NAIST-IS-DD1561022

Doctoral Dissertation

Improving and Expanding Gaming Experiences based on Cloud Gaming

Kar Long Chan

February 19, 2018

Graduate School of Information Science
Nara Institute of Science and Technology

A Doctoral Dissertation
submitted to Graduate School of Information Science,
Nara Institute of Science and Technology
in partial fulfillment of the requirements for the degree of
Doctor of ENGINEERING

Kar Long Chan

Thesis Committee:

Professor Hajimu Iida	(Supervisor)
Professor Hirokazu Kato	(Co-supervisor)
Associate Professor Kohei Ichikawa	(Co-supervisor)
Associate Professor Yasuhiro Watashiba	(Co-supervisor)

Improving and Expanding Gaming Experiences based on Cloud Gaming*

Kar Long Chan

Abstract

In gaming industry, Cloud Gaming is a new form of gaming service on trend trying to serve millions of players around the world with novel gaming experience. It aims to provide high-quality gaming service at any device, including thin clients which are incapable of handling high-definition gaming softwares. Players are only required to use any device that can connect to cloud servers for receiving streaming data through network and display game contents. Ideally Cloud Gaming is a promising service of providing novel gaming experience but in reality, various technological barriers make Cloud Gaming not comprehensively feasible for every type of game, such as first person shooting game which requires fast responsiveness. In addition, most existing cloud service, which streams encoded video sequence back to the client, is difficult to catch up with the rising demands for graphic quality. In this dissertation, we aim at addressing both of the above issues by providing solutions that could potentially improve user's experience of playing games delivered on Cloud Gaming, as well as expanding the usage of Cloud Gaming on new generation gaming experiences. First, for addressing the graphics quality issue, we propose a Hybrid-Streaming System that takes the respective benefits from Cloud Gaming and traditional gaming to provide highly-accessible gaming with close-to-original graphics quality. The system distributes rendering operations to game player's PC and Cloud server to achieve the desired improvement by utilising graphics processing power from both sides. Quantitative result shows graphics quality's improvement of the proposed system over traditional Cloud

*Doctoral Dissertation, Graduate School of Information Science,
Nara Institute of Science and Technology, NAIST-IS-DD1561022, February 19, 2018.

Gaming system while maintaining acceptable network bandwidth consumption. Moreover, since rendering tasks are reasonably distributed, server's workload is mitigated.

Furthermore, we explore methods that could make Cloud Gaming available for VR gaming, which comparatively has stricter latency requirement. As such, we propose utlizing Recurrent-Neural-Network-based Head-motion prediction model to compensate the inevitable latency issue in a Cloud Gaming environment and the random nature of player's head motion. Our results show that with an assumption of a normal Cloud Gaming environment at 150ms latency, not only the model could well predict head-motion within the period, but also it could fit well with different player's motion.

Keywords:

Cloud Gaming, Graphics Quality, Hybrid Streaming, VR gaming, Prediction Model, Head-Motion

Contents

List of Figures

1 Introduction

1.1 Background

Video Game industry has been considered as an essential sector in the media industry, joining with the other two sectors including the movie industry and the music industry. The global market of video game is growing at a rapid rate, with an expectation of the market worth rising from $67 billions in 2013 to $82 billions in 2017. In order to dynamically adapt to the ever changing and highly competitive business environment, video game industry is rennovating their services with the help of cloud technology. Efficiently utilising large amount of data through widely deployed data centre helps to form the diversity of video game industry nowadays. Forms of video game business utilising cloud technology include gaming services for social usage, which can be recognised by many existing Massively Multiplayer Online Role-Playing Games (MMORPG) such as World of Warcraft * and Finally Fantasy XIV †, together with browser-based games provided by social network based services. Another cloud-based video game business can refer to game contents delivery or distribution service, as Steam ‡ operated by Valve Corporation§ is among the most famous platforms in this segment of video game business. Cloud Gaming is an uprising solution that aims at providing compute-intensive games, which usually require high-quality rendering in real-time, to any device in anywhere. It is a relatively new business on trend making rapid development in term of technology and scale, and an estimation of reaching 8 billion US dollars by 2017 shows the evidence of tremendous market potential of cloud gaming.

*World of Warcraft: http://us.battle.net/en/int?r=wow
†Final Fantasy XIV: http://jp.finalfantasyxiv.com/lodestone/
‡Steam: http://store.steampowered.com/
§Valve Corporation: http://www.valvesoftware.com/

1.2 Concept of Cloud Gaming

Taking the advantages of utilizing reliable, elastic and high-performance computing resources in cloud infrastructure, Cloud Gaming, also regarded as gaming on demand, is an emerging gaming service that envisions a future of novel gaming experiences for millions of game players. Cloud Gaming shifts heavy workloads of game processing from player's device to a powerful cloud server. Within the overall structure, the actual interactive gaming applications are stored at the server and get executed once requested by the players. The resulted game scenes, which are rendered through the server's GPU, are streamed back to the player's device as an encoded video over a network with sufficient bandwidth, such as 5Mbps recommended by OnLive[¶] [1]. The method of streaming game contents as video sequences is called image-based streaming. Another method, which is called instruction-based streaming, delivers game contents by streaming 3D commands to the client's side, as such the game contents are locally rendered and shown on client's display. At the client's side, control events from mice, keyboards, joysticks and other types of input devices are accurately recorded and transmitted from the client's machine back to the cloud servers for corresponding manipulation of game logics. A brief structure of Cloud Gaming is shown at Figure 1.1.

The On demand nature of Cloud Gaming draws significant attention from both clients and game developers for various reasons. As for the benefits of clients, Cloud Gaming frees clients from the complication of game software installation and management of compatibility with hardwares. With the mere requirement of thin client, which is a device being able to connect to network and display the received game scenes, clients are granted with more different choices of platforms including PCs, laptops, tablets and smart phones to play games. In addition, clients could pay less costs for more game choices. In term of advantages gained by game developers, Cloud Gaming eases the possible incompatibility issues between hardwares and softwares. Therefore, developers are easier to adapt their game softwares to more different platforms, thus decreasing the production costs and increasing the net revenues. Features of Cloud Gaming envision a promising future of providing million clients with novel gaming experience, as it has been an

[¶]OnLive: https://onlive.com/

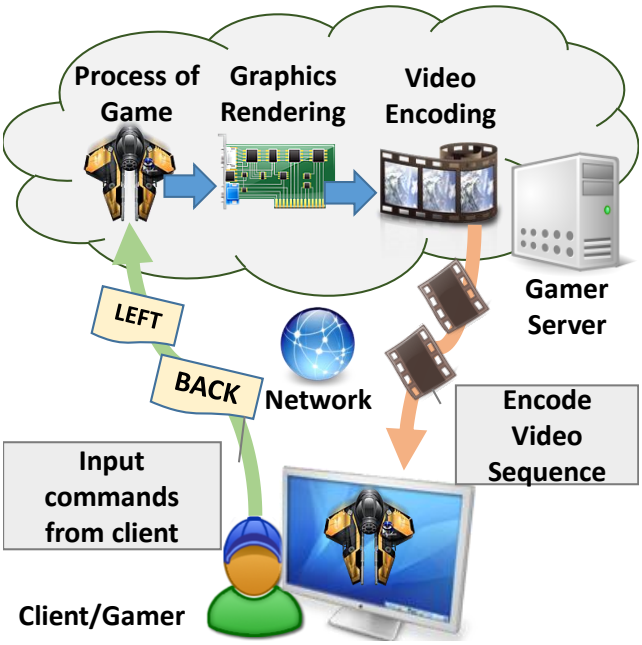


Figure 1.1: Brief Structure of Cloud Gaming

active topics both in industries and research fields recently. OnLive [2] is a leading Cloud Gaming service provider based in San Francisco USA providing gaming softwares from over 50 publishers. Sony's acquisition of Cloud Gaming company Gaikai [3] in the year of 2012 shows great interests toward Cloud Gaming from leading media enterprises. GamingAnyWhere [4], developed by Chen et al, is an open-source Cloud Gaming platform available for the use of research.

1.3 Conducted Evaluations of Cloud Gaming Service

With all the potential benefits, it has been a recent active topic both in industries and research fields. Sony Entertainment's[‖] acquisition of Cloud Gaming companies of Gaikai[**] in 2012 [5] and OnLive in 2015 [6] shows great interests toward Cloud Gaming from leading gaming enterprise. On the other hand, GamingAnyWhere [4]

[‖]Sony Computer Entertainment Inc: http://www.scei.co.jp/
[**]Gaikai: https://www.gaikai.com/

is an open-source Cloud Gaming platform available for the use of research. With a forecast of reaching 8 millions US dollars market worth by 2017 [7], Cloud Gaming proves to be a new area with tremendous market potential.

As an ideal concept, Cloud Gaming is considered as a game-changer of future gaming experience. However, network constraints inevitably introduce significant challenges that hinder Cloud Gaming from being competitive with traditional gaming platforms. Particularly, responsiveness and graphics quality have been the long term factors affecting player's Quality of Experience(QoE) in Cloud Gaming [8, 9]. Compared to conventional live video feeding, more rigid real-time responsiveness is required in Cloud Gaming in order to maintain good enough QoE [4, 10]. Furthermore, previous findings show that players are also sensitive to changes in graphics quality as well [11]. For improving QoE in Cloud Gaming, wide ranges of ideas are proposed to address the respective issues such as alleviating response latency [12, 13] or sending more reliable and higher quality video [14]. Moreover, OnLive and Gaikai, as two notable Cloud Gaming companies prior to their acquisitions by Sony Entertainment, were reasonably successful to compromise network constraints with their own technologies and delivered relatively good gaming service [15]. However, the actual delivered gaming quality is still nowhere close to match with the same gaming application being locally processed by traditional PC platforms or major game consoles. The newtork constraints nature of Cloud Gaming also hinder the concept from being applied to new gaming experenice, such as VR/AR gaming.

1.4 Research Motivation

At current stage, Cloud Gaming is an intriguing technology that likely satisfies casual gaming demands. However, it is still by far an unsatisfactory option for gamers, as here referred to game players with high gaming demands and they usually own specified gaming devices. With a strong base of researches being conducted to propel a rapid development, the increasingly matured Cloud Gaming technology will inevitably play more significant role in future gaming environment, together with the concurrently advancing traditional gaming platforms such as PC or game consoles. With this future perspective in mind, the goal of this

research is to bring novel gaming experience by making connection between Cloud Gaming, traditional PC gaming platform and also new generation gaming experience, specifically, VR gaming. First, we aim at drawing more positive attentions from gamers by improving graphics quality delivered on Cloud Gaming, as currently the quality of game scenes streamed as encoded video is degraded from original. Furthermore, utilization of gamer's PC platform in Cloud Gaming can also share workload, thus achieving better efficiency in term of handling more gaming requests at cloud server. Secondly, we propose using Machine Learning Technique to predict player's head motion as a mean to compensate the latency problem in Cloud Gaming environment. As such, it could potentially expand the application of Cloud Gaming on VR gaming.

the goal of this research can be summarized as follows:

- Improving Graphics Quality delievered on Cloud Gaming:
  - Based on existing Cloud Gaming infrastructure, a prototype Hybrid streaming system that distributes rendering operations to server and gamer's PC is proposed.
  - Improve graphics quality of Cloud Gaming by utilizing available rendering power from both server and gamer's PC.
  - Evaluate graphics quality, network bandwidth consumption and GPU usage of the proposed system by comparing with an existing Cloud Gaming platform.
- Creating a prediction model to compensate the latency issue in Cloud Gaming
  - Collect head motion data and build three different kinds of prediction models.
  - Evaluate the three models with cross validation.

1.5 Organization of Dissertation

Our dissertation is structured as follows:

- Chapter 2: Proposing an Hybrid-Streaming systed for improving graphics quality delivered on Clud Gaming
  - First we mention the two related techniques that our system is based on.
  - Provide an overview of the Hybrid-Streaming System, followed by the corresponding implementation.
  - Evaluating the Hybrid-Streaming system in terms of graphics quality improvement, incurred server workload and network bandwidth consumption.
  - Discussing some limitations of the proposed system
- Chapter 3: Head-Motion Prediction Model for expanding the usage of Cloud Gaming to VR gaming
  - We mention about the rising popularity of VR gaming and factor that hinders it from reaching more customers.
  - Propose using Cloud Gaming to improve the accessibility of VR gaming. But multiple difficulties need to be addressed. Our work focus on addressing the latency issues.
  - Introduce Some related works about providing VR experience remotely
  - Propose using RNN and Moving Average to predict head-motion as a mean to compensate the latency issue.
  - Evaluate our results and discuss about our findings

A last we will conclude our works and give an outlook about possible future directions.

2 Research 1: Improving Graphics Quality delievered on Cloud Gaming

a challenging objective of developing a sustainable Cloud Gaming service is to maintain and improve client's Quality of Experience (QoE), as network constraints play critical roles in affecting the system performance [8, 9]. In general, interaction delay and graphic quality are two significant criteria that determine client's QoE. In Cloud Gaming, rigid real-time responsiveness is demanded in order to achieve good enough QoE [4, 10], so currently most related researches focus on alleviating interaction delay [12, 13]. On the other hand, findings from a conducted subject test show that clients are sensitive to changes in graphic quality and smoothness during gameplay, which implies that graphic quality notably affects QoE of Cloud Gaming as well [11]. Furthermore, alongside the advancement of in-game visual effects and high resolution display, client's demands for gaming with more realistic graphics on their devices are uprising. However, in term of graphic quality, there is an obvious gap between traditional local rendering and Cloud Gaming's streaming encoded video, which the graphic quality is degraded from the original.

With this consideration, this study aims to enhance graphic quality delivered on existing Cloud Gaming system. Especially we address the use case of PC, which is not necessary to be a significant powerhouse, but rendering-capable to a certain extent. Based on the two existing streaming methods which will be introduced in later section, our approach is to allocate available rendering power at client's side to achieve the improvement of graphic quality. Furthermore, the distribution of rendering tasks also mitigates server's workload as well.

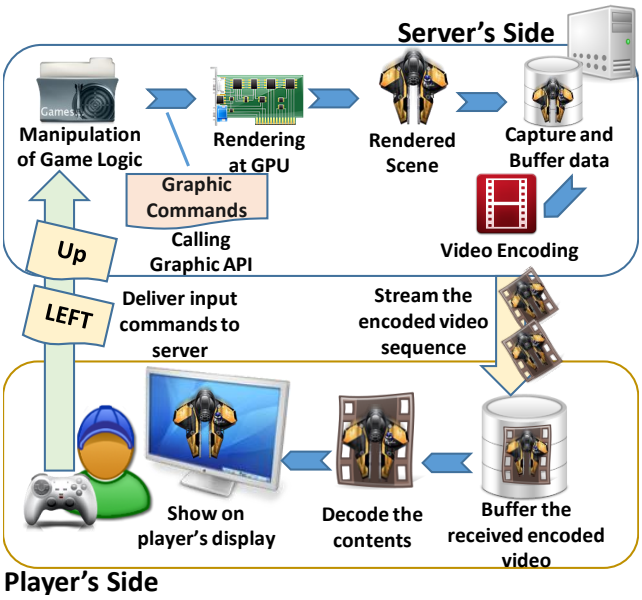


Figure 2.1: Structure of Image-based Streaming

2.1 Related Techniques

Cloud gaming system can be regarded as a type of real-time remote rendering system, as the technology is similar to application such as Remote Desktop. However, dedicated video player and encoder are usually specifically designed for the use of cloud gaming, which guarantee an environment to handle more rigid real-time response. In general, the streaming approaches of cloud gaming system can be categorised into two types, image-based streaming and instruction-based streaming. These two methods differ from each other in how the game contents are streamed from the server to the client.

2.1.1 Image-based streaming

As shown in Figure 2.1, in the Cloud Gaming environment adopting Image-based Streaming, game logics, which drive the progression of gameplay, are manipulated according to client's inputs at the server CPU. Afterwards, 3D graphics rendering operations are processed through a dedicated GPU. The rendered game contents are then captured, encoded into video and streamed to the game player's device.

Upon receiving the encoded video, game player's device decodes the contents and finally shows the corresponding frame on display.

An advantage of Image-based Streaming is streaming game contents as encoded video, since it consumes low bandwidth which is suitable in a typical network environment. Previous research reveals that only 1.5-5Mbps bandwidth is required for streaming game contents at 720p video [4]. Nvidia Grid* provides better streaming quality at 1080p, which requires bandwidth of 15Mbps. Furthermore, since decoding can be processed through low-cost decode chips which are massively embedded in game player's device, this approach is ideal for thin client running on resource-constrained devices. However, in the scenario of unstable network environment, graphics quality may be further degraded because the contents are streamed at lower resolution in order to adapt to the highly-burden connection.

Given the wide availability of service based on this method, most commercial Cloud Gaming companies, such as Nvidia GeForce Now†, PlayStation Now‡ and Steam In-Home Streaming§ are applying Image-based Streaming for delivering game contents.

On the other hand, GamingAnyWhere is the first and the only available open source Cloud Gaming platform [4], which is built as an Image-based Streaming structure. Designed to be highly extensible, the platform allows developers easily refer to programming interfaces of the modules and extend the capabilities of the platform. The platform is also portable to different environments such as Windows, Mac OSX and Linux, providing rich availability for testing. Furthermore, a large number of audio and video codecs are supported as well. According to the evaluation done in previous research [4], GamingAnyWhere outperforms the other two cloud gaming platforms including OnLive and StreamMyGame in term of response and graphic quality under a good network environment. Another advantage demonstrated by GamingAnyWhere is that the system introduces less network traffic compared to the other two. However, it is not as stable as OnLive if the environment becomes varied. Our proposed hybrid system is implemented based on GamingAnyWhere.

*Nvidia Grid: http://www.nvidia.com/object/cloud-gaming.html
†Nvidia GeForce Now: http://www.nvidia.co.jp/object/game-streaming-with-geforce-now-jp.html
‡PlayStation Now: https://www.playstation.com/en-gb/
§Steam In-Home Streaming: http://store.steampowered.com/streaming/?l=english

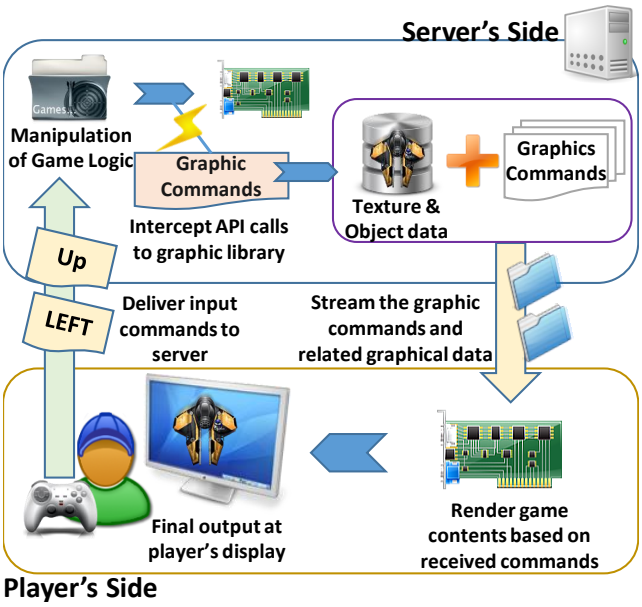


Figure 2.2: Structure of Instruction-based Streaming

In this research, Cloud Gaming generally refers to the system applying Image-based Streaming. Even though it is the more preferred methods, the delivered graphics quality is significantly degraded from original. Therefore, this is the particular issue we address on in this research.

2.1.2 Instruction-based streaming

As another method for delivering game contents, Instruction-based Streaming is varied in the way of where the rendering operation is performed. As indicated in Figure 2.2, after game logics are calculated at server, API calls to graphics library for corresponding rendering are intercepted. The intercepted API functions, or referred as graphics commands, are then compressed and sent to client's device. In addition, related 3D data such as geometry mesh and texture are streamed to the game player's device as well. Soon after the arrival of the data, 3D rendering according to the received graphics commands is processed on-site at the player's device. Put simply, game logics are calculated at the server, while the corresponding game contents are rendered locally at the player's device.

Eisert et al [16] implemented a low-latency Instruction-based Streaming within

Game@Large framework [17,18]. The model intercepts every API calls to OpenGL library[¶] as well as SDL library[‖], and tokenised them for reducing overhead during network streaming. In addition, caching techniques are status simulation are applied as well to mitigate network resource consumption.

The biggest advantage of Instruction-based Streaming over Image-based Streaming is the preserved original graphics quality, as the actual rendering is operated at the player's device. Furthermore, without the heavy burden of 3D graphics rendering, Cloud server is more efficient in term of simultaneously handling more player's requests. However, player is required to have a device not only compatible with the delivered 3D graphics commands such as OpenGL or DirectX[**], but also powerful enough to process high quality rendering in real time.

On the other hand, Instruction-based streaming could provide more stable gaming experience in a changing network environment because only light-weight graphics instruction are required to stream to the client side. However, at the start of Instruction-based Streaming, 3D resources such as texture data are streamed to the client side as well, which may incur network load ranging from 6Mbps up to 80Mbps [19] depending on the in-game scenarios. Once enough 3D resources are accumulated at the client side, only the light-weight graphics instructions are streamed to client. Furthermore, since individual connection consume less bandwidth, platform applying instruction-based stream could accommodate more clients simultaneously.

Compared to Image-based Streaming, Instruction-based Streaming is not suitable for resource-constrained devices, but should present as a viable option for gamer's PC, which is usually equipped with proper GPU for gaming.

2.1.3 Other Approaches

Beside the previous two major streaming methods, an approach of video streaming with post-rendering operation is also proposed [20]. This method is majorly designed for cloud-based mobile game usage. Compared to broadband network, wifi connection of mobile device implies much smaller bandwidth and possible

[¶]OpenGL: https://www.opengl.org/
[‖]SDL: https://www.libsdl.org/
[**]DirectX: https://www.microsoft.com/ja-jp/directx/

longer latency. Therefore, this approach aims to enhance video encoding for cloud-based mobile gaming service by taking the advantage of the run-time graphics rendering contexts form the 3D game engine. In this method, the modified encoder selects a set of key frames in the video sequence, uses a 3D warping algorithm based on the context data from the game engine to interpolate other non-key frames. The interpolation allows encoding warping residues with much lower bit rate, while maintaining or even improving video quality.

Another proposed method aims at streaming high quality graphics to mobile devices with low latency in the in-home environment, by exploiting a distributed rendering setup at the server [21]. The interactive multimedia application is processed in the distributed environment, while the output of the contents is compressed as ETC1 coding. Since ETC1 coding is natively supported by OpenGL ES which is the main 3D graphics library in mobile device, the application contents can be conveniently represented as OpenGL ES texture on the mobile display. Due to the powerful computing offered by the distributed setup, this streaming solution is also well suited for real-time visualisation of data intensive computations, which belong to the aspect of big data and HPC.

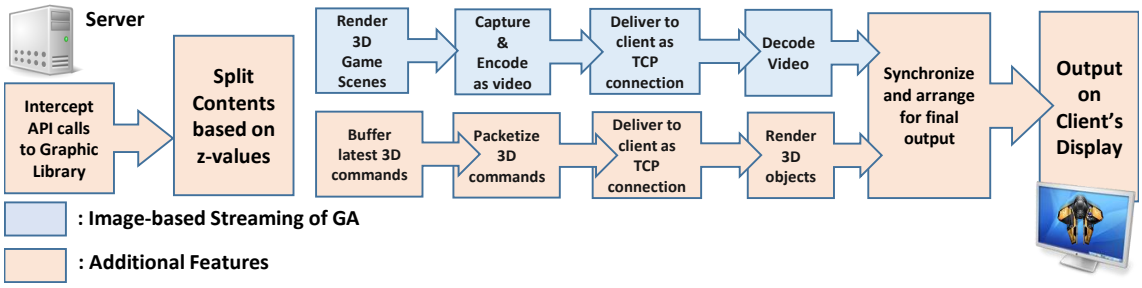


Figure 2.3: Data Flow of Proposed Hybrid-Streaming

2.2 Proposed Method: Hybrid-Streaming System

The goal of this research is to deliver novel Cloud Gaming experience that also likely satisfies demands of gamers. As such, we propose a streaming infrastructure which is based on Image-based Streaming, and integrate the mechanism of Instruction-based Streaming to achieve the desired improvement by exploiting on-site 3D rendering power at gamer's PC.

Our proposed Hybrid-Streaming System, which means utilizing both Cloud Gaming streaming methods, distributes partial game data to be streamed as 3D graphics commands and rendered locally at gamer's PC, while keeping partial rendering tasks to be processed at resourceful Cloud server and then streamed as video sequence. Graphics improvement is mainly contributed by the portion of game contents rendered by gamer's PC, while the system which is built upon the infrastructure of Image-based Streaming maintains highly available gaming experience. However, applying the proposed method also inevitably increase the GPU workload at gamer's PC. As such, there is a tradeoff between graphics quality and GPU workload. How such tradeoff is treated from the perspective of our system standpoint will be thoroughly discussed in later section.

2.3 System Structure Overview

Figure 2.3 presents the overall structure of our Hybrid-Streaming System. The blue boxes indicate the original data flow of Image-based Streaming which the

proposed system is based on, while the orange boxes refer to the additional features for achieving the overall infrastructure.

Within the system, how the final representation of game contents is correspondingly composed from the products of two different streams is an important design objective, which largely depends on the way of splitting the game contents at the server. By comparing the depth value, all the objects are separated into two groups, an upper layer which contains shallower objects (closer to client's view), and a lower layer which contains deeper objects (further away from client's view). Considering that the contents delivered through Image-based Streaming result in video frame without any depth factor, the game contents represented as this form should be treated as background. For this purpose, objects belonging to the lower layer are streamed as video sequence, which undergoes the original process Image-based Streaming. On the other hands, objects belonging to the upper layer are streamed as graphics commands. Therefore, once 3D rendering is accomplished through the GPU in gamer's PC, the products can be overlaid on top of the background filled by the contents from Image-based Streaming. The graphical representation of the final product is indicated by Figure 2.4.

In the following subsections, data manipulation in Image-based Streaming will be explained by exploring GamingAnyWhere(GA), which our work is based on. Furthermore, the additional features to achieve Hybrid-Streaming System will be introduced as well.

2.3.1 Data Flow of GamingAnyWhere's Image-based Streaming

In GA, the Data Flow refers to streaming audio and video frames from the server to the game player. In this research we focus on the graphics data manipulation throughout the Data Flow, indicated as the blue boxes in Figure 2.3.

Capturing original Game Contents and Encoding as Video

After the game source is processed and rendered at the server, a designated video capturer implemented in GA is triggered to capture the contents in a polling manner. The captured data are then buffered and encoded by encoder module.

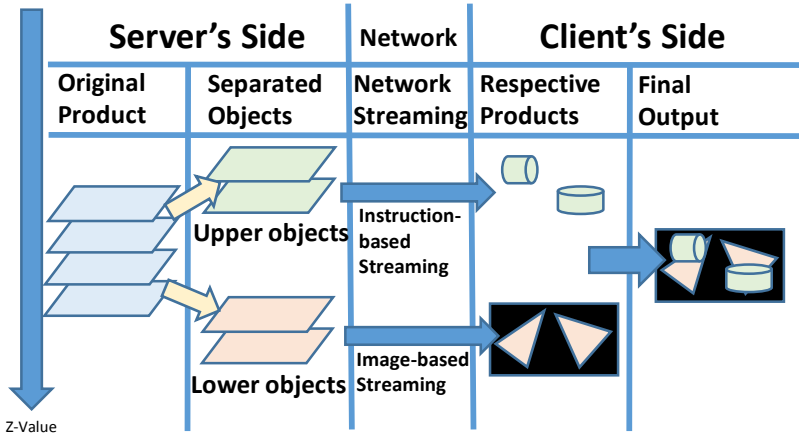


Figure 2.4: graphicsal representation of constructing the final output from two streaming

Given the highly extensible libavocode-based encoder module used in GA, our focus is on the rich customizable features, which enable us to easily adjust the encoding parameters that affect the delivered video quality.

Network Streaming

In GA, the encoded video frame, represented as RTSP and RTP/RTCP based packets, is delivered to gamer's PC. RTSP and RTP/RTCP are both specific protocol designed for the purpose of streaming media in the network. In GA, such packets are streamed in a connection of either TCP or UDP protocol, which can be specified based on user's preference. In order to handle the possible situation of packet loss, TCP protocol is set in our system.

Decoding Video

At GA's client module, only packets representing the most current encoded video frame delivered from server are buffered at the decoder. Once the video is decoded, the buffer is cleared for the next frame.

The state-of-art mechanism of GA's handling video source achieves satisfying results according to previous evaluations, which reveal that GA incurs low network load while the encoded video is delivered at relatively good quality [4]. It gives

us confidence that no significant modification for the process of Image-based Streaming is necessary in our system.

2.3.2 Proposed Mechanism for Hybrid Streaming

For achieving the Hybrid-Streaming System as shown in Figure 2.3, features including splitting the game contents, Instruction-based Streaming, synchronizing data from two streaming and correspondingly arranging the final output are mandatory.

Splitting the Contents

In order to distinguish if an object belongs to the upper layer or the lower layer during game processing, it is necessary to dynamically intercept API calls to the graphics library and looks for the most updated z-value of each object's center point. In this research, we only consider the case of OpenGL. The interception can be easily performed by applying existing application such as GLInterceptor. A threshold of z-value is defined for classifying which layer the object belongs, and usually in a manner of distributing the majority of the contents to be processed at Cloud server.

As to achieve the minimum required implementation for evaluating graphics quality, we manually manipulate depth value of every object. As such, we can explicitly split the contents to be processed as either Image-based Streaming or Instruction-based Streaming.

Contents' Processing Prior to Streaming

After the game contents classified as lower layer is rendered at the server, it undergoes the normal process of Image-based Streaming of GamingAnyWhere, as mentioned in last subsection.

As for the Instruction-based Streaming, the intercepted OpenGL instructions for rendering the upper-layer objects are overridden for not being processed through the server's GPU, but instead packetized for streaming to the gamer's device. Considering that usually same sequence of OpenGL instructions are routinely called for performing a particular task, we encrypt the sequence as a fixed code

indicating the corresponding task. As such, rather than delivering every native OpenGL instructions individually to gamer, a light-weight graphics commands set, which include a code and related parameters are streamed to gamer's device. Furthermore, regarding other graphics data such as objects vertices and textures that potentially incur heavy network load, a copy of such data is streamed in advance and saved at gamer's PC prior to actual gaming. Therefore, during gameplay, only the graphics commands set is streamed to the gamer's PC through the network based on a TCP connection. Such connection is separated from the one of Image-based Streaming.

Data Synchronization

Synchronizing data of two streams is another critical design objective. GamingAnyWhere's source capturing rate, which can be defined manually, is usually different from the API intercept rate, which follows the actual cycle of game processing. Therefore, the latest graphics commands which render the upper layer objects are buffered and updated in every cycle. Whenever GamingAnyWhere's source capturing is triggered, the most currently buffered graphics commands together with related parameters are packetized and streamed to thd gamer's device. After the graphics commands are sent, the buffer is cleared and continues for the next iteration of API interception. Since the contents of the two streams undergo separated TCP connection, a precise global time stamp is added to the respective headers during packetization. The global timestamp is for the client module to recognize the same-frame contents from the two streams. An indication of the buffering process is shown by Figure 2.5.

Final Arrangement at Client's Device

After the video is decoded from Image-based Streaming, an additional step is to assign a suitable depth value to the video frame for maintaining the contents to be always at the back of the locally rendered objects.

For the on-site rendering, a parallel running thread is responsible for receiving the inputs from Instruction-based Streaming and decoding the contents afterwards. Based on the decoded code which represents a particular rendering task, corresponding sequence of OpenGL functions are called and assigned with the

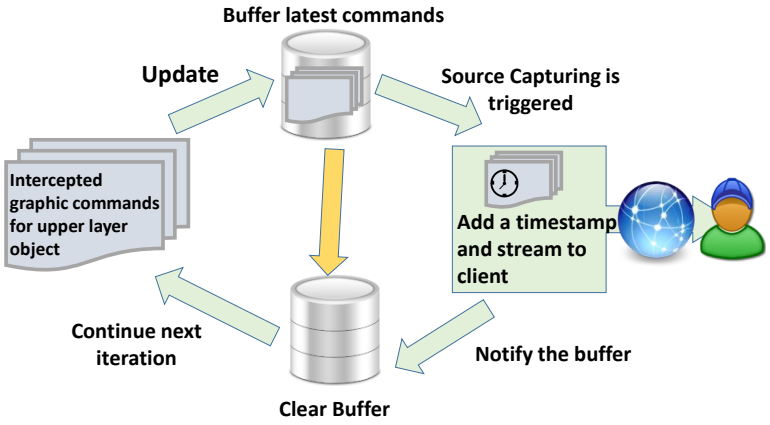


Figure 2.5: The cycle of buffering latest graphics commands for upper layer objects and streaming to gamer's PC

received parameters. As such, the local rendering at the gamer's device can be achieved.

Since decoding video frame from Image-based Streaming and local rendering based on Instruction-based Streaming are two separated processes at the client module, the embedded global timestamp is checked for confirming the same-frame contents before updating the next game frames. Once the confirmation is made, the game frame, which is composed of the rendered 3D objects overlaying on the video frame, is updated to the gamer's display.

2.4 Implementation

We implemented a prototype system based on the open source GamingAnyWhere platform. The main purpose of the prototype is to simulate the output of our system, which allows us to achieve the current main goal of evaluating graphics quality. By simulating the expected output of our system and conducting preliminary evaluations on graphics quality, bandwidth consumption and GPU usage at the server and also the gamer's PC, we can compare the results with the approach of Image-based streaming and eventually prove our concept. For this reason, instead of a full function of dynamically intercepting graphics commands, the graphics commands together with related parameters are streamed directly from the source of demo application. At the gamer's PC, the corresponding OpenGL objects are rendered based on the received instructions together with necessary texture data which are saved in advance. Such objects are eventually overlaid on top of the contents streamed as Image-based Streaming.

2.4.1 Setting Image-based Streaming in GamingAnyWhere

We installed the GamingAnyWhere pre-compiled package, which comes with both a client module and a server module, respectively on two machines, as one is for the server and another one is for the client. In order to stream a game in GamingAnyWhere, the platform provides a convenient way of capturing game frame by executing a simple command line in CLI. The command takes a configuration file as a parameter, in which we can specify the name of the gaming window. Many other parameters such as encoding bits, frequency of capturing and pixel formats can be adjusted within the configuration file as well.

The applied encoder is x264, which is the default configuration of GamingAnyWhere, and the data are streamed as H.264/MPEG-4 AVC format. Figure 2.6 shows the essential setting within configuration files. Figure 2.6(a) is the basic setting of the capturing video source, which includes the name of the demo application's window. Figure 2.6(b) shows the configuration of specifying the encoder to be used, which is x264 and the desired frame rate at 24fps for the out- put video. Specific parameters such as encoding bitrate at 3000kbps can be configured

```
[ga-server-periodic]
find-window-name = Demo
enable-audio = true
capture-cursor = true
```

(a) Configuration for capturing source

```
[video]
video-encoder = libx264
video-decoder = h264_dxva2 h264
video-fps = 24
video-renderer = hardware
```

(b) Specify encoder and capture rate

```
[video]
video-specific[b] = 3000000 # --bitrate
video-specific[refs] = 1 # --ref
video-specific[me_method] = dia # --me dia, equivalent to epzs
video-specific[me_range] = 16 # --merange
video-specific[g] = 48 # --keyint (gop size)
video-specific[slices] = 4 # --slices
video-specific[threads] = 4 # --threads
```

(c) Encoder parameters

Figure 2.6: Configuration for Image-based Streaming

as well, which is shown in Figure 2.6(c).

2.4.2 Instruction-based Streaming

The current Instruction-based Streaming sends 3D graphics commands directly from our created demo application. Operating in the manner of request-and-response, the graphics commands are only delivered when requested by the modified GA's client module. In this module, a separated thread for requesting graphics commands was developed. This thread, which operates in parallel with the original GamingAnyWhere decoding thread for video sequence, periodically sends requests to the server for graphics commands. Furthermore, a lock is used to synchronise the contents from both streams into the same frame.

As for the construction of the proposed light-weight graphics commands set, multiple OpenGL instructions responsible for a particular task are represented as a Function ID. Tasks here refer to altering the object's location, rotation,

scale, environment lighting effects and camera positions. For corresponding rendering operation to be performed properly at the client side, data that should be packetized includes:

- Object ID which indicates the object to be rendered (1 byte).
- Function ID which refers to the task to be performed (1 byte).
- Each task is followed by a set of corresponding parameters. For example, float values of X, Y, Z coordinates for setting the location of object (12 bytes).

Multiple functions can be aggregated into a single packet by simply putting graphics commands set one after another. Once receives the packet, the client module decodes the packetized contents and calls the corresponding OpenGL instructions to render the objects. The current packet layout is designed only for the minimum required data in this preliminary implementation, as it can be adjusted for adopting more patterns in favor of more complex gaming application.

2.4.3 Arrangement at Client Module

The original client module of GamngAnyWhere applies SDL library for directly presenting the decoded video contents on display. In order to switch to OpenGL-based operation, the module is modified to map the decoded contents as texture on an OpenGL-based rectangular-shaped polygon, which represents the background object of the final output. By manually setting a suitable depth value for this polygon object, the background contents are always further away than any locally rendered 3D objects.

As mentioned in last subsection, a separated thread was implemented to receive the packetized graphics commands and decode the contents. Based on the received graphics commands, Assimp[††], which is a library to import 3D model formats, is utilised to import the corresponding 3D models and OpenGL functions are called accordingly to render the objects into the buffer, which also contains the polygon object representing background contents.

[††]Assimp: http://www.assimp.org/

Finally, once the buffered products are called for updating the next frame on display, the locally rendered 3D objects accordingly overlay on top of the background contents streamed as encoded video from server.

2.5 Measurement

As for our current goal, we compared the graphics quality of the prototype Hybrid-Streaming System to original Image-based Streaming system of GamingAnyWhere. Besides, we also preliminarily evaluated GPU usage of both cloud server and gamer's PC. In addition, network bandwidth consumption was also examined.

Considering that using existing game software as a test case is sophisticated for the prototype stage of the system, we instead created a simple interactive demo application for evaluation.

2.5.1 Demo application for current evaluation

Before creating this demo application, which is OpenGL-based, we established a design objective to base on: the application should have clearly distinguished front objects and background objects, which enables us to easily split the contents into Image-based Streaming and Instruction-based Streaming. The created demo application is shown in Figure 2.7(a), in which one jet-fighter [‡‡] is presented. The main background, which includes the scene of mountain landscape, is created by applying the concept of skybox. Skybox is a cube object that the surface is filled with the assigned texture. It is always aligned to remain stationary with the viewer, while all the other objects must be at least closer than the skybox to the viewer. As such, it is always classified as background object and delivered to gamer's PC by Image-based Streaming.

The jet-fighter 3D model in foreground is constructed from multiple files which contain coordinates of vertices, texture image and mapping location. Applying Assimp allows us to conveniently load all the necessary 3D data and save it into the vertex buffers for constructing the desired 3D object on display. With the library, adding more 3D objects into the scene is also a task of ease. In addition, the location, rotation and scale of every object can be adjusted together with the camera position and certain lighting effects.

Since the location of every object can be explicitly defined in the demo, we can also conveniently classify if the 3D object belongs to lower layer, which is streamed from the server as video sequence, or upper layer, which is rendered

[‡‡]Blender 3D objects free for use: http://tf3dm.com/3d-model/sukhoi-t-50-pak-fa-4724.html

(a) With one jet-fighter

(b) With three jet-fighters

Figure 2.7: Demo Application

at the local device. For example, Figure 2.7(b) demonstrates the case of three jet-fighter objects. The two closer jet-fighters can be classified for Instruction-based Streaming, while the further one together with the landscape background are streamed as encoded video.

2.5.2 System Environment and Testing Scenarios

As for the hardware setup, the server machine is equipped with an Intel Core i7 4770K 3.5GHz CPU, a Asus Nvidia GeForce GTX780 graphics card DirectCU

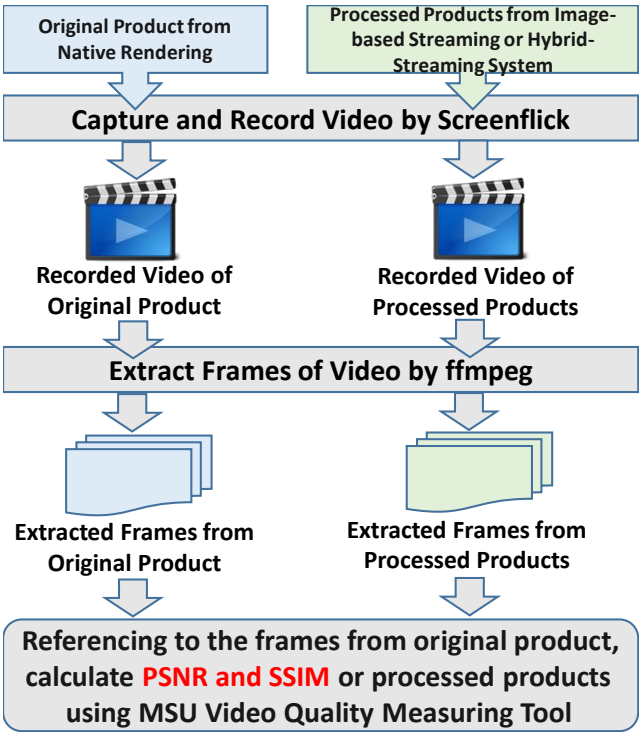


Figure 2.8: Procedure of measuring graphics quality

ii OC(Core clock at 889MHz with 3072MB GDDR5 video memory)[§§] and 16GB system memory, running on a 64-bit CentOS Linux system. On the other hand, the gamer's PC (client's side) is a MacBook Pro equipped with a 2.6GHz Intel Core i7 CPU, a Nvidia GeForce GT750M (Core clock at 926MHz with 2048MB GDDR5 video memory) with 16GB system memory. The Image-based Streaming of Gamin- gAnyWhere is set to stream the demo application at the resolution of 1280 x 720. The testing was conducted on a private local network.

Furthermore, all the measurements were conducted respectively on the demo application with different numbers of jet-fighter objects being loaded.

2.5.3 Procedure of Measuring graphics Quality

Referring to previous researches related to cloud gaming [4, 19], Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity (SSIM) are measured to evaluate

[§§]GTX 780 DirectCU II: https://www.asus.com/graphics-Cards/GTX780DC2OC3GD5/

graphics quality.

PSNR is used to measure the quality of reconstruction of lossy compression, which refers to the output from either Image-based Streaming or Hybrid-Streaming System in this research. It measures the ratio between the signal, which is the original quality produced by traditional graphics process- ing, and the noise introduced by compression. The higher PSNR value is the better the graphics quality it indicates. Typically the range of PSNR is between 30 and 50 dB.

SSIM is an index for determining the similarity between two images. It refers to the measurement of image quality referencing to an initial uncompressed image. The range of SSIM is from -1 to 1 while the value closer to 1 means higher similarity between the processed image (Image-based Streaming or Hybrid-Streaming System) and the reference image (produced by traditional local rendering).

In order to take the measurement of PSNR and SSIM, a window capturing software Screenflick[¶¶] is used to capture a short 60fps video of the demo application running in Image-based Streaming and the proposed Hybrid-Streaming at the gamer's PC. As for the reference source, we capture a short video of the demo natively running on our Linux server as well. The setting of window capturing is configured to be as high quality as possible for the purpose of retaining the original quality. After acquiring a video source for each mode, we used a function of ffmpeg[***] to extract every frame from each video source. Referencing to the frame of original quality, a chosen frame from Image-based Streaming and the Hybrid Streaming is used for calculating PSNR and SSIM respectively, which can be conveniently achieved by applying MSU Video Quality Measurement Tool[†††]. The overall measurement procedure is indicated in Figure 2.8.

2.5.4 Measuring GPU usage at server and gamer's PC

With the proposed system, the GPU usage of server is expected to mitigate since partial rendering tasks are offloaded to the gamer's PC. Simultaneously, GPU at the gamer's PC should be properly utilized without overloaded. Therefore, we investigated GPU usage at both sides under different situations of various

[¶¶]Screenflick: http://www.araelium.com/screenflick

[***]ffmpeg: https://www.ffmpeg.org/

[†††]MSU Video Quality Measurement Tool: http://compression.ru/video/quality_measure/

numbers of 3D objects being displayed in the demo application. We utilized tools, which include atMonitor[‡‡‡] (for MacBook Pro) and nvidia-settings (for CentOS), to take the measurement of GPU usage at each second. In addition, we logged all the data during the demo contents' being streamed to gamer's PC in either Hybrid Streaming or Image-based Streaming for one minute. At last, we took the average value of the logged GPU usage. Same measurement is repeated for scenarios of displaying different number of jet-fighter objects.

2.5.5 Measuring Network Traffic Load

Developing a Cloud Gaming system that is viable in common broadband network environment is also critical, as previous evaluation reveals that GamingAnyWhere's Image-based Streaming delivers relatively good gaming quality while attaining reasonably low traffic load. Since the implemented Instruction-based Streaming in our idea can introduce additional overhead, Ip-Traf[§§§], which is a network monitoring software, is used to measure average consumed bandwidth of demo contents being streamed for one minute. The measurement was taken under the scenario of twelve jet-fighter objects.

2.5.6 Results

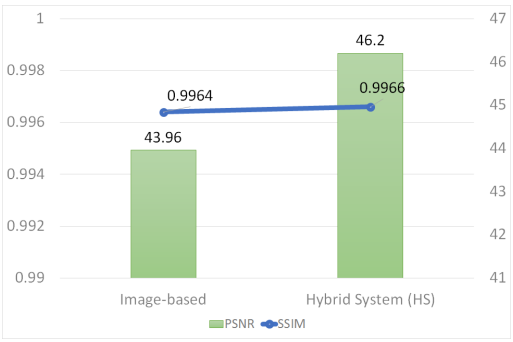
In this section, the measurement results are presented.

Results of PSNR and SSIM

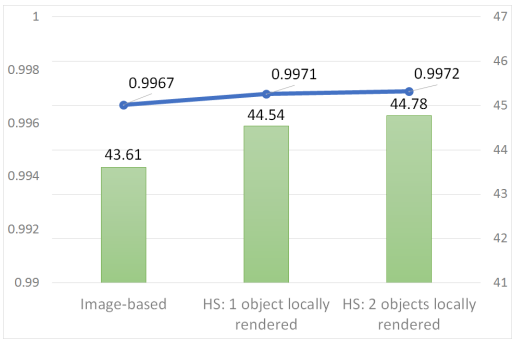
Figure 2.9(a) shows the PSNR and SSIM of the test case with one jet-fighter displaying in the demo application. The proposed Hybrid-Streaming System (HS), in which the jet-fighter object is locally rendered at the gamer's PC, achieves 46.20dB compared to 43.96dB from GamingAnyWhere's Image-based Streaming. As for the SSIM, the Hybrid-Streaming System also achieves better result than Image-based Streaming, but the difference is not significant.

In the second case which two jet-fighter objects are displayed, we conducted evaluations on two scenarios for the Hybrid-Streaming System. The first scenario

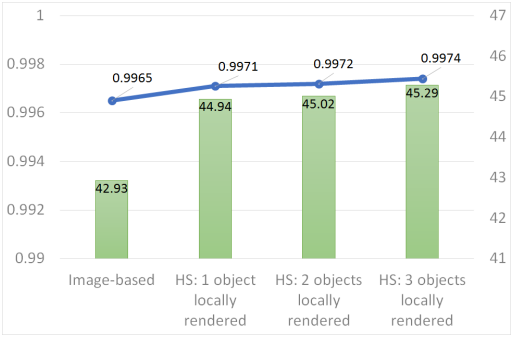
[‡‡‡]atMonitor: http://www.atpurpose.com/atMonitor/
[§§§]IpTraf: http://iptraf.seul.org/



(a) With one jet-fighter



(b) With two jet-fighters



(c) With three jet-fighters

Figure 2.9: Run charts of PSNR and SSIM in all the test cases

is streaming one jet-fighter as 3D graphics commands to be rendered at gamer's PC, while another one is delivered as video sequence. The second scenario is streaming both the two jet- fighters as 3D graphics commands. As shown in Figure 2.9(b), the first scenario achieves PSNR of 44.54, while the second scenario achieves slightly better result at 44.78dB. Both hybrid scenarios are better than the 43.61dB measured in Image-based Streaming. SSIM results from three scenarios are not significantly varied, but the Hybrid-Streaming System attains better results.

In the last test case, three jet-fighter objects are displayed in the demo application. Similar to the second case, graphics quality was evaluated based on various scenarios of streaming different number of jet-fighter objects as 3D graphics commands. As shown in Figure 2.9(c), all the scenarios of applying the Hybrid-Streaming System achieve overall better results than Image-based Streaming in

Table 2.1: Used bandwidth of delivering demo contents from server to gamer's PC under the test scenario of twelve jet-fighters

Number of objects streamed as graphics commands set	Consumed Bandwidth
None (Only encoded video)	402.25KB/s
3 jet-fighters	401.06KB/s
6 jet-fighters	406.39KB/s
9 jet-fighters	403.50KB/s
12 jet-fighters	401.25KB/s

terms of both PSNR and SSIM. Furthermore, while comparing only between the scenarios of Hybrid-Streaming System, better PSNR is achieved by streaming more jet-fighter objects as 3D graphics commands.

GPU Usage at Server and Gamer's PC

The respective GPU usage at cloud server and gamer's PC was investigated under the scenario of displaying more 3D objects, which include twelve jet-fighters, in the demo application. The purpose is to demonstrate changes of GPU usage along different rendering complexity, which is indicated by polygons count (One jet-fighter object is constructed from 25091 polygons, and the skybox (background scenery) is constructed from 12 polygons). In the test scenario, we took measurements from multiple cases of distributing different number of jet-fighter objects to be rendered at either side. As shown in Table 2.2, a decrease of GPU usage at server is indicated while rendering fewer jet-fighter objects. Simultaneously, GPU usage at gamer's PC increases as more jet-fighters are assigned to be rendered locally.

Network Bandwidth Consumption

For evaluating the additional overhead incurred by Instruction-based Streaming, we measured the consumed network bandwidth under the scenario of twelve jet-fighter objects. As shown in Table 2.1, all the measured scenarios of applying Hybrid-Streaming System incur rather trivial additional bandwidth over the case of Image-based Streaming (Streaming none object as graphics commands set). Throughout the observation, significant difference or trend was not observed on the network usage.

Table 2.2: GPU usages of server and gamer's PC under the test scenario of twelve jet-fighters

Number of distributed jet-fighter objects (with polygon counts) at server	Number of distributed jet-fighter objects (with polygon counts) at gamer's PC	GPU usage at server	GPU usage at gamer's PC
12 jet-fighters (301104 polygons)	0 jet-fighter (2 polygons)	18%	4.70%
9 jet-fighters (233121 polygons)	3 jet-fighters (77705 polygons)	16%	5.63%
6 jet-fighters (155418 polygons)	6 jet-fighters (155408 polygons)	15%	6.21%
3 jet-fighters (75285 polygons)	9 jet-fighters (225821 polygons)	14%	6.63%
0 jet-fighter (12 polygons)	12 jet-fighters (310814 polygons)	12%	6.98%

2.6 Discussion

In this session, the results from the conducted measurements are investigated. In addition, we will discuss about the novelty of the system by comparing with several related works.

2.6.1 Investigation on Graphics Quality

Evaluation on graphics quality from previous sections shows that our proposed Hybrid-Streaming System outperforms GamingAnyWhere's Image-based Streaming, as shown in Figure 2.9. When compared in PSNR, Hybrid-Streaming System demonstrates clean advantage over Image-based Streaming. As shown in Figure 2.9(a), our model achieves up to 2.24dB better. Furthermore, assigning more contents to be locally rendered at gamer's PC achieves higher PSNR, which is shown in both Figure 2.9(b) and Figure 2.9(c). This is reasonable because rendering at the gamer's PC can preserve the fidelity of the graphics. On the other hand, in the case of SSIM, our model retains a slightly better result than Image-based Streaming, but the difference is not significant.

Although gamer's experience was not evaluated subjectively at current stage, the PSNR/SSIM difference in the experiment provides us with insights of how it may indicate the subjective experience. According to previous works [22, 23], SSIM is sensitive in different aspect from PSNR. PSNR attempts to quantify the error between a distorted image and a reference image, while SSIM measures the degradation of structural information. The work mentions that SSIM should reflect better subjective information since the human visual perception is highly adapted to extract structural information. It also shows that the SSIM results align more consistently with the results from subjective evaluation using Mean Opinion Score than PSNR.

As for our evaluations, the Hybrid-Streaming system achieved both better PSNR and SSIM results than the Image-based Streaming. However, the variance of SSIM is rather small, which indicates that from a subjective perspective, it is not easy to distinguish the quality difference between Hybrid-Streaming and Image-based Streaming. On the other hand, higher PSNR achieved by Hybrid-Streaming system implies that higher fidelity of the graphics is preserved. It can be proved

by our system's achieving sharper edge and better graphical representation of texture. However, such details only become more visible when viewing closely to the graphics. We expect that by using more complicated gaming application, the perceived changes of graphic quality should be more obvious from a subjective perspective.

2.6.2 Investigation on GPU usages

The measured GPU usages of the server and the gamer's PC demonstrate expected tendency. When more objects are delivered to the gamer's PC as graphics commands set, GPU usage at the server, which is mainly consumed by objects' renderings and video encoding, decreases gradually. On the other hand, as more objects are assigned to be rendered in the gamer's PC, the machine's GPU usage, which majorly includes video decoding and objects' renderings, increases. Furthermore, even though the server is equipped with a more capable GPU, usage is overall higher than the one at the gamer's PC. It can be due to the reason that video encoding is responsible for most of the workload, while decoding at the gamer's PC is relatively light-weight. In addition, another finding is that the GPU utilization at both sides is low (Averagely less than 20% at the server, and less than10% at the gamer's PC). It is considered that the created demo application is not calculation-intensive, and without any significantly complicated effects. We expect that by testing with a more formal gaming application, the GPUs at both sides can be utilized more efficiently.

2.6.3 Investigation on network bandwidth

As for the results of the measured network bandwidth, mainly two findings can be summarized. First, the overhead incurred by the additional Instruction-based Streaming mechanism in our proposed system is relatively trivial. According to the data presented in Table 2.1, the bandwidth is mostly consumed by delivering the contents as Image-based Streaming. It is expected that more sophisticated graphics commands in formal gaming application will likely lead to larger overhead. Even so, as Hybrid-Streaming System saves related graphical data such as textures and geometry meshes at gamer's PC prior to actual gaming, low-bit

overhead should be maintained by streaming only the light-weight graphics commands set. Furthermore, another finding is that delivering more 3D objects as Instruction-based Streaming do not imply a trend of increasingly utilitzed bandwidth. According to the measurements, solely Image-based Streaming actually incurs higher consumed bandwidth than the case of streaming three jet-fighters as graphics commands set.

2.6.4 Related Works

In term of utilizing client's device for rendering, Nan et al [24] proposed a method of jointly using video and 3D graphics streaming to construct a cloud gaming framework, aiming at achieving low-bit rate transmission. In this proposed method, two stages of processing can be defined. As for the first stage, game contents are streamed as normal video streaming, but simultaneously 3D data such as geometry mesh and textures are transmitted to client side and saved at a buffer. As time goes by, more 3D data are accumulated at the client's buffer and the frame rendered from this local buffers tends to be closer to the original data. Soon as all the graphics data are received at the client, the system can shift to the second stage of only streaming 3D commands for rendering the graphics locally at the client side, thus achieving low-bit rate eventually. Comparing to this method of progressively shifting from Image-based Streaming to Instruction-based Streaming, our system utilizes both streaming in parallel. Without possibly incurring unevenly heavy workload on either side, our system intends to allocate render power from both sides to retain availability of Cloud Gaming service, and improve the graphics quality as well.

For improving graphics quality in a Cloud Gaming environment, Shi et al. [20] proposed an approach of video streaming with post-rendering operation, which is majorly designed for cloud-based mobile game usage. Concerning the limited network environment, the approach aims to enhance video encoding by taking the advantage of the run-time graphics rendering contexts from 3D game engine. The modified encoder selects a set of key frames within video sequence, uses a 3D warping algorithm based on the context data from the game engine to interpolate other non-key frames. The interpolation allows encoding warping residues at a much lower bit rate, while maintaining or even improving video quality. On the

other hand, Pohl et al. [21] proposed a method for streaming high quality graphics to mobile devices with low latency in an in-home environment, by exploiting a distributed rendering setup at the server. After the powerful server environment processes the interactive multimedia application (Gaming application was used as a test case), the contents are encoded as ETC1 coding, which can be conveniently represented as OpenGL ES texture on mobile display. Compared to these two methods, the established service target of Hybrid-Streaming System is gamer's PC, which is significantly more powerful than mobile device. Furthermore, from a technological perspective, the two methods achieve high quality video streaming through modifying the encoding scheme, as our approach utilizes local rendering to preserve partial game contents as original quality.

2.7 Limitations of Proposed System

The Hybrid-Streaming system is built upon the Image-based Streaming structure while applying the mechanism of Instruction-based Streaming to achieve the improvement of graphics quality, as well as the mitigation of server's workload. On the other hand, the main tradeoff for these improvements is the increased GPU workload at client's PC. Therefore, in this section we will clarify our system's standpoint by thoroughly discussing the tradeoff from the perspective of a Cloud Gaming system. Furthermore, there are multiple concerned limitations at the current prototype stage of the system. The limitations include contents separation in more complicated application, applying post-effect, inevitable waiting time before first-time gameplay, latency constraints and required special modification in the application source code. In order to further develop our system to be more practical in common usage, we will address each of the above limitations and provide our planned solutions in the following subsections.

2.7.1 System's Standpoint

From general Cloud Gaming perspective, the increased workload at client's machine in our system can be considered as a limitation. However, it is our approach that intends to utilize the available GPU processing power at client's machine to achieve the improvement of high-quality gaming graphics.

A typical cloud gaming system only requires client's machine to decode video sequence, which is a light-weight operation and thus suitable for resource constrained thin-client. However, it is more likely for gamers to own a machine that is possibly not very powerful, but capable enough to handle graphics rendering to a certain extent. Therefore, when gamers use Cloud Gaming, the mere operation of decoding only occupies a relatively small workload when compared to the full potentials of their machines. In addition, even though they can enjoy immediate gaming, the degraded graphic quality hardly satisfies the gamers' demands for high fidelity graphics. As such, we address this problem and propose a novel Cloud Gaming system that can particularly benefit this group of customers. Therefore, in our system, utilizing local rendering power inevitably induces higher workload at client side, but this is a reasonable tradeoff in terms of graphics quality and

resource usage at the server. Furthermore, compared with Instruction-based Streaming, the Hybrid-Streaming system represents a more flexible option to attain hight-fidelity graphics rather inducing too much workload at gamer's PC.

As a limitation, our system is not applicable to devices that are without any graphics processing unit, since ultimately our system requires client's device to be capable of graphics rendering to a certain extent. Furthermore, if multiple graphics-related applications are processed simultaneously at client's device, the heavy-loaded environment may hinder a smooth gaming experience. Therefore, it is necessary for our system to achieve the balance between GPU workload of client's device and the delivered graphics quality.

Currently our system focuses on games which are for single-player. In the case of multi-player games which involve multiple connections from different players, it is necessary to maintain fairness of gaming quality among the players. Therefore, it is necessary to retrieve rendering resource information at player's side in order to determine how much server resource should be allocated for each connected player. As an example, for players with more powerful rendering resource, game contents can be delivered only via Instruction-based Streaming. On the other hands, Hybrid-Streaming method could be used for players with less powerful gaming machine.

2.7.2 Contents separation in complicated renderings

Currently, each object, as a whole mesh structure, is distinguished as the upper layer or the lower layer based on the pre-defined z-value threshold. The separation is viable in the demo application. However, in more complicated cases such as a foreground object being placed on the background, or the foreground being occluded by background objects, the currently implemented solution may not be applicable. This can be due to the reason that the objects, such as ground, may span across the upper layer and the lower layer. That is, certain object cannot be clearly distinguished as the upper layer or the lower layer based on the z-value threshold.

For addressing this issue, we consider to improve the approach of separation by utilizing the characteristics of viewing frustum. A viewing frustum refers to a three-dimensional region which is visible on the screen. The region volume is

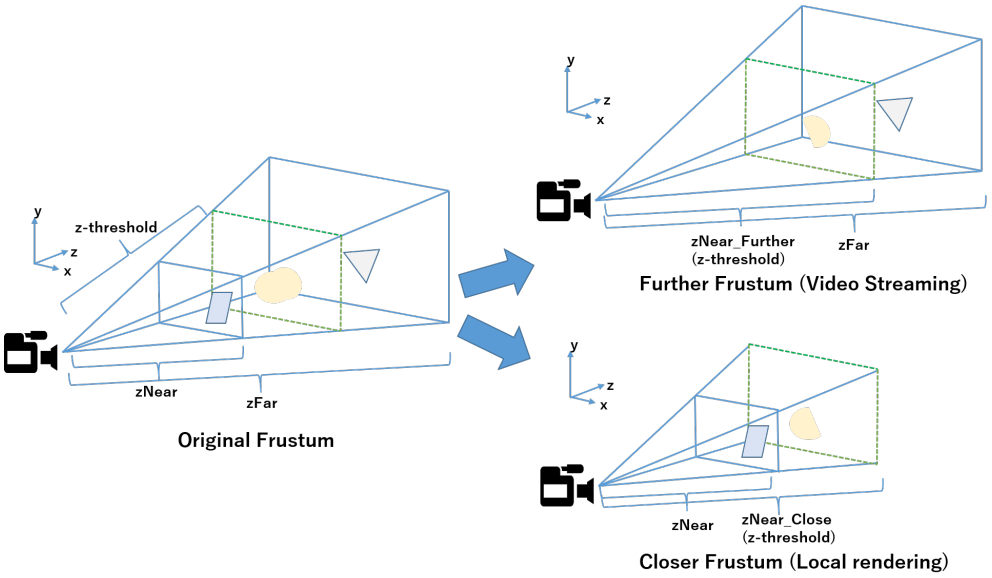


Figure 2.10: Division of viewing frustum based on the z-value threshold

specified by the values of near, far, left, right, bottom and top. The primitives, which include points, lines or triangles, are clipped out from the region if they are outside the boundaries. Therefore, instead of classifying all the objects into two groups, the continuous development is to divide the viewing frustum into two regions, the closer frustum and the further frustum, based on the specified z-value threshold. The division is shown in Figure 2.10. After the division, the contents within the closer frustum, which has the original near value and the z-value threshold as its far value, are streamed as 3D instructions to be rendered at the gamer's PC. On the other hand, the contents within the further frustum, which has the original far value and the z-value threshold as its near value, are processed at the server and streamed to the gamer as video sequence.

In addition, for the objects which cross the boundary of the z-value threshold, both the client's device and the server need to render the same objects. For the portion that belongs to the closer frustum, the respective clip plane based on the far value (z-value threshold) clips out the portion that belongs to the further frustum. Similarly, for the portion that belongs to the further frustum, the respective plane based on the near value (z-value threshold) clips out the portion that belongs to the closer frustum. Therefore, the final output is appropriately

shown on client's display by aligning the further frustum (video streaming) with the closer frustum (local rendering.)

The idea of separating viewing is considered to be applicable to any genre of gaming applications. Even though further evaluations are required to be affirmative with our assumption, we consider that game applications which have clearly distinguished foreground objects and background objects should benefit the most from our method. Such genres of games include First Person Shooters such as Call of Duty series, scroller games such as Super Mario, Fighting Games such as Street Fighter series and etc.

2.7.3 Applying post-effect

At current stage, post-effect was not applied on the demo application. However, post-effect is often manipulated by shader program, so such program can be streamed to client's PC prior to actual gaming as well. Furthermore, as shown in the previous section, the original view frustum is divided into two parts, which include the closer frustum (upper layer) and the further frustum (lower layer). Contents in both frustums undergo normal rendering process, which includes utilizing the shader program to manipulate the post-effects, respectively at client's PC and cloud server. However, at the client side, the special rectangular object for representing the video contents streamed from the server should not be applied any post-effect since the contents needed to be kept as original. When the lower layer contents (video streaming) are aligned with the upper layer contents (local rendering), the system should achieve the output with the post-effect being properly applied on.

In addition, the post-effects rendering offloaded to client's PC may additionally increase the GPU workload at client's PC. Therefore, it is necessary for our system to balance the delivered graphics quality and client's GPU workload which includes the task of rendering post-effects

2.7.4 Waiting time prior to first-time gameplay

The necessity of downloading assets prior to game play in the implementation is regarded as a limitation of our system. Even though it is a one-time process, it is

inevitable that the client needs to wait before the first-time gameplay. Therefore, the current implementation may not compromise fully the benefit of cloud gaming.

Referring to previous work conducted by Nan et al [24], a feasible solution to eliminate the waiting time is delivering the contents fully in Image-based Streaming at the beginning, and simultaneously downloading the asset to client's machine. Once enough asset data is accumulated at client side, the system can shift to our proposed Hybrid-Streaming. However, this method may induce heavy network load since asset data can be large, thus further investigation is required.

2.7.5 Latency Constraints

At the current stage of development, since we primarily focus on improving graphic quality delivered on Cloud Gaming, we are not leveraging any additional technique being applied on either image- or instruction-based streaming to improve latency. Therefore, the latency of our system is based on Image-based Streaming, and larger than Instruction-based streaming. It is due to the fact that Image-based Streaming has heavier flow of task together with larger video data to be transferred through the network. However, according to the evaluations performed by previous work, the Image-based Streaming platform of GA, which our work is based on, achieves rather low processing latency. Therefore, under an environment with sufficient network speed, our system should provide game players with smooth gaming experiences.

Furthermore, since GA is designed to be highly extensible and portable, the existing works which include motion prediction [12] and resource allocation [13] are considered to be applicable to our system for improving the latency.

2.7.6 Modification required to apply Hybrid-Streaming System

The Hybrid-Streaming system is built upon the Image-based Streaming framework of GA, which is able to capture and encode the output of target application without altering its source code. However, for implementing the mechanism of Instruction-based streaming in our system, special modifications are required in the source code of target application. The modification includes adding new

functions to separate the contents into the upper layer and the lower layer, as well as the network streaming for graphics instructions. At current prototype stage where we use a relatively simple application, the task of modification is light-weight.

For practical usage, it is necessary for developer to adapt the Hybrid-Streaming method by performing corresponding modification. However, we consider that it is challenging to make any direct modification to the source code because the target application can be large and complicated. Therefore, our next goal is to include all the necessary functions in a library which can be easily adapted by Cloud Gaming frameworks and without any necessity to modify the application source code. As such, game developer can specify a z-value threshold and the framework, which utilizes our specialized library, can automatically perform a sequence of tasks which include intercepting graphics commands, separating the contents and streaming the contents to client's PC. Furthermore, by monitoring the GPU workload at client's PC, the z-value threshold can be dynamically adjusted to ensure task balancing.

2.8 Research 1: Conclusions and Extensions of the work

In this work, we present our idea of making the connection between existing Cloud Gaming system and traditional PC platform. While attaining Cloud Gaming's highly available gaming experience, we especially aim at delivering improved quality that potentially satisfies high demands from gamers. For this established goal, the contribution of this work can be summarized as follows:

- Upon existing Cloud Gaming infrastructure, a Hybrid-Streaming System that applies two streaming approaches is proposed.
- With the proposed system, rendering tasks are distributed to server and gamer's PC. As such, more available rendering power can be effectively utilized.
- A prototype of the system was implemented and evaluations show that it achieves improved graphics quality over traditional Cloud Gaming platform, while maintaining reasonable network load.

Currently the Hybrid-Streaming System is still at the prototype stage, as more thorough investigations are needed to evaluate the idea as a prospective Cloud Gaming platform. As for the extension of this work, the following three concerns can be addressed:

- **If gamer's PC is capable, is it better to keep the traditional way of gaming** Traditional approach of game processing at local PC still preserves the best gaming quality. However, gaming applications, especially those regarded as Triple-A titles, are increasingly complicated, which put heavier workload on the GPU. Through the proposed idea, such workload can be shared between the cloud server and the gamer's PC. Not only original graphics quality can be preserved to a certain extent, but also highly accessible gaming experience can be achieved. Preliminary investigation on GPU usage in current work gives us insight that we can improve the system by dynamically assigning rendering tasks based on continuous monitoring of GPU usage on both sides.

- **Is gamer's actual experience improved while applying the proposed idea?** Numerical results indicated desired improvement in term of graphics quality on Hybrid-Streaming System. Furthermore, the results also provide insights into how subjective experience may be preserved. However, evaluating gamers' Quality of Experience (QoE) based on feedbacks is more critical, because ultimately Cloud Gaming, as a service, is client-centered. Subjective test involving actual clients to measure QoE based on Mean Opinion Scores [25] is planned to be conducted.

- **Is the Hybrid-Streaming System compatible with every genre of games?** In the demo application, foreground objects and background objects are clearly defined, as we could explicitly split the contents based on the depth value. Our next goal is to implement a new separation approach which is based on the viewing frustum. We consider that with the new approach, the Hybrid-Streaming System can be compatible with any genre of games.

Furthermore, we will continue to improve the applicability of the Hybrid-Streaming System by applying it on a larger scale testing environment, such as Nvidia Grid [26].

3 A Head-Motion Prediction Model for expanding the usage of Cloud Gaming to VR gaming

In this chapter, we will mention about our perspectives on improving the accessibility of VR gaming by applying Cloud Gaming techniques. Based on this goal, we propose a head-motion prediction model tha could compensate the inevitable latency issue in a Cloud Gaming system.

3.1 Introduction

VR technologies have been widely used in various professional fields as well as for medical usages. However, most of these VR simulations require heavy local-setup and traditional head-mounted display (HMD) is usually high-cost. Therefore, it is proven to be almost impossible for normal users to access to any of these VR experiences.

Thanks to the advancement of VR technology, the recent introduction of more compact and affordable HMDs becomes a game-changer. With several representative HMDs such as Oculus Rift*, HTC Vive† and Playstation VR‡, normal users are granted more accessibility to VR experiences. In addition, the increasingly more resourceful smartphone device also represents another mean

*Oculus Rift: https://www.oculus.com/rift/
†HTC Vive: https://www.vive.com/
‡Playstation VR: http://www.jp.playstation.com/psvr/

for users to experience VR. By inserting smartphone into specialized head mount device such as GearVR [§], user could conveniently access to media contents presented in VR.

In the gaming industry, companies are investing large amount of resources into developing games for the emerging VR devices. Compared to traditional gaming, VR gaming provides game players with unparalleled immersion into the virtual world, higher dimensional of controls and more instinctive interactions with game elements [27]. However, for maintaining a highly immersive VR experiences, properties including graphics quality and responsiveness are considered to be essential. By graphics quality, we refer that game environment presented in the HMD needs to be realistic and in high-definition. As for the responsiveness, users' motions, especially of the head, need to be reflected as swiftly as possible in the virtual game world by the backend system that processes the VR contents. Without meeting the above requirements, motion-sickness may happen, which largely hampers players' VR experiences. Therefore, compared to traditional gaming, a more demanding and costly high-spec backend machine is usually required for VR gaming. According to Nvidia, for a smooth VR gaming experience, it is required to run the game at 2K resolution and 90fps, compared to 1080p resolution and 30fps for typical PC gaming[¶]. Furthermore, Kanter et al [28] also reveals that the actual number of pixels rendered per second for VR systems at a good frame latency is roughly 4 times higher that average PC display witch 1080p resolution, or twice of a high-end PC display at 2K resolution. Zhong et al [29] also mentions that the frame rate of 90Hz is required to ensure a smooth user experience, compared to 60Hz of traditional PC gaming. Furthermore, future VR targeting at a frame rate of 120Hz will pose increasing challenges. Therefore, the requirement of a more demanding gaming environment can still hinder the popularity among many potential users who may not be able to afford such gaming system. Furthermore, the application is event more challenging for the untethered wireless environment of using smartphone as the VR device.

With this in consideration, we aim at further improve the accessibility of VR gaming which is based on the current compact HMD device. As such, we propose

[§]GearVR: http://www.galaxymobile.jp/gear-vr/

[¶]https://venturebeat.com/2015/12/30/to-handle-vr-graphics-gaming-pcs-have-to-be-7-times-more-powerful/

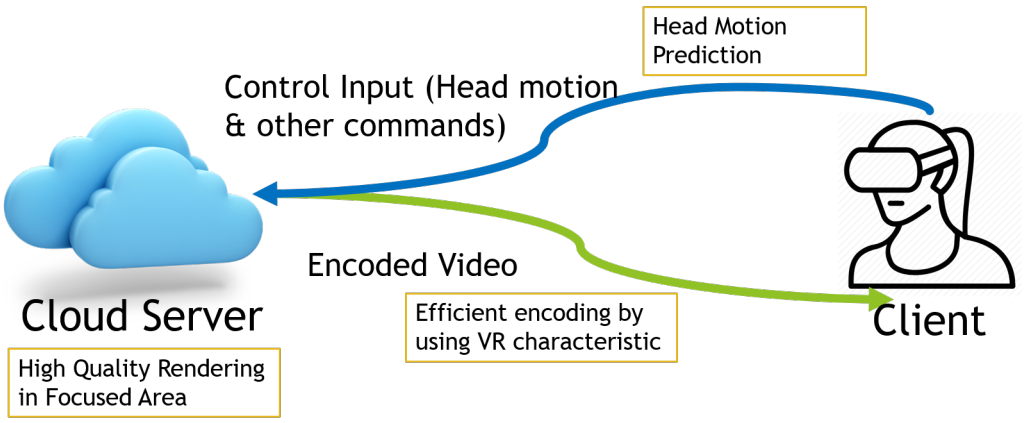


Figure 3.1: Issues to be addressed in a cloud-based VR gaming environment

exploiting the technique of Cloud Gaming to achieve the improvement in need. By leveraging elastic and high-performance computing resources, the intensive workloads of processing VR contents can be offloaded to powerful Cloud Server. The rendered game contents are delivered back to the client's HMD as a sequence of encoded video over a broadband network. Compared to traditional gaming, VR gaming requires additional inputs such as the rotation angle and the position of HMD device to be recorded and sent back to the Cloud Server for the manipulation of game logics. Therefore, cloud gaming potentially frees players from buying expensive gaming machines and allows them to play high-quality VR games on any devices, which include varied choices of smartphones.

We consider that Cloud Gaming is a prospective option for improving the accessibility of VR gaming, but multiple difficulties need to be carefully addressed before making it viable for practical usage. In the following section we will describe the considered difficulties and introduce previous works that may possibly address the respective issues. Afterwards, we select the issue that affects VR gaming experience the most and address

3.2 Expected Difficulties and Potential Solutions

As mentioned in the previous section, rapid responsiveness and high-definition graphics quality are two crucial factors to preserve highly immersive VR experience. However, dealing with these two factors present to be even more challenging in Cloud Gaming because of the inevitable network latency and limited transmission bandwidth. Therefore, as shown in Fig. 3.1, based on the unique features of VR gaming, we especially address the following three points and introduce several previous works that potentially address the respective issues:

3.2.1 Head Motion Input

Besides the input commands from keyboard or gamepads, head motion is another critical input in VR gaming. Rapidly and accurately sensing the inputs from player's head-motion and seamlessly having the corresponding feedbacks in the virtual game environment is fundamental in VR gaming. Otherwise, it may result in motion-sickness which negatively affect player's experience. The rigorous requirement is considered to be even a bigger challenge in the usage of Cloud Gaming, due to the inevitable network latency and the occasional packet loss. For example, in a smooth network environment, the round-trip delay occurred in Cloud Gaming is approximately at 150ms [4].

Potential solution based on previous work: Several works have been conducted to reduce the interaction latency in a VR system. Johnson et al. [30] proposed using low-cost motion trajectory prediction techniques to compensate the high end-to-end latency in a smartphone-based HMD environment with a hand motion tracking device. Three trajector prediction techniques were used to predict the hand motion in 300ms. As a result, linear extrapolation was the only effective way to predict the motion. Even though head-motion was not the input being predicted in this work, Predicting high-diemensional motion in 300ms at a reasonable accuracy by such low-cost method indicates that the technique is applicable in a Cloud Gaming environment (with normal latency at 150ms). Furthermore, Soccini et al. [31] proposed a method of predicting which area the player is gazing at in immersive virtual environment without using any eye-tracking

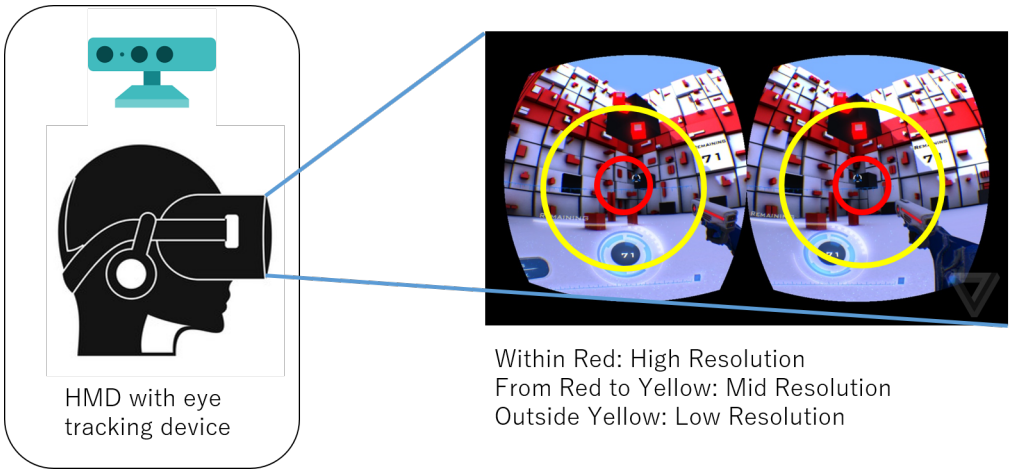


Figure 3.2: Foveated Rendering

devices. The proposed solution is based on a combination of the features of the images and head motions as inputs of a Deep Convolutional Neural Network to map 2D gaze coordinates in the image plane. The research shed light on predicting player's focus area in HMD without using additional expensive devices, which indicates a good synergy with the network environment of Cloud Gaming because less information is needed to be sent through the network.

3.2.2 High-quality rendering in real time

Compared to traditional gaming, our ocular system tends to be more sensitive to in-game details in VR gaming. Therefore, for smooth VR gaming experience, it is essential to not only render graphics in high-quality but also to sustain high frame rate. Powerful servers in Cloud Gaming are certainly capable of processing the heavy tasks of VR gaming, but as a cloud service, it is also crucial to allocate resources efficiently and fairly among clients. Thus, rather than incurring excessive workload on the server, it is more reasonable to look for a more efficient way of processing VR game contents that can be adapted to a Cloud Gaming environment.

Pontential solution based on previous work: Foveated 3D rendering is a popular technique being used to lower the VR computational requirements [32] . As shown in Figure 3.2, eye tracking is used to determine the area where player is focusing and only the corresponding area is needed to be rendered in greater

details. In addition, the surrounding area can be rendered in lesser details and thus mitigating the overall workload at the cloud server. Furthermore, based on foveated rendering, Pohl et al. improved the performance with optimizations by taking HMD's lens astigmatism into consideration [33]. As a result, the performance is increased by 20%. The corresponding continuous work [34] proposes method that even further speeds up the rendering by determining the respective viewable area when player looks at different point on the HMD display. As such, the renderings for the unviewable areas are skipped by reusing pixels from previous frame. The evaluation indicates that rendering can be sped up by 2 times by using the proposed method. Even though the above works were conducted in a local environment, the same techniques could be easily applied in Cloud Gaming and greatly mitigate the workload of the Cloud server. However, one concern is that eye tracking device, which is usually not embedded in most current HMDs, is required in this method. Therefore, building a generic model based on eye tracking data may present to be more practical for the Cloud Gaming environment.

3.2.3 Efficient Encoding for Transmission

In Cloud Gaming, encoding the game contents as video sequence is a fundamental step. Not only the data needs to be efficiently compressed for streaming in network, but also the quality of original game contents should be preserved reasonably. Therefore, it is necessary to optimize the encoding method for coping with the even more severe requirement of VR gaming usage.

Potential solution based on previous work: Currently most real-time entertainment applications such as streaming of audio and video are accomplished on top of MPEG-DASH standards, which the actual implementation is open for development. Timmer et al. [35] proposed a cloud encoding service and HTML5-based adaptive streaming that enables highest quality and low start-up delay. The work claims that the proposed service is able to perform streaming at Ultra-High-Definition (UHD) and high frame rates, such as 4k resolution video at 60fps. Furthermore, it is also compatible with the playback of VR. The versatility of the service indicates a good match with the requirement of smooth VR gaming, but there is no mentioning about the bandwidth consumption in network. In order to build a Cloud-based VR gaming system which does not over allocate

network resource, we consider that more efficient encoding can be achieved by utilizing the similarity between the left eye contents and the right eye contents, which is a unique characteristic of VR gaming.

The mentioned previous works were mostly conducted at small scale, which is either in a wired environment or a local wireless network. As for designing a proper Cloud Gaming system for VR gaming, the demanding service should not be an one-to-one, but one-to-many platform under a geographically distributed environment. Therefore, we consider that in order to make Cloud Gaming a viable platform for providing good-enough VR gaming experiences, all the above issues should be further addressed in the context of global network. However, solving all the above difficulties and constructing a comprehensive Cloud Gaming system for VR gaming are not the focus of this research. Instead, among all the above issues, we select the one that should affect the quality of gaming experience the most and address it by proposing a solution. Previous studies showed that in immersive virtual environments, player is able to detect latency lower than 16ms [36]. The study implies that the inevitable network latency will largely degrade player's VR experience if proper countermeasure is not applied because in a normal Cloud Gaming system, the RTT is at approximately 150ms. Since head-motion is the most fundamental input that determines where the player is viewing in an immersive virtual environment, we decided to address this particular issue and propose a method that compensates the influence of network latency. Rather than mitigating the round trip time (latency), we propose using a prediction model for predicting head-motion at cloud server. Therefore, the next few frames of game contents can be pre-rendered based on the prediction and streamed back to client more rapidly, without the need to wait for every head-motion commands' incoming from the client. The goal of applying such method is to maintain smooth VR experience without players' feeling the network latency. Our work can be summarized as follows:

- Collecting head-motion data (represented in quaternion) through a consumer-level HMD.
- Based on Machine Learning technique, constructing a prediction model for each set of data.

- Cross-validation for the generic prediction model.

In the following sections, we will briefly introduce several related works. Afterwards we will explain our approaches to build the prediction model for head-motion, followed by corresponding results and discussions.

3.3 Related Works

The future vision of this research is providing highly accessible VR gaming experience based on Cloud Gaming technique, while currently there exist several platforms or works that similarly attempt to provide wireless or cloud-based VR gaming experience. Trinus VR [37] is a platform that provides players with VR gaming experience without the need to purchase additional expesnvie hardware, because smartphone is used as the HMD. Motion data collected from sensors of player's smartphone is transformed to a PC at local side for processing game logics. The resulted contents are then streamed back to the smartphone. Trinus VR is designed to be easy to setup and the system provides a number of settings for different display size. Furthermore, Trinus VR supports both wired and in-house wireless end-to-end connections, while wired connection is suggested to provide the most sustainable performance. Even though no quantitative result is mentioned, we subjectively evaluated VR experience via local wireless connection, and we discovered that there is sensible latency between real head-motion and contents reflected on our smartphone device. We consider that Trinus VR is not yet a platform for streaming VR gaming contents in a large-scale environment, but it can be used a test-bed for the related purpose.

Cuevro et al [38] from Microsoft Research proposes a system called Matia that supports VR gaming experience on resource-constrained mobile devices by offloading processing to Cloud server. The system is claimed to attain quality, responsiveness and mobility. This work renders a wide field-of-view (FOV) panoramic stereo video to handle unexpected head movement or network fluctuations. Furthermore, Foveation technique is also applied to reallocate pixels to areas where the user is most likely to view. As only the idea is presented in the work, there is no evaluation showing the efficiency of the method. We consider

that both Cuevro et al's work and our work have similar goal, but the focus area of optimization is different.

Bao et al [39] proposes a Motion-Prediction-Based Transmission for 360-Degree Videos in an attempt to significantly reduce the bandwidth consumption. Compared to regular video, the 360-degree video transmission imposes 4-6x the bandwidth of a regular video with the same resolution. It is due to the reason that the whole image, including the portion that user is not viewing, needs to be streamed to client's HMD because the view angle is not known in prior at the server. For addressing the issue, the author proposes using a prediction model to determine which area the user is viewing. Ideally, if the prediction model could predict user's motion at 100% accuracy, only the portion of image that user is view needs to be streamed, which could reduce bandwidth consumption by 80%. However, due to the random nature of user's head motion, further technique is required for assuring the quality of experience. According to their studies, their prediction model could well predict the motion in 100 to 500ms. Furthermore, the work predicts the prediction deviation as well to decide the amount of redundancy to be transmitted. Such prediction is used as a mean to handle the gap between prediction result and true viewing point for maintaining user's viewing experience. As a result, the proposed system achieves up to 45% bandwidth reduction with less than 0.1% failure ratio. One significant point regarding this research is that the authors applied Machine Learning (ML) technique, specifically Neural Network (NN), to build the prediction model and achieve the best result among the other two methods. Therefore, from technical perspective, this work is potentially the closest to our work. However, we focus on exploiting ML-based prediction model to compensate network latency for streaming VR gaming contents, while they optimize bandwidth consumption of streaming 360-Degree Video.

3.4 Head-Motion Prediction Model

In this section, we will explain our approach of building a prediction model for head-motion.

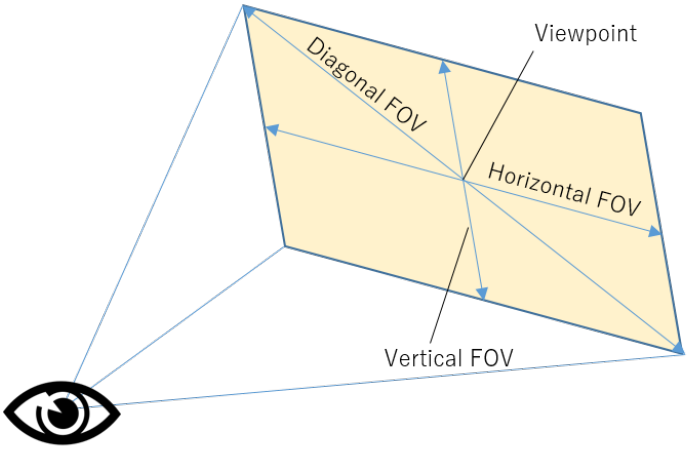


Figure 3.3: Field of View

3.4.1 Head-Motion in VR Gaming

In video gaming, Field of View (FOV) [40], which means the extent of the observable world that can be seen, is used to determine the corresponding portion that needs to be rendered and displayed on player's end. As shown in Figure. 3.3, Vertical FOV is the range of angle from the top to the bottom, horizontal FOV means the furthest left to the furthest right while the diagonal FOV is from the top-left corner to the bottom-right. In addition, viewpoint refers to the center of the image that the player is watching.

As for VR gaming, vertical, horizontal, diagonal FOV are fixed values defined by the HMD specification. For example, HTC Vive [41], which is the HMD that we use in our work, has 145-degree diagonal FOV, 100-degree horizontal FOV and 110-degree vertical FOV. Therefore, the view contents are mainly determined by the viewpoint, which depends on the pitch, yaw and roll angles as shown in Figure.3.4. These three Euler's angles (pitch, yaw, roll) correspond to head's rotation around the X, Y, Z axis, which are values recorded by sensor embedded in the HMD. In our work, the values of these three angles are represented in the units of quaternions, which include four values: qw, qx, qy and qz. Therefore, in this work, our goal is to create an individual prediction model for each of these four quaternion values.

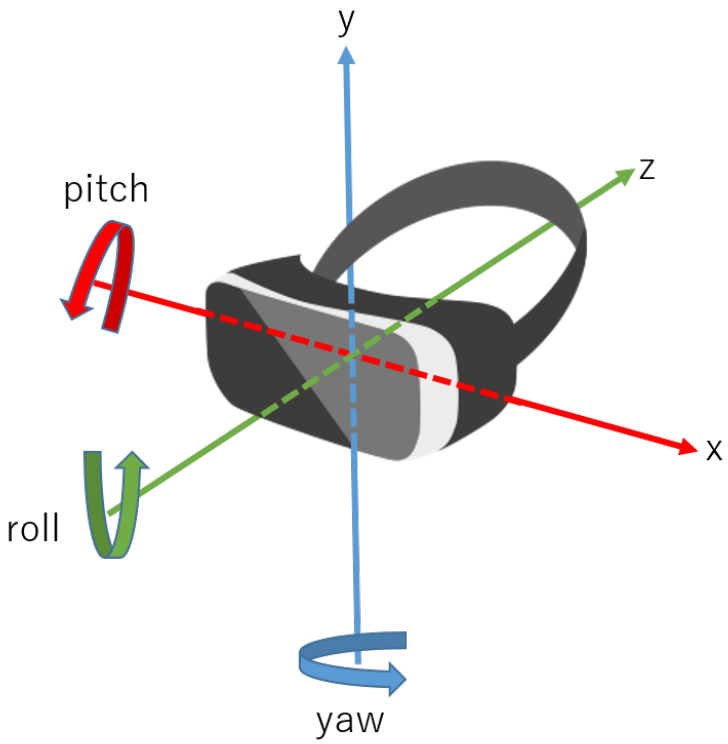


Figure 3.4: Pitch, Yaw, Roll angles

3.4.2 Requirement for prediction window

In the last subsection, we define that we need to build prediction models respectively for the four quaternion values, which are the qw, qx, qy and qz. In this section we mention about the criteria of building such prediction model. In order words, it is crucial to select an appropriate prediction window, which is denoted as Tp. The longer the Tp is, the more time available to transmit the predicted frame, but the less accurate the prediction is. Figure.3.5 shows an overall system of Cloud-based VR gaming that includes several fundamental steps. The first step is the data upstreaming which is required to transmit player's viewpoint from the HMD to the cloud server. The next step is the game process, which includes processing data (quaternions in our case) received from player's HMD, calculating game logics and rendering the game contents. As for the last step, it includes encoding the contents and streaming the contents back to player's HMD. As such,

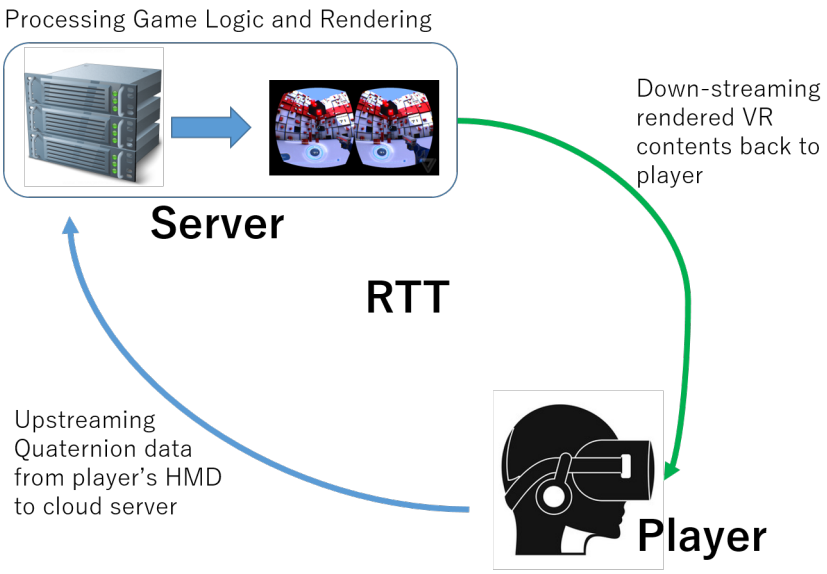


Figure 3.5: Steps that are required in a Cloud-based VR gaming system

Tp should be a value determined within the Round-Trip-Time (RTT), which is the sum of required times for the above steps.

To put it simply, an appropriate value of Tp depends on how the Cloud Gaming system is setup in the network. Referred to previous work conducted by Bao et al, 100 to 500ms is a reasonable range for Tp in the scenario of streaming 360-degree video. Concerning the fact that VR gaming requires much more rigid responsiveness because tasks are processed in real-time, we decided to define our prediction window Tp at 70ms. The value is reasonable because it is approximately the sum of the content processing time (20ms) at cloud server and the time required to stream the content back to player's end (50ms) in a normal broadband network circumstance. Concerning that an usual RTT of Cloud Gaming is at 150ms for normal gaming, we assume that this value as the worst scenario of applying Cloud Gaming to VR gaming, which requires up to 2 to 3 times faster responsiveness. As such, there is around 80ms(150ms - 70ms - 50ms) for upstreaming the quaternion data from player's HMD to cloud server. Assuming that the upstream time is same as the downstream time (50ms), there remains 30ms for accumulating as much previous data as possible to build up the sequence for prediction. In our case, we were able to sample a data at every 2ms, which indicates 15 samples

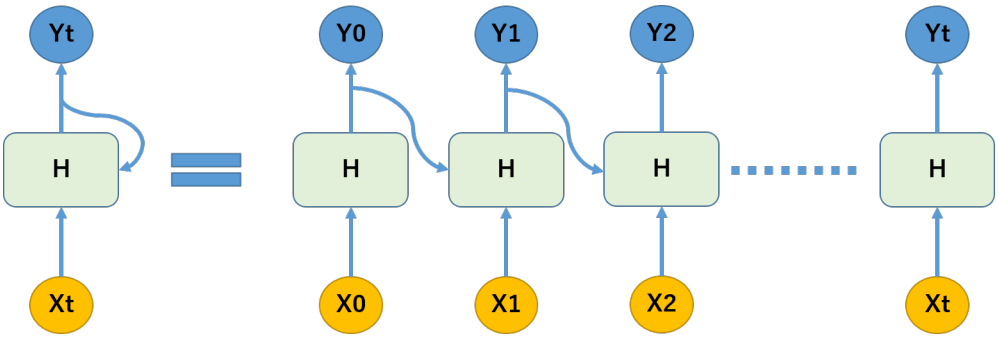


Figure 3.6: An overview of RNN

within 30ms to make a rather accurate prediction at Tp equals 70ms. The details will be mentioned in implementation section.

3.4.3 Prediction

After defining the data to be predicted and the requirement of prediction, we set up our head-motion prediction algorithm for predicting future motions. In our work, we consider that the feature of motion prediction only depends on the past and current motions of the players. Some other potential features, such as game types, interactive in-game elements and sound directions, could play significant role in affecting the prediction. These features will be considered in the future for further improving and generalizing the prediction model.

For predicting future motion, simple technique such as Naive Model, which applies the last observed value as the prediction value, can be used for cost-effective prediction. However, according to previous work, Naive Model has the worst prediction performance for head-motion. Since head-motion is a time-series data, another efficient way that we consider is calculating moving average, which refers to using the average value of previous samples as the prediction value. The performance of moving average will be shown in later section. Furthermore, Linear Regression is another popular method being applied to make time-series prediction. In previous work, Linear Regression performs relatively well in term of head-motion prediction. However, compared to watching 360-degree movie, player tends to have more sudden and polarized random movements. Thus, it is more preferable to apply a prediction algorithm that could handle such nonlinearity

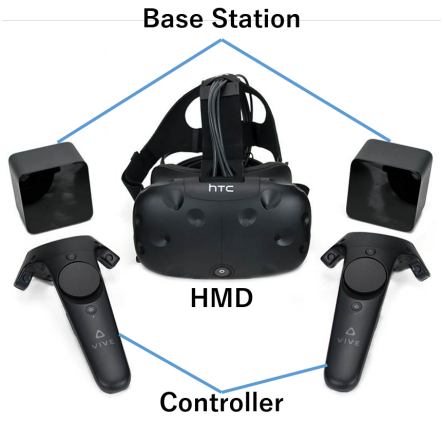
better. In this case, Neural Network is potentially the most suitable method. However, traditional neural network is not designed to use its reasoning about previous events to figure out the later ones. As such, the Nerual Network we mention here specifically refers to Recurrent Neural Networks (RNN), which allows information to persist. As shown in Figure.3.6 , by unrolling a RNN loop, it is actually a concatenation of multiple copies in which the output of last network is fed into the next network. One problem regarding RNN is that if the gap between relevant information and the prediction becomes large, the model is unable to learn to connect the information. As for our goal to build a generic prediction model, it is necessary to persist all necessary information long enough in order to make the prediction generalized. Therefore, Long Short Term Memory networks [42], named as LSTMs, is applied in our work. LSTMs is a special kind of RNN which specializes at remembering information for long periods.

As a summary of the prediction strategy in our work, we applied both Moving Average and LSTMs to create our prediction model. We compare the performances of these two method based on the MSE results retrieved from the models of each person's quaternion data. Furthermore, we also conducted 10-fold cross validation to evaluate if the LSTMs prediction model is overfit or not. The results will be shown in later section.

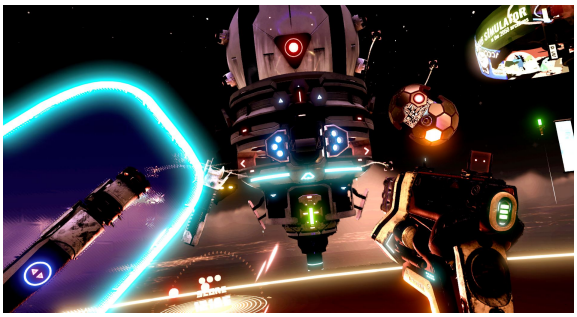
3.5 Implementation

In order to collect data and build prediction model for head-motion, we setup an experimental environment with HTC Vive. As shown in Figure.3.7(a), We demonstrated the setting in an Open Campus that occurred in November 2017, and a total of 45 volunteers joined the experiment. Each participant was asked to play a VR shooting game titled as "Space Pirate Trainer" [43] for 4 minutes. Before each participant played the game, she/he was asked to fill in a simple questionnaire about her/his gaming experiences and whether she/he has ever experienced VR before. After her/his VR section, the player was asked to answer if she/he felt about the experience. During each participant's VR section, the quaternion data of head-motion was simultaneously logged at our gaming server.

(a) Experiment setup in Open Campus



(b) HTC Vive HMD



(c) Tested Game: Space Pirate Trainer

Figure 3.7: Experiment setup for our work

3.5.1 Hardware and Software

In our hardware setup, the game server machine is equipped with an Intel Core i7 4770K 3.5GHz CPU, a Asus Nvidia GeForce GTX780 graphics card DirectCU ii OC(Core clock at 889MHz with 3072MB GDDR5 video memory)[‖] and 16GB system memory, running on a 64-bit Microsoft 10 system. As mentioned previously, we use the HTC Vive as our HMD hardware, as shown in Figure.3.7(b).It is considered that HTC Vive provides the most comprehensive VR gaming experience compared to other market rivals, due to the Lighthouse tracking system [**?**]. Lighthouse is a laser-based positional tracking system developed by Valve, which

[‖]GTX 780 DirectCU II: https://www.asus.com/graphics-Cards/GTX780DC2OC3GD5/

Table 3.1: Data of Participants in our experiment

Age	Male	Female
5-20	29	10
20-30	1	0
30-40	0	2
40-50	2	1
Total	32(71%)	13(29%)

is able to accurately tracks not only the orientation of the HMD as well as the controllers, but also the positions of the devices. The Base Stations are able to sense the movement of player within a maximum range of 4m x 4m area. In our experimental setting, a 2m x 2m was reserved for participants to move around during their VR sections. Furthermore, Lighthouse tracking system is able to perform sampling rate at 1000Hz.

As for the gaming software, we selected Space Pirate Trainer, which is a VR shooting game shown in Figure.3.7(c). The gameplay requires player to use their controllers on both hands, which are represented as a pair of weapons, to shoot down continuous waves of enemies. The player could also shift to different weapon or even shield for better attack or defense. As the wave goes on, the difficulties of enemies increase. In each round of gameplay, player is granted 3 health points, and game-over screen shows up when the player lost all his health points. Unless the 4-minute section was over, The participants in our experiment were instructed to continue a new round of gameplay even after game-over. We consider that the game is a suitable choice for our experience because the participant needs to continuously move their heads to aim at the flying enemies, which should create desired data for our prediction model. Moreover, the gameplay is simple enough that even participants who have their first-time VR experience in our experiment, they could easily get used to the controls in a rather short period.

Table 3.2: A example of dataset from one participant

Index	quaternion w	quaternion x	quaternion y	quaternion z
1	0.753446	0.0109914	-0.657079	-0.0210848
2	0.754943	0.0109193	-0.655353	-0.0213258
3	0.758531	0.0107283	-0.651181	-0.0219136
4	0.761459	0.0104646	-0.64774	-0.0224307
5	0.765097	0.0100987	-0.643419	-0.0231574

3.5.2 Subjects and Questionnaire

In our experimental demo during open campus, a total of 45 volunteers joined the experiment, in which each of them conducted the 4-minute VR section as mentioned in previous section. Most of the participants are visitors of the open campus, and most of them had their first VR experience in our experiment section. Ash shown in Table 3.1, the age distribution is as follows: 87% of them are between age from 5 to 20, 2% of them are at their 20s, 4% of them are at their 30s and the remaining 7% is between 40 to 50. Furthermore, 29% of the participants are females.

Furthermore, all participants were required to answer some questions regarding their video gaming habits. Questions include whether the participants play video games from time to time, how often do they play video games every week and what type of games do they play the most. Furthermore, the questionnaire also questions if the participant had ever experienced VR before our experiment section. After the participant finishes her/his section, she/he is required to fill in if she/he has any uncomfortable feeling such as nausea and dizziness. Participants could also optionally fill in more details about their experiences during the experiment section. The answers to the questionnaires will be later summarized in discussion section, as it will provide further insight to our prediction model.

3.5.3 Data Collection

Our overall data contains 45 view from all the participants, and a total of 180 minutes of head-motions were recorded. As mentioned in previous section,

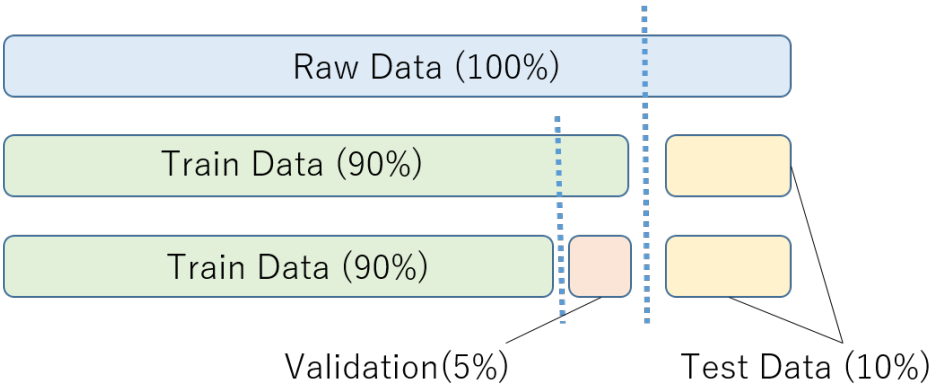


Figure 3.8: Train Data and Test Data for RNN

Lighthouse tracking system is able to sample at 1000Hz, which means 1 sample per millisecond. Our data collecting software at the game server can approximately log the data at the same rate, but due to the processing load on hardware, the actual sample rate written to log file is around 2ms. Furthermore, rather than utilizing the whole sequence of 4-minute data, we extract a one-minute segment in the middle, which the segment is from second to third minute. It is due to the reason in the first minute the participant needed to adjust herself/himself into the immersive VR environment, and we pick the following minute to guarantee our experiment dataset is consistent. Therefore, in each participant dataset, each 4 quaternion value contains 30000 samples, which means in total we have 5400000 samples from all the 45 participants. An example of dataset is shown in Table 3.2.

3.5.4 Prediction Models and Validation

In this section, we discuss about building the prediction models. It mainly consists of defining suitable input data (observed data) and output data (target prediction data) for RNN training, building the two prediction models respectively based on LSTM and Moving Average, and validating the prediction models.

Since the prediction of future head-motion is based on the current and past status, the four quaternion values, qw, qx, qy, qz are treated as features in the predictions models. Assuming qw_t, qx_t, qy_t, qz_t are denoted as the four quaternion values at time t. Furthermore, $qw_{t1:t2}$, $qx_{t1:t2}$, $qy_{t1:t2}$ and $qz_{t1:t2}$ refer to the

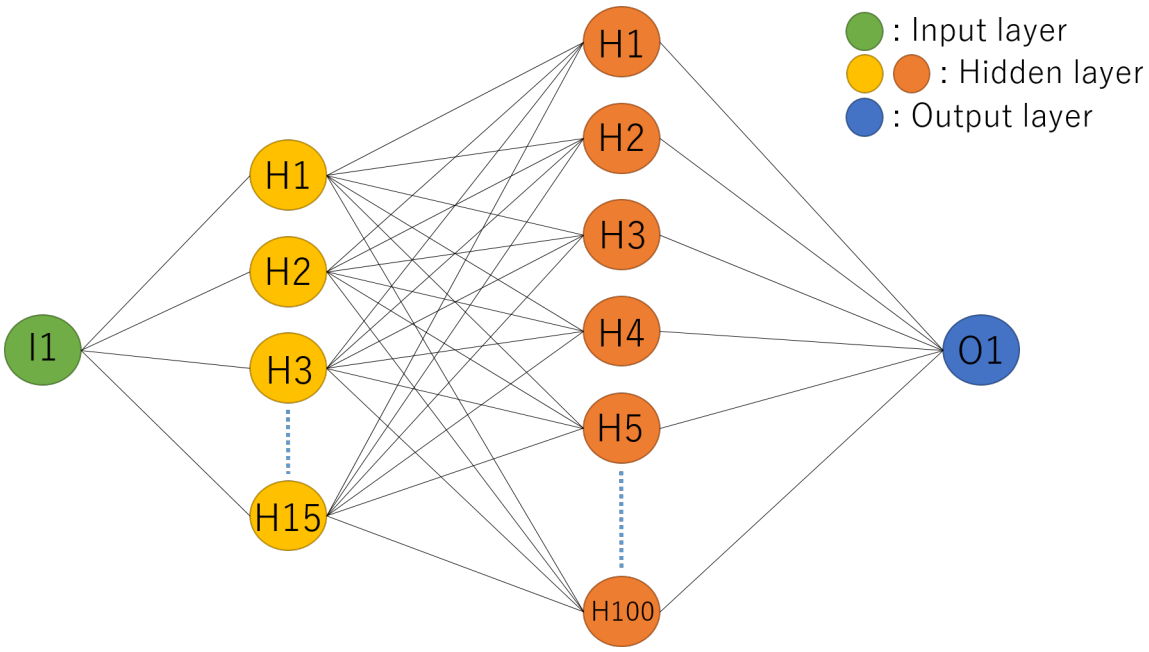


Figure 3.9: The RNN used in our work

sequence of quaternion data fomr time t1 to time t2, respectively. For example, in our case, $qw_{t1:t2}$ = (qw_{t1}, qw_{t1+2}, qw_{t1+2} ....,qw_{t2}) and $qz_{t1:t2}$ = (qz_{t1}, qz_{t1+2}, qz_{t1+2} ....,qz_{t2}). Therefore, with observed data $qw_{t1:t2}$, $qx_{t1:t2}$, $qy_{t1:t2}$ and $qz_{t1:t2}$, we aim at predicting (qw_{t2+Tp}, qx_{t2+Tp}, qy_{t2+Tp}, qz_{t2+Tp}). In last section, we made our assumption that in order to predict head-motion at 150ms with a Tp at 70ms, the model is given approximately a 30ms window to accumulate a sequence of observed data. In addition, the interval between each sample of quaternion data is 2ms. For facilitating our study, we define this observed data is a sequence of 15 continuous data (30ms/2), thus t2 equals t1+30 while the target prediction data is at t2+70. Furthermore, we treat each quaternion as an individual time-series regression problem and train separate models respectively for the prediction of the 4 quaternion values.

As for the prediction, we first build a prediction model based on the simple method of Moving Average. The 30000 samples of each quaternion value are grouped into sets of 15 continuous data, such as in the form of $qz_{t1:t2}$, and we calculate the average of each set. The result is used as the prediction value at

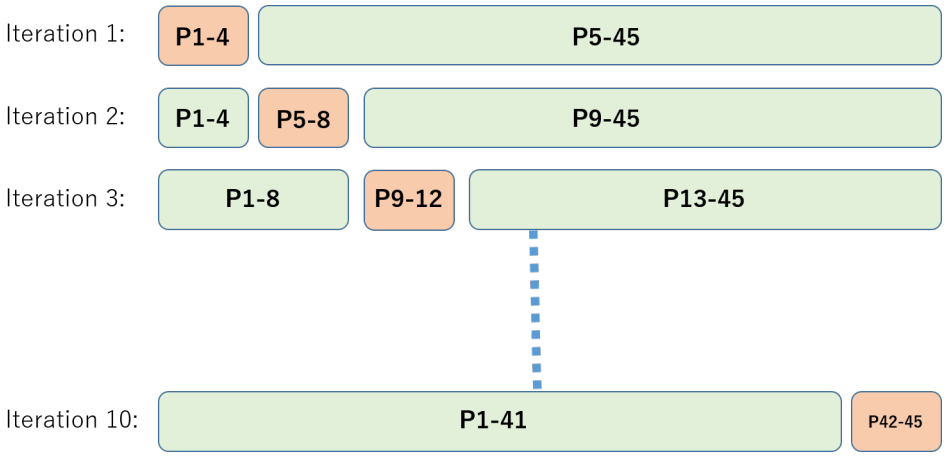
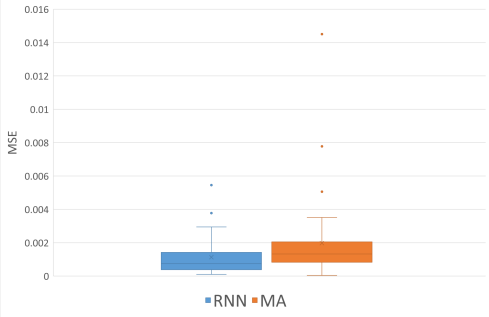


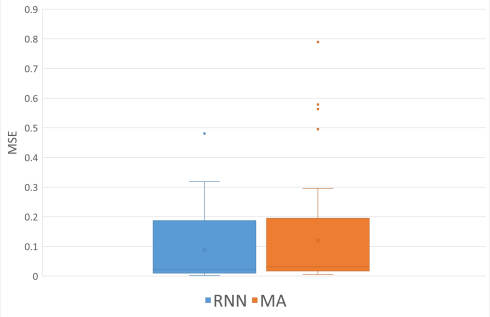
Figure 3.10: 10-fold cross validation

t2+70, which will be compared with the actual target data at t2+70 from the samples for evaluation. As for the RNN model, we need to define the input data, which consists of sets of 15 continuous data, and the output data, which are the corresponding t2+70 target values for training the model. Furthermore, as shown in Figure. 3.8, the whole set of 30000 samples is separated into train data, which is used to train the RNN, and test data for evaluating the accuracy. 5% of the train data is also used for validation. The ratio of train data and test data is 9:1. For building the LSTM-based RNN, we utilized Keras which is based on Tensorflow [44]. The RNN, which is shown in Figure. 3.9, consist of one input layer which feeds in a sequence of size 15, two hidden layers which one layer has 15 LSTM neurons and another has 100 LSTM neurons, and a single output layer. Furthermore, the model training is set to have a batch size of 512 with 10 epoch. Each participant's data is trained separately, which means that each participant has four prediction models respectively for the four quaternion values.

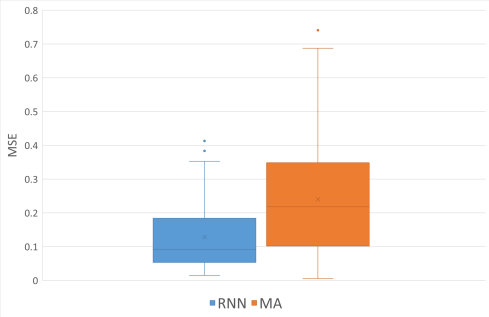
For evaluation, we feed in the test data and calculate the mean absolute error (MSE). Afterwards, we compare the result with the MSEs from the method of Moving Average. Furthermore, training the data separately may lead to overfit, so we also took all the samples from 45 participants as a whole and performed 10-fold cross evaluation, as shown in Figure. 3.10. In each iteration, the data of 41 participants was used to train a comparatively more generalized prediction model, and the data of remaining 4 participants was used as test data for evaluation. All



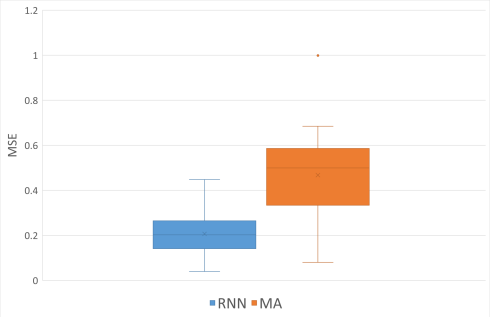
(a) box plot of qw prediction model trained by Moving Average and RNN



(b) box plot of qx prediction model trained by Moving Average and RNN



(c) box plot of qy prediction model trained by Moving Average and RNN



(d) box plot of qz prediction model trained by Moving Average and RNN

Figure 3.11: Performance Comparison between RNN and Moving Average

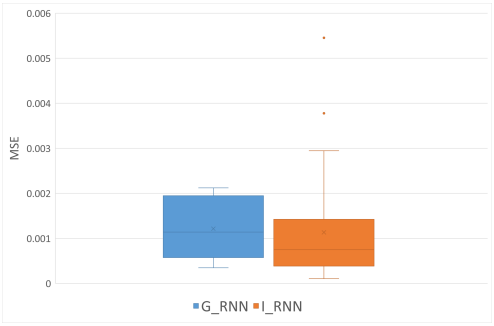
the results will be shown in the next section.

3.6 Results

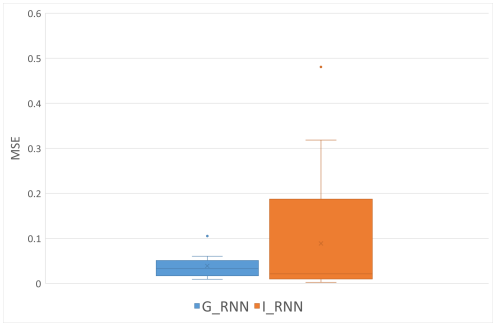
After training the mentioned prediction models, we utilize both Mean Absolute Error (MSE) to evaluate the result.

3.6.1 Results of RNN and Moving Average

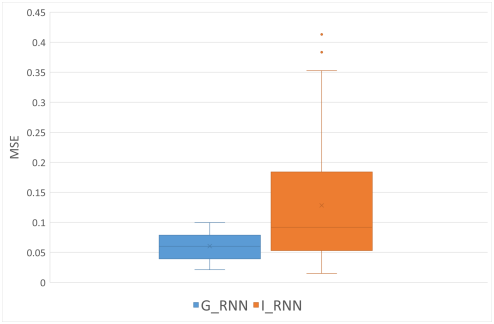
Both Table. 3.4 and Table. 3.5 show the MSEs of all the prediction models we built, which include the individual model of qw, qx, qy and qz respectively trained by using Moving Average (MA) and RNN. As we can see from both



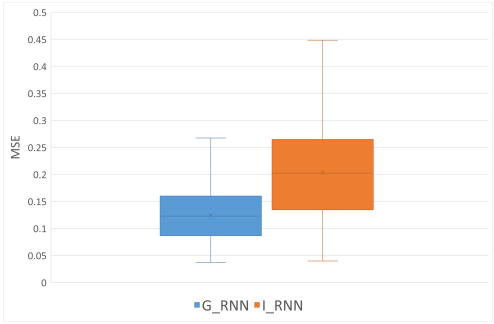
(a) box plot of qw prediction by individual RNN and generalized RNN



(b) box plot of qx prediction by individual RNN and generalized RNN



(c) box plot of qy prediction by individual RNN and generalized RNN



(d) box plot of qz prediction by individual RNN and generalized RNN

Figure 3.12: Performance Comparison between individual RNN and generalized RNN

tables and Figure. 3.11, overall the prediction model trained by RNN outperforms the one from Moving Average in term of accuracy. The difference is especially more obvious in the case of qw, qy and qz. In the case of qx, both models have relatively closer result, but it is not difficult to conclude that RNN is the better one. Furthermore, RNN prediction model also demonstrates higher consistency with shorter high-density region, closer extremes as well as fewer outliers. In addition, we also show qw, qx, qy and qz prediction example from one participant by plotting both the true data and prediction value within a 6000ms (3000 indexes in the graphs) region, which are shown from Figure. 3.13 to Figure. 3.16. It becomes more obvious that RNN could make better prediction pattern compared with Moving Average, especially when the data movement tends to be more

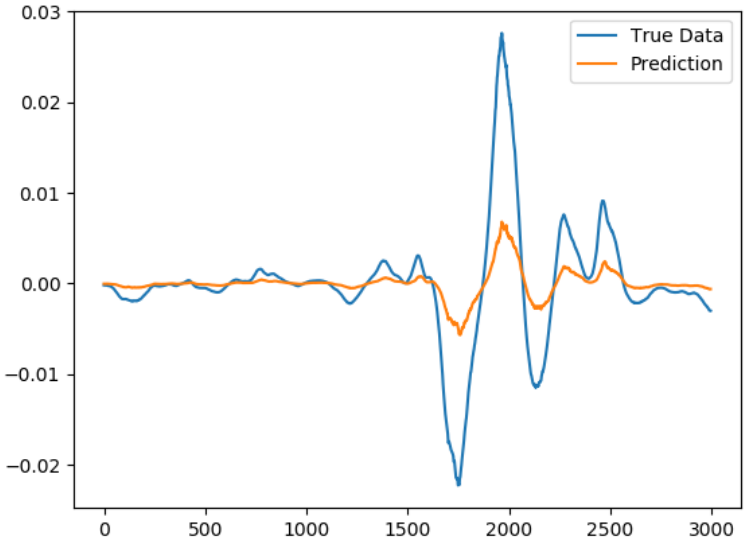
Table 3.3: Result comparison between Individual RNN and Generalized RNN

	qw		qx		qy		qz	
	I_RNN	G_RNN	I_RNN	G_RNN	I_RNN	G_RNN	I_RNN	G_RNN
Average	0.001132	0.001214	0.0887	0.0388	0.1282	0.0603	0.2064	0.1244
Median	0.000749	0.001073	0.0211	0.0134	0.0918	0.0533	0.2021	0.1226

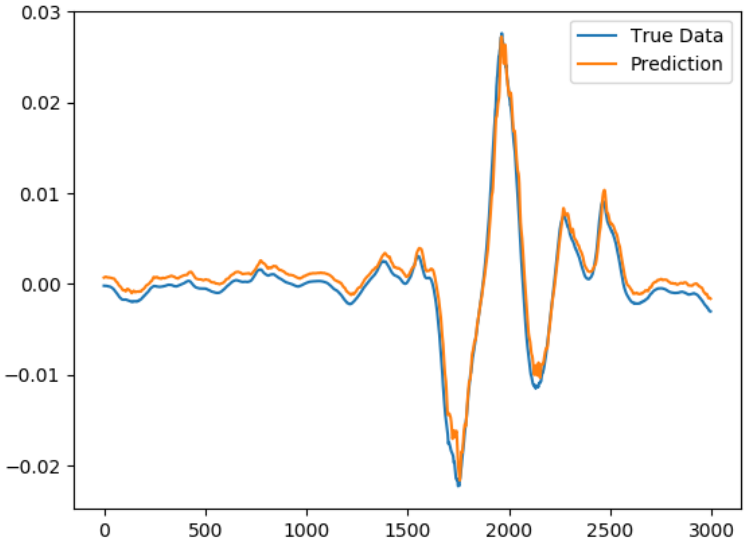
linear. However, in the cases of both qy and qz shown in Figure. 3.15 and Figure. 3.16, both models cannot predict the moving distance well when there is a spike movement, where RNN performs slightly better than Moving Average.

3.6.2 Results of 10-fold cross validation

In the last subsection, the result indicates that RNN is the better way to predict the future value. However, training individual prediction model for each participant's data could make the model overfit for the corresponding dataset, which makes the model not generalized for being used in other dataset. Therefore, 10-fold cross validation is applied for validating if the prediction model can be trained for generalizing the usage on different participant's dataset. After the 10 iterations of cross validation, we calculate both the average and median of MSEs retrieved from all the iterations, which the results are shown in Table 3.3. Furthermore, we also compare the performance of the generalized RNN model trained in 10-fold cross validation (denoted as G_RNN) with individual RNN model (denoted as I_RNN), which the results are presented in 3.12. In the case of qw, I_RNN has overall better prediction in term of accuracy. However, G_RNN has more consistent performance indicated by closer extreme values and outliers. Furthermore, for qx, qy and qz, G_RNN has much better performance than I_RNN in terms of both accuracy and consistency. The RNN performance can be further explained from Figure. 3.13 to Figure. 3.16. In the case of qw and qx, prediction results from both G_RNN and I_RNN are able to accurately follow the trend of the true data. However, in the case of qz, where there is spike movement, G_RNN could predict such sudden change better than I_RNN.

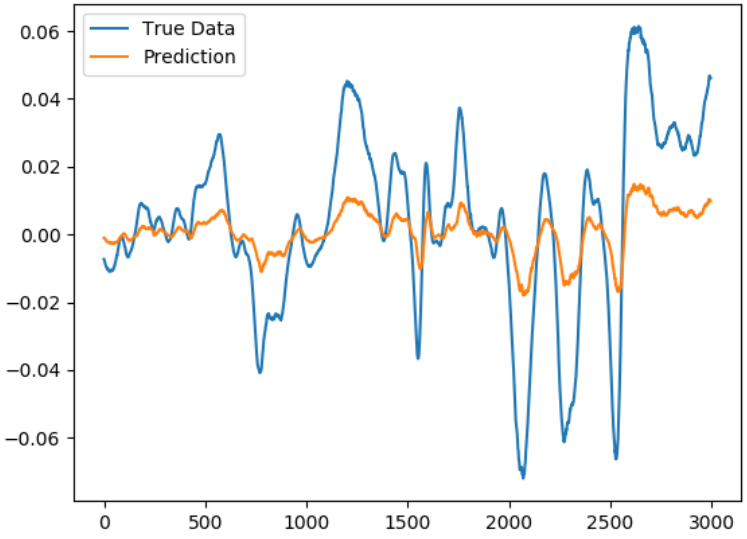


(a) qw prediction using Moving Average

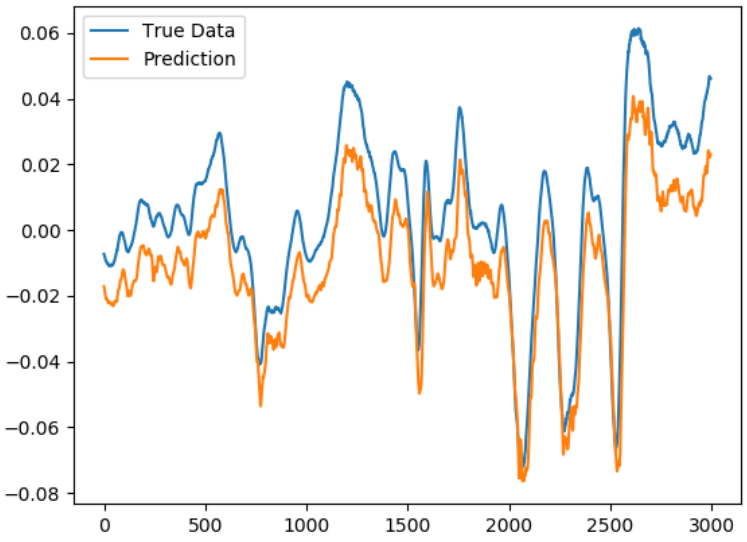


(b) qw prediction using RNN

Figure 3.13: Plot of qw from one of the participants

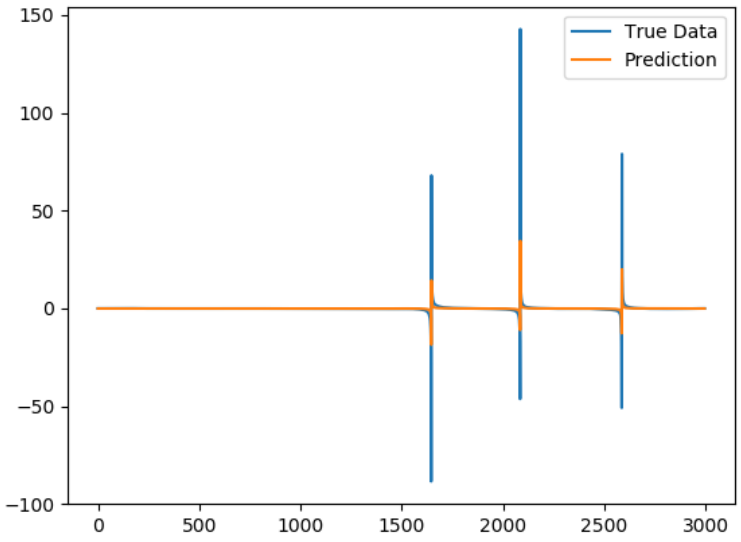


(a) qx prediction using Moving Average

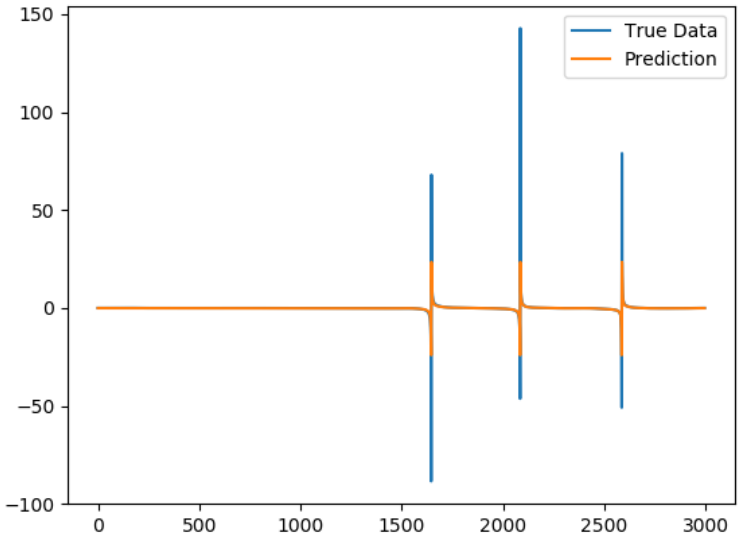


(b) qx prediction using RNN

Figure 3.14: Plot of qx from one of the participants

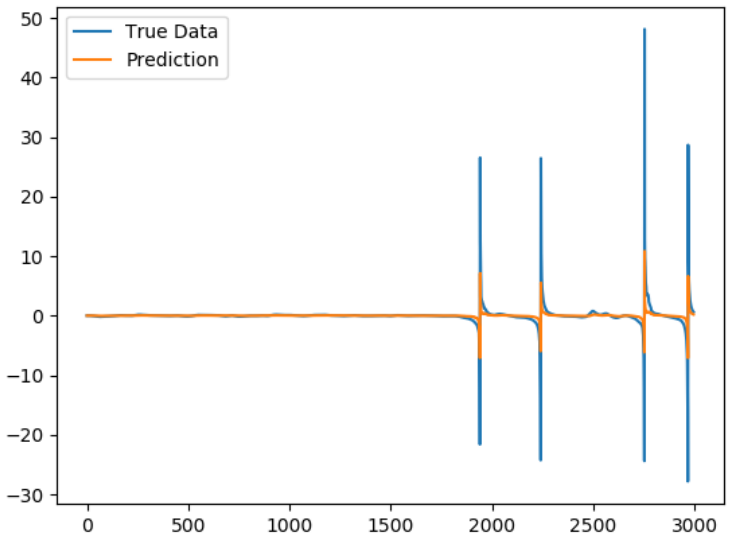


(a) qy prediction using Moving Average

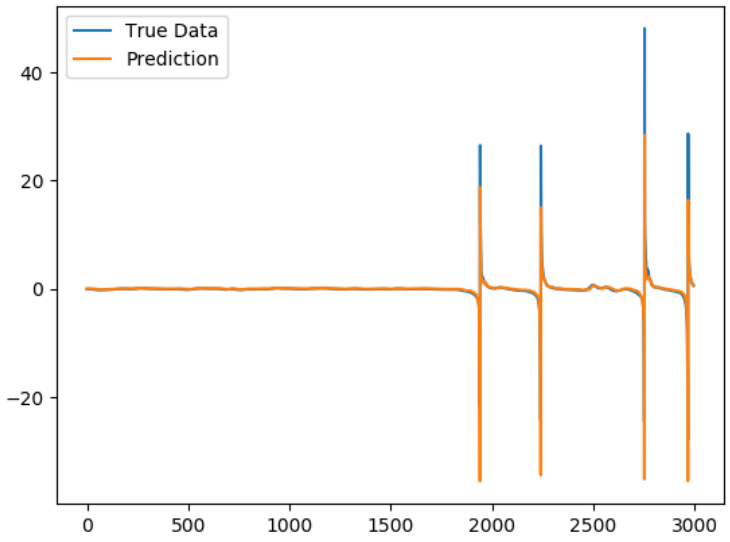


(b) qy prediction using RNN

Figure 3.15: Plot of qy from one of the participants

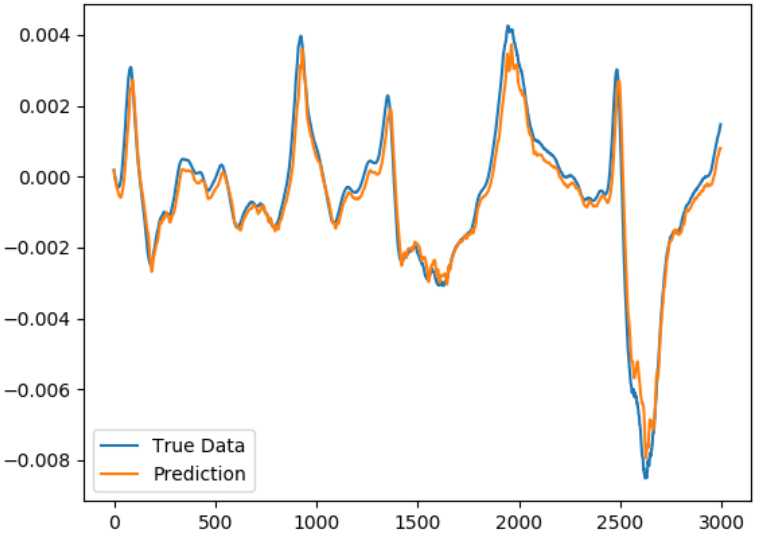


(a) qz prediction using Moving Average

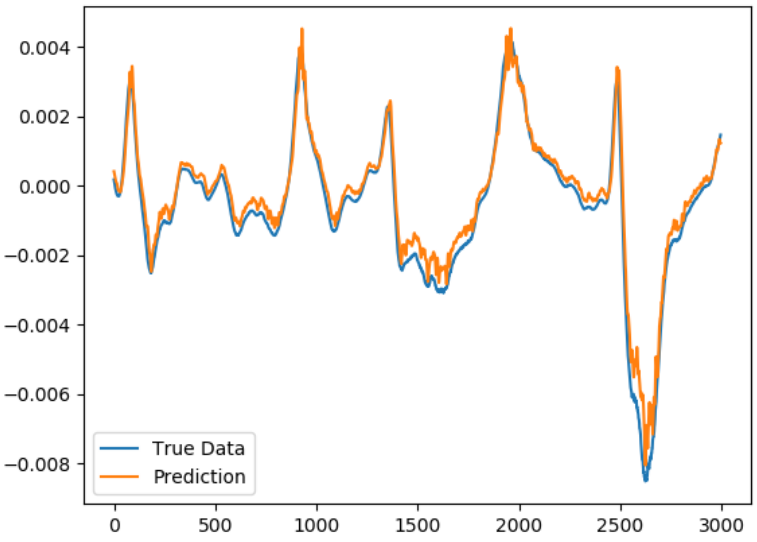


(b) qz prediction using RNN

Figure 3.16: Plot of qz from one of the participants

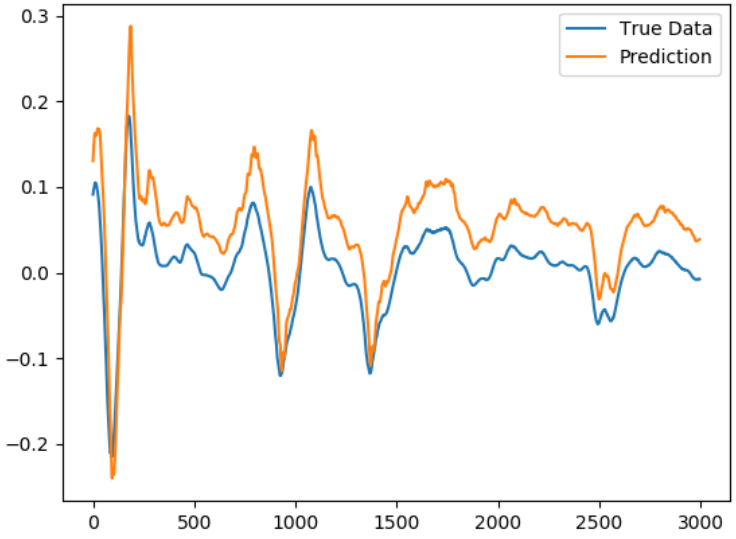


(a) qw prediction using individual RNN

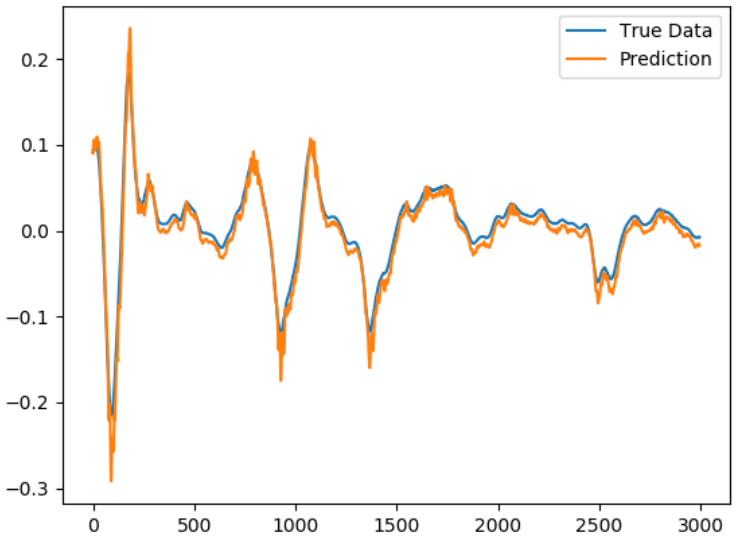


(b) qw prediction using generalized RNN

Figure 3.17: Comparison of qw between individual RNN and generalized RNN

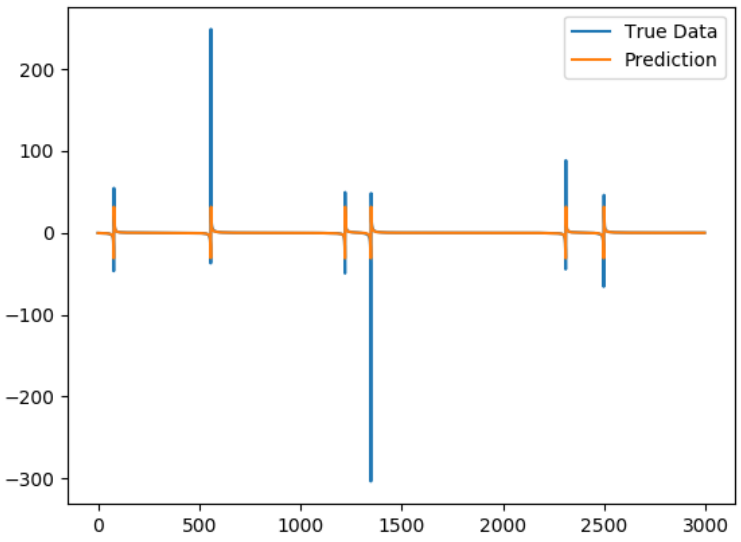


(a) qx prediction using individual RNN

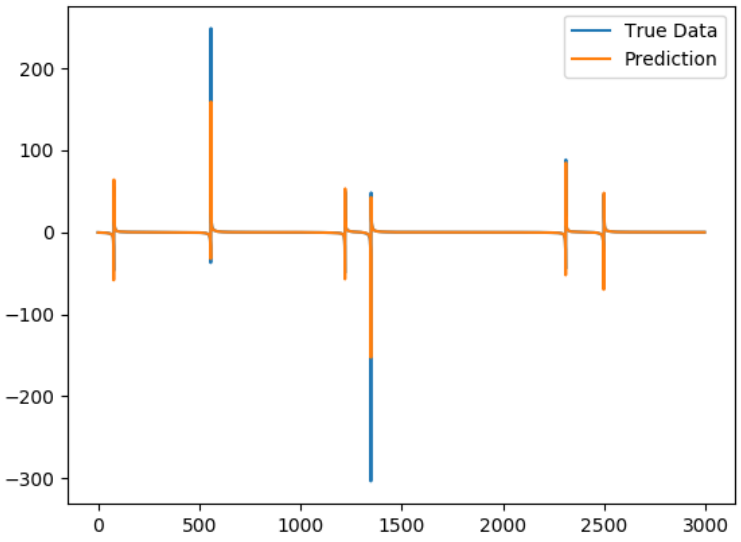


(b) qw prediction using generalized RNN

Figure 3.18: Comparison of qx between individual RNN and generalized RNN

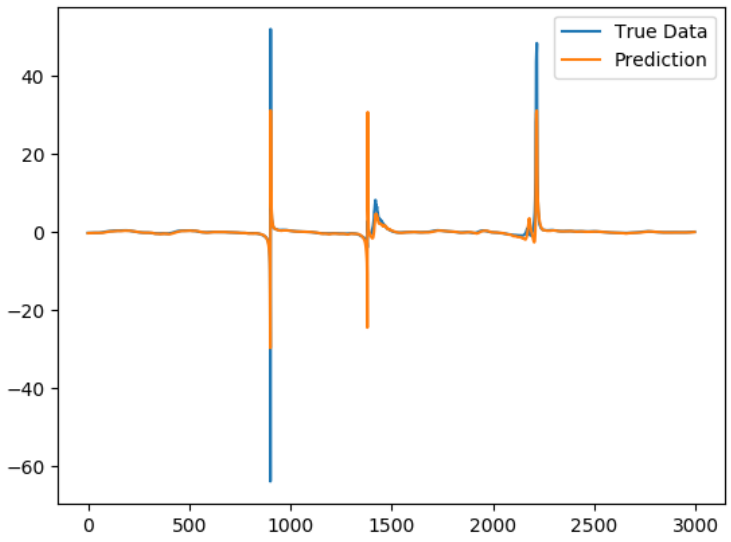


(a) qy prediction using individual RNN

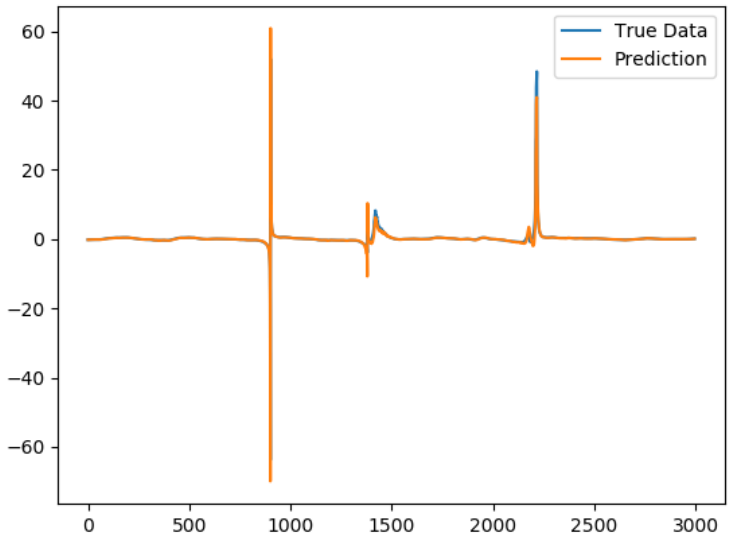


(b) qy prediction using generalized RNN

Figure 3.19: Comparison of qw between individual RNN and generalized RNN



(a) qz prediction using individual RNN



(b) qz prediction using generalized RNN

Figure 3.20: Comparison of qz between individual RNN and generalized RNN

Table 3.4: MSE results from Moving Average and RNN (This table is showing data from participant 1-30)

Person	qw		qx		qy		qz	
	MA	RNN	MA	RNN	MA	RNN	MA	RNN
1	0.003343	0.002158	0.5631	0.2169	0.0610	0.0149	0.5730	0.3242
2	0.000712	0.002548	0.2024	0.2309	0.2290	0.0747	0.2859	0.0856
3	0.001268	0.000657	0.0111	0.0043	0.0402	0.0220	0.5168	0.1946
4	0.003511	0.001111	0.0337	0.0126	0.3672	0.0760	0.6213	0.2815
5	0.003182	0.002359	0.4953	0.4807	0.0297	0.0687	0.9989	0.4481
6	0.000680	0.000203	0.0235	0.0411	0.1114	0.0876	0.0826	0.0395
7	0.001409	0.000748	0.0091	0.0069	0.0051	0.1169	0.2768	0.0788
8	0.000496	0.000582	0.2030	0.0943	0.1486	0.0722	0.5691	0.2643
9	0.000973	0.000104	0.0863	0.2452	0.2138	0.0487	0.0799	0.0600
10	0.002873	0.000245	0.0503	0.2189	0.3274	0.0959	0.2641	0.1457
11	0.001378	0.005453	0.0274	0.0048	0.1637	0.0808	0.0866	0.0824
12	0.000826	0.000256	0.0375	0.0196	0.3006	0.0563	0.3983	0.1928
13	0.007775	0.002664	0.0816	0.3182	0.1084	0.1612	0.4090	0.1415
14	0.002007	0.000724	0.0147	0.0067	0.2747	0.2249	0.5849	0.1408
15	0.005058	0.000656	0.0795	0.1981	0.1613	0.1791	0.4018	0.1995
16	0.000517	0.001107	0.0237	0.0026	0.4500	0.0497	0.4332	0.1885
17	0.001856	0.001986	0.0247	0.0315	0.4956	0.2353	0.5793	0.2056
18	0.001081	0.002945	0.5785	0.2897	0.4308	0.2624	0.4820	0.3620
20	0.000896	0.000339	0.0252	0.0159	0.2176	0.1696	0.4798	0.2028
21	0.001135	0.000631	0.0073	0.0023	0.2362	0.1938	0.6767	0.2858
22	0.000042	0.00038	0.2176	0.0091	0.2190	0.4129	0.5251	0.2147
23	0.014507	0.001687	0.0337	0.0802	0.2607	0.0972	0.2512	0.0695
24	0.002984	0.000935	0.7893	0.2824	0.5240	0.1678	0.6778	0.3140
25	0.001173	0.000364	0.0108	0.0024	0.3852	0.0874	0.5241	0.2423
26	0.001201	0.00145	0.0086	0.0142	0.0283	0.1598	0.6550	0.2775
27	0.000847	0.000955	0.0257	0.0198	0.0309	0.1855	0.6846	0.2504
28	0.000621	0.000749	0.0082	0.0125	0.2284	0.0263	0.3220	0.1320
29	0.000725	0.000184	0.1721	0.0301	0.2061	0.0201	0.2687	0.0726
30	0.001602	0.000396	0.0259	0.0136	0.0303	0.0396	0.5716	0.2504

Table 3.5: MSE results from Moving Average and RNN (This table is showing data from participant 31-45

Person	qw		qx		qy		qz	
	MA	RNN	MA	RNN	MA	RNN	MA	RNN
31	0.000926	0.003777	0.0283	0.0117	0.0326	0.0506	0.2460	0.2880
32	0.001697	0.00042	0.2957	0.1540	0.0994	0.0414	0.2334	0.0724
33	0.001137	0.000248	0.2211	0.1248	0.2318	0.1120	0.3911	0.1721
34	0.001171	0.000334	0.0262	0.0486	0.7405	0.3834	0.4388	0.2595
35	0.000485	0.000393	0.0174	0.0177	0.3819	0.2234	0.5548	0.2015
36	0.000551	0.001183	0.0053	0.0034	0.0202	0.0620	0.6292	0.2647
37	0.002652	0.001134	0.0153	0.0143	0.3552	0.1781	0.6797	0.3398
38	0.001407	0.001338	0.2803	0.2079	0.6873	0.3526	0.5873	0.2314
39	0.000338	0.000592	0.1283	0.0510	0.1893	0.1314	0.6129	0.2397
40	0.001954	0.001815	0.0605	0.0223	0.0558	0.0269	0.4385	0.1835
41	0.002043	0.001141	0.0160	0.0094	0.2182	0.0776	0.3687	0.1259
42	0.001438	0.001041	0.0266	0.0106	0.3266	0.1893	0.5504	0.1701
43	0.002697	0.000999	0.0171	0.0051	0.5700	0.2092	0.5388	0.2258
44	0.001624	0.000266	0.2697	0.2789	0.1809	0.0519	0.5865	0.3720
45	0.002050	0.000579	0.0368	0.0313	0.1780	0.0621	0.4162	0.1864
Average	0.001974	0.001132	0.1208	0.0887	0.2340	0.1282	0.4694	0.2064
Median	0.001324	0.000749	0.0310	0.0211	0.2180	0.0918	0.4994	0.2021

3.7 Discussion

In this section, we will discuss about the evaluations, threats to validity, application in actual VR games as well as the questionnaire results.

3.7.1 Evaluations

Prediction's accuracy and consistency

According to our last section, in terms of both accuracy and consistency, prediction model trained by using LSTMs-based Recurrent Neural Networks outperform the one from Moving Average. When the data point does not change drastically, Moving Average has comparatively closer performance to RNN. However, due to the continuous random nature of head-motion as well as high range of movement within short time, RNN prediction model is the preferred solution to cope with the nonlinear changes. Furthermore, the conducted 10-fold cross validation also reveals that fitting the prediction model with multiple participant's data as training set could make the model generalized for other participants' dataset. The resulted model performs as good as, or even better than the prediction model trained by individual participant's data. However, we also observed that the generalized RNN model seems to have more shaky predictions, and we consider that other means such as methods proposed by [45] can be applied to smooth out the prediction.

Furthermore, our prediction was performed in the scenario of RTT at 150ms, which is assumed to be the worst case of applying Cloud Gaming to VR gaming. Therefore, in usual case, the RTT is shorter, which indicates smaller prediction window. With shorter prediction window, prediction with higher accuracy is expected.

Resource required for training

On the other hands, even though the corresponding data was not logged in this work, the time required for training the prediction model should be considered. Obviously, Moving Average is the fastest prediction model because the prediction value requires only a simple arithmetic calculation, which in our case each MA-based prediction model usually took multiple seconds. In the case of RNN, the

training process is the most time-consuming, as it took approximately 5 minutes to complete the training for all the 4 quaternion values of each participant. Therefore, it also implies that each iteration of cross validation took more than 2 hours. The required time can be shortened by having different RNN setting such as fewer epochs or neurons, but it leads to lower accuracy of the model. Utilizing GPU processing power for training RNN is another practical way to speed up the performance. In Cloud Gaming, clusters of GPUs in powerful servers provide an appropriate environment for such purpose. However, properly allocating resources to efficiently train a more comprehensive prediction model based on larger dataset is required for delivering the actual service.

Difference from previous study

As mentioned in previous section, from technical perspective, Bao et al's work is potentially closest to our work, due to the reason of applying Machine Learning Technique for prediction. However, there is a significant difference in term of application scenario. In our work, we focus on exploiting ML-based prediction model to compensate network latency for streaming VR gaming contents, while Bao et al optimizes bandwidth consumption of streaming 360-Degree Video. In other words, the requirement for prediction is different. For VR gaming usage, the prediction window is smaller (150ms in our case) due to more rigid RTT, compared to 100-500 ms in Bao et al's work. Furthermore, unlike the usage of video streaming which stores all image data at cloud server, each immediate frame is created in real time for VR gaming. Therefore, the prediction accuracy is required to be significantly higher.

3.7.2 Threats to Validation

RNN Model setting

In this research, we apply the same RNN model to train all the data, based on our assumption for prediction requirement mentioned in previous section. We defined that in a normal Cloud Gaming streaming environment with RTT at around 150ms, there is approximately 30ms window to accumulate a sequence of data to predict the value at Tp equals 70ms. Furthermore, we also assumed

Roll_angle = arctan((2*(qw*qx+qy*qz))/(1-2*(qx^2+qy^2)))

Pitch_angle = arcsin(2*(qw*qy – qz*qx))

Yaw_angle = arctan((2*(qw*qz+qx*qy))/(1-2*(qy^2+qz^2)))

Figure 3.21: Conversion to Euler Angles from quaternions

that our software of capturing each head-motion data point at 2ms, which implies a sequence of 15 data points within the 30ms window. Result shows that the RNN model fits well with our assumption setting. However, in the actual network, latency tends to be varied, which means RTT is different from time to time. As well, based on the types of games, the sampling rate could be different. Therefore, it is possible that our RNN model could not suitably fit with such changing conditions. Even though making our prediction model flexible and dynamically adapt to various conditions is not the focus of this work, undoubtedly it needs to be further addressed for practical usage.

Representation of the data

In our work, our method collects and predicts the head-motion represented in the four quaternion values: qw, qx, qy and qz. It is due to the fact that these four values are the raw data retrieved from the Lighthouse tracking system. In order to convert from the representation of quaternions to Euler angles which include roll, pitch and yaw, it is necessary to follow the equation shown in Figure. 3.21. However, we decided to skip the conversion for saving small amount of time in our environment which requires intensive responsiveness. Therefore, it is a fact that the predicted roll, pitch and yaw angles calculated from the predicted quaternion values maybe varied from the actual angle at different deviation. However, from our presented results in last section, our prediction model, especially the generalized RNN model performs rather close prediction compared with the actual data, so the deviation from predicted angle should be not far off from the actual angle. For handling the inevitable deviation, other means can be applied, which will be

discussed in later section.

Testing subjects

One threat to validation regarding our work is about the distribution of our testing subjects (participants). As shown in table 3.1, 39 out of 45 participants (87%) are under 20 years old. This is not a factor we could control because we conducted our experiment on an Open Campus Day, where a lot of visitors are teenagers and children. This may indicate a lack of diversity in term of participants' ages. Moreover, even though most of the players equipped the HTC Vive HMD without any issue, the already light-weight device still presented to be slightly heavy for some younger participants, which may limit their head movements during the gameplay. As an extension of this work, gathering dataset from more participants with more diverse ages may help to train a more comprehensive prediction model.

Types of Games and VR System

In our experimental setup, we apply "Space Pirate Trainer", which is a VR-only shooting game as our immersive VR gaming software for the participant to experience with. As mentioned in previous section, the control requirements for the chosen VR game is simple that even first-time player could easily get used to. As the head-motion data is retrieved from participants' playing this game, it also indicates that our RNN model is specifically trained for head-motions involved in the corresponding gameplay. Therefore, it is possible that the trained model cannot be fitted well with the other types of games, which involve different patterns of head movements. It is due to the reason that in Space Pirate Trainer, most of the interactive elements, such as the enemies, only show up in the front direction. Most of the time participants only need to look forward or up to interact with the in-game elements, as they seldom turn around or look behind. We consider that for practical usage, there is a need to train varied models respectively for different types of games, but further investigation is needed to judge if this viewpoint is valid.

Furthermore, we conducted the experiment using HTC Vive, and we cannot confirm if other VR devices such as Oculus Rift or smartphone devices collect and process head-motion data in the similar way. However, we consider that it should

be easy to adapt the model to different devices because ultimately, it belongs to the same time-series regression problem.

3.7.3 Application in practical usage

In our work, we propose using RNN as the preferred method for predicting head-motion and present the corresponding result, without actually applying it in the actual gaming software. However, we consider that the model is highly adaptable to gaming software, without necessity to largely alter the software source code. Moreover, there is a need to retrain or modify the RNN model to fit with different types of games or network conditions. Such related steps should be performed and evaluated prior to deploying on the Cloud Gaming environment.

On the other hands, as mentioned in previous subsection, even though the model trained by RNN could predict quaternion values at high accuracy, the predicted head-motion (roll, pitch and yaw) represented by those values is most likely not 100% correct compared to the true data. Therefore, for dealing with such inevitable deviations, corresponding countermeasure is required. One way is that we could define threshold values based on the error deviations retrieved from evaluating the prediction model. We can classify the deviations into multiple patterns based on the derivative of changes, which mean that the bigger change is, the bigger error it usually indicates. For each of these patterns, we assign a corresponding threshold value. In the Cloud Gaming server, GPUs render redundant contents based on the threshold value. Therefore, the redundant content can be used to compensate the deviation between the predicted result and the true representation.

Furthermore, the above method may lead to higher bandwidth because of the redundancy. As such, foveated rendering and or reapplying pixels from previous frame on border content can be used to maintain efficiency.

3.7.4 Questionnaire Results

During our experiment sections, we also required each of our participants to fill in a simple questionnaire regarding their gaming habits as well as VR experiences. According to our collected data, 39 out of the 45 participants play video games for at least one hour per week regularly, with a maximum value at 10 to 20 playing

hours per week. Therefore, we consider that most of our participants are familiar with the concept of video gaming. Furthermore, among these 39 participants, 16 of them had VR experiences through other means prior to our experiments. The VR experiences include mostly watching 360-degree video, special navigation in sky, VR-involved roller coaster in amusement park. However, only one participant had experience with VR gaming. Therefore, for our participants, VR is relatively not a new concept but VR gaming is a new experience for most of the participants. Our chosen VR gaming software, "Space Pirate Trainers", can be considered as suitable because even first-time player could easy get used to the control.

What is more, our participants also provided us with their subjective comments after their VR gaming experiences through our experiment. Most participants were impressed with the immersive feeling during the gameplay, which can be shown by some comments (translated from Japanese) such as "The attacks from enemies feel very real" and "The environment looks very real but a bit scary". Furthermore, even though most participants expressed the easiness of the gameplay, but some participant found the control is "difficult as first time" and "a bit weird at the beginning". Moreover, 44 out of 45 participants felt physically fine after their VR section with only 1 participant felt dizzy. Despite of the dizziness, the participant expressed great VR experience throughout his section. Many of the participants mentioned theirs desire to have the VR gaming system at home because of the novel gaming experiences, but several of them have the concerns about HMD weight because some younger participants had difficulties to look upward. Regarding this issue, we consider that for further expanding the popularity of VR gaming, it is necessary to make the whole system light-weight with better accessibility. As such, our goal of applying Cloud Gaming on VR gaming is a prospective solution.

3.8 Summary

In this work, we state about the rising popularity of VR gaming and also the barriers that hinder such experience from reaching out to larger market. For improving the accessibility of VR gaming, we propose applying Cloud Gaming technique. However, for making Cloud Gaming a viable option for VR gaming, multiple technical issues needed to be further address. The issues include more

rigid responsiveness, high-quality graphics rendering in real-time, and more efficient transcoding for streaming the VR contents. We also provided a list of existing works that may resolve the respective issues, but none of these previous works were conducted in the context of Cloud Gaming. Furthermore, among the mentioned issues, we consider that rigid end-to-end latency affects the quality of experience in VR gaming the most. Therefore, we propose using head-motion prediction as a mean to compensate the inevitable latency.

For building the prediction model, we define that the four quaternion values: qw, qx, qy and qz are the target data to be predicted. These four values represent the orientation of head-motion. In addition, based on the common condition of a Cloud Gaming environment, we also establish our prediction requirement. As for the prediction models, we propose using RNN and Moving Average as the two candidates of prediction methods. In our actual experiment, we collected head-motion data from 45 participants when they experience VR gaming via our chosen shooting game. Afterwards, we trained the prediction models for each quaternion value based on the two mentioned method, and compared their performances. The conducted evaluation reveals that prediction of RNN outperforms Moving Average in terms of both accuracy and consistency. Furthermore, 10-fold cross validation also indicates that a generalized RNN prediction model has very stable performance.

As an extension of this work, different prediction models can be trained and evaluated by using different types of VR games. In addition, for actually applying the prediction model in a Cloud Gaming environment, it is necessary to deal with the inevitable deviation between prediction and the actual value. As a countermeasure, rendering and streaming redundant in-game contents can be considered.

In our work, we only consider head-motion but a truly immersive VR gaming experience involves body motions as well as the orientations of controllers. Several previous works [46–48] also utilized RNN to train models for predicting body-motion. These are the features that can be further explored for the usage in Cloud Gaming. As well, our previous work Hybrid Streaming [49] method can be utilized for processing some front-ground objects at the client side, as well as mitigating the workload at cloud server.

4 Conclusion

In this dissertation, we present our viewpoint that Cloud Gaming is an uprising, novel gaming service with tremendous potential market value. The purpose of cloud gaming is to provide highly accessible game contents with high-quality to any device at anywhere. However, previous works reveal that maintaining sustainable quality of experience in Cloud Gaming is challenging, because of the inevitable network latency. The limitations imposed in network environment could largely hinder QoE delivered on Cloud Gaming from being competitive with traditional gaming platform. Particularly, real-time responsiveness and graphics quality, which are the two elements that affect player's QoE the most, are addressed extensively by a number of existing works. With a strong base of researches being conducted to propel a rapid development, the increasingly matured Cloud Gaming technology will inevitably play more significant role in future gaming environment, together with the concurrently advancing traditional gaming platforms such as PC or game consoles. With this future perspective in mind, the goal of this research is to bring novel gaming experience by making connection between Cloud Gaming, traditional PC gaming platform and also new generation gaming experience, specifically, VR gaming. As such, by addressing the mentioned two issues, We mainly conducted the following two works:

- Addressing the graphics quality: We propose a Hybrid-Streaming system that utilizes the rendering power at server and client's local PC as a mean to improve the graphics quality
- Addressing the network latency: For potentially expanding the usage of Cloud Gaming on VR gaming, we propose using Machine-Learning-based technique to predict player's head motion as a way to compensate the network latency.

In our first work, the proposed Hybrid-Streaming system is based on two existing streaming methods: Image-based streaming and Instruction-based streaming. In Hybrid-Streaming system, in-game contents are separated into two groups, the foreground object and the background object, based on their depth values. The background objects were processed via Image-based streaming, through which the corresponding contents are rendered at cloud server and streamed back to client's device as encoded video. As for the foreground objects, the contents are rendered locally by exploiting client's GPU, thus the fidelity of graphics can be maintained. At last the foreground objects (original quality) overlay on top of the background objects (encoded quality). We implemented a prototype of Hybrid-Streaming system based on an open source cloud gaming platform called Gaminganywhere, and conducted evaluation on graphics quality by comparing with traditional streaming method used in Cloud Gaming. The result reveals that our Hybrid-Streaming System outperforms the traditional streaming method, where all contents are at encoded quality, in term of graphics quality. Furthermore, further evaluation also shows that server workload is mitigated because rendering tasks are shared with GPU at client's end. As well, the incurred network is maintained at a reasonable level.

In our second work, we state about the increasing popularity of VR gaming and propose applying Cloud Gaming technique as a way to improve the accessibility of VR gaming to more potential customers. However, for making Cloud Gaming as a practical option, multiple difficulties need to be further addressed. Particularly, we address the inevitable latency issue in network environment because it is the most significant factor that affects VR gaming experience. For this reason, we propose head-motion prediction as a mean to compensate the latency issue. In our experiment, we collected head-motion data from 45 subjects and created two prediction models respectively based on Moving Average and LSTM-based Recurrent Neural Networks. We evaluated two prediction models and the result reveals that RNN predictions outperform Moving Average in terms of both accuracy and consistency. Furthermore, 10-fold cross validation also indicates a more generalized RNN prediction model has even stabler performance. Moreover, during our experiment, we also collected feedbacks from the 45 subjects regarding their VR section. The qualitative evaluation shows that the VR gaming experiences

received by the subjects are positive.

For our two conducted research, various directions of extensions can be considered. For Hybrid-Streaming system, it can be further evaluated with actual gaming softwares, and a more concrete mechanism for splitting game contents can be discovered to cope with more complicated scenarios. As for the Head-motion prediction model, applying countermeasure for prediction errors can further improve the practicality of the method. Further, beside head-motion, VR in-game elements can be considered as additional input features for training the prediction model, which may lead to even more comprehensive performance. Cloud Gaming system is a metropolitan consisting of technologies from different fields. Regarding the prospects of future Cloud Gaming Development, several significant research problems spanning over a wide spectrum of different directions can be concerned: The distributed systems [50–52], human-computer interaction [53, 54], quality of experience [55–57], resource allocation/virtualisation [58–60], and dynamic adaption [61, 62]. Addressing these issues could largely help to make Cloud Gaming system more profitable, and more successful to provide high quality and highly accessible services to clients. Considering that Cloud Gaming is still a relatively new research field and there are lot of works needed to be done, dedication is needed to keep brining up novel ideas and contributing to this field in years to come.

Acknowledgements

Being able to complete my academic journey in NAIST as a doctoral student is considered to be one of my most precious and honorable experiences in my life. Not only I could live in the well-equipped, friendly environment of NAIST, and also I have been able to study and conduct research under valuable guidances of many brigt professors. Receiving generous helps, advices and caring from many great friends and great people around me propel me this far to achieve my doctoral dissertation. Time I have stayed in NAIST is so meaningful and priceless, and I would like to sincerely express my gratitude to many helpful ones.

First and foremost, I would like to show my deep appreciation to Professor Hajimu Iida, who grants me the tremendous opportunity to study in Laboratory for Software Design and Analysis. His enthusiasm to educate students and his very bright attitudes create a very positive, healthy research environment of SDLab, in which I have been able to passionately conduct my research and spend 5 wonderful years with the SDLab family. His advices and guidances on my research also give me confidence to continue in the field of Cloud Gaming. As well, Prof. Iida also encouraged me to participate in GEIOT, which largely helps me to pave my way to challenge start-up business in Japan after graduation.

To Professor Hirokazu Kato, I have been feeling so grateful and fortunate to learn essential knowledge of Computer Graphic in his lecture, because such knowledge is largely related to my research on Cloud Gaming. As my dissertation committees, his suggestions given in my Seminar presentation also benefits me with improving some critical areas of my research. Also Prof. Kato also gave me precious advices and guidances when I joined GEIOT, during which I was able to develop more mature sense of business.

Next I would like to also have my deepest thanks to Associate Professor, Kohei Ichikawa. Professor Ichikawa has been providing guidance since 2010, as I was

an internship student in Osaka University. By that time he suggested me to research on visualisation, which eventually drives me to conduct my main study in Cloud Gaming. In the past 5 years, his continuous helps for my research and his supportive attitude to push me forward help me to successfully accomplish the doctoral course. My gratitude towards Pro. Ichikawa is beyond words because without him, I would not have accomplished my journey in NAIST.

Moreover, I would also like to express my gratitude to Associate Professor, Yasuhiro Watashiba. Professor Watashiba gave me valuable advices on writing conference papers, through which I become more skillful in expressing my research professionally in the academic field.

I would like to thank my dissertation committee. Thank you so much for reviewing my dissertation and for many insightful comments and suggestions, which help me to improve the dissertation.

Also I want to specially thank Professor Shinji Shimojo and Associate Professor Susumu Date in Cyber Media Centre of Osaka University for providing a precious opportunity of internship at their lab three years ago. Experience from my internship is a deciding factor for me to pursue my Doctoral study in Japan, which leads me to having Doctoral study in NAIST at present. Furthermore, under their guidances, I obtained important skills and knowledges of Computer Science, through which I became more passionated to continue my study in this field.

In the summer of year 2014, I was offered a great chance to having a one-and-half-month internship at Kasetsart University in Thailand. Through this one month, I was able to receive a lot of significant advices from assistant professor Putchong Uthayopas and associate professor Chanting Chantrapornchai. Their concerns on network environment inspire me with a lot of ideas that could be shaped up for future research. Furthermore, Thai students and friends that I met are very friendly, and their hospitalities make my stay in Thailand so memorable.

Being able to study with many bright students in SDLab makes my Doctral course better than the best. I want to thank to all SDLab members for providing helps to live in japan. Their passions in study and their generous advices on my research encourage me to do better and work harder. Especially I want to express my appreciation to Xin Yang for his advices and inspirations of expanding my research interest into gamification, which I find myself deeply interested in. His

making Furthermore, I also want to specially thank to Erina Makihara, Yoshida Takanobu, Naoki Nakayama and Uemura Kohei for their guidances on Japanese languages, as I become more proficient to communicate with Japanese people around. The time I have spent with SDlab people will be always deep in my heart.

Finally and the most importantly, from the bottom of my heart, I want to have my most sincere thank you to my beloved parents, younger sister and my grandparents. For your unconditional love and endless supports since everything began, I am able to overcome difficulties and achieve what I have now. It is wordless to express my greatest gratitude to you, who let me to chase for my dreams. By completing this Doctoral course, I am able to finally fulfill my promise that I made with you since I was young. I hope to dedicate this dissertation to you.

References

[1] Ryan Shea, Jiangchuan Liu, EC-H Ngai, and Yong Cui. Cloud gaming: architecture and performance. *IEEE Network*, 27(4):16–21, 2013.

[2] Onlive web page. `https://www.onlive.com`, 2014.

[3] Gaikai web page. `https://www.gaikai.com`, 2014.

[4] Chun-Ying Huang, Cheng-Hsin Hsu, Yu-Chun Chang, and Kuan-Ta Chen. Gaminganywhere: An open cloud gaming system. In *Proceedings of the 4th ACM multimedia systems conference*, pages 36–47. ACM, 2013.

[5] Tomio Geron. Sony to acquire cloud gaming startup gaikai for $380 million. http://www.forbes.com/sites/tomiogeron/2012/07/02/sony-to-acquire-cloud-gaming-startup-gaikai-for-380-million, 2012.

[6] Josh Lowensohn. Sony buys streaming games service onlive only to shut it down. http://www.theverge.com/2015/4/2/8337955/sony-buys-onlive-only-to-shut-it-down, 2015.

[7] Cloud Gaming Report 2012. Distribution and monetization strategies to increase revenues from cloud gaming. http://www.cgconfusa.com/report/documents/Content-5minCloudGamingReportHighlights.pdf, 2012.

[8] Kuan-Ta Chen, Chun-Ying Huang, and Cheng-Hsin Hsu. Cloud gaming onward: research opportunities and outlook. In *2014 IEEE International Conference on Multimedia and Expo Workshops (ICMEW)*, pages 1–4. IEEE, 2014.

[9] Victor Clincy and Brandon Wilgor. Subjective evaluation of latency and packet loss in a cloud-based game. In *2013 Tenth International Conference on Information Technology: New Generations (ITNG)*, pages 473–476. IEEE, 2013.

[10] Kuan-Ta Chen, Yu-Chun Chang, Po-Han Tseng, Chun-Ying Huang, and Chin-Laung Lei. Measuring the latency of cloud gaming systems. In *Proceedings of the 19th ACM international conference on Multimedia*, pages 1269–1272. ACM, 2011.

[11] Michael Jarschel, Daniel Schlosser, Sven Scheuring, and Tobias Hoßfeld. An evaluation of qoe in cloud gaming based on subjective tests. In *2011 Fifth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS)*, pages 330–335. IEEE, 2011.

[12] Lingfeng Xu, Xun Guo, Yan Lu, Shipeng Li, Oscar C Au, and Lu Fang. A low latency cloud gaming system using edge preserved image homography. In *2014 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6. IEEE, 2014.

[13] Wei Cai, Conghui Zhou, Victor Leung, and Min Chen. A cognitive platform for mobile cloud gaming. In *2013 IEEE 5th International Conference on Cloud Computing Technology and Science (CloudCom)*, volume 1, pages 72–79. IEEE, 2013.

[14] Davy De Winter, Pieter Simoens, Lien Deboosere, Filip De Turck, Joris Moreau, Bart Dhoedt, and Piet Demeester. A hybrid thin-client protocol for multimedia streaming and interactive gaming applications. In *Proceedings of the 2006 international workshop on Network and operating systems support for digital audio and video*, page 15. ACM, 2006.

[15] Bryce Mariano and Simon GM Koo. Is cloud gaming the future of the gaming industry? In *2015 Seventh International Conference on Ubiquitous and Future Networks (ICUFN)*, pages 969–972. IEEE, 2015.

[16] Peter Eisert and Philipp Fechteler. Low delay streaming of computer graphics. In *15th IEEE International Conference on Image Processing, 2008*, pages 2704–2707. IEEE, 2008.

[17] Audrius Jurgelionis, Philipp Fechteler, Peter Eisert, Francesco Bellotti, Haggai David, Jukka-Pekka Laulajainen, Richard Carmichael, Vassilis Poulopoulos, Arto Laikari, P Perälä, et al. Platform for distributed 3d gaming. *International Journal of Computer Games Technology*, 2009:1, 2009.

[18] Itay Nave, Haggai David, Alex Shani, Y Tzruya, A Laikari, P Eisert, and P Fechteler. Games@ large graphics streaming architecture. In *IEEE International Symposium onConsumer Electronics (ISCE 2008)*, pages 1–4. IEEE, 2008.

[19] A Jurgelionis, F Bellotti, A De Gloria, P Eisert, JP Laulajainen, and A Shani. Distributed video game streaming system for pervasive gaming. *STreaming Day 2009*, September 2009.

[20] Shu Shi, Cheng-Hsin Hsu, Klara Nahrstedt, and Roy Campbell. Using graphics rendering contexts to enhance the real-time video coding for mobile cloud gaming. In *Proceedings of the 19th ACM international conference on Multimedia*, pages 103–112. ACM, 2011.

[21] Daniel Pohl, Stefan Nickels, Ram Nalla, and Oliver Grau. High quality, low latency in-home streaming of multimedia applications for mobile devices. In *2014 Federated Conference on Computer Science and Information Systems (FedCSIS)*, pages 687–694. IEEE, 2014.

[22] Alain Hore and Djemel Ziou. Image quality metrics: Psnr vs. ssim. In *ICPR*, volume 34, pages 2366–2369, 2010.

[23] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

[24] Xiaoming Nan, Xun Guo, Yan Lu, Yifeng He, Ling Guan, Shipeng Li, and Baining Guo. A novel cloud gaming framework using joint video and graphics

streaming. In *2014 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6. IEEE, 2014.

[25] Fernando Kuipers, Robert Kooij, Danny De Vleeschauwer, and Kjell Brunnström. Techniques for measuring quality of experience. In *Wired/wireless internet communications*, pages 216–227. Springer, 2010.

[26] Qingdong Hou, Chu Qiu, Kaihui Mu, Quan Qi, and Yongquan Lu. A cloud gaming system based on nvidia grid gpu. In *2014 13th International Symposium on Distributed Computing and Applications to Business, Engineering and Science (DCABES)*, pages 73–77. IEEE, 2014.

[27] Chek Tien Tan, Tuck Wah Leong, Songjia Shen, Christopher Dubravs, and Chen Si. Exploring gameplay experiences on the oculus rift. In *Proceedings of the 2015 Annual Symposium on Computer-Human Interaction in Play*, pages 253–263. ACM, 2015.

[28] David Kanter. Graphics processing requirements for enabling immersive vr. *AMD White Paper*, 2015.

[29] Ruiguang Zhong, Manni Wang, Zijian Chen, Luyang Liu, Yunxin Liu, Jiansong Zhang, Lintao Zhang, and Thomas Moscibroda. On building a programmable wireless high-quality virtual reality system using commodity hardware. In *Proceedings of the 8th Asia-Pacific Workshop on Systems*, page 7. ACM, 2017.

[30] Matthew Johnson, Irene Humer, Brian Zimmerman, Joshua Shallow, Liudmila Tahai, and Krzysztof Pietroszek. Low-cost latency compensation in motion tracking for smartphone-based head mounted display. In *Proceedings of the International Working Conference on Advanced Visual Interfaces*, pages 316–317. ACM, 2016.

[31] Agata Marta Soccini. Gaze estimation based on head movements in virtual reality applications using deep learning. In *Virtual Reality (VR), 2017 IEEE*, pages 413–414. IEEE, 2017.

[32] Brian Guenter, Mark Finch, Steven Drucker, Desney Tan, and John Snyder. Foveated 3d graphics. *ACM Transactions on Graphics (TOG)*, 31(6):164, 2012.

[33] Daniel Pohl, Xucong Zhang, and Andreas Bulling. Combining eye tracking with optimizations for lens astigmatism in modern wide-angle hmds. In *Virtual Reality (VR), 2016 IEEE*, pages 269–270. IEEE, 2016.

[34] Daniel Pohl, Xucong Zhang, Andreas Bulling, and Oliver Grau. Concept for using eye tracking in a head-mounted display to adapt rendering to the user's current visual field. In *Proceedings of the 22nd ACM Conference on Virtual Reality Software and Technology*, pages 323–324. ACM, 2016.

[35] Christian Timmerer, Daniel Weinberger, Martin Smole, Reinhard Grandl, Christopher Müller, and Stefan Lederer. Transcoding and streaming-as-a-service for improved video quality on the web. In *Proceedings of the 7th International Conference on Multimedia Systems*, page 37. ACM, 2016.

[36] Stephen R Ellis, Mark J Young, Bernard D Adelstein, and Sheryl M Ehrlich. Discrimination of changes of latency during voluntary hand movement of virtual objects. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, volume 43, pages 1182–1186. SAGE Publications Sage CA: Los Angeles, CA, 1999.

[37] Trinus vr web page. `https://www.trinusvirtualreality.com/#home`, 2017.

[38] Eduardo Cuervo and David Chu. Poster: mobile virtual reality for head-mounted displays with interactive streaming video and likelihood-based foveation. In *Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services Companion*, pages 130–130. ACM, 2016.

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

[40] Field of view. https://en.wikipedia.org/wiki/Field_of_view, 2017.

[41] Htc vive hardware specification. https://www.vive.com/us/product/vive-virtual-reality-system/, 2017.

[42] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[43] Space pirate trainer. http://www.spacepiratetrainer.com/#spthomepage, 2017.

[44] Tensorflow. https://www.tensorflow.org/, 2017.

[45] Julieta Martinez, Michael J Black, and Javier Romero. On human motion prediction using recurrent neural networks. *arXiv preprint arXiv:1705.02445*, 2017.

[46] Katerina Fragkiadaki, Sergey Levine, Panna Felsen, and Jitendra Malik. Recurrent network models for human dynamics. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4346–4354, 2015.

[47] Jun Liu, Amir Shahroudy, Dong Xu, and Gang Wang. Spatio-temporal lstm with trust gates for 3d human action recognition. In *European Conference on Computer Vision*, pages 816–833. Springer, 2016.

[48] Ashesh Jain, Amir R Zamir, Silvio Savarese, and Ashutosh Saxena. Structural-rnn: Deep learning on spatio-temporal graphs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5308–5317, 2016.

[49] Kar-Long Chan, Kohei Ichikawa, Yasuhiro Watashiba, Putchong Uthayopas, and Hajimu Iida. A hybrid-streaming method for cloud gaming: To improve the graphics quality delivered on highly accessible game contents. *Int. J. Serious Games*, 4(2), 2017.

[50] Sharon Choy, Bernard Wong, Gwendal Simon, and Catherine Rosenberg. The brewing storm in cloud gaming: A measurement study on cloud to end-user latency. In *Proceedings of the 11th annual workshop on network and systems support for games*, page 2. IEEE Press, 2012.

[51] Richard Süselbeck, Gregor Schiele, and Christian Becker. Peer-to-peer support for low-latency massively multiplayer online games in the cloud. In *Network and Systems Support for Games (NetGames), 2009 8th Annual Workshop on*, pages 1–2. IEEE, 2009.

[52] Hao Tian, Di Wu, Jian He, Yuedong Xu, and Min Chen. On achieving cost-effective adaptive cloud gaming in geo-distributed data centers. *IEEE Transactions on Circuits and Systems for Video Technology*, 25(12):2064–2077, 2015.

[53] Xin Yang, Kar-Long Chan, Papon Yongpisanpop, Hideaki Hata, Hajimu Iida, and Kenichi Matsumoto. Human software interaction in software development community. 2015.

[54] Elena Tuveri, Luca Macis, Fabio Sorrentino, Lucio Davide Spano, and Riccardo Scateni. Fitmersive games: Fitness gamification through immersive vr. In *Proceedings of the International Working Conference on Advanced Visual Interfaces*, pages 212–215. ACM, 2016.

[55] Sebastian Möller, Dennis Pommer, Justus Beyer, and Jannis Rake-Revelant. Factors influencing gaming qoe: Lessons learned from the evaluation of cloud gaming services. In *Proceedings of the 4th International Workshop on Perceptual Quality of Systems (PQS 2013)*, pages 1–5, 2013.

[56] M Shamim Hossain, Ghulam Muhammad, Biao Song, Mohammad Mehedi Hassan, Abdulhameed Alelaiwi, and Atif Alamri. Audio–visual emotion-aware cloud gaming framework. *IEEE Transactions on Circuits and Systems for Video Technology*, 25(12):2105–2118, 2015.

[57] Yeng-Ting Lee, Kuan-Ta Chen, Han-I Su, and Chin-Laung Lei. Are all games equally cloud-gaming-friendly?: an electromyographic approach. In *Proceedings of the 11th annual workshop on network and systems support for games*, page 3. IEEE Press, 2012.

[58] Hua-Jun Hong, De-Yu Chen, Chun-Ying Huang, Kuan-Ta Chen, and Cheng-Hsin Hsu. Qoe-aware virtual machine placement for cloud games. In *Network*

and Systems Support for Games (NetGames), 2013 12th Annual Workshop on, pages 1–2. IEEE, 2013.

[59] David Finkel, Mark Claypool, Sam Jaffe, Thinh Nguyen, and Brendan Stephen. Assignment of games to servers in the onlive cloud game system. In *Network and Systems Support for Games (NetGames), 2014 13th Annual Workshop on*, pages 1–3. IEEE, 2014.

[60] Zhou Zhao, Kai Hwang, and Jose Villeta. Game cloud design with virtualized cpu/gpu servers and initial performance results. In *Proceedings of the 3rd workshop on Scientific Cloud Computing*, pages 23–30. ACM, 2012.

[61] J Laulajainen, Tiia Sutinen, and Sari Jarvinen. Experiments with qos-aware gaming-on-demand service. In *Advanced Information Networking and Applications, 2006. AINA 2006. 20th International Conference on*, volume 1, pages 805–810. IEEE, 2006.

[62] Shaoxuan Wang and Sujit Dey. Rendering adaptation to address communication and computation constraints in cloud mobile gaming. In *Global Telecommunications Conference (GLOBECOM 2010), 2010 IEEE*, pages 1–6. IEEE, 2010.

Publication List

Published Journal:

[1] Kar-Long Chan, Kohei Ichikawa, Yasuhiro Watashiba, Putchong Uthayopas, and Hajimu Iida, "A Hybrid-Streaming Method for Cloud Gaming: To Improve the Graphics Quality delivered on Highly Accessible Game Contents," *International Journal of Serious Games*, vol. 4, no. 2, 2017.

Published Conference Papers with Review:

[2] Kar-Long Chan, Kohei Ichikawa, Yasuhiro Watashiba, Putchong Uthayopas, and Hajimu Iida, "A Hybrid Game Contents Streaming Method: Improving Graphic Quality Delivered on Cloud Gaming," *15th International Conference of Entertainment Computing 2016*, Vienna, Austria, Proceedings, pp. 149-160, September 28-30, 2016.

[3] Kar-Long Chan, Kohei Ichikawa, Yasuhiro Watashiba, and Hajimu Iida, "Cloud-Based VR Gaming: Our Vision on Improving the Accessibility of VR Gaming," *2017 International Symposium on Ubiquitous Virtual Reality (ISUVR)*, Nara, Japan, Proceedings, pp. 24-25, June 27-29, 2017.

Published Domestic Workshop paper:

[4] Kar-Long Chan, Kohei Ichikawa, Yasuhiro Watashiba, Putchong Uthayopas, and Hajimu Iida, "A Hybrid Game Contents Streaming Method to Improve Graphic Quality Delivered by Cloud Gaming," *IPSJ Special Interest Group on Entertainment Computing (SIGEC)*, SIG Technical Report, vol. 2014-EC-34, no. 1, pp. 1-6, December 12, 2014.

Others:

[5] Kar-Long Chan, Kohei Ichikawa, ""A Hybrid Game Contents Streaming Method to Improve Graphic Quality Delivered by Cloud Gaming," *Pacific Rim Application and Grid Middleware Assembly (PRAGMA Workshop 28)*, Poster, April 25, 2015.

[6] Xin Yang, Kar-Long Chan, Papon Yongpisanpop, Hideaki Hata, Hajimu

Iida, and Ken-ichi Matsumoto, " Human Software Interaction in Software Development Community," *IPSJ Winter workshop 2015*, Okinawa, Japan, pp. 5-6, January, 2015.