Beyond the Touch: Interaction-Aware Mobile Gamecasting with Gazing Pattern Prediction

Cong Zhang*, Qiyun He*, Jiangchuan Liu*, Zhi Wang†
*School of Computing Science, Simon Fraser University, Canada
†Graduate School at Shenzhen, Tsinghua University, China
Email: congz@cs.sfu.ca, qiyunh@cs.sfu.ca, jcliu@cs.sfu.ca, wangzhi@sz.tsinghua.edu.cn

***Abstract*—Recent years have witnessed an explosion of gamecasting applications in the market, in which game players (or *gamers* in short) broadcast their game scenes in real-time. Such pioneer applications as YouTube Gaming, Twitch, and Mobcrush have attracted a massive number of online broadcasters, and each of them can attract hundreds or thousands of fellow viewers. The growing number however has created significant challenges to the network and end-devices, particularly considering bandwidth- and battery-limited smartphones or tablets are becoming dominating for both gamers and viewers.**

Yet the unique touch operations of the mobile interface offer opportunities, too. In this paper, our crowdsourced measurement reveals that strong associations exist between the gamers' touch interactions and the viewers' gazing patterns. Motivated by this, we present a novel interaction-aware optimization framework to improve the energy utilization and stream quality for *mobile gamecasting (MGC)*. Our framework incorporates a touch-assisted prediction module to extract association rules for gazing pattern prediction and a tile-based optimization module to utilize energy on mobile devices efficiently. Trace-driven simulations illustrate the effectiveness of our framework in terms of energy consumption and streaming quality. Our user study experiments also demonstrate much improved (3%-13%) quality satisfaction than the state-of-the-art solution with similar network resources.

I. Introduction

With the widespread penetration of high-performance mobile devices and communication networks, the *mobile gamecasting* (MGC) service is getting increasingly popular among gamers. Such MGC applications as YouTube Gaming[1], Twitch[2], and Mobcrush[3], have shifted the paradigm of multimedia content generation and dissemination, which distributes mobile game sessions from the gamer's (i.e., broadcaster's) personal mobile device to a large population of viewers. Recent research from Google [1] suggests that about a third of U.S. mobile gamers are defined as "avid gamers", who spend more than nine hours a week on average playing mobile games on smartphones; And these "avid gamers" access mobile games and YouTube at similar times throughout the day. Another news from Twitch also shows that the mobile viewers in all gamecasting channels account for 35% of monthly viewership [2]. Promoted by the prosperous game markets and MGC applications, the growing number of gamers and viewers in turn poses significant challenges to the existing live video broadcasting platforms, particularly with mobile broadcasters and viewers.

Due to the intrinsic sensitivity to latency, MGC applications cannot simply rely on existing solutions for quality and energy optimization [3], e.g., ON-OFF transmission [4] or local cache scheduling [5]. Some research efforts [6], [7] have focused on optimizing the stream uploading stage, which however may affect the viewers' Quality-of-Experience (QoE) if the stream quality fluctuates greatly. Recently, the rapid development of eye-tracking researches makes foveated-aware optimization of viewers' watching experience possible, which has seen use cases for content distribution and live streaming [8] [9]. They are not targeting MGC applications, and typically need eye-tracking peripherals to collect viewers' gazing data in real-time, which in turn produces extra energy consumption on mobile devices.

Thanks to the unique features from mobile games, our crowdsourced measurements show that strong correlations exist between the gamers' interactions on the touch screen and the viewers' eye-gazing patterns in the MGC context. Motivated by this observation, we propose a novel interaction-aware optimization framework that guides mobile gamecasting in advance, even before the source encoding step. The target and key challenges towards designing the framework lie in: (1) understand the characteristics of the gamers' interactions and the viewers' gazing patterns with dynamic game strategies and eye movements; (2) online predict without the assistant from eye-gazing peripherals considering the energy saving on mobile devices; (3) design an optimization strategy to improve the energy efficiency and adjust the streaming quality using the predicted gazing patterns.

To this end, we build association rules between the gamers' interactions and viewers' gazing patterns from the trace-data, and classify the users' behaviors into distinct groups, including single-touch, press-drag, pan, and zoom for touch interactions, and area-fixation, smooth-pursuit, and scene-saccade for gaze patterns, corresponding to the steady, slow, and fast movements of human eyes. Our framework then incorporates a touch-assisted prediction (TAP) module and a tile-based optimization (TBO) module. The former achieves offline training and online prediction through building the association rules [10], and the latter improves the energy efficiency in mobile devices using a tile-based quality selection with bandwidth and QoE constraints. Trace-based simulations

[1]https://gaming.youtube.com/
[2]https://www.twitch.tv/
[3]https://mobcrush.com/

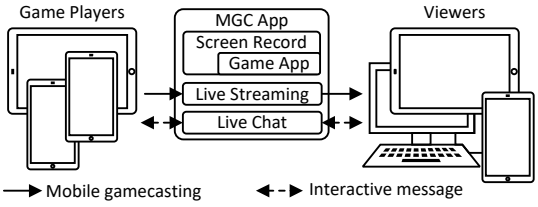


Fig. 1: A generic architecture of MGC applications

and user study demonstrate that our framework achieves noticeable better QoE under similar network constraints.

The remainder of this paper is organized as follows. Section II briefly introduces the MGC background and related work. Section III presents our observations in MGC applications and motivations in this paper. We propose the framework design of the interaction-aware optimization in Section IV. Section V and VI describe the collection and classification of gamers' touch interactions and viewers' gazing patterns, respectively. Section VII introduces the touch-assisted prediction using data traces collected from two types of users. Section VIII presents the tile-based optimization and solution. Section IX evaluates the performance of our framework via both trace-driven simulations and user study. We present further discussion after concluding the paper in Section X.

II. Background and Related Work

Recent years have witnessed an explosion of gamecasting applications, in which gamers broadcast their game scenes in real-time [11], [12]. Such pioneer applications as YouTube Gaming, Twitch, and Mobcrush have attracted a massive number of online broadcasters, and each of them can attract hundreds or thousands of fellow viewers. In this section, we first briefly introduce a generic architecture of MGC applications, as shown in Figure 1. A typical MGC application maintains the following services: (1) *live streaming service*, which not only fulfills the encoding, ingesting, transcoding, and distribution of live streams, but also implements the screen recording functionality on mobile devices; and (2) *live chat service*, which exchanges users' messages through the IRC (Internet Relay Chat) or proprietary chatting protocols. A representative scenario is illustrated as follows (see Figure 2): a gamer first launches a mobile game using an MGC application, e.g., YouTube Gaming, on the smartphone. The top bar on the screen provides the controllers for mobile cameras, microphone, screen recording, and other configurations. After clicking the screen recording button, the MGC application encodes the recorded game scene as a live stream and transmits it to the ingesting server. After multi-version transcoding, the segments of this live stream are delivered to heterogeneous viewers. During the gamecasting, the gamer and the viewers can closely interact by lively discussing the game strategies with the chat service.

The MGC applications have attracted a massive number of geo-distributed gamers and viewers, who are increasingly using mobile devices, e.g., smartphones, tablets, and phablets, to broadcast and watch mobile gamecasting. These high-performance mobile devices suffer from energy constraints with the built-in batteries, but also enjoy opportunities with the novel operation interfaces, in particular, the touch screens.

A. Touch-assisted Applications

Owing to the fast development of mobile devices, plenty works have devoted to analyzing touch behaviors for specific applications, e.g., recognizing users [13], [14] and unlocking smartphone [15]. A recent work [16] explored the instant video clip scheduling problem based on the unique scrolling behaviors on mobile devices. Under the bandwidth and energy constraints, they developed an effective algorithm to solve corresponding sub-problems and achieved a significant improvement in viewing experience. Our work is motivated by these works; yet we focus on the gamers' touch behaviors to predict the viewers' gazing patterns for MGC applications.

B. Region-of-Interest Optimization

There have been significant studies on region-based optimization in streaming applications [17], [18]. Most of them target on the Region-of-Interest (ROI) detection, which is used in visual attention models under the network bandwidth and video quality constraints. Through assigning suitable bits setting for ROI and non-ROI regions, these models achieve better visual quality with limited bandwidth. The ROI prediction in these works however depends on a saliency model, which analyzes neighbor frames of live streams, but cannot completely capture the patterns of human gaze. To overcome these challenges, recent works have designed content transmission based on the user's gaze information for live streaming services [8] and cloud gaming systems [9]. These works need extra supports from eye-tracking devices, e.g., web-camera, to capture gazing data during the whole process. Our work differs from them in that we predict viewers' eye-gazing patterns based on the cross-domain inputs, i.e., gamers' interactions in mobile gamecasting applications.

III. Motivation

In this paper, we optimize the MGC applications from a new perspective. That is, through analyzing gamers' interactions with the touch screen, we predict viewers' eye-gazing patterns towards energy-efficient streaming. We seek to first answer the following question: *How a gamer's interactions affect the viewer's gazing patterns (including the focusing regions and movement)?* To investigate the associations between them, we capture the gamers' game interactions and viewers' gazing data through our testbed, which consists of a smartphone, an eye-tracking device, and a desktop PC. We connect the smartphone with the desktop PC to record gamers' touch events and deploy the eye-tracking device to capture viewers' gazing points. The details about the testbed configurations will be introduced in Section V and Section VI, respectively. Here, we first highlight our findings.

Our testbed experiments show that the gamers' touch interactions change mobile game scenes and objects, which in turn pilot the viewers' gazing patterns. Figure 3 gives two

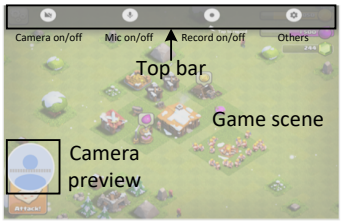


Fig. 2: MGC interface

(a) Example A

(b) Example B

(c) Gazing pattern comparison

Fig. 3: The motivations in our study

examples to illustrate their associations using the viewers' gazing heatmap. In Figure 3a, a gamer designs one attacking path to deploy soldiers from region #1 to #7. Consequently, the viewers' gazing points also follow this path. This example implies that the viewing regions may correspond with the touch interaction regions but have a temporal delay. In Figure 3b, the gamer touches the button in region #1 and activates the system setting options. Afterwards, the viewer focuses on reading the information of each button. This example shows that the viewing regions do not always have the spatial correlation with the touch regions, but their associations still can be found through analyzing the gamers' touch interactions and viewers' gazing patterns together.

Yet there is a new question: if a gamer considers the strategies through investigating the game scenes first, and then decides the touch interaction in the next step, *why we cannot directly rely on the gamer's gazing data for the viewer's gazing prediction?* The first reason is that such a strategy needs to capture the gamers' eye movement in real-time, which needs either plugin supplements (e.g., mobile camera) with higher energy consumption or expensive eye-tracking glasses. Another reason is that differences exist between gamers and viewers. To illustrate the second reason clearly, we use an example[4] to compare the gazing differences between gamers and viewers without gazing heatmap points in Figure 3c. In this figure, we plot the circle regions to exhibit the gazing movement of a gamer and a viewer, respectively. We can find that the gamer proactively focuses on lots of areas to determine the next action (i.e., closing Town Hall information board). The gazing sequence is: Storage Capacity in Town Hall (region #1), Current Storage values (region #3), and then Close icon (region #4). On the other hand, the viewer pays more attention to the information board in region #2, and then reads the introduction of Town Hall in region #5. Since the gamers' observations determine their game strategies and touch interactions, they tend to observe lots of details from the current game scene, which means that they have more complex and unpredictable gazing patterns comparing with the viewers. From this example, we also see the association sequence between the gamers' and viewers' behaviors: gamers' gazing behaviors $\rightarrow$ gamers' game strategies $\rightarrow$ gamers' touch interactions $\rightarrow$ viewers' gazing patterns.

[4]Example link: https://youtu.be/EP2v9m9d15E

IV. Interaction-Aware Design

Motivated by these observations, we design an interaction-aware optimization framework for MGC, as shown in Figure 4a. Compared with the original MGC framework, our design incorporates two new modules: Touch-Assisted Prediction (TAP) and Tile-Based Optimization (TBO). The TAP module predicts the viewer's gazing patterns through ingesting touch interactions from gamers and relays the results to the TBO module, which then accordingly optimizes the energy and bandwidth consumption.

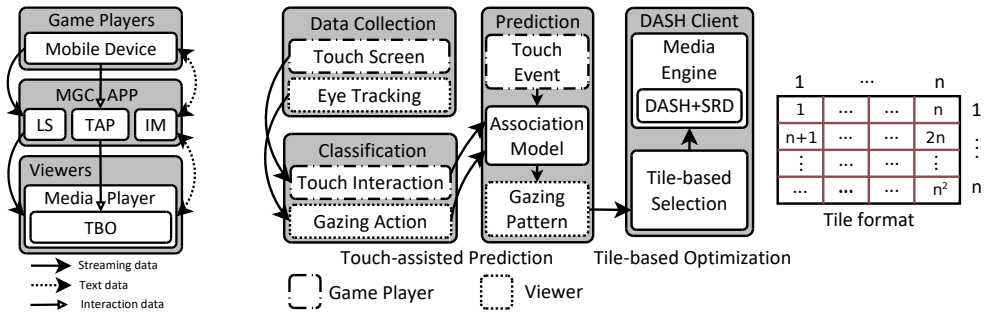
A. Touch-assisted Prediction Design

As shown in the left part of Figure 4b, the touch-assisted prediction involves three steps: Data Collection, Data Classification, and Gazing Prediction. We highlight their design concepts here and present the details of each step in the following sections.

- Data Collection: We first collect the raw touch events from the gamers' mobile devices and the viewers' gazing points based on an eye-tracking device. All the data will be formatted and processed towards the next step;
- Data Classification: According to game-specific rules, we classify the touch events and gazing points into predefined groups. The gamers' touch interactions include single-touch, press-drag, pan, and zoom. The viewer's gazing patterns consist of area-fixation, scene-saccade, and smooth-pursuit;
- Gazing Prediction: The main part of this step is the association model, which is derived from an association rules learning. The prediction module receives gamers' touch interactions and obtains the corresponding viewers' gazing patterns during the mobile gamecasting.

B. Tile-based Optimization Design

In the tile-based optimization module, as shown in the right part of Figure 4b, our framework is practical with Spatial Representation Description (SRD) feature in recent MPEG-DASH (Dynamic Adaptive Streaming HTTP) amendment [19]. SRD works well to stream a part of spatial subparts (i.e., tiles) of a video to viewers' devices. We assume that viewers' mobile devices parse mobile gamecasting using a media engine with DASH(SRD). A tile-based optimization algorithm is then designed to adjust the quality of each tile according to the predicted viewers' gazing patterns. As such, on the viewer's device, this module can adaptively determine



(a) Generic framework (b) Design details in our framework

Fig. 4: Interaction-aware optimization framework in MGC

[2213.341595]	EV_ABS	ABS_MT_TRACKING_ID	00000024
[2213.341638]	EV_ABS	ABS_MT_POSITION_X	00000201
[2213.341655]	EV_ABS	ABS_MT_POSITION_Y	0000040b
[2213.341671]	EV_ABS	ABS_MT_TOUCH_MAJOR	00000008
[2213.341683]	EV_ABS	ABS_MT_TOUCH_MINOR	00000007
[2213.341750]	EV_SYN	SYN_REPORT	00000000
[2213.415273]	EV_ABS	ABS_MT_TRACKING_ID	ffffffff
Timestamp	Event Type	Multi-Touch Event	Value

Fig. 5: Event sample

the quality of each tile in a certain segment, jointly considering the viewers' QoE and bandwidth constraints. Each segment has $n - by - n$ tiles (the default n is set to be 5 in our study). In Section VIII, we formulate this problem mathematically and solve it heuristically.

V. Understanding Game Touch Interactions

Because the gamers in MGC applications control and manage the stream sources, it is important to understand the characteristics of their touch interactions. In this section, we investigate gamers' touch interactions based on real-world data traces and classify game-specific touch behaviors.

A. Touch Data Collection

To collect user's touch interactions, we install the Android Debug Bridge (ADB)[5] on a desktop PC (DELL Optiplex 7010) that connects to a mobile phone (Samsung Galaxy S5 with Android 6.0.1). After running the following command:

```
adb shell getevent -lt /dev/input/event2
```

the touch screen events of S5 will be printed with a timestamp on the command terminal. The parameter `/dev/input/event2` (corresponding to the touch screen of S5) filters out the events triggered by other sensors, such as the proximity sensor, the light sensor, etc.

Figure 5 shows one event sample, where each line has four fields: timestamp, event type, multi-touch event, and value. This sample represents a single-touch interaction, which means a user quickly touches the screen center once. The dynamics of *touch position* (ABS_MT_POSITION_X/Y) and *touch area* (ABS_MT_TOUCH_MAJOR/MINOR) are recorded in a trace file. We write a Python script to extract the event properties of each tracking record from the original data trace. The formatted tracking record is a five-attribute tuple: {tracking_ID, start_timestamp, position_array, area_dynamics, duration}.

[5]ADB is a versatile command line tool that allows users communicate with connected Android devices.

TABLE I: The statistics of touch data

Game ID	Game Name	# of Events	# of Interactions
G1	Clash of Clans	65940	684
G2	HearthStone	33422	250
G3	Clash Royale	37151	421

To understand the characteristics of distinct gamers, we recruited ten volunteers who are familiar with popular mobile games. Each of the volunteers individually plays a game on S5 for two minutes. To explore the impact of game genre, we select three popular games: G1-Clash of Clans[6], G2-HearthStone[7], G3-Clash Royale[8] and capture the screens during game playing. Table I presents the details of our data traces. We find that the number of interactions is mostly determined by the gamers' preferences. There is no strict proportion between the number of events and the number of interactions. Therefore, we further investigate which interaction is used during the game playing.

B. Interaction Classification

According to the official description of touch actions in Android Developers[9], we define the following four game-specific interactions: *Single-touch* (ST), *Press-drag (PD)*, *Pan* (PA), and *Zoom* (ZM).

- Single-touch (ST): Users press the touch screen once and release quickly, e.g., pressing a button or activating an object in a game;
- Press-drag (PD): This is a general touch action in mobile games. For example, gamers move a card onto a target area or deploy attacking path using soldiers;
- Pan (PA): This action is similar to scroll screen; but it is an omnidirectional one-finger gesture in mobile games. Pan can be used to expand the field of view when the game scene is beyond the screen size;
- Zoom (ZM): Zoom is a key multi-touch action. Gamer's two fingers are used to scale up/down the game view. One typical application is to increase the investigating details before attacking enemy camps.

[6]https://clashofclans.com/
[7]http://us.battle.net/hearthstone/
[8]https://clashroyale.com/
[9]https://material.google.com/patterns/gestures.html

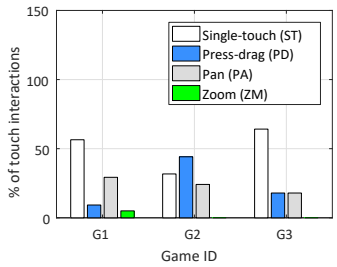


Fig. 6: The classification of touch interactions

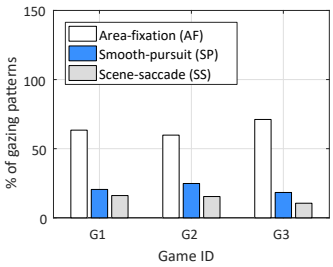


Fig. 7: The classification of gazing patterns

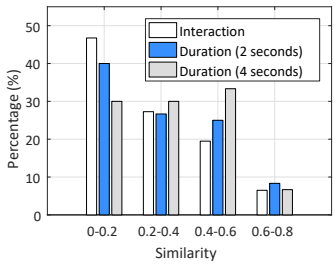
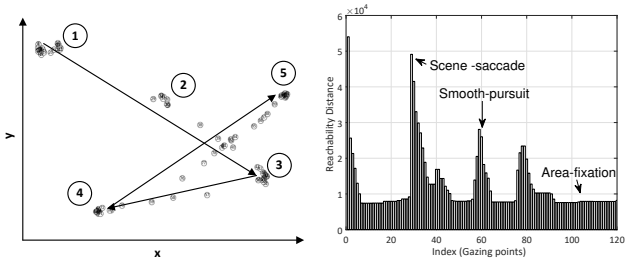


Fig. 8: The similarity of gazing patterns



(a) Gazing points (b) OPTICS result

Fig. 9: One example of gazing points classification

These touch actions are frequently used in mobile games. We use a decision tree [10] to classify them using five key features extracted from the data: duration, position number, direction, maximum touch area, and minimum touch area. Based on the definitions of touch interactions, this decision tree has four outputs: single-touch, press-drag, pan, and zoom. To train the decision tree and test the accuracy, we use 300 interactions labeled by the gamers. Finally, this decision tree achieves an average accuracy of 97%, which is adequate for the association analysis in our framework.

Figure 6 shows the interaction distribution of each game. We observe that three games have significant differences in touch interactions, which are affected by game genres. For example, single-touch is the most frequent interaction in G1 and G3, while G2 gamers have much more press-drag interactions, dragging a card to attack opponents. Since there is no scale up/down interaction in G2 and G3, we cannot find any zoom interaction from data traces.

VI. Insights into Viewers' Gazing Patterns

In this section, we first propose the data collection method in viewers' gazing investigation. Then, we analyze the characteristics of viewers' gazing data traces.

A. Gazing Data Collection

We choose Tobii eyeX[10] as the eye-tracking device to collect the users' gazing data due to its affordable price, suitable sampling rate, and high accuracy. It is connected to a desktop PC (DELL Optiplex 7010) through USB 3.0 port and attached to the frame of a 27-inch monitor (DELL U2715H), as shown in Figure 10. The eye-tracking device consists of

[10]http://www.tobii.com/xperience/

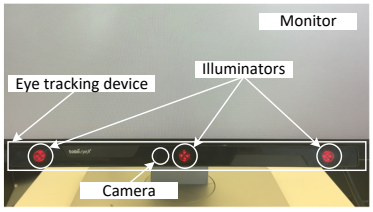


Fig. 10: The eye tracking device in our test-bed

three illuminators and one camera. The illuminators create the pattern of near-infrared light on viewer's eyes. Then, the camera captures high-resolution images of the viewer's eyes and the patterns. The eye-tracking algorithms in the device analyze these images and feedback the corresponding gazing positions on this monitor. The ten volunteers mentioned earlier have also assisted us to collect gazing data as viewers, who have personal profiles to calibrate the eye-tracking device before data collection. Each viewer watches three two-minute game videos selected from a gamer in Section V. We write a Python script to extract gazing data traces, including gazing positions and timestamps.

B. Gazing Classification

To analyze the characteristics of viewers' gazing patterns in MGC applications, we focus on the following three gazing patterns: *area-fixation* (AF), *smooth-pursuit* (SP), and *scene-saccade* (SS), which are based on the state-of-the-art eye-tracking research [20].

- Area-fixation (AF): The viewer's gaze is on a fixed area. Typically, viewers spend more time to watch the center of the game scene when there is no any gamer's interaction;
- Smooth-pursuit (SP): The viewer's gaze smoothly follows the movement of game objects, e.g., dragging gamer's builds and moving game cards;
- Scene-saccade (SS): The viewer scans the scene of the gamecasting, e.g., reading a notice board or descriptions.

Figure 9a shows several examples of these patterns. In this figure, we use labeled circles to indicate the area-fixation regions, where viewers focus on them in a short term. Due to the changes of game scenes, viewers quickly move gazing areas from region #1 to region #2, which corresponds to the scene-saccade pattern. Yet the slow movements among areas #3, #4, and #5 are considered as the smooth-pursuit pattern; that is, viewer's eyes follow the movements of a game object.

After investigating the gazing points, we find that perfectly classifying gazing points into three patterns is impossible due to the data noise. As such, we first pre-process the gazing points to improve the accuracy of classification. Because area-fixation has cluster feature in the 2D space and each gazing point also has a temporal dimension, i.e., timestamp, we employ the OPTICS (Ordering Points To Identify the CluSter-ing) algorithm [21] to process the time-series gazing points. The OPTICS algorithm can find the density-based clusters (i.e., area-fixation in our data traces) through calculating the distance of two gazing points. Moreover, this distance also reflects the speed of the eye movement from one area to another. Figure 9 shows a pre-process example. Figure 9a depicts five area-fixation regions, one scene-saccade pattern (from area #1 to #2), and three smooth-pursuit patterns (#2→#3, #3→#4, and #4→#5). Figure 9b shows the corresponding pre-process results using the OPTICS algorithm. We extract the features of the three gazing patterns based on the results from the OPTICS algorithm. Similar to the classification of touch interactions, 100 labeled patterns are used to train the decision tree, which achieves an average accuracy of 96%.

Figure 7 plots the distribution of gazing patterns in three videos. The feature of human eye movement determines that area-fixation accounts for the largest proportion in all cases. On the other hand, we can still observe the impact of touch interactions. For example, in G2, the percentage of SS is larger than the results of other two games. Similar insights also can be drawn from G3 in Figure 6 and 7, where PA and SS both have the lowest proportions.

To illustrate the associations between touch interactions and gazing patterns, we also choose two viewers to explore the similarity[11] of their gazing tiles. We divide these gazing tiles into different time windows using two approaches: (1) Interaction: we use the duration of every interaction trace to determine the size of each window, which is the minimum time slot for every association analysis; (2) Duration: considering that the duration of streaming segments are fixed in practical gamecasting applications, we divide the gazing data according to the segment duration. We plot the similarity of gazing data in Figure 8. From this figure, we observe that more than 50% of gazing tiles have more than 0.2 similarity for all approaches. Particularly, in Duration approach, we adopt two settings, i.e., 2 seconds and 4 seconds, and observe the percentage of similarity in 0.4-0.6 is higher than the Interaction approach, which implies that the similarity of gazing tiles exists in different viewers for the same touch interactions and the association exists between touch interactions and gazing patterns. Based on these investigations, we next model the associations between the gamers' interactions and the viewers' gazing patterns.

[11]In this paper, we use Jaccard Similarity, which is a statistic used for comparing the similarity of sample sets. $Jaccard(A,B) = \frac{|A \bigcap B|}{|A \bigcup B|}$

TABLE II: Encoding example

Data traces	Encoded transactions
{ST, (365, 632)};{AF, (250,430),(255,439)}	T_1. {115, 215}
{PD, (375, 700), (300, 1000)}; {SP, (280,500),(255,900)}	T_2. {125, 126, 225, 226}

VII. Touch-assisted Association Learning

Association rules describe the strong relations of items that occur frequently together in a data set. For instance, in market sale analysis, they can be used to discover the frequent combination of sales and product placements. In this section, we first introduce the preliminaries of association rule learning, and then build up such associations in our application scenario, addressing such challenges as transforming our data into the available format. In particular, we employ the classical Apriori algorithm [10] to build the associations between the gamers' touch interactions and the viewers' gazing patterns. The transformation however is applicable if other advanced association algorithms are to be used.

A. Preliminary

The inputs of association rules learning contain: (1) *items data*, A_i, where item $A_i \in I, i = \{1, \cdots, M\}$, and (2) a *transaction data set*, T, which consists of a set of transactions $< T_i, \{A_p, \ldots, A_q\} >$, where T_i is a transaction identifier and $A_i \in I, i = \{p, \cdots, q\}$. A collection of zero or more items is defined as an itemset. If an itemset contains k items, it is called k-itemset. An association rule is defined as an implication expression of form $X \rightarrow Y$, where $X \bigcap Y = \emptyset$ and $X, Y \subseteq I$. The *item support count* $\delta(X)$ of itemset X gives the number of transactions that contain a particular itemset. $\delta(X) = |\{T_i | X \in T_i, T_i \in T\}|$. To find the frequent occurring patterns, the support and confidence also need to be defined. Support determines how often a rule is applicable to a given data set, while confidence determines how frequently items in Y appear in transactions that contain X. The support $s(X \rightarrow Y)$ is defined as follows: $\delta(X \bigcup Y)/M$. The confidence of a rule $X \rightarrow Y$ is accordingly defined as $c(X \rightarrow Y) = \delta(X \bigcup Y)/\delta(X)$

To discover frequent patterns and build reasonably strong associations, we must specify two thresholds: *minimum support*, s', and *minimum confidence*, c'. The first step is to find all the itemsets that satisfy the threshold s'. These itemsets are called *frequent itemsets*. The second step is to extract all the rules from the frequent itemsets found in the previous step, if the confidence of these rules is no less than threshold c'. The second step is straightforward, while the first step needs more attention since it involves searching all possible itemsets. The Apriori algorithm employs a bottom-up approach by identifying the frequent individual items in the transaction data set and extending them to larger itemsets while the items satisfy the minimum support threshold. The frequent itemsets determined by the Apriori algorithm are then used to determine the association rules.

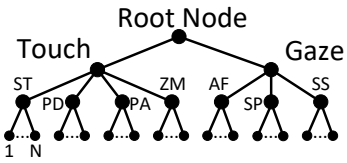


Fig. 11: The encoding tree of touch and gaze data

B. Touch-assisted Association

To map our data into the corresponding transaction T_i, we treat the touch interactions and gazing patterns as items and each transaction in our study as a combination of several touch items and gazing items from the start time of one touch to the start time of next touch. To simplify the discussion and fit the tile-based optimization context, we divide the positions of touch and gazing data into N tiles with the same size, where $N = n^2, n \in \mathbb{Z}^+$. Each item is then encoded based on the tree structure in Figure 11. Table II shows one encoding example. In this example, we set $N = 9$. $\{ST, (365, 632)\}$ and $\{AF, (250, 430), (255, 439)\}$ mean that (1) the gamer touches $(365, 632)$ once before the next interaction; (2) the viewer's gazing points contain $(250, 430), (255, 439)$, which is an area-fixation. The encoded transaction T_1 includes two items 115 (Touch-ST-5) and 215 (Gaze-AF-5). The rationale of this encoding method is that each encoded item contains all key information we needed. Notice that, we have already remove a duplicated item (215).

Algorithm 1 Touch-assisted Association Learning Algorithm

Input:
(1) A_1, Encoded touch interactions with timestamps
(2) A_2, Encoded gazing patterns with timestamps
(3) minimum support threshold s'
Output: Association Rules
1: $T \leftarrow$ Transaction(A_1, A_2);
// Generate transactions using the timestamps of the traces in A_1 and A_2
2: $G \leftarrow$ Apriori(T, s');
// Generate frequent itemsets using the Apriori Algorithm
3: **for** each frequent itemset g in G **do**
4: **if** all items in g come from the same type **then**
5: Remove g from G;
6: **end if**
7: **end for**
8: **return** G;

Using the Apriori algorithm, we can find all frequent l-itemsets for mining strong association rules in the encoded data set. Since we aim to find the association rules between touch interactions and gazing patterns, we remove the frequent itemsets that only contain one type of behaviors, e.g., $\{211, 212, 235\}$ and $\{119, 118, 127\}$. We summarize the touch-assisted association learning algorithm in Algorithm 1.

Given the frequent items, we can acquire the association rules to predict the viewer's gazing patterns. Through employing association rules for new touch interactions, the TAP module will predict the gazing patterns before the gamer's next touch interaction. As mentioned, the gazing region (i.e., the order of tile) is related to the viewer's QoE. Since there are three gazing patterns, we have four sets: L_1, including the AF regions, L_2, including the SP regions, L_3, including the SS regions, and L_4, including other regions. Finally, set $\{L_1, L_2, L_3, L_4\}$ is sent to the TBO module. Next, we illustrate how to utilize this information to optimize the energy consumption on mobile devices in the MGC scenario.

VIII. Tile-based Optimization

In this section, we first model the tile-based optimization problem with bandwidth and QoE constraints and transform it into an equivalent problem. We then present an efficient solution to solve it.

A. Problem Formulation

For ease of exposition, we assume that the duration D of a segment varies from a few seconds to several minutes (e.g., two seconds in our experiments). Each stream segment has N tiles and each tile has V versions. We thus denote each tile as $t_{i,j}$, $i \in \{1, \cdots, N\}, j \in \{1, \cdots, V\}$. We use $v(t_{i,j}) = j$ to denote the version information of a tile. Let $d_{i,j}$, $e_{i,j}$, and $q_{i,j}$ be the size, downloading energy consumption, and the quality of tile $t_{i,j}$, respectively. The information of file size $d_{i,j}$ can be acquired from streaming servers, the downloading energy consumption $e_{i,j}$ can be estimated based on the research [22], and the quality $q_{i,j}$ depends on the encoding bitrate. We use S_w to denote the index of the segment being watched and S_c to denote the index of the latest cached segment in DASH client. The tile-based optimization in MGC can thus be formulated as to maximize tile quality per energy consumption.

$$Maximize: \sum_{i=1}^{N}\sum_{j=1}^{V} \frac{q_{i,j}}{e_{i,j}} x_{i,j} \tag{1}$$

subject to:
Streaming Availability Constraints (2) and (3)

$$\sum_{i=1}^{N}\sum_{j=1}^{V} d_{i,j} x_{i,j} \leq B \cdot D \cdot (S_c - S_w) \tag{2}$$

$$\sum_{j=1}^{V} x_{i,j} = 1, i \in \{1, \cdots, N\}, x_{i,j} = \{0, 1\} \tag{3}$$

Foveated Quality Constraints (4), (5) and (6)

$$v(t_{i,a}) \geq \alpha, t_{i,a} \in L_1 \tag{4}$$

$$0 \leq v(t_{i,a}) - v(t_{i,b}) \leq \beta, t_{i,a} \in L_k, t_{i,b} \in L_{k-1} \tag{5}$$

$$v(t_{i,a}) - v(t_{i,b}) = 0, t_{i,a}, t_{i,b} \in L_k, k \in \{1, \cdots, 4\} \tag{6}$$

where B is the average bandwidth, and α and β are two tunable parameters. The rationale of Streaming Availability Constraints is as follows: (1) all tiles in a segment should be downloaded completely before the cache becomes empty, which guarantees a smooth playback of the live stream; and (2) the clients only need to download one quality version for every tile, which avoid extra bandwidth consumption. In Foveated Quality Constraints, two tunable parameters α and β control the range of tile version based on the prediction results.

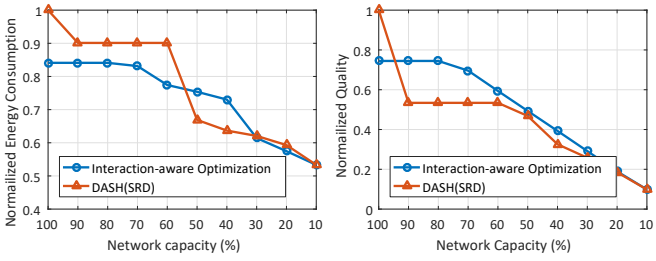
B. Solution

If we only consider the streaming availability constraints, the tile-based optimization problem can be transformed into the following Multiple Choice Knapsack (MCK) problem: there are N mutually disjoint classes $C_1, \cdots, C_N$ of items (i.e., tiles) to be picked into a knapsack (i.e., segment) of capacity $B \cdot D \cdot (S_c - S_w)$. Each item $j \in C_i$ has a profit $q_{i,j}/e_{i,j}$ and a weight $d_{i,j}$, and the problem is to choose exactly one item from each class such that the profit sum is maximized without exceeding the capacity of the corresponding weight sum. Through introducing the binary variables $x_{i,j}$, which take on 1 if and only if item j is chosen in class C_i, we can transform our problem into a corresponding MCK problem with known practically efficient solutions available (e.g., pseudopolynomial-time dynamic programming) [23]. It is worth noting that the optimal solution of this MCK problem may not meet the foveated quality constraints. We therefore first narrow down the solution space according to Foveated Quality Constraints before employing dynamic programming. If there is no feasible solution, we can adjust α and β to enlarge the solution space. Finally, we can obtain the version selection of each tile. We employ the trace-driven simulation to explore how to adjust these parameters in Section IX-A.

IX. Performance Evaluation

In this section, we evaluate our interaction-aware optimization through the trace-driven simulations and user study under various settings. A rooted Samsung Galaxy S5 with Android 4.4.2 is connected to a PC (DELL Optiplex 7010). Then, we collect the communication data using *Android tcpdump*[12] and retrieve the tile files using *wget*[13] on this S5 in a campus network. To measure the energy consumption, we supply the power of this S5 using a Monsoon power monitor[14], which connects to a PC through a USB connection and feedbacks the energy consumption to this PC in real-time. During the energy measurement, we turn off the mobile screen to guarantee that the power is consumed by the tile transmission only.

A. Trace-driven Simulation



(a) Energy consumtpion (b) Stream quality

Fig. 13: Comparison of our approach and DASH(SRD)

We first investigate the impact of parameters α and β in terms of energy consumption, data transmission, and streaming quality. We divide a 2-second segment into 25 tiles, which are encoded at eight bitrates. Based on the predicted gazing patterns, as shown in Figure 12a, we conduct the tile-based optimization to determine which tile should be obtained and collect the corresponding data traces. Through investigating these traces, we plot the normalized results under different α and β settings in Figure 12b, 12c, and 12d. We observe that α has higher impact than β; therefore, if there is no feasible solution, the optimization algorithm first adjusts β, and then changes α. On the other hand, the algorithm also attempts multiple settings of α and β to achieve a higher stream quality. In our simulation, we set $\alpha = 8$ and $\beta = 1$ in default. When the algorithm extends the solution space, it will fix the value of $\alpha - \beta$ and consider the solution space in multiple settings (e.g., $\alpha = 8, \beta = 2$ and $\alpha = 7, \beta = 1$).

To explore the effectiveness of our solution, we also explore the performance of the tile-based optimization under different network capacities through throttling the bandwidth on the mobile device. We compare our method with the original DASH(SRD) selection, in which all tiles in a segment have the same version. Figure 13 plots the results. In Figure 13a, we observe that our method has lower energy consumption except for two data points at 50% and 40% of network capacity. The reason is that the original DASH adaptation may reduce the quality of tiles suddenly to accommodate the decrease of bandwidth, while our method fully utilizes the available bandwidth to optimize the tile quality per energy consumption. Figure 13b shows that our solution optimizes the tile quality selection with lower energy consumption except for the case of 100% of network capacity, which is determined by our optimization objective, i.e., efficient energy utilization.

B. User Study

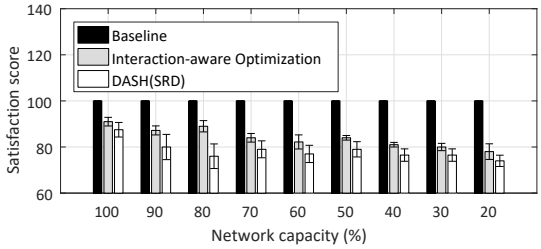
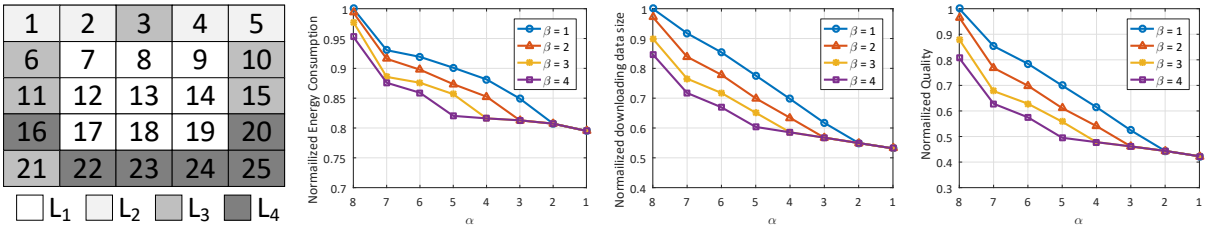


Fig. 14: Satisfaction Score

We further conduct user study to examine the QoE of our solution. We first select a 10-second video clip and generate 25 tiles at 8 versions for all the segments using *ffmpeg*[15], *x264 encoder*[16], and *mp4box*[17]. According to the corresponding touch interactions, we predict gazing pattern sets and output the video clips under different network capacities. We also produce DASH(SRD) video clips as the corresponding comparisons. As such, there are two clip-sets from our approach and DASH(SRD), respectively.

[12] http://www.androidtcpdump.com/
[13] https://www.gnu.org/software/wget/
[14] https://www.msoon.com/LabEquipment/PowerMonitor/
[15] https://ffmpeg.org
[16] http://www.videolan.org/developers/x264.html
[17] https://gpac.wp.mines-telecom.fr/mp4box



(a) Gazing pattern sets (b) Energy consumption (c) Data transmission (d) Stream quality

Fig. 12: Impacts of α and β

A desktop PC simultaneously plays two video clips under the same network capacity in random windows. The viewers compare their quality differences and evaluate them via satisfaction scores from 1(worst) to 100(best). Figure 14 shows that the evaluation results under different network capacities. We use the original video clip as the baseline; hence all satisfaction scores are 100. From this figure, we observe that our approach always achieves a higher score (3%-13%) than DASH(SRD).

X. Conclusion and Further Discussion

In this paper, we explored the emerging mobile gamecasting systems, in which both stream sources and receivers are mobile devices. Through trace analysis of a crowdsourced data collection, we identified the relations between the touch interactions of the gamers (broadcasters) and the gazing patterns of the viewers. Motivated by this, we proposed an interaction-aware optimization framework that includes two novel designs: (1) a touch-assisted prediction builds association rules offline and fulfills viewers' gazing pattern prediction online; and (2) a tile-based optimization for energy consumption and quality selection under limited network capacity. The experiment results showed that our solution effectively utilizes the available bandwidth with better tile quality and more efficient energy consumption. The user study also proved that it steadily improves the viewers' satisfactions (from 3% to 13%). We are currently integrating the whole framework with practical implementation to further evaluate the performance from different perspectives, such as the parameter selection in the optimization algorithm, the impact of the number of tiles, and the energy utilization under different wireless networks.

Acknowledgment

This publication was made possible by NPRP grant #[8-519-1-108] from the Qatar National Research Fund (a member of Qatar Foundation). The findings achieved herein are solely the responsibility of the authors. Dr. J. Liu's work is also supported by the Natural Sciences and Engineering Research Natural Sciences and Engineering Research Council (NSERC) of Canada. Dr. Z. Wang's work is supported in part by SZSTI under Grant No. CXZZ20150323151850088.

References

[1] Google, "The rise of avid mobile gamers on youtube," https://goo.gl/ofbNuf, March 2016.
[2] Twitch, "Retrospective," https://www.twitch.tv/year/2015, Dec. 2015.
[3] W. Hu and G. Cao, "Energy-aware Video Streaming on Smartphones," in *IEEE INFOCOM, 2015*.
[4] M. A. Hoque, M. Siekkinen, J. K. Nurminen, and M. Aalto, "Dissecting Mobile Video Services: An Energy Consumption Perspective," in *IEEE WoWMoM, 2013*.
[5] X. Li, M. Dong, Z. Ma, and F. C. Fernandes, "GreenTube: Power Optimization for Mobile Videostreaming via Dynamic Cache Management," in *ACM MM, 2012*.
[6] S. Wilk, R. Zimmermann, and W. Effelsberg, "Leveraging Transitions for the Upload of User-generated Mobile Video," in *ACM MoVid, 2016*.
[7] S. V. Rajaraman, M. Siekkinen, V. Virkki, and J. Torsner, "Bundling Frames to Save Energy While Streaming Video from LTE Mobile Device," in *ACM MobiArch, 2013*.
[8] J. Ryoo, K. Yun, D. Samaras, S. R. Das, and G. Zelinsky, "Design and Evaluation of a Foveated Video Streaming Service for Commodity Client Devices," in *ACM MMSys, 2016*.
[9] H. Ahmadi, S. Zad Tootaghaj, M. R. Hashemi, and S. Shirmohammadi, "A Game Attention Model for Efficient Bit Rate Allocation in Cloud Gaming," *Multimedia Systems*, vol. 20, no. 5, pp. 485–501, 2014.
[10] X. Wu, V. Kumar, J. Ross Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. J. McLachlan, A. Ng, B. Liu, P. S. Yu, Z.-H. Zhou, M. Steinbach, D. J. Hand, and D. Steinberg, "Top 10 Algorithms in Data Mining," *Knowledge and Information Systems*, vol. 14, no. 1, pp. 1–37, 2008.
[11] R. Shea, D. Fu, and J. Liu, "Towards Bridging Online Game Playing and Live Broadcasting: Design and Optimization," in *ACM NOSSDAV, 2015*.
[12] Q. He, J. Liu, C. Wang, and B. Li, "Coping With Heterogeneous Video Contributors and Viewers in Crowdsourced Live Streaming: A Cloud-Based Approach," *IEEE Transactions on Multimedia*, vol. 18, no. 5, pp. 916–928, 2016.
[13] S. M. Kolly, R. Wattenhofer, and S. Welten, "A Personal Touch: Recognizing Users Based on Touch Screen Behavior," in *ACM PhoneSense, 2012*.
[14] Y. Chen, J. Sun, R. Zhang, and Y. Zhang, "Your Song Your Way: Rhythm-based Two-factor Authentication for Multi-touch Mobile Devices," in *IEEE INFOCOM, 2015*.
[15] A. De Luca, A. Hang, F. Brudy, C. Lindner, and H. Hussmann, "Touch Me Once and I Know It's You!: Implicit Authentication Based on Touch Screen Patterns," in *ACM CHI, 2012*.
[16] L. Zhang, F. Wang, and J. Liu, "Mobile Instant Video Clip Sharing: Modeling and Enhancing View Experience," in *IEEE IWQoS, 2016*.
[17] Y. Sun, I. Ahmad, D. Li, and Y.-Q. Zhang, "Region-based Rate Control and Bit Allocation for Wireless Video Transmission," *IEEE Transactions on Multimedia*, vol. 8, no. 1, pp. 1–10, 2006.
[18] Z. Li, S. Qin, and L. Itti, "Visual Attention Guided Bit Allocation in Video Compression," *Image and Vision Computing*, vol. 29, no. 1, pp. 1 – 14, 2011.
[19] O. A. Niamut, E. Thomas, L. D'Acunto, C. Concolato, F. Denoual, and S. Y. Lim, "MPEG DASH SRD: Spatial Relationship Description," in *ACM MMSys, 2016*.
[20] A. Duchowski, *Eye Tracking Methodology: Theory and Practice*. Springer Science & Business Media, 2007, vol. 373.
[21] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander, "OPTICS: Ordering Points to Identify the Clustering Structure," *SIGMOD Rec.*, vol. 28, no. 2, pp. 49–60, 1999.
[22] W. Hu and G. Cao, "Energy Optimization Through Traffic Aggregation in Wireless Networks," in *IEEE INFOCOM, 2014*.
[23] H. Kellerer, U. Pferschy, and D. Pisinger, *Introduction to NP-Completeness of Knapsack Problems*. Springer, 2004.