Viewport Prediction for Live 360-Degree Mobile Video Streaming Using User-Content Hybrid Motion Tracking

XIANGLONG FENG*, Rutgers University, USA and University of Nebraska-Lincoln, USA
VISWANATHAN SWAMINATHAN, Adobe Research, USA
SHENG WEI*, Rutgers University, USA and University of Nebraska-Lincoln, USA

360-degree video streaming has been gaining popularities recently with the rapid growth of adopting mobile head mounted display (HMD) devices in the consumer video market, especially for live broadcasts. The 360-degree video streaming introduces brand new bandwidth and latency challenges in live streaming due to the significantly increased video data. However, most of the existing bandwidth saving approaches based on viewport prediction have only focused on the video-on-demand (VOD) use cases leveraging historical user behavior data, which is not available in live broadcasts. We develop a new viewport prediction scheme for live 360-degree video streaming using video content-based motion tracking and dynamic user interest modeling. To obtain real-time performance, we implement the Gaussian mixture model (GMM) and optical flow algorithms for motion detection and feature tracking. Then, the user's future viewport of interest is generated by leveraging a dynamic user interest model that weighs all the features and motion information abstracted from the live video frames. Furthermore, we develop two enhancement techniques that take into consideration of user feedback for fast error recovery and view updates. Consequently, our predicted viewports are irregular and dynamically adjusted to cover the maximum portions of the actual user viewports and thus ensure a high prediction accuracy. We evaluate our viewport prediction approach using a public user head movement dataset, which contains the data of 48 users watching 6 360-degree videos. The experimental results show that the proposed approach supports sophisticated user head movement patterns and outperforms the existing velocity-based approach in terms of prediction accuracy. In addition, the motion tracking scheme introduces minimum latency overhead to ensure the quality of live streaming experience.

CCS Concepts: • **Human-centered computing → Portable media players**;

ACM Reference Format:
Xianglong Feng, Viswanathan Swaminathan, and Sheng Wei. 2019. Viewport Prediction for Live 360-Degree Mobile Video Streaming Using User-Content Hybrid Motion Tracking. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 3, 2, Article 43 (June 2019), 22 pages. https://doi.org/10.1145/3328914

1 INTRODUCTION

360-degree video streaming, or Virtual reality (VR) video streaming, has been gaining tractions recently with the rapid growth of adopting mobile head mounted display (HMD) devices in the consumer video market [10, 11, 32]. For example, NBC provided 85 hours of VR coverage for the 2016 Rio Olympics [2], which was available on

*The author conducted this work while studying or working at the two universities.

Authors' addresses: Xianglong Feng, Rutgers University, 94 Brett Rd, Piscataway, NJ, USA, 08854, University of Nebraska-Lincoln, 256 Avery Hall, Lincoln, NE, USA, 68588-0115, xf56@scarletmail.rutgers.edu; Viswanathan Swaminathan, Adobe Research, 345 Park Ave, San Jose, CA, USA, 95110, vishy@adobe.com; Sheng Wei, Rutgers University, 94 Brett Rd, Piscataway, NJ, USA, 08854, University of Nebraska-Lincoln, 256 Avery Hall, Lincoln, NE, USA, 68588-0115, sheng.wei@rutgers.edu.

2474-9567/2019/6-ART43 $15.00
https://doi.org/10.1145/3328914

Samsung Gear VR devices [32]. MLB launched VR-enhanced broadcasts in June 2017 [35] with the “At Bat" app on Google’s Daydream headset [10]. The 360-degree video presents viewers with a fully immersive virtual environment controllable by head movements mimicking the physical world experience.

Given that most of the VR video use cases are sports events, a live VR broadcast is significantly more attractive than the video-on-demand (VOD) service [38] because of the timeliness of the sports events. However, most of the existing VR video deployments and studies to date have been focused on VOD only. For example, live broadcasts for the 2016 Rio Olympics were not deployed at scale as compared to VOD [2]. Also, live streaming was not available in MLB’s VR broadcast [35]. The lack of live broadcasts has significantly hindered a wider adoption of VR video in its most promising market and thus impacted the more prevalent deployment.

The big gap between the market requirement and the actual deployment is caused by the brand new challenges in VR video streaming, which cannot be addressed by the state-of-the-art streaming technologies for traditional videos (e.g., HD and 4K). The VR video involves full 360-degree frames, which contain 6 to 8 times of video content or resolution compared to the traditional video. In other words, VR video requires 6X to 8X processing capabilities of the existing video codec to deliver the same video quality as traditional videos, which is extremely challenging. Without fully addressing this technical challenge, the VR video market would suffer from bottlenecks in bandwidth usage and user experiences [3, 12, 26, 28, 30, 36, 41].

To address the series of issues caused by the significantly increased data volume in VR video, several recent research efforts have focused on VR viewport prediction, which aims to save the bandwidth consumption by predicting the user’s view of interest and streaming the portion of the video that is likely to be watched with high priority. There are two categories of viewport prediction approaches that have been studied by the community: (1) conducting machine learning on the *past viewing behavior* of a large number of users who have watched the same or similar videos [8, 15, 16]; and (2) near-term viewport prediction based on the user’s *current viewing behavior* in the streaming session [3, 12, 17, 18, 28–30], such as the head position and the velocity of moving, by adopting extrapolations that the current head movement direction and speed would sustain for a period of time. Among these existing approaches, category (1) applies only to the VOD case, because the past viewing behavior is not available for live video that is streamed for the first time; category (2) could support the live scenario, but the extrapolation-based approach only applies to the simple movement pattern heading towards one single direction, and a common change in direction conducted by the viewer would easily result in inaccurate prediction results. In addition, the community has also developed algorithms and protocols from the system perspective aiming to improve the performance and scalability of VR streaming, such as the tiled streaming mechanism [6, 27, 43], tile optimization [43, 45–47], and optimizations via adopting new network protocols (e.g., HTTP/2 [27] and QUIC [19]). Most of the existing works are very effective in improving the performance and scalability of the streaming system. However, the existing protocols and algorithms are orthogonal to the viewport prediction mechanism, and they often assume a "perfect" viewport prediction scheme, which has not been achieved. More importantly, no existing research efforts have targeted live VR streaming.

We develop a new viewport prediction scheme that works with live 360 video streaming and complicated user head movement patterns. Different from all the existing approaches on the historical or current *user behavior*, our approach employs a user model combining the *video content* and the *user interests* for viewport prediction. Our key idea is that, even though the user behavior is hardly predictable, there is often a close correlation between the user’s viewport of interest and the moving objects in the video. It is because most users would focus and follow the most active objects in the 360-degree frames, which are often the objects in motion (e.g., the ball in a soccer game). Therefore, we employ computer vision-based techniques to detect and track the objects in motion, which serve as the predictor of the user’s future viewport of interest. In particular, we develop a set of algorithms to accommodate the various user movement patterns following a *Tracking-Recovery-Update-Evaluation (T-R-U-E)* workflow:

- *Tracking: VR motion tracking algorithm*, in which we apply the Gaussian mixture model (GMM) to detect the objects in motion [23], as well as the Shi-Tomasi algorithm [33] to detect and track the representative features of the moving objects;
- *Recovery: User feedback-based error recovery algorithm*, in which we consider the actual user viewports at runtime to automatically correct potential predication errors;
- *Update: Dynamic viewport update algorithm*, which dynamically adjusts the size of the predicted viewport to cover the potential viewports of interest while maintaining the lowest possible bandwidth consumption; and
- *Evaluation: Empirical user/video evaluation*, in which we build a prototype of the VR viewport prediction approach and evaluate it using empirical 360-degree videos and representative head movement datasets.

Our evaluation results reveal that the proposed algorithms outperform the velocity-based method by supporting all the representative head movement patterns in typical VR videos with low error rate. Our algorithms are the first to achieve bandwidth savings in live VR broadcast, and they can be seamlessly deployed into existing VR video systems that require viewport prediction. By presenting this work, we aim to provide the community with a new category of live VR viewport prediction method, which leverages a hybrid *video content* and *user interest* model and targets the *live streaming* use case.

2 PRELIMINARIES: LIVE VR STREAMING

2.1 Workflow of Live VR Streaming

A live VR streaming system delivers a live broadcast event from the live source to the end VR user, as shown in Figure 1. The system includes a server component and a client component connected by a network (i.e., typically a content distribution network). In the server component, the *live camera* captures the 360-degree views of the live event and passes the raw video frames to a *video packager*. Then, the *video packager* partitions the received video frames into small segments and hosts them on a *web server* for the client to request. In the client component, the *mobile HMD* first requests the video segments containing 360-degree views from the *web server* via HTTP and stores them into a video playback buffer (typically a few seconds) to accommodate for network jitters. Also, it receives the instant view orientation data from the motion sensor based on the user's actual head movements. With the 360-degree frames and the view orientation information, the VR library on the HMD generates and renders the specific VR views (for left and right eyes) that are a subpart of the 360-degree view. Finally, the *end user* watches the generated VR views and gains the VR experience.

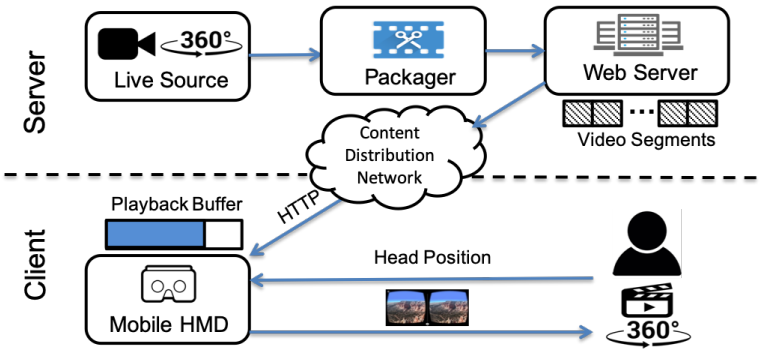


Fig. 1. Workflow of a typical live 360 video streaming system.

There are two unique features in the live VR streaming as compared to the traditional 2D video streaming. First, the VR system is *user-interactive*, in which the choice of video viewport is switched from the server (in

traditional video streaming) to the client (in VR video streaming) via head movements. Second, the final view presented to the user is a subpart of the entire 360-degree frame transmitted via the network. Both features determine that there is room for bandwidth optimization as long as the viewport of interest is known, which is heavily dependent on the user behavior (i.e., the head movement pattern).

2.2 User Head Movement Pattern

In live VR streaming experiences, the users' head movement pattern is complicated as they have the full freedom of choosing the viewports. However, the movements would not be fully random but correlated with the movements of the objects that attracted the users' interests. In particular, we observe four types of movement patterns that are prevalent in many 360-degree videos, which we represent using *i-j* move, where *i* represents the number of objects in motion, and *j* represents the average number of directions for each moving object. Obviously, both *i* and *j* would influence the user head movement pattern, assuming the intention of the user is typically watching the objects of interest in the video. We demonstrate the four movement patterns in Figure 2 and describe them as follows.

- *1-1* move, in which there is a single object of interest in the video, which moves along one single direction (e.g., a moving car on a straight road);
- *1-n* move, in which there is a single object of interest, which moves along multiple (i.e., *n*) directions (e.g., the ball in a soccer game);
- *m-n* move, in which there are multiple (i.e., *m*) objects of interest, which move along multiple (i.e., *n*) directions each (e.g., two groups of marching horses splitting into two different directions); and
- *Arbitrary* move: in which the user does not follow any objects of interest but switch viewports arbitrarily (e.g., watching around to check the surroundings and identify objects of their interests).

The various movement patterns pose significant challenges to the viewport prediction schemes. For example, the existing prediction method for live VR streaming, namely the velocity-based method [12, 28], is only effective towards the *1-1* move and would not apply to the other three movement patterns. It is because the velocity-based method cannot quickly adapt to the direction changes, considering that it must predict future viewports for a relatively long time duration equivalent to the length of the buffer (i.e., typically a few seconds), during which the probability of direction changes is high. Our aim in this paper is to develop a viewport prediction scheme that supports all the four head movement patterns.

3 SYSTEM ARCHITECTURE

Figure 3 shows the overall architecture of our motion tracking-based viewport prediction framework, which is deployed in the video packager as part of the server component presented in Figure 1. The shaded blocks represent the new components we develop in this work, and the unshaded block (i.e., the segmenter) represents the original component in the video packager. The core functionality of the motion tracking module depends on two video processing procedures on the raw video frames received from the live video camera: (1) motion detection, which differentiates the moving objects in the foreground from the static background; and (2) feature selection and tracking, which selects representative features to identify and thus track the moving objects. Both motion tracking and feature selection combined enable the system to track the moving objects, which are likely to be the user's viewport of interest.

Furthermore, we analyze the user viewing behavior and formulate the user interest model using the Bayes method. Based on the analysis, we apply two types of techniques to either recover from or prevent the prediction errors, in which case the predicted viewport is different from the actual user view. First, we develop an error recovery mechanism to dynamically adapt the predicted viewport to the actual user view once an error is detected. Although it cannot eliminate the error that has already occurred, it prevents the viewport prediction mechanism

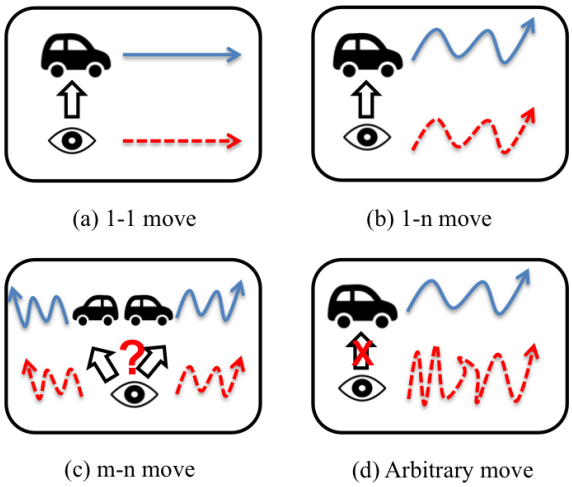


Fig. 2. Typical user movement patterns in live 360 video streaming.

from issuing more erroneous predictions. Second, to reduce the number of prediction errors, we develop a dynamic viewport update algorithm that generates variable sizes of viewports by considering both the tracked motion and the user's current velocity. In this way, the predicted viewport has a significantly higher probability of covering the potential user views even if the user's head movement is following one of the complicated moving patterns (e.g., *m-n* move).

While the proposed motion tracking and error handling mechanisms are computation-intensive, all the components are deployed in the video packager, which runs at the server side that has sufficient computing capabilities without consuming significant client-side resources. Instead, our approach offloads the workload of VR view generation from the client to the server, which further reduces the client overhead, such as latency and power, as well as the network overhead, such as bandwidth.

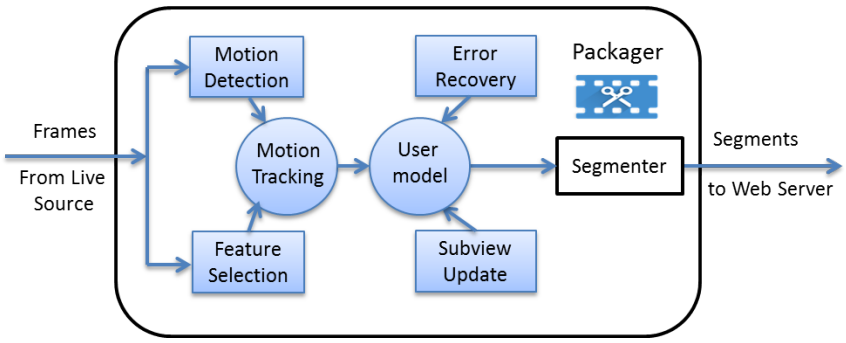


Fig. 3. Overall architecture of the motion tracking-based viewport prediction system.

4 LIVE VR VIEWPORT PREDICTION

4.1 Motion Tracking-based VR Viewport Prediction

In most VR videos, the viewer typically spends the majority of time watching active objects that are in motion. Therefore, the motion of the objects in the video provides us with an effective indication of the user's viewport of interest. In other words, the VR viewport prediction problem can be partially converted into a motion tracking problem based on the VR video content, which forms the basis of our solution.

To achieve the goal of motion tracking in VR videos, there are two major challenges that we must address: (1) *motion detection*, in which we detect the objects in motion by extracting the moving foreground from the background of the video; and (2) *feature selection*, in which we identify representative features in the detected moving objects so that they can be re-identified and thus tracked in the future frames. In this work, we apply the Gaussian mixture models (GMM) [48] to accomplish motion detection, as well as the Shi-Tomasi algorithm [33] to address the motion tracking problem, which we describe in details following a motivating example shown in Figure 4. Figure 4(a) shows the sample VR video we adopt in this research, which involves a group of marching horses as the major active objects [20].

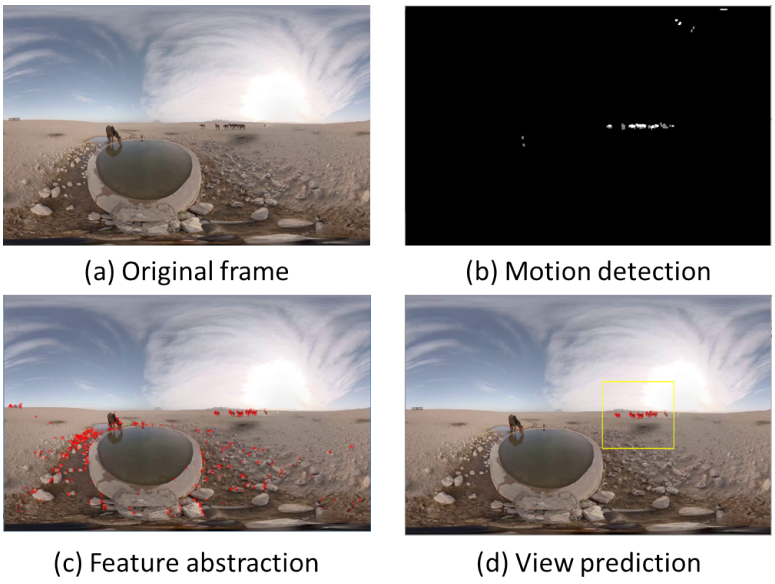


Fig. 4. Demonstration of the motion tracking-based VR viewport prediction (video courtesy of [20]).

Step 1: Motion detection. We apply the GMM foreground subtraction algorithm [48] to detect the moving regions (i.e., regions that contain moving objects) in the video, as shown in Figure 4(b), where the marching horses are marked with small white blocks.

Step 2: Feature selection and filtering. We employ the Shi-Tomasi algorithm [33] to detect representative features from the video, which are shown in Figure 4(c) with the red tags. These features are representative because they cover the special textures or structures in the video. Although the marked features provide a full representation of the video, there is typically a huge number of features that are challenging if not impossible to track, as demonstrated in Figure 4(c). Therefore, we apply two filtering methods to reduce the number of features for tracking. First, we compare the features from the current frame with those from the previous frame and only

keep those that exist in both frames. Second, we employ the motion detection results in Step 1 and filter out the features that are on static objects (i.e., in the background). Figure 4(d) shows the remaining features after the two filtering procedures, which we adopt in the motion tracking scheme.

Step 3: Viewport generation. The remaining features after Step 2 would typically spread in a broad area that cannot be covered by one single user view. In other words, there may be multiple moving objects in the entire 360-degree video that the viewer may choose to watch, as represented by the *m-n* move. To address this problem, we develop a systematic method to generate the single predicted viewport containing the objects that are most likely to be watched. Our intuition is that viewers typically pay most of their attentions on two types of objects: (1) those that are closest to the viewer; and (2) those that are "significant" in terms of physical shapes. We note that both types of objects would contain the most dense and the largest number of features and, therefore, we can calculate the center of the predicted user view by calculating the barycenter of all the remaining features. In particular, given the remaining list of features, $\overrightarrow{F} = [f_1, f_2, f_3....., f_k]$, where each $f_i (i = 1...k)$ represents the pixel coordinates of the feature $f_i =< f_i^{(x)}, f_i^{(y)} >$, the center coordinates of predicted view (l_x, l_y) can be calculated as:

$$l_x = \frac{1}{k}\sum_{i=1}^{k} f_i^{(x)}; \qquad l_y = \frac{1}{k}\sum_{i=1}^{k} f_i^{(y)} \tag{1}$$

Note that the predicted center coordinates may not be the exact center of the user's actual view, even if the predicted area contains the objects watched by the viewer. Therefore, the size of the predicted view may be larger than that of the actual view. To accommodate for this potential mismatch, we apply a scaling factor Sc while generating the predicted view. In particular, given the size of the user view S_{user}, the size of the predicted view is calculated as $S_{Pre} = Sc \cdot S_{user}$. Figure 4(d) shows the final viewport prediction results with $Sc = 1.2$, which contains the group of horses that have the largest number of features.

4.2 User Feedback-based Error Recovery

The aforementioned motion tracking-based viewport prediction method is effective for prediction under standard movement patterns, such as *1-1* move and *1-n* move. For sophisticated movements such as *m-n* move and *arbitrary* move, the approach may fail due to the unique feature in VR streaming that the user has full control over their viewports of interest via head movements. Consequently, our content-based view prediction approach may be subject to a high error rate. We address this problem by developing a prediction error recovery scheme driven by user feedback at runtime. As presented in Figure 1, the video packager can retrieve the feedback information of the user's actual view via the reverse path from the user through HMD and web server. With the feedback information, our error recovery mechanism takes effective in the following two scenarios.

Scenario 1: No objects in motion. In the event where the motion detection scheme could not identify any objects in motion, it is highly likely that the user is watching static objects, creating an exceptional case that causes our motion tracking-based prediction to fail. In this scenario, we consider that the determining factor for the viewport prediction has switched from the video and the user to solely the user. Therefore, we apply the velocity-based method [12, 28] that considers only the user behavior for view prediction. Such a decision can be triggered by the motion detection module once a no-motion result is present. Then, once the user data is obtained from the feedback path, we can generate a user viewport location vector $\overrightarrow{L} = [l_1, l_2, l_3....., l_M]$, where l_i represents the user viewport location in the i^{th} frame, and M represents the number of frames in the video playback buffer. The user viewport velocity can be calculated as:

$$\overrightarrow{V} = \frac{(\overrightarrow{l_2 - l_1}) + (\overrightarrow{l_3 - l_2})....(\overrightarrow{l_M - l_{M-1}})}{M-1} = \frac{(\overrightarrow{l_M - l_1})}{M-1} \tag{2}$$

The predicted location in the next frame, l_{M+1}, can be calculated as:

$$l_{M+1} = l_M + \overrightarrow{V} \tag{3}$$

Scenario 2: Mismatch between the predicted view and the actual user view. Once the motion tracking scheme detects that the user's actual view is different from what has been predicted, it triggers a recovery process by tracking the objects that the user is actually watching. In particular, we apply the motion tracking method within the actual user view and determine the velocity of the moving objects that the user is actually watching. Given the matched features from the previous frame $\overrightarrow{FA} = [fA_1, fA_2, fA_3....., fA_p]$ and the ones from the current frame $\overrightarrow{FB} = [fB_1, fB_2, fB_3....., fB_p]$, the velocity is calculated as:

$$V_x = \frac{1}{p}\left(\sum_{i=1}^{p} fB_i^{(x)} - \sum_{i=1}^{p} fA_i^{(x)}\right), \quad V_y = \frac{1}{p}\left(\sum_{i=1}^{p} fB_i^{(y)} - \sum_{i=1}^{p} fA_i^{(y)}\right) \tag{4}$$

Assuming the predicted view center is (l_x, l_y), the modified view center is calculated as (l_x+V_x, l_y+V_y).

4.3 Dynamic Viewport Update Algorithm

The aforementioned error recovery mechanism can detect and recover from the erroneous viewport predictions under the *m-n* and *arbitrary* moves. However, since the recovery is after the fact and focuses on avoiding future errors, the VR streaming system would have already conducted re-transmissions of the wrongly predicted views, which causes additional bandwidth overhead. In this subsection, we develop a dynamic viewport update algorithm that aims to prevent prediction errors before the fact.

Our key idea is to enlarge the predicted viewport to cover all the potential moving objects under the *m-n* and *arbitrary* moves with high probability and, more importantly, dynamically adjust the size of the view to achieve an efficient bandwidth usage. To achieve this goal, we leverage the concept of tiles available in the H.265 video codec [14] in our algorithm design. Given a 360-degree VR video, we equally divide each 360-degree frame into $N = n \times n$ grids, and each grid is a tile. Therefore, the predicted viewport can be formed as a subset of the N tiles.

To better leverage the tile-based method, we analyze the user viewing behavior with the Bayes method. We assign each tile with one independent Bayes model, so that each tile can be updated independently. Assuming X represents the user viewport, Y represents the static content, and Z represents the moving objects, the user's viewport X in the future can be calculated as $P(X|Y,Z)$, where Y and Z are independent since the moving objects and the static background are independent in the video. Meanwhile, the user watching the static background/features can be represented by $P(X|Y)$, and the user watching the moving features can be represented by $P(X|Z)$. Given the online feedback of the user view (as the user's choice), where the user watching behavior on static background/features and the moving features are represented as $P(Y|X)$ and $P(Z|X)$ respectively, the user's viewport in the future can be calculated as:

$$P(X|Y,Z) = \frac{P(Y|X) \cdot P(Z|X) \cdot P(X)}{P(Y,Z)} \tag{5}$$

For each tile T_i, as soon as the user starts watching, we can obtain the Priori probability $P(Y_i|X_i)$ and $P(Z_i|X_i)$ for this particular video and user of the ith tile by the feedback. Then, based on the Bayesian model, we can calculate the Posterior probability to predict the user view $P(X_i|Y_i,Z_i)$ in the future.

We define two attributes for each tile: (1) the current status, which indicates if the tile belongs to the predicted view (i.e., labeled as *PREDICTED*) or not (i.e., labeled as *NONPREDICTED*); and (2) the lifetime, which indicates how long the tile will stay in the predicted view. For example, we can define three levels of lifetime, namely *ZERO*, *MEDIUM*, and *HIGH*, the values of which can be customized in the real system depending on the specific users and videos. The lifetime value will decrease by one level at the interval of buffer length, and it can also be

set to increase or decrease by different number of levels. We develop a 3-step workflow to achieve the optimal viewport prediction, namely system initialization, frame-level update, and buffer-level update.

System Initialization. During initialization, our view update algorithm labels all the N tiles as the predicted view, by setting the attribute "status" as *PREDICTED* and setting the attribute "lifetime" to *MEDIUM* . In other words, the system sends the entire 360-degree frame to the user as bootstrap. At this point, the lifetime level increases from *ZERO* to *MEDIUM*. The reason why we assign all the tiles to the predicted view and grant them with *MEDIUM* lifetime is to allow for the "look around" type of movement when the viewer first starts the video session, which may cover an arbitrary viewport in the 360-degree frame. Once the system is initialized, there are two major check points where the view update algorithm would take effective, namely the frame-level update and the buffer-level update.

Frame-level update. Given each frame, we apply a modified motion tracking algorithm that selects the features in the motion area without further filtering using the density of the features. Instead, we employ multiple views (i.e., with multiple tiles) to cover an enlarged area containing all the features on the moving objects as the predicted viewport. In this way, we are able to accommodate for the user behavior in the *m-n* move. To achieve this goal, we design a frame-level algorithm, where we label the chosen tiles as *PREDICTED* and set their lifetimes to *HIGH*, under the intuition that the objects in motion or the user preferred static background/features would typically stay within the user's viewport of interest for an extended period of time.

Buffer-level update. We retrieve the user feedback (i.e., the user's actual view) at the interval of buffer length, which enables us to conduct the following two types of adjustments to the tile attributes, as shown in Pseudocode 1. First, for the tiles that have overlap with the user's actual view, we set them as *PREDICTED* with *HIGH* lifetime, because they are factually in the user's viewport of interest. Second, for each tile that does not overlap with the user's actual view and that is currently in the predicted view, we decrease their lifetime by 1. If such an operation causes the lifetime to reach *ZERO*, we reset the status of the corresponding tile to *NONPREDICTED*, i.e., move it out of the predicted viewport.

Pseudocode 1 Buffer-level Update.

```
1:  procedure BuffLevelUpdate(T[N])                          ▷ Update each tile
2:      i ← N
3:      while i ≠ 0 do
4:          if T[i]_area ⋂ UserView ≠ ∅ then
5:              T[i].status ← PREDICTED
6:              T[i].lifetime ← HIGH
7:          else
8:              if T[i].status == PREDICTED then
9:                  T[i].lifetime = T[i].lifetime − 1;
10:             end if
11:         end if
12:         if T[i].lifetime == ZERO then
13:             T[i].status = NONPREDICTED;
14:         end if
15:         i ← i − 1
16:     end while
17: end procedure
```

5 EXPERIMENTAL RESULTS

5.1 Experimental Setup

5.1.1 Test Video and User Dataset. To fully evaluate our viewport prediction mechanism, we employ two different VR video head movement datasets: (1) a lab dataset obtained from our IRB-approved user study, involving 4 users recruited from our lab (3 graduate students and 1 undergraduate student) watching 10 test videos. The 4 users have basic understandings about VR video streaming and are aware of the purpose of our research. We screen-recorded their viewing session on the HMD to use as the actual user viewport for the prediction accuracy evaluation. (2) a public VR user dataset [42] involving 48 users watching 6 VR videos. These users are not associated with our lab or institution, and they are not aware of our specific research about viewport prediction. The dataset contains timestamped user head movement data for the evaluation of viewport prediction accuracy.

Table 1 describes the 10 test videos in our lab study. The 10 videos cover a variety of scenarios including animals, humans, indoors/outdoors, real world/virtual world, etc. The frame rate of this set of videos is 30 FPS and the resolution is 1280×720. Table 1 also presents an approximate estimate of the object moving pattern in each video, where *L*, *M*, and *H* represent *Low*, *Medium*, and *High* activities, respectively. The estimate demonstrates the broad range of video patterns covered by our test benchmark. Furthermore, we invite 4 lab users to watch the 10 videos in order to collect the actual head movement data for analysis and evaluate the performance of the viewport prediction algorithms. Note that each of the 4 users watches all the 10 videos and, therefore, we have involved 40 test cases covering a diverse set of user-video combinations. More importantly, our experiments have very well covered all the 4 head movement patterns analyzed in Section 2.2, which is the key technical requirement for the evaluation.

Table 1. Test videos in lab user study including an approximate estimate of object moving patterns (*L*-Low; *M*-Medium; *H*-High).

Video Name	1-1 move	1-n move	m-n move
Cat [1]	L	L	M
Airbus [5]	L	L	M
Video Game [22]	H	L	L
Cow [7]	L	L	H
Giraffe [21]	H	M	L
Girls Dancing [37]	L	L	H
Surveillance [34]	M	M	M
Marine [39]	L	L	M
Lady Singing [4]	H	H	L
Wild Horse [20]	L	L	H

Table 2 shows the 6 test videos we obtained from the public dataset [42], which are all captured by fixed cameras. In the evaluation, we downsample the original videos to 1280×720 resolution for performance considerations, and the frame rates of the videos are 30 FPS, 29.9 FPS or 25 FPS.

5.1.2 Evaluation System. We deploy our viewport prediction prototype on two PCs: (1) PC1 is equipped with Intel i7 1.8GHZ CPU and 8GB RAM; and (2) PC2 is equipped with Intel E5-2623 v3 3.0GHz CPU and 32GB RAM. We employ PC1 for the experiments with our lab dataset and PC2 for the experiments with the public dataset.

Table 2. Test videos from public dataset [42].

Index	Video Name	Category	Content
0	Korean	Performance	Weekly Idol-Dancing
1	VoiceToy	Performance	Damai Music Live-Vocie Toy
2	Female Basketball	Sport	Female Basketball Match
3	Fighting	Sport	SHOWTIME Boxing
4	Anitta	Performance	Gear 360: Anitta
5	Reloaded	Performance	VR Concert

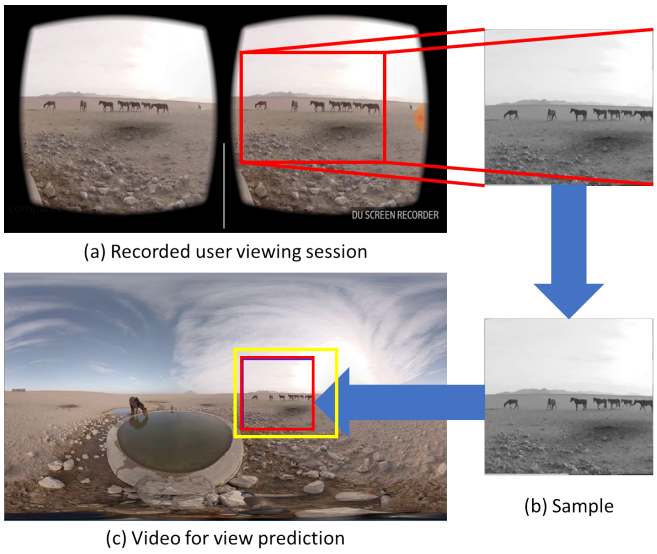


Fig. 5. The evaluation workflow comparing the predicted view with the user's actual view for accuracy evaluation (video courtesy of [20]).

The development environment is Visual Studio 2017 with an external software library OpenCV 3.2 [24], which provides the computer vision functions.

5.1.3 Evaluation Method. We employ a content-based evaluation method for the lab study, in which we record the user's VR viewing session and compare it with the predicted views using image processing techniques. Figure 5 shows our evaluation workflow for the lab user study. We first record the user's VR viewing session that represents the user's actual viewport for each frame, as shown in Figure 5(a). Then, we pick the user's actual viewport (in Figure 5(b)) and locate it in the original video frame by using the correlation convolution algorithm [25], as shown in Figure 5(c). After that, we conduct a viewport matching process to evaluate the accuracy of the viewport prediction. We consider there is a viewport match if the predicted viewport fully covers the user's actual viewport. This is a customized metric based on the intersection over union (IOU) metric considering the specific context of VR video streaming. In VR video streaming, the user's experience will be considered as compromised if the coverage of the user viewport is not 100%. Also, different from the traditional computer vision use cases

where IOU is applied, our predicted viewports are of dynamic sizes based on the error recovery and view update algorithms. Combining the two reasons above, we realize that the original IOU metric does not directly apply to this viewport prediction use case. Therefore, we customize it to a binary "match/not match" metric for each individual frame (i.e., "match" if the coverage of user viewport is 100% and "not match" otherwise). Then, we define the prediction accuracy of the entire video as the percentage of frames that have a "match" result. Finally, we conduct such matching and comparison for the prediction results of all the frames and define the accuracy of prediction as the percentage of the video frames whose viewports are predicted accurately. Note that for the evaluation using public dataset [42], since it provides timestamped user head movement data, we directly use the coordinates to conduct the viewport matching.

5.1.4 Subjects for Evaluation and Comparison. We evaluate our viewport prediction approach by comparing four related methods, including (1) "Velocity", which represents the velocity based method [12, 28] discussed in Section 1; (2) "Tracking", which represents the motion tracking-based method presented in Section 4.1; (3) "Tracking+Recovery", which represents the user feedback-based error recovery method presented in Section 4.2; and (4) "Tracking+Update", which represents the dynamic viewport update method presented in Section 4.3.

5.2 Prediction Accuracy Evaluation Using Lab Dataset

Figure 6 demonstrates a snapshot of the viewport prediction results in 3 out of the 10 test videos from the lab dataset, watched by User *A*, by applying the "Tracking+Update" algorithm presented in Section 4.3. (The snapshots for the rest of the videos are available at the project repository [9]). In each video, the 360-degree frame is divided into 10×10 tiles represented by the grids with thin blue edges. The grids with thick pink edges represent the selected tiles forming the predicted view. We observe that the selected tiles cover all the objects in the user's view, and they only occupy a subset of all the frames, indicating bandwidth savings.

Fig. 6. Demonstration of viewport prediction using 3 test videos from the lab dataset (video courtesy of [20][5][22]).

Furthermore, to quantitatively evaluate the accuracy of the viewport prediction methods, we adopt the evaluation method presented in Section 5.1 with four different methods under various buffer lengths. Figure 7 shows the evaluation results of User *A* watching the Wild Horse video. The prediction accuracy of our Tracking+Update method (with 10×10 tiles and *HIGH*=3 lifetime) demonstrates steady and premium accuracy under all the buffer lengths tested, which is around 88%. The accuracy of the Tracking only method is 18% with little dependency on the buffer length. The accuracy results of the Velocity method and the Tracking+Recovery method are influenced by the buffer length (e.g., the accuracy of the Velocity method drops to below 10% with 4s buffer length). Our observation is that for a longer buffer length, the user movement and the video content would have significant changes in direction, which would compromise the accuracy of the Velocity and Tracking+Recovery approaches. However, our Tracking+Update method demonstrates superior accuracy because it tracks the exact motions of the objects and dynamically update the predicted viewport.

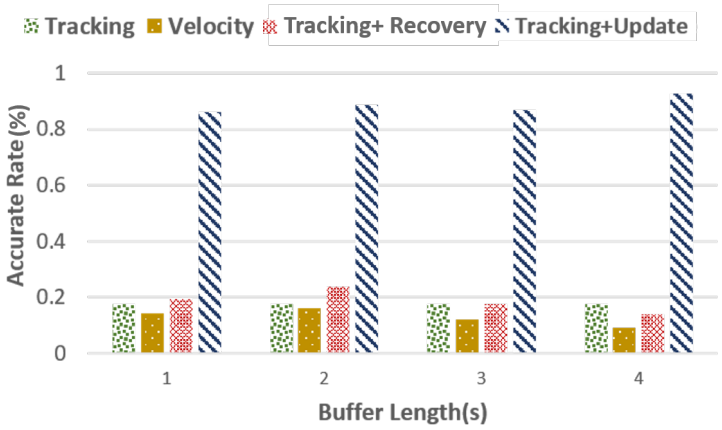


Fig. 7. Accuracy evaluation of the four approaches for User A watching the Wild Horse video.

Furthermore, Tables 3 shows the accuracy evaluation results of the Tracking+Update method when the 4 users are watching the 10 videos. The results confirm the accuracy results in Figure 7 with different user behavior and video types, where we observe that our Tracking+Update method achieved around or above 80% of accuracy with most of the users/videos under test.

Table 3. Prediction accuracy of the Tracking+Update method when 4 lab users watching 10 test videos from the lab dataset (%).

(%)	Airbus	Cat	Video Game	Cow	Giraffe	Girls Dancing	Surveillance	Marine	Lady Singing	Wild Horse
User A	95	83	69	89	83	84	79	60	89	86
User B	92	76	73	83	92	84	71	63	83	79
User C	92	55	77	66	80	84	88	73	88	79
User D	87	75	63	89	79	82	79	66	91	96

5.3 Bandwidth Evaluation Using Lab Dataset

We further evaluate the bandwidth consumption of the Tracking+Update approach, as it requires an enlarged user view and would not achieve the bandwidth savings of a regular view. Figure 8 shows the bandwidth usage rate (i.e., percentage of bandwidth usage compared to the case of no viewport prediction) of the four methods in the Wild Horse video, with 2-sec buffer length, 10×10 tiles, and lifetime 3. We observe that the bandwidth usage of the first three methods is below 20% because of the fixed-size predicted view. The bandwidth is 9.76% when the three approaches make the correct prediction, and it becomes 19.52% if the prediction is wrong. According to its design, the Track+Update approach requires a 100% bandwidth usage at the beginning, and it later decreases to around 30%, indicating a more than 60% bandwidth saving, which is premium considering the remarkable accuracy.

Furthermore, Figure 9 shows the bandwidth usage of the 4 users watching 4 of the 10 videos, when the Tracking+Update method is applied. These 4 videos have diverse complexity in the content, which results in significantly different bandwidth usage. For example, there are videos that contain very few objects and the

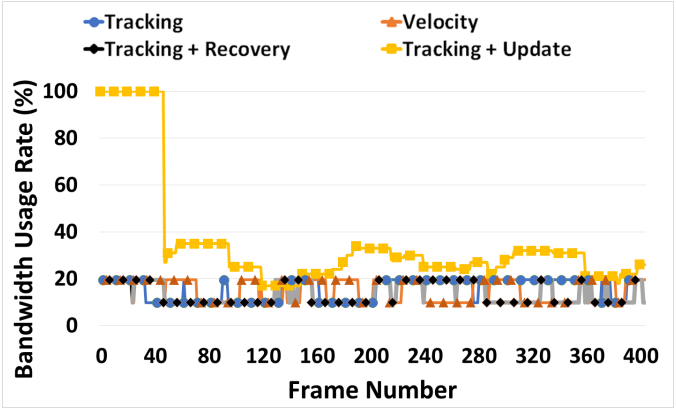


Fig. 8. Bandwidth evaluation of the four methods with 2-sec buffer length, 10×10 tiles, and lifetime 3 (for the Wild Horse video).

bandwidth usage is quite low (e.g., "Lady Singing"), and the bandwidth usage is around 30%. Some videos have complicated stories that make the bandwidth usage vary during the playback (i.e.,the "Surveillance" and "Video Game"). In all the test videos, we observe that our method saves a considerable amount of bandwidth (i.e., 20% to 80%).

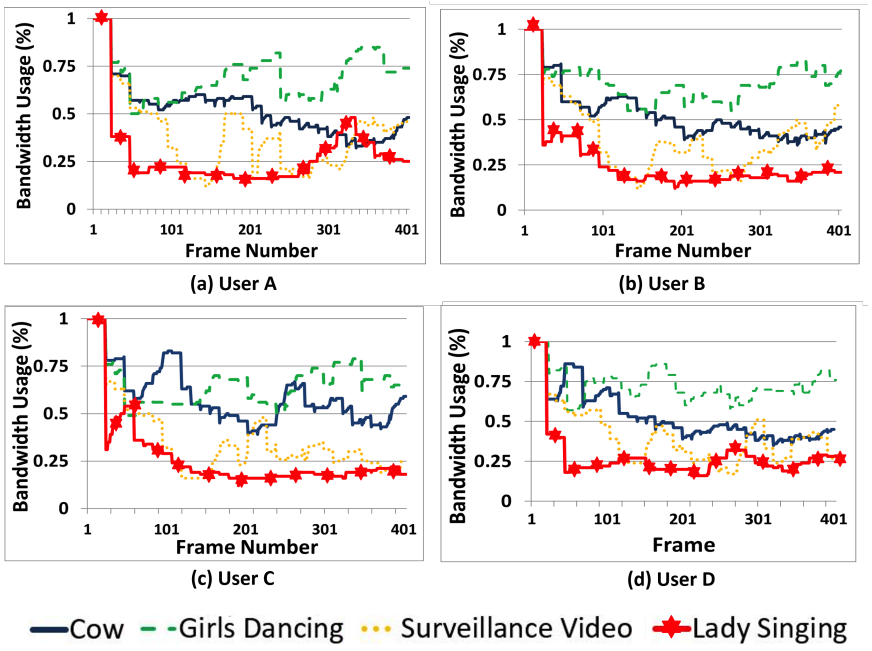
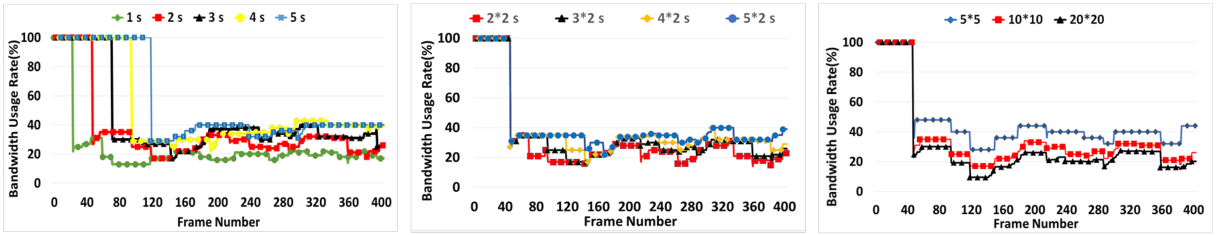


Fig. 9. Bandwidth usage evaluation of 4 Users watching 4 test videos from the lab dataset.

5.4 Performance Analysis of the Tracking+Update Algorithm Using Lab Dataset

The Tracking+Update algorithm involves 3 main parameters that may influence the performance of viewport prediction, namely the buffer length, the lifetime, and the number of tiles. Since for most combinations of these parameters, the accuracy of the algorithm is higher than 80% (as discussed in Section 5.2), we focus on evaluating how these parameters would influence the bandwidth savings using the Wild Horse video with User B from our lab dataset.

5.4.1 Buffer Length. We set the tile number to 10×10 and the lifetime to 3. The bandwidth usage rate of the Tracking+Update Algorithm under different buffer lengths is shown in Figure 10(a). We observe that the larger buffer lengths result in longer duration of higher bandwidth usage in the beginning frames. Considering the accuracy under different buffer length, as shown in Figure 7, using a 2-sec buffer length would be a premium option, which is also a typical buffer length for live video streaming.



(a) Buffer Length (b) Lifetime (c) Number of Tiles

Fig. 10. Bandwidth usage rate for Tracking + Update method under different parameters using the Wild Horse video with User B from the lab dataset.

5.4.2 Lifetime. We set the buffer length as 2s and use 10×10 tiles while changing the lifetime from 2 to 5 buffer lengths. Figure 10(b) shows the real time bandwidth usage results. We observe that the system achieves lower bandwidth usage with smaller lifetime values (i.e., more frequent updates), while the bandwidth drops at the beginning of the video start at the different times depending on the chosen lifetime value.

5.4.3 Number of Tiles. We further evaluate the influence of the tile numbers by fixing the buffer length to 2s and the lifetime value to 3. Figure 10(c) shows the bandwidth results with varying tile numbers of 5×5, 10×10, and 20×20. We observe that a larger tile number saves more bandwidth, as it makes the predicted view more fine-grained to cover the target objects more accurately. In terms of computation time, since our viewport prediction algorithm is just selecting the tiles to match the predicted area, the different tiles sizes/numbers would only make minor difference in the viewport prediction time. From the video encoding/decoding point of view, since each tile is a independent unit for video coding in parallel, larger tile numbers could reduce the encoding/decoding time. That being said, however, one important downside of larger tile numbers is that it could significantly reduces the coding efficiency (i.e., the video compression ratio), due to the potentially higher chance of partitioning and thus coding the continuous pixels separately in different tiles, as pointed out by recent research in the community [43]. In summary, the tile number/size indeed introduces tradeoffs between bandwidth savings/computation time and coding/compression efficiency.

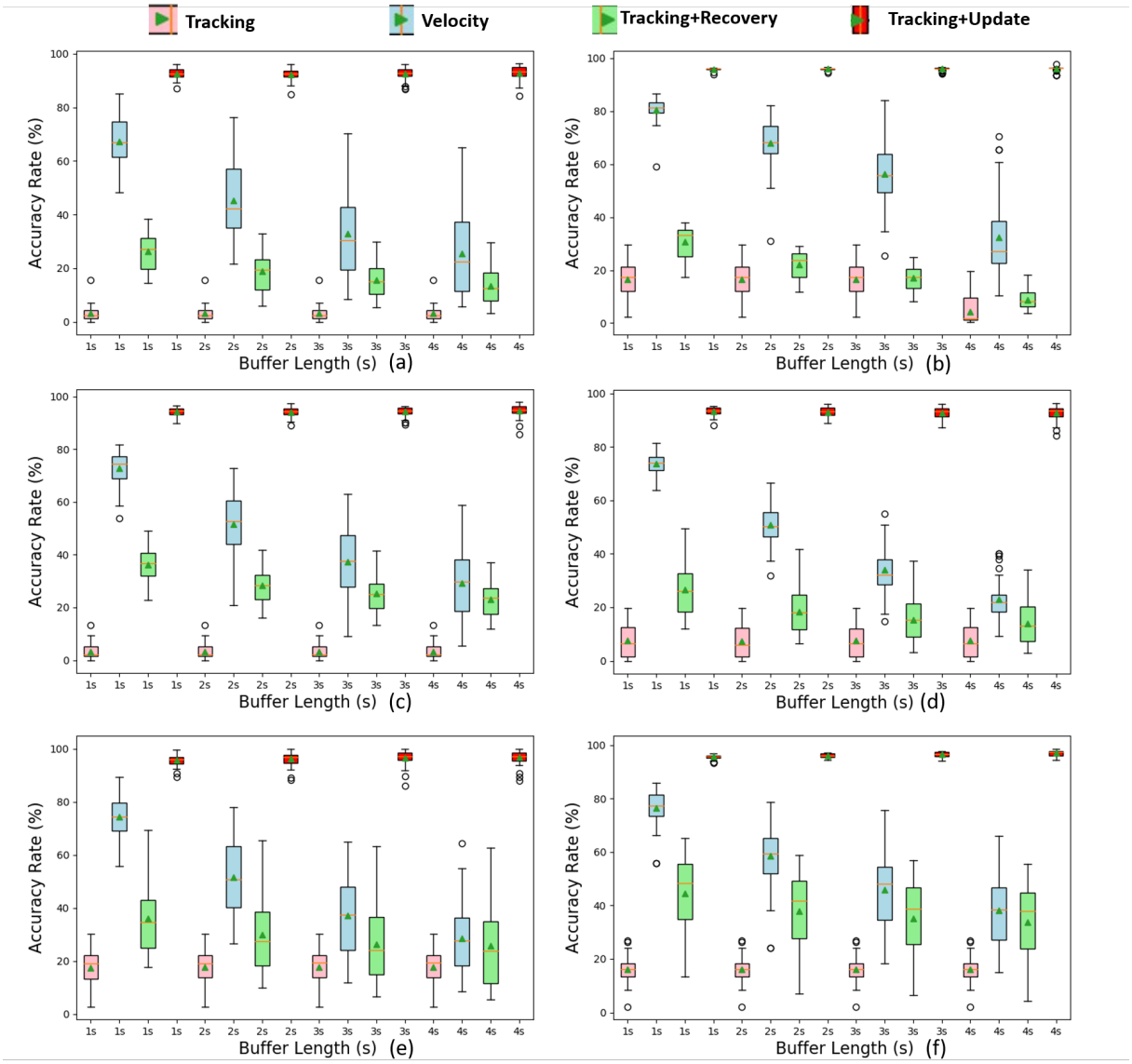


Fig. 11. Accuracy of the 4 viewport prediction methods under 4 buffer lengths over 48 users from the public dataset [42]: (a) Anitta Video, (b) Fighting Video, (c) VoiceToy Video, (d) Female Basketball Video, (e) Reloaded, and (f) Korean Video.

5.5 Accuracy and Bandwidth Evaluation Using Public Dataset

In addition to the lab dataset, we also evaluate our viewport prediction approach using the larger-scale public dataset [42], which involves 48 users watching 6 VR videos. Figure 11 shows the accuracy of the 4 viewport prediction methods for the 6 videos. Each sub-figure contains the results for a particular video, with a boxplot [13]

covering the results of the 48 users under different buffer lengths. We can observe that our Tracking+Updating method obtains the highest prediction accuracy (above 95%) compared to the other three methods in all 6 videos and all 4 buffer length settings. Also, the prediction accuracy in Tracking+Updating is more concentrated among 48 users, indicating more consistent and reliable accuracy over different users. The second highest prediction accuracy is obtained by the Velocity-based method. However, similar with our observations in the lab dataset experiments, the performance of this method is heavily impacted by the buffer length. Also, the Velocity-based method results in a much larger variation of the accuracy among different users compared to the Tracking+Updating method.

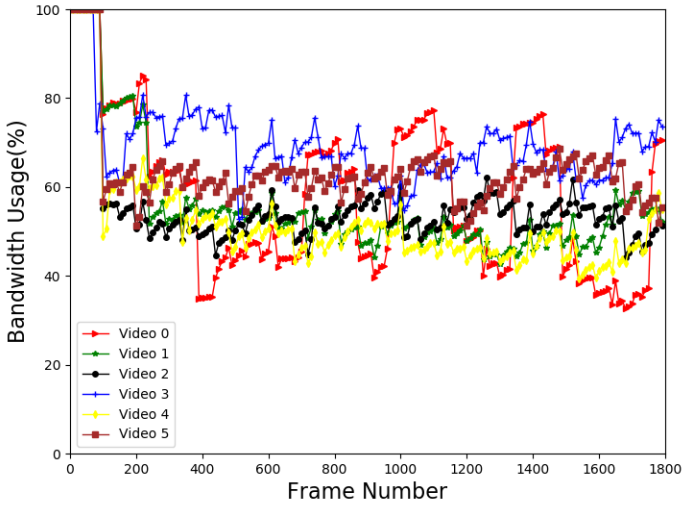


Fig. 12. The bandwidth usage for each of the 6 videos averaged over 48 users from the public dataset [42].

Furthermore, Figure 12 shows the average bandwidth usage of the Tracking+Update method over 48 users watching the 6 videos, and the boxplots [13] in Figure 13 summarizes the detailed statistics for each individual user. We observe from Figure 12 that, among different videos, the bandwidth usages for Video 1, 2, 4 and 5 are concentrated in a short range from 40% to 60%. For video 0 and video 3, the bandwidth usages vary significantly during the whole video session. We analyzed the content of video 0 and video 3 further and found that they involve two cameras each, and the video content keeps alternating between the two camera sources. Every time the video switches to a different camera, the changing background would introduce noise to our algorithm and result in fluctuations in the predicted viewport. In Video 3, the bandwidth varies more because of the higher frequency of changes in lighting. We observe from Figure 13 that Video 3 (i.e., the fighting video) has the lowest variation in bandwidth usage among the 48 users, which is because the interesting view in that video is mostly static and concentrated in a small area.

Table 4 shows the comparison of bitrates for the 6 test videos with and without applying the proposed viewport prediction mechanism (i.e., Tracking+Update), where the reduced bitrates after viewport prediction are shown as the average over all the 48 users. We observe that the original bitrates of the videos are around 5.6 Mbps, which are reduced to around 3.0 Mbps after applying our viewport prediction approach. Note that the resolutions of the test videos are 1280×720, which are down-sampled from 2560×1440 and 3840×2160 from the original

dataset. We intentionally conduct the downsampling to achieve short video processing and viewport prediction time, making it practical for live video system. Then, we can apply the viewport prediction results from the down-sampled videos to the original high-resolution videos, achieving the same percentage of bitrate/bandwidth savings.

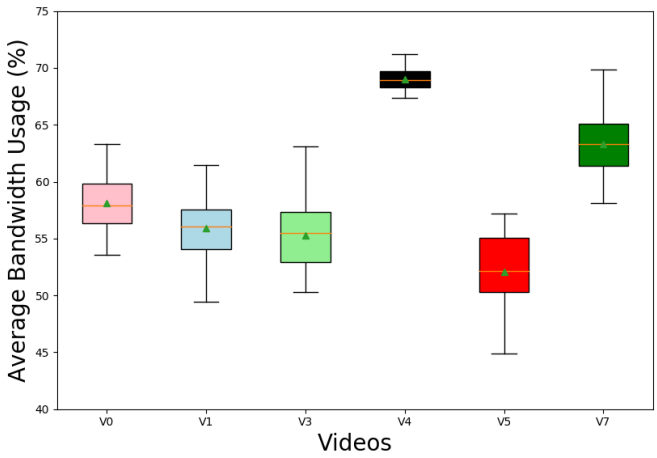


Fig. 13. The statistics (boxplot) of bandwidth usage for 48 users from the public dataset [42] averaged over all the frames in the video.

Table 4. Comparison of bitrates for the 6 test videos from the public dataset [42] with and without applying the proposed viewport prediction mechanism. The reduced bitrates after viewport prediction are shown as the average over all the 48 users.

.

	Korean	VoiceToy	Female Basketball	Fighting	Anitta	Reloaded
Original bitrate (Mbps)	5.683	5.620	5.640	5.654	5.624	5.750
Bitrate with viewport prediction (Mbps)	3.580	3.091	3.891	3.166	3.262	2.992

5.6 Live Latency Overhead

We evaluate the latency overhead of our approach by setting a timer for the processing of each frame using the Wild Horse video with User B from the lab dataset. Figure 14 shows the distribution of the latency overhead caused by the Tracking+Update approach over 404 data points of live latency that we collected from the video session. We observe that most of the values are at around 120 ms. To ensure a low latency live streaming, we can adopt a selective motion tracking method, in which we only process a subset (e.g., 7) of frames per second that introduces a less than 1 second latency overhead. While the viewport prediction does generate non-trivial delays (around 120 ms per frame and less than 1 second total live latency with selective processing), we argue that such latency can be pipelined with the other time consuming video operations under the a few seconds of segment duration. Therefore, the additional delay with viewport prediction will not noticeably increase the total live latency.

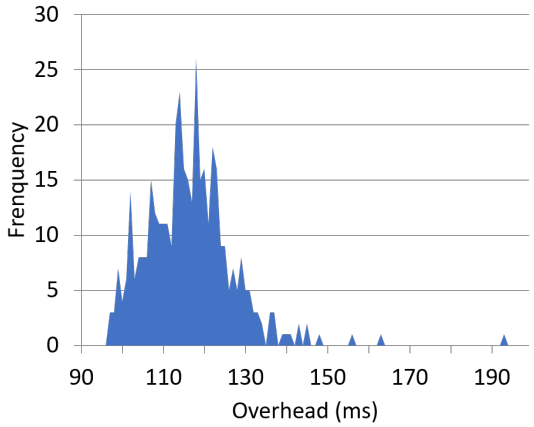


Fig. 14. The latency overhead of the Tracking+Update approach using the Wild Horse video with User B from our lab dataset.

6 LIMITATIONS AND ROBUSTNESS OF THE VIEWPORT PREDICTION MECHANISM

In this section, we discuss the scope and limitations of our viewport prediction scheme, with a focus on its robustness against the potential impacts posed by certain VR video properties, such as video scene change and video distortion. Also, we discuss and summarize the rationale behind the GMM algorithm and the Bayes methods adopted in this work compared to the alternative approaches.

6.1 Scope of Supported Video Types and Saliency Factors

Our intuition of using motion for saliency originated from the sports videos where the viewers typically follow the players or the ball, and we later extended this observation to other types of videos including animal marching, dancing, surveillance, etc. The experiments with both our own user study and the public dataset justified our intuition about motion-based saliency in these types of videos. However, we acknowledge that there exist other types of videos where the saliency may be determined by other factors like contrast and object appearance, where our motion-based approach may not apply.

Also, since our viewport prediction method relies on the motion detection mechanism adopted, the accuracy and robustness of the prediction may be subject to the complexity of the video scenes and, in particular, the motion status of the background in the video. For instance, a sudden video scene change initiated by the camera could result in an artificial motion that is not relevant to the viewer's viewport of interest. Also, if the background of the target video is constantly in motion, it may mislead the motion detection algorithm to report motion even if the objects in the foreground stand still. Both cases would cause false positives in the motion detection results and thus in turn impact the accuracy or robustness of the viewport prediction. We acknowledge that our current method can only work with VR videos shot by static 360-degree cameras.

6.2 Distortions in VR Video

Different from traditional 2D videos, the VR videos must be projected from a 360-degree sphere to a 2D surface before it can be rendered on the display. There are several projection methods that have been adopted [40][31], but all of them would inevitably cause distortions in the video due to geometric difference between 3D and 2D planes. While the distortion may cause significant problems in many video processing schemes that reply on the content of the video, we argue that our content-based viewport prediction mechanism is immune to such

impacts. It is because the distortions would only change the sizes of the target objects but not the motion and, therefore, the GMM algorithm would function equally well on the distorted portion of the video. Also, the same applies to the feature-based motion tracking algorithm since the features are the representative structures that would not be impacted by scaling, rotation, or distortion.

6.3 The GMM Algorithm

In addition to GMM, we have also considered 5 other motion detection methods presented and compared in [44], namely TPC, LBP, MRF, VIBE, and PBAS. As reported by [44], while 3 out of the 6 methods (GMM, TPC, and LBP) achieved equally good accuracy in motion detection, the GMM has the best frame rate performance [44]. We eventually employ GMM in this work considering the application is live VR streaming that requires real-time performance.

6.4 The Bayes Method

Compared to other related methods, such as hidden markov chain (HMM) and Recurrent Neural Network (RNN), the advantage of the Bayes method is that it requires simpler computation and thus helps achieve a light-weight viewport update algorithm suitable for our live streaming scenario. However, one potential downside of the Bayes method is that it cannot deal with a large number of inputs in a longer sequence, in which case methods like RNN would be a better fit. In the VR video streaming application, since the inputs are of moderate size which consist of only the user features, the background, and user head movement data, we chose to use the Bayes method for the consideration of computation overhead. A more detailed quantitative comparison between the Bayes and HMM methods can be found in the open source project depository of this work [9].

7 CONCLUSION

We for the first time developed a hybrid user and video content-based viewport prediction mechanism for reducing the bandwidth consumption in live mobile VR streaming. To achieve an accurate viewport prediction, we develop a motion tracking algorithm that predicts the user's viewport by identifying representative moving objects in the 360-degree frames. Furthermore, we develop two enhancement methods, namely error recovery and view updates, to recover from and prevent the prediction errors, respectively. Our evaluation results with 4 users and 10 videos from our own user study, along with 48 users and 6 videos from a public VR user dataset prove the premium accuracy and bandwidth savings of our approach compared to the existing velocity-based approach. The full sets of evaluation results, test videos, user traces, demonstrations, and source code of this work can be found in our open source project depository [9].

ACKNOWLEDGMENTS

This work was supported in part by the National Science Foundation under Awards CNS-1750867, CNS-1912593 and in part by the gift donation from Adobe Research.

REFERENCES

[1] AdventureAlly. 2016. 5 Cute Cats VR (360-degree video). Video. Retrieved April 29, 2019 from https://www.youtube.com/watch?v=pHwvVaiw5a8

[2] Edward C. Baig. 2016. NBC's Rio Olympics are in VR, but on a delay. Retrieved April 29, 2019 from https://www.usatoday.com/story/tech/columnist/baig/2016/08/05/nbcs-rio-olympics-vr-but-delay/88294986/

[3] Yanan Bao, Huasen Wu, Tianxiao Zhang, Albara Ah Ramli, and Xin Liu. 2016. Shooting a moving target: Motion-prediction-based transmission for 360-degree videos. In *IEEE International Conference on Big Data*. 1161–1170.

[4] Bjőrk. 2015. Bjőrk: Stonemilker (360-degree Virtual Reality). Video. Retrieved April 29, 2019 from https://www.youtube.com/watch?v=gQEyezu7G20

[5] Blick. 2015. 360-degree Cockpit View. Video. Retrieved April 29, 2019 from https://www.youtube.com/watch?v=HEEIzZ7UjRg&t=353s

[6] R. Brandenburg, R. van Koenen, and D. Sztykman. 2017. CDN Optimization for VR Streaming. In *International Broadcasting Convention*.
[7] 360 Thrill Daxon. 2015. 360-degree Getting Licked by a Cow in Ireland 4K. Video. Retrieved April 29, 2019 from https://www.youtube.com/watch?v=Q_BavaspcFc
[8] Ching-Ling Fan, Jean Lee, Wen-Chih Lo, Chun-Ying Huang, Kuan-Ta Chen, and Cheng-Hsin Hsu. 2017. Fixation Prediction for 360-Degree Video Streaming in Head-Mounted Virtual Reality. In *ACM Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)*. 67–72.
[9] Xianglong Feng, Viswanathan Swaminathan, and Sheng Wei. 2018. Repository for Project LiveMotion. Retrieved April 29, 2019 from https://github.com/hwsel/LiveMotion
[10] Google. 2019. Google Daydream. Retrieved April 24, 2019 from https://vr.google.com/daydream/
[11] Will Greenwald. 2019. The Best VR (Virtual Reality) Headsets of 2019. Retrieved April 28, 2019 from https://www.pcmag.com/article/342537/the-best-virtual-reality-vr-headsets
[12] Mohammad Hosseini and Viswanathan Swaminathan. 2016. Adaptive 360 VR Video Streaming: Divide and Conquer. In *2016 IEEE International Symposium on Multimedia (ISM)*. 107–110.
[13] J. D. Hunter. 2007. Matplotlib: A 2D graphics environment. *Computing In Science & Engineering* 9, 3 (2007), 90–95. https://matplotlib.org/api/_as_gen/matplotlib.pyplot.boxplot.html
[14] ISO/IEC. 2015. Information technology – High efficiency coding and media delivery in heterogeneous environments – Part 2: High efficiency video coding, ISO/IEC 23008-2:2015. Retrieved April 24, 2019 from https://www.iso.org/standard/67660.html
[15] Evgeny Kuzyakov, Shannon Chen, and Renbin Peng. 2017. Enhancing high-resolution 360 streaming with view prediction. Retrieved April 24, 2019 from https://code.facebook.com/posts/118926451990297/enhancing-high-resolution-360-streaming-with-view-prediction/
[16] Xing Liu, Qingyang Xiao, Vijay Gopalakrishnan, Bo Han, Feng Qian, and Matteo Varvello. 2017. 360-Degree Innovations for Panoramic Video Streaming. In *ACM Workshop on Hot Topics in Networks (HotNets)*. 50–56.
[17] Aditya Mavlankar and Bernd Girod. 2009. Pre-fetching Based on Video Analysis for Interactive Region-of-interest Streaming of Soccer Sequences. In *International Conference on Image Processing (ICIP)*. 3025–3028.
[18] Aditya Mavlankar and Bernd Girod. 2010. *Video Streaming with Interactive Pan/Tilt/Zoom*. Springer Berlin Heidelberg, 431–455.
[19] Neil McKay. 2018. Streaming VR video to PlayStation VR. Retrieved April 24, 2019 from https://community.akamai.com/community/media-delivery/blog/2017/04/24/streaming-vr-video-to-playstation-vr
[20] Photos of Africa VR Safari. 2016. 360 VR Video of Namib Desert Wild Horses at Klein-Aus Vista. Video. Retrieved April 29, 2019 from https://www.youtube.com/watch?v=8kqde9BlRO8&t=2s
[21] Photos of Africa VR Safari. 2017. 360 VR Video of Giraffe Manor, The Safari Collection - Kenya. Video. Retrieved April 29, 2019 from https://www.youtube.com/watch?v=2ANabu-EuHg
[22] Clash of Clans. 2015. Clash of Clans 360: Experience a Virtual Reality Raid. Video. Retrieved April 29, 2019 from https://www.youtube.com/watch?v=wczdECcwRw0
[23] OpenCV. 2017. Background subtraction. Retrieved April 29, 2019 from https://docs.opencv.org/3.2.0/d1/dc5/tutorial_background_subtraction.html
[24] OpenCV. 2019. The OpenCV library. Retrieved April 23, 2019 from https://opencv.org/
[25] OpenCV. 2019. Template Matching. Retrieved April 29, 2019 from https://docs.opencv.org/3.2.0/de/da9/tutorial_template_matching.html
[26] George Papagiannakis, Gurminder Singh, and Nadia Magnenat-Thalmann. 2008. A Survey of Mobile and Wireless Technologies for Augmented Reality Systems. *Comput. Animat. Virtual Worlds* 19, 1 (2008), 3–22.
[27] Stefano Petrangeli, Viswanathan Swaminathan, Mohammad Hosseini, and Filip De Turck. 2017. An HTTP/2-Based Adaptive Streaming Framework for 360 Virtual Reality Videos. In *ACM Multimedia Conference (MM)*. 306–314.
[28] Stefano Petrangeli, Viswanathan Swaminathan, Mohammad Hosseini, and Filip De Turck. 2017. Improving Virtual Reality Streaming Using HTTP/2. In *ACM Multimedia Systems Conference (MMSys)*. 225–228.
[29] Feng Qian, Bo Han, Qingyang Xiao, and Vijay Gopalakrishnan. 2018. Flare: Practical Viewport-Adaptive 360-Degree Video Streaming for Mobile Devices. In *International Conference on Mobile Computing and Networking (MobiCom)*. 99–114.
[30] Feng Qian, Lusheng Ji, Bo Han, and Vijay Gopalakrishnan. 2016. Optimizing 360 Video Delivery Over Cellular Networks. In *ACM Workshop on All Things Cellular: Operations, Applications and Challenges*. 1–6.
[31] David Salomon. 2007. *Transformations and projections in computer graphics*. Springer Science & Business Media.
[32] Samsung. 2019. Samsung GearVR. Retrieved April 29, 2019 from http://www.samsung.com/us/mobile/virtual-reality/gear-vr/gear-vr-with-controller-sm-r324nzaaxar/
[33] Jianbo Shi et al. 1994. Good features to track. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*. 593–600.
[34] SoKrispyMedia. 2015. Security Camera. Video. Retrieved April 29, 2019 from https://www.youtube.com/watch?v=Lipit2-WHIU
[35] Todd Spangler. 2017. MLB to Launch VR-Enhanced Broadcasts, but Live Games Won't Be in 360 Video. Retrieved April 29, 2019 from https://variety.com/2017/digital/news/mlb-virtual-reality-vr-enhanced-broadcasts-1202434545/

[36] Afshin TaghaviNasrabadi, Anahita Mahzari, Joseph D. Beshay, and Ravi Prakash. 2017. Adaptive 360-degree video streaming using layered video coding. In *IEEE Virtual Reality (VR)*. 347–348.
[37] 360 Videos Technology. 2015. Girl Dancing. Video. Retrieved April 29, 2019 from https://www.youtube.com/watch?v=UhJvAXTMxvg
[38] Viddler. 2014. Live Streaming vs. Video On-demand (VOD). Retrieved April 24, 2019 from https://www.viddler.com/blog/cracking-the-code-on-video-live-streaming-vs-video-on-demand/
[39] 360 video AirPano. 2017. 360-degree Diving with Sharks. Video. Retrieved April 29, 2019 from https://www.youtube.com/watch?v=VNChunf5RKQ&t=35s
[40] Eric Weisstein. 2019. Equirectangular Projection. Retrieved April 29, 2019 from http://mathworld.wolfram.com/EquirectangularProjection.html
[41] Cedric Westphal. 2017. Challenges in networking to support augmented reality and virtual reality. *IEEE ICNC* (2017).
[42] Chenglei Wu, Zhihao Tan, Zhi Wang, and Shiqiang Yang. 2017. A Dataset for Exploring User Behaviors in VR Spherical Video Streaming. In *Proceedings of the 8th ACM on Multimedia Systems Conference*. ACM, 193–198.
[43] Mengbai Xiao, Chao Zhou, Yao Liu, and Songqing Chen. 2017. OpTile: Toward Optimal Tiling in 360-degree Video Streaming. In *ACM Multimedia Conference (MM)*. 708–716.
[44] Chia-Hung Yeh, Chih-Yang Lin, Kahlil Muchtar, Hsiang-Erh Lai, and Ming-Ting Sun. 2017. Three-Pronged Compensation and Hysteresis Thresholding for Moving Object Detection in Real-Time Video Surveillance. *IEEE Transactions on Industrial Electronics* 64, 6 (2017), 4945–4955.
[45] Matt Yu, Haricharan Lakshman, and Bernd Girod. 2015. Content Adaptive Representations of Omnidirectional Videos for Cinematic Virtual Reality. In *International Workshop on Immersive Media Experiences*. 1–6.
[46] Alireza Zare, Alireza Aminlou, Miska M. Hannuksela, and Moncef Gabbouj. 2016. HEVC-compliant Tile-based Streaming of Panoramic Video for Virtual Reality Applications. In *ACM Multimedia Conference (MM)*. 601–605.
[47] Chao Zhou, Mengbai Xiao, and Yao Liu. 2018. ClusTile: Toward Minimizing Bandwidth in 360-degree Video Streaming. In *IEEE Conference on Computer Communications (INFOCOM)*. 962–970.
[48] Zoran Zivkovic. 2004. Improved adaptive Gaussian mixture model for background subtraction. In *IEEE International Conference on Pattern Recognition (ICPR)*, Vol. 2. 28–31.

Received August 2018; revised February 2019; accepted April 2019.