# Transcoding-Enabled Edge Caching and Delivery for Tile-Based Adaptive 360-Degree Video Streaming

Qiaoyu Lu, Chenglin Li, Junni Zou, Kexin Tang, Qi Wang, Hongkai Xiong
*School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, Shanghai, China*
{luqiaoyu,lcl1985,zou-jn,tkx1994-china,wang_qi,xionghongkai}@sjtu.edu.cn

***Abstract*—The high-definition and panorama characteristics of 360-degree video make its transmission in existing networks more challenging. To alleviate this problem, this paper leverages transcoding-enabled edge caching in tile-granularity and makes joint caching, transcoding and delivery decisions among multiple edge servers for adaptive 360-degree video streaming. We formulate the joint caching, transcoding and delivery decision-making problem as a multivariate nonlinear integer programming problem, aiming at minimizing the aggregate network operational cost and balancing the server load, subject to storage and computation capacity constraints of the edge servers. Through decomposition and relaxation approach, the original intractable optimization problem is solved by an effective iterative algorithm. Finally, simulation results show that the proposed iterative algorithm outperforms the existing schemes in terms of resource utilization, cost reduction and server coordination.**

***Index Terms*—Adaptive 360-degree video streaming, tile-granularity, edge caching, delivery, edge computing**

## I. Introduction

360-degree videos capture scenes from all directions using omnidirectional cameras, which provide the viewers with an immersive experience by free direction navigation. Compared to traditional videos, the high-definition and panorama characteristics of 360-degree video impose some challenges on its transmission over current networks. A user wearing a head-mounted display (HMD) can view at any time a partial scene of the entire 360-degree video, namely the *field of view* (FoV) of the user. Thus, streaming only the FoV of interest offers an effective bandwidth-saving approach. In practice, such a transmission adaptability can be realized by combining tiling and adaptive streaming techniques [1]. In tile-based adaptive streaming, a 360-degree video is segmented into several spatial tiles which are further divided into several temporal chunks. Each chunk in each spatial tile can then be encoded and delivered at different quality levels, adaptive to the user's interest and bandwidth constraint. However, since the video server is usually located far away from end-users, large latency and network congestion may occur at peak traffic hours during which the number of users competing for the bandwidth resource will increase dramatically.

Intuitively, edge caching/computing technique can be adopted to alleviate the amount of traffic at peak hours. Specifically, edge caching utilizes the storage of edge servers that locate much closer to end-users to pre-fetch videos during off-peak hours, and then use the locally cached videos to serve users' requests at peak hours. Edge computing exploits the powerful computation capacity of the edge servers to transcode video representations for end-users on the fly, so as to alleviate storage burden and compensate cache misses in edge caching. Existing works have explored a lot in edge caching for both traditional videos and 360-degree videos, e.g., distributed edge caching [2], FoV-aware caching update strategy [3], tile-based caching optimization [4]. However, studies on transcoding-enabled edge caching that collaboratively utilizes the storage and computation resources are still limited to traditional video streaming [5], [6], which also neglects the streaming in tile-granularity. In addition, the delivery decision determining which edge server serves which end-user can be jointly scheduled with caching and transcoding decision to further increase the resource utilization during the streaming process. However, existing works usually consider a partially joint decision [5] or a single task decision [4].

In this paper, we propose a transcoding-enabled edge caching scheme in tile-granularity and make a joint caching, transcoding and delivery decision among multiple edge servers for adaptive 360-degree video streaming. Our contributions are three-folded. 1) This is the first work to our knowledge to consider transcoding-enabled edge caching in tile-granularity for adaptive 360-degree video streaming. We focus on making the joint caching, transcoding and delivery decision among edge servers. Hence our scheme integrates the benefits of streaming at tile-granularity and the coordination among caching, transcoding and delivery. 2) We design mathematical models and formulate a multivariate nonlinear integer programming problem, aiming at minimizing the aggregate network operational cost and balancing the server load, subject to the storage and computation capacity constraints of edge servers. 3) Through decomposition and relaxation, the original intractable optimization problem is solved by an effective iterative algorithm, which shows outstanding performance in resource utilization, cost reduction and server coordination.

## II. System Models and Problem Formulation

Consider a scenario where the remote server stores the entire 360-degree video tile set, but locates far away from the end-users. At the network edge, multiple edge servers with certain storage and computation capabilities are surrounded by multiple end-users. Each user has connections to one or several edge servers and vice versa, resulting diverse server-user delivery pairs. The functional structure of the proposed transcoding-enabled edge caching system is illustrated in Fig. 1. One edge server is selected as the decision-making edge server, which contains modules for collecting and predicting requests from users and a decider module primarily responsible for making the joint caching, transcoding and delivery decision. Other edge servers act as the execution edge servers, consisting of modules for caching, transcoding and delivery. One decision

**978-1-7281-3723-0/19/$31.00 ©2019 IEEE**

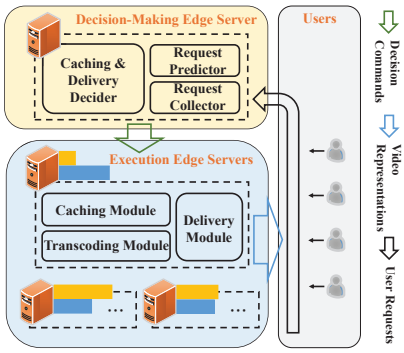



Fig. 1. Functional structure of edge servers.

cycle includes one caching phase and multiple subsequent transcoding and delivery phase. At the caching phase, the decision-making edge server makes analysis of the server capacities and user request characteristics, and then sends out the optimal caching decisions to the execution edge servers. The execution edge servers then catch tile representations from the remote server accordingly. At the transcoding/delivery phase, the decision-making edge server again analyzes and determines the optimal delivery pairs, and then the execution servers establish delivery paths and send cached or newly transcoded tile representations to the users.

Suppose that a number of $L$ 360-degree videos indexed by $l \in \{1, 2, \cdots, L\}$ are stored at the remote server. Each video is segmented into $N$ spatial tiles and $M$ temporal chunks, indexed by $n \in \{1, 2, \cdots, N\}$ and $m \in \{1, 2, \cdots, M\}$, respectively. Then each chunk in each tile is encoded into $Q$ quality levels indexed by $q \in \{1, 2, \cdots, Q\}$. Let $z_{l,m,n,q}$ denote the 360-degree video representation that is the $q$-th quality level of the $m$-th chunk in the $n$-th tile of the $l$-th video file, which is regarded as the minimum unit in video transmission. Thus the set $\mathcal{Z} = \{z_{l,m,n,q} | \forall l, m, n, q\}$ contains all tile representations that could be transmitted to users. We denote the size of a specific tile representation $z$ as $R_z$.

The connections between the edge servers and users can be represented as a bipartite graph $\mathcal{G} = \{\mathcal{S}, \mathcal{U}, \mathcal{E}\}$, where $\mathcal{S}$ is the set of $S$ execution edge servers, $\mathcal{U}$ represents the set of $U$ users, and an edge $(s, u) \in \mathcal{E}$ indicates a connection between user $u$ and server $s$. Accordingly, we denote the set of adjacent edge servers of user $u$ as $\mathcal{N}(u)$. We assume that the time is slotted and indexed by $t$. A transcoding/delivery operation executes within each time slot, while a caching operation spans $T$ time slots. In other words, one caching decision made at $t = 1$ is followed by $T$ transcoding/delivery decisions individually made at time slots $t \in \mathcal{T} = \{1, 2, \ldots, T\}$.

We use the notation $x_z^u(t) \in \{0, 1\}$ to represent the user request at time slot $t$, where $x_z^u(t)$ equals one when user $u$ requests a representation $z$ at time slot $t$ and zero otherwise. At the transcoding/delivery phase, $x_z^u(t)$ can be collected in real time at the beginning of each time slot $t$, which are probably generated by some rate adaptation algorithms like in [7]. At the caching phase, we leverage historic request traces collected previously and the tile heat map (e.g., derived by [8]), to predict future user requests when making caching decision.

We set two groups of decision variables: the distributed caching variables, $\alpha_z^s \in \{0, 1\}$, where $\alpha_z^s = 1$ represents that edge server $s$ caches representation $z$ and $\alpha_z^s = 0$ otherwise; and the delivery pair variables, $\beta_u^s(t) \in \{0, 1\}$, where $\beta_u^s(t) = 1$ represents that at time slot $t$, edge server $s$ serves user $u$ and $\beta_u^s(t) = 0$ otherwise. It is noted that the transcoding decision is implicated by these two groups of variables. Specifically, $x_z^u(t)\beta_u^s(t)(1 - \alpha_z^s) = 1$ indicates that edge server $s$ transcodes representation $z$ for user $u$ at time slot $t$. Based on these notations, we can express the caching storage consumption for edge server $s$ within $T$ time slots as:

$$B_{ca}^s = \sum_{z \in \mathcal{Z}} \alpha_z^s R_z. \tag{1}$$

Denoting the marginal operational cost for caching as $\omega_{ca}$, the total caching cost on $S$ edge servers in $T$ time slots is:

$$C_{ca} = \omega_{ca} \sum_{s \in \mathcal{S}} B_{ca}^s = \omega_{ca} \sum_{s \in \mathcal{S}} \sum_{z \in \mathcal{Z}} \alpha_z^s R_z. \tag{2}$$

Similar to [5], the transcoding cost is assumed proportional to the output bitrate (or size). Therefore, the computation resource consumption of edge server $s$ and the total transcoding cost at time slot $t$ can be respectively formulated as:

$$B_{tr}^s(t) = \sum_{u \in \mathcal{U}} \sum_{z \in \mathcal{Z}} x_z^u(t) \cdot \beta_u^s(t) \cdot (1 - \alpha_z^s) R_z, \tag{3}$$

$$C_{tr}(t) = \omega_{tr} \sum_{s \in \mathcal{S}} B_{tr}^s(t), \tag{4}$$

where $\omega_{tr}$ is the marginal operational cost for transcoding.

To realize effective coordination (load balance) among multiple edge servers, we denote $D_{max}(t)$ as the maximum delay caused by transcoding tasks of edge servers at time slot $t$,

$$D_{max}(t) = \max_{s \in \mathcal{S}} \left( \frac{W \cdot \sum_u \sum_z (x_z^u(t) \cdot \beta_u^s(t) \cdot (1 - \alpha_z^s) R_z)}{f_s} \right), \tag{5}$$

where $W$ is the required CPU cycles for transcoding one bit, and $f_s$ is the CPU frequency of edge server $s$. Minimizing $D_{max}(t)$ can thus ensure that the transcoding tasks are relatively evenly distributed among edge servers to avoid load imbalance which may lead to the waste of resources as well as a long waiting queue. Given $\omega_l$ as the marginal cost for load imbalance, $C_l(t) = \omega_l D_{max}(t)$ then represents the load imbalance cost at time slot $t$.

Therefore, we formulate the joint caching, transcoding and delivery decision-making problem as **P1**, aiming at minimizing the aggregate network operational cost and balancing the server load, subject to storage and computation capacity constraints of the edge servers.

$$\textbf{P1:} \quad \min_{\boldsymbol{\alpha}, \boldsymbol{\beta}} \quad C_{ca} + \frac{1}{T} \sum_{t \in \mathcal{T}} \{C_{tr}(t) + C_l(t)\}$$

$$\text{s.t.} \quad B_{ca}^s \le C A_{max}^s, \quad \forall s \in \mathcal{S}, \tag{6}$$

$$B_{tr}^s(t) \le T R_{max}^s, \quad \forall s \in \mathcal{S}, \forall t \in \mathcal{T}, \tag{7}$$

$$\sum_{s \in \mathcal{N}(u)} \beta_u^s(t) = \mathbf{1}(\sum_z x_z^u(t)), \forall u \in \mathcal{U}, \forall t \in \mathcal{T}, \tag{8}$$

$$\alpha_z^s \in \{0, 1\}, \quad \forall s \in \mathcal{S}, \forall z \in \mathcal{Z}, \tag{9}$$

$$\beta_u^s(t) \in \{0, 1\}, \quad \forall s \in \mathcal{S}, \forall u \in \mathcal{U}, \forall t \in \mathcal{T}. \tag{10}$$

The objective function is to minimize the sum of caching cost and the average transcoding cost and load imbalance cost over $T$ time slots. Storage resource keeps being consumed once the caching operation is complete, while the computation resource can be reused across time slots. Therefore, we employ the average transcoding cost occurred in $T$ time slots to measure the computation resource consumption. This accords with the aim of this paper, which devotes to reducing the network operational cost and balancing the server load. Eqs. (6) and (7) specify that the caching storage and transcoding resource consumption could not exceed the corresponding thresholds of each edge server, denoted by $CA^s_{max}$ and $TR^s_{max}$, respectively. Eq. (8) requires that each user is assigned exactly one neighbour edge server for the delivery of the requested tile representations, which assures efficient and reliable request responses. Here, $\mathbf{1}(\cdot)$ is an operation such that $\mathbf{1}(x) = 1$ when $x > 0$ and $\mathbf{1}(x) = 0$ when $x = 0$. The constraints in Eqs. (9) and (10) define the feasible regions for decision variables $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$.

## III. Proposed Solution and Algorithm

Observing **P1**, the nonlinearity is introduced by the operation $\mathbf{1}(\cdot)$ and the product of $\alpha^s_z$ and $\beta^s_u(t)$. Therefore, it is a multivariate nonlinear integer programming problem with coupling variables, which is discrete and intractable. Classical convex optimization approaches are not applicable, and the branch and bound algorithm for integer programming induces exponential complexity due to the huge search space. Hence, to derive the solution of **P1** effectively, we adopt the problem decomposition and relaxation approach, following the idea of block-coordinate descent algorithm [9].

Specifically, suppose that we have obtained the optimal value of $\boldsymbol{\alpha}$ as $\boldsymbol{\alpha}^*$, then **P1** evolves into a simpler problem **P2**, an integer programming problem without coupling variable $\boldsymbol{\alpha}$.

$$\textbf{P2:} \quad \min_{\boldsymbol{\beta}} \quad \frac{1}{T}\sum_{t\in\mathcal{T}}\{C_{tr}(t) + C_l(t)\}$$
$$\text{s.t.} \quad (7), (8), (10).$$

We can further decompose **P2** into a series of **P3** with respect to each time slot $t$, the solution of which together form the solution of **P2**, i.e., $\boldsymbol{\beta} = \{\boldsymbol{\beta(t)}|t \in \mathcal{T}\}$. The reason is that when the caching placement decision is determined in advance, the delivery pair decision at each time slot $t$ is independent of each other. Thus, we have:

$$\textbf{P3:} \quad \min_{\boldsymbol{\beta(t)}} \quad C_{tr}(t) + C_l(t)$$
$$\text{s.t.} \quad B^s_{tr}(t) \le TR^s_{max}, \quad \forall s \in \mathcal{S}, \tag{11}$$
$$\sum_{s\in\mathcal{N}(u)} \beta^s_u(t) = \mathbf{1}(\sum_z x^u_z(t)), \quad \forall u \in \mathcal{U}, \tag{12}$$
$$\beta^s_u(t) \in \{0,1\}, \quad \forall s \in \mathcal{S}, \forall u \in \mathcal{U}. \tag{13}$$

To solve **P3**, an auxiliary variable $\gamma$ is introduced and the term $C_l(t)$ in the objective function of **P3** is replaced with $\gamma$. Then, a series of auxiliary inequality constraints shown in Eq. (14) are added to **P3**. In this way, the max-operation in the objective function of **P3** is converted into linear constraints, resulting in an integer linear programming (ILP) problem. Furthermore, the continuous variable relaxation $\beta^s_u(t) \in [0,1]$ is adopted to relax **P3** to a linear programming (LP) problem, which can be easily resolved by the interior point method. When the relaxed solution is obtained, the discrete optimal solution can be derived through rounding approximation.

$$\omega_l \cdot \frac{W \cdot \sum_u \sum_z (x^u_z(t) \cdot \beta^s_u(t) \cdot (1-\alpha^s_z) R_z)}{f_s} \le \gamma, \quad \forall s \in \mathcal{S}. \tag{14}$$

Similarly, if we have obtained the optimal $\boldsymbol{\beta}$ as $\boldsymbol{\beta}^*$, **P1** becomes **P4**:

$$\textbf{P4:} \min_{\boldsymbol{\alpha}} \quad C_{ca} + \frac{1}{T}\sum_{t\in\mathcal{T}}\{C_{tr}(t) + C_l(t)\}$$
$$\text{s.t.} \quad (6), (7), (9).$$

Through auxiliary variable introduction and continuous variable relaxation, we can obatain an approximate optimal solution for **P4**. Based on the above problem decomposition, we can iteratively solve **P4** and a series of **P3** until they converge to obtain the joint caching decision. The delivery decision of each time slot $t$ can be derived by solving **P3** when the real-time requests arrive. Therefore, the algorithm for our proposed scheme can be summarized as Algorithms 1 and 2.

**Algorithm 1** Caching Algorithm

**Input:** Network topology $\mathcal{G}$; predicted user requests $x^u_z(t)$ from recent traces and tile heat map; and edge server parameters.
**Output:** $\boldsymbol{\alpha}$.
1: Initialize $\boldsymbol{\alpha} = \mathbf{0}$;
2: **repeat**
3: Let $\boldsymbol{\alpha}^* = \boldsymbol{\alpha}$;
4: **for** each $t \in \mathcal{T}$ **do**
5: Obtain $\boldsymbol{\beta(t)}$ using Algorithm 2;
6: **end for**
7: $\boldsymbol{\beta}^* = \{\boldsymbol{\beta(t)}|t \in \mathcal{T}\}$;
8: Formulate **P4** and do transform and relaxation;
9: Solve transformed **P4** with interior point method, solution $\overline{\boldsymbol{\alpha}}$;
10: Make approximations to get $\boldsymbol{\alpha}$: $\overline{\boldsymbol{\alpha}} \to \boldsymbol{\alpha}$;
11: **until** convergence.

**Algorithm 2** Transcoding/Delivery Algorithm

**Input:** Network topology $\mathcal{G}$; collected user requests $x^u_z(t)$; caching decisions $\boldsymbol{\alpha}^*$; and edge server parameters.
**Output:** $\boldsymbol{\beta(t)}$.
1: Formulate **P3** and do transform and relaxation;
2: Solve transformed **P3** with interior point method, solution $\overline{\boldsymbol{\beta}}(t)$;
3: Make approximations to get $\boldsymbol{\beta(t)}$: $\overline{\boldsymbol{\beta}}(t) \to \boldsymbol{\beta(t)}$.

## IV. Performance Evaluation

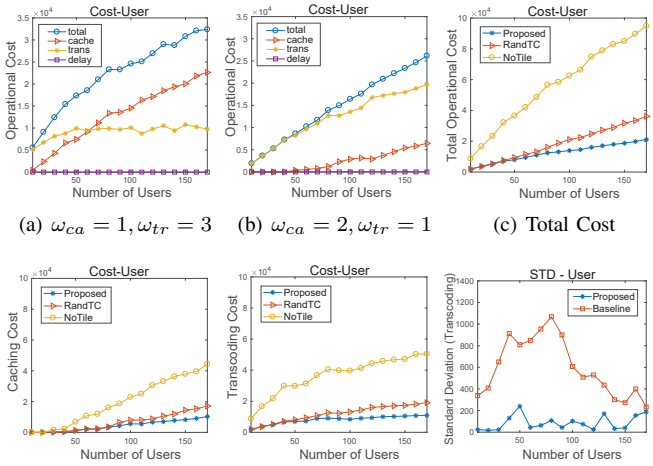
We use the test sequences from [8] to perform numerical simulations. Each 360-degree video is encoded into 4 quality levels and sliced into $8 \times 4$ tiles and chunks of 2s. The popularity of video contents follows Zipf distribution with parameter 0.56; the quality level preference is random; the chunk popularity follows a truncated exponential distribution in [6] with parameter $\mu = 4.6, K_l = 0.98$ and the tile preference follows the tile heat map. In an edge network, four execution edge servers are deployed and users are randomly

distributed. The number of users ranges from 10 to 170. We set $W = 20$ and $f_s = 500000$. $t$ is set to 10 s and $T$ to 100 s. To verify that the proposed scheme can find a good trade-off in utilizing two kinds of resources of the edge servers, substantially reduce network cost and achieve great load balance in server coordination, we show performance comparison in the following three aspects: resource utilization, cost reduction and server coordination.

*Resource utilization*: Figs. 2(a) and 2(b) show that the proposed scheme can achieve intelligent resource utilization between caching and transcoding when the number of users and the marginal operational cost change. When the number of users is small, the proposed scheme prefers transcoding to serve users no matter which kind of marginal operational cost is higher. This is because sparse users, namely sparse requests, exhibit low representation preferences, in which case caching easily misses and thus is inefficient. As the number of users and their requests increases, the proposed scheme can reach a proper trade-off between storage and computing resource utilization to reduce the total cost. Specifically, it intelligently adjusts the amount of video representations to cache and prioritizes the use of cheap resources.

*Cost reduction*: To illustrate the cost reduction of the proposed scheme, two other schemes are compared with: 1) random tile caching (RandTC) scheme, which randomly caches tiles instead of caching according to the tile heat map when a video chunk is predicted to be requested, and 2) non-tiling (NoTile) scheme, which gives up the tile-granularity decision and just caches/transcodes all tiles when a video chunk is requested. Figs. 2(c)- 2(e) present the total operational cost, the caching cost and the transcoding cost achieved by the three schemes. We observe that with the increase of number of users and requests, all cost increases. When there are only a few users, RandTC scheme and the proposed scheme perform equally well. But when the number of users becomes large, the proposed scheme outperforms RandTC. This results from that when the number of users is small, the tile heat map shows less accuracy in tile prediction. However, when it becomes large, the tile heat map comes into effect thus the proposed scheme outperforms RandTC. NoTile scheme always performs worse than others because processing the whole frame without considering the FoV characteristic brings unnecessary cost.

*Server coordination*: The standard deviation (STD) of transcoding resource consumption of $S$ servers is calculated to measure the server load balance, i.e., server coordination. Specifically, we compare our proposed scheme with a baseline scheme, which makes joint caching, transcoding and delivery decisions without considering the server load balance (i.e., removing the term $C_l(t)$ in the objective function of **P1**). In Fig. 2(f), the proposed scheme achieves a lower standard deviation than the baseline scheme, which indicates that at the transcoding and delivery phase, transcoding tasks are allocated as evenly as possible among the 4 edge servers to assure load balance. Note that the standard deviation varies with the number of users. This actually results from the different network topologies and user requests. Sometimes, the STD




(a) $\omega_{ca} = 1, \omega_{tr} = 3$ (b) $\omega_{ca} = 2, \omega_{tr} = 1$ (c) Total Cost

(d) Caching Cost (e) Transcoding Cost (f) Server Load

Fig. 2. Performance evaluation of the proposed scheme.

of the baseline scheme comes close to that of the proposed scheme. This is because random allocation adopted by the baseline can accidentally achieve the optimal allocation but it is not always guaranteed, while the proposed scheme is.

## V. Conclusion

We proposed a transcoding-enabled edge caching and delivery scheme in tile-granularity for adaptive 360-degree video streaming and made joint caching, transcoding and delivery decision for multiple edge servers. A multivariate nonlinear integer optimization problem has been formulated to minimize the aggregate network operational cost and to balance the server load. Through decomposition and relaxation approach, the original intractable optimization problem was solved by an iterative algorithm, the superior performance of which has been shown from the simulation results.

## Acknowledgement

This work was supported in part by the National Natural Science Foundation of China under No. 61871267, Grant 61831018, Grant 61529101, Grant 61622112, No. 61931023, and No. 61972256.

## References

[1] T. Stockhammer, "Dynamic adaptive streaming over HTTP: standards and design principles," in *Proc. ACM MMSys*, 2011.
[2] N. Golrezaei and *et al.*, "Femtocaching: Wireless video content delivery through distributed caching helpers," in *Proc. INFOCOM*, 2012.
[3] A. Mahzari, A. Taghavi Nasrabadi, and *et al.*, "FoV-aware edge caching for adaptive 360 video streaming," in *Proc. ACM MM*, 2018.
[4] G. Papaioannou and I. Koutsopoulos, "Tile-based caching optimization for 360 videos," *Proc. ACM MobiHoc*, 2019.
[5] Y. Jin and *et al.*, "Optimal transcoding and caching for adaptive streaming in media cloud: An analytical approach," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 25, no. 12, pp. 1914–1925, 2015.
[6] G. Gao, W. Zhang, and *et al.*, "Towards cost-efficient video transcoding in media cloud: Insights learned from user viewing patterns," *IEEE Trans. on Multimedia*, vol. 17, no. 8, pp. 1286–1296, 2015.
[7] S. Rossi and L. Toni, "Navigation-aware adaptive streaming strategies for omnidirectional video," in *Proc. IEEE MMSP*, 2017.
[8] X. Corbillon, F. De Simone, and G. Simon, "360-degree video head movement dataset," in *Proc. ACM MMSys*, 2017.
[9] Y. Xu and W. Yin, "A block coordinate descent method for regularized multiconvex optimization with applications to nonnegative tensor factorization and completion," *SIAM Journal on Imaging Sciences*, vol. 6, no. 3, pp. 1758–1789, 2013.