國立清華大學電機資訊學院資訊系統與應用研究所
碩士論文

Institute of Information Systems and Applications
College of Electrical Engineering and Computer Science
National Tsing Hua University
Master Thesis

在不損失虛擬實境畫面品質的前提下減輕頭戴式顯示器使用者的隱私威脅

Mitigating Privacy Threats of HMD Users Without Degrading the Visual Quality of VR Applications

魏宥絲
Yu-Szu Wei

學號：110065527
Student ID:110065527

指導教授：徐正炘 博士
Advisor: Cheng-Hsin Hsu, Ph.D.

中華民國 112 年 8 月
August, 2023

清華大學
TSING HUA UNIVERSITY
自強不息 厚德載物

中文摘要

虛擬實境的應用程式時常需要追蹤使用者的資料以提供沉浸式的體驗， 而這些資料中，使用者的軌跡為最基本的追蹤資料。 然而，許多研究指出，這些軌跡會使使用者暴露在危險之中，比如說，洩漏他們的身分。 因為現有的資料集沒有包含隱私敏感資料來研究虛擬實境的隱私議題 我們首先蒐集了一個六自由度的虛擬實境資料集以研究這些議題。 我們接著使用此資料集做了相關的研究， 研究的結果激發了我們的隱私保護機制的設計。 雖然已經有許多現有的虛擬實境隱私保護機制， 但這些機制不是沒有保護使用者軌跡，就是沒有考慮軌跡的時間相關性。 因此，我們設計了「干擾器」，在考慮時域與空間域的情況下，即時的加噪音在軌跡上。 除此之外，我們還提出了「補償器」以補償因為加了噪音的軌跡而位移的場景， 進而提高視覺品質。 我們的實驗結果顯示了我們解決方案的優點: (i) 與最先進的方法相比，我們的干擾器減少了最多0.4 的重新識別率， (ii) 在相同的隱私設置下，我們的干擾器比最先進的方法進步了 2.5 dB 的 PSNR， 0.1 的 SSIM，以及 10 的 VMAF， (iii)我們的補償器進一步將視覺品質提高最多6.83 dB(PSNR)、0.45(SSIM)和34.57(VMAF)。

Abstract

Virtual Reality (VR) applications usually require users' tracking data to provide an immersive experience. Among the tracking data, trajectory is the most basic in VR. However, several prior works have shown that the trajectories may threaten the users, for example, by revealing their identities. To study the privacy issues in VR, we first collected a 6DoF VR dataset since existing datasets lack the privacy-sensitive data types to do so. We then conducted prior tests with our dataset, which inspired the design of our privacy-preserving approach. Though there are some existing privacy-preserving approaches to protect VR users from privacy threats, these approaches neither consider VR users' trajectories nor perturb the trajectory considering the temporal correlation of the trajectory. Hence, we designed our disturber, perturbing the trajectory in both temporal and spatial domains on-the-fly. Moreover, we propose a compensator for the shifted scene after perturbation to improve the degraded visual quality. Our evaluation results show the merits of our solution: (i) our disturber alone reduces at most 0.4 re-identification rate compared to the state-of-the-art approach, (ii) our disturber alone outperforms the state-of-the-art approach by 2.5 dB in PSNR, 0.1 in SSIM, and 10 in VMAF under the same privacy settings, and (iii) our compensator further improves the visual quality by at most 6.83 dB in PSNR, 0.45 in SSIM, and 34.57 in VMAF.

致謝

在研究所的兩年間有許多我想要感謝的人。 首先，我想要感謝我的指導教授，徐正炘教授。 他總是很樂於與我討論研究上遇到的難題，並盡可能的給予我幫助。 當我犯錯時，他也不會過於苛責，只要求我們盡快的將錯誤改正，並從錯誤中學習。 他也很積極的讓我們與其他國內外學者們交流，使我們的視野更加開闊。 在徐教授的實驗室裡我學到了很多，除了資訊工程相關的知識與技術和統整規劃能力，更多的是處理事情的態度與責任心及毅力。 接著我想感謝陪伴我兩年多的實驗室夥伴們， 尤其是唐盛銘、鄭至雅及石桂華三位好友從碩一以來的陪伴， 讓我在做研究的路上不會孤單。 我還想感謝學長洪梓寬在我碩一時給了我許多研究上的建議與幫助。 另外，特別感謝給與我的研究重大幫助的唐盛銘與兩位小幫手鄭幸怡與孫元駿， 有你們的幫忙才能讓我順利投稿了兩篇文章，並參與 MMSys 2023 的研討會。 我還想感謝男友廖翊丞，即便所學領域不同，仍然耐心的傾聽我研究上遇到的瓶頸， 也會在我忙得焦頭爛額的時候幫助我處理許多日常的瑣事， 並給予我很多精神上的支持。 最後，我想要感謝我的父母，給予我各方面的支援來栽培我， 讓我毫無後顧之憂的完成兩年的碩士生涯。

Acknowledgments

There are many people I appreciate during my two-year master's life. First, I would like to thank my advisor, Professor Cheng-Hsin Hsu. He is always happy to discuss the challenges I face in my research and makes an all-out effort to help me. He seldom blamed me when I made mistakes. Instead, he asked me to fix them as soon as possible and learn from them. He also actively introduced me to other researchers to exchange my research topic and learn from them, which broadened my horizons. I learned much in Prof. Hsu's lab, not only the knowledge and technologies related to computer science and the ability to summarize and plan things but also the good attitude, responsibility, and perseverance when handling a project. Next, I thank my labmates who have been with me for these two years, especially my buddies, Sheng-Ming Tang, Chih-Ya Cheng, and Gui-Hua Shi. I knew I was not alone in the research life with their company. I also want to thank Tzu-Kuan Hung for his advice and help when I was a first-year graduate student. Additionally, I want to thank Sheng-Ming Tang and my two assistants, Shin-Yi Zheng and Yuan-Chun Sun, for their great help with my research. I submitted two papers successfully and got the opportunity to attend the 2023 MMSys conference with their help. I also genuinely appreciate my boyfriend, Yi-Cheng Liao. He patiently listened to my research problems when I hit a wall in my research, though he isn't majoring in computer science. He also helped me with daily routines and provided mental support when my hands were full. Last but not least, I would like to thank my parents for their full support, which provided me with an excellent environment to focus on research.

Contents

中文摘要 **i**

Abstract **ii**

致謝 **iii**

Acknowledgments **iv**

1 Introduction **1**
- 1.1 Contributions 5
- 1.2 Organizations 7

2 Background **8**
- 2.1 Networked VR Systems 8
- 2.2 Differential Privacy 9
- 2.3 AutoRegressive Model 10
- 2.4 Linear Minimum Mean Squared Error 11

3 Related Work **13**
- 3.1 6DoF VR Dataset 13
- 3.2 Privacy-Preserving Methodologies 15

4 VR Dataset **17**
- 4.1 Collection Testbed 17
- 4.2 Dataset Collection 20
- 4.3 Privacy Threats Investigations 28

5 Privacy Threats **31**
- 5.1 Threat Model 31
- 5.2 The Placement of the Disturber 32

6 Privacy Threats Mitigation **33**
- 6.1 Overview 33
- 6.2 Trajectory Disturber 34
- 6.3 Perturbed View Compensation 36
- 6.4 Implementations 38

TSING HUA UNIVERSITY

7 Evaluations **40**
7.1 Re-identification Attack 40
7.2 Visual Quality Assessment 41
7.3 User Study Design 42
7.4 Results 42

8 Conclusion and Future Work **59**
8.1 Concluding Remarks 59
8.2 Future Work 60

Bibliography **64**

清華大學
TSING HUA UNIVERSITY

List of Figures

1.1 A VR user interacts with a VR application via the HMD and two hand-held controllers. 2

1.2 Unique privacy-sensitive data that our dataset provides: (a) physical RGB video, (b) physical depth video, (c) answer to the demographic questionnaire. 3

1.3 The concept of the privacy-threats mitigation for VR applications. 4

1.4 The rendered scenes: (a) original view with 110° FoV, (b) perturbed view with 110° FoV, (c) original view with 104° FoV, and (d) compensated view with 104° FoV. 6

2.1 A general networked VR system. 8

4.1 The setup of our testbed. 18

4.2 The considered scenes: (a) city, (b) nature, (c) office, and (d) gallery. . . . 20

4.3 The distribution of average frame rates in Unity. 24

4.4 The sampling rates of individual sensors in the cleaned-up dataset. The error bar of RGBD is too short to be visible. 25

4.5 Directory structure. 26

4.6 Sample lines of the demographic and VR background answer file. 26

4.7 Sample lines of the City scene's experience answer file. 27

4.8 Sample lines of a City scene's log file. 27

4.9 Sample frames of physical RGBD videos: (a) RGB frame (b) Depth frame. 27

4.10 Comparison of feature importance from the different sensors. 28

4.11 Comparison of feature importance from the different HMD data. 29

4.12 The identification rate of the RF and SVM with different levels of noise. . 29

6.1 Our privacy-preserving system architecture. 33

6.2 Our proposed trajectory disturber. 34

6.3 Our proposed view compensator. 37

6.4 Sample view compensation to avoid shaky views due to perturbations. . . 37

TSING HUA UNIVERSITY

7.1 The sample viewport pair of the Office scene. 42
7.2 Re-identification rate under different ϵ for DO and MG. 43
7.3 Visual quality with and without compensation. 44
7.4 Privacy-quality tradeoff achieved by different privacy-threat mitigation approaches, with visual quality in: (a) PSNR, (b) SSIM, and (c) VMAF. . 45
7.5 Visual quality comparison of the four scenes under different ϵ with DO: (a) PSNR, (b) SSIM, and (c) VMAF. 46
7.6 Rendered city building: (a) with and (b) without compensation. 47
7.7 Visual quality comparison of the Gallery scene under different ϵ with DO : (a) PSNR, (b) SSIM, and (c) VMAF. 48
7.8 Visual quality comparison of the Nature scene under different ϵ with DO: (a) PSNR, (b) SSIM, and (c) VMAF. 49
7.9 Visual quality comparison of the Office scene under different ϵ with DO: (a) PSNR, (b) SSIM, and (c) VMAF. 50
7.10 Visual quality comparison of the City scene under different ϵ with DO: (a) PSNR, (b) SSIM, and (c) VMAF. 51
7.11 Visual quality comparison among the four considered scenes with $\epsilon = 3.33$ using DO: (a) PSNR, (b) SSIM, and (c) VMAF. 52
7.12 Visual quality comparison among the four considered scenes with $\epsilon = 50$ using DO: (a) PSNR, (b) SSIM, and (c) VMAF. 53
7.13 Visual quality comparison among the four considered scenes with $\epsilon = 100$ using DO: (a) PSNR, (b) SSIM, and (c) VMAF. 54
7.14 Re-identification rate of different w under different ϵ with DO. 54
7.15 Visual quality of our disturber with different w under different ϵ using DO: (a) PSNR, (b) SSIM, and (c) VMAF. 55
7.16 Privacy-quality tradeoff achieved by our disturber with different w: (a) PSNR, (b) SSIM, and (c) VMAF. 56
7.17 User experience scores (a), (c) from Nature and (b), (d) from Office, where (a), (b) give visual quality scores and (c), (d) give dizziness scores. 57
7.18 Re-identification rate with and w/o adding perturbations to the orientations under different ϵ. 57
7.19 Visual quality of our disturber with and w/o adding perturbations to the orientations under different ϵ: (a) PSNR, (b) SSIM, and (c) VMAF. . . . 58

List of Tables

4.1 Our Demographic Questionnaire. 21
4.2 Our VR Background Questionnaire. 22
4.3 Our Experience Questionnaire. 23
4.4 Fractions of Removed Outliers 24

7.1 Average Re-Id Rate with Std 48
7.2 Average Visual Quality with Std 49

清華大學
TSING HUA UNIVERSITY

清華大學
TSING HUA UNIVERSITY

Chapter 1

Introduction

Increasingly more Head Mounted Displays (HMDs) have been released by key manufacturers, like HTC, Pico, Meta, and Apple, which enable novel Virtual Reality (VR) usage scenarios [107, 112], such as healthcare [48, 90], training [48, 57], sports, tourism [11], and entertainment. In these scenarios, each user uses an HMD and two hand-held controllers [5, 145] to interact with VR applications as shown in Fig. 1.1. Both HMDs and controllers come with sensors like accelerometers and gyroscopes, which produce time-series sensor readings on *locations* (x, y, and z as Cartesian coordinates) and *orientations* (i, j, k, and l as quaternions). We collectively refer to the locations and orientations of an HMD and two controllers at a moment as a *pose*, and a time-series of poses as a *trajectory*.

Because of the immersive experience offered by HMD-enabled VR applications, their great market potential has been well recognized. For instance, the global VR market is projected to attract 171 million VR users by 2023 [62] and to reaches 20.9 billion USD by 2025 [42]. To draw and retain more VR users, Machine Learning (ML) algorithms have been adopted by many VR applications to improve their visual quality. For example, HMD viewports [141] have been predicted by various neural networks [31, 32, 54] for more optimized tiled streaming of 360° videos or 3D virtual worlds. Like other data-driven optimization approaches [63], detailed sensor data are collected from VR users' HMDs and the controllers for higher prediction accuracy and visual quality of streamed content [50]. Doing so, however, could lead to privacy threats [41, 134], including but not limited to VR users' identity [88], height/fitness [93], and typed text [113], which are often overlooked because trajectories are collected in the background. Moreover, because of the large amount of various data types that VR can collect and the unclear policies for VR companies to deal with users' data, the potential privacy threats for VR users are unlimited [41].

To study these privacy threats in VR, we first turn to existing datasets collected from VR applications. Most of the VR datasets are collected with the users consuming 360°

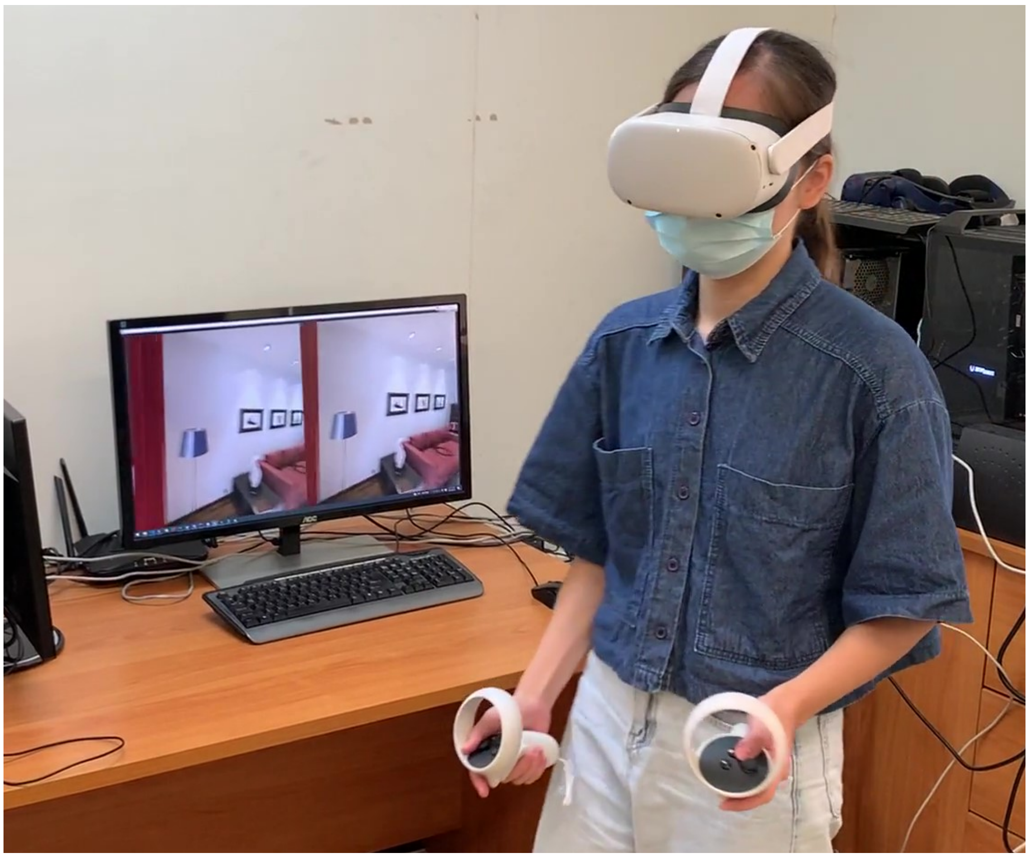
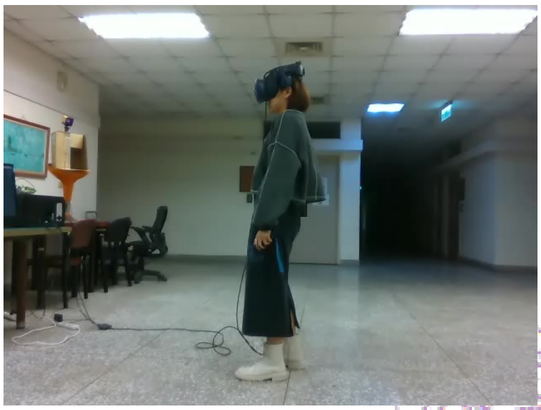
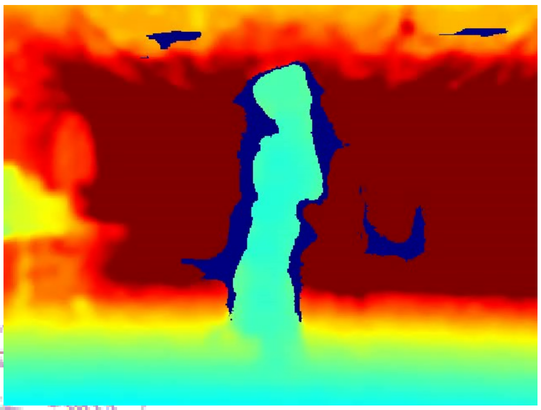


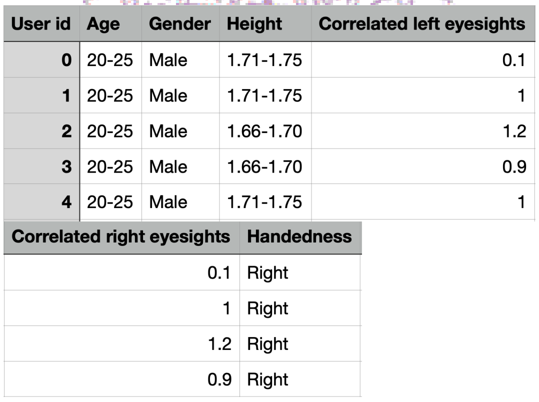
Figure 1.1: A VR user interacts with a VR application via the HMD and two hand-held controllers.



(a)



(b)



User id	Age	Gender	Height	Correlated left eyesights	Correlated right eyesights	Handedness
0	20-25	Male	1.71-1.75	0.1	0.1	Right
1	20-25	Male	1.71-1.75	1	1	Right
2	20-25	Male	1.66-1.70	1.2	1.2	Right
3	20-25	Male	1.66-1.70	0.9	0.9	Right
4	20-25	Male	1.71-1.75	1		

(c)

Figure 1.2: Unique privacy-sensitive data that our dataset provides: (a) physical RGB video, (b) physical depth video, (c) answer to the demographic questionnaire.

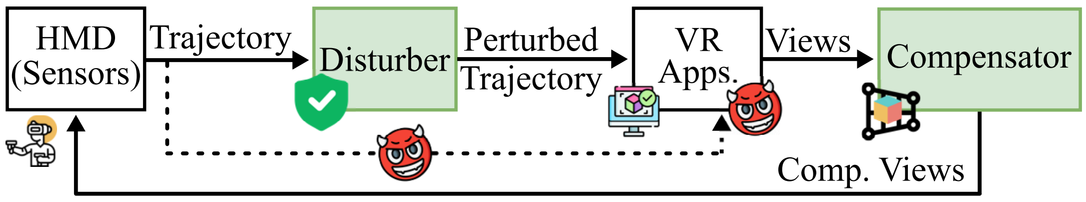


Figure 1.3: The concept of the privacy-threats mitigation for VR applications.

videos [1, 19, 36, 44, 50, 51, 73, 122, 138–140]. However, here, we focus on the behaviors of the VR users exploring 3D virtual worlds. There are quite a few VR datasets collected with 3D virtual worlds [20, 26, 52, 146]. We found that all existing public 3D-virtual-world VR datasets were collected for enhancing the quality of users' experience instead of studying the users' privacy. To investigate VR users' privacy comprehensively, some of the privacy-sensitive data types as shown in Fig. 1.2, such as the users' actual appearances while experiencing VR as shown in Fig. 1.2(a) and Fig. 1.2(b), and some demographic data of the users as shown in Fig. 1.2(c), should be included in the dataset either for being the ground-truth data or for more intensive privacy study. Though there are two datasets collected for studying privacy issues, which were collected with 360° videos [88], and a 3D virtual world [93], these datasets were collected only for the authors' research and were kept private. Moreover, some of these datasets [20, 93] focus on the scenarios in which users are completing some missions, for example, playing games. However, we want to study the privacy issues when users are just exploring 3D virtual worlds naturally without doing any task. Therefore, we designed a collecting procedure and set up the collecting testbed, recruiting 31 people to provide a public 6DoF (6 Degrees of Freedom) VR dataset of 3D virtual worlds. Besides those basic VR sensor data, including VR users' trajectories and eye gaze data, the dataset also contains the privacy-sensitive data of the VR users. The dataset is not only collected to study the privacy threats in VR, but also to investigate other potential privacy issues in VR, such as evaluating the efficiency of a privacy-preserving approach to protect VR users from privacy threats.

With the dataset, we can study how attackers can threaten VR users, and figure out the approaches to defeat those privacy threats. Fig. 1.3 illustrates a possible attacker that analyzes the trajectory from HMD sensors to threaten the privacy of a VR user, who is immersed in VR applications. If the raw trajectory is directly streamed to other entities without perturbation as shown by the dashed line in Fig. 1.3, user privacy may be threatened. Privacy threats can happen not only when the trajectories are being streamed, but also any entities that are not the users themselves. The encryption [125] cannot solve the problem since the raw trajectories are disclosed after decryption. A sample threat is the *re-identification* attack [88], referring to recognizing the identity of a VR user using their trajectory alone. One way to mitigate privacy threats is to *systematically* add noise to trajectories (or other extracted features) to get *perturbed trajectories*, which are less vulnerable to privacy threats. Here, by "systematically", we refer to random noise from some mathematical frameworks that ensure a VR user's privacy among a set of VR users, known as a *population*. One popular framework is *differential privacy* [21], which offers a control knob ϵ for trading off individuals' privacy and the population's statistics. In general, higher ϵ values lead to more accurate statistics at the expense of revealing more privacy of individuals. Differential privacy has been adopted in some VR applications to answer questions like: (i) What is the average gaze fixation time when reading a textbook [116]? and (ii) What are the objects gazed at by under-performing students [71]? Following a similar approach, and building upon it, we introduce a *disturber* that adds noise, or perturbations to individual VR users' trajectories to mitigate privacy threats as shown in Fig. 1.3. By properly adjusting ϵ, VR users could trade off the amount of revealed privacy and the achieved visual quality of VR applications. Even *if* an attacker gathers perturbed trajectories from the population, they will still have a hard time launching attacks guaranteed by the differential privacy framework.

In this thesis, we first collected a 6DoF VR dataset in 3D virtual worlds for our proposed privacy-preserving approach. Next, we set out to develop a disturber for trajectories of VR users to mitigate privacy threats. A straightforward way to design a disturber is to generate a fixed *offset* for each element of a VR user's trajectory, and add the same offsets to the whole trajectory [93]. Such an approach, however, ignores the temporal correlation among the poses of each trajectory. To cope with that, our disturber adopts the recently-proposed approach [144] to sequentially add different offsets drawn from a probability distribution to individual poses over time. While doing so, intuitively, leads to stronger protection against privacy threats, perturbed trajectories incur negative impacts on the visual quality of rendered scenes. More specifically, the rendered views are likely to be *shaky*. Fig. 1.4 shows the rendered view of a pose at a moment in time. The rendered view with a perturbed pose is askew, as shown in Fig. 1.4(b), compared to the rendered view with the original pose, as shown in Fig. 1.4(a). While the perturbed rendered views at different moments skew differently, the VR users may encounter severe cybersickness. To deal with this issue, we introduce a *compensator* that implements an efficient *image warping* algorithm [76, 120] to transform every rendered image from its perturbed pose back to that from its original pose. As shown in Fig. 1.4(d), by doing so, we turn the shaky rendered views caused by perturbed trajectories back to normal, compensated views.

1.1 Contributions

This thesis makes the following contributions:

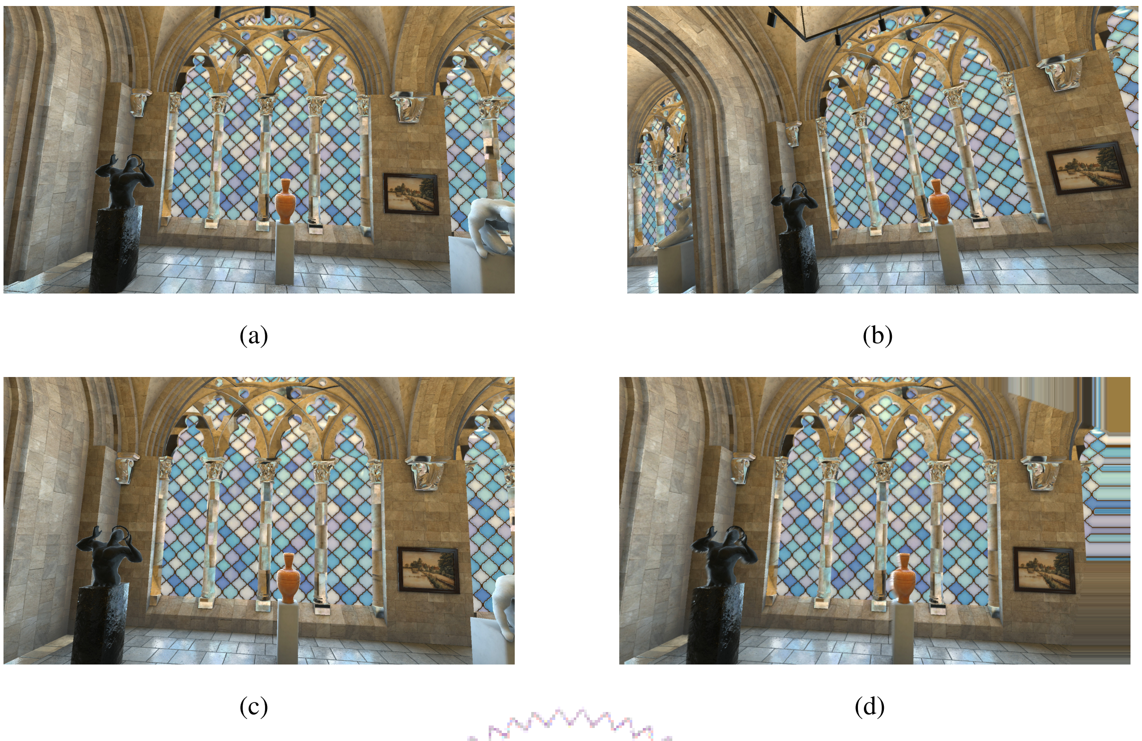


Figure 1.4: The rendered scenes: (a) original view with 110° FoV, (b) perturbed view with 110° FoV, (c) original view with 104° FoV, and (d) compensated view with 104° FoV.

- We collected and released one of the very first 3D-virtual-world 6DoF VR datasets for VR developers and researchers to investigate privacy issues in VR. The scripts that we use to log the sensor data are also included. Prior datasets either collected with 360° videos [1, 19, 36, 44, 51, 73, 122, 138–140], which is 3DoF, or lacked privacy-sensitive data of the subjects to complete such investigation more intensively and comprehensively [20, 26, 52, 146].
- We developed a disturber to add perturbations to time-series of poses on-the-fly to mitigate privacy threats. Prior works either ignored the temporal correlation [69, 93] or added perturbations offline [15, 17, 71, 116].
- We created a compensator to warp rendered views to eliminate the shakiness caused by perturbations introduced by the disturber. Doing so allows us to avoid visual quality degradation due to perturbations.
- We realized an ML algorithm for the re-identification problem [88] to exemplify privacy threats, and evaluate the effectiveness of our solution. We conducted extensive evaluations with our public VR dataset [137] captured from 3D virtual worlds to derive the *privacy-quality tradeoff* for our disturber with and w/o the compensation.

Our evaluation results demonstrate the merits of our solution: (i) our disturber alone reduces at most 0.4 re-identification rate compared to the state-of-the-art approach [93] under the same ϵ values, (ii) our disturber alone outperforms the state-of-the-art approach by 2.5 dB in PSNR, 0.1 in SSIM, and 10 in VMAF under the same privacy settings, and (iii) our compensator further improves the visual quality by at most 6.83 dB in PSNR, 0.45 in SSIM, and 34.57 in VMAF.

1.2 Organizations

We first introduce the limitations of the existing 6DoF VR dataset and privacy-preserving approaches for VR applications, and the challenges to protect VR users' trajectories in Chapter 1. Chapter 2 gives some background knowledge of the VR network system, the differential privacy framework, the AutoRegressive model, which we use to model VR users' trajectories, and the Linear Minimum Mean Square Error estimator that we used to estimate the VR users' trajectories. We next survey the existing 6DoF VR datasets and the perturbations related to VR data in Chapter 3. We present the collecting procedure, the content of our 6DoF VR dataset, and some sample usages of our dataset, which inspired the idea of our proposed privacy-threat mitigation methodology in Chapter 4. In Chapter 5 we introduce the privacy threats with the described networked VR system, list the existing privacy threats in the literature, and discuss the pros and cons of the placement of the perturbation along the networked VR system. We then elaborate on each component, the operations of the components with the defined notations, and the implementation of our proposed perturbations and compensation approaches respectively in Chapter 6. Then we evaluate the efficiency of our disturber with a state-of-the-art privacy-preserving approach using our 6DoF VR dataset in Chapter 7. Last, we summarize the whole thesis and list the future works in Chapter 8.

Chapter 2

Background

In this chapter, we introduce some pieces of background knowledge that support this thesis, i.e., the Networked VR Systems, Differential Privacy, AutoRegressive Model, and Linear Minimum Mean Squared Error.

2.1 Networked VR Systems

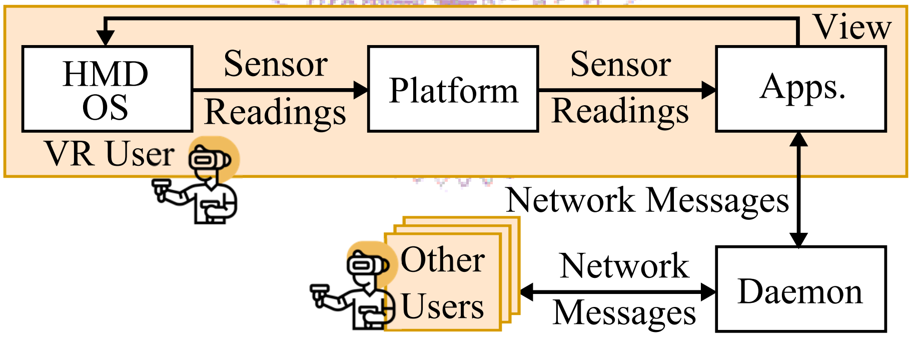


Figure 2.1: A general networked VR system.

Fig. 2.1 shows a networked VR system [50, 93], generally including five entities:

1. **HMD OS.** The operating system on the HMD that a VR user is equipped with. It tracks the user's sensor data to enable VR services. Usually, the user is also equipped with two hand controllers, and the controllers are also tracked for the VR services.

2. **Platform.** The VR platform, such as SteamVR [115] and VIVEPORT [49], acts as the bridge between VR users and VR service providers. It provides APIs for VR applications to access the tracking data to support their service to the users. On the other hand, it provides a comprehensive VR service to the user, such as providing lots of VR applications for the users to install and access.

3. **Application.** VR applications provide VR services to users. A VR application can be a VR game like Beat Saber [38], or a VR healthcare system like RelieVRx [6], which can be developed with a game engine, such as Unity [129]. Usually, VR applications are installed on users' PCs. Users can access the applications by connecting to the PC. VR applications use the API provided by the VR platform to access users' tracking data and other needed information to support their services to the users.

4. **Daemon.** In cases where the VR application involves online interaction with other players, the ser daemon on an external server is utilized to connect all the users for exchanging data. Examples of providers of such servers include Eclipse [25] and Fozzy [35].

5. **Other Users.** Other users of a VR application. Similarly, the users also have their own HMD, VR platform, and installed VR application.

In the networked VR system, one or multiple *VR users* are connected to one or multiple *servers* via the Internet. Every VR user interacts with a virtual world in 6DoF through an HMD and two controllers. The HMD/controller sensor readings are sent into a VR *platform* which host various VR *applications*. Generally, developers of VR platforms closely work with providers of servers. Various messages are exchanged among VR platforms used by individual VR users. Both the sensor readings and network messages carry VR users' trajectories to enable immersive experiences for VR users.

2.2 Differential Privacy

Differential privacy is a privacy framework that utilizes mathematics to quantify the amount of privacy that a *privacy mechanism* provides. Here, the privacy mechanism is a randomized algorithm that provides privacy protection for data. There are numerous famous companies that leverage differential privacy to protect their users. For example, Apple and Google use differential privacy to collect their user data from their browsers, Safari and Chrome, respectively, for various use cases such as the preferences of media playback [28, 123]. Differential privacy was first invented to quantify the privacy amount for statistical datasets [133], which guarantees that the answer to a *query*, which means any analyses or statistics for the dataset, does not mainly depend on any entry in the dataset. In other words, whether an entry is in the dataset or not does not affect the result of the query too much. For example, there is a dataset D containing the height of ten people. The dataset is kept private, while the average height of the ten people is public. In this case, the query is "what's the average height of D". Now, one of the ten people's height is removed from the dataset and leads to another dataset D', which is also private. If an attacker now queries for the average height of D', then the height of the removed person is disclosed. Fortunately, if the average height is protected by a privacy mechanism before being released, the removed data can be kept confidential as well. There are various kinds of differential privacy [133], such as *ϵ-Differential Privacy* [21], *(ϵ, δ)-Differential privacy* [22] and *geo-indistinguishability differential privacy* [4]. Among them, ϵ-Differential Privacy is one of the most popular types of differential privacy. It uses a parameter ϵ to represent the *privacy budget*, which determines the most amount of privacy leakage. A larger value of ϵ means more privacy budget and leads to less privacy protection. More specifically, let $\mathcal{M}$ be a random algorithm to protect the dataset D, and D' is the dataset that is one entry different from D. S is a subset of the output of the query. We say that the privacy mechanism $\mathcal{M}$ is ϵ-differential privacy if it satisfies

$$\mathcal{P}[\mathcal{M}(\mathcal{D}) \in \mathcal{S}] \leq \exp(\epsilon) \cdot \mathcal{P}[\mathcal{M}(\mathcal{D}') \in \mathcal{S}]. \tag{2.1}$$

Several privacy mechanisms are developed to achieve differential privacy, such as the *Laplace mechanism* [22], the *Gaussian mechanism* [23], and the *Exponential mechanism* [23]. The Laplace mechanism uses Laplace distribution to generate random perturbations, while the Gaussian mechanism generates random perturbations with Gaussian distribution. The scale b of the distribution is determined by $\Delta f/\epsilon$, where Δf is the l_1 *sensitivity* of a query, which is the amount of change in the query output when one entry is added or removed. The Exponential mechanism requires a *utility function* $util(\cdot)$ to determine the quality of the perturbed data r' based on its original data r, where r is generated from

$$r' \sim \exp(\frac{\epsilon \cdot util(r, r')}{2 \cdot \Delta_{util}}). \tag{2.2}$$

2.3 AutoRegressive Model

The *AutoRegressive (AR) model* [142] is a random process; it is a classic linear model that leverages mathematics and statistics to model the time series data [72, 75]. In AR, the current data are derived from the previous data. We can model the AR model with order p, which is denoted as AR(p), as:

$$V_t = \Sigma_{i=1}^{p} \phi_i V_{t-i} + U_t, \tag{2.3}$$

where $\phi_1, \phi_2, \ldots, \phi_p$ are the model parameters, and U_t is white noise. Therefore, the *First-order AR model*, AR(1), is:

$$V_t = \phi_1 V_{t-1} + U_t. \tag{2.4}$$

Here, U_t is the white noise with zero mean and variance σ_U^2, where $U_t \overset{\text{iid}}{\sim} \mathcal{N}(0, {\sigma_U}^2)$, where $\mathcal{N}(\cdot)$ denotes a normal distribution, and ϕ_1 is the autocorrelation ρ of the model. If $|\phi_1| < 1$, then the means of V_t, $E(V_t), \forall t$ are identical and equal to zero, and the variance of V_t is

$$ {\sigma_v}^2 = \frac{{\sigma_U}^2}{1 - {\phi_1}^2} = \phi^2 {\sigma_{v-1}}^2 + {\sigma_U}^2. \tag{2.5} $$

The Gaussian AR process [135] is one of the commonly used First-order AR in various domains:

$$ V_t = \alpha + \rho V_{t-1} + U_t, t \geq 1, \tag{2.6} $$

where $V_0 \overset{\text{iid}}{\sim} \mathcal{N}(\mu, {\sigma_v}^2)$, and ρ is the autocorrelation of $\{V_t\}_{t=1}^T$. If $\{V_t\}_{t=1}^T$ is a stationary Markov process, then $V_t \sim \mathcal{N}(\mu, {\sigma_v}^2)$, and $\alpha = \mu(1 - \rho)$, where $\mu = 0$.

2.4 Linear Minimum Mean Squared Error

We use *Linear Minimum Mean Squared Error (LMMSE)* to determine the estimated function. LMMSE is an estimate method that minimizes the *Mean Square Error (MSE)* of a linear model. Let $x \in X$ be an unknown random variable, $y \in Y$ be a known variable, and $\hat{x}(y)$ be an estimator of x given by y:

$$ \hat{x} = \hat{x}(y). \tag{2.7} $$

The MSE of the estimated value $\hat{x}$ is

$$ MSE = \mathbb{E}(\hat{x} - x)^2, \tag{2.8} $$

and the *Minimum Mean Square Error (MMSE)* estimator is

$$ MMSE = \arg\min_{\hat{x}(y)} \mathbb{E}(\hat{x} - x)^2. \tag{2.9} $$

If the estimated function is a linear model, where

$$ \hat{x} = \hat{x}(y) = ay + b, \tag{2.10} $$

then the MMSE of the function, which is the LMMSE, is

$$ MSE = \mathbb{E}[(ay + b - x)^2] \tag{2.11} $$

with the following properties:

1. $a = \hat{a} = \frac{\rho_{XY}\sigma_X\sigma_Y}{{\sigma_Y}^2}$
2. $b = \hat{b} = \mathbb{E}(X) - \hat{a}\mathbb{E}(Y)$

3. The $LMMSE = (1 - \rho_{XY}^2){\sigma_X}^2$

4. $\mathbb{E}[(X - \hat{a}Y - \hat{b})Y] = 0$

where ρ_{XY} is the correlation coefficient of X and Y, and σ_X and σ_Y are the variance of X and Y respectively. Then the formula for $\hat{X}$ is

$$\hat{X}(Y) = \rho_{XY}\frac{\sigma_X}{\sigma_Y}(y - \mathbb{E}(Y)) + \mathbb{E}(X). \tag{2.12}$$

TSING HUA UNIVERSITY

Chapter 3

Related Work

In this chapter, we first introduce the existing VR datasets, including those collected with both 360° videos and 3D virtual worlds. Next, we present existing privacy-preserving approaches for eye gaze data that are collected from both non-VR and VR applications, and the trajectory of VR users.

3.1 6DoF VR Dataset

Datasets of 360° video. 360° video streaming is one of the early VR applications [50], and there are quite a few datasets [1, 19, 36, 44, 51, 73, 122, 138–140] for diverse research purposes. Datasets [19, 73, 138, 139] are mainly used for designing and evaluating viewport/gaze prediction algorithms, where the head and/or eye movement of users are collected. For example, Lo et al. [73] recorded HMD's 6DoF data and used HMD's yaw, pitch, and roll to render saliency maps [14, 96] and motion maps [58] of users, and Xu et al. [139] recorded the eye and head movements of 31 HMD users when watching 360° videos to evaluate gaze prediction algorithms. There are datasets [1, 51] that contain eye movement [68] only, which are used for classifying eye movement [81, 91, 109]. For example, Agtzidis et al. [1] recorded and classified low-level eye movement into four categories: fixation, saccade, smooth pursuit, and noise, and Hu et al. [51] considered four high-level movements: free viewing, visual search, saliency, and track items. To understand the relationship between simulator sickness [60] and user head rotations, Fremerey et al. [36] collected a 360° video streaming dataset of head orientations and simulator sickness levels. Besides, several datasets [44, 122, 140] were gathered to analyze the correlation among biometrics, head/eye movement, and user emotion. We note that some 360° video datasets [36, 44] do record users' head positions. However, as users can only move in 3DoF when watching 360° videos, changing the head positions incurs no effect on the rendered views, and the displacements of positions in these datasets have been found to be small. *However, these datasets do not include personal attributes (e.g., age, gender, and height), and thus cannot be used to evaluate privacy-preserving approaches for personal attributes.*

Undisclosed dataset of 360° video related to privacy. Miller et al. [88] collected a dataset which includes the 6DoF trajectories from HMDs and two hand-held controllers of VR users viewing five different 360° videos. They then used the dataset to evaluate the feasibility of identifying the users' identities with three different machine learning algorithms, i.e., the k-Nearest-Neighbors (kNN), the Random Forest, and the Gradient Boosting Machine (GBM). The result shows that the identification rate can achieve 95% with that large amount of users. Although the dataset consists of data from 511 users, it has not been released and does not provide privacy-sensitive data about the subjects. Besides, all datasets of 360° video are not collected in the 3D virtual world with 6DoF interactions. Therefore, they cannot be used to investigate the privacy issue when introducing other types of user data, such as actual appearance.

Datasets of 3D virtual worlds. There are some datasets [20, 26, 52, 146] that record HMD user behavior when they were exploring 3D virtual worlds. Datasets [26,52] include trajectories of head movement, key-strokes of the two hand controllers, gaze, and scenes viewed by users. The former evaluated gaze prediction algorithms with their dataset and found that non-eye sensor data lead to more accurate gaze prediction, and the latter analyzed the correlation between the user gaze and head position. Dong et al. [20] collected users' trajectories of head movement and heart rate data while they were playing different kinds of VR games, i.e., Aircar [40], Beat Saber [38], Moss [102], Arizona Sunshine [121], and SUPERHOT [124]. The subjects were asked to fill out a simulator sickness questionnaire for studying the impact of each VR game on the cybersickness in VR. Datasets [146] provide the trajectories of 3D eye movement when users explore a virtual museum. The dataset was used to train a gaze prediction model. However, these datasets did not include the actual appearance of users, which may cause more severe privacy leakage or personal attributes, and thus cannot be used as a comprehensive dataset to investigate privacy issues.

Undisclosed dataset of 3D virtual worlds related to privacy. Nair et al. [92] designed an innocent-looking VR game and recruited 30 users to play the game. During the game, the users were requested to complete multiple tasks to escape the rooms, such as pressing a button, speaking out the word written on the wall, and mimicking the poses drawn on the wall. While the users were playing the game, some data, such as the 6DoF trajectories of users' HMDs and controllers and their voices were secretly collected. They then accurately inferred over 25 personal data attributes of the 30 users from these collected data, The inferred data include biometrics data such as height and fitness, environment such as room size and geolocation, device specifications, acuity, which are the grades of the Montreal Cognitive Assessment (MoCA) [55] test, and demographic data such as language and gender. The inferred results show that attackers can successfully harvest VR users' data in VR applications to threaten VR users' privacy. Although they collected multiple personal data attributes as ground-truth values, the dataset has not been released.

Consequently, we conclude that *there is no public VR dataset of 3D virtual worlds that can be used for comprehensive investigations of privacy issues*.

3.2 Privacy-Preserving Methodologies

Perturbations for eye gaze data collected from 2D content. To preserve viewers' privacy, perturbations have been added to eye gaze traces collected when users watch content on 2D monitors [37, 69, 71]. For example, Fuhl et al. [37] leveraged reinforcement learning [7] to add noise to the scanpath images generated from eye gaze traces. An autoencoder [143] is used to generate the scanpath images with lower resolution, which is seen as a method of adding perturbations to the images. The lower-resolution image is then fed into a classification agent to predict the privacy level of the perturbation by using different benign and malicious applications, such as document-type classification and gender inference. A manipulation agent then evaluates the predicted privacy level, scores the prediction, and manipulates the autoencoder based on the score. Liu et al. [71] added perturbations to aggregated saliency maps with Gaussian and Laplace differential privacy, and derived the privacy-utility tradeoff with the mean square error and the cross-correlation before/after adding perturbations. These two works focused on aggregated statistics, such as scanpath images and saliency maps, in an offline fashion. In contrast, Li et al. [69] introduced a perturbation system for eye gazes on-the-fly using the variant of differential privacy, i.e., the *geo-indistinguishability* [4] and *w-event differential privacy* [59]. The system can be integrated into different kinds of eye-tracking applications. *Different from these papers on 2D monitors, our work focuses on trajectories from VR users' HMDs.*

Perturbations for eye gaze data collected from VR. Although perturbations have also been added to eye gaze traces collected from HMDs, most prior studies [15, 17, 116] focused on extracted eye features [68], such as fixations, saccades, and blinks. For example, Steil et al. [116] collected an eye gaze dataset containing 52 eye movement features of 20 subjects. They then added noise to the extracted features using exponential differential privacy. They then derived a privacy-utility tradeoff using the document-type classification problem, which estimates the document type gazed by an HMD user. David-

John et al. [17] adopted *plausible deniability* [12] and presented *k-anonymity*, which was adapted from *k-same-select* [43], to design a privacy-preserving approach for eye gaze features, with mathematical definitions but limited utility degradation, and also in the document type classification problem. Bozkir et al. [15] considered temporal correlation in eye gaze traces by extending the Fourier perturbation algorithm [103], and then added noise to the extracted features. Very few works directly added perturbations to eye gaze traces from HMD users. To the best of our knowledge, there is only a prior paper [18] that perturbed eye gaze traces, and derived privacy-utility tradeoffs using applications like gaze prediction algorithm and saliency map generation. *Different from these works that considered eye gazes only, our paper considers trajectories from VR users' HMDs.*

Perturbation for VR trajectory data. Adding perturbations to non-eye-gaze data from VR users has only been recently explored. For example, Wei et al. [136] added noisy tiles around the tiles within a VR user's viewport when streaming a 360° video to them. They then empirically derived the tradeoff between viewport prediction accuracy and viewing experience. With the defined degree of privacy function, they found that the noisy tiles may degrade an algorithm but improve the overall user QoE. *Such an approach, however, is tightly coupled with VR applications, i.e., 360° tiled video streaming.* Nair et al. [93] is probably the closest work to ours. They added perturbations to HMD trajectories using Laplace differential privacy. Particularly, they first drew random perturbations from a probability distribution for features, such as height and fitness, which are inferred from the user's VR pose, They then converted perturbed features back to perturbed trajectories. *Different from their work, our paper takes the temporal correlation among poses in each trajectory into account when adding perturbations directly to trajectories.*

Chapter 4

VR Dataset

We elaborate on the collection testbed, the collecting procedures, and the data cleanup manner in this chapter. We then use the dataset to study the VR privacy threats and the potential privacy-preserving approaches for VR trajectory, which inspire the development of our privacy-threat mitigation system.

4.1 Collection Testbed

This section describes the hardware and software used to collect our dataset.

Hardware. Fig. 4.1 illustrates our collection testbed. We first introduce the employed hardware pieces.

- **VR PC.** It is a PC with an Intel i9-9920X CPU, 64 GB RAM, and an NVIDIA GeForce RTX 3080 Ti GPU. We installed Windows 10 on this PC to render HMD views and collect data.
- **HMD and controllers.** We chose HTC VIVE Pro Eye and its corresponding controllers, mainly for its built-in eye tracker. The HMD is connected to the VR PC with DP 1.2 and USB-C 3.0 cables. A converter box combines the two cables into a single cable to the HMD.
- **VIVE Beacons.** The HMD localization is realized with the help of two VIVE beacons from HTC.
- **RGBD PC.** It is another PC with an Intel i7-10875H CPU and 32 GB RAM for RGBD video recording from the RGBD camera.
- **RGBD Camera.** Some VR applications require users' actual appearance to reconstruct the virtual scenes they are in [16]. For instance, in teleconferences, we may want to talk with avatars similar to the remote participants' actual looks. Therefore, we use an Intel RealSense D435 camera to capture HMD users' actual appearance in RGB and depth videos while exploring 3D virtual worlds. The camera is con-

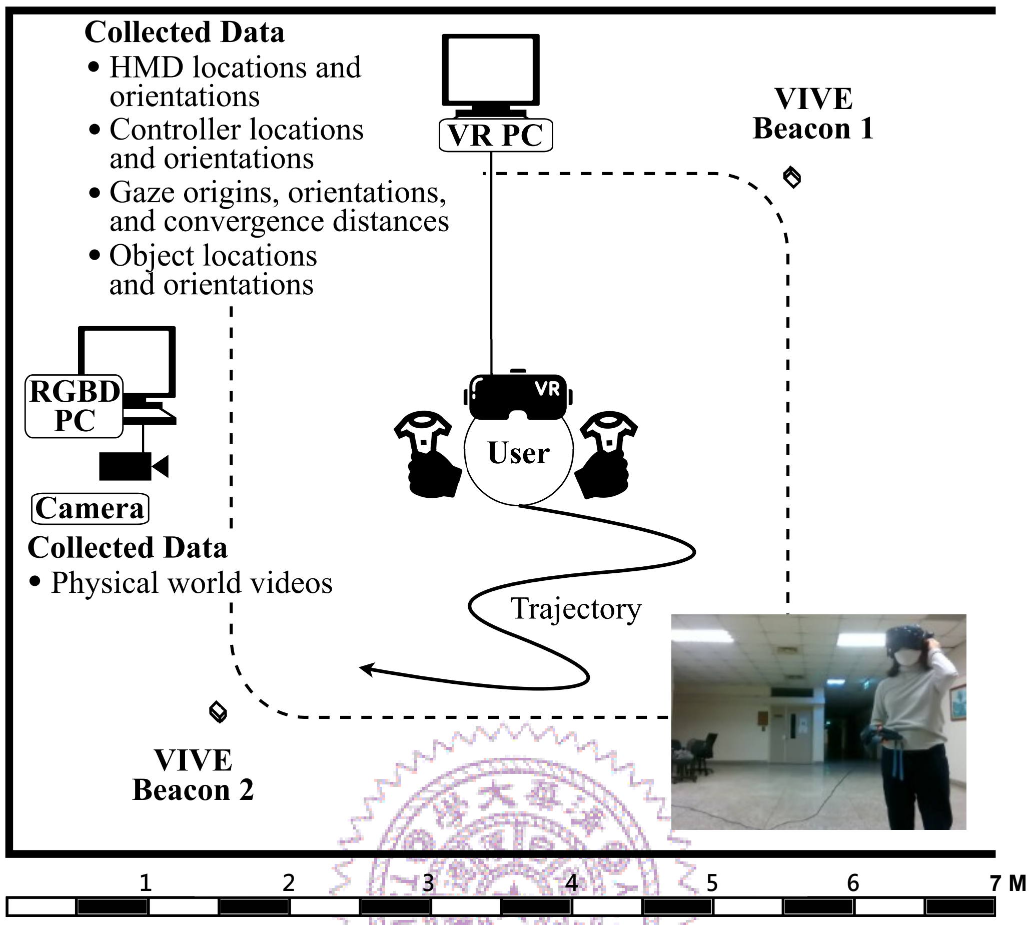

Collected Data
• HMD locations and orientations
• Controller locations and orientations
• Gaze origins, orientations, and convergence distances
• Object locations and orientations
VR PC
VIVE Beacon 1
RGBD PC
VR
User
Camera
Collected Data
• Physical world videos
Trajectory
VIVE Beacon 2
1
2
3
4
5
6
7 M

Figure 4.1: The setup of our testbed.

nected to the RGBD PC with a USB-C 3.0 cable.

These hardware pieces are placed in a hallway of our EECS building. Within a 4 $\times$ 4 m^2 space, we configure virtual barriers at the center, the dimensions of which are shown as the dashed box in Fig. 4.1. Users are warned when approaching the barriers for safety. Two beacons are placed at opposite corners just outside the barriers. We use Network Time Protocol (NTP) to align the clocks of different PCs, These clocks provide synchronized timestamps of collected samples. To use our dataset, one can align samples captured by different sensors using the timestamps in the granularity of milliseconds.

Software. Next, we present the key software applications and libraries adopted by our collection testbed.

- **Rendering engine.** We use Unity [129] version 2021.3 as our rendering engine. We create a Unity VR project and import different *scene* assets from the Unity Asset Store [130] to create multiple virtual worlds. We configure the rendering settings through Unity GUI, such as the frame rate and camera parameters.
- **World creation Toolkit.** XR Interaction Toolkit [132] is a package for creating virtual worlds. We employ version 2.2.0, which provides a game object called *XR*

Origin representing the HMD user in a virtual world. XR Origin is composed of (i) a camera representing the HMD user and (ii) two game objects representing the left and right controllers. We place an XR Origin at a suitable location in each scene.

- **Scripting language.** We use the Unity scripting API [131] to sense and record HMD user movements.
- **Eye tracker.** We adopt Tobii XR API [126] for accessing the eye gaze data, which include: (i) eye origins in 3D coordinates, (ii) eye orientations in normalized vectors, and (iii) convergence distances.
- **Camera interface.** We use a python wrapper [105] of Intel RealSense SDK 2.0 [104] to acquire physical RGBD videos of the subjects from the RGBD camera.

Building upon the above software, we developed a suite of tools as follows:

- **Sensor logger.** It consists of two modules for HMDs and eye trackers, respectively. The HMD module is written as Unity scripts to record: (i) the locations and orientations of the HMD with `GetComponent(···)`, (ii) the locations and orientations of the controllers with `GameObject.Find(···)`, and (iii) the key strokes of the controllers with `TryGetFeatureValue(···)`. The eye-tracking module adopts Tobii XR API's `GetEyeTrackingData(···)` to track the user gaze. All the sensor data from the HMD, controllers and eye trackers, except the object locations and orientations, are timestamped and stored in Comma-Separated Values (CSV) files. The object locations and orientations are saved in JavaScript Object Notation (JSON) files.
- **RGBD recorder.** We implement a Python-based RGBD recorder on top of the RGBS camera interface. We invoke `config.enablestream()` to enable the streams for both RGB and depth cameras. We initialize the recording pipeline by calling `pipeline.start(config)`. We set the depth sensor with several options including `gain`, `enable_auto_exposure`, `laser_power`, and `visual_present`. After that, a while loop starts for recording the RGB and depth frames into mp4 files with OpenCV [99]. The raw depth values are also saved as npy files.
- **Data visualizer.** We implement a visualizer to convert sensor data into videos to inspect sensor data and remove outliers. The visualizer only supports CSV files. In particular, we use `pandas` to read these data files and `matplotlib` to plot the sensor data into images. Last, we adopt `ffmpeg` to concatenate a sequence of images into videos.

The tools are included in our dataset.

Considered Scenes. We consider VR applications where HMD users freely explore VR worlds without performing assigned tasks. We built four scenes with diverse charac-

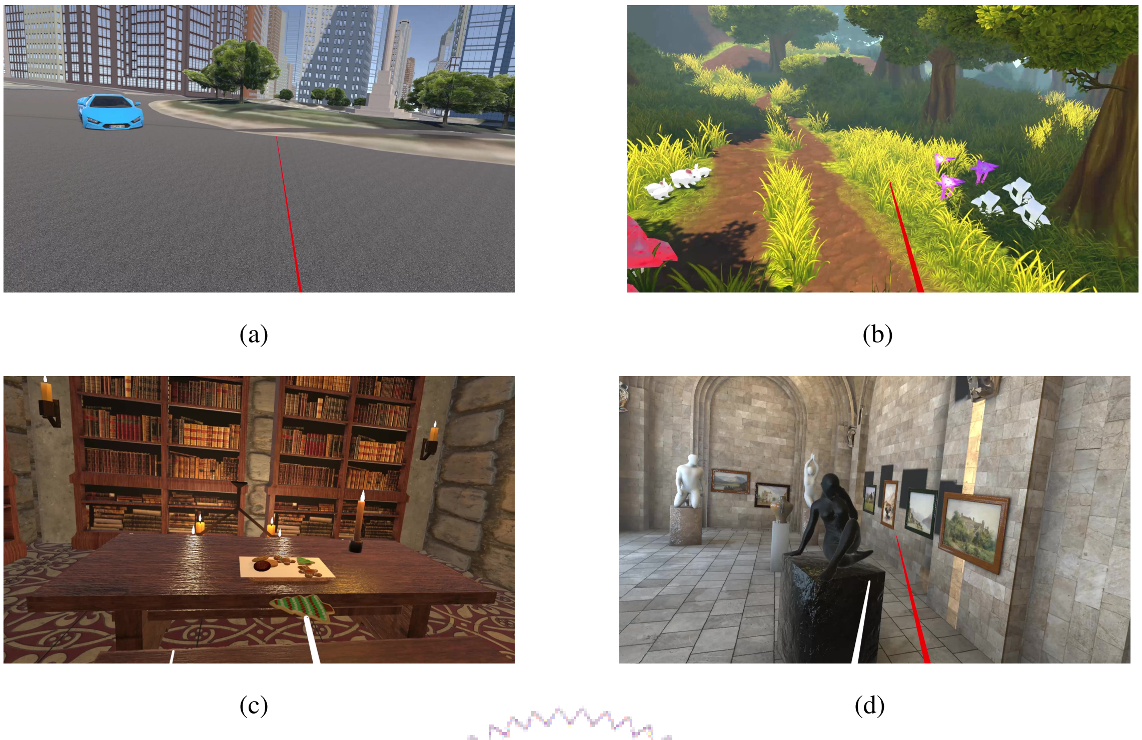


Figure 4.2: The considered scenes: (a) city, (b) nature, (c) office, and (d) gallery.

teristics: two outdoor ones, which are *city* [39] and *nature* [111]; and two indoor ones, which are *office* [117] and *gallery* [65]. We chose these scenes since they differ in the types of views and the space size. For the two outdoor scenes, the city has a larger space than nature; for the two indoor scenes, the gallery has a larger space than the office. See Fig. 4.2 for sample rendered HMD views. In each scene, we added 2–5 objects, such as cars, cookies, and statues, for users to interact with. Such interactions may lead to unnoticed privacy leakage. For instance, left-handed subjects may interact with the objects using their left hands more often. We only collected the dataset with four scenes due to the time limitation. It requires about 20 minutes for each subject to finish the collection with the four scenes, which does not include the time for adapting the unstable hardware. The total collection for all 31 subjects lasts a week to accommodate every subject's time. However, it is acceptable to consider new scenes in the collection. To add a new scene, import the scene to your Unity project, create an XR Origin game object, and add the provided tracking scripts, i.e., TrackingObj.cs and Tracking.cs, as components of your XR Origin.

4.2 Dataset Collection

In this section, we introduce our dataset.

Procedure. We recruit a set of subjects for collecting the dataset. Each subject

Table 4.1: Our Demographic Questionnaire.

Age	☐ 15-20 ☐ 21-25 ☐ 26-30 ☐ 31+
Gender	☐ Male ☐ Female
Height (m)	☐ 1.50-1.55 ☐ 1.56-1.60 ☐ 1.61-1.65 ☐ 1.66-1.70 ☐ 1.71-1.75 ☐ 1.76-1.80 ☐ 1.81+
Correlated Eyesights	Left: ________ Right: ________
Handedness	☐ Left ☐ Right

TSING HUA UNIVERSITY

first signs the informed consent and fills out the demographic questionnaire and the background of the subjects' VR experience. The content of the two questionnaires is listed in Table 4.1 and Table 4.2 respectively. Our research assistants helped the subject set up the HMD and controllers. The subject then calibrated the HMD eye tracker before the scenes were rendered. We rendered the four considered scenes sequentially for the subject to explore. Each scene lasted for two minutes. We skipped 2.5 seconds at the beginning and end when recording the sensor data to prevent missing data samples due to high and fluctuating system workload. Between any two scenes, the subject filled in an experience questionnaire to provide feedback on the immersive level of the virtual world. The questions of the experience questionnaire are listed in Table 4.3. The subject answered each question verbally with help from the research assistants. We gave each subject a 30-second break before the next scene started to avoid fatigue. Typically, it took a subject 20 minutes to complete their data collection. The subjects were allowed to skip any question or quit the data collection session at any time. Among 31 subjects, none of them opted for those two options.

For each subject, we collected the following data.

- **HMD locations and orientations.** The 3D world coordinates and quaternions of

Table 4.2: Our VR Background Questionnaire.

How many times have you used VR before?	☐ 0 ☐ 2-5 ☐ 6-10 ☐ 11+
How often did you experience motion sickness when using VR?	☐ 1 (Never) ☐ 2 (After 1 minute) ☐ 3 (After 3 minutes) ☐ 4 (After 5 minutes) ☐ 5 (Always)

the HMD in the VR world over time are recorded.

- **Controller locations and orientations.** The controller coordinates, and quaternions are saved
- **Controller key strokes.** We record the keys pressed by subjects to interact with the added, interactable objects during their exploration. More specifically, we record the (i)*grip* button states and (ii) fingertip 2D coordinates on the *touchpad* of HTC VIVE controllers.
- **Eye gaze.** It is composed of (i) eye origin in 3D coordinates, (ii) eye orientations in 3D vectors, and (iii) convergence distance in meters.
- **Object locations and orientations.** The 3D world coordinates and quaternions of each interactable object in the virtual world.
- **Physical RGBD videos.** The RGBD videos of subjects' appearance and movement. Both the RGB and depth channels are encoded by H.264 into mp4 files. We also save the raw depth values in npy files.
- **Questionnaire.** We collect demographic and VR background questionnaires once for each subject and an experience questionnaire after the subject explores a scene.

We configure the data logger and other software modules to collect data samples at the following rates: (i) HMD, controller, and object locations and orientations, as well as controller key strokes at 50 Hz, and (ii) RGBD videos at 30 Hz. Note that the views in HMD can be rendered offline by the Unity engine using the collected locations and orientations of the HMDs, controllers, and objects. The resulting virtual world RGBD videos may be useful when investigating the privacy concerns of VR applications.

Data Cleanup. During our data collection, we noticed that the eye gaze data from Tobii XR API [126] may be invalid due to some situations, for example, when the subject is not looking at the screen [127].

Table 4.3: Our Experience Questionnaire.

How is the overall quality?	☐ 1 (Very bad) ☐ 2 ☐ 3 ☐ 4 ☐ 5 (Great)
How is the visual quality?	☐ 1 (Very bad) ☐ 2 ☐ 3 ☐ 4 ☐ 5 (Great)
Do the objects move as you expect?	☐ 1 (Very bad) ☐ 2 ☐ 3 ☐ 4 ☐ 5 (Extremely)
How is the immersive level?	☐ 1 (Very bad) ☐ 2 ☐ 3 ☐ 4 ☐ 5 (Great)
Would you continue exploring the scene under the current system quality and immersive level?	☐ Yes ☐ No

TSING HUA UNIVERSITY

After collecting sensor data from 31 subjects, we ran a sanity check and found that 7 subjects suffered from 10+% loss on gaze samples in at least one scene. Therefore, we excluded these 7 subjects from our dataset, which then contained 24 subjects.

Although we configured the testbed to save HMD, controller, and object samples right after each frame of the HMD view is rendered in Unity, the VR PC cannot always keep up with the target frame rate of 50 Hz. Fig. 4.3 gives the distributions of individual subjects' average Unity frame rates in different scenes. This figure reveals that City has a lower frame rate (about 40 Hz) than other scenes. A closer look indicates that City is almost 20 times larger than the three other scenes regarding volume. Therefore, rendering City incurs higher computational complexity and lower unity frame rate. Because of the non-trivial workload, we found the timestamps of sensors (other than the RGBD camera)

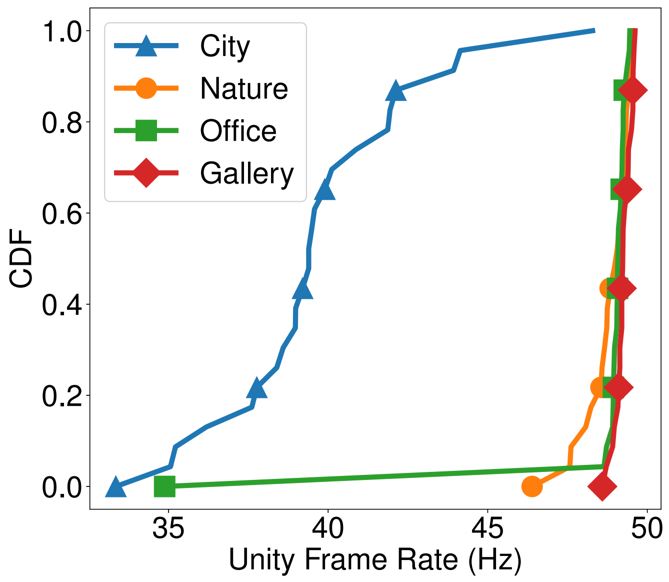


Figure 4.3: The distribution of average frame rates in Unity.

Table 4.4: Fractions of Removed Outliers

	Mean	**Std.**
HMD	1.2%	3.2%
Left Ctrl.	3.5%	5.1%
Right Ctrl.	3.0%	3.5%
Gaze	5.1%	4.0%

occasionally suffer from jitters. We, therefore, slightly augment the timestamps to equally space sampling falling in the same duration of 1 second.

Theoretically, the sampling rates of sensors connected to the VR PC should align with the unity frame rate. However, we inspected the collected sensor data and found that the gaze data may be invalid in some situations. Moreover, we found that the sampling rates of HMD, controller, and object are the same as the frame rate of Unity, and our dataset may only miss some gaze samples. In particular, we miss 1.65% gaze samples on average, with a standard deviation of 1.62% across all 96 subject-scene pairs.

Like other measurement studies, our collected sensor data may suffer from noise that needs cleaning. For example, when a subject moves to some blind spots of VIVE beacons, the locations and orientations of the HMD and controllers could be incorrect. We first visualize the collected sensor data to find outliers, then propose the following heuristics to detect and remove outliers.

- **HMD.** Samples falling outside of the $4 \times 4\ m^2$ barrier box are removed.

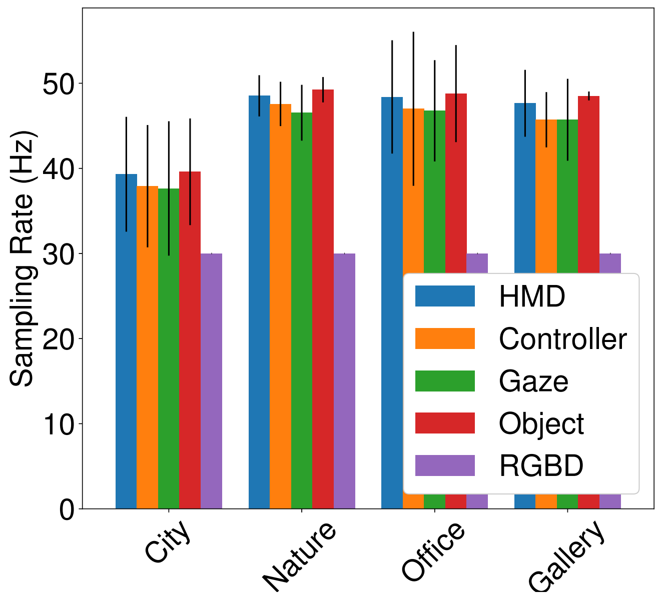


Figure 4.4: The sampling rates of individual sensors in the cleaned-up dataset. The error bar of RGBD is too short to be visible.

- **Controller.** Samples with a distance longer than 1.1 m from the HMD are removed. We get 1.1 m by measuring the distance of the tallest subject, 1.86 m.
- **Gaze.** Samples with a distance longer than 0.1 m from the HMD are removed. We get 0.1 m from the physical size of our HMD.

Table. 4.4 gives the fractions of removed outliers across 24 subjects. The cleaned-up sensor data are put together into our dataset.

Basic Statistics. We notice that samples from different sensors are not synchronized in our dataset. Fig. 4.4 summarizes the average sampling rates of different sensors across 24 subjects, where error bars indicate 95% confidence intervals. While the detailed demographic data of our subjects are given in the dataset, we give high-level statistics below.

- **Gender.** 75.0% of subjects are male.
- **Age.** All subjects are of 20–30 years old.
- **Height.** Most subjects' height is between 1.70–1.75 m.
- **Handedness.** 91.7% of subjects are right-handed.
- **VR experience.** Half of the subjects have no VR experience.
- **Simulator sickness.** 9 subjects experienced simulator sickness.

Directory Structure. Fig. 4.5 shows the structure of our dataset. We briefly introduce the top-level directories in the following.

- **Questionnaires.** They include the demographic and experience questionnaires as two pdf files. The VR background questionnaire is merged with the demographic

```
├── Questionnaires
│   ├── demographic.pdf
│   ├── experience.pdf
│   └── answers
│       ├── demographic.csv
│       ├── City_exp.csv
│       ├── Nature_exp.csv
│       ├── Office_exp.csv
│       └── Gallery_exp.csv
└── Tools
    ├── SensorLoggers
    │   ├── Tracking.cs
    │   └── TrackingObj.cs
    ├── realsense.py
    ├── visualize.py
    └── README.md

├── City
│   ├── Trajectory
│   │   ├── city_$U.csv
│   │   └── city_$U.json
│   └── PhysicalRGBD
│       ├── city_$U.mp4
│       ├── city_depth_$U.mp4
│       └── city_$U.txt
├── Nature
│   └── ...
├── Office
│   └── ...
├── Gallery
│   └── ...
├── README.md
└── change.log
```

Figure 4.5: Directory structure.

```
user id, answers to the demographic questions and VR background questions
0, 20-25, Male, 1.71-1.75, 0.1, 0.1, Right, 0, 1 (Never)
1, 20-25, Male, 1.71-1.75, 1, 1, Right, 2-5, 1 (Never)
2, 20-25, Male, 1.66-1.70, 1.2, 1.2, Right, 2-5, 1 (Never)
3, 20-25, Male, 1.66-1.70, 0.9, 0.9, Right, 10, 4 (After 5 minutes)
4, 20-25, Male, 1.71-1.75, 1, 1, Right, 0, 1 (Never)
...
```

Figure 4.6: Sample lines of the demographic and VR background answer file.

one as one pdf file. The inputs from all subjects are saved in CSV files. The demographic inputs of all subjects are saved in the demographic.csv file, and the structure of the file is shown in Fig. 4.6. The experience inputs of all subjects exploring each scene are stored in the $scene_experience.csv file, where $scene represents the scene name. Fig. 4.7 shows the sample lines of the City experience answer.

- **Tools.** Two scripts, TrackingObj.cs and Tracking.cs, are provided for capturing the sensor data with and without interactable objects. To use these two scripts, import the scripts into your Unity project, and add the scripts as components of your XR Origin. We also give realsense.py to capture the RGBD videos. In addition, data_visualizer.py is the script to visualize the sensor data.
- **City (and other scenes).** The sensor data from City (and other scenes) are stored in this folder. The trajectory from HMDs, controllers, objects, and eye trackers

```
user id, answers to the experience questions
0, 3, 3, 3, 3, y
1, 2, 3, 3, 2, n
2, 4, 4, 4, 4, y
3, 4, 5, 3, 4, y
4, 3, 3, 2, 4, y
...
```

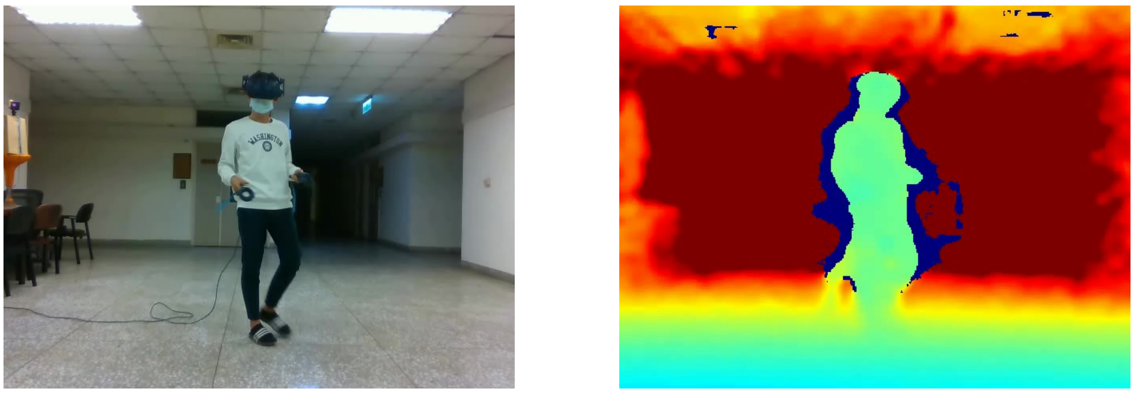
Figure 4.7: Sample lines of the City scene's experience answer file.

```
Unity time, head pose, left/right ctrl. pose and key strokes, eye pose and conv., ntp time, user id
0.08000000, -724.80690000, 26.66229000, -1040.00900000, ..., 1673440396.17647076, 3
0.09999999, -724.80690000, 26.66229000, -1040.00900000, ..., 1673440396.23529434, 3
0.12000000, -724.81010000, 26.66180000, -1040.01400000, ..., 1673440396.29411793, 3
0.14000000, -724.81510000, 26.66104000, -1040.02100000, ..., 1673440396.35294151, 3
0.16000000, -724.82840000, 26.65618000, -1040.04300000, ..., 1673440396.41176510, 3
...
```

Figure 4.8: Sample lines of a City scene's log file.

are stored in the city_$U.csv and city_$U.json files in the Trajectory folder, where $U represents the subject id. Fig. 4.8 shows the sample lines of a log file. Note that we do not remove the outliers in the JSON files due to the continuity of the frames while rendering the HMD view. Under physical RGBD, the physical RGB and depth videos are saved as city_$U.mp4 and city_depth_$U.mp4, respectively. Fig. 4.9 shows the frames of a sample RGB video and a sample Depth video. The timestamps of RGBD frames are saved in city_$U_t.txt.

We include a README.md in several folders to guide researchers, engineers, and hobbyists to use our dataset.



(a) (b)

Figure 4.9: Sample frames of physical RGBD videos: (a) RGB frame (b) Depth frame.

4.3 Privacy Threats Investigations

In this section, we illustrate some privacy threats with our dataset, which can inspire the design of privacy-persevering approaches.

Infer personal attributes: We use height and handedness as a toy example. To infer the subjects' height, we first calculate the distance from the XR origin and the sensed HMD locations y coordinate to present the subjects' height in Unity. The distance is the height of the subject in meters since a unit in Unity equals one meter in the real world. We then calculate these distances from each scene for each subject and average the distances to get the average height of each subject. The average height of each subject is added by 0.1 meters to adjust the distance from the eyes to the top of their head. Next, we compare the inferred height of each subject (denoted as h) to the ground truth (represented as $[h_a, h_b]$). The inference is considered correct if $h \in [h_a - 0.05, h_b + 0.05]$. Note that the added value for the range is for the error caused by the subjects' movements, such as jumping up and squatting down. The accuracy of inference is **75%**, which can be improved in the future. For handedness, we calculate the pressed times of grip buttons on both the left and right controllers respectively for each subject. If the pressed times on the left controller are more than that on the right controller, we infer that subject is left-handed; otherwise, the subject is inferred to be right-handed. The inferred results are also compared to the actual demographic data, and the inference accuracy is 58.33%. The inference accuracy for both cases is not so good; therefore the heuristic inference algorithms can be improved in the future.

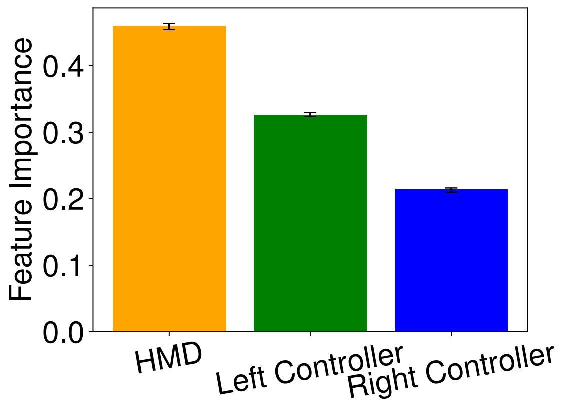


Figure 4.10: Comparison of feature importance from the different sensors.

Identify users: A more severe issue is that users may be identified with sensor data of HMD. As in [88], we classified the trajectory, i.e., the HMD and two controllers' locations and orientations, in our dataset to demonstrate the user identification issue. There are various classification algorithms, such as Random Forest (RF), k-Nearest-Neighbors (kNN), Gradient Boosting Machine (GBM), and Support Vector Machine (SVM). Among

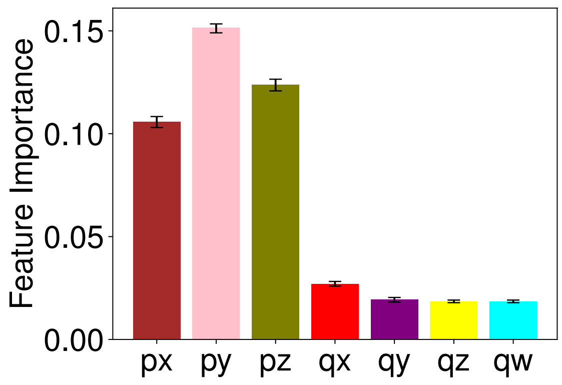


Figure 4.11: Comparison of feature importance from the different HMD data.

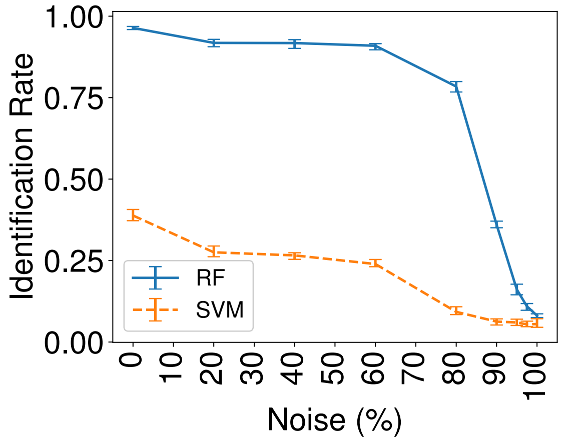


Figure 4.12: The identification rate of the RF and SVM with different levels of noise.

these classifiers, we chose the RF and SVM in our experiment. A subject exploring a scene is referred to as a *task*. The samples with outliers are skipped in this experiment. We extract the features, i.e., the mean, standard deviation, median, maximum, and minimum values for the samples within each second. Then, there are at most 115 sets of features for each task. We use the Scikit Learn [67] library to train RF and SVM classifiers. For each task, the ratio of training and testing sets is 80:20. The results are averaged over ten times of training and testing. The average identification rate of RF can achieve **96.41%**. We then use the feature importance metric to analyze the dominance of the features in user identification. Feature importance evaluates the significance of each feature to the final identification; the sum of the feature importance scores is one. Fig. 4.10 shows the feature importance score of each sensor. We gathered all the feature importance scores of the features derived from HMD, left and right controllers, respectively, and found that the sensor data of HMD is the most important. Similarly, we found that among the sensor data of HMD, the y coordinate of the HMD location is the most important as shown in Fig. 4.11, which is consistent with Miller et al. [88].

Adding noise to preserve privacy: Adding noise to the raw data can preserve privacy. Similar to adding Gaussian noise to eye gaze data [18], we use adding it to the trajectory to demonstrate the impact of noisy data on user identification.

Gaussian noise is added to each dimension of each sample, i.e., each value v of each dimension in each sample will become $\{v + N(0, \sigma)\}$. When adding noise, the utility loses. We use the percentage of samples in each second added by noise to represent the noise level. In the experiment, we set noise levels as 0%, 20%, 40%, 60%, 80%, 90%, 95%, 97.5%, and 100%. We trained and tested the classifier ten times for each noise level and calculated the mean accuracy. The mean and standard deviation σ of the Gaussian noises were set to zero and ten. Fig. 4.12 shows the mean identification rate under each noise level for the RF and SVM classification algorithms. We can see that the identification rate fluctuates when the noise level is lower than 70%, but degrades drastically after that. When the noise level reaches 100%, it is hard to identify the HMD users by their sensed data.

These prior tests show that the privacy of a VR user can be threatened by revealing their trajectory. Therefore, it is essential to design a privacy-threat mitigation approach to protect VR users' trajectories. Adding extra noise (perturbation) to users' trajectories is one of the promising ways to achieve that.

TSING HUA UNIVERSITY

Chapter 5

Privacy Threats

In this chapter, we first introduce the threat model of our system, and list existing privacy threats in VR. Next, we discuss the possible placements of our proposed disturber with the pros and cons and how we can implement it for each placement.

5.1 Threat Model

As mentioned in Sec. 2.1, each entity in the networked VR system closely works together with each other carrying VR users' trajectories to provide VR services. Therefore, each of the entities can be a potential malicious attacker that intercepts VR users' trajectories. For example, attackers may plug in malicious firmware to a try-out HMD in a video game store to secretly steal the trajectories from the customers. Once the attackers get the VR users' trajectories, they can analyze the trajectories to reveal users' privacy and threaten the users.

In the literature, several possible privacy threats have been recognized in VR applications [88, 92, 113, 134], For instance, Wang et al. [134] reported that metaverses often dictate their users to provide personal data, such as gaze movement, facial expression, gait, and voice, which could negatively affect HMD users' safety in both the virtual and physical worlds. In addition, Nair et al. [93] showed that VR gamers may be unaware of the leakage of their personal data, such as height and fitness, once their in-game behaviors are analyzed. Similarly, Miller et al. [88] demonstrated that the identity of HMD users can be recognized if their HMD and controller trajectories are inspected, which is called a re-identification attack. Slocum et al. [113] also showed that what the user is typing can be inferred from head motions alone. Besides, gaze data are also privacy-sensitive. Most of the HMDs are equipped with an eye-tracker nowadays to enable some VR applications, such as streaming optimization [74], foveated rendering [10, 33, 83, 84, 100], redirect walking [66, 119], etc. Gaze data can reveal users' preferences [27], psychological states [87], and diseases [45] and even recognize them [24, 46, 87]. Steil et al. [116] conducted an online survey on the privacy issue of gaze data and found that people have privacy concerns about their gaze data and are only willing to share their gaze data under certain restrictions.

5.2 The Placement of the Disturber

To mitigate such privacy threats, our proposed disturber can be deployed in HMDs, VR platforms, or VR applications. Deploying disturbers on the HMDs/controllers provides the strongest protection while supporting all VR platforms and applications. However, implementing disturbers in hardware incurs high engineering complexity. One possible solution is to adopt ALVR [3] to connect their HMD to their PC through Wi-Fi and to implement the disturber inside the ALVR client. In contrast, disturbers can be implemented in software and deployed in VR platforms. For example, a wrapper function can be added to a Unity [129] plugin running on SteamVR [115]. By instrumenting the wrapper function to intercept and perturb HMD/controller trajectories, it could work with any VR applications on that platform. Last, implementing disturbers in VR applications requires the least engineering effort and could incorporate application-specific optimization. However, duplicated efforts are needed across multiple VR applications. *In summary, our disturber can be implemented in HMDs, VR platforms, and VR applications.* Moreover, our compensator can be implemented next to the rendering engines, like Unity in either VR platforms or applications. In this thesis, we implemented the disturber in the VR application to get the evaluation results as soon as possible. The performances of the disturber algorithm are the same in these different placement settings. The placements only relate to the entities that can get the ground-truth poses.

Chapter 6

Privacy Threats Mitigation

In this chapter, we first give the whole picture of our system, and list all the components and the data flow. Next, we specify our disturber and compensator, elaborating on how they work. Finally, the software pieces we use to implement the system are presented.

6.1 Overview

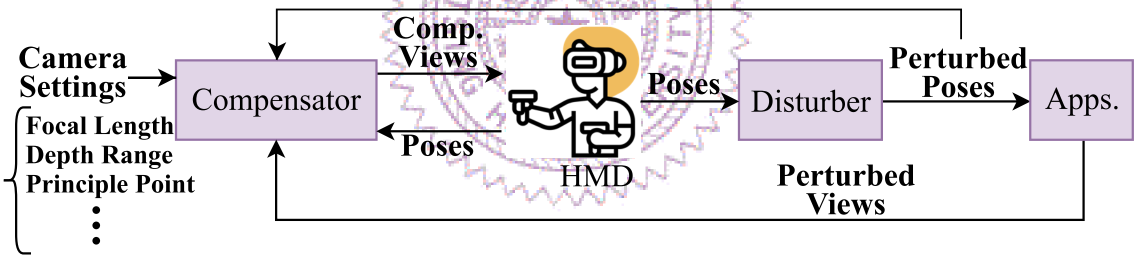


Figure 6.1: Our privacy-preserving system architecture.

Fig. 6.1 shows the overview of our privacy-preserving system, which contains four key components:

- **HMD.** The HMD that a VR user accesses to 3D virtual worlds.
- **Disturber.** Our disturber for protecting user trajectory, which adds perturbations to the trajectory before transferring it to other components.
- **Applications (Apps).** The VR applications that provide VR services for the user. In this thesis, we adopt a VR application that provides 3D virtual worlds for VR users to explore, and renders the scene views with the user's poses.
- **Compensator.** Compensator warps the noisy views and transforms them into compensated views as similar to the original views as possible.

The VR user's poses are tracked and fed into our disturber for perturbation. The disturber changes the poses into perturbed poses and transfers them to the VR application. After the perturbation, the application uses the perturbed poses to generate the perturbed views for the VR user. The application only renders the views by using the input poses in our case, however, for the VR applications in reality, there may be other actions depending on the application. The perturbed views are then sent to the compensator for the scene compensation. The compensator takes the perturbed views, perturbed and original poses, and the camera setting, such as the focal length, depth range, and principle point, to transform the perturbed views into the compensated views. The compensated views are then sent back to the VR user. Note that both the disturber and the compensator need to be implemented at the place that users trust, and hence can access the original poses.

6.2 Trajectory Disturber

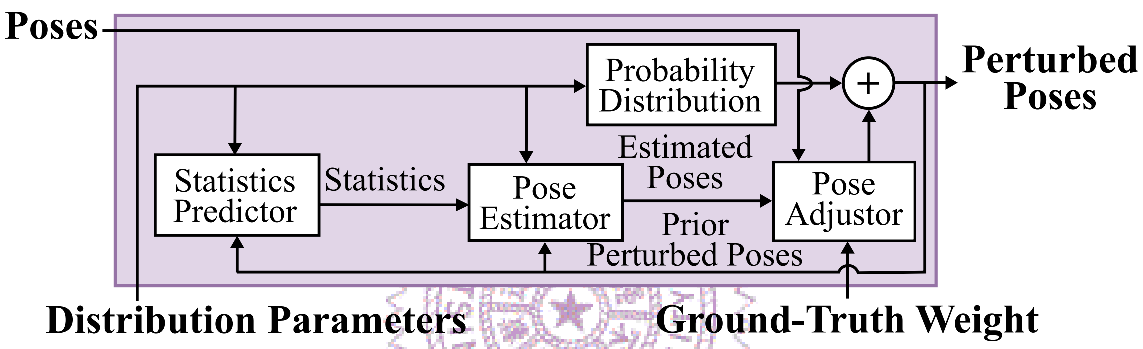


Figure 6.2: Our proposed trajectory disturber.

Design objectives. The purpose of the disturber is to perturb a VR user's trajectory in both the temporal and spatial domains on-the-fly. Doing this in the differential privacy framework is *inherently* challenging because the disturber has no knowledge of the required statistics from the whole trajectory, as some poses happen in the future. Therefore, we need to predict the statistics of the whole trajectory based on prior poses. Another challenge is to find a *good* tradeoff between the incurred perturbations and the degraded visual quality. Hence, we introduce two system parameters as control knobs for VR users to exercise the tradeoff.

Overview. Fig. 6.2 gives the design of our disturber, which sequentially takes *poses* as input and generates a series of *perturbed poses*. Two system parameters: *distribution parameters* and *ground-truth weight* are used to control the distribution and severity of random perturbations. More specifically, the disturber is composed of four components: (i) probability distribution, (ii) statistics predictor, (iii) pose estimator, and (iv) pose adjuster. Here, the *probability distribution* generates random noise that is added to poses as perturbations. Different probability distributions have been adopted in the differential privacy framework, including Laplace, Gaussian, and Binomial [21]. Each probability distribution takes one or multiple distribution parameters like variance. We use the Laplace distribution throughout this paper if not otherwise specified. In addition, the *statistics predictor* predicts the statistics, such as mean and autocorrelation, of the whole trajectory based on prior perturbed poses. These statistics are dictated by the differential privacy framework and are used as input by the *pose estimator*, which estimates the current pose using prior perturbed poses. The estimated pose is inevitably inaccurate compared to the actual one. The *pose adjuster* takes a system parameter called ground-truth weight and fuses the ground-truth and estimates poses. Last, random noise is added to the adjusted pose for the perturbed pose. Among these four components, the probability distribution introduces perturbations in the spatial domain, while the pose estimator does that in the temporal domain.

Notations We next present the operations perturbed with the disturber while developing notations. Each pose or perturbed pose at a moment is a 21-dimension vector containing the locations and orientations of an HMD and two controllers. Each dimension is perturbed independently, and we omit some mathematical detail for brevity in the following. We denote the (input) pose and (output) perturbed pose at time t as V_t and P_t, respectively. The poses and perturbed poses over a time duration with T samples are denoted as $\{V_t\}_{t=1}^{T}$ and $\{P_t\}_{t=1}^{T}$. We consider three statistics of the whole trajectory: mean μ, variance σ^2, and autocorrelation ρ. At time t, these statistics are predicted by the statistics predictor given the Laplace distribution parameter var and prior perturbed poses, $\{P_1, P_2, \ldots, P_{t-1}\}$. These statistics are fed into the pose estimator for an estimated pose $\hat{V_t}$. $\hat{V_t}$ and V_t are passed to the pose adjuster to be fused into an adjusted pose $A(\hat{V_t})$ with the ground-truth-weight w. $A(\hat{V_t})$ incorporates the temporal-domain perturbations. At time t, a random noise $n_t \sim L(0, var)$ is used to generate the spatial-domain perturbations, where $L(0, var)$ is a zero-mean Laplace random variable. Last, we sum $A(\hat{V_t})$ and n_t up to get P_t.

Procedure. We assume that each dimension of a VR trajectory defined on the same probability space, for example, the location x, y, and z are three dimensions defined on the same probability space, is a random variable from a Gaussian distribution. The VR trajectory with all the dimensions together is a *stationary sequence*, which is a random sequence whose joint probability distribution is invariant over time. Therefore, the trajectory can be modeled using the classic linear model, which leverages mathematics and statistics to model the time series data. Many linear models have been proposed, such as the AutoRegressive (AR), Moving Average (MA), and AutoRegressive Moving Average (ARMA) models [72,75]. Among them, we choose to model each trajectory as a Gaussian AR process [135], that is,

$$V_t = \alpha + \rho V_{t-1} + U_t, t \geq 1. \tag{6.1}$$

$V_0 \overset{\text{iid}}{\sim} \mathcal{N}(\mu, {\sigma_v}^2)$, ρ is the autocorrelation of $\{V_t\}_{t=1}^{T}$, and U_t is a white noise. If $\{V_t\}_{t=1}^{T}$ is a stationary Markov process, then $V_t \sim \mathcal{N}(\mu, {\sigma_v}^2)$, and $\alpha = \mu(1-\rho)$, where $\mu = 0$. We use *Linear Minimum Mean Squared Error (LMMSE)* to determine the model parameters α and ρ for the estimated function. As mentioned in Chapter 2, the LMMSE of a linear function is $(1-\rho_{XY}^2){\sigma_X}^2$. Therefore, considering the Gaussian AR, the LMMSE estimation of $\hat{V}_t$ given V_{t-1} is:

$$\hat{V}_t = \mu(1-\rho) + \rho V_{t-1}. \tag{6.2}$$

If we use the prior perturbed pose $P_{t-1} = A(\hat{V}_{t-1}) + n_t$ to estimate the current pose following Zhang et al. [144], then

$$\hat{V}_t = \mu(1-\rho\frac{\sigma^2}{\sigma^2+var}) + \rho\frac{\sigma^2}{\sigma^2+var}P_{t-1}. \tag{6.3}$$

where P_{t-1} is the preceding (input) pose, var is the distribution parameter, and μ, σ, ρ are statistics of the whole trajectory. Unfortunately, μ, σ, and ρ cannot be computed at time t, and we let $\hat{\mu}_{t-1}$, $\hat{\sigma}_{t-1}$, and $\hat{\rho}_{t-1}$ be the predicted statistics. Among them, $\hat{\mu}_{t-1}$ and $\hat{\sigma}_{t-1}$ can be calculated using $\{P_1, P_2, \ldots, P_{t-1}\}$ and $\hat{\rho}_{t-1}$ can be computed following Huitema and McKean [53]. Applying the predicted statistics to Eq. (6.3), we have:

$$\hat{V}_t = \hat{\mu}_{t-1}(1-\hat{\rho}_{t-1}\frac{\hat{\sigma}_{t-1}^2}{\hat{\sigma}_{t-1}^2+var}) + \hat{\rho}_{t-1}\frac{\hat{\sigma}_{t-1}^2}{\hat{\sigma}_{t-1}^2+var}P_{t-1}. \tag{6.4}$$

We also follow Zhang et al. [144] to compute adjusted pose $A(\hat{V}_t)$ as the following weighted sum:

$$A(\hat{V}_t) = (1-w)\hat{V}_t + wV_t. \tag{6.5}$$

Note that by keeping w secret, we create additional burdens to attackers. With the output of the probability distribution, $n_t \sim L(0, var)$, we write the perturbed pose as:

$$P_t = A(\hat{V}_t) + n_t = (1-w)\hat{V}_t + wV_t + n_t, \tag{6.6}$$

This concludes our disturber design, which satisfies the two design objectives.

6.3 Perturbed View Compensation

Design objectives. The purpose of the view compensator is to warp each RGB-D image rendered by a VR application with the perturbed pose P_t to an RGB image viewed at the (original) pose V_t. There exist two design objectives for the compensator: (i) high visual quality and (ii) short execution time. Fortunately, image warping [76], a.k.a. view synthesis [8, 9, 34, 56, 64, 85, 86] is a fairly mature technique. Adding to that, the perturbations imposed on trajectories are rather small and controllable (via var and w in our proposed disturber). Hence, our job is to find a fast enough *synthesizer* achieving reasonable visual quality.

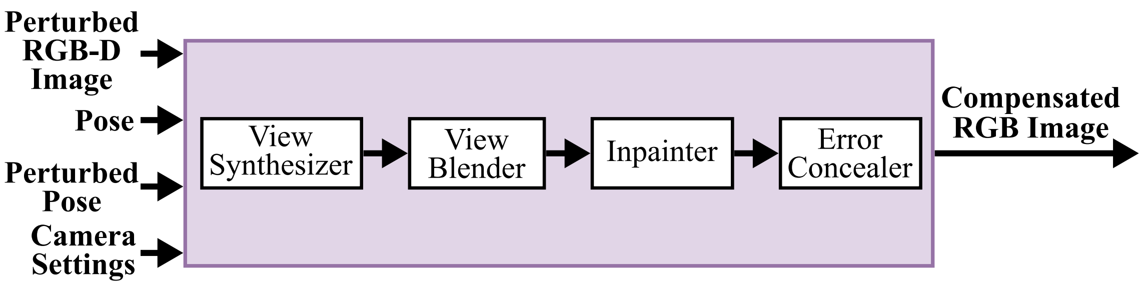


Figure 6.3: Our proposed view compensator.

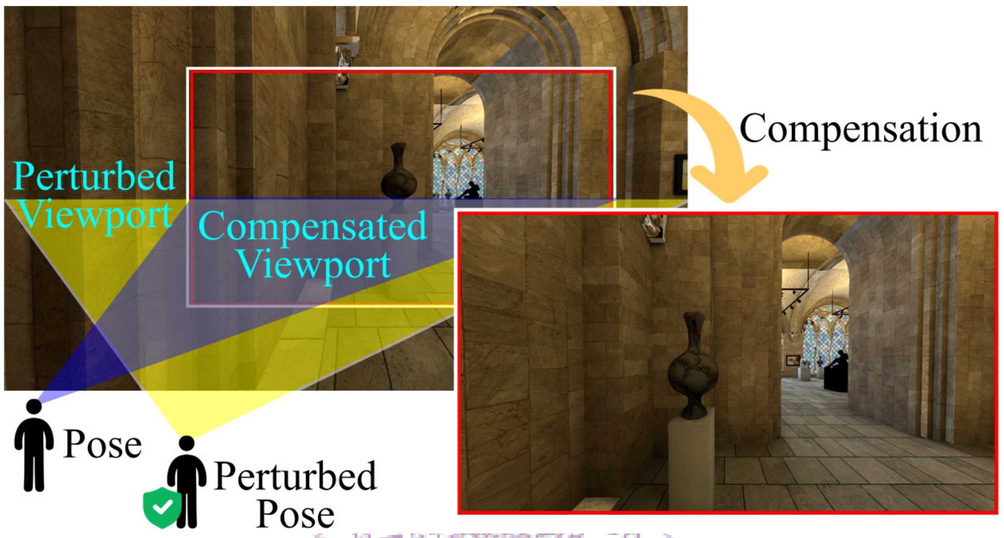


Figure 6.4: Sample view compensation to avoid shaky views due to perturbations.

Design choices. View synthesizers warp pixels from one or multiple input RGB-D images to a VR user's HMD viewport based on the D (depth) channel. There are quite a few Depth Image-Based Rendering (DIBR)-based view synthesizers [34], such as View Synthesis Reference Software (VSRS) [114], Versatile View Synthesizer (VVS) [56], Reference View Synthesizer (RVS) [64], and View Weighting Synthesizer (VWS) [108]. There are also neural-network-based view synthesizers [8, 9, 85, 86]. Between these two types of synthesizers, neural-network-based ones typically run much slower, taking seconds, if not minutes, to synthesize a viewport. Following Fachada et al. [30] and Sun et al. [120], we built our compensator on RVS [64] for a good tradeoff between visual quality and execution time, while other synthesizers can be easily dropped in if needed. We note that our compensator only uses one input RGB-D image, while most synthesizers can take multiple input images, for potentially better visual quality.

Overview. Fig. 6.3 gives the high-level workflow of our compensator. It takes the following inputs: (i) perturbed RGB-D image from the rendering engine, (ii) (original) pose, (iii) perturbed pose, and (iv) camera settings. The key camera settings are the two *Field-of-Views (FoVs)* of the *perturbed* and *compensated viewports*. Fig. 6.4 reveals the relation between these two viewports. To maintain the quality of compensated viewport, the perturbed FoV *must* be larger than compensated FoV, so that most pixels in the compensated viewport fall in the perturbed viewport. In our compensator, we employ the same resolution of input and output RGB(-D) images, denoted as $W \times H$. We let θ_p and θ_c be the vertical FoVs of perturbed and compensated viewports, respectively. Their horizontal FoVs are calculated by the vertical FoVs with a constant aspect ratio $\frac{W}{H}$. Our compensator produces compensated RGB images for VR users' HMDs.

Procedure. The compensator consists of four components: view synthesizer, view blender, inpainter, and error concealer. The *view synthesizer* warps input RGB-D images utilizing the computed disparity. The *view blender* blends wrapped pixels from multiple input RGB-D images based on per-pixel quality levels. It then determines the final pixel value for the synthesized RGB image. The next two components handle the exceptions. In particular, the *inpainter* fills in the blank pixels that are disoccluded in the input RGB-D images by interpolating with the surrounding pixels. Last, the *error concealer* deals with missing output RGB images by replaying the preceding successfully-compensated RGB image. Missing output RGB images could happen when the compensated viewport falls outside of the perturbed viewport, possibly caused by large perturbations. Another possibility is that input RGB-D images from rendering engines contain incomplete depth values in complex 3D scenes.

HUA UNIVERSITY

6.4 Implementations

Disturber. We implemented the disturber in Python. For coordinate transformations between quaternions and Euler angles, we use scipy [110]. For the random noise generation, we adopt numpy [97]. We create the perturbed trajectories with Python in advance for experiments. We also port our code to a Unity [129] project with version 2021.3.11f1. In that project, we install Math.NET Numerics [78] from the NuGetForUnity [80] package for probability distributions. Math.NET Numerics provides a namespace called `MathNet.Numerics.Distributions` [79], which contains many methods for computations related to different kinds of distributions, such as Laplace and Normal. We use the `Normal` type to implement the Gaussian noise sampler, and the `Laplace` type to implement the Laplace noise sampler. Porting the disturber to a Unity project makes it possible to execute it in real-time and is compatible with most VR applications since many of them are implemented with Unity.

Compensator. We also implemented the compensator using MPEG-I's RVS reference software [29], which is written in C++. We employ Unity [129] to render RGB-D images from perturbed poses, and FFmpeg [128] to convert the image format from PNG to YUV for RVS. Like other MPEG reference software, RVS is highly configurable, and thus we develop Python scripts to generate configuration files for RVS to specify camera settings, such as their locations, orientation, focal length, the paths of the input perturbed RGBD images and the output compensated images, the original pose, etc.

TSING HUA UNIVERSITY

Chapter 7

Evaluations

In this chapter, we conduct experiments to evaluate our proposed privacy-threat mitigation method. Two analyses are instrumented: the re-identification attack is used to evaluate the privacy-protecting level, and the visual quality assessment is for evaluating the quality of the rendered views after perturbation. A small-scale user study is designed to realize the true impact of the perturbation and the compensation to users. We then discuss the results of the experiments.

7.1 Re-identification Attack

We implemented a Random Forest (RF) based re-identification classifier to evaluate the protection provided by different mitigation solutions. We select 75 features from prior-arts on identification and authentication in VR applications [2, 70, 77, 88, 89, 98, 101], which include: (i) velocity and angular velocity of each HMD and its controllers, (ii) the minimum, average, and maximum distances between each HMD and its controller, (iii) the minimal, mean, and maximal locations/orientation of each HMD and its controllers, among others. We consider trajectories with longer than 2000 poses in our experiments (about 1/24 of trajectories were removed) and apply a 50-pose sliding window (equivalent to 1 s, as the dataset was collected at 50 Hz). By doing so, each trajectory can be turned into 1956 feature vectors, which are divided into 5 folds for cross-validation. Through some pilot tests, we found the best number of estimators to be 150 and the maximum depth to be 15. We use these two hyperparameters throughout the experiments and result in the average re-identification (re-id) rate across the 5 experiments. We run all experiments on a workstation with an Intel i9-9920X CPU, 64 GB RAM, and an NVIDIA GeForce RTX 3080 Ti GPU. We report the average performance results with 95% confidence intervals whenever possible.

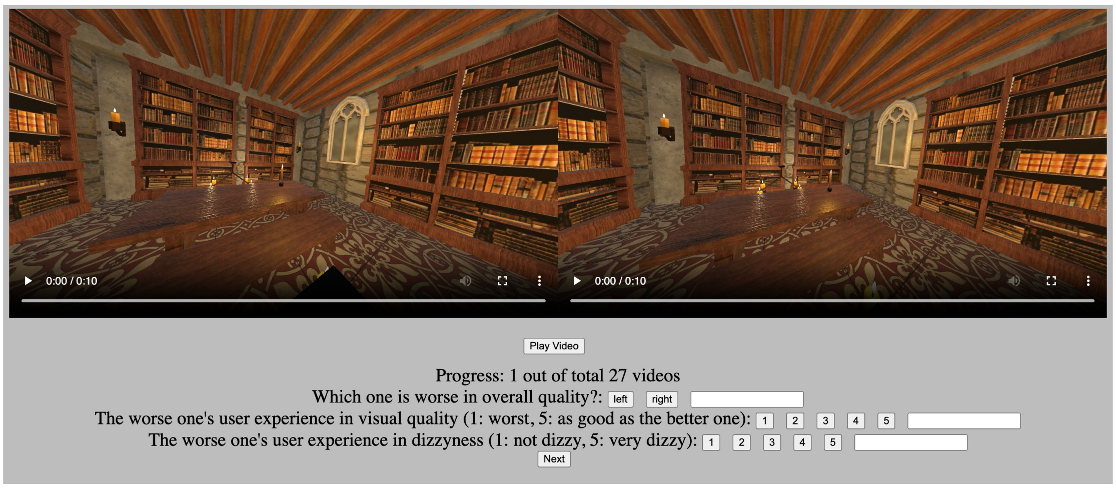
7.2 Visual Quality Assessment

We employ a 6DoF VR dataset [137] that collects 24 VR users' trajectories in four 3D virtual scenes to drive our experiments. We compare two of our proposed solutions: *Disturber Only (DO)* and *Disturber with Compensator (DC)*, against the state-of-the-art MetaGuard (MG) [93]. Notice that MetaGuard adds perturbations to multiple personal attributes (rather than trajectories directly), such as user height and gender, and then projects the perturbed attributes back to perturbed locations (rather than trajectories). They use *Bounded Laplace Mechanism* [23] to sample the random perturbations. Bounded Laplace Mechanism limits the amount of the sampled perturbation to bound the original data within a reasonable scope after adding the perturbation. For fair comparisons, we also add perturbations to locations only if not otherwise specified. In MetaGuard, we consider attributes that are most relevant to VR user locations, i.e., height, room, squat depth, wingspan, and arm length, and apply $\epsilon \in \{0.000001, 1, 5, 10, 20, 30, 50\}$ to these features. For our solution, we first determined the amount of the perturbation for each trajectory by the percentage $n\%$ of the perturbation in the whole trajectory. Since the dataset was collected in a 4×4 area when a user stands in the middle of the area, the maximum movements for each dimension (x, y, and z) are 2. Therefore, the var of the sampling probability distribution is $2 \times n\%$. We further transform the var to ϵ with $\epsilon = \delta f / var$, where δf is the sensitivity of the trajectory in each dimension. The δf here is 4 since the dataset is collected in a 4×4 area, and then the maximum difference for two poses in the same dimension is 4. We select $n \in \{1, 2, 4, 6, 10, 30, 60\}$ based on a prior test to roughly align the resulting visual quality with MetaGuard. After transforming, we have $\epsilon \in \{$3.33, 6.67, 20, 33.33, 50, 100, 200$\}$. Our solutions take a few additional parameters: (i) weight $w \in \{0.1, \mathbf{0.3}\}$, (ii) perturbed FoV $\theta_p \in \{\mathbf{115}°, 125°\}$, and (iii) compensated FoV $\theta_c = \mathbf{104}°$, where bold font indicates default values.

We consider three popular visual quality metrics: Peak Signal-to-Noise Ratio (PSNR) [47], Structural Similarity (SSIM) [47], and Video Multimethod Assessment Fusion (VMAF) [94], by comparing the quality of VR users' HMD viewports from DO, DC, and MG against the ground-truth viewports without perturbations. Computing visual quality for all VR users is time-consuming. Hence, we randomly select six sample users with diverse moving distances from each of the four scenes: City, Gallery, Nature, and Office. The selected users' ids are as follows:

- **City**: $\{$0, 3, 4, 6, 17, 18$\}$
- **Gallery**: $\{$3, 7, 8, 9, 17, 22$\}$
- **Nature:** $\{$1, 3, 6, 7, 9, 17$\}$
- **Office**: $\{$6, 9, 11, 13, 20, 21$\}$

7.3 User Study Design



Progress: 1 out of total 27 videos
Which one is worse in overall quality?: left right
The worse one's user experience in visual quality (1: worst, 5: as good as the better one): 1 2 3 4 5
The worse one's user experience in dizzyness (1: not dizzy, 5: very dizzy): 1 2 3 4 5

Figure 7.1: The sample viewport pair of the Office scene.

Last, we conduct a small-scale user study to qualify the effectiveness of our compensator on visual quality and dizziness with eight subjects. We chose two 3D scenes, Nature and Office for the outdoor and indoor environments. We chose the values of ϵ by aligning the re-id rates of MG and DO since we are curious about the user experience under the same privacy protection level for the two algorithms. We select $\epsilon \in \{0.000001, 5, 10\}$ for MG, and $\epsilon \in \{33.33, 50, 100\}$ for DO. We use a MacBook Pro laptop equipped with a 13.3-inch (diagonal) LED-backlit Retina display with IPS technology, 2560 native resolution at 227 pixels per inch, and which supports millions of colors. We play the ground-truth viewport along with impaired viewports, which came from either MG, DO or DC, as shown in Fig. reffig:userStudyVP. The placement of the two viewports is random. After watching each pair of viewports, we ask each subject the following questions about their user experience:

- Which viewport is worse in terms of overall quality?
- How would you rate the worst viewport's user experience in visual quality? Between 1 (Unacceptable) and 5 (as good as the better one).
- How is the dizziness when you watch the worst viewport? 1 (none) to 5 (severe).

We filter out outlier scores by checking the answers to the first questions: about 15% of them were dropped. We report the scores in Sec. 7.4.

7.4 Results

Our disturber leads to lower re-identification rates. Fig. 7.2 shows the re-id rates of DO and MG under different ϵ values. The re-id rate without any privacy-threat mitigation

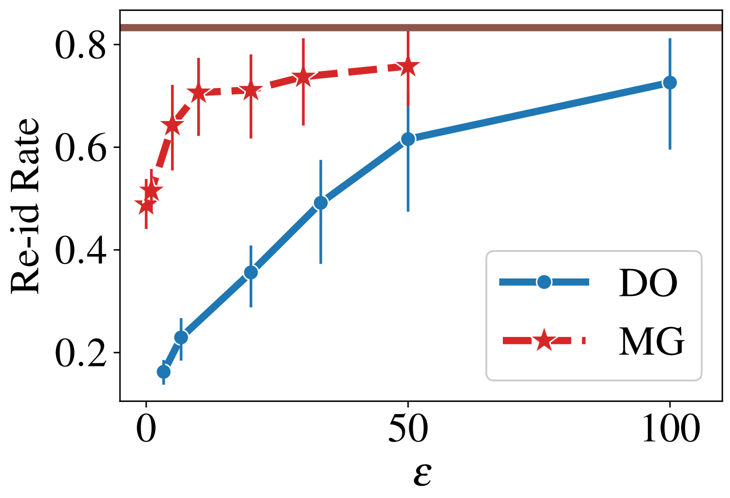
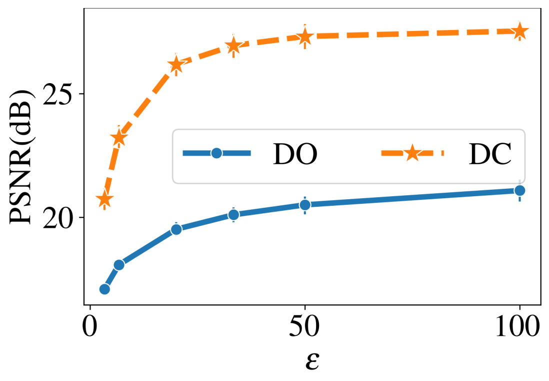


Figure 7.2: Re-identification rate under different ϵ for DO and MG.

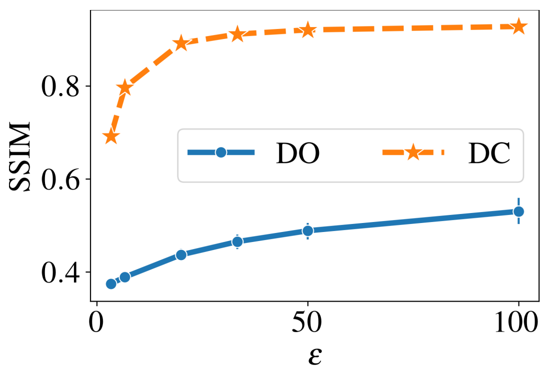
清華大學
TSING HUA UNIVERSITY

is 0.83, shown as the horizontal dashed line in the figure. Compared to the state-of-the-art MG, our DO reduces the re-id rate by up to 0.4. Such improvement may be attributed to the fact that the re-identification classifier considers temporal correlation in trajectories, which is not protected by MG. *We conclude that perturbing the trajectories in the temporal domain preserves more user privacy.* Moreover, when ϵ approaches 0, the re-id rate of DO also approaches 0.1; in contrast, the re-id rate of MG is still above 0.45. This is because MG leverages the Bounded Laplace Mechanism to sample the random perturbation, and the amount of the perturbation is limited to preserve the visual quality. Our system does not need to limit the perturbation amount to preserve the visual quality with our compensator. *Therefore, we conclude that our disturber can protect users' privacy more efficiently.* Additionally, the uneven curve of MG results may be because the perturbations are first sampled for the attributes rather than for the trajectory.

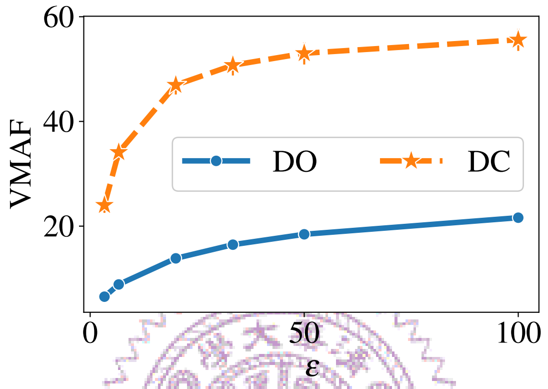
Our compensator mitigates the degradation of visual quality due to perturbation.



(a)



(b)



(c)

Figure 7.3: Visual quality with and without compensation.

Fig. 7.3 compares the overall visual quality from DO and DC across all four 3D scenes under different ϵ values. We observe that our compensator improves the visual quality by 6.8 dB at most and 5.9 dB on average, and the SSIM and VMAF boosts are up to 0.4 and 32, respectively, under the same value of ϵ. *Hence, we conclude that our compensator successfully improves the degraded visual quality while providing strong privacy protection.*

Our solution provides strong protection while delivering good visual quality.

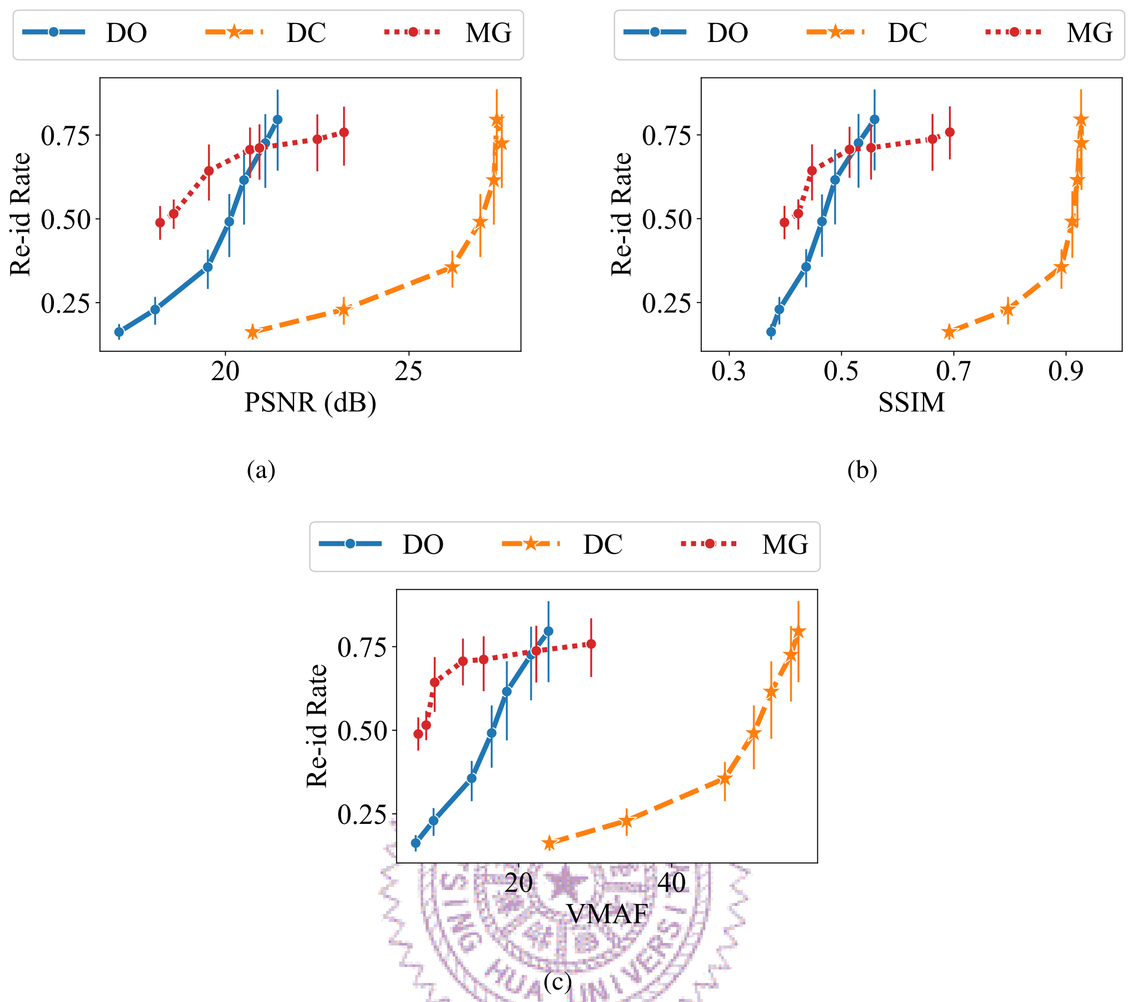
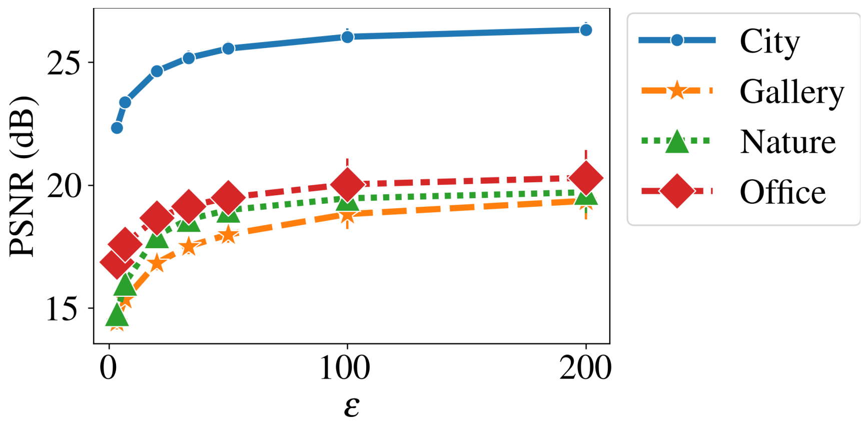


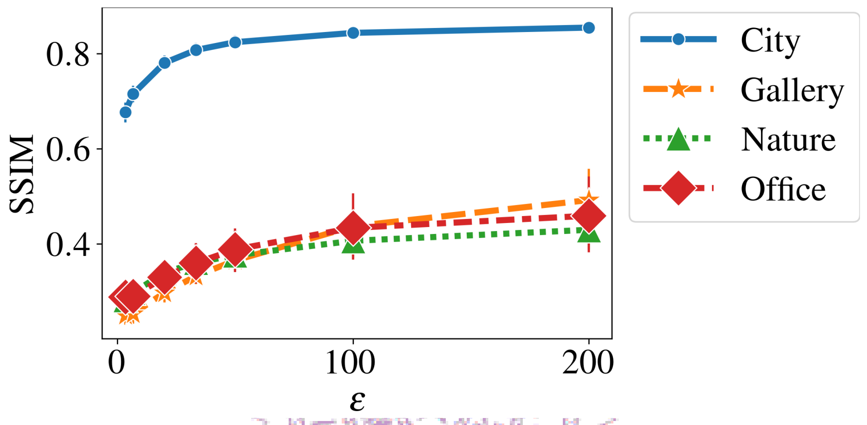
Figure 7.4: Privacy-quality tradeoff achieved by different privacy-threat mitigation approaches, with visual quality in: (a) PSNR, (b) SSIM, and (c) VMAF.

We plot the *privacy-quality tradeoff* in Fig. 7.4, which maps the privacy level to the visual quality. According to Fig. 7.4, DO achieves better visual quality than MG when the re-id rate is between 0.10 and 0.73. Even though MG delivers better visual quality when the re-id rate is > 0.73, MG fails to mitigate the threats at such a high rate. In fact, DO lowers the re-id rate by almost half compared to MG under the same visual quality. We also observe that with compensation, DC achieves strong privacy protection while achieving very good visual quality. Compared to DO, DC improves by up to 6.83 dB in PSNR, 0.45 in SSIM, and 34.57 in VMAF at the same re-id rate. *The tradeoff curves reported here validate the excellent privacy-quality tradeoff of our solutions.*

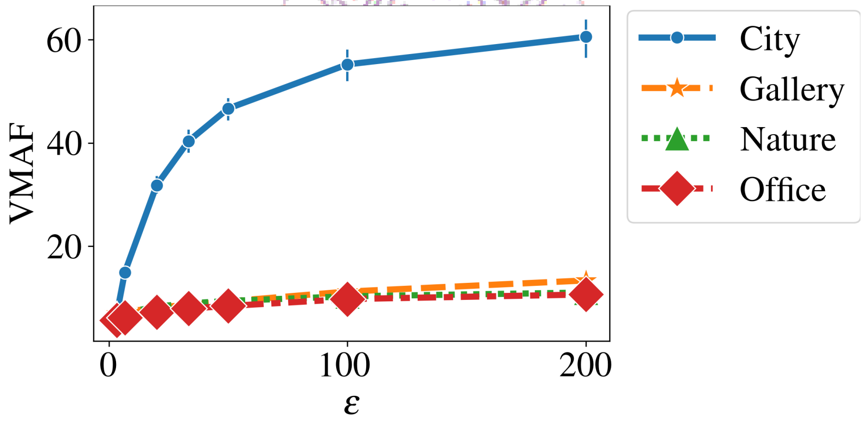
Implications of diverse characteristics of 3D scenes.



(a)



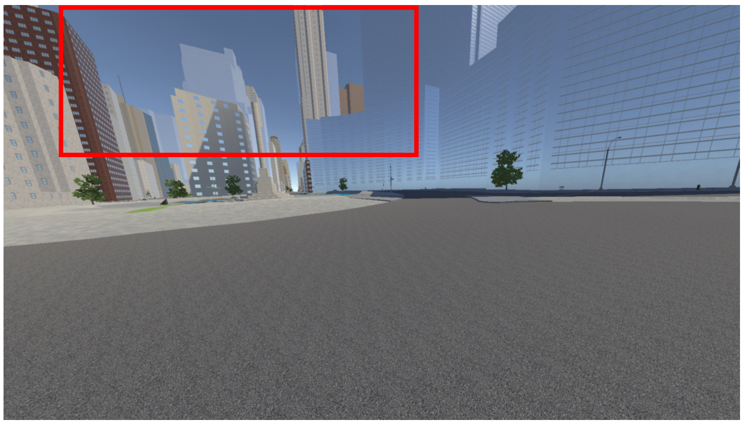
(b)



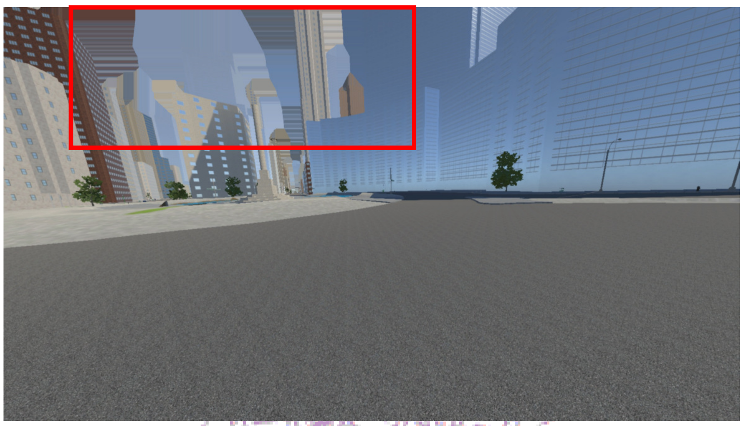
(c)

Figure 7.5: Visual quality comparison of the four scenes under different ϵ with DO: (a) PSNR, (b) SSIM, and (c) VMAF.

Fig. 7.5 shows the visual quality of the four scenes under different values of ϵ. Among the four scenes, the visual quality of City outperforms the others at all time. This may be caused by the size of City (128×50×128 m^3) being considerably larger than the others. Moreover, City is a vast scene with wide roads and a large sky, for which the adjacent pixels are less different from each other, as shown in Fig. 7.6(a). Therefore, the perturbed



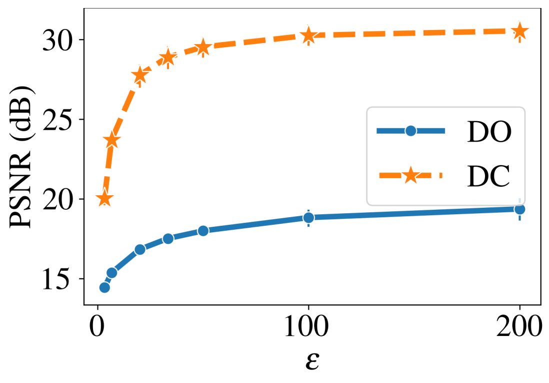
(a)



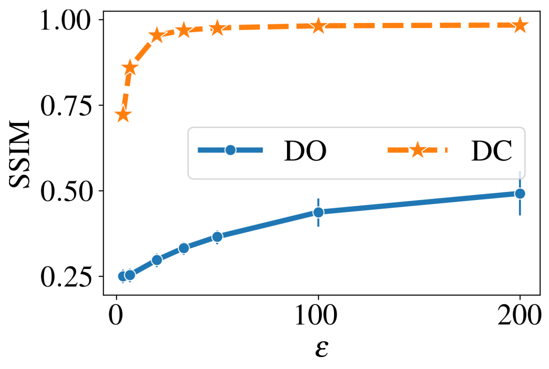
(b)

Figure 7.6: Rendered city building: (a) with and (b) without compensation.

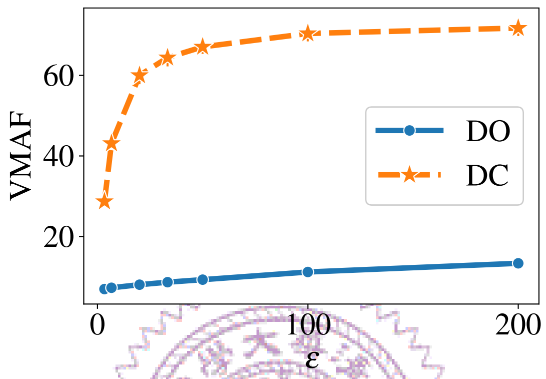
image may not be influenced too much when the poses are changed, which results in relatively better visual quality than the other scenes. However, the quality of all the scenes is improved when DC is applied except for City. Fig. 7.7 shows the quality comparison of DO and DC under different values of ϵ for the Gallery scene, the visual quality after compensation is improved by at least 5 dB in PSNR, 0.5 in SSIM, and 20 in VMAF. The improvement increases steadily with larger ϵ. The same trend of visual quality can be found in Fig. 7.8 for the Nature scene, and Fig. 7.9 for the Office scene. In contrast, Fig. 7.10 shows the quality comparison of DO and DC under different values of ϵ for the City scene. As shown in the figure, applying DC to City lowers the visual quality. More specifically, the performance of DC is slightly enhanced from $\epsilon = 3.33$ to $\epsilon = 50$, and gets worse when the value of ϵ is larger than 50; however, the visual quality of DC is lower than that of DO at all time. Fig. 7.11, Fig. 7.12, and Fig. 7.13 show the quality of each scene with ϵ value equal to 3.33, 50, and 100, respectively, with and w/o compensation. The visual quality of DC is close to DO, and even lower than DO for the City scene, while the visual quality enhanced steadily with DC for the other scenes. This is because there



(a)



(b)



(c)

Figure 7.7: Visual quality comparison of the Gallery scene under different ϵ with DO : (a) PSNR, (b) SSIM, and (c) VMAF.

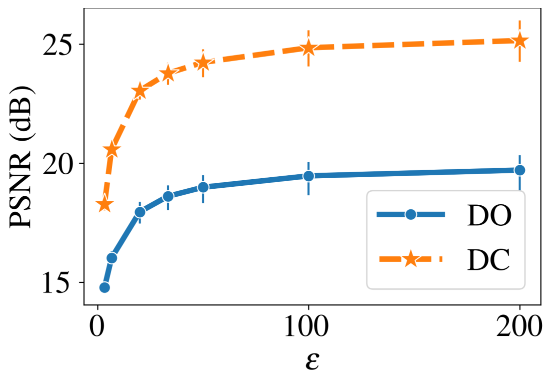
are many buildings with a sky background for the City scene. RVS uses the texture of adjacent pixels for inpainting when there is disocclude. Most of the disoccluded sky areas in City are inpainted with buildings after DC is applied, as shown in Fig. 7.6 with the red box, which degrades the visual quality of the City scene with DC.

Implications of different parameters. Fig. 7.14 shows the re-id rate of different

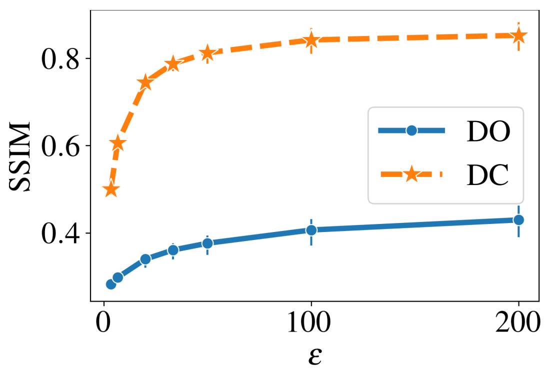
Table 7.1: Average Re-Id Rate with Std

Weight w	**0.1**	**0.3**	**0.5**	**0.7**
Re-id Rate	$0.419\ (\pm 0.259)$	$0.483\ (\pm 0.253)$	$0.493\ (\pm 0.249)$	$0.495\ (\pm 0.244)$

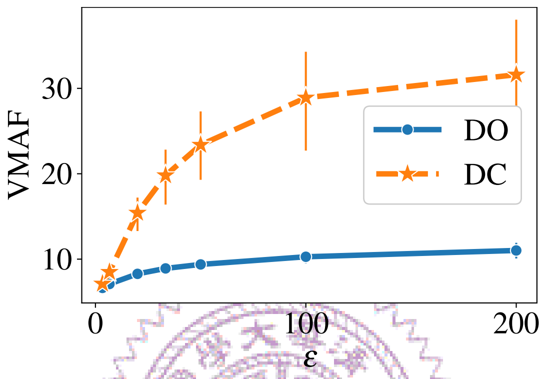
w under different ϵ. The re-id rates of $w = 0.1$ are much lower than those of the other w. Though the re-id rates of $\{0.3, 0.5, 0.7\}$ weight setting are almost the same in the figure, we can observe slight differences between them according to Table 7.1. Table 7.1 compares the re-id rate of $\{0.1, 0.3, 0.5, 0.7\}$ weight setting. According to the table, the re-id rate is reduced when w is decreased, and is reduced by 0.06 on average between w equals 0.1 and 0.3. Therefore, lower w can lead to a lower re-id rate and protect more privacy. On the other hand, Fig. 7.15 shows the visual quality of different weight settings



(a)



(b)



(c)

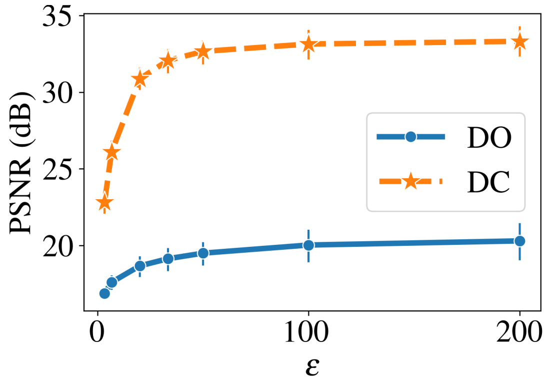
Figure 7.8: Visual quality comparison of the Nature scene under different ϵ with DO: (a) PSNR, (b) SSIM, and (c) VMAF.

under different ϵ. The lower w leads to lower visual quality, but the decrease is less than 2 dB in PSNR, 0.1 in SSIM, and 5 in VMAF. We also tune the θ_p to see if larger perturbed FoV can improve the compensation. However, according to Table 7.2, larger θ_p does not lead to better performance of compensation. This may be caused by the way RVS handles the inpainting that we discussed above. Moreover, in a smaller scene like Office, larger rendered FoV may only include more pixels far from the compensated viewport, which is ineffectual for compensation.

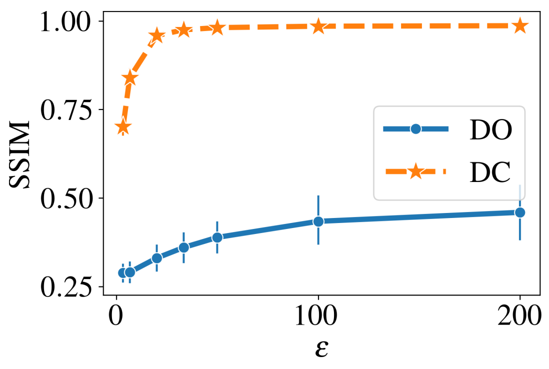
Our compensator successfully improves user experience. Fig. 7.17 shows that overall, after applying for compensation, the subjects' visual quality and dizziness scores are both improved. In each of the subfigures, at most one subject reports a worse score

Table 7.2: Average Visual Quality with Std

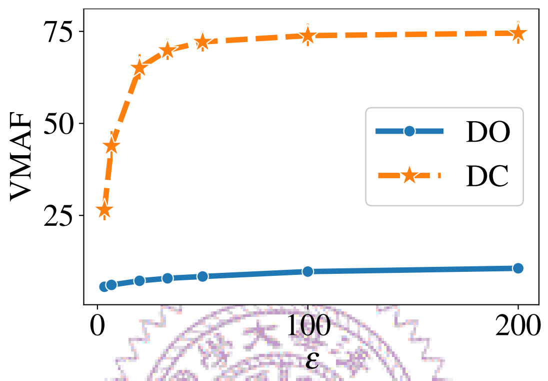
Metric	$\theta_p = 115°$	$125°$
PSNR (dB)	25.62 ($\pm$4.57)	25.39 ($\pm$4.20)
SSIM	0.87 ($\pm$0.12)	0.86 ($\pm$0.12)
VMAF	45.86 ($\pm$21.43)	41.08 ($\pm$19.51)



(a)



(b)



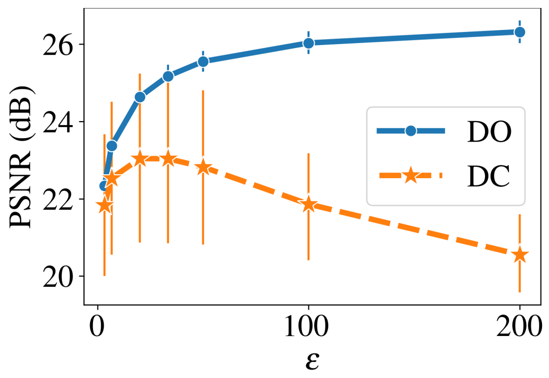
(c)

TSING HUA UNIVERSITY

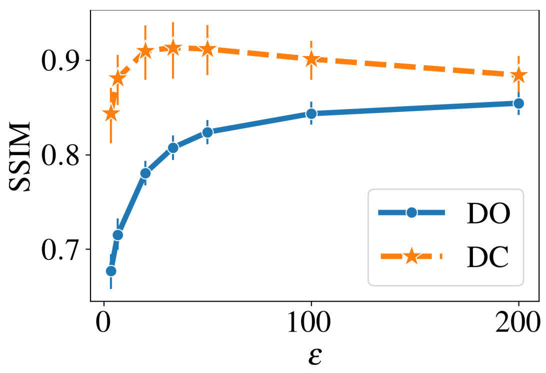
Figure 7.9: Visual quality comparison of the Office scene under different ϵ with DO: (a) PSNR, (b) SSIM, and (c) VMAF.

with DC, compared to DO. *In summary, the mean opinion score of visual quality is increased by 0.6 (out of 5) on average.* However, the user experience of DC is worse compared to MG. Though DC smooths the shaky perturbed views, the compensated views may contain some distortion due to the inpainting, which degrades users' experience. This can be improved by a better inpainting method in the future. Note that the shaky views are caused by the random perturbation sampled from the probability distribution. Without the additive random perturbation, DO can provide smoother views than MG, but less privacy protection.

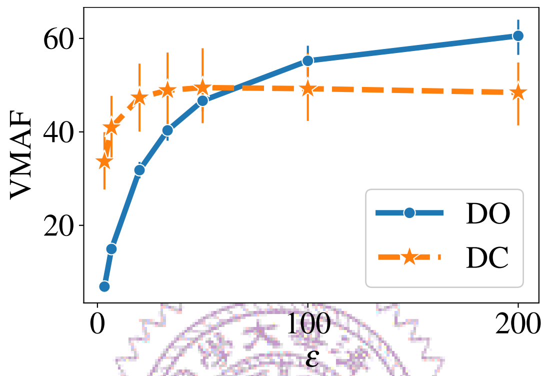
Adding perturbations to the users' locations only is more efficient than adding perturbations on both locations and orientations. We also conducted a small experiment of adding perturbation on both locations and orientations of users' trajectories with $\epsilon = \{100, 66.67, 50, 40\}$. Fig. 7.18 shows the re-id rate with and w/o adding perturbations to the orientations. The re-id rate is reduced at most 0.125 with the perturbations on orientations under the same ϵ. However, the visual quality degrades drastically as shown in Fig. 7.19. Though the visual quality can be enhanced with the compensation, *adding perturbations to the locations only is suggested since it offers strong enough privacy pro-*



(a)



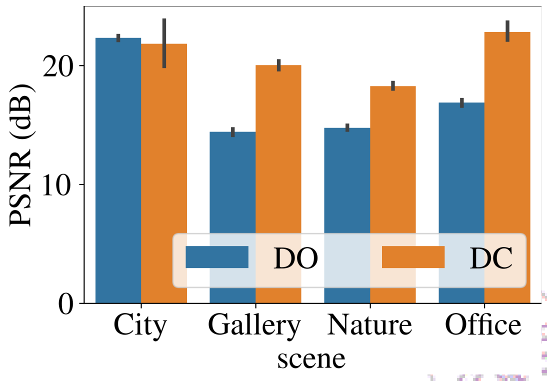
(b)



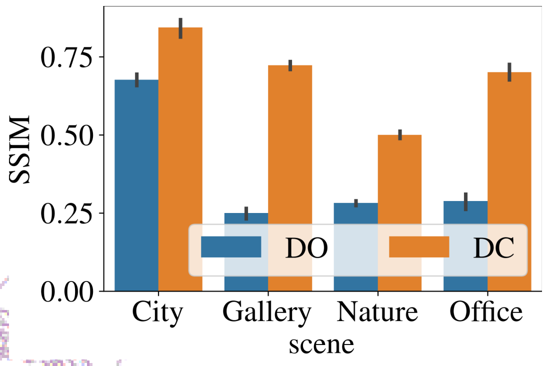
(c)

Figure 7.10: Visual quality comparison of the City scene under different ϵ with DO: (a) PSNR, (b) SSIM, and (c) VMAF.

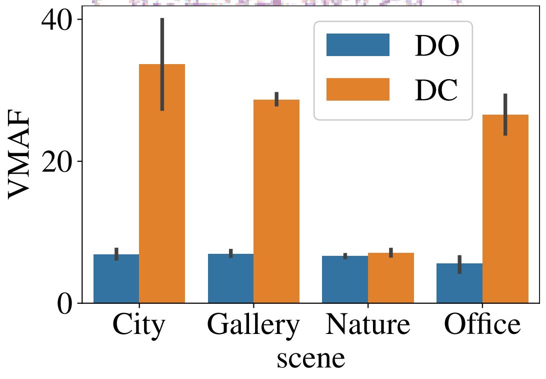
tection while maintaining good visual quality.



(a)

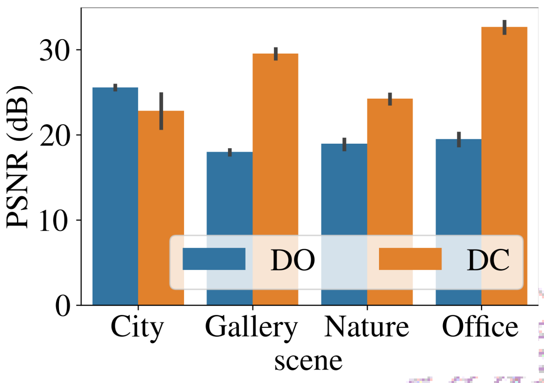


(b)

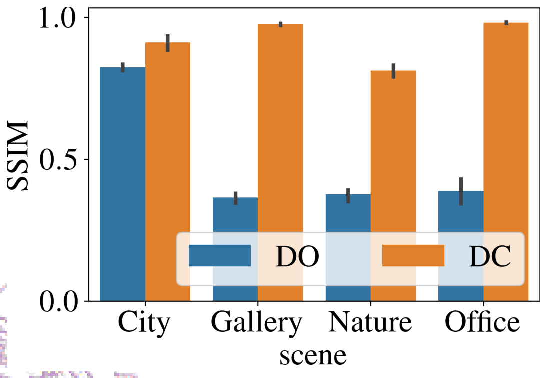


(c)

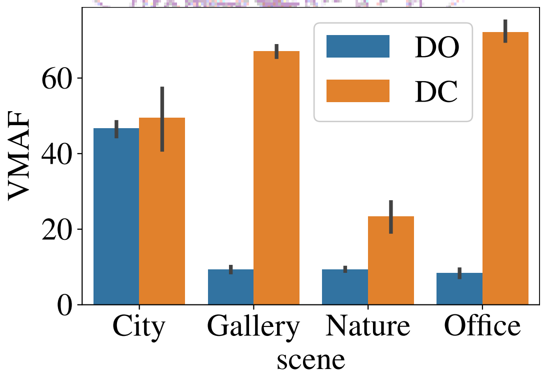
Figure 7.11: Visual quality comparison among the four considered scenes with $\epsilon = 3.33$ using DO: (a) PSNR, (b) SSIM, and (c) VMAF.



(a)



(b)



(c)

Figure 7.12: Visual quality comparison among the four considered scenes with $\epsilon = 50$ using DO: (a) PSNR, (b) SSIM, and (c) VMAF.

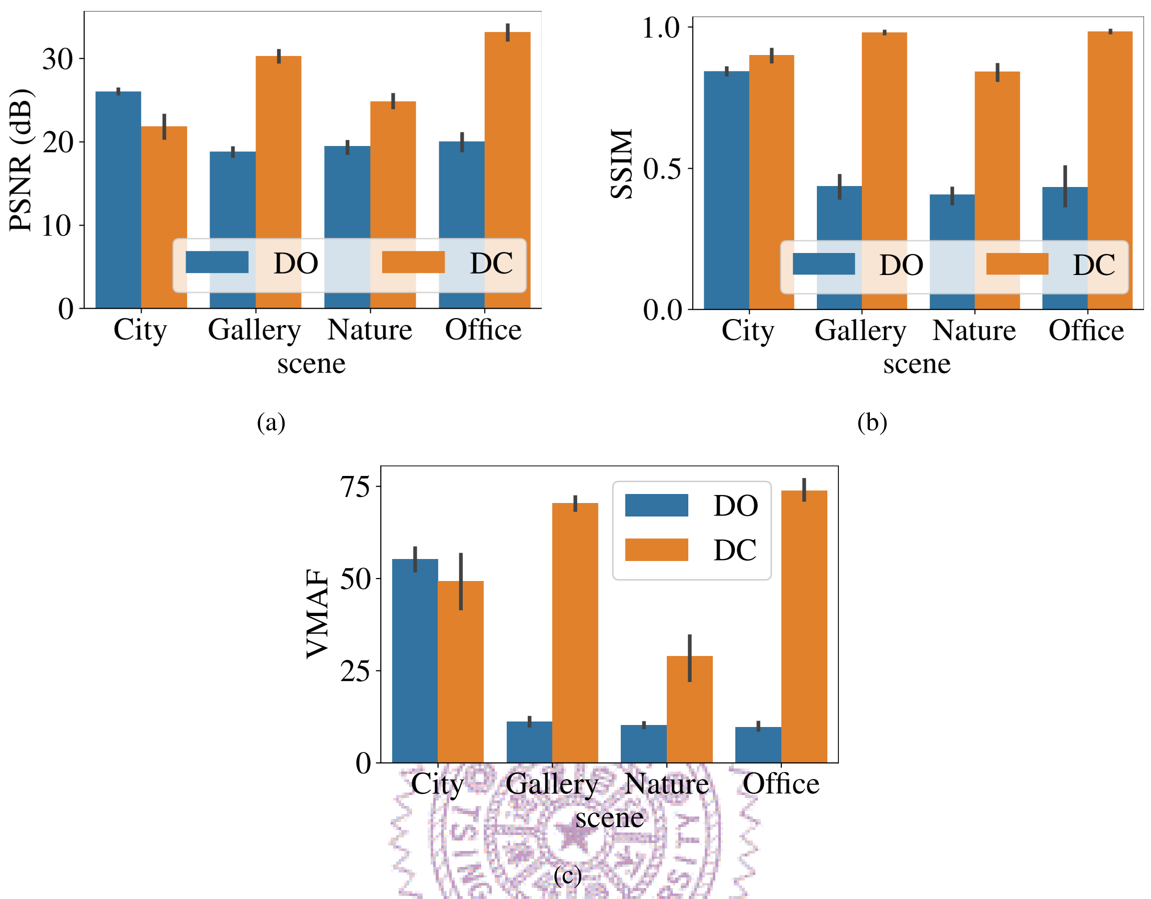


Figure 7.13: Visual quality comparison among the four considered scenes with $\epsilon = 100$ using DO: (a) PSNR, (b) SSIM, and (c) VMAF.

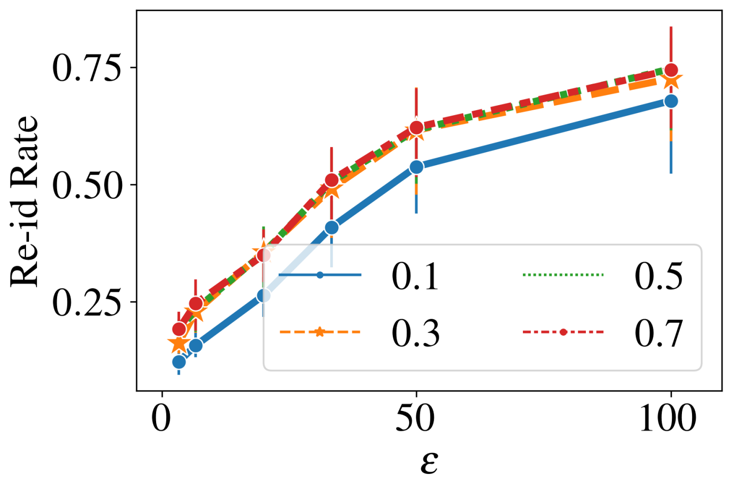
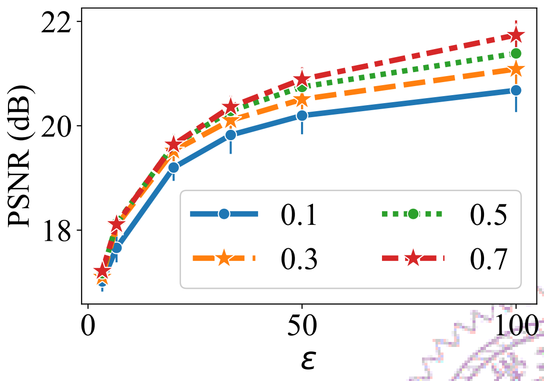
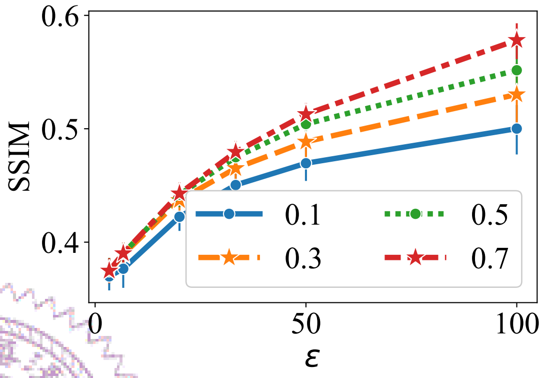


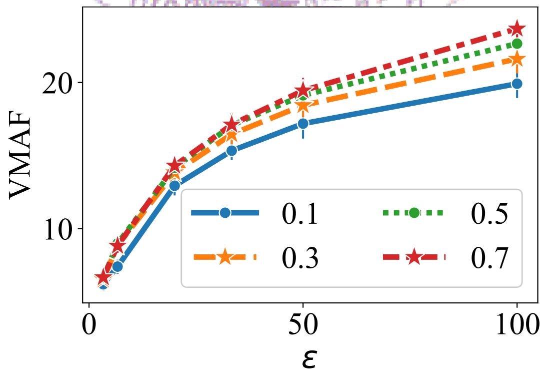
Figure 7.14: Re-identification rate of different w under different ϵ with DO.



(a)

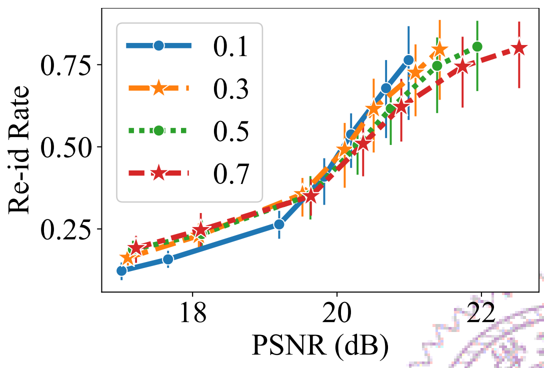


(b)

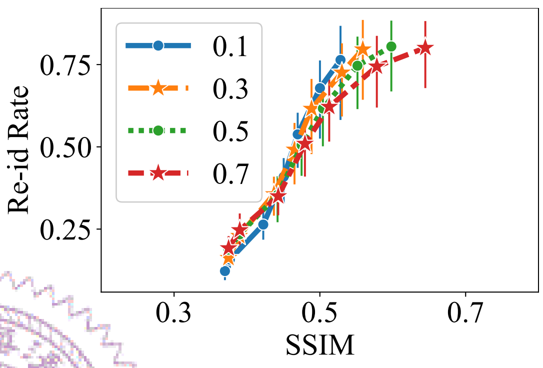


(c)

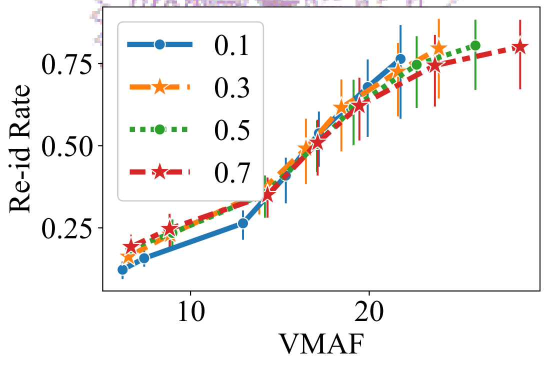
Figure 7.15: Visual quality of our disturber with different w under different ϵ using DO: (a) PSNR, (b) SSIM, and (c) VMAF.



(a)



(b)



(c)

Figure 7.16: Privacy-quality tradeoff achieved by our disturber with different w: (a) PSNR, (b) SSIM, and (c) VMAF.

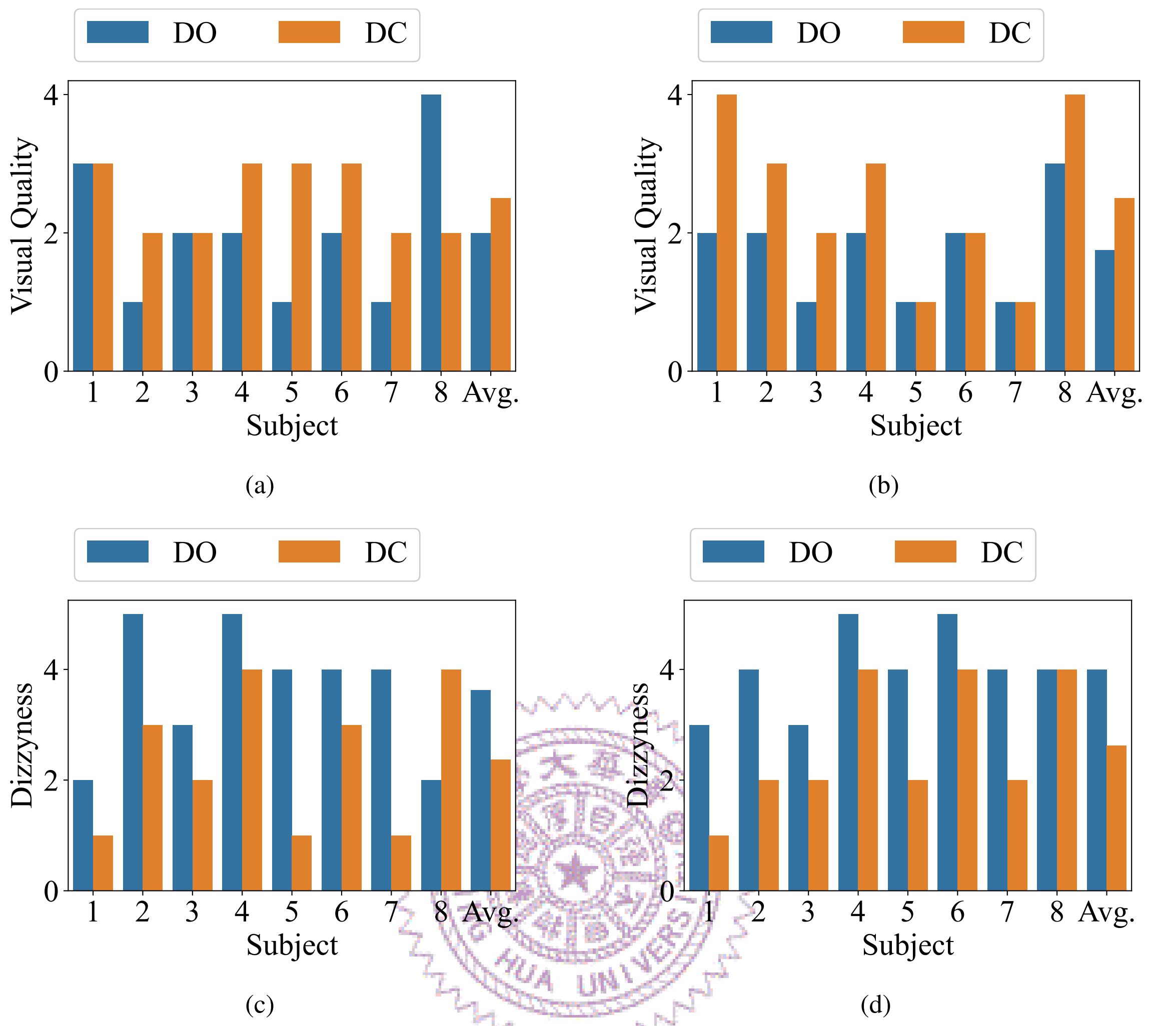


Figure 7.17: User experience scores (a), (c) from Nature and (b), (d) from Office, where (a), (b) give visual quality scores and (c), (d) give dizziness scores.

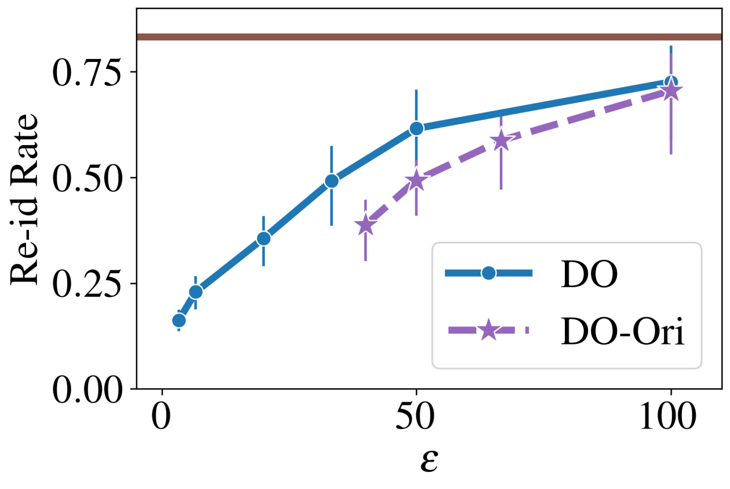


Figure 7.18: Re-identification rate with and w/o adding perturbations to the orientations under different ϵ.

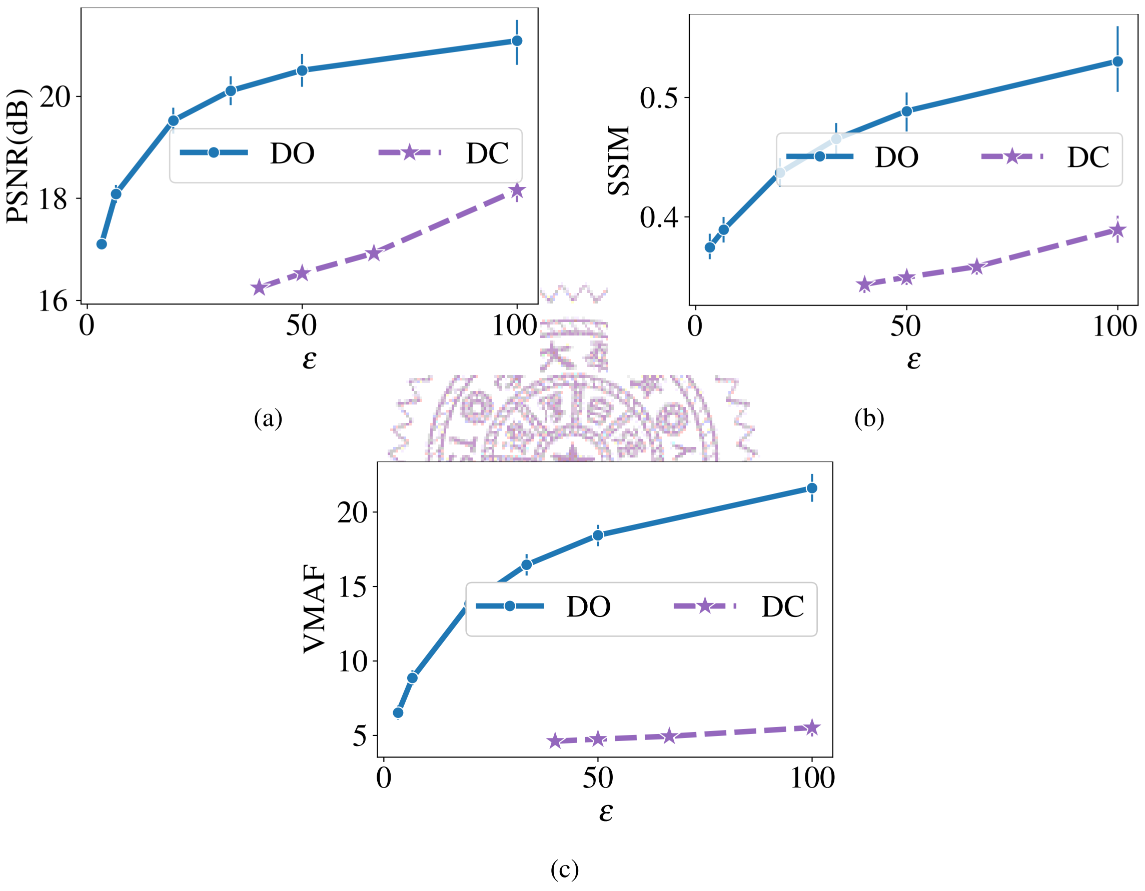


Figure 7.19: Visual quality of our disturber with and w/o adding perturbations to the orientations under different ϵ: (a) PSNR, (b) SSIM, and (c) VMAF.

Chapter 8

Conclusion and Future Work

In this chapter, we first conclude our work and then list the future work to improve our privacy-threat mitigation system.

8.1 Concluding Remarks

In this thesis, we design a collecting procedure to collect a 6DoF VR dataset for 3D virtual worlds to investigate the privacy issues in VR. The dataset includes the basic sensor data in VR, i.e., HMD locations and orientations, controller locations and orientations, key strokes of the controllers, objects locations and orientations, and the privacy-sensitive data, i.e., physical RGBD videos and the personal attributes of the subjects, which prior datasets do not contain and hence fail to do such investigation comprehensively and intensively. With the dataset, we test the feasibility of some proposed privacy threats for VR trajectory. We show that with the dataset, the VR users' height and handedness can be inferred with some trivial heuristic algorithms, and the users' identities can be identified with their VR trajectories, which refers to the re-identification attack. We then add Gaussian perturbations to the trajectories and found that the perturbations can shield the users from the re-id attack. These prior tests inspire the development of our proposed privacy-threat mitigation approach. We then propose a privacy-threat mitigation system to protect VR users with HMDs and controllers. We achieve that in two steps. First, we develop a disturber to add perturbations in both temporal and spatial domains to the time-series poses on-the-fly. The temporal perturbations are added by estimating the current pose based on the previous pose, and the spatial perturbations are sampled from a probability distribution to add on. Next, we build a compensator to warp rendered viewports of perturbed poses to eliminate the shakiness caused by the perturbations. Extensive objective and subjective experiments demonstrate the merits of our disturber and compensator: they achieve an excellent tradeoff between privacy and visual quality. In particular, our evaluation results demonstrate that our disturber alone reduces at most 0.4 re-identification rate compared to the state-of-the-art approach [93] while improving the visual quality by 2.5 dB in PSNR, 0.1 in SSIM, and 10 in VMAF.

8.2 Future Work

This work can be extended in multiple directions:

- **Make the privacy threat more comprehensive.** We assume that attackers know the user identity of part of a trajectory that is collected from the *target 3D virtual world* in the thesis, which is not realistic. Here, a wanted 3D virtual world means the 3D virtual world from which the trajectories that the attackers want to get the users' identities are collected. In reality, attackers may have the whole user identities of the trajectories collected from a group of people exploring one 3D virtual world, and want to get the identities of the trajectories of the same people exploring other target 3D virtual worlds. For example, an attacker has user identities of the trajectories of ten people exploring the City scene, and wants to identify the same ten people from their trajectories exploring the Gallery scene. However, this is challenging due to the different movement and behavior patterns in different kinds of 3D virtual worlds. For example, users may likely move faster in a broader 3D virtual world than a narrow one. These patterns should be investigated and considered when training the re-id model, more specifically, the features that represent these patterns should be extracted from the trajectory. On the other hand, other privacy threats can also be used to evaluate the privacy protection level. For example, one of the threats is to analyze the words or sentences that VR users are typing by using their HMD poses alone [113]. This is one of the most crucial threats since this may reveal users' passwords for VR applications or other privacy-sensitive information that is included in their sentences. Moreover, since there are various types of privacy threats while there are still many potential privacy threats that have not been discovered, it is essential to comprehensively and generally categorize the privacy threats in VR. Based on the categories, we can normalize the proposed privacy-threat mitigation approach to protect VR users from the categorized privacy threats but not only one certain privacy threat, i.e., re-id attack.
- **Extend the privacy-threat mitigation methods for other data types.** In this thesis, we only consider perturbing the locations and orientations of trajectories, though the performance of including the orientations is not that good. Other VR sensor data should also be included and protected. The sensor data that are sequential time-series, such as the eye gaze data, can utilize our system with the current

version. In the future, our proposed privacy-threat mitigation system can be extended to protect other data, such as the images of the users' facial appearances or their whole body while using VR. These kinds of images are sensed by facial trackers or external cameras for VR applications that need to reconstruct users' avatars with their real appearances, or detect the facial expressions and postures. The leakage of these images is much more serious since both the appearance of users and their living environments are discovered by attackers. To protect the image data, the approaches for de-identification of still images can be utilized [106]. For example, trivial approaches like blurring and pixelation images, or advanced approaches like *k-Same* [95] and *k-Same-Select* [43].

- **Determine the optimal input parameters.** There are two input parameters of our disturber, i.e., w and $epsilon$. These two parameters together determine the re-id rate and visual quality, and hence lead to the privacy-quality tradeoff we introduce in Chapter 7. However, though users can abide by the tradeoff curve to tune the parameters, the curve may not be precise enough since it is derived from several sample experience settings. In the future, we can mathematically formulate the tradeoff curve, and estimate the curve with optimization algorithms, and find the optimal input parameters under certain privacy levels and quality levels.
- **Create an end-to-end real-time privacy-threat mitigation system.** In this thesis, we generate the perturbed trajectories at first, then replay the trajectories in the VR application to capture the perturbed images, and then compensate for the perturbed images with the compensator. Though all of these steps are done on-the-fly, they are not integrated as a pipeline. In reality, after a user generates his/her pose, the pose should be perturbed, be used by the application to generate the perturbed view, and the perturbed view is compensated and sent back to the user right away, which means all the operations for one pose should be done together at once rather than finishing one operation for the whole trajectory and then moving on to the other operation. Therefore, the two main components illustrated in Chapter 6 should be integrated as an end-to-end system. This can be done by implementing both components in a Unity project, or implementing them as a Unity plug-in. One of the challenges is that the input image format of the software we use to implement the compensator, which is RVS, needs to be YUV; however, the YUV image is too large and leads to an overlong running time for our system. Therefore, we should solve the problem either by considering other synthesizing software or by implementing the compensator ourselves. Moreover, the system should also serve privacy protection in real time since the VR applications always run in real-time with users' instant actions. Our disturber requires less than 0.0027 seconds on

average to perturb one pose, which is good for a real-time system. However, the compensation requires 1.24 seconds on average to compensate for one perturbed image, which is too long for a real-time system. The duration does not include the running time of generating the required camera config files. Therefore, the main challenge to make the system real-time is the running time of the compensation, which can be further improved with other view synthesizers leveraging GPU [13] in the future.

- **Conduct a user study to model the privacy-QoE tradeoff.** In this thesis, we only consider the visual quality of the rendered views to derive the privacy-quality tradeoff. However, the user experience is not only related to the visual quality, but also other factors such as cybersickness. Therefore, the user QoE should be considered. In the future, the user study can be extended into a large-scale one, with more subjects involved and using a VR application rather than watching the rendered 2D videos. With the result of the user study, we can model the user QoE with our system, and hence derive the privacy-QoE tradeoff.
- **Different approaches and algorithms for implementing the proposed privacy-threat mitigation.** In the thesis, we model the users' VR trajectories with the Gaussian AR model. However, other AR models, such as Binomial AR [82], can be leveraged to model VR trajectories. Other approaches for modeling time-series sequences can also be applied. Next, we implement pose estimation and the probability distribution in the disturber with LMMSE estimation and Laplace distribution. However, other estimators for linear models, such as Maximum Likelihood Estimator (MLE), can also be utilized [118]. Moreover, in Chapter 7, we attempted to add perturbations to the orientations in users' trajectories. However, it turned out that the visual quality degrades substantially while the re-id rates are only reduced up to 0.125. This may be because the estimation of the poses is not accurate enough. The optimization algorithms, such as Stochastic Gradient Descent (SGD) [118] and Adam [61], which are the most used optimization algorithms for machine learning and deep learning, can also be utilized for more precise estimation. On the other hand, other probability distributions, such as the Gaussian distribution and exponential distribution, can also be utilized to sample the additive perturbations for the VR trajectories. For the compensation, we use the Reference View Synthesizer (RVS) in this thesis to warp the perturbed images. We can use other more powerful synthesizers that leverage GPU to accelerate the computation. We simply apply the preceding successfully compensated RGB image when encountering a compensation error. However, there are plenty of image error concealment methods that we can utilize for better error handling. In this thesis, we implement the privacy-threat

mitigation approach in VR applications. In the future, we can develop the proposed approach on user HMD/controllers to achieve stronger privacy protection.

清華大學
自強不息 厚德載物
TSING HUA UNIVERSITY

Bibliography

[1] I. Agtzidis, M. Startsev, and M. Dorr. 360-degree video gaze behaviour: A ground-truth data set and a classification algorithm for eye movements. In *Proc. of ACM International Conference on Multimedia (MM)*, pages 1007–1015, Nice, France, October 2019.

[2] A. Ajit, N. Banerjee, and S. Banerjee. Combining pairwise feature matches from device trajectories for biometric authentication in virtual reality environments. In *Proc. of IEEE International Conference on Artificial Intelligence and Virtual Reality (AIVR)*, pages 9–97, San Diego, USA, December 2019.

[3] alvr org. Stream VR games from your PC to your headset via Wi-Fi, 2023. https://github.com/alvr-org/ALVR.

[4] M. Andres, N. Bordenabe, K. Chatzikokolakis, and C. Palamidessi. Geo-indistinguishability: Differential privacy for location-based systems. In *Proc. of ACM SIGSAC conference on Computer & Communications Security (CCS)*, pages 901–914, Berlin, Germany, November 2013.

[5] C. Anthes, R. Garcia, M. Wiedemann, and D. Kranzlmuller. State of the art of virtual reality technology. In *Proc. of IEEE Aerospace Conference*, pages 1–19, Big Sky, USA, March 2016.

[6] AppliedVR. Take control of your chronic lower back pain, 2023. https://www.relievrx.com/.

[7] K. Arulkumaran, M. Deisenroth, M. Brundage, and A. Bharath. Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*, 34(6):26–38, 2017.

[8] J. Barron, B. Mildenhall, M. Tancik, P. Hedman, R. Martin-Brualla, and P. Srinivasan. Mip-NeRF: A multiscale representation for anti-aliasing neural radiance fields. In *Proc. of IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5855–5864, Montreal, Canada, October 2021.

[9] J. Barron, B. Mildenhall, D. Verbin, P. Srinivasan, and P. Hedman. Mip-NeRF 360: Unbounded anti-aliased neural radiance fields. In *Proc. of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5470–5479, New Orleans, USA, June 2022.

[10] B. Bastani, E. Turner, C. Vieri, H. Jiang, B. Funt, and N. Balram. Foveated pipeline for AR/VR head-mounted displays. *Information Display*, 33(6):14–35, 2017.

[11] J. Beck, M. Rainoldi, and R. Egger. Virtual reality in tourism: a state-of-the-art review. *Tourism Review*, 74(3):586–612, 2019.

[12] V. Bindschaedler, R. Shokri, and C. Gunter. Plausible deniability for privacy-preserving data synthesis. *arXiv:1708.07975*, 2017.

[13] D. Bonatto, S. Fachada, S. Rogge, A. Munteanu, and G. Lafruit. Real-time depth video-based rendering for 6-DoF HMD navigation and light field displays. *IEEE access*, 9:146868–146887, 2021.

[14] A. Borji, M.-M. Cheng, Q. Hou, H. Jiang, and J. Li. Salient object detection: A survey. *Computational visual media*, 5:117–150, 2019.

[15] E. Bozkir, O. Gunlu, W. Fuhl, R. Schaefer, and E. Kasneci. Differential privacy for eye tracking with temporal correlations. *Plos one*, 16(8):1–22, 2021.

[16] P. Caserman, A. Garcia, and S. Gobel. A survey of full-body motion reconstruction in immersive virtual reality applications. *IEEE Transactions on Visualization and Computer Graphics*, 26(10):3089–3108, 2019.

[17] B. David, K. Butler, and E. Jain. For your eyes only: Privacy-preserving eye-tracking datasets. In *Proc. of ACM Symposium on Eye Tracking Research & Applications (ETRA)*, pages 1–6, Seattle, USA, June 2022.

[18] B. David, D. Hosfelt, K. Butler, and E. Jain. A privacy-preserving approach to streaming eye-tracking data. *IEEE Transactions on Visualization and Computer Graphics*, 27(5):2555–2565, 2021.

[19] E. David, J. Gutierrez, A. Coutrot, M. Da, and P. Callet. A dataset of head and eye movements for 360° videos. In *Proc. of ACM Multimedia Systems Conference (MMSys)*, pages 432–437, Amsterdam, Netherlands, June 2018.

[20] J. Dong, K. Ota, and M. Dong. Why VR games sickness? an empirical study of capturing and analyzing VR games head movement dataset. *IEEE MultiMedia*, 29(2):74–82, 2022.

[21] C. Dwork. Differential privacy. In *Proc. of International colloquium on automata, languages, and programming (ICALP)*, pages 1–12, Venice, Italy, July 2006.

[22] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. *Journal of Privacy and Confidentiality*, 7(3):17–51, 2016.

[23] C. Dwork and A. Roth. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3–4):211–407, 2014.

[24] S. Eberz, G. Lovisotto, K. Rasmussen, V. Lenders, and I. Martinovic. 28 blinks later: Tackling practical challenges of eye movement biometrics. In *Proc. of ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 1187–1199, London, UK, November 2019.

[25] Eclipse. Eclipse, 2023. https://tinyurl.com/bddxda4z.

[26] K. Emery, M. Zannoli, J. Warren, L. Xiao, and S. Talathi. OpenNEEDS: A dataset of gaze, head, hand, and scene signals during exploration in open-ended VR environments. In *Proc. of ACM Symposium on Eye Tracking Research and Applications (ETRA)*, pages 1–7, Virtual, May 2021.

[27] S. Eraslan, Y. Yesilada, and S. Harper. Scanpath trend analysis on web pages: Clustering eye tracking scanpaths. *ACM Transactions on the Web*, 10(4):1–35, 2016.

[28] U. Erlingsson, V. Pihur, and A. Korolova. Rappor: Randomized aggregatable privacy-preserving ordinal response. In *Proc. of ACM SIGSAC conference on computer and communications security (CCS)*, pages 1054–1067, Scottsdale, USA, November 2014.

[29] S. Fachada, D. Bonatto, A. Schenkel, B. Kroon, and B. Sonneveldt. RVS software, 2023. https://gitlab.com/mpeg-i-visual/rvs/-/tree/master.

[30] S. Fachada, D. Bonatto, A. Schenkel, and G. Lafruit. Free navigation in natural scenery with DIBR: RVS and VSRS in MPEG-I standardization. In *Proc. of IEEE International Conference on 3D Immersion (IC3D)*, pages 1–6, Brussels, Belgium, December 2018.

[31] C. Fan, J. Lee, W. Lo, C. Huang, K. Chen, and C. Hsu. Fixation prediction for 360 video streaming in head-mounted virtual reality. In *Proc. of ACM Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)*, pages 67–72, Taipei, Taiwan, June 2017.

[32] C. Fan, S. Yen, C. Huang, and C. Hsu. Optimizing fixation prediction using recurrent neural networks for 360 video streaming in head-mounted virtual reality. *IEEE Transactions on Multimedia*, 22(3):744–759, 2019.

[33] J. Fang, K. Lee, T. Kamarainen, M. Siekkinen, and C. Hsu. Will dynamic foveation boost cloud VR gaming experience? In *Proc. of ACM Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV)*, pages 29–35, Vancouver, Canada, June 2023.

[34] C. Fehn. Depth-image-based rendering (DIBR), compression, and transmission for a new approach on 3D-TV. In *Proc. of SPIE on Stereoscopic Displays and Virtual Reality Systems (SD&A)*, pages 93–104, San Jose, USA, May 2004.

[35] Fozzy. Fozzy game servers, 2023. https://tinyurl.com/bdh7abyc.

[36] S. Fremerey, A. Singla, K. Meseberg, and A. Raake. AVtrack360: An open dataset and software recording people's head rotations watching 360° videos on an HMD. In *Proc. of ACM Multimedia Systems Conference (MMSys)*, pages 403–408, Amsterdam, Netherlands, June 2018.

[37] W. Fuhl, E. Bozkir, and E. Kasneci. Reinforcement learning for the privacy preservation and manipulation of eye tracking data. In *Proc. of International Conference on Artificial Neural Networks (ICANN)*, pages 595–607, Bratislava, Slovakia, September 2021.

[38] B. GAMES. Beat saber, 2023. https://beatsaber.com/.

[39] Geopipe. Real new york city vol. 2, 2023. https://tinyurl.com/28y93nbn.

[40] L. Giant Form Entertainment. Aircar, 2023. https://tinyurl.com/5xhcp4xr.

[41] A. Giaretta. Security and privacy in virtual reality–a literature survey. *arXiv:2205.00208*, 2022.

[42] N. Gilbert. 74 virtual reality statistics you must know in 2021/2022: Adoption, usage & market share, 2022. https://financesonline.com/virtual-reality-statistics/.

[43] R. Gross, E. Airoldi, B. Malin, and L. Sweeney. Integrating utility into face de-identification. In *Proc. of International Workshop on Privacy Enhancing Technologies (PETS)*, pages 227–242, Cavtat, Croatia, May 2005.

[44] Q. Guimard, F. Robert, C. Bauce, A. Ducreux, L. Sassatelli, H. Wu, M. Winckler, and A. Gros. PEM360: A dataset of 360° videos with continuous physiological

measurements, subjective emotional ratings and motion traces. In *Proc. of ACM Multimedia Systems Conference (MMSys)*, pages 252–258, Athlone, Ireland, June 2022.

[45] R. Hessels, C. Kemner, C. van den Boomen, and I. Hooge. The area-of-interest problem in eyetracking research: A noise-robust solution for face and sparse stimuli. *Behavior research methods*, 48:1694–1712, 2016.

[46] C. Holland and O. Komogortsev. Biometric identification via eye movement scanpaths in reading. In *Proc. of IEEE International joint conference on biometrics (IJCB)*, pages 1–8, Washington DC, USA, October 2011.

[47] A. Hore and D. Ziou. Image quality metrics: PSNR vs. SSIM. In *Proc. of IEEE International Conference on Pattern Recognition (ICPR)*, pages 2366–2369, Istanbul, Turkey, August 2010.

[48] M. Hsieh and J. Lee. Preliminary study of VR and AR applications in medical and healthcare education. *J Nurs Health Stud*, 3(1):1–5, 2018.

[49] HTC. VIVEPORT, 2023. https://www.viveport.com/?hl=zh-TW.

[50] M. Hu, X. Luo, J. Chen, Y. Lee, Y. Zhou, and D. Wu. Virtual reality: A survey of enabling technologies and its applications in IoT. *Journal of Network and Computer Applications*, 178:102970, 2021.

[51] Z. Hu, A. Bulling, S. Li, and G. Wang. EHTask: Recognizing user tasks from eye and head movements in immersive virtual reality. *IEEE Transactions on Visualization and Computer Graphics*, 29(4):1992 – 2004, 2021.

[52] Z. Hu, C. Zhang, S. Li, G. Wang, and D. Manocha. SGaze: A data-driven eye-head coordination model for realtime gaze prediction. *IEEE Transactions on Visualization and Computer Graphics*, 25(5):2002–2010, 2019.

[53] B. Huitema and J. McKean. Autocorrelation estimation and inference with small samples. *Psychological Bulletin*, 110(2):291–304, 1991.

[54] G. Illahi, M. Siekkinen, T. Kamarainen, and A. Yla. Real-time gaze prediction in virtual reality. In *Proc. of ACM International Workshop on Immersive Mixed and Virtual Environment Systems (MMVE)*, pages 12–18, Athlone, Ireland, June 2022.

[55] P. Julayanont and Z. Nasreddine. Montreal cognitive assessment (MoCA): Concept and clinical review. *Cognitive Screening Instruments: A Practical Approach*, pages 139–195, 2017.

[56] J. Jung and P. Boissonade. VVS: Versatile view synthesizer for 6-DoF immersive video. working paper or preprint, 2020.

[57] D. Kaminska, T. Sapinski, L. Wiak, T. Tikk, R. Haamer, E. Avots, A. Helmi, C. Ozcinar, and G. Anbarjafari. Virtual reality and its applications in education: Survey. *Information*, 10(10):318, 2019.

[58] Y. Kavak, E. Erdem, and A. Erdem. A comparative study for feature integration strategies in dynamic saliency estimation. *Signal Processing: Image Communication*, 51:13–25, 2017.

[59] G. Kellaris, S. Papadopoulos, X. Xiao, and D. Papadias. Differentially private event sequences over infinite streams. *Proc. of VLDB Endowment*, 7:1155–1166, 2014.

[60] R. Kennedy, N. Lane, K. Berbaum, and M. Lilienthal. Simulator sickness questionnaire: An enhanced method for quantifying simulator sickness. *The International Journal of Aviation Psychology*, 3(3):203–220, 1993.

[61] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv:1412.6980*, 2014.

[62] C. Kolmar. 25 amazing virtual reality statistics [2022]: The future of VR + AR, 2022. https://www.zippia.com/advice/virtual-reality-statistics/.

[63] G. Kougioumtzidis, V. Poulkov, Z. Zaharis, and P. Lazaridis. A survey on multimedia services QoE assessment and machine learning-based prediction. *IEEE Access*, 10:19507–19538, 2022.

[64] B. Kroon and G. Lafruit. Reference view synthesizer (RVS) 2.0 manual, 2018.

[65] LaikaBossGames. Art gallery museum, 2023. https://tinyurl.com/4bmjtbty.

[66] E. Langbehn, F. Steinicke, M. Lappe, G. Welch, and G. Bruder. In the blink of an eye: Leveraging blink-induced suppression for imperceptible position and orientation redirection in virtual reality. *ACM Transactions on Graphics*, 37(4):1–11, 2018.

[67] S. Learn. Scikit-learn, 2023. https://scikit-learn.org/stable/.

[68] R. Leigh and D. Zee. A survey of eye movements: characteristics and teleology. *The Neurology of Eye Movements; Oxford University Press: New York, NY, USA*, pages 3–19, 2006.

[69] J. Li, A. Roy, K. Fawaz, and Y. Kim. Kalεido: Real-time privacy control for eye-tracking systems. In *Proc. of USENIX Security Symposium*, pages 1793–1810, Virtual, August 2021.

[70] J. Liebers, M. Abdelaziz, L. Mecke, A. Saad, J. Auda, U. Gruenefeld, F. Alt, and S. Schneegass. Understanding user identification in virtual reality through behavioral biometrics and the effect of body normalization. In *Proc. of ACM CHI Conference on Human Factors in Computing Systems*, pages 1–11, Yokohama, Japan, May 2021.

[71] A. Liu, L. Xia, A. Duchowski, R. Bailey, K. Holmqvist, and E. Jain. Differential privacy for eye-tracking data. In *Proc. of ACM Symposium on Eye Tracking Research & Applications (ETRA)*, pages 1–10, Denver, USA, June 2019.

[72] Z. Liu, Z. Zhu, J. Gao, and C. Xu. Forecast methods for time series data: A survey. *IEEE Access*, 9:91896–91912, 2021.

[73] W.-C. Lo, C.-L. Fan, J. Lee, C.-Y. Huang, K.-T. Chen, and C.-H. Hsu. 360° video viewing dataset in head-mounted virtual reality. In *Proc. of ACM on Multimedia Systems Conference (MMSys)*, pages 211–216, Taipei, Taiwan, June 2017.

[74] P. Lungaro, R. Sjoberg, A. Valero, A. Mittal, and K. Tollmar. Gaze-aware streaming solutions for the next generation of mobile VR experiences. *IEEE Transactions on Visualization and Computer Graphics*, 24(4):1535–1544, 2018.

[75] G. Mahalakshmi, S. Sridevi, and S. Rajaram. A survey on forecasting of time series data. In *Proc. of IEEE International Conference on Computing Technologies and Intelligent Data Engineering (ICCTIDE)*, pages 1–8, Kovilpatti, India, January 2016.

[76] W. Mark. *Postrendering 3D Image Warping: Visibility, Reconstruction, and Performance for Depth-image Warping*. PhD thesis, The University of North Carolina, 1999.

[77] F. Mathis, H. Fawaz, and M. Khamis. Knowledge-driven biometric authentication in virtual reality. In *Proc. of Extended Abstracts of the ACM CHI Conference on Human Factors in Computing Systems*, pages 1–10, Honolulu, USA, April 2020.

[78] Math.NET. Math.net numerics, 2023. https://numerics.mathdotnet.com/.

[79] MathNet.Numerics. Mathnet.numerics.distributions, 2023. https://tinyurl.com/8aczzcyy.

[80] P. McCarthy. NuGetForUnity, 2023. https://tinyurl.com/3u446efc.

[81] G. McConkie. Evaluating and reporting data quality in eye movement research. *Behavior Research Methods & Instrumentation*, 13(2):97–106, 1981.

[82] E. McKenzie. Some simple models for discrete variate time series. *JAWRA Journal of the American Water Resources Association*, 21:645–650, 1985.

[83] X. Meng, R. Du, and A. Varshney. Eye-dominance-guided foveated rendering. *IEEE transactions on visualization and computer graphics*, 26(5):1972–1980, 2020.

[84] X. Meng, R. Du, M. Zwicker, and A. Varshney. Kernel foveated rendering. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 1(1):1–20, 2018.

[85] B. Mildenhall, P. Srinivasan, M. Tancik, J. Barron, R. Ramamoorthi, and R. Ng. NeRF: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, December 2021.

[86] B. Mildenhall, P. P. Srinivasan, R. Ortiz-Cayon, N. K. Kalantari, R. Ramamoorthi, R. Ng, and A. Kar. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. *ACM Transactions on Graphics*, 38(4), 2019.

[87] A. Millen and P. Hancock. Eye see through you! eye tracking unmasks concealed face recognition despite countermeasures. *Cognitive Research: Principles and Implications*, 4(23):1–14, 2019.

[88] M. Miller, F. Herrera, H. Jun, J. Landay, and J. Bailenson. Personal identifiability of user tracking data during observation of 360-degree VR video. *Scientific Reports*, 10(1):1–10, 2020.

[89] R. Miller, N. Banerjee, and S. Banerjee. Within-system and cross-system behavior-based biometric authentication in virtual reality. In *Proc. of IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)*, pages 311–316, Atlanta, USA, March 2020.

[90] J. Moline. Virtual reality for health care: a survey. *Virtual Reality in Neuro-Psycho-Physiology*, 44:3–34, 1997.

[91] S. Munn, L. Stefano, and J. Pelz. Fixation-identification in dynamic scenes: Comparing an automated algorithm to manual coding. In *Proc. of ACM symposium*

on Applied Perception in Graphics and Visualization (APGV)*, pages 33–42, Los Angeles, USA, August 2008.

[92] V. Nair, G. Garrido, and D. Song. Exploring the unprecedented privacy risks of the metaverse. *arXiv:2207.13176*, 2022.

[93] V. Nair, M. Garrido, and D. Song. Going incognito in the metaverse. *arXiv:2208.05604*, 2022.

[94] Netflix. VMAF - video multi-method assessment fusion, 2021. https://github.com/Netflix/vmaf.

[95] E. Newton, L. Sweeney, and B. Malin. Preserving privacy by de-identifying face images. *IEEE transactions on Knowledge and Data Engineering*, 17(2):232–243, 2005.

[96] E. Niebur. Saliency map. *Scholarpedia*, 2(8):2675–2675, 2007.

[97] Numpy. Legacy random generation, 2023. https://tinyurl.com/mr3fdaza.

[98] I. Olade, C. Fleming, and H. Liang. Biomove: Biometric user identification from human kinesiological movements for virtual reality systems. *Sensors*, 20(10):2944, 2020.

[99] OpenCV. Opencv modules, 2023. https://docs.opencv.org/4.7.0/.

[100] A. Patney, M. Salvi, J. Kim, A. Kaplanyan, C. Wyman, N. Benty, D. Luebke, and A. Lefohn. Towards foveated rendering for gaze-tracked virtual reality. *ACM Transactions on Graphics*, 35(6):1–12, 2016.

[101] K. Pfeuffer, M. Geiger, S. Prange, L. Mecke, D. Buschek, and F. Alt. Behavioural biometrics in VR: Identifying people from body motion and relations in virtual reality. In *Proc. of ACM CHI Conference on Human Factors in Computing Systems*, pages 1–12, Glasgow, UK, May 2019.

[102] Polyarc. Moss, 2023. https://tinyurl.com/59buur7s.

[103] V. Rastogi and S. Nath. Differentially private aggregation of distributed time-series with transformation and encryption. In *Proc. of ACM SIGMOD International Conference on Management of Data (SIGMOD)*, pages 735–746, Indianapolis, USA, June 2010.

[104] I. RealSense. Intel realsense sdk 2.0 (v2.53.1), 2023. https://github.com/IntelRealSense/librealsense/.

[105] I. RealSense. pyrealsense2 2.53.1.4623 project description, 2023. https://pypi.org/project/pyrealsense2.

[106] S. Ribaric, A. Ariyaeeinia, and N. Pavesic. De-identification for privacy protection in multimedia content: A survey. *Signal Processing: Image Communication*, 47(2):131–151, 2016.

[107] S. Saini, S. Malhi, B. Saro, M. Khan, and A. Kaur. Virtual reality: A survey of enabling technologies and its applications in IoT. In *Proc. of International Conference on Electronics and Renewable Systems (ICEARS)*, pages 597–603, Tuticorin, India, March 2022.

[108] B. S. Salahieh, J. Jung, and A. Dziembowski. Test model 10 for MPEG immersive video. 2021.

[109] D. Salvucci and J. Goldberg. Identifying fixations and saccades in eye-tracking protocols. In *Proc. of ACM symposium on Eye Tracking Research & Applications (ETRA)*, pages 71–78, Palm Beach Gardens, USA, November 2000.

[110] Scipy. scipy.spatial.transform.rotation, 2023. https://tinyurl.com/2p8zatbr.

[111] Shapes. Nature starter kit 2, 2023. https://tinyurl.com/yuruhcux.

[112] M. Slater and M. Sanchez-Vives. Enhancing our lives with immersive virtual reality. *Frontiers in Robotics and AI*, 3:74, 2016.

[113] C. Slocum, Y. Zhang, N. Abu, and J. Chen. Going through the motions: AR/VR keylogging from user head motions. In *Proc. of USENIX Security Symposium*, pages 159–174, Anaheim, USA, August 2023.

[114] O. Stankiewicz, K. Wegner, M. Tanimoto, and M. Domański. Enhanced view synthesis reference software (VSRS) for free-viewpoint television. 2013.

[115] SteamVR. Steamvr, 2023. https://tinyurl.com/yptjjtjs.

[116] J. Steil, I. Hagestedt, X. Huang, and A. Bulling. Privacy-aware eye tracking using differential privacy. In *Proc. of ACM Symposium on Eye Tracking Research & Applications (ETRA)*, pages 1–9, Denver, USA, June 2019.

[117] O. Studio. Free medieval room, 2023. https://tinyurl.com/mwmfft2v.

[118] X. Su, X. Yan, and C. Tsai. Linear regression. *Wiley Interdisciplinary Reviews: Computational Statistics*, 4:275–294, 2012.

[119] Q. Sun, A. Patney, L. Wei, O. Shapira, J. Lu, P. Asente, S. Zhu, M. McGuire, D. Luebke, and A. Kaufman. Towards virtual reality infinite walking: Dynamic saccadic redirection. *ACM Transactions on Graphics*, 37(4):1–13, 2018.

[120] Y. Sun, S. Tang, C. Wang, and C. Hsu. On objective and subjective quality of 6DoF synthesized live immersive videos. In *Proc. of ACM Workshop on Quality of Experience in Visual Multimedia Applications (QoEVMA)*, pages 49–56, Lisboa, Portugal, October 2022.

[121] A. Sunshine. Arizona sunshine, 2023. https://www.arizona-sunshine.com/.

[122] L. Tabbaa, R. Searle, S. Bafti, M. Hossain, J. Intarasisrisawat, M. Glancy, and C. Ang. VREED: Virtual reality emotion recognition dataset using eye tracking & physiological measures. *ACM Interact. Mob. Wearable Ubiquitous Technol.*, 5(4):1–20, 2022.

[123] A. D. P. Team. Learning with privacy at scale, 2017. https://tinyurl.com/3r7tbw8r.

[124] S. Team. Superhot vr, 2023. https://tinyurl.com/2bhhzmj3.

[125] E. Thambiraja, G. Ramesh, and D. Umarani. A survey on various most common encryption techniques. *International Journal of Advanced Research in Computer Science and Software Engineering*, 2(7):226–233, 2012.

[126] Tobii. Tobii XR API, 2023. https://developer.tobii.com/xr/develop/unity/.

[127] Tobii. Troubleshoot, 2023. https://developer.tobii.com/pc-gaming/unity-sdk/troubleshoot/.

[128] S. Tomar. Converting video formats with FFmpeg, 2006. https://tinyurl.com/3tr3nuvt.

[129] Unity. Unity, 2023. https://unity.com/.

[130] Unity. Unity asset store, 2023. https://assetstore.unity.com/.

[131] Unity. Unity scripting API, 2023. https://docs.unity3d.com/ScriptReference/.

[132] Unity. XR interaction toolkit, 2023. https://tinyurl.com/m37py5ty.

[133] I. Wagner and D. Eckhoff. Technical privacy metrics: a systematic survey. *ACM Computing Surveys*, 51(3):1–38, 2018.

[134] Y. Wang, Z. Su, N. Zhang, R. Xing, D. Liu, T. Luan, and X. Shen. A survey on metaverse: Fundamentals, security, and privacy. *IEEE Communications Surveys & Tutorials*, 25(1):319–352, 2022.

[135] W. Wei. *Time Siries Analysis*, volume 2. Oxford University Press, 2013.

[136] X. Wei and C. Yang. FoV privacy-aware VR streaming. In *Proc. of IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1515–1520, Austin, USA, April 2022.

[137] Y. Wei, X. Wei, S. Zheng, C. Hsu, and C. Yang. A 6DoF VR dataset of 3D virtualworld for privacy-preserving approach and utility-privacy tradeoff. In *Proc. of ACM Multimedia Systems (MMSys)*, pages 444–450, Vancouver, Canada, June 2023.

[138] C. Wu, Z. Tan, Z. Wang, and S. Yang. A dataset for exploring user behaviors in VR spherical video streaming. In *Proc. of ACM on Multimedia Systems Conference (MMSys)*, pages 193–198, Taipei, Taiwan, June 2017.

[139] Y. Xu, Y. Dong, J. Wu, Z. Sun, Z. Shi, J. Yu, and S. Gao. Gaze prediction in dynamic 360° immersive videos. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5333–5342, Salt Lake City, USA, June 2018.

[140] T. Xue, A. El, T. Zhang, G. Ding, and P. Cesar. CEAP-360VR: A continuous physiological and behavioral emotion annotation dataset for 360 VR videos. *IEEE Transactions on Multimedia*, 25:243 – 255, 2021.

[141] A. Yaqoob, T. Bi, and G.-M. Muntean. A survey on adaptive 360 video streaming: Solutions, challenges and opportunities. *IEEE Communications Surveys & Tutorials*, 22(4):2801–2838, 2020.

[142] G. Yule. VII. on a method of investigating periodicities disturbed series, with special reference to Wolfer's sunspot numbers. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, 226(636-646):267–298, 1927.

[143] J. Zhai, S. Zhang, J. Chen, and Q. He. Autoencoder and its various variants. In *Proc. of IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 415–419, Miyazaki, Japan, October 2018.

[144] X. Zhang, M. Khalili, and M. Liu. Differentially private real-time release of sequential data. *ACM Transactions on Privacy and Security*, 26(1):1–29, 2022.

[145] J. Zheng, K. Chan, and I. Gibson. Virtual reality. *IEEE Potentials*, 17(2):20–23, 1998.

[146] Y. Zhou, T. Feng, S. Shuai, X. Li, L. Sun, and H. Duh. EDVAM: A 3D eye-tracking dataset for visual attention modeling in a virtual museum. *Frontiers of Information Technology & Electronic Engineering*, 23(1):101–112, 2022.

清華大學 TSING HUA UNIVERSITY