Bachelor Project

Czech Technical University in Prague

F3

Faculty of Electrical Engineering
Department of Radioelectronics

Image Quality Assessment in Immersive Image Display Systems

Hodnocení kvality obrazu v systémech pro imerzivní reprodukci obrazu

Adam Pinsker

Supervisor: Ing. Karel Fliegel, Ph.D.
May 2023

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Pinsker** Jméno: **Adam** Osobní číslo: **499105**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra radioelektroniky**

Studijní program: **Elektronika a komunikace**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Hodnocení kvality obrazu v systémech pro imerzivní reprodukci obrazu

Název bakalářské práce anglicky:

Image Quality Assessment in Immersive Image Display Systems

Pokyny pro vypracování:

Give an overview of recent research and software tools in the field of image quality assessment and Quality of Experience (QoE) for immersive omnidirectional image displays. Focus also on the possibility to capture eye-tracking data in head-mounted displays for virtual reality environments. Develop software tools to perform related subjective experiments and verify the functionality in the pilot study.

Seznam doporučené literatury:

[1] Huang, M., et al., Modeling the Perceptual Quality of Immersive Images Rendered on Head Mounted Displays: Resolution and Compression, IEEE Transactions on Image Processing, 2018.
[2] Vlahovic, S., Suznjevic, M., Skorin-Kapov, L., A survey of challenges and methods for Quality of Experience assessment of interactive VR applications, Journal on Multimodal User Interfaces, 2022.

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Karel Fliegel, Ph.D. katedra radioelektroniky FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **14.02.2023** Termín odevzdání bakalářské práce: **26.05.2023**

Platnost zadání bakalářské práce: **22.09.2024**

Ing. Karel Fliegel, Ph.D.	doc. Ing. Stanislav Vítek, Ph.D.	prof. Mgr. Petr Páta, Ph.D.
podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

. Datum převzetí zadání

Podpis studenta

Acknowledgements

I would like to thank my supervisor, Ing. Karel Fliegel, Ph.D., for his guidance in preparing this work.

Declaration

Prohlašuji, že jsem předloženou práci vypracoval samostatně, a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with methodical instructions for observing the ethical principles in the preparation of university theses.

In Prague, 26. May 2023

Abstract

The aim of this thesis was to create an automated tool, that serves as a platform for data analysis from subjective image quality testing in virtual reality. A comparison with various objective metrics is implemented with the utilization of eye-tracking data from subjective testing for possible improvement of the effectiveness of some objective metrics. The goal was also the automation of the tool, so the user does not have to do any complex preparation. The tool as a whole is a ready-to-use application to analyse and project data.

An overview is also given of the current state of subjective testing of image quality and quality of experience in virtual reality together with objective metrics comparison for omnidirectional images.

Keywords:
Virtual reality, Eye-Tracking,
360° image compression
Omnidirectional projection,
Subjective quality of experience

Supervisor: Ing. Karel Fliegel, Ph.D.
CTU FEE,
Technická 6,
16000 Prague 5

Abstrakt

Cílem této práce bylo vytvoření automatizovaného nástroje, který slouží jako platforma pro analýzu dat ze subjektivního testování kvality obrazu ve virtuální realitě. Porovnání s objektivními metrikami je implementováno s využitím dat očních pohybů ze subjektivního testování pro možné vylepšení efektivity objektvních metrik. Cílem také byla automatizace nástroje, aby uživatel nemusel dělat žádné složité přípravy. Nástroj jako celek je aplikace připravena k použití pro analýzu a projekci dat.

Je podán také přehled o současném stavu subjektivního testování kvality obrazu a kvality prožitku ve virtuální realitě spolu s porovnáním současných metrik pro objektivní testování pro všesměrovové obrázky.

Klíčová slova:
Virtuální realita, Sledování očních pohybů,
Komprese 360° obrázků,
Všesměrová projekce,
Subjektivní kvalita prožitku

Překlad názvu: Hodnocení kvality obrazu v systémech pro imerzivní reprodukci obrazu

Contents

Figures

Tables

Chapter 1

Introduction

Image quality assessment is a necessity for optimising the quality of digital visual content in virtual reality (VR). Since making subjective tests for determining image quality is not feasible on a large scale, various objective metrics have been proposed such as PSNR, MSE or SSIM. All of these metrics have their deficiencies and some can be improved. Such an improvement can be the utilization of eye-tracking data to localize key areas of images on which people tend to focus.

To this date, only a few testbeds have been created to interactively measure and analyse the quality of experience (QoE) on a large scale in VR and none of them utilize eye-tracking data for objective metrics analysis. Such systems can then measure objective metrics' efficiency and compare them to each other. It is also beneficial when improving given metrics to quickly analyse their improvements.

In this work, in chapter 2, a brief insight is given into omnidirectional projections with their benefits and disadvantages. An overview of current objective metrics is given in chapter 3 with their effectiveness comparison. Then, in chapter 4.1 factors affecting users' QoE are talked about. Chapter 4 gives an overview of QoE structure and possible approaches and recommendations are given for subjective QoE testing along with existing testbeds and applications for measuring subjective QoE and image quality.

Finally, in chapter 5 a pilot testbed for analysing image quality data is introduced. It serves as a platform for analysing subjective image quality data and comparing them to publicly available objective metrics. Those metrics can be then altered using eye-tracking data captured during subjective testing and possible improvements demonstrated.

Chapter 2

Omnidirectional video projections

Immersion achieved by virtual reality is created by the sense of a human actually being in the world rather than looking at one as an observer. This is primarily achieved due to the projection of the world, that is being presented to us. Just like a regular monitor, virtual reality has a display by which the user can look at the world. The main difference is, however, that the image in the display is not static related to the user's orientation. By rotating the head, it is possible to see the whole world around and not be limited to just a small section of it. This type of projection is called omnidirectional. An illustration is shown in figure 2.1.

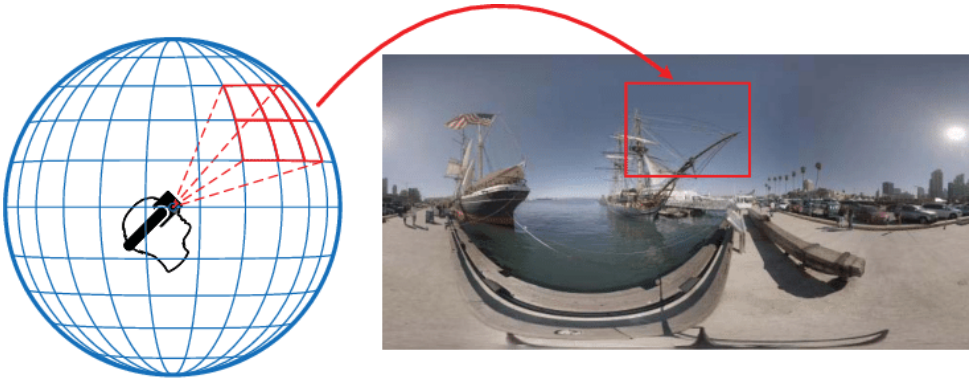

Figure 2.1: Showcase of omnidirectional projection (image taken from [27]).

Converting content or outright capturing one in this projection is a much more difficult task compared to traditional flat images or videos. Since storing an image as a sphere surface is not very practical, various methods have been developed to overcome this issue. All these methods have one thing in common – storing the omnidirectional space as a flat image. Some kind of warping or uneven folding is then required to achieve such a thing.

The main goal is to create the best quality content with the lowest possible amount of data. Other factors, for example, are the complexity of creating the content, though this is largely a problem only for live-streamed content, such as games etc.

There are two main categories for omnidirectional video projections –

viewport-independent projection (VIP) and viewport-dependent projection (VDP). Practical examples of such methods are in the next two sections.

2.1 Viewport independent projection (VIP)

The difference between VIP and VDP is the way they are being streamed to the user. Using VIP, the whole world is being available at all times. This reduces possible latency as no feedback is required of the user's head rotation. The downside is, however, a higher required bitrate, since the whole 360° world is being transmitted rather than a limited section of it. Such a method of only a limited section streaming is utilized by VDP.

2.1.1 Map-based projection

Generally, one of the simplest, map-based projections consists of 1 image that is warped around the user, creating a 360° image. Equirectangular Projection (ERP) is the most common one, used for example in 2D world maps. An example is shown in figure 2.2. This method suffers greatly from oversampling since the poles must be stretched to create a 2D image. Cylindrical Equal-Area Projection (EAP) partially solves this problem by multiplying every vertical layer by $\cos(\theta)$, where θ is the latitude. This solves the oversampling problem but also increases distortion [1].

Figure 2.2: Equirectangular projection (image adapted from [26]).

2.1.2 Patch-based projection

Patch-based projection is a more complex method. It solves the oversampling and distortion problem from ERP and EAP by splitting the space into several areas and stitching them together. This, however, creates unnatural boundaries between every two areas that may be visible, thus lowering the overall QoE. Generally, the more faces there are, the smoother edges there

will be and the oversampling rate will be lower. Choosing the right method is, therefore, crucial to ensure a smooth picture and a feasible bitrate.

One of the most popular patch-based projections is a Cube Map Projection (CMP) shown in figure 2.3 (right) [10], where the world around the user is interpreted as a cube. When unfolded, 6 individual square images are created. Many other possible methods use the same principle as CMP but with an increased number of surfaces. An example of these methods may be octahedron projection, icosahedron projection or dodecahedron projection shown in figure 2.3 (left) [10].

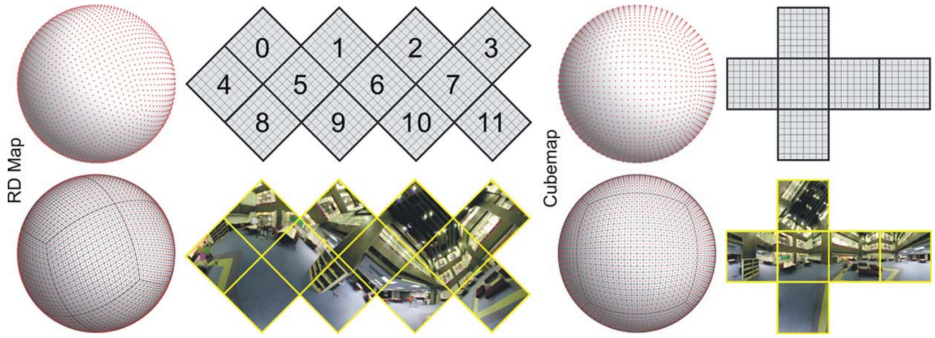

Figure 2.3: (left) Rhombic dodecahedron map and (right) Cube-map (image taken from [10]).

2.1.3 Tile-based projection

Another possibility is splitting the sphere into several horizontal tiles, where those near the poles have a lower sampling rate. Stitching the images together is not as big of a problem, since all the stitches are only in the horizontal direction [1].

2.2 Viewport dependent projection (VDP)

Using VDP, only a small section of the world is being streamed to the user, based on his current orientation. This greatly reduces the bitrate. However, it also requires a two-way connection to get users' head orientation. This is not only more complicated to create but may also significantly increase latency [1]. A possible VDP solution is, for example, a pyramid projection shown in figure 2.4 proposed by Facebook [11].

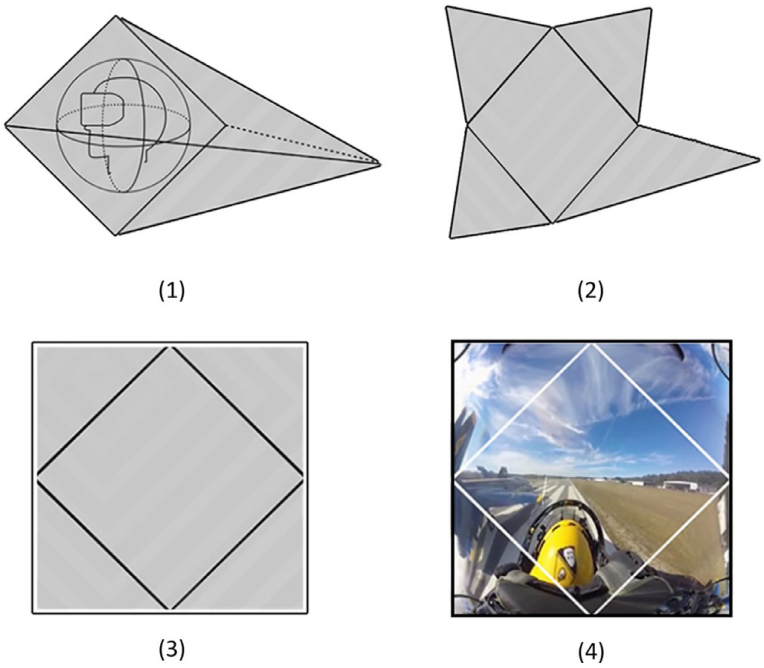

Figure 2.4: The process of the pyramid projection (image taken from [11]).

Chapter 3

Metrics for omnidirectional QoE

QoE measurements are crucial to get an essential feedback. There are two possible approaches of getting such feedback. The first option is subjective testing on a large enough pool of people. This can result, if conducted correctly, in a very accurate measurement – methodologies of conducting such tests correctly are briefly explored in section 4.2. The main downside of such testing is, however, the time required, which can increase dramatically, if accurate results are needed.

For this reason, objective metrics can be used. An objective metric can be any calculation based on input data – for example an image. These input data, however, must be empirically measured and most importantly, do not include human bias. This does not mean such a metric will be more accurate. If a given calculation is not optimal, it will usually result in a completely different result from subjective testing.

In this thesis, such objective metrics are complex computer-calculated algorithms for image quality assessment. Their main benefit, compared to subjective testing, is their speed.

3.1 Mean Opinion Score (MOS) and correlation coefficients

Let's first start with a subjective metric to be later able to compare objective metrics to subjective scores, thus, analysing their performance.

The most common subjective metric is Mean Opinion Score (MOS). It can be used for testing of any kind that includes humans. Its principle is very simple, as scores of a given object – an image for example – are collected from each person. Those scores are on a scale from 1 to 5, 1 being the worst, 5 being the best. All scores are then averaged creating a unified mean opinion score [14].

To be able to compare such a metric with its objective counterpart, correlation needs to be calculated. This can be done by using Pearson's linear correlation coefficient (PLCC) or Spearman's rank order coefficient (SROC). They both measure how given two functions deviate from each other in a parametric graph. Both these metrics fit the resulting points by a curve and

measure the average distance of the point from the curve. PLCC utilizes a straight line as a curve, thus, being linear, whereas SROC limits itself only by using a monotonic function as a fit. An illustration of PLCC is shown in figure 3.1. SROC is shown in figure 3.2.

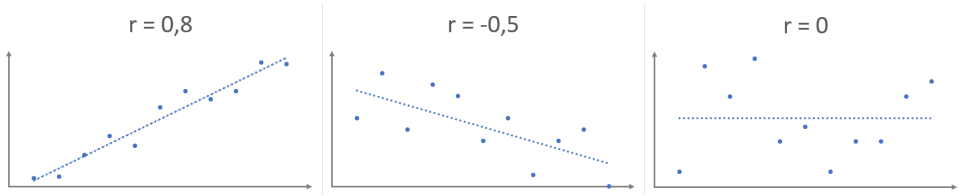

Figure 3.1: Examples of PLCC fitting and possible resulting values.

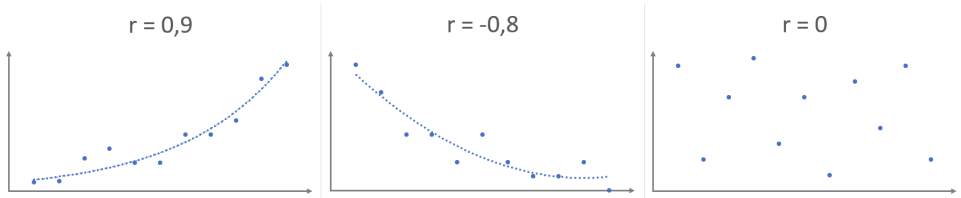

Figure 3.2: Examples of SROC fitting and possible resulting values.

3.2 Mean Square Error (MSE)

As a first objective metric, Mean square error (MSE), was chosen. since it is a conventional metric, that has been used for many centuries for various applications.

It measures the difference between, for example, two signals or images. Its principle is fairly simple. It first calculates the value difference between two pixels. This difference is then squared and values across the whole image are averaged [25]. The calculation is shown in equation 3.1.

$$MSE = \frac{\sum (y_i - \hat{y}_i)^2}{n} \tag{3.1}$$

3.3 Peak signal-to-noise ratio (PSNR)

One of the most common metrics after MSE is Signal-to-Noise ratio (SNR). It is derived from MSE as it is only a decibel version of it.

However, for conventional 2D pictures or videos, Peak Signal-to-Noise Ratio (PSNR) is usually the more common version. Its computation is, again, very simple as it is only SNR referenced to a peak value in an image. The computation of PSNR is shown in equation 3.2.

$$PSNR = 10 \times \log_{10} \left(\frac{MAX_I^2}{MSE} \right) \tag{3.2}$$

In practice, it indicates how much noise there is in a picture compared to its maximum possible quality. However, it was concluded, that it has a fairly low relation to what a human actually perceives as a video quality [12] and has its limitation even in a 2D picture. One of its main drawbacks is the lack of assessment of the actual footage content, which has a significant impact on human perception of QoE. Some improvements specifically for omnidirectional projection have been created, like Spherical PSNR (S-PSNR) but these do not perform much better. Calculations were made [7] of the PLCC and SRCC scores of PSNR and S-PSNR shown in table 3.1. All tests were performed in head-mounted displays using the SUN 360 Database[1] [28] and images created by [7].

Corr	PSNR	S-PSNR
PLCC	0.52	0.62
SRCC	0.58	0.67

Table 3.1: Performance of S-PSNR and PSNR in terms of PLCC and SRCC [7].

3.4 Gradient similarity index (GSI)

Gradient similarity index (GSI) was proposed [17] as a metric based on the principle that humans are more sensitive to edge regions. It focuses on capturing the high-frequency information and edge structures present in the images. It is particularly useful for evaluating image quality in scenarios where the distortion mainly affects the edges or gradient characteristics of the image, such as compression artefacts or image denoising. It was measured that GSI reaches a PLCC score of 0.90 and an SRCC score of 0.89 [4]. All tests were performed in an omnidirectional environment on the IQA database[2].

3.5 Visual saliency-based index VSI

Visual saliency-based index (VSI) was proposed [15] as a metric based on an assumption that an image's visual saliency map has a close relationship with its perceptual quality. It explores the visual saliency map in two stages. At the stage of local quality map computation, the VS map is taken as an image feature, while at the quality score pooling stage it is used as a weighting function to characterize the importance of a local image region [15]. It reaches a PLCC score of 0.91 and an SRCC score of 0.90 [4]. All tests were performed in an omnidirectional environment on the IQA database.

[1] https://vision.cs.princeton.edu/projects/2012/SUN360/data/
[2] http://database.mmsp-kn.de/iqa-experts-300.html

3.6 Structural similarity index (SSIM)

Structural similarity index (SSIM) [13] is a metric that takes into account human visual perception. It measures both local and global similarities of an image to its reference image.

SSIM calculations can be split into three components. The first is luminescence which is the brightness of a given pixel. The second is the contrast which is the brightness difference between two pixels. The third component is structural information. This component looks for various patterns along the image with regard to its reference.

All three components generate a map with values for each pixel between 0-1. The final value is then calculated as a mean along all those pixels and elements. It reaches a PLCC score of 0.51 and an SRCC score of 0.35 [4]. Though being fairly low, usually, it performs better. All tests were performed in an omnidirectional environment on the IQA database[3].

3.7 Multiscale SSIM (MS-SSIM)

Multiscale structural similarity index (MS-SSIM) is an advanced version of SSIM performed on various scales of the given image with its downsampled versions [29]. It can usually perform slightly better than regular SSIM with a PLCC score of 0.68 and an SRCC score of 0.67 [4]. All tests were performed in an omnidirectional environment on the IQA database.

3.8 Feature similarity index FSIM

Feature similarity index (FSIM) was proposed [16] as a feature-based metric. It is based on the assumption that humans perceive images mainly based on their two salient low-level features – phase congruency and gradient magnitude, which represent their complementary aspects of the image visual quality. It reaches a PLCC score of 0.92 and an SRCC score of 0.91 [4]. All tests were performed in an omnidirectional environment on the IQA database.

[3] http://database.mmsp-kn.de/iqa-experts-300.html

3.9 Conclusion

Table 3.2 [7, 4] summarizes all metrics performance. The best-performing metric is FSIM closely followed by VSI and GSI. On the other hand, traditional PSNR performs by far the worst.

Corr	**FSIM**	VSI	GSI	MS-SSIM	V-PSNR	SSIM	PSNR
PLCC	**0.9171**	0.9060	0.8992	0.68	0.6617	0.51	0.5088
SRCC	**0.9110**	0.9020	0.8901	0.67	0.6051	0.35	0.4984

Table 3.2: Performance of FSIM, VSI, GSI, MS-SSIM, SSIM, V-PSNR and PSNR in terms of PLCC and SRCC [7, 4].

Chapter 4

QoE measurements

In this broad chapter, section 4.1 discusses the structure of QoE and quantifies features and factors contributing to overall QoE. Then, section 4.2 gives recommendations of testing methodology for subjective testing in VR and finally section 4.3 gives a brief overview of current testbeds and applications used for such testing.

4.1 QoE features and factors

The overall QoE can be structuralized into several specific features [5], such as image quality, fragmentation, immersion, cybersickness and attractiveness. Most of these features cannot be objectively measured. However, they can be further subdivided into factors which are measurable qualities of the videos, such as compression rate, resolution, lag, etc. Figure 4.1 shows the composition of features and factors on the overall QoE.

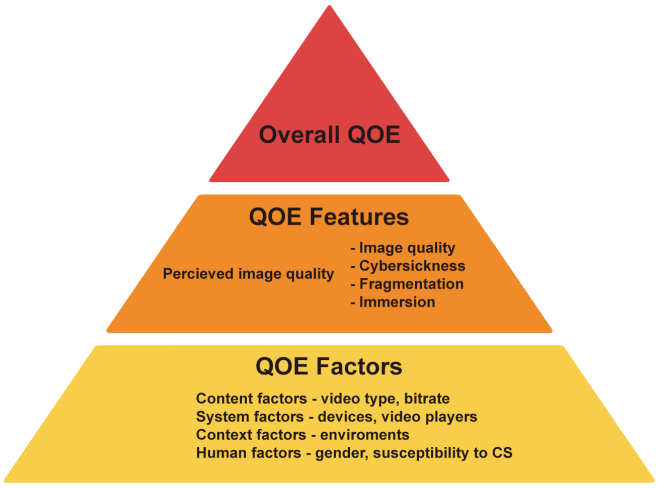

Figure 4.1: The compositions of QoE and their definitions.

4.1.1 QoE factors

In this section, all QoE factors are described as basic stones upon which QoE can be later created.

Compression rate

When working with media, minimizing its size is desired, since it is easier to work with it and share it. The most obvious solution is compression, which can dramatically reduce size while maintaining good or even the same quality. Figure 4.2 [3] shows different quantization parameters of H.264/AVC codec with various resolutions for three different content types with low, medium and fast motion complexity, in that order. Videos used in the study were created by [3].

Increasing the compression parameters up to a certain level does not have a large impact on the video quality. For UHD and QHD those parameters are up to 32 and for FHD up to 28 [3].

Complex and slow-paced videos are generally more sensitive to increasing compression [6], as subjects have more time to "zoom in" on the video and analyze all the imperfections.

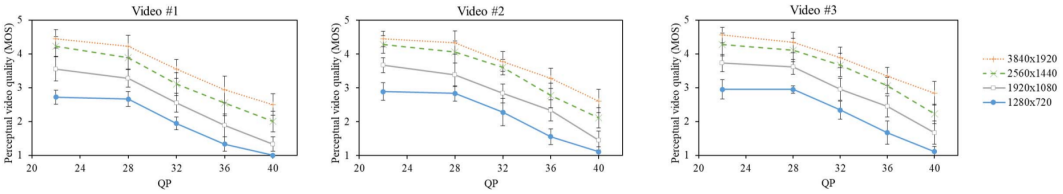

Figure 4.2: Perceptual video quality vs. encoding parameters (taken from [3]).

Resolution and bitrate

Resolution and bitrate are definitely two of the most important factors contributing to QoE and overall video complexity. Figure 4.3 [3] shows the acceptability rates vs. bitrates for three different content types with low, medium and fast motion complexity, in that order. Videos used in the study were created by [3].

It can be seen that decreasing the bitrate to about 1/4 of its maximum potential can do little to no harm to its perceived quality. However, based on fixed bitrate, quality can vary widely depending on content type. If the projected video has dynamic content, its acceptability quickly decreases. In terms of resolution, the most significant difference in quality is between HD and FHD, while FHD is still not providing acceptable results. QHD and UHD have fairly similar results so QHD could be used as an optimal solution since it has a lower bitrate and a high quality across a wide range of bitrates [3].

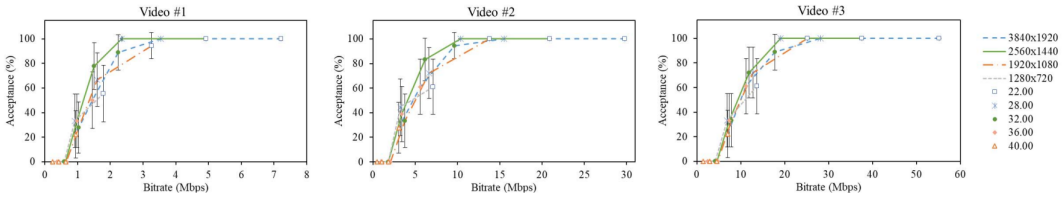

Figure 4.3: Acceptability rates vs. bitrates (taken from [3]).

Stalling events

A very important and often overlooked factor is stalling events. They can have a large impact on the overall quality and cause severe cybersickness. It was investigated [9], that the effects of frame freezing on the overall quality can be acceptable for a 100-300 ms lag. Observations were made [2] of how longer stalling events have an impact on the overall quality. The results are shown in figure 4.4 [2]. Videos used in the study were taken from Sintel[1] and ZDF documentary "Vulkane"[2]

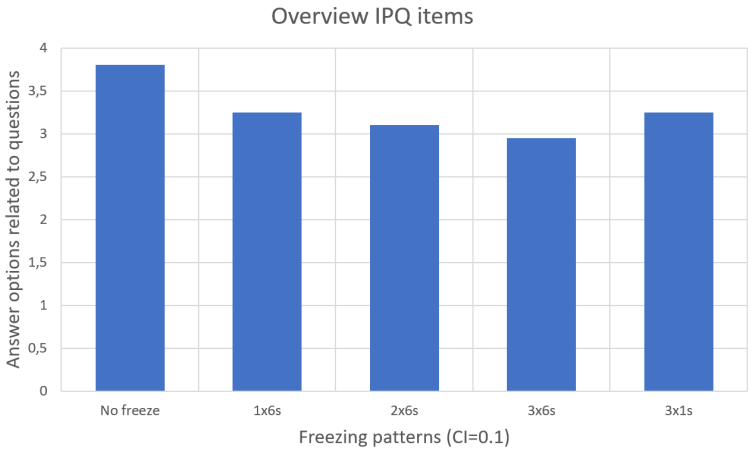

Figure 4.4: Scores of the four IPQ questions as influenced by stalling for the 'VR move' condition (5=strong sense of presence/immersion, 1=weak sense of presence/immersion) (graph adapted from [2]).

4.1.2 QoE features

All of the above-mentioned measurable factors contribute to several features, such as Image Quality (IQ), FraGmentation (FG), IMmersion (IM), Cyber-Sickness (CS) and ATtractiveness (AT) [5]. All these features can not be objectively measured. Therefore, when creating a new product, it is a good practice to focus on these features, since they are the "source" of QoE, not the factors themselves.

[1] https://durian.blender.org/

[2] https://www.zdf.de/dokumentation/terra-x/ 3d-360-grad-immersiver-film-vulkane-100.html

It was shown that cybersickness is influenced more by content type than bitrate [5] since content with a high optical flow can cause much more severe cybersickness than low-quality content. That makes sense since CS is created when there is a difference between the user's head movement and movement in the scene [3].

4.2 QoE modeling methods and testing conditions

When conducting a test, setting up the right conditions and designing content is just as important as the metrics used for analyzing results. In this section, recommendations about various modeling methods and testing conditions will be given. However, since subjective QoE measurements are a wide topic to discuss, all recommendations in this section, only apply to QoE measurements in virtual reality.

4.2.1 QoE modeling methods

The first thing to do when creating a test is to focus on what exactly we want to measure. If we are trying to measure the impact of quantization on the overall quality, changing other parameters during the test such as resolution or lag is counter-productive, as it will only bring bias to the MOS score according to ITU-T P.919 [8]. If testing quantization while changing the resolution is required, various resolutions should be separated.

The presented content should have a variety of spatial and temporal complexity and cover a wide range of stimuli so that the results could be used for a variety of applications [8]. The original video content should be obtained in the highest possible quality and only aggravated. Never should it be upscaled, for example. The content should be presented in a pseudo-random order [8].

Before the actual testing, all subjects should be presented with the best and worst quality image or video available so that they can experience the rating protocol and do not have different expectations according to [7]. A short period of training should also be included so that everyone becomes familiar with the methodology [8].

It should be noted [2] that creating a complex questionnaire with detailed sections can lead to subjects misunderstanding them or becoming confused. It can compromise the whole test as everyone will score differently. The questionnaire should be as simple and short as possible while providing all the necessary data. A detailed explanation of the testing must be provided to the subjects.

4.2.2 QoE testing conditions

Every user has a different notion of the outside world, therefore basic information about them is needed, such as age, gender and prior VR experience. VR experience is probably the most important one [2], as people might have different expectations. First-time users, for example, might be immersed in the VR world and not pay attention to such details or have a higher chance of having cybersickness. Basic screening for visual acuity should be made [8]. At least 28 subjects should participate in the test to obtain truly representative data [8].

All tests should be carried out in a quiet and calm environment. If the test is made for a static scene – the subject can not move in the world, only look around – all subjects should be seated on a swivel chair, so they are able to turn around, but not move in the space to prevent cybersickness [8].

The optimal duration for a single stimulus, e.g. video or picture, should be about 20 seconds [7]. 10 seconds is too short for the subject to orient himself in the world and 40 seconds is too long and fatigue can be encountered. Table 4.1 [7] shows the MOS standard deviation σ for 10, 20 and 40 seconds viewing sequences. During a single session, each subject's participation should be limited to 1.5-hour rating stimuli and no more than 25 minutes continuously [8]. More than 25 minutes can cause dizziness and cybersickness [7].

Duration	VS#1	VS#2	VS#3	VS#4	VS#4	Avg.
10s	9.4	6.5	14.7	15.5	12.5	11.7
20s	6.5	6.1	5.7	7.4	6.4	6.4
40s	6.3	6.6	5.6	9.3	5.5	6.7

Table 4.1: Standard deviation σ of MOS of repeated image samples over five random viewing sequences (VS) (table adapted from[7]).

4.3 Existing testbeds for measuring subjective QoE in VR

Several successful testbeds for subjective QoE in a VR environment have already been created.

One of them is VRate[3] [18]. VRate is an open-source asset for Unity3D that allows the integration of various questionnaires, quality ratings and experience measurements in VR. It provides a VR environment for subjective self-assessed experience measures. It consists of two modules. The first module is for the user, which provides an environment for playing omnidirectional video and an interface to enable the ratings. The second module is for the operator, which enables him to control the whole process. The modules have been successfully tested in a QoE study [21] with 27 users and a UX study with 48 users. In

[3] http://vrate.tech-experience.at/

the future, the authors want to implement various objective metrics and head and eye-tracking for better data analysis.

Another interesting tool is Testbed for subjective evaluation of omnidirectional visual content[4] [19]. The application allows viewing omnidirectional images and video with various projections and provides researchers with a tool to perform subjective QoE testing in VR. The application is open-source and enables the implementation of testers' own projections and parameters. The platform also tracks head direction and time spent on various stimuli.

The last testbed mentioned in this work is the Framework to evaluate omnidirectional video coding schemes[5] [20]. It is a platform for the evaluation of the coding efficiency of various omnidirectional content. It also enables head motion tracking.

[4]https://github.com/mmspg/testbed360-android
[5]https://github.com/mattcyu1/omnieval

Chapter 5

Tools for image quality data analysis

This chapter focuses on the practical part of my thesis. I am creating a pilot application for data analysis of subjective image quality assessment in virtual reality. Section 5.1 of this chapter briefly describes the process of gathering data for my application and their structure. Section 5.2 describes the application's structure and explains all its components. Section 5.3 explores the process of analyzing subjective scores. Section 5.4 goes through the eye-tracking data analysis, with a subsequent section 5.5 explaining in greater detail the implementation of those data for objective assessment of image quality. Section 5.6 gives an overview and serves as a manual for potential users explaining all the graphical user interface (GUI) components and input parameters with their impact on the final scores. Finally, section 5.7 shows sample outputs of the application and discusses them.

5.1 Data gathering for evaluation

Before creating an automated application for data evaluation, the data themselves need to be gathered. This was done using an application currently under development by a master student at the Department of Radioelectronics, FEE CTU in Prague[1]. The application can also perform eye-tracking which can be later examined. The currently used headset is HTC Vive Pro Eye with hand controllers and base stations for tracking user movement.

The usage of the application by the user is very simple. There are currently 116 different images that are subsequently projected in VR around the user. For every scene, there is a 10-second period where they can examine the scene. However, if they quickly decide that the scene is very low quality, they can close both eyes for 1 second and skip the current scene. Between projecting every scene, the user must give an evaluation of the scene quality. They can do this by using the MOS scale. There are 5 cubes projected in front of them. On the left, there is a red cube representing the worst score – 1, and on the right, there is a green cube representing the best score – 5. Users can choose any of those ratings by simply touching the cube using a controller they have been given.

[1] Bc. Jakub Špaňár

Before evaluating all of the scenes, users need to fill in a simple questionnaire asking for their name, age, past VR usage experience, dioptries rating of their spectacles, if they are using any, and whether they have participated in any kind of subjective image quality testing in the past. Before and after all the testing they also need to fill in a simulator sickness questionnaire [24]. It contains several health conditions like their current level of nausea, vertigo, headache and so on. Before the testing, a brief explanation of the application's features was given to all users. The process is shown in figure 5.1.

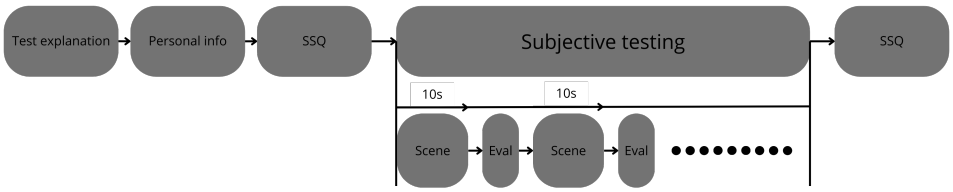

Figure 5.1: Flow of subjective testing.

There are 5 different scenes used for the testing – 02_Hokkaido, Biscayne, Flowers, Telescope and Trains. They have been provided by the OMNIQAD database[2] created at Brno University of Technology, and FEE CTU in Prague [26]. The quality of those 5 scenes is then gradually lowered by 4 compressions – AVIF, HEIC, JPG and JXL. The number of each quantisation parameter (QP) varies for each compression. In total, there are 116 unique images that are projected in front of the subjects.

Output data provided by the testing platform are mostly in the form of .txt files. The first and probably most important file is the one with the actual subjective scores everyone has given. It contains the name of the scene, (QP) and compression in one word – for example Biscayne60AVIF. This is followed by the actual subjective score of the scene. An example of an output file with subjective scores is in figure 5.2. Note that the file is not complete as it contains 116 rows.

A second .txt output file contains eye-tracking data. These contain cartesian coordinates – x, y and z values – together with time, which is measured every 100 ms. Relation between coordinates and the projected scene is shown in figure 5.3.

All 4 values are separated by a semicolon and space. Those data are grouped by every scene, the name of which is written above those data. A randomly chosen txt eye-tracking file is shown in figure 5.4. Note that the file is not complete.

[2]https://doi.org/10.5281/zenodo.7607071

```
02_Hokkaido59AVIF 3
Flowers20HEIC 5
Flowers13JXL 5
Biscayne14HEIC 4
02_Hokkaido16HEIC 3
Trains61AVIF 4
Telescope53AVIF 2
Telescope9JPG 1
02_Hokkaido0JXL 2
02_Hokkaido29HEIC 1
Biscayne47AVIF 3
Trains23HEIC 4
Flowers18JPG 4
Telescope56AVIF 2
```

Figure 5.2: An example of an output file containing subjective scores.

```
02_Hokkaido59AVIF
-4.987001; 0.3540808; 0.06672812; 0.2
-4.725479; 0.2532109; 1.614229; 0.3
-4.647421; 0.1466075; 1.838474; 0.4
-4.712827; 0.148028; 1.663535; 0.5
-4.792686; 0.5829196; 1.30014; 0.6000003
-4.775608; 1.242167; 0.8065929; 0.7000005
-4.81683; 1.146207; 0.6959562; 0.8000008
-4.752483; 1.350886; -0.7674742; 0.900001
-4.74516; 1.367489; -0.7832193; 1.000001
-4.721788; 1.48033; -0.716476; 1.100002
-4.435492; 1.651672; -1.611955; 1.200002
-4.480608; 1.611859; -1.525146; 1.300002
-3.869655; 1.666543; -2.692286; 1.400002
-1.469335; 1.240779; -4.615358; 1.500003
-1.695421; 1.309032; -4.517963; 1.600003
-0.2942981; 1.242161; -4.834298; 1.700003
-0.4059036; 1.299535; -4.811076; 1.800003
-0.5518327; 1.289865; -4.799139; 1.900004
```

Figure 5.4: An example of an output file containing eye-tracking data.

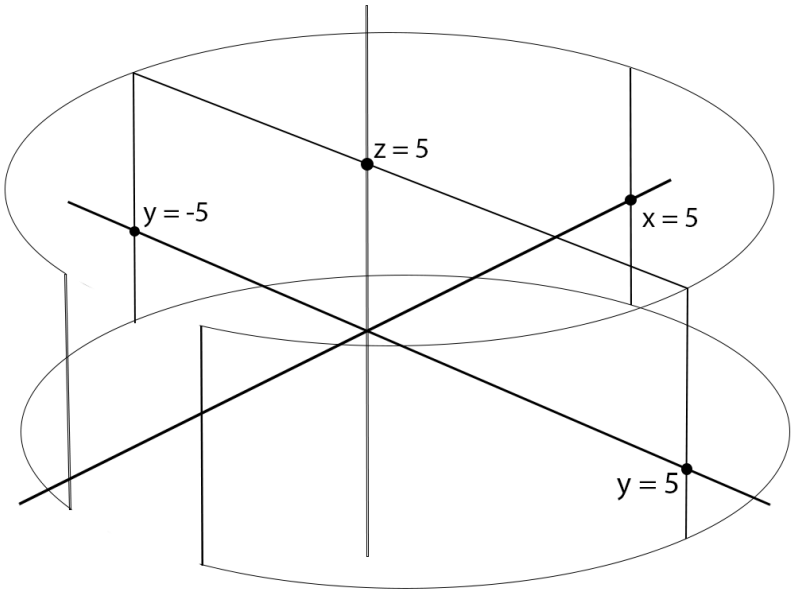

Figure 5.3: Illustration of the scene and coordinates realtion.

5.2 Basic application structure

The application's basic functionality needs to be divided into three main parts. There needs to be a separate block for processing received data and for data projection, since a real-time demonstration is not feasible due to time-consuming computations. There will also be an intermediate step between those two blocks, as all processed data will be first stored before projecting. The general scheme is shown in figure 5.5.

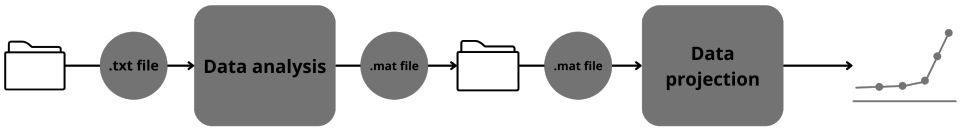

Figure 5.5: General structure of application design.

A simple GUI described in 5.6 needs to be created not only for data projection but also for processing data control. It needs to be specified what files will be loaded and input parameters need to be added. Since the whole application is coded in Matlab, it is worthwhile creating the GUI in Matlab's

App Designer[3], which is an easy-to-use environment for GUI creation.

The GUI itself has three pages. The first page is for loading and analysing the data. The second page shows the processed data and the third page shows sample images from testing overlaid with eye-tracking data.

5.3 MOS data analysis

The main back-end function is called *main()* and is called whenever data need to be analysed by the load page of the GUI. When called, there are several input arguments defining the load content and parameters for objective image quality metrics. After receiving the load content that consists of several scenes and compressions, multiple tables are created with MOS scores sorted using scenes by the *sceneSort()* function. When tables are created, a *graphSave()* function is called multiple times with a single scene for every call as an input parameter. *GraphSave()* then calls all the required functions that calculate objective metrics, and stores all those data in a new struct variable that is then saved as a .mat file into a separate folder. Those data can be subsequently loaded and projected in the GUI. The general scheme is shown in figure 5.6.

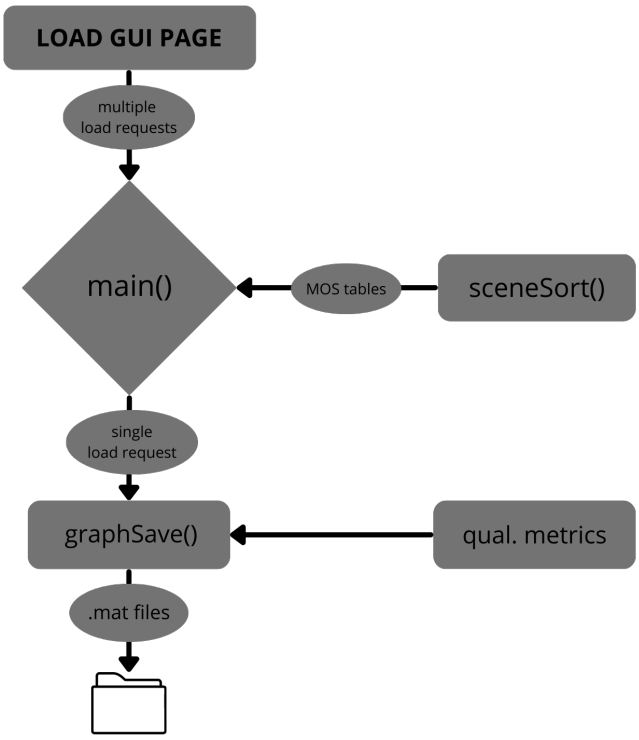

Figure 5.6: General structure of application communication.

[3]https://ch.mathworks.com/products/matlab/app-designer.html

Several functions handle .txt file loading, as multiple formats were used to store data. A function called *txtLoad()* loads subjective scores and the second function called *eyeLoad()* loads eye-tracking data. Both these functions output the loaded data into a *txtSort()* function which formats the given data. It uses *textFormat()* to split scene names, since they are formatted as one string, shown in the previous figure 5.2.

Each subject's score is then checked using the *outliers()* function to determine whether they are not too different from other subjects' score. In this function, every score for a given scene is checked across all subjects and if the given score, according to, by default, the three-sigma rule, is outside more than three standard deviations from the other scores, the subject is logged. Such computation is shown in equation 5.1 and 5.2. If those equations are true than the subject is logged. The coefficient three is set by default but the user can input his own number.

$$score < mean\_score(MOS) - 3 \times standard\_deviation(MOS) \tag{5.1}$$

$$score > mean\_score(MOS) + 3 \times standard\_deviation(MOS) \tag{5.2}$$

In the end, if any of the subjects have more than, by default, 10% of all their scores outside the three standard deviations interval, the given subject is removed from any other analysis. Such computation is shown in equation 5.3. If the equation is true, given subject is removed from any other analysis. The coefficient 10 is set by default but can be altered by the user.

$$number\_of\_loggs > \frac{numb\_of\_subjects}{10} \tag{5.3}$$

After checking for consistency, all subjects' MOS data are averaged for each scene using the *avgData()* function. Output from this function is then finally sent to *sceneSort()* and then to *main()*.

When loading sorted MOS to the *main()* function using the *sceneSort()*, tables with all the used scenes and compression are required for addressing and inputting correct data along the whole saving process. These tables are created using a *sceneTable()* function, which gets data from *outliers()*, since those do not need averaging. A diagram of the whole system is shown in figure 5.7.

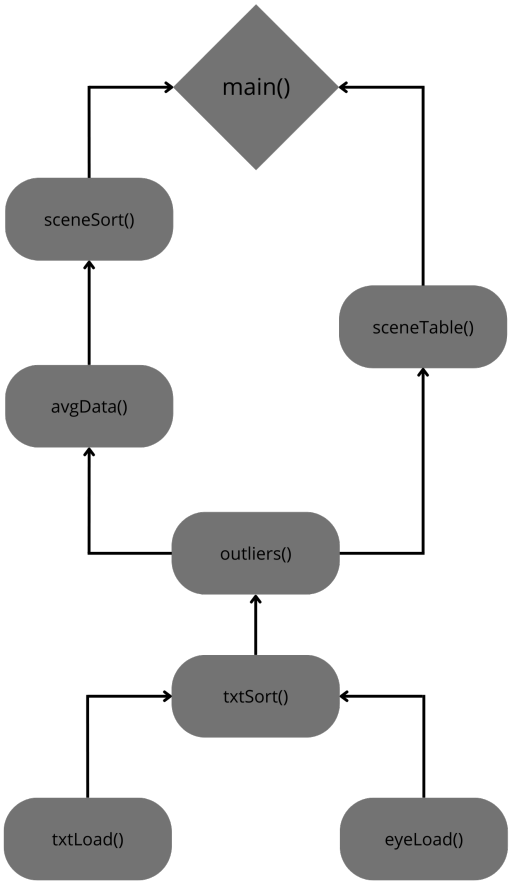

Figure 5.7: General process of analysing data.

5.4 Analyzing eye-tracking data

When loading eye-tracking data, coordinates conversion needs to be done. This is all done using the *eyeLoad()* function which loads all eye-tracking .txt files, sorts them and then converts them.

Loaded data are in 3D cartesian coordinates, since the environment they were captured in utilized them. However, this is not feasible for further use, as mapping on 2D images is required. Conversion to spherical coordinates is desirable where the horizontal angle is represented by θ, the vertical angle is represented by ϕ and the distance from the origin is represented by r. Since ERP projection has been used to warp 2D images around the user, all points are on a surface of a sphere and they have the same r. After conversion, only θ and ϕ are used, representing x and y coordinates respectively. The

conversion is shown in figure 5.8.

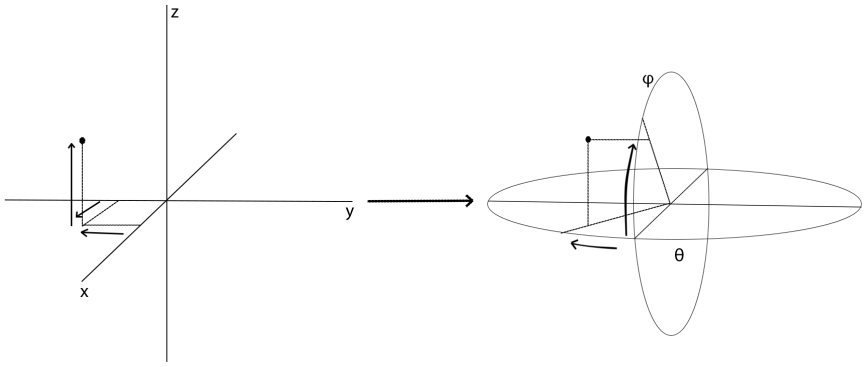

Figure 5.8: Illustration of conversion from cartesian to spherical coordinates.

In Matlab, a specialized function exists for conversion between cartesian and spherical coordinates. The function is *cart2sph()* and has output parameters θ, ranging from -π to +π, ϕ, ranging from -$\pi/2$ to +$\pi/2$, and r. Conversion equation for θ is 5.5, for π is 5.6 and for r is 5.4.

$$r = \sqrt{x^2 + y^2 + z^2} \tag{5.4}$$

$$\theta = \tan\left(\frac{y}{x}\right) \tag{5.5}$$

$$\varphi = \tan\left(\frac{\sqrt{x^2 + y^2}}{z}\right) \tag{5.6}$$

After conversion, both θ and ϕ have been additionally normalised to the range of 0-1 for easier future implementation. Demonstration of the resulting coordinates is shown in figure 5.9.

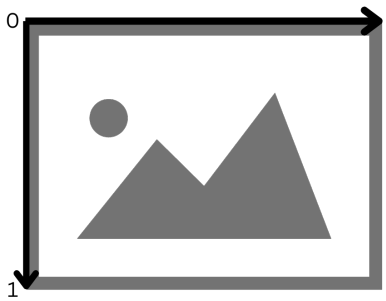

Figure 5.9: Resulting coordinates with reference to a sample image.

5.5 Objective metrics calculation

With images provided from subjective testing, various objective metrics like MSE 3.2, PSNR 3.3, SSIM 3.6, MS-SSIM 3.7 or FSIM 3.8 can be calculated and then compared with MOS. In the application, all those metrics are implemented at this stage, with possible more metrics to be added in the future.

5.5.1 Metrics used and their Matlab implementation

Matlab's own library provides a range of image quality metrics[4], which makes them fairly easy to use in my application.

Output parameters of these functions can be generally split into two types in Matlab. The first type is an output of just the estimated quality as a single value. The second type is the quality value and a map of quality values for every pixel of the image – similarity map. The second option will become useful when utilizing eye-tracking data in the next chapter. Out of all the above-mentioned metrics, in Matlab PSNR, MSE and FSIM output only a single estimated quality value and SSIM and MS-SSIM output quality value and a similarity map.

FSIM is the only metric used that has no Matlab implementation. However, Matlab implementation can be found on GitHub [22]. The code was imported into my project as the *FSIM()* function.

Calculation of objective metrics is done in *graphSave()* function by calling *metricsSort()*. Input parameters of this function are a load table of all the scenes to be evaluated, a metrics table, which defines what metrics are to be calculated, and a few other parameters used to calculate the similarity map.

Inside the *metricsSort()* function, metrics without an estimated similarity map are calculated directly by calling the appropriate function. For PSNR the function being *psnr()* and for MSE *immse()*. A diagram of the functions is shown in figure 5.10.

5.5.2 Eye-tracking utilization for SSIM

Apart from the overall similarity value, SSIM and MS-SSIM metrics create a similarity map defining the estimated similarity value between every pixel of the given image and its reference. Before calculating the overall similarity value by averaging all the pixels, the map can be weighted to make some areas more prominent than others. This can be particularly useful in scenes where a large part of the image is a blue sky for example. When compressing, these areas' quality will mostly look the same or at least will not deteriorate nearly as fast as those with complex content. In the end, after averaging all the values, the score will likely be higher than the subjective MOS score. The reason for this, apart from the core metric's efficiency, is human perception. Humans will mostly look at those places where they can determine image

[4]Image Processing Toolbox is required

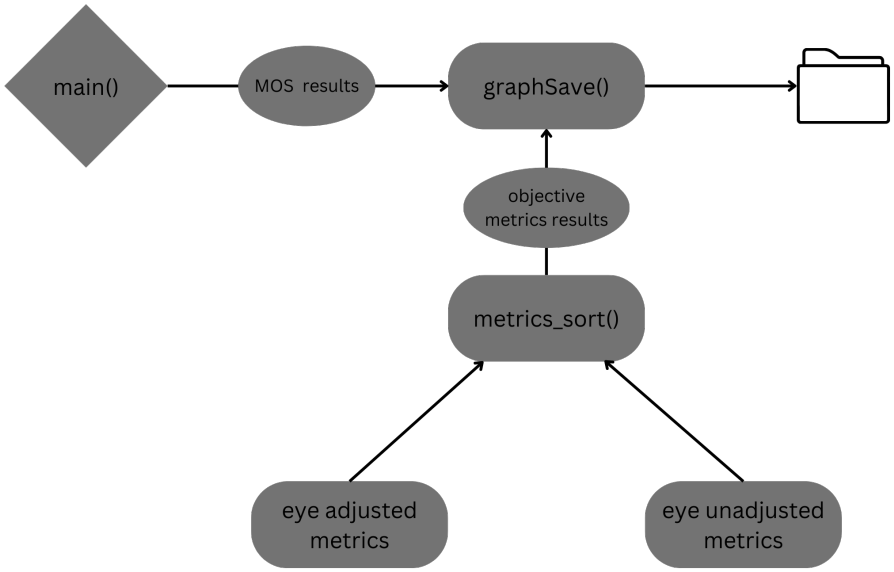

Figure 5.10: A diagram of a structure of functions processing image quality metrics.

quality the best. Those places are areas with complex content that will suffer substantially more from compression, compared to those depicting blue sky. An illustration is in figure 5.11. Note the difference between the house and the sky.

In order to alleviate this deficiency, eye-tracking data can be used to create a weight map of a given image. Function *weightMap()* creates this map, based on several input factors. The first input parameter is the eye-tracking data, which are being collected by the *averagedScores()* function. Since we want eye-tracking data from all subjects, all the points are added together into one big table that usually consists of around 500 points for a given testing methodology. The second parameter is given image resolution. By default, a weight map will be created, with a resolution matching that of the image it is to be placed over. However, since most images are at 8K resolution, calculations might take too long, so it is possible to scale the weight map using input parameter scale_factor. It ranges from 1 being no change, to 16, being 1/16 of the original image width and height. This parameter also scales the similarity map to match the weight map.

When creating a weight map, all of the eye-tracking data points are mapped to the corresponding pixels on the map. In order to do this, they need to be multiplied by the image width and height resolution. This can be done since all the values of the eye-tracking data are normalised to the range of 0-1. This, however would create only around 500 points on a huge map of approximately 32 000 000 pixels. Some blur is necessary to spread the points over a larger area and smooth the sharp edges these single points would make. A demonstration of such a blur is in figure 5.12.

Gaussian blur is the most fitting choice for this purpose, since its simple

Figure 5.11: The difference between uncompressed and compressed images (image taken from [26]).

use in Matlab and the possibility to adjust the amount of blur. Matlab has a specific function for this called *imgaussfilt()*, with input parameters being the image itself and sigma being the standard deviation with which it can adjust the amount of blur.

Setting the right sigma is crucial, since it reflects the human visual system's characteristics. When focusing on a single point, the human eye can not see sharply over its whole field of view (FOV) but only over a small area. A human can see sharply only about 2°-8° of the FOV depending on what is he looking at [23]. A different value would be given for reading a book and for looking at a landscape in the distance. An illustration of the human FOV is in figure 5.13.

It would be desirable for the user to input a FOV value when loading data, in order to be able to experiment. Such values need to be then recalculated to sigma values. In fact, in the case of spherical projection two sigma values need to be calculated – one for horizontal blur and the second for vertical blur. This is due to the nature of spherical projection where horizontal coordinate values are 0°-360° but vertical are only 0°-180°. In terms of equirectangular projection, which is used, different numbers of pixels will represent 5° horizontally and vertically. Calculation of horizontal sigma is

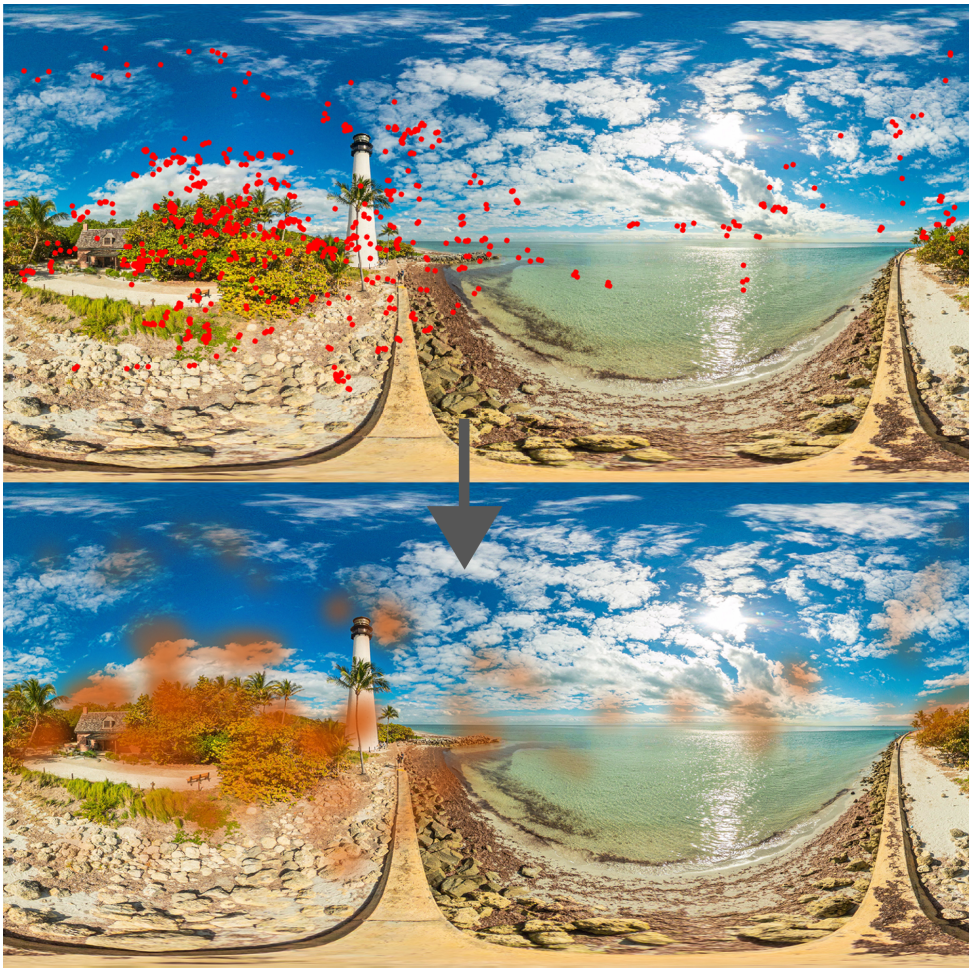

Figure 5.12: Sample eye-tracking data and desired heatmap (image taken from [26]).

shown in equation 5.7 and for vertical sigma in equation 5.8. The reason they are divided by four comes from the nature of gaussian distribution and its standard deviation. If a single value should be blured only inside the desired FOV, the interval needs to be shorten otherwise about 32% of the blur would be outside of it. An illustration is shown in figure 5.14. The tighter scope is without division, the wider scope is with division by 4.

$$\sigma_{hor} = FOV \times \frac{(res_{hor}/360)}{4} \tag{5.7}$$

$$\sigma_{ver} = FOV \times \frac{(res_{ver}/180)}{4} \tag{5.8}$$

Figure 5.13: Human eye FOV demonstration.

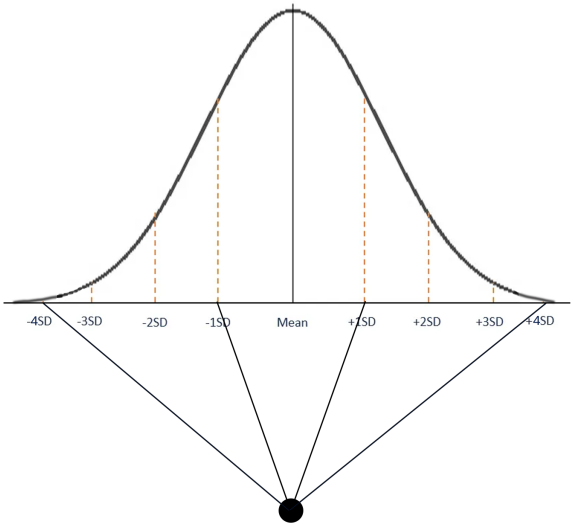

Figure 5.14: Illustration of FOV in relation to standard deviaton.

These sigma values are then input in the Gaussian filter *imgaussfilt()* as a vector, resulting in a smoother weight map that corresponds with sharp human FOV.

The created weight map is then sent to the *ssim_compare()* function. There, the calculations of the SSIM and MS-SSIM values and maps are done using the *ssim()* and *multissim()* functions, where multissim stands for MS-SSIM metric.

The resulting maps of the metrics consist of three parts – the luminance

term, the contract term and the structural term [13]. Each part is then multiplied by the appropriate weight map value, using matrix multiplication. After all three multiplications are done, all values are averaged to a single value representing the final SSIM or MS-SSIM.

5.6 Application usability

This chapter describes components of all three pages of the GUI with descriptions and recommendations of possible input settings.

All data displayed in the following figures are measured on the OMNIQAD database[5].

5.6.1 Load page

This page serves for creating data analysis. Users can choose from all the scenes that are available in the application's directory and calculate objective metrics on them. The load page of GUI is shown in figure 5.15.

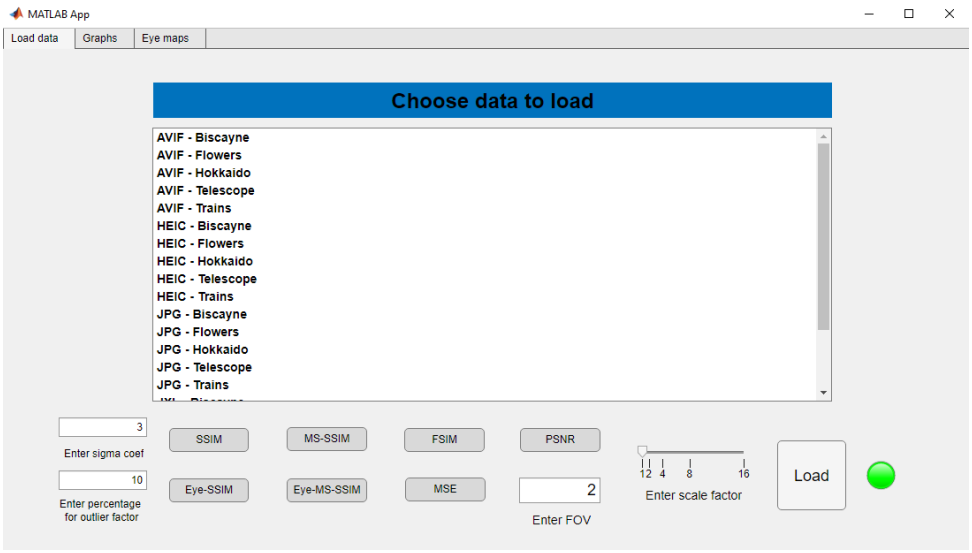

Figure 5.15: Default load page of GUI.

At the centre of the page, there is a list of possible load options. They are sorted first by their compression and then by the scene. All items on the list are dynamically loaded when starting the application. Multiple scenes can be loaded simultaneously, which also decreases the time needed for calculations since MOS is calculated first and used in all load scenes.

In the bottom left, there are two fields for sorting subjects in the *outliers()* function. 'Sigma' alters the standard deviation distance and 'outlier coef' alters the percentage from which subjects will be removed.

[5] https://doi.org/10.5281/zenodo.7607071

To the right, there are several buttons that load objective metrics. Eye-SSIM and Eye-MS-SSIM mean the metric will utilize a weight map for metric calculations. With this option, an editable field with FOV is added for the possibility to adjust the weight map. Values between 1-8 degrees create a reasonable weight map given human eye sharp FOV. Increasing this value can, however, increases the time it takes to generate such a map. For this reason, a scale factor slide is added to reduce the weight map resolution which otherwise matches the resolution of the image it is supposed to weigh. This scale factor will divide both dimensions of the image so the resolution then decreases quadratically.

Finally, on the bottom right, there is a load button to load all the selected scenes with an indicator that shows if there are any calculations currently undergoing. Green means no calculations are being done so the user can load scenes and red means calculations are undergoing.

5.6.2 Graphs page

This second page, shown in figure 5.16, serves for presenting analysed data. Users can choose from the list of scenes on the left. It is possible to show multiple scenes simultaneously so various scenes can be compared.

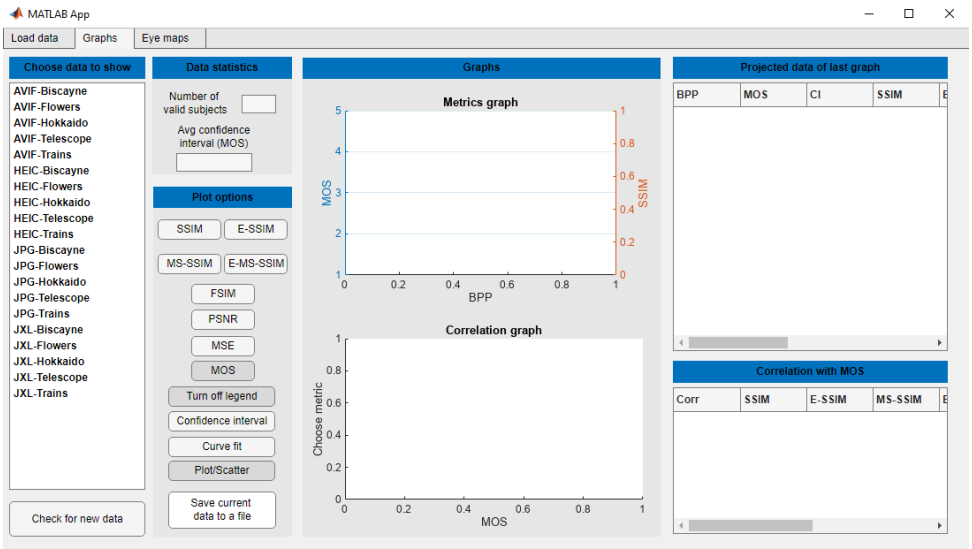

Figure 5.16: Default graphs page of GUI.

In the centre, two graphs are located. The top one shows the dependence of MOS and other metrics on bits per pixel (BPP). The bottom one shows the correlation values between MOS and currently chosen objective metrics.

In the data statistics panel, the number of subjects that generated those graphs is shown with an average confidence interval of all MOS points currently shown.

Beneath, there is the plot options panel where users can choose metrics to show. This will affect both the top and bottom graph. A few other options

are added such as 'Confidence interval' options, which shows the confidence interval along every MOS point shown in the graph. Curve fitting the data is also possible with the 'Curve fit' button and toggling between scatter and plot is possible. If needed, the user can turn off the legends so they do not take up too much space. Finally, a 'Save' button is added for the user to be able to save the last entered data to a .CSV file.

On the right, two panels are located. The top one shows all data currently projected in a graph and the bottom one shows correlation values between MOS and all available metrics. Spearman's rank correlation coefficient and Pearson correlation coefficient are calculated.

5.6.3 Eye-tracking maps page

The last page of the GUI, shown in figure 5.17, is just a simple showcase of all eye-tracking data being done during the testing. Users can choose among all available scenes dynamically since everything is calculated when the application launches.

The items in the list contain three parts, the first being the name of the scene, the second being compression used and the third one being quality level measure in BPP.

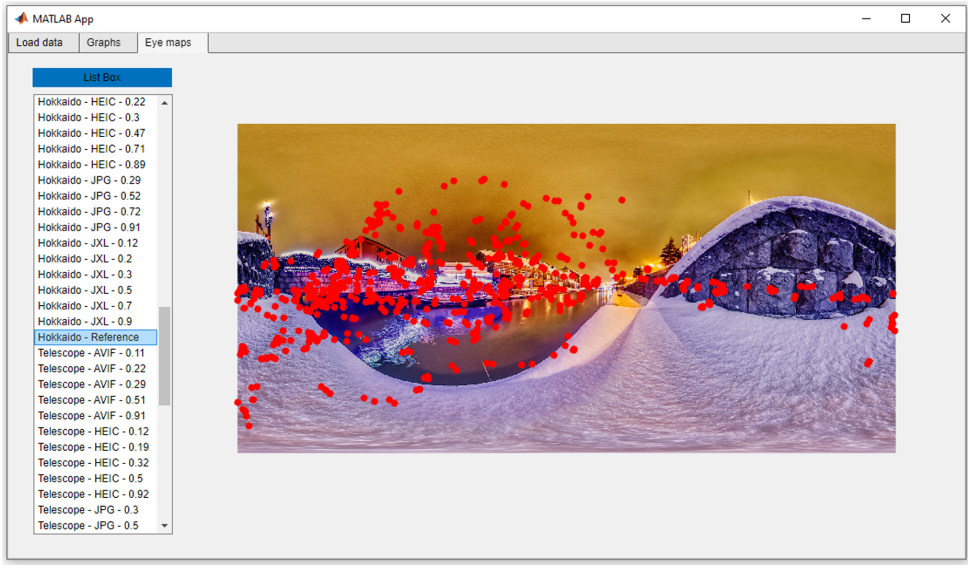

Figure 5.17: Default eye-tracking maps page of GUI.

5.7 Data demonstration

In this section, a very brief demonstration of outputs is shown. Since only 12 subjects participated in the testing, no valid conclusions can be made. However, looking at the graphs, it is possible to see trends in the data.

In figure 5.18 a sample data output is shown on the AVIF-Biscayne scene for 2° FOV and scale factor 1. In the upper graph MOS, SSIM and Eye-SSIM

metrics are displayed based on BPP. A similar trendline is clearly visible. At the bottom, there is a parametric graph showing the dependency of SSIM and Eye-SSIM on MOS. The scatter is quite consistent, indicating a fairly good correlation. This is confirmed by the correlation values of PLCC and SRCC 0.9125 and 0.8333 for SSIM and 0.9208 and 0.7857 for Eye-SSIM.

It can be seen that, by utilizing eye-tracking data, PLCC for SSIM grew from 0.9125 to 0.9208. However, SRCC dropped from 0.8333 to 0.7857. This kind of behaviour appears for all metrics showing somewhat minimal effect of the weighting.

Also, with the given amount of subjects, results are statistically unreliable. Further testing needs to be done to prove the platform useful for real-world applications. Tuning the input parameters and algorithm for metrics weighting is also required to yield better correlation results.

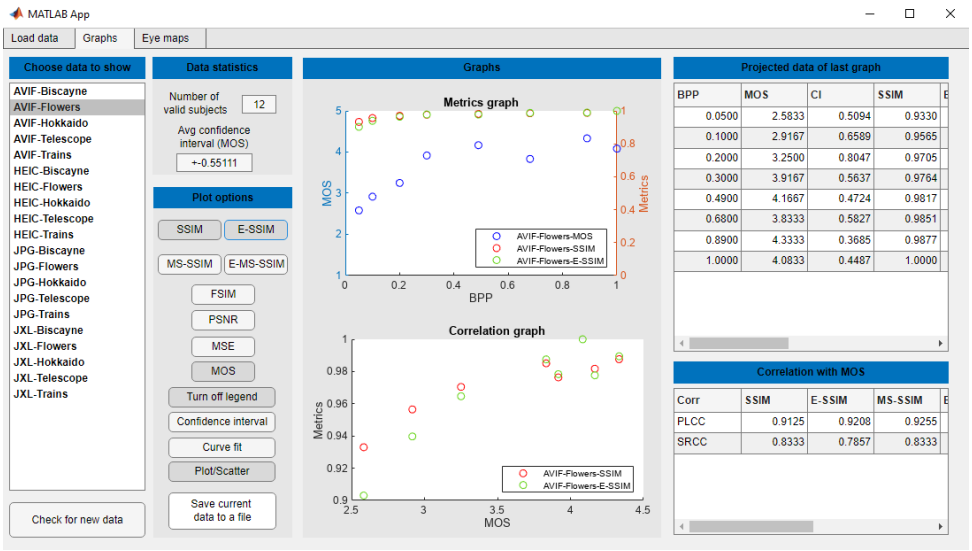

Figure 5.18: A sample showcase of displayed data – AVIF-Flowers, MOS, SSIM and MS-SSIM metrics showed, scatter option.

In figure 5.19 MOS and SSIM is shown with curve fit performed. In the upper graph the fit curve replaces unfitted data. In the bottom graph the fit curve is added to the normal scatter.

In figure 5.20 MOS and SSIM are displayed with the 'plot' option rather than 'scatter' and with confidence intervals on. It can be seen that the intervals are fairly large but the general trend still remains. In the lower graph a zig-zag line can be seen. This is created due to the drop of MOS in BPP between 0.6 and 0.8.

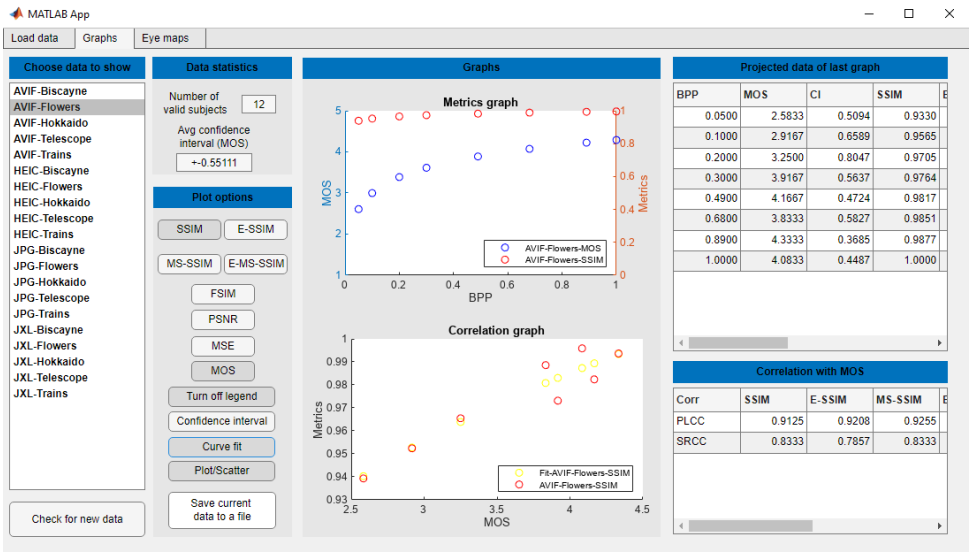

Figure 5.19: A sample showcase of displayed data – AVIF-Flowers, MOS and SSIM metrics showed, curve fit on, scatter option.

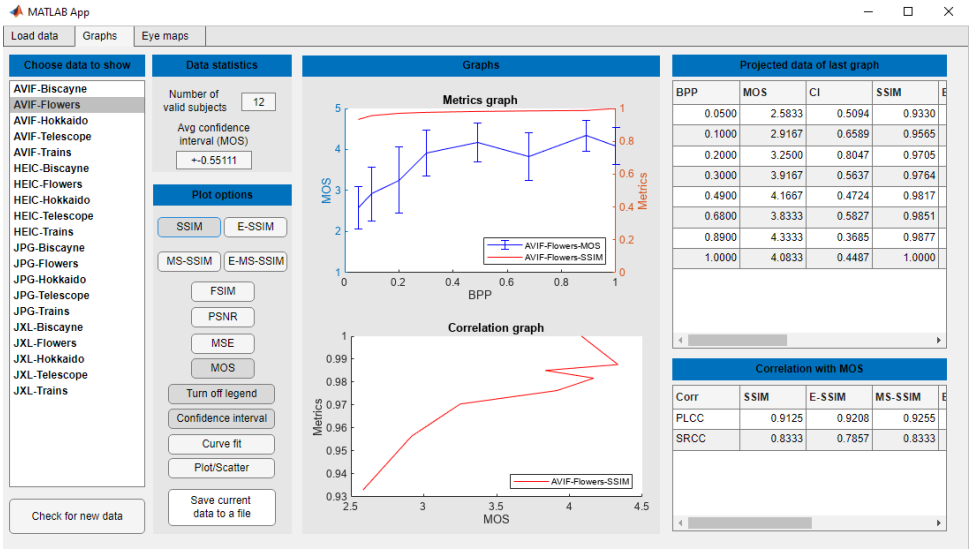

Figure 5.20: A sample showcase of displayed data – AVIF-Flowers, MOS and SSIM metrics showed, confidence interval on, plot option.

5.8 Discussion

In this chapter, an automated tool for image quality data analysis was made. Its easy-to-use environment enables users to fully focus on research rather than processing measured data.

Objective metrics can be created as well and compared to subjective testing results. As a major contribution, the utilization of eye-tracking data has been made to possibly improve objective metrics performance. The outcomes of eye-tracking utilization can not be concluded since not enough testing has been made. However, with tuning and further improvements, results can be promising.

Chapter 6

Conclusion

In this work, I tried to create an automated platform for image quality analysis. The tool was supposed to analyze both subjective scores and obejctive metrics like SSIM or PSNR which could be then compared to each other. A promising new approach for objective metrics improvement by utilizing eye-tracking data was supposed to be tested.

In the theoretical part of my thesis I reviewed several omnidirectional video projection methods and compared objective metrics for image quality assessment such as PSNR, MSE, VSI, SSIM and FSIM, with FSIM turning out to be the most effective one.

Then, I reviewed various factors impacting the QoE in virtual reality such as bitrate, compression rate or stalling events. All those factors could be then included in QoE features such as image quality, fragmentation, immersion and cybersickness, which all contribute to the overall QoE.

Before doing a practical experiment, I also made a brief recommendation for subjective image quality testing methodology in virtual reality and showed a couple of existing testbeds for such testing.

In the practical part of my thesis, I introduced an automated tool for image quality analysis. The easy-to-use environment of the tool enables users to fully concentrate on the results and eliminates prolonged data sorting and calculation. It also enables the comparison with objective metrics, such as MSE, PSNR, SSIM, MS-SSIM and FSIM, with possible further addition. A large part of the thesis was given on the potential for eye-tracking data utilization. Suitable metrics, which created a map, were weighted using those data. This enables testing for any potential improvement of those metrics if the right input data is given.

So far not a very significant improvement with eye-tracking base weightmaps was exhibited among the scenes. Therefore, further improvement of the application will be required together with tuning the input parameters.

Bibliography

[1] CHEN, Zhenzhong, Yiming LI a Yingxue ZHANG. Recent advances in omnidirectional video coding for virtual reality: Projection and evaluation. Signal Processing. 2018, 146, 66-78 [cit. 2022-12-04]. [Online]. ISSN 01651684. doi:10.1016/j.sigpro.2018.01.004

[2] SCHATZ, Raimund, Andreas SACKL, Christian TIMMERER a Bruno GARDLO. Towards subjective quality of experience assessment for omnidirectional video streaming. 2017 Ninth International Conference on Quality of Multimedia Experience (QoMEX). [Online]. IEEE, 2017, 2017, 1-6 [cit. 2023-02-03]. ISBN 978-1-5386-4024-1. doi:10.1109/QoMEX.2017.7965657

[3] TRAN, Huyen T. T., Nam Pham NGOC, Cuong T. PHAM, Yong Ju JUNG a Truong Cong THANG. A subjective study on QoE of 360 video for VR communication. 2017 IEEE 19th International Workshop on Multimedia Signal Processing (MMSP). [Online]. IEEE, 2017, 2017, 1-6 [cit. 2023-02-04]. ISBN 978-1-5090-3649-3. doi:10.1109/MMSP.2017.8122249

[4] DUAN, Huiyu, Guangtao ZHAI, Xiongkuo MIN, Yucheng ZHU, Yi FANG a Xiaokang YANG. Perceptual Quality Assessment of Omnidirectional Images. 2018 IEEE International Symposium on Circuits and Systems (ISCAS). [Online]. IEEE, 2018, 2018, 1-5 [cit. 2023-02-04]. ISBN 978-1-5386-4881-0. doi:10.1109/ISCAS.2018.8351786

[5] FAN, Ching-Ling, Tse-Hou HUNG a Cheng-Hsin HSU. Modeling the User Experience of Watching 360° Videos with Head-Mounted Displays. ACM Transactions on Multimedia Computing, Communications, and Applications. 2022, 18(1), 1-23. [Online]. [cit. 2023-02-03]. ISSN 1551-6857. doi:10.1145/3463825

[6] YAO, Shun-Huai, Ching-Ling FAN a Cheng-Hsin HSU. Towards Quality-of-Experience Models for Watching 360° Videos in Head-Mounted Virtual Reality. 2019 Eleventh International Conference on Quality of Multimedia Experience (QoMEX). [Online]. IEEE, 2019, 2019, 1-3 [cit. 2023-02-04]. ISBN 978-1-5386-8212-8. doi:10.1109/QoMEX.2019.8743198

[7] HUANG, Mingkai, Qiu SHEN, Zhan MA, Alan Conrad BOVIK, Praful GUPTA, Rongbing ZHOU a Xun CAO. Modeling the Perceptual

Quality of Immersive Images Rendered on Head Mounted Displays: Resolution and Compression. [Online]. IEEE Transactions on Image Processing. 2018, 27(12), 6039-6050 [cit. 2023-02-03]. ISSN 1057-7149. doi:10.1109/TIP.2018.2865089

[8] ITU-T. (2019). ITU-T P.919: Subjective video quality assessment methods for multimedia applications, accessed January 12, 2023. [Online]. Available: https://www.itu.int/rec/T-REC-P.919

[9] KARA, Peter A., Werner ROBITZA, Maria G. MARTINI, Chaminda T.E.R. HEWAGE a Fatima M. FELISBERTI. Getting used to or growing annoyed: How perception thresholds and acceptance of frame freezing vary over time in 3D video streaming. [Online]. IEEE, 2016, 2016, 1-6 [cit. 2023-02-04]. ISBN 978-1-5090-1552-8. doi:10.1109/ICMEW.2016.7574686

[10] CHI-WING FU, LIANG WAN, TIEN-TSIN WONG a CHI-SING LEUNG. The Rhombic Dodecahedron Map: An Efficient Scheme for Encoding Panoramic Video. [Online]. IEEE Transactions on Multimedia [online]. 2009, 11(4), 634-644 [cit. 2023-02-04]. ISSN 1520-9210. doi:10.1109/TMM.2009.2017626

[11] Facebook, "Next-generation video encoding techniques for 360 video and VR", accessed November 29, 2022. [Online]. Available: https://code.facebook.com/posts/1126354007399553/next-generation-video encoding

[12] UMAR, A. S., R. M. SWASH a A. H. SADKA. Subjective quality assessment of 3D videos. [Online]. IEEE Africon '11. IEEE, 2011, 2011, 1-6 [cit. 2023-02-05]. ISBN 978-1-61284-992-8. doi:10.1109/AFRCON.2011.6071968

[13] WANG, Z., A.C. BOVIK, H.R. SHEIKH a E.P. SIMONCELLI. Image Quality Assessment: From Error Visibility to Structural Similarity. [Online]. IEEE Transactions on Image Processing. 2004, 13(4), 600-612 [cit. 2023-05-21]. ISSN 1057-7149. doi:10.1109/TIP.2003.819861

[14] Dialpad, "Mean Opinion Score (MOS)", accessed February 2, 2023. [Online]. Available: https://www.dialpad.com/glossary/mos-score/

[15] ZHANG, Lin, Ying SHEN a Hongyu LI. VSI: A Visual Saliency-Induced Index for Perceptual Image Quality Assessment. [Online]. IEEE Transactions on Image Processing. 2014, 23(10), 4270-4281 [cit. 2023-02-05]. ISSN 1057-7149. doi:10.1109/TIP.2014.2346028

[16] LIN ZHANG, LEI ZHANG, XUANQIN MOU a D. ZHANG. FSIM: A Feature Similarity Index for Image Quality Assessment. [Online].IEEE Transactions on Image Processing. 2011, 20(8), 2378-2386 [cit. 2023-02-05]. ISSN 1057-7149. doi:10.1109/TIP.2011.2109730

[17] ANMIN LIU, WEISI LIN a M. NARWARIA. Image Quality Assessment Based on Gradient Similarity. [Online]. IEEE Transactions on Image Processing. 2012, 21(4), 1500-1512 [cit. 2023-02-05]. ISSN 1057-7149. doi:10.1109/TIP.2011.2175935

[18] REGAL, Georg, Raimund SCHATZ, Johann SCHRAMMEL a Stefan SUETTE. VRate: A Unity3D Asset for integrating Subjective Assessment Questionnaires in Virtual Environments. 2018 Tenth International Conference on Quality of Multimedia Experience (QoMEX). [Online]. IEEE, 2018, 2018, 1-3 [cit. 2023-03-09]. ISBN 978-1-5386-2605-4. doi:10.1109/QoMEX.2018.8463296

[19] UPENIK, Evgeniy, Martin RERABEK a Touradj EBRAHIMI. Testbed for subjective evaluation of omnidirectional visual content. 2016 Picture Coding Symposium (PCS). [Online]. IEEE, 2016, 2016, 1-5 [cit. 2023-03-09]. ISBN 978-1-5090-5966-9. doi:10.1109/PCS.2016.7906378

[20] YU, Matt, Haricharan LAKSHMAN a Bernd GIROD. A Framework to Evaluate Omnidirectional Video Coding Schemes. 2015 IEEE International Symposium on Mixed and Augmented Reality [Online]. IEEE, 2015, 2015, 31-36 [cit. 2023-03-09]. ISBN 978-1-4673-7660-0. doi:10.1109/ISMAR.2015.12

[21] SCHATZ, Raimund, Georg REGAL, Stephanie SCHWARZ, Stefan SUETTC a Marina KEMPF. Assessing the QoE Impact of 3D Rendering Style in the Context of VR-based Training. 2018 Tenth International Conference on Quality of Multimedia Experience (QoMEX). [Online]. IEEE, 2018, 2018, 1-6 [cit. 2023-03-09]. ISBN 978-1-5386-2605-4. doi:10.1109/QoMEX.2018.8463411

[22] GitHub, "FSIM-FSIMc-matlab", accessed May 21, 2023. [Online]. Available: https://github.com/sunxirui310/FSIM-FSIMc-matlab/blob/master/FSIM.m

[23] Reichl, J. (n.d.). "Zorné pole", accessed May 21, 2023. [Online]. Available: http://fyzika.jreichl.com/main.article/view/488-zorne-pole?fbclid=IwAR17IfIvXeG78OSoTiePtGbTgPvkk4xNKIenrMepUefktPSS2skb8DQVw58

[24] KENNEDY, Robert S., Norman E. LANE, Kevin S. BERBAUM a Michael G. LILIENTHAL. Simulator Sickness Questionnaire: An Enhanced Method for Quantifying Simulator Sickness. The International Journal of Aviation Psychology [Online]. 1993, 3(3), 203-220 [cit. 2023-05-25]. ISSN 1050-8414. doi:10.1207/s15327108ijap0303_3

[25] Statistics By Jim, "Mean Squared Error (MSE)", accessed May 22, 2023. [Online]. Available: https://statisticsbyjim.com/regression/mean-squared-error-mse/

[26] SIMKA, Marek, Ladislav POLAK, Jan KUFA, Martin NOVOTNY, Adam ZIZIEN a Karel FLIEGEL. 2023 33rd International Conference Radioelektronika (RADIOELEKTRONIKA) [Online]. IEEE, 2023 [cit. 2023-05-24]. ISBN 979-8-3503-9834-2. doi:10.1109/RADIOELEKTRONIKA57919.2023.10109005

[27] ZHANG, Wei, Wenjie ZOU, Fuzheng YANG, Lucie LEVEQUE a Hantao LIU. The Effect of Spatio-temporal Inconsistency on the Subjective Quality Evaluation of Omnidirectional Videos. ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) [Online]. IEEE, 2019, 2019, 4055-4059 [cit. 2023-05-24]. ISBN 978-1-4799-8131-1. doi:10.1109/ICASSP.2019.8682221

[28] JIANXIONG XIAO, K. A. EHINGER, A. OLIVA a A. TORRALBA. Recognizing scene viewpoint using panoramic place representation. 2012 IEEE Conference on Computer Vision and Pattern Recognition [Online]. IEEE, 2012, 2012, 2695-2702 [cit. 2023-05-24]. ISBN 978-1-4673-1228-8. doi:10.1109/CVPR.2012.6247991

[29] Vicuesof, "SSIM/MS-SSIM", accessed May 25, 2023. [Online]. Available: https://vicuesoft.com/glossary/term/ssim-ms-ssim/

Appendix A

List of electronic attachments

The electronic attachments are following

- avgData.m - averages subjective scores – creates MOS scores
- bitrates.m - assigns bit per pixel value to quantization parameter
- eyeLoad.m - loads eye-tracking data
- FSIM.m - FSIM metric, downloaded
- graphSave.m - saves created data for 'Graphs' tab of GUI to load
- GUI.mlapp - graphical user interface
- imageLoad.m - loads image in appropriate folders
- loadTable.m - creates a list of possible load options for 'Load data' tab of GUI
- main.m - main function from which all others are called
- metrics_sort.m - calculates some metrics and calls other function to do so
- outliers.m - removes subjects if their scores are too different from others
- sceneSort.m - creates tables with data with scene and compression data
- sceneTable.m - creates tables with information of all possible scenes and compression
- SortAvgData.m - loads reference images and eye-tracking data for 'Eye map' tab of GUI
- ssim_compare.m - calculates SSIM an MS-SSIM
- textFormat.m - formats scene names into usable form
- txtLoad.m - loads .txt file with subjective scores

- txtSort.m - handles .txt loading by combining txtLoad, eyeLoad and textFormat
- weightMap.m - creates a weightmap for SSIM and MS-SSIM
- Exported_data/ - a folder for containing data saved in 'Graph' tab of GUI
- Eye_movements/ - a folder for containing measured eye-tracking data
- MOS/ - a folder for containing measured subjective scores
- Saved_data/- a folder for containing temporarily saved data for 'Graphs' tab of GUI to show
- Scenes/ - a folder for containing all images used in testing
- Scenesref/ - a folder for containing refernce image (best quality) from testing
- Scenes_simp/ - a folder for containing small size images from testing to be used for displaying graphs in 'Eye maps' tab