Spatial Visibility and Temporal Dynamics: Rethinking Field of View Prediction in Adaptive Point Cloud Video Streaming

Chen Li, Tongyu Zong, Yueyu Hu, Yong Liu, Yao Wang
New York University
USA
{chen.lee,tz1178,yh3986,yongliu,yw523}@nyu.edu

Abstract

Field-of-View (FoV) adaptive streaming significantly reduces bandwidth requirement of immersive point cloud video (PCV) by only transmitting visible points inside a viewer's FoV. The traditional approaches often focus on trajectory-based 6 degree-of-freedom (6DoF) FoV predictions. The predicted FoV is then used to calculate point visibility. Such approaches do not explicitly consider video content's impact on viewer attention, and the conversion from FoV to point visibility is often error-prone and time-consuming. We reformulate the PCV FoV prediction problem from the cell visibility perspective, allowing for precise decision-making regarding the transmission of 3D data at the cell level based on the predicted visibility distribution. We develop a novel spatial visibility and object-aware graph model (**CellSight**) that leverages the historical 3D visibility data and incorporates spatial perception, occlusion between points, and neighboring cell correlation to predict the cell visibility in the future. We focus on multi-second ahead prediction to enable the use of long pre-fetching buffers in on-demand streaming, critical for enhancing the robustness to network bandwidth fluctuations. CellSight significantly improves the long-term cell visibility prediction, reducing the prediction Mean Squared Error (MSE) loss by up to 50% compared to the state-of-the-art models when predicting 2 to 5 seconds ahead, while maintaining real-time performance (more than 30fps) for point cloud videos with over 1 million points.

CCS Concepts

• **Information systems** → **Multimedia streaming**; • **Computing methodologies** → *Machine learning approaches.*

Keywords

Point Cloud Video, Field of View (FoV) Prediction, Immersive Video Streaming, 6 Degree of Freedom (DoF), Virtual Reality (VR), GNN

ACM Reference Format:

Chen Li, Tongyu Zong, Yueyu Hu, Yong Liu, Yao Wang. 2025. Spatial Visibility and Temporal Dynamics: Rethinking Field of View Prediction in Adaptive Point Cloud Video Streaming. In *ACM Multimedia Systems Conference 2025 (MMSys '25), March 31-April 4, 2025, Stellenbosch, South Africa.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3712676.3714435

MMSys '25, March 31-April 4, 2025, Stellenbosch, South Africa

ACM ISBN 979-8-4007-1467-2/2025/03
https://doi.org/10.1145/3712676.3714435

1 Introduction

Augmented Reality (AR) and Virtual Reality (VR) applications are gaining popularity rapidly. Streaming high-quality immersive videos, such as 360-degree videos and point cloud videos, to viewers is one of the most critical components for the wide adoption of AR/VR. It can also be integrated with interactive methods, such as the user force-aware approach in [33], further enhancing the user experience. However, immersive videos require significantly higher bandwidth than the traditional 2D planar videos. For example, a point cloud video consisting of 1 million points per frame requires streaming bandwidth of more than 120 Mbps even with lossy compression [22], while future production of high quality point cloud can scale up to 10M points per frame. A promising solution is FoV adaptive streaming that only streams video content within a viewer's current viewport. For 360-degree video, if the viewport is 120 degree (horizontal) by 90 degree (vertical), only 1/6 video data fall into a viewport, resulting into 6-fold bandwidth reduction. For point cloud video, due to points occlusion in 3D space, not all points falling into a viewport are visible. One can save even more bandwidth by not just omitting the points outside the viewport, but also removing the hidden points within the viewport. The complete point cloud frame illustrated in Fig. 1 contains more than 1 million points. The number of points visible for the viewport given in Fig. 1(a) are less than 150k, after removing the points outside the viewport and the occluded points. In cell-based PCV streaming, a point cloud frame is divided into cells, as shown in Fig. 1(c), and each cell is independently coded and transmitted.[1] To deliver the actual view in Fig. 1(b), only cells covering those visible points in Fig. 1(c) need to be transmitted, leading to 7-fold of streaming bandwidth reduction.

In immersive video streaming, a viewer can freely change her viewpoint (X, Y, Z), as well as her view angle *(yaw, pitch, roll)*, resulting in a total of 6 Degree-of-Freedom (6DoF). Cell-based FoV adaptive streaming dynamically prefetches cells that are predicted to be visible into local streaming buffer based on the viewer's past viewport trajectory. The key challenge is to accurately predict *cell visibility* in real time so that more bandwidth can be allocated to prefetch cells with higher visibility. The visibility of a cell can be simply measured by its overlap ratio with the viewport. Considering occlusion, cell visibility should be refined by discounting occluded points. Finally, cells at different viewing distances contribute differently to the viewer's viewing experience [9, 36]. More precise cell

[1] To clarify the terminology, we use "cell" to denote the geometry-based PCV unit, as described in [9].

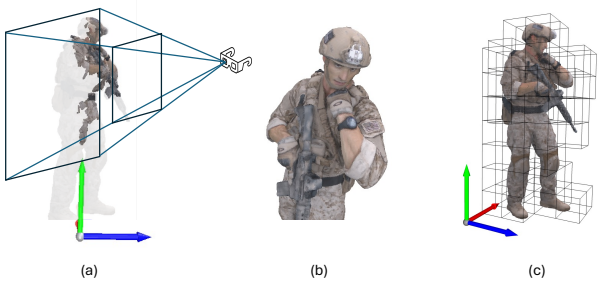


Figure 1: Illustration of PCV Point Visibility. As in Fig. 1(a), the content within the pyramid, defined by the near plane and far plane, is potentially visible in the viewport, while any content outside the pyramid will not be visible. Furthermore, some points inside the viewport are still not visible if occluded by other points. When a viewer watches the point cloud content from the viewpoint indicated in Fig. 1(a), only the highlighted part in Fig. 1(a) would be visible. The actual view for the viewer is shown in Fig. 1(b). If the point cloud is divided into 3D cells like in Fig. 1(c), only the cells covering the visible points need to be transmitted.

visibility metric should also take into account the cell-viewpoint distance. For on-demand PCV streaming, a relative long streaming buffer, e.g., 2 to 5 seconds, is preferred to provide sufficient margin for smooth video streaming and video processing. Consequently, predicting cell visibility in medium and long future horizons has become an important and challenging research problem.

Naturally, most of the existing FoV-adaptive PCV streaming studies [9, 11, 16, 20] predict point/cell visibility in two steps: 1) predict the viewer's future viewport based on her past viewport trajectory; 2) calculate visible points and cell visibility using the predicted viewport. This approach has several drawbacks: 1) trajectory-based viewport prediction does not explicitly consider the impact of video content on the viewer's attention; 2) small errors in any 6DoF viewport coordinates prediction may lead to large errors in visible points prediction; and 3) Hidden Point Removal (HPR) at high point density is time consuming. These limitations motivate us to rethink FoV prediction for PCV streaming: *Can we directly predict cell visibility based on the viewer's viewport trajectory, cell visibility history, and spatial features of objects to be viewed in PCV?* There are several advantages of this direct approach: 1) by directly predicting cell visibility based on the cell visibility history (we assume that the viewer's past viewport is fed back to the sender so that the sender can produce the visibility history data), we avoid the potential error amplification in the process of mapping 6DoF viewport to cell visibility; 2) using the spatial features of PCV objects to be viewed (for which we also have the ground-truth in on-demand streaming) as an input, we explicitly take into account the impact of PCV objects and their movements on the viewer's attention; 3) PCV object movements and viewer viewpoint movements are both continuous in time and space. As a result, point occlusion and visibility vary continuously in time and space. Motivated by these observations, we propose a TransGraph-GRU based spatial-temporal model for predicting 3D cell visibility. For the spatial aspect, a graph-based model is well-suited because if one cell lies within the FoV, its neighbors are likely to be visible as well. For the temporal aspect, an RNN-based model effectively captures dynamic dynamics [35], such as when a cell begins moving out of the FoV, making it likely to remain less visible in the near future. We opt for GRU over LSTM due to its lighter computational overhead—a critical factor when managing many cells. Prior work [10] has demonstrated the efficiency of integrating GRU with graph structures. Additionally, Transformer-based techniques [25] can effectively learn hidden spatial dependencies among neighboring cells, further enhancing visibility prediction. A well-trained spatial-temporal machine learning model can thus leverage these continuous dynamics to achieve accurate point/cell visibility prediction. Building on these insights, we make the following contributions in this paper:

- We design **CellSight**, a novel PCV FoV prediction framework that directly predicts the cell visibility in a future frame to a viewer based on the viewer's past viewport trajectory and the point cloud spatial features to be viewed.
- We construct and predict a set of cell visibility features that quantify the importance of streaming a cell by considering its overlap ratio with the viewport, the fraction of visible points after occlusion, and its angular span in the viewport determined by the viewing distance.
- We develop a spatial-temporal graph model which can capture the spatial and temporal correlations of cell visibility in consecutive PCV frames for accurate prediction.
- Through comprehensive experiments, we demonstrate that CellSight can generate real-time cell visibility predictions, and improve the long-term prediction accuracy of the state-of-the-art methods by up to 50% on the real point cloud video and viewport trajectory datasets.

2 Related Work

Immersive video streaming, whether for 360-degree (3DoF) videos or point cloud (6DoF) videos, demands strategies to reduce bandwidth usage while preserving user experience. In this section, we review prior work for 3DoF and 6DoF streaming scenarios and discuss the limitations that motivate our approach.

3DoF 360-degree Video Streaming. To mitigate the high bandwidth consumption inherent in 360-degree video streaming, many studies focus on predicting the user's Field-of-View (FoV) so that only the most relevant parts of the video are streamed at high quality. For instance, [2, 18] leverages personalized and cross-user behaviors to forecast future FoV, while [8] enhances prediction accuracy by incorporating content-based features in addition to historical FoV data. Furthermore, [19] extends these ideas by using variations of Long Short-Term Memory (LSTM) models to predict future FoV from past viewport trajectories, and even applies 2D spatial and temporal ConvLSTM models to forecast tile visibility. Although effective for 360-degree video, these methods rely on the assumption that all pixels within the 360-degree view are visible—a condition that does not hold in the more complex PCV context.

6DoF Point Cloud Video Streaming. Point cloud videos introduce additional challenges due to occlusions and vastly larger data volumes. In these scenarios, closer objects can obscure those further away, making the prediction of visibility more complicated.

Research in this area has addressed quality assessment, compression,reconstruction, segmentation and system design etc. For example, subjective studies in [23] and objective evaluations in [1, 29] have characterized the quality of point cloud videos, while [3] explores quality assessment using 3D visual saliency. To handle high data volumes, progressive coding strategies [24] and real-time 3D compression frameworks [4] have been proposed. Additional work includes human avatar reconstruction from few-shot views [32], and medical point cloud image segmentation [6]. Besides, system-level approaches for PCV streaming have been explored [5]. For example, [21] developed a mobile transcoding framework, and [12] proposed a real-time, human-centered method that segments the human body from the static background to reduce bandwidth overhead. When it comes to FoV prediction for point cloud videos, the challenges intensify due to 6DoF. [26] employing a 3DoF Bundle Adjustment to estimate 6DoF estimation. In [11], a combined LSTM and Multi-Layer Perceptron (MLP) model is used to predict user motion and head orientation for the next frame—yielding a very short prediction horizon (approximately 11 ms). The Vivo system [9] extends this horizon to 200 ms by incorporating factors like distance and occlusion, yet its trajectory-based prediction remains limited for long-term prediction. Similarly, CaV3 [20] improves prediction accuracy by integrating content features, but still suffers from the inherent limitations of trajectory-based approaches over long prediction horizons.

Our Contribution: CellSight. Unlike the existing methods that predominantly rely on trajectory-based FoV prediction, our approach, **CellSight**, leverages spatial and temporal models to capture detailed 3D cell visibility patterns. This design allows us to extend the accurate FoV prediction horizon to 5 seconds, thereby providing the streaming system with a substantially larger optimization window. In doing so, CellSight improves the robustness of adaptive streaming systems against network condition fluctuations.

3 Background and Problem Formulation

3.1 Point Cloud Video

A point cloud video (PCV) consists of a sequence of frames, each of which is a cloud of points on the surface of objects in a captured scene. Each point is described by it 3D coordinate and color. It can be rendered on 2D displays or VR goggles based on the user's view point, allowing users to explore the 3D scene from any angle and depth with 6 degrees of freedom (X, Y, Z, yaw, pitch, roll). We assume a PCV is represented by a geometry-based method and can be partitioned into 3D cells. Octree is one example that has been used in previous PCV streaming system [9]. For streaming, each PCV frame is typically partitioned into small 3D cells, which can be encoded independently at multiple quality levels [30], as shown in Fig. 1(c). For a given viewer viewport, only a small subset of cells are visible. If we know exactly which cells are visible, we just need to stream those visible cells at the highest quality allowed by the bandwidth. To deal with the unavoidable cell visibility prediction errors, we can also stream some cells with low visibility at low quality similar to the 360-degree video streaming strategy [28]. The final PCV quality perceived by a viewer and the PCV streaming overhead is largely determined by the accuracy of cell visibility prediction.

Symbol	Description
T_τ	6-DoF coordinates trajectory at time τ
$\mathcal{T}_h$	history trajectory
C	3D cell set
O^τ	graph occupancy features at time τ
F^τ	graph viewport overlap ratio feature at time τ
V^τ	graph occlusion-aware visibility features at time τ
E^τ	graph other features at time τ
A^τ	graph angular span feature at time τ
B^τ	graph visible angular span feature at time τ
G_τ	all graph raw features at time τ
S_τ	hidden states output of GRU at time τ
$\tilde{H}_\tau$	hidden states output of GNN at time τ
o_i^τ	occupancy feature of cell i at time τ
f_i^τ	viewport overlap ratio feature of cell i at time τ
v_i^τ	occlusion-aware visibility feature of cell i at time τ
α_i^τ	angular span feature of cell i at time τ
β_i^τ	visible angular span feature of cell i at time τ
e_i^τ	other features of cell i at time τ

Table 1: Notation table

3.2 Problem Formulation

We assume that the client has a frame buffer of length f, so that when the client is displaying frame h, the server needs to send frame $h+f$. Our goal is to predict the cell visibility for a future frame $h+f$ from the viewer's past history viewport trajectory, the past point cloud frames and the future point cloud frames up to frame $h+f$. We denote the viewer's history viewport trajectory by

$$\mathcal{T}_h = \{T^1, ..., T^h\}$$

where $T^\tau = (x^\tau, y^\tau, z^\tau, \psi^\tau, \theta^\tau, \phi^\tau)$ is the 6-DoF coordinates at time τ, and the corresponding PCV frame sequence by $\mathcal{P}_h = \{P^1, ..., P^h\}$. Each frame P^τ is partitioned into a set C of cells, each cell contains a set of points. The visibility of a cell i at time τ is defined as the number of visible points in that cell, denoted by as v_i^τ. The cell visibility vector for frame τ is $V^\tau = \{v_i^\tau, i \in C\}$. Our goal is to predict the cell visibility for a future frame Y^{h+f} at time $\tau = h+f$ from $\mathcal{T}_h, \mathcal{P}_h, P^{h+f}$ using a learned function $\mathcal{F}$:

$$Y^{h+f} = \mathcal{F}(\mathcal{T}_h, \mathcal{P}_h, P^{h+f}),$$

where Y^{h+f} denotes cell visibility features to be defined in Sec. 4.2. In principle, all frames in $\mathcal{P}_{h+f}$ can be used for prediction. For simplicity, we use $\mathcal{P}_h$ and P^{h+f}.

4 Methodology

We propose a graph-based prediction model that exploits Spatial Visibility and Temporal Dynamics of PCV. Fig. 2 presents an overview of CellSight. We divide the space covered by the entire point cloud video into multiple equal-sized cells. For the example in Fig 3, the space is partitioned into $5 * 6 * 8$ cells and all cells form a grid-like graph. Neighboring cells in the same frame have strong visibility correlations, which can be exploited by a graph model. Each node in the graph model corresponds to a cell. Each node has its neighboring cells as neighbors in the graph. For example, a cell can have

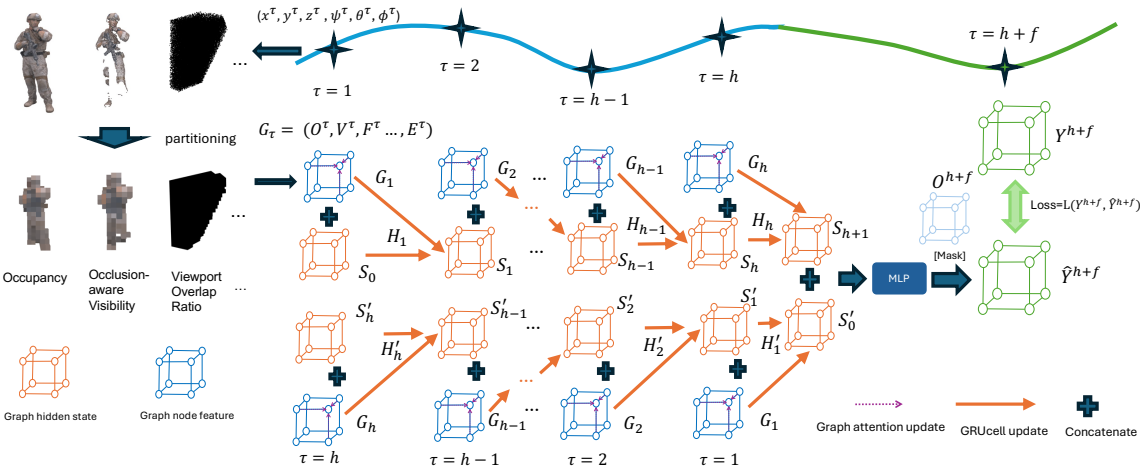


Figure 2: Overview of our cell visibility prediction system.

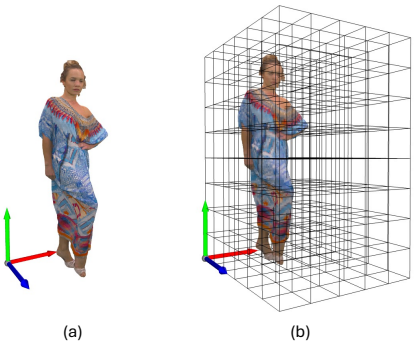


Figure 3: (a) is an example frame of a point cloud video. (b) shows how the entire 3D space is divided into 3D cells. We build a graph treating each cell as a node in the graph.

6 neighbors sharing the same side, or up to 26 neighbors sharing the same corner/edge/side. To simplify the illustration, we use a 8-node graph to represent the whole graph in Fig. 2. For each node, we have several features for visibility prediction. The curve on the top is the viewer's 6DoF viewport trajectory, illustrating one of the 6DoF coordinates $(x, y, z, \psi, \theta, \phi)$ at each frame time τ for a point cloud sequence set $\mathcal{P}^{h+f}$. For each frame, based on the 6DoF coordinates and P^τ, we can calculate, for each cell i, occupancy feature, visibility features, as well as location and distance features, which will be discussed in the following. We use $G_\tau = [O^\tau, F^\tau, V^\tau, ..., E^\tau]$ to represent the node features for each frame. We use a temporal Bidirectional GRU model and a spatial transformer-based graph model to capture patterns in viewer attention and cell visibility. GRU model captures each node's temporal pattern over time, which is encoded into the hidden state. In the graph model, each node aggregates its neighbor's information and hidden state from GRU, as the dash line shows. The bidirectional GRU can go through the history forward and backward, and have an enhanced hidden state to capture the history pattern. The details for Graph and GRU will be introduced in Section. 4.4. After we get the S_{h+1} and $S_0^{'}$ from bi-directional GRU, a shared MLP is used to predict the cell visibility for each cell at the target time stamp $\tau = h + f$. The final prediction output $\hat{Y}^{h+f}$ can be different cell visibility features designed to quantify the importance of a cell for the rendered view in different scenarios. The predicted cell visibility will be used by streaming algorithms to optimize bandwidth allocation among cells and improve streaming QoE. This proposed solution can be applied for video on demand or live streaming that can tolerate some playback lags, e.g. 1-5 second. Longer prediction accuracy of the solution will give more flexibility for the buffer-based streaming algorithms. We will introduce those features considering the cell's overlap ratio with the viewport, occlusions among points, and the viewing distance in Sec. 4.2. Before the final output, the O^{h+f} can be optionally applied as a mask to refine the predicted visibility $\hat{Y}^{h+f}$, since in the streaming system, the server has the point cloud at frame $h+f$. In the following, we will introduce how we construct features, including Cell Occupancy Feature, Cell Visibility Features, and Locality and Distance Features.

4.1 Cell Occupancy Feature

After partitioning the point cloud video into cells, we can get the number of points in each cell. Since viewer's attention can be driven by the objects in the point cloud video, the point density in a cell will affect the viewer's view interest for it and can be used as an important feature for cell visibility prediction. For example, work [14] studied the impact of video content on user FoV trajectory and observed that "users' gaze often follows the activity of the object inside the video", "for volumetric scenes with small movements, the gaze may move back and forth with irregular movement, but it generally still focuses on the target object". Clearly, if a cell is empty, it is unlikely to be viewed. We introduce the cell occupancy feature for frame τ as:

$$O^\tau = \{o_i^\tau, i \in C\}, \tag{1}$$

where o_i^τ is the number of points in cell i of frame τ.

4.2 Cell Visibility Features

We now introduce different cell visibility features that quantify the contribution of a cell to the viewing quality by considering different factors.

4.2.1 Viewport Overlap Ratio. Viewport Overlap Ratio is the overlap ratio between the current viewport and each cell as the cell-based viewport feature. To simplify the notation, we will use viewport feature in the rest of the paper. To obtain the overlap ratio between the viewport and cell i, we randomly generate R_i virtual points in cell i, and using the intrinsic and extrinsic matrices associated with the viewing camera (used for rendering according to the viewer's FoV) to calculate that. More specifically, given a set $\mathbf{P}$ of points in the world coordinates, we transform them to the viewing camera coordinates using:

$$\mathbf{P}_{\text{cam}} = \mathbf{E} \cdot \begin{bmatrix} \mathbf{P} \\ 1 \end{bmatrix},$$

where $\mathbf{E}$ is the extrinsic matrix determined by which represents the camera's orientation (yaw, pitch, roll), and a translation vector, which specifies the camera's coordinate system relative to the world coordinate system. We then project the transformed points onto the image plane using:

$$\mathbf{P}_{\text{img}} = \mathbf{I} \cdot \mathbf{P}_{\text{cam}},$$

where $\mathbf{I}$ is the intrinsic matrix, which depends on the intrinsic camera parameters including focal-length, etc. After we map all points on the camera's image, we can filter the points in actual FoV based on image dimensions and depth:

$$\text{Set of Points in Viewport} = \left\{ \mathbf{P}_{\text{img}} : \begin{vmatrix} 0 \le \mathbf{P}_{\text{img}}[0,:] < \text{width} \\ 0 \le \mathbf{P}_{\text{img}}[1,:] < \text{height} \\ d_{\text{near}} < \mathbf{P}_{\text{img}}[2,:] < d_{\text{far}} \end{vmatrix} \right\}$$

where d_{near} and d_{far} are the near and far plane as illustrated in Fig. 1. Denoting the number of points in the Set of Points in Viewport by K_i^τ at time τ in cell i, the viewport overlap ratio feature is defined as:

$$F^\tau = \left\{ f_i^\tau = \frac{K_i^\tau}{R_i}, i \in C \right\} \tag{2}$$

where R_i is the number of virtual points we randomly generated in cell i, and it's time independent.

4.2.2 Occlusion-aware Visibility . As we mentioned earlier, unlike 360-degree video, point cloud video has a unique property that requires consideration of occlusion in 3D space for any given 6DoF viewport. The number of visible points within a cell is crucial for describing occlusion. Given the 6DoF viewport coordinates and the point cloud object, we can use the HPR algorithm [15] to determine the number of visible points in a cell. However, applying HPR to high-density point clouds is time-consuming. In [9], the authors propose a cell-based occlusion estimation method. However, cell-based methods can introduce significant quantization errors. To achieve a better balance between accuracy and computational overhead, we calculate HPR on voxels instead of individual points, where each voxel covers an area of around $(1cm)^3$ cube in the 3D space. We assign the visibility of the voxel to all the points within. HPR on voxels has small accuracy loss from HPR with the original PCV, but remains more precise than the cell-based occlusion estimation. This enables real-time cell visibility calculations, achieving an average of approximately 45 fps on 8i data. After removing the hidden points, we apply intrinsic and extrinsic matrix mapping to the remaining points to exclude those points outside of the FoV. The final cell visibility feature for frame τ is then:

$$V^\tau = \left\{ v_i^\tau = \frac{Q_i^\tau}{N_i^\tau}, i \in C \right\}, \tag{3}$$

where N_i^τ is the number of points in cell i at time τ after downsampling, and Q_i^τ is the number of visible downsampled points. A further illustration for these three features is shown in Fig. 4.

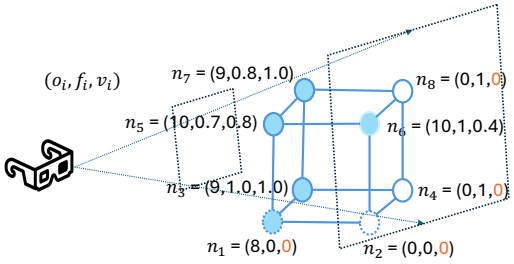


Figure 4: Given a viewer's 6DoF and the point cloud frame in 3D cells, we can get the occupancy feature, viewport overlap ratio feature and occlusion-aware visibility feature, as o_i, f_i, v_i. We have 8 nodes in total, and 5 of the nodes with color are occupied by points and have different numbers of points. For example n_6 has $o_i = 10$ points in total, the entire cell falls in the view port and hence $f_i = 1$, some points are occluded by other points and $v_i = 0.4$. For nodes without points, the occlusion-aware visibility feature is set as 0.

4.2.3 Angular Span and Visible Angular Span. Viewing distance is another important consideration for PCV streaming. An early subjective study on image viewing quality in [31] showed that the human perceived quality for an object depends on the *angular resolution*, i.e. the number of points per degree, which depends on the viewing distance and physical size of the object, as well as the image resolution. A recent subjective study on PCV in [9] also suggested that viewers' QoE changes significantly when they view a rendered point cloud object at different distances. Motivated by those studies, we model a cell's visual impact on the viewer as a function of the viewing distance. As illustrated in Fig. 5, all visible points in a cell are projected into a finite angular range within the viewer's viewport. The angular span depends on the distance of the cell to the viewpoint. A cell far away from the viewpoint contributes to only a small angular range of the viewport, and the angular resolution may saturate even at low point density [9, 36]. As a result, it has low impacts on the viewer's QoE and interest. Since we want to predict the utility of streaming each cell, we propose a new cell visibility feature to quantify the viewing distance effect. Similar to the pan-in-degree in [36], if a cell is completely inside the viewport and fully visible, as illustrated in Fig. 5, its contribution to the viewport can be quantified by the projected angular range θ,

which can be estimated as:

$$\theta = 2 \arctan\left(\frac{l}{2d}\right) \tag{4}$$

For a cell partially overlapping with the viewport, its contribution should be discounted by its overlap ratio with the viewport defined in (2):

$$A^\tau = \{\alpha_i^\tau = f_i^\tau * \theta_i^\tau, i \in C\}, \tag{5}$$

which is called *cell angular span* for cell i at time τ in the rest of the paper. Finally, if a cell is partially visible (due to occlusion), its contribution should be discounted by its visibility weight defined in (3):

$$B^\tau = \{\beta_i^\tau = v_i^\tau * \theta_i^\tau, i \in C\}, \tag{6}$$

which can be called the *cell visible angular span* for cell i at time τ.

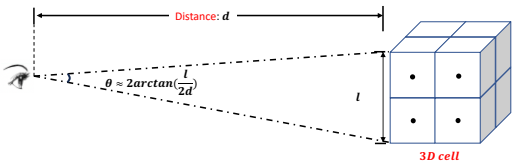


Figure 5: Inside a viewport, all points of a cell are viewed within an angular span determined by the viewing distance.

4.3 Locality and Distance Features

Other features $E^\tau = \{e_i^\tau, i \in C^\tau\}$ for cell i include the cell center coordinates and the distance from the cell center to the viewer's viewpoint. These features help the model account for changes in the viewer's position over time, similar to the way x, y, z coordinates are used in trajectory-based methods. It is important to note that the cell/node index remains fixed throughout the duration of the point cloud video.

4.4 TransGraph and GRU Model

We utilize a Transformer-based graph network [25] to build our graph model. The Transformer-based approach employs attention mechanisms [17] of transformers to capture dynamic relationships between neighbors. Given the hidden state $S_\tau = \{s_i, i \in C\}$ at time τ, we first concatenate S_τ with the raw node feature G_τ to form new feature for all cells:

$$H_\tau = \{h_i^\tau = s_i^\tau \oplus g_i^\tau, i \in C\} = S_\tau \oplus G_\tau \tag{7}$$

We then update the feature at each node using a weighted average of transformed features at its neighboring nodes, described by the following equations:

$$q_{c,i}^\tau = W_{c,q} h_i^\tau + b_{c,q} \tag{8}$$

$$k_{c,j}^\tau = W_{c,k} h_j^\tau + b_{c,k} \tag{9}$$

$$\alpha_{c,ij}^\tau = \frac{\langle q_{c,i}^\tau, k_{c,j}^\tau \rangle}{\sum_{u \in \mathcal{N}(i)} \langle q_{c,i}^\tau, k_{c,u}^\tau \rangle} \tag{10}$$

$$v_{c,j}^\tau = W_{c,v} h_j^\tau + b_{c,v} \tag{11}$$

$$\hat{h}_i^\tau = \sum_{j \in \mathcal{N}(i)} \alpha_{c,ij}^\tau v_{c,j}^\tau \tag{12}$$

where $W_{c,q}, W_{c,k}, W_{c,v}$ and $b_{c,q}, b_{c,k}, b_{c,v}$ are trainable weights and bias matrix and $\langle q, k \rangle = \exp\left(\frac{q^T k}{\sqrt{d}}\right)$ is the exponential scale dot-product function and d is the hidden size of each head [25]. We denote the above operations over all nodes collectively as

$$\hat{H}_\tau = TransGraph(H_\tau) \tag{13}$$

This is a single-layer graph model update; multiple layers can be used sequentially based on the output of the previous layer. For the GRU model, similar to the approach in [10], we employ a bi-directional GRU to capture temporal patterns. At each time step, the original hidden state s_i^τ at node i is updated to $\hat{h}_i^\tau$ using the graph operation described above, which propagates information from its neighbors in the graph. This updated hidden state will then serve as the hidden state for the next time step in the GRU model in both directions:

$$S_{t+1} = GRU_f(\hat{H}_t, G_t) \tag{14}$$

$$S'_{t-1} = GRU_r(\hat{H}'_t, G_t) \tag{15}$$

At the end of the GRU, one MLP module, which is shared by all cells, will take the combined hidden states S_{h+1} and $S_0^{'}$ of each cell to predict the visibility or viewport overlap ratio of this cell.

Since in a streaming system, when the system decides to send frame P^{h+f}, we can use occupancy feature at time $h+f$, O^{h+f}, as the mask on the output of MLP module to get the final prediction $\hat{Y}^{h+f}$. This masking operation, for example, can correct the situation when the predicted visibility is high but the actual occupancy is zero. The mask will be used for when Y^{h+f} is visibility feature V^τ and visible angular span feature B^τ.

$$\hat{Y}^{h+f} = M^{h+f} \circ \hat{Z}^{h+f} \tag{16}$$

where M^{h+f} is an indicator variable defined as:

$$m_i = \begin{cases} 1 & \text{if } o_i \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

and Z^{h+f} is the output of MLP module. A binary mask is applied here because if a cell has no points, there is no need to transmit that cell. Conversely, we do not rely on the number of points within a cell, as having more points does not necessarily equate to higher visibility, which ultimately depends on the viewer's interest.

Combining all the elements, CellSight is capable of generating the cell features in real time using the historical viewport trajectory and point cloud video data. Conceptually, the graph model captures the interaction of the cell features among neighboring nodes, while the GRU model captures the temporal dynamics of the cell features. By concatenating the cell feature with the cell's hidden state as the input feature to the graph model, and using the output of the graph model as the updated hidden state of the GRU model, our overall model is able to exploit the impact of the hidden states of neighboring cells on the temporal dynamics of a chosen cell. The model's output may be cell visibility based on the streaming system or cell-based viewport feature, and the prediction accuracy will be assessed in the following section.

5 Evaluation

5.1 Dataset

We use two public point cloud video datasets with user FoV traces to train and evaluate **CellSight**. The first dataset is 8i [7], which contains four videos, `longdress`, `loot`, `redandblack`, and `soldier`, each with over one million points per frame, along with a 6DoF viewing navigation trajectory dataset [27]. Trajectories were collected from 26 users watching these videos (looped every 5 seconds) at 30 fps. Although the PCV loops every 5 seconds due to its original design, user trajectories remain continuous and can vary from loop to loop. Consequently, the only truly repeating feature is the cell occupancy feature (e.g., the number of points in each cell). All the other features reflect continuous user FoV trajectories and thus do not repeat. As a result, the looping content has minimal impact on our evaluation, since most features remain distinct and capture realistic viewing behaviors. In total, these videos amount to approximately 40k frames. We use the trajectories from the first three videos (`longdress`, `loot`, `redandblack`) for training, and the trajectory from the fourth video (`soldier`) is split into testing (first half) and validation (second half) sets. The second dataset is the Full Scene Volumetric Video Dataset (FSVVD) [13], which represents complex indoor scenes. Its accompanying trajectory dataset [14] comprises data from over 10 users watching six different point cloud videos. We selected four videos—`Chatting`, `Pulling trolley`, `News interviewing`, and `Sweeping`—and focused on the 12 users who watched all four. We excluded the other two videos because they contain dual full-scene rooms and differ significantly in scale. Similarly, we use the trajectories from the first three videos for training, while the last video (`Sweeping`) is equally divided between testing and validation. Since FSVVD is recorded at 60 fps, we downsample both the videos and the user trajectories to 30 fps to align with the 8i data. Overall, FSVVD provides over 50k frames. Notably, user FoVs in FSVVD are more dynamic than those in 8i, making prediction more challenging at longer prediction intervals.

5.2 Implementation Details

We pre-process the point cloud videos with Open3D [34] and generate cell features on a laptop with Apple M1 Pro chip, while training the graph model on an NVIDIA A100 GPU. Our graph model is implemented using PyTorch 2.3.0 and CUDA 12.2. We set the learning rate to 1e-4 and use a latent feature dimension of 128 for both the graph model and the GRU. Unless otherwise specified, a history length of 90 frames (3 seconds) is used for all methods. The model is trained for 30 epochs, with early stopping applied if the validation loss does not decrease for 5 consecutive epochs. For the 8i dataset, the entire space is partitioned into $5 \times 6 \times 8 = 240$ cells. The point cloud videos span approximately 1.8 m in height (Y-axis), with each 3D cell measuring around 0.2 m along all dimensions. For FSVVD, the space is partitioned into $7 \times 5 \times 8$ cells. The viewing camera image resolution is (1920*1080). The intrinsic matrix of camera is $\begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$, where $f_x = f_y = 525$, $c_x = 1920/2$, $c_y = 1080/2$.

To deploy CellSight in a real system, we must consider two main sources of latency: feature calculation latency and model inference latency. Take the 8i dataset as an example: on Mac M1 Pro Chip, we achieved 32.5fps without occlusion-aware features and 18.1fps with occlusion-aware features. And model inference time is 0.279 seconds. The GPU memory usage is 45GB to train the model when the batch size is 32. For faster HPR, the downsampled frame has around 15k points.

5.3 Baseline Methods and Evaluation Metrics

We compare our proposed model against several trajectory-based baseline models to evaluate its performance in predicting cell visibility. The trajectory-based models first predict the future viewport 6DoF coordinates $(\hat{x}, \hat{y}, \hat{z}, \hat{\psi}, \hat{\theta}, \hat{\phi})$ based on the historical viewport trajectory. These predicted coordinates are then used to calculate cell features defined in the Methodology section. In contrast, CellSight directly predicts cell features, including cell viewport overlap ratio, angular span, occlusion-aware visibility, and visible angular span, which are ready to be directly used in point cloud streaming systems for bit-rate allocation to 3D cells.

For trajectory-based methods, the angle values of ψ, θ, and ϕ wrap around, e.g. from 2π to 0. There are different approaches to address this issue. In our approach, we convert the orientation coordinates to the sine and cosine domains, using two coordinates to represent each angle coordinate. After prediction, we convert these coordinates back to the original angle value using the arctangent function. The baselines are as follows:

- **Linear Regression (LR):** We employ linear regression to predict each 6DoF coordinate individually by using a linear combination of historical values over a fixed window. This simple model serves as a fundamental baseline and, as demonstrated in [9], can sometimes yield higher FoV prediction accuracy than MLP. To accommodate different scenarios, we implement two variants—LR90 and LR30—using history window lengths of 90 and 30 frames, respectively.
- **Truncated Linear Regression (TLR):** This method leverages the most recent monotonically increasing or decreasing segment of the history window to linearly extrapolate future values. TLR has shown good performance in short-term FoV prediction [19] and, similar to LR, is applied to predict each coordinate individually.
- **Multi-task Multilayer Perceptron (MLP):** Following [11], we utilize a feedforward neural network to capture nonlinear relationships between the input features and the future FoV. As demonstrated in [11], predicting all coordinates simultaneously using the entire history window can enhance performance. Consequently, we use an MLP model with two hidden layers, each comprising 60 neurons in a fully connected architecture, and employ the ReLU activation function. This configuration is consistent with [9, 11, 20].
- **Multi-task LSTM (LSTM):** LSTM, a type of recurrent neural network (RNN), is well-suited for sequence prediction by learning long-term dependencies in time series data. In line with [11], we implement a two-layer LSTM with 60 neurons per layer to predict future FoV coordinates from historical data, with all coordinates predicted simultaneously.

There are also some other SOTA methods, e.g. [9, 20]. For [20], it is not open-sourced and the system is hard to reproduce. But we

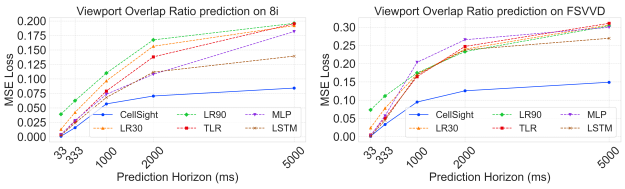
infer its performance based on its reported improvement over [9]. Authors of [9] built a point cloud streaming system considering Viewport Visibility (VV), Occlusion Visibility (OV), and Distance Visibility (DV), which are counterparts of our cell-based viewport feature, visibility feature and angular span feature. MLP and LSTM are used for FoV prediction. For the cell visible angular span prediction, we use MLP and LSTM to compare its performance with CellSight. Besides, we also use point-wise HPR to estimate the occlusion visibility to improve the its cell-based occlusion estimation.

We evaluate the performance of CellSight and the baselines by comparing the MSE across all cells for the predicted frame, including empty cells. This is because some cell may be temporarily empty when the objects move out of the space occupied by the cell. In CellSight, we will predict the visibility weight for each cell and the predicted weights can be directly used to determine the bit-rates allocated to all cells, not just cells falling into the predicted viewport. MSE is an accuracy metric for the visibility weight prediction. More accurate prediction leads to better cell bit-rate allocation for improved streaming quality. Besides, we use the R^2 score to assess the MSE relative to the variance of the ground truth variables, defined as $R^2 = 1 - \frac{\text{MSE}}{\text{GT variance}} = 1 - \frac{\sum_i \sum_f (y_{if} - \hat{y}_{if})^2}{\sum_i \sum_f (y_{if} - \bar{y})^2}$. In our experiments, we calculate residual variance and total variance over all cells and all frames. It measures the proportion of variance in the dependent variable that is predictable from the independent variables. An R^2 score of 1 indicates perfect prediction, and a ground truth mean prediction will a score of 0. It can be negative when model introduces more error than simply using the mean of the target data as the prediction. Good visual results are demonstrated when $R^2 >$ 0.43 in Fig. 11.

5.4 Prediction without Considering Occlusion: Cell Viewport Overlap Ratio and Angular Span

In this section, we will evaluate the viewport and angular span prediction. As discussed earlier, occlusion takes time to calculate, and even though we can downsample the point cloud video for HPR to achieve a real-time prediction, it is still a big overhead for a streaming system, which may be serving many clients. For some practical streaming system, sometimes it may be more appropriate to predict cell viewport overlap ratio without considering occlusion. We also notice that in 8i and FSVVD, if we only transmit the points inside the viewport (without considering occlusion), the total traffic for streaming can still be reduced to 1/2 and 1/6 of the original traffic. We apply CellSight on the cell viewport overlap ratio prediction and angular span prediction.

5.4.1 Viewport Overlap Ratio. For cell viewport overlap ratio prediction, the input features are occupancy feature, cell-based viewport feature, and other features, i.e., $G = \{O, F, E\}$. The Model will predict F. Fig. 6a illustrates the MSE for viewport overlap ratio prediction on the 8i dataset, demonstrating that CellSight consistently outperforms the others across varying prediction horizons, with substantially more gains at longer horizons. Similarly, Fig. 6b depicts the MSE for the prediction on the FSVVD dataset, where CellSight also achieves the lowest error compared to baseline methods, highlighting its robustness and accuracy.



(a) 8i Cell Viewport Overlap Ratio **(b) FSVVD Cell Viewport Overlap Ratio**

Figure 6: MSE losses for Cell Viewport Overlap Ratio

Table. 2 presents the R^2 scores for viewport overlap ratio prediction on the 8i dataset across different prediction horizons. Our proposed model consistently achieves the highest R^2 scores at all time steps. And the performance advantage of CellSight becomes more pronounced as the prediction horizon increases. Results on FSVVD dataset in Table. 3 show similar trend to 8i dataset, and CellSight outperforms the baselines, especially at longer prediction horizons. Since the user's trajectory data are more dynamic in FSVVD datset, some methods even produce negative R^2 scores at long time horizons, indicating poor performance. But CellSight still maintains positive scores, demonstrating its robustness and predictive capability even under challenging conditions.

Table 2: Viewport Overlap Ratio R^2 Scores on 8i

Time (ms)	LR30	LR90	TLR	MLP	LSTM	CellSight
33	0.941	0.824	0.982	0.983	0.989	**0.995**
333	0.809	0.718	0.879	0.873	0.893	**0.928**
1000	0.563	0.504	0.644	0.669	0.693	**0.743**
2000	0.294	0.246	0.378	0.514	0.497	**0.682**
5000	0.135	0.114	0.116	0.178	0.370	**0.618**

Table 3: Viewport Overlap Ratio R^2 Scores on FSVVD

Time (ms)	LR30	LR90	TLR	MLP	LSTM	CellSight
33	0.876	0.621	0.978	0.987	**0.995**	0.994
333	0.597	0.425	0.738	0.706	0.769	**0.828**
1000	0.112	0.100	0.153	-0.049	0.127	**0.512**
2000	-0.240	-0.195	-0.268	-0.362	-0.220	**0.355**
5000	-0.562	-0.558	-0.595	-0.539	-0.385	**0.232**

We also visualize the predicted viewport overlap ratio. We choose 2 second ahead as the prediction horizon to visualize it. In Fig. 7, the MSE loss for CellSight and LSTM are 0.05 and 0.14, respectively. Visually, our predicted viewport overlap ratio is closer to the ground-truth, especially at the edges of the viewport. LSTM prediction missed a large part of the leg, hands and gun, since its FoV coordinates were overly influenced by the historical trajectory alone. Our prediction can cover more around the edges of the viewport. In addition, CellSight exhibits smoother boundaries compared to trajectory-based models. This smoothness arises because neural networks typically filter out high-frequency signals, resulting in a naturally smoother output. When the predicted cell visibility is used to determine streaming bandwidth allocation, cells surrounding the FoV are also allocated with non-zero rates. This increases the streaming system's robustness against FoV prediction errors.

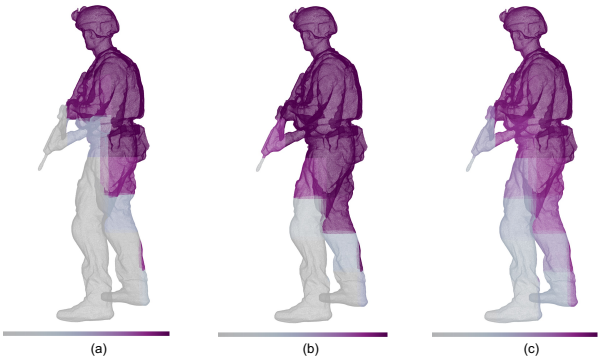


Figure 7: Cell viewport overlap ratio prediction results on the 8i dataset, where the prediction horizon is 2000ms. Visual comparison of predicted viewport using (a) LSTM model, (b) Ground Truth, and (c) CellSight. The color represents the prediction confidence, transitioning from dark (low confidence) to bright (high confidence).

Beyond using one video for testing/validation to provide a straightforward demonstration of generalization, we also conduct cross-validation by rotating the testing/validation videos among four PCVs. The cross-validation results are shown in Table. 4. CellSight has lower MSE and higher R^2 compared with LSTM, which further demonstrates the generalization of CellSight.

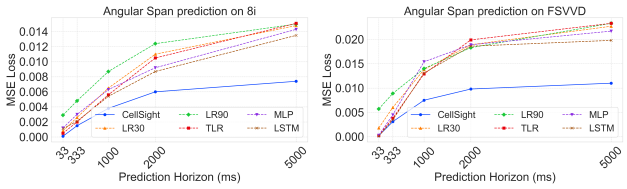
Table 4: Viewport Overlap Ratio Prediction Cross Validation on 8i

Testing	CellSight		LSTM	
	MSE↓	R^2↑	MSE↓	R^2↑
Longdress	0.0714	0.675	0.095	0.573
Redandblack	0.0610	0.720	0.100	0.545
Loot	0.0683	0.679	0.099	0.534
Soldier	0.0710	0.682	0.112	0.497
Average	**0.0679**	**0.689**	0.1015	0.5373

5.4.2 Angular Span. To predict the angular span for each cell, the history angular spans for all cells are added beyond the features of viewport overlap ratio prediction feature set, i.e., $G = \{O, F, E, A\}$. The model will predict A. The prediction MSE is reported in Fig. 8a and Fig. 8b respectively. CellSight consistently outperforms baselines, except in one case for the FSVVD dataset. In that case, LSTM is slightly better than CellSight for the very short predicted horizon of 33 ms (1 frame). R^2 score is displayed in Table 5 and Table 6.

Table 5: Angular Span R^2 Scores on 8i

Time (ms)	LR30	LR90	TLR	MLP	LSTM	CellSight
33	0.957	0.835	0.975	0.934	0.941	**0.994**
333	0.857	0.733	0.890	0.835	0.882	**0.917**
1000	0.639	0.516	0.687	0.651	0.696	**0.787**
2000	0.387	0.308	0.418	0.490	0.516	**0.668**
5000	0.174	0.162	0.156	0.199	0.246	**0.585**



(a) 8i Angular Span **(b) FSVVD Angular Span**

Figure 8: MSE losses for Angular Span

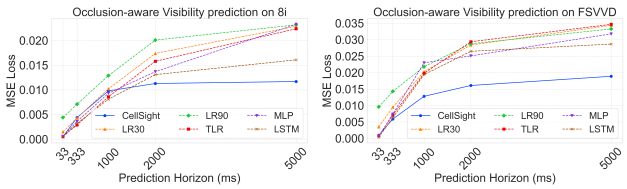
Table 6: Angular Span R^2 Scores on FSVVD

Time (ms)	LR30	LR90	TLR	MLP	LSTM	CellSight
33	0.881	0.617	0.978	0.983	**0.993**	0.988
333	0.594	0.403	0.743	0.698	0.766	**0.794**
1000	0.069	0.058	0.131	-0.033	0.117	**0.493**
2000	-0.275	-0.227	-0.334	-0.269	-0.251	**0.338**
5000	-0.524	-0.560	-0.564	-0.453	-0.329	**0.261**

5.5 Prediction considering Occlusion: Cell Occlusion-aware Visibility and Visible Angular Span

To achieve more bandwidth saving in point cloud streaming, FoV prediction can additionally consider occlusion after HPR. In this section, we evaluate the occlusion-aware visibility and visible angular span predictions.

5.5.1 Occlusion-aware Visibility. For the cell occlusion-aware visibility prediction, beyond all features used in cell viewport overlap ratio prediction, the cell occlusion-aware visibility history is added, i.e., $G = \{O, F, V, E\}$. The model will predict V. In Fig. 9a, while our proposed model demonstrates superior performance in longer prediction horizons, some baseline methods, such as LSTM, achieve slightly better results in shorter horizons due to the less dynamic nature of user behaviors in the 8i dataset. However, CellSight's long-term accuracy shows its robustness and effectiveness. For the more dynamic user FoV data FSVVD, as shown in Fig. 9b, CellSight consistently achieves the best performance across all prediction horizons. This highlights the adaptability and reliability of CellSight in handling scenarios with frequent changes in user focus, making it well-suited for dynamic FoV prediction. R^2 scores on 8i and FSVVD datasets are shown in Table 7 and Table 8. CellSight achieves the best performance for medium and long term horizon, especially on more dynamic FSVVD dataset.



(a) 8i Occlusion-aware Visibility **(b) FSVVD Occlusion-aware Visibility**

Figure 9: MSE losses for Occlusion-aware Visibility

Table 7: Occlusion-aware Visibility R^2 Scores on 8i

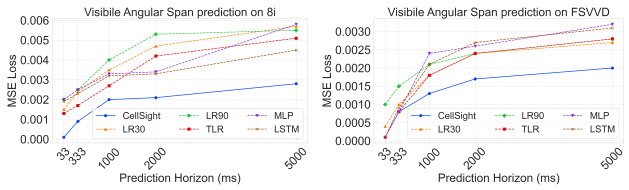
Time (ms)	LR30	LR90	TLR	MLP	LSTM	CellSight
33	0.965	0.897	0.986	0.985	**0.990**	0.987
333	0.898	0.836	**0.934**	0.916	0.926	0.896
1000	0.761	0.702	0.803	0.786	**0.814**	0.774
2000	0.595	0.537	0.636	0.684	0.699	**0.739**
5000	0.468	0.475	0.494	0.473	0.637	**0.736**

Table 8: Occlusion-aware Visibility R^2 Scores on FSVVD

Time (ms)	LR30	LR90	TLR	MLP	LSTM	CellSight
33	0.883	0.683	0.974	0.978	**0.988**	0.969
333	0.685	0.530	0.765	0.757	0.792	**0.807**
1000	0.331	0.284	0.349	0.244	0.362	**0.578**
2000	0.067	0.074	0.053	0.193	0.144	**0.480**
5000	-0.094	-0.046	-0.092	0.001	0.094	**0.402**

5.5.2 Visible Angular Span. To consider viewport, visibility and angular span all together, we evaluate CellSight when predicting the visible angular span β on the two datasets. We incorporate the cell's visible angular span feature β history along with the history of all the other features, i.e., $G = \{O, F, V, E, A, B\}$. The model will predict B. Figure 10a and Fig. 10b show that CellSight consistently achieves the best performance, particularly for longer prediction horizons and even in less dynamic user scenarios, highlighting its ability to effectively predict visible angular span. These results show that CellSight is not only accurate in combining viewport, visibility, and angular features but also robust across datasets with varying levels of user interaction dynamics. Table 9 and Table 10 present the R^2 scores for visible angular span for 8i and FSVVD.

Additionally, we visualize the visible angular span prediction on one frame, where prediction horizon is 2000ms in FSVVD data in Fig. 11, where the prediction MSE values for CellSight and LSTM are 0.0016 and 0.0026, respectively, which is close to the average MSE across all frames for both methods in Fig. 10b. The prediction by CellSight closely matches the ground-truth, whereas LSTM produces significantly different results due to error amplification in the degrees of freedom, leading to substantial discrepancies in the final visible cell angular spans.



(a) 8i Visible Angular Span **(b) FSVVD Visible Angular Span**

Figure 10: MSE losses for Visible Angular Span

6 Conclusion & Future Work

In this paper, we introduce **CellSight**, a novel approach for predicting long-term cell visibility in PCV. CellSight leverages both the spatial and temporal dynamics of PCV content and viewer behaviors to directly predict cell visibility for streaming systems. It outperforms state-of-the-art methods in terms of accuracy and robustness, particularly at long prediction horizons (beyond 2 seconds). By integrating Transformer-based Graph Neural Networks with recurrent neural networks, CellSight efficiently captures the complex interactions between PCV content and viewer interests, as well as the correlations among neighboring cells. Unlike the trajectory-based FoV prediction methods, which often overlook the full spatial context, our method yields more accurate and stable predictions for long-term 6-DoF FoV prediction.

For future work, we plan to utilize the predicted cell visibility metrics to guide cell-level bandwidth allocation in on-demand PCV streaming. Furthermore, while the current approach relies on a centralized architecture—where all computations are performed on the streaming server—we intend to explore distributed computing strategies to reduce server load and enhance system scalability. We have made the code and technical report publicly available[2] to support further research and development in this area.

Table 9: Visible Angular Span R^2 Scores on 8i

Time (ms)	LR30	LR90	TLR	MLP	LSTM	CellSight
33	0.873	0.827	0.888	0.826	0.829	**0.990**
333	0.788	0.777	0.855	0.779	0.794	**0.924**
1000	0.687	0.650	0.766	0.692	0.722	**0.828**
2000	0.587	0.550	0.639	0.707	0.717	**0.822**
5000	0.504	0.530	0.562	0.503	0.614	**0.760**

Table 10: Visible Angular Span R^2 Scores on FSVVD

Time (ms)	LR30	LR90	TLR	MLP	LSTM	CellSight
33	0.871	0.649	0.961	0.965	**0.974**	0.955
333	0.671	0.484	**0.741**	0.728	0.713	0.728
1000	0.388	0.297	0.400	0.175	0.286	**0.544**
2000	0.189	0.210	0.206	0.136	0.111	**0.436**
5000	0.124	0.095	0.087	-0.041	-0.008	**0.364**

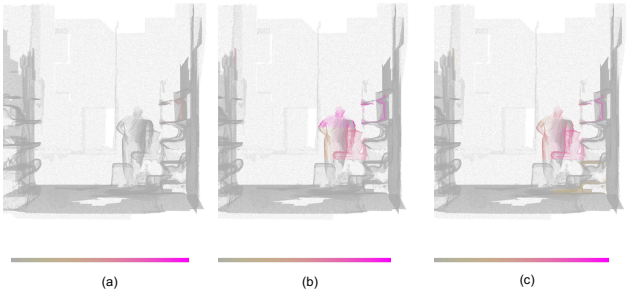


Figure 11: Visible angular span prediction for each cell on one frame of FSVVD dataset, where the prediction horizon is 2000ms. (a) LSTM model, (b) Ground Truth, and (c) CellSight. The color transition from gray to red corresponds to visible angular span from small to large.

7 Acknowledgments

This work was supported in part by the National Science Foundation (NSF) grant 2312839.

[2]https://github.com/chenli1996/CellSight

References

[1] Evangelos Alexiou, Yana Nehmé, Emin Zerman, Irene Viola, Guillaume Lavoué, Ali Ak, Aljosa Smolic, Patrick Le Callet, and Pablo Cesar. 2023. Subjective and objective quality assessment for volumetric video. In *Immersive Video Technologies*. Elsevier, 501–552.

[2] Yixuan Ban, Lan Xie, Zhimin Xu, Xinggong Zhang, Zongming Guo, and Yue Wang. 2018. Cub360: Exploiting cross-users behaviors for viewport prediction in 360 video adaptive streaming. In *2018 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE, 1–6.

[3] Salima Bourbia, Ayoub Karine, Aladine Chetouani, Mohammed El Hassouni, and Maher Jridi. 2024. Blind point cloud quality assessment via 3D visual saliency and point-based neural network. In *The 13th International Conference on Image Processing Theory, Tools and Applications*.

[4] Ruopeng Chen, Mengbai Xiao, Dongxiao Yu, Guanghui Zhang, and Yao Liu. 2023. patchVVC: A real-time compression framework for streaming volumetric videos. In *Proceedings of the 14th Conference on ACM Multimedia Systems*. 119–129.

[5] Jaeyeol Choi, Jong-Beom Jeong, Soonbin Lee, and Eun-Seok Ryu. 2022. Overview of the Volumetric Video Capturing System for Immersive Media. In *2022 13th International Conference on Information and Communication Technology Convergence (ICTC)*. IEEE, 574–577.

[6] Yuxin Du, Fan Bai, Tiejun Huang, and Bo Zhao. 2023. Segvol: Universal and interactive volumetric medical image segmentation. *arXiv preprint arXiv:2311.13385* (2023).

[7] Eugene d'Eon, Bob Harrison, Taos Myers, and Philip A Chou. 2017. 8i voxelized full bodies-a voxelized point cloud dataset. *ISO/IEC JTC1/SC29 Joint WG11/WG1 (MPEG/JPEG) input document WG11M40059/WG1M74006* 7, 8 (2017), 11.

[8] Ching-Ling Fan, Jean Lee, Wen-Chih Lo, Chun-Ying Huang, Kuan-Ta Chen, and Cheng-Hsin Hsu. 2017. Fixation prediction for 360 video streaming in head-mounted virtual reality. In *Proceedings of the 27th workshop on network and operating systems support for digital audio and video*. 67–72.

[9] Bo Han, Yu Liu, and Feng Qian. 2020. ViVo: Visibility-aware mobile volumetric video streaming. In *Proceedings of the 26th annual international conference on mobile computing and networking*. 1–13.

[10] Hangtao He, Linyu Su, and Kejiang Ye. 2023. Graphgru: A graph neural network model for resource prediction in microservice cluster. In *2022 IEEE 28th International Conference on Parallel and Distributed Systems (ICPADS)*. IEEE, 499–506.

[11] Xueshi Hou and Sujit Dey. 2020. Motion Prediction and Pre-Rendering at the Edge to Enable Ultra-Low Latency Mobile 6DoF Experiences. IEEE Open Journal of the Communications Society 1 (2020), 1674–1690.

[12] Kaiyuan Hu, Yongting Chen, Kaiying Han, Junhua Liu, Haowen Yang, Yili Jin, Boyan Li, and Fangxin Wang. 2023. LiveVV: Human-Centered Live Volumetric Video Streaming System. *arXiv preprint arXiv:2310.08205* (2023).

[13] Kaiyuan Hu, Yili Jin, Haowen Yang, Junhua Liu, and Fangxin Wang. 2023. FSVVD: A dataset of full scene volumetric video. In *Proceedings of the 14th Conference on ACM Multimedia Systems*. 410–415.

[14] Kaiyuan Hu, Haowen Yang, Yili Jin, Junhua Liu, Yongting Chen, Miao Zhang, and Fangxin Wang. 2023. Understanding user behavior in volumetric video watching: Dataset, analysis and prediction. In *Proceedings of the 31st ACM International Conference on Multimedia*. 1108–1116.

[15] Sagi Katz, Ayellet Tal, and Ronen Basri. 2007. Direct visibility of point sets. In *ACM SIGGRAPH 2007 papers*. 24–es.

[16] Muhammad Jalal Khan, Abdelhak Bentaleb, and Saad Harous. 2021. Can accurate future bandwidth prediction improve volumetric video streaming experience?. In *2021 International Wireless Communications and Mobile Computing (IWCMC)*. IEEE, 1041–1047.

[17] Chen Li, Xiaoyu Wang, Tongyu Zong, Houwei Cao, and Yong Liu. 2023. Predictive edge caching through deep mining of sequential patterns in user content retrievals. *Computer Networks* 233 (2023), 109866.

[18] Chen Li, Tingwei Ye, Tongyu Zong, Liyang Sun, Houwei Cao, and Yong Liu. 2023. Coffee: Cost-Effective Edge Caching for 360 Degree Live Video Streaming. *arXiv preprint arXiv:2312.13470* (2023).

[19] Chenge Li, Weixi Zhang, Yong Liu, and Yao Wang. 2019. Very long term field of view prediction for 360-degree video streaming. In *2019 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*. IEEE, 297–302.

[20] Junhua Liu, Boxiang Zhu, Fangxin Wang, Yili Jin, Wenyi Zhang, Zihan Xu, and Shuguang Cui. 2023. Cav3: Cache-assisted viewport adaptive volumetric video streaming. In *2023 IEEE Conference Virtual Reality and 3D User Interfaces (VR)*. IEEE, 173–183.

[21] Yu Liu, Bo Han, Feng Qian, Arvind Narayanan, and Zhi-Li Zhang. 2022. Vues: Practical mobile volumetric video streaming through multiview transcoding. In *Proceedings of the 28th Annual International Conference on Mobile Computing And Networking*. 514–527.

[22] Ohji Nakagami, Sebastien Lasserre, Sugio Toshiyasu, and Marius Preda. 2023. White paper on G-PCC. In *ISO/IEC JTC 1/SC 29/AG 03 N0111*. https://www.mpeg.org/wp-content/uploads/mpeg_meetings/142_Antalya/w22804.zip

[23] Minh Nguyen, Shivi Vats, Sam Van Damme, Jeroen Van Der Hooft, Maria Torres Vega, Tim Wauters, Christian Timmerer, and Hermann Hellwagner. 2023. Impact of quality and distance on the perception of point clouds in mixed reality. In *2023 15th International Conference on Quality of Multimedia Experience (QoMEX)*. IEEE, 87–90.

[24] Michael Rudolph, Aron Riemenschneider, and Amr Rizk. 2024. Progressive Coding for Deep Learning based Point Cloud Attribute Compression. In *Proceedings of the 16th International Workshop on Immersive Mixed and Virtual Environment Systems*. 78–84.

[25] Yunsheng Shi, Zhengjie Huang, Shikun Feng, Hui Zhong, Wenjin Wang, and Yu Sun. 2020. Masked label prediction: Unified message passing model for semi-supervised classification. *arXiv preprint arXiv:2009.03509* (2020).

[26] Han Song, Zhongche Qu, Zhi Zhang, Zihan Ye, and Cong Liu. 2024. Advancements in translation accuracy for stereo visual-inertial initialization. In *2024 9th Asia-Pacific Conference on Intelligent Robot Systems (ACIRS)*. IEEE, 210–215.

[27] Shishir Subramanyam, Irene Viola, Alan Hanjalic, and Pablo Cesar. 2020. User centered adaptive streaming of dynamic point clouds with low complexity tiling. In *Proceedings of the 28th ACM international conference on multimedia*. 3669–3677.

[28] Liyang Sun, Fanyi Duanmu, Yong Liu, Yao Wang, Yinghua Ye, Hang Shi, and David Dai. 2019. A two-tier system for on-demand streaming of 360 degree video over dynamic networks. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* 9, 1 (2019), 43–57.

[29] Jeroen van der Hooft, Maria Torres Vega, Christian Timmerer, Ali C Begen, Filip De Turck, and Raimund Schatz. 2020. Objective and subjective QoE evaluation for adaptive point cloud streaming. In *2020 twelfth international conference on quality of multimedia experience (QoMEX)*. IEEE, 1–6.

[30] Lisha Wang, Chenglin Li, Wenrui Dai, Junni Zou, and Hongkai Xiong. 2021. QoE-driven and tile-based adaptive streaming for point clouds. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 1930–1934.

[31] Joyce HDM Westerink and Jacques AJ Roufs. 1989. Subjective image quality as a function of viewing distance, resolution, and picture size. *SMPTE journal* 98, 2 (1989), 113–119.

[32] Xihe Yang, Xingyu Chen, Daiheng Gao, Shaohui Wang, Xiaoguang Han, and Baoyuan Wang. 2024. Have-fun: Human avatar reconstruction from few-shot unconstrained images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 742–752.

[33] Yunxiang Zhang, Benjamin Liang, Boyuan Chen, Paul M Torrens, S Farokh Atashzar, Dahua Lin, and Qi Sun. 2022. Force-aware interface via electromyography for natural VR/AR interaction. *ACM Transactions on Graphics (TOG)* 41, 6 (2022), 1–18.

[34] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. 2018. Open3D: A Modern Library for 3D Data Processing. *arXiv:1801.09847* (2018).

[35] Tongyu Zong, Chen Li, Yuanyuan Lei, Guangyu Li, Houwei Cao, and Yong Liu. 2022. Cocktail edge caching: Ride dynamic trends of content popularity with ensemble learning. *IEEE/ACM Transactions on Networking* 31, 1 (2022), 208–219.

[36] Tongyu Zong, Yixiang Mao, Chen Li, Yong Liu, and Yao Wang. 2023. Progressive Frame Patching for FoV-based Point Cloud Video Streaming. *arXiv preprint arXiv:2303.08336* (2023).