Viewing the 360° Future: Trade-Off Between User Field-of-View Prediction, Network Bandwidth, and Delay

Shahryar Afzal
Computer Science and Engineering
University of California, Riverside
Riverside, CA, USA
safza001@ucr.edu

Jiasi Chen
Computer Science and Engineering
University of California, Riverside
Riverside, CA, USA
jiasi@cs.ucr.edu

K. K. Ramakrishnan
Computer Science and Engineering
University of California, Riverside
Riverside, CA, USA
kk@cs.ucr.edu

***Abstract*—Predicting a user's field-of-view (FoV) accurately can help to significantly reduce the high bandwidth requirements for 360° video streaming, as it enables sending only the tiles corresponding to the predicted FoV. Since many approaches for user head-orientation (*i.e.,* FoV) prediction have been proposed in the literature, ranging from simple linear regression to more complex neural networks, it is difficult to comprehensively decide which method to use. Towards resolving this gap in knowledge, in this work we benchmark user prediction algorithms over an aggregation of multiple datasets and study the implications of this analysis. Our results demonstrate that it is indeed difficult for any prediction algorithm to accurately predict a user's FoV beyond a very short future time window of approximately 300 ms. We also observe that users' viewing behavior is dominated by sideways head movement, rather than up-and-down. These findings have implications on network bandwidth, latency, and playback buffering at the client: (1) Extra "padding" tiles are needed around the user's FoV in order to correct for prediction errors; in particular, a rectangular padding achieves lower stall rate than square padding, for the same bandwidth usage; (2) Video playout buffers, network delay, and jitter need to be small in order to avoid stale predictions of the user's field-of-view, which are only valid 300 ms into the future; (3) Per-video and per-user personalization of the padding can save bandwidth for slow-moving users or videos. We mathematically quantify these tradeoffs and present simulation results to demonstrate these findings and implications. Our results have implications for FoV prediction methods in future 360° streaming systems.**

I. Introduction

360° videos provide a more immersive user experience, and entertainment providers are increasingly offering 360° videos on their online platforms (*e.g.,* YouTube, CNN). However, from a networking perspective, 360° videos pose a significant challenge as they require substantial amounts of bandwidth (20-30 Mbps) in comparison to standard two-dimensional (3-6 Mbps). There have been a number of efforts to reduce the bandwidth consumption for delivering 360° videos [1], [2], [3]. One of the most common approaches is to deliver only a portion of the 360° video. As opposed to regular videos where the whole scene is viewed by the user, in 360° videos, only a portion of the scene – the field of view (FoV) – is viewed by the user at any given time, reducing the bandwidth usage by almost 80% [4], [5]. This partial viewing is supported by modern video codecs such as HEVC [6], [4], which allow for partitioning the video into tiles (illustration in Fig. 1).

Being responsive to what the user is currently viewing in her FoV requires the system to support an extreme level of interactivity: the user's head orientation needs to be determined, the FoV computed, and the corresponding tiles of the 360° video delivered from the server. Academia and industry have been actively studying FoV prediction (*i.e.,* how to predict the user's head orientation at some point in the future), and using this to request the tiles of the 360° video encompassing the expected FoV of the user, thereby both reducing the amount of bandwidth consumed while at the same time ensuring that the user's quality-of-experience (QoE), in terms of infrequent stalls, is maintained [7], [8], [9], [10]. Most of the systems designed to efficiently stream 360° videos [2], [11], [1] depend on accurate user FoV predictions.

Challenges: Predicting the user's head orientation/FoV and choosing which portions of the video to transmit is challenging, especially as we seek to predict further and further ahead into the future if the system has a large playback delay. These challenges are caused by the erratic nature of the user's head movement, as the user's head orientation depends on a number of characteristics, such as the user's usual behavior itself (relatively active users vs. passive users), the nature of the video (content with lot of action and movement vs. relatively static content), objects and actions of interest in the content, location of those interesting objects in the content (always in front vs. being located all around), the amount of inherent motion in the content, etc. Tiling of the video helps to an extent, because of the coarser spatial granularity at which the prediction can be made, but cannot fully alleviate user prediction challenges.

Related work: There have been a number of methods proposed to predict the user's head orientation or FoV. These approaches range from simple linear regression methods [2], [1], [12] or k-nearest neighbor [7], [8] to more sophisticated machine learning such as decision trees or neural networks [7], [9]. The inputs to these prediction methods are typically the recent history of head orientation of the current user, and optionally the historical data of previous users viewing the same 360° video in the past, and the output is the predicted head orientation/FoV of the current user. These prediction methods have been typically designed and evaluated ad hoc across disparate datasets, both public [13], [14], [10] and

proprietary [7], [12], [2], [1].

Goals: Thus, the first goal of this work is to systematically benchmark the performance of the various user prediction methods across a large pool of publicly available datasets, to establish useful guidelines on which user prediction methods provide the best performance, and is the foundation for the rest of the paper.

The second goal of this work is to understand how user behavior observed in these datasets can provide opportunities to improve user QoE. Our benchmarking of the user prediction methods reveals that the prediction accuracy drops very quickly as we increase how much into the future we seek to predict. One common approach to compensate for such prediction inaccuracies is to introduce a certain amount of padding around the FoV, and transmit these additional padding video tiles also, at the expense of additional bandwidth consumption. We leverage two key observations of user behavior to optimize this padding: (a) Users tend to look side-to-side rather than up-and-down, and (b) Users who are very active when viewing one 360°video also tend to be active when viewing other 360°videos. We leverage these two observations to customize the padding shapes and sizes, achieving fewer stalls for the same bandwidth consumption, or less bandwidth consumption for the same stall frequency.

The third goal of this work is to understand the impact of the aforementioned low prediction accuracy on the 360°video playout delay and acceptable network latencies. Typically, the playout delay/buffer is used to accommodate the latency for obtaining the video tiles and, more importantly, "ride out" the network delay variations (jitter) in the delivery of the 360°video. How far ahead the FoV prediction method is expected to predict is related to the playout delay size, which is in turn a function of the network round-trip time and the de-jitter budget. To keep the head orientation prediction accuracy within reasonable levels, we argue that the playout delay plus network round trip time needs to be very small (∼300 ms from our results). Hence, the network delay and jitter also have stringent upper bounds. To quantify these network requirements, we provide formulas to describe the above tradeoffs between the playout delay, network delay, and network jitter. While existing 360°streaming systems set the playout buffer in an ad hoc way based on typical configuration parameters, such as the video chunk length of 1-4 seconds [12], in this work we argue that the playout delay should instead be set based on its relationship with FoV prediction accuracy and network delay. We believe this argument is one of the strongest reasons for requiring a very low latency, low jitter network, as well as a small playout delay, for effectively streaming 360°videos.

In summary, our main contributions are as follows:

- We systematically compare the predictability of the user's head orientation across several previously proposed prediction methods that rely on user history. We show that the ability to predict the user's head orientation further into the future (e.g., 30 or 60 frames into the future) is extremely poor across all of the prediction algorithms, including popular methods such as LSTM neural networks and linear regression. Rather, prediction methods are only effective over a very short-term lookahead future of ∼300 ms. (§II-B)
- Based on the observation that users tend to move their heads sideways more than up and down, we show that sending additional padding tiles in a rectangular shape around the user's FoV is more effective than a square padding [12], [15]. These rectangular paddings provide operating points at the Pareto-frontier of the stall-bandwidth tradeoff; in other words, with rectangular paddings, we can reduce the number of stalls for the same bandwidth, or consume less bandwidth for the same number of stalls. (§III)
- Based on the observation that users' activity level (in terms of head movement) tend to be consistent across videos, we propose that padding size be personalized per user. Smaller rectangular paddings can be used for low-activity users, providing bandwidth savings. Additionally, we study the possibility of customizing padding per-video, but find that per-user customization provides larger bandwidth savings for the same stall rate. (§IV)
- Since our comparison of FoV prediction methods reveals that we can only predict over the very short term, any reduction of the bandwidth for delivering 360°video to the user has to be based on the very short term expectation of what the user's FoV will be. This requires the playout delay at the client be very small, and the round-trip latency to the server, where the requested content is located, plus the network delay variation (jitter) has to be correspondingly small. We characterize the relationship between network delay, jitter, playout delay, and how far into the future the FoV needs to be predicted. (§V)

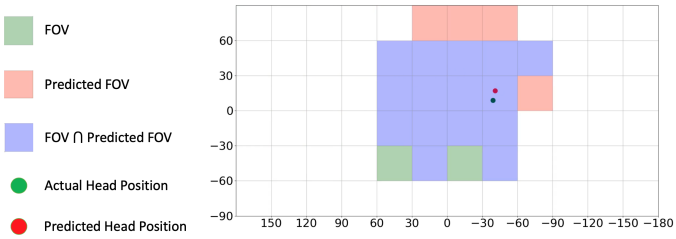


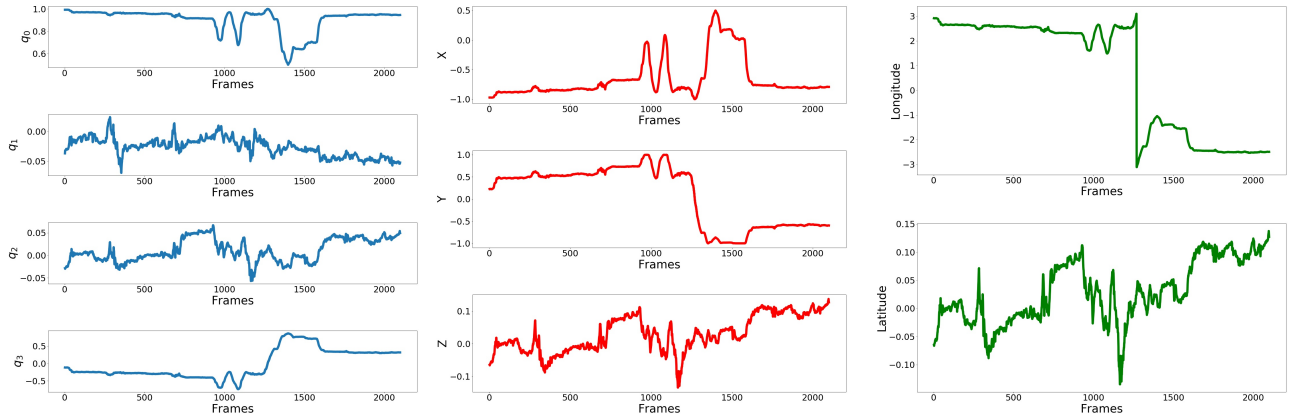
Fig. 1: Example relationship between a user's actual FoV, and the user's predicted Fov, with tiling. The user's FoV is projected from the sphere onto a 2D plane, resulting in the irregular shapes shown above in the figure.

II. Effective User Head-Orientation Prediction

In this section, we introduce the datasets, metrics, and head orientation prediction methods. We then compare the effectiveness of these prediction methods that are commonly considered in the 360°video delivery arena.

A. Setup

1) Datasets: We use three publicly available datasets. Each dataset contains the head orientation traces of some number of users watching 360°videos.



(a) Quaternions (q_0, q_1, q_2, q_3) (b) Cartesian coordinates (x, y, z) (c) Longitude/latitude

Fig. 2: Examples of different representations of the same head orientation trace. The same trace looks quite different under different representations, which can then affect the prediction method.

- **Dataset 1 [13]:** There are 59 users watching 7 different videos, with average video length of 3.5 minutes.
- **Dataset 2 [14]:** There are 48 users watching 18 videos, with average video length of 3.8 minutes. In the first 9 videos, users are allowed to explore the videos as they wish, while in the latter 9 videos, users are told they will be quizzed on the video content. Unless otherwise specified, we consider the first 9 videos in this work in order to be comparable with datasets 1 and 3.
- **Dataset 3 [10]:** There are 45 users watching 208 videos. Each video is watched by a subset of all users (ranging from 28-34 users). The video length ranges from 20-60 seconds.

Head-orientation representation: There are multiple ways to represent the user's head orientation, including *angular orientation* (used by [2], [11], [9]), latitude/longitude, *Cartesian coordinates* on the sphere or video plane (used by [10]), *Quaternions* (used by [13], [14]), and *Axis-angle* representations. Latitude/longitude is similar to angular orientation (yaw, pitch, and roll), except neglecting the roll dimension (which is empirically uncommon among users). Quaternion and axis-angle representations are commonly used in virtual reality [16], and represent rotation in 3D space by defining a vector (axis of rotation) $\mathbf{v} = (v_1, v_2, v_3)$ and an angle θ. The axis-angle representation is written as (θ, v_1, v_2, v_3), while the quaternion representation is defined as $\left(\cos\left(\frac{\theta}{2}\right), v_1 \sin\left(\frac{\theta}{2}\right), v_2 \sin\left(\frac{\theta}{2}\right), v_3 \sin\left(\frac{\theta}{2}\right)\right)$. Examples of different representations are shown in Fig. 2.

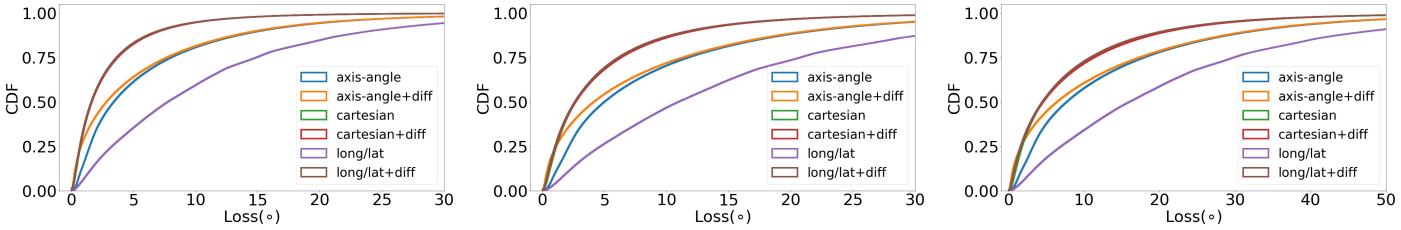
Dataset pre-processing: We have undertaken a number of steps to pre-process the datasets. In our experience, these pre-processing steps, particularly the head orientation representation and time series differencing, can have a significant impact on the performance of the prediction methods (see Sec. II-B).

1) **Interpolation:** Each dataset records the head orientations with a different sampling frequency. Since we are doing per-frame prediction, we desire one data point per frame; therefore, we remove extra data points per frame, and also perform linear interpolation (using the SLERP method [17]) to make the sampling frequencies consistent across the three datasets.
2) **Conversion to common representation:** As discussed in Section II-B, we convert the datasets to a common head orientation representation - namely, the Cartesian representation.
3) **Differencing:** Differencing (*i.e.,* computing the difference between the current data point and the most recent data point) [18] is a well-known time series analysis technique, and helps by making the time series have a more stable mean and variance, which is then easier to model/predict.
4) **Remove discontinuities:** Because the head rotations are limited to a specific range, *e.g.,* [-180°, +180°] for longitude or [-1, +1] for the axis-angle representation, we apply a modulo of the interval length to the input data to avoid these discontinuity issues.

2) Evaluation Metrics: The evaluation metrics considered in this paper are the following:

- ***Prediction Loss (in °)***: Prediction loss is defined as the angular distance (smallest angle) between the actual and the predicted user's head orientation.
- ***Stalls***: We consider a stall to occur if any part of the user's FoV is not available in the playout buffer, and requires the client to request the missing information from the server.
- ***Bandwidth consumed (kbps)***: The bitrate of each tile is estimated as the total bitrate of the video divided by the total number of tiles. The total bandwidth consumed is the sum of the bitrates of all the tiles sent to the user.

3) User Head-Orientation Prediction Methods: We have chosen a variety of prediction methods to estimate the users' head orientation, ranging from simple methods such as linear regression and k-nearest neighbors, to machine learning methods such as decision trees and neural networks. Our goal is capture the spirit of the main methods proposed in the literature, and perform apples-to-apples comparison of them.



(a) Lookahead = 7 (b) Lookahead = 10 (c) Lookahead = 15

Fig. 3: CDF of prediction loss for different head orientation representations and lookaheads values. Cartesian coordinates plus time differencing has the lowest prediction loss.

All prediction methods are trained on each video's data using a randomly selected subset of training users (70%), and tested on the remaining users (30%). We also perform k-fold cross validation, with the value of $k = 5$.

- **Naive** (used by [15]): This baseline prediction method chooses the current head orientation as the predicted user's head orientation at a future time instant.
- **Linear or Ridge Regression (RR)**: (used by [2], [12], [1]): Linear regression is a simple and popular method for user head orientation prediction. In ridge regression, a regularization parameter is added that tends to avoid complex models with many parameters, in order to avoid overfitting, while still achieving high prediction accuracy.
- **K-nearest neighbors (KNN)**: (used by [7], [8]) This method finds the K historical users, across all time, whose subsequence of head orientations are closest to the test user's current head orientation subsequence. The average of these K historical users' head orientations is output as the predicted future head orientation of the test user.
- **Decision trees (XGBoost)**: (used by [7]) XGBoost is a popular machine learning library [19] using decision trees. Since XGBoost by default only supports scalar outputs predictions, we train and predict each individual coordinate of the head orientation separately.
- **Long short-term memory (LSTM)**: (used by [11], [7], [10]) LSTM is a neural network architecture geared towards time series prediction. It consists of cells and gates, and uses feedback loops to maintain state and so produce better predictions than the standard feedforward neural networks.

***Hyperparameter selection*:** For each of the prediction methods described above, there are a number of hyperparameters that can be set/optimized. Unless otherwise noted, we perform grid search to find the best hyperparameter values. The hyperparameters common to all prediction methods include:

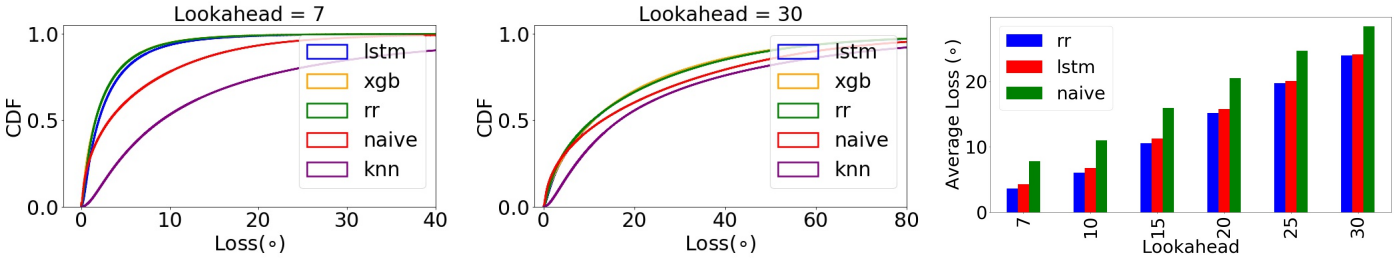
- ***Lookahead (L)***: The lookahead value specifies how many frames into the future the head orientation prediction method should predict.
- ***History window (H)***: The history window is how many frames in the past should be used to predict the future head orientation of the user. In the case of KNNs, the history window is defined as the subsequence length. A larger window size provides more information about the user and potentially better predictions, but increases the computational complexity.

There are also several hyperparameters specific to each prediction method. Based on the grid search, for KNN, we selected $K = 5$ and $H = 1$. For XGBoost, $H = 10$. For Ridge Regression, the penalty weight (α) is selected from 1, 0.1, 0.01, 0.001, 0.0001, and $H = 30$. For LSTM, $H = 10$, and there are two LSTM layers (the size of each layer and the learning rate is determined by the grid search) and one output layer (a dense layer with 4 neurons).

B. Comparison of User Prediction Algorithms

In this subsection, we benchmark the performance of the user head orientation prediction algorithms across datasets, carefully examining the impact of various parameters such as head orientation representation and prediction lookahead on the prediction loss. Our main findings are that: (1) Head orientation representation and time differencing significantly impacts prediction loss; (2) LSTMs, linear regression, and decision trees have very similar prediction loss across all the considered user prediction algorithms; and (3) Prediction loss strongly depends on how far into the future the prediction algorithm needs to predict (*i.e.,* the lookahead value). Very low lookahead values (*e.g.,* 10 frames $\sim$300 ms) are needed to achieve prediction loss of less than 7°.

Impact of head orientation representation: Fig. 2 shows an example of the same user trace plotted with different representations. We can see that depending on the representation, the data values change over time more smoothly or with higher variance, which can impact the ease of prediction. Therefore we compare the user head orientation prediction loss with the different head orientation representation formats using ridge regression in Fig. 3 (results for other prediction methods are similar). The cases where differencing is applied is labeled with "+ diff". The Cartesian coordinates representation has the lowest loss, followed by axis-angle and longitude/latitude. This pattern persists across all lookahead values Fig. 3. Therefore, we proceed with using the Cartesian representation (+diff) for all datasets. Note that while differencing is a common technique in time series analysis, it is typically not considered by existing FoV prediction methods [2], [11], [1]. Using differencing results in a prediction loss improvement of approximately 5° on average across all lookahead values



(a) Comparison of user prediction algorithms (lookahead L = 7).

(b) Comparison of user prediction algorithms (lookahead L = 30).

(c) Impact of prediction lookahead.

Fig. 4: Comparing loss of the various prediction methods across all datasets. LSTM, ridge regression, and decision trees have the lowest loss. Prediction loss is strongly dependent on how far into the future the prediction method needs to predict.

and representations, and hence we recommend the use of differencing in future 360°streaming.

Which prediction method has the lowest loss? We next compare the prediction loss of the various user prediction methods across all users and datasets. The CDF of the loss of each frame across all videos and users are shown in Figs. 4a and 4b, for two different lookahead values. The median loss of ridge regression, XGBoost, LSTM, naive, and KNN is 24.0°, 24.0°, 24.1°, 28.5°, and 34.1°, respectively, for a 30-frame lookahead, with similar patterns for a 7-frame lookahead. We observe that the prediction methods can be split into two clusters: one cluster with relatively high prediction accuracy (XGBoost, LSTM, RR, Naive), and the other cluster with low accuracy (KNN). Intuitively, the relatively large gap between the performance of the methods in the two clusters can be explained by the fundamental nature of these algorithms. KNN tries to find the most similar user(s) and copy the similar users' head orientations to make predictions. On the other hand, XGBoost, LSTM, and linear regression can model user behavior through their internal parameters (determined during the training process), and find trends in user behavior, resulting in better predictions. Unless otherwise specified, for the remainder of this work, we use the Linear Regression model as it has very similar performance to the other "good" prediction methods (XGBoost and LSTM).

We also note that the Naive method has fairly good performance, somewhere in between the sophisticated linear regression and the KNN method. While simple, it has several advantages: (1) prediction has negligible latency and compute requirements, which can be helpful on resource-constrained mobile devices; (2) there is no training process, making this method usable for live videos that lack historical training data; and (3) for video-on-demand, if additional viewing data is collected from users, the naive method does not require re-training, unlike the other prediction methods which would require re-training to update the models. Thus the Naive method can be considered a good candidate for user head orientation prediction in 360°streaming systems.

Per-user comparison: We also investigate the prediction accuracy of individual users. Our hypothesis is that users with more head movement should tend to have higher prediction loss. To investigate this, we plot the average head movement

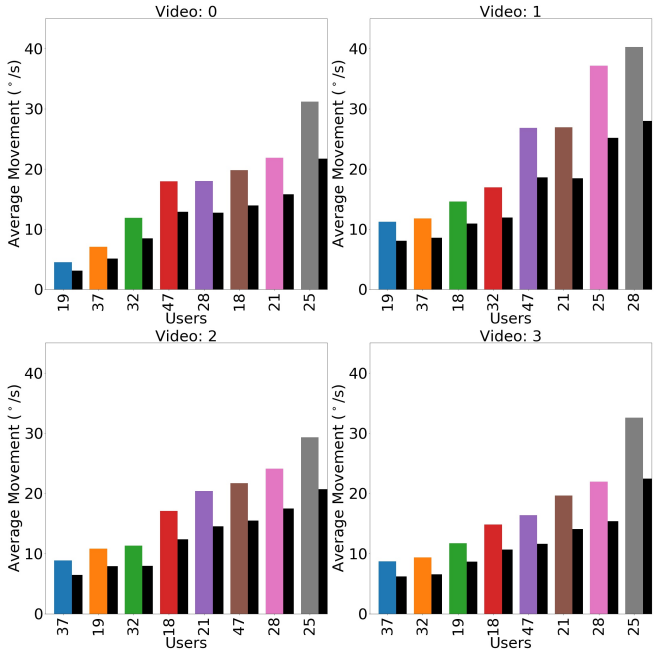
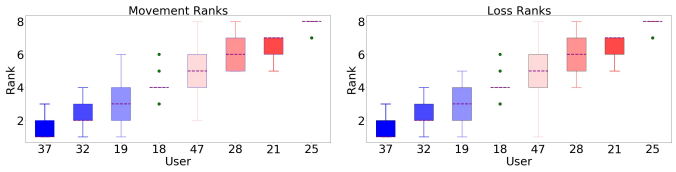


Fig. 5: Users' average head movement (colorful bars) is correlated with their prediction loss (black bars).

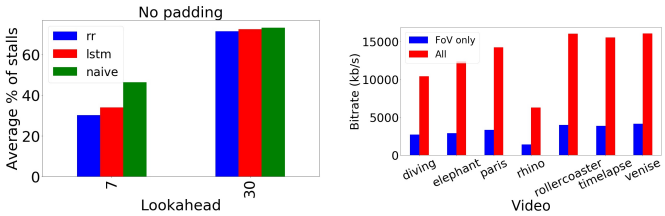
within one second for each user and each video, along with the corresponding prediction loss. We plot examples of 4 videos in Fig. 5 (for L=30). We make several observations: (1) The relative amount of head rotations for each of the different users remain about the same across different videos, *i.e.,* the users who move a lot in a given video tend to move a lot while viewing other videos as well. For example, user 37 has the lowest average movement in 2 of the 4 videos, and the second-lowest average movement in the remaining 2 videos. (2) Prediction loss is positively correlated with the magnitude of the user's movement. The black narrow bars in Fig. 5 show the average prediction loss, and their height appears to be positively correlated with the colorful bars representing the user's average head movement. This implies it is harder to predict the FoV of the users who tend to move a lot, while users who move less will have better prediction accuracy.

Combining the above two observations, users who tend to be difficult to predict in one video also tend to be difficult



(a) Rank of user movement across videos (b) Rank of user head orientation prediction loss across videos

Fig. 6: Rank of user movement and prediction loss (compared to their peers) across videos (dataset 2, experiment 1). Users tend to be consistent across videos, both in terms of user movement and prediction loss.



(a) % of stalls across all datasets, without padding

(b) Bandwidth savings of perfect prediction vs. sending all tiles, for select videos (dataset 1)

Fig. 7: Importance of FoV-based 360°streaming.

to predict in other videos. To investigate this further, we calculate the average head movement of each user for each video, and rank the user's head movement compared to her peers. We then plot the distribution (box plot) of each user's rank across videos. We repeat this calculation for each user's prediction loss as well, and show all the results in Fig. 6. As we can see, the order of the users are the same in both the movement ranking and the prediction loss ranking, suggesting the correlation between movement and prediction loss. This suggests opportunities for personalizing each user's tile selection policy, as users who have more head movements may require more extra padding tiles to account for their movements. This implication is discussed in Section IV.

Impact of lookahead on prediction loss: We next evaluate the impact of different prediction lookahead values on the prediction loss, *i.e.,* how far into the future can the user orientation prediction method predict? We sweep lookahead across several values of $L = [7, 10, 15, 20, 25, 30]$ frames, re-training each prediction method every time. The average prediction loss of LSTM, linear regression, and Naive for different lookahead values is shown in Fig. 4c (we focus on these three algorithms because they have relatively low prediction loss compared to the other methods, as discussed earlier in this section). We can see that short-term predictions, such as 7 frames ahead, result in lower prediction loss (*e.g.,* 3.6°average loss when $L = 7$ for RR); while, predicting further ahead into the future, such as 1-2 seconds ahead, results in higher loss (*e.g.,* 24°average loss when $L = 30$ for RR). Such a 24°prediction error in the vertical direction would, for example, result in approximately 24°/90°≈25% of the user's FoV area being missing, causing a stall while the system requests those missing tiles. Requesting the missing tiles to correct the stall incurs at least an extra round-trip time, plus the transmission and playout buffer delay. Even if some extra visual information from around the FoV (*i.e.,* padding) was sent to accomodate for the prediction error, this would still require a prediction accuracy of less than the size of that padding. For example, a default 15°of padding in the vertical direction [15] would require prediction loss of less than 15°to avoid a stall, on average. It is well known that users are highly sensitive to stalls [20]. So, the tolerable prediction loss should be much less than 15°to avoid a stall in the majority of cases.

In current practice, lookahead values are often set based on chunk duration. 360°videos are typically temporally split into chunks of 1-second long or more, so prior work on 360°streaming (*e.g.,* [2], [11], [12]) mainly sets the user prediction lookahead value to 1 second or more. For example, Petrangeli *et al.* [12] predicts 1 second into the future, while Flare [2] predicts 3 seconds into the future. However, based on our results, we argue that a lookahead value of 1 second or more is very difficult to predict accurately (*i.e.,* has high prediction loss of 24° when $L = 30$), potentially leading to stalls and wasted network bandwidth by sending wrongly predicted tiles. Hence we believe that user head orientation prediction methods should set a very short lookahead value, in order to avoid a number of associated problems: stalls, wasted network bandwidth and overall poor QoE.

III. Mitigating Network Bandwidth Needs and Stalls with FoV Padding

Building on predicting the user's head orientation, we now compute the tiles corresponding to the user's FoV based on her head orientation, and discuss how extra padding tiles can be used to overcome FoV prediction errors, at the expense of a carefully-managed increase in bandwidth usage. In particular, we leverage the natural tendency of users to look side-to-side rather than up-and-down and propose an asymmetric, wide padding shape to account for user prediction errors.

Tile computation: We split each video into 72 equal-sized tiles, with 6 tiles per column and 12 tiles per row of the rectangular video [7], and define the FOV as 90°× 90° [7]. To convert the user's head orientation, which is ouput by the user head orientation prediction methods, into the set of tiles actually viewed by a user, we take the point representing the center of the user's FoV, find the area on the surface of the sphere representing a 90°× 90°rectangle area (the FoV) around that point, and compute the corresponding region of the video using the equirectangular projection. Finally, we choose the video tiles overlapping with the specified region. An example is shown in Fig. 1.

Padding to reduce stalls: We first motivate why padding is needed to reduce the fraction of frames that may contribute to stalls. In our first experiment, we evaluate a baseline approach of sending only exactly the tiles needed to cover the FoV of the user, *i.e.,* a default 90°x 90°FoV. Fig. 7a illustrates how the lookahead (choosing 7 and 30 frame lookahead (which we select as being at the two ends of the range of reasonable

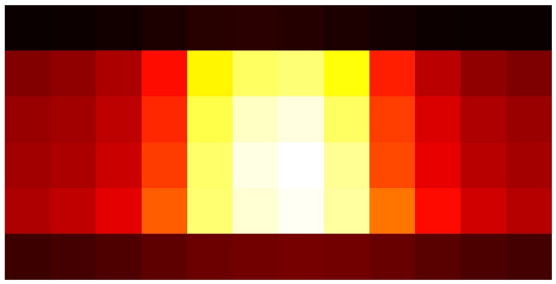


Fig. 8: Heatmap of all users' FoVs, across all datasets.

lookahead values based on our experiments)) impacts the percentage of frames that have a stall (across all videos, datasets, and head orientation prediction algorithms). Similar to the trend in Fig. 4c, as the lookahead increases, the fraction of stalls increases. This is because as the lookahead increases, the head orientation prediction loss increases, causing missing tiles and thus stalls, and the client has to request the missing tiles. Furthermore, we observe that the absolute values of the stall percentage is quite high (>30% of frames have stalls when lookahead is 7 frames, and more than 70%(!) of frames have stalls when the lookahead is 30 frames), suggesting that a baseline approach of only sending the minimum tiles needed to cover the FoV is clearly insufficient. Since prediction errors are likely inevitable, especially for predicting further into the future, additional padding tiles outside of the FoV need to be added to mitigate the number of stalls [12]. Adding more padding tiles decreases the risk of stalling, but increases the network bandwidth requirement. In the extreme case, the maximum size padding surrounding the predicted FoV would mean the client will download all the tiles for each frame.

Optimizing padding shape to trade bandwidth for stalls: Next, we study how to select the tiles that should be part of the padding. Going beyond the simple intuitive approach of selecting the tiles that are directly adjacent to the tiles in the user's FoV (*e.g.,* symmetrically encircling the FoV), we leverage a key insight about the users' viewing behavior: most of the movement results from users looking side-to-side. This is demonstrated in Fig. 8, which shows a heatmap of all videos from all three datasets. We find the tiles inside the FoV of each user and count the number of times each tile is present inside the FoV. The brighter a tile is in Fig. 8, the larger number of times it has appeared inside the FoV of the users. This suggests generally it is more likely that the users move along the horizontal axis (*i.e.,* change in longitude), and prediction errors are also more likely along the horizontal axis. Thus, padding horizontally around the user's FoV is likely to help reduce stalls more than adding padding vertically.

To show the implications of this, we plot the fraction of stalls vs. bandwidth for different padding shapes and sizes in Fig. 9. Bandwidth is normalized by the total bit rate of each 360°video, and stalls are normalized by the total number of frames in each video. The green and dashed red lines correspond to the same lookahead value (7, 10 or 15 frames) for the square and rectangular paddings. For example, consider a lookahead of $L = 10$ frames and a

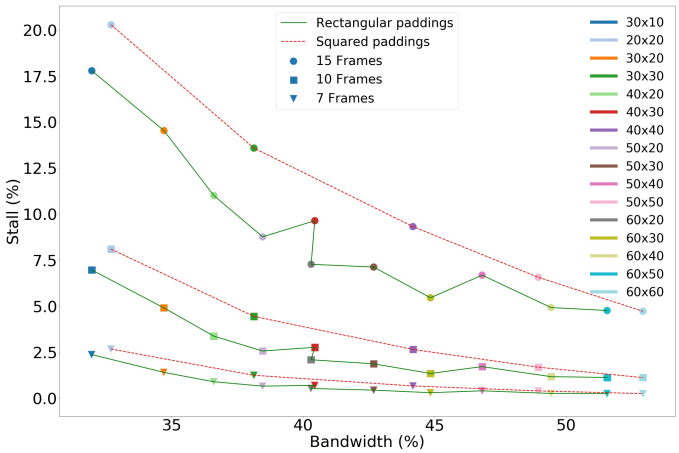


Fig. 9: Tradeoff between bandwidth usage and stalls, for different padding shapes and lookaheads. Rectangular paddings are at the frontier of the region, *i.e.,* they are Pareto-efficient.

30°×20° padding as the "baseline" (this padding is in addition to to the default FoV). A square 30°×30° padding represents expanding the FOV vertically, while a 40°×20° padding represents a horizontal expansion. The square 30°×30° padding decreases stalls by 0.46% compared to the baseline 30°× 20° padding, for a 3.4% increase in total bandwidth. However, the wide 40°×20°reduces stalls even further (1.53%), with a lesser amount of additional bandwidth (only 1.89% more bandwidth compared to the baseline) needed than the 30°× 30° padding. Similar arguments hold when comparing the 40°×30°padding with the 40°×40°padding (vertical expansion), and the 50°×30°padding (horizontal expansion). Overall, the horizontal, wide padding shapes tend to lie at the Pareto frontier of the tradeoff between bandwidth and stalls.

Having made the case for rectangular paddings (horizontal expansion), we now explore further the bandwidth/stall tradeoffs for different lookahead values. Intuitively, the padding shape has a much bigger role at higher lookahead values (*e.g.,* $L = 15$) , because higher lookahead values tend to result in higher prediction losses (Sec. II-B). For example, in Fig. 9, the rectangular 50°×20°padding, when $L = 7$ frames, saves 0.59% of stalls for the same bandwidth usage as compared to the square 30°×30°padding; making the same comparison when $L = 15$ frames, the rectangular padding saves 4.82% of stalls. However, no matter the lookahead value, the horizontal padding shapes (solid line in Fig. 9) tend to lie at the Pareto frontier of the bandwidth-stall tradeoff, and square padding shapes (dashed lines in Fig. 9) should be avoided since they are interior points in the tradeoff region. Plotting the results in another way, in Fig. 10, we see the average stall rates across all videos across all datasets for different lookahead values. We can easily see that the rectangular padding (50°×20°or 60°×30°) is more effective in terms of reducing the stall rate than the square padding (30°×30°) for the same bandwidth. This pattern is consistent across all lookahead values.

These results give us the opportunity to choose the padding size based on the the preferred values for stall rate and

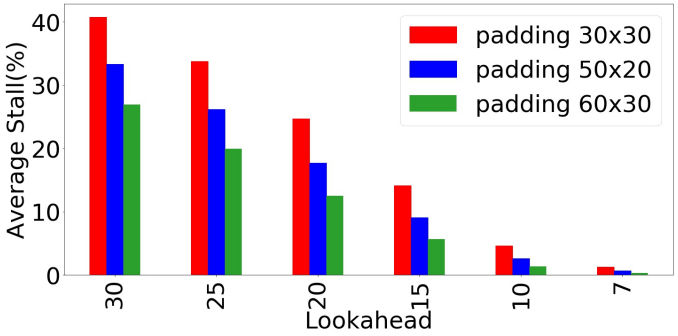


Fig. 10: Dependency between lookahead and stalls for different padding sizes (dataset 1).

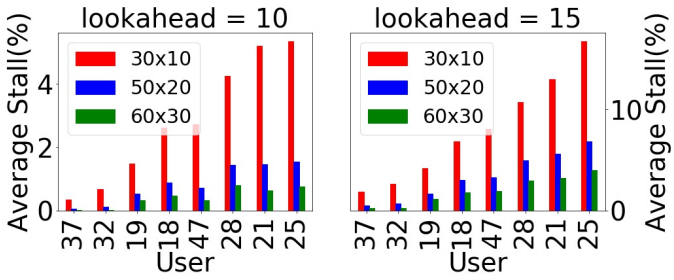


Fig. 11: Average stalls of selected users (dataset 2, exp. 1).

bandwidth usage. For example, if we choose to predict 10 frames ahead ($L = 10$) and want to have a stall rate of less than 5%, then we need a padding size of 30°×20°. In this case the bandwidth consumption is ∼35% (of the complete video's bandwidth). Or, if we choose to predict 15 frames ahead, which enables us to have a larger playback buffer, we will need at least a 60°×30°padding which consumes ∼45% bandwidth on average. For the remainder of this paper's experimental results, unless otherwise stated, we fix the padding shape and lookahead value. Based on the shape of the curve in Fig. 9, we focus on 50°×20°padding and $L = 10$ as achieving a good balance between the fraction of stalls and bandwidth usage, with less than 5% stall rate (closer to about 3%) and only 40% of the video's total bandwidth.

IV. Personalization Per User and Per Video

A. Per User Personalization

Given that we use machine learning techniques to predict the user head orientation and derive the expected FoV in the near future, we seek ways to exploit these predictions to mitigate the bandwidth demands even further. Recall that in Sec. II-B, we observed that the viewing behavior of different users is different, and a user that has less movement across one video has the same behavior across other videos. As a result, we seek to leverage these user level characteristics to customize the client's requests to the server for the corresponding video tiles. The primary customization we explore is how to provide variable-sized padding depending on the user. The goal is to ensure their QoE is preserved while maximizing bandwidth savings. Note that this personalization is enabled by prediction.

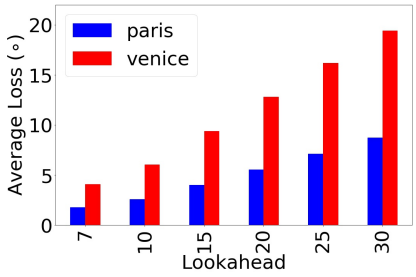


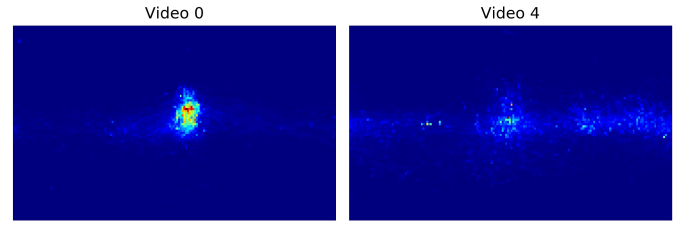
Fig. 12: Prediction loss is lower for the "Paris" video due to audio narration biasing the users' FoVs towards a common region in the video (dataset 1).

Without the guidance from the prediction algorithms (e.g., if one were to use the naive method [15]), it would be difficult to customize the padding on a per-user basis, requiring a separate means for learning/predicting individual user behavior.

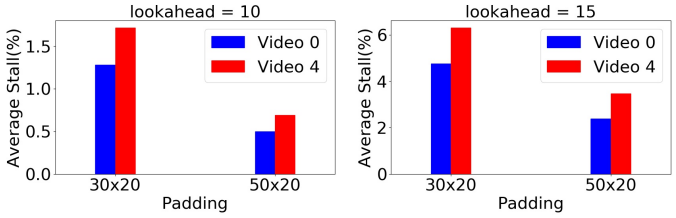
When the user does not move much, how much smaller can the padding be? We show in Fig. 11 that reducing the padding to a 30°×10° size still allows the users with small head movement (*e.g.,* users 37, 32, 19 from Sec. II-B) to have no more than 5% stalls for lookahead = 15. As a 50°×20°padding results in a 38.45% bandwidth usage (compared to streaming the entire 360°video), while a 30°×10°padding would only use 31.96% bandwidth, a ∼6% saving in bandwidth can be obtained as a result of personalization. Thus, customization gives us another lever for mitigating the significant bandwidth needs of 360 °video. This customization would have to be done online in an incremental manner, using features learned from the the user's historical behavior. This could be done in a holistic fashion, for example using neural networks (as used for non-360°video [21]) to both predict the FoV and which padding shape to select, or in a modular fashion, for example by creating a machine learning model to classify users as high or low activity users, mapping the classification result to padding size, and combining this with the FoV prediction to choose the set of tiles to deliver. In both cases, the system would have to continuously receive feedback about the user's behavior and adapt the padding size online as more information is collected.

B. Per Video Personalization

Guided videos: We can also consider adapting to individual videos instead of individual users, especially for delivering stored video, by learning the behavior of all of the users viewing a particular video. This is based on the intuition that some videos result in very active viewing patterns, while other videos result in more stationary viewing patterns across users. For example, Fig. 12 shows the average prediction loss of two videos from the Dataset 1. "Paris" and "Venice" are two videos of the same "sightseeing" genre. However, the average prediction loss is much lower for the video "Paris" than video "Venice". The reason is that Paris is an audio-guided video, where a narrator points out the interesting regions to watch in the video and the users, more or less, follow the guide's directions. Thus, for such videos where the users are biased



(a) Heatmap of users' FoVs



(b) Percentage of stalls

Fig. 13: User viewing behavior and stall probability for two videos. Video 0 has more concentrated user viewing patterns, and hence experiences fewer stalls, for different lookahead and padding sizes.

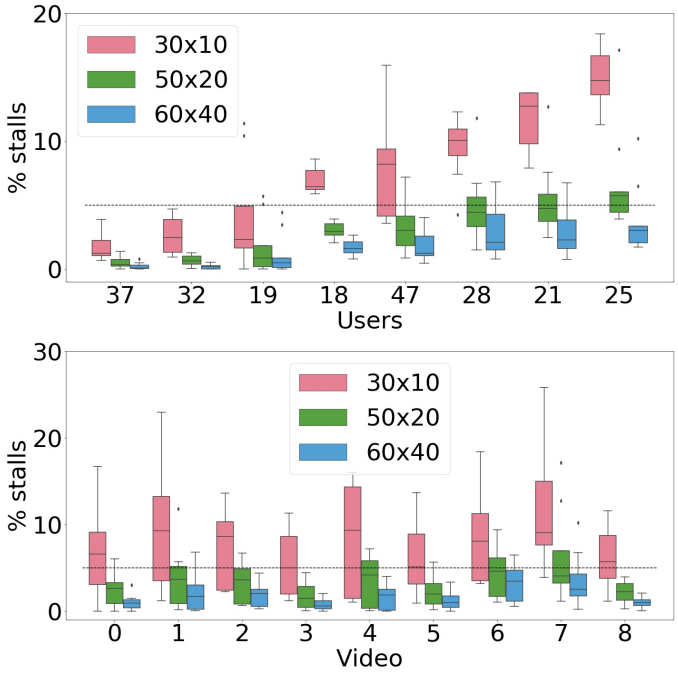


Fig. 14: Per-video vs. per-user customization

towards watching a specific region of the video or move in a certain direction, the prediction methods tend to have higher accuracy because they learn these patterns from the training process. Thus, it is easier to predict the user's head movement for some videos.

Videos with disparate viewing patterns: Some videos also tend to have different viewing patterns based on the nature of their content. To show this, we choose two specific "unbiased" videos from Dataset 2, experiment 1: video 0 and video 4. Fig. 13a shows the corresponding heat maps of the center of the users' FoVs. As we can see, in video 0, center of the FoVs are focused on one part of the frame, unlike video 4 where users view much more of the video, in the horizontal direction. Fig. 13b shows the average percentage of stalls across all users for videos 0 and 4, for several lookahead values. The difference between the stall results for these videos matches the inference we make from the heat maps, in that video 4 has a higher average stall percentage for the same padding size. Further, the figure indicates that video 4 might benefit from a larger padding (50°×20°) to get the stalls below 5% across all users, while for video 0, it may be sufficient to use a smaller (30×10) padding. However, the differences are not too large.

Besides padding, we also experimented with using features from the video itself, such as motion vectors and pixel intensities, to improve prediction accuracy and reduce stalls. However, despite extensive experiments incorporating such features into LSTMs, we were unable to find any performance improvement (typically incorporating motion features degraded prediction accuracy by 1-2°), and therefore we did not pursue content features further, and instead focused on customized padding per-video.

C. *Per-video vs. per-user personalization*

Which of the above aspects - per user or per video personalization - is more beneficial or dominant? We carefully examine the stalls experienced by the test users for 3 padding shapes, 30°×10°, 50°×20°, and 60°×30°, across all 9 videos in dataset 2, experiment 1. We show the box plots in Fig. 14. There are two takeaways from these results: (a) The variation in stall rate for the same user watching different videos is relatively small (small size of boxes in top plot of Fig. 14). For example, user 37 has very similar stall rates across all the 9 videos that this user watched. (b) The variation in stall rate for the same video watched by different users is relatively high (large size of boxes in bottom plot of Fig. 14). This is due to the fact that different users have different behavioral patterns when watching the same video. For these reasons, per-user customization seems to be more beneficial than customization on a per-video basis. For example, if we consider user 37 and try to pick a good personalized padding size, we can see that this user is not very active and a 30°×10° padding would suffice to keep the average stall rate below 5% (the horizontal dashed line), across videos watched by that user. The same applies to user 32 and 19. We need bigger padding sizes for more active users, *i.e.,* 50°×20°for users 18 and 47, and 60°×40°for users 28, 21, and 25.

On the other hand, choosing the padding size based on the video leads to a one size fits most approach, as all but 2 of the videos in Fig. 14 bottom plot require a 60°×40° padding size to stay under a 5% stall target (horizontal dashed line), missing on opportunities for bandwidth savings compared to per-user personalization, in which a 30°×10°or 50°×20° padding suffice for the majority of users.

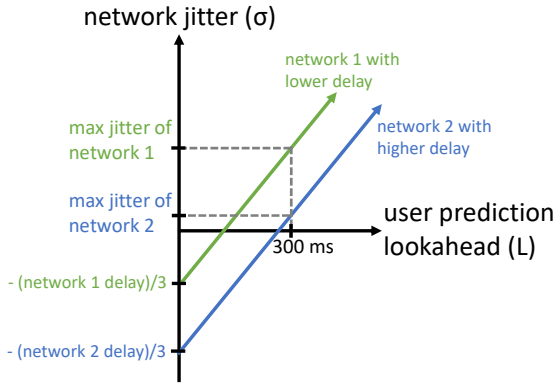


Fig. 15: Relationship between network delay, jitter, and prediction lookahead.

V. Relationship between lookahead, playout delay, and network delay

The low lookahead value of 300 ms discussed in Sec. III has implications for the frames' playout delays and the tolerable network delay and jitter for streaming 360°videos. Specifically, predicting the user's head orientation only 300 ms into the future in order to achieve low stalls implies that the total time from the tile request to rendering at the client, *i.e.,* the round-trip time plus the frame playout delay, must also be less than 300 ms. Otherwise, the prediction will be stale. This is in contrast to prior work that sets the prediction lookahead value to much higher values, *e.g.,* 2 seconds [11], [22], 3 seconds [2], 1-4 seconds [12], 2-6 seconds [8], *etc.*, which allows for much higher network delays.

We can derive the relationship between user prediction lookahead, playout delay, network delay, and network jitter as follows. The playout delay of a frame is commonly set as [23]:

$$T_{playout}[i] = 3\sigma \tag{1}$$

where $T_{playout}[i]$ is the time from when a frame i arrives at the client to when it is displayed to the user, and σ is the standard deviation of the packet inter-arrival time as measured by the client. The factor of 3 is due to approximating the inter-packet arrival time as a Gaussian distribution (which has been found to be a reasonable approximation in practice [24]), and allowing for 3 standard deviations of possible delay. Then the total round-trip delay from when a frame is requested to when it is received and displayed is:

$$T_{playout}[i] + T_{network}[i] + T_{render} \leq L \tag{2}$$

where $T_{network}[i]$ is the network delay RTT experienced by frame i, T_{render} is the frame rendering time, and L is how far ahead the user prediction method is must predict (*i.e.,* the lookahead value).

Eqn. 2, while simple, gives a rule of thumb for the tolerable network RTT and jitter based on the user head orientation prediction method's lookahead parameter. Fig. 15 illustrates Eqn. 2 and shows the maximum tolerable jitter values for different network RTTs and lookahead values. For example, network 1 (green) has a lower RTT than network 2 (blue), so it has a higher tolerable jitter. Network 2 (blue) with a higher RTT has a lower maximum tolerable jitter, or could even be infeasible for 360°video streaming if the network RTT or jitter is high enough. These tolerable network delay values are much more stringent than those typically allowed in regular non-360°video today. For example, the 5G QoS guidelines [25] for regular non-360°buffered video allow a packet delay budget of up to $T_{network} = 300$ ms, or approximately 600 ms round-trip, for which there is no feasible jitter value that satisfies the inequality in Eqn. 2, when the lookahead is 300 ms. However, the 5G guidelines for voice and live streaming guarantee a much stricter packet delay budget of $T_{network} = 100$ ms, implying that the jitter needs to be $\sigma \leq 33$ ms in order to have an acceptable low probability of stalls. (Note that here we assume rendering time T_{render} is negligible, although the formula allows for non-zero values). Overall, these numerical examples and equations demonstrate that in order for the user FoV prediction to be useful, 360°video streaming is only effective for very low network delay RTTs and jitter.

Another way of interpreting Eqn. 2 is based on real-world observed network performance. An RTT of 122 ms to request and retrieve a 360°video frame from a mobile edge cloud (as reported in [15]), an an approximate jitter of 51 ms (calculated by approximating the measured 90th and 10th percentile as 4σ), gives a total lookahead of 275 ms, which is within the recommended 300 ms to achive low prediction loss. However, the network delay values observed for a centralized cloud site [15] (212 ms RTT, 58 ms jitter) would result in a lookahead value of 386 ms, which is barely outside our recommended 300 ms. Thus, while some types of network deployments (*e.g.,* edge-based cloud architectures) can meet the stringent requirements of 360°video streaming, other deployments (*e.g.,* centralized cloud) may find it more difficult to meet the latency requirements, and the level of interactivity, in 360°video streaming. While Freedom [15] also relates network delay to 360° streaming performance on edge networks, our ability to use predictions and *user-customized rectangular padding* significantly improves the user QoE and network bandwidth requirements, even in a more challenging cloud deployment with higher latencies.

VI. Related Work

User head orientation and FoV prediction: Head orientation and FoV prediction approaches in the literature range from simple linear regression [2] and KNNs [8] to more complex models such as neural networks. Neural network variants employed for user head orientation prediction include LSTMs [7], LSTMs with content features [10], or attention-based neural networks [9]. However, such approaches are evaluated across disparate datasets and compared against subsets of prediction methods, whereas this work benchmarks such prediction methods on common datasets, focusing on methods that rely on user history only which are commonly used in 360°streaming systems [2], [11].

Tile encoding: The HEVC codec [6] allows developers to spatially partition the video into multiple tiles and encode each tile individually. Some works use homogeneous tiling schemas (same sized tiles) and some use heterogeneous tiling schemas [26], [27], [28], [29], [30], [31]. This work focuses on tiles produced by an equirectangular projection, which is one of the most widely used projection schemes, but its core results on user head orientation prediction can be applied to tiles of any shape.

360°tile selection: Flare [2], Rubiks [11], Pano [1], and Petrangeli *et al.* [12] are 360°streaming systems, incorporating FoV prediction modules and/or tile selection modules. Our insights into FoV prediction performance can lead to improvements in these systems' FoV prediction modules, and also has implications for their playout buffer setting (Sec. V).

Padding: Padding can be chosen as surrounding the FoV symmetrically [12], [32], [33], or irregularly based on complex algorithms such as model-predictive control or knapsack problems [8], [2], [11], [1]. In contrast, this work finds a middle ground by proposing an asymmetric padding that is wider than it is tall, providing a simple padding selection mechanism that outperforms existing symmetric padding and can avoid complex tile selection algorithms. Furthermore, this work also explores customized, per-video and per-user padding settings in order to further optimize streaming performance.

VII. Conclusions

In this paper, we took a detailed look at user FoV prediction for 360°video streaming. We found that there was little difference across popular FoV prediction algorithms, including linear regression and LSTM neural networks. However, head orientation prediction loss was acceptably low only when predicting in the near short-term (*e.g.,* 300 ms ahead). This has implications for how the video playout delay buffers are configured in 360°streaming systems, as well as tolerable network delay and jitter, as short-term predictions imply short playout delay buffers and stringent demands on network latency. We also explored how observations of user behavior can be used to optimize the shape and size of any additional "padding" tiles. A rectangular padding can substantially mitigate prediction errors, while only adding a relatively small amount of additional network bandwidth, and still maintaining a low video stall ratio. Overall, these observations have implications for how 360°streaming systems are configured in the future. Future work includes profiling users for personalized padding, as well as improving performance on 5G cellular networks.

Acknowledgements

This work has been supported in part by NSF grants CNS-1817216 and 1763929.

References

[1] Y. Guan, C. Zheng, X. Zhang, Z. Guo, and J. Jiang, "Pano: Optimizing 360 video streaming with a better understanding of quality perception," in *ACM SIGCOMM*, 2019.

[2] F. Qian, B. Han, Q. Xiao, and V. Gopalakrishnan, "Flare: Practical viewport-adaptive 360-degree video streaming for mobile devices," in *ACM MobiCom*, 2018.

[3] M. Xiao, C. Zhou, Y. Liu, and S. Chen, "Optile: Toward optimal tiling in 360-degree video streaming," in *ACM Multimedia*, 2017.

[4] M. Graf, C. Timmerer, and C. Mueller, "Towards bandwidth efficient adaptive streaming of omnidirectional video over http: Design, implementation, and evaluation," in *ACM MMSys*, 2017.

[5] S. Afzal, J. Chen, and K. K. Ramakrishnan, "Characterization of 360-degree videos," in *ACM SIGCOMM Workshop on Virtual Reality and Augmented Reality Network*, 2017.

[6] "Hevc," https://www.itu.int/rec/T-REC-H.265.

[7] X. Hou, S. Dey, J. Zhang, and M. Budagavi, "Predictive view generation to enable mobile 360-degree and vr experiences," in *ACM SIGCOMM Workshop on Virtual Reality and Augmented Reality Network*, 2018.

[8] Y. Ban, L. Xie, Z. Xu, X. Zhang, Z. Guo, and Y. Wang, "Cub360: Exploiting cross-users behaviors for viewport prediction in 360 video adaptive streaming," in *IEEE ICME*, 2018.

[9] J. Yu and Y. Liu, "Field-of-view prediction in 360-degree videos with attention-based neural encoder-decoder networks," in *ACM Workshop on Immersive Mixed and Virtual Environment Systems*, 2019.

[10] Y. Xu, Y. Dong, J. Wu, Z. Sun, Z. Shi, J. Yu, and S. Gao, "Gaze prediction in dynamic 360 immersive videos," in *IEEE CVPR*, 2018, pp. 5333–5342.

[11] J. He, M. A. Qureshi, L. Qiu, J. Li, F. Li, and L. Han, "Rubiks: Practical 360-degree streaming for smartphones," in *ACM MobiSys*, 2018.

[12] S. Petrangeli, V. Swaminathan, M. Hosseini, and F. De Turck, "An http/2-based adaptive streaming framework for 360 virtual reality videos," in *ACM Multimedia*, 2017.

[13] X. Corbillon, F. De Simone, and G. Simon, "360-degree video head movement dataset," in *ACM MMSys*, 2017.

[14] C. Wu, Z. Tan, Z. Wang, and S. Yang, "A dataset for exploring user behaviors in vr spherical video streaming," in *ACM MMSys*, 2017.

[15] S. Shi, V. Gupta, and R. Jana, "Freedom: Fast recovery enhanced vr delivery over mobile networks," in *ACM MobiSys*, 2019.

[16] S. LaValle, *Virtual Reality*. Cambridge University Press.

[17] K. Shoemake, "Animating rotation with quaternion curves," *SIGGRAPH Comput. Graph.*, vol. 19, no. 3, pp. 245–254, Jul. 1985.

[18] P. J. Brockwell and R. A. Davis, *Introduction to time series and forecasting*. springer, 2016.

[19] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *ACM KDD*, 2016.

[20] A. Balachandran, V. Sekar, A. Akella, S. Seshan, I. Stoica, and H. Zhang, "Developing a predictive model of quality of experience for internet video," in *ACM SIGCOMM*, 2013.

[21] H. Mao, R. Netravali, and M. Alizadeh, "Neural adaptive video streaming with pensieve," in *ACM SIGCOMM*, 2017.

[22] X. Corbillon, G. Simon, A. Devlic, and J. Chakareski, "Viewport-adaptive navigable 360-degree video delivery," in *IEEE ICC*, 2017.

[23] C. Perkins, *RTP: Audio and Video for the Internet*. Addison-Wesley Professional, 2003.

[24] S. B. Moon, J. Kurose, and D. Towsley, "Packet audio playout delay adjustment: performance bounds and algorithms," *Multimedia systems*, vol. 6, no. 1, pp. 17–28, 1998.

[25] 3GPP, "TS 38.300: NR; Overall description; Stage-2," June 2019.

[26] R. Skupin, Y. Sanchez, C. Hellge, and T. Schierl, "Tile based hevc video for head mounted displays," in *2016 IEEE International Symposium on Multimedia (ISM)*, Dec 2016, pp. 399–400.

[27] K. K. Sreedhar, A. Aminlou, M. M. Hannuksela, and M. Gabbouj, "Viewport-adaptive encoding and streaming of 360-degree video for virtual reality applications," in *IEEE ISM*, 2016.

[28] J. Le Feuvre and C. Concolato, "Tiled-based adaptive streaming using mpeg-dash," in *ACM MMSys*, 2016.

[29] K. Misra, A. Segall, M. Horowitz, S. Xu, A. Fuldseth, and M. Zhou, "An overview of tiles in hevc," *IEEE Journal of Selected Topics in Signal Processing*, vol. 7, no. 6, pp. 969–977, Dec 2013.

[30] Y. Snchez, R. Skupin, and T. Schierl, "Compressed domain video processing for tile based panoramic streaming using hevc," in *IEEE ICIP*, 2015.

[31] C. Zhou, M. Xiao, and Y. Liu, "Clustile: Toward minimizing bandwidth in 360-degree video streaming," in *IEEE INFOCOM*, 2018.

[32] S. Shi, V. Gupta, M. Hwang, and R. Jana, "Mobile vr on edge cloud: a latency-driven design," in *ACM MMSys*, 2019.

[33] Y. Bao, H. Wu, T. Zhang, A. A. Ramli, and X. Liu, "Shooting a moving target: Motion-prediction-based transmission for 360-degree videos," in *IEEE Big Data*, 2016.