

DESIGN OF A PRIVACY-AWARE TEMPORAL SUPERSAMPLING ARCHITECTURE

AKANKSHA DIXIT



DEPARTMENT OF ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY DELHI
APRIL 2026

© Indian Institute of Technology Delhi (IITD), New Delhi, 2026

DESIGN OF A PRIVACY-AWARE TEMPORAL SUPERSAMPLING ARCHITECTURE

by

AKANKSHA DIXIT

Department of Electrical Engineering

Submitted

in fulfillment of the requirements of the degree of Doctor of Philosophy

to the



INDIAN INSTITUTE OF TECHNOLOGY DELHI

APRIL 2026

Certificate

This is to certify that the thesis titled **DESIGN OF A PRIVACY-AWARE TEMPORAL SUPER-SAMPLING ARCHITECTURE** being submitted by **Ms. AKANKSHA DIXIT** for the award of **Doctor of Philosophy** in Electrical Engineering is a record of bona fide work carried out by her under my guidance and supervision at the Department of Electrical Engineering, Indian Institute of Technology Delhi. The work presented in this thesis has not been submitted elsewhere, either in part or full, for the award of any other degree or diploma.

Smruti R. Sarangi
Professor
Department of Electrical Engineering
Indian Institute of Technology Delhi
New Delhi- 110016

Acknowledgements

I would like to start with expressing my deepest gratitude to **Prof. Smruti R. Sarangi** for his unwavering guidance and support throughout my PhD. His relentless pursuit of excellence inspired me to push my limits and grow both academically and personally. A crucial lesson I learned from him is the importance of maintaining a big-picture perspective and addressing problems with a long-term vision. This has transformed my approach to problem-solving, enabling me to tackle complex issues more effectively.

I would also like to thank my Student Research Committee: **Prof. Sumantra Dutta Roy**, **Prof. Se-shan Srirangarajan**, and **Prof. Rohan Paul**, for their insightful comments, constructive feedback, and continuous guidance during the course of my research.

I am grateful to my friends at IIT Delhi for many wonderful conversations, both technical and non-technical. I especially appreciate my seniors and friends, Dr. Priyanka Singla and Dr. Nivedita Shrivastava, for their advice on planning research and tackling challenges, as well as their encouragement during difficult times. I want to extend a special thanks to my labmates, who have become dear friends: Shruti, Ani, and Rohith, for always being there and making this journey enjoyable and memorable.

I am deeply grateful to my parents for their unconditional love, constant support, and unwavering belief in me. Their encouragement allowed me to devote myself fully to this research. They are the pillars of my life and have always reminded me that with hard work, anything is possible and that I should focus on the effort rather than the outcome. Above all, I thank GOD for all and everything.

I also acknowledge the use of OpenAI's ChatGPT to assist with language refinement in this thesis.

Akanksha Dixit

Abstract

Graphics applications have become an integral part of diverse domains, ranging from entertainment and education to healthcare and scientific visualization. These applications take various forms. They can be immersive such as virtual reality (VR) and augmented reality (AR), or non-immersive, such as traditional desktop-based applications. They can be deployed across multiple platforms such as desktops, mobiles and headsets. Each form and platform presents unique performance constraints, yet a universal expectation remains: delivering a high Quality of Experience (QoE) to the users. Meeting this expectation requires addressing multiple challenges: ensuring low latency and high frame rendering rates, achieving high visual quality, maintaining realism, and safeguarding user security and privacy. This thesis takes a holistic approach to meeting these requirements while operating within the constraints of the underlying hardware resources.

We begin with the problem of high latency in VR systems, specifically the motion-to-photon delay (MPD), which is the time between a user’s head movement and the corresponding update on the display. In VR, even slight delays can disrupt the user’s sense of presence, making the MPD more critical. A common MPD reduction technique is Asynchronous Time Warping (ATW), which reprojects the already rendered frame based on the most recent head position. This helps display the frame corresponding to the latest possible head position, reducing the MPD. However, the challenge is to invoke ATW at the right instance within the refresh cycle; otherwise, it will increase the MPD. Fixing a time for invoking ATW does not work because ATW shares resources with the rendering pipeline, leading to unpredictable latency due to preemption and context switches. Hence, we propose, *PredATW*, a predictor that monitors the status of hardware resources and the rendering process to accurately estimate ATW latency, thereby reducing MPD and minimizing missed deadlines. Our predictor has an error of only 0.22 ms across several popular VR applications. Moreover, as compared to the baseline architecture, we reduce deadline misses by 80.6%.

While *PredATW* reduces the MPD by correctly invoking ATW, it is important to note that ATW has its own limitations. Specifically, it only corrects for head rotation, not translation, and primarily enhances perceived responsiveness by utilizing the most recent position data. However, it does not increase the frame rate. This makes ATW effective at relatively modest refresh rates spanning from 60 to 90 Hz, which are typical for VR headsets, but it falls short for modern monitors with high refresh rates (240 to 360 Hz) that require higher frame rates. To increase the frame rate, techniques such as *frame generation* or *temporal supersampling* have been proposed, which predict intermediate frames from already rendered frames.

Existing temporal supersampling methods are *interpolation* and *extrapolation*. Interpolation, which uses both past and future frames, offers higher quality but is best for doubling frame rates, as further upsampling adds latency and complexity. Extrapolation relies only on past frames, enabling continuous frame generation with lower latency, making it more suitable for real-time applications. In this work, we target a $4\times$ frame rate increase (from 60 Hz to 240 Hz), making extrapolation the preferred choice. However, no single existing extrapolation algorithm performs consistently well across all scene types. Each one is tailored to specific scenarios and exhibits strengths and weaknesses depending on the context. Hence, we propose *X4Rate*, a frame generation framework that combines multiple existing methods to deliver results far superior to any individual algorithm by selecting the best algorithm for the current scene. This adaptive selection ensures high visual quality consistently because when one method underperforms, others compensate. *X4Rate* achieves $4\times$ the rendering rate while delivering high-quality frames, outperforming the nearest competitor, ExtraNet, with a 22.13% improvement in the PSNR.

Building on this, we address the inherent limitation of relying solely on existing extrapolation methods by designing our own. We propose a novel lightweight frame extrapolation algorithm that exploits the perceptual sensitivity to segment each frame into foreground and background regions and employs a novel neural network to generate the final extrapolated frame. Additionally, a wavelet transform (WT)-based filter pruning technique is applied to compress the network, significantly reducing the runtime of the extrapolation process. Our results demonstrate that *PatchEX* achieves a 61.32% and 49.21% improvement in PSNR over the latest extrapolation methods ExtraNet and ExtraSS, respectively, while being $3\times$ and $2.6\times$ faster (resp.).

Finally, we address the data security and privacy challenges in AR systems. By blending the real and virtual worlds, AR devices can inadvertently expose sensitive visual data—ranging from bystanders’ identities to confidential physical documents. Existing privacy-preserving methods are often too computationally heavy for real-time execution on resource-constrained AR headsets. To solve this, we introduce *ARGuard*, a lightweight, real-time framework that integrates

a saliency-aware object detection network with a novel irreversible obfuscation algorithm. *AR-Guard* operates at 60 fps on Microsoft HoloLens 2, detecting and protecting sensitive regions with 12.6% and 15.8% higher accuracy in detection and tracking, respectively, and achieving a 60.23% reduction in privacy leakage compared to the nearest competing method, *BystandAR*.

To summarize, this thesis presents a unified framework for addressing both performance and privacy in graphics applications. Through innovations in latency reduction, high-quality extrapolation, and real-time privacy preservation, our work advances the state-of-the-art in delivering secure, smooth, and immersive experiences on consumer-grade hardware.

संक्षेप

मनोरंजन और शिक्षा से लेकर स्वास्थ्य सेवा और वैज्ञानिक विज्ञान तक, ग्राफिक्स अनुप्रयोग विविध क्षेत्रों का एक अभिन्न अंग बन गए हैं। ये अनुप्रयोग विभिन्न रूप धारण करते हैं, जिन्हें इमर्सिव के रूप में वर्गीकृत किया जाता है, जैसे वर्चुअल रियलिटी (वी आर), ऑगमेंटेड रियलिटी (AR), या गैर-इमर्सिव, जैसे पारंपरिक डेस्कटॉप-आधारित अनुप्रयोग। ये अनुप्रयोग डेस्कटॉप, मोबाइल और हेडसेट सहित कई प्लेटफॉर्म पर तैनात किए जाते हैं। प्रत्येक रूप और प्लेटफॉर्म अद्वितीय प्रदर्शन बाधाएँ प्रस्तुत करता है, फिर भी एक सार्वभौमिक अपेक्षा बनी रहती है: उपयोगकर्ताओं को उच्च अनुभव गुणवत्ता (QoE) प्रदान करना। इस अपेक्षा को पूरा करने के लिए कई चुनौतियों का समाधान करना आवश्यक है: कम विलंबता और उच्च फ्रेम रेंडरिंग दर सुनिश्चित करना, उच्च दृश्य गुणवत्ता प्राप्त करना, यथार्थवाद बनाए रखना और उपयोगकर्ता सुरक्षा और गोपनीयता की रक्षा करना। यह शोध प्रबंध अंतर्निहित हार्डवेयर संसाधनों की सीमाओं के भीतर काम करते हुए इन आवश्यकताओं को पूरा करने के लिए एक समग्र दृष्टिकोण अपनाता है।

हम वी आर सिस्टम में उच्च विलंबता की समस्या से शुरुआत करते हैं, विशेष रूप से मोशन-टू-फोटॉन विलंब (एम-पी-डी), जो उपयोगकर्ता के सिर की गति और डिस्प्ले पर संबंधित अपडेट के बीच का समय है। वी आर में, थोड़ी सी भी देरी उपयोगकर्ता की उपस्थिति की भावना को बाधित कर सकती है, जिससे एम-पी-डी और भी महत्वपूर्ण हो जाता है। एम-पी-डी कम करने की एक सामान्य तकनीक एसिंक्रोनस टाइम वॉर्पिंग (ए-टी-डब्ल्यू) है, जो पहले से रेंडर किए गए फ्रेम को सबसे हालिया हेड पोजीशन के आधार पर रीप्रोजेक्ट करती है। इससे एम-पी-डी को कम करते हुए, नवीनतम संभावित हेड पोजीशन के अनुरूप फ्रेम को प्रदर्शित करने में मदद मिलती है। हालाँकि, चुनौती रिफ्रेश चक्र के भीतर सही समय पर ए-टी-डब्ल्यू को ट्रिगर करने में है; अन्यथा, यह अभी भी एम-पी-डी को बढ़ाता है और सबसे खराब स्थिति में रिफ्रेश की समय सीमा चूक सकता है। निश्चित समय निर्धारण विश्वसनीय रूप से काम नहीं करता क्योंकि ए-टी-डब्ल्यू रेंडरिंग पाइपलाइन के साथ संसाधन साझा करता है, जिससे प्रीएम्पशन और संदर्भ स्विच के कारण अप्रत्याशित देरी होती है। इसलिए, हम Pred ए-टी-डब्ल्यू, एक ऐसा प्रेडिक्टर प्रस्तावित करते हैं जो हार्डवेयर संसाधन की स्थिति और रेंडरिंग कार्यभार की निगरानी करके ए-टी-डब्ल्यू विलंबता का सटीक अनुमान लगाता है, जिससे एम-पी-डी कम होता है और छूटी हुई समय-सीमाएँ न्यूनतम होती हैं। हमारे प्रेडिक्टर में कई लोकप्रिय वी आर अनुप्रयोगों में केवल 0.22 ms की त्रुटि है। साथ ही, बेसलाइन आर्किटेक्चर की तुलना में, हम समय-सीमा चूक को 80.6% तक कम करते हैं।

जहाँ वी आर हेडसेट आमतौर पर अपेक्षाकृत कम रिफ्रेश दरों पर काम करते हैं, वहीं आधुनिक मॉनिटर अब 360 हर्ट्ज़ तक पहुँच जाते हैं। हालाँकि ये डिस्प्ले ज्यादा स्मूथ मोशन और कम आर्टिफैक्ट्स जैसे कि जडर और मोशन ब्लर का वादा करते हैं, लेकिन अगर जी-पी-यू रिफ्रेश दर के बराबर तेज़ी से फ्रेम रेंडर नहीं कर पाता है, तो इनके फ़ायदे सीमित हो जाते हैं और यहाँ तक कि नवीनतम जी-पी-यू भी इतनी तेज़ रेंडरिंग गति को लगातार बनाए रखने में संघर्ष करते हैं। जब रेंडरिंग और रिफ्रेश दरें बेमेल होती हैं, तो उपयोगकर्ताओं को स्क्रीन फटना और रुक-रुक कर चलने का सामना करना पड़ सकता है। इस समस्या के समाधान के लिए, फ्रेम जनरेशन या टेम्पोरल सुपरसैमप्लिंग जैसी तकनीकें प्रस्तावित की गई हैं, जो पहले से रेंडर किए गए फ्रेमों से मध्यवर्ती फ्रेमों का अनुमान लगाती हैं, जिससे प्रत्येक फ्रेम के लिए पूरी रेंडरिंग पाइपलाइन को निष्पादित किए बिना अनुमानित फ्रेम दर बढ़ जाती है। मौजूदा टेम्पोरल

सुपरसैंपलिंग विधियाँ इंटरपोलेशन और एक्सट्रपोलेशन हैं। इंटरपोलेशन, जो पिछले और भविष्य दोनों फ्रेमों का उपयोग करता है, उच्च गुणवत्ता प्रदान करता है लेकिन फ्रेम दरों को दोगुना करने के लिए सबसे अच्छा है, क्योंकि आगे अपसैंपलिंग से विलंबता और जटिलता बढ़ जाती है। एक्सट्रपोलेशन केवल पिछले फ्रेमों पर निर्भर करता है, जो कम विलंबता के साथ निरंतर फ्रेम निर्माण को सक्षम बनाता है, जिससे यह वास्तविक समय के अनुप्रयोगों के लिए अधिक उपयुक्त हो जाता है। इस कार्य में, हमारा लक्ष्य 4x फ्रेम दर वृद्धि (60 हर्ट्ज़ से 240 हर्ट्ज़ तक) है, जिससे एक्सट्रपोलेशन पसंदीदा विकल्प बन जाता है। हालाँकि, कोई भी मौजूदा एक्सट्रपोलेशन एल्गोरिथम सभी प्रकार के दृश्यों में लगातार अच्छा प्रदर्शन नहीं करता है। प्रत्येक एल्गोरिथम विशिष्ट परिदृश्यों के लिए तैयार किया गया है और संदर्भ के आधार पर अपनी शक्तियाँ और कमज़ोरियाँ प्रदर्शित करता है। इसलिए, हम X4Rate का प्रस्ताव करते हैं, जो एक फ्रेम निर्माण ढाँचा है जो वर्तमान दृश्य के लिए सर्वोत्तम एल्गोरिथम का चयन करके किसी भी व्यक्तिगत एल्गोरिथम से बेहतर परिणाम देने के लिए कई मौजूदा विधियों को जोड़ता है। यह मूल रूप से एक निर्णय पूर्वसूचक (मेटा-लर्नर) है जो रनटाइम पर गतिशील रूप से सर्वश्रेष्ठ प्रदर्शन करने वाले फ्रेम एक्सट्रपोलेशन एल्गोरिथम का चयन करता है। यह अनुकूली चयन लगातार उच्च दृश्य गुणवत्ता सुनिश्चित करता है क्योंकि जब एक विधि कम प्रदर्शन करती है, तो अन्य उसकी भरपाई कर देते हैं। X4Rate उच्च-गुणवत्ता वाले फ्रेम प्रदान करते हुए 4 गुना रेंडरिंग दर प्राप्त करता है, जो निकटतम प्रतिद्वंद्वी, एक्सट्रानेट से बेहतर प्रदर्शन करता है, और पी एस एन आर में 22.13% सुधार करता है।

इसी आधार पर, हम अपनी स्वयं की एक्सट्रपोलेशन विधियाँ डिज़ाइन करके केवल मौजूदा एक्सट्रपोलेशन विधियों पर निर्भर रहने की अंतर्निहित सीमा का समाधान करते हैं। हम एक नवीन हल्के फ्रेम एक्सट्रपोलेशन एल्गोरिथम प्रस्तावित करते हैं जो प्रत्येक फ्रेम को अग्रभूमि और पृष्ठभूमि क्षेत्रों में विभाजित करने के लिए अवधारणात्मक संवेदनशीलता का उपयोग करता है और अंतिम एक्सट्रपोलेशन फ्रेम उत्पन्न करने के लिए एक नवीन तंत्रिका नेटवर्क का उपयोग करता है। इसके अतिरिक्त, नेटवर्क को संपीड़ित करने के लिए एक वेवलेट ट्रांसफॉर्म-आधारित फ़िल्टर प्रूनिंग तकनीक लागू की जाती है, जिससे एक्सट्रपोलेशन प्रक्रिया का रनटाइम काफी कम हो जाता है। हमारे परिणाम दर्शाते हैं कि PatchEX नवीनतम एक्सट्रपोलेशन विधियों एक्सट्रानेट और एक्सट्राएसएस की तुलना में क्रमशः 61.32% और 49.21% पी एस एन आर में सुधार प्राप्त करता है, जबकि क्रमशः 3 गुना और 2.6 गुना तेज़ है।

अंत में, हम AR प्रणालियों में डेटा सुरक्षा और गोपनीयता चुनौतियों का समाधान करते हैं। वास्तविक और आभासी दुनिया को मिलाकर, AR उपकरण अनजाने में संवेदनशील दृश्य डेटा को उजागर कर सकते हैं—जिसमें दर्शकों की पहचान से लेकर गोपनीय भौतिक दस्तावेज़ तक शामिल हैं। मौजूदा गोपनीयता-संरक्षण विधियाँ अक्सर संसाधन-सीमित AR हेडसेट पर वास्तविक समय में निष्पादन के लिए कम्प्यूटेशनल रूप से बहुत भारी होती हैं। इसे हल करने के लिए, हम ARGuard प्रस्तुत करते हैं, जो एक हल्का, वास्तविक समय ढाँचा है जो एक प्रमुखता-जागरूक वस्तु पहचान नेटवर्क को एक नए अपरिवर्तनीय अस्पष्टीकरण एल्गोरिथम के साथ एकीकृत करता है। ARGuard माइक्रोसॉफ़्ट होलोलैन्स 2 पर 60 fps पर संचालित होता है, संवेदनशील क्षेत्रों का पता लगाता और उनकी सुरक्षा करता है। क्रमशः 12.6% और 15.8% अधिक सटीकता के साथ, पहचान और ट्रैकिंग में, और सबसे अच्छी मौजूदा विधि, BystandAR की तुलना में गोपनीयता रिसाव में 60.23% की कमी प्राप्त करता है।

संक्षेप में, यह शोध-प्रबंध ग्राफ़िक्स अनुप्रयोगों में प्रदर्शन और गोपनीयता दोनों को संबोधित करने वाला एक एकीकृत ढाँचा प्रस्तुत करता है। विलंबता में कमी, उच्च-गुणवत्ता वाले एक्सट्रपोलेशन, और

वास्तविक समय गोपनीयता संरक्षण में नवाचारों के माध्यम से, हमारा कार्य उपभोक्ता-स्तरीय हार्डवेयर पर सुरक्षित, सुचारु और इमर्सिव अनुभव प्रदान करने में अत्याधुनिक तकनीक को आगे बढ़ाता है।

Contents

Certificate	i
Acknowledgements	iii
Abstract	v
1 Introduction	1
2 <i>PredATW</i>: Reducing MPD in VR	7
2.1 Introduction	7
2.2 Background	11
2.2.1 The VR-Architecture Loop	11
2.2.2 Inter-Frame Similarity	11
2.2.3 ML-based Prediction Models	12
2.3 Motivation	14
2.3.1 Overview of the Benchmarks	14
2.3.2 Methodology	15
2.3.3 Experimental Setup	17
2.3.4 Latency of GPU-Accelerated Asynchronous Time Warp	17
2.3.5 Inter-Frame Similarity	20
2.3.6 Resolution of the Human Eye and Low MPD	22
2.4 Implementation	24
2.4.1 Simulation Environment	24
2.4.2 Identifying the Features	24
2.4.3 Prediction Model	26
2.4.4 Workflow of <i>PredATW</i>	29
2.4.5 Hardware Accelerator	30
2.5 Results and Analysis	31
2.5.1 Performance of Various Predictors	31
2.5.2 Effect of the Predictor on the MPD	32

2.5.3	Sensitivity Analysis	35
2.5.4	Explanability of the Predictor	35
2.5.5	Effect of Underlying Hardware on the Predictor Performance	37
2.5.6	Overheads of <i>PredATW</i>	39
2.5.7	Comparison with the State-of-the-Art	40
2.5.8	Solution Space Exploration (CPU, GPU, Accelerator)	41
2.6	Related Work	44
2.7	Conclusion	45
3	<i>X4Rate</i>: 4× Frame Rate Upsampling	47
3.1	Introduction	47
3.2	Background and Related Work	49
3.2.1	Interpolation Vs Extrapolation	49
3.2.2	Meta Learning (MtL) and the Multi-Arm Bandit (MAB) Problem	50
3.2.3	State-of-the-art Frame Extrapolation Algorithms	50
3.3	Characterization	51
3.3.1	Overview of the Benchmarks	52
3.3.2	Performance Trade-offs in Extrapolation Algorithms	52
3.4	Methodology and Implementation	55
3.4.1	Problem Formulation	55
3.4.2	Meta-Features	55
3.4.3	Meta-Learner Architecture	57
3.5	Results and Analysis	59
3.5.1	Comparison with State-of-the-Art Extrapolation Algorithms	59
3.5.2	Breakdown of the Predictions	60
3.5.3	Runtime Performance	61
3.5.4	Ablation Study	61
3.5.5	Performance Comparison Against Other Meta-Learners	61
3.5.6	Comparison with State-of-the-Art Interpolation Algorithms	63
3.5.7	Frame Rate (FPS)	64
3.6	Conclusion	65
4	<i>PatchEX</i>: High-Quality Real-Time Frame Extrapolation	67
4.1	Introduction	67
4.2	Background and Related Work	71

4.2.1	Real-Time Frame Generation	71
4.2.2	Image Warping	74
4.2.3	Foreground Bias Effect in Human Vision	75
4.3	Characterization and Motivation	75
4.3.1	Overview of the Datasets	76
4.3.2	Foreground-Background Segmentation	77
4.3.3	Challenges in Frame Extrapolation	78
4.4	Methodology	79
4.5	Implementation	80
4.5.1	Mask Generation: Segmentation and Disocclusion	81
4.5.2	Inpainting and Shading Correction Network	82
4.5.3	Neural Network Compression	86
4.6	Results and Analysis	88
4.6.1	Comparison with Frame Extrapolation Methods	88
4.6.2	Ablation Study	91
4.6.3	Performance Comparison with Frame Interpolation Methods	92
4.6.4	Runtime Performance of <i>PatchEX</i>	95
4.6.5	Comparison with VR-style Extrapolation Methods	96
4.6.6	Comparison with Commercial Extrapolation Methods	97
4.6.7	Performance Analysis for High-Resolution Frames	98
4.6.8	Sensitivity Analysis	99
4.6.9	Failure Cases	99
4.7	Conclusion and Future Work	100
5	ARGuard: Real-time Privacy in AR	103
5.1	Introduction	103
5.2	Background and Related Work	105
5.2.1	Attacks on AR Systems	105
5.2.2	State-of-the-art Privacy Preservation Methods	106
5.2.3	Identifying Private/Sensitive Information	108
5.2.4	Obscuration/Obfuscation Techniques	109
5.3	Methodology	109
5.4	Overview	111
5.5	Implementation	112

5.5.1	Saliency-aware Object Detection	112
5.5.2	Object Tracking and Obfuscation (Joint Network)	115
5.5.3	Algorithmic Workflow of <i>ARGuard</i>	119
5.6	Results and Analysis	119
5.6.1	Overview of the Dataset	120
5.6.2	Performance Comparison with State-of-the-art	121
5.6.3	Runtime Performance	123
5.6.4	Privacy Preservation Effectiveness	125
5.7	Conclusion and Future Work	126
6	Conclusion and Future Work	127
6.1	Contributions	127
6.2	Future Work	128
	Bibliography	131
	Appendix	151
A	<i>PredATW</i>	151
A.1	MPD Variation in Similar Frames	151
A.1.1	Impact of Extraneous Factors:	151
A.1.2	Butterfly Effect	153
A.1.3	Comparison with Real-world Data	154
A.2	Correlation of HPCs with the MPD	156
A.3	Effect of the PID Controller on Prediction Errors	157
A.4	Importance of MPD	158
B	<i>PatchEX</i>	159
B.1	Performance of Various Warping Methods	159
B.2	Dataset	159
B.3	LBP Computation	161
B.4	Variation in the Frame Rate	161
B.5	Deformable Convolution	163
B.6	Discussion with the Concurrent Work	164
	List of Publications	165
	Biography	167

List of Figures

1.1	An overview of the work done	4
2.1	Possible scenarios in the GPU-accelerated ATW algorithm. $F1$, $F2$, and $F3$ are frames. $A1$, $A2$, and $A3$ are ATW tasks. $P2$, and $P3$ are preemption latencies. RD stands for the refresh deadline.	9
2.2	Overview of the VR-architecture Loop	11
2.3	The ATW latency	19
2.4	Variation in the ATW latency (Standalone mode)	20
2.5	Macroblock similarity percentage between two consecutive frames (X)	21
2.6	Percentage variation from the last rendered frame's value (denoted by X)	23
2.7	Time-series plot of various hardware performance counters	27
2.8	Design of the proposed predictor. $\langle f \rangle$, $E(t)$, and e_t are features, the controlled error, and the prediction error at time t , respectively.	29
2.9	System Architecture	30
2.10	MAE of the different prediction models	32
2.11	MAE of the different prediction models with the PID component	32
2.12	Improvement in MAE of decision tree after applying the PID control	33
2.13	Effect of the predictor on the refresh deadline miss-rate	34
2.14	Effect of various feature combinations on the prediction error	35
2.15	Percentage of the test data points containing a feature in their decision path	36
2.16	Frequency of each feature in the decision path for different test points	37
2.17	Variation in the ATW latency for various platforms	38
2.18	Performance of PredATW under different design scenarios	40
2.19	MPD of various state-of-the-art solutions	41
3.1	Performance of various extrapolation algorithms based on the scene attributes	53
3.2	Effect of various feature combinations on the prediction accuracy	62
3.3	Network architecture of the RL model	63

3.4	Performance comparison of various meta-learners	64
4.1	The solution space for frame generation at 720p. Each solution is run on an NVIDIA RTX 4080 GPU. The detailed system configuration is shown in Table 4.4.	69
4.2	Interpolation and extrapolation explained. F_i is the rendered frame. R and D represent the rendering time and refresh latency, respectively.	72
4.3	Example views from a few sample scenes	77
4.4	Disoccluded regions in a frame	79
4.5	Dynamic changes in the appearance of shadows between two successive frames, F_t and F_{t+1}	80
4.6	Sample of G-buffers	81
4.7	a Frame F_t ; b Scene depth; c Motion vector; d Foreground mask; e Disocclusion mask.	82
4.8	The neural network architecture for fixing the invalid pixels.	82
4.9	Illustration of the sampling locations using 3×3 standard and deformable convolutions. Left figure: regular sampling grid (blue points) of standard convolution. Right figure: deformed sampling locations (dark red points) with augmented offsets (black arrows) in deformable convolution.	84
4.10	Visual comparison of the frame extrapolation methods: ExtraNet [1] and ExtraSS [2]	90
4.11	Visual comparisons against two frame interpolation methods: EMA-VFI [3] and Softmax-splatting (SS) [4]	93
4.12	Quality comparison with the VR-style extrapolation method ASW and NVIDIA DLSS 3	97
4.13	Performance on high-resolution frames	98
4.14	Failure cases for our method.	100
5.1	A read attack in an AR system	106
5.2	Problems with the two possible approaches	108
5.3	Performance of various obscuration methods. <i>Image adapted from ICDAR 2015 [5]</i>	110
5.4	End-to-end workflow of <i>ARGuard</i> for real-time privacy protection	112
5.5	The neural network architecture for detecting sensitive regions.	114
5.6	The neural network architecture for tracking and inpainting.	117

5.7	Visual comparison of state-of-the-art privacy preservation methods in AR. Bystanders' privacy in the ground truth images is safeguarded by masking their faces to make them indiscernible.	124
A.1	Time-series plot of various hardware performance counters	153
A.2	Illustration of the butterfly effect	154
A.3	Illustration of the butterfly effect	154
A.4	Normalized variance in the ATW latency (note the similarity between simulated and real-world data)	155
A.5	The atw latency across similar frames captured from Oculus Quest 2	155
A.6	Time-series plot of various hardware performance counters for real-world data (normalized values)	156
A.7	Correlation of the identified micro-architectural parameters with the MPD	156
A.8	Histogram of the prediction errors without the PID controller	157
A.9	Histogram of the prediction errors with the PID controller	158
B.1	Comparison of various warping techniques	160
B.2	Variation in the total rendering time	162
B.3	Top 10 high-latency steps in the rendering process	162

List of Tables

2.1	VR benchmarks	15
2.2	Platform Configuration	18
2.3	Frame-specific attributes	22
2.4	List of features	27
2.5	The average duration between ATW completion and the display refresh point (milliseconds) when a deadline is not missed.	34
2.6	Platform Configuration	37
2.7	Overheads of the hardware accelerator	40
2.8	All possible solutions for designing an ATW latency predictor. R, A, and P stand for rendering, ATW and prediction, respectively.	43
2.9	A comparison of related work	45
3.1	A comparison of related work	51
3.2	List of UE scenes used for all the experiments. The input resolution for all the scenes is 720p.	52
3.3	Platform configuration	52
3.4	Runtime (ms) of state-of-the-art extrapolation methods at 720p resolution.	54
3.5	Possible scenarios based on the predicted algorithm at time instant t	55
3.6	Features defining the scenes' characteristics	56
3.7	Quantitative comparison of various state-of-the-art extrapolation methods against <i>X4Rate</i> in terms of PSNR (dB) and SSIM. Throughout this chapter, the best and second-best results of each test setting are highlighted in bold red and <u>underlined blue</u> , respectively.	60
3.8	Breakup of the predictions made by the predictor (in %)	60
3.9	Average latency (in <i>ms</i>) of <i>X4Rate</i> at 720p resolution.	61
3.10	Quantitative comparison of various interpolation methods against <i>X4Rate</i> in terms of PSNR (dB) and SSIM.	64
3.11	Final FPS achieved by <i>X4Rate</i>	65

4.1	Conditions and resulting presentation latency for interpolation and extrapolation techniques with associated latency constraints.	72
4.2	A comparison of related work	73
4.3	Graphics benchmarks	76
4.4	Platform Configuration	76
4.5	Statistics of the training and testing dataset	86
4.6	Quantitative comparison of various extrapolation methods against <i>PatchEX</i> in terms of PSNR (dB), SSIM, LPIPS and VMAF. Throughout this chapter, the best and second-best results of each test setting are highlighted in bold red and underlined blue, respectively.	89
4.7	Region-wise performance comparison of various extrapolation methods against <i>PatchEX</i> in terms of PSNR (dB).	91
4.8	Quantitative comparison of various variants of <i>PatchEX</i> in terms of PSNR (dB) and SSIM. <i>w/o FS</i> refers to without foreground-background segmentation. <i>w/o DC</i> refers to the case where the deformable convolution layer is not used. <i>w/o \mathcal{L}_p</i> refers to without perceptual loss taken into account.	92
4.9	Quantitative comparison of various interpolation methods against <i>PatchEX</i> in terms of PSNR (dB), SSIM and VMAF.	94
4.10	Runtime (ms) breakdown of <i>PatchEX</i> at various resolution levels	95
4.11	Runtime (ms) comparison with the state-of-the-art methods at various resolution levels	96
4.12	Layer-wise comparison of parameters, inference time, and FLOPs before and after pruning at 720p resolution.	96
4.13	Performance of <i>PatchEX</i> in terms of average PSNR (dB), SSIM, and LPIPS at various resolution levels.	98
5.1	A comparison of related work	108
5.2	Description of symbols used in the loss function components.	113
5.3	Platform Configuration	120
5.4	AR Dataset Composed of Corner-Case Scenarios	121
5.5	Performance comparison of various state-of-the-art methods with <i>ARGuard</i> in terms of F1 score, IoU, and mAP for the sensitive region detection. Throughout this chapter, the best and second-best results of each test setting are highlighted in bold red and <u>underlined blue</u> , respectively.	122

5.6	Performance comparison of various state-of-the-art methods with <i>ARGuard</i> in terms of F1 score, IoU, and mAP for tracking the sensitive regions.	122
5.7	Quantitative comparison of various inpainting methods against <i>ARGuard</i> in terms of PSNR (dB) and SSIM.	123
5.8	Runtime (ms) comparison with the state-of-the-art methods at various resolution levels	125
5.9	Performance comparison of various obfuscation methods against <i>ARGuard</i> in terms of privacy leakage.	126