Applications of Remote Reality in Agro-Informatics

Filip Findura

Bachelor's thesis
2021

Tomas Bata University in Zlín
Faculty of Applied Informatics

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav automatizace a řídicí techniky

Akademický rok: 2020/2021

ZADÁNÍ BAKALÁŘSKÉ PRÁCE
(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Filip Findura**
Osobní číslo:	**A18607**
Studijní program:	**B3902 Inženýrská informatika**
Studijní obor:	**Informační a řídicí technologie**
Forma studia:	**Kombinovaná**
Téma práce:	**Aplikace vzdálené reality v agro-informatice**
Téma práce anglicky:	**Remote Reality in Agro-informatics Applications**

Zásady pro vypracování

1. Seznamte se s aktuální problematikou využití vzdálené reality napříč různými obory.
2. Vypracujte informační přehled shrnující současný stav řešené problematiky.
3. Porovnejte používané technologie pomocí případových studií.
4. Navrhněte vhodné technologické řešení.
5. Experimentálně vyzkouší přenos obrazu přímo z hydroponického skleníku, nebo ekvivalentního prostředí.
6. Shrňte nabyté zkušenosti v závěru a doporučte vhodné směry budoucího vývoje.

Forma zpracování bakalářské práce:	**Tištěná/elektronická**
Jazyk zpracování:	**Angličtina**

Seznam doporučené literatury:

1. GURBAN, Eugen Horatiu a Gheorghe-Daniel ANDREESCU. Greenhouse environment monitoring and control: State of the art and current trends. Environmental Engineering and Management Journal [online]. 2018, 17(2), 399-416. ISSN 1582-9596. Dostupné z: doi:10.30638/eemj.2018.041
2. MARQUES, Gonçalo, Diogo ALEIXO a Rui PITARMA. Enhanced Hydroponic Agriculture Environmental Monitoring: An Internet of Things Approach. RODRIGUES, João M. F., Pedro J. S. CARDOSO, Jânio MONTEIRO, Roberto LAM, Valeria V. KRZHIZHANOVSKAYA, Michael H. LEES, Jack J. DONGARRA a Peter M.A. SLOOT, ed. Computational Science &#x2013; ICCS 2019 [online]. Cham: Springer International Publishing, 2019, 2019-06-08, s. 658-669. Lecture Notes in Computer Science. ISBN 978-3-030-22743-2. Dostupné z: doi:10.1007/978-3-030-22744-9_51
3. MAVRIDOU, Efthimia, Eleni VROCHIDOU, George A. PAPAKOSTAS, Theodore PACHIDIS a Vassilis G. KABURLASOS. Machine Vision Systems in Precision Agriculture for Crop Farming. Journal of Imaging [online]. 2019, 5(12). ISSN 2313-433X. Dostupné z: doi:10.3390/jimaging5120089
4. GUPTA, Himanshu a Roop PAHUJA. Estimating Morphological Features of Plant Growth Using Machine Vision. International Journal of Agricultural and Environmental Information Systems [online]. 2019, 10(3), 30-53. ISSN 1947-3192. Dostupné z: doi:10.4018/IJAEIS.2019070103
5. KACIRA, M. a P. P. LING. Design and Development of an Automated and Non-contact Sensing System for Continuous Monitoring of Plant Health and Growth. Transactions of the ASAE [online]. 2001, 44(4). ISSN 2151-0059. Dostupné z: doi:10.13031/2013.6231
6. LING, P. P., G. A. GIACOMELLI a T. RUSSELL. Monitoring of plant development in controlled environment with machine vision. Advances in Space Research [online]. 1996, 18(4-5), 101-112. ISSN 02731177. Dostupné z: doi:10.1016/0273-1177(95)00866-D

Vedoucí bakalářské práce:	**Ing. Bc. Pavel Vařacha, Ph.D.** Ústav informatiky a umělé inteligence
Datum zadání bakalářské práce:	**15. ledna 2021**
Termín odevzdání bakalářské práce:	**17. května 2021**

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky
DĚKAN
L.S.

doc. Mgr. Milan Adámek, Ph.D. v.r.
děkan

prof. Ing. Vladimír Vašek, CSc. v.r.
ředitel ústavu

Ve Zlíně dne 15. ledna 2021

ABSTRACT

This work discusses the contemporary state of smart greenhouse technologies and solutions for computer-mediated reality. It addresses the challenges of live video streaming for remote reality and identifies the requirements for high quality user experience in remote reality. Furthermore, it provides an experimental comparison of coding formats to determine a suitable technological solution in regards to the considerations of live video streaming from a greenhouse.

Keywords: remote reality, video streaming, video coding format, smart greenhouse

ABSTRAKT

Tato práce se zabývá současným stavem technologií chytrých skleníků a počítačově zprostředkované reality, popisuje problémy spojené s živým přenosem videa pro účely vzdálené reality a identifikuje požadavky na vzdálenou realitu nutné pro zprostředkování dobré zkušenosti uživatele. Dále pak zpracovává experimentální porovnání formátů kódování videa za účelem výběru nejvhodnějšího technologického řešení pro živý přenos videa ze skleníku.

Klíčová slova: vzdálená realita, přenos videa, formát kódování videa, chytrý skleník

TABLE OF CONTENTS

INTRODUCTION

The ability to gather data has grown immensely in the past years, to the point where data acquisition is rarely a limiting factor anymore, but rather the availability of the gathered data and their evaluation is the focus of further development. This thesis discusses the availability angle of data processing and application, or specifically the options for a low-latency real-time data transfer from a hydroponic greenhouse for use in a remote reality, taking the limitations of such an environment into account.

Hydroponics is a method of plant cultivation where soil is substituted with a nutrient-enriched water solution [1], though an inert substrate is often used for mechanical support of the plants, due to the difficulties of otherwise supporting their growth [2]. As the plants are thus grown in a fully artificial environment, understanding how they interact with it and anticipating their needs or reactions to a change in this environment can help increase growth efficiency, prevent yield losses and ensure proper health of the plants [3], consequently facilitating a decrease in consumption of resources in crop farming, including space1), while increasing the yield thanks to a closely controlled environment optimized for problem-free plant growth.

While multiple methods of plant health monitoring and automated control can be employed, human supervision is still desired or even required for cases impossible to cover by contemporary smart monitoring systems. Especially with the recent necessity of frequent remote work [4], allowing a human operator to access a real-time data feed from their greenhouse can simplify this oversight and reduce the work hours otherwise necessary for commuting to the location, or even between multiple locations.

1)Especially with the expected upswing in vertical farming. See chapter 5.

I. THEORETICAL PART

1 MOTIVATION

This bachelor thesis supports project No. FW01010381, "Inteligentní robotická ochrana zdraví ekosystému hydroponického skleníku" (*Intelligent robotic health protection of the hydroponic greenhouse ecosystem*) [5], which is a part of the *TREND Program* under the patronage of the Technology Agency of the Czech Republic [6] (hereafter referred to as "the Project"). The Project's objective is to design and manufacture a robotic system facilitating plant health inspection and monitoring through remote reality, with the aim of increasing the level of control, coordination and communication permitted to the supervisors, biologists and consultants of a hydroponic greenhouse, thus increasing their work efficiency.

The robotic system shall be capable of independent movement within a greenhouse and possess a remote-controlled arm with a high-definition camera. The resulting video feed shall be streamed real-time to allow the construction of a remote reality for an end user, first through a web page and later through a set of virtual reality goggles with full-scope freedom of view around the position of the robot. The video resolution shall be high enough for a professional examination of the plants.

In addition, as a later part of the Project, a software shall be developed for automatic screening of plant health and detection of diseases, pests or growth deficiencies. [7]

1.1 Thesis aims

This thesis encompasses the following goals:

1. Get acquainted with the current usage of remote reality across different fields[1)].
2. Compile an overview of the contemporary technologies available[1)].
3. Compare and contrast case studies for individual technologies[2)].
4. Propose a technological solution for the supported Project[2)].
5. Experimentally test video transmission and streaming from a hydroponic greenhouse, or an equivalent environment[3)].

[1)]See the Analytical Part.
[2)]See the Technical Report.
[3)]See chapter 9.3.

6. Summarize the findings and propose a future course of development for the Project[4].

This thesis focuses on establishing the best video characteristics for video streaming and video compression techniques for remote reality purposes.

In the Theoretical Part, an overview of remote reality and video streaming concepts is provided.

In the Analytical Part, a comparison of the currently available remote reality technologies and smart greenhouse solutions is presented to determine a baseline for the experimental part.

In the Technical Report, real-time video streaming is experimentally tested and a comparison of the results made in regards to the above-mentioned remote reality considerations.

Finally, a future course of development of the remote reality component of the Project is proposed.

[4]See chapters 9.2.4, 9.3.4 and Conclusion.

2 REMOTE REALITY

Remote reality provides a computer-mediated immersive 3D environment that the user can examine at their leisure. Unlike virtual reality, remote reality is based on a footage of an existing location, either pre-recorded (for 3D films) or available through a real-time video stream [8], as is the case of the Project.

When it comes to remote reality (or other types of computer-mediated reality, such as virtual or augmented reality), latency[1)] is the number one concern [9]. Without low latency, the user experience cannot be good, as any swifter motion of the head will leave the rendered scene lagging behind the movement. At best, this causes a visible misalignment or stuttering of the environment, but it may also lead to motion sickness in some users. [10]

Latency is less pressing in case of slow movements and relatively unchanging scenery and brightness, but for the purposes of a good virtual reality experience, it should still not exceed 20 ms, or the aforementioned problems might start to emerge. However, latency as low as 7 ms is recommended. [11]

Before we tackle the problem of reducing the latency of our remote reality video feed, though, we need to examine the considerations of video streaming.

[1)]See chapter 3.1.

3 VIDEO STREAMING

In digital communication, "streaming" is a process of continuously receiving and presenting data to an end user while further data are still being delivered from the provider over the Internet [12]. As such, data streaming presents unique challenges contrary to non-streaming delivery of data (downloading the whole data file before usage), especially in regards to the limits of bandwidth for data transmission and the possibility of lag[1)] or buffering, skipping and freezing.

In case of video streaming, several steps of the streaming process can be identified. First, the video feed has to be captured by a camera. Then the audiovisual data have to be encoded for transmission and published through a streaming channel[2)]. Then, the data have to be delivered and distributed to the end users, and finally decoded to play the video.

If at any point enough latency is introduced to make the next frame of the video unavailable in time, the video will lag.

3.1 Latency

Latency is the time delay between the request for a video frame and the actual time that the transfer is received. As such, it is an important concern for video streaming, as low latency must be ensured for quick response time and smooth video screening.[13]

Latency is especially important for live streaming, where it is balanced against other concerns of video quality.

3.2 Live streaming

Live streaming is the delivery of content directly as it is being produced in real-time. Where a pre-recorded video can be compressed and uploaded to a streaming service *before* it is published for streaming to consumers, all of these procedures need to be performed in real time for live streaming and introduce latency to the stream.

As such, live streaming suffers even more from the aforementioned challenges of deliv-

[1)] A delay between an input and a reply/reaction.

[2)] Here lies the fundamental difference between streaming a pre-recorded video and live streaming a video (see chapter 3.2).

ering good quality of content within the constraints of bandwidth and while avoiding lag.

Before we further discuss the means of encoding and delivery of a live video stream, several key concepts of digital video must be understood.

3.3 Resolution

The resolution of a digital image (either a still picture or a video frame) is the number of distinct pixels that the image is composed of and dictates its level of detail. Usually, it is given as the number of pixels in each dimension, *width* × *height*. Thus for example a video with a resolution of 1280 × 720 would have each frame 1280 pixels wide and 720 pixels high. Another common way of quoting the resolution of an image is by its total pixel count (usually expressed in megapixels).

The display resolution of a monitor, however, is usually described by the common name of a standard display resolution, with some of the popular standards shown in table 3.1. Alternately, a standard resolution may be referred to only by its vertical pixel count, so a Full HD video resolution might be given as 1080p[3].

Table 3.1 Standard resolutions by common names

Name	Resolution (px)
HD	1280 × 720
HD+	1600 × 900
Full HD	1920 × 1080
2K	2560 × 1440
4K	3840 × 2160
8K	7680 × 4320

3.4 Frame rate and Refresh rate

Frame rate and refresh rate are two interconnected concepts related to the speed at which the individual images comprising a video are drawn.

[3] The *p* standing for progressive scanning, a video format where the lines of each frame are both scanned and drawn progressively in sequence. Contrast to interlaced video format used in analog television systems, where the odd and even lines are drawn alternately.

3.4.1 Frame rate

Frame rate is the frequency at which the consecutive frames of a video should be displayed, expressed in *frames per second* (fps). When describing a video, its frame rate can be added to its shortened resolution, so a 1080p30 video would be Full HD with 30 fps.

A low frame rate results in a "stuttering" video as the viewer starts to notice the sequence of still images instead of the illusion of a continuous motion. Similarly, rapid movements at lower frame rates result in motion blurring, as the brain has to deduce the intermediate movements between frames.

A high frame rate can keep even fast movement smooth, but at the cost of increased file size. Thus lower frames per second are acceptable for videos with no abrupt changes, while a video containing quick movements or rapid brightness differences requires a higher frame rate to prevent blurring.

3.4.2 Refresh rate

Refresh rate is the frequency at which a monitor updates the displayed image, expressed in hertz (Hz). This usually equals the frame rate of the displayed video, though video of both higher and lower frame rate than the refresh rate of a monitor can be displayed by dropping or doubling some frames, respectively.

In the context of remote reality, refresh rate also has to be considered for the purposes of movement tracking speed, as refresh rate limits the speed at which the remote reality headset can react to the head movements of the user, and thus update their field of view. [14]

Therefore, remote reality requires higher refresh rate than other types of video feeds. Refresh rate below 60 Hz is not recommended due to motion sickness concerns [15], while the suggested refresh rate for virtual reality is at least 90 Hz. [16]

To provide a completely smooth experience during rapid head movements, refresh rates upwards of several hundred hertz are presumed to be needed [17], though such numbers are virtually unachievable due to bit rate concerns.

3.5 Bit rate

Bit rate determines how many bits of data are processed every second. It is usually expressed in kilobits or megabits per second (kbps or Mbps, respectively). Bit rate is a very important statistic for a live-streamed video, as the bandwidth[4)] is the main limiting factor in continuous delivery of data. [18]

When streaming, higher resolution and frame rate result in a larger volume of data to be transferred, and thus a higher bit rate. Should the network bandwidth be insufficient for the given bit rate, latency will be introduced into the data stream, resulting in visible buffering or freezing of the video. [19]

As network bandwidth is based on the available hardware and wireless or wired networking, and thus hard to change, it is the bit rate that has to be reduced to low enough levels for the available bandwidth. Therefore, the video has to be compressed through the use of a codec.

3.6 Codecs and Coding formats

Uncompressed video files are extremely large:

$$\textit{Uncompressed Video Size per Second (bps)} =$$

$$\textit{Frame Rate (fps)} \times \textit{Resolution (px)} \times \textit{Bit Depth (bits)}$$

A single second of an HD video at 30 fps and with the common bit depth of 8 bits would result in a file size of over 221 megabits (or 27.6 MB), with corresponding bit rate. As already mentioned, there is a pressing need for data compression.

Here, two often mixed up terms come into play:

A **codec** (portmanteau of *coder-decoder* [20]) is a device or software for encoding and later decoding data for transmission.

A **coding format** is the data compression algorithm used to reduce the size of the video file and its bit rate. Coding formats are sometimes also called "video coding

[4)] The maximum rate of data transfer.

standards", as they are the technical specification of the algorithm, whereas the specific implementation of a coding format is a codec.

As our objective is to transfer a live video stream with limited bandwidth[5)] and low latency, we will be searching for a suitable coding format that can ensure both good compression ratio and speed. Suitable coding formats will be further discussed in chapter 7.1 and the chosen formats will be experimentally compared in the Technical Report.

3.7 Container formats

An encoded video feed is normally embedded into a container format (or wrapper), a file format that can store multiple data feeds (such as a video feed and the accompanying audio feed) along with the metadata relating to those feeds, subtitles and more.

For the purposes of testing multiple coding formats, the Matroska multimedia container will be used in the experimental part of this work. Matroska (file extension *.mkv*) is a free, open-standard container format that declares compatibility with multitude coding formats and a robust streaming support as its core goals [21]. It also allows combining multiple video feeds into a stereoscopic video.

3.8 Monoscopic and Stereoscopic video

One final concern for the discussion of remote reality video streaming is the use of a monoscopic versus stereoscopic video feed.

Monoscopic video is a regular video, composed of a single video feed. For the purposes of 3D film-making, it is often projected on a virtual sphere centered around the viewer.

Stereoscopic video, on the other hand, is composed of two video feeds captured by two cameras set close to each other[6)] and representing the two human eyes. Therefore it cannot be viewed without a headset, as each video feed is presented to a different eye, allowing the brain the calculate the depth of the shown footage. Thus stereoscopic video provides a more immersive experience of a 3D environment, simulating the way one normally sees the world around them. [23]

[5)] Given the goal of streaming from a free-moving robot in a greenhouse, where only a wireless network covering a large area will be available.

[6)] At the average interpupillary distance, that is the distance between pupils. [22]

As monoscopic video lacks a proper sense of depth, it generally provides a less realistic experience. However, it is also easier to produce than stereoscopic footage, as no special equipment is necessary; more versatile, as no changes are needed to display it on devices other than a virtual reality headset; and more resource-effective, as it requires only a single video feed.

Multiple ways of streaming stereoscopic video exist. In addition to using a container format with two video feeds, it is also possible to double the frame rate, then use even and odd frames to transfer right and left eye footage within a single feed, or stream two video feeds separately and synchronize then on the receiving end via a time code. In all cases, however, assuming the same resolution and frame rate, a stereoscopic video will take up double the data of a monoscopic video, which generally means that a monoscopic video can afford to be shot in a higher quality than a stereoscopic one.

3.9 Communication protocols

Communication protocols are defined systems of data transfer between two or more electronic devices, including the rules and syntax for message composition, communication synchronization and error correction or recovery. [24]

For the purposes of testing a live video stream transfer through a server, the Real-Time Streaming Protocol (RTSP) was chosen, as it is a well-established protocol specifically designed for low-latency media streams.

Unlike other data transfer protocols, RTSP is stateful, and thus keeps a set of information needed to track the current session. RTSP only has to perform the time-intensive session starting once, unlike stateless web transfer protocols such as HTTP. This allows RTSP to achieve much higher speeds of data transfer. [25]

RTSP can achieve a very low latency thanks to the efficient Real-Time Transport Protocol (RTP) protocol used for the transmission of the media itself. In order to decrease latency, RTP sends the requested data separated into small packets suitable for quick transmission between the producer and the consumer(s).

The setup of a RTSP server will be further discussed in chapter 9.3.2.

3.10 Video quality

While looking for an efficient video compression tool to achieve a low bit rate on the video stream, we also have to consider video degradation by the encoding process. Assuring a good quality of the resultant video is as important for the user experience as transmitting the video feed without perceptible latency.

Video quality can be evaluated either subjectively by a human viewer, or objectively with the help of a set of mathematical models that assess the level of artifacts and distortion introduced to the video feed by the encoding.

As we will have access to both the original and the transcoded video, we can employ full reference video quality methods [26], which compute the quality difference between the original and encoded feed.

3.10.1 PSNR

The Peak Signal-to-Noise Ratio (PSNR) is the ratio between the maximum possible power of a signal and the power of the noise introduced into its representation. PSNR is the most frequently used objective image quality metric, though it does not always correlate well with human-perceived differences in quality. [27]

PSNR is expressed on the logarithmic scale in decibels, with higher values denoting less noise introduced. Values below 30 dB generally denote low quality video with visual artifacts, while values over 45 dB imply high quality, with little perceivable benefits for higher PSNR. [28]

3.10.2 SSIM

The Structural Similarity Index Measure (SSIM) evaluates the changes in the structural quality of a video feed as perceived by a viewer, and thus is better suited for video quality assessment from the users' point of view. [29]

SSIM measures the similarity between two signals with unitless values ranging from 0 to 1; 1 indicating that the two signals are perfectly structurally identical, while a 0 means there is no structural similarity. Values over 0.95 denote very good quality, while a video with SSIM of 0.99+ contains no perceptible imperfections. [30]

3.10.3 VMAF

The Video Multi-Method Assessment Fusion (VMAF) is a user perception video quality assessment tool developed by Netflix to gauge the parameters of different video coding formats, codecs and encoding settings for their streaming services. [31]

VMAF combines several quality metrics into a single score on the scale of 0 to 100, indicating the structural similarity between two signals. VMAF thus resembles SSIM, though it uses a set of algorithms to correct for possible errors in video assessment.

However, for the purposes of this thesis, a video quality comparison using the standard metrics will be sufficient to ensure no massive drops in video quality due to the encoding.

II. ANALYTICAL PART

4 SPECTRAL IMAGING IN PLANT MONITORING

As the demands on agricultural productivity grow, the implementation of innovative technical solutions is becoming necessary to meet those demands. Precision agriculture controlled by artificial intelligence, or smart farms connected through an Internet of Things approach are some of the solutions being actively pursued [32], but all such solutions depend on a way of monitoring and diagnosing the crop growth.

While this work is primarily focused on monitoring a greenhouse through a real-time video feed in visible light, another method of plant monitoring has to be mentioned, as it is widely used for non-destructive estimation of plant health. [3] [33] [34]

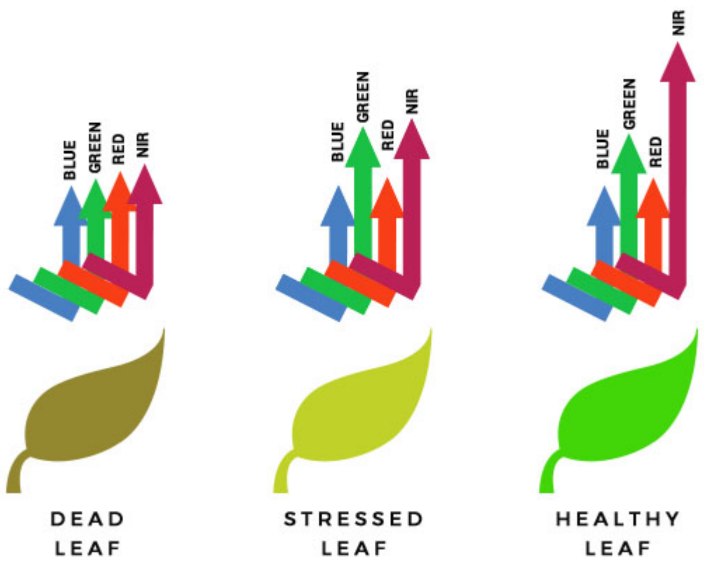

Figure 4.1 NIR vs visible light reflection in plants [35]

Spectral imaging is the collection of information from both visible and non-visible parts of the electromagnetic spectrum. As various types of materials reflects different parts of the electromagnetic spectrum in distinct ways, spectral imagery can help with monitoring of conditions undetected by visible light imagery. [36]

In recent years, a steady increase in the usage of spectral imaging could be seen in agriculture, especially of infrared monitoring of the development and health of crops via colour-infrared (CIR) imaging.

Colour-infrared imagery is a type of false-colour imaging offering a visible-light representation of images based in a portion of the electromagnetic spectrum known as near-infrared[1] (NIR).

For example, whereas the absorption of visible light is not that dissimilar for leaves both dried-out and healthy, the level of NIR reflection is much higher for healthy leaves, thus plant health can be detected through NIR reflection[2]. [35]

Various implementations of plant monitoring systems based on spectral imagery can be found in the smart greenhouse solutions examined in the following chapter.

[1]Wavelengths from 800 to 2 500 nm. [37]
[2]See figure 4.1.

5 SMART GREENHOUSE SOLUTIONS

The population of the Earth is on a steady rise and the food production needs to increase rapidly. Conventional farming is already struggling to support this development, as extensive farming strains the limits of available farmland [38], and according to the Food and Agriculture Organization, food production must double by 2050 to meet the demand, as the world population is expected to reach 9.6 billion by then. [39]

This, therefore, leads to a greater push for adopting intensive farming techniques, including vertical greenhouse hydroponic farming. [40]

Vertical farming presents a possibility of a sustainable and efficient agricultural production, as a smart greenhouse with water recycling system and automated light, nutrient and atmosphere control may achieve up to twenty times the amount of crops produced per acre with 90% less water expended than traditional methods [41]. In addition, according to the World Health Organization, about 80% of the current global population resides in urban areas, where the incentive to use the space-wise vertical farming is even greater. [42]

The monitoring and data management of smart greenhouse farms will thus likely be of utmost importance in the coming years.

5.1 Available solutions

As was previously mentioned, multiple smart greenhouse systems already exist and are available on the market. Notable solutions, both established and up-and-coming, were examined for comparison with the Project's proposed solution and shall be discussed below, ordered alphabetically.

5.1.1 GRoW by METOMOTION

GRoW (Greenhouse Robotic Worker) is a multipurpose robotic system for greenhouse automation[1)], created under the European Union's Horizon 2020 research and innovation program grant [44]. Its intended application is robotic harvesting of produce, presently only of tomatoes.

[1)]See figure 5.1.

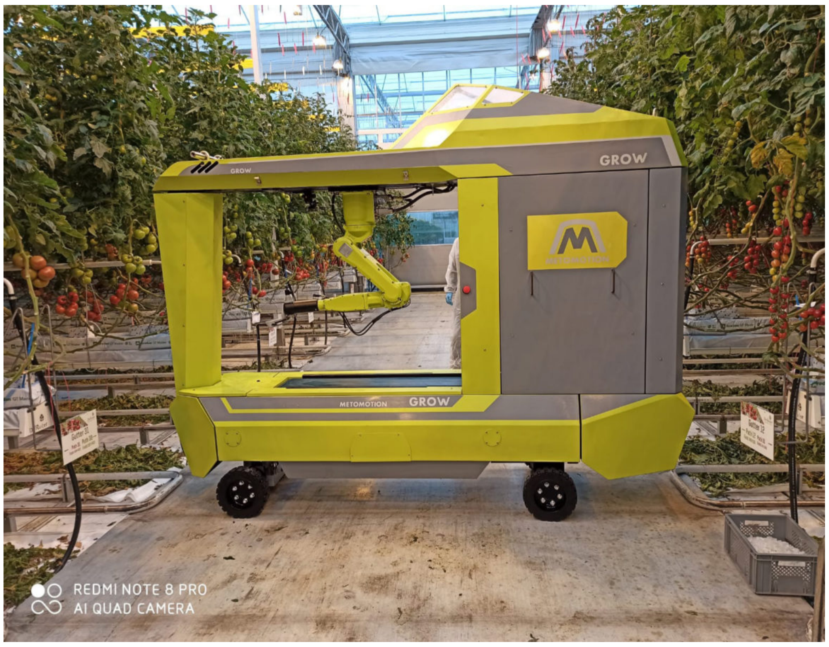

Figure 5.1 GRoW as featured in the Irrigation Leader magazine [43]

GRoW is designed for an easy integration into an existing greenhouse. It features a 3D vision system and computer vision algorithms, multiple robotic arms with a proprietary harvesting manipulator for damage free harvesting, an autonomous movement system and an on-board boxing system. [45]

5.1.2 LUNA by iUNU

LUNA is a greenhouse AI platform that uses computer vision to monitor crop growth in a greenhouse. The solution gathers data by way of a rail-mounted monitoring platform[2)], while a central processing unit prepares data aggregations, statistics and projections available through a web interface of a mobile application. In addition, the solution incorporates notifications for various detected problems including growth rate, greenhouse temperature, or pests; and a real-time video transmission from the monitoring platform. [46]

5.1.3 PlantEye by Phenospex

PlantEye is a multi-spectral 3D laser scanner designed to monitor and analyse plants. It is constructed to resist adverse conditions, such as direct sunlight or rain, and can scan thousands of plants daily, each scan comprising of multiple morphological and

[2)]See figure 5.2.

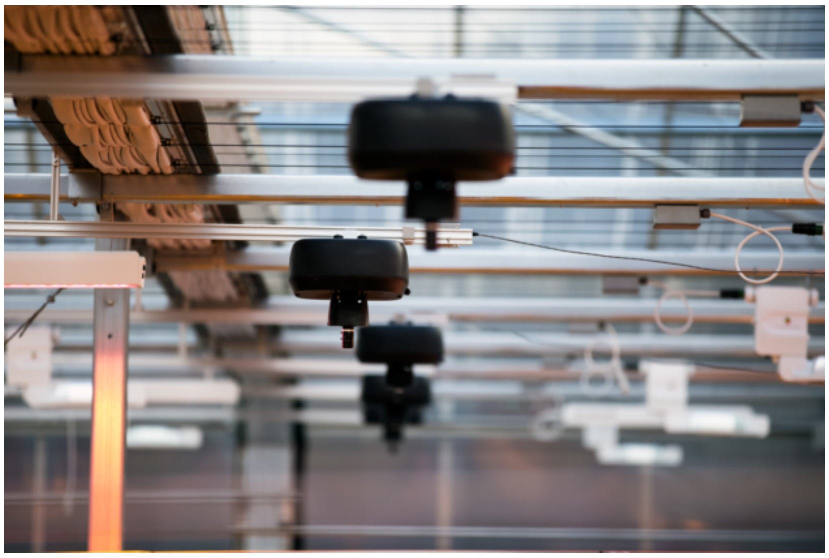
Figure 5.2 The rail system in a greenhouse using LUNA [46]

physiological parameters such as biomass, height, leaf area and projected area, RGB and NIR color, greenness or chlorophyll levels of each plant. [47]

HortControl software is then used to combine these parameters into a 3D model of the plant[3], allowing a comprehensive examination the plant and evaluation of its health, or any growth faults and disease symptoms. [48]

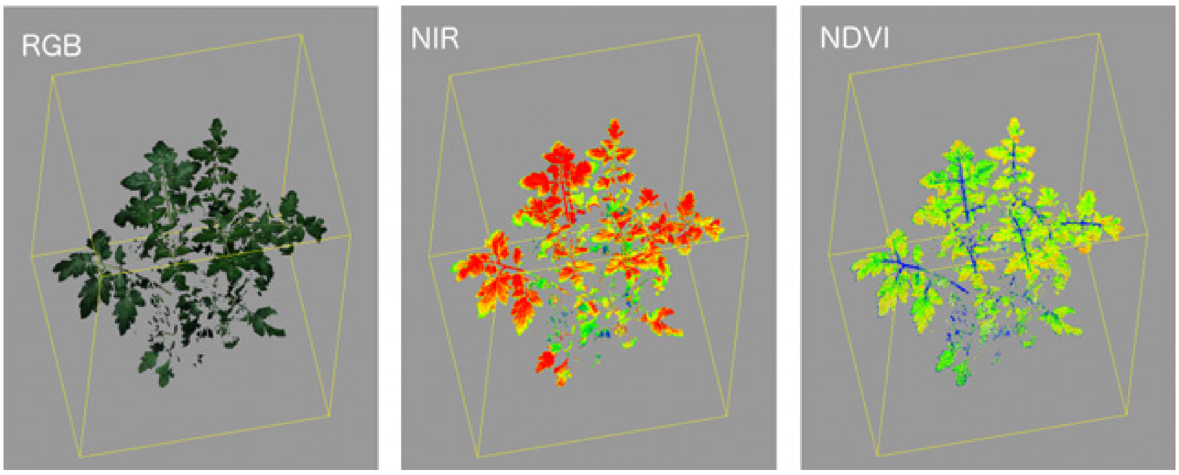

Figure 5.3 A 3D point model of a tomato plant with different spectral information, as captured with PlantEye [47]

Phenospex offers several installations of PlantEye:

1. **TraitFinder**, a scanning station[4] [49], or the smaller tabletop version **MicroScan** [50].

[3]See figure 5.3.
[4]See figure 5.5.

2. **FieldScan**, a large-scale platform[5] for outdoor use. [51]

3. A rail-mounted system for greenhouses.

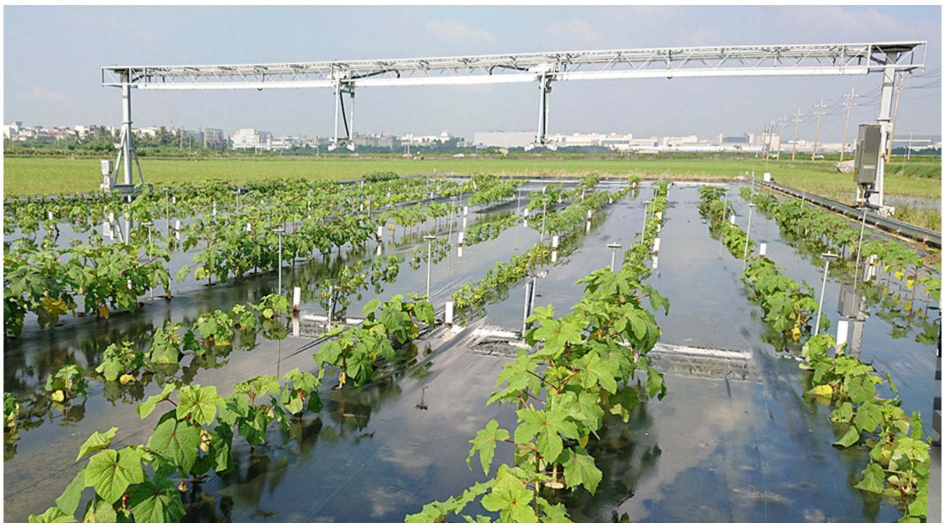
Figure 5.4 FieldScan deployed in a field in Taiwan [51]

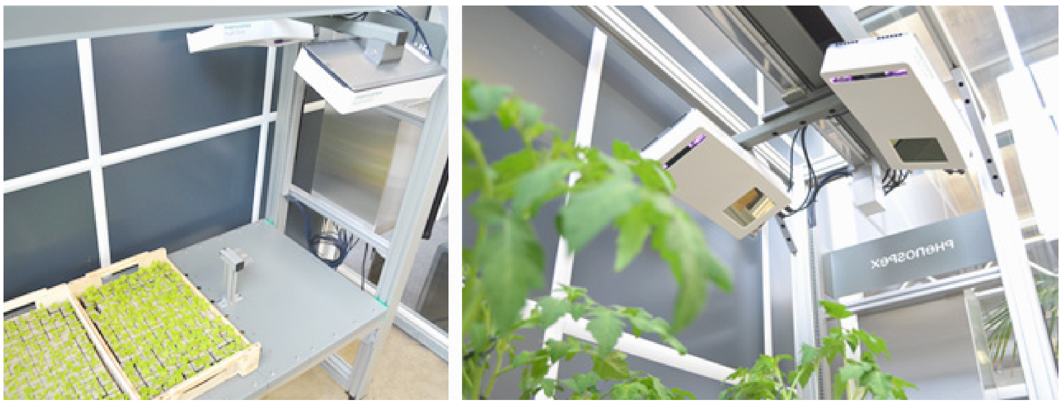
Figure 5.5 TraitFinder example use [49]

5.1.4 Virgo by Root AI

Virgo[6] is a rail-mounted plant harvesting robot currently in development. It features a computer vision and AI software capable of analyzing crop ripeness, automatically harvesting ready produce via a specialized gripper. Setting itself apart from other harvesting robots, Virgo assesses the shape of the crop, from strawberries to apples or cucumbers, and adjust its harvesting strength and technique accordingly, preventing damage to the produce. [53]

[5] See figure 5.4.
[6] See figure 5.6.

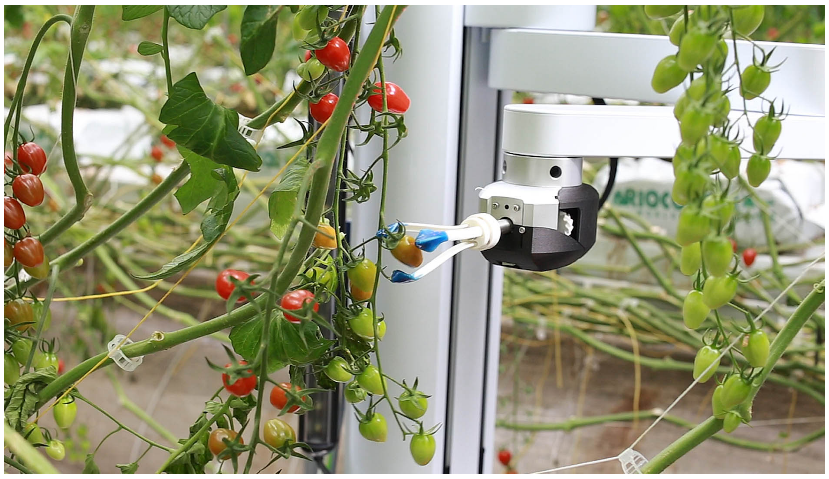

Figure 5.6 Virgo [52]

5.2 Technology used

The above-mentioned companies have been contacted in an effort to perform a deeper analysis of their solutions as a basis for this work's objective, but no answer was received.

6 REMOTE REALITY SOLUTIONS

In this chapter, we shall establish the quality expectations for a remote reality video stream, based on the available solutions for remote video monitoring of health and various virtual reality solutions.

6.1 Telemedicine

Telemedicine is a modern discipline joining health-related services with informational technologies to provide remote clinical services to patients in remote locations where access to medical care is limited [54]. Telemedicine solutions have a similar focus and face challenges similar to the Project, as both aim to provide a professional user with a reliable remote video feed access to a subject for diagnostic and monitoring purposes.

As such, an inquiry was made into the camcorder technologies used in telemedicine regarding the video feed characteristics utilized for diagnostic purposes, to establish a point of reference for plant health monitoring. Table 6.1 shows a comparison of cameras available from the main telemedicine equipment manufacturers.

Table 6.1 Telemedicine cameras comparison

Name	Manufacturer	Resolution (px)	Frame rate (fps)
VersaScope [55]	AMD Telemedicine	3264 × 2448	30
DE605 General Examination Camera [56]	Firefly	2592 × 1944	30
i1000MD [57]	GlobalMed	1920 × 1080	60
TotalExam 3.2 [58]	GlobalMed	1280 × 720	60
TotalExam HD [59]	GlobalMed	3840 × 2160	60 (HDMI)[1)] 30
GEIS Teleconsultation Camera [60]	visionflex	1920 × 1080	30

As we can see, the resolution offered by telemedicine cameras ranges from HD to 4K. However, the only camera recording in HD is the TotalExam 3.2 camera, superseded by GlobalMed's newer TotalExam HD camera. As such, a resolution of Full HD to 4K could be considered the current norm for remote medical examination practices. Resolution higher than Full HD, though, is only offered at 30 fps, with the exception of the TotalExam HD camera when connected through an HDMI cable directly to a tablet or laptop. [61]

[1)] Higher frame rate available only when connected directly to a telemedicine station.

Principally, telemedicine cameras do not offer video quality better than any regular web camera, their advantages lie elsewhere. Telemedicine cameras are water-resistant and designed to be washed or disinfected, have inbuilt lighting for ease of examination and come with a pre-set streaming software.

For the purposes of the Project, though, a regular web camera should be sufficient until further functionality is required, like spectral imaging mentioned in chapter 4.

6.2 Virtual reality headsets

A virtual reality headset is a device granting the user access to a virtual reality, though it can also be used for the purposes of remote reality. The headset is usually composed of a head-mounted display capable of providing a separate video feed for each eye[2], a set of headphones and head motion tracking sensors.

A comparison of currently available virtual reality headsets was made for the purposes of establishing the target video characteristics of a remote reality video stream; presented in no particular order in table 6.2.

Table 6.2 Virtual reality headsets comparison

Name	FoV[3] (°)	Resolution (px per eye)	Refresh rate (Hz)
HP Reverb Virtual Reality Headset - Professional Edition [62]	114	2160 × 2160	90
HTC Vive [63]	110	1080 × 1200	90
HTC Vive Pro [64]	110	1400 × 1600	90
HTC Vive Cosmos [65]	110	1440 × 1700	90
Oculus Quest [66]	100	1440 × 1600	72
Oculus Quest 2 [67]	100	1832 × 1920	72/90[4]
Oculus Rift CV1 [68]	110	1080 × 1200	90
Oculus Rift S [67]	115	1280 × 1440	80
Oculus Go [69]	101	1280 × 1440	60
Sony PlayStation VR [70]	100	1960 × 1080	90/120[4]

Most virtual reality headsets offer non-standard resolutions, but fall within the general vicinity of a 2K resolution. Notable exceptions are the HP Reverb Virtual Reality Headset - Professional Edition with a non-standard 4K resolution and the Sony PlayStation VR with a non-standard Full HD resolution. It should be noted that the quoted reso-

[2]See chapter 3.8.
[3]Field of view (in degrees).
[4]Recommended and maximum value.

lutions are per eye, so assuming a stereoscopic video stream, double resolution has to be considered for data transmission purposes unless a monoscopic video is used.

Except for Oculus Go, Oculus Rift S and Oculus Quest, all virtual reality headsets support 90 Hz refresh rate, which as already mentioned is the current recommendation for virtual reality game development. [71]

6.2.1 Mobile phone headsets

In addition to standalone virtual reality headsets, certain platforms support plugging a mobile phone into a head-mounted holder and experiencing virtual reality with an application such as Gear VR [72] from Samsung or Daydream View [73] from Google.

However, even though it is possible to use a mobile phone headset in lieu of a specialized headset and certain sources project a steady rise in the usage of mobile virtual reality [74], the providers of the largest mobile virtual reality platforms have already discontinued their support [75], citing technical constrains and limitations of the devices, less immersive experience compared to full virtual reality headsets and consumer dissatisfaction with the service. [76]

In addition, as the self-contained headsets slowly drop in price, mobile virtual reality can no longer even claim to be the cheaper option. [77]

7 VIDEO STREAMING

In this chapter, the coding formats chosen for the experimental testing in Technical Report will be discussed, along with bit rate estimation for the individual coding formats.

7.1 Video coding format

While the two primary concerns are, as already mentioned, data compression efficiency and encoding speed, other considerations also apply and will help narrow down which coding formats to include in the Technical Report.

For the purposes of the Project, it is necessary to find a coding format designed with high resolution video streaming in mind, with mature support both by stable codec implementations and by hardware and end-point provider compatibility. In addition, proprietary coding formats with restrictive licenses are undesirable for legal reasons.

Therefore, four coding formats were chosen, presented in alphabetical order:

7.1.1 AV1

AOMedia Video 1 is a coding format developed by the Alliance for Open Media (AOMedia) with the support of Google, Amazon, Cisco, Microsoft, Mozilla and Netflix. It is a relatively young format, first announced on 1 September 2015, and promises large improvements in comparison to other mainstream coding formats, though so far it seems the performance of its codecs has not yet settled. [78]

AV1 is royalty-free to use and specifically intended for open-source projects.

7.1.2 VP9

VP9 is a coding format created by Google as a competition to the MPEG-H formats (mentioned below). VP9 has been in development since 2011 and has support in both web browsers and on mobile video players, though the main platform for VP9 encoded videos remains Google's YouTube.

VP9 is royalty-free to use.

7.1.3 H.264

H.264 Advanced Video Coding (AVC) is a MPEG-H coding format which currently dominates the video streaming world. Estimations claim that up to 80% of all online videos use the AVC coding format and 91% of online video producers use it for some of their content. [9]

It is an older coding format, with the original specifications approved in March 2003, which on the other hand grants it a well-developed compatibility with most browsers and devices, plus it is still actively maintained and developed[1]. AVC supports resolutions up to 8K, but to reach lower bit rates at high resolutions, H.264 generally requires the use of lossy compression.

H.264 is royalty-free for non-commercial use, though by 2027 the patents will have expired and license will no longer be necessary. [79]

7.1.4 H.265

H.265 High Efficiency Video Coding (HEVC) is a MPEG-H coding format designed as a successor to AVC, offering much lower bit rates at the same video quality. Its specifications were approved on 25 January 2013, and even though the adoption of H.265 for wider use was rather slow, it has a solid and active support. By 2019, HEVC was the second most widely used video coding format. [9]

One of the main reasons for the slow adoption of H.265 has been the uncertain situation about its licensing, where until 2018, HEVC video streaming providers could be charged with royalties under certain circumstances. [80]

7.1.5 H.266

Though it was not considered for this work, H.266 Versatile Video Coding (VVC) as the newest MPEG-H coding format already exists, albeit the standard only finalized on 6 July 2020 [81]. While it promises up to 50% decrease in bit rate and has the explicit aim to facilitate 4K video streaming [82], the coding format is not yet usable due to non-existent support. It may be worth to revisit once it has matured enough to offer a stable codec implementation.

[1] The latest version 26 was released on 13 June 2019.

Table 7.1 Bit rate calculated using online tools

Resolution	Coding format	Frame rate (fps)	Bit rate (kbps) Bandwidth calculator [83]	Bit rate (kbps) Video surveillance calculator [84]
4K	H.264	90	54 200	64 999
		60	36 200	43 333
		30	18 100	21 660
	H.265	90	40 200	50 460
		60	26 800	33 640
		30	13 400	16 820
FullHD	H.264	90	13 600	16 540
		60	11 500	11 030
		30	5 800	5 510
	H.265	90	10 100	12 840
		60	6 700	8 560
		30	3 400	4 280
HD	H.264	90	7 700	7 350
		60	5 100	4 900
		30	2 600	2 450
	H.265	90	4 500	5 710
		60	3 000	3 810
		30	1 500	1 900

7.2 Bit rate estimation

As compression efficiency of a coding format varies based on the contents of the video[2)], there is no static ratio of compression for each coding format.

In this chapter, we will establish a reference point for bit rate requirements by consulting helper tools and sources of video streaming information.

7.2.1 Bit rate calculators

While it is possible to find bandwidth calculation tools, they only cover selected coding formats (mostly H.264 and H.265) and do not offer any insight into the algorithms used to reach their results. Their outputs, as shown in table 7.1, can nonetheless be consulted for reference.

[2)] A non-moving object with little variation in brightness will result in a much higher compression rate than a fast, dynamic movement with changing environment and lighting.

7.2.2 Video streaming providers

Another guideline can be found by referencing the video streaming bit rates recommended by some of the major providers of video streaming platforms. A summary can be found in table 7.2.

Table 7.2 Recommended bit rate by video streaming provider

Provider Resolution	Frame rate Bit rate (Mbps)	
YouTube [85]	**30 fps**	**60 fps**
4K	13-34	20-51
2K	6-13	9-18
Full HD	3-6	4.5-9
HD	1.5-4	2.2-6
Twitch [86]	**30 fps**	**60 fps**
Full HD	4.5	6
HD	3	4.5
Wowza [87]	**30 fps**	**60 fps**
4K	8	12-20
Full HD	3.2	4.4-6
HD	1.6	2.6-4
Dacast [88] [89]	**30 fps**	**60 fps**
4K	-	20
2K	-	15
Full HD	4.5	7
HD	1.5	5
IBM Cloud Video [90]	**30 fps**	**60 fps**
4K	8-14	-
Full HD	4	8
HD	1.2	4

While the recommended bit rate for 4K streams varies by the provider, all recommended values for Full HD streams fall within range of 4.4 Mbps to 9 Mbps, with an average of 6.4 Mbps (standard deviation 1.7 Mbps) and a median of 6 Mbps.

It should be noted, however, that the streaming bit rate recommendations do not necessarily correspond with low-latency streaming.

7.3 Alternate encoders

7.3.1 Hardware acceleration

Video encoding is a very resource-intensive process, so even with a well-chosen coding format for quick encoding at a good level of compression, the hardware used will play an important role in the minimum achievable latency.

Hardware acceleration is a way of further increasing the performance of video processing by offloading some tasks[3)] from CPU to the more specialized GPU, where it can be performed more efficiently, or even concurrently with other calculations. With a well-chosen GPU, 2 to 5 times speed-up of encoding and decoding is likely possible. [91] [92]

However, the encoding results are fully dependant on the hardware acceleration system used and coding format support differs by GPU [93]. It is not within the scope of this work to perform an adequate experimental comparison of the available implementations of different vendors, though such comparison would be worth pursuing in the next stage of the Project's technological review.

7.3.2 Cloud encoding

In addition to encoding a video immediately after is is captured, nowadays it is possible to use a cloud-based encoding service, which frees the user from the necessity to purchase the hardware and maintain the encoding software.

However, as the encoding is performed on the provider's servers, the raw video to be encoded has to be delivered to the server in full, which makes it unsuitable for live streaming, as the bandwidth requirements would be enormous[4)]. On the other hand, cloud transcoding[5)] might be useful should the Project require multiple video streams at different formats, frame rates or resolutions. [94]

[3)] In video processing, mostly encoding and decoding.

[4)] See chapter 3.6

[5)] Where encoding refers to the application of a codec on a raw video file, transcoding means changing an already encoded video to a new coding format, file format, resolution, frame rate, etc. Overall, though, both processes are largely the same, as in both cases the video feed has to be encoded anew.

8 PROJECT CONCERNS

Unlike the majority of the smart greenhouse solutions described in chapter 5 that are either installed in place or move on rails, BERABOT[1)] is designed for free movement in a greenhouse. This, however, places certain limits on network availability for video streaming, as a wireless network will need to be used. Therefore, the solution has to account for a lower available bandwidth[2)] and the coding format should deliver low bit rate videos.

On the other hand, the Project's aim of live streaming for a remote reality necessitates an ultra-low latency[3)] and thus a high encoding speed, but also a high frame rate to prevent the possible problems discussed in chapter 3.4.2. In addition to that, the robot-mounted camera must have high resolution to allow professional examination of the plants.

While in the first stage of the Project the video will only be streamed to a web page and thus the demands on low latency and high quality can be lowered, for the eventual remote reality streaming a resolution of Full HD or better and a frame rate of at least 60 fps has to be recommended, based on the findings in chapter 6.

Unfortunately, the concerns of low latency and high quality go directly against each other, as improving on one will have a negative impact on the other no matter the technological solution chosen. Consequently, a serviceable compromise has to be found.

It should also be noted, though, that a still photo will generally achieve a better resolution than a video, given the same hardware setup, especially as it needs not meet the limits of a real-time video stream. Therefore, it can be suggested to also allow the robot to take high resolution photos to supplement its remote reality function.

1)See chapter 1.

2)Due to the restrictions on movement during the writing of this work, it was unfortunately unfeasible to take a measurement of the connection speed in the greenhouse where BERABOT shall be deployed.

3)See chapter 2.

III. TECHNICAL REPORT

9 CODING FORMATS COMPARISON

In this chapter, a series of tests will be effected to examine the performance of the coding formats chosen in chapter 7.1.

9.1 Video processing

Due to the technology stack delineated in chapter 3, FFmpeg (Fast Forward MPEG) version 4.3.2 was determined as the best tool for video processing in the experimental comparison of the coding formats. FFmpeg is a free open-source software project that supports encoding, transcoding and transrating[1)] video in all chosen coding formats, along with offering RTSP streaming and a multitude of tools for video quality comparison and performance benchmarking.

Table 9.1 Codecs used for individual coding formats

Coding format	Codec
H.264	libx264
H.265	libx265
VP9	libvpx-vp9
AV1	libaom-av1

Table 9.1 shows the codecs selected for encoding the individual coding formats. For H.264 and H.265, the recommended open-source codecs were chosen, both created by VideoLAN. For VP9 and AV1, the official codecs by the coding format creators were used.

9.2 Video transcoding

A single video was transcoded to the four selected coding formats to study the difference in their encoding times and the resultant bit rates when identical video feed is used. This test was carried out on a free test video file [95] with the audio stream removed for more precise comparison of the video stream compression.

```
Name:                Slide_4K_90FPS_test_noaudio.mkv
Coding format:       H.264
Resolution:          3840x2160
Frame rate:          90 fps
Bit rate:            33 273 kbps
```

1) That is changing the frame rate of the video feed.

```
Duration:              18.42 sec
Size:                  76.6 MB
```

The test video was transcoded to 4K, Full HD and HD resolutions at frame rates of 90, 60 and 30 fps. Encoding duration was measured by FFmpeg's *-benchmark* option.

9.2.1 Test 1, Recommended settings

The first suite of tests was carried out on a machine running Linux Mint 19 Tara with the following CPU specifications:

```
$ lscpu
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Little Endian
CPU(s):                8
On-line CPU(s) list: 0-7
Thread(s) per core:  2
Core(s) per socket:  4
Socket(s):             1
NUMA node(s):          1
Vendor ID:             GenuineIntel
CPU family:            6
Model:                 42
Model name:            Intel(R) Core(TM) i7-2760QM CPU @ 2.40GHz
Stepping:              7
CPU MHz:               1153.670
CPU max MHz:           3500,0000
CPU min MHz:           800,0000
BogoMIPS:              4784.46
Virtualization:        VT-x
L1d cache:             32K
L1i cache:             32K
L2 cache:              256K
L3 cache:              6144K
NUMA node0 CPU(s):   0-7
```

The recommended settings for high encoding speed and good level of compression were used for the transcoding[2] and the results can be found in table 9.2. Quality metrics were gathered from the resulting videos[3], as found in tables I and II in appendix 4.

This test should establish the baseline comparison of the coding formats' performance[4].

[2] See appendix 1 for the benchmark script used.

[3] See appendix 3 for the script used to gather the quality metrics.

[4] See tables 9.3 and 9.4.

Table 9.2 Bit rates and encoding time for recommended (test 1) settings

Resolution	Coding format	Frame rate (fps)	File size (MB)	Bit rate (kbps)	Encoding duration (minutes)
4K	H.264	90	73.7	32 059	5.20
		60	58.3	25 406	3.57
		30	38.0	16 601	2.51
	H.265	90	20.6	8 963	14.62
		60	17.6	7 651	10.55
		30	13.8	6 025	7.55
	VP9	90	47.9	20 812	18.67
		60	34.2	14 852	12.41
		30	19.6	8 527	6.88
	AV1	90	37.2	16 171	677.95 (11.30 hrs)
		60	26.9	11 682	446.04 (7.43 hrs)
		30	15.3	6 669	225.23 (3.75 hrs)
Full HD	H.264	90	23.3	10 144	1.81
		60	19.7	8 600	1.35
		30	14.1	6 152	1.06
	H.265	90	7.2	3 111	4.08
		60	6.2	2 705	3.03
		30	5.0	2 189	2.24
	VP9	90	20.7	9 006	7.80
		60	15.1	6 544	5.27
		30	8.7	3 778	2.99
	AV1	90	15.6	6 797	251.62 (4.19 hrs)
		60	11.3	4 908	173.76 (2.90 hrs)
		30	6.5	2 825	81.44 (1.36 hrs)
HD	H.264	90	11.2	4 848	0.86
		60	9.7	4 194	0.77
		30	7.3	3 146	0.64
	H.265	90	3.9	1 710	2.05
		60	3.5	1 504	1.55
		30	2.8	1 230	1.13
	VP9	90	12.4	5 379	4.01
		60	9.1	3 944	2.90
		30	5.3	2 287	1.71
	AV1	90	9.1	3 974	139.42 (2.32 hrs)
		60	6.7	2 891	92.61 (1.54 hrs)
		30	3.9	1 678	51.18

Table 9.3 Average proportional difference in bit rate when comparing the coding format in the row against the coding format the in column; recommended (test 1) settings

	H.264	H.265	VP9	AV1
H.264	–	301%	140%	185%
H.265	34%	–	47%	62%
VP9	75%	226%	–	132%
AV1	57%	170%	76%	–

Table 9.4 Average proportional difference in encoding duration when comparing the coding format in the row against the coding format in the column; recommended (test 1) settings

	H.264	H.265	VP9	AV1
H.264	–	43%	29%	1%
H.265	239%	–	70%	2%
VP9	355%	152%	–	3%
AV1	11 685%	4 966%	3 267%	–

9.2.2 Test 2, Fastest settings

The second suite of tests was carried out on the same machine as test 1. The codec settings were, however, changed to tune for lowest possible latency[5]. Quality metrics were gathered for the resulting videos[6], as found in tables III and IV in appendix 4.

This test should establish how encoding duration and bit rates are impacted by focusing more on low latency of the video stream then the level of compression. The results can be found in table 9.7. The comparative performance of the codecs can be found in tables 9.5 and 9.6.

An inverse test focusing on lossless compression over encoding speed was not performed, as all the codecs feature options for a high quality compression, that nonetheless cannot be used for encoding a real-time stream due to a massive increase in latency.

9.2.3 Test 3, Hardware difference

A third suite of tests was carried out on a server running Ubuntu 20.04 LTS with the following CPU specifications:

[5]See appendix 2 for the benchmark script.

[6]The same script as for the quality metrics of test 1 was used. See appendix 3.

```
$ lscpu
Architecture:                    x86_64
CPU op-mode(s):                  32-bit, 64-bit
Byte Order:                      Little Endian
Address sizes:                   46 bits physical, 48 bits virtual
CPU(s):                          40
On-line CPU(s) list:             0-39
Thread(s) per core:              2
Core(s) per socket:              10
Socket(s):                       2
NUMA node(s):                    2
Vendor ID:                       GenuineIntel
CPU family:                      6
Model:                           85
Model name:                      Intel(R) Xeon(R) Silver 4210 CPU @ 2.20GHz
Stepping:                        7
CPU MHz:                         1000.018
CPU max MHz:                     3200.0000
CPU min MHz:                     1000.0000
BogoMIPS:                        4400.00
Virtualization:                  VT-x
L1d cache:                       640 KiB
L1i cache:                       640 KiB
L2 cache:                        20 MiB
L3 cache:                        27.5 MiB
NUMA node0 CPU(s):               0-9,20-29
NUMA node1 CPU(s):               10-19,30-39
```

Table 9.5 Average proportional difference in bit rate when comparing the coding format in the row against the coding format the in column; fastest (test 2) settings

	H.264	H.265	VP9	AV1
H.264	–	366%	137%	265%
H.265	27%	–	38%	72%
VP9	75%	275%	–	192%
AV1	39%	144%	52%	–

Table 9.6 Average proportional difference in encoding duration when comparing the coding format in the row against the coding format in the column; fastest (test 2) settings

	H.264	H.265	VP9	AV1
H.264	–	45%	64%	1%
H.265	269%	–	154%	3%
VP9	166%	68%	–	2%
AV1	10 734%	3 740%	5 899%	–

The same benchmark as in test 1 was used, to establish the impact of increased computational resources on the codecs' performance.

The results can be found in table 9.10. Quality metrics were then gathered for the resulting videos, as found in tables V and VI in appendix 4. The comparative performance of the codecs can be found in tables 9.8 and 9.9.

Table 9.7 Bit rates and encoding time for fastest (test 2) settings

Resolution	Coding format	Frame rate (fps)	File size (MB)	Bit rate (kbps)	Encoding duration (minutes)
4K	H.264	90	103.2	44 835	0.87
		60	83.1	36 146	0.70
		30	60.9	26 489	0.50
	H.265	90	28.2	12 243	4.26
		60	22.9	9 028	3.12
		30	16.7	7 272	1.95
	VP9	90	80.7	35 065	2.08
		60	58.0	25 232	1.55
		30	32.9	14 313	0.95
	AV1	90	38.6	16 762	199.78 (3.33 hrs)
		60	27.9	12 134	136.67 (2.28 hrs)
		30	16	6 943	74.18 (1.24 hrs)
Full HD	H.264	90	34.5	14 969	0.60
		60	28.6	12 439	0.54
		30	21.0	9 130	0.47
	H.265	90	9.4	4 076	1.35
		60	7.9	3 413	1.06
		30	5.9	2 570	0.77
	VP9	90	30.7	13 325	1.05
		60	22.2	9 671	0.85
		30	12.7	5 541	0.63
	AV1	90	16.4	7 118	66.13 (1.11 hrs)
		60	11.9	5 167	44.92
		30	6.8	2 978	24.31
HD	H.264	90	18.3	7 961	0.53
		60	15.1	6 580	0.50
		30	11.2	4 875	0.45
	H.265	90	4.9	2 138	1.02
		60	4.2	1 829	0.84
		30	3.2	1 406	0.66
	VP9	90	17.5	7 586	0.74
		60	12.8	5 547	0.62
		30	7.4	3 212	0.51
	AV1	90	9.7	4 206	36.01
		60	7.1	3 067	24.60
		30	4.1	1 783	13.43

Table 9.8 Average proportional difference in bit rate when comparing the coding format in the row against the coding format the in column; server (test 3)

	H.264	H.265	VP9
H.264	–	303%	1 479%
H.265	33%	–	488%
VP9	7%	21%	–

Table 9.9 Average proportional difference in encoding duration when comparing the coding format in the row against the coding format in the column; server (test 3)

	H.264	H.265	VP9
H.264	–	46%	19%
H.265	226%	–	42%
VP9	545%	244%	–

Due to the length of encoding, the tests for AV1 coding format were not finished. See chapter 9.2.4 for further information on AV1 encoding times.

9.2.4 Video transcoding results

In this set of tests, the aim was to ascertain the encoding times and bit rates produced by the selected coding formats when used on an identical video feed. All tables showing a proportional percentage comparison between the coding formats were calculated with the values for each format averaged over all tested resolutions and frame rates in the given test[7]. See figures 9.1 and 9.2[8] for a graphical comparison of the codecs' performance.

The tests have shown AV1 as an obvious exception in the encoding speed, with encoding on average 32 times slower than VP9, 45 times slower than H.265 and 115 times slower than H.264. This corroborates a recent performance test of H.265 versus AV1 encoding using both CPU codecs and hardware accelerated encoders. The study claims that on CPU, AV1 performs 50 to 14 times worse than H.265, depending on the set-up, and with a hardware accelerated encoder, AV1 is still 2 times slower than H.265. [78]

It should be noted that AV1 has shown remarkable results in test 2, where its fastest settings have produced only a marginally higher bit rate video at less than third the encoding duration than in test 1 (table 9.11), which unfortunately still left it at hours-long encoding times. As the goal of the Project is to deliver a low-latency video stream,

[7] Tables 9.3, 9.4, 9.5, 9.6, 9.8, 9.9, 9.11 and 9.12

[8] Both created in R using the *ggplot* library. [96]

Table 9.10 Bit rates and encoding time on server (test 3)

Resolution	Coding format	Frame rate (fps)	File size (MB)	Bit rate (kbps)	Encoding duration (minutes)
4K	H.264	90	74.8	32 517	1.04
		60	59	25 655	0.76
		30	38.5	16 707	0.49
	H.265	90	20.6	8 963	2.81
		60	17.5	7 651	2.03
		30	13.8	6 025	1.26
	VP9	90	4.2	1 813	5.86
		60	3.1	1 361	4.05
		30	1.9	851	2.14
	AV1	90	2.8	1 249	2856.23 (47.60 hrs)
		60	2.4	1 040	2358.79 (39.31 hrs)
		30	1.4	617	1124.95 (18.74 hrs)
Full HD	H.264	90	23.5	10 228	0.46
		60	20	8 695	0.41
		30	14.2	6 183	0.34
	H.265	90	7.2	3 111	1.13
		60	6.2	2 705	0.85
		30	5	2 189	0.61
	VP9	90	1.7	748	3.38
		60	1.3	597	2.50
		30	0.9	386	1.46
	AV1	90			
		60	Not performed due to length of encoding.		
		30			
HD	H.264	90		4 852	0.32
		60		4 210	0.28
		30		3 169	0.26
	H.265	90		1 710	0.76
		60		1 504	0.58
		30		1 230	0.44
	VP9	90		495	2.12
		60		383	1.60
		30		272	1.01
	AV1	90			
		60	Not performed due to length of encoding.		
		30			

Table 9.11 Proportional difference in bit rate and encoding duration when comparing the fastest (test 2) settings against the recommended (test 1) settings

	Bit rate	Encoding duration
H.264	147%	29%
H.265	125%	32%
VP9	159%	14%
AV1	104%	29%

Table 9.12 Proportional difference in bit rate and encoding duration when comparing the the recommended settings on 40 CPU (test 3) against 8 CPU (test 1)

	Bit rate	Encoding duration
H.264	100%	25%
H.265	100%	22%
VP9	9%	38%

AV1 does not seem like a suitable candidate for a coding format.

On the other end of the spectrum, H.264 achieved the fastest encoding times in all tests, on average half that of H.265, but also produced the highest bit rates, over 3 times the bit rate of H.265 in all tests. As the bandwidth available in a greenhouse will be limited, it is unlikely that H.264 would be a suitable solution.

This leaves us with two coding formats often termed direct competitors. In the first test, VP9 performed worse than H.265 on its recommended settings, both in achieved bit rates and encoding duration. Similar results were reached by a Netflix study, concluding that overall H.265 performs 19% to 22% better than VP9 [97]. In the second test, a divergent performance can be noticed, where VP9 achieved about one third better encoding time than H.265, but at the cost of nearly triple the bit rate of H.265.

In both tests 1 and 2, H.265 produces the lowest bit rate from all compared coding formats, which is especially evident at 4K resolution.

Quality metrics[9)] are comparable between the coding formats, with both PSNR and SSIM averaging on the high quality mark[10)]. A certain drop in quality can be seen in test 2, though, owning to the faster encoding speed settings.

Another drop in quality can be consistently noticed for sub-90 fps frame rates when

[9)]See appendix 4, with PSNR and SSIM respectively in tables I and II for test 1, tables III and IV for test 2 and tables V and VI for test 3.

[10)]That is 45+ dB for PSNR and 0.95+ for SSIM, as noted in chapter 3.10.

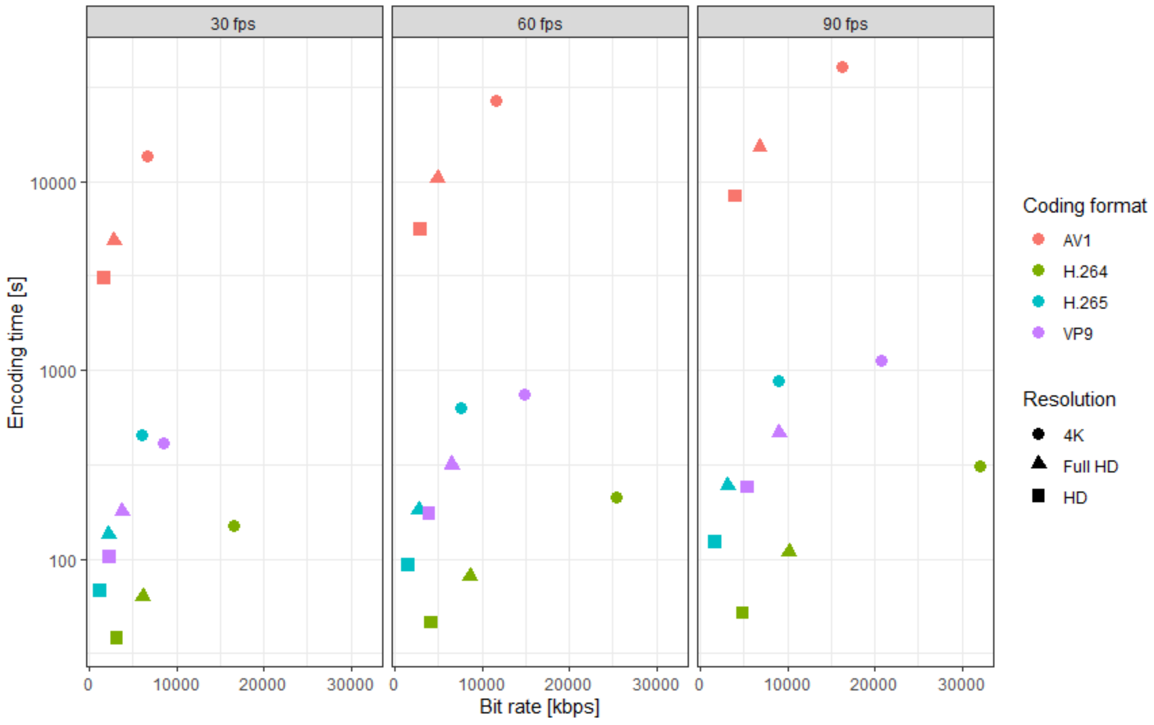

Figure 9.1 Recommended (test 1) settings

compared to the corresponding video at 90 fps, especially among the minimum values for a given metric. This decrease might be partially introduced even by the quality evaluation process itself, though, as both metrics necessitate the original video and transcoded video to be of the same resolution and frame rate before calculating the quality ratio. Because the video was transformed for lower resolution or frame rate, the original video has to be transformed accordingly for the evaluation, potentially increasing the divergence between the video feeds.

In test 3, the effects of increasing the CPU available for the encoding were examined. For the coding formats H.264 and H.265, the encoding duration decreased linearly with the increase in available CPUs, while the resulting bit rate remained essentially constant (table 9.12).

For VP9, however, the bit rate was 10 times smaller, while the encoding time decreased to about one third of the first test. The recommended settings for VP9 are thus seemingly less predictable when the hardware is changed. A possible cause for this is the necessity to use one-pass encoding for low latency video streaming, as two-pass encoding is the recommended method in the libvpx-vp9 codec. Multiple features and settings are only available in a two-pass mode, which is unfortunately not suitable for real-time video stream encoding. [98]

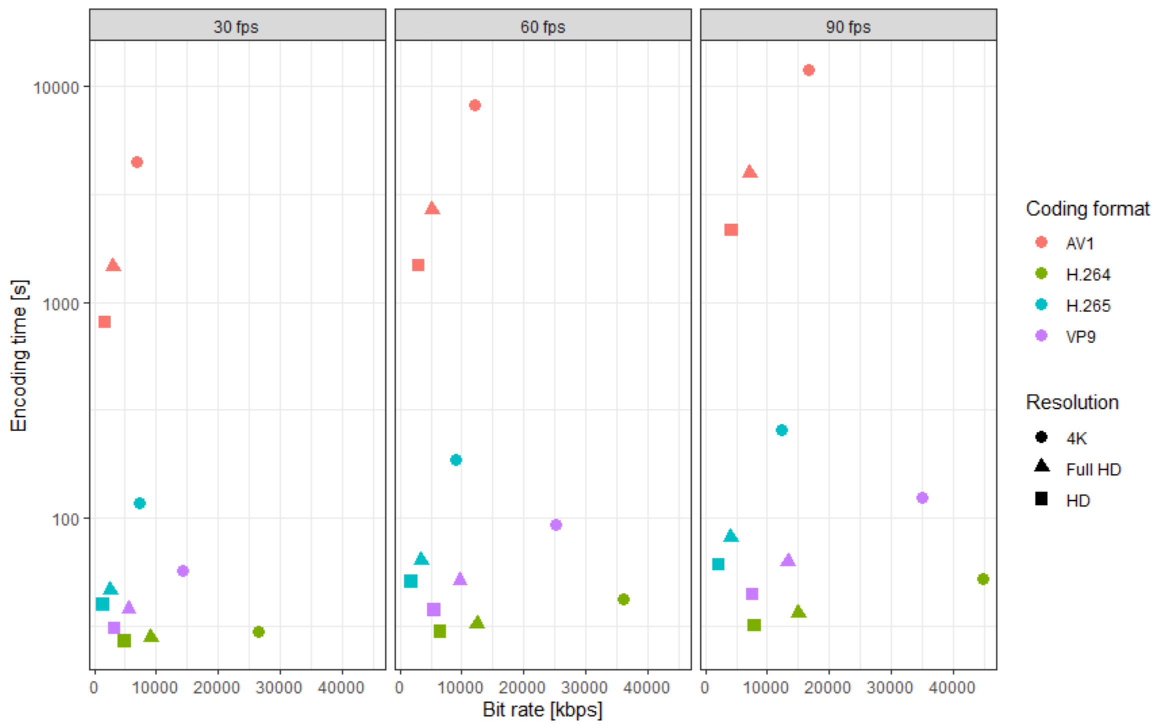

Figure 9.2 Fastest (test 2) settings

For AV1, the third test was aborted due to the extreme length of encoding. In the part of the test that was performed, the bit rate was 13 times smaller, resulting in the lowest bit rates of all codecs for that part of the test, yet the encoding time increased to 469% when compared against the corresponding values for AV1 in test 1. The recommended settings therefore vastly favour low bit rate over encoding time.

9.3 Video streaming

A video feed was streamed from a greenhouse[11)] to study the latency introduced into the video feed.

Given the hardware available in the greenhouse, a video at HD resolution and a frame rate of 30 fps was streamed. The video was produced as an uncompressed feed in the *rawvideo* format, then encoded into the select coding format and streamed to a video player[12)]. The same machine as in chapter 9.2.1 was used for the encoding.

To properly measure the latency between the time of receiving the raw video feed and the time of receiving the streamed video feed, a FFmpeg filter was used to embed

[11)]Due to the restrictions on movement during the writing of this work, a garden greenhouse was selected as a suitable equivalent environment to a hydroponic greenhouse.

[12)]The *ffplayer* utility of FFmpeg was used.

Table 9.13 Video streaming on localhost (test 4) results

Coding format	Latency (ms)	Bit rate (kbps)
H.264	1 012	9 752
H.265	1 725	2 430
VP9	3 062	4 749

a timestamp when encoding the *rawvideo* stream and a second timestamp when the streamed video was decoded for displaying[13]. The difference in the timestamps then marks the latency between capturing the video and playing it for the given frame.

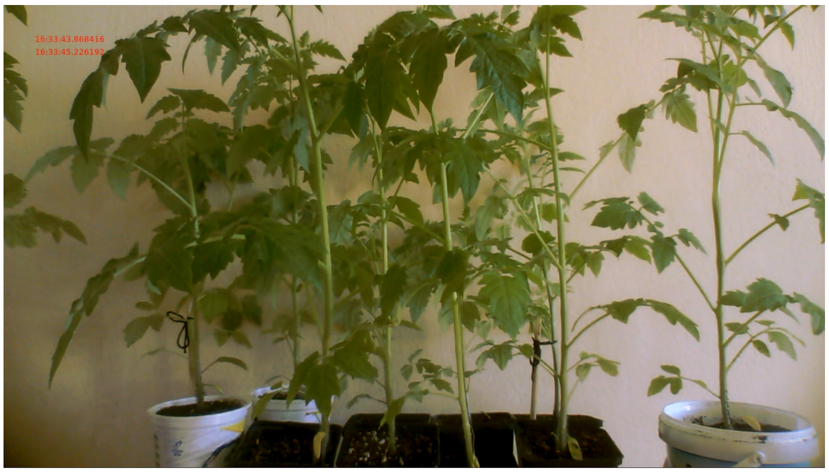

Figure 9.3 A footage of tomato plants streamed over a RTSP server, with timestamps added for measuring latency

For each coding format, ten measurements of latency were taken and averaged. Due to its very high encoding times in the previous set of tests that would make it unsuitable for low-latency streaming, AV1 was not included in these tests.

9.3.1 Test 4, Localhost streaming

The first suite of tests was intended to ascertain the base latency of the given streaming setup. The encoded videos were streamed on localhost directly to a video player[14], eliminating any network latency.

The results can be found in table 9.13. The comparative performance of the codecs can be found in tables 9.14 and 9.15.

[13] See figure 9.3.

[14] See appendix 5 for the configuration used.

Table 9.14 Average proportional difference in bit rate when comparing the coding format in the row against the coding format the in column; localhost (test 4) streaming

	H.264	H.265	VP9
H.264	–	401%	205%
H.265	25%	–	51%
VP9	49%	195%	–

Table 9.15 Average proportional difference in latency when comparing the coding format in the row against the coding format in the column; localhost (test 4) streaming

	H.264	H.265	VP9
H.264	–	59%	33%
H.265	170%	–	56%
VP9	303%	178%	–

9.3.2 Test 5, RTSP streaming

For the second suite of tests, a containerized RTSP server [99] was set up on Digital Ocean, to find any latency difference between streaming on localhost and over a RTSP channel[15]. The average ping of the server during the test was 26.3 ms, with the highest recorded ping of 38.8 ms. The network had an upload speed of 9.92 Mbps and a download speed of 28.37 Mbps, as measured by the Global Broadband Speed Test. [100]

The results can be found in table 9.16. The comparative performance of the codecs can be found in tables 9.17 and 9.18.

[15] See appendix 6 for the configuration used.

Table 9.16 Video streaming over RTSP server (test 5) results

Coding format	Latency (ms)	Bit rate (kbps)
H.264	1 265	9 554
H.265	1 972	2 176
VP9	3 668	4 317

Table 9.17 Average proportional difference in bit rate when comparing the coding format in the row against the coding format the in column; RTSP (test 5) streaming

	H.264	H.265	VP9
H.264	–	439%	221%
H.265	23%	–	50%
VP9	45%	198%	–

Table 9.18 Average proportional difference in latency when comparing the coding format in the row against the coding format in the column; RTSP (test 5) streaming

	H.264	H.265	VP9
H.264	–	58%	34%
H.265	172%	–	59%
VP9	290%	169%	–

9.3.3 Test 6, Bit rate difference

During the previous two tests, all codecs have shown a notable differences in bit rates for individual video streams, even though the settings were not changed. This was especially in contrast to the stable bit rates of H.265 in tests 1 and 3. Therefore, a hypothesis was formed that the environment during the streaming has a strong effect on the resultant bit rate; fast movements and brightness changes in particular.

In this test, the hypothesis was examined by comparing the bit rate of a video stream of an unchanging environment (a wall in a room with a constant level of lighting) against a stream of environment with a high amount of movement and brightness changes[16]. The results can be found in table 9.19, with the unchanging environment values labeled as minimum bit rate and the latter environment labeled as maximum bit rate.

[16] Rapid, abrupt movements of the camera and the lighting being repeatedly turned off and on were used.

Table 9.19 Bit rate difference based on environment (test 6)

Coding format	Bit rate min (kpbs)	Bit rate max (kpbs)	Percentage difference
H.264	9 082	14 677	62%
H.265	2 145	3 449	61%
VP9	4 174	6 824	63%

Table 9.20 Proportional difference in bit rate when comparing streaming (test 4) against transcoding (test 2) with the same settings

Coding format	Bit rate
H.264	200%
H.265	173%
VP9	148%

Table 9.21 Proportional difference in latency when comparing RTSP (test 5) against localhost (test 4) streaming

Coding format	Latency
H.264	125%
H.265	114%
VP9	120%

9.3.4 Video streaming results

In this set of tests, the aim was to compare the latency of encoding a live video stream between the coding formats, plus find other sources of latency and possible ways to reduce or remove them.

The comparative differences between H.264 and H.265 closely mirrored those observed in test 2, with H.264 achieving a lower latency at the cost of a large increase in bit rate. The resultant bit rate for both was higher than in the transcoding test (table 9.20), though, owning to the use of a setting for low-latency streaming, *-tune zerolatency*. Without zero latency set, the bit rates produced were mostly in line with the transcoding test, but the encoding duration increased to nearly triple the results in tests 4 and 5.

On the other hand, VP9 performed significantly slower when encoding live video stream than when transcoding a video, even though the same settings were applied. Further investigation into the reasons for this might be warranted[17)], but overall VP9 proved to offer worse results than H.265.

The latency achieved even in test 4 did not reach the stated goal of sub-20 ms latency for remote reality live stream [11], but between 2 and 5 times lower encoding duration should be reachable with hardware acceleration[18)] on the same machine, with further speed-up possible with a better machine, as shown in test 3.

[17)]Though once again, the limited settings of one-pass encoding in libvpx-vp9 might be at blame.

[18)]See chapter 7.3.1.

Table 9.22 Comparison of encoding stereoscopic vs monoscopic video, with a proportional comparison of bit rate and encoding duration

Coding format	File size (MB)	Bit rate (kbps)	Bit rate vs monosc.	Encoding duration (minutes)	Enc. duration vs monosc.
H.264	28.2	12 239	199%	1.87	176%
H.265	10.0	4 377	200%	4.50	201%
VP9	17.4	7 556	200%	6.30	211%
AV1	13.0	5 651	200%	163.71 (2.73 hrs)	201%

While the added latency from streaming over a remote server in test 5 was still outweighted by the latency introduced via encoding the video (table 9.21), network latency is much harder to counter than an encoding latency. Given that the average ping of the streaming server was higher than the ideal latency for remote reality streaming, network latency might prove a notable obstacle to a smooth video stream into a virtual reality headset.

Test 6 has demonstrated a strong correlation between increased level of movement or brightness change and an increase in the bit rate of a video. All tested coding formats have shown nearly two-third increase in bit rate when the feed captured a rapidly changing environment, which could prove problematic with a limited streaming bandwidth.

This problem can be offset by configuring a maximum bit rate[19)], though the quality of the video will suffer from the forced heavier compression, likely resulting in a motion blur.

9.4 Stereoscopic video

To verify the hypothesis that encoding a stereoscopic video will double the workload of the codec[20)], a stereoscopic video was created by combining two video feeds into a single video file, then transcoded[21)] on the same machine as described in chapter 9.2.1. Only stereoscopic videos in Full HD resolution with a frame rate of 30 fps were used, to reduce the duration of the test. The resultant data were compared with the corresponding values for the monoscopic videos from test 1.

The benchmark test has shown that both the bit rate and the encoding time of the

[19)] All tested codecs offer a bounded bit rate setting.

[20)] See chapter 3.8.

[21)] See appendix 7 for the script used.

stereoscopic videos are nearly exactly double of the monoscopic videos, as shown in table 9.22.

Therefore, if a stereoscopic video is to be streamed and assuming bandwidth cannot be easily changed at the location, the video resolution or frame rate has to be decreased by half to maintain the same latency level of the stream.

CONCLUSION

This thesis set out to investigate alternate solutions and technological options in support of a project aiming to construct a greenhouse monitoring robot[22] and to propose a real-time video streaming method for use in this remote reality monitoring tool. The comparison presented in this work will hopefully help in the progress of the Project and set the course for its further development.

An overview of the current state of remote reality[23] and of contemporary solutions for smart greenhouse farming[24] was compiled and contrasted with the Project. A set of considerations for live streaming and computer-mediated reality was identified[25] and the requirements for a high quality user experience in remote reality were determined[22]. Furthermore, a suggestion for the minimum characteristics of the video streamed from BERABOT was established[26].

Finally, coding formats were selected for experimental comparison[27] and a set of tests including a video transmission and streaming from a greenhouse was used to determine the coding format most suitable as a technological solution for the Project, given the previously established considerations[28].

H.265 High Efficiency Video Coding was discovered to be the most likely candidate for low-latency real-time video streaming, as it reliably achieves low bit rates with fast encoding speed and high video quality, and therefore offers a good compromise between the two main concerns of a remote reality video streaming setup.

H.264 Advanced Video Coding proved the fastest video encoder, but the efficiency of its compression turned out too low to deliver high resolution content without greatly increased bandwidth requirements.

VP9 produced somewhat unreliable results. Overall, its performance did not meet the standards set by H.265, though it remains its main contender for an efficient modern coding format.

AV1 has shown to be not ready for real-time encoding, as it provided a very good com-

22) See chapter 1.
23) See chapter 6.
24) See chapter 5.
25) See chapter 2.
26) See chapter 8.
27) See chapter 7.1.
28) See the Technical Report.

pression level, but with an extremely long encoding duration. With the development of all tested codecs ongoing, though, this situation may change in the future.

While the criteria for a remote reality video stream were not met in this work, a course for future work was found. Another set of tests using a selection of hardware encoders from different vendors is recommended, to determine the best possible encoding speed with hardware acceleration.

In addition, as FFmpeg offers a very large number of settings for each codec, further tests with different encoder settings could be expanded upon once the actual limits of bandwidth in the greenhouse where BERABOT shall be deployed are known, to determine the best possible setting combination for the specific limits placed on the video stream.

REFERENCES

[1] Gericke, W. F.: HYDROPONICS–CROP PRODUCTION IN LIQUID CULTURE MEDIA. *Science*, volume 85, no. 2198, February 1937: pp. 177–178, ISSN 0036-8075, 1095-9203, doi:10.1126/science.85.2198.177.
URL `https://www.sciencemag.org/lookup/doi/10.1126/science.85.2198.177`

[2] Resh, H. M.: *Hydroponic Food Production A Definitive Guidebook for the Advanced Home Gardener and the Commercial Hydroponic Grower, Seventh Edition*. 2016, ISBN 978-1-4398-7869-9, oCLC: 957230513.

[3] Story, D.; Kacira, M.: Design and implementation of a computer vision-guided greenhouse crop diagnostics system. *Machine Vision and Applications*, volume 26, no. 4, May 2015: pp. 495–506, ISSN 0932-8092, 1432-1769, doi:10.1007/s00138-015-0670-5.
URL `http://link.springer.com/10.1007/s00138-015-0670-5`

[4] Baker, M.: Survey Reveals 82% of Company Leaders Plan to Allow Employees to Work Remotely Some of the Time.
URL `https://www.gartner.com/en/newsroom/press-releases/2020-07-14-gartner-survey-reveals-82-percent-of-company-leaders-plan-to-allow-employees-to-work-remotely-some-of-the-time`

[5] TA ČR: Starfos: Inteligentní robotická ochrana zdraví ekosystému hydroponického skleníku.
URL `https://starfos.tacr.cz/cs/project/FW01010381`

[6] TA ČR: Technology Agency of the Czech Republic.
URL `https://www.tacr.cz/en/`

[7] Kaleta, M.: BERABOT.
URL `https://berabot.com/`

[8] Boult, T.: Remote Reality Overview.
URL `https://vast.uccs.edu/~tboult/remote-reality.html`

[9] Lederer, S.: 2019 Video Developer Report - The Future of Video: AV1 Codec, AI & Machine Learning, and Low Latency. September 2019, section: Blog Post.
URL `https://bitmovin.com/bitmovin-2019-video-developer-report-av1-codec-ai-machine-learning-low-latency/`

[10] Stauffert, J.-P.; Niebling, F.; Latoschik, M. E.: Effects of Latency Jitter on Simulator Sickness in a Search Task. In *2018 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, Reutlingen: IEEE, March 2018, ISBN 978-1-5386-3365-6, pp. 121–127, doi:10.1109/VR.2018.8446195.
URL `https://ieeexplore.ieee.org/document/8446195/`

[11] Valve: Latency – the sine qua non of AR and VR. August 2013.
URL `https://web.archive.org/web/20130826171927/http://blogs.valvesoftware.com/abrash/latency-the-sine-qua-non-of-ar-and-vr`

[12] Amazon Web Services (AWS): What Is Streaming Data?
URL `https://aws.amazon.com/streaming-data/`

[13] Valve: Ramblings in Valve Time. August 2013.
URL `http://blogs.valvesoftware.com/abrash/`

[14] Valve: Why virtual isn't real to your brain. August 2013.
URL `https://web.archive.org/web/20130826195424/http://blogs.valvesoftware.com/abrash/why-virtual-isnt-real-to-your-brain`

[15] Buker, T. J.; Vincenzi, D. A.; Deaton, J. E.: The Effect of Apparent Latency on Simulator Sickness While Using a See-Through Helmet-Mounted Display: Reducing Apparent Latency With Predictive Compensation. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, volume 54, no. 2, April 2012: pp. 235–249, ISSN 0018-7208, 1547-8181, doi:10.1177/0018720811428734.
URL `http://journals.sagepub.com/doi/10.1177/0018720811428734`

[16] Oculus: Asynchronous Spacewarp.
URL `https://developer.oculus.com/blog/asynchronous-spacewarp/`

[17] Valve: Down the VR rabbit hole: Fixing judder. August 2013.
URL `https://web.archive.org/web/20130823111544/http://blogs.valvesoftware.com/abrash/down-the-vr-rabbit-hole-fixing-judder/`

[18] Ozcelik, I. M.; Ersoy, C.: Low-Latency Live Streaming Over HTTP in Bandwidth-Limited Networks. *IEEE Communications Letters*, volume 25, no. 2, February 2021: pp. 450–454, ISSN 1089-7798, 1558-2558, 2373-7891, doi:10.1109/LCOMM.2020.3030887.
URL `https://ieeexplore.ieee.org/document/9223713/`

[19] Valve: Why virtual isn't real to your brain: judder. August 2013.
URL `https://web.archive.org/web/20130827003420/http://blogs.`

valvesoftware.com/abrash/why-virtual-isnt-real-to-your-brain-judder/

[20] Merriam-Webster: Definition of CODEC.
URL https://www.merriam-webster.com/dictionary/codec

[21] Matroška: What is Matroska?
URL https://matroska.org/what_is_matroska.html

[22] Li, D.; Xu, L.; Tang, X.-s.; et al.: 3D Imaging of Greenhouse Plants with an Inexpensive Binocular Stereo Vision System. *Remote Sensing*, volume 9, no. 5, May 2017: p. 508, ISSN 2072-4292, doi:10.3390/rs9050508.
URL http://www.mdpi.com/2072-4292/9/5/508

[23] Vetro, A.; Wiegand, T.; Sullivan, G. J.: Overview of the Stereo and Multiview Video Coding Extensions of the H.264/MPEG-4 AVC Standard. *Proceedings of the IEEE*, volume 99, no. 4, April 2011: pp. 626–642, ISSN 0018-9219, 1558-2256, doi:10.1109/JPROC.2010.2098830.
URL http://ieeexplore.ieee.org/document/5705534/

[24] Encyclopedia Britannica: Protocol (computer science).
URL https://www.britannica.com/technology/protocol-computer-science

[25] Watequlis S. et al.: Study of Performance of Real Time Streaming Protocol (RTSP) in Learning Systems. *International Journal of Engineering Technology*, volume 7: p. 216, ISSN 2227-524X, doi:10.14419/ijet.v7i4.44.26994.
URL https://www.sciencepubco.com/index.php/ijet/article/view/26994

[26] Chikkerur, S.; Sundaram, V.; Reisslein, M.; et al.: Objective Video Quality Assessment Methods: A Classification, Review, and Performance Comparison. *IEEE Transactions on Broadcasting*, volume 57, no. 2, june 2011: pp. 165–182, ISSN 0018-9316, 1557-9611, doi:10.1109/TBC.2011.2104671.
URL http://ieeexplore.ieee.org/document/5710601/

[27] Huynh-Thu, Q.; Ghanbari, M.: Scope of validity of PSNR in image/video quality assessment. *Electronics Letters*, volume 44, no. 13, 2008: p. 800, ISSN 00135194, doi:10.1049/el:20080522.
URL https://digital-library.theiet.org/content/journals/10.1049/el_20080522

[28] Ozer, J.: How to Choose and Use Objective Video Quality Benchmarks. December 2017.

URL `https://www.streamingmedia.com/Articles/ReadArticle.aspx?ArticleID=122050&pageNum=2`

[29] Wang, Z.; Bovik, A.; Sheikh, H.; et al.: Image Quality Assessment: From Error Visibility to Structural Similarity. *IEEE Transactions on Image Processing*, volume 13, no. 4, April 2004: pp. 600–612, ISSN 1057-7149, doi:10.1109/TIP.2003.819861.
URL `http://ieeexplore.ieee.org/document/1284395/`

[30] Ozer, J.: Mapping SSIM and VMAF Scores to Subjective Ratings. July 2018.
URL `https://streaminglearningcenter.com/learning/mapping-ssim-vmaf-scores-subjective-ratings.html`

[31] Netflix: VMAF. April 2021, original-date: 2016-02-08T18:41:38Z.
URL `https://github.com/Netflix/vmaf`

[32] Meola, A.: Smart Farming in 2020: How IoT sensors are creating a more efficient precision agriculture industry.
URL `https://www.businessinsider.com/smart-farming-iot-agriculture`

[33] Hetzroni, A.; Miles, G.; Engel, B.; et al.: Machine vision monitoring of plant health. *Advances in Space Research*, volume 14, no. 11, November 1994: pp. 203–212, ISSN 02731177, doi:10.1016/0273-1177(94)90298-4.
URL `https://linkinghub.elsevier.com/retrieve/pii/0273117794902984`

[34] Polder, G.; van der Heijden, G.; Young, I.: Tomato sorting using independent component analysis on spectral images. *Real-Time Imaging*, volume 9, no. 4, August 2003: pp. 253–259, ISSN 10772014, doi:10.1016/j.rti.2003.09.008.
URL `https://linkinghub.elsevier.com/retrieve/pii/S107720140300069X`

[35] Altavian: What is CIR Imagery and what is it used for?
URL `https://www.altavian.com/blog/2016/8/cir-imagery`

[36] Blackburn, G. A.: Hyperspectral remote sensing of plant pigments. *Journal of Experimental Botany*, volume 58, no. 4, November 2006: pp. 855–867, ISSN 0022-0957, 1460-2431, doi:10.1093/jxb/erl123.
URL `https://academic.oup.com/jxb/article-lookup/doi/10.1093/jxb/erl123`

[37] Shimadzu: Near-Infrared Region Measurement and Related Considerations.
URL `https://www.shimadzu.com/an/service-support/technical-support/analysis-basics/tips-ftir/nir1.html`

[38] Ray, D. K.; Mueller, N. D.; West, P. C.; et al.: Yield Trends Are Insufficient to Double Global Crop Production by 2050. *PLoS ONE*, volume 8, no. 6, june 2013: p. e66428, ISSN 1932-6203, doi:10.1371/journal.pone.0066428.
URL `https://dx.plos.org/10.1371/journal.pone.0066428`

[39] Food and Agriculture Organization of the United Nations (editor): *The future of food and agriculture: trends and challenges*. Rome: Food and Agriculture Organization of the United Nations, 2017, ISBN 978-92-5-109551-5, oCLC: ocn979567879.

[40] Tomlinson, I.: Doubling food production to feed the 9 billion: A critical perspective on a key discourse of food security in the UK. *Journal of Rural Studies*, volume 29, January 2013: pp. 81–90, ISSN 07430167, doi:10.1016/j.jrurstud.2011.09.001.
URL `https://linkinghub.elsevier.com/retrieve/pii/S0743016711000830`

[41] Walters, K. J.; Behe, B. K.: Historical, Current, and Future Perspectives for Controlled Environment Hydroponic Food Crop Production in the United States. volume 55, 2020: p. 10.

[42] Pulighe, G.; Lupia, F.: Food First: COVID-19 Outbreak and Cities Lockdown a Booster for a Wider Vision on Urban Agriculture. *Sustainability*, volume 12, no. 12, june 2020: p. 5012, ISSN 2071-1050, doi:10.3390/su12125012.
URL `https://www.mdpi.com/2071-1050/12/12/5012`

[43] Leader, I.: Metomotion's Robotic Tomato Harvester. April 2020.
URL `https://irrigationleadermagazine.com/metomotions-robotic-tomato-harvester/`

[44] Horizon 2020.
URL `https://ec.europa.eu/programmes/horizon2020/en`

[45] METOMOTION: Greenhouse Robotic Worker.
URL `https://metomotion.com/`

[46] iUNU: Precision Agriculture For Indoor Growing.
URL `https://iunu.com/`

[47] Phenospex: PlantEye F500 - Multispectral 3D laser scanner for plant phenotyping.
URL `https://phenospex.com/products/plant-phenotyping/planteye-f500-multispectral-3d-laser-scanner/`

[48] Phenospex: HortControl – Plant Data Management Software.
URL `https://phenospex.com/products/plant-phenotyping/science-hortcontrol-data-management-software/`

[49] Phenospex: TraitFinder.
URL `https://phenospex.com/products/plant-phenotyping/traitfinder-for-lab-and-greenhouse-phenotyping-automation/`

[50] Phenospex: MicroScan for small digital phenotyping tasks.
URL `https://phenospex.com/microscan-for-small-digital-phenotyping-tasks/`

[51] Phenospex: FieldScan.
URL `https://phenospex.com/products/plant-phenotyping/fieldscan-high-throughput-field-phenotyping/`

[52] Twitter: Root AI (@RootRobots).
URL `https://twitter.com/RootRobots`

[53] Root AI.
URL `https://root-ai.com/`

[54] Coombes, C. E.; Gregory, M. E.: Current and Future Use of Telemedicine in Infectious Diseases Practice. *Current Infectious Disease Reports*, volume 21, no. 11, November 2019: p. 41, ISSN 1523-3847, 1534-3146, doi:10.1007/s11908-019-0697-2.
URL `http://link.springer.com/10.1007/s11908-019-0697-2`

[55] AMD Telemedicine: Intraoral Dental Camera | Oral Care Device. May 2020.
URL `https://amdtelemedicine.com/product/intraoral-dental-camera/`

[56] Firefly Global: Telemedicine Examination Camera.
URL `https://fireflyglobal.com/de605-general-examination-camera/`

[57] GlobalMed: i1000MD Video Conferencing Camera.
URL `https://www.globalmed.com/solutions/connected-devices/conference-cameras/i1000md-camera/`

[58] GlobalMed: TotalExam 3.2.
URL `https://www.globalmed.com/wp-content/uploads/2021/01/Product-Sheet_TotalExam3.2_Jan2021-1.pdf`

[59] GlobalMed: TotalExam HD.
URL `https://www.globalmed.com/wp-content/uploads/manuals/MAN-600040-TEHD-user.pdf`

[60] MedicalExpo: GEIS - Teleconsultation camera by Visionflex.
URL `https://www.medicalexpo.com/prod/visionflex/product-127373-941685.html`

[61] MedicalExpo: TotalExam® HD Camera - GlobalMed.
URL `https://pdf.medicalexpo.com/pdf/globalmed/totalexam-hd-camera/82214-168187.html`

[62] Hewlett-Packard: HP Reverb G2 VR Headset.
URL `https://www8.hp.com/us/en/vr/reverb-g2-vr-headset.html`

[63] HTC: VIVE.
URL `https://www.vive.com/us/product/`

[64] HTC: VIVE Pro.
URL `https://www.vive.com/us/product/`

[65] HTC: VIVE Cosmos.
URL `https://www.vive.com/us/product/vive-cosmos/overview/`

[66] Oculus: Oculus Rift S: VR Headset for VR-ready PCs.
URL `https://www.oculus.com/rift-s/`

[67] Oculus: Oculus Go: Stand-alone VR Headset.
URL `https://www.oculus.com/go/`

[68] Niora: Oculus Rift CV1 - Review.
URL `https://www.niora.net/en/p/oculus_rift_cv1`

[69] Oculus: Oculus Go: Stand-alone VR Headset.
URL `https://www.oculus.com/go/`

[70] Sony: PlayStation VR | Live the game in incredible virtual reality worlds.
URL `https://www.playstation.com/en-us/ps-vr/`

[71] Oculus: Asynchronous Timewarp Examined.
URL `https://developer.oculus.com/blog/asynchronous-timewarp-examined/`

[72] Oculus: Gear VR Powered by Oculus: A Portable VR Headset With Smartphone.
URL `https://www.oculus.com/gear-vr/`

[73] Google: Daydream.
URL `https://arvr.google.com/daydream/`

[74] IndustryResearch: Virtual Reality (VR) Market 2021-2024 | Insight by Recent Developments, Emerging Trends, Overview by Industry Share and Size Forecast to 2024 – Clark County Blog.
URL `https://clarkcountyblog.com/uncategorized/248255/virtual-reality-vr-market-2021-2024-insight-by-recent-developments-emerging-trends-overview-by-industry-share-and-size-forecast-to-2024/`

[75] Koksal, I.: Google Discontinues The Daydream VR Headset. Section: Enterprise Tech.
URL `https://www.forbes.com/sites/ilkerkoksal/2020/10/03/google-discontinues-the-daydream-vr-headset/`

[76] 3DMaster: Why the era of Samsung Gear VR is coming to an end.
URL `https://viscircle.de/why-the-era-of-samsung-gear-vr-is-coming-to-an-end/?lang=en`

[77] Robertson, A.: Phone-based VR is officially over. October 2019.
URL `https://www.theverge.com/2019/10/16/20915791/google-daydream-samsung-oculus-gear-vr-mobile-vr-platforms-dead`

[78] Ozer, J.: AV1 Is At Least 70% Faster Than I Thought. July 2020.
URL `https://streaminglearningcenter.com/codecs/av1-is-at-least-70-faster-than-i-thought.html`

[79] OSnews: US Patent Expiration for MP3, MPEG-2, H.264.
URL `https://www.osnews.com/story/24954/us-patent-expiration-for-mp3-mpeg-2-h264/`

[80] Ozer, J.: HEVC Advance Cuts Content Fees on Streaming. March 2018.
URL `https://www.streamingmedia.com/Articles/ReadArticle.aspx?ArticleID=123828`

[81] JVET: Joint Video Experts Team.
URL `https://www.itu.int:443/en/ITU-T/studygroups/2017-2020/16/Pages/video/jvet.aspx`

[82] ScreenRant: H.266 Codec Streams 4K Video Using 50% Less Data. July 2020, section: Tech.

URL `https://screenrant.com/h266-4k-8k-video-codec-data-usage-explained/`

[83] Kašpar, M.: Bandwidth calculator | CCTV Calculator.
URL `https://www.cctvcalculator.net/en/calculations/bandwidth-calculator/`

[84] Arxys: NVR Bandwidth And Storage Calculator Video Surveillance.
URL `https://www.arxys.com/bandwidth-storage-calculator/`

[85] Google, YouTube: Recommended upload encoding settings.
URL `https://support.google.com/youtube/answer/1722171?hl=en`

[86] Twitch: Twitch Video Encoding/Bitrates.
URL `https://stream.twitch.tv/encoding/`

[87] Wowza: Encoding best practices for Wowza Streaming Cloud.
URL `https://www.wowza.com/docs/how-to-encode-source-video-for-wowza-streaming-cloud`

[88] Dacast: VOD Video Renditions & Bitrates for On Demand Content.
URL `https://www.dacast.com/support/knowledgebase/vod-renditions-bitrates/`

[89] Dacast: What is Live Video Cloud Encoding? April 2021.
URL `/blog/cloud-transcoding/`

[90] IMB: Internet connection and recommended encoding settings.
URL `https://support.video.ibm.com/hc/en-us/articles/207852117-Internet-connection-and-recommended-encoding-settings`

[91] Arzumanyan, R.: Turing H.264 Video Encoding Speed and Quality. February 2019.
URL `https://developer.nvidia.com/blog/turing-h264-video-encoding-speed-and-quality/`

[92] Blinov, V.: Intel vs Nvidia: a comparison of video encoding using GPUs. *Elecard: Video Compression Guru.*
URL `https://www.elecard.com/page/article_intel_vs_nvidia`

[93] FFmpeg: Hardware Acceleration Intro.
URL `https://trac.ffmpeg.org/wiki/HWAccelIntro`

[94] Dacast: Cloud Video Encoding vs. Transcoding Services To Know in 2021. May 2021.
URL `https://www.dacast.com/blog/auto-transcoding-cloud-video-encoding-software/`

[95] Xtremerides: Go-Pro 5 Ultra HD 4K 90FPS on waterslides (test video). 2017.
URL `https://www.youtube.com/watch?v=1QYbdmhunow`

[96] Wickham, H.: *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York, 2016, ISBN 978-3-319-24277-4.
URL `https://ggplot2.tidyverse.org`

[97] Ozer, J.: Netflix Finds x265 20% More Efficient than VP9. September 2016.
URL `https://www.streamingmedia.com/Articles/ReadArticle.aspx?ArticleID=113346`

[98] FFmpeg: Encode VP9.
URL `https://trac.ffmpeg.org/wiki/Encode/VP9`

[99] Ros, A.: aler9/rtsp-simple-server. April 2021, original-date: 2019-12-28T20:08:43Z.
URL `https://github.com/aler9/rtsp-simple-server`

[100] Ookla: The Global Broadband Speed Test.
URL `https://www.speedtest.net/`

LIST OF ABBREVIATIONS

bps	Bits per second
CIR	Colour-infrared imaging
CPU	Central processing unit
fps	Frames per second
GPU	Graphics processing unit
NIR	Near-infrared imaging
PSNR	Peak signal-to-noise ratio
px	Pixel
RGB	Red, green, blue
RTP	Real-time transport protocol
RTSP	Real-time streaming protocol
SSIM	Structural similarity index measure
VMAF	Video multi-method assessment fusion

LIST OF FIGURES

LIST OF TABLES

LIST OF APPENDICES

APPENDIX A I. TEST 1 BENCHMARK SCRIPT

test1_bench.sh

```
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -report -benchmark -c:v
    libx264 Slide_4K_90fps_h264.mkv
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -report -benchmark -c:v
    libx264 -filter:v "fps=60" Slide_4K_60fps_h264.mkv
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -report -benchmark -c:v
    libx264 -filter:v "fps=30" Slide_4K_30fps_h264.mkv

ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -report -benchmark -c:v
    libx265 Slide_4K_90fps_h265.mkv
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -report -benchmark -c:v
    libx265 -filter:v "fps=60" Slide_4K_60fps_h265.mkv
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -report -benchmark -c:v
    libx265 -filter:v "fps=30" Slide_4K_30fps_h265.mkv

ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -report -benchmark -c:v
    libvpx-vp9 Slide_4K_90fps_vp9.mkv
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -report -benchmark -c:v
    libvpx-vp9 -filter:v "fps=60" Slide_4K_60fps_vp9.mkv
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -report -benchmark -c:v
    libvpx-vp9 -filter:v "fps=30" Slide_4K_30fps_vp9.mkv

ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -report -benchmark -c:v
    libaom-av1 -strict -2 Slide_4K_90fps_av1.mkv
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -report -benchmark -c:v
    libaom-av1 -strict -2 -filter:v "fps=60" Slide_4K_60fps_av1.mkv
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -report -benchmark -c:v
    libaom-av1 -strict -2 -filter:v "fps=30" Slide_4K_30fps_av1.mkv

ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -report -benchmark -c:v
    libx264 -filter:v "scale=-1:1080" Slide_FullHD_90fps_h264.mkv
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -report -benchmark -c:v
    libx264 -filter:v "scale=-1:1080,fps=60" Slide_FullHD_60fps_h264.
    mkv
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -report -benchmark -c:v
    libx264 -filter:v "scale=-1:1080,fps=30" Slide_FullHD_30fps_h264.
    mkv

ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -report -benchmark -c:v
    libx265 -filter:v "scale=-1:1080" Slide_FullHD_90fps_h265.mkv
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -report -benchmark -c:v
    libx265 -filter:v "scale=-1:1080,fps=60" Slide_FullHD_60fps_h265.
    mkv
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -report -benchmark -c:v
    libx265 -filter:v "scale=-1:1080,fps=30" Slide_FullHD_30fps_h265.
    mkv

ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -report -benchmark -c:v
    libvpx-vp9 -filter:v "scale=-1:1080" Slide_FullHD_90fps_vp9.mkv
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -report -benchmark -c:v
    libvpx-vp9 -filter:v "scale=-1:1080,fps=60" Slide_FullHD_60fps_vp9
    .mkv
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -report -benchmark -c:v
    libvpx-vp9 -filter:v "scale=-1:1080,fps=30" Slide_FullHD_30fps_vp9
    .mkv

ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -report -benchmark -c:v
    libaom-av1 -strict -2 -filter:v "scale=-1:1080"
    Slide_FullHD_90fps_av1.mkv
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -report -benchmark -c:v
    libaom-av1 -strict -2 -filter:v "scale=-1:1080,fps=60"
    Slide_FullHD_60fps_av1.mkv
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -report -benchmark -c:v
    libaom-av1 -strict -2 -filter:v "scale=-1:1080,fps=30"
    Slide_FullHD_30fps_av1.mkv

ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -report -benchmark -c:v
    libx264 -filter:v "scale=-1:720" Slide_HD_90fps_h264.mkv
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -report -benchmark -c:v
    libx264 -filter:v "scale=-1:720,fps=60" Slide_HD_60fps_h264.mkv
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -report -benchmark -c:v
    libx264 -filter:v "scale=-1:720,fps=30" Slide_HD_30fps_h264.mkv

ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -report -benchmark -c:v
    libx265 -filter:v "scale=-1:720" Slide_HD_90fps_h265.mkv
```

```
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -report -benchmark -c:v libx265 -filter:v "scale=-1:720,fps=60" Slide_HD_60fps_h265.mkv
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -report -benchmark -c:v libx265 -filter:v "scale=-1:720,fps=30" Slide_HD_30fps_h265.mkv

ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -report -benchmark -c:v libvpx-vp9 -filter:v "scale=-1:720" Slide_HD_90fps_vp9.mkv
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -report -benchmark -c:v libvpx-vp9 -filter:v "scale=-1:720,fps=60" Slide_HD_60fps_vp9.mkv
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -report -benchmark -c:v libvpx-vp9 -filter:v "scale=-1:720,fps=30" Slide_HD_30fps_vp9.mkv

ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -report -benchmark -c:v libaom-av1 -strict -2 -filter:v "scale=-1:720" Slide_HD_90fps_av1.mkv
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -report -benchmark -c:v libaom-av1 -strict -2 -filter:v "scale=-1:720,fps=60" Slide_HD_60fps_av1.mkv
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -report -benchmark -c:v libaom-av1 -strict -2 -filter:v "scale=-1:720,fps=30" Slide_HD_30fps_av1.mkv
```

APPENDIX A II. TEST 2 BENCHMARK SCRIPT

test2_bench.sh

```
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -report -benchmark -preset ultrafast -tune zerolatency -c:v libx264 Slide_4K_90fps_h264.mkv
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -report -benchmark -preset ultrafast -tune zerolatency -c:v libx264 -filter:v "fps=60" Slide_4K_60fps_h264.mkv
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -report -benchmark -preset ultrafast -tune zerolatency -c:v libx264 -filter:v "fps=30" Slide_4K_30fps_h264.mkv

ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -report -benchmark -preset ultrafast -tune zerolatency -c:v libx265 Slide_4K_90fps_h265.mkv
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -report -benchmark -preset ultrafast -tune zerolatency -c:v libx265 -filter:v "fps=60" Slide_4K_60fps_h265.mkv
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -report -benchmark -preset ultrafast -tune zerolatency -c:v libx265 -filter:v "fps=30" Slide_4K_30fps_h265.mkv

ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -report -benchmark -deadline realtime -cpu-used 8 -c:v libvpx-vp9 Slide_4K_90fps_vp9.mkv
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -report -benchmark -deadline realtime -cpu-used 8 -c:v libvpx-vp9 -filter:v "fps=60" Slide_4K_60fps_vp9.mkv
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -report -benchmark -deadline realtime -cpu-used 8 -c:v libvpx-vp9 -filter:v "fps=30" Slide_4K_30fps_vp9.mkv

ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -report -benchmark -c:v libaom-av1 -strict -2 -cpu-used 8 -row-mt 1 Slide_4K_90fps_av1.mkv
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -report -benchmark -c:v libaom-av1 -strict -2 -cpu-used 8 -row-mt 1 -filter:v "fps=60" Slide_4K_60fps_av1.mkv
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -report -benchmark -c:v libaom-av1 -strict -2 -cpu-used 8 -row-mt 1 -filter:v "fps=30" Slide_4K_30fps_av1.mkv

ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -report -benchmark -preset ultrafast -tune zerolatency -c:v libx264 -filter:v "scale=-1:1080" Slide_FullHD_90fps_h264.mkv
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -report -benchmark -preset ultrafast -tune zerolatency -c:v libx264 -filter:v "scale=-1:1080,fps=60" Slide_FullHD_60fps_h264.mkv
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -report -benchmark -preset ultrafast -tune zerolatency -c:v libx264 -filter:v "scale=-1:1080,fps=30" Slide_FullHD_30fps_h264.mkv

ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -report -benchmark -preset ultrafast -tune zerolatency -c:v libx265 -filter:v "scale=-1:1080" Slide_FullHD_90fps_h265.mkv
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -report -benchmark -preset ultrafast -tune zerolatency -c:v libx265 -filter:v "scale=-1:1080,fps=60" Slide_FullHD_60fps_h265.mkv
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -report -benchmark -preset ultrafast -tune zerolatency -c:v libx265 -filter:v "scale=-1:1080,fps=30" Slide_FullHD_30fps_h265.mkv

ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -report -benchmark -deadline realtime -cpu-used 8 -c:v libvpx-vp9 -filter:v "scale=-1:1080" Slide_FullHD_90fps_vp9.mkv
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -report -benchmark -deadline realtime -cpu-used 8 -c:v libvpx-vp9 -filter:v "scale=-1:1080,fps=60" Slide_FullHD_60fps_vp9.mkv
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -report -benchmark -deadline realtime -cpu-used 8 -c:v libvpx-vp9 -filter:v "scale=-1:1080,fps=30" Slide_FullHD_30fps_vp9.mkv

ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -report -benchmark -c:v libaom-av1 -strict -2 -cpu-used 8 -row-mt 1 -filter:v "scale=-1:1080" Slide_FullHD_90fps_av1.mkv
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -report -benchmark -c:v libaom-av1 -strict -2 -cpu-used 8 -row-mt 1 -filter:v "scale=-1:1080,fps=60" Slide_FullHD_60fps_av1.mkv
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -report -benchmark -c:v libaom-av1 -strict -2 -cpu-used 8 -row-mt 1 -filter:v "scale=-1:1080,fps=30" Slide_FullHD_30fps_av1.mkv

ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -report -benchmark -preset ultrafast -tune zerolatency -c:v libx264 -filter:v "scale=-1:720" Slide_HD_90fps_h264.mkv
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -report -benchmark -preset ultrafast -tune zerolatency -c:v libx264 -filter:v "scale=-1:720,fps=60" Slide_HD_60fps_h264.mkv
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -report -benchmark -preset ultrafast -tune zerolatency -c:v libx264 -filter:v "scale=-1:720,fps=30" Slide_HD_30fps_h264.mkv

ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -report -benchmark -preset ultrafast -tune zerolatency -c:v libx265 -filter:v "scale=-1:720" Slide_HD_90fps_h265.mkv
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -report -benchmark -preset ultrafast -tune zerolatency -c:v libx265 -filter:v "scale=-1:720,fps=60" Slide_HD_60fps_h265.mkv
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -report -benchmark -preset ultrafast -tune zerolatency -c:v libx265 -filter:v "scale=-1:720,fps=30" Slide_HD_30fps_h265.mkv

ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -report -benchmark -deadline realtime -cpu-used 8 -c:v libvpx-vp9 -filter:v "scale=-1:720" Slide_HD_90fps_vp9.mkv
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -report -benchmark -deadline realtime -cpu-used 8 -c:v libvpx-vp9 -filter:v "scale=-1:720,fps=60" Slide_HD_60fps_vp9.mkv
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -report -benchmark -deadline realtime -cpu-used 8 -c:v libvpx-vp9 -filter:v "scale=-1:720,fps=30" Slide_HD_30fps_vp9.mkv

ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -report -benchmark -c:v libaom-av1 -strict -2 -cpu-used 8 -row-mt 1 -filter:v "scale=-1:720" Slide_HD_90fps_av1.mkv
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -report -benchmark -c:v libaom-av1 -strict -2 -cpu-used 8 -row-mt 1 -filter:v "scale=-1:720,fps=60" Slide_HD_60fps_av1.mkv
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -report -benchmark -c:v libaom-av1 -strict -2 -cpu-used 8 -row-mt 1 -filter:v "scale=-1:720,fps=30" Slide_HD_30fps_av1.mkv
```

APPENDIX A III. QUALITY METRICS SCRIPT

test_stats.sh

```
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -i Slide_4K_90fps_h264.mkv -lavfi psnr=stats_file=psnr_4K_90fps_h264.txt -f null -
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -i Slide_4K_60fps_h264.mkv -lavfi '[0]fps=60[a];[a][1]psnr=stats_file=psnr_4K_60fps_h264.txt' -f null -
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -i Slide_4K_30fps_h264.mkv -lavfi '[0]fps=30[a];[a][1]psnr=stats_file=psnr_4K_30fps_h264.txt' -f null -
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -i Slide_4K_90fps_h265.mkv -lavfi psnr=stats_file=psnr_4K_90fps_h265.txt -f null -
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -i Slide_4K_60fps_h265.mkv -lavfi '[0]fps=60[a];[a][1]psnr=stats_file=psnr_4K_60fps_h265.txt' -f null -
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -i Slide_4K_30fps_h265.mkv -lavfi '[0]fps=30[a];[a][1]psnr=stats_file=psnr_4K_30fps_h265.txt' -f null -
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -i Slide_4K_90fps_vp9.mkv -lavfi psnr=stats_file=psnr_4K_90fps_vp9.txt -f null -
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -i Slide_4K_60fps_vp9.mkv -lavfi '[0]fps=60[a];[a][1]psnr=stats_file=psnr_4K_60fps_vp9.txt' -f null -
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -i Slide_4K_30fps_vp9.mkv -lavfi '[0]fps=30[a];[a][1]psnr=stats_file=psnr_4K_30fps_vp9.txt' -f null -
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -i Slide_4K_90fps_av1.mkv -lavfi psnr=stats_file=psnr_4K_90fps_av1.txt -f null -
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -i Slide_4K_60fps_av1.mkv -lavfi '[0]fps=60[a];[a][1]psnr=stats_file=psnr_4K_60fps_av1.txt' -f null -
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -i Slide_4K_30fps_av1.mkv -lavfi '[0]fps=30[a];[a][1]psnr=stats_file=psnr_4K_30fps_av1.txt' -f null -
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -i Slide_FullHD_90fps_h264.mkv -lavfi '[0]scale=-1:1080[a];[a][1]psnr=stats_file=psnr_FullHD_90fps_h264.txt' -f null -
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -i Slide_FullHD_60fps_h264.mkv -lavfi '[0]scale=-1:1080,fps=60[a];[a][1]psnr=stats_file=psnr_FullHD_60fps_h264.txt' -f null -
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -i Slide_FullHD_30fps_h264.mkv -lavfi '[0]scale=-1:1080,fps=30[a];[a][1]psnr=stats_file=psnr_FullHD_30fps_h264.txt' -f null -
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -i Slide_FullHD_90fps_h265.mkv -lavfi '[0]scale=-1:1080[a];[a][1]psnr=stats_file=psnr_FullHD_90fps_h265.txt' -f null -
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -i Slide_FullHD_60fps_h265.mkv -lavfi '[0]scale=-1:1080,fps=60[a];[a][1]psnr=stats_file=psnr_FullHD_60fps_h265.txt' -f null -
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -i Slide_FullHD_30fps_h265.mkv -lavfi '[0]scale=-1:1080,fps=30[a];[a][1]psnr=stats_file=psnr_FullHD_30fps_h265.txt' -f null -
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -i Slide_FullHD_90fps_vp9.mkv -lavfi '[0]scale=-1:1080[a];[a][1]psnr=stats_file=psnr_FullHD_90fps_vp9.txt' -f null -
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -i Slide_FullHD_60fps_vp9.mkv -lavfi '[0]scale=-1:1080,fps=60[a];[a][1]psnr=stats_file=psnr_FullHD_60fps_vp9.txt' -f null -
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -i Slide_FullHD_30fps_vp9.mkv -lavfi '[0]scale=-1:1080,fps=30[a];[a][1]psnr=stats_file=psnr_FullHD_30fps_vp9.txt' -f null -
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -i Slide_FullHD_90fps_av1.mkv -lavfi '[0]scale=-1:1080[a];[a][1]psnr=stats_file=psnr_FullHD_90fps_av1.txt' -f null -
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -i Slide_FullHD_60fps_av1.mkv -lavfi '[0]scale=-1:1080,fps=60[a];[a][1]psnr=stats_file=psnr_FullHD_60fps_av1.txt' -f null -
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -i Slide_FullHD_30fps_av1.mkv -lavfi '[0]scale=-1:1080,fps=30[a];[a][1]psnr=stats_file=psnr_FullHD_30fps_av1.txt' -f null -
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -i Slide_HD_90fps_h264.mkv -lavfi ’[0]scale=-1:720[a];[a][1]psnr=stats_file=psnr_HD_90fps_h264.txt’ -f null -
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -i Slide_HD_60fps_h264.mkv -lavfi ’[0]scale=-1:720,fps=60[a];[a][1]psnr=stats_file=psnr_HD_60fps_h264.txt’ -f null -
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -i Slide_HD_30fps_h264.mkv -lavfi ’[0]scale=-1:720,fps=30[a];[a][1]psnr=stats_file=psnr_HD_30fps_h264.txt’ -f null -
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -i Slide_HD_90fps_h265.mkv -lavfi ’[0]scale=-1:720[a];[a][1]psnr=stats_file=psnr_HD_90fps_h265.txt’ -f null -
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -i Slide_HD_60fps_h265.mkv -lavfi ’[0]scale=-1:720,fps=60[a];[a][1]psnr=stats_file=psnr_HD_60fps_h265.txt’ -f null -
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -i Slide_HD_30fps_h265.mkv -lavfi ’[0]scale=-1:720,fps=30[a];[a][1]psnr=stats_file=psnr_HD_30fps_h265.txt’ -f null -
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -i Slide_HD_90fps_vp9.mkv -lavfi ’[0]scale=-1:720[a];[a][1]psnr=stats_file=psnr_HD_90fps_vp9.txt’ -f null -
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -i Slide_HD_60fps_vp9.mkv -lavfi ’[0]scale=-1:720,fps=60[a];[a][1]psnr=stats_file=psnr_HD_60fps_vp9.txt’ -f null -
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -i Slide_HD_30fps_vp9.mkv -lavfi ’[0]scale=-1:720,fps=30[a];[a][1]psnr=stats_file=psnr_HD_30fps_vp9.txt’ -f null -
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -i Slide_HD_90fps_av1.mkv -lavfi ’[0]scale=-1:720[a];[a][1]psnr=stats_file=psnr_HD_90fps_av1.txt’ -f null -
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -i Slide_HD_60fps_av1.mkv -lavfi ’[0]scale=-1:720,fps=60[a];[a][1]psnr=stats_file=psnr_HD_60fps_av1.txt’ -f null -
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -i Slide_HD_30fps_av1.mkv -lavfi ’[0]scale=-1:720,fps=30[a];[a][1]psnr=stats_file=psnr_HD_30fps_av1.txt’ -f null -
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -i Slide_4K_90fps_h264.mkv -lavfi ssim=stats_file=ssim_4K_90fps_h264.txt -f null -
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -i Slide_4K_60fps_h264.mkv -lavfi ’[0]fps=60[a];[a][1]ssim=stats_file=ssim_4K_60fps_h264.txt’ -f null -
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -i Slide_4K_30fps_h264.mkv -lavfi ’[0]fps=30[a];[a][1]ssim=stats_file=ssim_4K_30fps_h264.txt’ -f null -
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -i Slide_4K_90fps_h265.mkv -lavfi ssim=stats_file=ssim_4K_90fps_h265.txt -f null -
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -i Slide_4K_60fps_h265.mkv -lavfi ’[0]fps=60[a];[a][1]ssim=stats_file=ssim_4K_60fps_h265.txt’ -f null -
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -i Slide_4K_30fps_h265.mkv -lavfi ’[0]fps=30[a];[a][1]ssim=stats_file=ssim_4K_30fps_h265.txt’ -f null -
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -i Slide_4K_90fps_vp9.mkv -lavfi ssim=stats_file=ssim_4K_90fps_vp9.txt -f null -
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -i Slide_4K_60fps_vp9.mkv -lavfi ’[0]fps=60[a];[a][1]ssim=stats_file=ssim_4K_60fps_vp9.txt’ -f null -
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -i Slide_4K_30fps_vp9.mkv -lavfi ’[0]fps=30[a];[a][1]ssim=stats_file=ssim_4K_30fps_vp9.txt’ -f null -
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -i Slide_4K_90fps_av1.mkv -lavfi ssim=stats_file=ssim_4K_90fps_av1.txt -f null -
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -i Slide_4K_60fps_av1.mkv -lavfi ’[0]fps=60[a];[a][1]ssim=stats_file=ssim_4K_60fps_av1.txt’ -f null -
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -i Slide_4K_30fps_av1.mkv -lavfi ’[0]fps=30[a];[a][1]ssim=stats_file=ssim_4K_30fps_av1.txt’ -f null -
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -i Slide_FullHD_90fps_h264.mkv -lavfi ’[0]scale=-1:1080[a];[a][1]ssim=stats_file=ssim_FullHD_90fps_h264.txt’ -f null -
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -i Slide_FullHD_60fps_h264.mkv -lavfi ’[0]scale=-1:1080,fps=60[a];[a][1]ssim=stats_file=ssim_FullHD_60fps_h264.txt’ -f null -
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -i Slide_FullHD_30fps_h264.
    mkv -lavfi ’[0]scale=-1:1080,fps=30[a];[a][1]ssim=stats_file=
    ssim_FullHD_30fps_h264.txt’ -f null -
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -i Slide_FullHD_90fps_h265.
    mkv -lavfi ’[0]scale=-1:1080[a];[a][1]ssim=stats_file=
    ssim_FullHD_90fps_h265.txt’ -f null -
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -i Slide_FullHD_60fps_h265.
    mkv -lavfi ’[0]scale=-1:1080,fps=60[a];[a][1]ssim=stats_file=
    ssim_FullHD_60fps_h265.txt’ -f null -
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -i Slide_FullHD_30fps_h265.
    mkv -lavfi ’[0]scale=-1:1080,fps=30[a];[a][1]ssim=stats_file=
    ssim_FullHD_30fps_h265.txt’ -f null -
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -i Slide_FullHD_90fps_vp9.
    mkv -lavfi ’[0]scale=-1:1080[a];[a][1]ssim=stats_file=
    ssim_FullHD_90fps_vp9.txt’ -f null -
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -i Slide_FullHD_60fps_vp9.
    mkv -lavfi ’[0]scale=-1:1080,fps=60[a];[a][1]ssim=stats_file=
    ssim_FullHD_60fps_vp9.txt’ -f null -
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -i Slide_FullHD_30fps_vp9.
    mkv -lavfi ’[0]scale=-1:1080,fps=30[a];[a][1]ssim=stats_file=
    ssim_FullHD_30fps_vp9.txt’ -f null -
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -i Slide_FullHD_90fps_av1.
    mkv -lavfi ’[0]scale=-1:1080[a];[a][1]ssim=stats_file=
    ssim_FullHD_90fps_av1.txt’ -f null -
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -i Slide_FullHD_60fps_av1.
    mkv -lavfi ’[0]scale=-1:1080,fps=60[a];[a][1]ssim=stats_file=
    ssim_FullHD_60fps_av1.txt’ -f null -
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -i Slide_FullHD_30fps_av1.
    mkv -lavfi ’[0]scale=-1:1080,fps=30[a];[a][1]ssim=stats_file=
    ssim_FullHD_30fps_av1.txt’ -f null -
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -i Slide_HD_90fps_h264.mkv -
    lavfi ’[0]scale=-1:720[a];[a][1]ssim=stats_file=ssim_HD_90fps_h264
    .txt’ -f null -
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -i Slide_HD_60fps_h264.mkv -
    lavfi ’[0]scale=-1:720,fps=60[a];[a][1]ssim=stats_file=
    ssim_HD_60fps_h264.txt’ -f null -
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -i Slide_HD_30fps_h264.mkv -
    lavfi ’[0]scale=-1:720,fps=30[a];[a][1]ssim=stats_file=
    ssim_HD_30fps_h264.txt’ -f null -
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -i Slide_HD_90fps_h265.mkv -
    lavfi ’[0]scale=-1:720[a];[a][1]ssim=stats_file=ssim_HD_90fps_h265
    .txt’ -f null -
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -i Slide_HD_60fps_h265.mkv -
    lavfi ’[0]scale=-1:720,fps=60[a];[a][1]ssim=stats_file=
    ssim_HD_60fps_h265.txt’ -f null -
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -i Slide_HD_30fps_h265.mkv -
    lavfi ’[0]scale=-1:720,fps=30[a];[a][1]ssim=stats_file=
    ssim_HD_30fps_h265.txt’ -f null -
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -i Slide_HD_90fps_vp9.mkv -
    lavfi ’[0]scale=-1:720[a];[a][1]ssim=stats_file=ssim_HD_90fps_vp9.
    txt’ -f null -
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -i Slide_HD_60fps_vp9.mkv -
    lavfi ’[0]scale=-1:720,fps=60[a];[a][1]ssim=stats_file=
    ssim_HD_60fps_vp9.txt’ -f null -
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -i Slide_HD_30fps_vp9.mkv -
    lavfi ’[0]scale=-1:720,fps=30[a];[a][1]ssim=stats_file=
    ssim_HD_30fps_vp9.txt’ -f null -
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -i Slide_HD_90fps_av1.mkv -
    lavfi ’[0]scale=-1:720[a];[a][1]ssim=stats_file=ssim_HD_90fps_av1.
    txt’ -f null -
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -i Slide_HD_60fps_av1.mkv -
    lavfi ’[0]scale=-1:720,fps=60[a];[a][1]ssim=stats_file=
    ssim_HD_60fps_av1.txt’ -f null -
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -i Slide_HD_30fps_av1.mkv -
    lavfi ’[0]scale=-1:720,fps=30[a];[a][1]ssim=stats_file=
    ssim_HD_30fps_av1.txt’ -f null -
```

APPENDIX A IV. QUALITY METRICS RESULTS

This appendix shows the quality metrics gathered in tests 1 to 3.

Table I PSNR values for test 1

Resolution	Coding format	Frame rate (fps)	PSNR min	PSNR max	PSNR average	PSNR median
4K	H.264	90	42.28	92.66	53.98	53.15
		60	8.19	92.83	46.08	51.08
		30	9.04	91.33	44.82	50.88
	H.265	90	37.39	65.03	47.28	47.155
		60	8.20	73.82	41.68	47.00
		30	9.04	74.68	41.30	48.17
	VP9	90	43.82	86.14	52.34	51.48
		60	8.20	86.16	44.62	50.32
		30	9.04	85.92	43.27	50.07
	AV1	90	44.73	84.37	52.52	51.64
		60	8.20	84.16	44.86	50.88
		30	9.04	84.7	43.32	50.53
Full HD	H.264	90	37.34	82.26	48.31	47.98
		60	8.19	82.57	42.41	45.86
		30	9.04	82.97	41.56	46.20
	H.265	90	34.30	58.88	44.82	44.80
		60	8.20	64.76	39.85	44.39
		30	9.04	64.93	39.39	45.43
	VP9	90	40.95	87.77	49.66	48.83
		60	8.19	87.77	42.76	47.68
		30	9.04	87.77	41.46	47.31
	AV1	90	41.88	84.49	49.92	49.17
		60	8.19	83.15	42.98	48.22
		30	9.04	86.36	41.57	48.00
HD	H.264	90	35.69	80.34	45.94	45.98
		60	8.20	80.47	40.65	44.13
		30	9.05	80.57	39.85	44.52
	H.265	90	32.35	59.03	43.14	43.22
		60	8.20	58.92	38.55	42.72
		30	9.05	61.67	38.09	43.77
	VP9	90	39.32	93.10	47.93	47.45
		60	8.20	91.90	41.58	46.23
		30	9.05	93.10	40.31	45.84
	AV1	90	40.3	89.33	48.32	47.99
		60	8.20	90.04	41.88	46.43
		30	9.05	92.91	40.51	46.19

Table II SSIM values for test 1

Resolution	Coding format	Frame rate (fps)	SSIM min	SSIM max	SSIM average	SSIM median
4K	H.264	90	0.96	0.99	0.99	0.99
		60	0.43	0.99	0.97	0.99
		30	0.56	0.99	0.97	0.99
	H.265	90	0.88	0.99	0.99	0.99
		60	0.43	0.99	0.97	0.99
		30	0.56	0.99	0.96	0.99
	VP9	90	0.96	0.99	0.99	0.99
		60	0.43	0.99	0.97	0.99
		30	0.56	0.99	0.96	0.99
	AV1	90	0.95	0.99	0.99	0.99
		60	0.43	0.99	0.97	0.99
		30	0.56	0.99	0.96	0.99
Full HD	H.264	90	0.92	0.99	0.99	0.99
		60	0.43	0.99	0.96	0.99
		30	0.56	0.99	0.95	0.99
	H.265	90	0.86	0.99	0.98	0.99
		60	0.43	0.99	0.96	0.98
		30	0.55	0.99	0.95	0.99
	VP9	90	0.91	0.99	0.99	0.99
		60	0.43	0.99	0.96	0.99
		30	0.56	0.99	0.95	0.99
	AV1	90	0.89	0.99	0.99	0.99
		60	0.43	0.99	0.96	0.99
		30	0.55	0.99	0.95	0.99
HD	H.264	90	0.43	0.99	0.96	0.99
		60	0.43	0.99	0.93	0.99
		30	0.43	0.99	0.92	0.99
	H.265	90	0.43	0.99	0.95	0.98
		60	0.42	0.99	0.93	0.98
		30	0.43	0.99	0.92	0.98
	VP9	90	0.43	0.99	0.96	0.99
		60	0.42	0.99	0.93	0.99
		30	0.43	0.99	0.92	0.99
	AV1	90	0.43	0.99	0.97	0.99
		60	0.42	0.99	0.93	0.99
		30	0.43	0.99	0.92	0.99

Table III PSNR values for test 2

Resolution	Coding format	Frame rate (fps)	PSNR min	PSNR max	PSNR average	PSNR median
4K	H.264	90	39.25	77.85	49.69	49.46
		60	8.19	77.87	43.74	49.22
		30	9.04	82.17	43.7	50.39
	H.265	90	37.64	75.46	49.27	49.38
		60	8.19	68.44	42.82	48.73
		30	9.04	73.02	42.18	48.71
	VP9	90	43	64.04	50.57	50.2
		60	8.2	68.52	43.29	49.45
		30	9.04	64.87	41.96	49.13
	AV1	90	44.4	81.82	52.17	51.37
		60	8.2	81.73	44.57	50.61
		30	9.04	82.58	43.02	50.26
FullHD	H.264	90	36.38	74.13	46.78	46.81
		60	8.19	77.34	41.44	46.44
		30	9.04	80.57	41.08	47.05
	H.265	90	34.26	66.57	46.19	46.31
		60	8.19	68.29	40.75	45.89
		30	9.05	69.11	39.99	46.09
	VP9	90	40.46	62.75	48.41	48.25
		60	8.2	63.44	41.74	47.11
		30	9.05	62.93	40.42	46.52
	AV1	90	41.65	87.2	49.76	48.97
		60	8.19	80.27	42.73	47.97
		30	9.04	86.76	41.42	47.72
HD	H.264	90	34.71	75.32	45.07	45.125
		60	8.2	75.01	40.04	44.68
		30	9.05	79.03	39.45	45.13
	H.265	90	32.09	93.1	44.35	44.55
		60	8.2	67.33	39.22	43.78
		30	9.05	73.96	38.56	44.24
	VP9	90	38.79	62.22	46.86	46.9
		60	8.2	62.39	40.61	45.79
		30	9.05	63.07	39.32	45.2
	AV1	90	40.27	92.91	48.24	47.82
		60	8.2	88.2	41.71	46.35
		30	9.05	89.37	40.28	46.16

Table IV SSIM values for test 2

Resolution	Coding format	Frame rate (fps)	SSIM min	SSIM max	SSIM average	SSIM median
4K	H.264	90	0.68	0.99	0.98	0.99
		60	0.43	0.99	0.96	0.99
		30	0.55	0.99	0.96	0.99
	H.265	90	0.94	0.99	0.99	0.99
		60	0.43	0.99	0.97	0.99
		30	0.56	0.99	0.96	0.99
	VP9	90	0.88	0.99	0.99	0.99
		60	0.43	0.99	0.97	0.99
		30	0.55	0.99	0.96	0.99
	AV1	90	0.95	0.99	0.99	0.99
		60	0.43	0.99	0.97	0.99
		30	0.56	0.99	0.96	0.99
FullHD	H.264	90	0.67	0.99	0.97	0.99
		60	0.43	0.99	0.95	0.98
		30	0.55	0.99	0.95	0.99
	H.265	90	0.88	0.99	0.98	0.99
		60	0.43	0.99	0.96	0.98
		30	0.55	0.99	0.95	0.98
	VP9	90	0.85	0.99	0.98	0.99
		60	0.43	0.99	0.96	0.99
		30	0.55	0.99	0.95	0.99
	AV1	90	0.89	0.99	0.99	0.99
		60	0.43	0.99	0.96	0.99
		30	0.55	0.99	0.95	0.99
HD	H.264	90	0.43	0.99	0.95	0.98
		60	0.42	0.99	0.93	0.98
		30	0.43	0.99	0.92	0.98
	H.265	90	0.43	0.99	0.95	0.98
		60	0.42	0.99	0.93	0.98
		30	0.43	0.99	0.92	0.98
	VP9	90	0.43	0.99	0.96	0.99
		60	0.42	0.99	0.93	0.99
		30	0.43	0.99	0.92	0.99
	AV1	90	0.43	0.99	0.97	0.99
		60	0.43	0.99	0.93	0.99
		30	0.43	0.99	0.92	0.99

Table V PSNR values for test 3

Resolution	Coding format	Frame rate (fps)	PSNR min	PSNR max	PSNR average	PSNR median
4K	H.264	90	42.18	92.82	53.81	53.04
		60	8.19	92.74	46.05	50.98
		30	9.04	90.56	44.84	50.92
	H.265	90	37.39	65.03	47.28	47.155
		60	8.2	73.82	41.68	47
		30	9.04	74.68	41.3	48.17
	VP9	90	31.43	66.16	40.76	41.055
		60	8.22	58.8	36.32	39.46
		30	9.06	72.06	35.33	39.65
FullHD	H.264	90	37.31	82.47	48.32	48
		60	8.19	82.49	42.44	45.86
		30	9.04	83.18	41.56	46.18
	H.265	90	34.3	58.88	44.82	44.8
		60	8.2	64.76	39.85	44.39
		30	9.04	64.93	39.39	45.43
	VP9	90	29.16	84.2	40.19	39.485
		60	8.24	83.22	36.04	38.88
		30	9.06	85.7	34.74	37.71
HD	H.264	90	35.75	80.46	45.95	45.995
		60	8.2	80.32	40.65	44.15
		30	9.05	80.36	39.84	44.51
	H.265	90	32.35	59.03	43.14	43.215
		60	8.2	58.92	38.55	42.72
		30	9.05	61.67	38.09	43.77
	VP9	90	27.63	93.1	39.21	38.41
		60	8.24	93.1	35.44	37.31
		30	9.08	85.26	34.32	37.5

Table VI SSIM values for test 3

Resolution	Coding format	Frame rate (fps)	SSIM min	SSIM max	SSIM average	SSIM median
4K	H.264	90	0.96	0.99	0.99	0.99
		60	0.43	0.99	0.97	0.99
		30	0.56	0.99	0.97	0.99
	H.265	90	0.88	0.99	0.99	0.99
		60	0.43	0.99	0.97	0.99
		30	0.56	0.99	0.96	0.99
	VP9	90	0.43	0.99	0.95	0.98
		60	0.43	0.99	0.94	0.98
		30	0.43	0.99	0.93	0.97
FullHD	H.264	90	0.91	0.99	0.99	0.99
		60	0.43	0.99	0.96	0.99
		30	0.56	0.99	0.95	0.99
	H.265	90	0.86	0.99	0.98	0.99
		60	0.43	0.99	0.96	0.98
		30	0.55	0.99	0.95	0.99
	VP9	90	0.67	0.99	0.95	0.97
		60	0.42	0.99	0.94	0.96
		30	0.52	0.99	0.93	0.96
HD	H.264	90	0.43	0.99	0.96	0.99
		60	0.42	0.99	0.93	0.99
		30	0.43	0.99	0.92	0.99
	H.265	90	0.43	0.99	0.95	0.98
		60	0.42	0.99	0.93	0.98
		30	0.43	0.99	0.92	0.98
	VP9	90	0.43	0.99	0.93	0.96
		60	0.42	0.99	0.91	0.95
		30	0.43	0.99	0.90	0.95

APPENDIX A V. TEST 4 BENCHMARK SCRIPT

The following commands were used separately:

```
ffmpeg -s 1280x720 -r 30 -i /dev/video0 -vcodec libx264 -preset ultrafast -tune zerolatency -vf "drawtext=expansion=normal:fontfile=/usr/share/fonts/truetype/ttf-dejavu/DejaVuSans.ttf:reload=1:textfile=time.txt: x=50: y=50: fontcolor=red" -f matroska - | ffplay -preset ultrafast -tune zerolatency -vf "drawtext=expansion=normal:fontfile=/usr/share/fonts/truetype/ttf-dejavu/DejaVuSans.ttf:reload=1:textfile=time.txt: x=50: y=70: fontcolor=red" -
ffmpeg -s 1280x720 -r 30 -i /dev/video0 -vcodec libx265 -preset ultrafast -tune zerolatency -vf "drawtext=expansion=normal:fontfile=/usr/share/fonts/truetype/ttf-dejavu/DejaVuSans.ttf:reload=1:textfile=time.txt: x=50: y=50: fontcolor=red" -f matroska - | ffplay -preset ultrafast -tune zerolatency -vf "drawtext=expansion=normal:fontfile=/usr/share/fonts/truetype/ttf-dejavu/DejaVuSans.ttf:reload=1:textfile=time.txt: x=50: y=70: fontcolor=red" -
ffmpeg -s 1280x720 -r 30 -i /dev/video0 -strict experimental -c:v libvpx-vp9 -deadline realtime -cpu-used 8 -vf "drawtext=expansion=normal:fontfile=/usr/share/fonts/truetype/ttf-dejavu/DejaVuSans.ttf:reload=1:textfile=time.txt: x=50: y=50: fontcolor=red" -f matroska - | ffplay -deadline realtime -vf "drawtext=expansion=normal:fontfile=/usr/share/fonts/truetype/ttf-dejavu/DejaVuSans.ttf:reload=1:textfile=time.txt: x=50: y=70: fontcolor=red" -
```

APPENDIX A VI. TEST 5 BENCHMARK SCRIPT

In this test, a video stream for each coding format was first published to a RTSP server using the corresponding command from the *rtsp_publish.sh* script, then played with the corresponding command from the *rtsp_play.sh* script.

rtsp_publish.sh

```
ffmpeg -s 1280x720 -r 30 -i /dev/video0 -vcodec libx264 -preset
    ultrafast -tune zerolatency -rtsp_transport tcp -vf "drawtext=
    expansion=normal:fontfile=/usr/share/fonts/truetype/ttf-dejavu/
    DejaVuSans.ttf:reload=1:textfile=time.txt: x=50: y=50: fontcolor=
    red" -f rtsp rtsp://<server IP>:8554/mystream

ffmpeg -s 1280x720 -r 30 -i /dev/video0 -vcodec libx265 -preset
    ultrafast -tune zerolatency -rtsp_transport tcp -vf "drawtext=
    expansion=normal:fontfile=/usr/share/fonts/truetype/ttf-dejavu/
    DejaVuSans.ttf:reload=1:textfile=time.txt: x=50: y=50: fontcolor=
    red" -f rtsp rtsp://<server IP>:8554/mystream

ffmpeg -s 1280x720 -r 30 -i /dev/video0 -strict experimental -c:v
    libvpx-vp9 -deadline realtime -cpu-used 8 -rtsp_transport tcp -vf
    "drawtext=expansion=normal:fontfile=/usr/share/fonts/truetype/ttf-
    dejavu/DejaVuSans.ttf:reload=1:textfile=time.txt: x=50: y=50:
    fontcolor=red" -f rtsp rtsp://<server IP>:8554/mystream
```

rtsp_play.sh

```
ffplay -tune zerolatency -rtsp_transport tcp -fflags -nobuffer -
    analyzeduration 1 -vf "drawtext=expansion=normal:fontfile=/usr/
    share/fonts/truetype/ttf-dejavu/DejaVuSans.ttf:reload=1:textfile=
    time.txt: x=50: y=70: fontcolor=red" rtsp://<server IP>:8554/
    mystream

ffplay -deadline realtime -rtsp_transport tcp -fflags -nobuffer -
    analyzeduration 1 -vf "drawtext=expansion=normal:fontfile=/usr/
    share/fonts/truetype/ttf-dejavu/DejaVuSans.ttf:reload=1:textfile=
    time.txt: x=50: y=70: fontcolor=red" rtsp://<server IP>:8554/
    mystream
```

APPENDIX A VII. STEREOSCOPIC VIDEO TEST SCRIPT

This script first combines two video feeds into a single stereoscopic video, then performs a benchmarked transcoding.

stereo.sh

```
ffmpeg -i Slide_4K_90FPS_test_noaudio.mkv -i Slide_4K_90FPS_test_noaudio.mkv -c copy -map 0 -map 1 Slide_4K_90FPS_test_stereo.mkv


ffmpeg -i Slide_4K_90FPS_test_stereo.mkv -report -benchmark -c:v libx264 -map 0:0 -map 0:1 -filter:v "scale=-1:1080,fps=30" Slide_FullHD_30fps_h264_stereo.mkv
ffmpeg -i Slide_4K_90FPS_test_stereo.mkv -report -benchmark -c:v libx265 -map 0:0 -map 0:1 -filter:v "scale=-1:1080,fps=30" Slide_FullHD_30fps_h265_stereo.mkv
ffmpeg -i Slide_4K_90FPS_test_stereo.mkv -report -benchmark -strict experimental -c:v libvpx-vp9 -map 0:0 -map 0:1 -filter:v "scale=-1:1080,fps=30" Slide_FullHD_30fps_vp9_stereo.mkv
ffmpeg -i Slide_4K_90FPS_test_stereo.mkv -report -benchmark -strict -2 -c:v libaom-av1 -map 0:0 -map 0:1 -filter:v "scale=-1:1080,fps=30" Slide_FullHD_30fps_av1_stereo.mkv
```