Received 24 July 2023, accepted 19 August 2023, date of publication 23 August 2023, date of current version 1 September 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3307710

RESEARCH ARTICLE

Velocity- and Error-Aware Switching of Motion Prediction Models for Cloud Virtual Reality

AIRLANGGA ADI HERMAWAN[1], YAKUB FAHIM LUCKYARNO[1], ISFAN FAUZI[1], DEREK KWAKU POBI ASIEDU[1], (Member, IEEE), TAE-WOOK KIM[2], DEOK-YOUNG JUNG[2], JIN SAM KWAK[3], AND JI-HOON YUN[1,4], (Senior Member, IEEE)

[1]Department of Electrical and Information Engineering, Seoul National University of Science and Technology, Seoul 01811, South Korea
[2]Clicked Inc., Seoul 03920, South Korea
[3]WILUS Institute of Standards and Technology, Gyeonggi 13595, South Korea
[4]Research Center for Electrical and Information Technology, Seoul National University of Science and Technology, Seoul 01811, South Korea

Corresponding author: Ji-Hoon Yun (jhyun@seoultech.ac.kr)

This work was supported in part by the Institute of Information Communications Technology Planning & Evaluation (IITP) funded by the Korean Government [Ministry of Science and ICT (MSIT)] under Grant 2017-0-00650, in part by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education under Grant NRF-2019R1A6A1A03032119, and in part by the Seoul National University of Science and Technology.

ABSTRACT Offloading virtual reality (VR) computations to a cloud computing entity can enable support for VR services on low-end user devices but may result in increased latency, which will lead to mismatch between the user's viewport and the received VR image, thus inducing motion sickness. Predicting future motion and rendering future images accordingly is a promising solution to the latency problem. In this paper, we develop velocity- and error-aware model switching schemes applicable to a wide range of existing motion prediction models. First, we consider the chattering problem of machine learning (ML)-based prediction models and the relationship between the velocity and the prediction error gap between an ML model and the case of no prediction (NOP). Accordingly, we propose a velocity-aware switching (VAS) scheme that combines the outputs from the ML model and the NOP case via a weight determined by the head motion velocity. Next, we develop an ensemble method combining a set of outputs from VAS and other models, called error-aware switching (EAS). EAS switches between model outputs based on the error statistics of those outputs under the parallel execution of multiple models, including VAS models. For EAS, schemes for both hard switching and soft integration of the model outputs are proposed. We evaluate the proposed schemes based on real VR motion traces for diverse ML-based prediction models.

INDEX TERMS Virtual reality, VR, cloud VR, motion prediction, machine learning, ensemble.

I. INTRODUCTION

Virtual reality (VR) services allow a user to immerse him-/herself in a virtual world by enabling the user to explore a virtual space, which is rendered in the form of stereoscopic images, in the same way he/she would in the real world, i.e., through head movements [1]. This process is achieved by means of a VR-dedicated headset device equipped with a head-mounted display (HMD) and a built-in inertial measurement unit (IMU) to capture the user's head motion. The user views a stereoscopic image of the virtual world that corresponds to his/her current viewport (as estimated from the latest head tracking data) on an HMD panel at an ultrashort viewing distance (several centimeters) through binocular magnifying lenses to enable a large field of view.

Offloading heavy VR computations to a network cloud or edge computing entity [2], [3], [4] possessing sufficient computational power and then wirelessly streaming the rendered VR image frames to the VR headset, which we call *cloud VR* in this paper, is rapidly emerging as a promising solution. The abundant processing and storage resources

The associate editor coordinating the review of this manuscript and approving it for publication was Tai-Hoon Kim.

found in the cloud can make high-quality and pay-as-you-go VR services available anytime and anywhere across the globe to a large group of online VR users with access to affordable low-end client VR headsets wirelessly connected to the Internet [2], [3], [4]. With the cloud providing the necessary resources to execute and render scenes, the client headsets are left only with the task of displaying the content streamed over the wireless network. Thus, higher user mobility is facilitated, and the headsets' battery life is also expected to be prolonged due to their light operations [5].

However, the offloading of VR computations to the cloud may result in increased *latency* [5]. The large volume of VR display data necessitated by the required ultrahigh resolution tends to result in a long latency due to the need to transport these VR data over the network. As the latency increases, the mismatch between the rendered VR image and the user's viewport at the time of scan-out also increases since the user is likely to continue to move after the image is rendered. A sufficiently large mismatch will lead to unpleasant physiological symptoms due to the lag between the vestibular and visual systems of the human body, commonly referred to as *motion sickness* [6]. The image reprojection technique [7], [8], [9]—in which a rendered frame is adjusted to account for head pose changes that occur after the scene is rendered—helps to reduce mismatch and alleviate motion sickness. However, as the end-to-end latency increases, large black borders may begin to appear after reprojection [5], [10], negatively impacting the user's quality of experience (QoE) [11].

Predicting future motion and rendering future frames accordingly is one solution to reduce or even eliminate the impact of latency in VR computing. In [12], two straightforward approaches for motion prediction were presented, one assuming constant velocity and the other assuming constant acceleration during the prediction interval. Both show acceptable prediction results for short prediction intervals but fail for long intervals because the assumptions become unaligned with reality. Multiple machine learning (ML)-based prediction solutions have also been proposed based on a linear regression model [13], multilayer perceptrons (MLPs) [14], [15], convolutional neural networks (CNNs) [16], recurrent neural networks (RNNs) [17], [18], [19], long short-term memory (LSTM) [20], [21], and gated recurrent units (GRUs) [20], [22], which learn the correlations between past head pose data and the future head pose. Another approach to motion prediction is to utilize the user's neck surface electromyographic (sEMG) data to make predictions using a trained artificial neural network, based on the fact that myoelectric signals precede exertion [23], [24].

In this paper, we develop a velocity-aware switching (VAS) scheme for motion prediction models that predict a VR user's future head orientation using only data from inertial sensors. This scheme is applicable in combination with a wide range of existing motion prediction models and with current VR headsets at no extra hardware cost. First, we consider the chattering problem of ML-based prediction models and the relationship between the velocity and the prediction error gap between an ML model and the case of no prediction (NOP). The related observations reveal that the chattering problem leads to increased errors at low velocity. We also show that applying low-pass filters to suppress chattering gives rise to a time lag in the resulting signals and thus is not suitable for motion prediction. Accordingly, we propose a VAS scheme in which the outputs from such an ML model and the NOP case are combined via a weight determined based on the head motion velocity such that the weight of the NOP output increases as the velocity decreases. Next, we develop an ensemble method combining a set of outputs from VAS and other models, called error-aware switching (EAS). EAS switches between model outputs based on the error statistics of those outputs under the parallel execution of multiple models, including VAS models. For EAS, schemes for both hard switching (EAS-H) and soft integration (EAS-S) of the model outputs are proposed. EAS-H selects the output of a single error-minimizing prediction model as the final output, while EAS-S combines the outputs of the considered models with individual weights determined based on their error statistics.

We evaluate the proposed schemes based on real VR motion traces captured from a VR headset for multiple test players, considering various combinations of ML models. The experimental results show that the VAS scheme always reduces the mean absolute error (MAE) by 7% to 25% in yaw and by 13% to 27% in pitch for anticipation times of 50 to 300 ms. EAS is shown to further decrease the MAE relative to VAS by up to 80% in yaw and 50% in pitch. To demonstrate the impact of the proposed schemes on VR quality enhancement, we also evaluate the level of black border generation, which is shown to be improved by the proposed schemes.

The rest of this paper is organized as follows. Recent studies related to VR motion prediction are reviewed and discussed in Section II. The system model is described in Section III. Section IV reports important observations on the chattering problem and the impact of velocity on this phenomenon as obtained from experiments. Sections V and VI describe the proposed schemes. Experimental results illustrating the performance gains of the proposed schemes are presented in Section VII, and the conclusion is given in Section VIII.

Remarks. We omit the results for the roll coordinate since the overall trends for the roll coordinate are very similar to those for the other two coordinates (yaw and pitch), and the level of movement is lower in the roll coordinate than in the other two coordinates.

II. RELATED WORK

Motion prediction using inertial sensors is cheap and suitable for VR since all VR headsets have built-in inertial sensors for head orientation tracking. In conventional VR systems, constant-rate-based prediction (CRP) and constant-acceleration-based prediction (CAP) have been considered [12], in which a constant velocity or a constant acceleration is assumed throughout a specified anticipation time for prediction; however, the use of these approaches is limited to short anticipation times of a few tens of milliseconds. In [25], the authors designed a Kalman filter for head motion prediction for short anticipation times. In [26], the authors proposed a double exponential smoothing filter method to predict the user's position and rotation in a manner equivalent to Kalman and extended Kalman filtering predictors but at a lower complexity. In [27], the probability distribution of the fixation-point prediction error was derived to be a normal distribution under certain assumptions, and a closed-form expression for the prediction of the future viewport at a given confidence level was proposed.

The utilization of ML algorithms has been considered in most research works on motion prediction. [13] showed strong short-term autocorrelations of viewer motions and developed linear regression models to predict the viewer's viewpoint and the prediction's deviation. In [14] and [15], the authors developed a prediction algorithm using an MLP and a framework for streaming 360-degree videos based on viewer motion prediction. The authors of [16] applied model-agnostic meta-learning to a one-dimensional CNN for predicting the head orientation of a new user. The authors of [20] proposed offline learning algorithms for head orientation prediction using linear regression (LR), MLP, and RNN models based on LSTM and GRU architectures. They also extended these algorithms to online learning, combined with proactive uplink retransmission for the stable collection of user motion data for training. In [17], the authors used an RNN to predict viewpoint preferences in continuous time slots. In [22], the authors used a GRU model for user motion prediction and other models for the estimation of the terahertz communication environment for wireless VR. Fan et al. [18] proposed the use of an RNN to predict the fixation point of the user, where the input to the RNN consists of the user motion data and the saliency map from the video frame and the output is the orientation of the user. A similar approach was also proposed by [19].

Approaches using sEMG data have also been proposed. Barniv et al. [23] captured multichannel sEMG signals from the user's neck muscles, obtained multiple features from the captured sEMG signals and input them into an Elman neural network (ENN) to be mapped to the head velocity. Polak et al. [24] used an MLP taking EMG signals and current kinematics data as input to produce two outputs: angular velocity and angular acceleration. However, sEMG-based methods are challenging to integrate with a VR headset due to the difficulty of integrating sEMG electrodes into a VR headset and the instability of the electrode contacts.

Some studies have investigated the benefits of incorporating additional data for motion prediction. In [28] and [29], a saliency map of the current image projected into the user's headset served as a basis for predicting head motion. Hou et al. [21] utilized an LSTM model taking tiles and other features from the current viewpoint as input to output the fixation point of the user. Stein et al. [30] trained an LSTM model using position, orientation, and eye-tracking features as input to predict the user's future position. These authors found that incorporating eye data was particularly beneficial in scenarios involving changes in walking speed. Additionally, an attempt has been made to predict the future eye gaze [31], demonstrating performance gains for foveated rendering in cloud VR.

Another approach addressed in the literature for achieving cloud VR is predictive rendering, which involves generating multiple candidate scenes by predicting user actions. Outatime [32] is a cloud gaming system that renders speculative frames for possible future outcomes. These frames are delivered to the client one round trip time ahead, allowing for quick recovery from mis-speculations when they occur. Ebner et al. [33] developed a VR display architecture for video see-through mixed reality that delivers focus cues across a large workspace. This architecture employs gaze-contingent layered displays and mixed reality focal stacks, leveraging a multiplane image representation of multiple views for rendering. Additionally, Liu et al. [34] introduced Vues, which adaptively transcodes a volumetric video frame into multiple 2D views based on user viewport prediction. The client then selects the view that optimizes the QoE from among the delivered candidates for display.

The present work focuses on either switching among or combining the outputs of multiple prediction models to overcome the challenges presented by ML-based models, thus achieving enhanced prediction accuracy, rather than on designing a novel ML-based model. Therefore, the schemes proposed in our work do not compete with other prediction models but rather are applicable in combination with a wide range of existing and future prediction models (possibly any type of model).

III. SYSTEM MODEL

A. OPERATIONAL PROCEDURES AND DATA FLOWS

Fig. 1 illustrates the content creation and streaming flow of the considered cloud VR system, in which a VR user establishes a VR session with a computing host of a cloud computing entity over a wireless Internet connection. The user's headset device continuously tracks the head pose of the user and periodically sends head pose data elements to the cloud. We consider a three-tuple of head pose data to specify the current orientation of the user's head: θ (pitch), ψ (roll), and ϕ (yaw), which are defined as rotations in a right-handed coordinate system [35]. Upon reception of the head pose data, the cloud renders a VR image corresponding to the received head pose of the user and transfers the image to the user over the network connection. Once the headset device receives an image, it stores the image in a playout buffer, and it scans the buffered images out to its display panel at a prespecified frame rate. Regarding notation, we denote the value of variable x at time t by $x[t]$. To denote a time series of x at times $t_1, t_2, \cdots$, we use $x[t_1, t_2, \cdots]$.

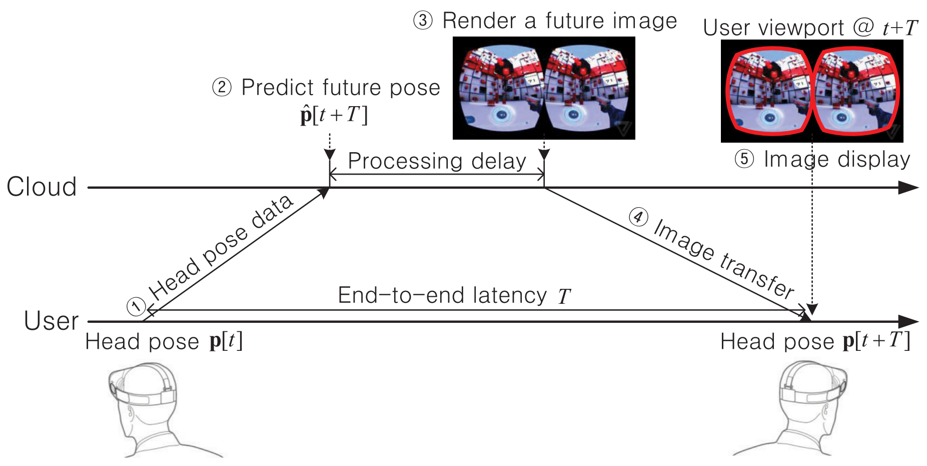


FIGURE 1. Content creation and streaming flow with head pose prediction for a VR display.

We assume that the head orientation measured at t by the IMU of the user's headset is given by $\mathbf{p}[t] = \langle\theta[t], \psi[t], \phi[t]\rangle$. Let the total end-to-end latency from when the sensor unit measures the user's head pose until the VR image corresponding to the respective sensor data is displayed at the user device be $T(>0)$. Accordingly, upon reception of the sensor data $\mathbf{p}[t]$, the VR host in the cloud must predict the user's head orientation at $t + T$. Thus, we call T the anticipation time for prediction. We denote the predicted orientation at $t + T$ by $\hat{\mathbf{p}}[t + T]$.

We wish to predict the user's head orientation at $t + T$ from the information available by t, which consists of a window of sensor data samples representing head orientations $\mathbf{p}[t_1, t_2, \cdots, t_W]$, angular velocities $\dot{\mathbf{p}}[t_1, t_2, \cdots, t_W]$ obtained from a gyroscope, and accelerations $\ddot{\mathbf{p}}[t_1, t_2, \cdots, t_W]$ obtained from an accelerometer, where W is the window size (the number of data samples serving as input), $t_k = t - (k - 1)\tau,\ k = 1, 2, \cdots, W$, and τ is the time interval between consecutive data samples. Accordingly, the predicted head orientation, denoted by $\hat{\mathbf{p}}[t+T] = \langle\hat{\theta}[t + T], \hat{\psi}[t + T], \hat{\phi}[t + T]\rangle$, can be defined as a function of the abovementioned sensor data. Thus, we have

$$\begin{aligned}\hat{\mathbf{p}}[t+T] = f_{\theta,T}(&\mathbf{p}[t_1, t_2, \cdots, t_W];\\ &\dot{\mathbf{p}}[t_1, t_2, \cdots, t_W];\\ &\ddot{\mathbf{p}}[t_1, t_2, \cdots, t_W]),\end{aligned} \tag{1}$$

where θ represents the model parameters of the function f. Then, the prediction error is defined as

$$\mathbf{e}[t + T] = \hat{\mathbf{p}}[t + T] - \mathbf{p}[t + T]. \tag{2}$$

For the prediction of N samples, we calculate the mean absolute error (MAE) of prediction as

$$\bar{\mathbf{e}} = \frac{1}{N}\sum_{k=1}^{N} |\mathbf{e}[k\tau]|. \tag{3}$$

The function (model) f and its parameter set θ need to be found so as to minimize the MAE $\bar{\mathbf{e}}$.

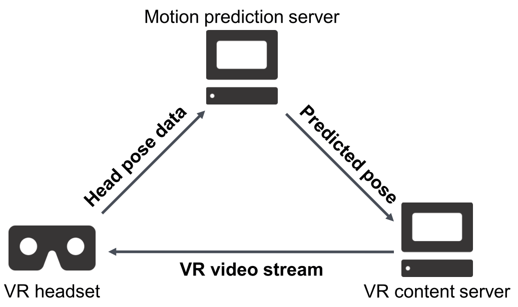


FIGURE 2. Experimental VR testbed system.

In the prediction framework of Eq. (1), the NOP case, i.e., the case in which the cloud simply uses the head pose data as they are received, is specified as $\hat{\mathbf{p}}[t + T] = \mathbf{p}[t]$.

B. VR TESTBED SYSTEM

The architectural components of our testbed system are illustrated in Fig. 2. This system consists of three main elements: (a) a content server, (b) a motion prediction server, and (c) a VR headset. The content and motion prediction servers are each connected to a wireless router using a Gigabit Ethernet connection, while the client VR headset (the Meta Quest 2 [36]) connects to the wireless router via an IEEE 802.11ac Wi-Fi interface. To ensure that the highest link speed of the Wi-Fi interface is consistently utilized, the headset is positioned close to the router. The essential experimental data for the testbed system can be found in [5].

The headset periodically transmits the user's most recent pose information to the motion prediction server at a frequency of 72 Hz. The motion prediction algorithm, operating on the motion prediction server, generates predicted head pose data and transfers these data to the content server. Based on these data, the content server carries out a 3D simulation and renders a VR viewpoint image corresponding to the predicted pose. The renderer's image output is then

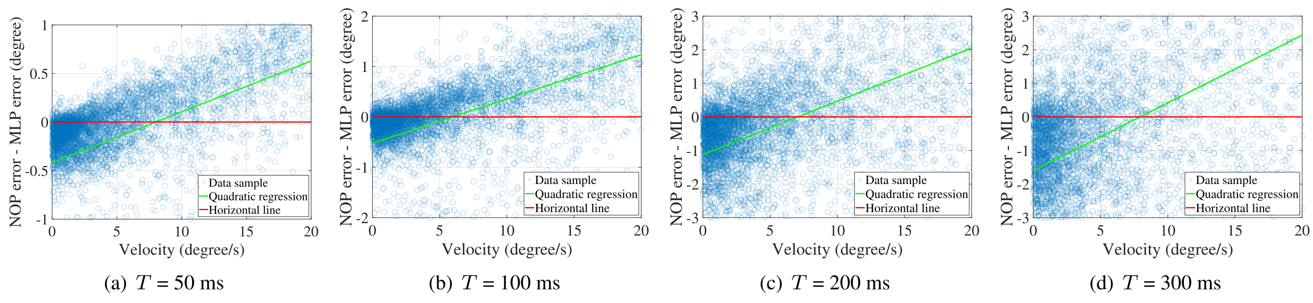


FIGURE 3. Scatter plots of NOP error - MLP error (yaw).

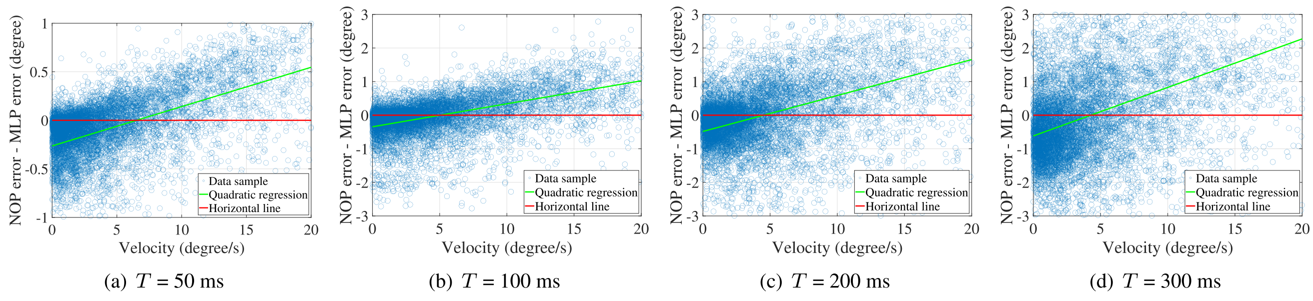


FIGURE 4. Scatter plots of NOP error - MLP error (pitch).

passed to a video encoder for compression before being transmitted back to the client headset (transmission of an uncompressed VR stream demands an exceptionally high data rate, which is currently infeasible with existing wireless technologies [37]). The rendering engine and encoder are configured to produce 72 frames per second, which equates to one frame every 13.89 ms. At the client headset, frame packets are received and assembled into frames. The frame data are then forwarded to the headset's video decoder. Once decoded, the current frame image is stored and remains in the playout buffer for a predetermined delay, regulating the end-to-end latency to a specified value. Finally, the image is reprojected based on the latest pose information obtained from the IMU. The reprojected image is placed in the frame buffer and ultimately displayed by the headset at the native refresh rate of the screen.

The streaming protocol for VR frames, which include audio data, is a custom-built protocol that utilizes the User Datagram Protocol (UDP). Meanwhile, the protocol for pose information relies on the Transmission Control Protocol (TCP)-based ZeroMQ messaging protocol [38]. In our experiments, we adjusted the playout delay parameter of the headset's playout buffer to account for various end-to-end latency values typically observed in cloud services, ranging from tens to several hundreds of milliseconds [39], [40], [41], [42].

IV. OBSERVATIONS OF THE IMPACT OF VELOCITY ON PREDICTION GAIN

Figs. 3 and 4 show the differences between the NOP error and the MLP error in the yaw and pitch coordinates, respectively, for varying anticipation times T (the architectural configurations of the ML models considered in this section are described in Section VII.A). A positive difference means that the MLP prediction is better than the NOP case, whereas a negative difference means that the MLP prediction is worse. These figures show that at a high velocity, the MLP prediction mostly performs better than the NOP case. However, as the velocity approaches zero, i.e., as the user moves less, there are many more frames in which the MLP output is worse than the NOP output.

Fig. 5 shows the time evolution of the true orientation, the NOP output and the outputs of two prediction models. The prediction outputs better follow the true orientation, while the NOP output shows a lagged pattern relative to the true orientation, with a lag equal to the anticipation time. However, the outputs of the prediction models contain noise. Consequently, when the motion is steady (the orientation change appears flat), the NOP output becomes closer to the true orientation, while the noise of the prediction models causes their outputs to be worse than the NOP case. A similar trend is also noticeable in the time evolution of the prediction error, as shown in Fig. 6. The errors of the prediction models fluctuate around zero, while the NOP errors exhibit higher and lower peaks. However, when the motion is steady (during the periods of 4–5 s and 8.5–10 s in yaw and the periods of 0.5–1.5 s and 8.5–10 s in pitch), the errors of the prediction models continue to fluctuate, sometimes with biases, while the NOP errors are close to zero.

It is expected that such prediction model noise can be filtered out by applying a low-pass filter. Fig. 7 shows

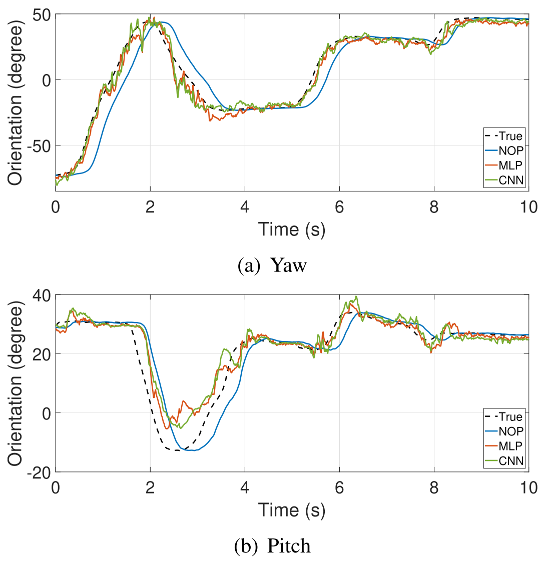


FIGURE 5. Time evolution of the prediction output (T = 300 ms).

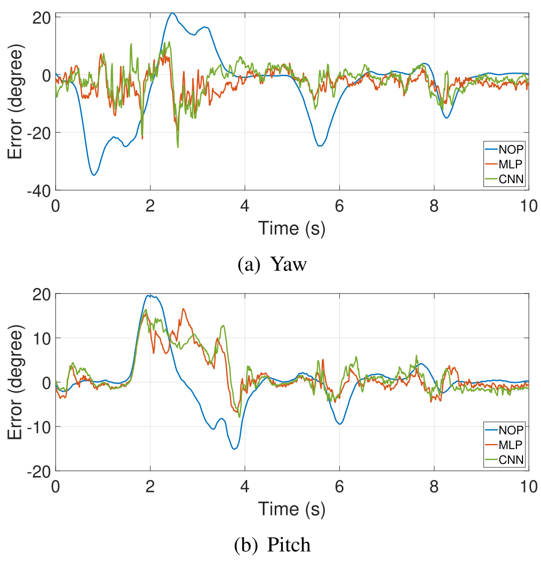


FIGURE 6. Time evolution of the prediction error (T = 300 ms).

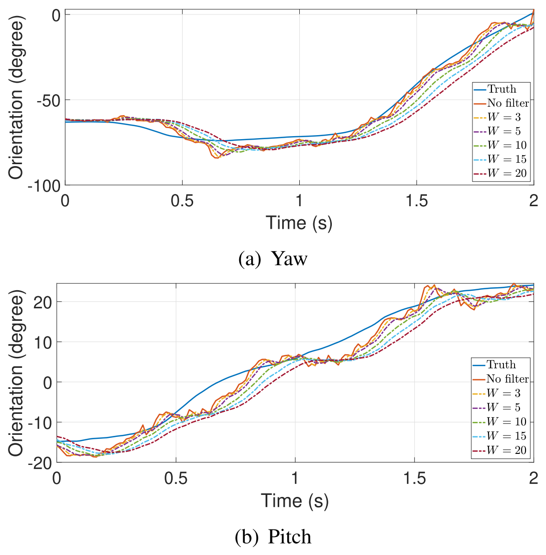


FIGURE 7. Moving average output (300 ms).

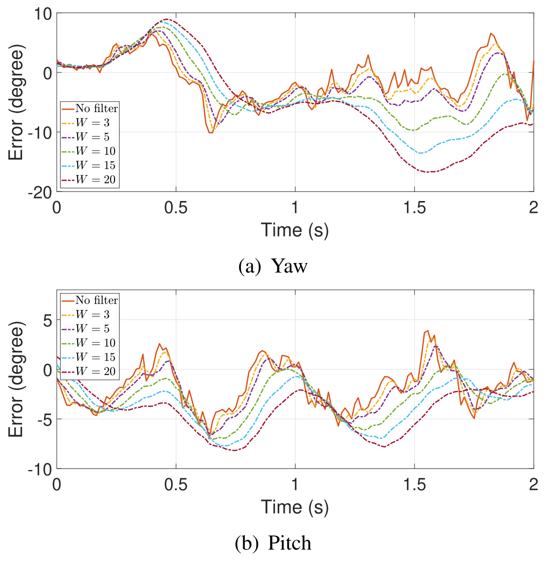


FIGURE 8. Error of the moving average (300 ms).

the prediction results obtained through moving averaging with different sliding window sizes. This figure shows that taking the moving average reduces the noise, as expected, and increasing the window size further decreases the noise level. However, it is also apparent that the moving average produces lagged output, and the lag becomes more severe as the window size increases. As shown in Fig. 8, the lag induced by the moving average operation causes the prediction error to increase as the window size increases. When a different filter—the Savitzky–Golay (SG) filter—is applied, similar trends are seen, as shown in Figs. 9 and 10.

The above observations can be summarized as follows:

- When the level of user motion is not high, the prediction results of ML models tend to become worse than the NOP results.
- This phenomenon arises from the noise (chattering) in the ML prediction results, which is consistently present even when the user motion is steady.
- A low-pass filter may reduce such prediction noise but adds a lag to the prediction results, thus ultimately increasing the prediction error.

V. VELOCITY-AWARE MODEL SWITCHING

We propose a soft switching scheme for a single prediction model that is based on the level of user motion as represented by the measured head velocity, which we call *velocity-aware switching (VAS)*. VAS integrates the output of a prediction model with the output in the NOP case, weighting each output based on the velocity. Let $\hat{\mathbf{p}}_{ml}$ be the output of an ML model, and let $\hat{\mathbf{p}}_{vas}$ be the integrated output from that ML model and

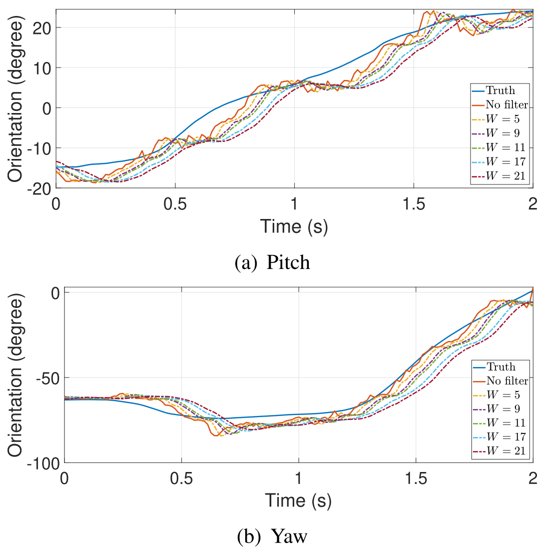


FIGURE 9. SG-filtered output (300 ms).

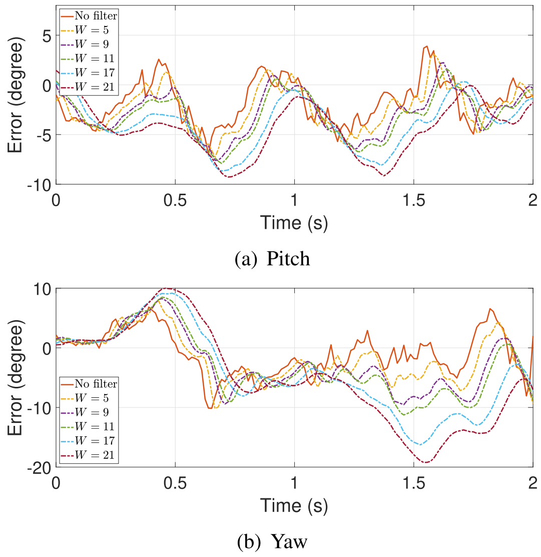


FIGURE 10. Error with the SG filter (300 ms).

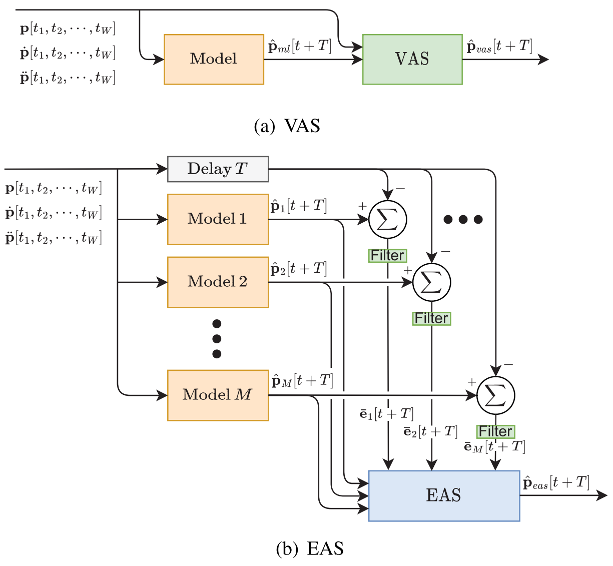

$\mathbf{p}[t_1, t_2, \cdots, t_W]$
$\dot{\mathbf{p}}[t_1, t_2, \cdots, t_W]$
$\ddot{\mathbf{p}}[t_1, t_2, \cdots, t_W]$
Model
$\hat{\mathbf{p}}_{ml}[t+T]$
VAS
$\hat{\mathbf{p}}_{vas}[t+T]$
(a) VAS
Delay T
Model 1
Model 2
Model M
$\hat{\mathbf{p}}_1[t+T]$
$\hat{\mathbf{p}}_2[t+T]$
$\hat{\mathbf{p}}_M[t+T]$
Filter
$\bar{\mathbf{e}}_1[t+T]$
$\bar{\mathbf{e}}_2[t+T]$
$\bar{\mathbf{e}}_M[t+T]$
EAS
$\hat{\mathbf{p}}_{eas}[t+T]$
(b) EAS

FIGURE 11. Architectures of the proposed switching schemes.

the NOP case as generated by VAS. $\hat{\mathbf{p}}_{vas}$ is expressed as

$$\begin{aligned}\hat{\mathbf{p}}_{vas}[t+T] &= \mathbf{w} \cdot \hat{\mathbf{p}}_{ml}[t+T] + (1-\mathbf{w})\mathbf{p}[t] \\ &= g(\mathbf{v}) \cdot \hat{\mathbf{p}}_{ml}[t+T] + (1-g(\mathbf{v}))\mathbf{p}[t], \qquad (4)\end{aligned}$$

where $\mathbf{w}$ is the weight given to the output of the ML model ($0 \preceq \mathbf{w} \preceq 1$) and is defined as a monotonically increasing function g of the velocity $\mathbf{v}(= \dot{\mathbf{p}})$ that becomes zero when $|\mathbf{v}| = 0$ and approaches one as $|\mathbf{v}|$ approaches infinity. Note that g is an elementwise function and returns a result of the same shape as the argument $\mathbf{v}$. The multiplication in Eq. (4) is also an elementwise operation.

In the following, we develop a sigmoid-based functional form for g:

$$g(\mathbf{v}) = \frac{1}{1+e^{-(\hat{\mathbf{v}}-\hat{\mathbf{v}}_{th})}}, \qquad (5)$$

where $\hat{\mathbf{v}}$ and $\hat{\mathbf{v}}_{th}$ are the normalized velocity and the normalized threshold velocity, respectively. These normalized velocities are obtained by dividing by the average velocity of the training samples, i.e.,

$$\hat{\mathbf{v}} - \hat{\mathbf{v}}_{th} = \frac{\mathbf{v} - \mathbf{v}_{th}}{|\mathcal{D}_t|^{-1} \sum_{i \in \mathcal{D}_t} \mathbf{v}_i}, \qquad (6)$$

where $\mathbf{v}$ and $\mathbf{v}_{th}$ are the velocity and threshold velocity, respectively, before normalization; $\mathcal{D}_t$ is the training dataset; and $\mathbf{v}_i$ is the velocity of sample i in the training dataset. The division in Eq. (6) is an elementwise operation. When $\hat{\mathbf{v}} = \hat{\mathbf{v}}_{th}$, $g(\mathbf{v}) = 0.5$, meaning that the same weight is applied to both the prediction model and the NOP case. As $\hat{\mathbf{v}}$ increases over $\hat{\mathbf{v}}_{th}$, $g(\mathbf{v})$ goes to one, prioritizing the output of the prediction model. As $\hat{\mathbf{v}}$ decreases below $\hat{\mathbf{v}}_{th}$, $g(\mathbf{v})$ goes to zero, applying a higher weight to the NOP output. $\mathbf{v}_{th}$ is determined during training such that VAS achieves the optimal performance for $\mathcal{D}_t$.

VI. ERROR-AWARE MODEL SWITCHING

We also develop an ensemble method for a set of model outputs, called *error-aware switching (EAS)*, which switches among model outputs based on the error statistics of those outputs under the parallel execution of multiple models, including VAS models. We propose two types of EAS, with hard switching (EAS-H) and soft integration (EAS-S) of the model outputs. The operations in the following equations are all elementwise operations.

A. EAS-H

EAS-H selects the output of a single model that achieves the lowest statistical prediction error among the different models. Let $\mathcal{M}$ be the set of models forming the ensemble. The final output $\hat{\mathbf{p}}_{eash}$ of EAS-H is determined as the output of a single model m, denoted by $\hat{\mathbf{p}}_m$, as follows:

$$\begin{aligned}&\hat{\mathbf{p}}_{eash}[t] = \hat{\mathbf{p}}_{\mathbf{m}^*[t]}[t]\\ &\text{s.t.}\\ &\mathbf{m}^*[t] = \arg\min_{m\in\mathcal{M}} \bar{\mathbf{e}}_m[t],\end{aligned} \tag{7}$$

where $\bar{\mathbf{e}}_m$ is the statistical prediction error of model m, obtained using the exponentially weighted moving average (EWMA) as

$$\bar{\mathbf{e}}_m[t] = (1-\alpha)\mathbf{e}_m[t] + \alpha\bar{\mathbf{e}}_m[t-\tau], \tag{8}$$

where α is the weight used for the EWMA calculation.

B. EAS-S

EAS-S integrates the outputs of all models in the ensemble using weights that depend on the prediction error statistics; specifically, a model with a lower statistical prediction error is given a higher weight. The output of EAS-S is obtained as follows:

$$\hat{\mathbf{p}}_{eass}[t] = \sum_{m\in\mathcal{M}} \mathbf{w}_m[t]\hat{\mathbf{p}}_m[t] = \sum_{m\in\mathcal{M}} h(\bar{\mathbf{e}}_m[t])\hat{\mathbf{p}}_m[t], \tag{9}$$

where the weight function h is a monotonically decreasing function that ranges from zero to one subject to the constraint $\sum_{m\in\mathcal{M}} h(\bar{\mathbf{e}}_m[t]) = 1$. One example of a possible h is

$$h(\bar{\mathbf{e}}_m[t]) = \frac{|\bar{\mathbf{e}}_m[t]|^{-k}}{\sum_{m\in\mathcal{M}} |\bar{\mathbf{e}}_m[t]|^{-k}}, \tag{10}$$

where $k(>0)$ is an error exponent that determines the effect of the error of a model on its weight.

VII. PERFORMANCE EVALUATION, ANALYSIS, AND DISCUSSION

A. EXPERIMENTAL SETUP

We used a Meta Quest 2 headset [36] to capture motion data. We used two VR content applications in our performance evaluations: a fruit harvesting game and a dragon riding experience, screenshots of which are exhibited in Fig. 12. In the fruit harvesting game, players pick fruits from trees and place them into a basket. Conversely, the dragon riding experience immerses the user in an aerial adventure atop a flying dragon, surveying war scenes from a bird's-eye view. While the participating users played these VR content applications, head motion data were recorded at 72 Hz into trace files to ensure that all algorithms could be run with the same input data to ensure fair comparisons. We captured motion traces, each approximately four minutes long, from ten users. The first 50% of each trace was used for training, and the rest was used for testing. We considered anticipation times of $T = 50$, 100, 200, and 300 ms. We used $W = 20$ samples, corresponding to a period of 0.28 seconds. Unless otherwise specified, $\mathbf{v}_{th}$ for VAS was set to the 30th percentile value of the training data for each trace, and the error exponent k for EAS was set to one.

Throughout the experiments, we considered multiple ML models to solve the motion prediction problem: an MLP,

(a) Fruit harvesting game

(b) Dragon riding experience

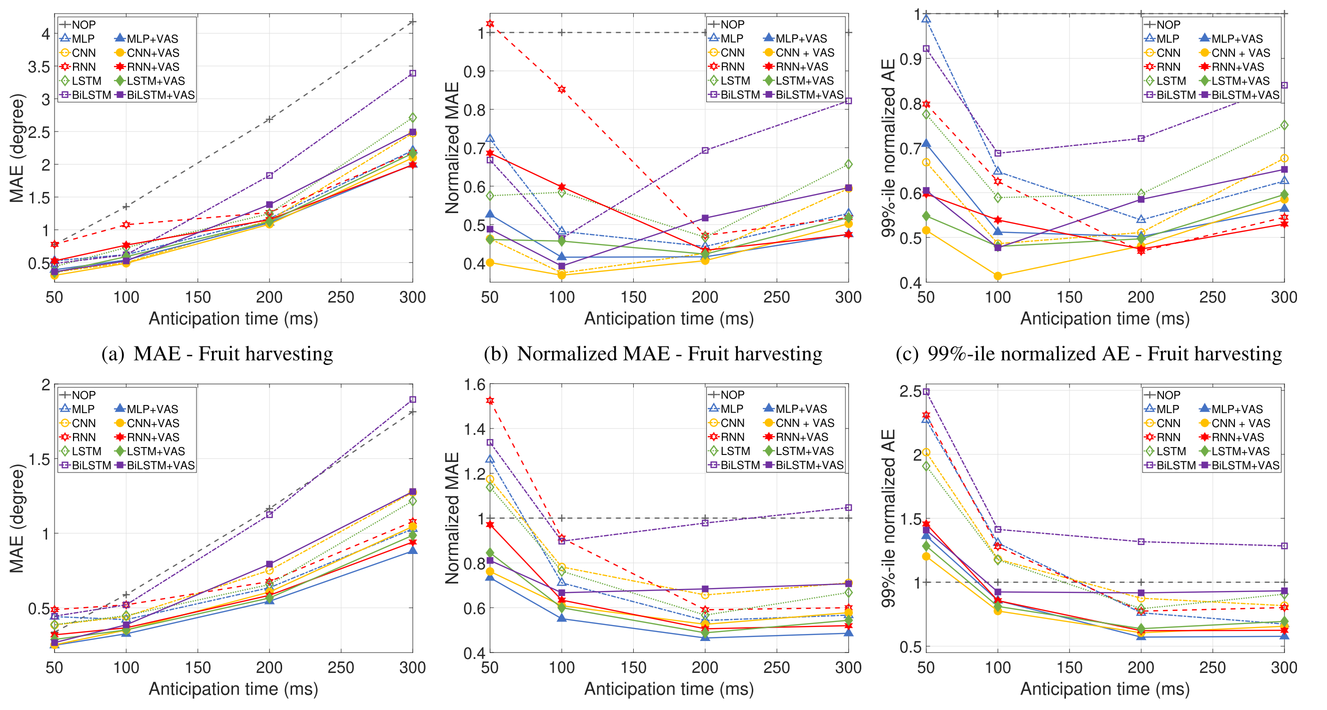
FIGURE 12. VR content applications used in experiments.

a CNN, an RNN, an LSTM network, and a bidirectional LSTM (BiLSTM) network. The parameters of the models were configured as follows. For the MLP model, we used three hidden layers with 30, 20, and 25 neurons in sequence. For the CNN model, we used 30×1 convolutional layers and a fully connected network with two hidden layers consisting of 20 and 25 neurons in the first and second layers, respectively. For the RNN and LSTM models, we used 30 units followed by a hidden layer with 9 neurons. For the BiLSTM model, 512 units and a hidden layer with 9 neurons were used. For the hidden layers of the models, the activation function was ReLU. The optimizer was the Adam optimizer, and the learning rate was set to 0.01.

B. EVALUATION RESULTS FOR VAS SCHEMES

Figs. 13 and 14 show the MAE, normalized MAE, and 99th percentile absolute error (AE) results in the yaw and pitch coordinates, respectively, for the VAS schemes with varying anticipation times. From Figs. 13(a)/(d) and 14(a)/(d), two common observations can be drawn for all schemes. First, the MAE in the yaw direction is higher than that in the pitch direction (e.g., the MAE in the NOP case ranges from approximately 0.8 to 4.2 (0.5 to 1.9) degrees for yaw and from 0.3 to 1.7 (0.2 to 0.8) degrees for pitch in the fruit harvesting game (dragon riding experience)) due to the higher level of yaw motion. Additionally, the results reflect the fact that the fruit harvesting game exhibits a higher level of motion than the dragon riding experience in both directions. Second, as the anticipation time increases, the MAEs of all schemes increase since the future motion becomes less correlated with the previous motion.

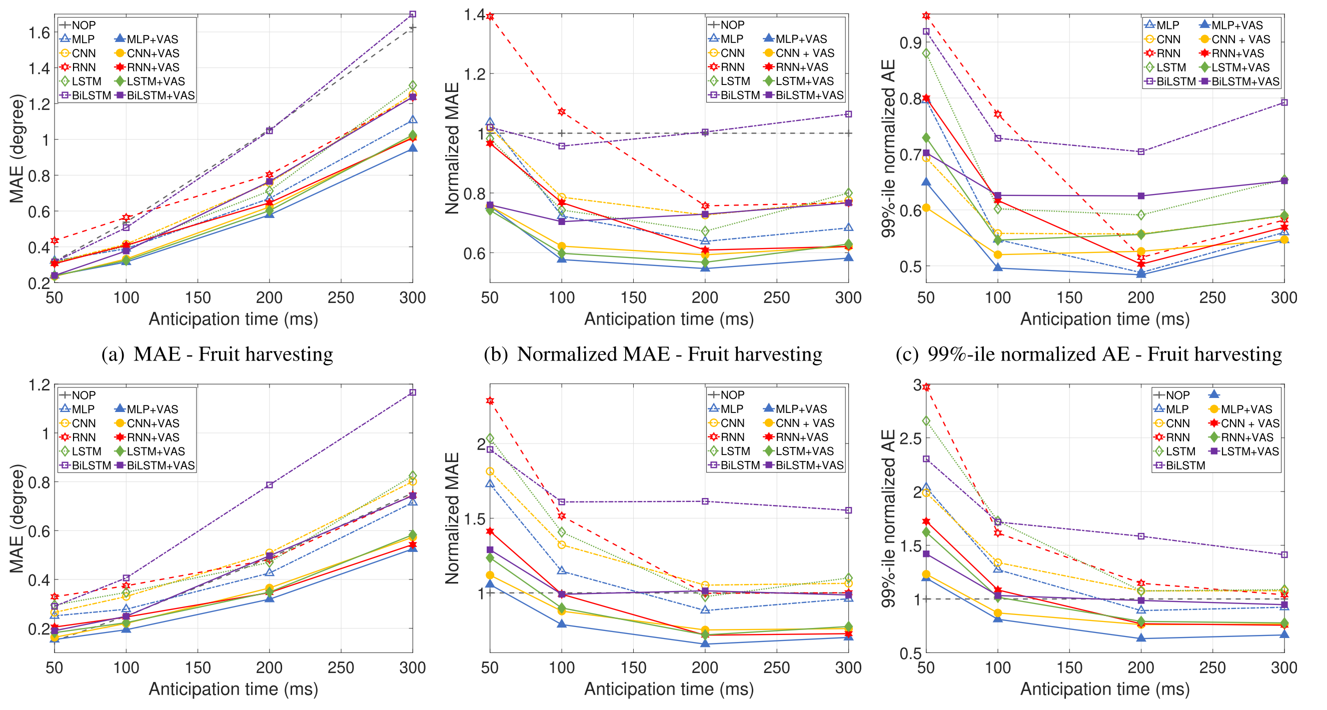
To compare the prediction performance of the different schemes, we first focus on Figs. 13(b) and 14(b), which



(a) MAE - Fruit harvesting (b) Normalized MAE - Fruit harvesting (c) 99%-ile normalized AE - Fruit harvesting

(d) MAE - Dragon riding (e) Normalized MAE - Dragon riding (f) 99%-ile normalized AE - Dragon riding

FIGURE 13. MAE, normalized MAE and 99th percentile normalized AE in the yaw direction vs. anticipation time for VAS schemes.



(a) MAE - Fruit harvesting (b) Normalized MAE - Fruit harvesting (c) 99%-ile normalized AE - Fruit harvesting

(d) MAE - Dragon riding (e) Normalized MAE - Dragon riding (f) 99%-ile normalized AE - Dragon riding

FIGURE 14. MAE, normalized MAE and 99th percentile normalized AE in the pitch direction vs. anticipation time for VAS schemes.

show the MAE normalized with respect to that in the NOP case for the fruit harvesting game. A normalized MAE of one corresponds to the MAE in the NOP case. If the normalized MAE is lower than one, the corresponding scheme performs better than NOP; otherwise, it performs worse. Most schemes outperform NOP, while some of them

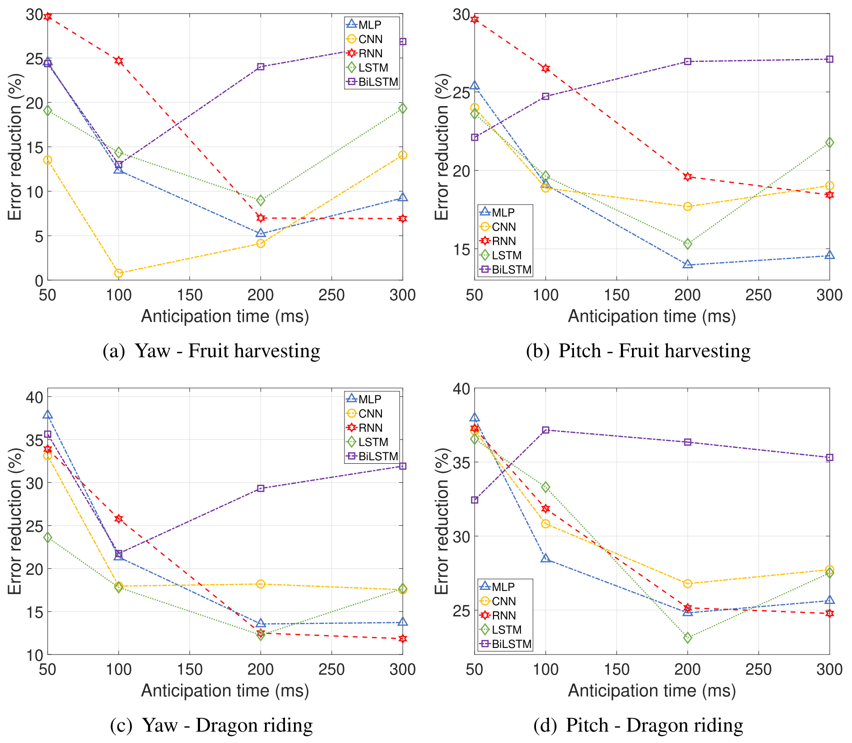


FIGURE 15. Error reduction gain vs. anticipation time for VAS schemes.

(RNN and BiLSTM without VAS) are worse than NOP for some anticipation time cases. For the yaw direction, CNN+VAS is the best overall, while MLP+VAS and RNN+VAS are slightly better than CNN+VAS for long anticipation times. For the pitch direction, MLP+VAS almost always achieves the lowest MAE for all anticipation times. Similar trends are observed in the 99th percentile normalized AE results shown in Figs. 13(c) and 14(c), indicating that MLP+VAS and CNN+VAS are overall the best schemes. These results show that the schemes with VAS generally outperform those without VAS, not only on average but also on most individual samples. In terms of the 99th percentile normalized AE, the prediction models both with and without VAS are always better than NOP. On the other hand, in the case of the dragon riding experience, we observe that for short anticipation times, the normalized MAEs of many schemes are higher than that of NOP, as illustrated in Figs. 13(e) and 14(e). MLP+VAS and CNN+VAS once again emerge as the top-performing schemes, surpassing NOP in all anticipation time cases except for 50 ms in the pitch direction.

The error reduction gains of VAS for each prediction model are shown in Fig. 15. These gains are always positive for all considered prediction models and anticipation times, confirming that VAS is effective in reducing the prediction error for a wide range of prediction models. The gains are especially high for the BiLSTM model since this model alone (without VAS) does not perform well compared to the other models, as shown in Figs. 13 and 14, and thus, there is considerable opportunity for VAS to enhance the model performance. In contrast, the gains of MLP+VAS and CNN+VAS, which are the best VAS schemes among those considered, are relatively low compared to the gains of the other schemes. This is because the MLP and CNN models alone already achieve the lowest MAEs among the models without VAS. Nevertheless, applying VAS to these models still leads to gains of up to approximately 25% for the fruit harvesting game and 38% for the dragon riding experience.

C. EVALUATION RESULTS FOR EAS SCHEMES

To evaluate the prediction performance of the EAS schemes, we consider three types of input models: no-VAS models only, VAS models only (denoted by "-VAS"), and all models (denoted by "-All"). We also consider gradient boosting (denoted by Boosting) as a benchmark ensemble method. To show the gains of EAS over the use of a single VAS scheme, we also include MLP+VAS and CNN+VAS, which were shown above to perform the best overall among the VAS schemes, in the result plots.

Figs. 16 and 17 show the MAE, normalized MAE, and 99th percentile normalized AE results in the yaw and pitch coordinates, respectively, for the EAS schemes with varying anticipation times. Figs. 16(a)/(d) and 17(a)/(d) show that although the EAS schemes achieve different levels of prediction performance with different input types, they always significantly outperform the Boosting schemes. For a detailed comparison among the schemes, we focus

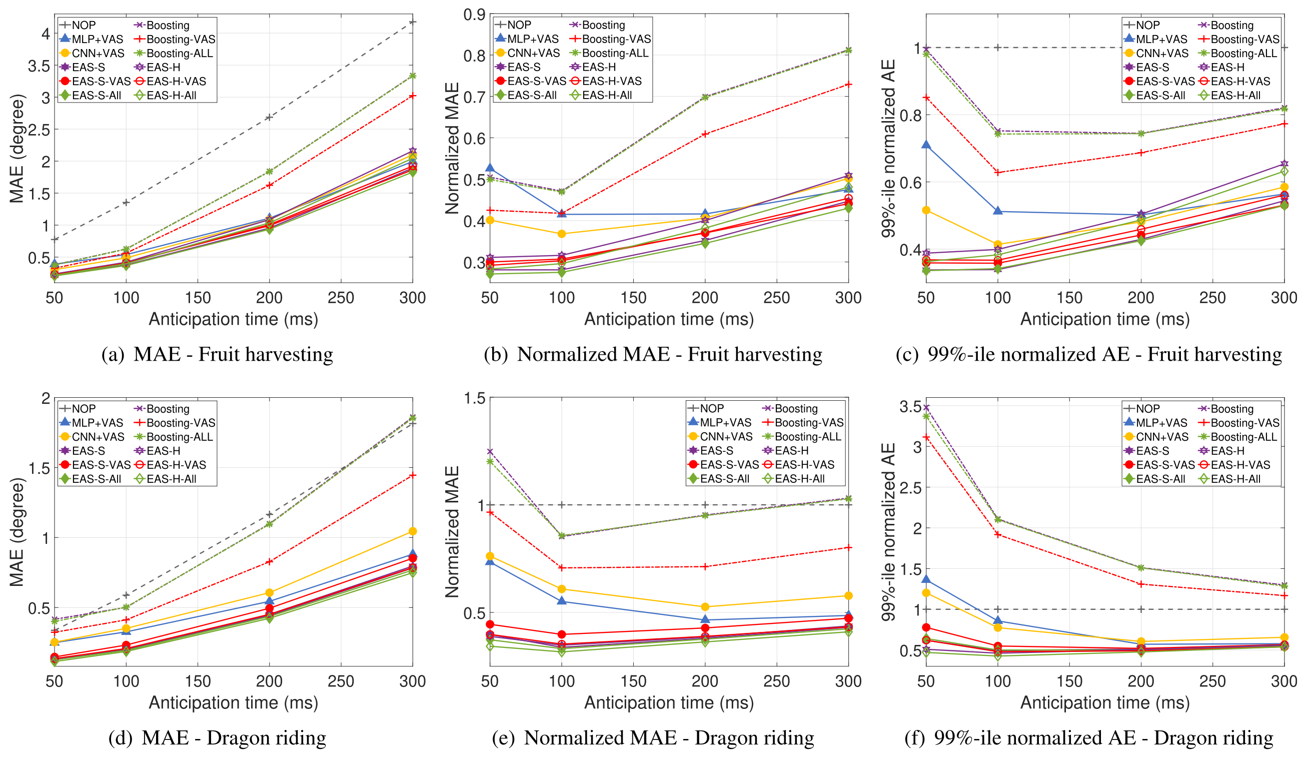


FIGURE 16. **MAE, normalized MAE and 99th percentile normalized AE in the yaw direction vs. anticipation time for EAS schemes.**

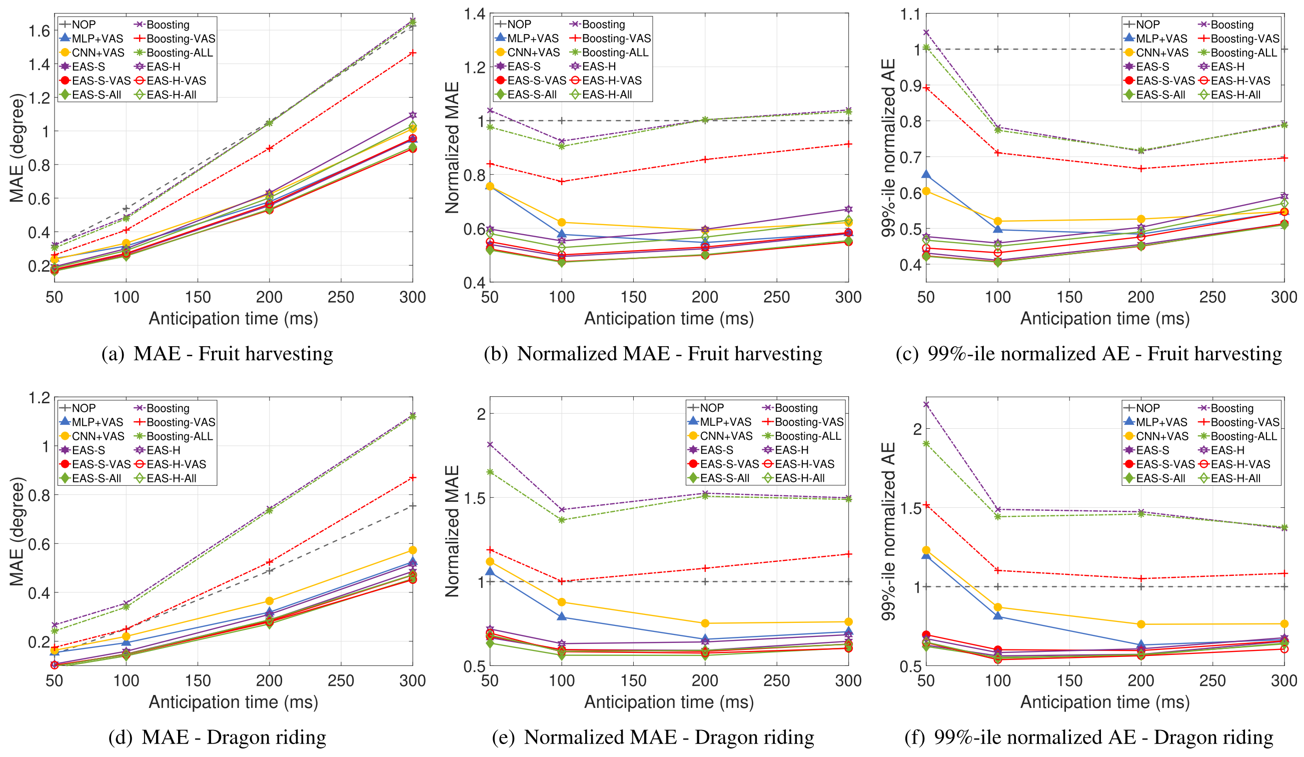


FIGURE 17. **MAE, normalized MAE and 99th percentile normalized AE in the pitch direction vs. anticipation time for EAS schemes.**

on the normalized MAE results shown in Figs. 16(b)/(e) and 17(b)/(e). All EAS schemes outperform MLP+VAS and CNN+VAS for short anticipation times in the fruit harvesting game and for all anticipation times in the dragon riding

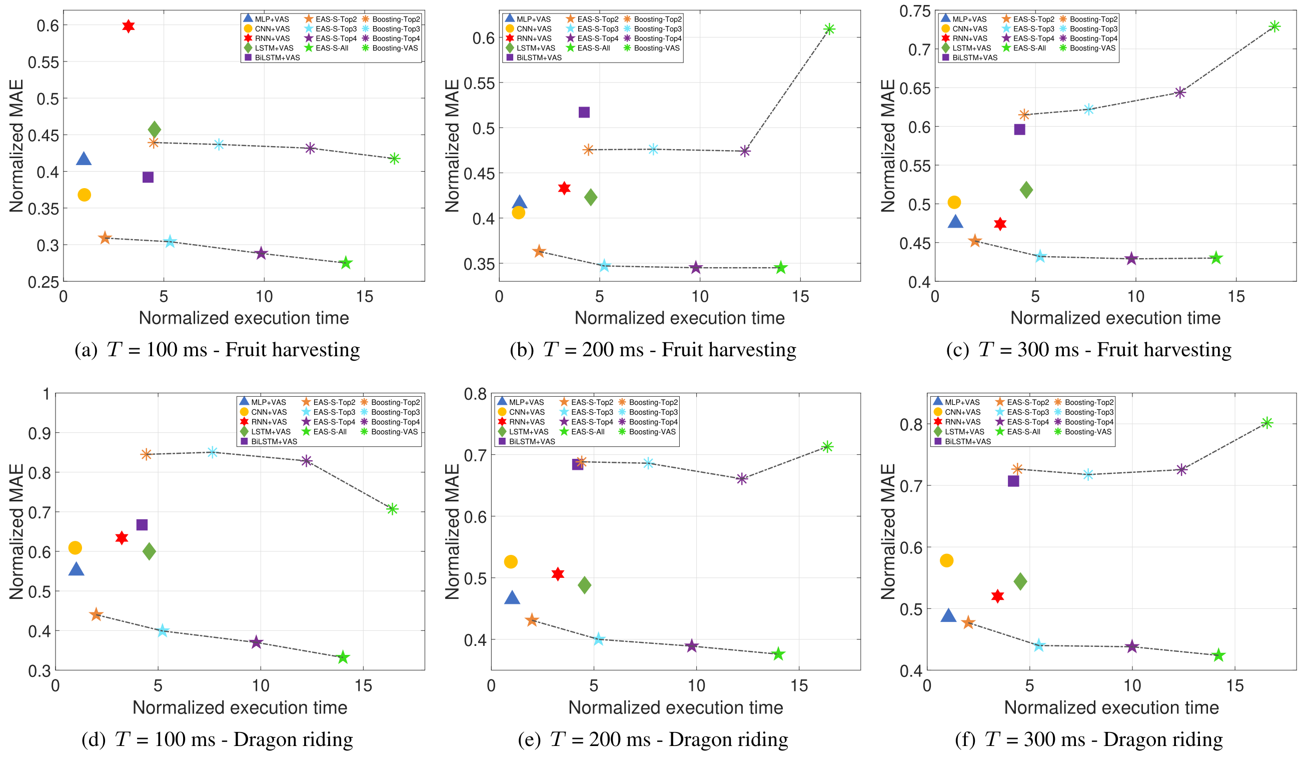


FIGURE 18. **Normalized MAE in the yaw direction vs. normalized execution time for different anticipation times.**

experience. As the anticipation time increases, however, the gap between the EAS schemes and MLP+VAS and CNN+VAS is reduced since the error estimates of EAS are based on older data and thus become less accurate. It is also shown that EAS-S mostly outperforms EAS-H for the same input models. In terms of the 99th percentile normalized AE results shown in Figs. 16(c)/(f) and 17(c)/(f), the gap between the EAS schemes and MLP+VAS and CNN+VAS becomes even larger for short anticipation times, although it is still reduced for longer anticipation times. When both the MAE and 99th percentile AE results in both the yaw and pitch coordinates are considered, EAS-S-All generally performs best, although its gain with respect to other EAS-S schemes is insignificant.

Next, we explore the balance between prediction accuracy and computational demand within the EAS framework. We consider an array of partial prediction combinations for EAS-S, specifically choosing a set number of top-performing models based on their MAE outcomes. The EAS-S approach that integrates the best x prediction models, determined as those with the x lowest MAEs, is denoted by EAS-S-Top$x$. It should be noted that a VAS scheme generates both VAS and non-VAS outputs for the associated prediction model, as illustrated in Fig. 11(a). Therefore, if a VAS scheme is used in EAS-S-Top$x$, we assume that both its VAS and non-VAS outputs are used. The VAS models are sequentially ranked as follows: CNN+VAS, MLP+VAS, RNN+VAS, LSTM+VAS, and BiLSTM+VAS. For instance, EAS-S-Top2 combines CNN+VAS, MLP+VAS, and their non-VAS counterparts. We measure the computational demand of each scheme in terms of its execution time normalized with respect to that of the MLP+VAS scheme, while the prediction accuracy is presented in terms of the normalized MAE.

Figs. 18 (yaw) and 19 (pitch) present scatter plots of the normalized MAE vs. normalized execution time for different anticipation times in the two VR applications. EAS-S-Top2 generally outperforms MLP+VAS and CNN+VAS, thereby establishing the efficacy of the EAS approach even when a limited assortment of prediction models is utilized. The execution time of EAS-S-Top2 is approximately twice that of MLP+VAS but less than half of those of RNN+VAS, LSTM+VAS and BiLSTM+VAS. The prediction accuracy is further improved with EAS-S-Top3 compared to EAS-S-Top2, but this comes at the cost of a significant increase in execution time due to the inclusion of RNN+VAS. Similarly, EAS-S-Top4 and EAS-S-All yield further enhanced prediction accuracy at the expense of increased execution time. In contrast, Boosting does not guarantee improved accuracy as more models are combined. Moreover, EAS-S-Top$x$ consumes significantly less execution time than its counterpart Boosting-Top$x$ for the same value of x, highlighting the efficiency of EAS-S's model-combination mechanism.

The impact of the error exponent k is shown in Fig. 20 for the EAS-S scheme with all input models (including models with and without VAS), which performs the best overall.

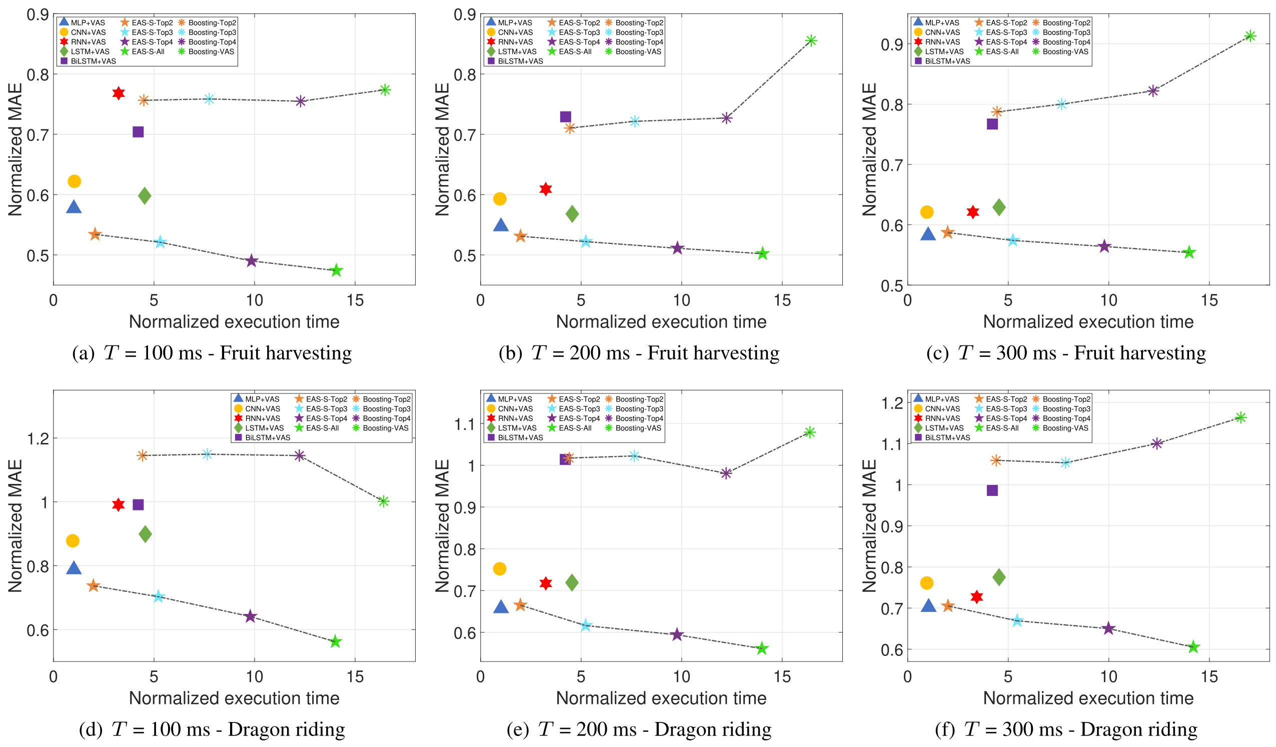


FIGURE 19. Normalized MAE in the pitch direction vs. normalized execution time for different anticipation times.

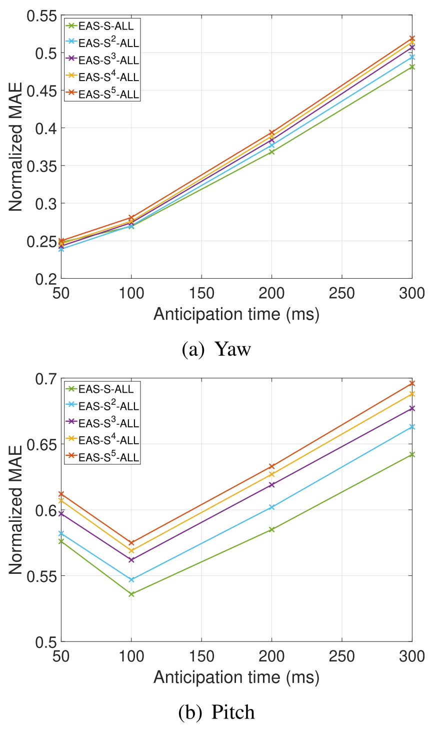


FIGURE 20. Impact of the error exponent on the prediction performance of EAS (in the fruit harvesting game).

For both yaw and pitch, a smaller k results in a lower MAE; note that $k = 1$ was used throughout the previous experiments.

D. IMPACT OF PREDICTION ON VR QUALITY

The image reprojection technique (also known as time warping) [7], [8], [9] mitigates VR motion sickness by reducing inconsistencies between the rendered image and the user's viewport at scan-out. This method adjusts rendered frames to account for head pose changes after the scene is rendered [7]. However, with increased end-to-end latency, larger black borders may appear post-reprojection, compromising user immersion [10], [43]. QoE research [11] has demonstrated that black borders significantly affect the user's gameplay experience, reducing the mean opinion score (MOS) as the presence of black borders increases. Consequently, smaller black border areas in the experimental results reported below imply an enhanced QoE.

We show the impacts of motion prediction on the formation of black borders in Figs. 21 and 22. The NOP case results in black borders occupying 8.2% to 12.5% of the viewport area on average for the considered range of anticipation times. As seen in Fig. 21(b), which shows the black border area normalized with respect to that in the NOP case, motion prediction significantly reduces black border generation, and VAS further mitigates this phenomenon. The improvement gain of VAS over NOP tends to increase with longer anticipation times, reaching 28% for MLP+VAS. The 99th percentile normalized results given in Fig. 21(c) show similar trends. As shown in Figs. 22(a) and (b), the gains of EAS over MLP+VAS and CNN+VAS in terms of the average black border area are not significant; however, the gains become more significant for the 99th percentile normalized black

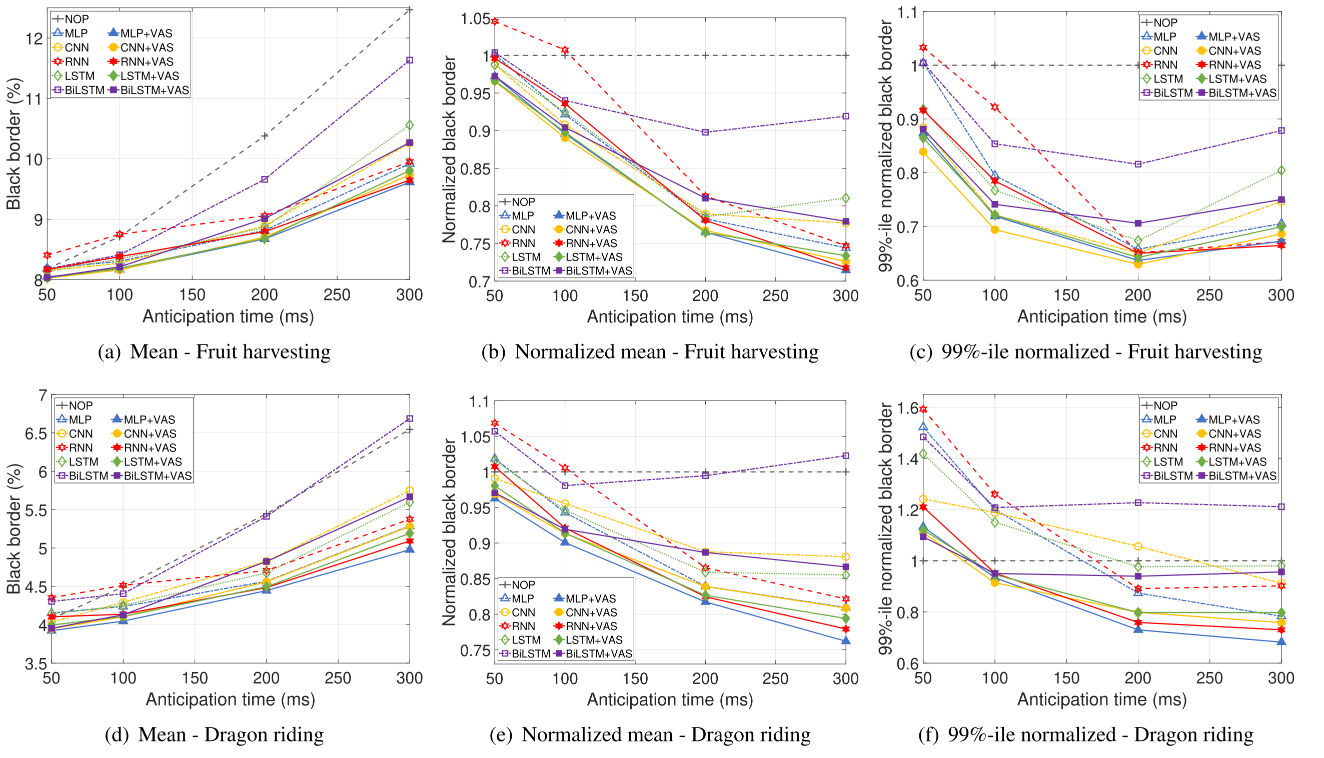


FIGURE 21. Mean, normalized mean and 99th percentile normalized black border areas vs. anticipation time for VAS schemes.

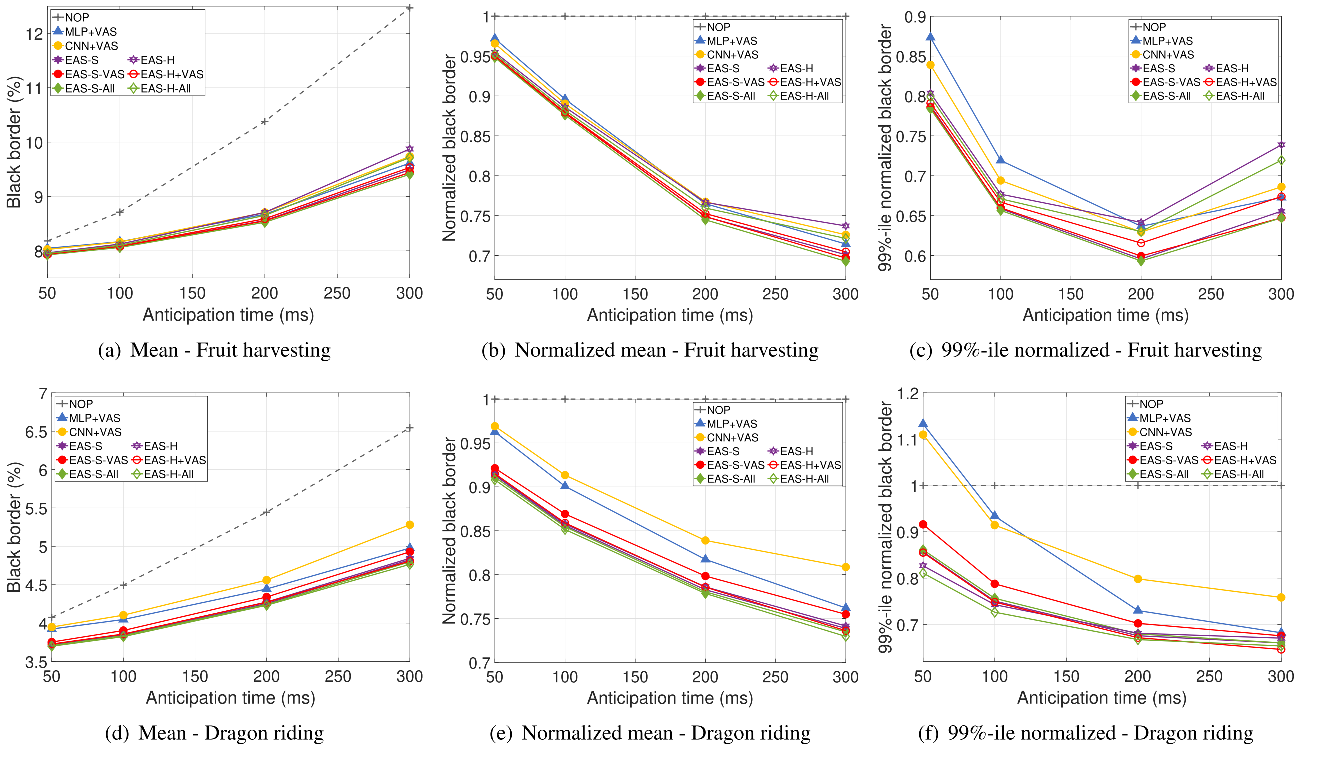
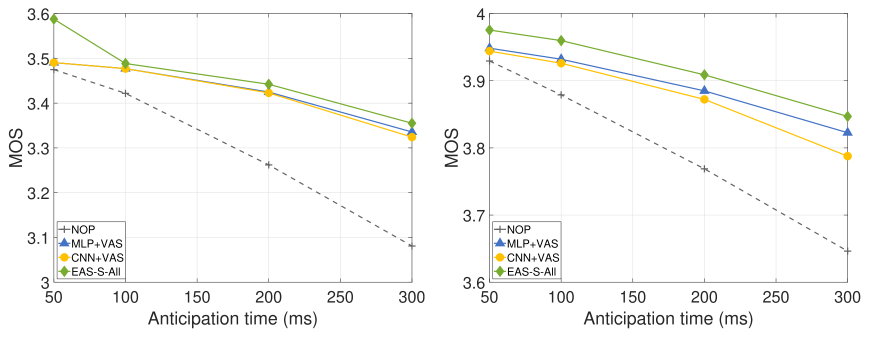


FIGURE 22. Mean, normalized mean and 99th percentile normalized black border areas vs. anticipation time for EAS schemes.

border area, as shown in Fig. 22(c), implying that EAS further enhances the worst-case visual quality of VR services.

We also present the MOS results with and without the proposed prediction schemes in Fig. 23. These results are



(a) Fruit harvesting (b) Dragon riding

FIGURE 23. MOS vs. anticipation time with and without the proposed prediction schemes.

obtained by mapping our black border data to the MOS values (ranging from one to five) from the previously cited QoE study [11]. This figure indicates that the MOS in the NOP case sharply declines as the anticipation time increases due to the expanded presence of black borders within the user's visual range. The results also indicate that our proposed schemes effectively enhance the user experience, as reflected by the higher MOS, reducing the impact of an increased anticipation time on QoE degradation. In the fruit harvesting game, the proposed schemes achieve approximately 9% higher MOS values than NOP for an anticipation time of 300 ms, and the MOS values with the different prediction schemes are comparable. In the dragon riding experience, EAS-S-All outperforms the other schemes, achieving a 5.5% higher MOS than NOP for an anticipation time of 300 ms.

VIII. CONCLUSION

We have developed velocity- and error-aware switching schemes for motion prediction models to solve the latency problem in cloud-assisted VR offloading systems. The proposed schemes are applicable in combination with a wide range of existing motion prediction models and with current VR headsets at no extra hardware cost. First, we considered the chattering problem of ML-based prediction models and the relationship between the head motion velocity and the prediction error gap between ML models and the NOP case. Accordingly, we proposed the VAS scheme, which combines the output of an ML model with the output in the NOP case using a weight determined by the head motion velocity. Next, we developed EAS schemes for integrating a set of outputs from VAS and other models based on the error statistics of the outputs of the individual models. Specifically, EAS-H and EAS-S were proposed for hard switching and soft integration, respectively, of the model outputs. Finally, the results of experimental evaluations demonstrated that on real VR motion traces, the proposed schemes always outperformed conventional ML-based prediction models in terms of prediction accuracy, thereby enhancing the VR visual quality.

Future research can further explore several promising possibilities. These include the fusion of data from multiple sources to improve predictions, integration with predictive rendering techniques, and the development of prediction models optimized to reduce the computational load. Furthermore, a detailed user experience study employing a variety of measurement tools, such as the Simulator Sickness Questionnaire (SSQ) [44] and the Igroup Presence Questionnaire (IPQ) [45], would be advantageous. Additionally, these approaches could be expanded to incorporate service platforms for augmented and mixed reality.

REFERENCES

[1] S. M. LaValle, *Virtual Reality*. Cambridge, U.K.: Cambridge Univ. Press, 2017.
[2] *Preparing for a Cloud AR/VR Future*, Huawei, Shenzhen, China, 2017.
[3] *Cloud VR Network Solution*, Huawei, Shenzhen, China, 2017.
[4] *Cloud AR/VR*, GSMA, London, U.K., 2019.
[5] B. W. Nyamtiga, A. A. Hermawan, Y. F. Luckyarno, T.-W. Kim, D.-Y. Jung, J. S. Kwak, and J.-H. Yun, "Edge-computing-assisted virtual reality computation offloading: An empirical study," *IEEE Access*, vol. 10, pp. 95892–95907, 2022.
[6] Huawei Technologies. (2019). *Cloud VR User Experience and Evaluation White Paper*. [Online]. Available: https://www.huawei.com/minisite/static/cloud-vr-user-experience-evaluation-white-paper-en.pdf
[7] M. Antonov. *Asynchronous Timewarp Examined*. Accessed: Aug. 24, 2023. [Online]. Available: https://developer.oculus.com/blog/asynchronous-timewarp-examined/
[8] D. Evangelakos and M. Mara, "Extended TimeWarp latency compensation for virtual reality," in *Proc. 20th ACM SIGGRAPH Symp. Interact. 3D Graph. Games*, Feb. 2016, pp. 193–194.
[9] T. C. Nguyen, S. Kim, J.-H. Son, and J.-H. Yun, "Selective timewarp based on embedded motion vectors for interactive cloud virtual reality," *IEEE Access*, vol. 7, pp. 3031–3045, 2019.
[10] Huawei Technologies. (2019). *Cloud VR Black Edge and Network Latency Relationship White Paper*. [Online]. Available: https://www-file.huawei.com/-/media/corporate/pdf/ilab/2019/cloud-vr-blackline-network-delay-en-v1.pdf
[11] J. Song, X. Mao, and F. Yang, "The impact of black edge artifact on QoE of the FOV-based cloud VR services," *IEEE Trans. Multimedia*, early access, Dec. 26, 2022, doi: 10.1109/TMM.2022.3232229.
[12] S. M. LaValle, A. Yershova, M. Katsev, and M. Antonov, "Head tracking for the oculus rift," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2014, pp. 187–194.
[13] Y. Bao, H. Wu, T. Zhang, A. A. Ramli, and X. Liu, "Shooting a moving target: Motion-prediction-based transmission for 360-degree videos," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2016, pp. 1161–1170.
[14] Y. Bao, T. Zhang, A. Pande, H. Wu, and X. Liu, "Motion-prediction-based multicast for 360-degree video transmissions," in *Proc. 14th Annu. IEEE Int. Conf. Sens., Commun., Netw. (SECON)*, Jun. 2017, pp. 1–9.

[15] Y. Bao, H. Wu, A. A. Ramli, B. Wang, and X. Liu, "Viewing 360 degree videos: Motion prediction and bandwidth optimization," in *Proc. IEEE 24th Int. Conf. Netw. Protocols (ICNP)*, Nov. 2016, pp. 1–2.
[16] W. F. Satrya and J.-H. Yun, "Combining model-agnostic meta-learning and transfer learning for regression," *Sensors*, vol. 23, no. 2, p. 583, Jan. 2023.
[17] X. Liu and Y. Deng, "Learning-based prediction, rendering and association optimization for MEC-enabled wireless virtual reality (VR) networks," *IEEE Trans. Wireless Commun.*, vol. 20, no. 10, pp. 6356–6370, Oct. 2021.
[18] C.-L. Fan, J. Lee, W.-C. Lo, C.-Y. Huang, K.-T. Chen, and C.-H. Hsu, "Fixation prediction for 360° video streaming in head-mounted virtual reality," in *Proc. 27th Workshop Netw. Operating Syst. Support Digit. Audio Video*, Jun. 2017, pp. 67–72.
[19] M. Rondon, L. Sassatelli, R. Aparicio-Pardo, and F. Precioso, "TRACK: A new method from a re-examination of deep architectures for head motion prediction in 360° videos," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 9, pp. 5681–5699, Sep. 2022.
[20] X. Liu, X. Li, and Y. Deng, "Learning-based prediction and proactive uplink retransmission for wireless virtual reality network," *IEEE Trans. Veh. Technol.*, vol. 70, no. 10, pp. 10723–10734, Oct. 2021.
[21] X. Hou, S. Dey, J. Zhang, and M. Budagavi, "Predictive view generation to enable mobile 360-degree and VR experiences," in *Proc. Morning Workshop Virtual Reality Augmented Reality Netw.*, Aug. 2018, pp. 20–26.
[22] X. Liu, Y. Deng, C. Han, and M. D. Renzo, "Learning-based prediction, rendering and transmission for interactive virtual reality in RIS-assisted terahertz networks," *IEEE J. Sel. Areas Commun.*, vol. 40, no. 2, pp. 710–724, Feb. 2022.
[23] Y. Barniv, M. Aguilar, and E. Hasanbelliu, "Using EMG to anticipate head motion for virtual-environment applications," *IEEE Trans. Biomed. Eng.*, vol. 52, no. 6, pp. 1078–1093, Jun. 2005.
[24] S. Polak, Y. Barniv, and Y. Baram, "Head motion anticipation for virtual-environment applications using kinematics and EMG energy," *IEEE Trans. Syst., Man, Cybern., A, Syst. Hum.*, vol. 36, no. 3, pp. 569–576, May 2006.
[25] S. Gül, S. Bosse, D. Podborski, T. Schierl, and C. Hellge, "Kalman filter-based head motion prediction for cloud-based mixed reality," in *Proc. 28th ACM Int. Conf. Multimedia*, Oct. 2020, pp. 3632–3641.
[26] J. J. LaViola, "Double exponential smoothing: An alternative to Kalman filter-based predictive tracking," in *Proc. workshop Virtual environments*, May 2003, pp. 199–206.
[27] T. C. Nguyen and J.-H. Yun, "Predictive tile selection for 360-degree VR video streaming in bandwidth-limited networks," *IEEE Commun. Lett.*, vol. 22, no. 9, pp. 1858–1861, Sep. 2018.
[28] A. D. Aladagli, E. Ekmekcioglu, D. Jarnikov, and A. Kondoz, "Predicting head trajectories in 360° virtual reality videos," in *Proc. Int. Conf. 3D Immersion (IC3D)*, Dec. 2017, pp. 1–6.
[29] Y. Zhu, G. Zhai, and X. Min, "The prediction of head and eye movement for 360 degree images," *Signal Process., Image Commun.*, vol. 69, pp. 15–25, Nov. 2018.
[30] N. Stein, G. Bremer, and M. Lappe, "Eye tracking-based LSTM for locomotion prediction in VR," in *Proc. IEEE Conf. Virtual Reality 3D User Interface (VR)*, Mar. 2022, pp. 493–503.
[31] A. Satriawan, A. A. Hermawan, Y. F. Luckyarno, and J.-H. Yun, "Predicting future eye gaze using inertial sensors," *IEEE Access*, vol. 11, pp. 67482–67497, 2023.
[32] K. Lee, D. Chu, E. Cuervo, J. Kopf, Y. Degtyarev, S. Grizan, A. Wolman, and J. Flinn, "Outatime: Using speculation to enable low-latency continuous interaction for mobile cloud gaming," in *Proc. 13th Annu. Int. Conf. Mobile Syst., Appl., Services*, May 2015, pp. 151–165.
[33] C. Ebner, S. Mori, P. Mohr, Y. Peng, D. Schmalstieg, G. Wetzstein, and D. Kalkofen, "Video see-through mixed reality with focus cues," *IEEE Trans. Vis. Comput. Graph.*, vol. 28, no. 5, pp. 2256–2266, May 2022.
[34] Y. Liu, B. Han, F. Qian, A. Narayanan, and Z.-L. Zhang, "Vues: Practical mobile volumetric video streaming through multiview transcoding," in *Proc. 28th Annu. Int. Conf. Mobile Comput. Netw.*, Oct. 2022, pp. 514–527.
[35] Oculus SDK Developer Guide. *Initialization and Sensor Enumeration*. Accessed: Aug. 24, 2023. [Online]. Available: https://developer.oculus.com/documentation/native/pc/dg-sensor/
[36] Meta. (2022). *Quest VR Headset*. Accessed: Aug. 24, 2023. [Online]. Available: https://www.meta.com/kr/en/quest/products/quest-2/
[37] I. F. Akyildiz and H. Guo, "Wireless communication research challenges for extended reality (XR)," *ITU J. Future Evolving Technol.*, vol. 3, no. 2, pp. 273–287, 2022.
[38] *ZeroMQ: An Open-Source Universal Messaging Library*. Accessed: Aug. 24, 2023. [Online]. Available: https://zeromq.org/
[39] *Azure Network Round-Trip Latency Statistics*. Accessed: Aug. 24, 2023. [Online]. Available: https://learn.microsoft.com/en-us/azure/networking/azure-network-latency
[40] *Naver Cloud Platform Global Latency Status*. Accessed: Aug. 24, 2023. [Online]. Available: https://www.ncloud.com/product/global/globalLatencyStatus
[41] *AWS Latency Monitoring*. Accessed: Aug. 24, 2023. [Online]. Available: https://www.cloudping.cloud/aws
[42] B. Charyyev, E. Arslan, and M. H. Gunes, "Latency comparison of cloud datacenters and edge servers," in *Proc. IEEE Global Commun. Conf.*, Dec. 2020, pp. 1–6.
[43] D. T. Tan, S. Kim, and J.-H. Yun, "Enhancement of motion feedback latency for wireless virtual reality in IEEE 802.11 WLANs," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2019, pp. 1–6.
[44] R. S. Kennedy, N. E. Lane, K. S. Berbaum, and M. G. Lilienthal, "Simulator sickness questionnaire: An enhanced method for quantifying simulator sickness," *Int. J. Aviation Psychol.*, vol. 3, no. 3, pp. 203–220, Jul. 1993.
[45] V. Schwind, P. Knierim, N. Haas, and N. Henze, "Using presence questionnaires in virtual reality," in *Proc. CHI Conf. Hum. Factors Comput. Syst.*, May 2019, pp. 1–12.

AIRLANGGA ADI HERMAWAN received the B.S. degree in electrical engineering from Gadjah Mada University, Yogyakarta, Indonesia, and the M.S. degree in computer science and engineering from the Eindhoven University of Technology, Eindhoven, The Netherlands. He is currently pursuing the Ph.D. degree in electrical and information engineering with the Seoul National University of Science and Technology, Seoul, South Korea. His current research interest includes virtual reality systems.

YAKUB FAHIM LUCKYARNO received the B.S. degree in engineering physics from Gadjah Mada University, Yogyakarta, Indonesia, and the M.S. degree in computer engineering from the King Mongkut's Institute of Technology Ladkrabang, Bangkok. He is currently pursuing the Ph.D. degree in electrical and information engineering with the Seoul National University of Science and Technology, Seoul, South Korea. His current research interest includes virtual reality systems.

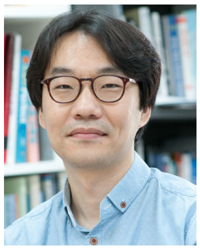
ISFAN FAUZI received the B.S. degree in electrical engineering from the Bandung Institute of Technology (ITB), Indonesia. He is currently pursuing the M.S. degree in electrical and information engineering with the Seoul National University of Science and Technology, Seoul, South Korea. His current research interest includes virtual reality systems.

DEREK KWAKU POBI ASIEDU (Member, IEEE) received the B.S. degree in biomedical engineering from the University of Ghana, Legon, Ghana, in 2011, and the M.S. and Ph.D. degrees in electronic engineering from Hanbat National University, Daejeon, South Korea. He was a Postdoctoral Fellow with the Wireless Intelligent Systems Laboratory, Hanbat National University, from September 2019 to December 2020. Afterwards, he has been with the Department of Electrical and Information Engineering, Seoul National University of Science and Technology, Seoul, South Korea, as a Research Assistant Professor, since May 2021. His research interests include machine learning-based and model-based applications in next-generation wireless communication, radio frequency wireless information and power transfer communication, and the symbiotic cognitive radio in Internet of Things networks.

TAE-WOOK KIM received the B.S. degree in computer science from the Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea, in 2007. He is currently a Software Engineer with Clicked Inc., Seoul, South Korea. His current research interests include real-time graphic rendering and streaming for virtual reality.

DEOK-YOUNG JUNG received the bachelor's and master's degrees in communication design from the College of Fine Arts, Hongik University, Seoul, South Korea, in 2000 and 2003, respectively. In 2002, he was with Bbox studio, a computer graphics studio for movies and commercial films. He won the Special Effects Award at the Daejong Film Festival in Korea for the film "Faceless Beauty." In 2004, he founded Computer Graphics Studio B1 for movies and games. In 2007, he established Studio Varsia, a mobile game developer. In addition, he established Clicked Inc., in 2013, and the CEO. Clicked is currently focusing on developing a number of XR contents and WiFi-based XR cloud streaming solutions.

JIN SAM KWAK received the B.S., M.S., and Ph.D. degrees in electrical engineering and computer science from Seoul National University, Seoul, South Korea, in 1998, 2000, and 2004, respectively. From 2004 to 2005, he was a Postdoctoral Research Associate with the School of Electrical and Computer Engineering, Georgia Institute of Technology. In 2006, he was also with the Wireless Networks and Communications Group (WNCG), The University of Texas at Austin, as a Postdoctoral Research Fellow. From 2007 to 2012, he was with LG Electronics as a Chief Research Engineer. Since 2012, he has been with the WILUS Institute of Standards and Technology, where he is currently the CEO and the Co-Founder. His research interests include next generation standards-driven technologies for LTE, (B)5G, Wi-Fi, and multimedia codec. During this time, he carried out research tasks on the IMT-advanced & its beyond and led the standards activities for wireless communications in IEEE 802 (especially IEEE 802.11/15/16/19), Wi-Fi Alliance (served as an Alternative Board Member), and WiMAX Forum.



JI-HOON YUN (Senior Member, IEEE) received the B.S. degree in electrical engineering and the M.S. and Ph.D. degrees in electrical engineering and computer science from Seoul National University (SNU), Seoul, South Korea, in 2000, 2002, and 2007, respectively.

He is currently a Professor with the Department of Electrical and Information Engineering, Seoul National University of Science and Technology (SeoulTech), Seoul. Before joining SeoulTech, in 2012, he was with the Department of Computer Software Engineering, Kumoh National Institute of Technology, as an Assistant Professor. He was a Postdoctoral Researcher with the Real-Time Computing Laboratory, The University of Michigan, Ann Arbor, MI, USA, in 2010 and a Senior Engineer with the Telecommunication Systems Division, Samsung Electronics, Suwon, South Korea, from 2007 to 2009. His current research interests include wireless multimedia systems and mobile applications.