Predicting the Performance of Virtual Reality Video Streaming in Mobile Networks

Roberto Irajá Tavares da Costa Filho
Federal University of RS - UFRGS
Porto Alegre, Brazil
roberto.costa@inf.ufrgs.br

Maria Torres Vega
Ghent University - imec
Ghent, Belgium
maria.torresvega@ugent.be

Marcelo Caggiani Luizelli
Federal University of Pampa
Alegrete, Brazil
marceloluizelli@unipampa.edu.br

Jeroen van der Hooft
Ghent University - imec
Ghent, Belgium
jeroen.vanderhooft@ugent.be

Stefano Petrangeli
Ghent University - imec
Ghent, Belgium
stefano.petrangeli@ugent.be

Tim Wauters
Ghent University - imec
Ghent, Belgium
tim.wauters@ugent.be

Filip De Turck
Ghent University - imec
Ghent, Belgium
filip.deturck@ugent.be

Luciano Paschoal Gaspary
Federal University of RS - UFRGS
Porto Alegre, Brazil
paschoal@inf.ufrgs.br

ABSTRACT

The demand of Virtual Reality (VR) video streaming to mobile devices is booming, as VR becomes accessible to the general public. However, the variability of conditions of mobile networks affects the perception of this type of high-bandwidth-demanding services in unexpected ways. In this situation, there is a need for novel performance assessment models fit to the new VR applications. In this paper, we present PERCEIVE, a two-stage method for predicting the perceived quality of adaptive VR videos when streamed through mobile networks. By means of machine learning techniques, our approach is able to first predict adaptive VR video playout performance, using network Quality of Service (QoS) indicators as predictors. In a second stage, it employs the predicted VR video playout performance metrics to model and estimate end-user perceived quality. The evaluation of PERCEIVE has been performed considering a real-world environment, in which VR videos are streamed while subjected to LTE/4G network condition. The accuracy of PERCEIVE has been assessed by means of the residual error between predicted and measured values. Our approach predicts the different performance metrics of the VR playout with an average prediction error lower than 3.7% and estimates the perceived quality with a prediction error lower than 4% for over 90% of all the tested cases. Moreover, it allows us to pinpoint the QoS conditions that affect adaptive VR streaming services the most.

MMSys'18, June 12–15, 2018, Amsterdam, Netherlands

ACM ISBN 978-1-4503-5192-8/18/06…$15.00
https://doi.org/10.1145/3204949.3204966

CCS CONCEPTS

• **Information systems** → **Multimedia streaming**; • **Human-centered computing** → **Virtual reality**; • **Networks** → *Network protocols*; *Public Internet*;

KEYWORDS

Virtual Reality, HTTP Adaptive Streaming, Quality of Service, Quality of Experience, Mobile Networks

ACM Reference Format:
Roberto Irajá Tavares da Costa Filho, Maria Torres Vega, Marcelo Caggiani Luizelli, Jeroen van der Hooft, Stefano Petrangeli, Tim Wauters, Filip De Turck, and Luciano Paschoal Gaspary. 2018. Predicting the Performance of Virtual Reality Video Streaming in Mobile Networks. In *MMSys'18: 9th ACM Multimedia Systems Conference, June 12–15, 2018, Amsterdam, Netherlands.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3204949.3204966

1 INTRODUCTION

The number and variety of Internet-based video applications do not cease to increase. In fact, IP video traffic is envisioned to experience a 9-fold growth between 2016 and 2021, accounting for 78% of the total mobile traffic by 2021 [5]. According to the same source, the traffic generated by Virtual Reality (VR) is expected to increase 11-fold by 2021 [5]. One key enabler for supporting such a consistent growth is the diffusion of Head Mounted Devices (HMD). HMDs are presenting high penetration rates as they (*i*) are becoming effective and affordable (*e.g.*, Google's Cardboard[1]), (*ii*) are already provided at no cost with certain smartphones (such as Samsung Gear VR[2]) and (*iii*) are being pushed by industry (*e.g.*, Facebook recently announced a permanent price drop in Oculus Go headset with the goal of reaching 1 billion VR users [3]).

In order to provide an immersive user experience, VR videos demand significantly higher bandwidths when compared to traditional video applications. These ultra-high bandwidths are not

[1]Google Cardboard https://goo.gl/DSquZf
[2]Gear VR https://goo.gl/7JdQm7

always available in wireless networks and are not easy to process by lightweight mobile devices. In fact, currently, the streaming of VR videos through mobile networks is far from optimal. A VR video contains a full 360° panoramic view, regardless of the fact that only a fraction of it, namely the viewport[3], is visible at any given instant. In an attempt to optimize bandwidth usage, a recent research path has proposed viewport-aware schemes for VR video streaming, based on the HAS (HTTP Adaptive Streaming) paradigm [10, 27]. HAS approach is focused on encoding the source content at multiple quality representations (bitrates), while each quality representation is time-segmented into small parts (*i.e.*, segments). To further optimize bandwidth usage, recent research investigations have proposed HAS variants in which quality representations are not only segmented in time but also spatially split into smaller pieces (*i.e.*, tiles) [6].

Bringing the 2D adaptive streaming techniques to the VR arena requires the VR videos to be encoded at different quality levels, temporally divided in segments and spatially tiled [19]. Then, during the streaming session, only the tiles within the viewport are streamed in high quality, while others are maintained at the lowest levels or not streamed at all [24]. To be effective, these techniques rely on viewport prediction algorithms, since the player needs to fill in a playout buffer with tiles that are expected to compose the viewport in the near future [23].

Although the use of viewport-aware techniques leads to the reduction of bandwidth consumption, the effects of network performance on VR video streaming still plays an important role on the user's perception of the services. Since the full panoramic view of a VR video usually demands a much higher bitrate, when compared to regular videos [7], even a fraction of it (viewport) may require high bitrates. Along these lines, recent investigations have emphasized the importance of the network effects on the perceived quality (Quality of Experience, QoE) of adaptive video streaming applications [8, 10, 13, 14, 16]. However, state-of-the-art approaches fall short in predicting the perceived quality for VR videos as they do not consider the spatial segmentation.

QoE has shown to be a critical factor for video applications [1, 18]. As such, both network operators and VR content providers are required to answer an important question: *considering the wide range of performance levels of IP networks, to which extent are the currently observable network conditions able to provide users of VR applications with adequate QoE?* Answering this question is remarkably complex due to two constrains. First, the influence of the network on VR video performance is unknown; and second, the state-of-the-art on video QoE estimation modeling does not consider the VR context.

In this paper, we present PERCEIVE (PERformanCe EstImation for VR vidEos), a method that aims to provide answer to both aspects. PERCEIVE is a two-stage adaptive VR performance assessment model that employs machine learning algorithms to first predict VR video playout performance, using network QoS indicators as predictors. Then, it uses the video playout performance metrics to model and estimate the end-user perceived quality. Evaluated in real-world 4G/LTE network conditions, PERCEIVE not only accurately predicts the VR videos performance over networks, but also allows us to pinpoint the QoS conditions that affect VR streaming services the most.

The remainder of this paper is organized as follows. In Section 2, we discuss the related work. In Section 3, we introduce the approach used for the tile-based adaptive VR video streaming. In Section 4, we describe PERCEIVE, the two-step performance prediction scheme that we propose. In Section 5, we report the evaluation carried out to prove concept and technical feasibility of the proposed approach. It includes details on the evaluation methodology, the generation of the dataset, analysis of the training set and results. Finally, our conclusions and key findings are presented in Section 6.

2 RELATED WORK

In this section, we provide a thorough description of the state-of-the-art. In Section 2.1, we present a brief introduction to adaptive streaming applied to the VR context and review the state-of-the-art approaches. In Section 2.2, we highlight the most significant QoE estimation models in literature.

2.1 Adaptive Tile-based and Viewport-aware Video Streaming

An aspect to consider in adaptive tile-based VR streaming services is viewport prediction, which allows to considerably optimize bandwidth usage. Since a full VR video can easily reach 8K video resolution [7], most video players rely on heuristic algorithms to predict near-future user's head movements. By considering next position prediction, the video player is able to request only tiles that are likely to be inside the viewport, which leads to reduced bandwidth utilization. To provide this prediction, heuristic algorithms consider variables such as the angular velocity of the user's head, movement patterns for previous viewers, video content (*e.g.*, in a football match users will most likely follow the ball's movements), among other factors [2]. By performing such prediction, the video player can reduce in up to 72% in bandwidth utilization [12].

In practice, the viewport prediction algorithm is responsible for keeping a small video playout buffer (*e.g.*, 2 seconds) with the tiles that are more likely to belong to the viewport in the near-future. To illustrate how the viewport prediction interacts with the playout buffer, consider the example of a user watching a tile-based VR video using a head-mounted display. Consider a given temporal segment S_k and a respective viewport V_k, as depicted in Figure 1 (a). At this moment, the video player is requesting high-resolution chunks only for tiles inside the viewport V_k. Then, based on the near-future viewport prediction for the next segment (S_{k+1}), the video player starts requesting high-resolution tiles for the viewport V_{k+1} (delimited by the right dashed square in Figure 1 (b)). As predicted, the viewer slightly moves to the right (see Figure 1 (c)). At this point, driven by the viewport predictor, the VR player starts requesting high-resolution chunks for the predicted viewport on the segment S_{k+2} (Figure 1 (d)), and so forth.

Several recent investigations [11, 23, 24] have focused on a common main objective: devising bandwidth-efficient adaptive VR video players while keeping QoE at acceptable levels. Taking viewport prediction information as input, most approaches rely on per-tile rate adaptation algorithms to reduce the amount of information to

[3] Also referred to as Field of View (FoV)

be downloaded by keeping only the viewport's tiles in high resolution. Qian *et al.* [24] present a viewport prediction scheme that considers users' head movements (traces) and relies on Weighted Linear Regression to predict users' head position for the next second. The same study indicates that the estimation accuracy can drop from 90% to approximately 60% when increasing the prediction window to 2 seconds. Fan *et al.* [11] consider HMD sensor information and content-related features (*i.e.*, image saliency maps and motion maps) to train two neural networks for prediction of viewport position. In turn, Hosseini *et al.* [23] propose an efficient 3D geometry mesh to represent tiled VR video in the 3D space, which is capable of reducing bandwidth consumption in up to 30% when compared to non-tiled videos. The work also relies on MPEG-DASH Spatial Representation Descriptor (SRD) to describe the spatial relationship of tiles in the 360-degree space, prioritizing tiles inside the viewport to be downloaded in high quality. Petrangeli *et al.* [23] propose an approach with the ability to reduce bandwidth consumption in up to 35% by relying on both an HTTP/2-based push mechanism and a viewport prediction scheme based on viewport speed.

As discussed, viewport prediction is a sensitive task, which might affect the user's perception in unexpected ways. Errors on the prediction of the viewport (*i.e.* the FoV that the user will look at in the next segment) may lead to partial or full degradation of the perception, even if the network conditions are enough to guarantee the user's QoE. This means that, during the streaming, two different processes (namely the viewport prediction and the effects of the network on the adaptive streaming performance metrics) will have a major influence on the user's QoE. In this work, we are interested on predicting the effects of networks on VR adaptive streaming in an isolated manner, without the influence of errors derived from wrong viewport prediction. Thus, for the analysis presented herein, we have assumed perfect prediction, *i.e.*, the adaptive streaming player knows exactly where the user is looking at every point in time.

2.2 Adaptive Video Streaming QoE Estimation

One of the biggest challenges of adaptive video streaming (2D, 3D or VR) applications is the accurate and real-time estimation of quality perceived by the users. And, based on it, the provision of a feedback loop to dynamically influence the quality adaptation. In state-of-the-art adaptive streaming approaches, the modeling of QoE has to rely on objective information obtained at the client, the server or the network-side.

Mao *et al.* [16] present a model to estimate QoE considering both network and application performance indicators measured at client's video player (*e.g.*, average bitrate, video stall events and bitrate changes) as inputs. Similarly, Jiang *et al.* [13] propose a Content Delivery Network (CDN) node selection approach that employs a Critical Feature Analytics (CFA) design to provide accurate QoE estimation. In this work, the authors also rely on information provided by client video players as input. Conversely, Xianshu *et al.* [14] propose a network path selection scheme that considers the bitrate measured at the server-side to produce simplified QoE estimation for adaptive video streaming applications.

Dimopoulos *et al.* [10] introduce a methodology for estimating QoE based on the analysis of encrypted HTTPS video streaming traffic observed in the network. Da Costa Filho *et al.* [8] propose an approach for QoE estimation based on network QoS indicators obtained through active measurements. Both investigations [8, 10] demonstrate it is possible to estimate video streaming QoE based on network-side information. Although these approaches may not be as accurate as the ones based on client- or server-side measurements, they have shown to result in a satisfactory level of accuracy with a crucial advantage: in addition to end-to-end QoE estimation, they allow for fast identification and isolation of network segments responsible for QoE impairments. In spite of the recent fundamental contributions for the video streaming evolution, the work on QoE modeling for adaptive video streaming has basically focused on 2D videos. Unlike 2D video content, VR presents significantly more complex elements to consider (*e.g.*, spatially segmentation, viewport prediction, and per-tile rate adaptation). Thus, the current QoE models are not suitable to estimate QoE for VR videos.

3 ADAPTIVE STREAMING OF VR VIDEOS USING TILES AND QUALITY ZONES

This section introduces the adaptive VR streaming approach for which PERCEIVE is envisioned. In order to reduce the bandwidth required for the streaming, it adopts a tiling structure, in which the videos are not only divided in temporal segments but are also spatially split in sections (tiles) [23]. In addition, tiles are grouped in quality zones prior to the streaming. Each of the zones is assigned a quality level according to the network conditions measured during the previous segment. In the next two subsections, both the structure and the adaptive streaming technique adopted in this work are presented.

3.1 Adaptive VR streaming structure: Spatial Tiles and Quality Zones

A VR video V can be represented by a set of k spatially divided zones $Z = \{Z_1, ..., Z_k\}$ such that $\bigcap_{\forall k} Z_k = \emptyset$. The same video V is temporally split into a discrete number of m segments $S = \{S_1, ..., S_m\}$ such that $\bigcup_{\forall m} S_m = V$. Each zone Z_k is composed of a set of tiles $t \in Z_k$. A tile t is time-divided into m chunks $C = \{C_{t_1}, ..., C_{t_m}\}$, and may assume different bitrates (qualities) $R(C_{t_m})$ over time. Finally, we refer to a segment as the set of all chunks for a given time frame such as $S_m = \bigcup_{\forall t} C_{t_m}$. In tile-based approaches, the encoding process defines how the video will be spatially divided (*i.e.*, tiling scheme), which bitrates will be available in the HAS context (*i.e.*, quality representations), and the segment length (*i.e.*, number of seconds).

An example of this type of structure is shown in Figure 2. There are three quality zones $Z = \{Z_1, Z_2, Z_3\}$, each one composed by a set of adjacent tiles. Z_1 is a set of tiles adjacent to the viewport center ($t_{28}, t_{29}, t_{36}, t_{37}$), Z_2 is the border of the viewport ($t_{43}, t_{44}, t_{45}, t_{46}, t_{38}, t_{30}, t_{22}, t_{21}, t_{20}, t_{19}, t_{27}, t_{35}$) and Z_3 is composed by all tiles outside the viewport.

3.2 Adaptive streaming heuristic

Algorithm 1 describes the adaptive streaming heuristic procedure adopted for this paper (adapted from [23]). The bitrate in a specific zone Z_k is named as $R(C_t)|^{Z_k}$. The algorithm receives as input *(i)*

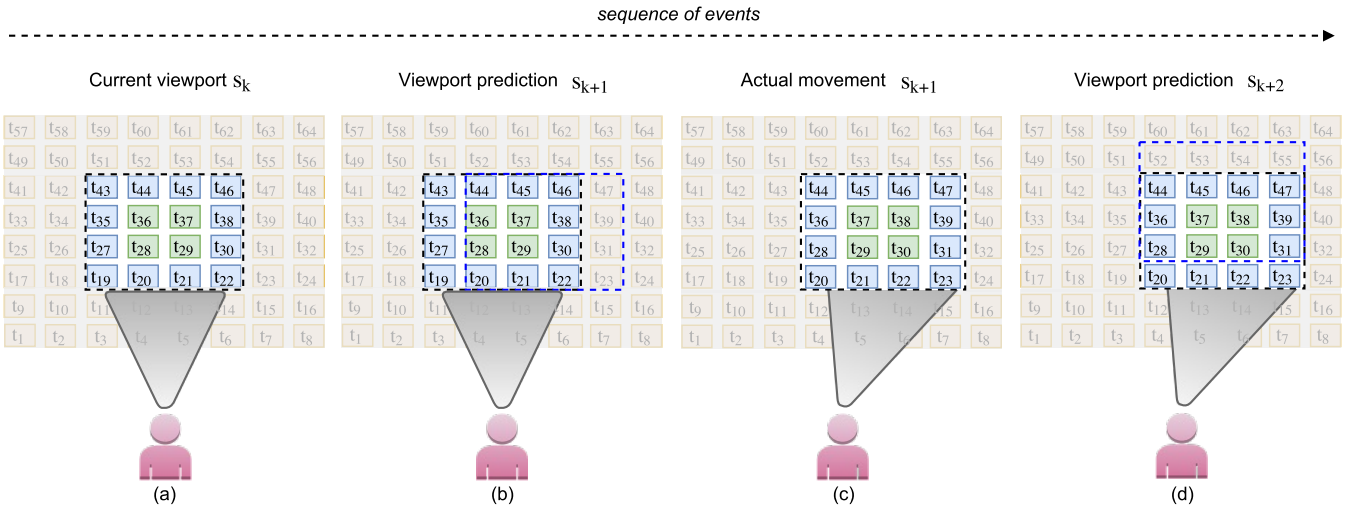


Figure 1: Example of the working principles of the viewport prediction.

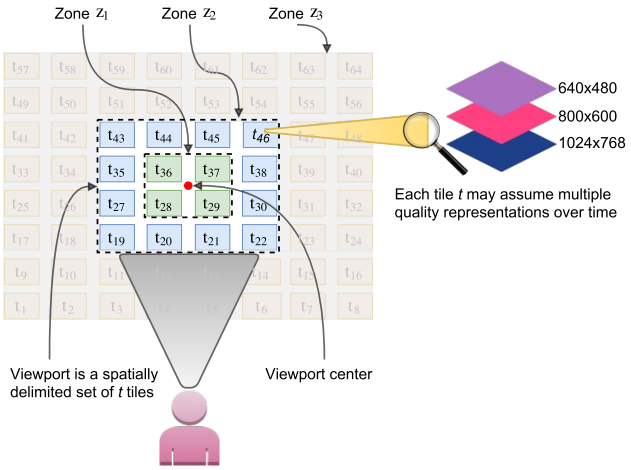


Figure 2: Example of an adaptive tile-based VR video structure split in 3 quality zones.

a reference to a VR video V, *(ii)* a set S describing the video segmentation and *(iii)* the available zones in video V. The heuristic described in Algorithm 1 works as follows. Once knowing the available bandwidth in the network, the VR player downloads tiles with the highest possible bitrates. First, the heuristic tries to increase the bitrates on the zones inside the viewport. Then, it repeats the procedure to stream tiles from the outer zones (observed in line 3). Observe that the heuristic does not increase the bitrate $R(C_t)|^{Z_{k+1}}$ on a subsequent zone Z_{k+1} in case the bitrate of zone Z_k is strictly upper-bounded by $R(C_t)|^{Z_{k+1}}$. In other words, it ensures that the bitrate of zone Z_{k+1} is always lower or equal to that of zone Z_k. Furthermore, it ensures that tiles within the same zone Z_k are streamed with the same bitrate $R(C_t)$, that is, $R(C_0)|^{Z_k} = R(C_1)|^{Z_k} = \cdots = R(C_t)|^{Z_k}$. If the available bandwidth were insufficient to download all the tiles in a zone on time (before display), the streaming would stall until the buffers were filled. Hence, the player ensures that all tiles are synchronized during the playout and that no black tiles appear. For more information on the working principles of the streaming heuristic, please refer to [23].

Algorithm 1 VR player heuristic adapted from [23].

Input: V: VR video
Input: S: discrete number of segments in VR video V
Input: $Z = \{Z_1, ..., Z_k\}$: k spatially divided zones in VR video V
1: **for** each video segment $S_i \in S$ from VR video V **do**
2: **for** each zone $Z_k \in Z$ **do**
3: *gather* tiles $t \in Z_k$ with maximum available bitrate $R(C_t)|^{Z_k}$, such that $(\forall k \geq 2) : R(C_t)|^{Z_k} \leq R(C_t)|^{Z_{k-1}}$ and $(\forall t \in Z_k) : R(C_0)|^{Z_k} = R(C_1)|^{Z_k} = \cdots = R(C_t)|^{Z_k}$

4 PERCEIVE: ADAPTIVE VR VIDEO PERFORMANCE PREDICTION

Figure 3 presents the block diagram of the proposed two-stage VR performance prediction method. The first stage is composed of four predictors, one per VR video application performance metric (*i.e.*, startup delay, quality, quality switches count and video stalls) [32]. As input, the predictors consider both the network Quality of Service (QoS) (*i.e.*, delay, packet loss and TCP throughput) and the tiling scheme. In the second stage, the user QoE is estimated by submitting the predicted application layer performance metrics to the proposed QoE model. PERCEIVE can dissect VR video playout performance by understanding two key processes, namely *(i)* the influence of the network performance on VR video player outputs; and *(ii)* how the user perceives the resulting video playout performance. However, both the VR video player and the QoE model are open questions in the sense that there is neither a reference player implementation nor a QoE model for VR videos. Considering the above, the proposed two-stage prediction allows both the playout

performance metrics predictors and the QoE model to be individually updated, without the need to rebuild the entire scheme. The following two subsections provide details and insights on each of the stages of which PERCEIVE is composed.

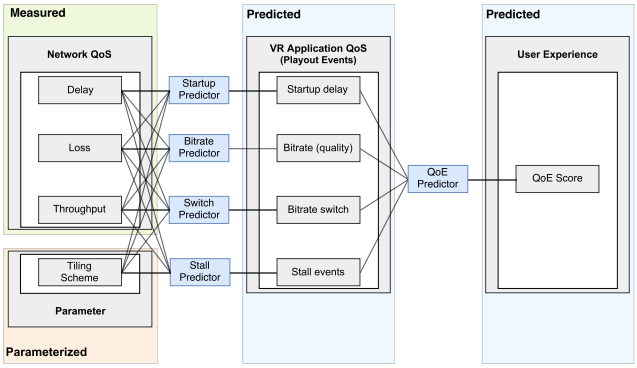


Figure 3: PERCEIVE two-stage quality prediction

4.1 Adaptive VR Video Playout Prediction

In the first stage of the method, the four most important playout performance metrics associated with the adaptive VR streaming, namely startup delay, quality level (bitrate), quality switches and stall time [32], are predicted based on network QoS inputs and the VR video structure. Each of them is independently predicted using regression trees as predictors (taking advantage of and adapting the 2D procedure proposed in previous research work [8]). Regression trees are employed due to three main reasons. First, they have shown to be an accurate machine learning method in related investigations [8, 26]. Second, they permit understanding complex and non-linear relationships between predictors and response variables in an intuitive and visual manner. This is a very important feature that allows to pinpoint the most influential inputs, which will be put to the test in the analysis of Section 5.3. Finally, once the prediction structures are generated, they can predict the response variable in linear time complexity, and can be easily integrated in third-party applications, which are fundamental aspects for network operators and content providers.

The selection of the input QoS parameters has been made based on state-of-the-art research studies on the QoS conditions that affect video streaming services the most [8, 20, 28]. These studies also concluded that TCP throughput is one of the most influential QoS metric when prediction QoE. Also, both network losses and delays have been demonstrated to be responsible for further degradation, depending on the type of streaming application used. In addition to these three network performance metrics, a fourth parameter, namely the tiling structure of the VR streaming, was included. The structure defines the number of tiles that need to be streamed to the client, thus it will heavily influence the VR playout performance. Once the four VR playout performance indicators are predicted, they serve as input to the second phase, the QoE model as it is presented in the next Section.

4.2 Adaptive VR QoE Estimation Model

The purpose of this second stage is to estimate the quality perceived by the end users (their QoE), considering the VR video playout performance metrics obtained from the previous stage of PERCEIVE (*i.e.,* startup delay, quality level (bitrate), quality switches and stall time).

The model proposed herein considers the state-of-the-art on QoE modeling for adaptive streaming applications in general and HAS in particular [16, 22, 32]. To the best of our knowledge, this is the first model to consider the concept of zones and tiles in a QoE estimation model for VR videos. These characteristics are crucial, given the fact that they allow coping with VR video attributes while providing flexibility to handle different video encoding strategies (*e.g.*, tiling scheme, viewport geometry and available quality representations).

Given the concepts of quality zones and tiles of the approach used for Adaptive VR streaming (Section 3), the QoE function is defined per zone as a function of the VR playout characteristics predicted by the previous stage within that quality zone (Section 4.1). This strategy is aligned with the notion that the influence of VR playout characteristics on user perception is different depending on the zone where they are observed (*e.g.*, quality switches for tiles outside the viewport are less important than quality switches inside the viewport). Thus, the per-zone quality function ($\phi(Z_k)$) is defined as the weighted sum of the four playout characteristics (Equation 1).

$$\phi(Z_k) = \overbrace{\sum_{\forall t \in Z_k} \sum_{\forall c \in C(t)} q(R(C_{t_m}))}^{\text{Quality}} - \overbrace{\mu \cdot \sum_{\forall t \in Z_k} \sum_{\forall c \in C_{t_m}} \left(\frac{d_c(R_c)}{C_c} - B_c \right)_+}^{\text{Stalls}} - \underbrace{\lambda \cdot \sum_{\forall t \in Z_k} \sum_{\forall c \in C_{t_m}} \left| q(R(C_{t_m+1})) - q(R(C_{t_m})) \right|}_{\text{Quality switches}} - \underbrace{\omega \cdot T_s}_{\text{Startup}} \quad (1)$$

In Equation 1, $R(C_{t_m})$ R represents the bitrate (*i.e.,* quality) of a given chunk. Recall that a tile t is time-divided into m chunks $C = \{C_{t_1}, ..., C_{t_m}\}$, (adapted from [16, 32]). Function q is a mapping function that translates the bitrate of chunk C_{t_m} belonging to tile t to the quality perceived by the user (*i.e.*, in terms of bitrate sensitivity). The second term of the Equation is used to track stall time. Stalls can be characterized either by tile (*i.e.*, it is possible to have stall in some tiles and video playout in other, for the same segment) or by segment (*i.e.*, the video will stall until all the tiles for a given segment have enough buffer). To keep the model as general as possible, we consider for each chunk c, that a stall event occurs when the download time $\frac{d_c(R_c)}{C_c}$ is higher than the playout buffer length (B_c) when the chunk download started. Hence, the total stall time is given by $\sum_{c=1}^{C} \left(\frac{d_c(R_c)}{C_c} - B_c \right)_+$. In addition, $|q(R_{ct+1}) - q(R_{ct})|$ considers the quality switches between consecutive chunks and T_s tracks the startup delay. Finally, constants μ, λ, ω are the non-negative weights used to tune the model for different user importance regarding QoE events. For example, a higher value of μ, with respect to the other weights, means that the user is more

susceptive to video stalls. Consequently, these events should affect the QoE indicator more severely.

Each of the zones within the VR video influences the perception of the user in a different manner. For example, tiles within the first or second zones (*i.e.*, the closest to the FoV of the user) will greatly steer the quality perceived by the user, while bad qualities on tiles of the edge zones will potentially go unnoticed. For this reason, the overall video QoE ($\phi(V)$) is modelled as a weighted linear sum of the QoE measurement per zone (Equation 2). Each weight ($\alpha_1, \alpha_2, ..., \alpha_k$) allows defining the relative importance of each zone when composing the video QoE. For example, the zones belonging to the viewport should have higher weights compared to the other zones.

The values for each α_n parameter should be derived from subjective tests. For example, considering a two-zone QoE scheme, values for α_1 (viewport) should be close to one, and values for α_2 (outside viewport) should be close to zero. When the QoE model is configured with more than two zones, it is necessary to determine α_n (testing values within a certain range) for each zone. In this case, subjective tests should systematically include incremental quality degradation, specifically in the intermediate zones, in order to measure the user's sensitiveness regarding quality issues in each zone.

$$\phi(V) = \alpha_1 \cdot \phi(Z_1) + \alpha_2 \cdot \phi(Z_2) + ... + \alpha_k \cdot \phi(Z_k) \quad (2)$$

5 EVALUATION

In this section, the evaluation of the PERCEIVE method is discussed. We start by presenting the procedure followed to evaluate the method in Section 5.1. Next, in Section 5.2, we introduce the generation of the dataset used for training and testing. In Section 5.3 we discuss and analyze the resulting VR playout predictors. This analysis provides insights on the dependency and predictability of each of the VR playout performance metrics given the QoS and tiling structure inputs. Finally, in Section 5.4 we present the prediction evaluation results for each of the five outputs of PERCEIVE (*i.e.*, the four VR playout performance metrics and the perceived quality).

5.1 Evaluation Methodology

In order to evaluate the performance of PERCEIVE, the procedure outlined by Figure 4 is followed. First, the datasets for training and testing have to be generated. Therefore, a VR video player is required to measure the VR video application playout performance metrics (*i.e.*, startup delay, average bitrate (quality), bitrate switches and video stalls) while subjected to real-world inputs, such as a realistic wireless networks measurements, VR tile-based videos and users' head track traces.

Next, the resultant datasets are given as input to the machine learning algorithm (responsible for learning the influence of the network QoS parameters and tiling scheme onto the VR playout characteristics). After the training phase, the resulting predictors can estimate the application layer performance only by means of the network parameters, and the considered tiling scheme. Finally, based on the VR playout performance metrics, the QoE can be

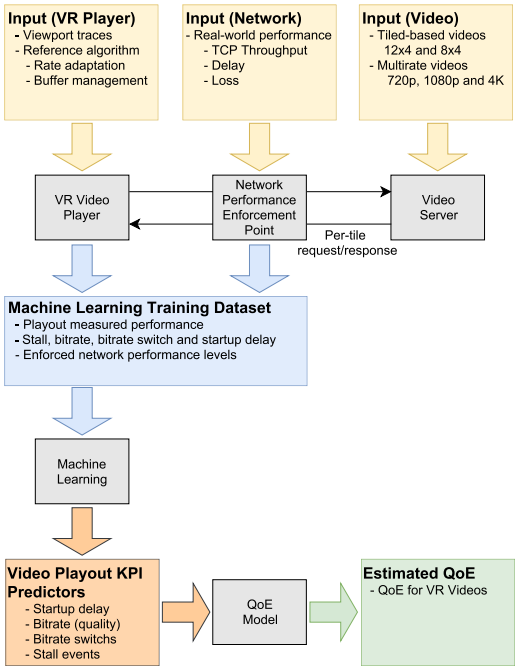


Figure 4: General evaluation methodology for PERCEIVE

estimated. The performance of PERCEIVE is assessed by means of the calculation of the normalized residual errors between predicted and measured values (r_i, Equation 3). In the equation, x is the ground truth, $\hat{x}$ is the prediction and N is the normalization factor (in this case the video duration).

$$r_i = |\hat{x}_i - x_i|/N \quad (3)$$

5.2 VR Dataset Generation

Each sample in the dataset contains the VR video tiling information, the three network QoS features and the respective video performance measured by the VR video player. To construct such dataset, the procedure presented in Section 5.1 is followed. Experiments are set, considering that a VR video player requests and processes tile-based VR videos from a web server (Apache 2 2.4.18-2). The network conditions are enforced by the Linux Traffic Control (TC) mechanism according to real-world network performance inputs. The experiments are built on top of a Linux Ubuntu 14.04 operating system, running on bare metal servers, where each server consists of a quad-core E3-1220v3 (3.1GHz) processor, with 16GB of RAM and two 10-gigabit network interfaces. Considering this infrastructure setup, we performed 1,524 video execution rounds, which resulted in more than 5,240 minutes of VR video playout.

Table 1 summarizes the input parameters values considered in the experiments. As network throughput input, the 4G/LTE measurements dataset of van der Hooft *et al.* [29] was selected. This dataset presents TCP throughput ranging from 0 Kb/s to 95 Mb/s as shown in Figure 5. For network packet loss, values between 0% and 13% were selected, in line with [8]. The network delay range was set from 1 to 130 ms. These values allowed us to assess the

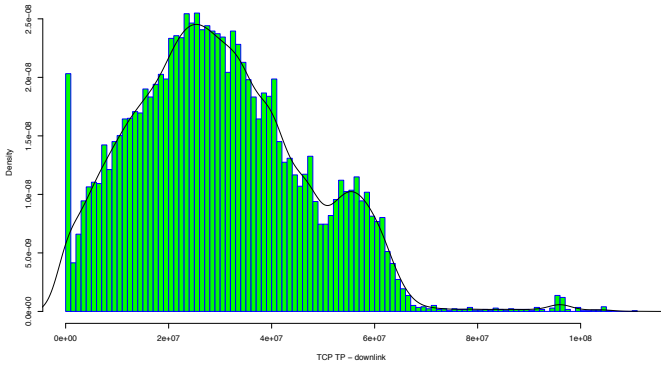


Figure 5: TCP throughput histogram of the 4G/LTE dataset of [29]

application performance from a very degraded delay performance (130 ms) down to the expected 5G delay (1 ms) [9].

Table 1: Input parameter configurations.

Metric	Short	Unit	Range
TCP throughput	TCPTP	Mb/s	0-95Mb/s ([29], Figure 5)
Packet Loss	PLR	%	0 – 13% (based on [8])
Delay	Delay	millisecond	1-130ms (based on [8, 9])
Tiling scheme	Tile	categorical	8×4 or 12×4 (based on [15, 24]

Two VR videos from Wu et al.' s dataset [31] (namely "Google Spotlight-HELP" and "Freestyle Skiing") were used for the streaming under the above described network conditions. For each video, we considered the available datasets regarding the users' head movements while watching it. As the original videos are not tile-based, they had to be re-encoded. After extracting the raw YUV files, making use of the Kvazaar encoder [30], the videos were re-encoded in a HEVC tile-based version, considering two tiling schemes: 8×4 and 12×4 [15, 24]. In addition, each tiling scheme was encoded to three quality representations, namely 720p (1.8Mb/s), 1080p (2.7Mb/s) and 4K (6Mb/s). Next, we used the MP4Box[4] application to pack the re-encoded videos into MP4 containers. Finally, we defined the segment duration of 1 second and used MP4Box to extract per-tile files and to generate the MPEG Dash Media Presentation Description (MPD) files considering multiple quality representations (Table 2). For the streaming heuristic (Section 3.2), there are three defined zones, where Zone 1 is the viewport center tile, Zone 2 groups the 8 tiles surrounding Zone 1, and all other tiles belong to Zone 3. Figure 6 shows the zone division for the 12×4 tiling scheme.

Table 2: Adaptive streaming configurations.

Videos	Qualities (bitrates)	Quality zones	Segment	Tiling
Google Spotlight Freestyle Skiing (Wu et al. [31])	720p - 1.8Mb/s 1080p - 2.7Mb/s 4K - 6Mb/s	Zone 1: 1 tile (central FoV) Zone 2: 8 tiles (adj. Zone 1) Zone 3: Rest	1 s	12×8 8×4

[4]MP4Box https://gpac.wp.imt.fr/mp4box/

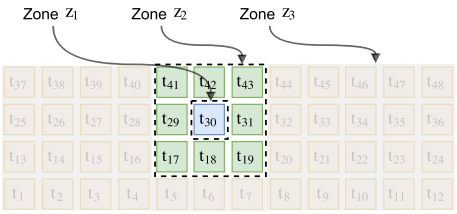


Figure 6: Viewport detail for the 12x4 tiling scheme

5.3 Resulting Predictors: VR Playout vs Network Conditions

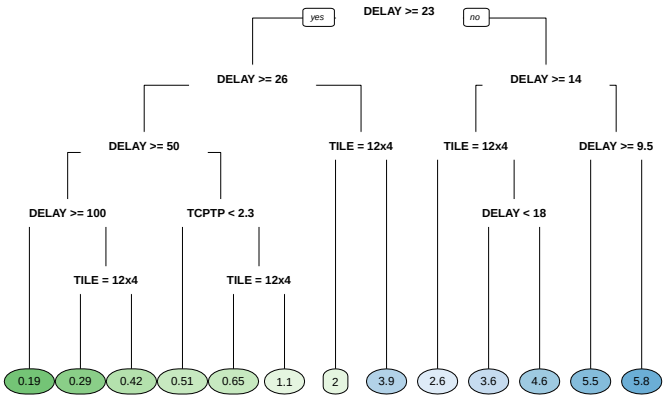
Based on the dataset, the regression trees were trained using a 10-fold cross-validation approach [15]. As each zone has independent quality behavior, both the quality and quality switches need to be learned per-zone. On the other hand, the startup and stall times are independent from the quality zone under scrutiny. Hence, they can be learned per video segment. Given the fact that there are three quality zones, eight regression trees were trained: three for Quality, three for Quality Switches, one for Stall time and one for Start time. All trees are optimally pruned [17], which means pruning until the cross-validation error is minimal and overfitting is avoided.

Before assessing the performance of the two-stage method, a thorough analysis of the regression trees was performed. This analysis aims at characterizing the relationship between the input parameters (network conditions and VR video structure) and the VR playout, allowing one to pinpoint to the most influential inputs.

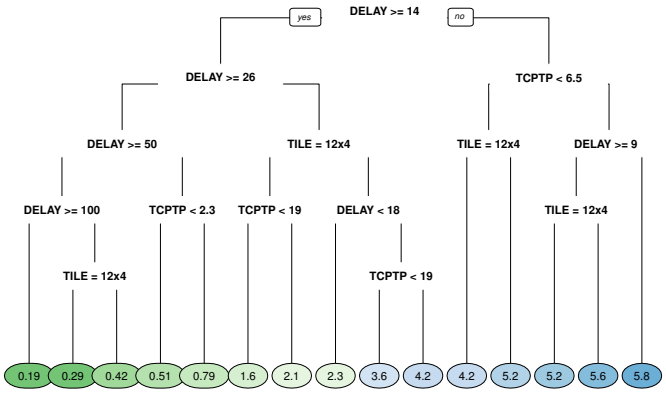
Figures 7 to 9 present the outcome predictors derived from the regression trees. All presented trees share two structural characteristics. First, although inversions may occur, usually the leftmost leaf node holds the lowest value for the predicted variable, and the value increases while moving towards the rightmost leaf node. Second, the closer to the root node, the more important the prediction feature (*i.e.*, delay, TCP throughput, loss and tile scheme).

Having a first look at the content of the trees, two observations can be made. First, network packet losses are not included in any of the trees. This means that the level of packet losses does not have influence on the VR playout performance metrics. Its effect will only be important as they affect the TCP throughput (higher network packet losses = lower TCP throughput). Furthermore, network delays turn out to be the most influential parameter on the VR playout.

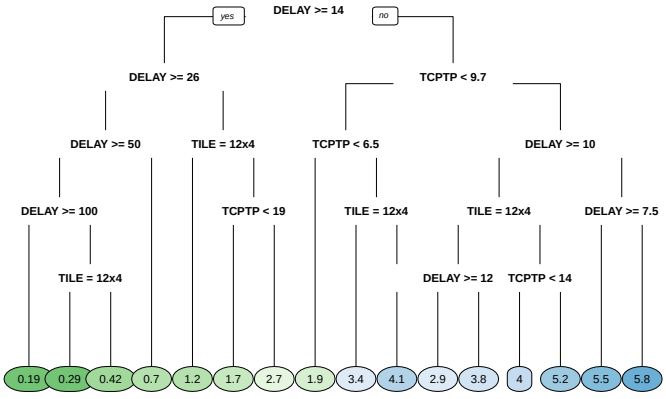
Regarding quality (by means of the average bitrate) (Figures 7(a) to 7(c)), let us consider the following aspects. The first decision taken in Zone 1, at the root node and, therefore, the most influential, is to understand if the network delay is greater than 23 ms (Figure 7(a)). The left branch ($Delay \geq 23ms$) is related to predicted quality not higher than 3.9 Mb/s, regardless of any other input value. In other words, even considering that the evaluated LTE network presents TCP throughput of up to 95 Mb/s, it is not enough to achieve the maximum bitrate (6.0 Mb/s - 4K), if the delay is higher than 23 ms. The reasoning behind this behavior is that each video segment (1 s) demands the download of 32 (8x4 tiling) or 48 (12x4) tiles. Despite the reuse of the TCP connection avoids the TCP slow-start restart [4]), the request/response overhead limits the throughput.



(a) Quality (average bitrate) - Zone 1 (Mb/s)

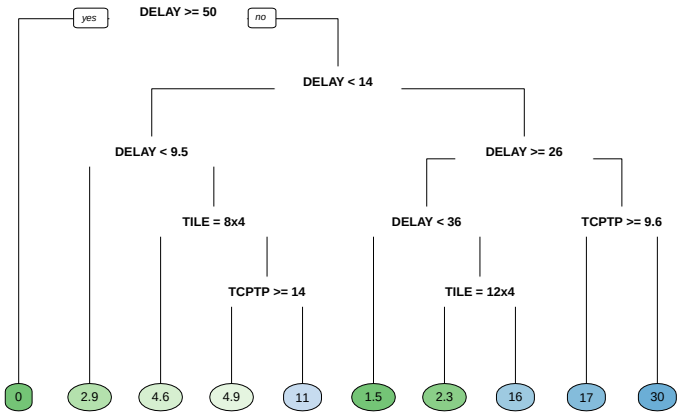


(b) Quality (average bitrate) - Zone 2 (Mb/s)

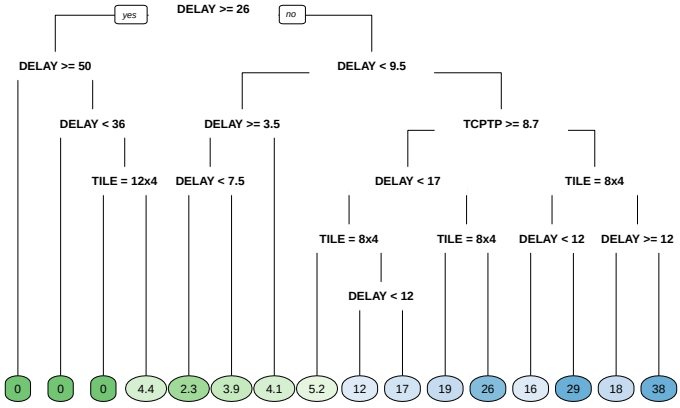


(c) Quality (average bitrate) - Zone 3 (Mb/s)

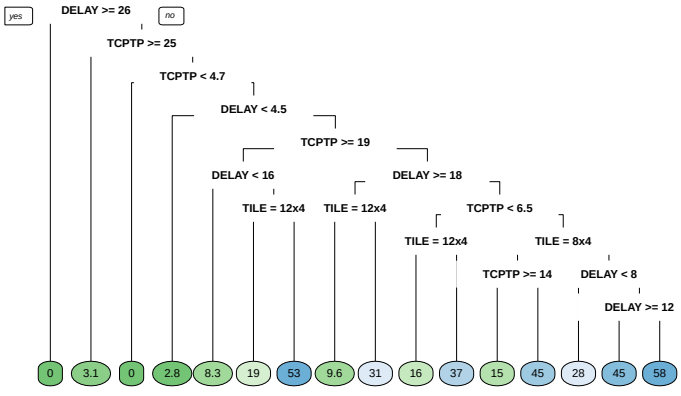
Figure 7: Regression tree representation for the predictors of the VR playout performance metric: quality. Leaf node colors go from dark green (for the lowest value for the predicted variable) to dark blue (the highest predicted value).



(a) Quality switch - Zone 1



(b) Quality switch - Zone 2



(c) Quality switch - Zone 3

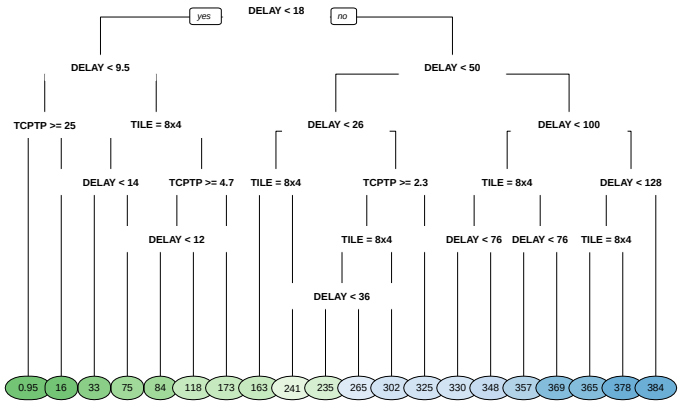
Figure 8: Regression tree representation for the predictors of the VR playout performance metrics: quality switches. Leaf node colors go from dark green (for the lowest value for the predicted variable) to dark blue (the highest predicted value).

For Zones 2 and 3 (Figures 7(b) and 7(c)), the quality predictors follow a very similar trend. However, in order to achieve the same level of average quality, they demand higher network performance than Zone 1. The right-most leaf of Zones 2 and 3 are a clear example of this behavior. To achieve the same quality (average bitrate of 5.8 Mb/s), Zone 2 requires a delay lower than 9 ms, and Zone 3 lower than 7.5 ms. Also, the values of TCP throughput to achieve intermediate average bitrates are higher for Zones 2 and 3 when compared with Zone 1. The main reason for such behavior comes from the rate adaptation heuristic, which prioritizes high bitrates for the tiles that are closest to the viewport's center (Section 3.2). Thus, intermediate network performance may be enough to keep Zone 1 at the highest available bitrate, while high levels of network performance allow increasing the bitrate for all zones.

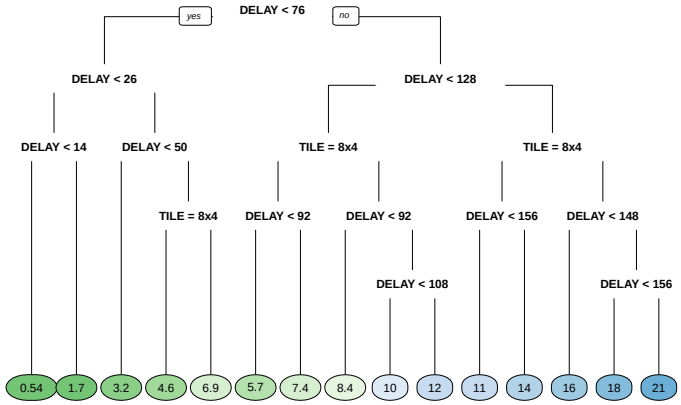
Quality switches (Figures 8(a) to 8(c)) provide valuable information in the context of HAS videos. For example, if no switches occur, the full video playout occurs in the lowest available resolution, meaning that the video player is unable to switch to higher bitrates, probably, due to insufficient network performance. In turn, when subject to excellent network performance conditions, most of HAS rate adaptation heuristics (including the one used in this paper) will stabilize at the highest available bitrate within a few switches. When considering real-world networks, if we have a look at the quality switches trained trees (Figures 8(a), 8(b) and 8(c)), it can be seen that the turning point from zero switches to maximizing the quality is a network delay of 50ms for Zone 1, and 26ms for Zones 2 and 3. However, by analyzing the rightmost leaf nodes of the decision trees for Zones 1, 2 and 3, one can observe that the maximum number of quality switches increases from Zone 1 towards Zone 3: (30, 38 and 58, respectively). This happens because, according to the considered heuristic for rate adaptation, the tiles inside Zone 3 will be the first ones to be switched to a lower resolution in case a network performance degradation is detected, followed by Zone 2 and, if the network performance degradation is severe, the Zone 1.

With respect to the cumulative stall time (Figure 9(a)), the resulting regression tree presents a wide range of predicted values (from 0.95 up to 384 seconds). One key aspect is related to the decision taken at the root node. As one can observe, if the delay is higher or equal to 18 ms, the minimum expected stall time is equal or higher than 163 seconds, independent of the tiling scheme or the available bandwidth (TCP throughput). Such high values would inflict a dramatic degradation on the perceived quality. In turn, for network delays lower than 9.5 s and TCP throughput equal or higher than 25 Mb/s, the expected stall time is minimal (0.95 seconds). It is worth mentioning that, even if the delay is lower than 9.5 s, if the TCP throughput is lower than 25 Mb/s, the expected stall time is 16 seconds. Also, in line with the aforementioned findings, the 12x4 tiling scheme leads to a significative higher amount of stall time for intermediate levels of network performance.

Finally, the regression tree for predicting startup delay is shown in Figure 9(b). In the considered VR video player, the startup delay is characterized as the elapsed time between the arrival of the request for the first tile and the completion of the buffer filling for all tiles for the first two segments. As the segment is relatively small, and considering the small file size of the tile chunks (on average 23 KB for 4K video resolution), the startup delay exclusively depends on



(a) Cumulative start time (seconds)



(b) Startup delay (seconds)

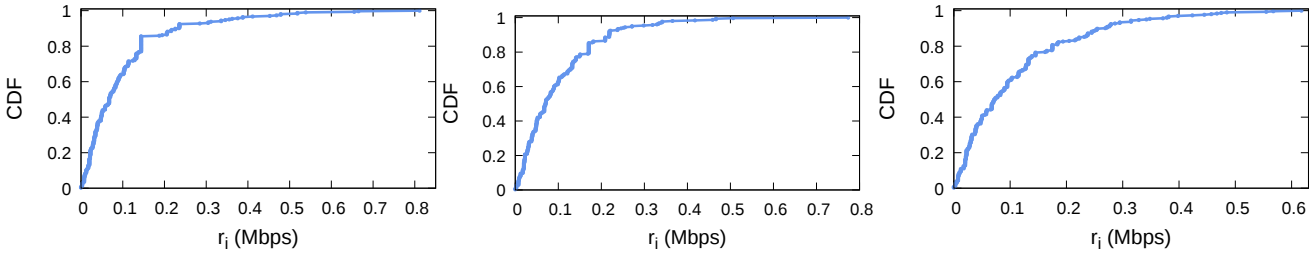
Figure 9: Regression tree representation for the predictors of the VR playout performance metrics: stall time and startup delay. Leaf node colors go from dark green (for the lowest value for the predicted variable) to dark blue (the highest predicted value).

the network delay. A delay lower than 26 ms is enough to provide an acceptable startup delay (smaller than 1.7 s). However, the best performance is achieved when the delay is lower than 14 ms (0.54 s).

5.4 PERCEIVE Results

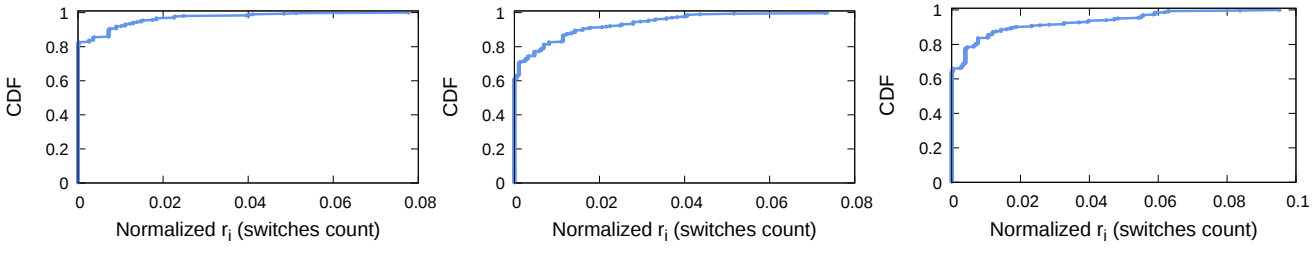
Aiming at determining the accuracy of the proposed predictors, the trained regression trees were used on unseen samples of the generated dataset, according to a 10-fold cross-validation scheme [15]. We considered as ground truth the performance measured by the reference VR video player when subjected to real-world network performance traces. In light of this, each test sample i contains the predictor variables (*i.e.*, TCP throughput, delay and tiling scheme), and the respective measured values for the performance metrics (*i.e.*, average bitrate, stall time, quality switch and startup delay).

Furthermore, based on the predicted VR playout characteristics, the QoE indexes were estimated by means of Equation 2. The parametric constants shown by the model were set to the values



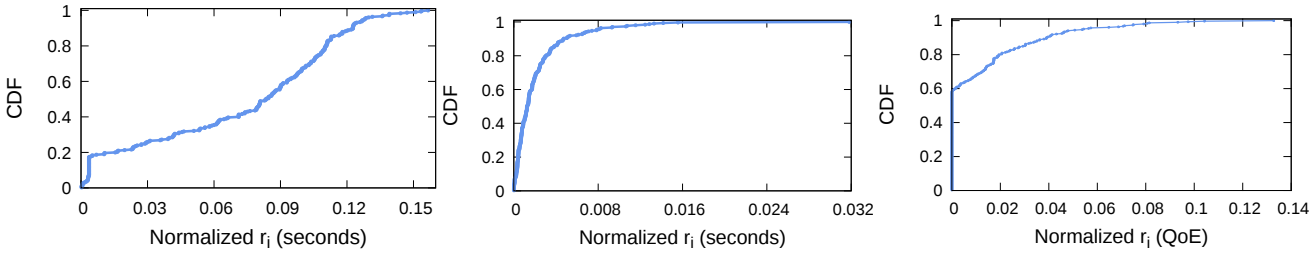
(a) Quality (average bitrate) - Zone 1 (b) Quality (average bitrate) - Zone 2 (c) Quality (average bitrate) - Zone 3

Figure 10: Residual error CDFs for quality (average bitrate)



(a) Quality switch - Zone 1 (b) Quality switch - Zone 2 (c) Quality switch - Zone 3

Figure 11: Residual error CDFs for the quality switch



(a) Cumulative Stall time (b) Startup delay (c) QoE

Figure 12: Residual error CDFs for stall time, startup delay and QoE estimation

presented in Table 3. Based on the results shown by Mao et al. [16], the q function was set to linear, where q is equal to the bitrate. In addition (also according to [16]), the stall and startup weights (μ and ω) were set to 4.3. The value of the quality switches constant (λ) was tuned to 1 [32]. Finally, the zones weights (α_1, α_2 and α_3) were empirically set to 0.7, 0.3 and 0, for Zone 1, Zone 2 and Zone 3. The reason behind setting α_3 to zero comes from the perfect prediction scenario considered in the evaluation. In such cases, the FoV will correspond 100% of tiles of Zones 1 and 2. Thus, there is no influence of the quality of Zone 3 on the user' s perception. In the case that perfect prediction would not be possible, the weights would need to be tuned accordingly.

The performance of the method is assessed by means of the residual error calculated between real data sample (entry in the training set) and the predicted one (as already introduced in Section 5.1 and Equation 3). With the purpose of generalizing the method for videos of arbitrary duration, the residual error for the metrics average bitrate, quality switch and startup delay are normalized by the factor N of the residual error equation, which corresponds to the

Table 3: Constants and function values assigned to the function to estimate QoE (refer to Equation 2)

Param.	Value
q	Linear
μ	4.3
ω	4.3
λ	1
α_1	0.7
α_2	0.3
α_3	0

considered video length (200 seconds). Figures 10, 11 and 12 show the Cumulative Distribution (CDF) of the residual error for the four VR playout performance metrics and the QoE estimation.

Looking at the quality prediction capacities of PERCEIVE (Figures 10(a) to 10(c)), it is possible to observe that the residual errors are very small (224 Kb/s and 220 Kb/s for Zones 1 and 2, respectively, for over 90% of the cases). If normalized by the maximum available quality (6.0 Mb/s), it represents only 3.73% and 3.67% of residual error. This means that in roughly 97% of the cases, the quality levels are correctly predicted. Even though the residual error for Zone 3 is slightly higher (4.5%), it is still within the acceptability range.

The accuracy of the quality switch prediction (Figures 11(a) to 11(c)) shows even better results. For over 90% of the samples, Zones 1, 2 and 3 present a residual error of $r_{i_1} \leq 0.00745$, $r_{i_2} \leq 0.01604$ and $r_{i_3} \leq 0.01877$, respectively. In line with the findings for the average bitrate prediction, Zone 3 presented a higher residual error (1.9%), as this is the zone with the highest number of quality switches during the video playout. In Figure 11(b) it is possible to observe that on over 80% of samples the residual error is zero. This is because the quality switch behavior for both extreme cases of the network performance is predictable: first, when the network performance is sufficiently high, the rate adaptation will stabilize at the highest representation, and no further quality switches are expected. Second, when the network performance is degraded, the rate adaptation will keep the video playout at the lowest available quality representation, and, similarly, no further switches are expected.

The stalling time (Figure 12(a)) shows an error close to 13% for over 90% of the testing samples. One main reason behind such increased residual error is the wide range of the predicted variable (as we saw in the regression tree of Figure 9(a)). Nevertheless, several samples in the training dataset presented zero seconds of stall time. We found that such predictable cases are associated with high levels of network performance. For each of these samples, a residual error of 0.95 was accounted (as 0.95 is the lowest predicted value). As the presented regression tree is the optimal prune, further growth would lead to overfitting, and thus a higher cross-validation error. Due to the relatively high stall time for intermediate and degraded network performance, the prediction performance is impaired as the network performance degrades. However, at high levels of stall time, the QoE is already completely degraded. Thus, the increased error does not impair the accuracy of the QoE estimation.

The final VR playout parameter, the startup delay (Figure 12(b)), is characterized as the elapsed time between the request of the video and the playout of the first segment. In the considered context, the startup delay prediction presented a well predictable pattern with $r_i \leq 0.00473$ for over 90% of the cases. Also, the regression tree presented a stable prediction performance across all the evaluated samples.

Finally, Figure 12(c) depicts the residual error for the QoE estimation. By applying the QoE model defined in Section 4.2 to each sample i, it is possible to estimate QoE for both the predicted playout values and the original ones. Then, the residual error can be calculated. Through this procedure, the QoE estimation error induced by the proposed prediction scheme can be assessed. As shown in Figure 12(c), the QoE estimation presents $r_i \leq 0.03922$ for over 90% of the cases.

6 CONCLUSION

Virtual Reality applications based on adaptive tile-based video streaming are booming, as VR content becomes available to the general public. To be able to cope with their ultra-high bandwidth and low latency requirements, network and services providers are required to assess the end-client perceived performance of such services.

In this paper we presented PERCEIVE, a novel VR performance evaluation method to assess the user's perception of the VR content when streamed through the network. By means of machine learning techniques applied to the network performance indicators, it predicts the adaptive VR performance both in terms of VR main playout parameters (quality, quality switches, stalling time and starting time) and the perceived QoE. To our knowledge, this is the first VR performance model.

PERCEIVE has been evaluated considering a real-world environment, in which VR videos are streamed while subjected to an LTE/4G network performance. Then, we assessed its accuracy by means of the residual error between the predicted and measured values. PERCEIVE is able to predict the playout performance metrics with an average prediction error lower than 3.7% and, the perceived quality with a prediction error lower than 4% for over 90% of all the tested cases. PERCEIVE not only provides very high prediction accuracy, but also allows analyzing the influence of networks on the VR streaming parameters. This feature has helped us pinpoint the network delay as the QoS feature that affects the transport of VR services the most.

We believe our work is one step forward in the assessment of VR applications performance, which is an open subject in the state-of-the-art on multimedia network management. Although the proposed QoE model has not been validated through subjective tests, we believe it is an acceptable approach considering the scope of this work. As we are evaluating the predictability of the QoE indicator based on the performance of the application layer, possible adjustments in the weights of Equation 1 will not affect the prediction error of the QoE indicator. Aiming at providing realistic weights for Equation 1, as future work, we intend to perform subjective tests of the proposed QoE model. We also intend to explore and improve the estimation capabilities of our approach, focusing on viewport prediction and on adaptive streaming heuristics.

ACKNOWLEDGEMENT

This research was performed partially within the project G025615N "Optimized source coding for multiple terminals in self-organizing networks" from the fund for Scientific Research-Flanders (FWO-V). This work was also partially funded by CAPES, CNPq, FAPERGS and IFSul PROPESP.

REFERENCES

[1] Arslan Ahmad, Alessandro Floris, and Luigi Atzori. 2016. QoE-centric service delivery: A collaborative approach among OTTs and ISPs. *Computer Networks* 110 (2016), 168 – 179. https://doi.org/10.1016/j.comnet.2016.09.022

[2] Y. Bao, H. Wu, T. Zhang, A. A. Ramli, and X. Liu. 2016. Shooting a moving target: Motion-prediction-based transmission for 360-degree videos. In *2016 IEEE International Conference on Big Data (Big Data)*. 1161–1170. https://doi.org/10.1109/BigData.2016.7840720

[3] BBC. 2017. Facebook: We want a billion people in VR. (2017). https://goo.gl/2LNuAo Accessed 16-October-2017.

[4] Ethan Blanton, Dr. Vern Paxson, and Mark Allman. 2009. TCP Congestion Control. RFC 5681. (Sept. 2009). https://doi.org/10.17487/RFC5681
[5] Cisco. 2017. *Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2016–2021.* Technical Report. Cisco Systems.
[6] C. Concolato, J. Le Feuvre, F. Denoual, E. Nassor, N. Ouedraogo, and J. Taquet. 2017. Adaptive Streaming of HEVC Tiled Videos using MPEG-DASH. *IEEE Transactions on Circuits and Systems for Video Technology* PP, 99 (2017), 1–1. https://doi.org/10.1109/TCSVT.2017.2688491
[7] X. Corbillon, G. Simon, A. Devlic, and J. Chakareski. 2017. Viewport-adaptive navigable 360-degree video delivery. In *2017 IEEE International Conference on Communications (ICC)*. 1–7. https://doi.org/10.1109/ICC.2017.7996611
[8] R. I. T. da Costa Filho, W. Lautenschlager, N. Kagami, V. Roesler, and L. P. Gaspary. 2016. Network Fortune Cookie: Using Network Measurements to Predict Video Streaming Performance and QoE. In *2016 IEEE Global Communications Conference (GLOBECOM)*. 1–6. https://doi.org/10.1109/GLOCOM.2016.7842022
[9] E. Dahlman, G. Mildh, S. Parkvall, J. Peisa, J. Sachs, Y. Selén, and J. Sköld. 2014. 5G wireless access: requirements and realization. *IEEE Communications Magazine* 52, 12 (December 2014), 42–47. https://doi.org/10.1109/MCOM.2014.6979985
[10] Giorgos Dimopoulos, Ilias Leontiadis, Pere Barlet-Ros, and Konstantina Papagiannaki. 2016. Measuring Video QoE from Encrypted Traffic. In *Proceedings of the 2016 Internet Measurement Conference (IMC '16)*. ACM, New York, NY, USA, 513–526. https://doi.org/10.1145/2987443.2987459
[11] Ching-Ling Fan, Jean Lee, Wen-Chih Lo, Chun-Ying Huang, Kuan-Ta Chen, and Cheng-Hsin Hsu. 2017. Fixation Prediction for 360 Video Streaming in Head-Mounted Virtual Reality. In *Proceedings of the 27th Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV'17)*. ACM, New York, NY, USA, 67–72. https://doi.org/10.1145/3083165.3083180
[12] M. Hosseini and V. Swaminathan. 2016. Adaptive 360 VR Video Streaming: Divide and Conquer. In *2016 IEEE International Symposium on Multimedia (ISM)*. 107–110. https://doi.org/10.1109/ISM.2016.0028
[13] Junchen Jiang, Vyas Sekar, Henry Milner, Davis Shepherd, Ion Stoica, and Hui Zhang. 2016. CFA: A Practical Prediction System for Video QoE Optimization. In *13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16)*. USENIX Association, Santa Clara, CA, 137–150. https://www.usenix.org/conference/nsdi16/technical-sessions/presentation/jiang
[14] Xianshu Jin, Hwiyun Ju, Sungchol Cho, Boyeong Mun, Cheongbin Kim, and Sunyoung Han. 2016. QoS routing design for adaptive streaming in Software Defined Network. In *2016 International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS)*. 1–6. https://doi.org/10.1109/ISPACS.2016.7824694
[15] Ron Kohavi et al. 1995. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Ijcai*, Vol. 14. Stanford, CA, 1137–1145.
[16] Hongzi Mao, Ravi Netravali, and Mohammad Alizadeh. 2017. Neural Adaptive Video Streaming with Pensieve. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication (SIGCOMM '17)*. ACM, New York, NY, USA, 197–210. https://doi.org/10.1145/3098822.3098843
[17] Sreerama K. Murthy. 1998. Automatic Construction of Decision Trees from Data: A Multi-Disciplinary Survey. *Data Min. Knowl. Discov.* 2, 4 (Dec. 1998), 345–389. https://doi.org/10.1023/A:1009744630224
[18] H. Nam, K. H. Kim, and H. Schulzrinne. 2016. QoE matters more than QoS: Why people stop watching cat videos. In *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*. 1–9. https://doi.org/10.1109/INFOCOM.2016.7524426
[19] Omar A. Niamut, Emmanuel Thomas, Lucia D'Acunto, Cyril Concolato, Franck Denoual, and Seong Yong Lim. 2016. MPEG DASH SRD: Spatial Relationship Description. In *Proceedings of the 7th International Conference on Multimedia Systems (MMSys '16)*. ACM, New York, NY, USA, Article 5, 8 pages. https://doi.org/10.1145/2910017.2910606
[20] Pradrip Paudyal, Federica Battisti, and Marco Carli. [n. d.]. Impact of video content and transmission impairments on quality of experience. *Multimedia Tools and Applications* 2016 ([n. d.]). https://doi.org/10.1007/s11042-015-3214-0
[21] PERCEIVE. 2018. Performance Estimation for VR Videos. (2018). https://github.com/rtcostaf/PERCEIVE Accessed 2-April-2018.
[22] Stefano Petrangeli, Jeroen Famaey, Maxim Claeys, Steven Latré, and Filip De Turck. 2015. QoE-Driven Rate Adaptation Heuristic for Fair Adaptive Video Streaming. *ACM Trans. Multimedia Comput. Commun. Appl.* 12, 2, Article 28 (Oct. 2015), 24 pages. https://doi.org/10.1145/2818361
[23] Stefano Petrangeli, Viswanathan Swaminathan, Mohammad Hosseini, and Filip De Turck. 2017. An HTTP/2-Based Adaptive Streaming Framework for 360 Virtual Reality Videos. In *Proceedings of the 2017 ACM on Multimedia Conference (MM '17)*. ACM, New York, NY, USA, 306–314. https://doi.org/10.1145/3123266.3123453
[24] Feng Qian, Lusheng Ji, Bo Han, and Vijay Gopalakrishnan. 2016. Optimizing 360 Video Delivery over Cellular Networks. In *Proceedings of the 5th Workshop on All Things Cellular: Operations, Applications and Challenges (ATC '16)*. ACM, New York, NY, USA, 1–6. https://doi.org/10.1145/2980055.2980056
[25] R Core Team. 2017. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. https://www.R-project.org/
[26] Muhammad Zubair Shafiq, Jeffrey Erman, Lusheng Ji, Alex X. Liu, Jeffrey Pang, and Jia Wang. 2014. Understanding the Impact of Network Dynamics on Mobile Video User Engagement. *SIGMETRICS Perform. Eval. Rev.* 42, 1 (June 2014), 367–379. https://doi.org/10.1145/2637364.2591975
[27] Iraj Sodagar. 2011. The MPEG-DASH Standard for Multimedia Streaming Over the Internet. *IEEE Multimedia* 18, 4 (2011), 62–67.
[28] M. Torres Vega, C. Perra, and A. Liotta. 2018. Resilience of Video Streaming Services to Network Impairments. *IEEE Transactions on Broadcasting* (2018). https://doi.org/10.1109/TBC.2017.2781125
[29] J. van der Hooft, S. Petrangeli, T. Wauters, R. Huysegems, P. R. Alface, T. Bostoen, and F. De Turck. 2016. HTTP/2-Based Adaptive Streaming of HEVC Video Over 4G/LTE Networks. *IEEE Communications Letters* 20, 11 (2016), 2177–2180.
[30] M. Viitanen, A. Koivula, A. Lemmetti, J. Vanne, and T. D. Hämäläinen. 2015. Kvazaar HEVC encoder for efficient intra coding. In *2015 IEEE International Symposium on Circuits and Systems (ISCAS)*. 1662–1665. https://doi.org/10.1109/ISCAS.2015.7168970
[31] Chenglei Wu, Zhihao Tan, Zhi Wang, and Shiqiang Yang. 2017. A Dataset for Exploring User Behaviors in VR Spherical Video Streaming. In *Proceedings of the 8th ACM on Multimedia Systems Conference (MMSys'17)*. ACM, New York, NY, USA, 193–198. https://doi.org/10.1145/3083187.3083210
[32] Xiaoqi Yin, Abhishek Jindal, Vyas Sekar, and Bruno Sinopoli. 2015. A Control-Theoretic Approach for Dynamic Adaptive Video Streaming over HTTP. *SIGCOMM Comput. Commun. Rev.* 45, 4 (Aug. 2015), 325–338. https://doi.org/10.1145/2829988.2787486

APPENDIX

This appendix provides a detailed description of the procedure to be followed in order to allow reproducibility of the experiments performed in this work. All the employed datasets and source code are available at the PERCEIVE repository [21].

In Section 1.1 we present an overview of the experimental design and its general setup. Next, the re-encoding procedure required to generate tile-based VR-videos is explained in Section 1.2 Then, in Section 1.3, the VR video player considered in this work is examined. Subsequently, both the network and the VR video playout datasets are thoroughly discussed in Section 1.4. Finally, in Section 1.5, we give an overview of the R scripts responsible for performing the machine learning task.

1.1 Experimental Procedure Overview and Specifications

The experimental procedure is split into three steps, as explained in Section 5.1 (Figure 4). First, the VR video player requests tile-based videos, from a web server, while subjected to controlled network conditions. The VR video player is responsible for measuring and recording the VR video playout performance, while the network conditions are enforced by Linux Traffic Control (TC) mechanism. Second, the VR video playout performance indicators are given as input to the machine learning process. At this stage, machine learning is responsible for characterizing how each network condition impacts the video playout performance. Finally, in the third step, an estimation of QoE is provided by giving the VR video performance as input to the QoE model.

To perform the first step, we employ three dedicated virtual machines deployed on the imec iLab.t Virtual Wall emulation platform[5]. The first machine was used to run the VR video player, while the second was used to host tile-based VR videos using a regular Apache web server. Through traditional IP routing and Linux Traffic Control (TC), the third machine was configured as a gateway between the other two, acting as a network condition enforcement point. Each virtual machine was configured with a quad-core Intel

[5] imec iLab.t: http://doc.ilabt.iminds.be/ilabt-documentation/virtualwallfacility.html

Xeon E3-1220 v3 CPU running at 3.10GHz, 15GB RAM, 16GB of storage and running Linux Ubuntu 14.04 (3.13.0-33). The full list of packages and its respective versions is available at PERCEIVE's repository [21] (Setup/packages.txt).

Steps two and three do not require any specific hardware or software specification. Step two was performed using R (1.0.143) [25], and for the third step we employed a simple electronic spreadsheet to compute the QoE model (Section 4.2) over the VR video playout performance indicators. After this overview, the remainder of this section will cover practical details of the main elements of the experiment.

1.2 Tile-based HAS VR-video Re-encoding

In order to generate tile-based HAS VR-videos, it was necessary to re-encode the original VR videos from Wu et al.' s dataset [31] (namely "Google Spotlight-HELP" and "Freestyle Skiing"). Herein, the re-encoding procedure is explained step-by-step.
After downloading the original VR-videos "Google Spotlight-HELP"[6] and "Freestyle Skiing"[7], the raw videos must be first extracted using the following command of FFMPEG[8]:

```
$ ffmpeg -i inVideo.mkv -c:v rawvideo
outVideo.yuv
```

Next, the HEVC tile-based version of the videos is generated using Kvazaar[9]. Kvazaar splits the videos based on the generated YUV file, the desired tiling scheme, resolution and frames per second (FPS), as shown in the following example. This command is to be executed per video quality.

```
$ kvazaar -i outVideo.yuv --input-res
3840x2160 -o outVideo12x4.hevc --tiles
12x4 --slices tiles --mv-constraint
frametilemargin -q 30 --period 30
--input-fps 30
```

Subsequently, each of the tiles of the VR-video is packed into an mp4 container employing the MP4Box software [10].

```
$ MP4Box -add outVideo12x4.hevc:split_tiles
-fps 30 -new video_tiled_4K_12x4.mp4
```

Finally, based on the desired length of the HAS segment, the per-tile per segment files of the VR-video are extracted. For example, the following command defines one second for the segment length, 12x4 tiling scheme and three video resolutions (720p, 1080p and 4K). This procedure also generates MPD files by using multiple quality representations.

```
$ MP4Box -dash 1000 -rap -frag-rap
-profile live -out has_tiled_12x4.mpd
../SOURCE/video_tiled_720_12x4.mp4
../SOURCE/video_tiled_1080_12x4.mp4
../SOURCE/video_tiled_4K_12x4.mp4
```

[6]https://youtu.be/G-XZhKqQAHU
[7]https://youtu.be/0wC3x_bnnps
[8]FFMPEG: https://www.ffmpeg.org/
[9]Kvazaar: https://github.com/ultravideo/kvazaar
[10]MP4box: https://gpac.wp.imt.fr/mp4box/

1.3 VR Video Player

Both the source code and binary for the VR video player are available at the PERCEIVE repository [21] (VR-player/Source and VR-player/bin respectively). The player provides support to variable tiling scheme and can be adapted to several QoE zone schemes (Section 3.1). Additionally, the player supports viewport traces (a previously recorded log regarding the user's head track) as input. The player is written in C language and employs Curl library to perform HTTP requests. The player also allows parameters to be passed through command line arguments. It is particularly useful when running large experiments, so that the player parameterization can be done dynamically by an external script. The full set of parameters is shown in Table 5. For example, the following player call is used for requesting the first 60 segments of the video named "video2", available at the IP "10.0.0.251", using the viewport trace stored in the file "user1/video2.txt", using 100 seconds timeout and a 12x4 tiling scheme. In this case, the resultant VR-video playout performance will be written in the file named "video2playout".

```
$ VR-player 10.0.0.251 video2 60
video2playout user1/video2.txt 100 4 12
```

Table 4: VR-video player command line arguments.

Sequence	Description
1	IP address of the video server
2	Video filename
3	Number of segments to download
4	Output filename (to write playout performance results)
5	Viewport trace filename (head track logs)
6	Session timeout (max number of seconds)
7	Number of vertical tiles (tiling scheme)
8	Number of horizontal tiles (tiling scheme)

1.4 Video Playout and Network Datasets

The file "Sample.csv" (directory "Network dataset" [21]) provides the 48 network conditions considered in our experiments. The conditions were extracted from [29] and adapted according the procedure described in Section 5.2. The range for each input parameter is summarized in Table 1. The configuration ID is the leftmost field in the file "Sample.csv", followed by the fields throughput TCP (Mb/s), delay (msec) and packet loss rate (%). After parsed, these values are given as input to the Linux TC, which act as a network condition enforcement point.

In turn, the file "playoutPerformance.txt" (directory "Playout performance dataset" [21]) provides the resultant output of the first step of the experimental procedure (described in Section 1.1). Furthermore, this is the same file given as input to the machine learning process (step two). Along with the network dataset, Table 5 summarizes the input parameters for generating the playout performance dataset.

Table 6 shows the set of fields of the resultant VR-video playout performance. Fields 1 - 5 are related to the video input parameters, listed in Table 5. Fields 6 - 8 are related to the network conditions. Fields 9 - 22 corresponds to the VR-video playout performance measured by the VR-video player. Finally, fields 23 - 25 are calculated as the number of per-zone tiles times the average bitrate for each video resolution. In the PERCEIVE repository, we provide a

Table 5: PERCEIVE video input parameters.

Parameter	Value/Range	Details
VR video	Google Spotlight-HELP and Freestyle Skiing	V1 and V2 (from [31])
Head track traces	Google Spotlight-HELP and Freestyle Skiing	V1 and V2 (from [31])
Video format	MP4 - HEVC tile-based and HAS	Using MP4Box
Video encoder	Kvazaar	Kvazaar encoder [30]
HAS	720p (1.8Mb/s), 1080p (2.7Mb/s) and 4K (6Mb/s)	Kvazaar encoder [30]
Segment size	1 second	From [11]
Tiling scheme	8x4 and 12x4	From [24]
Considered viewport	One central tile and eight border tiles	Section 3.1

bash script "addQuality.sh" (directory Scripts) which can be used to perform this computation.

Table 6: Fields sequence of the file "playoutPerformance.txt" (Field 1 is the leftmost value in the file).

Field	Description	Type/unit
1	Video ID	string
2	User ID	string
3	Tile format	horizontal X vertical
4	Network trace ID	string
5	Experiment round	integer
6	TCP throughput	Mb/s
7	Delay	msec
8	Packet loss	%
9	Number of tiles 720p for Zone 1	integer
10	Number of tiles 1080p for Zone 1	integer
11	Number of tiles 4K for Zone 1	integer
12	Number of tiles 720p for Zone 2	integer
13	Number of tiles 1080p for Zone 2	integer
14	Number of tiles 4K for Zone 2	integer
15	Number of tiles 720p for Zone 3	integer
16	Number of tiles 1080p for Zone 3	integer
17	Number of tiles 4K for Zone 3	integer
18	Number of quality switches for Zone 1	integer
19	Number of quality switches for Zone 2	integer
20	Number of quality switches for Zone 3	integer
21	Stall time	seconds
22	Startup delay	seconds
23	Average bitrate for Zone 1	Mb/s
24	Average bitrate for Zone 2	Mb/s
25	Average bitrate for Zone 3	Mb/s

1.5 Machine Learning

The directory "R Scripts" [21] provides all the source code used to generate the regression decision trees shown in Section 5.3. Each of the eight decision trees has its own source code (R script). In addition to the R tool [25], we employed the following packages: stargazer [11], gdata [12], rpart [13], tree [14] and rpart.plot [15]. Finally, it is worth mentioning that the trees shown in this work were obtained through their optimal prune. Which means that during the prune stage, we selected the complexity parameter (CP) associated with the minimum cross-validation error (xerror).

[11] https://cran.r-project.org/web/packages/stargazer/index.html
[12] https://cran.r-project.org/web/packages/gdata/index.html
[13] https://cran.r-project.org/web/packages/rpart/index.html
[14] https://cran.r-project.org/web/packages/tree/index.html
[15] https://cran.r-project.org/web/packages/rpart.plot/index.html