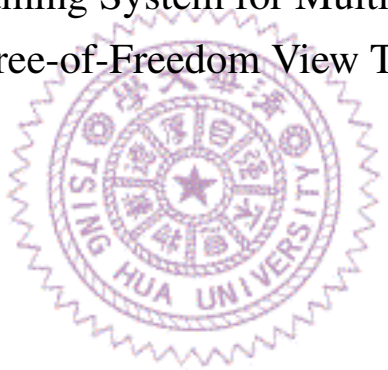
國立清華大學電機資訊學院資訊工程研究所
碩士論文
Department of Computer Science
College of Electrical Engineering and Computer Science
National Tsing Hua University
Master Thesis

支援多使用者六自由度線上瀏覽的盲串流系統
A Blind Streaming System for Multi-client Online 6-Degree-of-Freedom View Touring



唐盛銘
Sheng-Ming Tang

學號：110062572
Student ID:110062572

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

中華民國 112 年 6 月
June, 2023

國立清華大學
資訊工程研究所

碩士論文

支援多使用者六自由度線上瀏覽的盲串流系統

唐盛銘

Abstract

Online 6 degree of freedom (6-DoF) view touring has become increasingly popular due to hardware advances and the recent pandemic. One way for content creators to support many 6-DoF clients is by transmitting 3D content to them, which unfortunately leads to content leakage. Another way for content creators is to render and stream novel views for 6-DoF clients, which unfortunately incurs staggering computational and networking workloads. In this thesis, we develop a blind streaming system as a better solution that leverages cloud service providers between content creators and 6-DoF clients. The proposed blind streaming system has two core design objectives: (i) to generate high-quality novel views for 6-DoF clients without retrieving 3D content from content creators, (ii) to support many 6-DoF clients without overloading the content creators. We achieve these two goals in the following steps. First, we design a source view request/response interface between cloud service providers and content creators for efficient communications. Second, we present novel view optimization algorithms for cloud service providers to intelligently select the minimal set of source views while considering the workload of content creators. Third, we employ scalable client-side view synthesis for 6-DoF clients with heterogeneous device capabilities and personalized 6-DoF client poses and preferences. Our evaluation results demonstrate the merits of our blind streaming system; compared to the state-of-the-art solution, our system: (i) improves synthesized novel views by 2.27 dB in PSNR and 12 in VMAF on average, and (ii) reduces the bandwidth consumption by 94% on average. In fact, our blind streaming system approaches the performance of an unrealistic optimal solution with unlimited source views, achieving performance gaps as small as 0.75 dB in PSNR and 3.8 in VMAF. We also empirically demonstrate that our blind streaming system is not vulnerable to 3D content reconstruction algorithms such as Structure-from-Motion (SfM).

Acknowledgments

There are many people I appreciate during my master's life.

First, I would like to thank my advisor Cheng-Hsin Hsu for supporting my research during these years. He leads students with plans and tries to make everything organized and pushes us to keep up with the schedules. I never worry about my research and graduation because I know that once I follow the schedule and do my best, Cheng-Hsin will try his best to pave the way for me to chase our goals. Besides, Cheng-Hsin is a good consultant for research and career and also an open-minded person. Whenever I discussed my research plan with him, he always listened to my ideas carefully and tried to give me constructive feedback. Also, everyone can negotiate our schedule and research directions with Cheng-Hsin to strike a good balance between our limited time and research completeness. Furthermore, he is always happy to cover his students for international conferences and scholarship. He cares about his students and tries to make the most of every chance that would benefit or make the students better. For example, he wrote a cover letter for me for a student grant to Lisbon and for my internship at MediaTek.

Second, I would like to thank the Pan Wen-Yuan foundation for giving me a scholarship in 2022. I really appreciate their support for offering me daily expenses so that I could focus more on my research. Also, I am thankful for being able to go on a tour in IRTI in 2023, which helped me get to know more about the contribution of Mr. Pan Wen-Yuan and the responsibility we are going to take as engineers/researchers.

Third, I would like to thank MediaTek for offering me an internship opportunity in summer 2022. The faculties there are so nicely and always think on my side. They allowed me to work from home so that I can save my trips to go back and forth between our lab and the company. I learned a lot during that internship and felt so proud of myself that I could make a contribution to research. Among the faculties, I would like to give my special thanks to Alex who treats me so nice and is willing to write a cover letter for me for applying for the Pan Wen-Yuan scholarship. Moreover, I am also thankful for the help from Tim. He is a good mentor that is so kind and pointed out a clear way for me to go forward.

Last, I would like to thank my labmates. I won't forget I met those friends and shared those good days which enriched my master's life. We are not partners for research only, we are also partners for life, entertainment, and mental support. I really appreciate those moments that I chat with you about my worries, murmurs and everything, and you always pay full attention to providing me with constructive feedback. Thanks to all of you for providing me with such a comfortable environment with a good atmosphere. Among

the labmates, I would like to give special credits to Jia-Wei Fang, Yuan-Chun Sun, Ching-Ting Wang, and Kuan-You Lee for supporting me to write my publication in IXR'22. Without your help, I would not have been able to complete this work. Especially, I would like to offer my highest respect and appreciation to Yuan-Chun Sun who helped me throughout these years. He is so smart that he learns things quickly and is always be able to point out my blind spots. Furthermore, he is a responsible partner, and I never worry about the jobs I have assigned to him. I am really thankful for his help to make everything on schedule.

Contents

List of Figures

List of Tables


TSING HUA UNIVERSITY

Chapter 1

Introduction

Virtual Reality (VR) refers to virtual representations of real life including content, objects, avatars, and metaverses [7]. VR technologies have gained popularity in the recent years and have attracted content creators, such as game designers, to create 3D content on top of rendering engines, such as Unreal Engine [10] and Unity [12]. Furthermore, the metaverse market is expected to grow to 293.71 billion USD by 2027 [20]. In order to allow Head Mounted Display (HMD) clients to explore VR content freely as in real life, VR streaming systems should support 6 Degree-of-Freedom (6-DoF), in which HMD clients can: (i) move their position in three translational directions *x, y, z*, and (ii) rotate their orientation along three axes *roll, pitch, and yaw*. We collectively refer to a pair of position and orientation as a user *pose* at a moment and a time series of poses as *pose trajectories*. In the rest of this thesis, we refer to clients and users interchangably.

We study 6-DoF view touring systems, which enable various new applications. For example, online art galleries relax their geographical constraints and crowd limitations imposed by physical constraints. By moving art pieces online, HMD clients get to enjoy galleries anywhere in the world and can avoid unnecessary physical contact with people. Online house touring is another application, in which potential buyers could see houses anywhere and anytime. This saves sellers, agents, and buyers a lot of commute time, reducing the overhead on the real-estate industry. Naive 6-DoF view touring systems stream 3D content directly to HMD clients, which could lead to *Intellectual Property (IP) leakage*. That is, malicious clients could redistribute 3D content without permission from content creators.

To solve the IP leakage issue, we introduce *blind streaming*, which supports multiple 6-DoF HMD clients *without* sending 3D content to them. Building a scalable blind streaming system is no easy task, because its bandwidth consumption grows linearly as the number of HMD clients increases. Therefore, we employ *client-side view synthesis*, which uses multiple RGB-D *source views*, shot by either physical cameras or virtual ones

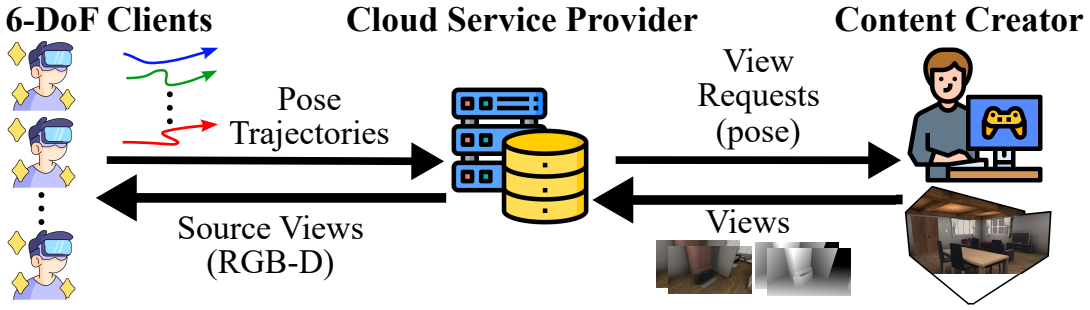


Figure 1.1: A blind streaming system.

if in the virtual world, with different poses to synthesize a *novel view* for each HMD client at any moment. The source views must be carefully selected to synthesize high-quality novel views. For instance, a region of novel view that is occluded in all source views would become a black hole, leading to inferior visual quality and driving HMD clients away from the view touring system. Hence, the selection of source views to maximize the quality of synthesized novel views is the crux for commercially-viable blind streaming systems.

Fig. 1.1 presents three entities of a blind streaming system: (i) *6-DoF clients* transmit their pose trajectories and synthesize their novel views using the source views from cloud service providers, (ii) *cloud service providers* process the pose trajectories and requests for several source views from content creators on behalf of 6-DoF clients, and (iii) *content creators* keep the 3D content with them, possibly in 3D mesh or point clouds, as long as the format can be rendered, and generate source views satisfying requests from cloud service providers. To the best of our knowledge, the design of blind streaming systems has not been explored. Our prior study [38] on blind streaming has several limitations in our prototype: (i) insufficient information for cloud service providers to select the optimal source views, (ii) heavy workload on content creators for computing *coverage ratio* between two pose trajectories, and (iii) lower novel view quality due to the insufficient computation capability of cloud service providers. To eliminate these limitations, we propose to: (i) transmit pixel-level auxiliary data from content creators to cloud service providers, (ii) offload some workload from content creators to cloud service providers, and (iii) synthesize novel views for individual 6-DoF clients.

1.1 Contributions

This thesis makes the following contributions:

1. (Ch. 4) We propose a content-creator-friendly blind streaming system that requires few resources from content creators and scales to many 6-DoF clients by properly

redistributing networking and computational workloads.

2. (Ch. 8) We develop a mechanism to estimate the *coverage map*, which tells whether individual pixels are covered or not, instead of a scalar coverage ratio between two RGB-D source views without accessing the original 3D content. To the best of our knowledge, our mechanism is the first of its kind in the literature.

3. (Ch. 5) We propose a suite of heterogeneous algorithms to select optimal source views. These algorithms exercise a trade-off between computational complexity and selection optimality. The best-performing algorithm improves the average novel view quality by 2.27 dB in Peak Signal-to-Noise Ratio (PSNR) [14] and 12 in Video Multi-Method Assessment Fusion (VMAF) [26], compared to our prior work [38].

1.2 Limitations

In the following discussion, we assume that novel view synthesis runs in real-time on HMD clients for achieving seamless experience in the VR world. State-of-the-art solutions are ready to be employed and demonstrated by a proof-of-concept project, Freeport Player [39] from Intel. However, Freeport player itself is not open-source; thus we offload novel view synthesis to a workstation to achieve the same effect. Furthermore, we also assume that lower network layers have implemented transport level features of zero packet loss, packet retransmission, or data recovery on packet loss. Second, we also assume that our system runs in a static network topology with sufficient bandwidth. Or, equivalently, proper congestion control algorithms have been employed to achieve the same effect. For content specifications, we design and evaluate our systems for static content as a starting point of blind streaming systems. Static content is less complex to analyze and implement, though our systems can be generalized to dynamic content by considering the dynamic ones as quasi-static content. That is, we consider the content static at the moment we are updating the source views.

1.3 Organization

We introduce inspiration from the IP leakage problem and applications for blind streaming systems, and summarize the contributions along with the challenges and limitations in Ch. 1. In Ch. 2, we give a high level overview of the concepts of head mounted displays and virtual reality, 6-DoF view touring, and common streaming media format including RGB-D images and meshes. We survey the literature and summarize prior work related

to either some components in a blind streaming system or some concepts that inspire our design in Ch. 3. Then, we introduce Ch. 4 which contains high-level design of the blind streaming system framework in terms of the three entities, 6-DoF client, cloud service providers, content creators, and their interactions. We also illustrate the time sequence and key components of how a blind streaming system works. Then, we formulate the blind streaming problem into a novel view optimization problem in Ch. 5. In the following chapters, we introduce each component in detail. We introduce the components including pose predictor, candidate generator, coverage estimator and solver in Ch. 6, Ch. 7, Ch. 8 and Ch. 9, respectively. Next, we introduce how we implement the blind streaming system prototype and evaluate system performance in terms of view quality, run time, and bandwidth consumption in Ch. 10. Last, we summarize the whole thesis and present that our work survives under the attack of Structure-from-Motion (SfM) in Ch. 11.

Chapter 2

Background

For better understanding of readers from different fields, we give background knowledge on the concepts of head mounted displays, 6-DoF, virtual reality, view touring applications, and common streaming media formats used in the thesis.

2.1 Head mounted display (HMD) and 6 degree of freedom

Figure 2.1: An Oculus Quest Pro.

Head mounted displays (HMD) are the devices for people to directly wear on their faces, which is why they are called head mounted. HMDs offer immersive experiences because our vision is basically filled with the built-in displays to block stimulus from the

outside world. Several manufacturers have produced HMDs, and an example is shown in Fig. 2.1. The demo sample is newly released by Meta [18]. Basic components include:

1. An accelerometer for measuring its acceleration in the 3D world.
2. A gyroscope for measuring its speed of rotation.
3. A Camera for detecting the distance from a user to an obstacle and for the user to see through to the outside world to ensure safety.
4. An embedded monitor to display the user interface or applications.
5. Network interfaces for wireless communication and experience.

Most of these devices are in the form of all-in-one, which means the HMD itself contains the Android operating system and is a complete entity capable of running Android applications. On the other hand, some of them are coupled with a high-end personal computer, which acts as an external device. Researchers prefer the former ones because they are relatively easy and free to develop applications. Modern HMDs operate in 6 degree of freedom (6-DoF) instead 3-DoF. Earlier products only operate in the rotational axes of *roll, pitch*, and *yaw* anchored at the *x, y*, and *z* axes respectively in the right-hand-convention. That means users can only rotate their heads, and their translational movement will be ignored and thus they are prone to motion sickness. Instead, 6-DoF supports another 3 translational axes of *x, y*, and *z*, which enables a more completely immersive experience. The main difference is illustrated in Fig 2.2, where 3-DoF+ means the combination of 3 rotational axes plus one translational axis.

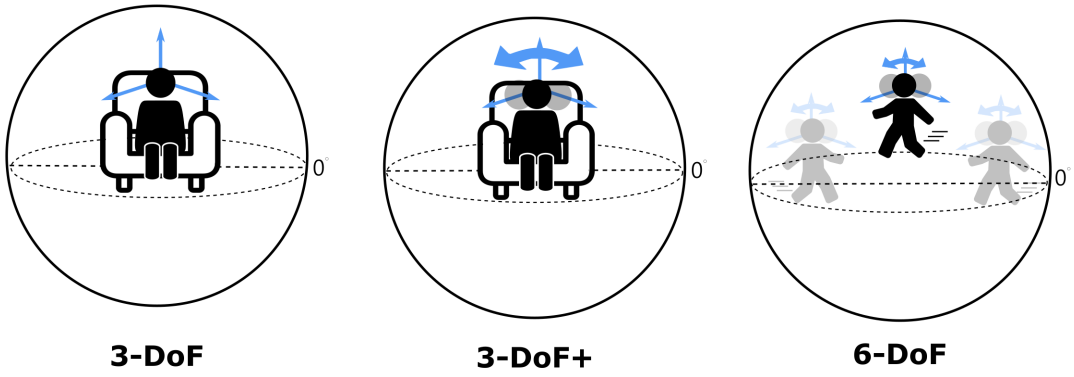


Figure 2.2: Illustrations for the progress of 3-DoF to 6-DoF.

2.2 Virtual reality (VR) and view touring

With advances in HMD technology, virtual reality (VR) has become popular in recent years. Virtual reality is a computer-generated virtual world taht aims to provide immersive

experience for HMD users. Many of the applications have arisen in the market including VR gaming, VR viewing and some engineering fields like remote control. In this thesis, we will focus on the VR view touring applications in which users are free to move around and explore the VR world. We chose them because they are common and fundamental for VR streaming applications. Research on VR view touring will also benefit other streaming-based applications because the skills and algorithms can be easily extended to other fields.

2.3 Common streaming media formats

Figure 2.3: Examples of a mesh, an RGB image and a D image.

In this section, we are going to introduce two common streaming media formats, namely meshes and images. Some examples are shown in Fig. 2.3.

Meshes are 3D models consist of sets of vertices and faces describing connections between vertices. Each vertex can be described by their position in x, y, and z, their colors in R, G, and B, and optionally their normal vectors for calculating different lighting conditions. Then, a flat surface described by a face will be interpolated by the 3 vertices belonging to the flat surface. Furthermore, meshes are so popular because affine transformation of meshes can be described by single matrix multiplication, which is efficient and can be easily parallelized by graphics processing units (GPUs). Though meshes are efficient to manipulate, obtaining meshes implies obtaining the whole 3D geometry, which is not preferred in blind streaming.

On the other hand, images are even simpler to manipulate. Images contain a set of pixels normally arranged horizontally and vertically, and each of the pixels contains value(s) describing their attributes. There are two common types of images, red-green-blue(RGB) and depth(D) images. RGB images describe color information by filling the values describing the strength of the three colors. On the other hand, D ioamges describe the depth values, or equivalently, the distance starting from the origin of the camera through the pixel to the physical entity. Note that RGB images do not necessarily describe the geometry seen from the camera, though there are works on inferring depth images from RGB images. However, D images actually leak the partial geometry seen from the camera.

Combining both, RGB-D images are pairs of the RGB and D iamges describing the full information of a physical entity from some aspects, and some novel view synthesizers could use this information to infer the view they have never seen. Images are natural for HMDs because applications have to eventually display images to the users. Moreover, image compression is mature so that transmission is both bandwidth and computationally efficient. Image manipulation can also be parallelized by affine transformation on GPUs.

Chapter 3

Related Work

Although blind streaming is a totally new concept, the following prior studies inspired our design.

3.1 Novel view synthesis

View synthesis has been done with different source view formats. Hladky et al. [13] invented a new 3D source view format to efficiently generate all novel views within a predefined box. However, their method requires 3D content and leads to IP leakage. Choi et al. [8] generalized single-value depth prediction from multiple cameras to a distribution for better novel view synthesis. Park et al. [28] presented a transformation-grounded image generation neural network for novel view synthesis from a single image. Attal et al. [2] transformed stereo 360° videos into multi-sphere images to enable 6-DoF synthesis in the vicinity of the sphere centers. MPEG developed an Immersive Video codec, called MIV [5], for novel view synthesis. It packs source views into *atlases* by removing duplicated content, and then employs a view synthesizer, such as RVS [19] to render novel views from multiple RGB-D source views. *Our blind streaming system also employs RVS for novel view synthesis, although other synthesizers can be readily adopted.*

3.2 Coverage optimization and view selection

Optimal source view selection is largely driven by coverage optimization among possible poses. In the literature, Wang et al. [43] designed an algorithm for arranging a fleet of drones to stream sports events by solving a matching problem between drones and possible poses. Suresh et al. [36] divided the terrain into 2D grids and used greedy algorithms to perform voting between grids and camera poses. Munishwar and Abu-Ghazaleh [24] maximized the number of covered targets given a fixed number of cameras. Zhang et

al. [46] proposed an algorithm to compute the pose trajectories of multiple cameras to ensure full coverage of a set of 2D points. Peng and Isler [30] designed an algorithm for computing optimal flying paths for aerial 3D reconstruction. Although the above studies [24, 30, 36, 43, 46] addressed the coverage optimization problem with different assumptions, their target is 3D content (re)construction. Hence, they are *overkills* for synthesizing novel views for 6-DoF clients in blind streaming systems.

There are also light-weight solutions for view selections, e.g., Hänel et al. [16] formulated the 6-DoF camera placement problem into a differentiable objective function interpolated from sampled objective function values, which could be efficiently solved using block-coordinate ascent. However, samples of the objective function are not directly available in blind streaming systems. Bonatto et al. [4] developed a system that enables 6-DoF navigation in real-time and searches for the optimal number of possible poses to meet the real-time requirement. As they assumed that the possible poses were given, their solution is not applicable to blind streaming systems. Tang et al. [38] is probably the closest work to this thesis. They proposed view selection algorithms for novel view synthesis given mutual coverage ratio in scalar between poses. They used vector and matrix operations to approximate set operations such as unions and intersections. Though they attempted to analytically bound their coverage ratio, their solution did not scale well when the problem size is large. This limitation results in inferior visual quality, as we will report in Ch. 10.

Chapter 4

High-Level Design

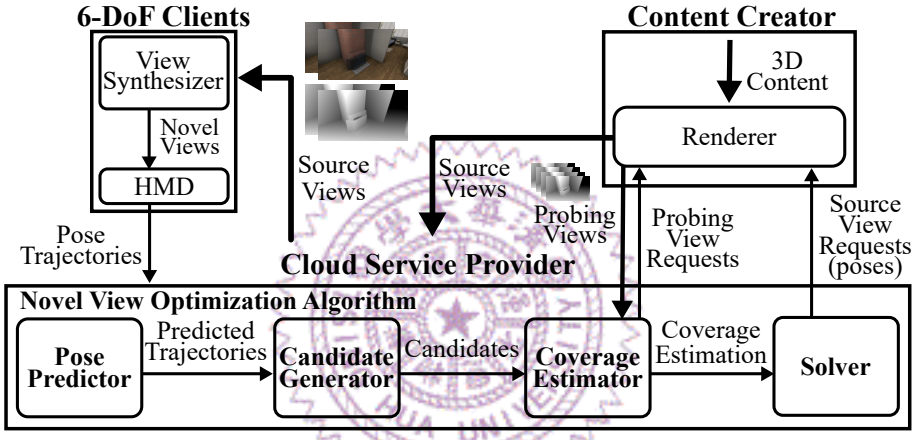


Figure 4.1: Key components of a blind streaming system.

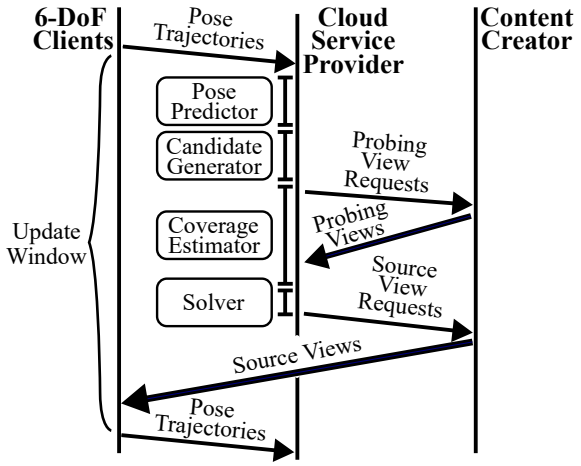


Figure 4.2: Operations of a blind streaming system.

In this chapter, we present the key components and operations of our blind streaming system. Fig. 4.1 gives the key components of the three entities. *6-DoF clients* consist of a *view synthesizer*, which synthesizes received RGB-D source views into novel views for

the current poses. Having a local view synthesizer allows 6-DoF clients to render at diverse frame rates depending on their hardware specifications, and mitigates the response delay due to network latency. *Cloud service providers* receive pose trajectories from 6-DoF clients and are responsible for selecting RGB-D source views. Without prior knowledge of 3D content, cloud service providers submit requests for *probing views* asking for low-resolution depth images at "carefully chosen" poses. Different from high-resolution RGB-D source views, probing views allow cloud service providers to "probe" the geometry of the 3D content, so as to quantify the overlapping "degree" between any two possible camera poses.

The objective of view selections is to maximize the overall quality of novel views across all 6-DoF clients. We refer to this problem as the *novel view optimization problem*, which is solved by four components: (i) a *pose predictor* predicts future pose trajectory within a short *update window* (usually a few seconds) using historical pose trajectories, (ii) a *candidate generator* concatenates all client pose trajectories into a long one, divides it into several partitions of poses, and transforms each partition into a representative pose as a source view candidate; by doing so, the candidate generator reduces the problem size, (iii) a *coverage estimator* retrieves probing views from content creators and computes mutual coverage maps of every pair of source view candidates, and (iv) a *solver* solves a mathematical optimization problem to select source views for 6-DoF clients. Note that *content creators* never transmit 3D content to cloud service providers to avoid IP leakage. Instead, content creators generate probing and source views using a *renderer* to satisfy the requests from cloud service providers. The renderer is implemented by combining a rendering engine [10, 12] with standardized video codecs, like MPEG H.264 [31] and MIV [22].

Fig. 4.2 gives the operations of our blind streaming system. We set the update window to be 50 frames, about 1–2 seconds, for fast adaptations, if not otherwise specified. In each update window, the following events occur: (i) the 6-DoF clients transmit their pose trajectories to the cloud service provider, (ii) the cloud service provider invokes the pose predictor and candidate generator, and coverage estimator, (iii) it also requests probing views from the content creator, (iv) upon getting source view candidates and receiving probing views, the cloud service provider invokes the coverage estimator to compute coverage maps, then the solver solves an optimization problem for source views, (v) the content creator sends the rendered source views through the cloud service provider to the 6-DoF clients, and (vi) the 6-DoF clients synthesize their novel views. After the current update window, the 6-DoF clients move into the next update window.

Chapter 5

Novel View Optimization: Problem and Solution

In this chapter, we formulate and solve the novel view optimization problem.

5.1 Problem formulation

Our goal is to optimize the summation of synthesized view quality for all 6-DoF clients. There are several aspects to be considered. First, to avoid overloading the cloud service provider and content creator, we set the budgets of: (i) source view requests to be N and (ii) probing view requests to be M. Second, we adapt the source views in recurring update windows. More specifically, we let $\mathcal{T}$ be the time slots of the upcoming update window, $\mathcal{S}_\mathcal{T}$ be the source views for $\mathcal{T}$, $\mathcal{P}_\mathcal{T}$ be the probing views for $\mathcal{T}$, $\mathcal{U}$ be the set of 6-DoF clients, and $v_{u,t}$ be the pose for a client $u \in \mathcal{U}$ at time slot $t \in \mathcal{T}$. Last, to estimate synthesized view quality, we use $\text{ql}(\cdot)$ to denote the quality model predicting the synthesized view quality seen at $v_{u,t}$ by considering the source views $\mathcal{S}_\mathcal{T}$ and the probing views $\mathcal{P}_\mathcal{T}$. We write the novel view optimization problem as a high-level formulation:

$$\begin{aligned} \underset{\mathcal{S}_\mathcal{T}}{\text{maximize}} \quad & \sum_{u \in \mathcal{U}} \sum_{t \in \mathcal{T}} \text{ql}(v_{u,t}, \mathcal{S}_\mathcal{T}, \mathcal{P}_\mathcal{T}) \\ \text{subject to :} \quad & |\mathcal{S}_\mathcal{T}| \leq N; \\ & |\mathcal{P}_\mathcal{T}| \leq M, \end{aligned} \tag{5.1}$$

However, not all content creators are willing to serve requests of probing views. We categorize the content creators into 2 groups. The first group consists of the content creators that are willing to serve only scalar coverage ratio among poses, called *S-CC*, while the other group consists of those who are willing to serve probing views, called *P-CC*. Scalar coverage ratio means the scalar representing the ratio of covered area relative to the whole image frame. In the following chapters, we will describe each component in

Fig. 4.1 for *S-CC* and *P-CC* respectively since we will leverage probing views and formulate the problem into a more exact form compared to the scenario that we only have scalar coverage ratio. We name the components for *S-CC* with prefix *S-* while those for *P-CC* with prefix *P-* to distinguish both more clearly if dedicated components must be specified for different kinds of content creators. We will add more details to the formulation and solve it in the following chapters.

5.2 System specification for S-CC and P-CC

Since there are two groups of content creators, we specify the components for both groups and give an overview for avoiding ambiguity in this section.

1. **Pose predictors** are shared components for both groups because it only considers client pose trajectories.

2. **Candidate generators** are dedicated. We have S-Cdd for S-CC, which only generates candidates when a new pose cannot cover as much as expected. On the other hand, we have P-Cdd for P-CC, which generates candidates by considering the workload for content creators and cloud service providers.

3. **Coverage estimators** are dedicated. We have $\text{cvg}_1(\cdot)$ and $\text{cvg}_2(\cdot)$ for S-CC, which use vectors and matrices as numerical approximation. On the contrary, we have $\text{cvg}_P(\cdot)$ for P-CC, which computes pixel level coverage estimation with the help of probing views.

4. **Solvers** are dedicated. We have *C1G, C2G, C2I* solvers for S-CC because it formulates coverage estimation with approximation in integer programming format. On the other hand, we have *Uni*, *BB*, *UM* solvers for P-CC which solve problems iteratively by leveraging the monotonic increasing property in the formulation.

Chapter 6

Pose Predictor

The pose predictor aims to predict client pose trajectories for the next update window. Though we adapt a pose predictor in the system design, it is actually optional in both groups of content creators. The main difference is that once the pose trajectories could be predicted, the source view update calculated by the cloud service provider can be done beforehand to compensate for the extra latency. That is, the extra latency from the time that pose trajectories are issued from the 6-DoF clients to the time that their displays are refreshed could be compensated for, and this kind of latency is called motion-to-photon latency. Higher latency often induces cybersickness among HMD clients and thus should be prevented if possible. However, the pose predictor plays an important role in the system because it basically determines the quality of inputs to the following components. Inaccurate pose trajectories could lead to suboptimal source view updates and thus have a negative impact on the system performance. The pose predictor can be built on prior arts, such as the Kalman filter based solution from Serhan et al. [11] and the long short-term memory (LSTM) solution from Hou et al. [15]. Since this is a well-studied problem, we assume a perfect pose predictor is adopted to exclude any unnecessary quality drop induced by the pose predictor.

Chapter 7

Candidate Generator

The design intuition of the candidate generator is that we want to formulate the novel view optimization problem as stated in Eq. 5.1 into a selection problem instead of a generation problem. A selection problem is usually easier to analyze compared to a generation problem because it only needs to focus on the limited solution space. That is, the selection of any number of source views is from the pool of source view candidates. However, the size of the solution space is crucial because a smaller space leads to suboptimal results, which is equivalent to undersampling of the pose trajectory, while a bigger one may make it hard to find the optimal solution within such a large space because we need to spend much more effort identifying the optimal solution from a huge number of solutions. There are additional reasons for having a candidate generator. First, in this stage, the candidate generator aims to provide an appropriate solution space for the rest of the components. Second, to cope with the vast 6-DoF search space of source view candidates, we decided to *select* or *generate* some poses from pose trajectories. The reason why we select or generate candidates from the pose trajectories themselves is that optimal source views are more likely to be shot near the poses. This implication is intuitive because the source view shot at a pose at least covers itself and those deviating little from the pose. The candidate generator works according to the following steps: We let M be the number of partitions.

1. The candidate generator concatenates all client pose trajectories into a single one with P poses in total.

2. It then divides the concatenated pose trajectory into M partitions evenly, and transforms each partition into a source view candidate (leading to M source view candidates, thus M probing views in the context of *P-CC* detailed in later chapters).

In the rest of this chapter, we propose two versions of candidate generator, selection-based *S-Cdd* and generation-based *P-Cdd* for *S-CC* and *P-CC* respectively.

We transform candidates from the partitions instead of enumerating some from the free 6-DoF space because optimal source views are usually closer to the client trajectories. In other words, randomly selecting the source view candidates is computationally inefficient.

7.1 Candidate generator for S-CC (S-Cdd)

To consider pose trajectories from multiple HMD clients, we concatenate all the pose trajectories into a long pose trajectory. Next, we introduce a temporal downsampling factor ds on the concatenated pose trajectory. A properly selected ds allows us to speed up candidate generation without scarifying the candidate quality too much. To *select* a pose as a source view candidate from the pose trajectories, we have to determine whether selecting a pose e as a source view candidate leads to enough contributions to the synthesized novel view under the situation that a set of poses $\mathcal{E} = \{e_0, e_1, \dots\}$ has been chosen as candidates. We use function $overlap(\mathcal{E}, e)$ to denote the *coverage ratio* of e on $\mathcal{E}$, which is defined as the fraction of meshes visible at any pose in $\mathcal{E}$ that can also be viewed from the pose e. With $overlap(\mathcal{E}, e)$, we cut the concatenated pose trajectory into multiple *partitions*, where each partition consists of a set of consecutive poses with high mutual coverage ratios. More specifically, we set the first pose in the concatenated pose trajectory as the *reference* pose e and scan through the following poses. For each pose e' in the scan, we compute $overlap(\mathcal{E}, e')$ and create a new partition if $overlap(\mathcal{E}, e') < thres$, where $thres$ is an empirically chosen threshold. The procedure continues until we reach the end of the pose trajectory. The last pose in a partition is used as the reference pose for identifying the end of the next partition.

The next design decision is how to select a representative pose from each partition to be a source view candidate. Let $\mathcal{A}_{e_1,e_2}$ be a partition that starts from pose e_1 to e_2. We consider three design alternatives:

$$cdds(\mathcal{A}_{e_1,e_2}) \in \{cdds_l(\mathcal{A}_{e_1,e_2}), cdds_m(\mathcal{A}_{e_1,e_2}), cdds_a(\mathcal{A}_{e_1,e_2})\},$$

where: (i) $cdds_l(\mathcal{A}_{e_1,e_2})$ selects the last pose of $\mathcal{A}_{e_1,e_2}$, (ii) $cdds_m(\mathcal{A}_{e_1,e_2})$ selects the middle pose within $\mathcal{A}_{e_1,e_2}$, and (iii) $cdds_a(\mathcal{A}_{e_1,e_2})$ selects the pose with a coverage ratio that is closest to the average coverage ratio of all the poses in $\mathcal{A}_{e_1,e_2}$. Among these three options: $cdds_l$ opts for the most significant view change, $cdds_m$ picks the moderate one, and $cdds_a$ goes for minimal mean squared error for coverage error estimation. Algorithm 1 gives the pseudocode of our candidate generator. The algorithm has a polynomial time complexity of $O(P)$. We define len$[e]$ as the number of frames in the partition associated with source view candidate e.

Algorithm 1 Candidate Generator

$\mathcal{F} \leftarrow concat()$ ▷ concatenate all pose trajectories
$\mathcal{F} \leftarrow downsample(\mathcal{F}, ds)$ ▷ down-sample concatenated pose trajectory $\mathcal{F}$
$e \leftarrow$ first pose in $\mathcal{F}$
$\mathcal{E} = \{e\}$ ▷ current set of candidates, initially contains the first view
for $e' \in \mathcal{F}$ **do**
  if $overlap(\mathcal{E}, e') \leq thres$ **then**
    $\mathcal{E} \leftarrow \mathcal{E} \cup cdds(\mathcal{A}_{e,e'})$
    $e \leftarrow e'$
  end if
end for
return $\mathcal{E}$

7.2 Candidate generator for P-CC (P-Cdd)

We first concatenate all client pose trajectories into a single one with P poses in total. The P-Cdd candidate generator aims to control the workload on the cloud service provider. The workload can be split into: (i) *candidate generation workload*, which is a function of the number of partitions (= no. probing views) M and the number of poses P, and (ii) *coverage estimation and solver workload*, which is adjustable by a *control knob* r of the cloud service provider. Here, r is a ratio of computational complexity normalized to a *baseline*, or say *unit workload*, denoted by $m = N/P$. That is, the cloud service provider can trade-off the running time and optimality of the coverage estimator and solver by adjusting r. Higher r values lead to better synthesized view quality at the expense of higher computational complexity on the cloud service provider. The total workload h can be written as:

$$h = \frac{M}{P} + r \cdot \frac{N}{P} = \frac{M}{P} + rm, \tag{7.1}$$

where the two terms represent the workloads from the candidate generator and coverage estimator/solver, respectively. Last, we let h with a default value of 0.15 be the computational complexity budget of the cloud service provider. The problem we have in-hand is: *Given N, h, and r, find the optimal M value and transform each partition into the most representative source view candidate.* We detail these two steps in the following.

7.2.1 Optimal number of partitions (M)

The objective of this step is to maximize the probability of successfully selecting the optimal N source views. We refer to this as *success* probability, and model it conservatively using a *random arbitrary-selection* policy since we have no access to 3D content

and client statistics. This policy consists of two stages: (i) M source view candidates are selected from P poses, where the optimal N source views are among them, and (ii) the optimal N source views are selected out of the M source view candidates. To write down the success probability, we introduce a few additional symbols. We let $l = rm$ be the coverage estimator/solver workload, if not otherwise specified and $k = M/N$ be the *redundant factor* between the number of candidates (equivalent to the number of probing views) and the number of source views. Some manipulations reveal that $1 \leq k \leq h/m$ as long as $l \geq 0$.

We detail the first stage of random arbitrary-selection policy in this paragraph. In this analysis, we assume that the optimal N source views are always among the P ones. The "random" means that, in the first stage, we select M source view candidates from the pool of P poses at random. That is, each pose has equal probability of being chosen as the M source view candidates. To include N optimal views in the M candidates, we focus on the choice made by the N optimal views. Without loss of generality, we assume that the P poses are selecting their seats out of P seats and the final M candidates will be those who sit in the first M seats. Then, the event of M candidates including the optimal N source views occurs if all N optimal source views select one of the first M seats, regardless of the order of choosing.

Now, we can write the probability of the first stage as:

$$\frac{M(M-1)\cdots(M-(N-1))}{P(P-1)\cdots(P-(N-1))} \geq \frac{(M-(N-1))^N}{P^N} = (\frac{mP(k-1)+1}{P})^{mP}. \quad (7.2)$$

For the second stage, we are going to make a selection of an arbitrary number of source views out of the M candidates under the assumption that the N optimal views are included in the M candidates. Instead of analyzing the probability of optimally selecting exactly the N optimal views out of M candidates, we analyze the arbitrary number of selections of choosing $0, 1, \ldots M$ source views out of M candidates. Imagine that the selection is done by allowing each candidate to express its willingness to be the final source view or not. If a candidate is willing to be the final source view, it would express a "yes". The selection will include only those who are willing to express "yes". The reason why we use an arbitrary number of selections is because we want to exclude the "answers" from the $(M-N)$ candidates that are not optimal. The "answers" from the $(M-N)$ ones are not important because we only focus on whether all the N optimal ones say "yes". We enumerate two examples for better understanding this probability. If the $(M-N)$ ones all express "yes", they only shift the probability of selections containing less that $(M-N)$ source views to 0. At the other extreme, if the $(M-N)$ ones all express "no", they shift the probability of selections containing more than N source views to 0.

Another assumption is that as l increases, the probability of those N optimal source

views which say "yes" will increase linearly while the $(M - N)$ suboptimal ones are left unaffected. This effect is designed to be consistent with the condition that higher workload leads to a higher probability of success. Then, for the second stage, the chance that an optimal source view is selected grows linearly as the workload l grows. We know that the chance is: (i) $1/k$ for $l = 0$, i.e., the cloud service provider spends no effort to make a selection, and (ii) 1 for $l = mk$, i.e., the cloud service provider has examined all source view candidates, and thus it always makes an optimal selection. For mathematical tractability, we consider the chance to be 1 when $l = m(k-1)$ in the following analysis. First, we approximate the chance in $l \in [0, m(k-1)]$ using a linear function:

$$1/k + (1 - 1/k)(h - mk)/m(k-1). \tag{7.3}$$

So far we have analyzed the probability of saying "yes" by an optimal source view. Next, we extend our analysis to the condition that all N optimal source views express "yes" by further assuming that the probability of saying "yes" is independent of each other. Then, the probability of all N optimal source views expressing "yes" is:

$$(\frac{1}{k} + \frac{(1 - \frac{1}{k})(h - mk)}{m(k-1)})^{mP}. \tag{7.4}$$

Considering that the two stages are independent, the success probability can be written as the multiplication of Eq. (7.2) and Eq. (7.4), i.e.:

$$(m(k-1) + \frac{1}{P})^{mP}(\frac{1}{k} + \frac{(1 - \frac{1}{k})(h - mk)}{m(k-1)})^{mP}. \tag{7.5}$$

To maximize the success probability, we solve the following formulation:

$$\begin{aligned} \underset{k}{\text{maximize}} \quad & (m(k-1) + \frac{1}{P})^{mP}(\frac{1}{k} + \frac{(1 - \frac{1}{k})(h - mk)}{m(k-1)})^{mP} \\ \text{subject to :} \quad & 1 \le k \le \frac{h}{m}. \end{aligned} \tag{7.6}$$

After solving its derivatives, there exists a unique local maximum with respect to k under the constraints.

$$k = \sqrt{\frac{(m+h)(mP-1)}{m^2 P}} \approx \sqrt{\frac{m+h}{m}} \text{ as } P \to \infty. \tag{7.7}$$

7.2.2 Source view transformation for all poses of each partition

Upon determining the optimal $M = kN$, we round it to a multiple of $|\mathcal{U}|$ for fairness to all 6-DoF clients. We equally split the concatenated pose trajectory into M partitions, and transform each partition into a representative source view candidate, cdd, as follows. We write each pose as a pair of position $p = (x, y, z)^T \in \mathbb{R}^3$ and orientation in quaternion

$q = (q_x, q_y, q_z, q_w)^T \in \mathbb{S}^3$, where $\mathbb{S}^3$ denotes the unit 3-sphere. Note that we opt for quaternion to avoid mathematical singularity and rotation order ambiguity compared to Euler angles. Next, cdd of a partition of L poses is transformed by: (i) computing the vector average among p in a partition, and (ii) solving a maximum eigenvalue problem of q. In particular, we solve the following problem for a unit quaternion [21]

$$\text{cdd} = (\overline{p}, \overline{q}) = (\frac{1}{L}\sum_{i=1}^{L} p_i, \underset{q \in \mathbb{S}^3}{\text{argmax}}\{q^T(\sum_{i=1}^{L} q_i q_i^T)q\}), \tag{7.8}$$

where (p_i, q_i) denotes the i^{th} pose in this partition.

Note that we also tried motion-based candidate generation. Our prior test revealed that the motion-based implementation cut the client pose trajectories into more than 3 times the optimal number of partitions (or equivalently the number of candidates). Furthermore, the runtime is at least 10 times longer than our proposed implementation. Such implementation is not efficient enough at this stage. In the following discussion, we omit the case of motion-based candidate generator.

Chapter 8

Coverage Estimator

Recall that the candidate generator produces M source view candidates. Let $\boldsymbol{s}$ be an M-dim Boolean index vector denoting a selection of M candidates, where the i^{th} element is 1 iff the i^{th} candidate is chosen.

8.1 Scalar coverage estimator for S-CC

We build a quality estimator to predict the synthesized quality of HMD view at an arbitrary pose e using $\boldsymbol{s}$ without carrying out computationally-demanding view synthesis. The quality estimator consists of two steps. First, we build a coverage function $cvg(\boldsymbol{s}, e)$ that estimates the coverage ratio of using $\boldsymbol{s}$ to synthesize e. We use coverage ratio as an initial approximation of synthesized HMD view quality, as a lower coverage ratio indicates more information loss during warping. Second, we develop a quality function $\text{qls}(g)$ to map coverage ratio g to a quality metric, such as PSNR (Peak Signal-to-Noise Ratio) [14], SSIM (Structure Similarity Index) [14], VMAF (Video Multimethod Assessment Fusion) [26], etc.

A naive way to construct the coverage function $cvg(\boldsymbol{s}, v)$ is to synthesize each view e with all $2^M - 1$ possible $\boldsymbol{s}$, in $\mathcal{Q}$, for a lookup table. Such an exhaustive approach is clearly not feasible. Thus, we opt to estimate the coverage ratios with regression analysis based on a limited number of *modeling samples* with all $\boldsymbol{s}$ having up to k_{max} 1s, where k_{max} represents the maximal number of virtual cameras in these modeling samples. That is, for each 3D content, we synthesize each HMD view e using all $\boldsymbol{s} \in \mathcal{Q}_{k_{\text{max}}} = \{\boldsymbol{s} \mid \mathbf{1}^T\boldsymbol{s} \leq k_{\text{max}}\} \subseteq \mathcal{Q}$. $\mathcal{Q}$ denotes the set of $\boldsymbol{s}$ having up to M 1s, while $\mathcal{Q}_{k_{\text{max}}}$ denotes the set of $\boldsymbol{s}$ having up to k_{max} 1s. The $\mathbf{1}$ is a column vector with all 1s, and its dimension depends on its context. We fit these modeling samples to a coverage estimator for content-dependent model parameters. One tricky question is to determine the HMD view quality of any $\boldsymbol{s}$ with more than k_{max} 1s: the intersections/unions among source view candidates were not

considered, which may lead to estimation errors. We next present two alternative coverage estimators to control the errors. These two alternatives, *first-order coverage estimator* and *second-order coverage estimator*, aim to achieve different degrees of error control with a diverse number of model parameters.

8.1.1 First order $cvg_1(\cdot)$

For the *first-order coverage estimator* $cvg_1(\boldsymbol{s}, e)$, we compute the best column vector $\boldsymbol{b}_e$ of dimension M such that $\boldsymbol{b}_e^T \boldsymbol{s} = union_e(\boldsymbol{s})$, where $union_e(\boldsymbol{s})$ is a function that returns coverage ratio from $\boldsymbol{s}$ to a source view candidate e. Due to the limited number of parameters, we can only minimize the squared error of all considered $\boldsymbol{s}$. Particularly, to choose an optimal $\boldsymbol{b}_e$, we solve a Quadratic Programming (QP) problem:

$$\begin{aligned} \boldsymbol{b}_e^* = \underset{\boldsymbol{b}_e}{\text{argmin}} \quad & \sum_{\boldsymbol{s} \in \boldsymbol{Q}_{k_{\max}}} [\boldsymbol{b}_e^T \boldsymbol{s} - union_e(\boldsymbol{s})]^2 \\ \text{subject to:} \quad & 0 \leq \boldsymbol{b}_e \leq 1, \end{aligned} \tag{8.1}$$

for the best $\boldsymbol{b}_e^*$.

8.1.2 Second order $cvg_2(\cdot)$

For the *second-order coverage estimator* $cvg_2(\boldsymbol{s}, e)$, we introduce more parameters in an M by M matrix B_e, with the aim of reducing the estimation error of intersection and union operations on up to two sets. We hope that the matrix will have the property $\boldsymbol{s}^T B_e \boldsymbol{s} = union_e(\boldsymbol{s})$. However, the number of parameters in B_e is not enough to fit all considered $\boldsymbol{s}$. Hence, we solve the following nonlinear optimization problem:

$$B_e^* = \underset{B_e}{\text{argmin}} \quad \sum_{\boldsymbol{s} \in \boldsymbol{Q}_{k_{max}}} [\boldsymbol{s}^T B_e \boldsymbol{s} - union_e(\boldsymbol{s})]^2, \tag{8.2}$$

for the best B_e^*.

Next, we empirically map the coverage ratio to quality metrics by regression with modeling samples along with rendered HMD views using 3D content with four bunnies (more detail on the content in Sec. 10.1.1). We first run the candidate generator on the aggregated pose trajectory. We randomly select five 2-sec update windows. For each update window, we synthesize the HMD views of all candidates with multiple random $\boldsymbol{s}$, where $\mathbf{1}^T \boldsymbol{s} \in \{1, 2, \ldots 7\}$. Particularly, for each number of selected candidates $\mathbf{1}^T \boldsymbol{s}$, we randomly select 1,000 $\boldsymbol{s}$ following the recommendation [6]. We also render the corresponding ground truth HMD views at all candidate poses for each $\boldsymbol{s}$. Last, we calculate the coverage ratios and quality values of all poses of individual $\boldsymbol{s}$. The results are used as modeling samples in the following regression analysis for a quality function $\text{qls}(g)$.

We adopt PSNR and SSIM as quality metrics in our analysis, while the same procedure can be applied to other metrics. We plotted the mappings between the coverage ratio and considered quality metrics, and observed a positive and nonlinear relationship between them. To build a regression model, we try a wide range of function families, including the tangent hyperbolic, exponential function, polynomial, and arc tangent functions. We write the best-fit function between the coverage ratio and quality metric as $f_{\text{qls}}(g)$, where g is the coverage ratio.

Initially, we planned to directly use $f_{\text{qls}}(g)$ for the quality function $\text{qls}(g)$. However, we soon realized that doing so would lead to large errors when the coverage ratio approaches 0, which is counter-intuitive. To avoid such behavior, we developed the following approach for a fit with the constraint: $\text{qls}(0) \approx 0$. In particular, we introduce a step function:

$$s(g) = 1/(1 + e^{-2xg}),$$

with large enough scalar x. Next, we bind $f_{\text{qls}}(g)$ by multiplying it with a factor related to the step function and adding an offset to bring the quality estimate to 0 at $g \approx 0$. In addition to being more consistent with our intuition, adding these two constraints is equivalent to compensating for the estimation error at $g \notin [0, 1]$. That is, we relax g to take values outside $[0, 1]$, which may expand our selections of solvers, as discussed later.

With the above constraints, we have the final quality function:

$$\text{qls}(g) = (1 - s(g - 1) - s(-g))f_{\text{qls}}(g) + s(g - 1)f_{\text{qls}}(1).$$

We name the regression model from the coverage ratio to PSNR and SSIM as $f_{\text{psnr}}(g)$ and $f_{\text{ssim}}(g)$, respectively; and the corresponding quality models as $\text{qls}_{\text{psnr}}(g)$ and $\text{qls}_{\text{ssim}}(g)$. Last, for the sake of presentation, we also write $f_{\text{cvg}}(g)$ and $\text{qls}_{\text{cvg}}(g)$ if the coverage ratio is used to approximate the quality metric. Fig. 8.2 illustrates the nonlinear correlation between the coverage ratio and quality metrics. We show sample fitting curves to SSIM in Fig. 8.2, in which the dashed curve demonstrates the effects of constraints approaching 0 and 1. We note that, for clarity, the dots in the figure represent the average quality values among all samples in 1% bins, as there were in total 60,225 modeling samples. The precise regression models are:

$$f_{\text{psnr}}(g) = 11.2 + 37.9g - 77.7g^2 + 64.3g^3,$$

and

$$f_{\text{ssim}}(g) = 48.8tan^{-1}(26.4g + 43.7) - 75.0.$$

The corresponding quality functions $\text{qls}_{\text{psnr}}(g)$ and $\text{qls}_{\text{ssim}}(g)$ can be readily derived.

8.2 Pixel level coverage estimator $cvg_P(\cdot)$ for P-CC

Modern view synthesizers [2,5,8,13,19,28] do not offer a closed-form quality estimation for novel views. Therefore, we develop the coverage estimator to compute a Boolean coverage map $\mathcal{C}_{1,2}$ representing which grids of the novel view seen from cdd_2 are covered by cdd_1, where cdd_1 and cdd_2 are two source view candidates. The coverage maps have a resolution WxH, that is 1/16 of novel views if not otherwise specified. In coverage maps, 1s represent *covered*, while 0s represent *not covered*. Particularly, the coverage maps of all pairs of source view candidates are computed by the depth images of probing views of these candidates, which have a resolution WxH. More precisely, each coverage map is computed in the following steps as summarized in Fig. 8.3:

1. Sending probing requests for probing views d_1 and d_2 of cdd_1 and cdd_2.
2. Creating 3D meshes m_1 from d_1.
3. Rendering a depth image $d_{1,2}$ of m_1 seen from cdd_2.
4. Removing disocclusion of $d_{1,2}$.
5. Outputting $\mathcal{C}_{1,2}$.

Most of the steps are self-explanatory, except *mesh creation* and *disocclusion removal*. We detail them below.

8.2.1 Mesh creation

This mesh creation module is used for creating a 3D content mesh to represent the partially revealed 3D content by a source view candidate and its corresponding probing view. The illustration of mesh creation is in Fig. 8.4. We use meshes because of the following benefits: (i) it is efficient to process in common 3D rendering engines like Open3D [47] and OpenGL [44], (ii) linear interpolation of geometry between vertices is automatically implemented, and (iii) coordinate transformation is efficient because it requires only one matrix multiplication. With d_1, we create 3D meshes m_1, by distorting an *image plane mesh* with WxH vertices as our geometry proxy of the revealed 3D content. The image plane meshes are first placed 1 unit away from the viewing direction of cdd_1. Then, we distort the meshes by moving the vertices along the *projection lines* according to their distances inferred from the probing view d_1 without breaking the edges. Here, the projection lines are the lines connecting cdd_1 and the mesh vertices.

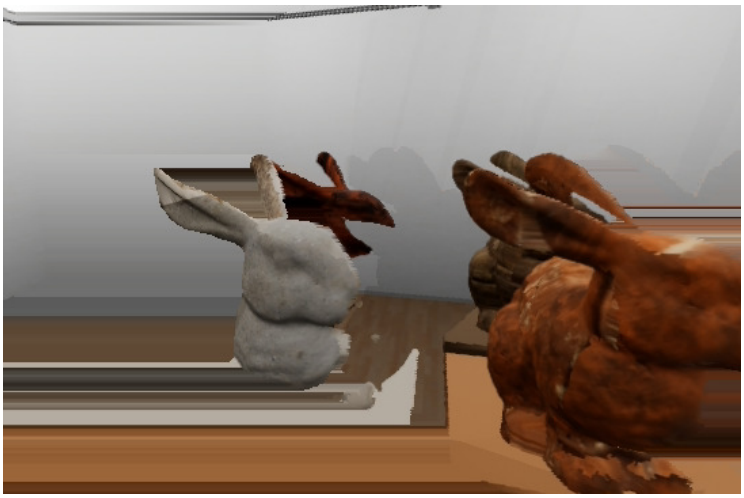
8.2.2 Disocclusion removal

We need to identify and remove the disocclusion grids in $d_{1,2}$. We build the $\mathcal{C}_{1,2}$ by initializing a matrix of dimension WxH with all 1s, and set those grids with disocculsion to 0. This is done by first computing the absolute difference map $d_{\text{abs}} = |d_{1,2} - d_2|$. There are three sources of non-zero values, called *inconsistency*, in d_{abs}: (i) $d_{1,2}$ has infinite depth values because those grids are not covered by cdd_1 at all, (ii) rendered $d_{1,2}$ suffers from depth re-projection error because d_1 has a reduced resolution, and (iii) disocclusion. The disocclusion grids are those with finite depth values in $d_{1,2}$ and larger than a *depth rejection threshold* $z = 10^{-2}$ unit. As shown in Fig. 8.5, the left figure highlights disocclusion in red rectangles. We omit discussion of the first case of inconsistency because it is trivial by checking whether the values are infinite. For the second and the third case, we identify disocclusion by checking whether those values are greater than z units.

By repeatedly computing coverage maps of all pairs of the source view candidates, we will have mutual coverage among all of them by sending M probing view requests.

(a)

(b)



(c)

Figure 8.1: Rendered HMD views with different coverage ratios, where: (a) 100.00% leads to a PSNR of 43.17 dB and an SSIM of 0.99, (b) 75.06% leads to a PSNR of 27.32 dB and an SSIM of 0.94, and (c) 50.42% leads to a PSNR of 20.55 dB and an SSIM of 0.80.

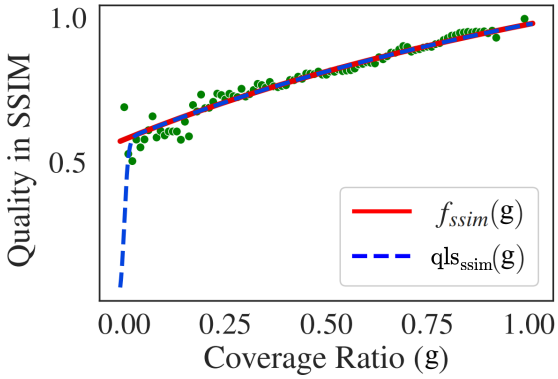


Figure 8.2: Sample regression models for SSIM.

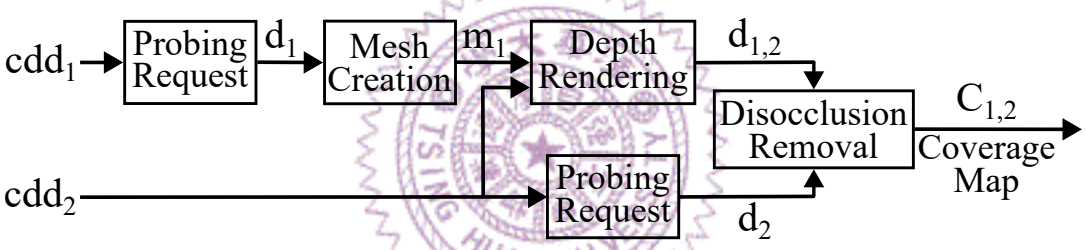


Figure 8.3: Steps of the quality estimator.

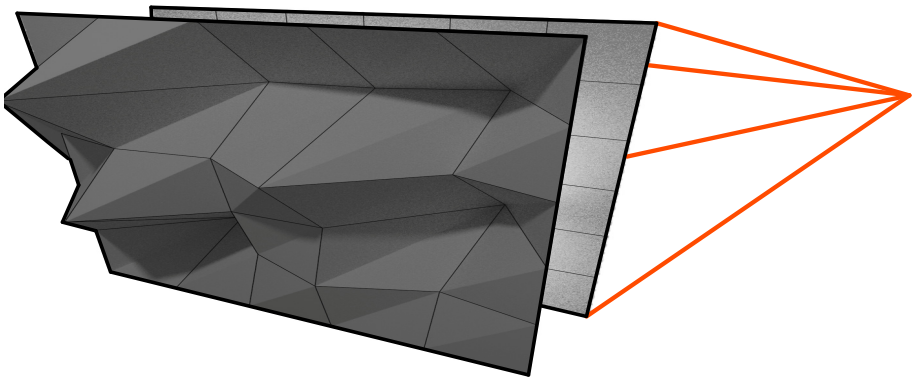


Figure 8.4: Mesh creation (dark gray mesh) by distorting an image plane mesh (light gray mesh) along the projection lines (red lines).

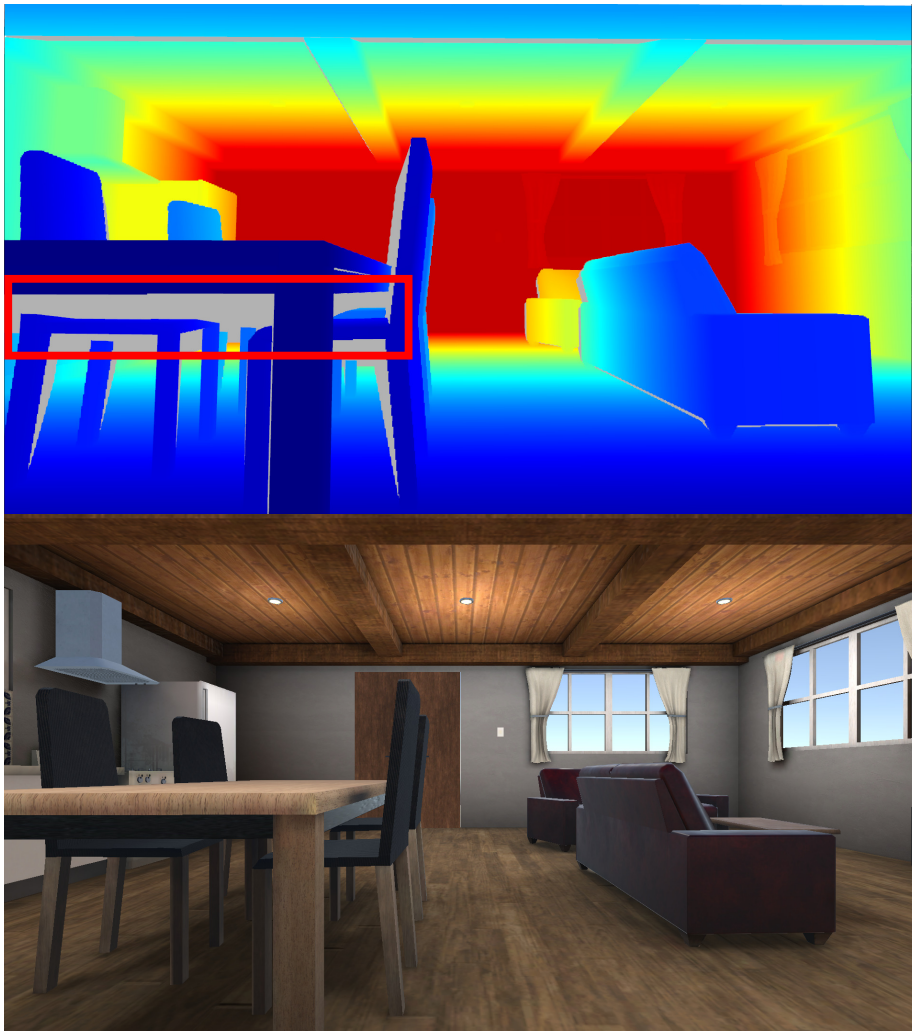


Figure 8.5: The top figure shows the results of back-projecting the pixel points (colored according to their coordinates) from the depth image, while the bottom figure shows the RGB image seen from the same pose. The red circles demonstrate where disocclusion occurs.

Chapter 9

Solver

In this chapter, we design solvers for S-CC and P-CC respectively because of the unique properties of their problem formulation.

9.1 Solver for S-CC

Thus far, we know how to estimate the synthesized HMD view quality at any pose e. Next, we aim to compute the best $\boldsymbol{s}^*$ that leads to the best quality of synthesized HMD views at all N candidates produced by the candidate generator. These candidates essentially represent all possible poses in the considered pose traces. We solve this problem with mathematical optimization, where the objective function is a weighted sum of the synthesized HMD view quality. More precisely, we let $\boldsymbol{w}$ be the weighting vector of all source view candidates. We consider two alternatives for $\boldsymbol{w}[e]$ of a source view candidate e: (i) $\boldsymbol{w}_e[e] = 1/N$ and (ii) $\boldsymbol{w}_c[e] = len(e)/(\mathbf{1}^T\boldsymbol{w})$.

9.1.1 Integer programming based solvers

We formulate our optimization problem into integer programming problems as we are making 0/1 decisions in the index vector $\boldsymbol{s}^*$. In the following context, we use $\boldsymbol{s}_{i_0,i_1,...}$ to represent the index vector with ones in the i_0^{th}, i_1^{th}, ... positions. In particular, we write the following formulations for $cvg_1(\boldsymbol{s}, e)$ and $cvg_2(\boldsymbol{s}, e)$, respectively:

$$\begin{aligned} &\underset{\boldsymbol{s}}{\text{maximize}} && \sum_{e \in \text{candidates}} \boldsymbol{w}[e]\text{qls}(\boldsymbol{b}_e^{*T}\boldsymbol{s}) \\ &\text{subject to :} && \boldsymbol{s} \in \mathcal{Q}; \end{aligned} \tag{9.1}$$

$$\begin{aligned} &\underset{\boldsymbol{s}}{\text{maximize}} && \sum_{e \in \text{candidates}} \boldsymbol{w}[e]\text{qls}(\boldsymbol{s}^T B_e^*\boldsymbol{s}) \\ &\text{subject to :} && \boldsymbol{s} \in \mathcal{Q}. \end{aligned} \tag{9.2}$$

These two formulations can be numerically solved with commercial or open-source integer programming solvers, such as CPLEX [17] and SCIP [3], which is denoted as ip in our discussion.

Algorithm 2 First-Order Greedy Solver

$\boldsymbol{q} \leftarrow zeros(M,)$
$\boldsymbol{b}_{\text{all}} \leftarrow \sum_{i=0}^{M} \boldsymbol{b}_i$
for $i \in \{0...M-1\}$ **do**
  $\boldsymbol{q}[i] = \boldsymbol{w}[i] q({\boldsymbol{b}^*_{\text{all}}}^T \boldsymbol{s}_i)$
end for
$\boldsymbol{q}, \boldsymbol{c} \leftarrow sorted(\boldsymbol{q})$
▷ gives sorted values and indices into the input array in descending order
return $\boldsymbol{c}[0:N]$

9.1.2 Greedy based solvers

However, IP solvers generally leverage the branch-and-bound approach for searching for the best feasible solution, which may result in a prohibitively long runtime. Therefore, we develop greedy algorithms referred to as gdy. Our greedy algorithms iteratively add one (for $cvg_1(\boldsymbol{s}, e)$) or two (for $cvg_2(\boldsymbol{s}, e)$) source view candidates that lead to the largest increase in the coverage ratio across all candidates. Algorithms 2 and 3 give the pseudocodes of these two gdy algorithms, where $\boldsymbol{s}_{i,j}$ denotes the vector with ones in the i^{th} and the j^{th} indices. The two greedy algorithms run in $O(M\text{lg}M)$ and $O(M^2)$, respectively.

Among the design alternatives of our novel view optimization algorithm for S-CC, the coverage models ($cvg_1(\cdot)$ versus $cvg_2(\cdot)$) and solvers (ip versus gdy) are the most critical decisions for trading off estimation error control, solution optimality, and runtime. Therefore, we define the following variants:

- **C1G** adopts the first-order coverage model $cvg_1(\boldsymbol{s}, e)$ and the greedy solver gdy, as summarized in Algorithm 2. This algorithm is provably optimal.
- **C2G** adopts the second-order coverage model $cvg_2(\boldsymbol{s}, e)$ and the greedy solver gdy, as summarized in Algorithm 3.
- **C2I** adopts the second-order coverage model $cvg_2(\boldsymbol{s}, e)$ and the integer programming solver ip for optimal solutions.

We note that we do not list **C1I** as a variant, because C1G already delivers optimal solutions at (much) shorter runtime than generic IP solvers.

Along with these variants, we also consider the following options:

- **Candidate generation strategy**: $cdds_l(\cdot)$, $cdds_m(\cdot)$, or $cdds_a(\cdot)$.
- **Modeling sample size**: $k_{\max}$.

- **Quality function**: $\text{qls}_{\text{cvg}}(g)$, $\text{qls}_{\text{psnr}}(g)$, or $\text{qls}_{\text{ssim}}(g)$.
- **Objective function weights**: $\boldsymbol{w}_e$ or $\boldsymbol{w}_c$.

In the next chapter, we empirically determine the best options for each algorithm variant.

9.2 Solver for P-CC

The solver selects N optimal source views out of M source view candidates based on coverage maps. We first model how a pixel in a novel view contributes to the synthesized novel view quality by (i) *coverage count* c which represents the number of source views covering the pixel, and (ii) *quality modeling function*, $f(c)$, which maps c onto a scalar in an interval $[0, 1)$, where

$$f(c) = 1 - e^{ac} \text{ for } a < 0, c \geq 0, \tag{9.3}$$

where a is a constant for a curve approaching 1 when c reaches 1. We let $a = \log 10^{-5}$ to give uncovered pixels higher priority to be covered, where -5 comes from the common resolution of 960x540 of our source views. Note that $f(c)$ satifies the following properties:

1. $f(c) = 0$ for $c = 0$ (zero coverage);
2. $f(c) \to 1$ as $c \to \infty$ (bounded quality);
3. $f(c_1) \geq f(c_2)$ for $c_1 \geq c_2$ (monotonic increase);
4. $f'(c_1) \leq f'(c_2)$ for $c_1 \geq c_2$ (quality saturation);

and $f(c)$ degenerates to a Boolean function $b(c)$ if $a \to \infty$:

$$\begin{aligned} b(c) &= 0 \text{ for } c \leq 0, \\ b(c) &= 1 \text{ for } c > 0. \end{aligned} \tag{9.4}$$

However, we will use $f(c)$ instead of $b(c)$ because we seek improvement from multiple view coverage and $b(c)$ cannot distinguish this condition.

Note that we use $f(c)$ as quality representation for the following two major points according to Sun et al. [35]: (i) the quality improves the most when $c = 0 \to 1$, and (ii) the quality continues to improve when $c = 1 \to \infty$ but it saturates. We do not claim that one-coverage per pixel is enough, but seek for improvement from multiple view coverage.

Next, we generalize our analysis to complete novel view and among multiple 6-DoF clients. The source view selections are represented by a sequence of Boolean decision

variables $\{s_j\} \in \{0,1\}^M$, where s_j denotes whether the j^{th} source view is selected. For convenience, we define a *null* sequence $\{s_j\} = \{0\}$ with all 0s. We also define: (i) $C_{j,i}$ is the coverage map of how cdd_j covers cdd_i, (ii) $\bigodot$ is a composite operation that first multiplies the two matrices elementwisely and sums over the elements inside the resulting matrix, (iii) $\mathcal{W}$ is an *averaging mask* with the same dimension as the probing views, and thus $\mathcal{W} \bigodot A$ averages the elements in a matrix A. Last, we define *aggregated quality* $\text{qlp}(\cdot)$ to be the expected overall quality under a source view selection $\{s_j\}$. With these symbols, we expand Eq. (5.1) using Eq. (9.3) into:

$$\begin{aligned} \underset{\{s_j\}}{\text{maximize}} \quad & \text{qlp}(\{s_j\}) = \mathcal{W} \bigodot \sum_{i}^{M} 1 - e^{a(\sum_j^M s_j C_{j,i})} \\ \text{subject to :} \quad & s_j = \{0,1\} \text{ for } 1 \le j \le M \\ & \sum_{j}^{M} s_j = N. \end{aligned} \tag{9.5}$$

In this formulation, the aggregated quality is monotonically increasing and is nonnegative with respect to more 1s in $\{s_j\}$ since $f(c)$ itself is also monotonically increasing. Mathematically speaking, increasing $\{s_j\}$ is defined to be:

$$\{s_j\}_1 \ge \{s_j\}_2 \text{ if } (s_j)_1 \ge (s_j)_2 \text{ for all } j, \tag{9.6}$$

where $(s_j)_k$ denotes the j^{th} element in sequence $\{s_j\}_k$. Such property can be leveraged by solver algorithms for effectively refining source view selections. We propose three solver algorithms for P-CC below.

9.2.1 Uniform (Uni)

We pick the source view candidates every fixed skips to ensure uniform source view distribution along the temporal domain and among all 6-DoF clients. Note that *Uni* does not require output from the the coverage estimator. The pseudocode is shown in Alg. 4.

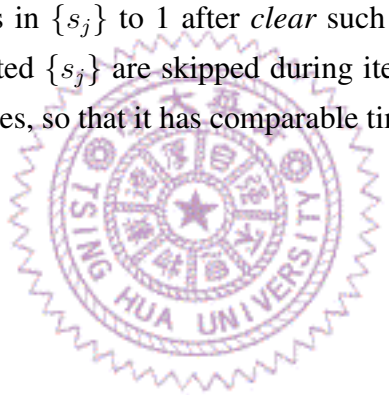
9.2.2 Branch & Bound (BB)

We initilize $\{s_j\} = \{0\}$ and mark all elements of $\{s_j\}$ as *undetermined*. We perform *branch* or *bound* operations on $\{s_j\}$ until the number of branching exceeds a pre-defined threshold *maxNodes*. At that time, the best known $\{s_j\}$ is returned. We empirically set *maxNodes* to 96 if not otherwise specified. Here, we define *ub*$(\{s_j\})$ as the aggregated quality of the sequence that sets all of undetermined 0s in $\{s_j\}$ to 1, which results in the maximum aggregated quality given that some of the elements have been determined, and the terminal sequence is a sequence with exactly N 1s. More precisely, if $\{s_j\}$ is not a

terminal sequence, we *branch* the current sequence $\{s_j\}$ into two sequences. The first one is created by setting one of its 0s to 1 such that the aggregated quality increases the most, and we mark the element s_j as *determined*, while the other one is created by setting s_j to 0 and we also mark it as *determined*. Moreover, we remove the current sequence $\{s_j\}$ from our searching list if either ub$(\{s_j\})$ is lower than our currently best aggregated quality or if it is a terminal sequence. Whenever we encounter a terminal sequence, we update the current best sequence if its aggregated quality is greater than before. The algorithm is presented as pseudocode in Alg. 5.

9.2.3 Uniform & Modify (UM)

We start from the $\{s_j\}$ returned from *Uni*, and we always iterate $\{s_j\}$ among all terminal sequences. More specifically, each iteration consists of two operations: (i) *clear*, which clears one of the 1s in $\{s_j\}$ such that the updated aggregated quality is maximal, and (ii) *set*, which sets one of the 0s in $\{s_j\}$ to 1 after *clear* such that the resultant aggregated quality is maximal. Duplicated $\{s_j\}$ are skipped during iterations. The algorithms stop after $M/(N+M) \cdot$ maxNodes, so that it has comparable time complexity as *BB*, and the best known $\{s_j\}$ is returned.



Algorithm 3 Second-Order Greedy Solver

$Q \leftarrow zeros(M, M)$

$B \leftarrow \sum_{e \in \text{candidates}} B_e$

for $i \in \{0...M-1\}$ **do**

  for $j \in \{0...M-1\}$ **do**

    $Q[i,j] \leftarrow \boldsymbol{w}[i]\text{qls}(\boldsymbol{s}_{i,j}^T B \boldsymbol{s}_{i,j})$

  end for

end for

$Q, \boldsymbol{c} \leftarrow sorted2(Q)$ ▷ gives sorted values and 2D indices into the input array; $\boldsymbol{c}$ can be viewed as a 3D array, and $\boldsymbol{c}[i]$ gives a 2D index(i,j)

$x, \text{acc} \leftarrow N, 0$

$Y \leftarrow []$

while $x > 1$ **do**

  $i, j \leftarrow \boldsymbol{c}[\text{acc}]$

  if $i \neq j$ **then**

    append i, j to Y

    $x \leftarrow x - 2$

    $\text{acc} \leftarrow \text{acc} + 2$

  end if

  if $i = j$ **then**

    append i to Y

    $x \leftarrow x - 1$

    $\text{acc} \leftarrow \text{acc} + 1$

  end if

  delete all indices that have either i or j after the acc$^{\text{th}}$ element of $\boldsymbol{c}$

end while

if $x > 0$ **then**

  delete all indices with its first element $\neq$ its second element

  $i, i \leftarrow \boldsymbol{c}[\text{acc}]$

  append i to Y

end if

return Y

Algorithm 4 Uniform

$\{s_j\} \leftarrow \{0\}$
for $j \in$ round$([0, M/N, 2(M/N), ..., M-1])$ **do**
    $s_j \leftarrow 1$
end for
return $\{s_j\}$

Algorithm 5 Branch & Bound (BB)

q, lb, sol, count $\leftarrow [\{0\}], 0, \{0\}, 0$ ▷ q is a max priority queue
while count $<$ maxNodes, and q is not empty **do**
    $\{s_j\} \leftarrow q$.pop()
    if $\{s_j\}$ is not a terminal state, and ub$(\{s_j\}) \geq$ lb **then**
        count $\leftarrow$ count $+ 1$
        for each $\{s_j\}_b$ by branching $\{s_j\}$ **do**
            if qlp$(\{s_j\}_b) >$ qlp(sol) **then**
                sol, lb $\leftarrow \{s_j\}_b$, qlp$(\{s_j\}_b)$
            end if
            push $\{s_j\}_b$ to q with priority qlp$(\{s_j\}_b)$
        end for
    end if
end while
return sol

Algorithm 6 Uniform & Modify (UM)

count, lb, sol $\leftarrow 0, 0, Uni()$
while (count $< \frac{M}{N+M} \times$ maxNodes) **do**
    count $\leftarrow$ count $+ 1$
    $\{s_j\} \leftarrow$ set(clear(sol))
    if qlp$(\{s_j\}) \geq$ qlp(sol) **then**
        sol $\leftarrow \{s_j\}$
    end if
end while
return sol

Chapter 10

Performance Evaluations

We implement and evaluate our proposed algorithms for both S-CC and P-CC in this chapter. In this chapter, we evaluate the performance of our proposed novel view optimization algorithm using real testbeds.

The VMAF we are using is version 2.3.1 and was pretrained by the VMAF authors. The dataset consists of 7 real-life videos and 2 animations, which is quite different from our photo-realistic content. As a result, we cannot make a fair comparison with VMAF. Nevertheless, we still report VMAF for the reader's reference.

10.1 Evaluations of S-CC

In this section, we are going to evaluate system performance for S-CC only. Evaluation of P-CC systems is in the next section.

10.1.1 Testbed implementation

To render the ground truth HMD views, we control virtual cameras in a photorealistic simulator, called AirSim [34], which is built upon Unreal Engine [10] and can be extended to evaluate networked systems [37]. For HMD view synthesis, we employ MPEG TMIV [23] from the MPEG Immersive Video (MIV) standard [22]. More specifically, we built a testbed as illustrated in Fig. 10.1, for our performance evaluations. This testbed consists of five key components: (i) HTC Vive Pro Eye that collects the 6-DoF pose trajectories, (ii) AirSim that renders source view and HMD views for given pose trajectories of 3D content, (iii) our novel view optimization algorithm that selects some source candidates for HMD view synthesis, (iv) a TMIV renderer that synthesizes HMD views, and (v) a metric evaluator that quantifies the objective performance by comparing the HMD and ground truth views. In addition to AirSim and TMIV, we develop scripts using the

Unreal Engine blueprint [41] to log pose trajectories of HTC HMD users, which consist of timestamps, location (x, y, z), and orientation (roll, pitch, yaw) at 30 fps (frames-per-second). We implement the metric evaluator in Python script to compute the objective metric values. The HTC Vive HMD is tethered to a workstation with an Intel i5-9600K CPU and an NVIDIA GeForce RTX 2080 Ti GPU. Based on user feedback, we configure AirSim/Unreal Engine to render at 90 fps to avoid cybersickness.

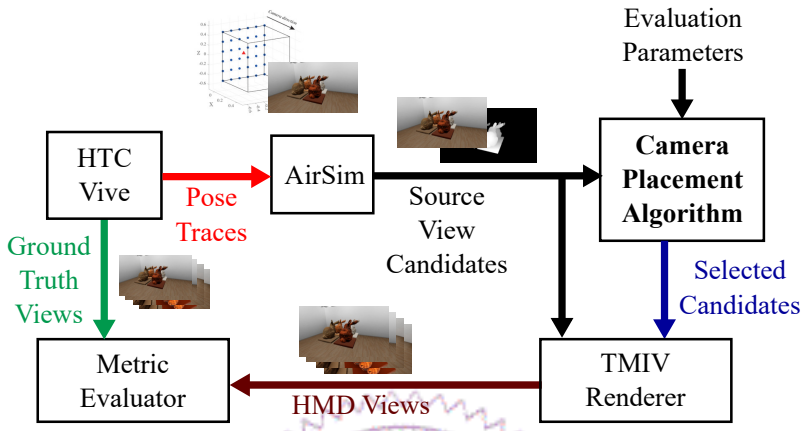


Figure 10.1: Testbed for performance evaluations.

10.1.2 Setup for S-CC

To fairly compare the performance of different solver algorithms, we first use our testbed to collect pose trajectories from 16 subjects (7 of them are female). These subjects are between 21-29 years old, with 20/25 (corrected) vision, and do not personally own HMDs. We build 3D content with a dimension of 30 m $\times$ 30 m $\times$ 10 m, which is large enough for each subject to navigate without bumping into the walls/ceiling/floor. We create two sample set of 3D content with one and four bunnies, where the height of each bunny is 5 m. We collect pose trajectories from each subject as follows.

1. First, we calibrate the HMD and align the physical origin with the content's origin.
2. We next place the bunnies at the eye-level of each subject.
3. We randomly place the subject in the 3D content, and make him/her face the bunnies.
4. We ask the subject to freely navigate in the 3D content.
5. We configure AirSim/Unreal Engine to prevent the subject from bumping into objects, like walls or bunnies.

After collecting all pose trajectories, statistics show that the shortest pose trajectories among the subjects is 62 seconds, so we cut the pose trajectories of all the other subjects into the same duration. Therefore, for each set of 3D content, we collect sixteen 62-second pose trajectories, where each pose trace contains 1860 poses. Note that we spent around 5 hours to rendering the views of each pose trace for our performance evaluations.

We use the collected dataset (3D content and pose trajectories) to run the experiments on our testbed to quantify the performance of our novel view optimization algorithm variants: C1G, C2G and C2I for S-CC. We are not aware of any prior work that solves the considered novel view optimization problem without content access. Hence, we compare against two algorithms: *Set Cover (SC)* and *Upper Bound (UB)*. SC works as follows, it: (i) identifies a set of 3D primitives observed by pose trajectories, (ii) creates a table from each source view candidate to covered primitives, and (iii) greedily selects the candidate that covers the most primitives. As a heuristic algorithm, SC has been shown to perform reasonably well in practical set cover problems [42, Chapter 2]. In addition, we emphasize that SC gets access to 3D primitives, and thus has an *edge* over our proposed C1G, C2G and C2I. We still consider SC as a baseline algorithm to be conservative. UB, on the other hand, chooses all source view candidates to serve as an impractical performance bound. We use UB for benchmarking purposes.

We set update window = 60, $thres = 0.7$, and $ds = 5$ throughout the evaluations. We vary the following parameters to study their implications for performance:

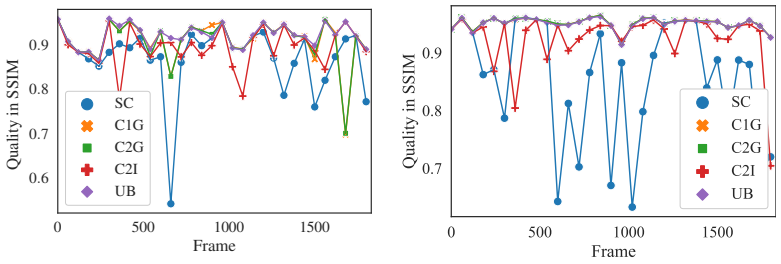
1. *cdds* $\in \{\underline{l}, m, a\}$,
2. $k_{\max} \in \{2, \underline{3}, 4\}$,
3. $\boldsymbol{w} \in \{\underline{\boldsymbol{w}_e}, \boldsymbol{w}_c\}$, and
4. $\text{qls}(g) \in \{\underline{q_{\text{cvg}}(g)}, q_{\text{psnr}}(g), q_{\text{ssim}}(g)\}$,

where underlines indicate default values. We also vary the number of source views $N \in \{\underline{16}, 17, \ldots M\}$ to adjust the resource scarcity. For the performance metrics, we use PSNR, SSIM, and VMAF to measure the HMD view quality. We also measure per-component and overall runtime on our i5 workstation. We report average results in figures and tables with standard deviations whenever possible.

10.1.3 Results for S-CC

Sample per-subject results. We compute the performance of individual subjects (pose trajectories) and give results[1] from sample subjects in Fig. 10.3. Fig. 10.2(a) shows

[1]Due to the space limit, we omit significance analysis and only discuss selected quality metrics here, and in the rest of this section for S-CC.



(a) (b)

Figure 10.2: Sample synthesized HMD view quality: (a) a sample subject in the one bunny content, and (b) a sample subject in the four bunnies content.

the SSIM dynamics from a random subject, in which a point represents average quality among 60 consecutive frames. Fig. 10.3(c) presents the SSIM results from sample subjects ranked by their UB performance in descending order. We observe that our C1G, C2G, and C2I algorithms generally outperform the baseline SC and approach the benchmark UB. For a more complete view, we report overall results from all subjects in the following.

Best *cdds* option for the candidate generator. As shown in Fig. 10.4 with improvement shown with respect to $cdds = cdds_l$, we conclude that different choice *cdds* leads to performance improvement for different algorithms. The candidate generator is a crucial component in the overall system since it determines the quality of inputs to all the following modules. Quality of inputs limits the performance upper bound given the best quality estimator and solver we can achieve. We perform experiments to study how different algorithms react to the different choice of *cdds*. Focusing on the UB algorithm, we find that $cdds_a$ improves quality. Thus, we recommend $cdds_a$ as the default option. We also found out that C1G and C2G perform best with $cdds_l$ due to the fact that greedy algorithms prefer a set of candidates with maximal difference among them. Meanwhile, we conclude that C2I prefers $cdds_m$ and $cdds_a$. The reason for $cdds_m$ is that it takes the user velocity into consideration implicitly. For example, if a user moves faster at the beginning and slower at the end of a content change (coverage ratio has fallen behind the $thres$ with respect to the reference), $cdds_m$ will choose the pose which is closer to the end, which will result in a greater number of frames to have high coverage ratio. On the other hand, $cdds_a$ provides higher accuracy of coverage ratio estimation inside a parti-

Table 10.1: Overall Coverage Ratio Achieved by UB with Different *cdds* Options

content	$cdds_l$	$cdds_m$	$cdds_a$
1 Bunny	$96.30\% \pm 3.57\%$	$95.81\% \pm 5.55\%$	$96.19\% \pm 3.99\%$
4 Bunnies	$96.01\% \pm 3.91\%$	$95.55\% \pm 5.88\%$	$95.81\% \pm 5.21\%$

tion. The integer programming solver prefers this option because it is more sensitive to the numerical perturbation. As a result, higher coverage ratio estimation implies a more precise solution. We recommend to use $cdds_l$ for greedy algorithms, and $cdds_a$ or $cdds_m$ for integer programming based algorithms. Our candidate generator supports three cdd options. To understand the quality of the resulting candidates, we report the overall coverage ratio resulting from UB in Table 10.1. This table reveals that cdd_l leads to the highest coverage ratio and the lowest standard deviation. Adding to that, cdd_l also has the lowest computational complexity. Hence, we present results from cdd_l in the rest of this thesis.

Best $k_{\max}$ option for individual coverage models. We analyze the implications of

Table 10.2: RMSE and Running Time of Different $k_{\max}$

$k_{\max}$	2		3		4	
Coverage Model	cvg_1	cvg_2	cvg_1	cvg_2	cvg_1	cvg_2
RMSE	0.44	0.18	0.43	0.08	0.49	0.08
Init. Time (s)	32.78		36.82		59.53	
Est. Time (s)	0.0078	0.0050	0.058	51.64	0.393	511.27

$k_{\max}$ in terms of computational times and Root Mean Square Error (RMSE) with each coverage model. The computational time is composed of: (i) the *initialization time* that generates the candidates and model data structure, and (ii) the *estimation time* that evaluates the coverage function. Note that these two metrics are measured with the coverage models: cvg_1 and cvg_2. Table 10.2 gives the overall results. We observe that cvg_2 has an RMSE reduction of at least 60% over cvg_1 with all $k_{\max}$. We also observe that the estimation time increases exponentially as $k_{\max}$ increases since the estimator needs to evaluate a least-squared-error problem with $O(2^{k_{\max}})$ terms. With cvg_2, the estimation time of $k_{\max} = 4$ has grown by 10 times compared to that of $k_{\max} = 3$; however, the RMSE remains the same. Hence, we recommend $k_{\max} = 3$ for a good tradeoff between the computational time and degree of error. We give results from $k_{\max} = 3$ below if not otherwise specified.

Our algorithm delivers high-quality synthesized HMD views. We report the overall performance of different algorithms in Fig. 10.5. Fig. 10.5(a) reveals that our algorithm achieves a comparable coverage ratio with SC. However, Figs. 10.5(b)–10.5(d) clearly demonstrate that our C1G, C2G, and C2I perform much better than SC in terms of PSNR, SSIM, and VMAF: boosts of 2.37 dB, 0.05, and 10.84 are observed in the one bunny scene, while boosts of 1.90 dB, 0,05, and 9.39 are observed in the four bunnies scene. We also observe that the four bunnies scene leads to worse results, which may be attributed to its increased complexity compared to the one bunny scene. Since our algorithm performs much better than SC, we no longer consider SC in the following. Another observation we

can make in Fig. 10.5 is that our C1G, C2G, and C2I approach the quality achieved by UB: gaps as small as 1.00 dB, 0.01, and 2.30 in PSNR, SSIM, and VMAF are reported. Note that UB selects *all* source view candidates, and could lead to higher throughput. We depict the throughput of the selected source views in Fig. 10.6, which shows that at least 28.43% higher throughput is incurred by UB, compared to our C1G, C2G, and C2I. In summary, our algorithm delivers quality fairly close to UB at a reduced network throughput.

Performance comparison among our algorithm variants. We compare the performance of different algorithm variants using sample winner-loser figures in Fig. 10.7. The percentages in this figure are calculated by comparing the coverage ratio between two algorithms per pose from all subjects. Notice that two algorithms may achieve the same coverage ratio with some poses, and these poses will not be counted in Fig. 10.7. This figure shows that C2I performs better than C1G and C2G in both scenes. C2I outperforms C2G probably because the solver ip gives better solutions than gdy, since they employ the same coverage model $cvg_2(\boldsymbol{s}, e)$. Moreover, C1G performs worse than C2G due to its simpler first-order coverage model. In summary, Fig. 10.7 confirms the effectiveness of more comprehensive: (i) (second-order) coverage model and (ii) (optimal) solver.

Objective function weights do not affect the algorithm performance with our dataset.

Table 10.3: Quality Metrics of Different Choices of $\boldsymbol{w}$

$slvr$	**PSNR**		**SSIM**		**VMAF**	
	$\boldsymbol{w}_e$	$\boldsymbol{w}_c$	$\boldsymbol{w}_e$	$\boldsymbol{w}_c$	$\boldsymbol{w}_e$	$\boldsymbol{w}_c$
SC	22.31	22.31	0.87	0.87	48.49	48.49
C1G	23.99	23.99	0.90	0.90	54.93	54.93
C2G	24.04	24.76	0.90	0.91	55.26	57.91
C2I	24.74	24.57	0.92	0.91	59.95	59.31
UB	25.85	25.85	0.93	0.93	63.10	63.10

As shown in Table 10.3, we observe that all the algorithms perform almost the same on both weighting vectors. The effect of weighting is not significant for all quality metrics except for a gain of 2 in VMAF for C2G. We analyze the distribution of $\boldsymbol{w}_c$ generated in our experiments, and the element-wise standard deviation from $\boldsymbol{w}_e$ is 0.012. One conclusion is that, for either $slvr$, the weighting effect is not strong enough to change a bit in $\boldsymbol{s}$. With our analysis, there is a distribution of such weighting vectors which will lead to the same solution by a solver. We also analyze and plot the choice of $f_{\text{qls}}(.)$ in Fig. 10.8 and find out that there is little effect among each choice. We conclude that either choice of $\boldsymbol{w}$ and $f_{\text{qls}}(.)$ makes no difference, and we leave it as the default values.

Our algorithm works well with the coverage ratio as an approximated quality

function. So far, we have been using the coverage ratio to approximate the HMD view quality. We vary the quality function $\text{qls}(\cdot)$ and find little, if any, impact on the resulting HMD view quality in Fig. 10.8. For example, with C1G in the one bunny scene, applying $\text{qls}_{\text{psnr}}(\cdot)$ only improves 0.0 dB of PSNR, resulting in no improvement, compared to applying $\text{qls}_{\text{cvg}}(\cdot)$. Similarly, with C2I in the four bunnies scene applying $\text{qls}_{\text{ssim}}(\cdot)$, the improvement of SSIM is merely 0.38. These results back up our earlier intuition: the coverage ratio can approximate the synthesized HMD view quality without the overhead of building the quality models. More specifically, instead of building such quality estimator of SSIM or PSNR as shown in Fig 8.2, we can just use the coverage ratio for quality function.

The performance of our algorithm better approaches that of UB with more source views. Last, we vary the number of source views $N \in \{16, 17, \ldots, M\}$ to see how much quality we could improve with more cameras[2]. Fig. 10.9 shows the gap between our algorithm and UB. We observe in Fig. 10.9(a) that in both scenes, the gap reduces to 0 for C1G, C2G, and C2I when N approaches M. A similar observation can be drawn for SSIM in Fig. 10.9(e). That is, our algorithm approaches optimum performance with enough number of source views.

Key findings. We found that: (i) our algorithm outperforms the baseline algorithm SC by at least 2.37 dB in PSNR, 0.05 in SSIM, and 10.84 in VMAF (see Fig. 10.5); (ii) the optimality gap of our algorithm is as small as 1.00 dB in PSNR, 0.01 in SSIM, and 2.30 in VMAF (see Fig. 10.5); and (iii) our algorithm further approaches the performance of impractical UB when the number of source views increases (see Fig. 10.9). Taking runtime into consideration, we recommend using C2I if the computational resources are abundant. Otherwise, we recommend using C2G for a good tradeoff between runtime and optimality.

10.2 Evaluations of P-CC

Note that when we are evaluating the candidate generator, we omit comparison of the proposed approach and some naive candidate generations such as random and downsampling since the best sampling methods are determined in Ch. 10.1.

10.2.1 System implementation

We implement the cloud service provider in Python 3.9 with the help of Pytorch [29] with Cuda [27] to facilitate GPU computation and FFmpeg [40] to encode/decode RGB-

[2]We empirically decided to start from 16 source views based on our pilot test results.

D views. Each key component of the novel view optimization algorithm (see Fig. 4.1) is implemented as a Python module for better maintainability. We implement the content creator with Open3D [47] to generate the source and probing views. Last, 6-DoF clients run on Oculus Quest 2 as their HMD. For client-side view synthesis, we opt for RVS synthesizer [19] on top of OpenGL [44] based on performance comparison [35]. Overall system implementation is summarized in Fig. 10.10. Unfortunately, common HMDs are not powerful enough to host the view synthesizer, and thus we offload it to a workstation.

To fairly compare the performance of various novel view optimization algorithms, we implement a pose trajectory collection system in Unity [12] and Air-Light-VR [1]. We consider three sets of content with different levels of texture details and geometry complexity: (i) *House* (17x13x5 meter^3, 156 objects) has 3 rooms with comprehensive texture and furniture, (ii) *Bigroom* (45x12x4 meter^3, 260 objects) has a room with robotic arms and simple texture, and (iii) *Smallroom* (7x12x4 meter^3, 360 objects) has a small server room with simple texture. Fig. 10.11 shows the three sets of content. We recruited 16 subjects between 21 and 29 years old, and collected 16 pose trajectories for each set of content, where each pose trajectory lasted for 30 seconds at 50 frames per second. We replayed them in Unity to render novel views at resolution 960x540 as the ground truth for comparison. After collecting the pose trajectories, we observed that users preferred close-up views in *Bigroom* and *Smallroom* since objects are smaller than those in *House*. To evaluate novel view optimization algorithms, we compared the synthesized novel views and the ground truth novel views for several quality metrics.

10.2.2 Experiment setup

For performance comparison, we employ *C2G* and *C2I* solver algorithms and the *S-Cdd* candidate generator algorithms. We also consider an unrealistic upper-bound *Opt* (= UB), and we will use them interchangably. We will study how the choice of candidate generator affects the system performance in later discussion.

Parameter options. Our parameterized algorithms come with the following parameters, where underlines indicate default options.

1. $N \in \{8, 16, \underline{24}, 32, 40\}$.

   i.e., $m = \{0.01, 0.02, \underline{0.03}, 0.04, 0.05\}$,

   and $M \in \{32, 32, \underline{48}, 64, 80\}$ according to Eq. (7.7).

2. solver $\in \{$*C2G*, *C2I*, *Uni*, *BB*, <u>UM</u>, *Opt*$\}$.

3. candidate generator $\in \{$*S-Cdd*, <u>P-Cdd</u>$\}$.

The underlining shows the default options, and other parameters are left to their default values if not otherwise specified. We measure runtime and bandwidth consumption for each round of experiments, and we assess quality in Peak Signal-to-Noise Ratio (PSNR) [14], Structural Similarity (SSIM) [14], and Video Multi-Method Assessment Fusion (VMAF) [26]. We run the algorithms on a workstation with an AMD Ryzen 7 5700X 8-core CPU and NVIDIA Geforce RTX 3090 Ti GPU. We report all statistics after averaging the results along the temporal axis and among all 6-DoF clients. All error bars are presented with 95% confidence intervals.

We will discuss how a specific parameter/component impacts the system performance while leaving others to default. In the rest of this thesis, all statistics are reported after averaging the results along the temporal axis and among all 6-DoF clients. All error bars are presented with 95% confidence intervals.

Visual inspection of sample results. We present sample synthesized results from the *House* content under default parameters in Fig. 10.12(a) which shows that our synthesized novel views are mostly of good visual quality. We also show two novel views with the highest and lowest quality as examples. Observing Fig. 10.12(c), we see some distortion and blur regions under the table and at the edges of the novel view. We conduct more extensive experiments in the following.

Quality improvement as N increases. We conduct experiments by varying the choice of N and leave other parameters to default. As shown in Fig. 10.13, quality increases rapidly when $N<24$ but grows slowly when $N \geq 24$ in terms of all quality metrics. This observation is consistent with the design of $f(c)$ in Ch. 9. Among these three quality metrics, our system is less stable for optimizing VMAF because the optimization objective in Eq. (9.5) optimizes aggregated quality without explicitly taking temporal continuity into account. Moreover, *Bigroom* and *Smallroom* result in lower quality because subjects prefer to take close-up views there. Hence, the same number of source views cover a smaller portion of *Bigroom* and *Smallroom*.

Bandwidth reduction by novel view synthesis. We demonstrate that our blind streaming system saves bandwidth by comparing it with traditional novel view streaming that sends customized novel views to individual clients. We use x264 [31] to encode source and novel views with $qp = 0$. We report source view bandwidth consumption in RGB and D separately in Fig. 10.14 with diverse choices of $N \in \{8, 16, 24, 32, 40\}$. The figure reveals that, with merely 16 clients, the noreal view synthesis cuts the bandwidth consumption by 94% at $N = 24$ for the content creator. The improvement will become even larger when the number of clients is increased. Based on the trade-off between quality Fig. 10.13 and bandwidth Fig. 10.14, we recommend using $N = 1.5|\mathcal{U}|$.

Proposed algorithms outperform state-of-the-art ones. We first compare *S-Cdd*

and *P-Cdd*. Because the *S-Cdd* generator generates candidates based on coverage drop, it results in less than 24 source view candidates under our default setup. Therefore, we have to set $N = 8$ for this comparison. Briefly, we feed the source view candidates to default *UM*. The results are shown in Fig. 10.15. We find that the *P-Cdd* outperforms *S-Cdd* in all quality metrics across all content. As a result, we suggest using *P-Cdd*, which would not: (i) produce non-uniform source view candidates or (ii) overload the content creator.

Next, we pass the source view candidates to different solver algorithms in $\{$*C2G*, *C2I*, *Uni*, *BB*, *UM*, *Opt*$\}$ and summarize the results with $N = 24$ in Fig. 10.16. Our proposed solvers algorithms for P-CC consistently outperform the state-of-the-art ones by at least 2.27 dB in PSNR, 0.04 in SSIM, and 12 in VMAF, and leave performance gaps as small as 0.75 dB in PSNR, 0.009 in SSIM, and 3.8 in VMAF for all 3D content. *Uni* solver performs relatively well in terms of VMAF since it guarantees uniform source view distribution along the temporal axis, which is only considered by VMAF. Based on Fig. 10.16, we recommend using *UM* solver for higher quality. Cross-referencing Fig. 10.15, we note that even with suboptimal source view candidates from S-Cdd, our UM solver algorithm still leads to fairly good quality.

Comparison of different content could also be inferred from Fig. 10.16. We observe that for high quality texture content such as *House*, clients prefer to watch the same spots. On the other hand, the other 2 sets of content consist of relatively low quality textures and lack some focus points. As a result, we conclude that our algorithms improve the quality further if the content itself has some focus points.

Runtime of our algorithms. We divide the total runtime into: candidate generator, coverage estimator, and solver. We report the runtime distribution with the default parameters in Fig. 10.17. Runtime for candidate generator is negligible since it only requires one scan through the pose trajectories. The coverage estimator consumes relatively longer runtime because of the coverage map computation (see Ch. 8) using Open3D [47], which is a CPU implementation. In fact the total runtime is dominated by the solver, which grows quadratically with N because it evaluates aggregated quality frequently for each update window. Note that we execute the solver algorithms on a single-threaded GPU. Neither do we parallelize the aggregated quality evaluations although several operations such as *branch*, *clear*, *set* in *BB* and *UM* are ready for multi-threaded executions. We opt not to enable multi-threading in solver algorithms, because the coverage estimator is single-threaded. Combining the results from Figs. 10.16 and 10.17, we suggest using the *Uni* solver, which runs faster than 100ms and is suitable for real-time operations. When runtime is not a concern, *UM* is recommended for the best quality.

Correlation between quality metrics and aggregated quality.

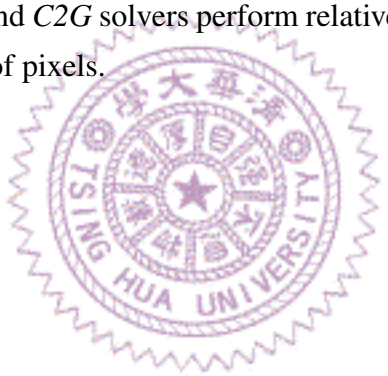
For justifying the representativeness of the aggregated quality, we report the corre-

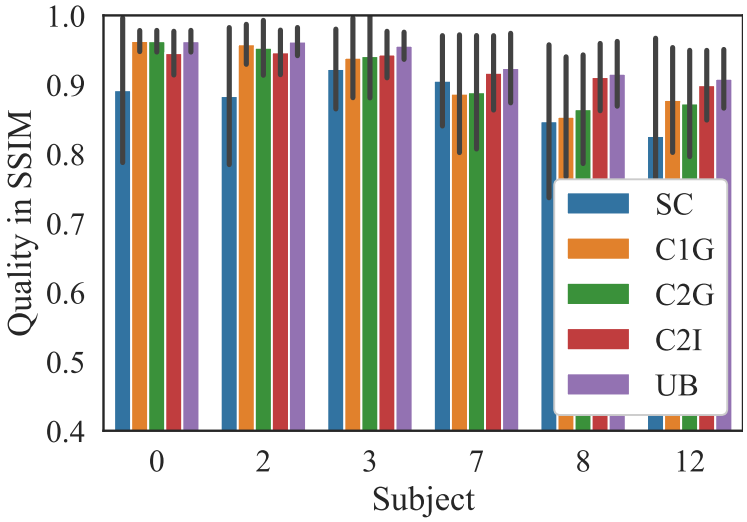
Table 10.4: Correlation Between Quality Metrics - Aggregated Quality

Metrics	PSNR	SSIM	VMAF
Pearson	0.41	0.66	0.51
Spearman	0.27	0.40	0.48

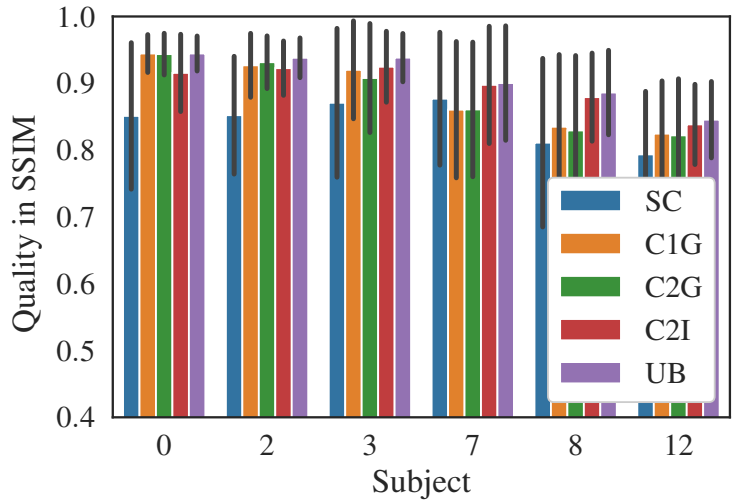
lation coefficients in Pearson [9] and Spearman [25] in Table 10.4 and Fig. 10.18. The correlation coefficients are calculated every source view update, and quality metrics are averaged across all novel views corresponding to that update. Statistics show that there exist positive relationships between the aggregated quality and the quality metrics.

Coverage counts distribution. We report counts of a random candidate for different levels of coverage and solvers with the *House* content in Fig. 10.19 by conducting a small experiment. We observe that the proposed *UB* solver performs as expected. It covers most of the pixels one time, and there are only a few pixels that get covered more than 2 times. On the other hand, the *C2I*, and *C2G* solvers perform relatively worse by the fact that they do not cover a huge portion of pixels.

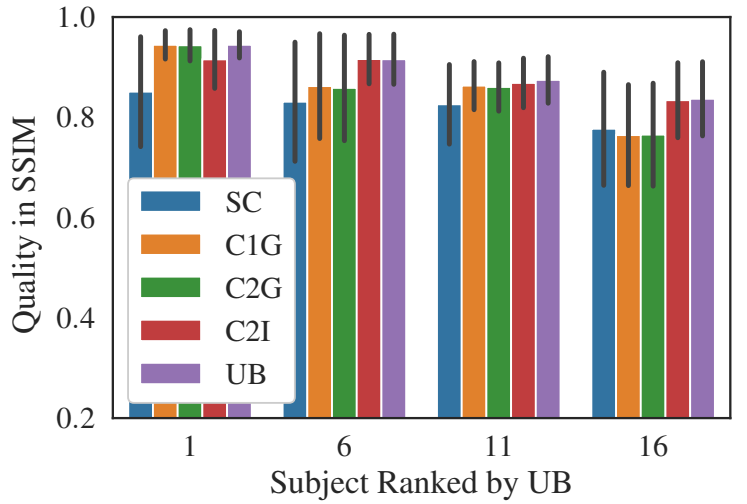




(a)



(b)



(c)

Figure 10.3: Sample synthesized HMD view quality: (a) for 1-bunny content, and (b)–(c) for 4-bunny content.

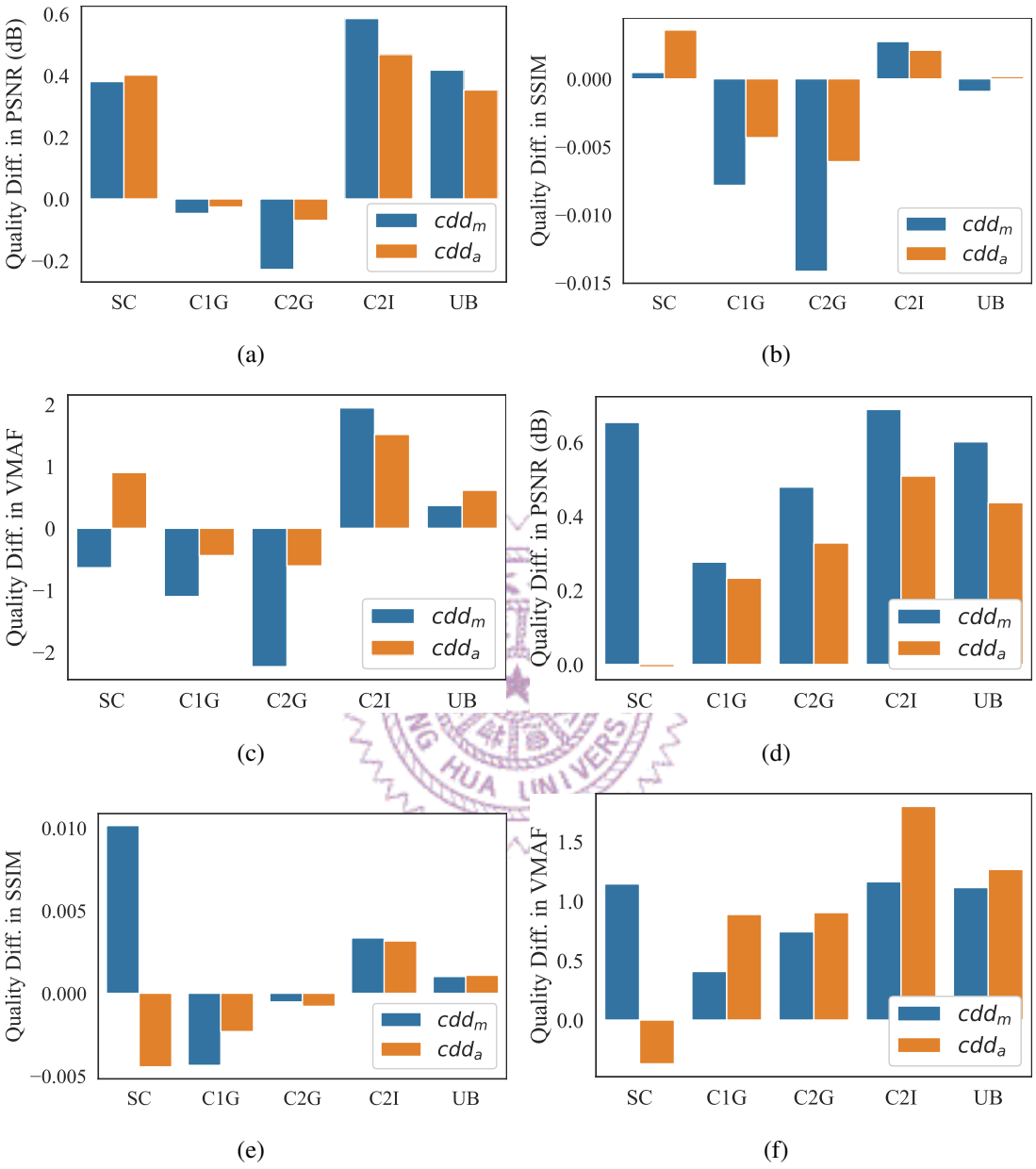


Figure 10.4: Effects of tuning cdd and different $slvr$. We choose cdd_a in each solver as the baseline. (a) Quality difference in PSNR for one bunny content, (b) Quality difference in SSIM for one bunny content, (c) Quality difference in VMAF for one bunny content, (d) Quality difference in PSNR for four bunny content, (e) Quality difference in SSIM for four bunny content, and (f) Quality difference in VMAF for four bunny content.

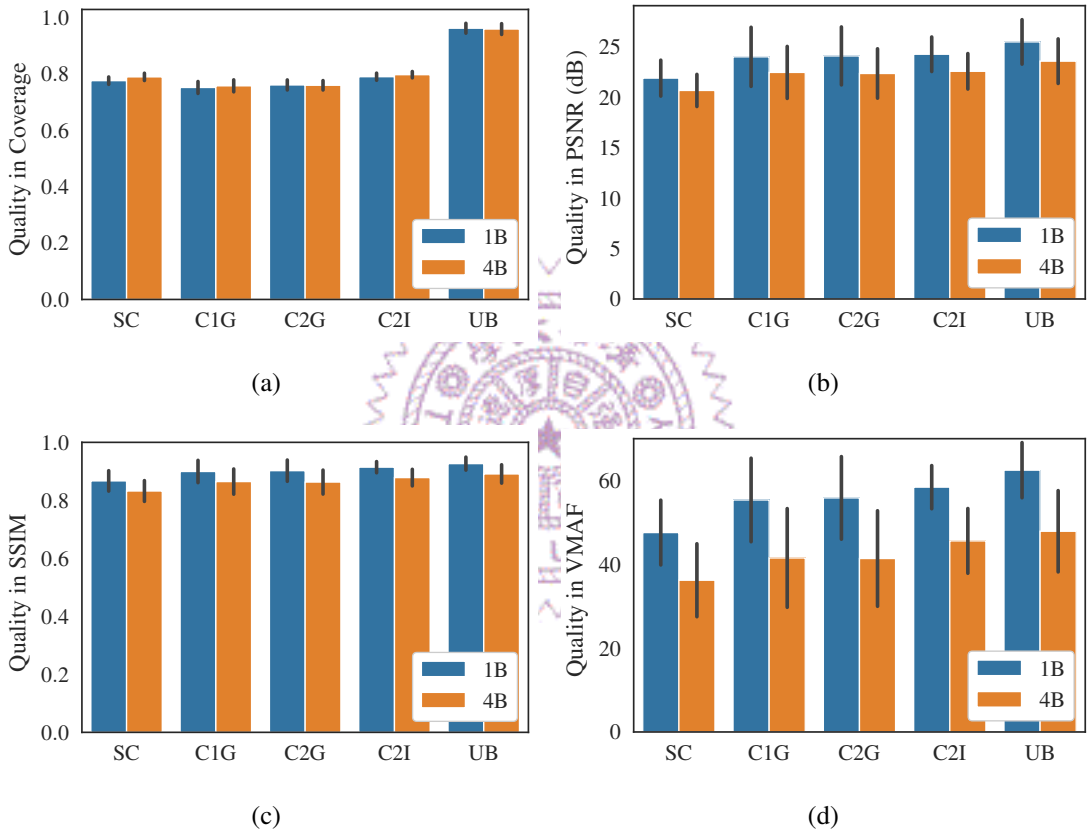


Figure 10.5: Overall performance achieved by different novel view optimization algorithms, in terms of: (a) coverage ratio, (b) PSNR, (c) SSIM, and (d) VMAF.

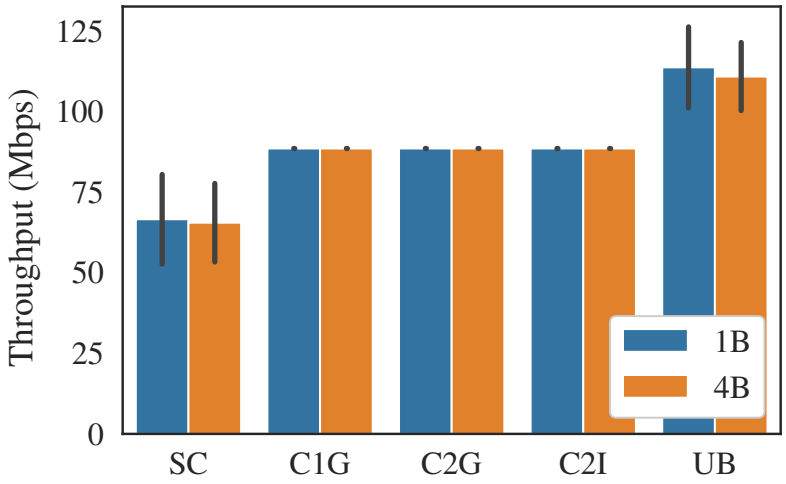


Figure 10.6: Network throughput caused by streaming selected source views.

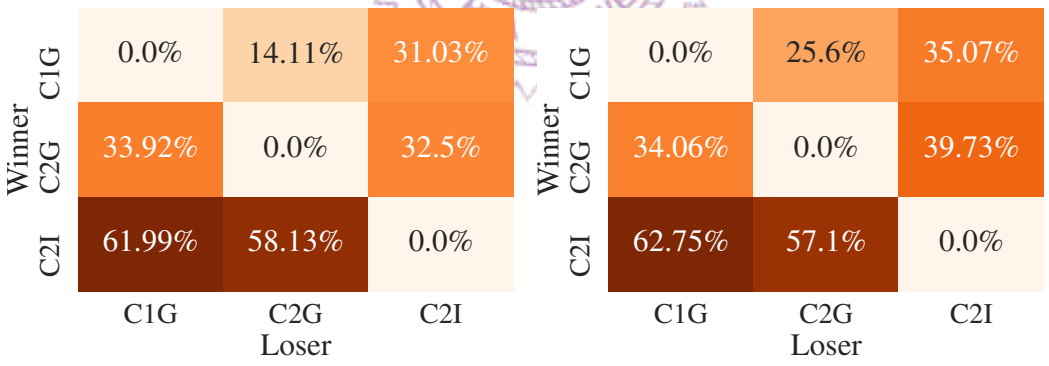


Figure 10.7: Winner-loser matrices based on the coverage ratio achieved by different algorithm variants: (a) one bunny and (b) four bunnies content.

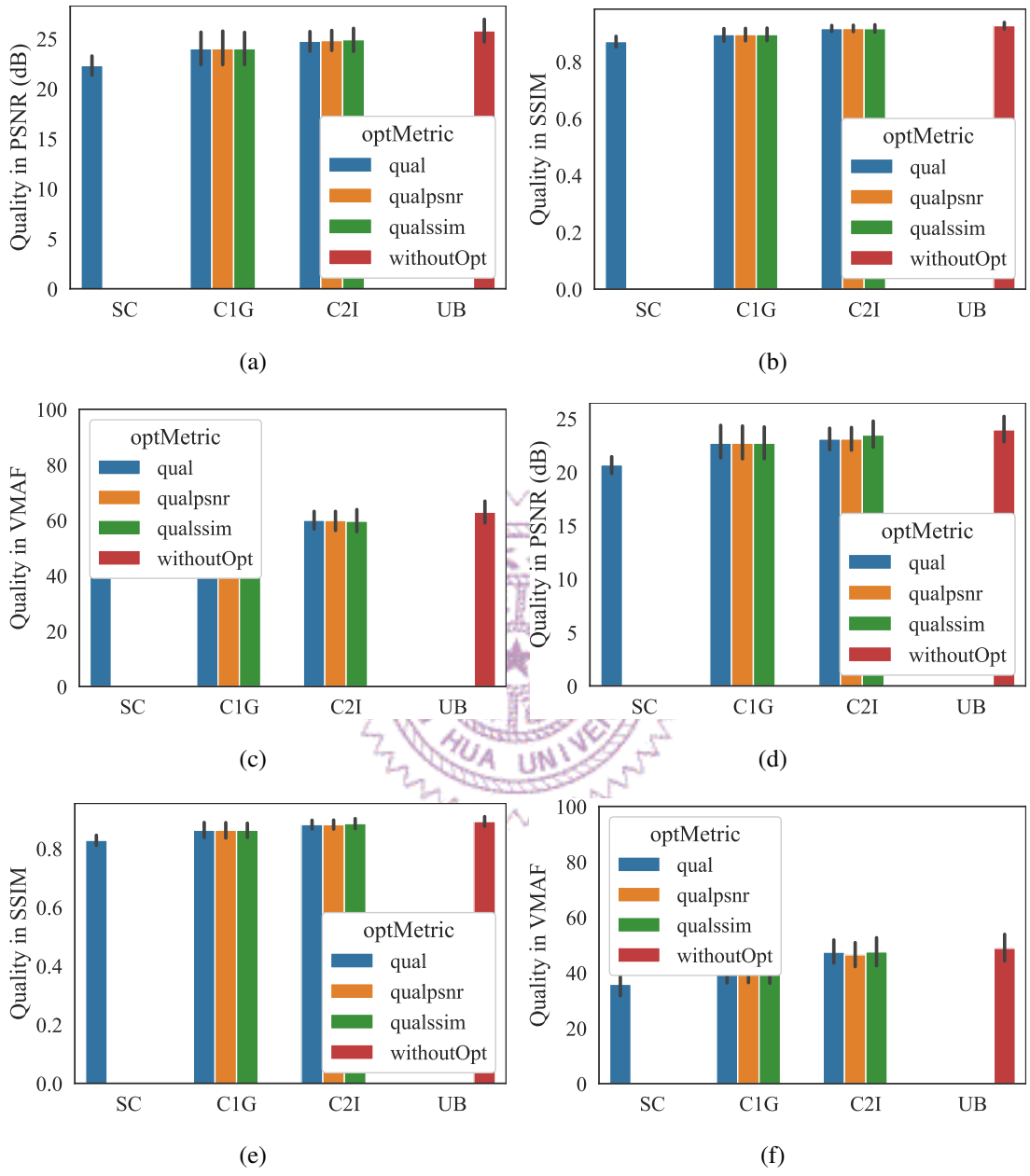


Figure 10.8: Results for optimizing with respect to different metrics using UB. (a) Quality of PSNR in one bunny scene, (b) Quality of SSIM in one bunny scene, (c) Quality of VMAF in one bunny scene, (d) Quality of PSNR in four bunny scene, (e) Quality of SSIM in four bunny scene, and (f) Quality of VMAF in four bunny scene.

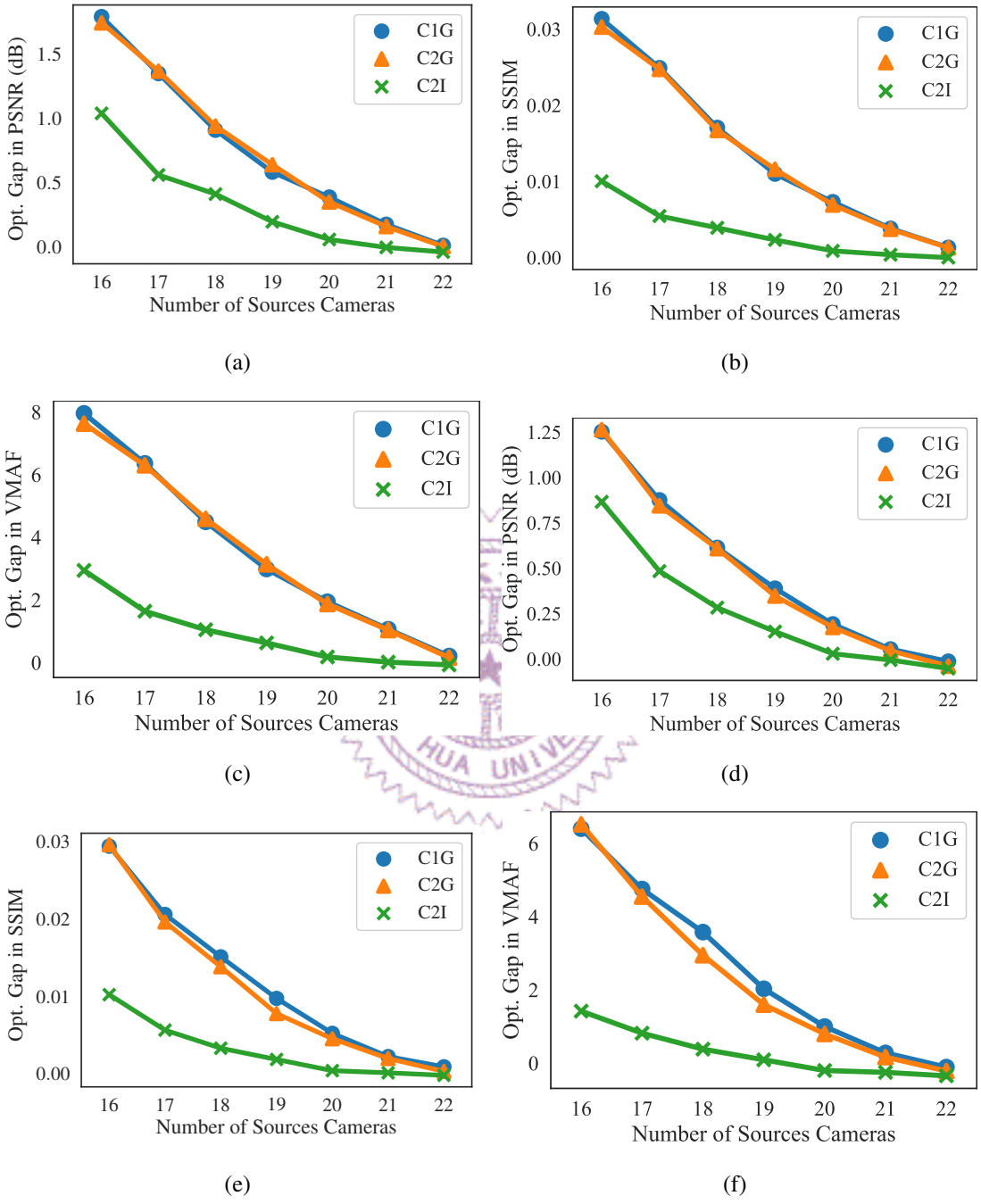


Figure 10.9: Effects of different number of source views. (a) PSNR results for one bunny scene, (b) SSIM results for one bunny scene, (c) VMAF results for one bunny scene, (d) PSNR results for four bunnies scene, (e) SSIM results for four bunnies scene, and (f) VMAF results for four bunnies scene.

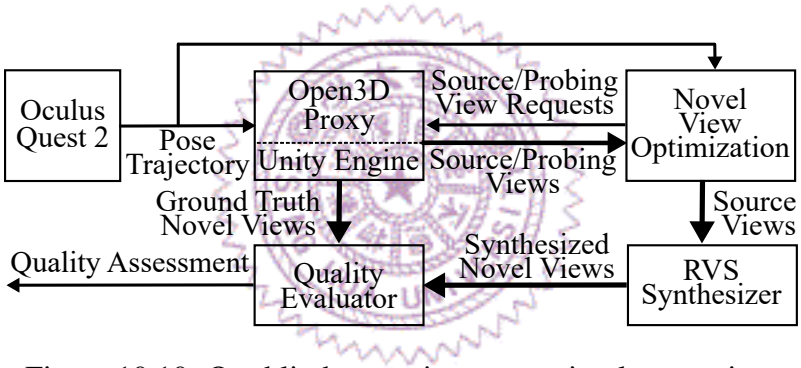
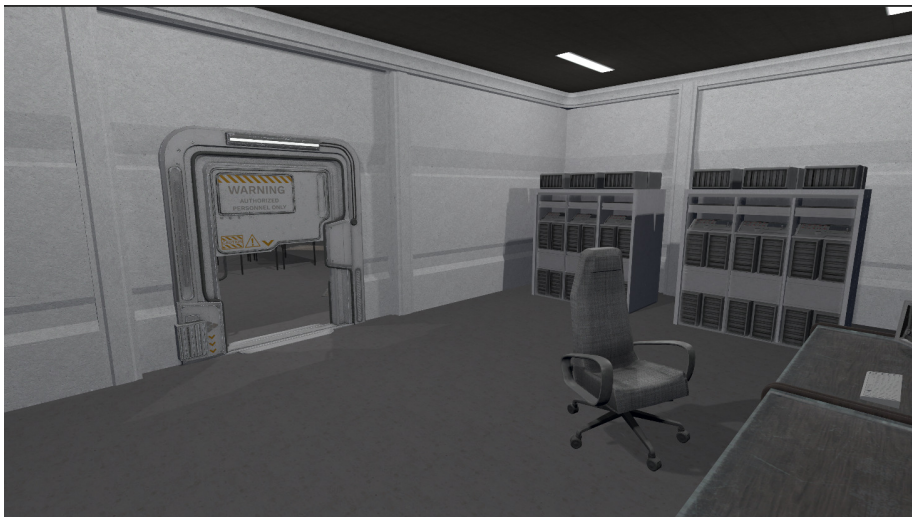


Figure 10.10: Our blind streaming system implementation.

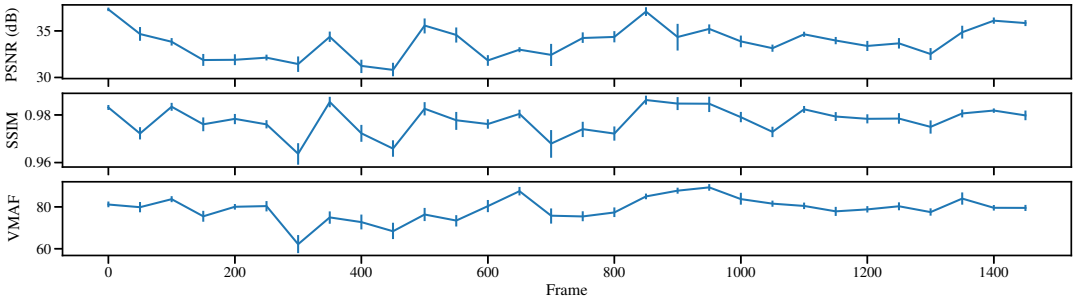
(a)

(b)



(c)

Figure 10.11: Considered 3D content: (a) House, (b) Bigroom, and (c) Smallroom.

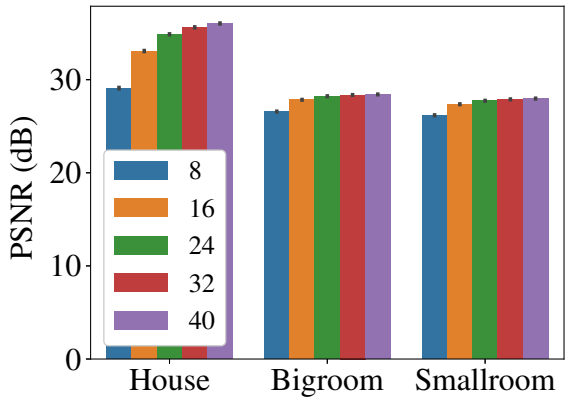


(a)

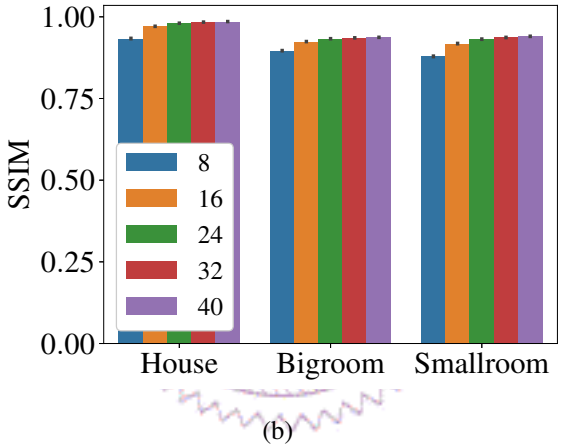
(b)

(c)

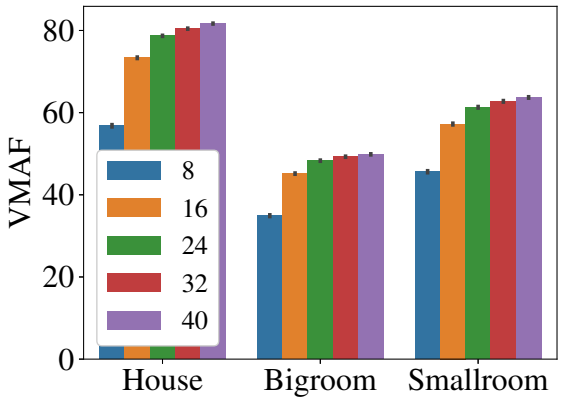
Figure 10.12: Sample synthesized novel views from House with default parameters: (a) average quality from a random client, (b) and (c) are the synthesized novel views with the highest and lowest PSNR values, respectively.



(a)



(b)



(c)

Figure 10.13: Quality improvement with increasing N: (a) PSNR, (b) SSIM, and (c) VMAF.

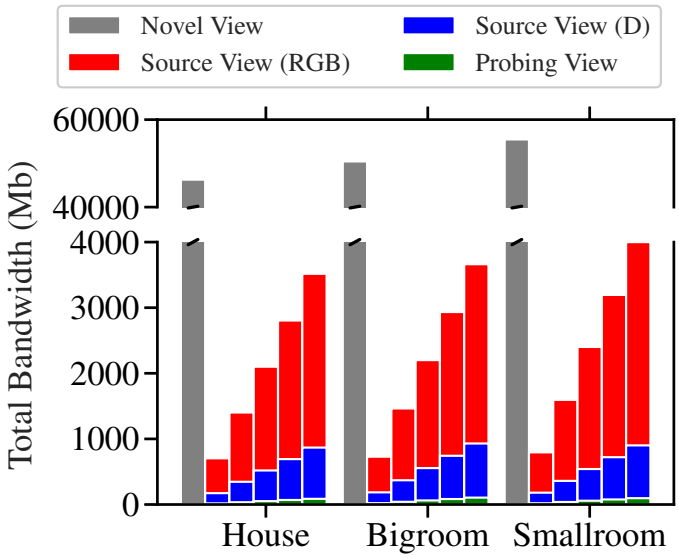
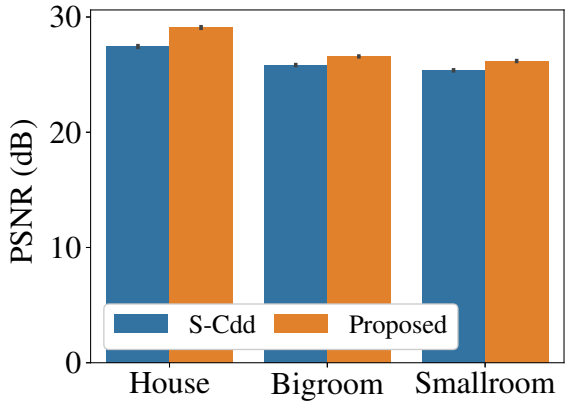
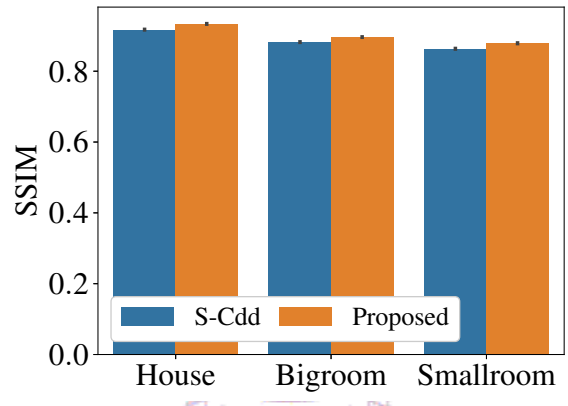


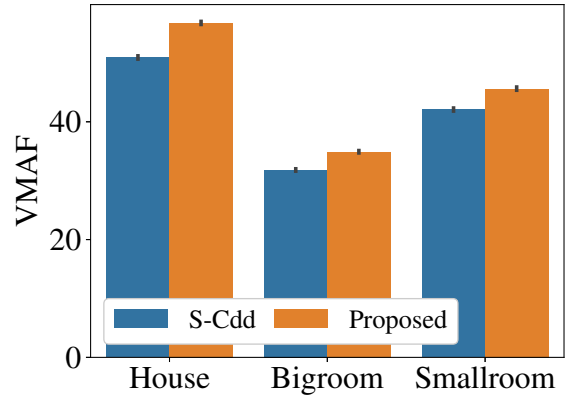
Figure 10.14: Bandwidth consumption and distribution. The source/probing views with $N \in \{8, 16, 24, 32, 40\}$ are reported.



(a)

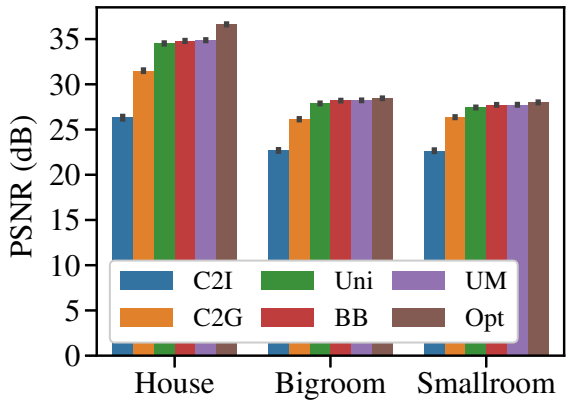


(b)

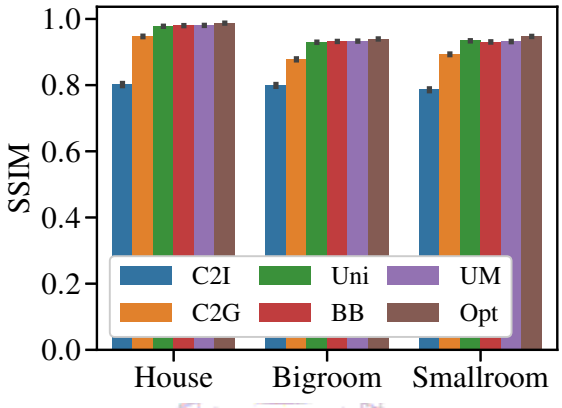


(c)

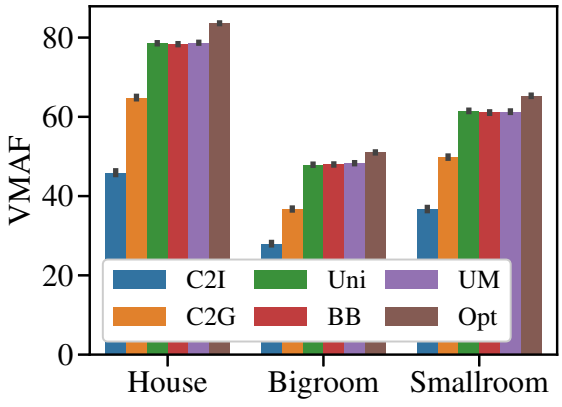
Figure 10.15: Performance evaluation of S-Cdd/P-Cdd with $N = 8$ and *UM* solver: (a) PSNR, (b) SSIM, and (c) VMAF.



(a)



(b)



(c)

Figure 10.16: Performance evaluation of solvers for P-CC with P-Cdd with $N = 24$: (a) PSNR, (b) SSIM, and (c) VMAF.

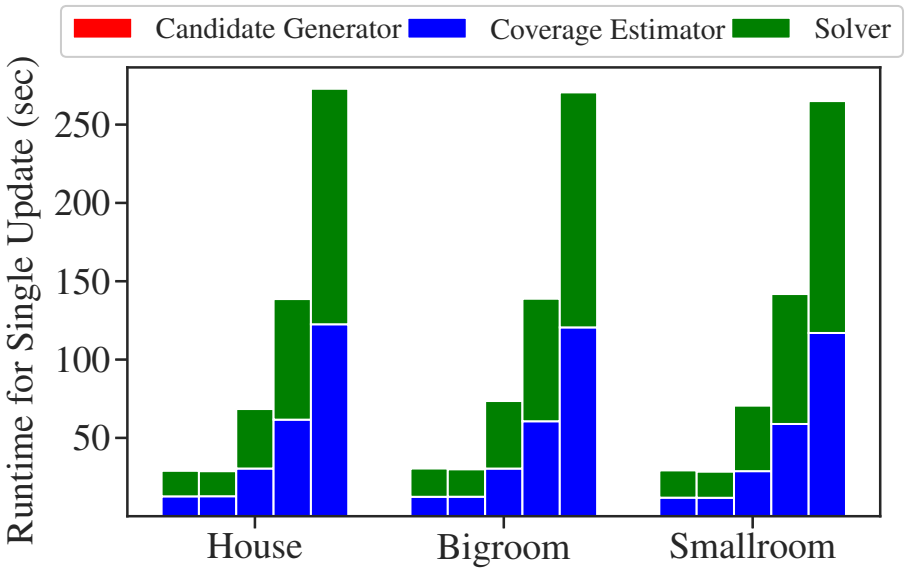


Figure 10.17: Runtime distribution for a sample update window. We vary the choice of N from 8 to 40 for each set of 3D content.

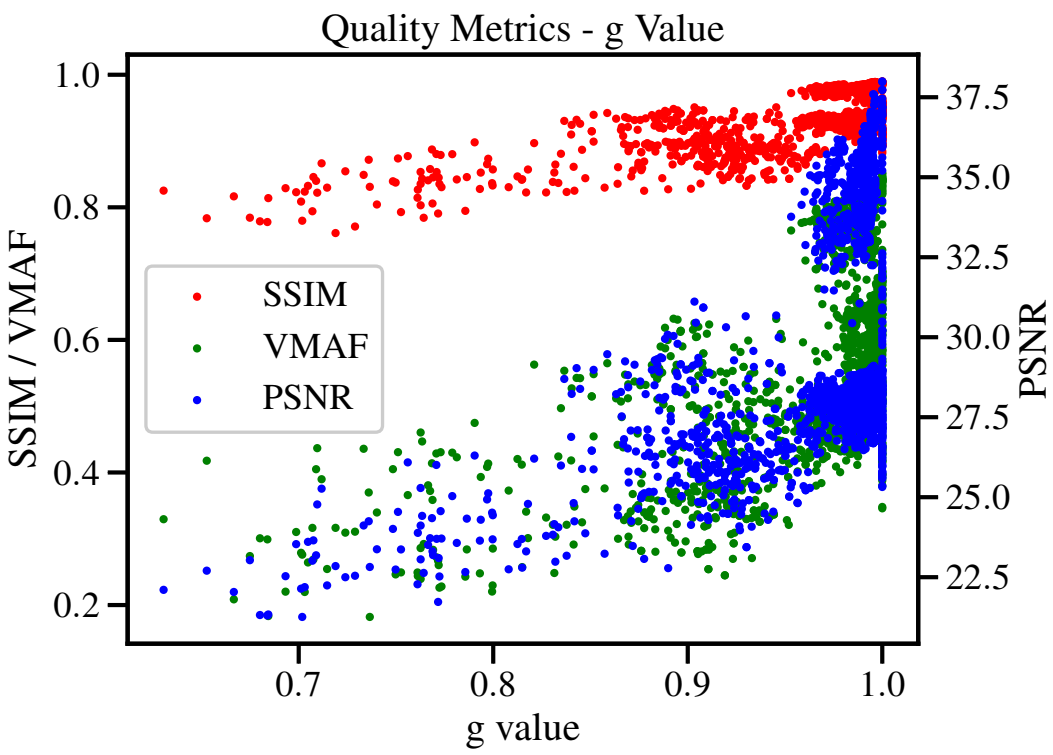


Figure 10.18: Correlation between quality metrics and aggregated quality.

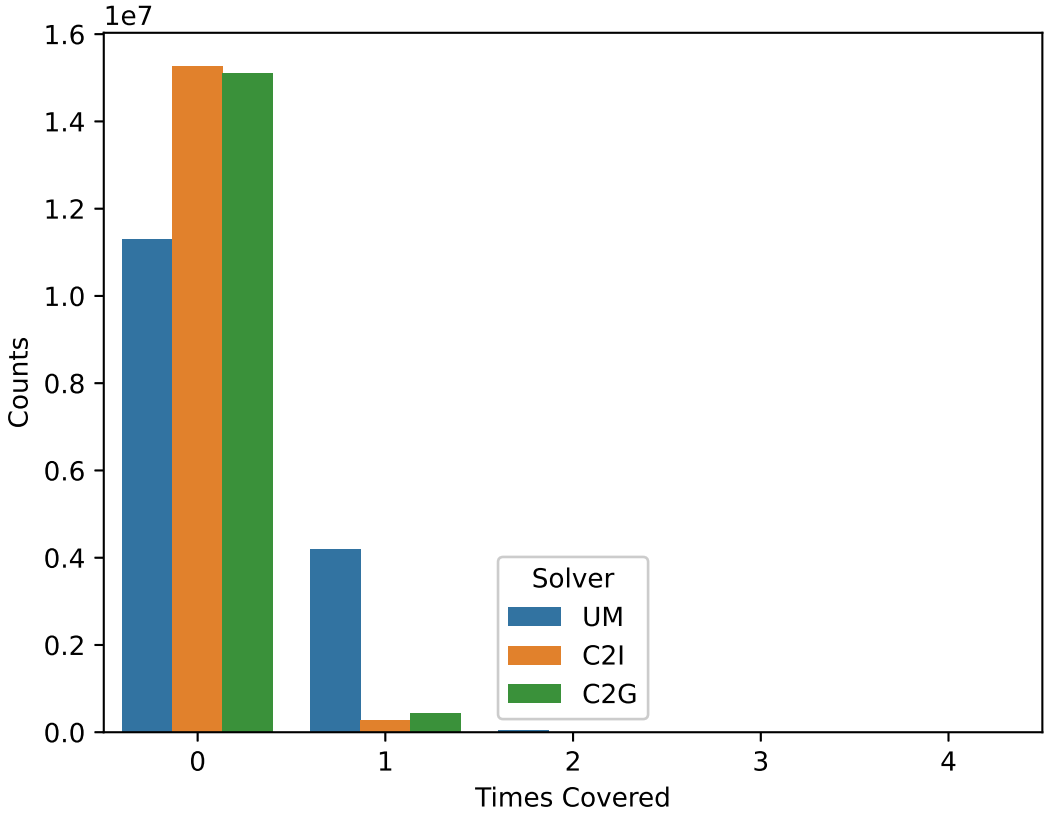


Figure 10.19: Counts of different levels of coverage.

Chapter 11

Conclusion

In this chapter, we are going to summarize what we have done in the past few years and explore possible directions to make our system even better.

11.1 Remarks

In this thesis, we designed, implemented, and evaluated a content-creator-friendly blind streaming system that serves many 6-DoF clients by synthesizing their novel views from carefully selected RGB-D source views. Our blind streaming system prevents IP leakage by only revealing partial 3D content with a few RGB-D source views and low-resolution depth probing views. We proposed a suite of novel view optimization algorithms to maximize the overall synthesized novel view quality across all 6-DoF clients using the coverage maps created from two depth images. We mathematically derived an optimal number of source view candidates to avoid overwhelming the content creator and cloud service provider. We also presented multiple iterative algorithms to select optimal source views from the candidates. We compare the proposed algorithms against the state-of-the-art ones, or say, the suite of algorithms for S-CC, which improve the visual quality by 2.27 dB in PSNR, 0.04 in SSIM, and 12 in VMAF on average, and leave small performance gaps with respect to an upper bound. Moreover, our experiments revealed that our algorithms are not sensitive to the choice of candidate generator and reduces content creator bandwidth consumption by 94%. We suggest using $N = 1.5|\mathcal{U}|$ with the *Uni* solver for real-time systems and *UM* solver for the best quality under abundant resources.

11.2 Attack using structure-from-motion (SfM)

We check if Structure-from-Motion (Sfm) algorithms can be used to attack our blind streaming system by reconstructing the 3D content from source views. We adopt a state-

of-the-art SfM algorithm, Colmap [33], and reconstruct a point cloud using 720 RGB source views at 960x540 resolution in *House*. Colmap ran for more than 10 hours on an Nvidia RTX 2080 Ti GPU, but still resulted in unacceptable 3D content. Fig. 11.1 gives sample ground truth and reconstructed 3D point cloud. We observed frequent artifacts, including: (i) incorrect object positions and (ii) large holes on the floor and other flat surfaces. Hence, we conclude that our blind streaming system can still protect IP leakage, even under attacks from a modern SfM algorithm.

11.3 Future work

This work can be extended in several directions, such as: (i) parallelism when solving the novel view optimization problem, (ii) employment of real-time view synthesis on HMDs, and (iii) formulation that takes SfM attacks and continuity into account.

The above directions improve the system performance in different ways. Some of them are for better computational efficiency, while some of them are for better user experience.

Computational parallelism. In Ch. 9, we have designed several algorithms that could possibly run in parallel. For instance, the *branch* operation in BB solver could be parallelized by computing all of them together. This example could also be applied to the *clear* and *set* operation of the UM solver. The computational dependency only exists between iterations, also called solution update. The branching-like operations are independent of each other inside an iteration. Therefore, correctly building graph representation of computational dependencies and identifying all independent operations are crucial for best usage of computational power.

Real-time view synthesis. In our system, we assume that HMDs are powerful enough to run in real-time. Actually, for our implementation, we offload rendering to a power PC. In fact, there are different synthesizers which even leverage GPUs more than ever before. We could incorporate such off-the-shelf synthesizers to improve our user experience.

Optimization for defense against SfM and continuity. In Ch. 10, we found that VMAF are relatively low compared to the other two metrics for P-CC systems. We concluded that the problem mainly arose from the reason that our optimization objective does not take temporal continuity into account. Addition is a commutative operation, and it cannot tell the order and distribution. The second consideration is defense against SfM attackers. Our formulation only represents coverage, but it does not consider the information the attackers are monitoring. A complete formulation will be that we also play the role of attackers and formulate the reconstruction quality into some numerical forms. The modeled reconstruction quality will be added to the formulation constraints such that we

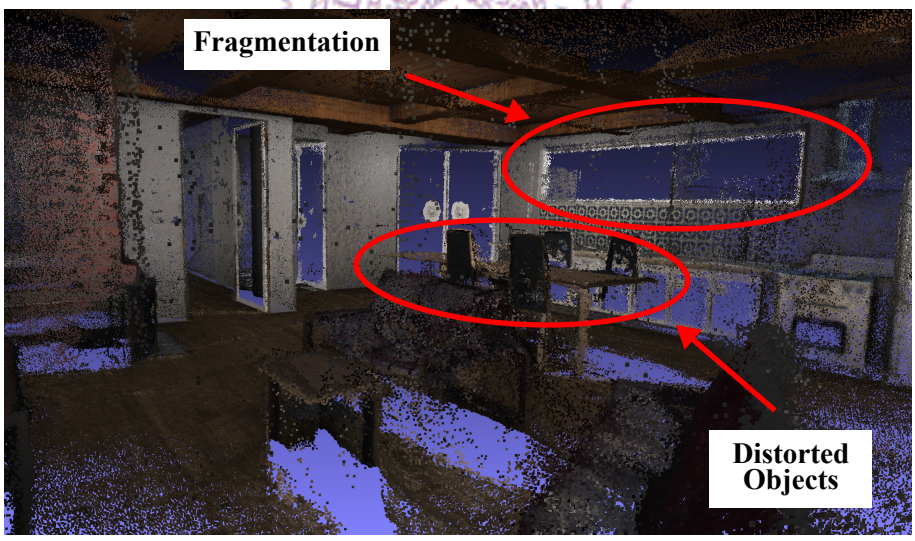
could also expect how much information is going to be harvested by the attackers.

Network simulation. In Ch. 10, we adapt our blind streaming system to diverse network status by adapting N. We assume that a proper bandwidth-to-N mapping exists for our systems. A complete evaluation could further include co-simulation with some well-known network simulator such as NS-3 [32].

Objective metrics for reconstructed mesh. In this thesis, we only use visual comparison and enumerate some artifacts. We also tried some point cloud alignment methods such as ICP [45]. However, due to the fact that only a few points are reconstructed, the computed transformation is meaningless and performs worse. We need to switch to evaluate reconstruction quality using assessing video quality metrics by rendering a huge amount of video data from the reconstructed 3D content, which could further strengthen the argument if correct coordinate transformation between the origin and reconstructed content can be identified.

Implementation for dynamic content. In this thesis, we evaluated how our systems perform in static content. We could extend the systems to dynamic content by implementing the following features: (i) source views need to be rendered frame by frame to synthesize continuous clients' novel views, and (ii) probing views may also be rendered frame by frame to better observe the dynamics of the 3D content.

(a)



(b)

Figure 11.1: 3D reconstruction of *House*: (a) ground truth mesh and (b) reconstructed point cloud with clear artifacts.

Bibliography

[1] alvr-org. ALVR - Air Light VR, 2023. https://github.com/alvr-org/ALVR.

[2] B. Attal, S. Ling, A. Gokaslan, C. Richardt, and J. Tompkin. MatryODShka: Real-time 6-DoF video view synthesis using Multi-Sphere images. In *Proc. of European Conference on Computer Vision (ECCV'20)*, pages 441–459, Glasgow, United Kingdom, August 2020.

[3] K. Bestuzheva, M. Besançon, W.-K. Chen, A. Chmiela, T. Donkiewicz, J. van Doornmalen, L. Eifler, O. Gaul, G. Gamrath, A. Gleixner, et al. The SCIP optimization suite 8.0. Technical report, Optimization Online, 2021.

[4] 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.

[5] J. Boyce, R. Doré, A. Dziembowski, J. Fleureau, J. Jung, B. Kroon, B. Salahieh, V. K. M. Vadakital, and L. Yu. MPEG immersive video coding standard. *Proc. of the IEEE*, 109(9):1521–1536, September 2021.

[6] P. B. Bullen. How to choose a sample size (for the statistically challenged), 2022. https://tools4dev.org/resources/how-to-choose-a-sample-size/.

[7] S.-C. Chen. Multimedia research toward the metaverse. *IEEE MultiMedia*, 29(1):125–127, 2022.

[8] I. Choi, O. Gallo, A. Troccoli, M. Kim, and J. Kautz. Extreme view synthesis. In *Proc. of IEEE/CVF International Conference on Computer Vision (ICCV'19)*, Seoul, Korea, October 2019.

[9] I. Cohen, Y. Huang, J. Chen, J. Benesty, J. Benesty, J. Chen, Y. Huang, and I. Cohen. Pearson correlation coefficient. *Noise Reduction in Speech Processing*, pages 1–4, 2009.

[10] Epic Games. Unreal engine, 2019. https://www.unrealengine.com.

[11] S. Gül, S. Bosse, D. Podborski, T. Schierl, and C. Hellge. Kalman filter-based head motion prediction for cloud-based mixed reality. In *Proc. of ACM International Conference on Multimedia (MM'20)*, pages 3632–3641, Seattle, United States, October 2020.

[12] J. Haas. A history of the Unity game engine. *Diss. Worcester Polytechnic Institute*, 483(2014):484, March 2014.

[13] J. Hladky, M. Stengel, N. Vining, B. Kerbl, H.-P. Seidel, and M. Steinberger. QuadStream: A quad-based scene streaming architecture for novel viewpoint reconstruction. *ACM Transactions on Graphics*, 41(6):1–13, November 2022.

[14] A. Horé and D. Ziou. Image quality metrics: PSNR vs. SSIM. In *Proc. of IEEE International Conference on Pattern Recognition (ICPR'20)*, pages 2366–2369, Istanbul, Turkey, August 2010.

[15] X. Hou and S. Dey. Motion prediction and pre-rendering at the edge to enable ultra-low latency mobile 6DoF experiences. *IEEE Open Journal of the Communications Society*, 1:1674–1690, 2020.

[16] M. L. Hänel and C.-B. Schönlieb. Efficient global optimization of non-differentiable, symmetric objectives for multi camera placement. *IEEE Sensors Journal*, 22(6):5278–5287, March 2022.

[17] IBM ILOG Cplex. V12. 1: User's manual for CPLEX. *International Business Machines Corporation*, 46(53):157, 2009.

[18] Kris Holt. Meta quest pro will soon support WiFi 6E, 2023. https://reurl.cc/3xxmvL.

[19] B. Kroon and G. Lafruit. Reference View Synthesizer (RVS) 2.0 manual, 2018.

[20] MarketWatch. Metaverse market global analysis 2023-2030, 2023. https://reurl.cc/944QrY.

[21] L. Markley, Y. Cheng, J. Crassidis, and Y. Oshman. Averaging quaternions. *Journal of Guidance, Control, and Dynamics*, 30(4):1193–1197, May 2007.

[22] MIV. MPEG IMMERSIVE VIDEO (MIV), 2022. https://mpeg-miv.org/.

[23] MPEG. The GitLab of mpeg test model for immersive video, 2019. https://gitlab.com/mpeg-i-visual/tmiv/-/tree/v10.0.1.

[24] V. Munishwar and N. Abu-Ghazaleh. Scalable target coverage in smart camera networks. In *Proc. of ACM/IEEE International Conference on Distributed Smart Cameras (ICDSC'10)*, pages 206–213, Atlanta, United States, August 2010.

[25] L. Myers and M. J. Sirois. Spearman correlation coefficients. *Encyclopedia of Statistical Sciences*, 12, 2004.

[26] Netflix. VMAF - Video Multi-Method Assessment Fusion, 2021.

[27] NVIDIA, P. Vingelmann, and F. H. Fitzek. CUDA, release: 10.2.89, 2020. https://developer.nvidia.com/cuda-toolkit.

[28] E. Park, J. Yang, E. Yumer, D. Ceylan, and A. Berg. Transformation-grounded image generation network for novel 3D view synthesis. In *Proc. of IEEE International Conference on Computer Vision and Pattern Recognition (CVPR'17)*, Honolulu, United States, July 2017.

[29] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[30] C. Peng and V. Isler. Adaptive view planning for aerial 3D reconstruction. In *Proc. of IEEE International Conference on Robotics and Automation (ICRA'19)*, pages 2981–2987, Montreal, Canada, May 2019.

[31] I. E. Richardson. *The H.264 Advanced Video Compression Standard*. Wiley Publishing, 2nd edition, 2010.

[32] G. F. Riley and T. R. Henderson. The NS-3 network simulator. In *Modeling and Tools for Network Simulation*, pages 15–34. Springer, 2010.

[33] J. L. Schönberger and J.-M. Frahm. Structure-from-motion revisited. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR'16)*, Las Vegas, United States, June 2016.

[34] S. Shah, D. Dey, C. Lovett, and A. Kapoor. Airsim: High-fidelity visual and physical simulation for autonomous vehicles. In *Field and Service Robotics*, pages 621–635. Springer, November 2018.

[35] Y.-C. Sun, S.-M. Tang, C.-T. Wang, and C.-H. Hsu. On objective and subjective quality of 6DoF synthesized live immersive videos. In *Proc. of ACM International Workshop on Quality of Experience in Visual Multimedia Applications (QoEVMA'22)*, pages 49–56, Lisboa, Portugal, October 2022.

[36] S. Suresh, A. Narayanan, and V. Menon. Maximizing camera coverage in multicamera surveillance networks. *IEEE Sensors Journal*, 20(17):10170–10178, September 2020.

[37] S.-M. Tang, C.-H. Hsu, Z. Tian, and X. Su. An aerodynamic, computer vision, and network simulator for networked drone applications. In *Proc. of ACM Annual International Conference on Mobile Computing and Networking (MobiCom '21)*, pages 831–833, New Orleans, United States, 2021.

[38] S.-M. Tang, Y.-C. Sun, J.-W. Fang, K.-Y. Lee, C.-T. Wang, and C.-H. Hsu. Optimal camera placement for 6 Degree-of-Freedom immersive video streaming without accessing 3D scenes. In *Proc. of ACM International Workshop on Interactive EXtended Reality (IXR'22)*, pages 31–39, Lisboa, Portugal, October 2022.

[39] The Freeport Player Authors. Freeport player demo (intel), 2022.

[40] S. Tomar. Converting video formats with ffmpeg. *Linux Journal*, 2006(146):10, 2006.

[41] Unreal Engine. Blueprints visual scripting in unreal engine, 2022. https://docs.unrealengine.com/5.0/en-US/blueprints-visual-scripting-in-unreal-engine/.

[42] V. V. Vazirani. *Approximation Algorithms*. United States, 2003.

[43] X. Wang, A. Chowdhery, and M. Chiang. Networked drone cameras for sports streaming. In *Proc. of IEEE International Conference on Distributed Computing Systems (ICDCS'17)*, pages 308–318, Atlanta, United States, June 2017.

[44] M. Woo, J. Neider, T. Davis, and D. Shreiner. *OpenGL programming guide: the official guide to learning OpenGL, version 1.2*. Addison-Wesley Longman Publishing Co., Inc., 1999.

[45] J. Zhang, Y. Yao, and B. Deng. Fast and robust iterative closest point. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(7):3450–3466, 2021.

[46] Q. Zhang, S. He, and J. Chen. Toward optimal orientation scheduling for full-view coverage in camera sensor networks. In *Proc. of IEEE International Conference*

on Global Communications Conference (GLOBECOM'16), pages 1–6, Washington, United States, December 2016.

[47] Q.-Y. Zhou, J. Park, and V. Koltun. Open3D: A modern library for 3D data processing, 2018. http://www.open3d.org/.