On Improving QoE of Remote Rendered Graphics

Gazi Karam Illahi
On Improving QoE of Remote Rendered Graphics

Gazi Karam Illahi

A doctoral thesis completed for the degree of Doctor of Science (Technology) to be defended, with the permission of the Aalto University School of Science, at a public examination held at the lecture hall T2 of the school on 7th of June 2024 at 12:00.

Aalto University
School of Science
Department of Computer Science
Supervising professor
Professor Antti Ylä-Jääski, Aalto University, Finland

Thesis advisor
DSc. (Tech.) Matti Siekkinen, Aalto University, Finland

Preliminary examiners
Professor Cheng-Hsin Hsu, National Tsing Hua University, Taiwan
Professor Patrick Le Callet, Polytech’Nantes Université de Nantes, France

Opponent
Professor Cheng-Hsin Hsu, National Tsing Hua University, Taiwan

Aalto University publication series
DOCTORAL THESES 122/2024

© 2024 Gazi Karam Illahi

ISBN 978-952-64-1866-7 (printed)
ISSN 1799-4934 (printed)
ISSN 1799-4942 (pdf)

Unigrafia Oy
Helsinki 2024

Finland
A new class of interactive multimedia experiences leverages real-time remote rendering with video encoding to provide high quality visual experiences on low end devices, the so called thin-clients. The basic architecture entails off-loading some or all the rendering calculations of a complex computer graphics scene to a remote server, often a cloud graphics server, which renders the scene, encodes it and sends it to a client as video. The video is then decoded by the thin-client and displayed to a user. Cloud gaming and Cloud Virtual Reality (VR) are two example use cases of such experiences. These applications have two principal constraints: downstream bandwidth and motion to photon (M2P) latency. Quality of experience (QoE) of such applications can be improved by reducing the downstream bandwidth needed for a given visual quality of the encoded video and by reducing the perceived M2P latency; that is the perceived latency between user action and corresponding frame update at the client. In this thesis, we investigate avenues to improve QoE of remotely rendered graphics applications by addressing the above constraints. We evaluate the feasibility of leveraging the characteristics of the Human Visual System (HVS) to reduce the downstream bandwidth needed for streaming high quality graphics videos. Specifically, we investigate the phenomenon of foveation in the context of real time video encoding and evaluate different parameterizations and schemes of foveated video encoding (FVE). We also investigate whether synergies exist between FVE and foveated rendering (FR). To address the challenge of low latency requirements for interactive remotely rendered graphics applications, we investigate Machine Learning (ML) based approaches to predict human motion kinematics used to render a scene by a rendering engine. Specifically, we investigate head pose and gaze prediction using past pose and gaze data. Accurate head pose and gaze information are critical for field of view (FoV) rendering and foveated encoding or rendering respectively. The investigated approaches focus on light weight data ingest and low latency inference in order to preclude introduction of additional latency in the rendering and media delivery pipeline.
This dissertation and the work presented in it are a result of support provided by colleagues, fellow researchers, family, and friends. I would like to thank my supervising professor, Professor Antti Ylä-Jääski, for his constant support and confidence in me throughout the PhD project. I am grateful to my thesis advisor, Matti Siekkinen, for giving me a foothold in the research world and helping me develop into a good researcher, in addition to collaborating on every research problem considered in this dissertation. Sincere thanks go to Teemu Kämäräinen for always being there to help and troubleshoot issues with our prototypes and experiments. I also wish to express deep gratitude to Professor Mario Di Francesco for being a mentor and helping substantively and not just with words.

Thanks also go to all my co-authors for making the articles in this thesis possible. Thanks also to all the colleagues at the Department of Computer Science who made working in the Computer Science building such a joy. Special thanks to Esa, Veronica, Ashutosh, Mohammad, Pranvera, Gopika, Abbas, Behrouz, Asad, Avleen, Manik, Samuel, Niina, and Sanna for all the discussions, chats, and check-ins. I am also thankful to the CS IT support for being the best IT support one could ask for.

I would like to express gratitude to the pre-examiners, Professor Cheng-Hsin Hsu and Professor Patrick Le Callet, for their insightful comments and feedback which improved the quality of this dissertation. Sincere thanks to Professor Cheng-Hsin Hsu for also agreeing to be the opponent.

About a year ago, after the substantive part of my PhD was finished, I was lucky enough to join an excellent research team at Nokia Technologies as a researcher. It has been a journey of constant discovery in multiple dimensions. I have a new appreciation for the work I did in my PhD while simultaneously recognizing the cutting-edge research done at Nokia Technologies. I would like to express my thanks to my colleagues at Nokia Technologies, especially Emre, Igor, Saba, and Serhan, who inspire me every day. Saba, an Aalto alumna herself, has been a great mentor, and I am grateful for her guidance.

I am extremely fortunate to be blessed with a loving family. No words can describe my gratitude to my dearly departed Dad, whom I will always miss.
In his roles as my father, my friend, and my patron, he taught me to be the person I am today. Without his infectious penchant for perseverance, I would not have pursued a career in research. I am grateful to my Mom, my sisters, my brothers, and my extended family for always being there for me, in joy and pain. Their love and support have been an unwavering safety net in my life. Sincere thanks to all the people and loved ones who touched my life during this PhD and supported me in their own different ways. Last but not least, thanks to my friends for tolerating my idiosyncrasies during these years, especially Tahir, Naveed, Salman, and Hashim.

Espoo, May 17, 2024,

Gazi Karam Illahi
## Contents

Preface 1  
Contents 3  
List of Publications 5  
Author’s Contribution 7  
List of Figures 9  
List of Tables 11  
Abbreviations 13  

### 1. Introduction 15  
1.1 Research Questions and Scope 18  
1.2 Methodology 20  
1.3 Summary of Contributions 21  
1.4 Structure 22  

### 2. Background 23  
2.1 Interactive Remotely Rendered Applications 23  
2.1.1 Cloud Gaming 24  
2.1.2 Cloud XR 25  
2.2 Human Visual System and Foveation 26  
2.3 Video Encoding 28  
2.4 Computer Graphics Rendering 29  
2.5 Time Series Forecasting 30  
2.6 Summary 30  

### 3. Reducing Bandwidth with Foveation 31  
3.1 Related Work 33  
3.2 System 36  
3.3 Foveated Video Encoding and Cloud Gaming 38
This thesis consists of an overview and of the following publications which are referred to in the text by their Roman numerals.


Author’s Contribution

Publication I: “Cloud Gaming with Foveated Video Encoding ”
Illahi designed and implemented the real-time foveated video encoding prototype, designed and implemented the foveation profiles and performed computational measurements. Van Gemert designed and conducted the user study and performed analysis of the user study results. Illahi and Van Gemert were the primary contributors in writing of the article. Siekkinen identified the problem space, contributed to system and experiment design, analysis of the results and to writing the document. Masala contributed to identifying entry points in the encoding pipeline for spatially varying quality to actuate foveation. Oulasvirta and Ylä-Jääski contributed the work at various stages of ideation and to writing and editing the article.

Publication II: “Foveated streaming of real-time graphics”
Illahi designed and implemented the foveated rendering and encoding prototypes, the foveation profiles, and the computational and subjective quality measurement experiments. Illahi was also the primary writer of the corresponding text the article. Kämäräinen designed, implemented and conducted the latency measurement experiments and was the primary writer of the corresponding text in the article. Siekkinen contributed to identifying the problem and solution spaces, analysis of the results and to writing the article. The prototypes developed in the work were based on prior remote rendering prototypes developed by Siekkinen, Kämäräinen and Ylä-Jääski. All co-authors contributed to the work at various stages of ideation, problem space identification and to writing and editing the article.
Publication III: “Learning to Predict Head Pose in Remotely Rendered Virtual Reality”

Illahi designed and implemented the LSTM based pose forecasting models presented in the article, the data pre-processing pipeline and conducted the accuracy evaluation of the developed models. Illahi was also the primary writer of the corresponding text in the article. Vaishnav implemented and evaluated, for accuracy, the baseline DESP based conventional pose predictor, its convolutional neural network based analogue and primarily wrote the corresponding text in the article. Kämäräinen designed, implemented and conducted the latency measurement experiments and primarily wrote the corresponding text in the article. Siekkinen and Francesco contributed to problem space identification, writing and editing the article. All authors contributed to ideation at various stages of the work.

Publication IV: “Real-time gaze prediction in virtual reality”

Illahi designed and implemented the LSTM based gaze forecasting model, the baseline models, the data pre-processing pipeline and conducted the evaluation of the developed model. Illahi was also the primary writer of the corresponding text in the article. Kämäräinen designed, implemented and conducted the latency measurement experiments and primarily wrote the corresponding text in the article. Siekkinen and Ylä-Jääski contributed to problem space identification and to ideation at various stages of the work. All authors participated in writing and editing the article.
List of Figures

2.1 Cloud Rendering Architecture ........................................ 24
2.2 eXtended Reality Spectrum, modified from [1] ..................... 25
2.3 Photo-receptor density in the human eye ............................ 27

3.1 A prototype remotely rendered interactive multimedia system. Figure copied from Publication I ....................... 37
3.2 Foveation and QO calculation. FW is the width of the output frame in pixels. Figures from Publication I .............. 38
3.3 Measured video bitrates with different games and parameterization of foveated streaming. FW is the width of the display in pixels. The box comprises the inter-quartile range, the red line in the middle of the box is the median, and the diamond denotes the mean. Copied from Publication I ......................... 40
3.4 Gaze tracking data from 15 minute gameplay sessions. a,b,c and d are gaze heatmaps, e and f are the CDFs of gaze moments: time periods where gaze lingers within a circular region of FW/8 (e) and FW/4 (f), g CDF of rate of gaze shifting .......... 41
3.5 QO_{max} and video quality in terms of (a) Mean Opinion Score with 95% confidence interval and (b) PSNR and EWPSNR. (c) illustrates the relationship between QO_{max} and MOS ............. 42
3.6 Foveation Samples with zoomed-in regions. .......................... 45
3.7 a) Frame sizes of frames rendered with Foveated Rendering (FR) and Normal Rendering (NR) with spatially uniform QPs b) FSSIM-Y of Foveated Warping (FW) and Foveated Video Encoding (FVE) frames with different parameterizations ............... 47
3.8 Encoding (a) and decoding (b) delays for FVE and FW .......... 48

4.1 Data processing pipeline of LSTM-based models. ................. 63
4.2 Data fusion model for pose and gaze, indicated with the p and g superscripts (respectively); x_t is the input sample at time t. .... 64
List of Figures

4.3 a) position and b) orientation errors for different representations of orientation. XR=position+quaternion, 6D=position+6D, 3P=three point, XE=position+Euler Angles. 2L, 3L and 4L show the number (2, 3 and 4 respectively) of LSTM layers in the LSTM-based models. ................................. 68

4.4 a) position and b) orientation errors for data modalities in addition to pose (XR), gaze data (G) and controller pose (C). Dlinear and the LSTM model trained with only pose data are provided as reference. Pose is represented as position vector+quaternion orientation. 2L, 3L and 4L show the number (2, 3 and 4 respectively) of LSTM layers in the LSTM-based models. ......................... 69

4.5 Heatmap of sanitized OpenNEEDs gaze data. ................. 71

4.6 Gaze Prediction Model: \( t \) is the current time, \( X_t-X_{t-200} \) and \( Y_t-Y_{t-200} \) is the gaze \( X, Y \) coordinate sequence from current time to approximately 200ms into the past. \( VG_t-V_{t-200}, VH_t-VH_{t-200}, AG_t-AG_{t-200} \) and \( AH_t-AH_{t-200} \) respectively are the gaze velocity, head rotational velocity, gaze acceleration and head rotational acceleration sequences from now to approximately 200ms into the past. \( pH \) is the prediction horizon; taken from Publication IV ................................................................. 72

4.7 a) CDF of the angular error for TDS1, TDS1 shares subjects with the training dataset b) Angular error for unseen VR gaming test data, at a prediction window of 100ms c) CDF of the angular error for GazeBase data with different configurations of input data. Figures taken from Publication IV ....................... 75
List of Tables

1.1 Research questions, research outputs and potential use cases . . 20

4.1 Summary statistics of accuracy evaluations. The cases are repre-
sented as `model_training mode_pose representation+additional
data modality`, training mode can be JT = Joint Training or ST
= Separate Training, for cases where position and orientation
components are learnt and forecast jointly and separately respec-
tively. Pose representation can be: XR = position+quaternion,
6D = position+6D, 3P = three point, XE = position+Euler Angles
and the additional data modalities can be gaze data (G) and
controller pose (C). 2L, 3L and 4L show the number (2, 3 and 4
respectively) of LSTM layers in the LSTM-based models. . . . . . 66

4.2 Summary of parameter count and inference times for different
models and runtimes. . . . . . . . . . . . . . . . . . . . . . . . . . . . 70

4.3 Evaluation results on test datasets. The Mean and Median Er-
rors are in units of degrees . . . . . . . . . . . . . . . . . . . . . . 73

4.4 Ablation study at prediction horizon 100ms, G = gaze, GV =
velocity, GA = gaze acceleration, HV = HMD velocity and HA =
HMD acceleration . . . . . . . . . . . . . . . . . . . . . . . . . . 74

11
Abbreviations

**API** Application Programming Interface

**CNN** Convolutional Neural Network

**DESP** Double Exponential Smoothing based Prediction

**DL** Deep Learning

**DNN** Deep Neural Network

**FoV** Field of View

**FR** Foveated Rendering

**FW** Foveated Warping

**FVE** Foveated Video Encoding

**GPU** Graphical Processing Unit

**HMD** Head Mounted Display

**HVS** Human Visual System

**IMU** Inertial Measurement Unit

**LSTM** Long Short Term Memory

**M2P** Motion to Photon

**ML** Machine Learning

**MR** Mixed Reality

**QoE** Quality of Experience

**QoS** Quality of Service

**RNN** Recurrent Neural Network
Abbreviations

**rtt**  round trip time

**SDK**  Software Development Kit

**VR**  Virtual Reality

**XR**  eXtended Reality
1. Introduction

Interactive multimedia applications with computer generated graphics, like video games and extended reality are burgeoning in popularity with the video game market size set to grow to more than USD 285 billion by 2027 [2] while the extended reality market is predicted to grow to USD 280 billion by 2028 [3]. Video games and extended reality applications both rely on real time rendering of relatively complex graphics scenes.

Video games have been developed since the advent of computers and have driven and benefited from advances in computer graphics processing. These advances in graphics processing have led to development of powerful graphical processing units (GPUs) and graphics algorithms capable of producing strikingly high quality photorealistic imagery. Increasingly, video games are being played on resource constrained devices like smartphones and tablets [2].

Extended reality is an umbrella term for a broad spectrum of immersive applications which typically cover a substantial fraction of a user’s visual field of view (FOV). These range from virtual reality (VR) and mixed reality (MR) to augmented reality (AR). These applications are either delivered on tethered head mounted displays (HMDs), standalone devices, or smartphones. In the tethered case, the HMDs are connected to powerful computers, which do the necessary processing.

Rendering a graphics scene is computationally intensive for non trivial graphics and especially so for complex graphics required for high visual quality. For an immersive experience, both video games and virtual reality applications need not only real-time rendering of complex graphics scenes, but also computing operations for coherence of the scene like maintaining physics and experience logic. For AR and MR applications, there are additional computational needs for tracking the HMD and processing the visual environment, typically a video stream of one or more cameras. The high computational requirements of a typical high quality video game or XR experience are obvious from the minimum hardware requirements specified by the developers. As illustrative examples, Elden Ring and Total War: Warhammer III two popular video games of 2022 recommend a minimum memory of 8Gb, a multi-core 64 bit processor capable of about 32bit 400 GFLOPS and a graphics processing unit (GPU) capable of up to
Introduction

32 bit 6 TFLOPS. Half-Life: Alyx and Hitman 3 two popular VR games in 2022 recommend similar or better hardware for a good quality of experience.

The success of cloud computing has led to interest in the paradigm of using cloud based remote rendering to ameliorate these hardware requirements. The basic architecture of such remote rendering systems entails off-loading some or all the rendering operations from a local client to a remote graphics server located in a so-called graphics cloud [4, 5]. The local client forwards all the information necessary to render a frame to the server, the server renders a frame based on this information and sends the frame back to the client which displays the frame to the user. The information sent to the server may include user control inputs for video games, head and hand motions for VR and a video stream of the environment in case of AR/MR. This paradigm is envisioned to enable high quality interactive multimedia experiences on low end thin client devices.

Remote rendering of interactive multimedia applications compares favorably with local rendering only if the visual quality of the video displayed at the thin client is equivalent to local rendering and the motion to photon (M2P) latency is not noticeable. Motion to photon latency refers to the latency between user control input, for example, a key press or a head movement to the corresponding frame update on the screen. The need for high quality video at the thin client translates to a high downstream bandwidth capacity requirement while the need for imperceptible M2P latency translates to a low system and network latency. These requirements have been extensively studied [6, 7, 8, 9, 10]. These works generally focus on M2P latency in remotely rendered systems, its causes and its effects on user quality of experience (QoE). While the effects of video quality on user QoE have also been studied [9], reducing bandwidth requirements specifically for interactive remotely rendered applications has seen less interest outside of general video coding research efforts for screen content.

A high downstream bandwidth requirement can not be always satisfied by a best effort network. High bitrate of a video stream can suffer from packet loss and jitter, which both have an impact on perceived quality in real time streaming scenarios [11, 12]. Packet loss, can lead to loss of or artefacts in one or more video frames in a bitstream, while jitter can force the decoding application to either introduce delay in presenting the decoded video frames or to discard frames entirely. In real-time streaming use cases, where packet re-transmission is not feasible, both of these effects are undesirable.

A high M2P latency can cause sensory mismatch between the human visual system and the human vestibular system. The human vestibular system is responsible for our sense of balance and spatial orientation [13] and is activated by head motions. A mismatch between the orientation and balance presented by visual stimuli and the orientation and balance perceived by the vestibular system can lead to discomfort and in severe cases nausea and other symptoms. When caused by VR, these symptoms are called VR sickness and have substantial negative impacts on user quality of experience [14].
This thesis investigates these twin challenges of downstream bandwidth and M2P latency facing interactive remote rendering application paradigm. Addressing these is essential for broad adoption of cloud based remote rendering as a viable and useful alternative to local rendering.

In this thesis we explore the foveated nature of the human visual system (HVS) as a means to reduce downstream bandwidth requirements for remotely rendered graphics. Foveation refers to the non-uniform visual acuity of the human eye, being highest directly along the optical axis of the eye lens and decreasing sharply away from the optical axis. Foveation, in the context of remote rendering, can be leveraged with foveated video encoding and foveated rendering. This thesis investigates the feasibility of real-time foveated encoding for cloud gaming for off the shelf games and with off the shelf hardware. To that end we implement and evaluate prototype systems. Prior work in the field has either considered custom made games or custom encoding pipelines or game attention models [15, 16]. In addition to foveated encoding, foveated content can be created at the rendering step as well, for example, using spatially varying sampling rates during the rendering process. Another possibility of creating foveated content is at the post-processing stage, wherein filters or geometric transforms may be applied to a rendered frame to produce a frame with a foveated spatial quality profile. This thesis also investigates and compares different approaches and corresponding parameterizations of creating foveated content in a remote rendering context. We propose a novel HVS mapped foveated warping method at the post processing stage which is suitable for remote rendering applications.

Further, in this thesis we explore approaches to mask latency in remotely rendered graphics, specifically by forecasting user related rendering parameters using machine learning. In interactive computer graphics applications, a frame is rendered based on the state of the application, virtual camera pose in the scene, camera projection, and user interactions. The user interactions can include interaction cues from input devices like key-press events, mouse events, mouse button presses, controller/joystick events etc. Gaze may also be used for rendering optimizations or interaction. While the state of the application and camera projection information is maintained and tracked by the application, the camera pose is typically controlled by user interaction as well. In case of XR rendering, the camera pose is controlled by the user head pose as reported by the HMD. Further, pose of some virtual objects in an XR scene may also be controlled by real-life location of trackable objects, such as hand-held controllers. Many of these rendering parameters, for example, HMD pose, controller pose, mouse/joystick movements and gaze location, depend on human body kinematics. Human body kinematic properties exhibit patterns [17], which can be leveraged to forecast the value of a kinematic property based on previous values in a sequence for remote rendering use cases. To this end, we design, train and test ML models to forecast user body kinematics, specifically head and gaze motion. The predicted motion can be used to render frames for user head pose and gaze coordinates which are yet to be reported by the client. We focus on forecasting
gaze for predictive foveation and forecasting the pose of an HMD for predictive rendering. These problems fit into the domain of time series forecasting which is a very well studied domain. While there has been a considerable amount of work in predicting pose or gaze for 360 degree video streaming, the methods developed therein are limited to three degrees of freedom (3DoF). Immersive VR entails the HMD having six degrees of freedom (6DoF), three translational and three rotational, which complicates the forecasting problem, primarily because rotation and consequently pose representations are not unique. Conventional pose representations have been shown to be sub-optimal for ML applications [18]. However, the effect of different pose representations in the context of pose forecasting for remotely rendered graphics has not been studied. Low latency requirements of remotely rendered graphics translate to short forecasting prediction horizons which may negate the shortcomings of conventional pose representations for ML. We explore conventional pose representations for pose forecasting and propose a novel pose representation, a novel data fusion approach as well as an ML pipeline for pose forecasting in the context of VR. Further, for gaze forecasting we develop a lightweight ML model which relies on only on a feature set based on past gaze data and optionally head rotation data to forecast future gaze. In doing so, we go in a direction orthogonal to most prior work in gaze prediction and consciously eschew use of visual stimuli or derivatives of visual stimuli in gaze prediction in order to avoid potential latency penalties due to intensive visual computations.

1.1 Research Questions and Scope

Improving the quality of experience of interactive remotely rendered applications necessarily entails investigating and ameliorating the two major challenges mentioned above: high downstream bandwidth and low M2P latency. As a means of reducing the high downstream bandwidth requirements, this thesis studies real time foveated content creation and encoding at the server side and content decoding and display at the client side.

RQ1: Is real time foveated encoding feasible for off the shelf games with consumer components and standard encoders?

To study feasibility of real time foveated encoding for remotely rendered applications, accurate real time gaze information is needed in addition to mechanisms for encoding a video stream with a non-uniform spatial quality profile to accrue bitrate savings. Further, for the foveated video to be visually comparable to spatially uniform quality video, a quality profile model which matches the HVS needs to be determined. We develop a prototype cloud gaming system capable of delivering a foveated video stream to a client fitted with an eye tracker. We propose an FVE encoding scheme and test different parameterizations of the the scheme to analyse whether bitrate savings can be achieved without loss of visual quality.
RQ2: **How can foveation at the rendering stage be used to improve real-time remotely rendered foveated content delivery?**

Foveation can be leveraged at the rendering stage by spatially varying the render quality according to the acuity profile of the HVS. We compare foveated rendering and foveated video encoding for remote rendering systems, in terms of bitrate, quality, resolution of the encoded frames and achievable resolution at the client. We also propose a novel scheme of foveated warping at a rendering server to improve the deliverable net resolution at the client.

The second challenge for interactive remotely rendered systems is reducing the perceptible latency. While it may not be possible to reduce latency beyond a certain lower limit dictated by geography and the physical layer of the network, the potential sensory mismatch caused by a high M2P latency can be masked by predictive rendering. Typically a remote rendering server renders a frame based on rendering parameters received from the client. These rendering parameters can be current camera position and orientation dictated by player point of view (POV) in video games and HMD pose in XR experiences, controller inputs, et cetera, and are generally abstractions of human body kinematic properties. Predictive rendering estimates these parameters for an estimated display time of the frame and then renders the frame based on the predicted rendering parameters. The estimated display time can be, for example, half of network round trip time (rtt) or half of estimated M2P latency.

RQ3: **In a remotely rendered VR system, is it possible to forecast user body kinematic properties which affect rendering given previous values of these kinematic properties?**

We consider head pose and visual gaze as representative user body kinematic properties which affect rendering. We design and evaluate ML based forecasting models to answer this question. Current head pose defines the FOV to be rendered for in VR/XR applications and current gaze is essential for HVS based rendering and encoding optimizations like those explored in RQ1 and RQ2.

Pose forecasting is complicated by the fact that pose has multiple possible mathematical representations, all of which are not necessarily optimal for ML [18]. We compare different pose representations and models for pose forecasting with a prediction horizon relevant for cloud VR to find suitable pose representations and ML models for pose forecasting. We propose a new pose representation for pose prediction in VR, in addition to presenting a pose prediction pipeline. In addition to we evaluate various time series forecasting methods for pose forecasting. Gaze forecasting, for low prediction horizons, is challenging because of various gaze biases that occur when humans view stimuli on a screen, especially in VR. Prior work in gaze prediction relies on information about visual stimuli or their visual saliency which can be computationally intensive. We propose a gaze processing pipeline and lightweight model for gaze forecasting, especially in VR, which relies on easily acquired numeric sensor data like previous pose and gaze.

The research questions, the main research outputs and application scenarios
are summarised in table 1.1

<table>
<thead>
<tr>
<th>Research Question</th>
<th>Main Research Output</th>
<th>Potential Use Cases</th>
</tr>
</thead>
<tbody>
<tr>
<td>RQ1</td>
<td>Foveated cloud gaming system, parameterization schemes</td>
<td>Cloud Gaming, Cloud XR</td>
</tr>
<tr>
<td>RQ2</td>
<td>Foveated rendering, warping and encoding system, parameterization schemes</td>
<td>Cloud XR, Cloud Gaming</td>
</tr>
<tr>
<td>RQ3</td>
<td>ML models, head pose and gaze data processing pipelines, a novel head pose representation, a novel data fusion layer</td>
<td>Cloud XR, Foveated Cloud Gaming</td>
</tr>
</tbody>
</table>

This thesis is bound in scope to interactive remotely rendered applications, and primarily aims to improve the user experience by reducing the downstream bandwidth needed to stream high quality graphics from a rendering server to a thin client and by masking the M2P latency by forecasting user kinematic properties which affect rendering. Improving video coding efficiency and reducing network or system latency in remote rendering has been extensively researched [10, 19, 20] and is not considered here in an explicit sense.

1.2 Methodology

The research questions enumerated above were answered by a systems approach of designing prototypes and models for the target use cases and empirical measurements and evaluations. All designs, implementations and empirical evaluations were guided by constraints set by practical real-time remote rendering applications.

To answer RQ1, we implemented a working cloud gaming system with foveated video encoding. We leveraged an open source cloud gaming system together with a consumer grade eye tracker to implement a prototype foveated video cloud gaming system. The implementation was designed to be usable with off the shelf games, without needing hooks into the game engine or re-implementation of an encoder decoder. This was achieved by using region of interest encoding in standard video encoders. The parameterization for the foveated encoding was inspired by the HVS. Similarly, to answer RQ2, we implemented a fully functional client-server system capable of foveated rendering and foveated video encoding. We built the server and client using the Unity game engine. The client application uses a consumer eye tracker (stand alone or in an HMD) for gaze tracking. Further, for usability with popular hardware and rendering pipelines, we leverage spatially variable rendering provided by Nvidia GPUs. We evaluated the parameterization of foveated video encoding and foveated rendering with
objective image quality assessments and limited subjective testing.

To answer RQ3, we developed Python Pytorch based scripts and ML models. The ML models and processing pipelines for human kinematic attribute forecasting were designed to rely on light weight data streams easily extractable from a VR rendering system without necessarily needing changes in, or hooks into the underlying VR/XR experience being executed. This and the design constraint of keeping latency low, dis-incentivizes use of image data or in-game virtual objects in our forecasting models, as extracting and processing image data can be computationally intensive and may incur a latency penalty. The approaches considered in state of the art work in the domain have typically relied on image data, image saliency data, scene object data, scene task data, or a combination of these. Further, the forecasting horizons considered in our work are in the range of realistic remote rendering scenarios. We evaluated the ML based prediction models using accuracy metrics relevant in the context of human body kinematic attributes. For gaze, we consider angular error between the predicted gaze and ground truth gaze. For evaluation of pose prediction models, we considered orientation and position errors separately.

1.3 Summary of Contributions

The contributions of this thesis, in line with the research questions and scope of this thesis, are in the domain of improving QoE of remote rendered real time interactive applications. The contributions are summarized below:

- Real time FVE for cloud gaming. This thesis establishes feasibility of real-time FVE with off the shelf encoders and suggests parameterizations for FVE which may accrue considerable bandwidth savings without impacting perceived quality.

- Real time Foveated Warp for cloud VR. This thesis proposes and tests a contextually novel real time post-processing frame warping scheme for spatially compressing technique which is guided by HVS and more specifically foveation. The proposed technique allows delivery of video whose display resolution is substantially higher than the resolution of the encoded version. This is achieved by allocating fewer pixels to perceptually less important regions of a frame, given a gaze point. The highest pixel resolution that is decodable by the so called “standalone” HMDs is often limited to an upper cap due to hardware limitations. FW may be used to deliver remotely rendered frames to these standalone HMDs such that the final display resolution is higher than the upper cap set by the hardware decoder.

- Comparison and analysis of real time foveated content generation for cloud VR. Three methods of creating foveated video in a remote rendering system
are implemented and analysed in this thesis. The three compared methods are FVE, FR and the proposed FW method. The findings of this analysis may help in informed design of remote rendered systems with foveation. A rule of thumb guideline that can be drawn from the analysis is that FR on its own does not provide bandwidth savings, while with proper parameterization FVE or FW may be used together with FR to accrue savings in computation and bandwidth. The choice between FVE and FW may be dictated by the use case.

• ML based real time gaze and pose predictors for VR. The thesis proposes ML based real time gaze and pose predictors for VR environments which use comparatively light weight and easy to extract data as inputs. The data pre-processing pipelines necessary for these models are also developed. These predictors do not rely on image based input data and can run in real time to predict pose and gaze for VR experiences for prediction horizons which are realistic for remotely rendered VR. These predictions may be used to compensate for motion to photon latency of remote rendered VR systems and thus improve QoE for the end user.

1.4 Structure

The rest of the thesis is structured into 4 chapters. Chapter 2 provides an overview of remotely rendered applications, HVS, video encoding, graphics rendering and time series forecasting. These areas provide a background of the work presented in this thesis. Chapters 3 and 4 provide a discussion on contributions of this thesis, in leveraging foveation for bitrate reduction and masking latency with forecasting respectively; contextualised with a discussion of the state of the art. Chapter 5 concludes the thesis, followed by the original publications included in this thesis.
2. Background

In this chapter, a background study of the concepts involved in this work is presented. We discuss interactive remotely rendered applications, especially the use cases of cloud gaming and cloud VR. We provide an overview of the phenomenon of foveation and its use in video and graphics. We also provide a brief background of time series forecasting which is relevant in the context of RQ3. Specifically, we discuss machine learning methods for time series forecasting.

2.1 Interactive Remotely Rendered Applications

Remote rendering, in a simple sense is the visual computing paradigm of decoupling the location of rendering operations for an application from the location where the rendered graphics are being interactively consumed, with a network link transporting the rendered graphics to the graphics consumption location. Remote rendering is at the heart of visual computing history, first starting off in the mainframe-terminal era as early as 1960s [21]. The design requirements and QoE expectations, obviously, were much different at the time. The shift towards personal computers happened synergistically with the popularity of graphical user interfaces. Major developments in semiconductors and computing hardware have made local rendering of applications with increasingly complex computer generated graphics possible and popular, both for productivity and leisure [22]. With the advent of cloud computing, the pendulum has swung back to executing applications remotely, at least partially, due to a variety of technical, business and usability factors [23]. Simultaneously, use of resource constrained devices like smartphones, tablets and netbooks has exploded. These two trends, together with increasing costs of hardware for high quality rendering, be it for video gaming or VR or computer aided (graphic) design, has renewed interest in the paradigm of remote rendering. Remote rendering is envisioned to provide high quality visual experiences on resource constrained devices and enable development of new business models [24].

Good QoE for remote rendering applications has two necessary constraints:
good stream quality, which translates to high downstream bandwidth and low latency [8]. This thesis focuses on these two constraints and hence remote rendering is the broad use case for **RQ1**, **RQ2** and **RQ3**.

The general architecture of a remote rendering application is illustrated in figure 2.1. In the simplest case, it comprises a thin client and a rendering server. A thin client executes a local application which intercepts control inputs and user actions including HMD and controller motion in case of XR use cases. The client application transmits these control inputs to the remote server. The remote graphics server runs an application engine which executes the experience and rendering logic. It receives the intercepted control inputs and replays them to the application engine, mimicking local input. The application engine renders a scene based on the control inputs, encodes it and transmits it to the thin client. Once received, the thin client decodes and displays this video, mimicking local rendering [4, 5]. While we characterized the rendering operations as producing a video stream containing the output of all rendering operations of a graphics application, remote rendering may also be partial in nature, in that not all rendering and experience logic may be offloaded to a remote server. In such an architecture, the thin client may dynamically offload execution of parts of rendering and experience logic to the server, while executing the non-offloaded parts locally. The remote rendering server may transmit rendering primitives and other synchronizing information instead of or in addition to fully rendered frames to the client [25, 5, 26].

### 2.1.1 Cloud Gaming

Cloud gaming is the most common leisure use case of the remote rendering paradigm discussed above. With reference to Figure 2.1, in cloud gaming, the remote server executes the rendering and game logic of a video game. In addition,
it replays the user control input, e.g. key presses, mouse, joystick, controller or touch screen input to the game engine as intercepted and transmitted by the client device. The clients for cloud gaming range from smart TVs, netbooks and desktop computers to tablet and smartphone devices. Although cloud gaming services have been available since 2000 [27, 24], recent entry by major information technology companies into the field has renewed consumer interest into cloud gaming services. These include services like Nvidia GeForce Now, Xbox Cloud Gaming, Amazon Luna and PlayStation Plus. By the time of writing this dissertation, active research into cloud gaming has also been ongoing for at least two dozen years, ranging from prototype systems [4], business cases [24] and QoE of cloud gaming and factors affecting it [9, 8]. M2P latency and stream quality have been identified as primary factors QoE of cloud gaming [9, 28].

### 2.1.2 Cloud XR

Extended Reality (XR) is an umbrella term to encompass a wide range of immersive experiences characterized by containing at least some virtual elements. Depending on the proportion of virtual elements to elements from the real world, XR experiences exist on a spectrum from Virtual Reality (VR) to Augmented Reality (AR) [1]. This is illustrated in Figure 2.2. The immersiveness entails at least some multi-sensory stimuli, including but not limited to, visual, auditory and haptic stimuli. AR is generally considered to comprise overlaying virtual objects over real objects in a real environment, allowing for interaction with the virtual objects. Mixed Reality (MR) takes the proportion of virtual objects and interaction further and is considered to blend real and virtual objects and allow for immersive interaction with both. In VR, the user is immersed in a wholly virtual environment with elements of interactivity. This thesis uses VR as a representative XR application for system design and experiments.

![Extended Reality Spectrum](image)

**Figure 2.2. eXtended Reality Spectrum, modified from [1]**

Typically VR is delivered using a near eye HMD. The HMD covers a user’s whole visual field of view and displays a computer generated virtual scene, accompanying audio is delivered using earphones. Other VR setups like cave VR are possible, but not very common as a consumer product.
may be delivered using hand held controllers or more elaborate haptic pads on the user's body. Immersiveness of the virtual scene is achieved by a one to one mapping of the user's head (specifically eyes) in the physical world to a (stereoscopic) virtual camera in the virtual scene being rendered, so that the rendered field of view corresponds to the user's movements. The mapping works using accurate real time sensors which track the user's head pose and optionally the pose of controllers that the user uses to interact with the virtual environment. Accurate and low latency tracking is essential for VR as any mismatch between the visual stimuli displayed in the HMD and the spatial orientation from the human vestibular system can result in severe degradation of user QoE and more problematically, cause the so called VR sickness [29, 30]. Pose tracking in state of the art commercial VR solutions is either inside out, outside in or a combination of the two. Inside out tracking uses sensors and cameras on the HMD for pose tracking. Outside in tracking uses stationary optical sensors outside the device and sensors on the HMD to track the pose. The stationary optical sensors generally track predefined markers on the HMD device within the tracking space.

Cloud XR applies the architecture of 2.1 to XR. The thin client being an HMD or a smartphone device, with means to track the head or device pose. Remote rendering for VR and XR exacerbates the tracking latency issue. In addition to the system and processing latency of the tracking system, remote rendering adds network latency and additional processing latency [10]. It is not possible to reduce all the additional latencies, however, masking the latency is a possible avenue of reducing the perceived M2P latency. RQ3 considered in this thesis attempts to provide ML methods to mask latency in cloud XR/VR situations by forecasting future head pose or gaze positions for which to render. Although this thesis considers Cloud VR use case of remotely rendered XR, the lessons learnt, especially in the context of RQ3, can be extended to other Cloud XR scenarios as well.

2.2 Human Visual System and Foveation

The human visual system consists of the eyes, ocular musculature and associated neural circuitry, including the visual cortex [31]. At a high level, the eye acts as the sensor which samples light from the visual field. The human eye is roughly a sphere with an opening, called pupil for light to enter. The pupil size and hence the amount of light entering the eye is controlled by iris muscles surrounding it and the opening itself contains a flexible lens whose thickness and hence power is controlled by ciliary muscles. The eye is filled with a transparent fluid called the vitreous humor. There is a transparent corneal layer over the lens and parts of the iris. Light entering the eye is focused by the cornea and the lens on to the retina. Retina refers to the inner surface of eye ball sphere. This surface is lined with neurons called photoreceptors which
Figure 2.3. Photo-receptor density in the human eye

are fired or activated by the light incident on them. These photoreceptors are connected to a variety of neurons, for example, collectors and retinal ganglion cells, which ultimately connect to the brain (visual cortex) via the optic nerve. There are two primary type of photoreceptors in our retina, rods and cones. Rods are responsible for vision under low illumination. This low light vision, called scotopic vision is monochromatic in nature. The cones are responsible for chromatic and photopic vision i.e. that is vision under well lit conditions[31]. The collector cells accumulate electrical signals from the photoreceptors when they are fired and transport them to the retinal ganglion cells. The ganglion cells have their own triggers of activation and act as a filter. The filtered signals are transported to the visual cortex via the neural fibers in the optic nerve. Both the density of photoreceptors and the density of neurons collecting signals from these photo-receptors is non-uniform. Further, the photoreceptors and the ganglion cells have functionally and morphologically different sub-types which have different activation regimes and consequently perform different tasks. The receptive field of ganglion cells, i.e., the number of photoreceptors they collect signals from, is also non-uniform. On the retina, fovea is the region located diametrically opposite to the lens. It is characterized by the highest concentration of cone photoreceptors. The density of cones in the retina drops dramatically with distance from the fovea. On the other hand, there are almost no rods in the fovea: their density is highest around the fovea and drops with eccentricity from the fovea. The ganglion cells corresponding to the fovea have a small receptive field, with a one ganglion receiving signals from one photoreceptor, while the ganglion cells corresponding to photoreceptors away from the fovea have a much larger receptive field, with many ganglions receiving signals from many photoreceptors.
from the fovea have a larger receptive field, with a one ganglion receiving signals from many photoreceptors[32]. This non-uniformity of the photoreceptors and the corresponding neural circuitry leads to a compression of the so-called raw retinal image as it is passed to the visual cortex. The ganglion cells also vary in their function in the vision process depending on their type and the neurons they correspond to in the visual cortex. The visual cortex neurons are also of different types, with different receptive fields and sensitivity to different aspects. Briefly, midget ganglion cells connect to neurons called P-cells. This neural pathway is sensitive to mid to high spatial frequencies and low temporal frequencies. Small bi stratified ganglion cells are responsible for carrying color information of visual stimuli in mid spatial and temporal frequency range. Another kind of ganglion cells, the parasol ganglion cells are responsible for carrying achromatic, i.e. intensity information for visual stimuli of with low spatial and mid temporal frequencies. In the primary visual cortex, simple cells are respond to contrast gradients of stationary or slow visual stimuli while so called complex cells respond to moving stimuli.

The human visual system is complex and not fully understood [31]. However, non-uniform distribution of photoreceptors in the retina and the fact that corresponding neural circuitry, including in the visual cortex as illustrated by cortical magnification[33], devotes more resources to the signals from the foveal region lead to a photopic visual acuity profile with the acuity being highest directly on the fovea-lens axis and dropping off exponentially with radial distance from the fovea-lens axis[34, 35, 32]. It is also manifest as the decrease in spatial frequency discernible by the human eye at a given viewing distance with radial distance from the fovea lens axis [31, 36]. This non-uniform acuity of the HVS is also referred to as foveation. The phenomenon of foveation can be leveraged to reduce resources needed for rendering or encoding graphics frames and is relevant to RQ1 and RQ2 considered in this thesis. However, it should be noted that foveation is just one aspect of the HVS. Other aspects, like contrast sensitivity both spatial and temporal, cortical magnification, saccadic omission, and aspects of scotopic vision play an important role in human vision. An excellent resource on these aspects can be found in [31]. Some of these aspects, when relevant are considered and introduced in this thesis.

2.3 Video Encoding

Video encoding uses signal processing and statistical techniques to reduce the size of image and video sequences for storage and transmission over networks. Current video encoding standards use the so-called block based encoding which leverages the fact that a picture or a video frame is an ordered sequence of pixels which can be grouped into blocks. Spatial and temporal similarities between these blocks are then used to represent one block as a function of one or more previously encoded blocks which allows it to be imperfectly reconstructed
from the previously encoded macroblocks. When this process is done over a whole frame, an imperfect frame can be reconstructed and only the functions used and the difference between this imperfectly reconstructed frame and the original frame needs to be stored to recreate the original frame. The macroblocks of the difference frame are decomposed using a frequency transform such as Discrete Fourier or Cosine transform. The size of the storage representation of the frame is reduced by quantizing the frequency transform coefficients. Further reduction in size can be achieved by using statistical properties of the quantized coefficients to encode them using fewer bits [20, 37]. Video Encoding is relevant to RQ1 and RQ2 considered in this thesis.

2.4 Computer Graphics Rendering

Computer graphics rendering, in the simplest terms, creates visual images computationally from programmed instructions. A graphics rendering engine is provided coordinates of so called geometric primitives, associated information and ordered instructions, it applies the instructions on the primitives and generates images comprising pixels which can be displayed on a display. For most three dimensional applications, a three dimensional scene comprising of hierarchically defined objects is defined by the application. The objects, their pose within the scene, their behaviour with time and in response to user interaction are also defined. The objects comprise of geometric primitives which together form geometric shapes. The geometric primitives can in turn be defined in terms of points in the 3D scene, so called vertices. Further, textures, color information and lighting to apply to these shapes are also provided. The actual rendering uses at least one virtual camera traversing the scene, e.g. in response to user controls, to capture a view frustum: a pyramidal space originating from the lens of the virtual camera. In order to generate the image captured by this virtual camera, a series of operations are undertaken on the vertices of the objects by a graphics rendering pipeline [38, 39, 40]. The most popular graphics pipeline can conceptually be divided into an application stage, a geometry stage and a rasterization stage, each of which may have pipeline like stages of its own. The application stage defines the 3D scene, the objects in it, animations, the interaction mechanisms and behaviour, et cetera. From the application stage the objects in the scene which are to be rendered are output in the form of rendering primitives like points, lines and triangles and associated information like textures and light sources. The geometry stage, broadly, processes each vertex of the rendering primitives, applying transforms (model view), vertex shading, projection, clipping to view volume and screen mapping. Vertex shading includes computing the effect of lighting on materials associated with the rendering primitives. The output i.e. processed vertices and associated data are fed to a rasterizer stage which converts this information into pixels drawn onto a screen or a render target. This includes triangle setup from the geometry
stage data and triangle traversal to find which triangles overlap with a pixel to generate fragments for the pixel. Finally, a merging operation computes the final pixel color from the fragments, associated data and fragment shading logic [38]. Graphics rendering is relevant particularly to RQ2 considered in this thesis.

2.5 Time Series Forecasting

Time series forecasting is the broad field concerned with forecasting values of a time varying quantity given values of the quantity at known times. Often, previous or historic values are available and present or future values which are not known yet are forecast. Time series forecasting has found major applications in areas as varied as stock market, consumer goods market to website traffic prediction [41]. Conventional methods of time series forecasting have relied on parametric, hand crafted methods to utilize properties of previous data to create models which can be used to predict future samples [42] such as autoregressive integrated moving average (ARIMA) models [43], exponential smoothing models [44, 45] and conventional state space models [46]. Owing to the success of machine learning models in other domains, there has been a spurt in using machine learning, especially deep learning methods for time series forecasting [41]. While, CNNs have been designed for time series forecasting as well [47], recurrent neural networks (RNNs) such as Long-Short Term Memory (LSTM) models [48] and models derived from them like Transformers [49] have been of particular interest to researchers for time series forecasting [50, 51, 52, 53]. This is motivated by the fact that time series forecasting fits very well in a recurrent computational graph and due to excellent results of recurrent models in the field of Natural Language Processing (NLP) [54]. NLP problems are viewed as sequence to sequence problems and time series forecasting problems can be easily re-projected as sequence to sequence problems. Time series forecasting is relevant particularly to RQ3 of this thesis.

2.6 Summary

In this chapter we briefly introduced the background concepts on which work conducted in this thesis relies. We introduced remote rendering especially cloud gaming and cloud XR which are the main use cases of the interactive remote rendering paradigm, the challenges of which this thesis attempts to improve from a quality of experience perspective. We also took a look at video encoding and computer graphics which are two primary technologies that interactive remote rendering applications rely on. Then we introduced human visual system and its non uniform acuity–courtesy of foveation. Finally, we also briefly introduced time-series forecasting. Foveation and time series forecasting are two domains we leverage to attempt to answer the research questions of this thesis.
3. Reducing Bandwidth with Foveation

One of the two primary challenges of real-time remotely rendered graphics is the need for high downstream bandwidth from the rendering server to the thin client. High fidelity video with what might be considered a sufficient minimum pixel resolution and fidelity at the time of writing this thesis, is invariably large in size with currently available and deployed video codecs, especially with low latency encoding. This is further exacerbated if the frame rate of the video is high. For interactive applications, a high frame rate is necessary for a good quality of experience [55, 56]. The downstream bandwidth required is especially challenging for remote VR applications as the minimum pixel resolution and frame rate needed for immersion is quite high compared to non-VR remotely rendered applications [5, 57].

Compressing video for efficient storage or transmission is a vast field of research, that has been ongoing for more than half a century [58]. Video streaming over the internet, on-demand or live, has exploded in popularity, constituting the bulk of internet traffic in 2022 [59]. Concurrent to the increase in video applications, there has been an increase in display and video resolution, with 1920x1080 pixels no longer being the highest possible available resolution for display screens or on demand videos. This has spurred intense work to create new video compression standards rapidly. As of 2021, Video compression standards from the Joint Video Team of ITU-T Video Coding Experts Group and ISO Moving Picture Experts Group (MPEG) are the most adopted video codecs, AVC being the most used video codec [60]. Following the release of the first version of AVC in 2003, JVT released multiple extensions for AVC. Further, JVT standardized two new generations of video codecs, HEVC and VVC, which are being gradually adopted [60]. VVC is supposed to provide 50% more compression efficiency than HEVC which itself is estimated to provide 50% more compression efficiency over AVC. Parallel to the developments by JVT, Google and AOMedia have released VP8, VP9 and AV1 as open source competitors to AVC, HEVC and VVC respectively. A primary factor affecting the adoption and popularity of a video codec is whether hardware encoding and decoding support is available for it in consumer devices, as manifest in the popularity of AVC.

Foveation of the human visual system can be exploited to address the band-
Reducing Bandwidth with Foveation

width challenge, by achieving compression gains on top of the compression attained by the underlying encoder. Referring to section 2.2, the acuity of the human visual system is non uniform, being very high directly in front of the lens. This region of high visual acuity covers just 2° of the human visual field which is approximately equal to a human thumb's width at arms length from the eye.

Considering foveation, encoding a video frame with uniform spatial quality may be a non-optimal use of the bit budget assigned to a frame. As the name suggests, foveated video encoding takes foveation into account while encoding a video frame [61]. Given a gaze point (tracked or predicted) a video frame is encoded with a quality profile that roughly matches the acuity profile of the HVS: high quality at the gaze point and lower quality elsewhere. This translates to using more bits at and around the gaze point(s) in a video frame and lesser bits elsewhere in the frame. When used for remotely rendered graphics, FVE may help in ameliorating high downstream bandwidth requirements of interactive applications.

In 3D rendering applications, foveation can also be leveraged to improve the perceived quality or to reduce the rendering cost of rendered frames. Most modern GPUs provide means to spatially vary the quality of a rendered frame, for example, by varying the shading rate or level of detail. This can be leveraged to implement so called Foveated Rendering (FR) wherein the quality of the rendered frame matches that of the HVS with respect to a gaze point provided by an application. In fact, Application Programming Interface (API) of a popular GPU manufacturer has options to implement FR in any 3D application with the condition that the API is provided a gaze point. FR, in addition to having the potential advantage of reducing the computational load at the rendering server [62], can produce foveated frames for a foveated video stream. Frames rendered with a uniform spatial quality can also be transformed to have a foveally varying spatial quality or resolution after being rendered, in a so-called post processing stage before encoding [63]. However, which foveation method leads to bandwidth savings or is advantageous for remotely rendered graphics needs to be investigated. Further, the parameterization for a combined FR and FVE strategy is also an open area of investigation. RQ2 of this thesis attempts to answer this problem.

This chapter presents our investigations into RQ1 and RQ2. RQ1 is investigated with a real-time FVE for cloud gaming implementation and extensive measurements of bandwidth and image quality as published in Publication I. RQ2 is investigated with an implementation of a FR and FVE prototype remote rendering system and comparing three different methods of delivering foveated frames to a client and evaluation of the methods with objective measurements.
3.1 Related Work

Foveated video encoding relies on gaze information to be available. Since each viewer can have a different gaze pattern, FVE with an ideal match between encoding quality profile and the acuity of the HVS would rely on real-time gaze information of the viewer. For video on demand applications it is not feasible to foveally re-encode a video every time it is requested, hence saliency or regions of interest have been suggested as a proxy for gaze location for the encoding process [64]. Most early work in foveated video encoding either assumed a temporally fixed gaze point at the center of the frame [65] or relied on saliency or regions of interest [64, 66] or manual selection of gaze points [67, 68]. A survey on algorithms for foveated video coding can be found in [69], most of the proposed methods entail either implementing non-standard codecs or making substantial changes to a standard codec.

Non-invasive gaze tracking has become more feasible in recent years with improvements in tracking algorithms as well as cheaper hardware for eye tracking. Consequently, relatively accurate consumer gaze trackers have become affordable, with many recent VR HMDs having built gaze trackers, for example those evaluated in [70]. Availability of gaze data makes foveated video streaming enticing especially for point to point video transmission and has lead to a spurt of research in the field. Ryoo et al. present a real-time foveated video streaming system based on commodity components in [71]. The system comprises a client with a webcam which is used to estimate gaze. The gaze and a bandwidth estimate is sent to a video streaming server. To actuate spatial non-uniformity in quality, each frame is divided into a grid of 16x9 cells and each cell is pre-coded in multiple versions of increasing quality. During streaming, high quality cells are streamed for the gaze estimate received from the client, the cells surrounding the high quality region have an intermediate quality and the rest of the cells have low quality. This method of foveal streaming is shown by the authors to reduce downstream bandwidth by up to 50% compared to uniform quality streaming without loss of user satisfaction. Hoessini et al. propose a similar solution for streaming 360° videos to a VR HMD, dividing the video into tiles and streaming only the high quality tiles to the viewers’ field of view. The practice of dividing a frame into rectangular cells or so called tiles to deliver frames with spatially varying quality has become popular with tile based encoding support provided in HEVC [72, 73]. This feature of HEVC has been particularly leveraged for streaming panoramic videos, for example, in both [74] and [75] HEVC compliant tiles are used to deliver panoramic video. In [74], the high quality tiles are streamed for a region of interest based on simulated movement while in [75] the high quality tiles are streamed for a viewer’s field of view in a VR HMD. A real time foveated encoding solution for panoramic video streaming which using motion constrained HEVC tiles is proposed in [76]. While using HEVC compatible tiles for region of interest or foveated encoding has multiple advantages, such as growing support for hardware acceleration...
Reducing Bandwidth with Foveation

for encoding and decoding and need for a single decoder at the client, spatially dividing a video into motion constrained independently decodable tiles reduces compression efficiency and can be computationally expensive. The problems are further exacerbated when considering large saccade motions wherein the gaze lands at a tile previously streamed at low quality which complicates inter frame prediction. In [76], this challenge is addressed by storing a video stream in two coded versions, one which is coded with intra (I) frame only and a second one which is coded with both I frames and predicted (P) frames, so that intra coded tiles can be dynamically inserted when needed.

None of the works discussed so far deal with streaming real time rendered graphics. Gaze contingent rendering and display has been a topic of research for some time now, especially in the context of psychophysics [77]. There is considerable work on FR to optimize usage of rendering resources. A relevant modern influential work in the field of FR is by Guenter et al. [78]. The authors propose a gaze contingent rendering scheme which renders three circular regions of progressively lower quality centered around the gaze point. The region at the gaze is rendered with the highest resolution, while the annular region surrounding the gaze region is rendered with an intermediate resolution and the rest of the frame is rendered with the lowest resolution. The three regions are (up)scaled to the same resolution, blended and an anisotropic filter is applied to create the final frame. The authors validate their FR schemes with a user study and report rendering acceleration by a factor of 5-6 for the same perceived quality. Subsequent to the work in [78] there have been many works in the field of FR, introducing different algorithms for FR or evaluating the parameterization for foveation [62, 79, 80, 81, 82, 83, 84], but the principle remains the same.

Some work has also been conducted to investigate latency constraints for FR in VR [85] which suggest a tolerable latency of 50–70ms. A survey on FR can be found in [77].

Substantial work has been conducted to study the parameterization of foveated imaging and display, be it in vision science, computer graphics or video coding. We briefly discuss works relevant to us [86, 87, 88, 82, 89]. Wiedmann et. al [86] study foveated video encoding for real-time streaming of natural videos, using a foveated encoding approach similar to that proposed in Publication I. The authors parameterize foveated encoding using subjective evaluations of Just Noticeable Distortion (JND). The results indicate up-to 68.88% bitrate savings with distortions noticeable by 25% of the study participants. The parameterization for the study in [86] was conducted after our work in Publication I and validates parameterization presented therein. Bokani [87], empirically evaluates foveated videos in a subjective study, wherein each foveated frame is compressed with a spatial quality profile consisting of 6 different quality regions: an octagonal region centered at the user gaze and 4 other concentric octagonal rings with progressively lower quality and the rest of the frame with the lowest quality. The quality difference between the different regions is not specified in [87], but is ostensibly decided based on the highest quality level set for the foveal region.
Bitrate savings of up to 33.2% are reported in [87] for foveated videos with the same quality in the foveal region as the source video. Lungaro et al. in [88] study QoE trade-offs for foveated content provisioning with subjective testing, with a focus on latency, with different radii of high quality "foveal" region. The authors conclude that even for latency levels of 4G networks, several foveated encoding parameterizations may be used to deliver foveated video with acceptable QoE. Swafford et. al in [82] evaluate perceptual quality of different FR methods both subjectively and computationally. The study results show performance gains without substantial degradation in perceived quality are possible with FR, provides guidelines for real-time FR parameterization and proposes a perceptually guided metric for FR evaluation. A comprehensive study of quality assessment for foveated video compression for virtual reality is conducted by Jin et. al. in [89]. In addition to conducting a thorough subjective evaluation of FVE with different parameterizations, authors in [89] provide test sequence databases and evaluate image quality assessment algorithms on the foveated content. Rai et. al in [90] study the role of distortions in the visual periphery in the context of natural fixation and observe that temporal distortions in the peripheral vision have more effect on perceptual experience than non-flickering spatial distortions.

Streaming real time rendered graphics with spatially non-uniform encoding is considered in [15, 16, 91, 92, 93, 94], specifically from a cloud gaming perspective. Ahmadi et al. [15] develop a game attention model based on both saliency maps and top down game object and task based priority maps to allocate encoding bits in a spatially varying fashion. Mohammadi et al. [16] propose a gaze contingent graphics streaming solution for cloud gaming, wherein graphics objects are streamed with higher quality if they are being gazed at by the player and lower quality otherwise, by varying QPs at the encoding stage. The proposed solution relies on the game engine providing object information to the encoder and uses MPEG-BIFS to stream the objects. Liu et al. [91] propose a spatially variant bit allocation scheme for cloud gaming using scene and depth information from the game engine to obtain a macroblock level attention map. Subsequently, QPs for each macroblock are calculated corresponding to the attention map. Hegazy et al. [92] propose a content aware video encoding (CAVE) scheme for cloud gaming wherein the quality of a video frame is varied based on importance of the objects in the frame as reported by the game engine. Le et al. [94] propose an end to end neural codec for cloud gaming which relies on rendering information from the game engine to leverage depth information, camera and in-game object motion to produce encoded streams which beat HEVC encoded streams rate-distortion metrics.

The works discussed above need access to the game engine or modifications to standard codecs or both, which makes them impractical to use with off the shelf games. Mossad et al. in [93] propose a deep learning frame work to extract regions of interest from rendered frames in real time. These regions of interest are then used in a standard encoder to differentially allocate bits in a
Reducing Bandwidth with Foveation

This approach has the potential to reduce downstream bandwidth requirements of cloud gaming considerably.

The approach proposed in this thesis is based on real-time gaze reported from the client. Consumer gaze trackers are becoming more affordable and are in fact inbuilt in many recently launched VR HMDs [70]. Further, with good tracking algorithms, web cameras which are so ubiquitous, can also be used to provide a reasonable gaze estimate [71, 95, 96, 97, 98].

3.2 System

The answer to RQ1 and RQ2, lies in evaluating FVE and FR for remotely rendered interactive graphics. Towards that goal, we implement two prototypes to study FVE and FR, respectively presented in Publication I and Publication II. On a high level, they follow the general architecture presented in 2.1, with a client and server architecture. However, the systems implemented in this thesis have a communication channel between the server and client which reports gaze to the server as tracked by a gaze tracker at the client. The high level system is presented in Figure 3.1.

The two implemented systems share the same general design. However, to answer RQ1, we targeted compatibility with off the shelf games to be able establish feasibility of FVE more substantively. Usability with off the shelf games which are based on disparate graphics engines and platforms is a desirable feature for a cloud gaming system. A cloud gaming system is usable with off the shelf games if it does not need code hooks into the game executable and if it can encode and process frames as they are presented for display, rather than needing modifications into the render engine as well. We used GamingAnywhere [4] as the cloud gaming chassis for our FVE system prototype. GamingAnywhere is an open source, modular and multi-platform cloud gaming software, with support for multiple encoders via ffmpeg [99]. We modified the server and client components of GamingAnywhere to implement FVE. Looking at Figure 3.1, the server executes the game logic which can be conceptualized as having a rendering module and a user interaction module. The GamingAnywhere server receives encoded user input events, decodes and replays them to the user interaction module of the game logic, while simultaneously capturing the rendered video from the rendering module and audio as output by the game logic. An encoder module encodes and multiplexes the video and audio into a stream which is served by a streaming server over a stateful streaming protocol, RTSP in this case, to the client. The GamingAnywhere client can be conceptualized to comprise an audio/video player and a user interaction module. A streaming client module receives the audio/video stream from the GamingAnywhere server, a decoder decodes it and it is displayed and played by the audio/video player. The user interaction module provides the user with input modalities to interact with the game. The user input interactions are captured, encoded and transmitted.
to the server. We modified the GamingAnywhere client to interface with Tobii

4C [100], a consumer grade gaze tracker, so gaze of a user playing a game on
the client is tracked, lightly encoded and transmitted to the GamingAnywhere
server. The encoded gaze is sent over a data channel separate from the au-
dio/video stream and the user interaction data channel. We modified the server
to receive and decode the gaze data and added logic to spatially vary the quality
of each frame according to the gaze. For encoding, we leveraged the x264 encoder
implementation of MPEG AVC [20].

To answer RQ2, we implemented a prototype remote rendering system wherein
the server is capable of both FVE and FR. In addition, the system is capable
of Foveated Warping, a technique of streaming foveated graphics we propose
in Publication II. The system was implemented using the popular game engine
Unity, with both the server and the client designed as Unity applications. The
server application uses an Nvidia GPU’s hardware acceleration for rendering
computations and variable rate shading (VRS) using the NVAPI rendering API
and video encoding using its H264 hardware encoder via the NVENC encoder
API. The client application can be run as a VR application on HTC Vive Pro,
Android devices or Oculus standalone HMD devices. The client application, in
addition to reporting HMD pose and player actions, also reports gaze data to the
server, if available. In the prototype, the server is configured to use the HMD
pose to control the camera position in the virtual scene being rendered while the
gaze data is used for FR, foveated video encoding or foveated warping.
3.3 Foveated Video Encoding and Cloud Gaming

Foveated video encoding has been studied for decades [101, 67, 63], driven by the desire to reduce size of compressed video. As mentioned earlier, additional compression beyond normal encoding is achieved in foveated encoding by spatially varying the quality profile of an encoded frame according to the acuity of the HVS. There have been different approaches to parameterize the quality fall off from the gaze point, all of which broadly attempt to match the HVS acuity profile [64, 65, 61, 63, 102, 71, 69, 88, 86, 103, 67]. Regardless of the parameterization used, foveated video encoding can result in substantial bitrate savings. However, for real time remotely rendered applications, implementation of such a system has to consider multiple challenges like real time gaze tracking, latency and tracking errors and encoding methodology. RQ1 of this thesis examines these issues and to answer RQ1, we implemented the cloud gaming system with real time foveated video encoding of section 3.2, evaluated the parameterization of FVE and its effect on video quality and bitrate. In this section we briefly present our implementation of FVE, parameterization and the results.

![Visual acuity of human eye](a) Visual acuity of human eye

![QO at different values of W](b) QO at different values of W

**Figure 3.2.** Foveation and QO calculation. FW is the width of the output frame in pixels. Figures from Publication I

As discussed in section 2.3, in block based encoding, quantization step incurs the most loss in detail. In the quantization step, frequency transform coefficients of macroblocks are quantized. In practice this takes the form of dividing the frequency transform coefficients with a quantization factor and rounding the results, so that most of the coefficients become zero. These quantized coefficients can be represented with fewer bits compared to unaltered coefficients. The higher the quantization factor used, lower the size in bits of the resulting coefficients but higher the loss in detail.

In our FVE scheme, we added a quantization offset on top of the quantization coefficient of each macroblock calculated by the underlying rate control algorithm. Our quality drop off curve models the acuity of HVS as a Gaussian. This is a conservative choice considering the visual acuity drop off from the gaze location is rather exponential. This choice is driven by the desire to ameliorate gaze tracker inaccuracy and m2p latency from gaze tracker to the screen update.
Reducing Bandwidth with Foveation

on the client. Figures 3.2a and 3.2b provide a side by side comparison of the acuity of HVS and value of quantization offsets with distance from gaze for three different quality drop off parameterizations. Based on gaze data analysis (see section 3.4), we observe that for the games considered in the evaluation, gaze location on the screen stays localized for a major fraction of the game play and that gaze traversals in subsequent frames beyond a distance of 12.5% of the screen size are rare. The quantization offset was calculated as:

$$QO(i,j) = QO_{max} \left( 1 - \exp\left(-\frac{(i-x)^2 + (j-y)^2}{2W^2}\right) \right)$$

(3.1)

Where $QO(i,j)$ is the quantization offset for a macroblock at $i$, $j$, where $i$ and $j$ are indices of the matrix of macroblocks comprising the frame, $QO_{max}$ is the maximum offset which is configurable by the server administrator (or a user), $x$ and $y$ are the indices of the macroblock corresponding to the gaze location, and $W$ is a measure of the size of the foveal region. $W$ is the full width at half maximum of the two dimensional Gaussian of Equation 3.1. Figure 3.2b illustrates quantization offsets for 3 values of $W$ when the maximum quantization offset $QO_{max}$ is set to 10. We parameterize $W$, which is a measure of size of the high quality region, in terms of frame width $FW$. This is because size of the frame often dictates the viewing distance which affects the acuity of human vision. The correlation of frame display size and viewing distance is an intuitive fact which is recognized and considered in standardized subjective video quality assessment protocols [104].

3.3.1 Evaluation

Answering RQ1, involves establishing that FVE, with suitable parameterization, leads to reduction in downstream bandwidth requirements without a discernible drop in perceived video quality. We investigated this from two perspectives. First we conducted a battery of tests to investigate the effect of different FVE parameterizations on bitrate. We selected a set of games representative of different genres for these tests. Specifically, we chose AssaultCube as a representative First Person Shooter game, Trine 2 as a representative platformer puzzle game and Little Racers Street as a representative birds eye view game. Second, we studied the effect of FVE parameterization on quality of the encoded video in terms of computational metrics of quality and verified the results with a subjective study for one genre of games.

Bandwidth

We parameterized FVE with different $QO$ and $FW$ values to investigate their effects on the resulting bitrates. The results of FVE parameterization on downstream bandwidth are shown in figure 3.3. The FVE parameterization in figure 3.3 uses the x264 encoder in crf rate control mode with adaptive quantization, with $crf = 28$ as recommended in [4]. The $QOs$ are applied on top of the $QP$ calculated by the $crf$ rate control algorithm. It is clear from figure 3.3, that
Reducing Bandwidth with Foveation

![Bitrate Comparison](image)

**Figure 3.3.** Measured video bitrates with different games and parameterization of foveated streaming. $FW$ is the width of the display in pixels. The box comprises the interquartile range, the red line in the middle of the box is the median, and the diamond denotes the mean. Copied from Publication I

varying the maximum quantization offset $QO_{max}$ has a stronger effect on bitrate than varying the radius of the high quality region, across different game genres. Reducing the size of the high quality region beyond $W = FW/8$ does not cause a correspondingly meaningful drop in bitrate as at the video resolution used in the measurements the number of macroblocks with low $QO$ offsets is already small at $W = FW/8$. Based on this insight and typical gaming scenarios, we selected $W = FW/8$ as a basis for FVE in further experiments. For a typical laptop screen DESKTOP screen viewing distance of about 50cm, and typical screen size of 50cm by 40cm, the foveal region is just 2cm wide, and $W = FW/8$ translates to a high quality circular region of 5cm diameter which superimposes on the foveal region with a reasonable margin for gaze errors due to latency or in tracking.

**Latency**

Interactive remotely rendered applications have tight latency requirements as discussed in earlier sections and cloud gaming is no exception. There is inherent latency in cloud gaming due to system factors on top of network latency [10]. For FVE to be feasible for cloud gaming, FVE should not introduce additional latency into the cloud gaming pipeline. Different latencies may be accrued due to FVE. There is latency between actual gaze translation due to eye movement and corresponding gaze point detection by the the eye tracker, transmission latency between the client and the server and the latency introduced into encoding process to spatially vary the quality. In our prototype gaze is detected, tracked and transmitted in parallel to the cloud gaming pipeline, so those latencies are
Reducing Bandwidth with Foveation

(a) AssaultCube  (b) LRS  (c) Formula Fusion  (d) Trine 2

Figure 3.4. Gaze tracking data from 15 minute gameplay sessions. a, b, c and d are gaze heatmaps, e and f are the CDFs of gaze moments: time periods where gaze lingers within a circular region of $FW/8$ (e) and $FW/4$ (f). g CDF of rate of gaze shifting

not additive to the end to end latency. Further, since the spatial variation of quality takes the form of relatively simple mathematical operations, the latency hit ought to be minimal. The end to end latency of FVE is most perceptible when gaze location changes between two subsequent frames. The gaze patterns of AssaultCube, Trine2, Little Racers Street and Formula Fusion, which is a racing game, are presented in figure 3.4. From the heatmaps of the games in 3.4 a-d the gaze patterns are localised around the center of the screen: highly so for Assault cube which is a First Person Shooter (FPS) game and not as highly for Little Racers Street which is birds eye view racing game. Looking at the gaze moments and rate of gaze change as well, it is clear that for majority of the time, consecutive gaze locations do not change substantially, which bodes well for foveated video encoding.

From a system latency perspective, an end to end latency of 100ms in cloud gaming is deemed acceptable [8]. A predecessor of the gaze tracker that was used in the prototype 3.2 has a latency of 50ms which when substituted in the device-to-kernel latency of a modern mobile device [10] in a cloud gaming system, suggests a sub 100ms end to end latency is still possible.

**Video Quality**

We computed video quality as Peak Signal to Noise Ratio (PSNR) and Eye Weighted Peak Signal to Noise Ratio (EWPSNR), with varying $QO_{max}$ and $W = FW/8$. For AssaultCube we also conducted a user study and collected Mean Opinion Scores (MOS). The user study was a controlled within-subjects study set-up according to [105] and conclusions in [106]. 12 study participants were recruited and asked to play the game under different foveation conditions whose presentation order was cycled with the participants. After each test condition,
the subjects rated various aspects of their experience on a 0-100 Likert scale. For details of the user study, please refer to Publication I. The results, converted to a MOS score, for video quality for AssaultCube are shown in figure 3.5. From the MOS, in conjunction with the bandwidth for each case it is clear that bandwidth reduction of up to 50% can be achieved with negligible loss in perceived quality. EWPSNR shows a similar trend, while PSNR, which gives equal weight to quality in the whole frame, shows a linear decrease in quality with increase in the maximum offset $Q_{O_{max}}$. It is evident that with proper FVE parameterization, it is possible to reduce the downstream bandwidth needed for a given perceived quality with foveated encoding in a cloud gaming scenario.

3.4 Rendering, Encoding and Foveated Streaming

As discussed in earlier sections, foveated video encoding is just one method of creating foveated video in a remote rendering system. In the pipeline of...
Reducing Bandwidth with Foveation

rendering and streaming graphics (as video) from a remote rendering server, foveated processing is possible at multiple stages: within the graphics pipeline, post render post-processing and at the video encoding stage. In the above section 3.3, the feasibility of FVE for off the shelf games in a cloud gaming environment is established. Within the rendering pipeline, foveated processing takes the form of FR. At the post-processing stage, it is also possible to use image transforming techniques like filtering and non-uniform sampling to generate a foveated image [63]. In Publication II, we proposed a novel real-time warping technique for the rendered frames at the post-processing stage which takes HVS into account, which we call Foveated Warping (FW). Further we evaluated and compared the proposed technique against FVE and FR to answer RQ2. Next we describe Foveated Warping, followed by a brief overview of our approaches to FR and FVE and the parameterization used in Publication II in the context of RQ2.

3.4.1 Foveated Warping

Foveated Warping, abbreviated as FW in this thesis, reduces the spatial (pixel) resolution of a frame corresponding to the HVS so that the number of pixels needed to represent a frame is reduced, which results in lower bitrate when the frame is encoded. In other words, FW spatially compresses a frame around a gaze point, the compression being a function of the distance from the gaze point; in order to correspond to the acuity of the HVS. Essentially, foveated warping is a mapping function from a frame with uniform spatial quality to a warped frame with fewer pixels from which an unwarped frame of original dimensions can be recreated by inverting the mapping. The unwarped frame has high quality at the point of gaze and the quality reduces with distance from the gaze location. Any foveation inspired spatial compression and decompression scheme has to consider the acuity profile of HVS given a gaze location. The acuity of HVS can be gauged by contrast sensitivity [107] which is measured empirically by displaying sinusoidal gratings as stimuli to test subjects. The frequency of these sinusoidal gratings per degree of visual angle discernible by the human eye varies with eccentricity of the stimulus from the fovea [31] and is characterized by the so called Contrast Sensitivity Function (CSF) and its inverse the Contrast Threshold Function (CTF). Based on the CSF used in video encoding applications [65, 69, 61, 64], and given \( C T_0 \) is the minimal contrast threshold, the cutoff frequency \( f_c \) at an eccentricity \( e \) is calculated as:

\[
f_c = \min \left( \frac{e_2 \ln \left( \frac{1}{C T_0} \right) \pi N v}{\alpha (e + e_2)}, \frac{\pi N v}{360} \right)
\]

(3.2)

where \( \alpha \) is spatial frequency decay constant, \( e_2 \) is half-resolution eccentricity constant, \( N \) is the image width and \( v \) viewing distance. The best parameterization of the constants \( \alpha, e_2 \) and \( C T_0 \) are reported as \( \alpha = 0.106, e_2 = 2.3 \) and \( C T_0 = 0.015625 \) in [67]. The term \( \frac{\pi N v}{360} \) in equation 3.2 is the display Nyquist frequency of the image.

43
Reducing Bandwidth with Foveation

When \( N \) is measured in pixels and \( v \) in picture widths, it corresponds to the maximum spatial frequency that can be displayed by an \( N \) pixel wide display without aliasing at a viewing distance of \( v \) display widths [108]. The eccentricity \( e \) for equation 3.2, given gaze location \( x_g, y_g \), can be calculated as:

\[
e(v, x, y) = \arctan \sqrt{\left(\frac{x - x_g}{Nv}\right)^2 + \left(\frac{y - y_g}{Nv}\right)^2}
\]

Our approach for FW takes an image \( I \) and a gaze fixation point at \( x_g, y_g \) to produce a foveated warped output image \( O \), such that at a given angular eccentricity from the gaze fixation point, the sampling rate of pixels from \( I \) to \( O \) is inversely proportional to the cut-off frequency. This can be achieved by transforming pixels in \( I \) to polar coordinates \( r, \phi \), centered at the gaze point \( x_g, y_g \), and scaling \( r \) by normalized cutoff frequency \( f_c \) to produce a warped image. \( O \) can be obtained by transforming the polar coordinates back to Cartesian coordinates. These operations result in (non-uniformly) sub-sampling pixels from \( I \) to \( O \). Since \( f_c \) is spatially varying, the sub-sampling is also spatially varying. Without sub-sampling, pixels in the output image \( O_{x_o,y_o} \) can mapped from pixels in the input image \( I_{x_i,y_i} \) by first transforming the pixels in input image into polar coordinates \( r, \phi \) as:

\[
r = \sqrt{(x_o - x_g)^2 + (y_o - y_g)^2}
\]

\[
\phi = \arctan \left( \frac{y_o - y_g}{x_o - x_g} \right)
\]

(3.3)

\( r \) and \( \phi \) can easily be converted back to Cartesian coordinates as:

\[
x_o = r \cos(\phi)
\]

\[
y_o = r \sin(\phi)
\]

For FW, to sub-sample pixels from \( I \) to \( O \), we scaled the radius \( r \) according to the normalized cutoff frequency \( f_{cn} \) at the output pixel. A pixel in the output image \( O_{x_o,y_o} \) is thus obtained as a pixel \( I_{x_i,y_i} \) from the input image, where:

\[
x_i = \frac{r}{f_{cn_o}} \cos(\phi)
\]

\[
y_i = \frac{r}{f_{cn_o}} \sin(\phi)
\]

(3.4)

where \( f_{cn_o} \) is the normalized cutoff frequency at a pixel with coordinates \( x_o \) and \( y_o \).

The mapping above corresponds to the cutoff frequency and to the spatial pixel resolution of HVS, but the transformation from Cartesian coordinates to scaled polar coordinates is hard to invert, because of the \( f_{cn_o} \) term in Equation 3.4. To overcome this challenge, we modelled the scaling term \( r/f_{cn_o} \) in equation 3.4 as a parabolic function \( p(r) = a + b \times r^2 \) so that Equation 3.4 becomes:

\[
x_i = p(r) \cos(\phi)
\]

\[
y_i = p(r) \sin(\phi)
\]

(3.5)
where $a$ and $b$ are parameterized empirically depending on the pixel dimensions of the frame and the viewing distance. A normally rendered frame and the corresponding warped frame and unwarped frame are shown in Figures 3.6a, 3.6c and 3.6d respectively.

### 3.4.2 Foveated Rendering

Foveated Rendering, abbreviated as FR in this thesis, can be actuated in multiple ways [62], we use Variable Rate Shading (VRS) as implemented by Nvidia in their GPUs, exposed to developers via the Nvidia VRWorks SDK. VRS actuates spatially varying quality by spatially varying the shading rate in a frame. Shading rate determines the sample to pixel ratio during shading, which determines the render quality. Lower number of samples per pixel translates to less detail, while higher number of samples per pixel translates to more detail [109]. FR is achieved by setting high shading rate at and around the gaze location and lower shading rate in areas spatially away from the gaze location. Nvidia VRWorks SDK allows setting shading rate in discrete steps across blocks of 16x16 pixels, which precludes using a smoothly varying shading rate profile in a frame.

### 3.4.3 Foveated Video Encoding

We implemented the FVE to compare against FR and FW in the same fashion as described in 3.3. In the second prototype we leveraged the Nvidia hardware H.264 encoder. We configured the encoder to accept an array of Quantization Offsets (QOs) to be applied to the Quantization Parameters (QPs) calculated by the rate control algorithm. As mentioned in Section 3.3, this approach allows for control of the total quantization factor applied to each macroblock and hence the quality across a frame.

### 3.4.4 Evaluation

For comparison of FW, FR and FVE we conducted computational and subjective evaluations of the second prototype discussed in 3.2. For computational evaluation, we configured the prototype to run a camera along the same trajec-
Reducing Bandwidth with Foveation in a 3D scene while producing a foveated video stream using FR only and then with FW and FVE. FW and FR streams were with warping and rendering parameterizations which aim to maintain similar spatial quality profiles. For FVE we parameterized the offset calculation so that the quality profiles of the output frame are similar to those used in FW and FW. The parameterizations for FW, FR and FVE considered a viewing distance of 3.5 frame widths and the frame size of 1088x1088 pixels, displayed at native resolution. In FW we parameterized \( p(r) \) with \( a = 0.112 \) and \( b = 2.2063 \) for \( r > 1/8 \) of the frame width, for \( r < 1/8 \) the normalized cut off frequency \( f_{cn} \) is one and hence we do not apply spatial compression. This translates to pixels within the foveal region (i.e \( r <= 1/8 \) of frame width) in the input image being mapped one to one to the high quality foveal region in the warped image while the rest of the pixels in the input image are scaled.

For FR, we divided the frame into a high quality region with an outer radius of 1/8 of the frame size, an intermediate quality annular transition region around the high quality region with an inner radius of 1/8 of the frame width and an outer radius of 1/6 of the frame width and the rest of the frame as low quality region. For high quality region, we set the shading rate to 1 shading pass per pixel, for the transition region 1 shading pass per 2x2 pixels and 1 shading pass per 4x4 pixels for the low quality region which comprises rest of the frame.

For FVE we adopted a QO calculation scheme that roughly matches the spatial quality profiles actuated for FW and FR. For a macroblock with normalized coordinates \( x, y \) and gaze fixation with normalized coordinates \( x_g, y_g \), the normalized pixel distance between the macroblock and gaze is \( r = \sqrt{(x-x_g)^2+(y-y_g)^2} \). We calculate the QO for the macroblock at \( x, y \) as:

\[
QO_{x,y} = 0, \quad r \leq 0.125
\]

\[
QO_{x,y} = QO_{\text{max}} \left( a + b \cdot r^2 \right), \quad r > 0.125
\]

(3.6)

For an input image of size 1088p x 1088p, we set \( a = 0.112 \) and \( b = 2.2063 \).

**Bandwidth and Quality**

We compared bitrate of frames obtained with FR and Normal Rendering (NR) with the same camera trajectory in a 3D scene and same encoding profiles with different QPs. The results are illustrated in Figure 3.7a. It is clear that FR does not lead to any bitrate savings when compared to NR frame. On the average, for the same encoding profile and QP, FR frames are the same or slightly bigger size than corresponding NR frames. This counter-intuitive result is because FR lowers inter-frame compression efficiency. We confirmed this assertion by comparing FR and NR frames compressed with intra-compression only which showed NR frames are bigger than FR frames on average. We already established in 3.3 that for normally rendered frames, FVE reduces bitrate compared to spatially uniform quality encoding for a given perceived quality. Next we compare the bitrates and quality of FR frames with FVE encoding or FR frames post-processed with FW and with normal encoding. The results are
shown in Figure 3.7b. It shows FVE with different QOs and FW with different QPs. The FVE condition with QO\_max = 0 corresponds to normal encoding. Expectedly, with increase in total quantization factor applied, the bitrate of frames decreases across FVE and FW. Both methods of foveation reduce bitrate compared to normal encoding. Excepting the extreme quantization profiles (e.g. QP = 30, QO\_max > 20) for a given FSSIM-Y, FVE achieves lower bitrate. These results are broadly also validated by DMOS scores in our user study, illustrated in Figure 3.7c. The user study followed ITU-T recommendations [104] and used Absolute Category-Hidden reference protocol for stimulus presentation and subjective rating. The test sequence was a 30 second exploration of a scene along a fixed trajectory, with different (QP, QO\_max ) values as test conditions against a (QP = 10, QO\_max = 0 ) reference. There are some unexpected irregularities.
Reducing Bandwidth with Foveation

around the conditions $QP = 20, QO_{\text{max}} = 0$ for FVE and $QP = 10$ for FW which we believe are due to temporal distortions becoming more perceptible at higher background quality, which is in conformance with observations in other work [90].

**Latency**

![figure](image)

(a) Encoding Delays (b) Decoding Delays

**Figure 3.8.** Encoding (a) and decoding (b) delays for FVE and FW

FW does not result in a rate-distortion improvement in terms of bitrate-FSSIM-Y, it does have the advantage of smaller pixel resolution for a given target render resolution. This can have advantages in latency of encoding and decoding and in delivering a high pixel resolution stream to mobile devices like smartphones and standalone HMDs, where the decoder has a cap on maximum decodable pixel resolution. We conducted latency measurements for both encoding and decoding for the different target render resolutions for FW and FVE cases on different hardware. For encoding latency measurements, we consider server grade hardware encoders. For decoding latency measurements, we consider a typical smartphone and a stand alone HMD as the representative thin clients. The results are shown in Figure 3.8. For a given target render resolution, FW results in lower encoding and decoding delays, especially so at higher target render resolutions.

### 3.5 Conclusion

In this chapter, we investigated avenues to leverage the non-uniform visual acuity of the HVS to reduce the downstream bandwidth requirements of remotely rendered interactive graphics applications. We established the feasibility of real time FVE for cloud gaming in Publication I and investigated alternative methods of generating foveated video frames from real time rendered graphics for remotely rendered graphics applications in Publication II.

We showed FVE is feasible for off the shelf games with consumer grade eye tracker. Further we proposed and evaluated FVE parameterizations and found
that some parameterizations can reduce bitrates by 25% to 40% without loss of perceived quality, but the parameterization depends upon game genre broadly and game in question specifically.

Further, we showed that FR with normal encoding, even though producing foveated video frames, does not result in a reduction of bitrate. However, FR together with FVE can reduce the downstream bitrate without affecting the perceived quality of the video stream. Further we proposed an HVS inspired warping technique FW which can reduce downstream bandwidth requirements and potentially encoding and decoding latency in remotely rendered applications. The proposed FW method also allows a client device to decode video whose render resolution is higher than the client's maximum decode resolution cap.
Real time remotely rendered graphics applications, as discussed previously, have challenging bandwidth and latency requirements. Latency in remotely rendered graphics is inevitable [10], due to inherent systemic factors. There are multiple components in the m2p latency of a remotely rendered system, for example, instrumentation, processing and network latencies. Instrumentation latencies include control delay at the client between a control signal originating from input devices to the time it is transmit to the server. Network latency includes access latency of the (radio) access network and the latency involved in the internet from the access point to the server. Within the server there are processing latencies which include the time taken to simulate the 3D scene, render and capture the graphics. Further, there is latency involved in encoding the frame. The network latency is again encountered in the frame delivery loop by the encoded frame. After the frame is delivered to the client, there is processing delay in the form of frame decoding latency and play out delay [10].

These latencies manifest themselves as motion to photon latency, which is the latency between user motion, be it an input event in cloud gaming or HMD/controller motion in cloud XR, to the corresponding frame update on the display at the client. In cloud gaming, this latency can cause decrease in quality of experience of the player as well as their in game performance like accuracy in aiming [9, 8, 28]. In cloud XR, as in locally rendered XR, motion to photon latency, in addition to decrease in QoE, can exacerbate symptoms of so called VR sickness. The amount of perceptible and acceptable latency is still being researched and appears to vary with task, head velocity and user among other factors [110, 111, 112]; from a minimum range of 3.2ms-51.0ms [111] to a maximum range of 180-320ms [110]. For remotely rendered VR, a latency of about 100ms is generally assumed to be acceptable and realistically achievable [113, 114].

The intuitive reason for latency in VR causing discomfort is the sensory mismatch it produces in the user, primarily between the visual and vestibular systems. Recall from section 2.1.2, rendering in HMD based VR experiences relies on accurately tracking the HMD pose in the physical world and mapping it to the location of a stereoscopic camera in the simulated 3D scene. Since
VR devices cover the entire field of vision of a user, motion to photon latency can manifest as a jarring discrepancy between motion sensed by the vestibular system and the motion perceived in the visual field by the eyes [115]. This mismatch can be accentuated by the human vestibulo-ocular reflex which refers to compensatory motion of human eyes when we move our heads for image stabilization [116]. Given the wide range of reported acceptable latencies for VR [110, 111, 112], latency is a factor even in locally rendered VR. This latency can come from the instrumentation used to track the HMD, the processing power of the HMD device or the tethered rendering computer as well as the refresh rate of the display. Although these latencies have been reduced with developments in hardware and HMD tracking instrumentation, predictive and filtering methods are still used to compensate for the residual latency and to remove noise [117]. Conventional Bayesian filters are attractive for this use case, owing to their proven accuracy at short prediction horizons and low computational complexity [118, 119]. Accurate predictions of pose of the HMD allows the rendering engine to render frames for temporally correct true pose of the HMD, in effect preventing sensory mismatch for the user.

Remotely rendered graphics are faced with inherently longer latencies than locally rendered graphics as discussed earlier. So, for remotely rendered VR, the prediction horizons for predicting a pose are substantially longer compared to locally rendered VR. Conventional pose prediction methods may not have the prediction accuracy adequate for good QoE in remotely rendered VR. RQ3 investigates what methods to use for such forecasting, primarily we consider HMD pose in VR and gaze of the user in VR. Accurate pose forecasting is essential for remote VR to have adequate QoE. We also consider gaze prediction as accurate gaze is needed for gaze based augmentations of remotely rendered graphics systems discussed in Chapter 3. Region of interest based encoding, which reduces bitrate, for conventional 2D video applications also benefits from accurate gaze prediction [120]. Although we consider XR applications for RQ3, the investigations may be useful for non-XR applications like Cloud Gaming where mouse or controller data may be considered as proxy for human kinematics and therefore predicted to compensate for latency [121].

Next in section 4.1 we discuss the background of time series prediction broadly and related work in head pose and gaze prediction specifically. In section 4.2 we describe the data we use in studying RQ3. This is followed by investigations of pose forecasting as conducted in Publication III in section 4.3. A discussion of gaze forecasting work carried out in Publication IV is presented in section 4.4. The chapter is concluded in 4.5.

It should be noted we use forecasting and predicting interchangeably in this thesis. However, in Publication III, we use the phrase “pose forecasting” to distinguish our work from the considerable amount of work in computer vision which focuses on estimating the pose of humans or objects from images or video frames and uses the phrase “pose prediction” for it.
4.1 Background and Related Work

In this section we briefly study the background and related works of pose and gaze forecasting. First, we briefly discuss visual attention modelling and saliency estimation work. We then review recent gaze and pose forecasting related work, starting with some background information on time series forecasting.

4.1.1 Visual Attention and Saliency Estimation

In computer science and signal processing, methods to predict and estimate where a user would look within a displayed picture have been long studied [122, 123] with applications in computer vision, computer graphics, robotics, health sciences, advertising, and image and video compression [124, 125]. Knowing which regions viewers are most likely to pay attention to in a video frame guides optimal spatial bit budget allocation [66, 120], which is the paradigm followed in Chapter 3. This research has drawn from multiple fields of science ranging from vision science, psychophysics of vision and computer vision to develop computational models of attention and saliency detection [126, 123]. Saliency detection techniques may use conventional algorithmic approaches [127, 128] or machine learning approaches [129, 130, 131]. Gaze tracking is a direct approach to gauge visual attention of a viewer being shown a visual stimulus. As such, gaze tracking is used to subjectively validate saliency models. Borji et. al [124] provide a look at various state of the art saliency models and a comprehensive comparison of these saliency models with subjective gaze data. Saliency models may be used for rate control directly, for example, using region of interest coding [66], or alternatively they may be used for gaze fixation prediction. Le Meur et. al. in [122] present a low level feature based saliency model for videos which leverages a fusion of both spatial and temporal attention. The model shows improved results when evaluated by predicting gaze fixations and compared to state of the art methods.

4.1.2 Gaze and Pose Forecasting

Gaze and (HMD) pose forecasting fall into the long studied domain of time series forecasting which studies predicting the future value of a time varying quantity based on its previous values. In the multimedia domain, pose and gaze forecasting have become especially relevant with the advent of immersive video streaming [132]. As such, gaze and pose forecasting can leverage saliency estimation techniques as discussed above in addition to historic gaze or pose information [133].

In the context of RQ3 specifically and immersive multimedia systems generally, time series forecasting mainly considers gaze and HMD pose forecasting [134, 121]. However, user kinematics forecasting can also be employed for user action prediction [121] for latency compensation. Within the immersive multimedia
field, eye gaze and HMD pose forecasting work is mainly directed towards 360° video streaming [133]. Eye gaze and HMD pose in 3 Degrees of Freedom (3DoF) is used as a proxy for the FOV of the user. Knowledge of FOV can be leveraged to reduce the downstream bandwidth requirements of omnidirectional video streaming by selectively streaming only the region of video in the user’s FOV or by streaming high quality video in the FOV and low quality video elsewhere [132].

Time Series forecasting

Time series forecasting, at its core, attempts to use values of a variable at past time steps to predict value(s) at future time step(s), as close to the eventual observed value(s) at the future time step(s). Generally, a sequence of variable values at past time steps are used to predict a sequence of values at future time steps. Mathematically, for a time varying quantity \( x_t \), the forecasting problem can be stated as: at time \( t = T \), given a sequence of historical data \( X_s = \{ x_t | t \in [T-L, T] \} \), where \( L \) is the sequence length of the historical data, forecast a sequence of values \( X_f = \{ x_t | t \in [T+1, H] \} \), where \( H \) is the prediction horizon, with the objective that the distance between forecasted values \( X_f \) and the (eventually) observed values at time steps \( t \in [T+1, T+H] \) is minimized. When \( H > 1 \), the forecasting is done for multiple steps and can be done iteratively or in one direct step [53]. \( X \) can be univariate or multivariate and the predictions can be univariate and multivariate depending on the use case.

Time series forecasting has a long history of applications and study as introduced in 2.5. Conventional models rely on domain knowledge and are hand-crafted [43, 44, 45, 46, 53, 135] while deep learning methods for time series forecasting [41] use an end to end learning approach. Recurrent neural networks [136], convolutional networks [47, 137] and Transformers [50, 51, 52, 53] have all been proposed for time series forecasting with results at par or better than state of the art conventional models.

Pose and Gaze forecasting for 360° Video

Building on the insight that human attention and hence gaze is correlated to saliency [124], pose and gaze forecasting techniques in 360° video streaming applications often leverage saliency information in addition to past pose and gaze data to predict future pose. Work in this domain aims to identify relevant spatial regions of future frames of panoramic video for streaming to a client, as such the actual predicted variable may be 3DoF pose [133], gaze [138], saliency [139] or FOV in tile coordinates wherein tiles are spatial regions of the video which may be streamed independently [140]. These models generally use recurrent neural architectures [134], although sparse directed graphs with hidden Markov models have also been proposed [141]. Rondon et al. [134, 133] do an in-depth analysis of recent work featuring such 3DoF methods, highlighting the shortcomings therein. They propose a novel recurrent model named TRACK. A building block of TRACK comprises recurrent layers for saliency and past orientation data,
Forecasting Pose and Gaze

further it introduces recurrent layers to perform fusion of the outputs of the saliency and orientation recurrent layers. These building blocks, configured in a sequence to sequence architecture form the complete model.

Prediction and forecasting solutions directed towards video on demand streaming applications generally have prediction horizons of the order of seconds. For remotely rendered interactive applications, these prediction horizons may not be ideal given the latency constraints. Further, interactive VR environments typically have 6 Degrees of Freedom (6DoF) which preclude use of 3DoF forecasting solutions, especially for pose.

Pose and Gaze forecasting for XR systems

HMD Pose prediction has been studied since the advent of HMDs for military applications [142]. In the context of VR HMDs, the earliest HMD pose prediction solutions mainly targeted suppressing instrumentation noise and predicting pose for the next frame to be rendered. These solutions leveraged developments in rigid body pose tracking from the aerospace field [143] and rely on statistical methods. A discussion and analysis of classical techniques of pose prediction can be found in [118].

Gaze tracking has been studied for a while, especially in vision and cognitive science fields [144]. There has been considerable work in gaze prediction across multiple fields, including psychology, behavioural sciences, cognitive sciences and in computer science [144]. In computer science, the focus has been on visual attention prediction for images [144, 66, 145]. Some work in gaze forecasting based on gaze motion has also been conducted in the field of gaze contingent displays [146, 147], generally using planar displays and with head motion restricted. Recently, there has been increased interest in predicting gaze in egocentric video leveraging machine learning approaches, e.g. [148] and the references therein. The task of gaze prediction in egocentric video is similar to VR scenes at a high level, but the application scenarios as well as the available data is different between gaze prediction in egocentric natural video and gaze prediction in VR scenes. Gaze prediction for egocentric video is generally not a real-time task [149].

Gaze Forecasting

As compared to gaze and pose prediction for 360° video streaming use cases, gaze and pose forecasting beyond the next time step or the next frame has been a relatively unexplored area in interactive VR applications, especially in the remote rendering paradigm. Some relevant recent works in gaze forecasting for interactive VR applications [150, 151, 152, 153] are discussed below.

Hu et. al. present SGaze, [150], DGaze [151] and FixationNet [152] for gaze and fixation prediction in VR environments, all being learning based models. SGaze uses saliency data of 35° of the visual field in addition to head motion to predict current gaze in a scene, as opposed to future gaze. DGaze, in addition to saliency and head motion data, uses scene object information for gaze prediction in dynamic VR environments. A variant of the solution also uses past gaze
data for gaze prediction and is called DGaze_ET by the authors. The proposed solution leverages CNNs to encode sequences of position and motion data, and SAM-ResNet [130] to extract saliency of rendered frames. The outputs are fed to a feed forward network to predict gaze position, either for the current time-step or for a time horizon in future. When compared to a copylast baseline, DGaze shows better performance beyond prediction horizons of around 340ms, while DGaze_ET shows better performance beyond prediction horizons of 100ms. Closer to 1000ms, both models have similar accuracy which is close to a gaze mean baseline predictor. FixationNet incorporates task information in addition to saliency and past head motion and gaze data to predict gaze fixations task oriented VR scenes.

The works by Hu et. al. show promising first results in human kinematics prediction in VR environments. However, these studies are based on data from VR scenes with limited interaction, task and content dynamics which may not reflect consumer VR experiences. Further, some of the data used in the models, like object position or task information may not be available, especially in VR scenes with scene complexity or some task complexity. Extracting saliency may introduce latency especially in a remotely rendered VR scene, considering that frame buffer capture and processing can be computationally heavy.

Rolff et al. [153] propose GazeTransformer, a Transformer [154] based model for gaze prediction in VR which eschews VR experience based data and uses data which can be obtained from the HMD itself. The input data modalities considered are past head data, gaze data, image data and task data. The authors consider different backends for image data processing, like grayscaling, downscaling, and various machine learning models. The authors consider task data to be derivable from gaze and head data modalities based on recent work [155]. This assumption, however, may impact latency performance of the model. The authors compare GazeTransformer with DGaze and FixationNet on the DGaze and FixationNet datasets and report performance improvements over both DGaze and FixationNet. An interesting observation from [153] is that the best performing model does not rely on any image data. This result is similar to observations by Rondo et al. [133] when comparing state of the art FoV prediction solutions in 360° video streaming. Since DGaze and FixationNet datasets are used in [153], another limitation of the work are the limited scene, interaction and task dynamics in the datasets used.

**Pose Forecasting**

Prediction of HMD pose in 6DoF for prediction horizons longer than the time to next frame is a relatively new area of research and has come into focus with remote rendering and volumetric multimedia streaming paradigms being studied now. Some recent relevant works for 6DoF pose forecasting [156, 157, 26, 158] are discussed below.

Hou et al. [156, 157] propose an remotely rendered VR system wherein the remote server is deployed at the network edge. To compensate for the motion to photon latency added by the remote rendering paradigm, the authors propose
Forecasting Pose and Gaze prediction of position and orientation separately using either a Multi Layer Perceptron (MLP) based model or an LSTM based model. The prediction horizons considered are for next frame which may be too optimistic for remotely rendered VR. Further, in the proposed solution head pose data is preprocessed using a Savitzky–Golay filter which can lead to data leakage across time steps as well as smoothing out of micro-motions.

Gul et al. [26] propose a Kalman filter with a 14D state vector as a predictor of HMD pose for a volumetric MR streaming system to ameliorate registration errors. The position and orientation components of the pose being expressed as a Cartesian coordinates and unit quaternions respectively. The state vector of the proposed Kalman filter comprises position and orientation vectors and their first time derivatives. The system model defined fails to account for non-unique nature of quaternion representation of orientation. Further, the solution is trained and tested on relatively short (60s) traces of data collected in a limited AR scene. Another limitation of the solution may be the preprocessing undertaken to up-sample the sampling frequency of the pose data from 60Hz to 200Hz. The up-sampling assumes uniform translational velocity between consecutive position samples and uniform angular velocity and axis of rotation for orientation.

Yoon et al. [158] propose to combine predictions from a consistent motion assuming linear predictor and an LSTM based deep learning predictor for 6DoF pose prediction in a remotely rendered AR system. Pose is represented as a tuple of position in Cartesian coordinates and orientation in Euler vector notation and a sequence of poses is input to the linear predictor and the deep learning predictor. The authors report best performance when the outputs of the linear predictor and the deep learning predictor are summed using a learned weight. The proposed solution shows good results for a prediction horizon of 150ms. The study is limited by the dataset used, which comprises of short (70s) traces of AR glass usage by 24 participants. Further, the proposed solution relies on access to raw IMU sensor data which may not be available for consumer HMD devices. Another possible limitation of the solution is the use of Euler vector rotation representation which may not be optimal for deep learning algorithms [18], an issue which we briefly discuss in section 4.3.1.

A good solution for human body kinematics forecasting in the context of remote rendering should make predictions fast enough so as to not block frame render cycles at the rendering server. Further, it should be usable with off the shelf computer graphics applications. This generally translates to using only data available from the Application Programming Interface (API) of the rendering engine and/or the thin client. Thin clients like laptops, tablets and smartphones can provide input events like key presses, mouse clicks, joystick-handheld controller device events etc. Devices like tablets, smartphones and HMDs, which may be used as thin clients for remotely rendered XR systems also provide orientation data of the device and gaze data if eye tracking is available. Rendered frames are also available from the rendering API and this fact is
leveraged by remote rendering solutions which work with off the shelf graphics applications. Application state information, like position of 3D objects within a graphics application, task information and such are generally not accessible directly and need hooks into the graphics application.

The investigation of RQ3 in this thesis considers only data which is accessible without needing hooks or modifications to off the shelf graphics application. Further, we make a design decision to not use rendered images in our solutions due to two primary reasons: First, image processing is compute intensive and may incur a latency in prediction as well as the capture and encoding pipeline of the remote rendering system; and second, at the prediction horizons considered acceptable for remotely rendered XR applications and cloud gaming, image data may not be very beneficial for gaze [153] or HMD pose prediction [133]. In the interactive immersive multimedia application system architecture introduced in 2.1, the prediction is envisioned to take place in the remote graphics server before the render and encode operations for a frame are executed.

4.2 Data

Good data is necessary for building robust deep learning models for prediction. A good quality dataset should contain data which is accurate, has adequate variability, large enough size and should capture possible underlying patterns exhaustively. For VR use cases this means the data set should comprise of data traces collected while subjects experience VR scenes which have dynamic content, broad spectrum of interaction paradigms and non confounding task choices. A major shortcoming of previous work in the domain of pose and gaze prediction is the datasets are limited either in size (small number of short traces), nature of content (no or little dynamic content), limited interaction (simple exploration of static content) or limited task choices (searching for or exploring a 3D object). In view of these limitations we used the OpenNEEDs [159] dataset as the primary dataset to investigate RQ3. OpenNEEDs dataset is a publicly available dataset of multiple data modalities collected from subjects exploring interactive open-ended virtual environments in 6DoF with a sampling frequency of 90Hz. The dataset is collected from 44 participants wearing an Oculus Rift based HMD with a FoV of 104°equipped with custom eye tracker, and using Oculus controllers to interact with the VR environment displayed in the HMD. The data modalities captured include HMD and controller pose, gaze vectors, pixel motion in screen space, color and depth images being displayed to the participant and orientation of interactive objects in the scene. Two VR scenes, one indoor and one outdoor, with the same interactive content were experienced by the participants. The interactive content was designed to elicit a range of task involvement and behaviour from the subjects, ranging from typical point and shoot gaming, reading, 3D object manipulation, throwing, free hand drawing and free exploration. The subjects were provided information about
the available task and interaction dynamics within the VR scene on a virtual clipboard without being given instructions to choose a particular task.

From the OpenNEEDs dataset we selected the data modalities that would typically be available from an off the shelf VR experience as well. These include HMD pose, controller pose, gaze vector and rendered image data. However, as discussed in section 4.1, we ignored image data because of potential latency implications of using image data for pose or gaze prediction and recent indications that image data may not be particularly helpful in gaze or pose prediction at the prediction horizons considered in this thesis for remotely rendered applications in the context of RQ3.

We also validated the gaze forecasting model developed in Publication IV with a small scale dataset of subjects playing off-the-shelf VR games that we collected. In this VR gaming dataset, we collected pose and gaze data from three subjects over six sessions, in each session the subjects played mini-games from Nvidia VR Funhouse\(^2\) and Steam Lab\(^3\). The mini-games in both VR Funhouse and Steam Lab have varied gameplay dynamics representative of different genres. At the start of each data collection session, calibration was carried out for the HMD eye tracker and the subjects were given the freedom to chose what mini-game they played. The equipment used was an HTC Vive Pro Eye HMD and a workstation powered by Intel Xeon W-2133 CPU, an Nvidia RTX 2080 GPU and 32 GB RAM with a custom Unity application collecting the data at 90Hz which is the nominal sampling rate of the HMD.

To investigate the impact of sampling frequency on gaze forecasting, we used GazeBase dataset [160]: a high sampling frequency (1000Hz) large-scale dataset containing gaze data of more than 300 participants performing various tasks with stimuli presented on a 2D display and head stabilized during data collection.

Next we discuss pose forecasting and gaze forecasting in sections 4.3 and 4.4, and present our results pursuant to investigations of RQ3.

### 4.3 Pose Forecasting

Pose forecasting, as discussed in this thesis, refers to forecasting HMD pose from data from the past, generally a sequence of data points in the immediate past. We make this distinction because pose prediction and pose estimation has been commonly used in the computer vision field for estimation of human head or face or other rigid body pose from images or image sequences [161]

To forecast pose from the data modalities discussed in 4.2, data representation and data preprocessing are the first steps that need to be addressed because they affect system models and their ability to forecast accurately [18].

\(^2\)https://store.steampowered.com/app/468700/NVIDIA_VR_Funhouse/
\(^3\)https://store.steampowered.com/app/450390/The_Lab/
4.3.1 Pose Representation

The pose of three-dimensional rigid body in a space is an intuitive concept to define "where and how" the body is situated in the space and can be thought of as the position of the rigid body together with an orientation with respect to a coordinate system in that space [162, 163].

In an Euclidean three dimensional space, relevant to real physical space, the position of a rigid body can be uniquely characterized using a position vector $p = [x, y, z]$ in Cartesian coordinates. Uniquely characterizing the orientation of a rigid body, however, is more complicated especially when considering rotation operations [164, 165, 166]. While there are many different representations of orientations like Euler angles, (Euler) angle-axis pair, Euler parameters, rotation vectors, rotation matrices, classical or modified Rodrigues parameters, exponential map, unit quaternions and Cayley-Klein parameters [167, 165, 164, 166], Euler angles, rotation matrices and quaternions are the most relevant for RQ3 as they are commonly used in computer graphics applications [168]. A detailed discussion of orientation representations can be found in [167].

Euler angles, as three rotations $[\alpha, \beta, \gamma]$ in a pre-agreed order around the three Cartesian axes $X, Y, Z$ are the most intuitive way to represent orientation. They are not very well suited for computational applications because of shortcomings like their local nature, non-uniqueness and occurrence of gimbal lock [168, 166].

Rotation matrices, as the name suggests, use a matrix to represent an orientation. The matrix is 3x3 and orthogonal and provides a unique representation for each possible orientation. However, rotation matrices are not compact and can be numerically unstable as orientation reaches a singularity. In the context of machine learning, the orthogonality constraint may unsuitable for learning [18].

Quaternions (as unit quaternions) with four components are a compact and numerically stable representation of orientation. A quaternion $q = (q_w, q_x, q_y, q_z)$ has a real or scalar part $q_w$, and and imaginary vector part $(q_x, q_y, q_z)$. However, quaternions suffer from non-uniqueness which can lead to numeric discontinuities in computations if some uniqueness mechanism is not enforced [169].

Pose of a rigid body can be have different representations depending upon the representation used for orientation. In the field of computer graphics graphics, quaternions are commonly used to represent orientation and consequently pose $\zeta$ is represented as $\zeta = (p, q)$ where $p$ is the position as a 3D vector and $q$ is a unit quaternion. This representation is commonly reported by VR and AR SDKs.

The orientation representations discussed above have shortcomings in the context of machine learning. Rotation matrices are continuous and unique, but are not compact and require an orthogonality constraint. Euler angles are intuitive and continuous locally, but suffer from gimbal lock and are non-unique. Quaternions are compact and continuous, but are non-unique. Recent work [18] has shown that these representations are not suited for machine learning applications. Consequently, pose parameterizations using any of these representations for orientations face the same issue.
A 6-dimensional (6D) representation of orientation suitable for machine learning proposed in [18] has shown encouraging results for pose estimation from images [170]. Given the prediction horizons relevant to this RQ3 of this thesis are in the range of 80-150ms and the kinematics are restricted by the physical aspects of human body, the relative deltas between two orientations are small. This thesis also investigates whether typical orientation representations are suitable in the context of HMD pose forecasting at such small time scales.

The pose representation for ML problem is complicated by the need for a differentiable distance metric between two poses, if pose is to be used as a single quantity in an ML application. Consequent to the variety in pose representations, a variety of distance metrics have also been proposed [163, 171, 172]. In the context of pose forecasting a distance metric between a ground truth pose and forecasted pose has to reflect the target use case i.e. of predictive graphics rendering. Given a frame rendered for a forecasted or past pose and a ground truth current pose, correcting for orientation errors has been found to be easier than correcting for translational errors [5, 173].

Positive results have been shown in pose estimation from image and video sequences with a pose distance metric that is a weighted combination of translational error and orientation error [174, 171]. In this thesis we use this approach when forecasting pose as a single vector.

Euclidean distance is a common distance metric for vectors used in machine learning applications for numeric data. However, pose represented as concatenated vector of a 3D position vector and orientation in most representations, e.g 3D Euler angles or 4D quaternions, don’t exist in the same geometric space necessarily. This precludes Euclidean distance between two pose vectors from being a good distance metric. In Publication III we proposed a pose representation which transforms orientation to the same geometric space as the position.

**Three Point Representation of Pose**

The proposed pose representation uses two vectors originating from the back of the head to the two eye positions to represent HMD pose. This representation can be also be visualized as a triangle with vertices at the back of the head, right eye and left eye or three points corresponding to the back of the head and the two eyes. To map a given pose to the proposed three point representation, we compute the position of the eyes and back of the head by adding offsets to the HMD position corresponding to average IPD and average head diameter for humans, followed by applying a rotation, equivalent to the orientation component of the pose, to the vectors so obtained. To map the three representation back to a pose representation with separate position and orientation, we first calculate the HMD position from the centroid of the triangle formed by the three points. From the HMD position, we calculate an un-rotated set of points corresponding to the eyes and back of the head. To uniquely map orientation [175], we use Kabsch algorithm [176] between the (rotated points) of the three point representation and the un-rotated points calculated from the HMD position.
4.3.2 Preprocessing

Data obtained from sensors is often noisy, which is true for the sensors on VR HMDs and controllers like Inertial Measurement Units (IMUs) and gaze trackers. In practice pose data is obtained by fusion of raw data from different kinds of sensors, for example, HTC Vive uses IMU and LightHouse Optical Tracking sensors to estimate pose of the HMD [117, 177, 178]. Common XR device runtimes provide only a filtered version of this data to an XR SDK [178, 177, 179, 180]. ML models show optimal learning and prediction performance when data preprocessing steps, like text to vector encoding or numeric normalization, are applied to the data [181].

We used OpenNEEDs as the dataset for our investigations into pose forecasting aspects of RQ3. HMDs often report data at slight but variable offsets from their nominal reporting frequency [177]. We analysed the inter-sampling windows and found the sampling interval to vary considerably. We re-sampled the data to have a constant sampling frequency if the deviation from the nominal sampling frequency was less than threshold: half the normal inter-sample time window. In case the deviation was longer than the threshold, we split the data trace and count the rest of the trace as a new session. For resampling, we used linear interpolation for position data and spherical linear interpolation for quaternions representing rotation [168]. We also enforced a quaternion sanitization regime to prevent consecutive orientations in a trace being represented by their antipodal equivalents. In practice we ensured that the geodesic distance between orientation components of two consecutive pose samples expressed as quaternions [169]. We also used delta poses as input data for some models. To calculate delta poses, we made sure to calculate position delta in Euclidean space and orientation delta in quaternion space, using quaternion operations for orientation where applicable [168]. An essential constraint that we enforced was that the sliding windows did not slide across trace boundaries between data traces collected during different sessions. This constraints prevents sudden artificial drifts between consecutive data samples. More details on preprocessing steps followed for pose data can be found in Publication III.

4.3.3 DNN Design

The pose forecasting problem, as mentioned in earlier sections, has been tackled both with conventional forecasting techniques and DNN based techniques. To answer RQ3 with respect to HMD pose, we considered RNN based LSTM models to forecast pose using past pose data and possibly additional modalities of data like controller pose information and gaze data.

LSTM models
LSTM models, being recurrent in nature, are considered a good fit for time series data applications as they have the capacity to learn over a temporal dimension.
Figure 4.1. Data processing pipeline of LSTM-based models.

Base on our preliminary experimental results we devised a generic data flow architecture for our models. The architecture, shown in Figure 4.1, uses a sequence of past delta-poses as input to the model whereas a residual connection carries the corresponding sequence of poses to a summation layer which sums the output of the model being considered with the residuals to predict pose for a given prediction horizon.

The LSTM models considered for evaluation comprise of multiple stacked LSTM layers, followed by two fully connected layers to reduce the dimensionality of the outputs from LSTM layers to desired dimensions. We empirically chose number the of LSTM layers and the size of their hidden state based on the model for each pose representation. Consequently, for pose representations wherein the position was represented as a Cartesian coordinate and the orientation as either Euler angles, quaternions or as a 6D vector, 4 LSTM layers were used. When pose was represented with our three point representation for HMD pose, we used 3 LSTM layers. Further we also chose different sizes for the hidden state of the LSTM layers for separate and joint prediction models depending upon model performance.

4.3.4 Data Fusion LSTM model

When multiple modalities of data are available to be used for a learning based task, they can be combined in three different ways: pre-fusion, mid-fusion and post-fusion. Prefusion refers to concatenating data from different modalities into a single vector before being fed into a model. Mid-fusion refers to inputting the different modalities separately, but combining intermediate outputs from each of them into a single vector within the model, the vector being input into the next stage of the model. Finally post-fusion refers to the method in which each modality is essentially fed into a separate leg of the model and final outputs of each leg are then combined. Pre-fusion and post-fusion of non-HMD pose data considered in Publication III i.e. controller pose and user eye gaze did not yield competitive results. To implement mid-fusion inside an LSTM model, we designed a novel data Fusion LSTM layer. We named the resulting LSTM model "Data Fusion LSTM" model.

The proposed Data Fusion LSTM model comprises of one novel data fusion LSTM layer made of custom LSTM cells. Each custom cell takes as input the HMD pose which is processed in a typical LSTM circuit and additional modalities
Figure 4.2. Data fusion model for pose and gaze, indicated with the $p$ and $g$ superscripts respectively; $x_t$ is the input sample at time $t$.

which are processed by a modified LSTM circuit which does not output a hidden state for a time step, unlike a typical LSTM circuit. An example of such a layer is shown in Figure 4.2, which considers HMD pose and the additional modality of gaze data. Figure 4.2 illustrates the computation of pose cell state $c_p^t$ and gaze cell state $c_g^t$ by corresponding LSTM circuits separately and the fusion of these cell states in the pose LSTM circuit. Also note that the hidden state $h_p^t$ is only computed and propagated forward for the pose circuit. The operations carried out by the model for can be described as:

$$f_p^t = \sigma(W_f^p[h_{t-1}^p; x_t^p] + b_f^p),$$

$$i_p^t = \sigma(W_i^p[h_{t-1}^p; x_t^p] + b_i^p),$$

$$o_p^t = \sigma(W_o^p[h_{t-1}^p; x_t^p] + b_o^p),$$

$$c_p^t = f_p^t \odot c_{t-1}^p + i_p^t \odot (\tanh(W_c^p[h_{t-1}^p; x_t^p] + b_c^p)),$$

$$f_g^t = \sigma(W_f^g x_t^g + b_f^g),$$
\[ i^g_t = \sigma(W^g_i x^g_t + b^g_i), \]
\[ o^g_t = \sigma(W^g_o x^g_t + b^g_o), \]
\[ c^g_t = f^g_t \odot c^g_{t-1} + i^g_t \odot (\tanh(W^g_c x^g_t + b^g_c)), \]
\[ c_t = W_p \odot c^p_t + W_g \odot c^g_t, \]
\[ h^p_t = o^p_t \odot \tanh(c_t), \]

where \( W_p \) and \( W_g \) are weight matrices for the cell state computed by the pose LSTM circuit and the gaze LSTM circuit. In Publication III, we used a 2 layer Fusion LSTM model, in which the first layer comprised of the data fusion LSTM layer presented above. Rest of the model was similar to the LSTM models discussed above.

**Other Models to Evaluate Against**

To evaluate whether it is possible to predict HMD pose and if so which approaches work, we selected the following representative models in addition to the LSTM, and data fusion LSTM based models.

- **DESP**: This model is state of the art DESP filter for pose prediction proposed in [119], configured to use a separate filter for predicting individual time series for the Cartesian coordinates \( x_t, y_t, z_t \) and quaternions \( q_w, q_x, q_y, q_z \).

- **DESPConv**: We developed this model as an implementation of DESP as a Convolutional Neural Network (CNN) [182]. Specifically, it consists of three convolutional layers (equivalent to weighted smoothing filters with learnable weights), the output from each being stacked which is input to a fully-connected layer for the actual prediction. The input time series sequences for position and orientation data (same as DESP), are each fed-forward individually through convolution and fully-connected layers to output corresponding position and orientation predictions.

- **TRACK**: This model is a state of the art LSTM-based predictor proposed in [133], for 3DoF pose prediction in streaming of 360°videos use cases. In TRACK mid-fusion of previous orientation and video frame data is leveraged to forecast future 3DoF pose (i.e., orientation), for prediction horizons in the order of seconds; We modified TRACK to accommodate 6DoF HMD pose, gaze and optionally controller pose as input data modalities to predict 6DoF HMD pose.

- **Dlinear**: This model is proposed as a simple DNN based baseline timeseries in [53] based on time series decomposition of input data into trend and remainder components. These components go through separate fully connected layers, the output of these layers being summed to compute the final prediction.
• Copylast, This refers to a case of no prediction as a baseline where the last known HMD pose point is used as the predicted value.

### 4.3.5 Summary of Results

Table 4.1. Summary statistics of accuracy evaluations, The cases are represented as \textit{model\_training mode\_pose representation+additional data modality}, training mode can be JT = Joint Training or ST = Separate Training, for cases where position and orientation components are learnt and forecast jointly and separately respectively. Pose representation can be: XR = position+quaternion, 6D = position+6D, 3P = three point, XE = position+Euler Angles and the additional data modalities can be gaze data (G) and controller pose (C). 2L, 3L and 4L show the number (2, 3 and 4 respectively) of LSTM layers in the LSTM-based models.

<table>
<thead>
<tr>
<th>Model</th>
<th>Position (mm)</th>
<th>Orientation (deg)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Median</td>
<td>Mean</td>
</tr>
<tr>
<td>DLinear_JT_XR</td>
<td>1.56</td>
<td>2.49</td>
</tr>
<tr>
<td>DESP</td>
<td>1.43</td>
<td>3.41</td>
</tr>
<tr>
<td>DESPConv</td>
<td>1.63</td>
<td>2.77</td>
</tr>
<tr>
<td>4L_ST_XR</td>
<td>1.00</td>
<td>1.70</td>
</tr>
<tr>
<td>4L_JT_XR</td>
<td>4.33</td>
<td>8.43</td>
</tr>
<tr>
<td>DLinear_JT_3P</td>
<td>2.35</td>
<td>3.29</td>
</tr>
<tr>
<td>DLinear_JT_6D</td>
<td>1.66</td>
<td>3.09</td>
</tr>
<tr>
<td>DLinear_JT_XE</td>
<td>1.36</td>
<td>2.80</td>
</tr>
<tr>
<td>3L_JT_3P</td>
<td>1.20</td>
<td>1.78</td>
</tr>
<tr>
<td>4L_JT_6D</td>
<td>2.45</td>
<td>4.10</td>
</tr>
<tr>
<td>4L_JT_XE</td>
<td>3.84</td>
<td>6.96</td>
</tr>
<tr>
<td>DLinear_JT_XRG</td>
<td>1.51</td>
<td>2.58</td>
</tr>
<tr>
<td>DLinear_JT_XRG_C</td>
<td>2.30</td>
<td>3.37</td>
</tr>
<tr>
<td>4L_JT_XRG</td>
<td>2.30</td>
<td>3.30</td>
</tr>
<tr>
<td>4L_JT_XRG_C</td>
<td>1.83</td>
<td>2.84</td>
</tr>
<tr>
<td>TRACK_JT_XRG</td>
<td>1.96</td>
<td>3.18</td>
</tr>
<tr>
<td>TRACK_JT_XRG_C</td>
<td>2.33</td>
<td>3.57</td>
</tr>
<tr>
<td>Fusion2L_JT_XRG</td>
<td>1.32</td>
<td>2.72</td>
</tr>
<tr>
<td>Fusion2L_JT_XRG_C</td>
<td>1.28</td>
<td>2.67</td>
</tr>
<tr>
<td>Copylast</td>
<td>4.33</td>
<td>8.42</td>
</tr>
</tbody>
</table>

In this section we summarize the results of our investigations of pose aspects of RQ3, primarily analysing whether forecasting HMD pose as a single quantity is possible with learning based methods. We attempt to answer the following questions in this regard: whether learning based methods have better forecasting accuracy than conventional methods, whether jointly forecasting the position and orientation components of pose is possible and if so what is the accuracy with respect to separate forecasting of position and orientation components of a
pose and whether auxiliary modalities of data, in addition to HMD pose, can be leveraged to improve pose forecasting accuracy. The results are summarized in Table 4.1.

In our analysis of conventional methods of forecasting vs learning based methods, we used a DESP model designed for pose prediction and shown to have equal or better performance when compared to Kalman Filters [119]. We compared this model with simpler learning based models: Dlinear and DESPConv, introduced in the previous section. It is clear that no prediction case: Copylast has the worst errors both in forecasting position and orientation. The learning based methods reduce the mean position error more compared to the DESP model, while the median position error is lower for DESP. However, a closer inspection of the distribution of errors (see Publication III) reveals better position accuracy for up to 99th percentile of measurements for learning based methods. For orientation forecasting, DESP, DESPConv and Dlinear have similar accuracy with errors almost half of the no prediction case. However, the error in no prediction case is sub 1°, so the results have to be considered carefully, especially because the margin of orientation error for most HMDs is close to this range. Considering learning based methods and comparing training and forecasting separately for position and orientation with the cases where forecasting is done jointly, using only pose data in position+quaternion (XR) representation, it is clear that separate forecasting with a 4 layer LSTM model (4L_ST_XR) has the best accuracy for position forecasting and close to best performance in orientation forecasting.

When evaluating different pose representations for jointly training and forecasting pose, our LSTM based model (3L_JT_3P) with 3 LSTM layers using our proposed three point representation from section 4.3.1 has the best position forecasting accuracy and a good to poor orientation forecasting accuracy, while the 4 layer model (4L_JT_XE) using position+Euler Angles representation has the highest orientation forecasting accuracy, but poor position forecasting accuracy. The 6D representation performs moderately well in both position and orientation prediction.

Considering modalities of data in addition to pose, our proposed 2 layer data Fusion Model using a fusion layer from section 4.3.4 performs the best in terms of both position and orientation forecasting, illustrating that additional modalities of sensor data in a VR environment may indeed improve forecasting accuracy when leveraged in a suitable fashion. The results for forecasting errors using different pose representations illustrated in Figure 4.3 and using additional modalities of data are illustrated in Figure 4.4. More illustrations of the results are provided in Publication III.

In order to analyse whether the models considered for accuracy performance can be run in real time for a VR application, we measured the inference times for each model on a rendering server class workstation with a state of the art GPU: Nvidia RTX 3080Ti. The inference times were measured both using Pytorch, which was also used for the model implementation, as well as TensorRT which is a high-performance inference runtime for production environments.
The inference times and number of model parameters, which can reflect model complexity are summarized in Table 4.2. From the results, it is clear that even though the range of number of parameters is quite large for models considered, even the most complex models can be run in real time with both TensorRT and Pytorch runtimes. The fact that data fusion based models considered here such as our Fusion Model and modified TRACK [134] can be run faster than frame rates of typical VR applications is promising.
4.4 Gaze Forecasting

Accurate gaze, as noted in the beginning of this chapter, is essential to deliver gaze-based augmentations explored in Chapter 3. If there is a mismatch between the gaze point used in FVE or FR and the current gaze point, the user might see a region of the video frame which has low quality, lowering QoE and possibly also causing visual discomfort. In this section we discuss our investigation into \textbf{RQ3} in the context of gaze, specifically in the context of head mounted VR/XR. This investigation was conducted in Publication IV. We propose a learning based LSTM model for gaze prediction which outperforms conventional prediction models as well as naive MLP and LSTM models. Keeping in view the real time...
Table 4.2. Summary of parameter count and inference times for different models and runtimes.

<table>
<thead>
<tr>
<th>Model</th>
<th>Parameters (k)</th>
<th>Inference time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>PyTorch</td>
</tr>
<tr>
<td>3L_JT_3P</td>
<td>2,310</td>
<td>1.7</td>
</tr>
<tr>
<td>4L_JT_6D</td>
<td>3,820</td>
<td>4.1</td>
</tr>
<tr>
<td>4L_JT_XE</td>
<td>11,302</td>
<td>5.9</td>
</tr>
<tr>
<td>4L_JT_XR</td>
<td>1,333</td>
<td>1.1</td>
</tr>
<tr>
<td>4L_JT_XRG</td>
<td>3,821</td>
<td>4.1</td>
</tr>
<tr>
<td>4L_JT_XRGC</td>
<td>3,835</td>
<td>4.1</td>
</tr>
<tr>
<td>4L_ST_XR</td>
<td>1,333</td>
<td>1.1</td>
</tr>
<tr>
<td>DESPConv</td>
<td>15</td>
<td>0.8</td>
</tr>
<tr>
<td>DLinear_JT_XR</td>
<td>2</td>
<td>0.5</td>
</tr>
<tr>
<td>DLinear_JT_6D</td>
<td>1</td>
<td>0.5</td>
</tr>
<tr>
<td>DLinear_JT_3P</td>
<td>2</td>
<td>0.5</td>
</tr>
<tr>
<td>DLinear_JT_XE</td>
<td>1</td>
<td>0.5</td>
</tr>
<tr>
<td>DLinear_JT_XRG</td>
<td>2</td>
<td>0.5</td>
</tr>
<tr>
<td>DLinear_JT_XRGC</td>
<td>2</td>
<td>0.5</td>
</tr>
<tr>
<td>Fusion2L_JT_XRG</td>
<td>9,223</td>
<td>10.1</td>
</tr>
<tr>
<td>Fusion2L_JT_XRGC</td>
<td>10,288</td>
<td>17.1</td>
</tr>
<tr>
<td>TRACK_JT_XR</td>
<td>3,259</td>
<td>7.9</td>
</tr>
<tr>
<td>TRACK_JT_XRGC</td>
<td>4,860</td>
<td>11.3</td>
</tr>
</tbody>
</table>

nature of VR, we use minimal data modalities to train the model, completely eschewing image data for reasons discussed earlier in the chapter. Further, we investigate whether the sampling frequency of the eye-tracker has any effect on gaze. In this investigation, however, we have to rely on a non-VR dataset and hence the results can be taken as indicative and not conclusive.

We considered realistic VR gaze data and investigate gaze forecasting for a prediction horizon of 150ms⁴. As discussed in section 4.2 we used OpenNeeds [159], GazeBase [160] and a small scale gaming dataset in-house collected for this investigation. Next in section 4.4.1 we discuss the preprocessing conducted on these data sets and the features used. We describe the developed model in section 4.4.2 and present a summary of results in section 4.4.3.

4.4.1 Preprocessing

The VR gaze data sets we used provide gaze as a 3D gaze vector, with the origin at corresponding eye location, or the so called center eye if a single

⁴At the time this work was conducted in Publication IV, we believed 150ms was a practical m2p latency estimate for remotely rendered VR. Subsequent work has shown that lower latencies are achievable and state of the art work in remote rendering VR [114, 113] consider 100ms as a more representative latency, which we used in Publication III.
value is output for both eyes. XR or VR run time SDKs usually provide gaze information in this form or as a pose value wherein the orientation provides the gaze direction. We instead preprocess gaze into a normalized point in screen space as this is more useful for graphics SDKs and video encoders. We used the viewing geometry of the HMD used for the data sets to project 3D gaze vectors into screen space gaze points. Further, from the corresponding head pose data we derive secondary features, such as gaze velocity on the screen, rotational velocity of the HMD along the shortest geodesic arc between two consecutive orientations and additionally gaze acceleration and HMD angular acceleration. Data is fed to the model as a sequence of samples of a subset of these features. We create the input training sequences as a sliding window over the time series within each trace from the datasets. An essential constraint that we enforced was that the sliding windows did not slide across trace boundaries between data traces collected during different sessions. This constraint prevents sudden artificial drifts between consecutive data samples. More details on data preprocessing used for gaze data can be found in Publication IV.

A heatmap of gaze data in OpenNeeds can be seen in figure 4.5, illustrating centre bias of gaze data [183].

![Figure 4.5. Heatmap of sanitized OpenNEEDs gaze data.](image)

### 4.4.2 Gaze Forecasting Model

Our gaze forecasting model, developed in Publication IV, is based on LSTMs. LSTMs being RNNs have the ability to learn temporal patterns, while also ameliorating the exploding and vanishing gradient problems of regular RNNs, allowing for learning long term dependencies [184]. Based on the fact that the horizontal and vertical components of human gaze in VR seem to have low correlation [183], our model comprises of two separate LSTM based branches, one for each component of the gaze. The model is illustrated in figure 4.6. The model accepts a pre-processed data sequence of 20 time steps, which for OpenNEEDs data is about 200ms of past data. The model uses sum of mean absolute error for each gaze coordinate as the loss function. Coordinates from
Forecasting Pose and Gaze

Figure 4.6. Gaze Prediction Model: $t$ is the current time, $X_t - X_{t-200}$ and $Y_t - Y_{t-200}$ is the gaze $X, Y$ coordinate sequence from current time to approximately 200ms into the past. $V_{Gt} - V_{t-200}$, $V_{Ht} - V_{Ht-200}$, $AG_{t} - AG_{t-200}$ and $AH_{t} - AH_{t-200}$ respectively are the gaze velocity, head rotational velocity, gaze acceleration and head rotational acceleration sequences from now to approximately 200ms into the past. $pH$ is the prediction horizon; taken from Publication IV

Each branch of the model are concatenated into the output. Each branch consists of 4 LSTM layers with a hidden state size of 64. Each branch uses two fully connected layers to reduce to output size to 1.

4.4.3 Summary of Results

We evaluated the model for inference accuracy as well as inference speed. The inference accuracy was calculated at four prediction horizons: 2, 4, 9 and 13 time steps in the OpenNEEDs dataset which correspond to 22.22ms, 44.44 ms, 100ms and 144.44 ms. In addition, we investigated the effect of gaze sampling frequency on inference accuracy using the GazeBase dataset. We report the errors in terms of angular error between the predicted gaze point and the ground truth gaze point, angular error being commonly used in the field of gaze estimation and prediction [134, 183]. For model training and evaluation, we divided the OpenNEEDs dataset into four disjoint subsets: two for training and validation and two for testing. The first test dataset (TDS1) shares some subjects with the training data set while the second test dataset (TDS2) does not share any subjects with the training datasets.

For evaluation, we used the last known gaze as the baseline, as we found it to be more accurate than commonly used baselines like gaze center or gaze mean used in recent works in the field [150, 151]. As a preliminary test, we compared our proposed model with a simple LSTM model, a multi layer perceptron and a state of the art linear regression model. Our results showed that our baseline i.e. last known gaze as the prediction outperformed mean gaze, center gaze as well as the MLP in accuracy. Linear regression and simple LSTM have better accuracy than the baseline, but our proposed model beats both the linear regression and the simple LSTM model. The summary results of the testing on OpenNEEDs dataset are presented in table 4.3.

To visualize the spread of the gaze errors, the CDFs of the errors for TDS1 at different prediction horizons are illustrated in figure 4.7a, showing both the results for baseline case and prediction of our models. From both the summary
Table 4.3. Evaluation results on test datasets. The Mean and Median Errors are in units of degrees

<table>
<thead>
<tr>
<th>Horizon (ms)</th>
<th>Case</th>
<th>Mean Error TDS1</th>
<th>Median Error TDS1</th>
<th>Mean Error TDS2</th>
<th>Median Error TDS2</th>
</tr>
</thead>
<tbody>
<tr>
<td>22.22</td>
<td>prediction</td>
<td>0.74</td>
<td>0.41</td>
<td>0.58</td>
<td>0.35</td>
</tr>
<tr>
<td></td>
<td>baseline</td>
<td>0.82</td>
<td>0.30</td>
<td>0.71</td>
<td>0.25</td>
</tr>
<tr>
<td>44.44</td>
<td>prediction</td>
<td>1.52</td>
<td>0.90</td>
<td>1.21</td>
<td>0.75</td>
</tr>
<tr>
<td></td>
<td>baseline</td>
<td>1.57</td>
<td>0.61</td>
<td>1.37</td>
<td>0.52</td>
</tr>
<tr>
<td>100</td>
<td>prediction</td>
<td>2.65</td>
<td>1.03</td>
<td>2.20</td>
<td>0.75</td>
</tr>
<tr>
<td></td>
<td>baseline</td>
<td>3.17</td>
<td>1.46</td>
<td>2.82</td>
<td>1.24</td>
</tr>
<tr>
<td>144.44</td>
<td>prediction</td>
<td>3.63</td>
<td>1.62</td>
<td>3.14</td>
<td>1.22</td>
</tr>
<tr>
<td></td>
<td>baseline</td>
<td>4.21</td>
<td>2.17</td>
<td>3.79</td>
<td>1.85</td>
</tr>
</tbody>
</table>

results in table 4.3 and the CDFs in figure 4.7a, it can be noted that at small prediction horizons, the proposed model and baseline have similar accuracy and the copy last baseline may even be a better predictor of gaze. However, as the prediction horizon increases, the model performance is better than baseline and it is especially effective in suppressing large errors compared to the copy last baseline as can be seen in 4.7a. Similar results of error distribution were obtained when testing the with TDS2 which contained data from unseen subjects, indicating model generalization.

We also tested the model against an in-house collected VR gaming dataset for a prediction horizon of 100ms to get an idea of its real world performance. The VR gaming dataset was divided into mutually exclusive test and train sets and a model trained on OpenNeeds training data was retrained with the train dataset from our VR gaming dataset. The errors of the baseline prediction and the model prediction are illustrated in 4.7b. It is clear that the model is able to reduce mean, median and interquartile range of the errors for the VR gaming dataset as well, indicating a positive outlook for real world applications. Our analysis of the results also revealed that most of the gaze errors occur when the gaze velocity is close to zero. This indicates that while prediction during a saccade is possible to learn, predicting saccade landing points is more difficult.

Consumer grade gaze trackers currently available for VR HMDs have sampling frequency of up-to 120Hz. Given the high possible velocities of eye movements, it is possible that sampling frequency of the gaze tracker may have an effect on prediction accuracy. To investigate this, we train and test our model with video viewing gaze data from the GazeBase dataset with a sampling frequency of 1000Hz. GazeBase is collected using conventional electronic displays and the head position is fixed using a chin brace, there is no rotational data available. We are able to use only gaze and gaze velocity data for our model. We train and test our model for a prediction horizon of 100ms with three configurations of
Forecasting Pose and Gaze

data: a) sampling frequency = 1000Hz, input sequence length = 200 steps; b) sampling frequency = 1000Hz, input sequence length = 20 steps and c) sampling frequency = 100 Hz, input sequence length = 20 steps. We sub-sampled the 1000Hz original data to create the configuration c.

The results are illustrated in 4.7c. The data configuration closest to consumer grade eye trackers, i.e. with 100Hz sampling frequency and 200ms sequence length seems to perform worse than the other two configurations when considering angular errors below 1°. Considering angular errors above 1.1°, all the data configurations have similar accuracy while having better accuracy performance than the baseline. Somewhat counter-intuitively, case b with the shortest sequence length in terms of time seems to be performing better than cases a and c. Overall, the sampling frequency does not seem to have a drastic effect on accuracy performance of the model.

We also investigated the effect of adding additional features in addition to gaze in the input data sequence in a data ablation study. The results, summarised in table 4.4, show that additional features do indeed improve the accuracy performance. To test if our proposed model can be run in real time, we performed inference time measurements using the PyTorch runtime on a top range and a mid range GPU. We measured median inference time on the top range GPUs RTX 3080 Ti to be 1.6ms and on the mid-range GPU RTX 2060 to be 2.0ms, showing our proposed model can be run in real time for current VR HMDs with refresh rates of upto 120Hz.

Table 4.4. Ablation study at prediction horizon 100ms, G = gaze, GV = velocity, GA = gaze acceleration, HV = HMD velocity and HA = HMD acceleration

<table>
<thead>
<tr>
<th>Data Input</th>
<th>Mean Error</th>
<th>Median Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>3.17</td>
<td>1.46</td>
</tr>
<tr>
<td>G</td>
<td>3.07</td>
<td>1.53</td>
</tr>
<tr>
<td>G + GV</td>
<td>2.99</td>
<td>1.17</td>
</tr>
<tr>
<td>G + GV + HV</td>
<td>2.86</td>
<td>1.62</td>
</tr>
<tr>
<td>G + GV + HV + GA + HA</td>
<td>2.65</td>
<td>1.03</td>
</tr>
</tbody>
</table>

4.5 Conclusion

In this chapter we investigated prediction of human body kinematics relevant to remotely rendered graphics particularly remotely rendered VR. The prediction horizons 100ms in Publication III and 150ms in Publication IV broadly represent the range that is considered achievable and feasible for remotely rendered VR [113, 114].

Accurate pose information is one of the most important parameters for re-
Forecasting Pose and Gaze

motely rendered VR as pose dictates the FOV that is rendered and a mismatch between the rendered FOV and the viewer expected FOV can result in not only lower QoE, but severe VR sickness. We proposed some machine learning based pose prediction approaches in Publication III. We evaluated the approaches against a state of the art classical DESP based pose prediction approach. Our investigations considered multiple aspects for pose prediction based on past pose information and optionally gaze and controller pose information. The aspects considered were: classical vs machine learning based pose prediction; joint and separate prediction of position and orientation components of pose; pose representation; and finally effect of additional data modalities on pose prediction. We also introduce a new three point VR HMD pose representation which we believe is more amenable to machine learning and propose a data fusion LSTM layer for better leveraging gaze and controller pose information for HMD pose prediction. We found, via extensive evaluation that ML based models outperform classical
models and while predicting position and orientation components of pose using separately trained models has the best accuracy, similar accuracy can be achieved with a single model by carefully selecting the data fusion strategy, the ML model and the pose representation. Our proposed models showed promising results in reducing pose prediction errors, especially extreme errors in position prediction which have the potential to impact user QoE drastically and which are harder to correct at a client device [5].

We also investigated gaze prediction in VR in Publication IV. Gaze in VR environments is prima facie difficult to predict without considering visual stimuli because of the various biases, center and horizon biases being the main ones, that human gaze exhibits in VR [183]. As such, gaze appears stationary most of the time and the movements are sporadic and ballistic in nature due to nature of saccadic motion. We proposed a lightweight LSTM based model which eschews visual information (rendered frames or saliency information) in the interest of low computational requirements, which reduces gaze prediction errors by up to 20% compared to the baseline of no prediction. We also showed that using HMD orientation data as an additional data modality (preprocessed into appropriate features) improves gaze prediction accuracy.

The ML based models developed in Publication III and Publication IV and summarized in this chapter were designed to be lightweight to satisfy the low latency constraints of a remotely rendered VR system. This is reflected in the inference time evaluations which show that the proposed models have a sufficiently low inference time on typical server architecture and inference of the next pose point or gaze point can be obtained well within the frame render period.
5. Conclusion

This thesis considers the field of real time interactive applications enabled by remote rendering. Remote rendering as a paradigm, though old, is seeing a renewed interest with the success of public cloud computing and enterprise and consumer interest in graphics intensive applications like high quality video games, VR, MR and XR. Since these graphics intensive applications require substantial rendering and energy resources, not available in typical consumer devices like smartphones, it may be advantageous to off load some or all the rendering operations to well provisioned servers and stream the rendered graphics back to the resource constrained devices, the so called thin clients. However, high downstream bandwidth is needed for the high quality rendered graphics to be streamed to the thin clients and the latency between motion on the client and the corresponding frame update (motion to photon latency) has to be low enough to be imperceptible to the user.

This thesis investigates these two challenges and attempts to propose solutions beyond the obvious ones i.e. provisioning higher link capacity, placement of rendering servers closer to the network edge etc. We explore these challenges in three research questions RQ1, RQ2 and RQ3. RQ1 asks "Is real time foveated encoding feasible for off the shelf games with consumer components and standard encoders?", while RQ2 asks "How can foveation at the rendering stage be used to improve real-time remotely rendered foveated content delivery?". These two questions, which deal with the bandwidth challenge are considered in chapter 3. RQ3 asks "In a remotely rendered VR system, is it possible to forecast user body kinematic properties which affect rendering given previous values of these kinematic properties?". This research question, which deals with the M2P challenge, is addressed in chapter 4.

In chapter 3, the non-uniform acuity of the HVS is exploited by leveraging developments in video encoding and graphics rendering which allow spatially non-uniform quality in an encoded or rendered frame. We propose a method and parameterization for real time foveated video encoding for cloud gaming. We develop a prototype cloud gaming system with the real time foveated video encoding and show that not only is it feasible with currently available consumer grade eye-trackers but also that without modifying underlying codecs’ bitrate
Conclusion

savings of up to 20% are possible using FVE. The FVE is transparent to the user and there is no additional perceivable latency or perceivable drop of QoE for the average user. RQ1 thus has an affirmative answer. The accrued bit rate savings are, however, genre specific and while we present a parameterization scheme for FVE which provides optimistic results, the parameterization would need to be tuned to different game genres and games.

We next implement and evaluate three different methods of producing foveated content in a real time remotely rendered system. The methods implemented are foveated video encoding, foveated rendering and foveated warping. Foveated warping is a contextually novel technique to spatially compress a rendered frame with angular pixel resolution of the uncompressed frame being proportional to the cutoff frequency of the HVS. We show that FR on its own does not accrue any bitrate savings of the encoded frames on its own and for best results FR should be combined with FVE. We further illustrate that while FW has lower quality for the same bitrate when compared to FVE, it may be useful in delivering HVS friendly video to thin clients so that the final render resolution is higher than the maximum decodable resolution by the client. This is enabled by spatially compressing the high resolution frames into encoded frames of a resolution which is decodable by the client. Decode and un-compression operations at the frame yield a frame with a resolution higher than the maximum resolution decodable by the client, while substantially preserving the perceived quality. FW may enable thin clients like average smartphones in the year 2023, which typically have a maximum decoding resolution of up to 2K pixels to receive and display videos which have effective resolutions of up to 4k to 6 K pixels. The answer to RQ2 is thus use case specific, FR on its own does not help in bitrate reduction of the video stream so produced. Given an FR stream, FVE has advantages for both server and bandwidth resources, while maintaining perceived quality. For the same FR stream, FW may have advantages in enabling thin clients constrained in the maximum resolution they can decode to realise HVS guided render resolutions which are higher than the maximum resolution that they can decode.

In chapter 4, we consider the other challenge facing remotely rendered applications, i.e. motion to photon latency. Specifically, we investigate machine learning based methods to combat the latency by forecasting rendering relevant human kinematic properties. The forecasted kinematic properties can be used to preemptively render frames for kinematic properties which are yet to be reported by the client and hence may be able to mask the latency between the client and the remote rendering server. We consider head pose and gaze as the human kinematic properties to be predicted in Publication III and Publication IV respectively. A key decision in design of these methods was eschewing visual data e.g. rendered frames, in the interest of inference speed, which has be faster than frame render periods of the remotely rendered application being considered. Head pose is the key rendering parameter in XR applications as it dictates the camera position and orientation, which is coupled to the HMD.
Any mismatch between the head pose used for rendering and the current head pose (as tracked by the HMD) can result in lowering of user QoE and even VR sickness. Accurate gaze on the other hand is key in gaze based augmentations discussed in chapter 3. A mismatch between the actual gaze and the gaze used for gaze based augmentations may lead to user gaze falling in low quality areas of a frame. Low video quality and switches between low and high video quality have negative impacts on user QoE [88, 185].

We investigated whether ML based pose forecasting has accuracy advantages over a state of the art conventional DESP based predictor. Further, we also investigated the feasibility of forecasting pose as a single quantity i.e. a single vector comprising of the position and orientation components and compared it to forecasting the pose constituent components of position and orientation separately. In the joint pose prediction evaluation, we considered different pose representations and the effect of additional data modalities, specifically gaze and controller pose data on HMD pose forecasting accuracy. We also propose a three point pose representation for HMD pose which we hypothesize is more amenable to joint pose forecasting. Further we propose an LSTM based data fusion layer which fuses HMD pose with gaze and controller pose data in a way which we hypothesize is more suitable for HMD pose forecasting. Our evaluations support our hypothesis regarding the three point pose prediction and data fusion layer with meaningful improvements in pose prediction accuracy accrued when using the proposed three point representation or when using the proposed data fusion layer.

We also propose a light weight gaze predictor for VR applications and investigate whether higher sampling frequency of the gaze data improves prediction accuracy. The proposed lightweight LSTM based gaze predictor outperforms copy last baseline and simple classical predictors like linear regression and simple MLPs and LSTMS. We also show that HMD orientation, as an additional data modality for gaze prediction, improves prediction accuracy. The models proposed for pose and gaze prediction were designed to run with low inference times and our inference time evaluations illustrate that. RQ3 is answered in affirmative, provided attention is paid to feature selection, data preprocessing, and selection of appropriate ML models. In the case of pose forecasting, the representation of pose used and whether the position and orientation components are predicted separately or jointly have an impact on prediction accuracy. In the case of gaze forecasting, how human gaze behaves in VR/XR environments, i.e. gaze biases like center and horizon bias, need to be considered.

In summary, in this thesis we propose and evaluate a set of actionable methods that can be deployed in remotely rendered interactive applications to: a) reduce the downstream bandwidth required for a particular perceived visual quality and b) reduce the perceived motion to photon latency by enabling predictive rendering.
5.1 Limitations

The user study in Publication I, considered only one genre of video games, FPS. Gaze in FPS games is center biased, which may not be the case for other game genres. A more robust approach would have been to include other games in the user study, particularly a game genre with a more spread out gaze pattern, for example, a game where the user has a birds eye perspective of the scene. However, FPS games have very tight latency constraints, especially when user performance is taken into account [186, 187], which can be tighter than other genres, and our user study establishes that even with such tight latency constraints, FVE is feasible and can provide bitrate savings.

The user study in Publication II was limited in number of participants. More participants in the study would have potentially improved the quality of the results. The number of participants was still within the recommended range specified by ITU [104], although on the lower end of the range. Further, the user study design had room for improvement as there was no verification of whether the user followed the instructions on where to fix their gaze. A more robust approach would have been to use a gaze tracker in the user study. However, the results from the user study broadly agree with the objective quality metrics which indicates participant conformance to the test protocol. Further, in case the participants did not conform to the test protocol, the user study opinion scores would represent results for test sequences with worse quality than the ones presented. Considering these mitigating factors, the user study provides a good foundation for further work.

Publication III should have included evaluation of different pose representations on different classical models as well for completeness. However, the work was quite expansive and increasing the scope further would not have been feasible considering resources. Further, classical models are easier to test and evaluation of different pose representations can be done as a separate work. Ideally, the models should have been tested on a data collected while users where experiencing off the shelf VR games. Data scarcity is a general problem in machine learning, especially so in XR which is only now seeing mainstream adoption. The dataset we used attempts to cover a broad range of visual, cognitive and task dynamics and should broadly correspond to these dynamics in off the shelf VR games as well.

For gaze prediction, presented in Publication III, an approach where gaze data is filtered into fixations, saccades and smooth pursuit in the pre-processing stage could have provided some meaningful insights into predictability of various gaze motions. Velocity threshold filters utilized for such categorization often need threshold parameters to be adjusted for each scenario [188] and what constitutes a fixation may also vary. Considering these complexities and the fact that the gaze reported by consumer gaze trackers is often already filtered using proprietary algorithms, forecasting gaze as such provides a valuable foundation for future work.
There are multiple directions in which the work presented in this thesis could be continued. One possible direction is in the domain of foveated video streaming. It would be interesting to understand how QoE is affected by FVE and its parameterization for different game genres, both on flat screens and in the context of XR experiences and developing a computational model for QoE vs FVE parameterization. Another aspect to investigate is dynamic adaptive FVE wherein FVE parameterization is based on network conditions and/or game/scene dynamics to optimize user QoE and channel capacity usage.

In the context of human kinematic attribute prediction, using classical kinematic models of head neck and hands, together with machine learning models could be an enticing direction to explore. It would also be interesting to explicitly decompose kinematic attribute sequences into identifiable patterns, as opposed to the implicit learning of those patterns by ML models. Another aspect would be exploring generative models for kinematic attribute forecasting. For business use cases with pass based access to cloud games/XR experiences, reinforcement learning is also an enticing direction to explore.

An ambitious future work direction is to refine the techniques developed in this thesis and implement them all in a functional prototype to study effects of latency, prediction error, and parameterizations of gaze based augmentations on user QoE. Various influence factors that could be considered in such a study would be XR content, interaction and task dynamics, latency, latency compensation, prediction error and FVE/FR/FW parameterization.

Further challenges in this domain that remain open to investigation are latency mitigation in the client during frame composition prior to display. Image based reprojection techniques which may leverage depth images in addition to color textures have shown promising results. Such techniques have been implemented by different HMD manufacturers with various names like space warp, time warp and asynchronous reprojection. Interesting challenges in this domain include transport and rate control of the depth images considering both available bandwidth and motion to photon latency. Depth images can be difficult to compress with appropriate precision as the range of depth values in a virtual scene can vary over orders of magnitude. Support for depth images is available in standard codecs as extensions and profiles, but optimized hardware acceleration of depth images is not so wide spread. Furthermore, new immersive codecs like MPEG-I: Visual Volumetric Video-based Coding (V3C) [189] are being standardized. Investigating performance of these codecs, in the context of remote rendering and considering HVS and QoE factors should reveal interesting results.

Another challenge, yet to be addressed is investigating interplay of rate control, foveation, image based reprojection, available bandwidth and motion to photon latency, both with conventional 2D codecs and immersive codecs.

Yet another aspect of remote rendering, especially in the context of XR, is
transport protocols and network infrastructure. This is an active field of research and standardization. Some work has analysed network protocols and traffic characteristics of commercial cloud gaming services [190] as have been network resource provisioning and latency aspects like server placement [10]. There have been developments in split rendering [191] and efforts to standardize aspects of not only remote and split rendering data exchange [192] but also support for enabling deployment of such services within wireless networks, both at infrastructure level [193] and protocol level [194]. However, investigations into implementations of remote rendered applications deployed using these paradigms and the end user QoE have not been pursued yet and will be an interesting area of research.
References


[99] FFmpeg Developers. FFmpeg.


References


94


