Abstract:

XXX

Problem statement:

Recently, Virtual Reality (VR) becomes increasingly more popular. It enables a wide array of novel applications in many domains, such as video streaming, computer games, occupational training, healthcare, manufacturing, etc. The global virtual reality market size was valued at USD 15.81 billion in 2020 and is expected to grow at a compound annual growth rate (CAGR) of 18.0% in the up coming years XXXcite1XXX. As the growth of metaverse, its market is expected to reach USD 872.35 billion in 2028 and register a revenue CAGR of 44.1% XXXcite2XXX. To emphasize the user immersive experience in metaverse, real-time rendering AR/VR devices are regarded as an important interaction layer to bridge the physical and virtual environment XXXcite3XXX. More and more companies devote their effort to the VR industry such as Meta XXXciteXXX or Google XXXciteXXX.

We consider supporting multiple users simultaneously viewing the live immersive videos of remote dynamic scenes on Head-Mounted Display(HMD), and also supply 6 Degree of Freedom(6DoF). In 6DoF applications, they allow users to have full control in their positions, which are (i) surge, heave, and sway, and their orientations, which are (ii) yaw, roll, and pitch. There are some ways to achieve the 6DoF immersive video, like point cloud or mesh, and we opt for the Depth Image-Based Rendering(DIBR) XXXciteXXX as a lower computing solution. To synthesize the user viewport in HMD, users are able to receive multiple source views, which are captured by RGBD cameras from the server-side, and utilize them to synthesize the viewport in client-side based on user's positions and orientations.

In real-time live streaming, the bandwidth of the Internet, network latency, and packet loss can be the main factors that could reduce users' immersive experiences. Traditional streaming does not scale well for multiple users. Besides, traditional immersive streaming has some critical problems that may cause high latency. As we showed in XXXFig1(a)XXX, traditional streaming comes in the pose-frame sequence and the users have to transmit their pose to the server-side and the server will synthesize the viewport according to the user pose. In this situation, the latency will count from the time pose is issued to the next frame arrival. However, synthesis-based streaming is capable to fix the high latency issue which we show the high-level concept in XXXFig1(b)XXX. Synthesis-based streaming is able to issue dense poses, and the latency will be composed of the computational time for a synthesized frame. Nevertheless, how to identify those crucial sources such that users could derive their desired viewports remains a challenging issue. In this way, we are trading users' workload for reduced latency. We will mainly focus on identifying critical visual sources.

To solve this challenging problem, we propose an algorithm to provide the source views' camera placements strategy that could support high-quality image synthesis at a limited bandwidth. Our algorithm takes 3D representations of a scene in which a user is traveling and quality index as its setup. The inputs of our algorithm are the predicted trajectories in a short period and the available bandwidth. The trajectory is composed of a series of 6DoF samples in a short period and assumed that we predicted it precisely. The available bandwidth is able to determine the number of cameras at a specific timestamp. According to the number of cameras, our system is able to provide a strategy for camera placement. This is a challenging problem because most of the image synthesizers do not have a closed-form representation for synthesized quality in terms of the amount of displacement from the source views. To completely implement our camera placement algorithm to the virtual or real system, there are still several research problems that need to be solved, which contain (i) creating a quality model, (ii) generating a data collection system, and (iii) adjusting the system for different synthesizer. More details of our tasks will be given and mentioned later.

Related work:

We have surveyed some papers on camera placement, quality model, synthesizers and so on. We will introduce them one by one in the following paragraphs.

Camera placement. Jing et al.~\cite{jing2016sampling} proposed a *sampling-based* approach. The solution space is constructed by dilating around the rough representation of the being covered object. Then, candidate camera position is randomly sampled within the solution space and viewing directions are calculated using potential field method, which is similar to Newton's law of universal gravitation. The candidates are selected greedily to reduce the expected number of cameras used. The sample-and-select process could be run over and over again until the coverage ratio is satisfied or a termination requirement is reached. Peng and Isler et al.~\cite{peng2019adaptive} proposed an *offset-and-fit* approach for adaptive aerial 3D reconstruction. Their goal is to minimize the length of aerial craft path to visit all necessary view ports. The surface vertices are first partitioned, elevated and fitted into several viewing planes. The final path is constructed by computing a back-and-forth path within individual planes. Finally, they connect all the viewing planes by computing the shortest Euclidean walk.

Quality models. Quality estimation is important because the overall performance depends highly on the utility function or the quality estimation is developed. The most common way of doing so is to design a function that takes camera poses and scene representation as inputs and outputs a scalar that estimates the final synthesized/rendered quality. Jing et al.~\cite{jing2016sampling} uses inverse of uncertainty. The estimated quality can be computed by using the angle between two cameras and their distance. Hepp et al.~\cite{Plan3D} design their quality model to be the amount of visual information to be product of functions of incidence ($vi_i(\tau, \nu)$) and resolution ($vi_r(\tau, \nu)$) respectively. The functions are designed to be exponentially falling with some specific thresholds. The total information discovered in a voxel from a view port is $\frac{1}{\xi} vi_i(\tau, \nu) vi_r(\tau, \nu)$. Roberts et al.~\cite{roberts2017submodular} use total surface light field covered of all surface points as their maximization target. Each surface point is attached with a virtual hemisphere centere. A camera's coverage associated with a surface point is defined to be the disk projected from the camera to the associated hemisphere. In the case that a voxel is blocked, the disk radius is set to zero. However, none of them integrate the rendered/synthesized image quality into their design.

MIV

Synthesizer

Synthesizers for immersive videos can be categorize into three types, Image Based Rendering (IBR), Depth Image Based Rendering (DIBR), and MutiPlane Images (MPI). IBR approaches utilize the images we captured by cameras to synthesize the target views, such like image morphing, interpolaton, and light modeling. DIBR approaches take depth information as input. These approaches map the pixels on to a 3D space using the provided depth map, and then project the pixels back to synthesize new views. MPI methods adapt machine learning techniques to train models that can convert 3D scenes into layers of images which different layers represent different depth.

Wizadwongsa et al. proposed NeX XXXciteXXX, a novel MPI synthesizer whinch utilizes neural network to adjust basis functions for color representation.

Problem solution:

Scenario. We plan to design a camera placement algorithm to provide a camera placement strategy and expand it for different synthesizer and implement it in a real immersive video streaming system. There are several design choices for the camera placement algorithm. The system could be either memorable or memoryless. The main difference is that the memory system would reuse the previous camera's placement and bandwidth thus maximizing the information we have gained over time. On the other hand, the memoryless system only looks into the present inputs and arranges the camera placement to achieve the best quality. However, the algorithm must combine with the quality model. The quality model should satisfy the following requirements, (i) the algorithm should place the cameras along with the samples in the next piece of the trajectory when the number of cameras is greater than the number of samples in the trajectory, (ii) the computation should be as fast as possible such that the placement is available in the next few sample periods. Symbols are listed in Table I before we state the optimization problem formally. The formal mathematical formulation for memorable systems can be written as

$$maximize_{\mathcal{C}_t} \sum_{i=t}^{t+n} Q(\mathcal{C}_t, v_i)\,,\ \ subject\ to\ |\mathcal{P}_t| = N_t, \mathcal{C}_t = \cup_{i=0}^{t} \mathcal{P}_i$$

For systems that have memoryless, we have

$$maximize_{\mathcal{C}_t} \sum_{i=t}^{t+n} Q(\mathcal{C}_t, v_i)\,,\ \ subject\ to\ |\mathcal{C}_t| = N_t$$

Research problems. We divide this proposal into four research problems, which are detailed below.

Camera placement algorithm. We simplify our problem by setting a fixed prediction horizon of length (n) in which the fixed number of target poses are given by some prediction algorithms. We assume that the source view poses are a subset of the target view poses, that is $\mathcal{S} \subseteq \mathcal{T}$. We transform the problem into a node selection problem based on the contribution of each node. A graph $G = (\mathcal{T}, \mathcal{E})$, where $\mathcal{E} = \{q(u,v) \mid \forall u, v \in \mathcal{T}\}$, and G is a fully connected, directed graph. The simple mathematical formulation for systems can be written as

$$maximize_{\boldsymbol{X}} \sum_{\boldsymbol{u}\in\mathbf{T}} \sum_{\boldsymbol{v}\in\mathbf{T}} x_u q(u,v)$$

$$subject\ to\ x \in \{0,1\} \forall x \in \boldsymbol{X}, \sum_{x\in\boldsymbol{X}} x \leq N$$

, where x_u denote the indicator for node u. The contribution in this formulation which is $q(u,v)$ is calculated by the quality model. To implement our system, there are some design requirements to make our node selection problem reasonable. First, we impose the constraints that $q(u,u) = 1 \forall u \in \mathcal{T}$ since the target pose is exactly the same as the source pose. Second, we need to have $\sum_{u\in\mathcal{T}} x_u q(u,v)$ exhibit positive relation to $m\left(\mathcal{I}_v, \widehat{\mathcal{I}_v}\right)$. In other words, the higher value of the former implies a higher value of the latter. Third, we expect the quality model is additive. That is, $m_{u_1,u_2}\left(\mathcal{I}_v, \widehat{\mathcal{I}_v}\right) \approx m_{u_1}\left(\mathcal{I}_v, \widehat{\mathcal{I}_v}\right) + m_{u_2}\left(\mathcal{I}_v, \widehat{\mathcal{I}_v}\right)$, where the subscripts denote the nodes that are selected to reconstruct v. Generally, the quality model is not commutative, that is, $q(u,v) \neq q(v,u)$.

Quality model. The main goal of the quality is to generate the relevance between source views, user pose traces, network bandwidth, and video quality. We opt to use generate a quality model based on view coverage. The Coverage quality model used the area in the target view v covered by source view u as the estimated quality. The view coverage can be calculated within several matrix multiplications by utilizing view unprojection and re-projection. In virtual scenes, we are able to calculate the coverage of viewport by Open3D XXXcite4XXX, as long as we have the 3D scenes information and camera parameters such as field of view(FOV), position, orientation, resolution, and projection model. However, the coverage is not additive since it intrinsically involves set operations like union and intersection. To fit the coverage quality model to the algorithm, we have two workarounds. First, we naively add all the coverage together but the overall optimization target will be over-estimated (q_{nav}). Second, we slightly modify the quality model by dividing the contribution to the overlapping area evenly by the number of views that cover it (q_{even}). Illustration~\ref{fig:coverageQualityModel} show the difference between the two approaches. The first approach gives total coverage of 0.8 while the latter gives $0.45 + 0.25 = 0.8$ if both source views are selected. However, in the case that only the left view is selected, the coverage for both models will be 0.5 and 0.45 respectively. Fortunately, the true solution of total quality will be bounded from both side, that is,

$$\sum_{u\in\mathcal{T}}\sum_{v\in\mathcal{T}} x_u q_{even}(u,v) \le \sum_{u\in\mathcal{T}}\sum_{v\in\mathcal{T}} x_u q(u,v) \le \sum_{u\in\mathcal{T}}\sum_{v\in\mathcal{T}} x_u q_{nav}(u,v).$$

The quality model will design in an empirical way and aim at specific scenes. We will take the source views position and orientation, viewport trajectories, and coverage as the inputs. Then the quality might output the value of quality metrics which might use as an important basis of contribution. To conduct these experiments, it is important for us to collect our own dataset.

Data collection system. The purpose of collecting our own dataset is to investigate the impacts of diverse settings on the synthesized target views. We use scenes in Unreal Engine marketplace, and capture captured the RGBD video clips by Airsim XXXciteXXX. Airsim is an open-source project that provides Application Programming Interface (API) for programmers in Unreal Engine. We develop a data collection system. It is able to collect the datasets that could cover a wide variety of usage scenarios with (i) different camera placement strategies, (ii) random target view trajectories mimicking real HMD users, (iii) scenes with diverse characteristics (e.g., lighting conditions, color tone, and dynamics). For the experiments of the quality model, we will capture the source view and the ground truth of the target view in virtual scenes as our dataset by this system.

MIV reference software. XXX

Synthesizer. In our experiments, we are going to first focus on the View Weighting

Synthesizer (VWS) XXXciteXXX. VWS is reproduce a scene by computing the weight of source views based on how similar the direction is to the target views. It is easier to understand and does not require any machine learning skills. We will conduct quality measurements on evaluating how well does our system work. We also look forward to fit our implementation into different synthesizers. It doesn't matter what kind of synthesizers the users are going to use, our system can generate the proper inputs for the synthesizer.

Tasks and schedule.

Next, we will list the concrete research tasks and schedule them.
Task 1. Generating the data capture system
Task 2. Creating the quality model
Task 3. Implement the quality model in the algorithm and test it in an empirical way
Task 4. Implement the quality model to the real-world simulation.
Task 5. Expanding the algorithm to different synthesizers.

XXXcite1XXX
https://www.grandviewresearch.com/industry-analysis/virtual-reality-vr-market
XXXcite2XXX
https://www.einnews.com/pr_news/558354492/metaverse-market-size-expected-to-reach-usd-872-35-billion-at-cagr-of-44-1-in-2028
XXXcite3XXX
https://dl.acm.org/doi/abs/10.1145/3474085.3479238?casa_token=mEMHG6j_t2cAAAAA%3ApCKoRdbxxGUrJsfSaS039xf5gmZNYfa1RaMPP1_D8HEV3cbAbYv0fPogS1885PTykOFbkWHbKwAKu-SU
XXXcite4XXX
Open3D: A Modern Library for 3D Data Processing

Table I

Symbol	Explanation
n	The number of samples contained in the trajectory
v_t	User pose sampled at time t, $(x, y, z, roll, pitch, yaw)$
c	Camera pose, $(x, y, z, roll, pitch, yaw)$
$\mathcal{U}$	Set of all free 6DoF camera pose
$\mathcal{C}_t$	Set of cameras considered at time t
$\mathcal{P}_t$	Set of cameras newly placed at time t
N_t	The number of cameras newly placed at time t
$q(c, v)$	Quality model, where c is single camera placement and v is a surface/point
$Q(C, v)$	Aggregated quality model, where C is the camera placement and v is a surface/point
θ	Angle between the look-at vector of the camera and line connecting the target point and the camera
$dist(p_1, p_2)$	distance between p_1 and p_2, p could be 6 DOF poses or 3D position
β	Fall-off factor used in the quality model
$pose\ (p)$	6-tuple, $(x, y, z, roll, pitch, yaw)$
$\mathcal{T}$	Set of target poses to be reconstructed
$\mathcal{S}$	Set of source poses used to reconstruct the target poses
$q(p_1, p_2)$	Quality contribution for using p_1 to reconstruct p_2
N	The number of source poses allowed
x_u	Binary indicator incidcating that the node u is selected (1) or not (0)
X	Collection of binary indicators of each node
$\mathcal{I}_u$	Ground truth image at node u
$\widehat{\mathcal{I}_u}$	Reconstructed image at node u
$m(\mathcal{I}, \hat{\mathcal{I}})$	Quality metrics for comparing the two images