Rate-Distortion Modeling of Synthesizer Immersive Video for 6DoF Interaction

Yuan Jun Sun
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

***Abstract*—This document is a model and instructions for LaTeX. This and the IEEEtran.cls file define the components of your paper [title, text, heads, etc.]. *CRITICAL: Do Not Use Symbols, Special Characters, Footnotes, or Math in Paper Title or Abstract.**

***Index Terms*—component, formatting, style, styling, insert**

I. Introduction

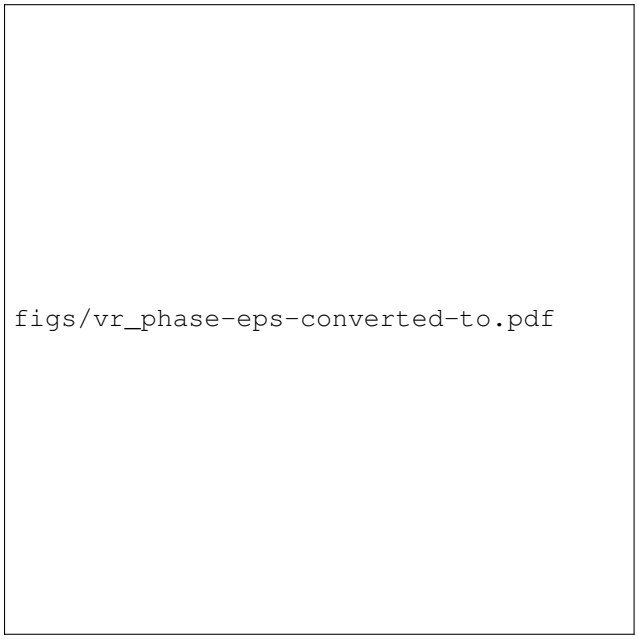


Fig. 1. Difference between 3DoF and 6DoF interactions.

Recently, Virtual Reality (VR) becomes increasingly more popular. It enables a wide array of novel applications in many domains, such as video streaming, computer games, occupational training, healthcare, manufacturing, etc. The market research also reports that foresee explosive growth of the VR market in the upcoming years [**?**]. More and more companies devote their effort to the VR industry such as Meta [**?**] or Google [**?**], [**?**].

One way to classify the VR applications is through the different *interaction techniques*, including *3DoF* and *6DoF* interactions. Because of the high popularity of 360° video streaming services, most users are familiar with the *3DoF* (Degree-of-Freedom) interactions, in which a user's viewport is determined by his/her head/HMD *orientation*. In the 3DoF VR application, the user's position in his/her coordinate such as standing up and walking around, his/her HMD viewport would not reflect the changes of positions. Therefore, the user will *not* feel he/she is moving in the virtual world, leading to an inferior *immersive* user experience. In *6DoF* interactions, the application will render the viewports depending on the user position and orientation. Different from the 3DoF interaction, the 6DoF interaction allows users to walk around the virtual world which optimizes the immersive user experience. Fig. **??** illustrates the difference between 3DoF and 6DoF interactions.

Supporting 6DoF Extended Reality (XR) using 360° videos is not an easy task, because, for every single position, a new 360° video needs to be captured. Even if we deploy dense 360° cameras, users may still miss smooth transitions at the positions between any two adjacent cameras. Hence, more descriptive 3D representations are required for enabling the truly immersive experience of 6DoF VR applications. Recently, MPEG-I (Moving Picture Expert Group - Immersive Group) has been actively developing MPEG Immersive Video (MIV) standard [**?**], [**?**] which can use for 6DoF video compression. It uses multi-view RGB-D video as the data representation and includes the integrated pipeline for encoding, decoding, synthesizing, and rendering. The Test Model for Immersive Video (TMIV) [**?**], [**?**], which is the reference software of MIV standard, has been released to show a reference implementation of MIV.

Besides, reducing bandwidth and maintaining high view quality is a bottleneck of 6DoF real-time streaming. Because of the limitation of bandwidth, performing the best quality by adaptive quality model for 6DoF immersive video is quite a challenging work. There are many existing rate control algorithms for video compression [**?**] [**?**] [**?**]. Based on the existed rate control algorithms, it is possible to optimize those models to predict the 6DoF video performance. Although, there are still many effects that could impact the 6DoF video quality, e.g., tile sizes [**?**], camera placements, complexity of scenes [**?**], number of groups, synthesizer, and a quantization parameter. In this paper, we conduct a rate-distortion model (RD-model) to predict the performance of 6DoF immersive video in common situations as we show in Fig. **??**. The model will design in an empirical way and aim at some vital

TMIV parameters. The main goal of this model is to generate relevance between TMIV parameters and quality metrics, e.g., PSRN, SSIM, and VMAF. Once the model generates, it is possible to use on estimate the 6DoF immersive video performance in different camera placements and also use it on real-time 6DoF streaming.

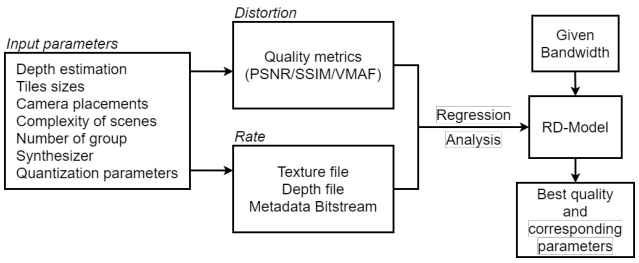


Fig. 2. RD-model workflow

II. Related Work

Objective quality assessment of live immersive video streaming has been conducted to understand how different settings affect the target view quality. For example, Cai et al. [**?**] compared the performance resulted by using different 2D video codecs to encode depth video in 6DoF streaming. Sebastian et al. [**?**] conducted a similar study on depth video compression. In contrast, Szekielda et al. [**?**] studied the implications of 2D encoding parameters on both RGB and depth videos. Software components other than 2D video codecs have also been exercised. For example, Fachada et al. [**?**] focused on the impact of different synthesizers under linear versus planar camera placement. Pre- or post-processing of the RGB and depth videos for higher coding efficiency was also investigated. For example, Jeong et al. [**?**] proposed to downsample the RGB video and upsample the depth video to increase the coding efficiency. Salahieh et al. [**?**] compared the performance of MIV reference software, called Test Model of Immersive Video (TMIV) [**?**] in the 3D domain. In particular, they considered different settings, including *single-* versus *multi-pass synthesis* and *MIV* versus *MIV View modes*. Last, the impacts of different camera (source view) densities on synthesized target view quality were evaluated in Ray et al. [**?**]. They concluded that higher camera densities lead to better perceived quality. Compared to our work, the aforementioned studies have two major limitation. First, almost all of them (except Ray et al. [**?**]) employed existing scene/trajectory datasets. Second, none of them considered *continuous* 6DoF trajectories of source *and* target views. The closest setup to continuous 6DoF trajectories was the *discrete* camera locations/orientations adopted in Fachada et al. [**?**] and Ray et al. [**?**]. *Our AirSim-based data collection tools offer the opportunity to generate large and flexible datasets with real 6DoF source/target trajectories.*

A. 2D videos

B. 360 videos

C. Synthesized video

Yangang Cai [**?**] compressed the depth map by AVC, HEVC, and AVS3. Their results show that using the AVS3 encoder to compress the depth maps can provide better virtual view performance and less bitrate. Basel Salahieh [**?**] evaluate the performance of the object-based solution. Their results show the pixel rate saving and bitrate distortion in the object-based situation. Xavier Corbillon [**?**] implemented a 6DoF VR application with a multi-camera system and analyzed two extreme optimal algorithms. Their results show that tiling is able to improve the service performance and the high cost for the proactive optimizing strategies By the previous experiments, both 6DoF video quality and bitrate have a negative correlation with QP.

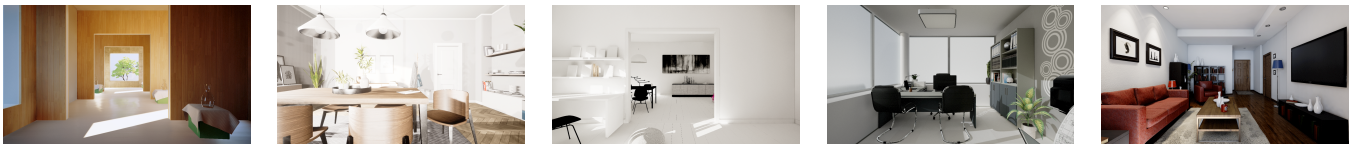
III. Data collection

A. Implementations

The purpose of collecting our own dataset is to investigate the impacts of diverse settings on the synthesized target views. This dataset has to cover a wide variety of usage scenarios with: (i) different camera placement strategies, (ii) random target view trajectories mimicking real HMD users, (iii) scenes with diverse characteristics (e.g., lighting conditions, color tone, and dynamics), we capture the source views by extending AirSim [**?**], which is an open source project that provides Application Programming Interface (API) for programmers. For example, we develop tools using camera control API to capture source views. Moreover, we adopt the random waypoint as the mobility model to generate random target view trajectories. In addition to random trajectories, we also implement tools to collect real HMD user trajectories using Unreal Engine. In a pilot test, we recruit three HMD users, play a random scene to them, and analyze their target view trajectories. We found that the dynamic ranges of roll/pitch/yaw are 80/80/100 degrees, and the angular velocity is between 8 and 80 degree/s. We also found the HMD users moves at a speed between 0.1 and 1 m/s. We use these statistics to generate random target view trajectories.

B. The Dataset

We select five scenes from Unreal Engine marketplace [**?**], and modify one of them into a dynamic scene. Fig. **??** gives sample video frame of these scenes (except the dynamic one). Table **??** summarizes the scenes with key characteristics. The synthesized target views from these scenes exhibit different complexities in terms of Temporal Information (TI) and Spatial Information (SI). The dynamic scene, *RealD*, is generated by adding a howling wolf and a rolling ball to *Real*. For each scene, we manually choose direction that have the richest set of visual features. We then place 36 cameras with one of the four placements: 6X6, 9X4, 12X3, and 18X2. Following MPEG's recommendations, all cameras and trajectories are confined in a $0.35 \times 1 \times 1\ m^3$ bounding box. We set the resolution and



(a) (b) (c) (d) (e)

Fig. 3. The considered scenes sorted in increasing complexity levels: (a) *Light*, (b) *Arch*, (c) *Xoio*, (d) *Office*, and (e) *Real*.

Field-of-View (FoV) of each camera to be 1280×720 and 90°, respectively. We generate 10 target view trajectories: t_1 to t_{10} using the random waypoint model. We also selected a real HMD user's trajectory from our pilot test that covers the largest surface of the scene. We refer to it as u_1. Each trajectory contains 90 samples of positions and orientations at 30 Hz.

TABLE I
SCENES IN OUR DATASET

Scene	# Obj.	# Mesh.	Space	Lighting	Color Tone	TI	SI
Light	52	51.3 K	Narrow	Bright	Warm	19.1	35.0
Arch	282	5.5 M	Wide	Bright	Warm	27.1	57.4
Xoio	125	2.8 M	Wide	Bright	Cold	26.2	57.8
Office	96	100.4 K	Narrow	Dark	Cold	29.6	62.2
Real	352	221.1 K	Narrow	Medium	Warm	34.7	66.7

One last complication is the different coordinate systems used by: (i) Unreal Engine (North-Eastern-up, left-handed), (ii) AirSim (North-Eastern-down, right-handed), and (iii) TMIV (North-Western-up, right-handed). We have implemented scripts to convert the coordinate systems. For each combination of the scene and camera placement (trajectories), we capture RGBD videos from individual cameras, which are *source views*. We also captured the RGBD video clips following individual target view trajectories, which are *target views*, or ground truth. Both source and target views are stored as raw videos. In summary, our dataset contains: (i) source view placements (trajectories), (ii) target view (trajectories), (iii) source views, and (iv) target views. We plan to make our tools and sample dataset public.

IV. MPEG IMMERSIVE VIDEO STANDARD

In this section, we briefly introduce the workflow and components of MIV codec [**?**], [**?**].

Fig. **??** shows the high-level workflow of MIV encoder. The inputs of MIV encoder are *source views*. Each source view is composed of attribute (texture) videos, geometric (depth) videos, and camera parameters. MIV encoder do the following process to compress source views:

- **Automatic parameter selection.** MIV encoder automatically calculate the parameters for compression, e.g., assessing geometric video quality, splitting source views into multiple group according to configuration, and labeling source views in each group.
- **Single-group encoders.** MIV encoder encodes each group of source views separately. In each group, the encoder chooses several views as the basic view according to the label of source view, and remove the duplicate area in other source views. The basic view and remaining area of other views are packed into rectangle video frames, which are called *atlases*. Fig. **??** show the example of atlases.

The outputs of MIV encoder are attribute atlases, geometric atlases, and metadata bitstream. The atlases are further compressed by video codec, and multiplexed with metadata bitstream as a single bitstream.

Fig. **??** shows the high-level workflow of MIV decoder. The inputs pf MIV decoder is the bitstream contains atlases bitstream, and metadata bitstream. The video decoder first be employed to decompress attribute atlases and geometric atlases. After that MIV decoder do the following process to decompress altases and synthesize the user's viewport.

- **Reconstruction process.** The MIV decoder reconstruct the source view by using the data in atlases.
- **Synthesizer.** The MIV decoder employ view synthesis techniques to synthesize the user's viewport according to user's position and orientation. Specifically, the synthesizer warp the pixel of each source view to user's viewport according to depth information, and blending the pixel values from each source views.
- **Inpainting.** After synthesis, the synthesized result may contain holes without information. The inpainting process uses the information from neighbor pixels to calculate pixel value for holes.

The outputs of MIV decoder are user's viewport synthesized according to user's position and orientation.

V. RD-MODELING

The target of our model is to predict the objective video quality which we choose VMAF as our main quality metric. VMAF is a quality metric developed by Netflix. It predicted the video quality by existing image metrics and some other features, such as Visual Information Fidelity (VIF), Detail Loss Metric (DLM), and Mean Co-Located Pixel Difference (MCPD).

We vary the following inputs of the model in our experiments:

- **QP** trades off the bitrate and RGBD video quality. MPEG group suggests using a smaller QP for depth video (80% of the corresponding RGB videos) because depth imposes significant impacts on synthesized quality [**?**]. We set (RGB video) QP $\in \{20, 36, 44, 48, 50\}$, targeting 5 to 50 Mbps total bitrates.

(a)

(b)

Fig. 4. The high-level overview of process flow of TMIV: (a) Encoder and (b) Decoder.

- **Target view trajectory** includes: (i) yaw, roll, and pitch, which specify the user *orientation* and (ii) surge, heave, and sway, which specify the user *position*.

Fig. 5. The example of atlases. The left picture is attribute atlas, and the right picture is geometric altas.