Deep Variational Learning for 360° Adaptive Streaming

QUENTIN GUIMARD, Université Côte d'Azur, CNRS, I3S, France
LUCILE SASSATELLI, Université Côte d'Azur, CNRS, I3S, Institut Universitaire de France, France
FRANCESCO MARCHETTI, Università degli Studi di Firenze, MICC, Italy
FEDERICO BECATTINI, Università degli Studi di Firenze, MICC, Italy
LORENZO SEIDENARI, Università degli Studi di Firenze, MICC, Italy
ALBERTO DEL BIMBO, Università degli Studi di Firenze, MICC, Italy

Prediction of head movements in immersive media is key to designing efficient streaming systems able to focus the bandwidth budget on visible areas of the content. However, most of the numerous proposals made to predict user head motion in 360° images and videos do not explicitly consider a prominent characteristic of the head motion data: its intrinsic uncertainty. In this article, we present an approach to generate multiple plausible futures of head motion in 360° videos, given a common past trajectory. To our knowledge, this is the first work that considers the problem of multiple head motion prediction for 360° video streaming. We introduce our discrete variational multiple sequence (DVMS) learning framework, which builds on deep latent variable models. We design a training procedure to obtain a flexible, lightweight stochastic prediction model compatible with sequence-to-sequence neural architectures. Experimental results on 4 different datasets show that our method DVMS outperforms competitors adapted from the self-driving domain by up to 41% on prediction horizons up to 5 sec., at lower computational and memory costs. To understand how the learned features account for the motion uncertainty, we analyze the structure of the learned latent space and connect it with the physical properties of the trajectories. We also introduce a method to estimate the likelihood of each generated trajectory, enabling the integration of DVMS in a streaming system. We hence deploy an extensive evaluation of the interest of our DVMS proposal for a streaming system. To do so, we first introduce a new Python-based 360° streaming simulator that we make available to the community. On real-world user, video, and networking data, we show that predicting multiple trajectories yields higher fairness between the traces, the gains for 20 to 30% of the users reaching up to 10% in visual quality for the best number K of trajectories to generate.

CCS Concepts: • **Human-centered computing** → **Virtual reality**; • **Information systems** → **Multimedia streaming**; • **Computing methodologies** → **Neural networks**.

Additional Key Words and Phrases: 360° videos, head motion, trajectory prediction, deep learning

1 INTRODUCTION

High-quality access to immersive environments, such as the Metaverse, is gaining traction in research interests. Embodied experiences through, e.g., Virtual Reality (VR), indeed require high level of system performance over multiple metrics, including quality in the field of view (FoV) and low motion-to-photon latency. Under constraining network conditions where wireless links or Internet congestion limit the available bandwidth and possibly increase latency, a general principle to optimize resource utilization is to focus the bandwidth budget

Authors' addresses: Quentin Guimard, quentin.guimard@univ-cotedazur.fr, Université Côte d'Azur, CNRS, I3S, Sophia Antipolis, France; Lucile Sassatelli, Université Côte d'Azur, CNRS, I3S, Institut Universitaire de France, Sophia Antipolis, France; Francesco Marchetti, Università degli Studi di Firenze, MICC, Firenze, Italy; Federico Becattini, Università degli Studi di Firenze, MICC, Firenze, Italy; Lorenzo Seidenari, Università degli Studi di Firenze, MICC, Firenze, Italy; Alberto Del Bimbo, Università degli Studi di Firenze, MICC, Firenze, Italy.

ACM 1551-6857/2024/1-ART
https://doi.org/10.1145/3643031

into the current FoV, and allow low-quality rendering elsewhere. In a seemingly simple modality such as 360° videos where the user can only rotate their head in 3 degrees of freedom, realizing this principle is already non trivial. The straight-forward solution is for the server to be informed by the client of the current head position and to immediately send the required sections of the sphere in high quality. However, this prevents the use of a (more than one second-) playback buffer at the client side, as the human user may have changed FoV position at a later playback time. However, playback buffers are a key ingredient of modern streaming systems to absorb network quality variations for the codec. Another solution is therefore to enable a longer playback buffer, of at least few seconds (we take 5 to 10 seconds in this article), but requires to predict where the user's FoV will be in the next few seconds. In this article, we present a solution to realize this approach.

Several prediction models have been published to attempt predicting head or gaze motion, yet mostly over time horizons less than 2 sec. and the vast majority of these models do not explicitly account for a constitutive component of human attention, which is its uncertainty. Very few models have considered this characteristic so far [9, 18, 48], but only heuristically for 360° videos. We illustrate this uncertainty in Fig. 1a, showing that close past trajectories often lead to diverse/distant future trajectories. This has long been identified in other application domains such as autonomous driving [3, 24] or human pose estimation [34]. Such an ambiguity in the data (a same input may be mapped to several outputs) leads to degraded performance and over-fitting. Considering uncertainty in optimization of resource allocation is therefore key to improve systems' performance, as shown in robotic planning [16] and regular video streaming considering bandwidth uncertainty [21, 47]. In the case of 360° video streaming, predicting a single trajectory concentrates the available bandwidth budget into a specific area, leading to high quality when the prediction is accurate, but low quality when the movements become unpredictable. Predicting multiple trajectories increases fairness by covering more area, accounting for the diversity of possible future viewports.

In this article, we introduce how we can predict multiple plausible trajectories and how 360° streaming can benefit from it. This article extends our previous publication [14]. We make three contributions:

• Motivated by our analysis of the difficulty of prediction on existing datasets, we present our discrete variational multiple sequence (DVMS) learning framework, which produces several plausible future continuation trajectories from a given past trajectory. It is applicable to the general family of encoder-decoder neural network architectures, and relies on the introduction of a latent variable z to condition the decoding of the past into multiple futures. We motivate our choices of training loss and procedure. We additionally introduce a method to estimate likelihoods of the generated future trajectories, which is leveraged in the streaming logic. We then assess the quality of prediction on the best-of-many performance metric, and show that our method DVMS outperforms competitors adapted from the self-driving domain by up to 41% on prediction horizons up to 5 sec., at lower computational and memory costs.

• [*New contribution*] We provide a detailed analysis of both the learned latent space where the encoding of the past trajectories lies, and of the influence of z on the connection between past and predicted trajectories. For both analyses, we connect latent space locations and values of z with physical properties.

• [*New contribution*] We then deploy an extensive evaluation of the interest of our DVMS proposal for a streaming system. To do so, we first introduce a new Python-based 360° streaming simulator that we make available to the community. We then provide an in-depth analysis of the performance of DVMS prediction when incorporated in a streaming system, obtaining results from nearly 5 million simulations using 3378 head motion traces from 132 different users watching 94 different videos, 40 different network traces with 5 different buffer settings and 7 viewport prediction algorithms, including state-of-the-art competitors and variants of DVMS. Our results show that predicting multiple trajectories (under constant bandwidth budget) yields a higher fairness between the traces of the user-video pairs, with less traces with the worst quality of experience (QoE) level, and a close (resp. slightly higher) number of traces with the maximum QoE when predicting with $K = 5$ (resp. choosing the best K

per trace) futures with DVMS. We also quantify that choosing the best K yields up to a 10% higher quality in the FoV (up to a 5% better QoE) for the 20% to 30% of traces with the highest prediction errors.

The article is organized as follows. Sec. 2 reviews major related works for uncertainty-based prediction 360° adaptive streaming. Sec. 3 presents an analysis of the prediction difficulty on reference dataset and prediction models. Sec. 4 presents our DVMS learning framework and its performance assessment. Sec. 5 details the analysis of the learned encoding latent space as well as the influence of the latent variable on the generated trajectories. Sec. 6 details the integration of DVMS into an adaptive streaming logic, motivating our design and implementation choices with existing works, and presenting our Python-based simulator created to remedy weaknesses of existing tools, and made available to the community. We detail the simulation settings and thoroughly analyze the simulation results to quantify the interest of DVMS on various conditions and metrics. Sec. 7 discusses the novelty and impact of DVMS, as well as possible improvements. Sec. 8 concludes the article and draws perspectives.

2 RELATED WORK

We position our work in two respects. We first review trajectory prediction for 360° adaptive streaming incorporating uncertainty consideration. We then provide a concise review of existing 360° streaming simulators. For an additional review of point-wise head or gaze motion prediction, we refer to [14, Sec 2.1].

2.1 Uncertainty-based prediction

How users explore in VR and what commonalities do their viewing patterns exhibit have fostered a lot of interest in the last few years [1, 8, 37]. Almquist et al. [1] showed that the viewing congruence heavily depends on the type of scene, while other works [8, 37] have shown that, upon entering a new scene, the user first goes through an exploration phase where movements are not strongly correlated with the visual content.

To study and cope with user movement uncertainty, several approaches have relied on hand-crafted adaptations [9, 18, 50]. In contrast to these works relying on single trajectory prediction trying to consider the error distribution around a single mode, our method provides diverse trajectories by design, in addition to their estimated likelihoods.

Recent works have presented deep learning approaches to consider prediction uncertainty [21, 47]. In contrast with the vast majority of approaches considering point-wise estimates of future bandwidth for adaptive streaming, both consider the uncertainty of bandwidth prediction in the decision problem of what encoding rate to choose for the next video chunks to send. Both derive probability distribution of the future throughputs, that they feed into an MPC algorithm. Yan et al. [47] designed a neural network to output a discretized probability distribution of predicted transmission times. Kan et al. [21] considered Bayesian neural networks (BNN) to output the probability distribution of future throughput, given the network's historical throughput. In a very recent work, Yang et al. [48] considered predicting multiple head trajectories but only for 360° images, not videos as we do. They consider head trajectory as a succession of fixations and saccades, and intend to learn to capture the uncertainty of head trajectories across different subjects. They resort to a Bayesian neural networks (BNN) approach, to predict, given an input 360° image, multiple head trajectories by sampling the weights of the neural network predictor, the inter-subject variance being modeled with a latent variable conditioning the weight distribution. This approach is the closest to our work, but it differs from ours in several aspects. It considers 360° images, not videos as we do. It generates trajectories for the entire viewing duration, and is meant to model the intrinsic variability between the users, generating the trajectory uncertainty. In our work, we generate future trajectories online over a prediction horizon of 5 seconds and considering past motion of the current user only. We therefore cope not only with inter-user variability, but also with intrinsic uncertainty of the data in how past is correlated with future motion, data uncertainty often referred to as aleatoric uncertainty. Also, BNN are computationally-heavy (the approach from Yang et al. [48] is not real-time) and fit accurately but to just one mode in the data [10]. In this article, we

consider a lightweight approach to multiple trajectory prediction, able to predict multiple modes for the future trajectory.

Tracking and trajectory prediction have been historically tackled with model-based approaches, such as Kalman filters [5], particle filtering [17, 31, 36], or other probabilistic methods [43]. More recently, many deep learning approaches to multiple trajectory prediction to forecast future positions of moving agents such as cars and pedestrians have been proposed. Compared to head motion prediction, where predictions are guided by content and user attitude, trajectory forecasting is a more constrained task due to social behavioral rules [15, 19, 49], inertia of moving agents and environmental constraints [3, 6, 25, 40]. Nonetheless, the ability to forecast a multimodal prediction is of fundamental importance for planning secure trajectories for autonomous vehicles. Due to space limitations, we refer to [14, Sec. 2.2] for a more detailed review of the literature in multiple trajectory prediction in robotics. In the present article, we leverage this domain knowledge by considering the variety loss [15] to enable the training of our DVMS model aiming to produce diverse plausible trajectories.

2.2 360° streaming simulators

Several tools have been made available in recent years in an effort to improve reproducibility in this field. Ribezzo et al. [30] released TAPAS-360° , an open-source emulator that enables designing and experimenting omnidirectional video streaming algorithms. Unfortunately, TAPAS-360° does not support tile-based streaming, but works with a set of pre-defined "views". This makes it impossible to use with tile-based bitrate adaptation algorithms, which are the most common type of bitrate adaptation algorithms for 360° video streaming. Jiang et al. [20] provide code for simulating 360° bitrate adaptation and motion prediction along with Plato , but the lack of documentation and obscure file structure makes it difficult to use, precluding other researchers from using it and test new algorithms. Spiteri [38] released , a simulation testbed for 360° videos as an extension of Sabre [39], an open-source simulation environment for ABR algorithms. While Sabre360 can be used to compare adaptive bitrate algorithms, it has some drawbacks, including an approximated considerations of stalls, impossibility to change the tiling scheme and unrealistic segment request procedure. Our simulator takes a lot of inspiration from Sabre360, which we consider to be the closest solution to the problem we want to solve. Our work aims at rectifying any shortcomings the existing solutions may have for comparing motion prediction and adaptive bitrate strategies in the context of 360° streaming.

3 MOTIVATION BEHIND MULTIPLE PREDICTION OF HEAD TRAJECTORIES

We first outline the 360° adaptive streaming problem we aim to solve in Sec. 3.1, then formally define the head motion prediction problem in Sec. 3.2. In Sec. 3.3, we analyze head motion data to quantify the diversity of futures corresponding to similar past trajectories, and show the need for multiple future predictions.

3.1 360° adaptive streaming problem

The core motivation for our contribution is to improve adaptive streaming for 360° videos by taking into account randomness of the environment in the optimization of resource allocation. Specifically, considering the optimization of spatial heterogeneous quality in streaming 360° videos, one has to consider the variations of network bandwidth and human head position, which both cannot be predicted perfectly. The primary objective of our study is to maximize the quality of experience (QoE) for users under bandwidth constraints, as formulated in the following equations. Such stochastic optimization can generally be approached in two ways. First, RL-based approaches [23, 44] do not split the problem into environment prediction and resource allocation, but rather tackle it end-to-end. Other recent works show the benefit, for regular video streaming [21, 47], of splitting the problem and designing a DNN to produce stochastic predictions of bandwidth, which are then considered as parameters in model predictive control (MPC). For example, Yan et al. [47] used dynamic programming to

maximize the expected cumulative QoE as shown in Eq. 1, where H is the look-ahead horizon for download, B_j is the playback buffer's level at chunk j, $QoE(\cdot)$ is the QoE function, K_i^s is chunk i in quality s, and $T(K_j^s)$ is the stochastic download time of this chunk.

$$\max_{K_i^s,\ldots,K_{i+H-1}^s} \sum_{j=i}^{i+H-1} \sum_{t_j} Pr[T(K_j^s) = t_j] QoE\left(K_j^s, K_{j-1}^s, B_j, t_j\right) \tag{1}$$

$$\max_{\{K_{i,l}^s\}_l,\ldots,\{K_{i+H-1,l}^s\}_l} \sum_{j=i}^{i+H-1} \sum_{l=1}^{L} \sum_{t_j} Pr[l \in FoV(j)] Pr[T(K_j^s) = t_j] QoE\left(\{K_{j,l}^s\}_l, \{K_{j-1,l}^s\}_l, B_j, t_j\right) \tag{2}$$

Such formulation enables buffering of H chunks to absorb bandwidth variations. Kan et al. [21] formulated this optimization by projecting the estimated bandwidth distribution onto a confidence interval. In the case of 360° streaming, the equivalent problem can be formulated, incorporating the distribution of the FoV position over the look-head horizon [35] as shown in Eq. 2, with $l \in \{1, L\}$ denoting the tile index, if we consider a tile-based formulation.

In this article, we provide a stochastic tool, the DVMS learning framework presented in Sec. 4.1, to estimate the distribution $Pr\,[l \in FoV(j)]$. To do so, we make a proposal to predict several K trajectories (series of centers of FoV) $\mathbf{y}_{t:t+H}^k$, for $k \in \{1, K\}$, with their estimated likelihood $Pr[\mathbf{y}_{t:t+H}^k | \mathbf{x}_{0:t}]$. If the problem is tile-based as above, then we can obtain $Pr\,[l \in FoV(j)]$ in Eq. 3.

$$Pr\,[l \in FoV(j)] = \sum_{k=1}^{K} Pr[l \in FoV(j) | \mathbf{y}_{i:i+H-1}^k] Pr[\mathbf{y}_{i:i+H-1}^k | \mathbf{x}_{0:i}] = \sum_{k:l \in \text{ FoV of center } y_j^k} Pr[\mathbf{y}_{i:i+H-1}^k | \mathbf{x}_{0:i}] \tag{3}$$

Once we have at our disposal multiple trajectory estimates and their respective likelihoods, we can express the distribution of the FoV position as a function of these estimates as seen in Eq. 3. This distribution can in turn be used in conjunction with the appropriate QoE function by an adaptive bitrate (ABR) algorithm to solve the optimization problem as formulated in Eq. 2.

3.2 Head motion prediction problem

The problem we consider is formally described as follows. We consider that a given 360° video v of duration T seconds is being watched by a user u. The head trajectory of the user is denoted $\mathbf{x}_{0:T}^{u,v}$, with $\mathbf{x}$ storing the head coordinates on the unit sphere.

Online single prediction problem: At any time t in $[0, T]$, predict $\mathbf{x}_{t:t+H}^{u,v}$ with an estimate $\mathbf{y}_{t:t+H}^{u,v}$, that is predict the future trajectory over a prediction horizon H, assuming only $\mathbf{x}_{0:t}^{u,v}$ is known. That is, we do not assume any knowledge of traces other than u on this video v. Hence, for lighter notations, we drop indices u and v from $\mathbf{x}_{0:t}^{u,v}$ and only write $\mathbf{x}_{0:t}$ and $\mathbf{y}_{0:t}$.

Online multiple future prediction problem: At any time t in $[0, T]$, predict K possible future trajectories $\mathbf{y}_{t:t+H}^k$, for $k = 1, \ldots, K$, to estimate $\mathbf{x}_{t:t+H}$. This is the general problem definition considered in related work [2, 4]. However, for optimization of heterogeneous quality decisions in a video streaming system, it is also important to estimate the likelihood of every such possible future trajectory. We therefore augment the multiple future prediction problem with estimation of likelihood $Pr[\mathbf{y}_{t:t+H}^k | \mathbf{x}_{0:t}]$. We propose a variational model in Sec. 4.1 to address this issue.

3.3 Analysis of the need for multiple prediction in head motion data

We now analyze the need for multiple prediction from two perspectives: from the data only, and from the performance of a given predictor on this data. We consider the test data of the MMSys18 dataset, described in

Sec. 4.3. In what follows, past (resp. future) trajectories are considered over a horizon of 1 sec. (resp. 5 sec.) as done in recent work [7, 33]. We exclude the 5% shortest past trajectories and the 5% shortest future trajectories from this analysis, as they may skew the distance calculations between pairs of trajectories.

Distance metric: We compare two trajectories P_1 and P_2 of equal length L using the average point-wise great-circle distance, defined in Eq. 4. We consider pairs of trajectories with the lowest distance to be the closest to each other.

$$d(P_1, P_2) = \frac{2}{L} \cdot \sum_{p_1 \in P_1, p_2 \in P_2} \arcsin\left(\frac{\|p_1 - p_2\|_2}{2}\right) \tag{4}$$

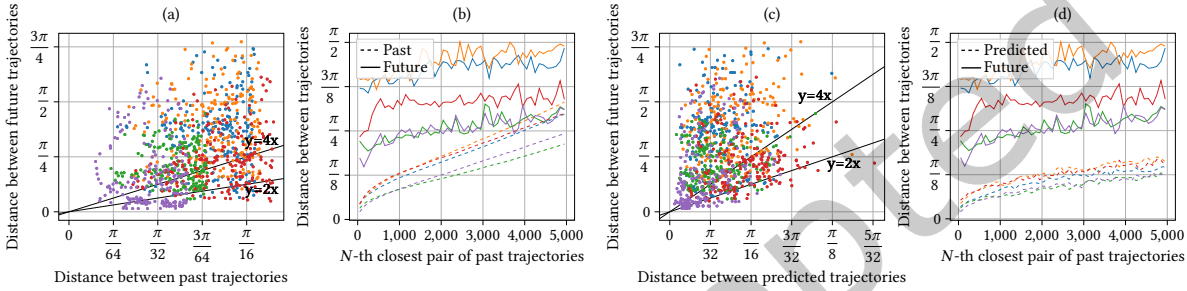


Fig. 1. Distances between pairs of past, future, and predicted trajectories for pairs of close past trajectories on the test videos of the MMSys18 dataset. The colors are associated with the video IDs and are the following: blue: PortoRiverside, orange: PlanEnergyBioLab, green: Waterpark, red: Warship, purple: Turtle.

We investigate how the distance between past trajectories relates to the distance between their corresponding true futures. To do so, for each timestamp of each video in the dataset, we consider all pairs of users, and select 200 pairs per video with the closest past trajectories. Every pair of users yields the distance between both past trajectories, and the distance between both respective true future trajectories. Fig. 1a represents the scatter plot of both distances for every pair. We observe that, for 200 pairs of closest past trajectories per video, **90%** of the corresponding future pairs have a distance more than twice the distance between their past trajectories (above the $y = 2x$ line). Also, we observe that for close past elements, more distant futures are produced, on this dataset, for exploration-type videos *PortoRiverside* and *PlanEnergyBioLab*. Specifically, 81% of the points are above the $y = 4x$ line for *PortoRiverside* and 85% of the points are above the $y = 4x$ line for *PlanEnergyBioLab*. Fig. 1b represents the distance between past elements and the distance between future elements, for every N-th closest pair, with $N \leq 5000$ (distances are smoothed with a moving average). It confirms that the average future distance is generally higher that the past distance, with a greater difference obtained for exploration videos.

This is an indication that relatively close past trajectories may lead to distinct/farther apart future trajectories, which may create difficulties when attempting to train a prediction model on such data. Indeed, a (neural) regressor trained with the regular mean square error (MSE) cannot map similar inputs to different outputs. We refer to [14, Sec. 3.3] showing that the diversity of ground truth data is not properly reproduced by a reference predictor considering both past motion and visual content. These two findings provide solid rationale for designing multiple trajectory prediction methods.

4 DEEP STOCHASTIC PREDICTION OF MULTIPLE HEAD TRAJECTORIES

We first present our proposal for a multiple prediction framework in Sec. 4.1, exemplified with an architecture in Sec. 4.2, and prediction performance resulted are presented in Sec. 4.3.

4.1 Discrete Variational Multiple Sequence (DVMS) prediction

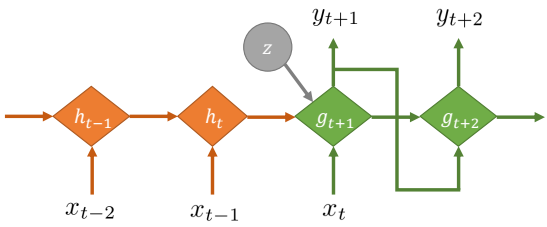


Fig. 2. Probabilistic graphical model of the proposed stochastic discrete variational multiple sequence (DVMS) prediction framework. A random variable is represented with a circle, a deterministic state with a diamond.

We present a new learning framework for multiple head motion trajectory prediction, named discrete variational multiple sequence (DVMS). In contrast to particle filtering, which approximates the posterior distribution in a sampling-based manner and considers a fully stochastic generative model where the state variable is stochastic and changes every time step, DVMS does not learn an approximate posterior of the state variable and does not consider a fully stochastic generative model. It builds on deep latent variable models like VAEs [22, 29]. We refer to [14, Sec. 4.1] for a background on deep generative models for sequences, which we could not include here due to space limitations. DVMS is designed to be compatible with any sequence-to-sequence architecture. The rationale for such design is as follows. Our goal is to design a framework for multiple prediction of head motion for deep architectures, which provides key properties:

P1 sufficiently diverse predictions $\mathbf{y}^k_{t:t+H}$, for $k = 1, \ldots, K$,
P2 state-of-the-art performance when $K = 1$,
P3 estimates of likelihoods of the predicted trajectories,
P4 flexibility and low computational cost.

Generative model: The probabilistic graphical model of DVMS is depicted in Fig. 2. For any encoder fed with past sequence $\mathbf{x}_{0:t}$, an embedding h_t is produced. This embedding is then concatenated with a unique latent variable z. The latent variable is key in our DVMS proposal. As opposed to using a singular model with increased variance, different values of the latent variable are meant to capture the diversity present in different *modes* of trajectories. The resulting concatenation produces the first hidden state g_t of the decoder. Considering that the encoder is made of recurrent connections with hidden state h_t, the generative model writes as Eq. 5, where $\mathbb{U}_{\mathcal{Z}_K}$ denotes the uniform distribution over discrete set $\mathcal{Z}_K$, and MLP stands for multi-layer perceptron to denote one or several fully connected (FC) layers. To generate multiple prediction, every $z_k \in \mathcal{Z}_K$ generates a future trajectory $\mathbf{y}^k_{t:t+H}$. To enable diverse predictions (P1), we do not constrain the distribution $p(z)$ we sample from to be conditioned on $x_{0:t}$ in test, in contrast to what was done by Alahi et al. [28], but instead draw z uniformly in $\mathcal{Z} \in [-1, 1]^d$ (where d is the dimension of vector z). To meet (P3), z is drawn from a discrete set $\mathcal{Z}$ with K elements. The same discrete set of values $\mathcal{Z}_K$ is then used at test time, leading to a deterministic model. If there is some stationarity in how likely is every trajectory produced from every z_k, then we can exploit this stationarity for likelihood estimation (P3). We therefore consider a discrete fixed set of possible z values to ease

this exploitation, which we describe in Sec. 5.2.

$$\begin{aligned} h_t &= \text{RNN}_{enc}\left(h_{t-1}, \mathbf{x}_{t-1}\right), \quad h_0 = 0 \\ z &\sim \mathbb{U}_{\mathcal{Z}_K} \\ g_{t+1} &= \text{MLP}(h_t, z) \\ \mathbf{y}_t &= \mathbf{x}_t \\ \mathbf{y}_{t+s} &= \text{FC}(g_{t+s}) + \mathbf{y}_{t+s-1}, \quad \text{for } s \geq 1 \\ g_{t+s} &= \text{RNN}_{dec}\left(g_{t+s-1}, \mathbf{y}_{t+s-1}\right), \quad \text{for } s \geq 2 \end{aligned} \tag{5}$$

Training procedure: To ensure (P2), we enforce the prior distribution $p(z)$ z is sampled from at training time to be the same as in test (contrary to work from Babaeizadeh et al. [2]), i.e., we do not consider an inference network. This allows to avoid the mismatch between $p(z)$ and $q(z|x_{0:T})$, which impedes the training convergence. However, doing so also adds noise to the sequence decoder which, if trained with gradient descent performed over every sampled trajectory obtained from z_k, for all $k \in \{1, K\}$, learns to discard the z input and only produces a single trajectory corresponding to the baseline, as described by Babaeizadeh et al. [2]. To avoid this phenomenon, we instead train our architecture with the *best of many samples* (BMS) loss [4], also named the *variety loss* [15, 41], defined in Eq. 6.

$$\mathcal{L}(\mathbf{x}_{0:t}, \theta) = \min_{k \in \{1,K\}} D\left(\mathbf{y}^k_{t:t+H}, \mathbf{x}_{t:t+H}\right) \tag{6}$$

where $D(\cdot)$ can be any distance between two trajectories on the sphere. This loss thus consists, for every past trajectory sample, in selecting sample z_{k^*} generating the best match to the single ground truth future. The gradient descent is hence performed only on a single k^* sample out of the trajectories generated by the model. With the variety loss, the model learns an efficient mapping between each z_k and a mode of trajectory. This reduces the variance by focusing the prediction around the different modes and prevents the architecture from learning to discard z as being an uninformative input for prediction.

DVMS is flexible (P4) because it can be used with any sequence-to-sequence architecture, being it an architecture processing video content [33] in case of streaming of stored content, or an architecture processing only the past user's trajectory [7] in case of live streaming. Indeed, Bayesian methods like BNN [26] and Monte-Carlo dropout [11] require to change how every network weight is considered in train (generating multiple weight samples). In contrast, DVMS only consists in adding a latent variable to modulate the initial state of the sequence-to-sequence decoder with a random component, independently of the actual structure of the sequence-to-sequence encoder and decoder.

DVMS is also lightweight (P4) because the additional training cost, w.r.t. the original sequence-to-sequence architecture, only comes from the latent variable z to be concatenated with the encoder's last hidden state (MLP to learn in Eq. 5). This additional cost is also limited because we do not learn an approximate posterior $q(z|\mathbf{x}_{0:t})$, that is an additional neural network (named *inference network* and used only in train), but rather directly sample z from $\mathcal{U}_{\mathcal{Z}_K}$ both in test and train.

All four properties (P1)-(P4) are experimentally evaluated in Sec. 4.3 .

4.2 Proposal of a DVMS-based architecture

To demonstrate the interest of the proposed DVMS learning framework of multiple head trajectory prediction, in this section we propose a simple architecture akin to those presented in [28, Fig. 2] or [33, Fig. 4]. This architecture is of type sequence-to-sequence and is represented in Fig. 3. It is however simplified compared to the previous literature, as we consider double-stacked gated recurrent units (GRU) instead of single or double-stacked LSTM. Here we purposefully do not consider the visual content in order to simplify the presentation and analysis of our contribution which is on the variational framework DVMS for multiple future prediction, and not on a

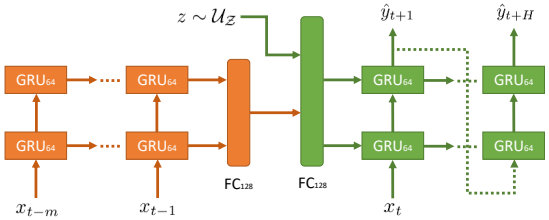


Fig. 3. Proposed example of a DVMS-based architecture.

specific neural architecture. So other architectures can be incorporated in our framework, such as based on more advanced recurrent techniques like transformers [7] or fusion of multimodal input considering the visual content [33]. We discuss more this compatibility in Sec. 7.

Architecture: We set $d = 1$ as the dimension of z. The encoder is made of a doubly-stacked GRU with 64 neurons (and default GRU activations). The final GRU's hidden state is then fed to a 128-neuron fully connected layer. The output of this layer is concatenated with z and fed to another 128-neuron fully connected layer. The decoder is also a doubly-stacked GRU with same hyper-parameters as the encoder. The past sequence is restricted to $\mathbf{x}_{t-m:t}$ with $m = 1$sec., matching recent work [7, 33], and we set $H = 5$sec. as the prediction horizon. The sampling rate of the scanpaths is 5Hz, thus the past (resp. future) sequences are 5 (resp. 25) sample-long.

Training procedure: The model is trained using the loss described in Eq. 6. Distance $D(\cdot)$ is taken as the cumulative Euclidean distance, that is $D\left(\mathbf{y}^k_{t:t+H}, \mathbf{x}_{t:t+H}\right) = \sum_{s=0.2}^{H} \left\|\mathbf{y}^k_{t+s} - \mathbf{x}^k_{t+s}\right\|^2$. The optimizer is *AdamW*, with a learning rate of $5 \cdot 10^{-4}$ and a batch size of 64.

4.3 Results on multiple trajectory prediction

In this section we assess (P1) the diversity of predictions, (P2) the performance for $K = 1$, and (P4) the computational cost. Likelihood estimation (P3) is addressed in Sec. 5.2. Results have been updated from our previous article [14] by adding results on a new dataset, MM18 [27], and by using the original code and weights provided by the authors for the *deep-position-only* baseline instead of a re-implementation.

4.3.1 Experimental settings.

Datasets: We consider four datasets of 360° videos with head motion traces:

- MMSys18 [8]: head motion traces of 57 subjects watching 19 360° videos, all lasting 20 seconds.
- CVPR18 [46]: head motion traces of 45 subjects watching 208 360° videos lasting from 15 to more than 80 seconds (36 seconds on average).
- PAMI18 [45]: head motion traces of 58 subjects watching 76 360° videos, lasting from 10 to 80 seconds (25 seconds on average).
- MM18 [27]: head motion traces of 48 subjects watching 9 360° videos, lasting from 19 to 49 seconds (30 seconds on average).

For all of these datasets, we use the same split as described in the supplemental material from by Romero et al. [33], such that there is no overlap between the videos in the train and test sets of CVPR18, PAMI18, and MM18, as well as no overlap between the users of MMSys18. Additionally, we do not make predictions for the first 6 seconds of the video with any of the considered competitors, as done by Romero et al. [33] to skip the user's initial exploration phase.

Metrics: When it comes to evaluating the quality of the multiple predictions, the major challenge is that several plausible futures may correspond to a single input, but the datasets provide only a single ground-truth future. The best way to assess (P1) is therefore to check if the known ground truth is covered by one of the few

predictions, while the others can efficiently explore the search space to cover the futures of close inputs. This can be done by using the *winner-take-all* or *best of many samples* (BMS) metric [4]. Therefore, as is usually done as standard practice in multiple sequence prediction [2, 24, 40], we report the BMS metric. Specifically, BMS at prediction step s is defined in Eq. 7, where the great-circle distance between points P_1 and P_2 on the unit sphere is $\text{gcd}(P1, P2) = \arccos(\sin\phi_1 \sin\phi_2 + \cos\phi_1 \cos\phi_2 \cos\lambda$ with ϕ the latitude and λ the absolute difference in longitude, and k^* is defined in Eq. 8.

$$\frac{1}{U}\frac{1}{V}\frac{1}{T}\sum_{u}\sum_{v}\sum_{t}\text{gcd}(y^{k^*}_{u,v,t+s}, x_{u,v,t+s}) \tag{7}$$

$$k^*(u,v,t) = \arg\min_{k}\sum_{s=0.2}^{H}\text{gcd}(y^{k}_{u,v,t+s}, x_{u,v,t+s}) \tag{8}$$

We report the BMS metric in figures, and we report in a more compact form in tables the average prediction error, which is the average over $s \leq H$ of the BMS metric, similarly to what Marchetti et al. reported [24]. For $K = 1$, the BMS metric is equal to the great-circle distance, hence enabling the assessment of (P2) with the same metric as used for single sequence prediction [7, 33].

Competitors: We compare our models with three competitors. As no competitor exists so far for multiple prediction of head motion, we adapt a recent method from the self-driving domain.

• *Deep-position-only*: *Deep-position-only* is a baseline introduced by Romero et al. [33]. It is a simple sequence-to-sequence LSTM taking past head positions as input. Additional details can be found in section 3.2 of [33]. Thanks to the reproducible framework they published [32], we were able to directly evaluate *Deep-position-only* with the provided code and model weights and achieve the same performance as reported.

• MANTRA-*adapted*: MANTRA is an approach described by Marchetti et al. [24] to predict the trajectory of other vehicles. It uses an auto-encoder in conjunction with a memory network. The auto-encoder is first trained to reconstruct future trajectories from past and future trajectories. A memory-writing controller is then trained to fill the memory with embeddings from the encoder. The memory takes the form of a (key, value) dictionary, where the embeddings of past trajectories are the keys that are used to retrieve the values, embeddings of future trajectories. At prediction time, embeddings of yet unseen past trajectories are computed and matched with keys from the memory. The K most similar keys are used to retrieve the K corresponding values, which are then fused with the embedding of the actual past and decoded into K predicted future trajectories. Memory is built at training time with the following procedure. During training, if none of the predicted trajectories is close enough (defined by a manual threshold) to the ground truth future trajectory, the embeddings (past and future) of this trajectory are added as new key and value to the memory. The loss for the writing controller is designed so that it only writes relevant trajectories into the memory. Embeddings that are too similar and do not help to decrease the prediction error are not added to the memory. At test time, the memory is read-only and filled with embeddings from the training set. For this model to work properly, the trajectories have to be normalized so that they are translation and rotation-invariant. Building from this approach, we build a MANTRA-*adapted* model as a multiple trajectory prediction baseline to be compared to our proposed model. The changes from the original MANTRA model are described as follows. The trajectories are 3-dimensional instead of 2-dimensional. We adapt the manual distance thresholds used for the writing controller with values that fit our data and give an acceptable memory size. We do not normalize the trajectories in the same way. As there is no rotation invariance in head motion, we carried out several tests with translation invariance (separating yaw and pitch). The results were best when only re-centering on yaw (longitude). The results were worse with re-centering both axes, only pitch or with no re-centering. Since video cue is not considered in DVMS, thus not providing any contextual information or map, MANTRA-*adapted* does not employ any contextual cue either, such as the "Iterative Refinement Module" [24], which normally integrates information from the map.

• VPT360: VPT360 is the recurrent transformer-based viewport prediction architecture presented by Chao et al. [7]. We do not reproduce their results because the code is not available at the time of writing, but we report the values presented in their work [7] on the MMSys18 dataset and compare DVMS with VPT360 on the exact same settings.

4.3.2 Experimental results.

Prediction error: Fig. 4 shows the prediction error (great-circle distance, BMS metric of DVMS for $K > 1$) of DVMS against against state-of-the-art single trajectory predictors on all four datasets. The shaded area represents the 95% confidence interval. Detailed prediction results on the same datasets are also available in the appendix in Tables 1, 2, 3 and 4. We observe that DVMS ($K = 1$) slightly outperforms both *Deep-position-only* (by **3.4%**) and VPT360 (by **2.3%**) on the MMSys18 dataset for when looking at the average prediction error over 5 seconds (prediction step $s \leq 5$sec.). DVMS ($K = 1$) also slightly outperforms *Deep-position-only* on the PAMI18 and MM18 datasets, by **2.1%** and **2.8%**, respectively, hence meeting (P2). On CVPR18, DVMS largely outperforms *Deep-position-only* by **20%**, which may suggest that *Deep-position-only* was not properly trained on this dataset. For (P1), we observe for $K = 2$ a **25%** reduction in prediction error for $s \leq 5$sec. with DVMS, compared to the single prediction competitors *Deep-position-only* and VPT360. For higher K, the error reduction increases, and tends to saturate for $K = 4$ then $K = 5$. DVMS hence meets both (P1) and (P2) on these datasets. Fig. 5 compares

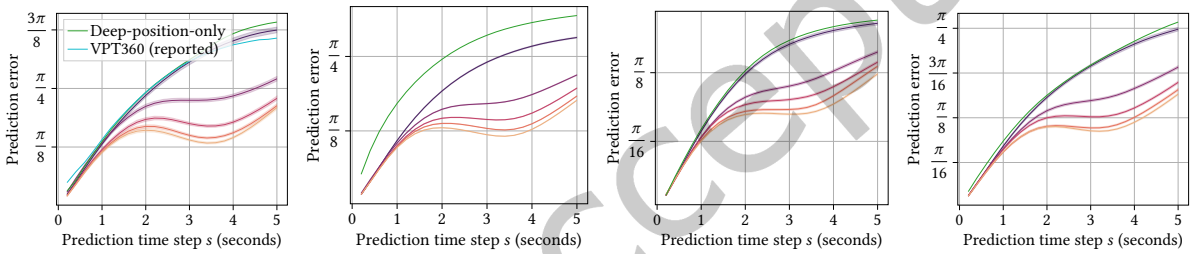


Fig. 4. Prediction error (great-circle distance) of DVMS (ours, same colors as Fig. 5) against state-of-the-art single trajectory predictors on various datasets. Datasets from left to right: MMSys18, CVPR18, PAMI18, MM18. Colors have the same meaning across all subfigures.

the performance of DVMS with the MANTRA-*adapted* competitor on the same datasets. Detailed prediction results can also be found in the appendix in Tables 1, 2, 3 and 4. We first notice that for every $K = 1, \ldots, 5$, DVMS consistently yields a lower prediction error than MANTRA-*adapted*. Also, we observe that MANTRA-*adapted* does not match the state of the art performance of *Deep-position-only* for $K = 1$. Over all datasets, for $s \leq 5$sec., the prediction gains of DVMS over MANTRA-*adapted* range from **26% to 47%** (average 36%) for $K = 1$, from **26% to 41%** (average 33%) for $K = 2$, from **27% to 37%** (average 33%) for $K = 3$, from **27% to 37%** (average 33%) for $K = 4$, and from **26% to 38%** (average 33%) for $K = 5$.

Constructing futures by combining past with future pieces from the training set does not seem sufficient for MANTRA-*adapted* to produce diverse enough futures, compared with DVMS which instead modulates the initial state of the sequence decoder with a random component. The results on all four datasets therefore show that DVMS is able to produce diverse predictions (P1), outperforming the multiple prediction competitor MANTRA-*adapted*, while providing comparable performance to state-of-the-art single trajectory predictors when $K = 1$ (P2).

We ran experiments where DVMS was tested on different datasets than it was trained on to assess its generalization capabilities. We report the cross-dataset performance in the appendix in Table 5. We can see that models trained on smaller datasets such as MMSys18 and MM18 struggle to generalize to other datasets, with a prediction error usually higher than the other models. However, we observe that models trained on CVPR18

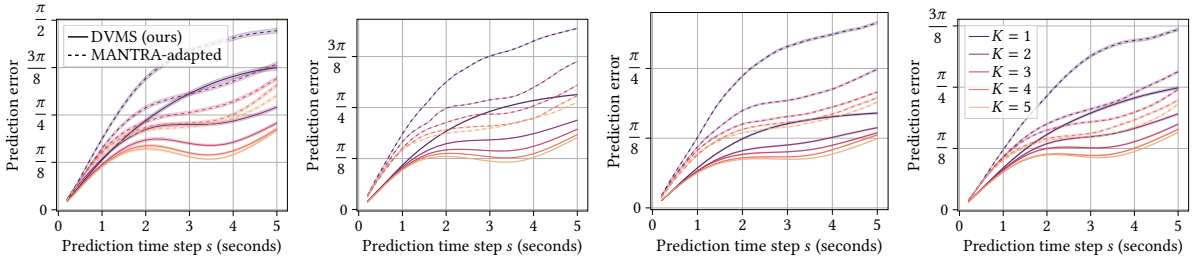


Fig. 5. Prediction error (BMS metric) of DVMS (ours, solid lines) against MANTRA-adapted (dashed lines) on various datasets. Datasets from left to right: MMSys18, CVPR18, PAMI18, MM18. Colors have the same meaning across all subfigures.

(the largest of the four datasets) tend to generalize well to other datasets, even outperforming models trained MMSys18 and MM18 on their own test datasets in some cases, without any kind of dataset-specific fine-tuning. We recommend using the CVPR18 dataset to train DVMS, as models trained on CVPR18 are always the best or second-best performing on any test dataset.

Computational cost: Hardware used to train and test the methods is a Nvidia RTX 3080 with 10GB of video RAM on a station with 128GB of RAM. Table 6 (appendix) shows that DVMS and MANTRA-*adapted* have significantly less weights than both single prediction methods *Deep-position-only* and VPT360. While DVMS has more neural network parameters than MANTRA-*adapted*, the execution time to generate a trajectory at test time is 14% less than MANTRA-*adapted*. This is due to MANTRA-*adapted* having to do memory lookup. Indeed MANTRA-*adapted* has an extra memory, which DVMS does not, and the size of this memory, shown in Table 7 (appendix) in percentage of the training set size, varies with the target accuracy and the dataset (and hence cannot be generalized to other datasets before actual training). Also, training MANTRA-*adapted* requires two phases, the first to train the auto-encoder, the second for the memory writing controller.

Summary: We have validated (P1), (P2), and (P4) on four datasets of head motion data: The error of DVMS significantly decreases when K increases, and it largely outperforms MANTRA-*adapted* (P1). DVMS matches or outperforms VPT360 and *Deep-position-only* for single trajectory prediction (P2). DVMS has significantly less parameters than VPT360 and *Deep-position-only* and has the lowest latency of all compared methods (P4). We validate (P3) in Sec. 5.2.

5 LATENT SPACE ANALYSIS AND LIKELIHOOD ESTIMATION

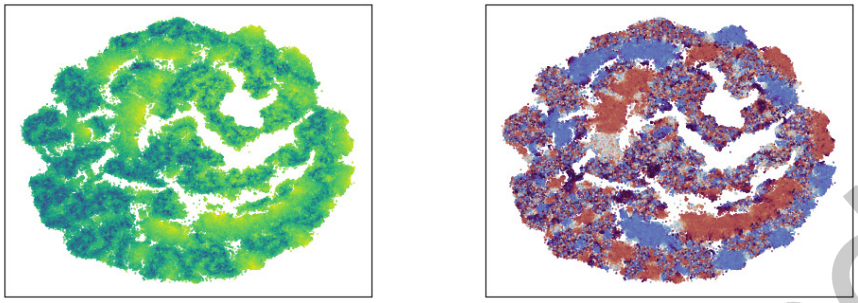
In this section, we first analyze the structure of the latent space learned from the trajectory data and we connect latent space locations and values of z with physical properties. We then present our method to estimate the likelihood of every of the K generated trajectories, instrumental to deploy DVMS in a streaming system in Sec. 6.1.2.

5.1 Linking latent space features to trajectory properties

The model learns a representation of the past trajectory before being combined with z to generate a future trajectory. In this section, we first show what trajectory properties the encoder is able to perceive, and then we analyze the influence that different values of z can have on the generated trajectories when combined with the output of the encoder.

5.1.1 Learned representation of past trajectories. We define as "embedding of the past trajectory" the output of the last layer of the encoder of DVMS, i.e., the output of the last orange layer in Fig. 3 (128 dimensions). Fig. 6 shows a 2D representation (obtained with t-SNE) of all the embeddings of the test set of the CVPR18 dataset obtained with a model trained to predict three trajectories ($K = 3$) on the train set of the CVPR18 dataset. Each

dot corresponds to a past trajectory, and embeddings that are similar in the 128-dimensional space should be close to each other on the 2D representation. In Fig. 6a, the embeddings dots were colored according to the speed



(a) Colored by past trajectory speed (b) Colored by past trajectory direction

Fig. 6. 2D representation of the embeddings of past trajectories learned by the encoder on the CVPR dataset

of their corresponding past trajectories. Blue dots represent embeddings of low-speed trajectories and yellow dots represent embeddings of high-speed trajectories. In Fig. 6b, the embeddings dots were colored according to the direction of their corresponding past trajectories. Blue dots represent embeddings of trajectories going to the left and red dots represent embeddings of trajectories going to the right. In both cases, we can see some areas where the colors are well separated, with yellow "high-speed areas" and blue "low-speed areas" in Fig. 6a, and blue "left-direction areas" and red "right-direction areas" in Fig. 6b. We can observe a correspondence between "high-speed areas" and areas where the direction is clearly left or right. "Low-speed areas" correlate with areas where the direction seems random.

Finding: DVMS is able to differentiate and represent different speeds and angles in its latent space. Specifically, it can easily identify high speed trajectories coming from left or right.

5.1.2 Influence of z. Depending on weight initialization, training set and data order, the model will map different trajectory features to z. Here we show an example for a model trained to predict three trajectories ($K = 3$) on the CVPR dataset.

To understand and evaluate the influence of z in the model, we show the estimated probability density functions (PDF) of the speed and direction of the output trajectories generated with each z_k. The approximate probability density functions are obtained through kernel density estimation (KDE), which we consider to be easier to read and understand than histograms for our data.

Fig. 7 shows the influence of z on trajectory speed. Fig. 7-left shows the distribution of the past and future trajectory speeds. Fig. 7-center shows the distribution of the output trajectory speeds generated with each z_k and that corresponding to the ground truth future (GT). Fig. 7-right shows the distribution of the ratios between past and future speed. A ratio of 10^0 (1) means that the generated/future trajectory has the same speed as its corresponding past. A ratio greater (resp. lower) than 1 means that the generated/future trajectory has a greater (resp. lower) speed than its corresponding past. We can see that predicted speed is always at least slightly lower than actual speed (which we see for all K on all datasets), most likely because predicting higher speed (longer trajectories) will lead to higher error on average. The model learns to be conservative by predicting shorter (lower speed) trajectories. We also see much less variance in predicted speed than in actual speed (which we also see for all K on all datasets), but having more predicted trajectories (higher K) allows for more diversity overall, since each z can cover different parts of the distribution. In this example, the model learns to "specialize" z: z_2 generates low speed trajectories while z_1 and z_3 give similar speeds, usually in the same order of magnitude as GT.

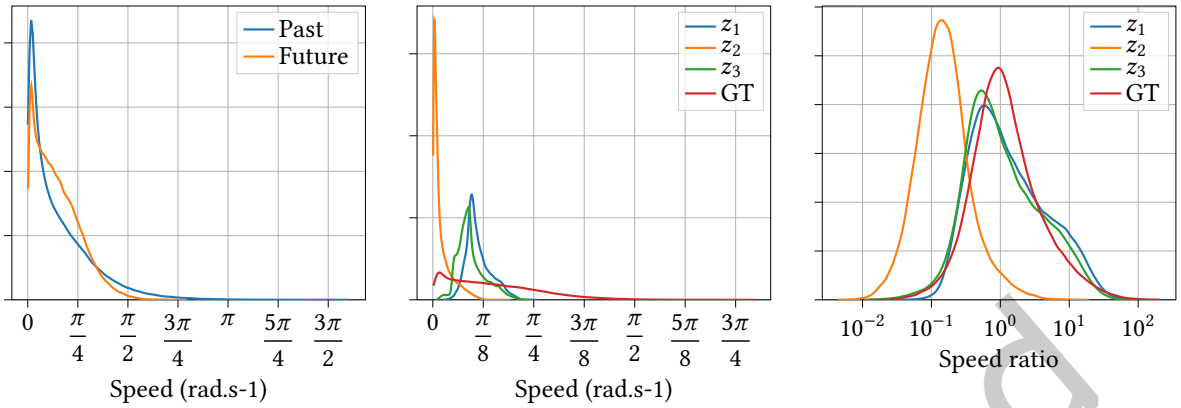


Fig. 7. Distribution of the trajectory speeds depending on z

Fig. 8 shows the influence of z on trajectory direction. Fig. 8-left shows the distribution of the past and future trajectory directions. Fig. 8-center shows the distribution of the output trajectory directions generated with each z_k compared to the ground truth future (GT). Fig. 8-right shows the distribution of the differences between past and future direction. A difference of 0 means that the generated/future trajectory kept going in the same direction as its corresponding past. A difference greater (resp. lower) than 0 means that the generated/future trajectory turned right (resp. left) relative to its corresponding past. The direction peaks that we observe on Fig. 8-left and

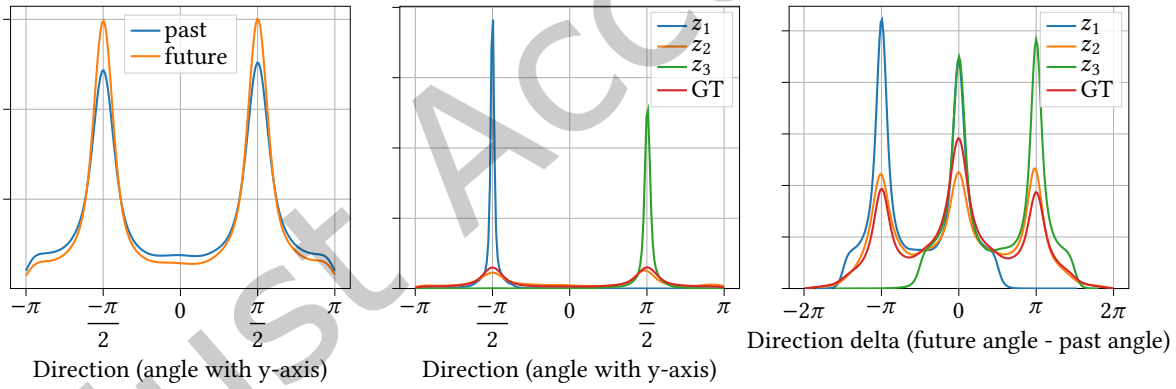


Fig. 8. Distribution of the trajectory directions depending on z

Fig. 8-center indicate more trajectories going left ($\frac{-\pi}{2}$) or right ($\frac{\pi}{2}$) than in other directions. This means there is a lot more horizontal head movement than vertical head movement, which is expected in head motion data. We can discern 3 peaks on Fig. 8-right. The first peak at $-\pi$ corresponds to cases where the past was going to the right but the future is going to the left, the second peak at 0 corresponds to cases where the past and the future go in the same direction, and the third peak at $+\pi$ corresponds to cases where the past was going to the left but the future is going to the right.

In this example, while z_1 and z_3 give similar speeds, the generated trajectories have completely different directions. With z_1, the generated trajectories always go in the left direction. With z_3, the generated trajectories always go in the right direction. The direction of trajectories generated with z_2 follow a distribution close to GT.

We can see that in order to go left, trajectories generated with z_1 will either continue in the same direction as the past or turn around to go left if the past was going right. Similarly, trajectories generated with z_3 will either continue or turn around in order to always go right.

Finding: DVMS learns to efficiently use the values of z by specializing them. Each z_k will generate a different *mode* of future trajectory. In this example, we have seen that one z_k is used to generate low speed futures, while the other two are used to generate higher-speed futures, but with opposite directions.

5.2 Exploiting properties of z to estimate trajectory likelihood

For a regression problem, the likelihood $Pr[\mathbf{y}^k_{t:t+H}|\mathbf{x}_{0:t}]$ of a future trajectory can be expressed with $\exp^{-D(\mathbf{y}_{t:t+H},\mathbf{x}_{t:t+H})}$, hence estimating the likelihood is equivalent to estimating the distance of a trajectory to the ground truth, that is the negative log-likelihood. We define in Eq. 9 $err^k_{u,v,t}$ the error of the k-th generated trajectory $\mathbf{y}^k_{t:t+H}$, the motion of user u on video v at timestamp t.

$$err^k_{u,v,t} = D\left(\mathbf{y}^k_{t:t+H}, \mathbf{x}_{t:t+H}\right) \tag{9}$$

With a variational framework, a standard approach to estimate the likelihood would be to rely on the model (whose parameters are set from the training data) and on the known past $\mathbf{x}_{0:t}$. In this work, we argue that this is not sufficient, and that other available information must be considered, namely the past generated trajectories $\mathbf{y}^k_{s:\min(s+H,t)}$, for all $k \in \{1, K\}$ and $s \in [0, t]$, and the errors obtained by every such trajectory when compared to the available ground truth at t $\mathbf{x}_{s:\min(s+H,t)}$. Indeed, these errors are informative of which z_k, for $k \in \{1, K\}$, have best coded the latent features connecting the future trajectory with the past trajectory. If the errors over the various z_k, for $k \in \{1, K\}$, have sufficient stationarity in time, then we can exploit such stationarity to estimate the likelihood of the predicted trajectories. We therefore define an estimate the estimate $\widehat{err}^k_{u,v,t}(r)$ of $err^k_{u,v,t}$ in Eq. 10.

$$\widehat{err}^k_{u,v,t}(r) = D\left(\mathbf{y}^k_{t-r:\min(t-r+H,t)}, \mathbf{x}_{t-r:\min(t-r+H,t)}\right) \tag{10}$$

where r is a past window of size controlling the age of the trajectory ground truth to produce the error estimate. Let us recall that the z-space is discrete, with $\mathcal{Z}_K = \{z_k\}_{k=1}^K$. This means that $err^k_{u,v,t}$ is predicted by the error produced by the trajectory $\mathbf{y}^k_{t-r:t-r+H}$ generated with the same z_k and predicted at time $t - r$ over a horizon H, but with the error only counted on the timestamps for which the ground truth $\mathbf{x}^{u,v}$ is available, i.e., on $[t - r, \min(t - r + H, t)]$.

The accuracy of this estimator therefore depends on the stationarity in time of the distribution of the error over the latent values z_k, for $k \in \{1, K\}$. In other words, if the error err^k of the trajectory $\hat{y}_k$ generated from z_k was low in the recent past, it is likely to still be low. We refer to [14, Sec. 5.2] where we have shown that the error for a given latent value z_k exhibits temporal stationarity that we can exploit to estimate the likelihood, and where we have evaluated the likelihood estimation on three datasets, hence validating (P3). We detail in Sec. 6.1.2 how the generated trajectories and their corresponding likelihoods are used to estimate the probability distribution of the future viewport by assigning "tile scores".

6 360° VIDEO STREAMING WITH DVMS

This section presents an extensive evaluation of the interest of our proposal DVMS in a 360° streaming system. We first present the simulation environment we consider: the requirements we set to match a realistic streaming system, the new tool we introduce to realize this environment, and how multiple trajectory prediction is leveraged in the streaming logic. We then present the simulation settings and results, quantifying the gains on different metrics.

6.1 Simulating 360° video adaptive streaming with SMART360

6.1.1 About SMART360. We have chosen to run simulations, as user experiments can be very costly. We aim at solving the problem defined in Sec. 3.1: a person watching 360° video in VR streamed over a network with variable bandwidth. The video is temporally divided in segments and spatially divided in tiles. Tiles for each segment are available in several quality levels, higher quality tiles requiring higher bit rates. Every Δ_{DL} seconds, the client uses an ABR algorithm to makes request for new tiles which are then downloaded and put in the buffer while the video is playing. The ABR algorithm is responsible for selecting the tiles and quality levels based on user viewport prediction, network bandwidth estimation and buffer levels. If the viewport contains a tile that is not present in the buffer in any quality level, the video playback is paused until the tile is downloaded (stall period). Despite the wide availability of head motion traces datasets, we could not find any suitable public software tool to simulate a realistic tile-based VR adaptive streaming system, flexible enough to work with any 360° video, head motion and network trace.

To overcome the drawbacks of the few existing emulation/simulation tools, we develop a Python-based simulation environment named SMART360. SMART360 simulates the adaptive streaming process for 360° videos. SMART360 offers a realistic streaming behavior, with stall events and ABR planning, along with highly-configurable code, with many inputs and settings. Owing to the lack of space, we refer to [13] for the details on SMART360. We implemented the ABR algorithm described in Sec. 6.1.2 and the DVMS-based model described in Sec. 4.2 in the simulation environment. The code and data are made available[1][2].

6.1.2 DVMS implementation in SMART360. In the SMART360 simulator, the client uses an ABR algorithm to make requests for new tiles every Δ_{DL} seconds. The ABR makes its quality allocation for incoming segments decisions based on tile scores, given by the viewport prediction algorithm. Before the prediction is made, all the tile scores are initialized to 0. DVMS outputs the predicted head positions for a given segment. In our case, DVMS outputs $5 \cdot K$ points, because the segments are 1 second long and the head motion trace sampling rate is 5 Hz. For each predicted position (FoV center), we calculate the list of tiles belonging to this FoV. The scores of the tiles belonging to this FoV are updated as follows: $\alpha \mathrel{+}= \frac{\mathcal{L}_k}{5 \cdot K}$, with α being the score of any tile belonging to the FoV calculated from a position of a predicted trajectory of likelihood $\mathcal{L}_k$. Since $\sum_{k \in K} \mathcal{L}_k = 1$, the maximum score for a tile that belong to all the predicted FoVs is 1. The remaining tiles are given a score inversely proportional to the distance to the viewport.

Adaptive bitrate (ABR) algorithm: The objective of the ABR algorithm is to maximize the expected QoE given the predicted viewport, the estimated network bandwidth and the buffer level. This task is achieved by selecting the right tiles to download in the right quality, such that the quality inside of the user's viewport is as high as possible, without any stall event. Every Δ_{DL} seconds, the ABR algorithm is used to produce a download schedule that will be sent as a request to the server. Adaptive bitrate algorithms for regular videos form a well-studied field, and many types of ABR strategies exist. We can define three categories of ABR strategies: rate-based, which base their decisions upon bandwidth estimation, buffer-based, which base their decisions upon buffer level, and hybrid approaches (among which there can be learning-based approaches), which can use both the estimated bandwidth and the buffer level. We chose to implement a simple hybrid ABR algorithm named *BaselineABR* for tile-based streaming that can demonstrate the advantages of multiple trajectory prediction. This algorithm was kept simple for an easier understanding of the streaming behavior with different viewport prediction algorithms. The objective of *BaselineABR* is to maximize the expected viewport quality, while maintaining a minimum buffer level B_{min}. A simplified version of the *BaselineABR* is described in Algo. 1 in the appendix.

[1]https://gitlab.com/SMART360/SMART360-simulator
[2]https://gitlab.com/SMART360/SMART360-preprocessing

6.2 Simulation settings

The results presented in Sec. 6.3 summarize metrics from **4,729,000 simulations** using **3,378 head motion traces** of **132 different users** watching **94 different videos**, **40 different network traces** with **5 different buffer settings** and **7 different viewport prediction algorithms**.

6.2.1 Videos. The simulations were run on 94 different videos coming from the test sets of three datasets the DVMS model was trained on (see Sec. 4.3.1). The original video files of each dataset were retrieved, split in a **12x6 tile layout** and re-encoded with *libx265*, using the HEVC compression standard. The tiles were each encoded in **five different quality levels** with different **constant rate factors (CRFs): 16, 22, 28, 34, and 40**. Finally, the videos were packaged in **1 second segments** for streaming delivery.

6.2.2 Head motion traces. Each video was watched by an average of around 36 users (range 28-58), which gives a total 3,378 head motion traces, coming from 132 different unique users. Each trace contains the head positions of the user with a 5 Hz sampling rate. These traces were not included in the training of the model, as they come from the test sets of the datasets.

6.2.3 Network traces. The simulations were run on 40 different 4G network traces from the 4G/LTE dataset published by van der Hooft et al. [42]. They are made of 40 traces of bandwidth and latency measurements along several routes in the city of Ghent, Belgium. For the comparisons between ABR algorithms to be relevant, the average throughputs of the network traces were scaled to approximately match the video bit rates, because we need to be in a situation where the algorithm has to adapt to the network constraints to make a difference in quality.

6.2.4 Buffer settings. The maximum size of the **buffer** was always set to **10 segments** (i.e., 10 seconds), because it is not possible to show the benefit of viewport prediction with larger values, since we only predict the future head positions 5 seconds ahead. The simulations were run for 5 different values of B_{min}, the ABR buffer constraint (see Sec. 6.1.2): 1, 2, 3, 4, and 5 seconds. With this parameter, we can tune the behavior of the ABR algorithm and show results for different levels of aggressiveness.

6.2.5 Prediction algorithms. 7 different prediction algorithms were tested in the simulations:

- *NoPred*: no assumption is made about the viewport location and the same score is given to all tiles before ABR allocation. This baseline serves as an example of "maximum fairness", everyone receiving the same quality. We choose to compare all prediction methods to this baseline to assess the QoE fairness gains, while showing a significantly better average QoE than this maximum fairness baseline.
- *StaticPred*: we assume that the person will stay still and that the viewport will not change in the near future. Tiles present in the viewport are given a score of 1.0, and remaining tiles are given a score inversely proportional to the distance to the viewport. This serves as a baseline for comparison.
- DVMS, $K = 1$: DVMS is used to predict one trajectory of the future head positions. Since there is only one trajectory, the likelihood of this trajectory is set to 1.0 and the tile scores are computed as described in Sec. 6.1.2.
- DVMS, $K > 1$ (2 to 5): DVMS is used to predict K possible trajectories of the future head positions. The respective likelihoods are based on the past error as described in Sec. 5.2. The tile scores are computed as described in Sec. 6.1.2.

In the following sections, we write DVMS-**k** instead of DVMS, $K = k$ to improve readability.

6.2.6 Metrics. We report results on two metrics in the following section:
Viewport quality: For each video segment, a person sees multiple tiles that were downloaded at a certain quality level. The quality level of a tile is approximately logarithmically proportional to its bitrate, because of the choice

of CRFs that was made in Sec. 6.2. The viewport quality metric is computed as the weighted average of the quality of the tiles that were seen during a segment. The weights are proportional to the duration for which this tile was visible during the segment.

Normalized QoE: After reviewing several references including QoE functions combining various components, we decided to make our own in order to avoid subjective hyper-parameter choices required by existing formula. We defined our own QoE function, the normalized QoE (Eq. 11) combines the viewport quality, the stall periods, the spatial quality variance, and the temporal quality variance in one metric, with a value between 0 and 1. T is the duration of the video and S is the stall duration over the simulation. $\overline{VQ}$ is the average viewport quality over the video, VQ_{max} is the maximum possible viewport quality for a frame. $\overline{SQV}$ is the average spatial quality variance (standard deviation of quality levels of the tiles in a viewport) over the video, SQV_{max} is the maximum possible spatial quality variance for a frame. TQV is the temporal quality variance (mean of absolute differences between average viewport quality of segments) over the video, TQV_{max} is the maximum possible temporal quality variance over a video.

$$QoE = \frac{T \cdot \overline{VQ}}{VQ_{max} \cdot (T+S)} \cdot \left(1 - \frac{\overline{SQV}}{2 \cdot SQV_{max}}\right) \cdot \left(1 - \frac{TQV}{2 \cdot TQV_{max}}\right) \tag{11}$$

6.3 Results

In this section, we compare the results for simulations with the viewport prediction algorithms presented in Sec. 6.2.5. For DVMS, only $K = 1$, $K = 5$, and $K = best$ are shown for better readability. The choice of number of trajectories to predict (K) is related to prediction uncertainty, where factors such as video features such as spatial and temporal information as well as user emotions can have a strong impact [12]. The "*best K*" shows the potential gains if we were able to use this information to choose the best K (ranging from 1 to 5) for each (user, video) pair, but even more gains could be found by dynamically adapting K during video playback, using head speed or past prediction error. In these experiments, we verified that the bandwidth consumption of all the compared algorithms was identical (not shown here due to space limitations).

6.3.1 Viewport quality and QoE gains of DVMS. We present results showing viewport quality and QoE gains in Fig. 9, where the simulations were run with $B_{min} = 1$, which is where differences between viewport prediction algorithms are the most visible. The order of performance between predictors is nearly identical for all values of B_{min}. As B_{min} increases, the advantage of viewport prediction slowly decreases. We provide detailed results in the appendix in Tables 8 and 9 for completeness.

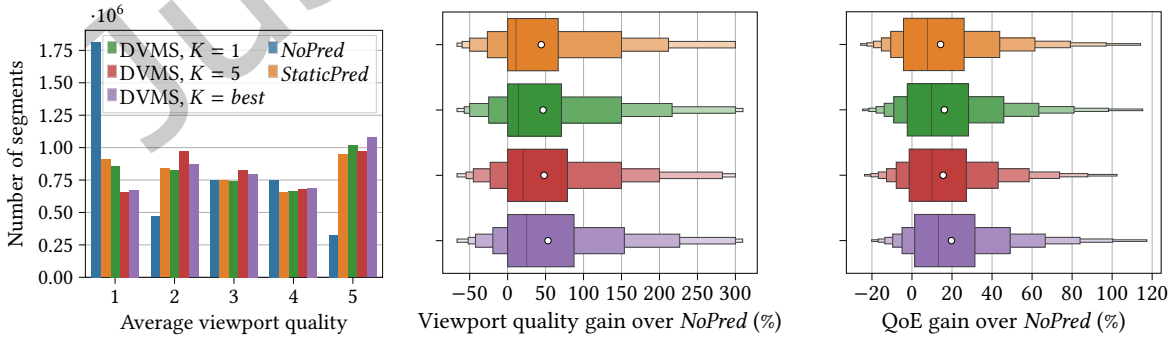


Fig. 9. Viewport quality and QoE gains over all simulations with $B_{min} = 1s$

Fig. 9-left shows the average viewport quality across all segments during the simulations. The average viewport quality is not necessarily an integer, each bin contains the number of segments with an average viewport quality lower or equal to its tick label, but greater than the preceding bins. *NoPred* (i.e., uniform spread of the quality budget) gives the most segments with the lowest quality and the fewest segments with the highest quality. We can see that *StaticPred* already gives significant quality improvements. This figure also illustrates a key difference between single and multiple trajectory prediction: DVMS-**5** gives fewer segments with a very low viewport quality, but also fewer segments with a very high viewport quality than DVMS-**1**. While DVMS-**1** leads to more segments with very low quality than DVMS-**5**, it is still fewer very-low-quality segments than *StaticPred*. While DVMS-**5** leads to fewer segments with very high quality than DVMS-**1**, it is still more very-high-quality segments than *StaticPred*. DVMS-**best** gives the best of both worlds with less segments with very low quality and more segments with very high quality.

Fig. 9-center shows the viewport quality gain over *NoPred* for all the played segments of the simulations. We can see that DVMS-**5** has an average viewport quality gain (**48.0%**) slightly better than the gain of DVMS-**1** (46.7%). The median gain is better: 50% of segments have a viewport quality improvement over 20.4% with DVMS-**5**, while 50% of segments have a viewport quality improvement over 14.3% with DVMS-**1**. Improvements are also more evenly distributed with DVMS-**5**, with **61.2%** of segments having an increased viewport quality (20.0% decreased, 18.8% unchanged), while 57.2% of segments had an increased viewport quality (20.4% decreased, 22.4% unchanged) with DVMS-**1**.

Fig. 9-right shows the QoE gain over *NoPred* for all simulations. We can see that DVMS-**5** has an average QoE gain (15.6%) slightly worse than the gain of DVMS-**1** (16.2%), but the median gain is slightly better: 50% of segments have a QoE improvement over 10.0% with DVMS-**5**, while 50% of segments have a QoE improvement over 9.8% with DVMS-**1**. Improvements are once again more evenly distributed with DVMS-**5**, with **71.9%** of simulations having an increased QoE, while **70.0%** of simulations had an increased QoE with DVMS-**1**.

Summary: These results highlight the aggressiveness of DVMS-**1**. Single trajectory prediction completely focuses the quality in a specific area: if the prediction is accurate, we can have very-high-quality segments, but if the prediction is inaccurate, we will have very-low-quality segments. Predicting more trajectories is a more conservative approach. Over all B_{min}, DVMS-**5** gives improvements comparable to DVMS-**1** on average but leads to better fairness between users than DVMS-**1**: there are more cases of segments where the quality is improved, slightly fewer cases with very high quality, considerably fewer cases with very low quality.

6.3.2 Link with prediction error. To confirm that this different distribution of quality gains between DVMS-**1** and DVMS-**5** is indeed due to the prediction error, we first look at the link between normalized QoE and prediction error in Fig. 10-left. The x-axis of this figure is the prediction centile: the average prediction error for all (users, videos) was sorted and equally distributed in 100 sorted bins, so that each bin has the same number of (users, videos). Unsurprisingly, there is a clear decreasing trend across all K: the QoE decreases when the prediction error increases.

We now show the average viewport quality (resp. QoE) of segments (resp. simulations) against DVMS prediction error deciles in Fig. 10-center (resp. 10-right). The process for deciles is the same as for centiles, but there are only 10 bins instead of 100. We can see that predicting one trajectory is better than predicting 5 trajectories when head motion is predictable and that the single trajectory is very accurate. However, when head motion becomes less predictable and prediction error increases, predicting 5 trajectories leads to higher viewport quality and QoE. As previously shown in Fig. 9, predicting multiple trajectories increases fairness between users, as individual bad predictions have a smaller negative effect on viewport quality and QoE than in the case of single trajectory prediction. Finally, we can see that the potential gain in visual quality (resp. QoE) when using DVMS-**best** over DVMS-**1** is around 10% (resp. 5%) for roughly 30% of the data, when the prediction error is at its highest (deciles 7-10).

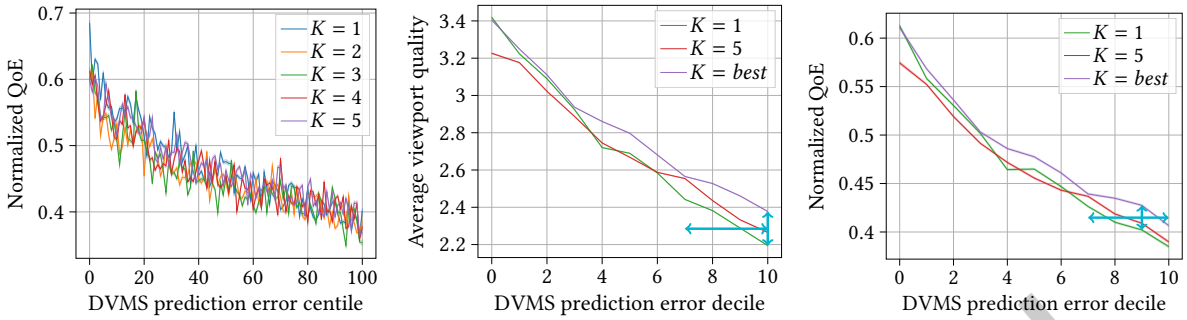


Fig. 10. Average viewport quality and QoE against DVMS prediction error quantiles. Horizontal blue arrow: 30% of the users, vertical arrow: 10% gain in visual quality, 5% gain in QoE.

7 DISCUSSION

We have presented the first proposal to generate multiple future plausible trajectories of user head or gaze motion in 360° videos. Our DVMS learning framework establishes a first baseline paving the way to more principled approaches to stochastic motion prediction in this domain. Such stochastic approach is also instrumental to enable the automatic triggering of interactive strategies when the predictability of the user motion is evaluated to be too low. While the average gains of DVMS compared to static prediction appear limited, our approach showed that increasing the number of predicted trajectories significantly increases the fairness between the users. These prediction gains are limited by the predictability of head motion, and depend on many factors, such as the type of video or the emotional state of the user [12, 33]. Applying the DVMS framework to content-aware architectures [33], in connection to user and video profiles, is expected to bring more gains and will enable to investigate more intricate connections between scene video content, user state and motion predictability. Indeed, we have shown the type of gains that DVMS can bring, particularly to reduce the number of (video,user) traces with lowest visual quality and QoE, while maintaining the number of traces to high-quality levels. Beyond enabling such a fairness increase, maximizing it however requires to dynamically adapt the number K of predicted trajectories to both the type of scene and the current state of the user, that is to the (video,user) pair, but also over time, considering changing types of scene and user attentional states. The more predictable the user motion (i.e., in sync with the content and with an attention-driving content with a low number of points of interest), the less the need for a high K. This is the subject of exciting future work, involving emotion recognition and learning.

8 CONCLUSION

In this article, we presented the first method for multiple head motion prediction in 360° videos, motivated by the user motion uncertainty yielding a high diversity of future trajectories. Our main contribution is a new learning framework, called DVMS, which builds on deep latent variable models and allows to predict multiple future trajectories from a given past. We design a training procedure to obtain a flexible and lightweight stochastic prediction model compatible with sequence-to-sequence architectures. We analyze the structure of the learned latent space and the influence of the latent variable on the generated futures, and are able to connect them with physical properties of the trajectories. We assess DVMS on 4 datasets and show that it outperforms competitors adapted from the self-driving domain by up to 41%, on prediction horizons up to 5 seconds. We then deploy an extensive simulation framework for which we introduce a new Python-based streaming simulator (made available to the community), and consider 4 different datasets of user, video and network bandwidth traces. We show that predicting multiple trajectories yields a higher fairness between the traces, the gains for 20% to 30% of the users reaching up to 10% in visual quality for the best number K of trajectories to generate for a given

trace. DVMS paves the way for multiple trajectory prediction in VR. In particular, DVMS can be adapted to 6DoF interactive environments, such as gaming or other forms of virtual social spaces, where both head-gaze and body movements need to be predicted. Adapting to these new environments brings numerous challenges. Open research questions still need to be addressed, such as the predictability of movements in 6DoF, 3D saliency estimation, or the consideration of additional structured descriptions of these environments.

ACKNOWLEDGMENTS

This work has been partly supported by the French government, through the UCA JEDI and EUR DS4H Investments in the Future projects ANR-15-IDEX-0001 and ANR-17-EURE-0004.
This work was partly supported by EU Horizon 2020 project AI4Media, under contract no. 951911 (https://ai4media.eu/).
This work was performed using HPC/AI resources from GENCI-IDRIS (Grant 2020-AD011012209).

REFERENCES

[1] Mathias Almquist, Viktor Almquist, Vengatanathan Krishnamoorthi, Niklas Carlsson, and Derek Eager. 2018. The prefetch aggressiveness tradeoff in 360° video streaming. In *9th ACM Multimedia Systems Conference*. 258–269.
[2] Mohammad Babaeizadeh, Chelsea Finn, Dumitru Erhan, Roy H. Campbell, and Sergey Levine. 2018. Stochastic Variational Video Prediction. In *6th International Conference on Learning Representations (ICLR)*.
[3] Lorenzo Berlincioni, Federico Becattini, Lorenzo Seidenari, and Alberto Del Bimbo. 2021. Multiple Future Prediction Leveraging Synthetic Trajectories. In *25th International Conference on Pattern Recognition (ICPR)*. 6081–6088.
[4] Apratim Bhattacharyya, Mario Fritz, and Bernt Schiele. 2018. Accurate and Diverse Sampling of Sequences based on a "Best of Many" Sample Objective. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.
[5] Eli Brookner. 1998. Tracking and Kalman filtering made easy. (1998).
[6] Ming-Fang Chang, John Lambert, Patsorn Sangkloy, Jagjeet Singh, Slawomir Bak, Andrew Hartnett, De Wang, Peter Carr, Simon Lucey, Deva Ramanan, et al. 2019. Argoverse: 3D tracking and forecasting with rich maps. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 8748–8757.
[7] Fang-Yi Chao, Cagri Ozcinar, and Aljosa Smolic. 2021. Transformer-based Long-Term Viewport Prediction in 360° Video: Scanpath is All You Need. In *IEEE 23nd International Workshop on Multimedia Signal Processing (MMSP)*.
[8] Erwan J. David, Jesús Gutiérrez, Antoine Coutrot, Matthieu Perreira Da Silva, and Patrick Le Callet. 2018. A dataset of head and eye movements for 360° videos. In *9th ACM Multimedia Systems Conference*. 432–437.
[9] Xiaoxiong Fan, Yun Cai, Yufei Yang, Tianxing Xu, Yike Li, Songhai Zhang, and Fanglue Zhang. 2021. Detection of scene-irrelevant head movements via eye-head coordination information. *Virtual Reality & Intelligent Hardware* 3 (2021), 14.
[10] Stanislav Fort, Clara Huiyi Hu, and Balaji Lakshminarayanan. 2020. Deep Ensembles: A Loss Landscape Perspective. https://openreview.net/forum?id=r1xZAkrFPr
[11] Yarin Gal and Zoubin Ghahramani. 2016. Dropout as a Bayesian approximation: representing model uncertainty in deep learning. In *33rd International Conference on Machine Learning (ICML)*. 1050–1059.
[12] Quentin Guimard and Lucile Sassatelli. 2022. Effects of Emotions on Head Motion Predictability in 360° Videos. In *14th International Workshop on Immersive Mixed and Virtual Environment Systems* (Athlone, Ireland). 37–43.
[13] Quentin Guimard and Lucile Sassatelli. 2023. SMART360: Simulating Motion Prediction and Adaptive BitRate STrategies for 360° Video Streaming. In *Proceedings of the 14th Conference on ACM Multimedia Systems* (Vancouver, BC, Canada). 384–390.
[14] Quentin Guimard, Lucile Sassatelli, Francesco Marchetti, Federico Becattini, Lorenzo Seidenari, and Alberto Del Bimbo. 2022. Deep Variational Learning for Multiple Trajectory Prediction of 360° Head Movements. In *13th ACM Multimedia Systems Conference* (Athlone, Ireland). 12–26.
[15] Agrim Gupta, Justin Johnson, Li Fei-Fei, Silvio Savarese, and Alexandre Alahi. 2018. Social GAN: Socially Acceptable Trajectories with Generative Adversarial Networks. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2255–2264.
[16] David Ha and Jürgen Schmidhuber. 2018. Recurrent World Models Facilitate Policy Evolution. In *Advances in Neural Information Processing Systems 31 (NeurIPS)*.
[17] Christoph Hermes, Christian Wohler, Konrad Schenk, and Franz Kummert. 2009. Long-term vehicle motion prediction. In *2009 IEEE intelligent vehicles symposium*. IEEE, 652–657.
[18] Han Hu, Zhimin Xu, Xinggong Zhang, and Zongming Guo. 2019. Optimal Viewport-Adaptive 360-Degree Video Streaming Against Random Head Movement. In *2019 IEEE International Conference on Communications (ICC)*. 1–6.

[19] Boris Ivanovic and Marco Pavone. 2019. The Trajectron: Probabilistic multi-agent trajectory modeling with dynamic spatiotemporal graphs. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. 2375–2384.
[20] Xiaolan Jiang, Yi-Han Chiang, Yang Zhao, and Yusheng Ji. 2018. Plato: Learning-based Adaptive Streaming of 360-Degree Videos. In *2018 IEEE 43rd Conference on Local Computer Networks (LCN)*. 393–400.
[21] Nuowen Kan, Chenglin Li, Caiyi Yang, Wenrui Dai, Junni Zou, and Hongkai Xiong. 2021. Uncertainty-aware robust adaptive video streaming with bayesian neural network and model predictive control. In *31st ACM Workshop on Network and Operating Systems Support for Digital Audio and Video*. 17–24.
[22] Diederik P. Kingma and Max Welling. 2014. Auto-Encoding Variational Bayes. In *2nd International Conference on Learning Representations (ICLR)*.
[23] Hongzi Mao, Ravi Netravali, and Mohammad Alizadeh. 2017. Neural Adaptive Video Streaming with Pensieve. In *Conference of the ACM Special Interest Group on Data Communication*. 197–210.
[24] Francesco Marchetti, Federico Becattini, Lorenzo Seidenari, and Alberto Del Bimbo. 2020. MANTRA: Memory Augmented Networks for Multiple Trajectory Prediction. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 7141–7150.
[25] Francesco Marchetti, Federico Becattini, Lorenzo Seidenari, and Alberto Del Bimbo. 2020. Multiple trajectory prediction of moving agents with memory augmented networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2020).
[26] Radford M. Neal. 2012. *Bayesian learning for neural networks*. Vol. 118.
[27] Anh Nguyen, Zhisheng Yan, and Klara Nahrstedt. 2018. Your Attention is Unique: Detecting 360° Video Saliency in Head-Mounted Display for Head Movement Prediction. In *ACM Int. Conf. on Multimedia*. 1190–1198.
[28] Behnam Parsaeifard, Saeed Saadatnejad, Yuejiang Liu, Taylor Mordan, and Alexandre Alahi. 2021. Learning Decoupled Representations for Human Pose Forecasting. In *2021 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*. 2294–2303.
[29] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. 2014. Stochastic Backpropagation and Approximate Inference in Deep Generative Models. In *31st International Conference on Machine Learning (ICML)* (Beijing, China). II–1278–II–1286.
[30] Giuseppe Ribezzo, Luca De Cicco, Vittorio Palmisano, and Saverio Mascolo. 2020. TAPAS-360°: A Tool for the Design and Experimental Evaluation of 360° Video Streaming Systems. In *28th ACM International Conference on Multimedia* (Seattle, WA, USA). 4477–4480.
[31] Branko Ristic, Sanjeev Arulampalam, and Neil Gordon. 2003. *Beyond the Kalman filter: Particle filters for tracking applications*.
[32] Miguel Fabián Romero-Rondón, Lucile Sassatelli, Ramón Aparicio-Pardo, and Frédéric Precioso. 2020. A unified evaluation framework for head motion prediction methods in 360° videos. In *11th ACM Multimedia Systems Conference (MMSys '20)*. 279–284.
[33] Miguel Fabián Romero-Rondón, Lucile Sassatelli, Ramón Aparicio-Pardo, and Frédéric Precioso. 2022. TRACK: A New Method From a Re-Examination of Deep Architectures for Head Motion Prediction in 360° Videos. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44, 9 (2022), 5681–5699.
[34] Christian Rupprecht, Iro Laina, Robert DiPietro, Maximilian Baust, Federico Tombari, Nassir Navab, and Gregory D Hager. 2017. Learning in an Uncertain World: Representing Ambiguity Through Multiple Hypotheses. In *2017 IEEE International Conference on Computer Vision (ICCV)*. 3591–3600.
[35] Lucile Sassatelli, Marco Winckler, Thomas Fisichella, Ramon Aparicio, and Anne-Marie Pinna-Déry. 2019. A New Adaptation Lever in 360° Video Streaming. In *29th ACM Workshop on Network and Operating Systems Support for Digital Audio and Video*. 37–42.
[36] Hedvig Sidenbladh, Michael J Black, and Leonid Sigal. 2002. Implicit probabilistic models of human motion for synthesis and tracking. In *Computer Vision—ECCV 2002: 7th European Conference on Computer Vision Copenhagen, Denmark, May 28–31, 2002 Proceedings, Part I 7*. Springer, 784–800.
[37] Vincent Sitzmann, Ana Serrano, Amy Pavel, Maneesh Agrawala, Diego Gutierrez, Belen Masia, and Gordon Wetzstein. 2018. Saliency in VR: How Do People Explore Virtual Environments? *IEEE Transactions on Visualization and Computer Graphics* 24, 4 (2018), 1633–1642.
[38] Kevin Spiteri. 2021. *Video Adaptation for High-Quality Content Delivery*. Ph. D. Dissertation.
[39] Kevin Spiteri, Ramesh Sitaraman, and Daniel Sparacio. 2018. From Theory to Practice: Improving Bitrate Adaptation in the DASH Reference Player. In *9th ACM Multimedia Systems Conference* (Amsterdam, Netherlands). 123–137.
[40] Shashank Srikanth, Junaid Ahmed Ansari, Sarthak Sharma, et al. 2019. INFER: INtermediate representations for FuturE pRediction. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2019)*.
[41] Luca Thiede and Pratik Brahma. 2019. Analyzing the Variety Loss in the Context of Probabilistic Trajectory Prediction. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. 9953–9962.
[42] Jeroen van der Hooft, Stefano Petrangeli, Tim Wauters, Rafael Huysegems, Patrice Rondao Alface, Tom Bostoen, and Filip De Turck. 2016. HTTP/2-Based Adaptive Streaming of HEVC Video Over 4G/LTE Networks. *IEEE Communications Letters* 20, 11 (2016), 2177–2180.
[43] Jürgen Wiest, Matthias Höffken, Ulrich Kreßel, and Klaus Dietmayer. 2012. Probabilistic trajectory prediction with Gaussian mixture models. In *2012 IEEE Intelligent Vehicles Symposium*. 141–146.
[44] Chenglei Wu, Zhi Wang, and Lifeng Sun. 2021. PAAS: a preference-aware deep reinforcement learning approach for 360° video streaming. In *31st ACM Workshop on Network and Operating Systems Support for Digital Audio and Video*. 34–41.
[45] Mai Xu, Yuhang Song, Jianyi Wang, Minglang Qiao, Liangyu Huo, and Zulin Wang. 2019. Predicting Head Movement in Panoramic Video: A Deep Reinforcement Learning Approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 41, 11 (2019), 2693–2708.

[46] Yanyu Xu, Yanbing Dong, Junru Wu, Zhengzhong Sun, Zhiru Shi, Jingyi Yu, and Shenghua Gao. 2018. Gaze Prediction in Dynamic 360° Immersive Videos. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 5333–5342.

[47] Francis Y. Yan, Hudson Ayers, Chenzhi Zhu, Sadjad Fouladi, James Hong, Keyi Zhang, Philip Levis, and Keith Winstein. 2020. Learning in situ: a randomized experiment in video streaming. In *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*. 495–511.

[48] Li Yang, Mai Xu, Yichen Guo, Xin Deng, Fangyuan Gao, and Zhenyu Guan. 2021. Hierarchical Bayesian LSTM for Head Trajectory Prediction on Omnidirectional Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2021).

[49] Ye Yuan, Xinshuo Weng, Yanglan Ou, and Kris M. Kitani. 2021. AgentFormer: Agent-Aware Transformers for Socio-Temporal Multi-Agent Forecasting. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. 9813–9823.

[50] Xue Zhang, Gene Cheung, Yao Zhao, Patrick Le Callet, Chunyu Lin, and Jack Z. G. Tan. 2021. Graph Learning Based Head Movement Prediction for Interactive 360 Video Streaming. *IEEE Transactions on Image Processing* 30 (2021), 4622–4636.

ONLINE APPENDIX

Table 1. Prediction error over all $s \leq H$ on the MMSys18 dataset. Lowest prediction error for a given K is underlined, lowest prediction error for all K is highlighted in **bold**.

Method		Average prediction error				
		$s \leq 1s$	$s \leq 2s$	$s \leq 3s$	$s \leq 4s$	$s \leq 5s$
VPT360 (reported) ($K = 1$)		0.239	0.438	0.603	0.726	0.809
Deep-position-only ($K = 1$)		0.261	0.450	0.598	0.721	0.818
MANTRA-adapted	$K = 1$	0.333	0.621	0.828	0.967	1.066
	$K = 2$	0.296	0.515	0.651	0.743	0.824
	$K = 3$	0.290	0.472	0.575	0.645	0.717
	$K = 4$	0.287	0.453	0.539	0.592	0.659
	$K = 5$	0.274	0.433	0.515	0.566	0.625
DVMS (ours)	$K = 1$	0.245	0.432	0.581	0.700	0.790
	$K = 2$	0.262	0.424	0.516	0.566	0.613
	$K = 3$	0.228	0.372	0.439	0.465	0.501
	$K = 4$	0.218	0.352	0.402	0.418	0.452
	$K = 5$	**0.216**	**0.343**	**0.386**	**0.397**	**0.432**

Table 2. Prediction error over all $s \leq H$ on the CVPR18 dataset. Lowest prediction error for a given K is underlined, lowest prediction error for all K is highlighted in **bold**.

Method		Average prediction error				
		$s \leq 1s$	$s \leq 2s$	$s \leq 3s$	$s \leq 4s$	$s \leq 5s$
Deep-position-only ($K = 1$)		0.369	0.529	0.637	0.713	0.768
MANTRA-adapted	$K = 1$	0.351	0.594	0.767	0.887	0.981
	$K = 2$	0.323	0.503	0.608	0.678	0.755
	$K = 3$	0.298	0.456	0.544	0.598	0.656
	$K = 4$	0.292	0.433	0.500	0.543	0.596
	$K = 5$	0.303	0.426	0.487	0.533	0.581
DVMS (ours)	$K = 1$	0.200	0.355	0.470	0.555	0.618
	$K = 2$	0.200	0.326	0.394	0.435	0.477
	$K = 3$	0.190	0.305	0.357	0.383	0.419
	$K = 4$	0.187	0.295	0.337	0.355	0.387
	$K = 5$	**0.186**	**0.287**	**0.321**	**0.335**	**0.366**

Table 3. Prediction error over all $s \leq H$ on the PAMI18 dataset. Lowest prediction error for a given K is underlined, lowest prediction error for all K is highlighted in **bold**.

Method		Average prediction error				
		$s \leq 1s$	$s \leq 2s$	$s \leq 3s$	$s \leq 4s$	$s \leq 5s$
Deep-position-only ($K = 1$)		0.140	0.239	0.311	0.361	0.396
MANTRA-adapted	$K = 1$	0.236	0.429	0.571	0.666	0.736
	$K = 2$	0.211	0.343	0.426	0.479	0.530
	$K = 3$	0.202	0.313	0.375	0.417	0.457
	$K = 4$	0.186	0.291	0.351	0.389	0.428
	$K = 5$	0.194	0.290	0.342	0.378	0.413
DVMS (ours)	$K = 1$	0.135	0.233	0.304	0.353	0.388
	$K = 2$	0.127	0.207	0.253	0.284	0.313
	$K = 3$	0.128	0.202	0.238	0.262	0.289
	$K = 4$	**0.124**	0.192	0.224	0.244	0.271
	$K = 5$	0.125	**0.189**	**0.218**	**0.235**	**0.258**

Table 4. Prediction error over all $s \leq H$ on the MM18 dataset. Lowest prediction error for a given K is underlined, lowest prediction error for all K is highlighted in **bold**.

Method		Average prediction error				
		$s \leq 1s$	$s \leq 2s$	$s \leq 3s$	$s \leq 4s$	$s \leq 5s$
Deep-position-only ($K = 1$)		0.183	0.301	0.392	0.466	0.529
MANTRA-adapted	$K = 1$	0.221	0.411	0.572	0.693	0.779
	$K = 2$	0.204	0.341	0.431	0.499	0.565
	$K = 3$	0.204	0.330	0.402	0.450	0.501
	$K = 4$	0.202	0.310	0.366	0.409	0.456
	$K = 5$	0.199	0.302	0.356	0.391	0.435
DVMS (ours)	$K = 1$	0.159	0.282	0.377	0.453	0.514
	$K = 2$	0.155	0.262	0.326	0.369	0.410
	$K = 3$	0.156	0.254	0.302	0.330	0.364
	$K = 4$	**0.148**	**0.236**	0.275	0.298	0.332
	$K = 5$	0.149	0.237	**0.273**	**0.292**	**0.322**

Table 5. Prediction error over $s \leq 5s$ when training and testing on different datasets. For a given test dataset and a given K, the lowest prediction error is highlighted in **bold**, the second lowest prediction error is underlined.

		Test dataset			
	Train dataset	MMSys18	CVPR18	PAMI18	MM18
$K = 1$	MMSys18	0.790	0.695	0.466	0.706
	CVPR18	**0.778**	**0.618**	0.397	0.561
	PAMI18	0.789	0.643	**0.388**	0.718
	MM18	0.881	0.796	0.802	**0.514**
$K = 2$	MMSys18	0.613	0.569	0.420	0.542
	CVPR18	**0.581**	**0.477**	0.321	0.437
	PAMI18	0.604	0.493	**0.313**	0.501
	MM18	0.698	0.606	0.530	**0.410**
$K = 3$	MMSys18	**0.501**	0.491	0.371	0.414
	CVPR18	0.503	**0.419**	0.295	**0.362**
	PAMI18	0.530	0.442	**0.289**	0.440
	MM18	0.597	0.533	0.487	0.364
$K = 4$	MMSys18	**0.452**	0.444	0.350	0.354
	CVPR18	0.460	**0.387**	0.275	0.341
	PAMI18	0.476	0.404	**0.271**	0.407
	MM18	0.531	0.481	0.435	**0.332**
$K = 5$	MMSys18	0.432	0.418	0.330	0.347
	CVPR18	**0.427**	**0.366**	0.261	**0.321**
	PAMI18	0.451	0.385	**0.258**	0.376
	MM18	0.483	0.434	0.364	0.322

Table 6. Computational cost of the different models.

Method	# parameters	Single sample prediction time (ms)
Deep-position-only	4.21M	46.98
VPT360	6.3M	N/A
MANTRA-adapted	**76k**	6.81
DVMS (ours)	110k	**5.87**

Table 7. Memory size (in number and percentage of training samples) of the MANTRA-*adapted* method for different number of predicted trajectories K across all the datasets.

Dataset	MMSys18	CVPR18	PAMI18	MM18
Training set size	12600	560342	271440	32160
K=1	6413 (50.90%)	258758 (46.18%)	79503 (29.29%)	10520 (32.71%)
K=2	3601 (28.58%)	142113 (25.36%)	34564 (12.73%)	4575 (14.23%)
K=3	2041 (16.20%)	83702 (14.94%)	20059 (7.39%)	2489 (7.74%)
K=4	1227 (9.74%)	50265 (8.97%)	10470 (3.86%)	1431 (4.45%)
K=5	**811 (6.44%)**	**36325 (6.48%)**	**7632 (2.81%)**	**797 (2.48%)**

Algorithm 1 Simplified *BaselineABR* logic

Input: Available bandwidth budget b, Tile scores $\alpha_{s,t}$, Indices of empty (segments, tiles) in buffer (S, T)
Parameters: Minimum buffer level B_{min}, Quality levels Q_k, $k = 1, ..., 5$, Score threshold $p = 0.2$
Output: Download schedule skd
$(S_{min}, T_{min}) = s_t < B_{min}, s_t \in (S, T)$ ▷ Indices of empty (segments, tiles) inferior to B_{min}
if $\text{Cost}(S_{min}, T_{min}, Q_1) > b$ **then**
    $skd \leftarrow S_{min}, T_{min}, Q_1$ ▷ Request all under B_{min} anyway
else
    $skd \leftarrow S, T, Q_5$ ▷ Initialize schedule with max quality
    while $\text{Cost}(skd) > b$ **do**
        $S, T, Q_k \leftarrow skd$
        $Q_k = \begin{cases} Q_k & \text{if } \alpha_{S,T} > p \textbf{ or } (Q_k == Q_1 \textbf{ and } s_t \in (S_{min}, T_{min})) \\ Q_{k-1} & \text{otherwise (remove from schedule if already min quality)} \end{cases}$
        $p = \min(p + 0.2, 1)$
    end while
end if
$skd \leftarrow \text{Sort}(skd, \alpha_{s,t})$ ▷ Sort with highest tile scores first

Table 8. Visual quality gains (in %) over *NoPred* for all segments during simulations for different values of B_{min}. We report average and median gains in the "Avg. / Med." columns. We report the proportion of segments (in %) for which there was an increase / decrease in viewport quality over *NoPred* in the "Inc. / Dec." columns (some segments keep the same quality). Best results are highlighted in **bold**, second best are underlined.

	$B_{min} = 1$		$B_{min} = 2$		$B_{min} = 3$		$B_{min} = 4$		$B_{min} = 5$	
	Avg. / Med.	Inc. / Dec.	Avg. / Med.	Inc. / Dec.	Avg. / Med.	Inc. / Dec.	Avg. / Med.	Inc. / Dec.	Avg. / Med.	Inc. / Dec.
StaticPred	44.3 / 11.1	54.7 / 22.2	39.8 / 4.8	51.2 / 22.1	36.1 / 0.0	48.7 / 22.1	32.6 / 0.0	45.3 / 21.8	29.1 / **0.0**	42.2 / 21.5
DVMS, $K = 1$	46.7 / 14.3	57.2 / 20.4	41.4 / 9.6	53.4 / 20.3	37.3 / 1.9	50.6 / 20.2	34.0 / 0.0	47.3 / 20.0	31.0 / **0.0**	44.5 / 19.6
DVMS, $K = 5$	48.0 / 20.4	61.2 / 20.0	42.3 / 14.3	58.0 / 19.8	36.9 / 9.7	54.1 / 19.6	33.6 / 2.2	50.8 / 19.3	30.4 / **0.0**	47.5 / 18.9
DVMS, $K = best$	**53.1** / **25.0**	**62.8** / **18.4**	**47.7** / **17.1**	**59.6** / **18.2**	**43.2** / **12.2**	**56.1** / **18.1**	**39.8** / **6.6**	**52.6** / **17.8**	**36.2** / **0.0**	**49.3** / **17.5**

Table 9. QoE gains (in %) over *NoPred* for all simulations for different values of B_{min}. We report average and median gains in the "Avg. / Med." columns. We report the proportion of simulations (in %) for which there was an increase in QoE over *NoPred* in the "Inc." columns. Best results are highlighted in **bold**, second best are underlined.

	$B_{min} = 1$		$B_{min} = 2$		$B_{min} = 3$		$B_{min} = 4$		$B_{min} = 5$	
	Avg. / Med.	Inc.	Avg. / Med.	Inc.	Avg. / Med.	Inc.	Avg. / Med.	Inc.	Avg. / Med.	Inc.
StaticPred	14.3 / 7.8	65.8	13.2 / 7.1	65.0	12.2 / 6.1	63.3	11.0 / 5.3	62.2	10.0 / 4.5	61.0
DVMS, $K = 1$	16.2 / 9.8	70.0	14.8 / 8.9	68.9	13.8 / 7.9	67.5	12.9 / 7.2	66.5	12.0 / 6.7	65.6
DVMS, $K = 5$	15.6 / 10.0	71.9	14.2 / 9.2	71.1	12.8 / 8.1	69.3	12.1 / 7.6	68.8	11.3 / 7.1	67.9
DVMS, $K = best$	**19.7** / **13.2**	**77.9**	**18.4** / **12.4**	**77.3**	**17.4** / **11.4**	**76.2**	**16.6** / **10.9**	**75.7**	**15.5** / **10.2**	**74.7**