EUROGRAPHICS Workshop on Intelligent Cinematography and Editing (2020)
M. Christie, H.-Y. Wu, T.-Y. Li and V. Gandhi (Editors)

User-Adaptive Rotational Snap-Cutting for Streamed 360° Videos

M. F. Romero Rondón[1,2], L. Sassatelli[1], R. Aparicio-Pardo[1] and F. Precioso[2]

[1]Université Cote d'Azur, CNRS, INRIA, I3S, Sophia Antipolis, France
[2]Université Cote d'Azur, CNRS, I3S, Sophia Antipolis, France

Abstract

Designing editing cuts for cinematic Virtual Reality (VR) has been under active investigation. Recently, the connection has been made between cuts in VR and adaptive streaming logics for 360° videos, with the introduction of rotational snap-cuts. Snap-cuts can benefit the user's experience both by improving the streamed quality in the FoV and ensuring the user sees important elements for the plot. However, snap-cuts should not be too frequent and may be avoided when not beneficial to the streamed quality. We formulate the dynamic decision problem of snap-change triggering as a model-free Reinforcement Learning. We express the optimum cut triggering decisions computed offline with dynamic programming and investigate possible gains in quality of experience compared to baselines. We design Imitation Learning-based dynamic triggering strategies, and show that only knowing the past user's motion and video content, it is possible to outperform the controls without and with all cuts.

CCS Concepts

• ***Computing methodologies*** → ***Virtual reality; Sequential decision making; Apprenticeship learning;*** • ***Information systems*** → ***Multimedia streaming;***

1. Introduction

The question of designing editing cuts to drive the user's attention in cinematic Virtual Reality (VR) has been under active investigation for the last 3 to 4 years. If driving the user's attention is critical for a director to ensure the story plot is understood, in this paper we investigate attention driving techniques from a different perspective: that of the multimedia networking community. The development of VR contents is persistently hindered by the difficulty of accessing them through regular Internet video streaming. Indeed, owing to the closer proximity of the screen to the eye and to the width of the spherical content, the data rate to stream 360° videos may be two orders of magnitude that of a regular video [BBMD17]. To decrease the amount of data to stream, a solution is to split the sphere into tiles and send in high resolution only the portion of the sphere the user has access to at each point in time, named the Field of View (FoV). This however requires to know the user's head position in advance, that is at the time of sending the content from the server. This can go from a few tens of seconds (low network delay) to a few seconds (extreme network delay or presence of a video playback buffer at the client to absorb network rate variations). Predicting the user's head motion is difficult and can be done accurately only over short horizons (less that 2s, see [NYN18, RRSAPP19]). If the server makes an error in prediction and the level of bandwidth is sufficient to attempt sending again in High Quality (HQ) tiles previously sent in Low Quality (LQ), then the prediction error translates into a higher level of network bandwidth consumption. In 360° video streaming, the consumed data rate therefore depends on the prediction error, that is on the user's motion, which in turn depends on the user's attentional process and hence on the editing cuts. Fig. 1 depicts this interplay, from which can arise interdisciplinary approaches to jointly design 360° video streaming algorithms and 360° film editing techniques.

In [DSS*18a], for the first time, Dambra et al. have showed that film editing can be helpful for streaming 360° videos by directing the user's attention to specific pre-defined Regions of Interest (RoI), thereby lowering the randomness of the user's motion and using this a-priori knowledge in the streaming decisions (consisting, at each point in time, in deciding which area/tile of which video segment to send in which quality). This is done by periodically regaining control on the user's FoV using so-called snap-changes, which are a type of intra-scene rotational cuts. They are interchangeably called snap-cuts. The details of the method and its positioning with respect to the relevant literature are provided in Sec. 2. Dambra et al. showed that it is beneficial both for application-level metrics (level of quality in FoV, consumed bandwidth) and user-experience metrics (user's angular speed, story understanding). Their results are a proof-of-concept where the set of cuts is pre-defined in an XML file by, e.g., the film director, and all cuts are executed at their corresponding time instants. However, snap-cuts are forced re-positioning corresponding to momentaneously taking some freedom off of the user. These intra-scene cuts can hence be envisioned as levers that the 360° video player can leverage to cope with an insufficient bandwidth and still succeeding in displaying HQ in the

DOI: 10.2312/wiced.20201125

Figure 1: *Streaming 360° videos: the sphere is tiled and each tile of the sphere is sent into low or high quality depending on the user's motion and network bandwidth.*

user's FoV (which gets re-positioned in front of the HQ area). Every cut may hence not always be necessary:

- if the user will be close enough to the FoV targeted by the cut: this is implemented by the 30°rule already implemented in the original method [DSS*18a] (with a moderate impact on the quality downloading decision)
- if the network bandwidth is high enough so that replacements are possible and not costly, and/or
- if the user moves slowly enough that the spatial qualities fetched earlier for each area/tile overlap sufficiently the FoV at the time of playback (so the cut will not help increase the quality in the FoV).

Whether or not a cut will be beneficial therefore depends on the user's motion and on the network conditions. Specifically, the trade-off involves: (i) a snap-change guarantees that the user will see the FoV desired by the director, and that HQ is displayed in this FoV, while (ii) not having a snap-change may preserve the level of presence and keep low the probability of disorientation.

In this article, we consider the network conditions being fixed and investigate how to optimize cut triggering to obtain best trade-off, by designing a user-adaptive editing policy for 360° video streaming.

Contributions:

• We model the dynamic decision problem of snap-change triggering as a model-free Reinforcement Learning (RL), for which we model the user's quality of experience as a reward function based on the quality in FoV penalized by the cut frequency. We express the optimum cut triggering decisions computed offline with dynamic programming, when the user's motion is known before but also after the cut decision time.

• We adopt a machine learning framework from the realm of Imitation Learning, namely Behavioral Cloning, to train different strategies aimed at approaching optimal decisions. We show that it is possible to improve the quality of experience by dynamically deciding to trigger snap-cuts, only knowing the past user's motion and video content, compared to the controls without and with all cuts.

After presenting the related works in Sec. 2, Sec. 3 presents the problem formulation. The design of the learning approaches is presented in Sec. 4, and Sec. 5 gathers the results and their analysis. Finally, the limitations of this preliminary work are discussed in Sec. 6, and conclusions are given in Sec. 7.

2. Related works

This section reviews the main categories of existing works relevant to the problem at hand. We provide successfully an overview of techniques to partly control the user's FoV, introduce the concept of adaptive 360° video streaming and the required adaptation to the user, and finally review some ML-based approaches for various questions arising with 360° videos.

2.1. Directing change of FoV

The works presented in [YLN*17, SMSR17, FT20] are aimed at assisting the user in moving in the virtual environment to increase comfort and/or lower sickness, while those presented in [PHA17, RBR18, RHB05, SPDW*18, DSS*18a] consist in operating FoV changes independently from the user's motion or will. In [SMSR17], amplified and guided head rotation are introduced for seated use of VR in HMDs: physical head rotation angles are magnified in the VE so that physically turning in a (limited) comfortable range can allow wider range. In [FT20], Farmani et al. show that it is possible to artificially reduce vection – the illusion of self-motion, which is connected to cybersickness – by snapping the viewpoint, reducing continuous viewpoint motion by skipping frames. They show that both rotational and translational snapping reduce cybersickness by 40% and 50 %, respectively. In [LCH*17], the authors compare an auto-pilot mode deciding which FoV is exposed to a seated user, independently of the user movements, with assisting the user by displaying an arrow of where to look. In [PHA17] and [RBR18], user-initiated (by pressing a button) and system-initiated rotational re-positioning of FoV are performed, independently of the user motion. The re-positioning is progressive and while participants could most easily track scene changes, they generally and unsurprisingly experienced sickness due to rotational vection. Much interestingly, it has been uncovered in [RHB05] that participants can automatically update their sense of spatial orientation during rotation using only visual cues, and that the accompanying physical rotation may not be necessary for fast and reflexive updating. Also, how cut frequency influences viewers' sense of disorientation and their ability to follow the story has been studied in [KLMP*17]. The results show that high editing cut frequency can be very well received, as long as the user's attention is appropriately guided at the point of cut. In [SPDW*18, DSS*18a], the authors introduce so-called dynamic editing with (rotational) snap-changes, combining the positive aspects of the above methods: to periodically regain control over the user's FoV at the time of 360° video playback (these intra-scene cuts are hence not built into the video file), the FoV is repositioned, in a snap, that is from one frame (image) to the next, in front of a pre-determined FoV (possibly decided by the director). To do so, only an additional XML file has to be downloaded at the begining of the video, and the custom player [DSS*18b] implements the snap-changes at the indicated times in front of the indicated angular sectors. Fig. 2 depicts the effect of a snap-change on the FoV. It is shown in [DSS*18a] that this strategy enables to reduce the average head motion speed by up to 30%, that these repositionings are mostly not noticed by the users when performed towards a meaningful RoI (possibly perceived as fast-cutting, which may sometimes feel like missing in cinematic VR), and when they notice it, no discomfort or sickness are yielded

as no motion is sensed, and the vestibular system is not involved. It is also shown that incorporating the knowledge of the future snap-changes into the adaptive streaming algorithm enables bandwidth savings, as described below.

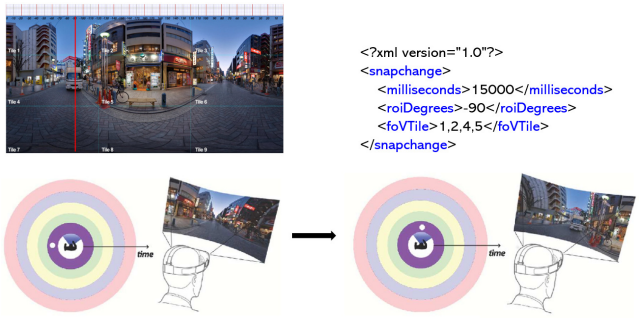

Figure 2: *Top left: Identification of the RoI targeted by the snap-change. Top right: Description of the list of snap-changes over the video as an XML file.*
Bottom: FoV re-positioning in front of the targeted RoI by the snap-change.

2.2. Adaptive streaming for 360° videos

Modern (regular non-360°) video streaming relies on the concept of HTTP Adaptive Streaming, whose most wide spread version is the MPEG-DASH standard [MPE14]. It consists in the video file being chunked into temporal segments of fixed duration (often 2 sec. or 5 sec.), each encoded into several quality levels, that is at different bitrates (often corresponding to resolutions). The client strives to (i) prevent playback interruptions by maintaining a non-empty playback buffer where a certain number of segments (or seconds of video) are stored in advance of playback, while (ii) fetching and displaying qualities as high as possible. To do so, the client runs a so-called adaptive streaming logic (or algorithm) which chooses which quality to request for every segment to the remote server, based on the network bandwidth varying over time. As explained in Sec. 1, in the case of 360° video streaming, a single segment does not correspond anymore to a single entity, but possibly to several tiles. The goal is to reduce the required bandwidth to stream 360° videos by requesting high quality for the tiles that will intersect the FoV of the user. The qualities to request for every tile of every segment must therefore adapt both to the network and the user dynamics, as represented in Fig. 3. The challenge is that at the time of the decision, the future of the network bandwidth (which will support the currently decided object to download) and the user motion (which will determine where will the FoV be at the time this segment is played out) are unknown. Most recent examples of strategies addressing this difficult problem are [ZZB*19, PBY*19], which strive to predict network bandwidth and user's motion with recurrent deep neural networks. The innovative interdisciplinary approach presented in [DSS*18a] consisted in not trying to predict the future user's motion (which is much difficult to do [RRSAPP19]), but instead in designing a 360° adaptive streaming algorithm which benefits from the knowledge of the future rotational cuts to better target which tile must be fetched in HQ. The bandwidth savings yielded by such approach have been shown to be substantial (up to 25%). In [SWFA19], a first attempt of learning how to trigger snap-changes of [DSS*18a] is demoed. It implements a deep Reinforcement Learning (RL) strategy to adapt to the user's motion, without proving that it can work better than a simple baseline, nor quantifying the gains. In the present article, we design a complete learning framework addressing the problem of FoV overlap (quality) prediction and snap-change triggering decision, to best trade between quality in FoV and user's freedom while streaming a 360° video over a bandwidth-limited network.

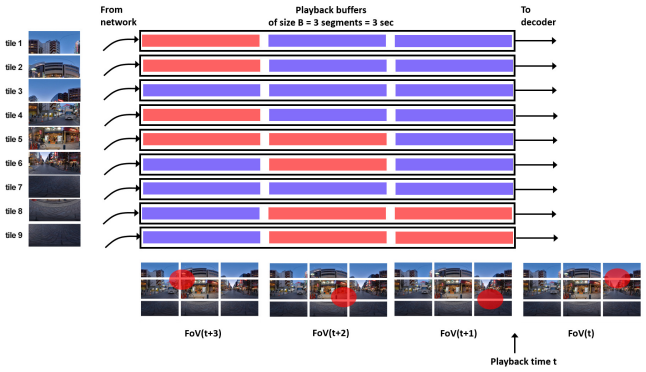

Figure 3: *Buffering process for a tiled 360° video. While segment n is being decoded at time t, segment $n+B$ is being downloaded. Red (resp. blue) rectangles represent tiles' segments in HQ (resp. LQ). Idealistically, the tiles in HQ must match the user's FoV at their time of playback.*

3. Problem definition

In this section, we first expose, in Sec. 3.1 all the system and model assumption we make (that we implement in our simulator used for training and analysis in 4 and Sec. 5), before formally describing the optimization problem in Sec. 3.2. The notations are provided in Table 1.

3.1. Assumptions

System assumptions: As aforementioned, a 360° streaming strategy has to be both network- and user-adaptive. In this work, we focus on how to adapt the frequency of the snap-changes to the user's motion and maximize their level of experience – a Quality of Experience (QoE) function defined in Sec. 3.2. Therefore, we set fix the underlying streaming strategy, detailed in Algo. 1, working as follows. We consider (wlog) that 2 quality levels are available. The qualities (HQ or LQ) for every tile of every segment are decided based on the current user's FoV at the time of downloading this tile (HQ for the tiles intersecting the FoV, LQ for the others). The tile's download can occur up to B seconds before its playback, where B is the size of the playback buffer. We consider the time to download a tile negligible. The downloading process pauses when the buffer is full, and resumes when at least one segment has been dequeued and fed to the video decoder. We consider that the list $\mathcal{C}$ of potential snap-changes is loaded with the video description file before the playback starts (each snap-change is described as

System param.	Definition
N ($\mathcal{N}$), M ($\mathcal{M}$)	number (set) of segments, tiles
C ($\mathcal{C}$)	number (set) of possible snap-changes
B	playback buffer size (in sec.)
$q_m(n) \in \{0,1\}$	quality level of tile m at segment n
$buf_m(t)$	num. of sec. stored in buffer of tile m at time t
$time(c)$	playback time of snap-change c
$FoV_{snap}(c)$	FoV targeted by snap-change c
$flag_{decided}(c)$	$=1$ if the decision for c is made, $=0$ ow.
$flag_{trigger}(c)$	$=1$ if c has been decided to be triggered, $=0$ ow.
$w_{past}(t)$	time interval $[0,t]$
$w_{fut1}(t)$	time interval $]t,t+B]$
$w_{fut2}(t)$	time interval $]t+B,t+B+D]$
Model param.	**Definition**
D	duration of snap-cut impact on user's motion
β	penalty for triggering a snap-change

Table 1: *Notation of the system and model parameters.*

depicted in Fig. 2-top right). We then augment the streaming strategy as described in Algo. 1: when the first tile of a new segment is about to be downloaded, if the list of possible snap-changes indicates that a snap-change may be trigger at this segment, then a decision function is called to decide whether this snap-change should be triggered to maximize the objective function described below. Designing such a snap-triggering decision function is the subject of this article.

QoE model assumptions: We model the user's reaction to the triggering of a snap-change. It has been shown in [DSS*18a, Fig. 11], with user experiments, that snap-changes decrease the head motion speed of users by up to 30%. In the simulations used in this article (later used to train classifiers), we use the user motion dataset presented in [DGC*18], which contains the head tracking data of 57 users exploring 19 videos. In this dataset, considering a FoV to be about 100° wide [FLL*17], it takes about 5 sec. for 50% of the users to shift their FoV entirely. This can be seen in [RRSAPP19, Fig. 7]. In this article, we adopt a preliminary simplistic user model: considering that a 30% decrease in angular speed would add 1.5 sec. to shift the field of view of half of these users, we consider in our simulator that right after each triggered snap-change, repositioning the user in front of a meaningful/interesting FoV, every user pauses for 2 sec. before resuming her motion.

3.2. Formulation of the optimization problem

To formulate the decision problem, we adopt a model for the level of user's experience under adaptive 360° streaming and snap-changes. We define an instantaneous reward obtained after the playout of segment n (of duration 1s in the simulator):

$$r(n) = q_{FoV}(n) - \beta^{t(n)-t_{last}(n)} . \quad (1)$$

The components are as follows. We define the average quality in the FoV over segment n's duration as $q_{FoV}(n) = \sum_{m=1}^{M} q_m(n)F(n,m)$, with $q_m(n)$ the quality level of tile m at segment n, and $F(n,m)$ the average fraction of FoV at segment n occupied by tile m. The second component represents the penalty incurred by triggering snap-changes: β is a penalty parameter, $t(n)$ is the playback time of segment n (close to n depending on playback interruption), and $t_{last}(n)$ is the time of the last triggered snap-change.

Algorithm 1: Downloading process: quality allocation and snap-change triggering decisions

Data: Current playback time t. Buffer states $buf_m(t)$, $\forall m \in \mathcal{M}$. List of possible snap-changes $c \in \mathcal{C}$. Initially $flag_{decided}(c) = 0$ and $flag_{trigger}(c) = 0$, $\forall s \in \mathcal{C}$.
Result: which segment n of which tile m to download next in which quality $q_m(n)$, whether to trigger a snap-change if n has one not decided yet, $\forall m \in \mathcal{M}$, $n \in \mathcal{N}$, $q_m(n) \in \{0,1\}$

```
if all M buffers are full then
    stay idle;

else
    set m to the tile index with the least full buffer;
    set n to the lowest value not having entered buffer m yet;
    q_m(n) = 0 # initialize with LQ;
    if there exists a snap-change c such that
    time(c)= nand flag_decided(c) = 0 then
        # Whether this snap-change will be triggered at time time(c)
        must be decided now;
        form environment state state(t);
        flag_trigger(c) =decision(state(t));
        flag_decided(c) = 1;

    identify the highest snap index c_last with flag_trigger(s_last) = 1;
    if c_last not empty and t ≤time(c_last)≤ n then
        if m intersects snap-change's FoV_snap(c_last) then
            q_m(n) = 1 # download in HQ;

    else
        if m intersects current user's FoV(t) then
            q_m(n) = 1 # download in HQ;
```

Let us denote by S_u the state of the streaming process (tiles' qualities stored in the buffer) and user's state (past head coordinates but also fatigue, likeliness to move) at the time of deciding for snap-change c. Given that the decision of triggering each snap is taken at the time of downloading it, hence of it entering the playback buffer, S_{u+1} is independent of what happened before u conditionally on S_u, and hence the Markov property is verified. The snap triggering problem is therefore a Markov Decision Process (MDP) defined by the tuple $(\mathcal{S},\mathcal{A},\mathcal{P},\mathcal{R},\gamma)$, where $P^a_{ss'} = Pr[S_{u+1}=s'|S_u=s,A_u=a]$, $R^a_s = \mathbb{E}[R_u|S_u=s,A_u=a]$. The time is paced with snap-triggering decisions, $a \in \{0,1\}$, and $R_u = \sum_{n\in N(u)} r(n)/|N(u)|$ where $N(u)$ is the segments played between both snaps decided at u and $u+1$. We define $\pi(a|s) = Pr[A_u = a|S_u = s]$ and formulate the snap-change triggering optimization problem as:

$$\pi^\star = \arg\max \mathbb{E}\Big[\sum_{u=0}^{C-1} \gamma^u R_u\Big] . \quad (2)$$

The optimal snap-triggering strategy must therefore trigger a snap-change when the contribution, to the cumulative reward, of the quality increase due to this snap exceeds the incurred penalty. It

therefore must trigger depending on the user's motion. The transition probability distribution $P^a_{ss'}$ is unknown as, to take online decisions sequentially, we cannot know how the user is going to move and react, given a certain state. The decision-making problem is hence that of model-free RL.

Finally, it is interesting to notice that other forms of similar problems have appeared in much different application domains. In [PGV*09], Pineau et al. present how to automatically learn an optimal neurostimulation strategy for the treatment of epilepsy. The goal is to trigger the minimum amount of neurostimulation (electric discharge from intra-cerebral electrodes) as a function of the EEG signal, so as to minimize the number of seizures.

4. Learning how to trigger a snap-cut

For the reasons exposed in Sec. 3.2, deciding dynamically during the video playback how to trigger a snap-change is a model-free RL problem. However, rather than designing an RL agent learning from its own observations, we instead leverage here the principle of learning from expert's demonstration, also known as Imitation Learning (IL) [OPN*18]. In our case, for the short videos considered, we indeed have access to an expert which is the *offline optimal* defined below in Sec. 4.1. We consider in this article the simplest version of IL, known as Behavioral Cloning (BC) [OPN*18]. It consists in supervising the training of the action policy/decision classifier (deciding whether or not to trigger the next snap-change), with the trigger labels provided by the *offline optimal*. A major design difficulty is the possibly loose correlation between the past motion at time t, and the decision to trigger snap c decided at time t but impacting user's motion only from time $t+B$, as represented in Fig. 4. We therefore split the snap-triggering problem at time t into 2 sub-problems: (P1) predicting the FoV overlap over $w_{fut2}(t)$ from $w_{past}(t)$ (as defined in Table. 1 and shown in Fig. 4), and (P2) deciding whether to trigger based on the series of predicted overlaps.

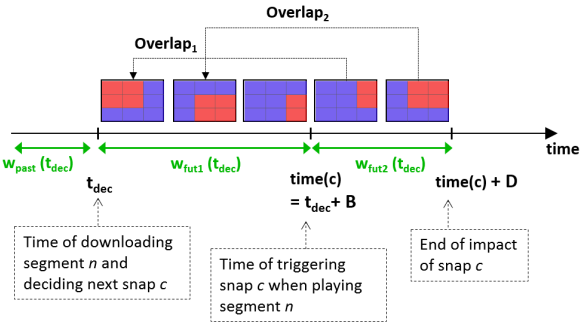

Figure 4: *Timing of the process: the tiles' qualities displayed at time t have been downloaded at time $t-B$. If segment n to be downloaded at time t_{dec} and to be played at $t_{dec}+B$, contains a possible snap-change c, either (i) this snap is not triggered, then the quality in the user's FoV at any $t \in w_{fut2}(t_{dec})$ is given by $overlap(t) = FoV(t) \cap FoV(t-B)$, or (ii) it is triggered, then only HQ is displayed in the FoV as the qualities fetched at $t-B$ are based on the snap-change's FoV $FoV_{snap}(c)$.*

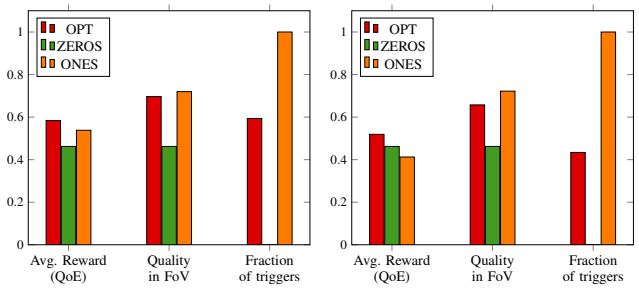

Figure 5: *Reward, quality and snap frequency for the optimal prediction and for the control baselines: always trigger (ONES) and never trigger (ZEROS). The parameters are: FoV Area: 100°x50°, regular snap interval of 3 sec., $D=2$, $B=4$ (Left) $\beta=0.2$, (Right) $\beta=0.35$.*

4.1. Definition of expert and baseline

We consider supervised learning and the labels being the optimal decisions. With the perfect knowledge of the user's motion over the entire video, we are able to compute the set of optimal decisions for each trace, corresponding to a given user watching a given video. This *offline optimal* solution is computed with Dynamic Programming. This is feasible in reasonable time only if the number of decisions is not too high. In Sec. 5, we consider 20 sec-long videos with a possible snap-changes every 3 sec., hence a total of 7 snap-changes. Fig. 5 illustrates, for $B=4$ and $\beta=0$ and 0.2, the gain of the *offline optimal* (*OPT* in legend) over two baselines: *ZEROS* where no snap-change is triggered, and *ONES* where every snap-change is triggered. Let us now present how to design online decision strategies able to outperform the baselines and approach the *offline optimal* as much as possible.

4.2. P1: Future overlap prediction

As shown in Fig. 4 and defined in Sec. 3.2, the quality in the user's FoV at any time t is given by $overlap(t) = FoV(t) \cap FoV(t-B)$. One set of inputs we feed the decision classifier with at the decision time t_{dec}, is therefore the prediction of the D values of overlap over $w_{fut2}(t_{dec})$, i.e., $FoV(t_{dec}+B+m-B) \cap FoV(t_{dec}+B+m)$ for $m \in \{0,\dots,D\}$. We consider two options to solve P1.

Past Overlap as estimate of future overlap: The D overlap values anterior to t_{dec} are considered as those over $w_{fut2}(t_{dec})$.

Future overlap estimation from FoV prediction: The deep neural network architecture named TRACK and introduced in [RRSAPP19] is used to predict the series of FoV positions between t_{dec} and $t_{dec}+B+D$, from (i) the series of FoV positions anterior to t_{dec} and (ii) the visual saliency extracted from the video content available for the entire video (streaming of pre-recorded content).

4.3. P2: Classify to decide

The goal of the decision step is to classify whether the snap-cut should be triggered (1) or not (0). A Decision Tree (DT) is used to perform this classification, as it provides a simple yet informative

structure that allows to investigate how the different configurations of the environment change the way the input features are used. The inputs are set to be as close as possible to the reward's components defined in Eq. 1: the overlap values predicted over $w_{fut2}(t_{dec})$ as described above, and input describing the relative snap position in time. The latter is made of the elapsed time since the last snap-cut triggered ($t(n) - t_{last}(n)$ in Eq. 1), and the remaining playback time after the playback time of the snap (to represent the likely impact of the snap, impacting less the cumulative reward as we reach the end of the video).

4.4. Training the Complete Framework

The training of the snap-cut decision framework is not in an end-to-end manner. TRACK is pre-trained to provide the overlap predictions, the DT decision classifier is trained on top. The dataset is split into 70% for training, 20% for testing and 10% for validation. The training, validation and test sets are the same for both overlap prediction and decision layers. We perform hyper-parameter tuning by varying the maximum tree depth in the range $[1,10)$, and selecting the depth yielding the highest F1-Score (defined as the harmonic mean of precision and recall) on the validation set. During train, the $t(n) - t_{last}(n)$ input above is fed from the *offline optimal*. However at test time, the input is determined from the previous decisions already made. The test is therefore made in an iterative process, where the previous decision will affect the future ones.

5. Results

We present here the performance in terms of reward, quality in FoV and fraction of snap-cut triggered, for the different baselines and proposed dynamic methods for different values of buffer size (B in sec.) and snap-cut penalty β. We identify the difficulties and discuss the limitations in Sec. 6.

Simulator settings: The results in this section are generated with an emulated streaming process developed in Python and serving as training and testing environment. Given the simple user's behavior model introduced in Sec. 3.1 where the user's reaction does not depend on the FoV targeted by the snap-cut, we consider equally spaced possible snap-changes to trigger, with the FoV for each picked uniformly at random in $[-180°, 180°[$. From Sec. 3.1, the snap-cut impact duration is $D = 2$s. One video segment corresponds to one second of playback. Playback starts at second 0, the first possible snap at second 1, and possible snap-changes are evenly spaced every 3s. With the 20 sec-long videos in the dataset of David et al. [DGC*18] described in Sec. 3.1, there are 7 possible snaps, occurring at times $\{1,4,7,10,13,16,19\}$.

Results on overlap prediction: As a preliminary result, Table 2 compares the error (overlap computed as orthodromic distance of the centers of the FoVs) of overlap prediction for both methods for P1 (PAST-Past overlap and TRACK), when varying the buffer size. We confirm the interest of employing a refined FoV predictor in lowering the overlap prediction error.

	B=2	B=3	B=4
PAST	40.79°	42.42°	45.54°
TRACK	36.07°	37.17°	36.74°

Table 2: *Absolute error of (TRACK) future overlap prediction using TRACK or (PAST) estimating the future overlap from the past overlap, varying the buffer size* $B \in \{2,3,4\}$.

Results on reward, quality and snap frequency:
Fig. 6 depicts the results in terms of reward, quality in the FoV and percentage of triggered snap-cuts. The values for playback buffer size B (still in sec.) and snap penalty β are the combination of $B \in \{2,3,4\}$ and $\beta \in \{0.2, 0.3, 0.35\}$. One baseline is added: *GT*, standing for Ground Truth, when the DT is fed with the GT overlap over $w_{fut2}(t_{dec})$, for every decision time. It refines the upper-bound accessible by the online methods predicting the future overlap.

First, general and expected trends can be observed from *OPT*. The optimal reward decreases when B or β increase. So does the quality. Indeed, the higher B, the lower the FoV overlap between t and $t-B$, for any time t, hence the lower the quality, hence the reward (we can trigger a snap only every 3 sec., i.e., every 3 segments, with these settings). The optimal fraction of triggers increases with B but decreases with β. Indeed, the higher B, the lower the FoV overlap, the more snap-cuts needed to get back a high quality. However the higher the snap-cut penalty β, the lower the amount of snap-cut allowed to not decrease the reward. Second, *GT* is relatively close to *OPT* when β is small, but gets away from *OPT* when β increases: this shows the need of not being myopic and considering a future horizon longer than $w_{fut2}()$ to make decisions, when the snap penalty β increases. Third, it is interesting to observe that the gap between *OPT* and the best of both *ZEROS* and *ONES*, where online methods can bring improvement, is greater for higher values of B and intermediate values of β (for example, $B = 3$ and $\beta = 0.2$, or $B = 4$ and $\beta = 0.3$). In these cases, the online methods *PAST* and *TRACK* (not assuming any knowledge of the future) are able to slightly outperform (in reward) the *ZEROS* and *ONES* baselines. However, when observing their snap-cut triggering decisions, we observe that for $\beta \geq 0.3$, they often are much more conservative than GT, triggering almost no snap-cut. In such case of high β, for high B (3 and even more so 4), the quality is much impacted by the lower accuracy of future prediction.

6. Discussion

This work defines and investigate how to learn to trigger (rotational) snap-cuts meant to jointly benefit the user and the streaming algorithm to increase the user's QoE. It is however a preliminary work, suffering from several limitations. First, in comparison with *GT*, we can observe that the performance of the proposed method *TRACK* (even more so *PAST*) is limited by the difficulty of predicting the overlap (the prediction horizon is $B+D$ sec., 4 to 6 sec. here). To overcome this difficulty, it will be important to extract less volatile features that can be exploited in deciding to trigger, particularly user or video profiles. Second, the considered model of user's reactions to a snap-cut (freezing for D sec.) and the QoE model (for the reward) are simplistic. Future work should carry out user experiments to determine accurate models, and considering complex or unknown reaction models will require to adopt other types of learning approaches (RL approaches where the test set is collected

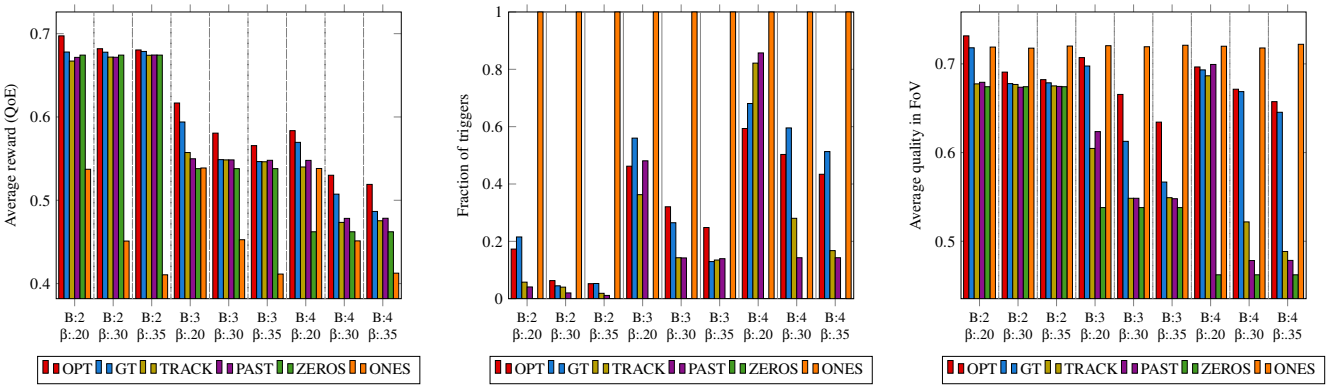

Figure 6: *(Left) Average reward (QoE), (Center) fraction of triggered snap-changes and (Right) average quality in FoV, for each method in OPT: Optimal, GT: Using the groundtruth future overlap as features, TRACK: Using the overlap computed with the prediction from TRACK [RRSAPP19], PAST: Using the overlaps before the decision, ZEROS: Never triggering a snap, ONES: Always triggering the snaps. The values are computed for each of the experiments, varying the buffer size B, and the penalty for triggering a snap-change* β.

using a behavior policy different from the target policy). Finally, the snap-cuts are modulated here to help the streaming process (using a playback buffer). However, we have considered the network state fix and the streaming logic independent of the available bandwidth. Future work will design advanced network-adaptive strategies making use of user- and content-adaptive snap-cutting strategies.

7. Conclusion

In this article, we have investigated how to learn to dynamically trigger rotational snap-cuts that re-position a user in front of new FoV, when watching a VR content streamed over the Internet. These snap-cuts can benefit the user's experience both by helping stream and improve the quality in the FoV, and ensuring the user sees RoIs important for the story plot. However, snap-cuts should not be too frequent and may be avoided when not beneficial to the streamed quality. We have formulated the snap-cut triggering optimization problem and investigated possible gains in quality of experience. We have shown that learning approaches are relevant to design online snap-cut triggering strategies to outperform baselines. Finally, we have identified the limitations of this preliminary work and the future steps to take.

References

[BBMD17] BASTUG E., BENNIS M., MEDARD M., DEBBAH M.: Toward interconnected virtual reality: Opportunities, challenges, and enablers. *IEEE Communications Magazine 55*, 6 (2017), 110–117. 1

[DGC*18] DAVID E. J., GUTIÉRREZ J., COUTROT A., DA SILVA M. P., CALLET P. L.: A dataset of head and eye movements for 360 videos. In *Proceedings of the 9th ACM Multimedia Systems Conference* (2018), pp. 432–437. 4, 6

[DSS*18a] DAMBRA S., SAMELA G., SASSATELLI L., PIGHETTI R., APARICIO-PARDO R., PINNA-DÉRY A.-M.: Film editing: New levers to improve vr streaming. In *Proceedings of the 9th ACM Multimedia Systems Conference* (New York, NY, USA, 2018), MMSys '18, ACM, pp. 27–39. URL: http://doi.acm.org/10.1145/3204949.3204962, doi:10.1145/3204949.3204962. 1, 2, 3, 4

[DSS*18b] DAMBRA S., SAMELA G., SASSATELLI L., PIGHETTI R., APARICIO-PARDO R., PINNA-DÉRY A.-M.: TOUCAN-VR. *Software* (DOI: 10.5281/zenodo.1204442 2018). URL: https://github.com/UCA4SVR/TOUCAN-VR. 2

[FLL*17] FAN C.-L., LEE J., LO W.-C., HUANG C.-Y., CHEN K.-T., HSU C.-H.: Fixation prediction for 360 video streaming in head-mounted virtual reality. In *ACM NOSSDAV* (2017), pp. 67–72. 4

[FT20] FARMANI Y., TEATHER R. J.: Evaluating discrete viewpoint control to reduce cybersickness in virtual reality. *Springer Virtual Reality* (2020). doi:https://doi.org/10.1007/s10055-020-00425-x. 2

[KLMP*17] KJÆR T., LILLELUND C. B., MOTH-POULSEN M., NILSSON N. C., NORDAHL R., SERAFIN S.: Can you cut it? an exploration of the effects of editing in cinematic virtual reality. In *Proceedings of the 23rd ACM Symposium on Virtual Reality Software and Technology* (New York, NY, USA, 2017), VRST '17, Association for Computing Machinery. URL: https://doi.org/10.1145/3139131.3139166, doi:10.1145/3139131.3139166. 2

[LCH*17] LIN Y.-C., CHANG Y.-J., HU H.-N., CHENG H.-T., HUANG C.-W., SUN M.: Tell me where to look: Investigating ways for assisting focus in 360 video. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (New York, NY, USA, 2017), CHI '17, Association for Computing Machinery, p. 2535–2545. URL: https://doi.org/10.1145/3025453.3025757, doi:10.1145/3025453.3025757. 2

[MPE14] MPEG DYNAMIC ADAPTIVE STREAMING OVER HTTP (DASH): Iso/iec 23009-1:2014, 2014. URL: https://www.iso.org/standard/65274.html. 3

[NYN18] NGUYEN A., YAN Z., NAHRSTEDT K.: Your attention is unique: Detecting 360-degree video saliency in head-mounted display for head movement prediction. In *ACM Int. Conf. on Multimedia* (2018), pp. 1190–1198. 1

[OPN*18] OSA T., PAJARINEN J., NEUMANN G., BAGNELL J., ABBEEL P., PETERS J.: An algorithmic perspective on imitation learning. *Foundations and Trends in Robotics 7*, 1-2 (2018), 1–179. 5

[PBY*19] PARK S., BHATTACHARYA A., YANG Z., DASARI M., DAS S. R., SAMARAS D.: Advancing user quality of experience in 360-degree video streaming. In *2019 IFIP Networking Conference (IFIP Networking)* (2019), pp. 1–9. 3

[PGV*09] PINEAU J., GUEZ A., VINCENT R., PANUCCIO G., AVOLI M.: Treating epilepsy via adaptive neurostimulation: a reinforcement learning approach. *International journal of neural systems 19*, 4 (2009), 227–240. 5

[PHA17] PAVEL A., HARTMANN B., AGRAWALA M.: Shot orientation controls for interactive cinematography with 360 video. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology* (New York, NY, USA, 2017), UIST '17, Association for Computing Machinery, p. 289–297. URL: https://doi.org/10.1145/3126594.3126636, doi:10.1145/3126594.3126636. 2

[RBR18] RAHIMI MOGHADAM K., BANIGAN C., RAGAN E. D.: Scene transitions and teleportation in virtual reality and the implications for spatial awareness and sickness. *IEEE Transactions on Visualization and Computer Graphics* (2018), 1–1. 2

[RHB05] RIECKE B. E., HEYDE M. V. D., BÜLTHOFF H. H.: Visual cues can be sufficient for triggering automatic, reflexlike spatial updating. *ACM Trans. Appl. Percept. 2*, 3 (July 2005), 183–215. URL: https://doi.org/10.1145/1077399.1077401, doi:10.1145/1077399.1077401. 2

[RRSAPP19] ROMERO RONDON M. F., SASSATELLI L., APARICIO PARDO R., PRECIOSO F.: Revisiting deep architectures for head motion prediction in 360 videos, 2019. arXiv:1911.11702. 1, 3, 4, 5, 7

[SMSR17] SARGUNAM S. P., MOGHADAM K. R., SUHAIL M., RAGAN E. D.: Guided head rotation and amplified head rotation: Evaluating semi-natural travel and viewing techniques in virtual reality. In *2017 IEEE Virtual Reality (VR)* (2017), pp. 19–28. 2

[SPDW*18] SASSATELLI L., PINNA-DÉRY A.-M., WINCKLER M., DAMBRA S., SAMELA G., PIGHETTI R., APARICIO-PARDO R.: Snapchanges: A dynamic editing strategy for directing viewer's attention in streaming virtual reality videos. In *Proceedings of the 2018 International Conference on Advanced Visual Interfaces* (New York, NY, USA, 2018), AVI '18, Association for Computing Machinery. URL: https://doi.org/10.1145/3206505.3206553, doi:10.1145/3206505.3206553. 2

[SWFA19] SASSATELLI L., WINCKLER M., FISICHELLA T., APARICIO R.: User-adaptive editing for 360 degree video streaming with deep reinforcement learning. In *Proceedings of the 27th ACM International Conference on Multimedia* (New York, NY, USA, 2019), MM '19, Association for Computing Machinery, p. 2208–2210. URL: https://doi.org/10.1145/3343031.3350601, doi:10.1145/3343031.3350601. 3

[YLN*17] YU R., LAGES W. S., NABIYOUNI M., RAY B., KONDUR N., CHANDRASHEKAR V., BOWMAN D. A.: Bookshelf and bird: Enabling real walking in large vr spaces through cell-based redirection. In *2017 IEEE Symposium on 3D User Interfaces (3DUI)* (2017), pp. 116–119. 2

[ZZB*19] ZHANG Y., ZHAO P., BIAN K., LIU Y., SONG L., LI X.: Drl360: 360-degree video streaming with deep reinforcement learning. In *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications* (2019), pp. 1252–1260. 3