Flexitile: A Flexible 360 Video Tiling and Streaming Method

Amir Ahmad Mohammadi[1*] and Mohammad Sharifkhani[1]

[1*]Department of Electrical Engineering, Sharif University of Technology, Azadi Ave., Tehran, 1458889694, Iran.

*Corresponding author(s). E-mail(s): amirahmad.mohamadi@ee.sharif.edu;
Contributing authors: msharifk@sharif.edu;

Abstract

Tiled-based streaming is a promising method for 360-degree video streaming that reduces transmission bandwidth while enhancing the quality of experience (QoE). However, the tiling layout and the bitrate of the tiles significantly affect the QoE. This paper proposes FlexiTile, a novel and flexible method that maximizes QoE in tiled-based 360-degree video streaming. FlexiTile achieves this through joint optimization of the tiling layout and bandwidth allocation for each tile. This optimization considers the rate-distortion characteristics of the video content and user viewing direction preferences. FlexiTile leverages a formulation based on the Set Cover problem and an efficient and accurate piece-wise linear estimation of the rate-distortion model for candidate tiles.

FlexiTile is evaluated under various transmission constraints and a subjective quality test. Compared to the baseline Notile tiling scheme, FlexiTile achieves a 1.9 dB average improvement in viewport PSNR using the BD metric. Additionally, it outperforms other state-ofthe-art methods in terms of viewing quality. Furthermore, FlexiTile demonstrates improvement over existing tiled-based streaming methods by optimizing tile bandwidth allocation. Experiments show a 1.3 dB average improvement in Uniform tiling (4×2) when the bandwidth allocation is based on the FlexiTile optimization method.

FlexiTile surpasses existing methodologies by providing adaptable tiling layout and bitrate allocation strategies, resulting in a notable enhancement of QoE.

Keywords: virtual reality, 360 video, video streaming, tile-based streaming, integer linear programming

1 Introduction

Over recent years, 360-degree video content (immersive videos) has garnered significant user interest. In this type of video content, by using a head-mounted display (HMD) that tracks the user's head movement, the virtual viewport within the video synchronizes with that of the user to induce an immersive experience.

However, achieving a high-quality immersive experience presents technical challenges. It necessitates high-resolution video at the user's viewport, coupled with fast tracking of head movements. This, in turn, demands a high-speed network connection and minimal latency in synchronizing the virtual viewport with the user's viewing

direction. Furthermore, high-resolution video streaming and decoding are power-intensive tasks, impacting the battery life of HMDs like smartphones. These combined technical hurdles complicate 360-degree video streaming. An interesting aspect to consider is that users typically only view a small portion of the surrounding environment at any given time. Studies have shown that for most of the playback duration, users only focus on roughly 10% of the 360-degree video content [1]. This observation suggests that compression (encoding) and bandwidth resources can be strategically allocated towards the user's viewport rather than transmitting the entire 360-degree sphere. This principle underpins various approaches designed to address the challenges associated with 360-degree video streaming.

Tiled-based streaming emerges as a promising solution to the aforementioned challenges in 360-degree video streaming [2], [3]. This approach partitions the video content spatially into smaller tiles, each encoded in various quality levels. Unlike requesting a large encompassing tile for the entire 360-degree view, the client device retrieves high-quality versions of tiles within the user's viewport and lower-quality versions for peripheral regions.

The tiling layout and quality levels assigned to each tile significantly influence the user's Quality of Experience (QoE) [4], [5]. To fully visualize a viewport, users must download all tiles overlapping their viewing area, even if the overlap is partial. Downloading these partially overlapping tiles can lead to inefficient bandwidth utilization as only a portion of the content is actually viewed. This can exacerbate buffering delays, especially when users seek high-quality playback. Given limited network bandwidth, minimizing the number and size of partially overlapping tiles becomes crucial. This naturally leads to using smaller tiles which consume less bandwidth, but this comes at the cost of reduced compression efficiency, ultimately impacting streaming bandwidth requirements.

This inherent trade-off highlights the need for optimization strategies to determine optimal tile sizes and corresponding bitrates for maximizing QoE. Beyond tile size and bandwidth, optimization should encompass additional factors such as storage capacity and computational power of the streaming server and processing capabilities of the user device for decoding the video stream. Formulating this optimization problem with all these resource constraints in a practical context is essential for comprehensively evaluating tiled-based streaming and establishing best practices to enhance the user's overall experience.

Research has addressed optimal tile and bitrate selection for 360-degree video streaming from two primary perspectives: client-side and server-side optimization.
Client-side approaches assume pre-defined tiling layouts and quality levels prepared on the server. Here, the focus lies on client-side tile selection to maximize user QoE. Key contributions from these methods include novel quality metrics and viewport prediction techniques to accurately predict the user's QoE in order to optimize his/her tile selection. Quality Metrics consider individual user experience, incorporating factors like buffer capacity and bitrate variation [6], [7]. While powerful for individual users, these methods are constrained by server-provided tiling layouts and quality levels. On the other hand, server-side approaches consider anticipated users' behavior to pregenerate

tiles at the server, aiming to deliver optimal QoE for average users. In fact, this provides the input to the former methods. This is the condition in video broadcasting where the goal is to optimize the broadcasting for all viewers instead of individual users. Server-side studies pursue various goals, with reducing streaming bandwidth being a primary focus [4], [8], [9], [10], [11], [12]. However, bandwidth-centric approaches can compromise QoE even when bandwidth is ample. Therefore, more recent server-side research prioritizes QoE. In this group, some methods focus on fixed tiling layouts, optimizing tile encoding bitrates based on network conditions to deliver the highest possible video quality (e.g., [1], [13], [14], [15]). However, joint optimization of tiling layout and bitrate could enhance QoE. User viewing preferences and tile rate-distortion characteristics could be leveraged to improve compression efficiency and reduce bandwidth without sacrificing video quality.

Achieving the highest possible QoE necessitates evaluating numerous potential tiling layouts and bandwidth distributions. Traditional approaches employ brute-force encoding of every possible tile at predefined quality levels [5], which is computationally infeasible for practical applications. Recent studies use heuristics to select a subset of near optimal tile layouts [6]. However, a more practical solution lies in the development of an analytical rate-distortion model encompassing all possible tiles, facilitating the formulation of tiled-based streaming as an optimization problem.

This paper proposes FlexiTile, a server-side optimization framework for maximizing user video quality in tiled-based streaming of 360-degree video content under constrained network bandwidth. FlexiTile leverages an accurate, piece-wise linear estimation of tile rate-distortion models and user viewport video quality. This linear model facilitates the joint optimization of tile layout and bandwidth allocation through a Mixed-Integer Linear Programming (MILP) formulation. In summary, this method has the following features:

1) The objective is to maximize the average users' viewports PSNR (VPSNR) by optimally utilizing available network bandwidth.

2) A crucial element of the optimization is the rate-distortion model for each potential tile. FlexiTile addresses the challenge of efficiently obtaining these models for all tiles without sacrificing quality through the use of a piece-wise linear estimation approach.

3) The MILP formulation allows for the incorporation of additional practical constraints into the optimization process. Examples include storage capacity limitations and decoder computational capabilities on the client-side. This flexibility enables FlexiTile to adapt to diverse streaming scenarios.

In this paper, the related works and background information are reviewed in Section 2 and the proposed method is explained in Section 3. Section 4 evaluates our method in different use-case scenarios and explains our findings. Finally, the paper is concluded in Section 5.

2 Related work

There two main areas that are addressed by this work are tiled-based streaming and video quality assessment methods. Thus, the related work to these two areas are explored.

2.1 Tile-based streaming

A significant number of tile-based streaming methods for 360-degree video content, historically employed uniform ($m \times n$) tiling [10], [11], [16], [17], [18], [19], [20]. This approach has been prevalent due to its native support in video coding standards since High-Efficiency Video Coding (HEVC) [16]. Tiling, as a video encoding technique, involves partitioning the video frame into a grid of independently decodable rectangular regions (tiles), which can be multiplexed into a single bitstream. This architectural feature facilitates a differentiated quality of service, enabling the tiles within the user's viewport to be encoded at a high fidelity, while peripheral tiles are compressed more aggressively at a lower quality.

The most recent video coding standard, H.266/Versatile Video Coding (VVC), was developed with intrinsic support for 360-degree video applications. Consequently, it incorporates enhanced coding tools and a specialized streaming format specifically designed to improve the efficiency and facilitate the streaming of tile-based video [21]. Despite these advancements, a primary challenge for such methods remains their dependency on specialized streaming clients and network infrastructure capable of managing tile selection in response to dynamic network bandwidth variations.

To enhance compression efficiency by accounting for the geometric distortions introduced by projecting spherical 360-degree video content onto a two-dimensional rectangular frame, alternative approaches have proposed heuristic tiling layouts tailored to specific projection formats. [22], [23], suggested that 360 videos with ERP projection can be tiled with two horizontal tiles at poles and four vertical tiles at the center part of the video. This heuristic comes from the fact that in ERP-projected videos the top and bottom parts are mapped from small areas in the 360-degree sphere. The purpose of these methods is to decrease the large bandwidth requirement of traditional streaming of the whole 360-degree video content and they do not explore further to enhance perceptual video quality.

The AdaptiveTiling [24] utilizes a greedy algorithm that commences by partitioning the entire frame into a multitude of small basic tiles. Subsequently, it attempts to merge these basic tiles to form larger tiles iteratively. At each iteration, guided by the viewers' preference map, the algorithm identifies the merge that minimizes the average streaming bandwidth by encoding the candidate merged tiles. However, due to the vast number of tiling possibilities that require encoding and the inherent suboptimality of the greedy algorithm, this approach becomes impractical, limiting the solution's performance.

OpTile [4] formulated the tiled-based streaming problem into a Mixed Integer Linear Programming (MILP) optimization. The MILP is based on *set cover* problem in which the

frame is split into small basic tiles. Possible merges of the basic tiles are modeled in the MILP as linear constraints. OpTile also adds storage cost to the *objective function* in order to minimize the storage needed for storing all resulting tiles at the streaming server as well as the streaming bandwidth. ClusTile [9] improves the bandwidth saving of OpTile by allowing the final tiling layout to have overlapping tiles which increases the storage cost. Moreover, it mitigates the complexity of OpTile's MILP in two ways; First, it clusters the viewers' preference map into a few representative viewports, and instead of solving its optimization for the entire set of viewports, it only considers these representative viewports. Second, it solves the problem for each cluster independently. This decreases the number of constraints in the MILP. Hence, the optimization time decreases as well. VASTILE [8] proposed a tiling method that reduces the processing time compared to the previous method. However, it lacks a proper bandwidth allocation algorithm which is the main reason that other methods have higher computational complexity.

The aforementioned methods prioritize bandwidth savings over improving the Quality of Experience (QoE), implicitly assuming that if encoding parameters remain constant, the tiling has no impact on QoE. However, this assumption is not accurate as maintaining consistent quality across all tiles necessitates meticulous adjustment of encoder parameters for each tile (because of different rate-distortion relationship for each tile). Furthermore, neglecting user-available bandwidth leads to lengthy buffering delays, significantly degrading QoE. These limitations have prompted the development of advanced tiling methods that optimize video quality while considering rate-distortion characteristics to effectively utilize available bandwidth [5], [6], [25].

For example, Chen et al. [6] proposed a Popularity-aware 360-degree video streaming method (Macrotile). This method utilizes macrotiles, large tiles that encompass the user's viewport, offering enhanced compression efficiency compared to the conventional tiles, thereby increasing QoE during video streaming. Macrotile construction is guided by the viewers' preference map. During streaming, the algorithm selects between macrotiles and conventional tiles to maximize QoE. However, due to the NP-hardness of the optimization problem, a greedy algorithm that prioritizes tiles with higher bitrates.

2.2 Quality assessment in 360-degree video streaming

Assessing the efficacy of tiled-based streaming methods in enhancing 360-degree video quality necessitates the employment of appropriate quality metrics. Traditionally, metrics like PSNR and SSIM have been applied to the projected 360-degree video. However, to account for the impact of projection, modified versions of these metrics have emerged, such as SPSNR and WSPSNR [7], [26].

Ozcinar et al. [5] proposed Visual Attention Spherical Weighted PSNR of the video (VASW-PSNR). The VASW-PSNR of the video is the combination of WSPSNR [27] of the

video and the viewers' preference map of the video. In VASW-PSNR, the error of each point in WSPSNR is multiplied by its viewing probability.

Pano [25] states that traditional quality metrics like PSNR [28] or SPSNR [27] are inadequate for 360-degree videos. To address this, Pano introduced three new quality factors: 1) relative viewing speed, 2) relative luminance difference, and 3) relative focus difference. Subjective tests demonstrated that these factors independently influence perceived quality. Pano then proposed a new quality metric, Peak Signal to Perceivable Noise Ratio (PSPNR), which incorporates the concept of Just Noticeable Difference (JND). JND represents the minimum discernible distortion level. PSPNR resembles PSNR but excludes pixels with noise levels below their respective JNDs. While PSPNR is a novel metric, extracting JNDs via subjective tests is impractical.

Recent studies have demonstrated that viewport-centric metrics, such as viewport PSNR, exhibit a stronger correlation with subjective quality perception compared to projection-based metrics [29], [30], [31]. Additionally, these viewport-centric approaches offer lower computational complexity and wider acceptance. Consequently, this study adopts average viewport PSNR (VPSNR) as the quality metric for tiling optimization. This choice simplifies the modeling of rate-distortion characteristics, leading to more efficient utilization of available bandwidth, ultimately enhancing QoE and minimizing buffering delays.

3 Proposed Method

The objective of FlexiTile is to maximize the average viewport video quality of the users watching a tiled-based 360-degree video under limited available network bandwidth. The video content model, the video quality model and the formulation are presented in this section.

3.1 Video Content Model

The video content is temporally split into segments of fixed duration (e.g., 2 sec.) for tiling optimization. Each segment is spatially split into $m \times n$ basic tiles. All possible merging combinations of basic tiles are named candidate tiles ($\{T_1,...,T_k\}$). For example, for a 20×10 basic tile layout, there are 200 basic tiles which make 11550 candidate tiles (This number is measured by counting all possible tiles constructed by merging a subset of these 200 basic tiles). A combination of candidate tiles that cover the entire sphere is named a tiling layout.

3.2 Video Quality Model

The expected PSNR of the video at the individual users' viewport (v) is chosen as the quality metric of the 360-degree video content [32]. The expectation function helps to catch an all-inclusive measurement of the perceived quality. It is named VPSNR to indicate that the PSNR is calculated at the viewport domain instead of the projection domain (Eq. (1)).

$$VPSNR = E_v[PSNR_v] \quad (1)$$

Where E_v is the expectation function over all possible viewports. In other words, FlexiTile maximizes VPSNR by minimizing the *average* distortion of the viewports of individual users, $d(v)$.

$$D = E_v[d(v)] \quad (2)$$

The optimization will be detailed in sub-section 3.3. The mathematical model for the distortion $d(v)$ for a given set of candidate tiles that cover viewport v is:

$$d(v) = \sum_i d_i do(i,v) \quad (3)$$

where d_i is the mean-square error of candidate tile T_i and $do(i,v)$ is the fraction of the distortion of candidate tile T_i that contributes to the distortion of the v. Clearly, if v completely covers candidate tile T_i, $do(i,v)$ is equal to 1. Otherwise, since distortion is not uniformly distributed over the pixels of tile T_i, $do(i,v)$ is not necessarily proportional to the overlap area of tile T_i with v.

Parameter d_i follows the rate-distortion curve of the candidate tile T_i. Thus, the rate-distortion model of all candidate tiles covering v is needed. Modeling the rate-distortion behavior of the encoders is studied in many papers for other video applications (e.g. [33], [34], [35]). It is shown that PSNR can be estimated using an exponential function that relates bit-rate r to PSNR.

$$PSNR \propto \alpha_i \log(r) + \beta_i \quad (4)$$

According to PSNR definition, d_i is logarithmically related to PSNR:

$$\log(d_i) = \alpha_i \log(r_i) + \beta_i \quad (5)$$

where α_i and β_i are parameters dependent on the content of tile T_i. Using Eq. (5), $d(v)$ is written as:

$$d(v) = \sum_i do(i,v) 10^{\alpha_i \log(r_i) + \beta_i} \quad (6)$$

To determine $do(i,v)$ an estimation of the spatial distribution of distortion in the video is needed. This estimation is facilitated through the utilization of a Distortion Distribution Map (DDM). DDM is a map showing the amount of distortion introduced to each pixel of a video after encoding the video to a different bitrate. An example of a distortion distribution map is shown in Fig. 1 which shows the distortion distribution when transcoding a video from 17.9 Mbps to 2 Mbps. Besides the encoding, projection

affects the distortion distribution. For example, ERP causes polar areas of the video to expand. Therefore, a large area in the upper part of the projected video is a small area in the user's viewport. Hence, its encoding distortion is not as significant as the distortion of the areas in the equator. To account for this effect, DDM is scaled by the effect of the projection. For an $m \times n$ ERP video the scaling factor for pixel (x,y) is defined as follows:

$$w(x,y) = \cos\left(\frac{\pi}{n}\left(y + \frac{1}{2} - \frac{n}{2}\right)\right) \tag{7}$$

Using the DDM, the $do(i,v)$ is the sum of distortion introduced to pixels in the overlap area of Tile T_i with viewport v.

To determine $d(v)$, it is also needed to extract α_i and β_i parameters for all overlapping candidate tiles. This necessitates the rate-distortion model for all candidate tiles. For this purpose, a rate-distortion model estimator is introduced in Sub-section 3.5 that accurately estimates the rate-distortion model of a given tile in a video without extracting and actually encoding the tile.

3.3 Optimization Problem Formulation

The partitioning or tiling problem is a *set cover* problem which is a well-studied problem in optimization theory with known solutions. Consider a set of $m \times n$ basic tiles and a collection of candidate tiles ($T\{T_1,\ldots,T_k\}$), each composed of a subset of these basic tiles. The objective is to find a tiling layout and bandwidth of the tiles that minimize the average distortion D of all viewports, thereby maximizing the VPSNR, while ensuring that all basic tiles are covered and the available bandwidth for each viewport is utilized to the fullest extent.

Eq. (8) shows the basic formulation of FlexiTile where T_i is a binary variable that represents the $i - th$ candidate tile. If $T_i = 1$ then the corresponding candidate tile is present in the final tiling layout. M is a matrix that identifies the constituent basic tiles of each candidate tile. If $M_{ji} = 1$ it means T_i contains basic tile j. The first constraint is the covering constraint that ensures that all basic tiles are covered by the final tiling layout. The final constraint mandates that the bandwidth allocation for tiles be optimized in a manner that guarantees that the required streaming bandwidth of the tiles covering any viewport does not exceed the network bandwidth limitation. The parameter $o(i,v)$ is the proportion of the area of tile T_i that overlaps with viewport v and the parameter $\lceil o(i,v) \rceil$ indicates whether tile T_i is a covering tile for viewport v or not. Finally, the objective is to minimize the average distortion of viewports which is defined in the previous sub-section.

$$\begin{aligned} &\text{minimize}: && D \\ &\text{s.t.} && \sum_i T_i M_{ji} \geq 1 \forall j \\ & && \sum_i \lceil o(i,v) \rceil r_i \leq R_{\text{limit}} \ \forall v \end{aligned} \tag{8}$$

Eq. (8) is a non-convex Mixed Integer Non-Linear Problem (MINLP) and it is not possible to find the optimal solution for this problem in a reasonable time even with a small subset of tiles. Therefore, to find a near-optimal solution, in a reasonable time, a linear estimation of the problem is proposed in the next sub-section.

3.4 Linear Estimation of the Formulation

A linear approximation of the tiling problem formulation (Eq. (8)) is presented as a Mixed Integer Linear Programming (MILP) that can be solved within a reasonable timeframe. To transform the MINLP into MILP, both the objective function and the constraints are reformulated as linear functions of the decision variables.

The nonlinearity in the objective function arises from the inherent nonlinearity of the rate-distortion model represented by Eq. (5). A piece-wise linear approximation is employed to replace this nonlinear function. Consequently, for each candidate tile T_i, the distortion d_i is modeled based on a set of l linear inequality constraints that are introduced to model the piece-wise linear approximation of Eq. (5). Fig. 2 demonstrates the method. In essence, instead of compelling the optimization to adhere to the nonlinear rate-distortion curves of the candidate tiles, the problem is reformulated to enforce the optimization to utilize the rate-distortion curve as the lower bound for the distortion of candidate tile T_i (i.e. d_i) at any rate (r_i). Since the rate-distortion curves exhibit convexity, minimizing the cost function minimizes the distortions to their lower bounds, which coincide with the rate-distortion curves.

$$\begin{aligned} &\text{minimize}: && \sum_v \sum_i T_i d_i do(i,v) \\ &\text{s.t.} && \sum_i T_i M_{ji} \geq 1 \forall j \\ & && \sum_i \lceil o(i,v) \rceil r_i \leq R_{\text{limit}} \ \forall v \\ & && d_i \geq a_{ij} r_i + b_{ij} \, 1 \leq j \leq l \end{aligned} \tag{9}$$

where

$$a_{ij} = \frac{d_{ij} - d_{i(j+1)}}{r_{ij} - r_{i(j+1)}}$$
$$b_{ij} = d_{i(j+1)} - r_{i(j+1)} a_{ij} \tag{10}$$

For each candidate tile T_i and linear piece j, a_{ij} and b_{ij} is obtained based on parameters α_i and β_i (Eq. (5)). The latter two parameters are estimated using a proposed method explained in the next sub-section.

Eq. (9) remains nonlinear due to the presence of the $T_i d_i$ quadratic term in the objective function and the $d_{ij} r_i$ quadratic term in constraints. In both quadratic terms, T_i is a binary variable indicating the inclusion of tile i in the final tiling layout. Given the binary nature of T_i, its impact is to eliminate r_i and d_i when candidate tile i is not included in the final tiling layout. To enforce this effect without the T_i coefficient, it is essential to set $r_i, d_i = 0$ when $T_i = 0$. To achieve this for r_i, an additional constraint is introduced to control the upper bound of r_i using T_i. When $T_i = 0$ the upper bound of r_i is effectively set to 0. For d_i the approach is the opposite since d_i is part of the objective function and its value is lower-bounded by the rate-distortion curve. Thus, to make $d_i = 0$ when $T_i = 0$, the lower bound must be set to zero. To accomplish this, the rate-distortion inequality constraints in Eq. (10) are modified, and T_i is multiplied to b_{ij}. Consequently, since r_i is equal to 0 when $T_i = 0$, the inequality constraints become $d_i > 0$. These modifications are reflected in Eq. (11):

$$\begin{aligned} \text{minimize}: \quad & \sum_{v} \sum_{i} do(i,v) d_i \\ s.t. \quad & \sum_{i} T_i M_{ji} \geq 1 \forall j \\ & \sum_{i} \lceil o(i,v) \rceil r_i \leq R_{\text{limit}} \ \forall v \\ & d_i \geq a_{ij} r_i + T_i b_{ij} \, 1 \leq j \leq l \\ & T_i \leq r_i \leq T_i R_{\text{limit}} \end{aligned} \tag{11}$$

Leveraging a linear approximation facilitates the optimization problem's solution within a computationally tractable timeframe. However, constructing Eq. (11) necessitates the rate-distortion model of all candidate tiles (for parameters a_{ij} and b_{ij}). Due to the substantial number of candidate tiles, computational efficiency is paramount. Therefore, the subsequent sub-section introduces a rate-distortion model estimator.

3.5 Rate-Distortion Model Estimation

Eq. (5) relates the bitrate of a video to the distortion (for all candidate tiles). This equation is needed to predict the distortion of a candidate tile at an arbitrary bitrate. Since the rate-distortion model of a candidate tile depends on its content and the

encoding parameters, to find parameters α_i and β_i for candidate tile T_i, first, the tile must be encoded at multiple bitrates to measure the induced distortion. Then, the bitrate-distortion *pairs* are fitted to Eq. (5) to estimate the model and the piecewise linear estimation. This process needs to be repeated for every candidate tile. Therefore, it takes an impractical time to find the rate-distortion model of all candidate tiles through encoding (For example, there are 11550 candidate tiles for 20×10 basic tiles). To speed up the process of rate-distortion estimation for all candidate tiles, a Multi-Layer Perceptron (MLP) regressor is used to estimate the PSNR of a candidate tile at a given bitrate based on the features of the comprising basic tiles. Therefore, instead of encoding a candidate tile at multiple bitrates, the bitrate-distortion *pairs* are obtained from the MLP regressor in the optimization process of FlexiTile. This significantly reduces the pre-processing time for the video tiling optimization of a video segment.

Assuming adaptive steaming video segments are 2 second long (a typical value in HTTP adaptive streaming), the MLP is trained with the features obtained from several 2-second video segments of different tile sizes at various bitrates and distortion as training samples. Clearly, the training phase of the MLP regressor is done only once using sufficient data-set video tile samples. During the tiling optimization phase of a video segment, the same features are extracted for all candidate tiles T_i to obtain parameters α_i and β_i that ultimately provides the rate-distortion curve of the tile for optimization.

Specific features in a video segment are selected to train and use the MLP regressor to estimate the distortion at a given bitrate. Assuming only one I-frame at the beginning of the video segment, the average gradient of the first frame of the tile, g, is used as a feature to represent the I-frame information. For the intra-frames information, the frame is split into 16×16 pixel blocks. The motion vectors are calculated based on the comparison between the first and the last frame of the segment for these blocks. Depending on how different the first and last frame of the segment is, the motion vectors of the blocks have three different patterns. Fig. 3 shows these 3 types of motion vectors. The first pattern (case a) is where the motion vectors remain within the same basic tile in a candidate tile. The second pattern (case b) is the motion vectors point to a place outside of the initial basic tile boundary yet within the original candidate tile (mv_b). These motion vectors change if the basic tile is cropped from the candidate tile and encoded separately. The third pattern (case c) is where the motion vectors not only point outside of the initial basic tile but also to a place outside of the boundary of the candidate tile (mv_b). If the tile is encoded independently, these motion vectors also change. The last two patterns of the motion vectors are important because they show the bitrate overhead of decoupling a basic tile from a candidate tile and the overhead of decoupling a candidate tile from the original video. Thus, the average distortion each motion vector in these two categories introduce is used to represent the sensitivity of the distortion of the candidate tile to the distortion of the comprising basic tiles ($d_{average}$).

Finally, the bitrate at which the tile is encoded is also used as a feature in the training phase. All the features are summarized in Table 1.

3.6 Implementation

The implementation of the proposed method's formulation is illustrated in Fig. 4. The figure provides an overview of the practical implementation of the FlexiTile method. The inputs to FlexiTile include 2-second streaming video segments, user viewports (or the access pattern map of the video), and the optimization constraints such as total network bandwidth limit and server storage limit. Prior to the MILP optimization, a pre-processing step provides the piece-wise rate-distortion models for all candidate tiles, objective function and tile overlapping coefficients needed for the optimization.

During the pre-processing stage, each 2-second video segment and the corresponding users' viewports are utilized to extract the basic tiles and the *Distortion Distribution Map* (DDM). The DDM and the viewports are employed to compute $do(i,v)$ as detailed in Eq. (6). The basic tiles are used to construct the candidate tiles. For a candidate tile T_i, the coefficients α_i and β_i are estimated using the MLP based rate-distortion estimator, which has already been trained using the features and video samples described in sub-section 3.5. With α_i and β_i available, the piece-wise linear parameters a_{ij} and b_{ij} is calculated to provide the constraints for MILP optimizer. With $do(i,v)$ and the other required video models available for all candidate tiles, *set cover* optimization is employed to solve Eq. (11) which is a MILP problem. This optimization problem can be efficiently solved using readily available *set cover* solver tools such as CPLEX.

In addition to the constraint on bandwidth, depending on the application, FlexiTile offers the flexibility of including other constraints such as server storage or decoder computational capability to the MILP solution. Modeling these constraints are explained in Appendix A.

Real-world network conditions often exhibit dynamic bandwidth fluctuations. Traditionally, tiled-based streaming approaches pre-encode video content into multiple quality levels for each tile. During playback, the client selects the desired quality level for each tile when requesting data from the server so that to maximize the QoE of the user. FlexiTile departs from this approach by generating distinct tiling layouts and quality levels tailored to specific network bandwidth conditions. This pre-generation considers user viewing preferences, allowing the server to anticipate client needs. Consequently, the client no longer needs to perform real-time optimization of tile quality levels. Server-side pre-optimization based on estimated bandwidth and viewport characteristics leads to a pre-selected, optimized result delivered to the client. This significantly reduces the computational complexity involved in client-side tile selection. While FlexiTile offers a pre-optimized solution, it also provides flexibility for further client-side refinement. The client can optionally select tiles from the broader set stored on the server, aiming to further maximize its individual QoE. This additional optimization layer caters to diverse user preferences or unexpected network variations, enhancing the adaptability of the streaming experience.

4 Experiments

A unified framework is constructed to assess and compare various tiling methods. The framework employed 4k 360-degree videos with ERP projection from the 360-video viewing dataset [36] as input videos. The dataset comprises the trajectory of 50 individual users viewing 10 distinct 4k 360-degree videos. From each video a 10-second clip was extracted and segmented into 2-second intervals. It has been demonstrated that the viewports of the 20 users are an accurate approximation of an all-users access map [5]. Therefore, to evaluate each tiling method the tiling layout and bandwidth allocation of the tiles are optimized using the trajectories of 20 users. Subsequently, the actual video tiles are extracted by cropping the original 360-degree video in accordance with the optimized tiling layout. In the next step, each tile is encoded at the respective optimal bitrate specified by the tiling method. For evaluation, the viewport videos of the other 30 test users are reconstructed based on the recorded head movement trajectory of each user to measure the PSNR of the video. Thereafter, the VPSNR of the video is calculated using Eq. (1) based on the PSNR of the 30 test users.

To generate the actual videos of candidate tiles, the platform employs FFmpeg [37] for tile encoding and measurement of both the actual PSNR and bitrate of the tiles. Considering that FlexiTile is encoder-agnostic and the core algorithm does not rely on a specific encoder, for the purpose of evaluation, all video files are encoded in H.265 format (X.265 Encoder [38]), because X.265 is an optimized, industrial-level implementation and more recent encoders are not widely adopted. The encoding parameters are selected in accordance with Apple's recommendations for HLS authoring [39]. The client's video decoder level is 5, enabling it to decode 4K videos at 30 frames per second. To construct the rate-distortion model estimator a Multilayer Perceptron (MLP) with three hidden layers of 40 nodes and a *tanh* activation function is utilized. The MLP is trained once using 6000 randomly selected 2-second tiles of varying sizes from different videos. The trained regressor achieves an accuracy of 94% for predicting the distortion at a given bitrate. The CPLEX optimizer [40] is employed to solve the MILP problems arising from various FlexiTile configurations, which will be discussed in detail later.

FlexiTile is a pre-streaming, server-side algorithm designed to optimize video tiling scheme. Utilizing a priori knowledge of viewer preferences, FlexiTile pre-processes video content to generate tiling scheme tailored for a defined set of available bandwidths, thereby enhancing streaming efficiency. Therefore, the client-side optimizations are out of scope of the algorithm. Considering this fact, in all simulations, perfect viewport prediction at the client device is assumed implying that the client device is sufficiently intelligent to request only the necessary tiles for viewing. The sole constraint is the client device's available bandwidth (that maps to one of the predefined server-side tiling schemes), which may result in buffering delays if the available bandwidth is insufficient to download the tiles of the segment within the required timeframe.

4.1 Configuration of Tiling Methods

The proposed method is evaluated against three reference methods: Notile (conventional streaming without tiling), Uniform 4×2 tiling (video is split into 4×2 equal tiles), and Uniform 10×5 tiling. Additionally, three state-of-the-art methods are considered for comparison: Ozcinar [5], VASTILE [8], and Macrotile [6]. For each method, the average PSNR and required downloading bandwidth across all users are calculated to determine the VPSNR and buffering delay of the evaluated method.

To evaluate the effectiveness of FlexiTile, three distinct configurations were employed: **FlexiTile-U4x2 and FlexiTile-U10x5**: These configurations utilize FlexiTile on uniform 4×2 and 10×5 tiling grids, respectively, aiming to optimize bandwidth allocation. **FlexiTile-General**: This configuration employs FlexiTile in its most general form, enabling simultaneous optimization of the number, geometry, and bandwidth of the tiles. The video is segmented into a 20×10 grid, resulting in 200 basic tiles. This translates to a field of view of approximately 18 degrees for each basic tile. While employing a large number of basic tiles enhances FlexiTile's adaptability, it comes at the expense of optimization speed. In contrast to conventional tiling approaches that apply bandwidth limitations to the entire 360-degree sphere, FlexiTile applies these limitations to individual input viewports. This strategy enables FlexiTile to exceed the bandwidth limit for the entire sphere without incurring buffering delays for individual test input viewports. As a result, other users experience minimal or no buffering delays. All FlexiTile configurations are implemented with a limited storage policy, ensuring that each basic tile is covered by only one tile in the resulting tiling layout. This constraint promotes efficient resource utilization and reduces storage requirements.

4.2 Comparison in Fixed Tiling Layouts

To assess the effectiveness of FlexiTile in optimizing bandwidth allocation, two fixed tiling configurations, Uniform 4×2 and Uniform 10×5, are compared with their respective FlexiTile-U counterparts. FlexiTile-U4x2 and FlexiTile-U10x5 are constrained to tiling layouts of 4×2 and 10×5, respectively, and focus on optimizing bandwidth distribution within these predefined tiling patterns.

Fig. 5 presents rate-distortion curves for Notile, Uniform 4×2 and Uniform 10x5 tiling alongside FlexiTile-U4x2 and FlexiTile-U10x5. The results demonstrate that Notile achieves superior video quality (VPSNR) compared to Uniform 4×2 and Uniform 10×5 tiling. This is attributed to the fact that while all these methods utilize the entire available bandwidth to cover the entire 360-degree sphere, in uniform tiling, compression efficiency is compromised due to the tiling process. FlexiTile-U4x2 exhibits an average VPSNR improvement of 1.3 dB over traditional 4×2 Uniform tiling, while FlexiTile-U10x5 demonstrates an even more significant enhancement of 1.73 dB. This superior performance of FlexiTile-U methods stems from two key factors: 1) **Viewport-Specific Bandwidth Allocation**: FlexiTile-U allocates the total available bandwidth to individual viewports rather than the entire 360-degree sphere. 2) **Rate-Distortion-Driven Bitrate**

Allocation: FlexiTile-U determines bitrate allocation based on the rate-distortion model of each tile, ensuring optimized bitrate distribution.

The comparative analysis demonstrates the superior performance of FlexiTileU4x2 and FlexiTile-U10x5 in achieving higher VPSNR compared to traditional tiling approaches. FlexiTile-U's viewport-specific bandwidth allocation and rate-distortion-driven bitrate allocation strategies contribute significantly to its enhanced performance.

4.3 Comparison in Joint Tiling Layout And Bitrate Allocation Optimization

The performance of FlexiTile in its generalized form (FlexiTile-G) which optimizes tile count, geometry, and bitrate, is assessed against Notile, VASTILE [8], Macrotile [6] and Ozcinar [5]. Since VASTILE and Macrotile do not have bitrate allocation algorithm, to have a fair comparison, FlexiTile is used to allocate the bitrate of their chosen tiles. These FlexiTile enhanced version of these methods are denoted as VASTILE-FE and Macrotile-FE. Fig. 6 illustrates the average VPSNR across users. FlexiTile-G demonstrates a VPSNR improvement of 1.92 dB compared to Notile, 1.25 dB compared to VASTILE, 0.19 dB compared to Macrotile, and 2.1 dB compared to Ozcinar, as measured using the Bjøtegaard delta [41]. Besides VPSNR, VMAF and SSIM were also measured for the extracted viewports. The result confirms that the VPSNR improvement in Notile, VASTILE and Ozcinar have meaningful VMAF and SSIM improvement as well (2 VMAF score and 0.01 SSIM score improvements are considered meaningful). FlexiTile's flexibility to optimize between tiling layout or bandwidth allocation and its MILP modeling contribute to achieving this performance enhancement.

4.4 Comparison in Buffering Delays

To evaluate both the video quality and the buffering delay of FlexiTile against other methods, a dataset of real LTE channel traces [42] is used with the head movement trace of the 30 test users to simulate a realistic viewing scenario. The streaming server is provided with pre-configured tiling layouts for five distinct bandwidth levels, allowing users to select the most suitable option based on their available bandwidth. Table 2 presents the average video quality and buffering delay for different tiling methods. The results demonstrate that while FlexiTile-G achieves the highest video quality (42.15 dB), its buffering delay is at most 3% higher than that of other methods.

4.5 Comparison in Computational Complexity

to evaluate the computational complexity of FlexiTile, FlexiTile-G the most general form of FlexiTile, is compared to FlexiTile-C, a variant that utilizes viewport clustering. In FlexiTile-C, numerous test viewports, representing the access pattern map of the video, are clustered into a smaller set of representative viewports. Each representative viewport encompasses the viewing area of all viewports within its cluster. This

clustering technique effectively reduces the number of constraints in the MILP problem, leading to a significant improvement in optimization speed.

Fig. 7 illustrates the comparison result. In this figure, the pre-processing of basic tiles, candidate tiles, and the calculation of the distortion distribution map are excluded as these operations remain identical for both methods. Additionally, Fig. 8, compares the video quality achieved by these two methods. Analyzing both figures reveal that employing 10 clusters reduces the complexity by up to 70% while maintaining comparable video quality.

4.6 Analysis of Available Bandwidth on Tiling Scheme

Our experiments provide an intriguing observation regarding the influence of available bitrate on the tiling scheme (i.e., layout and bitrate allocation). In most instances, the overall available bandwidth does not significantly impact the tiling layout. This observation is illustrated in Fig. 9, which depicts the tiling scheme of different videos optimized by FlexiTile for varying network bandwidth limitations (1 Mbps and 5 Mbps). While the tiling schemes for 1 Mbps and 5 Mbps are not identical, they exhibit notable similarities. This observation suggests that optimization can be performed for a single bandwidth level, and the same candidate tiles can be employed for other bandwidth levels, with their bitrates adjusted accordingly. This approach significantly reduces computational complexity on both the server and the client sides. On the server side, the number of optimization and encoding processes is reduced, while on the client side, tile representations are simplified, which in turn streamlines the optimization process for selecting the appropriate tiles and their corresponding bitrates for the viewer.

4.7 Subjective Tests

To complement the objective metrics, a subjective video quality assessment was conducted to evaluate the perceptual impact of the proposed encoding scheme and validate the findings derived from analytical evaluations. The test is based on Double Stimulus Continuous Quality Scale (DSCQS) test, adhering to the recommendations of ITU-R BT.500 [43], was employed. In this test DSCQS is a temporal consistency approach that presents participants with pairs of video sequences – a reference viewport and a viewport extracted from a tiling scheme encoded for 2 Mbps available bandwidth– and asks them to rate the quality difference on a continuous scale. This methodology minimizes adaptation effects and provides a robust measure of perceived quality.

Participants were naive viewers ranging from 25 to 37 years old. They were instructed to rate the quality of each viewport video sequence using a continuous slider scale ranging from 0 to 100. The Notile method is used as reference and other methods were compared to this reference method. To mitigate bias, the order of presentation of reference and other sequences was randomized.

The results of the DSCQS test were analyzed to obtain the Differential Mean Opinion Score (DMOS) which is presented partially in Table 3. The negative number of DMOS indicates that the scheme is better than the baseline. The results confirm that using FlexiTile (all variations) can lead to better perceived quality than the Notile baseline.

5 Conclusion

To enhance the quality of experience (QoE) in 360-degree tiled-based video streaming, we propose the FlexiTile method, which jointly optimizes the tiling layout and bandwidth allocation of the tiles. The optimization in FLexiTile is a Mixed Integer Linear programming based on set cover problem. The optimization employs linear estimation of the rate-distortion model for candidate tiles and distortion distribution maps (DDMs) to accurately estimate the distortion experienced by a user due to the overlap of their viewport with the candidate tiles. A multi-layer perceptron (MLP) regressor is utilized to predict the distortion of a tile at any given bitrate, eliminating the need for encoding the tile at multiple bitrates. Moreover, FlexiTile employs accurate linear models for the primary constraints in tiled-based streaming, including network bandwidth limitations, to jointly determine the optimal tiling layout and bitrate allocation that maximizes video quality.

FlexiTile uses the average viewport PSNR of users (VPSNR) for its quality metric. Our experiments show at least 0.19 dB improvement in viewport video quality compared to the state-of-the-art methods and 1.9 dB compared to traditional Notile streaming. Furthermore, FlexiTile can be employed to enhance the bitrate allocation of conventional fixed-tiled methods, leading to improved quality. Our experiments reveal 1.3 dB improvement in the quality of Uniform 4×2 tiling and 1.73 dB in the quality of Uniform 10×5 tiling.

Appendix A Modeling Various Constraints In FlexiTile

To demonstrate the versatility of the FlexiTile within a predefined bandwidth constrained environment, two representative design constraints are examined, chosen arbitrarily without compromising the generality of the findings.

A.1 Limited Server Storage Capacity

Conventional video tiling approaches utilize non-overlapping tiles. This strategy simplifies the client-side selection process by eliminating ambiguity in choosing the correct tiles. However, [9] demonstrates that employing overlapping tiles can enhance video quality. This improvement stems from the ability of overlapping tiles to leverage video redundancies, leading to more efficient compression.

While overlapping tiles offer quality benefits, they also incur increased storage requirements on the server due to the redundant information stored across tile boundaries. To mitigate this storage overhead, the MILP formulation incorporates a modification in the covering constraint (Eq. (A1)) that limits the number of times a basic tile can be covered by the final set of candidate tiles.

$$\begin{aligned} \text{minimize}: \quad & \sum_{v}\sum_{i} do(i,v)d_i \\ s.t. \quad & \sum_{i} T_i M_{ji} = S_{\text{limit}} \ \forall j \\ & \sum_{i} \lceil o(i,v) \rceil r_i \le R_{\text{limit}} \ \forall v \\ & d_i \ge a_{ij} r_i + T_i b_{ij} \, 1 \le j \le l \\ & T_i \le r_i \le T_i R_{\text{limit}} \end{aligned} \tag{A1}$$

The storage limit (S_{limit}) in Eq. (A1) significantly influences the resulting tiling strategy. In an extreme scenario where $S_{limit} = 1$, each basic tile is covered by only one candidate tile in the final solution. This scenario replicates the traditional nonoverlapping tile approach. Conversely, when $\sum_i T_i M_{ji} \geq 1$ (unlimited storage case), the optimization is unconstrained by tile boundaries, leading to a tiling layout with minimal distortion and superior quality. However, under this scenario, the client would require an additional optimization step to select the optimal set of tiles from the available overlapping tiles for a specific viewport to maximize the quality of the viewport video.

A.2 Limited Parallel Decoding Capability In The Client

Tiled-based streaming necessitates parallel decoding of tiles on the client-side. To ensure seamless playback for the viewer, the client must simultaneously decode all tiles within the current viewport. The number of concurrently decodable tiles is influenced the client processing capabilities. Typically, Head-Mounted Display (HMD) devices can decode videos concurrently up to the number of cores available in their processors. For instance, a quad-core processor can decode up to four videos simultaneously. Furthermore, each decoder possesses a maximum number of samples it can decode per second, referred to as the decoder level. This level, in conjunction with the desired frame rate, dictates the achievable video resolution. To incorporate these constraints into the formulation, two additional constraints are introduced (Eq. (A2)).

$$
\begin{aligned}
\text{minimize}: \quad & \sum_{v}\sum_{i} do(i,v)d_i \\
s.t. \quad & \sum_{i} T_i M_{ji} \geq 1 \forall j \\
& \sum_{i} \lceil o(i,v) \rceil r_i \leq R_{\text{limit}} \ \forall v \\
& d_i \geq a_{ij} r_i + T_i b_{ij} \, 1 \leq j \leq l \\
& T_i \leq r_i \leq T_i R_{\text{limit}} \\
& \sum_{i} \lceil o(i,v) \rceil T_i \leq P_{\text{limit}} \ \forall v \\
& T_i \sum_{j} M_{ji} \leq L_{\text{limit}} \ \forall i
\end{aligned}
\tag{A2}
$$

P_{limit} represents the client's limitation on the number of concurrently decodable tiles. It controls the total number of tiles within a viewport ($\sum_i \lceil o(i,v) \rceil T_i$) that can be decoded simultaneously. L_{limit} reflects the constraint imposed by the decoder level on the resolution of candidate tiles. It restricts the maximum number of basic tiles a single candidate tile can encompass. In fact, the last constraint ensures that the resolution of candidate tiles remains compatible with the decoder level by limiting the number of basic tiles covered by each candidate tile. By incorporating these constraints, the optimization considers both client processing capabilities and decoder limitations while selecting the optimal set of tiles for transmission.

References

[1] M. Hu, J. Chen, D. Wu et al., "TVG-Streaming: Learning User Behaviors for QoE-Optimized 360-Degree Video Streaming," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 31, no. 10, pp. 4107–4120, 2021, doi: 10.1109/TCSVT.2020.3046242.

[2] A. Yaqoob, M. A. Togou, G.-M. Muntean, "Dynamic Viewport Selection-Based Prioritized Bitrate Adaptation for Tile-Based 360° Video Streaming," *IEEE Access*, vol. 10, pp. 29377–29392, 2022, doi: 10.1109/ACCESS.2022.3157339.

[3] M. M. Hannuksela, Y.-K. Wang, A. Hourunranta, "An Overview of the OMAF Standard for 360° Video," in *2019 Data Compression Conference (DCC)*, 2019, pp. 418–427. doi: 10.1109/DCC.2019.00050.

[4] M. Xiao, C. Zhou, Y. Liu et al., "OpTile: Toward Optimal Tiling in 360-degree Video Streaming," in *Proceedings of the 25th ACM International Conference on Multimedia*, in MM '17. New York, NY, USA: Association for Computing Machinery, 2017, pp. 708–716. doi: 10.1145/3123266.3123339.

[5] C. Ozcinar, J. Cabrera, A. Smolic, "Visual Attention-Aware Omnidirectional Video Streaming Using Optimal Tiles for Virtual Reality," *IEEE J. Emerg. Sel. Top. Circuits Syst.*, vol. 9, no. 1, pp. 217–230, 2019, doi: 10.1109/JETCAS.2019.2895096.

[6] X. Chen, T. Tan, G. Cao, "Macrotile: Toward QoE-Aware and Energy-Efficient 360-Degree Video Streaming," *IEEE Trans. Mob. Comput.*, vol. 23, no. 2, pp. 1112–1126, 2024, doi: 10.1109/TMC.2022.3233022.

[7] N. Kan, J. Zou, C. Li et al., "RAPT360: Reinforcement Learning-Based Rate Adaptation for 360-Degree Video Streaming With Adaptive Prediction and Tiling," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 32, no. 3, pp. 1607–1623, 2022, doi: 10.1109/TCSVT.2021.3076585.

[8] C. Madarasingha, K. Thilakarathna, "VASTile: Viewport Adaptive Scalable 360-Degree Video Frame Tiling," in *Proceedings of the 29th ACM International Conference on Multimedia*, in MM '21. New York, NY, USA: Association for Computing Machinery, 2021, pp. 4555–4563. doi: 10.1145/3474085.3475613.

[9] C. Zhou, M. Xiao, Y. Liu, "ClusTile: Toward Minimizing Bandwidth in 360-degree Video Streaming," in *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*, 2018, pp. 962–970. doi: 10.1109/INFOCOM.2018.8486282.

[10] X. Corbillon, G. Simon, A. Devlic et al., "Viewport-adaptive navigable 360-degree video delivery," in *2017 IEEE International Conference on Communications (ICC)*, 2017, pp. 1–7. doi: 10.1109/ICC.2017.7996611.

[11] M. Graf, C. Timmerer, C. Mueller, "Towards Bandwidth Efficient Adaptive Streaming of Omnidirectional Video over HTTP: Design, Implementation, and Evaluation," in *Proceedings of the 8th ACM on Multimedia Systems Conference*, in MMSys'17. New York, NY, USA: Association for Computing Machinery, 2017, pp. 261–271. doi: 10.1145/3083187.3084016.

[12] A. T. Nasrabadi, A. Mahzari, J. D. Beshay et al., "Adaptive 360-degree video streaming using layered video coding," in *2017 IEEE Virtual Reality (VR)*, 2017, pp. 347–348. doi: 10.1109/VR.2017.7892319.

[13] J. Chakareski, R. Aksu, X. Corbillon et al., "Viewport-Driven Rate-Distortion Optimized 360º Video Streaming," in *2018 IEEE International Conference on Communications (ICC)*, 2018, pp. 1–7. doi: 10.1109/ICC.2018.8422859.

[14] L. Xie, Z. Xu, Y. Ban et al., "360ProbDASH: Improving QoE of 360 Video Streaming Using Tile-based HTTP Adaptive Streaming," in *Proceedings of the 25th ACM International Conference on Multimedia*, in MM '17. New York, NY, USA: Association for Computing Machinery, 2017, pp. 315–323. doi: 10.1145/3123266.3123291.

[15] C.-L. Fan, S.-C. Yen, C.-Y. Huang et al., "On the Optimal Encoding Ladder of Tiled 360° Videos for Head-Mounted Virtual Reality," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 31, no. 4, pp. 1632–1647, 2021, doi: 10.1109/TCSVT.2020.3007288.

[16] R. Ghaznavi-Youvalari *et al.*, "Comparison of HEVC coding schemes for tile-based viewport-adaptive streaming of omnidirectional video," in *2017 IEEE 19th International Workshop on Multimedia Signal Processing (MMSP)*, 2017, pp. 1–6. doi: 10.1109/MMSP.2017.8122227.

[17] F. Qian, L. Ji, B. Han et al., "Optimizing 360 video delivery over cellular networks," in *Proceedings of the 5th Workshop on All Things Cellular: Operations, Applications and Challenges*, in ATC '16. New York, NY, USA: Association for Computing Machinery, 2016, pp. 1–6. doi: 10.1145/2980055.2980056.

[18] F. Qian, B. Han, Q. Xiao et al., "Flare: Practical Viewport-Adaptive 360-Degree Video Streaming for Mobile Devices," in *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking*, in MobiCom '18. New York, NY, USA: Association for Computing Machinery, 2018, pp. 99–114. doi: 10.1145/3241539.3241565.

[19] D. Podborski *et al.*, "HTML5 MSE Playback of MPEG 360 VR Tiled Streaming: JavaScript Implementation of MPEG-OMAF Viewport-dependent Video Profile

with HEVC Tiles," in *Proceedings of the 10th ACM Multimedia Systems Conference*, in MMSys '19. New York, NY, USA: ACM, 2019, pp. 324–327. doi: 10.1145/3304109.3323835.

[20] R. Skupin, Y. Sánchez, C. Hellge, T. Schierl et al., "Region-Of-Interest Coding Schemes For Http Adaptive Streaming With VVC," in *2022 IEEE International Conference on Image Processing (ICIP)*, 2022, pp. 3196–3200. doi: 10.1109/ICIP46576.2022.9897576.

[21] M. M. Hannuksela, S. Deshpande, "VVC for Immersive Video Streaming," in *Proceedings of the 2nd Mile-High Video Conference*, in MHV '23. New York, NY, USA: Association for Computing Machinery, 2023, pp. 52–58. doi: 10.1145/3588444.3591004.

[22] M. Hosseini, V. Swaminathan, "Adaptive 360 VR Video Streaming: Divide and Conquer," in *2016 IEEE International Symposium on Multimedia (ISM)*, 2016, pp. 107–110. doi: 10.1109/ISM.2016.0028.

[23] S. Petrangeli, V. Swaminathan, M. Hosseini et al., "Improving Virtual Reality Streaming using HTTP/2," in *Proceedings of the 8th ACM on Multimedia Systems Conference*, in MMSys'17. New York, NY, USA: Association for Computing Machinery, 2017, pp. 225–228. doi: 10.1145/3083187.3083224.

[24] K. Q. M. Ngo, R. Guntur, W. T. Ooi, "Adaptive encoding of zoomable video streams based on user access pattern," in *Proceedings of the Second Annual ACM Conference on Multimedia Systems*, in MMSys '11. New York, NY, USA: Association for Computing Machinery, 2011, pp. 211–222. doi: 10.1145/1943552.1943581.

[25] Y. Guan, C. Zheng, X. Zhang et al., "Pano: optimizing 360° video streaming with a better understanding of quality perception," in *Proceedings of the ACM Special Interest Group on Data Communication*, in SIGCOMM '19. New York, NY, USA: Association for Computing Machinery, 2019, pp. 394–407. doi: 10.1145/3341302.3342063.

[26] J. Chen, M. Hu, Z. Luo et al., "SR360: boosting 360-degree video streaming with super-resolution," in *Proceedings of the 30th ACM Workshop on Network and Operating Systems Support for Digital Audio and Video*, in NOSSDAV '20. New York, NY, USA: Association for Computing Machinery, 2020, pp. 1–6. doi: 10.1145/3386290.3396929.

[27] S. Chen, Y. Zhang, Y. Li, Z. Chen, Z. Wang, "Spherical Structural Similarity Index for Objective Omnidirectional Video Quality Assessment," in *2018 IEEE International Conference on Multimedia and Expo (ICME)*, 2018, pp. 1–6. doi: 10.1109/ICME.2018.8486584.

[28] M. G. Martini, "Measuring Objective Image and Video Quality: On the Relationship Between SSIM and PSNR for DCT-Based Compressed Images," *IEEE Trans. Instrum. Meas.*, vol. 74, pp. 1–13, 2025, doi: 10.1109/TIM.2025.3529045.

[29] R. G. de A. Azevedo, N. Birkbeck, I. Janatra, B. Adsumilli, and P. Frossard, "Subjective and Viewport-based Objective Quality Assessment of 360-degree videos," *Electron. Imaging*, vol. 32, no. 9, pp. 284-1-284–1, 2020, doi: 10.2352/ISSN.2470-1173.2020.9.IQSP-284.

[30] R. G. de A. Azevedo, N. Birkbeck, I. Janatra, B. Adsumilli, and P. Frossard, "A Viewport-Driven Multi-Metric Fusion Approach for 360-Degree Video Quality Assessment," in *2020 IEEE International Conference on Multimedia and Expo (ICME)*, 2020, pp. 1–6. doi: 10.1109/ICME46284.2020.9102936.

[31] N. Birkbeck, C. Brown, and R. Suderman, “Quantitative evaluation of omnidirectional video quality,” in *2017 Ninth International Conference on Quality of Multimedia Experience (QoMEX)*, 2017, pp. 1–3. doi: 10.1109/QoMEX.2017.7965684.

[32] M. Yu, H. Lakshman, and B. Girod, “A Framework to Evaluate Omnidirectional Video Coding Schemes,” in *2015 IEEE International Symposium on Mixed and Augmented Reality*, 2015, pp. 31–36. doi: 10.1109/ISMAR.2015.12.

[33] F. Song, C. Zhu, Y. Liu, Y. Zhou, and Y. Liu, “A new GOP level bit allocation method for HEVC rate control,” in *2017 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*, 2017, pp. 1–4. doi: 10.1109/BMSB.2017.7986230.

[34] H. Guo, C. Zhu, S. Li, and Y. Gao, “Optimal Bit Allocation at Frame Level for Rate Control in HEVC,” *IEEE Trans. Broadcast.*, vol. 65, no. 2, pp. 270–281, 2019, doi: 10.1109/TBC.2018.2847445.

[35] M. Zhou *et al.*, “Global Rate-Distortion Optimization-Based Rate Control for HEVC HDR Coding,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 30, no. 12, pp. 4648–4662, 2020, doi: 10.1109/TCSVT.2019.2959807.

[36] W.-C. Lo, C.-L. Fan, J. Lee, C.-Y. Huang, K.-T. Chen, and C.-H. Hsu, “360° Video Viewing Dataset in Head-Mounted Virtual Reality,” in *Proceedings of the 8th ACM on Multimedia Systems Conference*, in MMSys’17. New York, NY, USA: Association for Computing Machinery, 2017, pp. 211–216. doi: 10.1145/3083187.3083219.

[37] S. Tomar, “Converting video formats with FFmpeg,” *Linux J*, vol. 2006, no. 146, p. 10, June 2006.

[38] “x265, the free H.265/HEVC encoder - VideoLAN.” Accessed: Aug. 27, 2025. [Online]. Available: https://www.videolan.org/developers/x265.html

[39] “HTTP Live Streaming (HLS) authoring specification for Apple devices,” *Apple Developer Documentation*. Accessed: Aug. 27, 2025. [Online]. Available: https://docs.developer.apple.com/documentation/http-live-streaming/hls-authoring-specification-for-apple-devices

[40] “IBM ILOG CPLEX Optimization Studio.” Nov. 2024. Accessed: Aug. 27, 2025. [Online]. Available: https://www.ibm.com/products/ilog-cplex-optimization-studio

[41] N. Barman, M. G. Martini, and Y. Reznik, “Bjøntegaard Delta (BD): A Tutorial Overview of the Metric, Evolution, Challenges, and Recommendations,” *ArXiv E-Prints*, p. arXiv:2401.04039, Jan. 2024, doi: 10.48550/arXiv.2401.04039.

[42] D. Raca, J. J. Quinlan, A. H. Zahran, and C. J. Sreenan, “Beyond throughput: a 4G LTE dataset with channel and context metrics,” in *Proceedings of the 9th ACM Multimedia Systems Conference*, in MMSys ’18. New York, NY, USA: Association for Computing Machinery, 2018, pp. 460–465. doi: 10.1145/3204949.3208123.

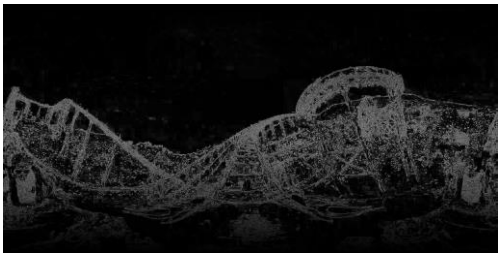
[43] S. Möller, “Quality of Video Transmission Systems,” in *Quality Engineering: Quality of Communication Technology Systems*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2023, pp. 95–120. doi: 10.1007/978-3-662-65615-0_6.

6 Captions

- **Fig. 1**: Distortion distribution map of a 2-second segment of a 360-degree video transcoded from 17.9 Mbps to 2 Mbps. (a) original video, (b) the DDM of the video
- **Fig. 2**: For all candidate tiles, the piece-wise linear approximation of the rate-distortion curve exploits its convexity property. Due to convexity, the actual distortion at any point is guaranteed to be less than or equal to the piece-wise line estimate. Consequently, minimizing distortion objective d_i subject to the corresponding linear constraints provides a provably tight lower bound on the true distortion, effectively achieving accurate estimation.
- **Fig. 3**: Different categories of motion vectors used for training the MLP regressor. Each candidate tile covers several basic tiles and each basic tile is split into 16×16 pixel blocks for motion vector computation. Motion vectors that remain within the basic tile. (b) Motion vectors that point to a place outside of the basic tile but within the candidate tile. c) Motion vectors that point to a place outside the candidate tile**.**
- **Fig. 4**: Overview of FlexiTile method framework
- **Fig. 5**: The rate-distortion curves of different videos encoded using uniform and FlexiTile-U methods
- **Fig. 6**: The rate-distortion curves of different videos
- **Fig. 7**: The computational complexity of FlexiTile and FlexiTile-C10
- **Fig. 8**: The rate-distortion comparison of FlexiTile-G and FlexiTile-C10
- **Fig. 9**: The tiling scheme of different videos (top) at 1Mbps (middle) and 5Mbps (bottom)
- **Table 1**: Features used to train the MLP regressor which estimates the PSNR for a given candidate tile at a given bitrate
- **Table 2**: Comparison of tested methods in terms of video quality and buffering delay
- **Table 3**: The computed DMOS from the result of the subjective video quality test

7 Figures

(a)



(b)

Fig. 1: Distortion distribution map of a 2-second segment of a 360-degree video transcoded from 17.9 Mbps to 2 Mbps. (a) original video, (b) the DDM of the video

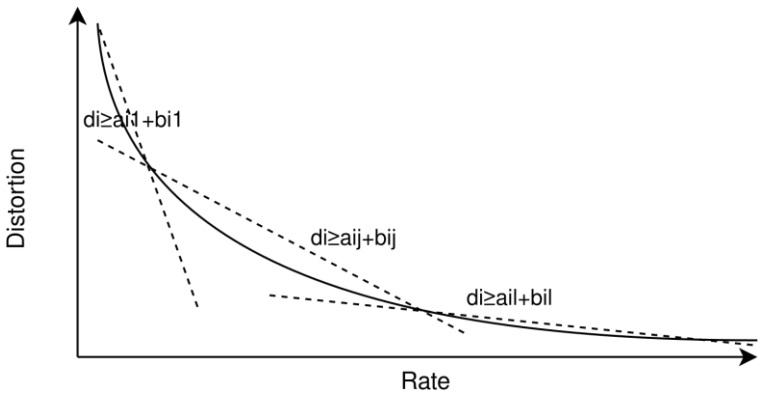


Fig. 2: For all candidate tiles, the piece-wise linear approximation of the rate-distortion curve exploits its convexity property. Due to convexity, the actual distortion at any point is guaranteed to be less than or equal to the piece-wise line estimate. Consequently, minimizing distortion objective d_i subject to the corresponding linear constraints provides a provably tight lower bound on the true distortion, effectively achieving accurate estimation.

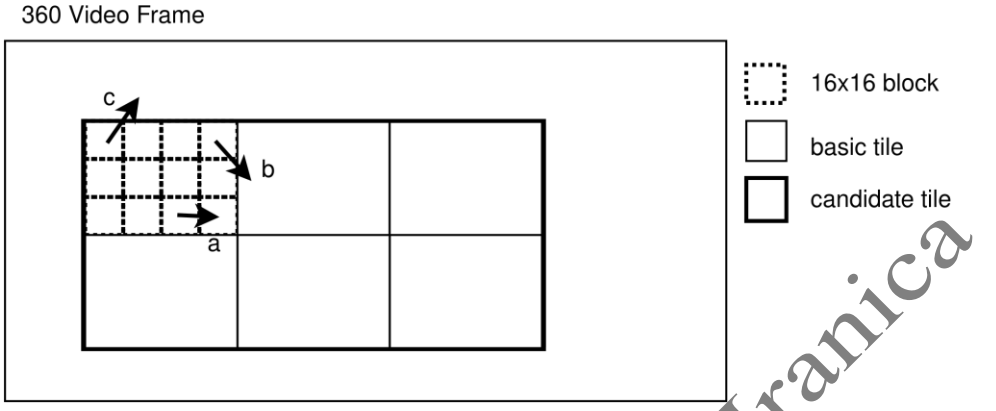


Fig. 3: Different categories of motion vectors used for training the MLP regressor. Each candidate tile covers several basic tiles and each basic tile is split into 16×16 pixel blocks for motion vector computation.
(a) Motion vectors that remain within the basic tile. (b) Motion vectors that point to a place outside of the basic tile but within the candidate tile. c) Motion vectors that point to a place outside the candidate tile

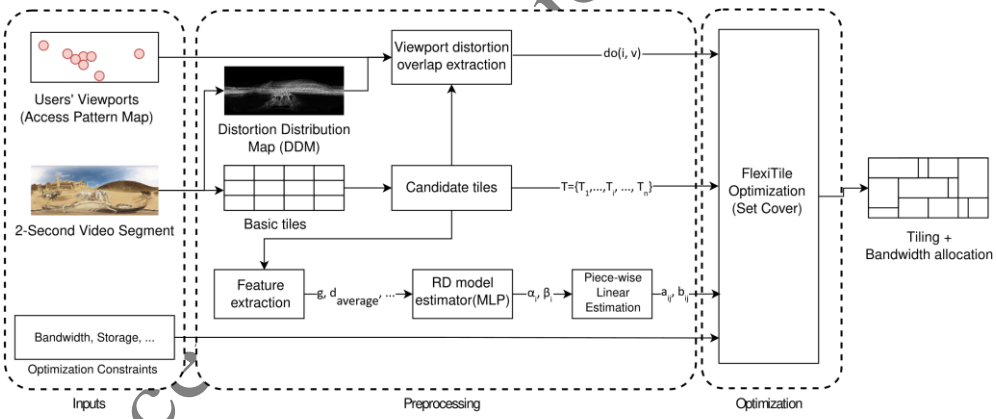


Fig. 4: Overview of FlexiTile method framework

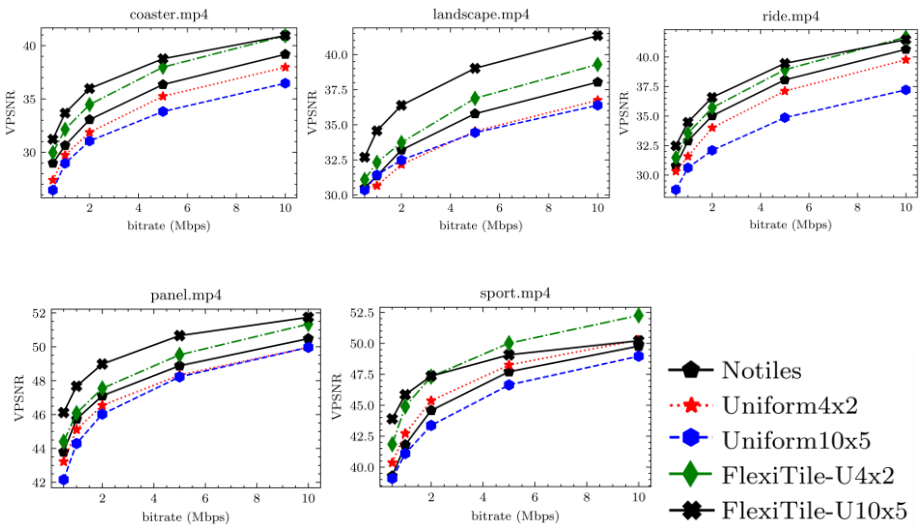


Fig. 5: The rate-distortion curves of different videos encoded using uniform and FlexiTile-U methods

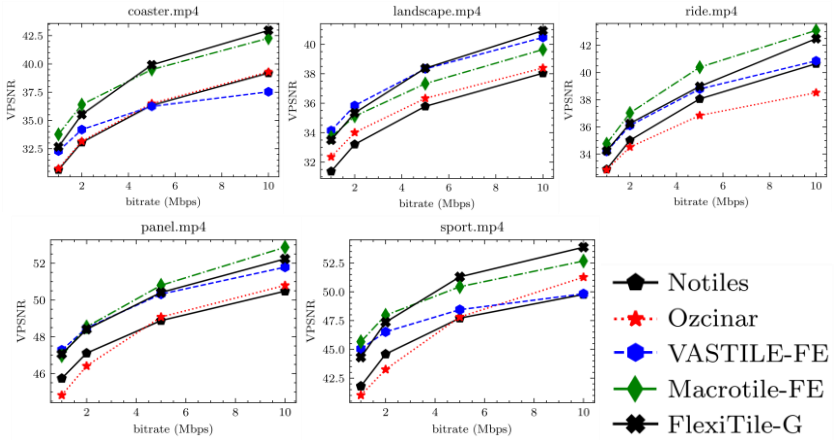


Fig. 6: The rate-distortion curves of different videos

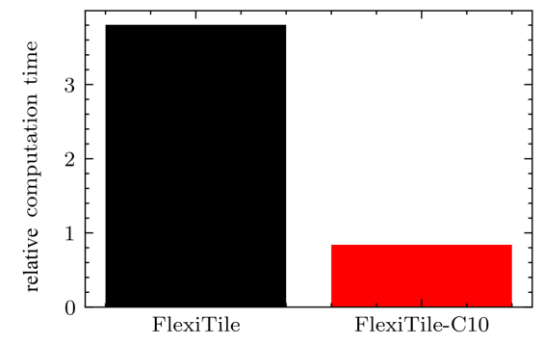


Fig. 7: The computational complexity of FlexiTile and FlexiTile-C10

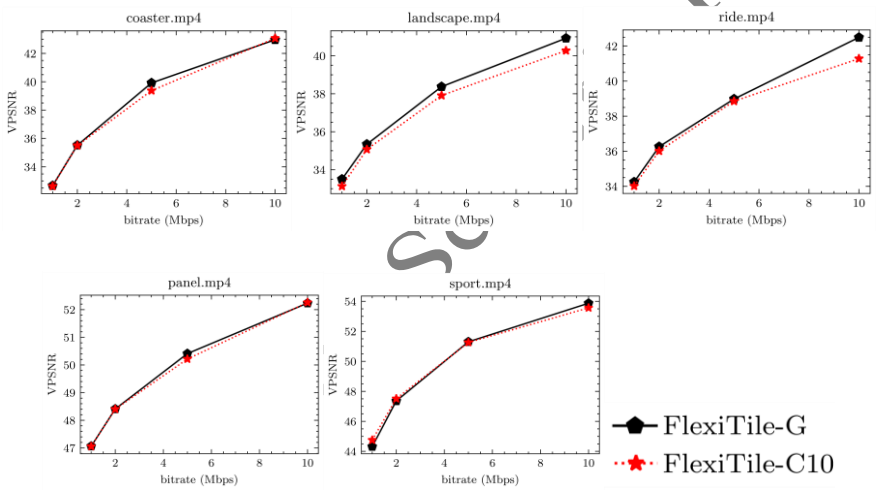
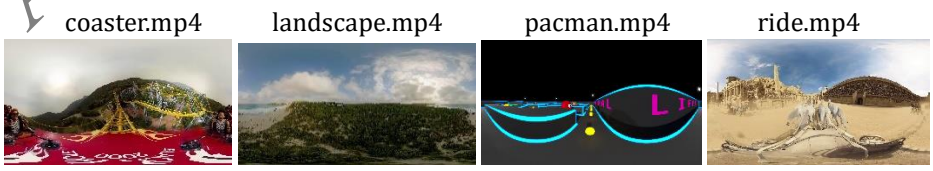


Fig. 8: The rate-distortion comparison of FlexiTile-G and FlexiTile-C10



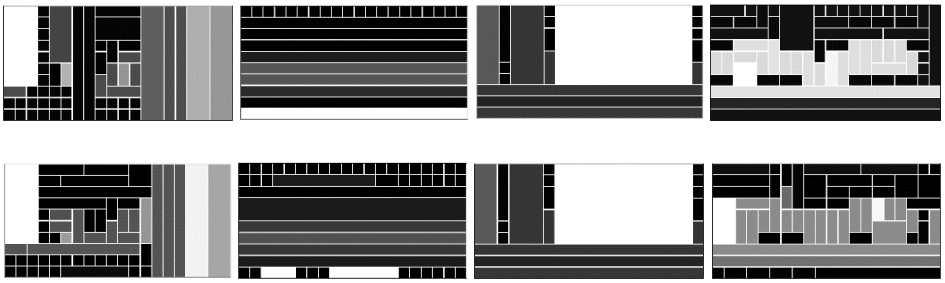


Fig. 9: The tiling scheme of different videos (top) at 1Mbps (middle) and 5Mbps (bottom)

8 Tables

Table 1: Features used to train the MLP regressor which estimates the PSNR for a given candidate tile at a given bitrate

g	The average gradient of the first frame (I-Frame) to consider the effect of intra coding
$\sum_i d_i$	Sum of distortion of all overlapping basic tiles.
$\sum_i m v_b$	The number of motion vectors that point to a block outside the boundary of their surrounding basic tile but inside the boundary of the surrounding candidate tile
mv_t	Number of motion vectors that point to a block outside of the boundary of their surrounding candidate tile.
$d_{average}$	Average effect of motion vector changes on distortion
n_t	Number of basic tiles overlapping with candidate tile t
r_i	The bitrate of the candidate tile

Table 2: Comparison of tested methods in terms of video quality and buffering delay

Tiling Method	Notiles	FlexiTile-U4x2	FlexiTile-U10x5	FlexiTile-G	Ozcinar	VASTILE-FE	Macrotile-FE
Mean VPSNR	40.10	41.33	41.15	42.15	39.97	40.61	41.91
Mean Buffering Delay (%)	12.91	10.46	12.48	13.32	11.89	11.96	11.93

Table 3: The computed DMOS from the result of the subjective video quality test

Viewport Video	Tiling Method	Average DMOS	Standard Deviation	95% Confidence Interval
coaster22	FlexiTile-U4x2	-7.08	18.06	7.23
	FlexiTile-G	-3.13	16.94	6.78
	Macrotile-FE	2.42	17.95	7.18

	VASTILE-FE	7.92	16.70	6.68
panel22	FlexiTile-U4x2	-6.71	7.68	3.07
	FlexiTile-G	-3.17	7.21	2.88
	Macrotile-FE	-5.33	9.64	3.86
	VASTILE-FE	-4.08	14.30	5.72
ride22	FlexiTile-U4x2	1.75	9.38	3.75
	FlexiTile-G	13.33	21.12	8.45
	Macrotile-FE	5.38	13.58	5.43
	VASTILE-FE	5.71	19.47	7.79
sport22	FlexiTile-U4x2	-3.08	7.80	3.12
	FlexiTile-G	-3.50	9.82	3.93
	Macrotile-FE	-4.29	9.31	3.72
	VASTILE-FE	-1.83	7.53	3.01
	Average DMOS	-0.356771		

9 Biography

Amir Ahmad Mohammadi received the B.Sc. and M.Sc. degrees in electrical engineering from Sharif University of Technology, Tehran, Iran, in 2012 and 2014, where he is currently pursuing the Ph.D. degree. His research concerns video encoding and multimedia streaming, focusing on optimizing streaming of 360-degree video content.

Mohammad Sharifkhani received the B.Sc. and M.A.Sc. degrees in electrical and computer engineering from the University of Tehran, Tehran, Iran, in 1998 and 2000, respectively. He received the Ph.D. degree from the University of Waterloo, Waterloo, ON, Canada, in 2006. He was a Postdoctoral Research Fellow at the University of Waterloo in 2007. He is currently an Associate Professor at the Department of Electrical Engineering, Sharif University of Technology, Tehran, Iran. Since 2008 he has published several scientific articles on the broad field of VLSI and Digital Systems. He served as technical committee member and reviewer of several IEEE conferences and journals, respectively. He founded several start-up companies on the broad field of video and image processing as well as machine intelligent systems. His current research is on low-power circuits and architectures, data converters, and application-specific processors for video and machine learning applications.