CLiFT: Compressive Light-Field Tokens for Compute Efficient and Adaptive Neural Rendering

Zhengqing Wang[1] **Yuefan Wu**[1] **Jiacheng Chen**[1]
Fuyang Zhang[1] **Yasutaka Furukawa**[1,2]

[1]Simon Fraser University [2]Wayve

Abstract

This paper proposes a neural rendering approach that represents a scene as "compressed light-field tokens (CLiFTs)", retaining rich appearance and geometric information of a scene. CLiFT enables compute-efficient rendering by compressed tokens, while being capable of changing the number of tokens to represent a scene or render a novel view with one trained network. Concretely, given a set of images, multi-view encoder tokenizes the images with the camera poses. Latent-space K-means selects a reduced set of rays as cluster centroids using the tokens. The multi-view "condenser" compresses the information of all the tokens into the centroid tokens to construct CLiFTs. At test time, given a target view and a compute budget (i.e., the number of CLiFTs), the system collects the specified number of nearby tokens and synthesizes a novel view using a compute-adaptive renderer. Extensive experiments on RealEstate10K and DL3DV datasets quantitatively and qualitatively validate our approach, achieving significant data reduction with comparable rendering quality and the highest overall rendering score, while providing trade-offs of data size, rendering quality, and rendering speed. Check demos and code on the project page: `https://clift-nvs.github.io`.

1 Introduction

Global consumption of visual media is skyrocketing, driven by platforms like Instagram, YouTube, and TikTok. Billions of photos and videos are captured, shared, and streamed daily, placing enormous demands on storage and bandwidth. The success of these platforms owes much to advances in visual data compression, allowing high-resolution content to be delivered efficiently across a range of devices and network conditions. From images to high-definition videos, modern compression algorithms have enabled rich visual experiences to become a ubiquitous part of everyday life.

Beyond passive viewing of recorded media, interactive novel view synthesis (NVS) is gaining momentum, allowing users to freely navigate virtual environments. Neural rendering techniques such as Neural Radiance Fields (NeRF) and 3D Gaussian Splatting (3DGS) are the driving forces. Recent research has explored efficiency of neural rendering pipelines, including compression of radiance fields [24, 35], sparse representations [17, 22], and adaptive quality rendering [9, 36]

Along the line of interactive visual media, reconstruction-free novel view synthesis is emerging as a promising direction. Models such as Large View Synthesis Models (LVSM) [12] and Scene Representation Transformers (SRT) [25] synthesize novel views directly, without defining bottleneck geometric and photometric representations by heuristics and reconstructing them. By avoiding explicit reconstruction, these methods would better handle scene dynamics and capture fine-grained visual details directly from the data. Compute-efficient representations and compute-adaptive rendering within these systems would unlock the full potential of NVS technology, driving new applications

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

across real estate (virtual property tours), entertainment (immersive media and games), online shopping (interactive product displays), and autonomous driving (simulation and model validation).

Towards realizing this potential, this paper introduces a new representation and rendering framework centered on the concept of Compressive Light-Field Tokens (CLiFT). CLiFT is a compact set of light field rays with compressed learned embeddings. Concretely, given a set of images with camera poses as input, multi-view encoder tokenizes the images with camera poses. A latent-space K-means algorithm selects a reduced set of rays as cluster centers. Intuitively, the resulting cluster centers preserve coverage due to geometric diversity among rays, while becoming denser in texture-rich regions. Multi-view "condenser" compresses the information of all the embeddings into the centroid tokens to construct CLiFTs At test time, given a target camera pose and a compute budget (i.e., the number of CLiFTs to use), the system collects the corresponding nearby tokens and synthesizes a novel view using a compute-adaptive renderer that is trained to handle a variable number of tokens.

We have evaluated the approach on RealEstate10K [37] and DL3DV [16] datasets, and compared with three state-of-the-art methods, LVSM [12] from the reconstruction-free approach, and MVSplat [3] and DepthSplat [34] from the reconstruction-based approach. Extensive quantitatively and qualitatively validate our approach, achieving significant data reduction with comparable rendering quality and the highest overall rendering score, while providing trade-offs of data size, rendering quality, and rendering speed.

2 Related Work

Light Field Imaging. Computational light field imaging techniques date back to the 1990s with notable work by Levoy *et al*. on Light Field Rendering [15] and Gortler *et al*. on Lumigraph [10]. Both introduced the idea of capturing a dense array of rays from multiple viewpoints, enabling novel view generation without geometry reconstruction. Subsequent efforts, such as multi-camera arrays [32] and hand-held plenoptic cameras [23], brought these ideas into practical systems. For example, commercial light field cameras like the Lytro (circa 2011) demonstrated post-capture refocusing and viewpoint adjustment. Recent works combine classical light field concepts with neural rendering to enable efficient and accurate novel view synthesis with view-dependent effects [5, 26, 28, 29]. The idea of using light fields (or rays) as a scene representation forms the core of our approach, where we combine this classical concept with modern neural rendering techniques.

Compressive Sensing. In the mid-2000s, compressive sensing emerged as a paradigm for capturing signals with far fewer samples than traditional Nyquist sampling theory would require, given the signal is sparse in some basis [1, 4]. In computational imaging, compressive sensing inspired novel camera designs that sample and project measurements into lower-dimensional spaces. Notable examples include the single-pixel camera [6] and coded aperture systems [20] that optically encode high-dimensional scene information into compressed sensor readings. Similar to compressive sensing, our approach selects which rays to store and how to represent them in a compact form for successful view synthesis. A key distinction is that our method learns to compress all the input rays into a compact set of representative rays using neural networks, whereas compressive sensing relies on predefined heuristics to drop information and project to a lower dimension.

Reconstruction-Based Novel View Synthesis. Early image-based rendering (IBR) systems relied on reconstructed scene geometry for novel view generation. Photo Tourism is a seminal example, using a planar geometric proxy for rendering—simple yet producing compelling visual experiences [27]. Subsequent work extended this idea by more accurate depth maps to enhance rendering quality [8]. More recently, Neural Radiance Fields (NeRF) introduced a continuous volumetric scene representation as a 5D function that maps spatial location and viewing direction to color and density using an MLP [21]. 3D Gaussian Splatting (3DGS) eliminates neural networks in favor of explicit point-based primitives, representing the scene as millions of 3D Gaussians and rendering images by projecting ("splatting") them with view-dependent shading [13]. These methods typically require per-scene optimization, lack generalization across scenes, and assume dense input coverage. To overcome the limitations, feed-forward Gaussian splatting methods [2, 3, 34] eliminate per-scene optimization by predicting Gaussian parameters in a single forward pass. In parallel, image and video generative models have been incorporated to refine NVS outputs, enhancing realism and temporal consistency [7, 31]. Recent methods like Reconfusion [33] and NerfDiff [11] take a step further by using generative refinement.

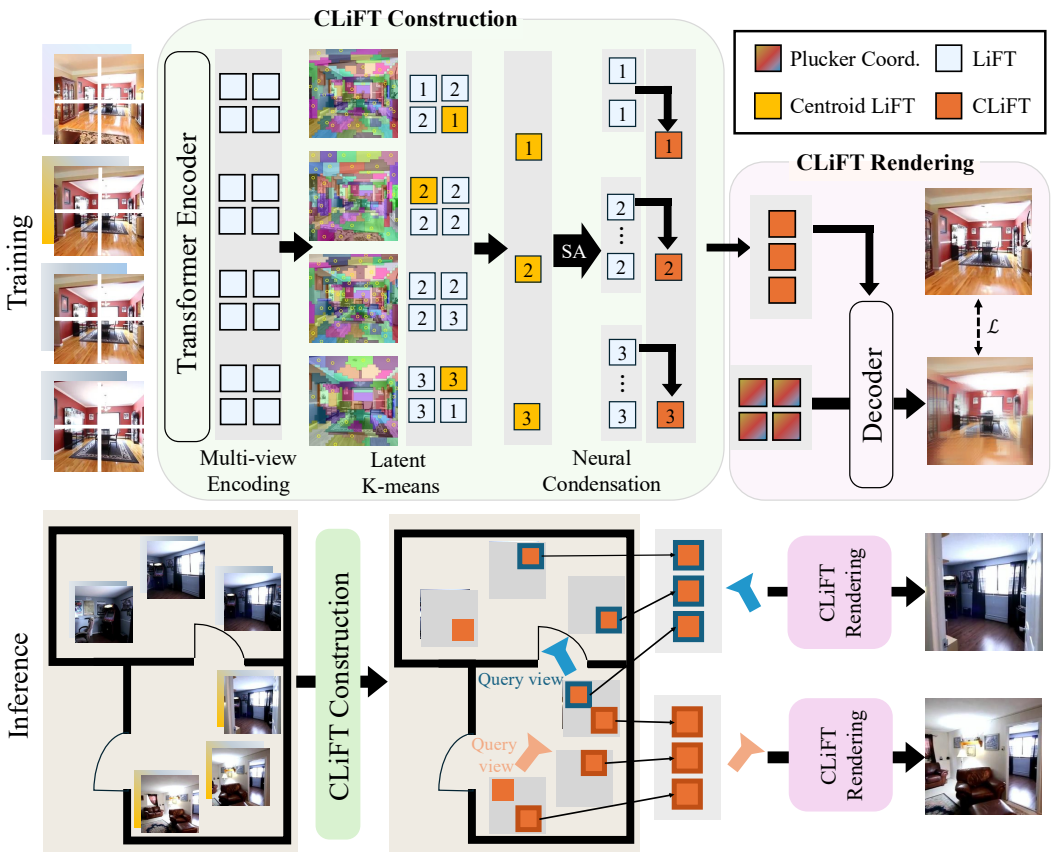

Figure 1: The training and the inference system overview. **Top:** The training consists of three steps: 1) Multi-view encoder, tokenizing the input images; 2) Latent K-means, selecting a representative set of tokens; and 3) Neural condensation, compressing the information of all the tokens into the representative set to produce Compressed Light-Field Tokens (CLiFTs). **Bottom:** At inference time, multi-view images are encoded into CLiFTs following the same process as in training. Given a target view, we collect a relevant set of CLiFTs with simple heuristics and render a novel view.

Reconstruction-Free Novel View Synthesis. Reconstruction-free novel view synthesis is gaining attention as an end-to-end solution. The Scene Representation Transformer (SRT) encodes a set of input images into a latent scene representation—a collection of tokens—and renders novel views from this latent in a single feed-forward pass [25]. Another representative work is LVSM (Large View Synthesis Model), a fully transformer-based pipeline that achieves state-of-the-art results from sparse inputs [12]. LVSM includes both an encoder–decoder transformer, which compresses input images into a fixed-length latent code before decoding novel views, and a decoder-only transformer that maps input views directly to output pixels. Multi-view generative models share similar characteristics, producing consistent images across views without reconstructing or generating explicit geometry [14, 18, 19, 30]. This paper advances the frontier of reconstruction-free NVS by introducing Compressive Light-Field Tokens: a compact, variable-size scene representation paired with a renderer that adaptively balances quality and resource usage on demand.

3 Compressive Light-Field Tokens (CLiFT)

A light field is a collection of rays, each associated with the radiance information. Compressive light-field tokens (CLiFTs) are a collection of rays, each associated with a latent vector to which a neural encoder compresses the geometry and radiance information of a scene (See Figure 1). Given a target camera pose, a traditional light field rendering synthesizes a color for each target ray, typically by linear interpolation of nearby input rays. CLiFT rendering synthesizes an image (not necessarily per ray) by using nearby CLiFTs via a neural renderer. The framework allows us to control data size,

rendering quality, and rendering speed by two hyper parameters 1) Storage CLiFT count (N_s), the number of tokens to represent a scene; and 2) Render CLiFT count (N_r), the number of tokens to use in rendering a frame. The section explains the problem, CLiFT construction, and CLiFT rendering.

3.1 Problem Definition

This paper tackles the problem of taking N_c images with the camera poses, and constructing N_s CLiFTs as a scene representation. Given a target view, the output is its synthesized image, where N_r CLiFTs tokens are to be selected and used for rendering. The evaluation is based on standard image reconstruction metrics, PSNR, SSIM, and LPIPS. A token (CLiFT) is a D dimensional embedding, associated with a ray. Ray geometries are represented in a global coordinate frame.

3.2 CLiFT Construction

Construction of compressed light field tokens (CLiFTs) involves three key steps: encoding multi-view images and camera poses, selecting representative rays, and condensing information into the selected set. The section explains the steps.

Multi-view Encoding. Given images with camera poses, we use a standard Transformer encoder to extract our Light Field Tokens (LiFT), which capture both geometry and appearance information. Specifically, for each pixel in each image, we concatenate the 6D Plücker coordinates of the corresponding ray with its normalized 3D color vector. We then patchify non-overlapping 8×8 regions, resulting in 576 = (3 + 6)×8×8 dimensional vectors. [1] Each vector is linearly projected to a token of dimension $D = 768$. This process converts N_c input images of resolution 256×256 into 1024 $\cdot N_c$ tokens. The Transformer encoder performs self-attention across all 1,024 $\cdot N_c$ tokens, producing a total of 4,096 LiFTs for a scene of 4 input views. Specifically, our encoder has six self-attention blocks, each comprising Self-Attention (8 heads), Add & LayerNorm, Feedforward Network, and another Add & LayerNorm. Unlike the encoder-decoder architecture in LVSM, which uses a learnable token to aggregate scene information, our method directly uses the outputs of Transformer encoder, which retains the geometry and appearance of a specific region.

Latent-space K-means for Ray Selection. Effective ray selection is crucial, as uniform ray selection across input images leads to two redundancies: 1) Appearance redundancy at texture homogeneous regions; and 2) Geometric redundancy at visual overlaps among different views. Our approach performs latent-space clustering to select representative rays that compactly represent an entire scene. K-means clustering algorithm finds the clusters from LiFT of all the images. The nearest neighbor sample from each center is retained as the cluster centroid. Centroid LiFT will be the storage CLiFTs, and K is set to N_s (i.e, the number of storage CLiFTs).

Neural Condensation. A lightweight transformer condenses information from all LiFTs into a set of centroid LiFTs, producing CLiFTs. Self-attention operates over the centroid LiFTs to enable information exchange across clusters. Within each cluster, cross-attention uses the centroid LiFT as the query and the remaining LiFTs as keys and values. To preserve the pretrained latent space, a zero-initialized linear layer aggregates features back into the centroid LiFTs.

Concretely, the condensation network consists of two transformer decoder blocks. Each block includes inter-cluster self-attention (8 heads), intra-cluster cross-attention (8 heads), and standard Add & LayerNorm operations. Let $T_k \in \mathbb{R}^D$ denote the embedding of the k-th centroid LiFT, and $\{T_{k,i} \mid i\}$ represent the remaining LiFTs assigned to the k-th cluster. SA$(\cdot)$, CA$(\cdot,\cdot)$, and FFN$(\cdot)$ denote the self-attention, cross-attention, and feed-forward network layers, respectively. W_z is a zero-initialized linear projection operator $W_z \in \mathbb{R}^{D\times D}$. Each decoder block updates the centroid

[1] As explained in §4, we use RealEstate10K [37] and DL3DV [16] datasets. Image resolutions are 256×256 and 256×448, respectively. The section uses RealEstate10K as an example to explain the feature dimensions. The patch size is the same for DL3DV, resulting in 1,792 $\cdot N_c$ tokens instead.

embeddings as follows, producing CLiFTs after the second block:

$$\{\hat{T}_k\} \leftarrow \text{SA}\left(\{\text{LN}(T_1), \text{LN}(T_2), \cdots\}\right), \tag{1}$$

$$\forall k \quad \hat{T}_k \leftarrow \text{CA}(\hat{T}_k, \{\text{LN}(T_{k,1}), \text{LN}(T_{k,2}), \cdots\}), \tag{2}$$

$$\forall k \quad \hat{T}_k \leftarrow \text{FFN}(\text{LN}(\hat{T}_k)), \tag{3}$$

$$\forall k \quad T_k \leftarrow T_k + W_z(\hat{T}_k). \tag{4}$$

3.3 CLiFT Rendering

CLiFTs enable a flexible and efficient rendering mechanism. Given a target view and the render CLiFT count (i.e., the number of tokens to use), we collect a set of relevant CLiFTs, which are then used by a neural renderer to synthesize the view.

Neural Renderer. The renderer is a simple Transformer decoder where the target view serves as the query and the selected CLiFTs serve as keys and values. Specifically, we initialize a 2D grid of Plücker coordinates at each pixel and patchify them. Each patch is represented as a 384-dimensional vector derived from the 6×8×8 Plücker coordinates, which is projected to 768 dimensions via a linear layer. The rendering network has six decoder blocks, each comprising Self-Attention (among query tokens), Add & LayerNorm, Cross-Attention (from CLiFTs to query tokens), Add & LayerNorm, Feedforward Network, and Add & LayerNorm. The number of heads is eight in SA and CA layers. The decoder output is passed through a linear projection, followed by a sigmoid activation to map it to the RGB space, and then an unpatchify operation to reconstruct the full-resolution image. We use a combination of L2 loss and perceptual loss (with a 0.5 weight) following LVSM [12], applied to the normalized RGB intensities. During training, we randomly vary the number of CLiFTs passed to the decoder, allowing it to learn how to handle different token counts and enabling dynamic trade-offs between rendering quality and computational cost.

Token Selection. The token selection algorithm employs a simple heuristic. Let N_r denote the number of CLiFTs used for rendering. To ensure spatial coverage of the target view, we divide the view into a 16×16 grid of patches and pad it with a 4-patch margin on all sides, resulting in a 24×24 grid. For each patch, we cast a ray through its center and retrieve the $N_r/(24 \times 24)$ closest CLiFTs from a pool of N_s storage CLiFTs. The distance between a patch and a CLiFT is computed using a heuristic based on their associated rays (ray origins and directions); details are provided in the supplementary material. Since CLiFTs may be selected multiple times, we greedily accumulate CLiFTs in order of their minimum distance to any patch ray until N_r unique tokens are selected.

4 Experiments

4.1 Experimental Setup

Datasets. We use two scene datasets RealEstate10K [37] and DL3DV [16], following the recent literature [3, 12, 34]. [2] We preprocess videos and create training/testing video clips in exactly the same as PixelSplat [2] for RealEstate10K and DepthSplat [34] for DL3DV. The only difference is the number of images to use from each clip for training and testing. Concretely, LVSM [12], MVSplat [3], and DepthSplat [34] used 2 images for training and testing for RealEstate10K. MVSplat and DepthSplat used 2-6 images for training and testing for DL3DV. We use 4-6 images for training and 4-8 images for testing in both datasets to handle larger scenes.

Evaluation Metrics. We evaluate the rendered image quality by PSNR, LPIPS, and SSIM. To assess computational efficiency, we report data size, rendering FPS, and rendering FLOPs, where FLOPs are theoretical numbers instead of measured ones.

Training Details. We adopt a two-stage training strategy. The first stage is to train the multi-view encoder where the neural renderer is directly connected to back-propagate gradients without the latent K-means or the condensation modules. In this case, all the tokens from the encoder are passed to the renderer through cross-attention. The second stage uses all the modules while freezing the multi-view encoder. Since K-means is not efficient for online sampling, we pre-compute the cluster assignments

[2] RealEstate10K is under Creative Commons Attribution 4.0 International License. DL3DV is under DL3DV-10K Term of use and Creative Commons Attribution-NonCommercial 4.0 International License.

offline and use them in the second stage. For experiments on the 256×256 RealEstate10K, the first and the second stages train for 90,000 steps with a batch-size 64 and 50,000 steps with a batch-size 80, respectively. For DL3DV (256×448), we finetune the RealEstate10K-pretrained model for 100,000 steps with a batch size of 24 in the first stage and 32 in the second stage. Both datasets use the same cosine learning rate scheduler with a 2500-step warmup. The peak learning rate is 4×10^{-4} for RealEstate10K and 2×10^{-4} for DL3DV, with the learning rate of the renderer scaled by 0.1 in the second stage. We use four NVIDIA RTX A100 GPUs to train our model. Training takes approximately 3 days on RealEstate10K and 5 days on DL3DV.

Baselines. We compare with three state-of-the-art methods, one from the reconstruction-free approach and the other two from the reconstruction-based approach.

• *LVSM* [12] is a state-of-the-art reconstruction-free method. While a publicly available checkpoint exists for RealEstate10K, it corresponds to a significantly larger model, featuring twice as many blocks in both the encoder and decoder, and was trained using 64 A100 GPUs with only 2 input views. To enable a fair comparison, we trained the LVSM-ED system under our setting: using 6 blocks in both the encoder and decoder, and 4 input views. Given the substantial compute requirements of LVSM (i.e., 64 A100 GPUs as reported by the authors), we chose to compare on RealEstate10K. As the LVSM code was not available during our initial experiments, we first reproduced their system independently, and later refined our implementation based on the official code after its release.

• *MVSplat* [3] and *DepthSplat* [34] are state-of-the-art feedforward splatting-based novel view synthesis systems. They achieve strong performance on benchmarks like RealEstate10K and DL3DV by leveraging explicit 3D scene representations and depth-guided 3D Gaussian splatting for rendering. We use publicly available checkpoints for MVSplat and DepthSplat (i.e., 2-view case for RealEstate10K and 2, 4, or 6-view cases for DL3DV).

4.2 Main Results

Figure 2 and Figure 4 show the main evaluation results on the RealEstate10K and DL3DV datasets. In the plots, the x-axis represents the data size of the scene representation: Storage CLiFTs for our method, decoder input tokens for LVSM, and splats for MVSplat and DepthSplat. The y-axis reports PSNR, SSIM and LPIPS.

None of the baselines support controlling the data size and require a separately trained model for each data point. Many baseline points are missing from the plots because we either use publicly available checkpoints (for MVSplat and DepthSplat) or train a new model (§4.1). In contrast, our method trains a single model per dataset and supports fine-grained control over both the storage data size and the render data size via the numbers of Storage (N_s) and Render (N_r) CLiFT tokens.

The plots show that CLiFT achieves comparable PSNR with approximately 5–7× less data size than MVSplat and DepthSplat, and about 1.8× less than LVSM, highlighting the effectiveness of our compressed tokens and overall scene representation. CLiFT also attains the highest overall PSNR with significantly lower data usage. Qualitative results in Figure 4 support these findings: our method preserves sharp appearance details that are closer to the ground truth and maintains high visual fidelity even under strong compression, with only minor loss in high-frequency content.

4.3 Ablation Studies

CLiFT Construction. To demonstrate the effectiveness of our individual components, Figure 3 compares three variants of our system: 1) The full system (blue); 2) The system without the condenser (green); and 3) The system without both the condenser and latent K-means (red), where tokens are selected randomly instead. We evaluate all variants on 92 randomly selected scenes to ensure consistent and fair comparisons, using an NVIDIA RTX A6000 GPU.

Figure 3 shows how rendering quality (PSNR, LPIPS, SSIM), rendering speed (FPS), and rendering cost (FLOPs: theoretical number instead of measured) change with different compression rates. Specifically, we fix the number of CLiFT tokens for storage (N_s) and rendering (N_r) to be the same, and vary them together. When the compression rate is low (<2×), all variants perform similarly since even randomly selected tokens provide reasonable coverage. However, at high compression rates, the gap between random selection and K-means selection becomes significant: random selection fails to capture representative tokens, while K-means selects tokens that better summarize the scene. Figure 5 visualizes the selected centroids and their associated cluster members. The clustering is performed

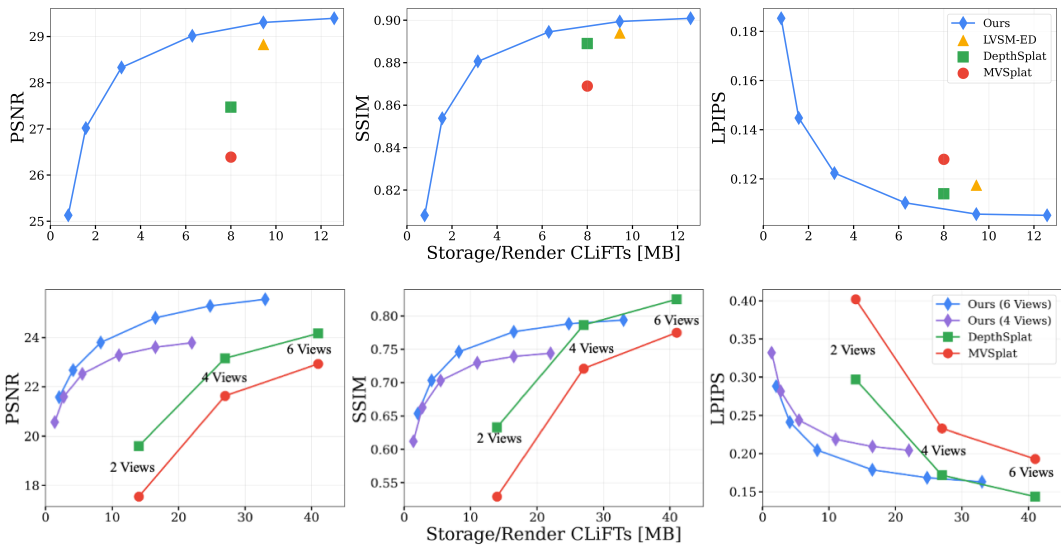

Figure 2: Main evaluation results on the RealEstate10K dataset (top) and the DL3DV dataset (bottom), comparing our approach with three baseline methods (LVSM-ED [12], DepthSplat [34], and MVSplat [3]). The x-axis is the data size of the scene representation, and the y-axis is the rendering quality (PSNR, SSIM and LPIPS). Our approach CLiFT can flexibly change the data size (i.e., number of tokens) with one trained model, achieving significant data size reduction with comparable rendering quality and the highest overall PSNR, while providing trade-offs among data size, rendering quality, and rendering speed.

Table 2: We fix the number of storage CLiFTs to represent a scene (N_s=4096), then vary the number of render CLiFTS (how many tokens to use for rendering) on-the-fly and measure rendering quality (PSNR), rendering speed (PSNR), and rendering cost (theoretical number as GFLOPs).

Metrics	Render CLiFTs				
	4096	3072	2048	1024	512
PSNR	26.72	26.71 (-0.01)	26.56 (-0.16)	25.75 (-0.97)	23.89 (-2.83)
GFLOPs	70.6	63.3 (-10%)	56.1 (-21%)	48.9 (-31%)	45.23 (-36%)
FPS	54.3	53.89 (-1%)	66.44 (+22%)	80.77 (+49%)	90.15 (+66%)

jointly across multiple views, encouraging tokens from different images to form non-redundant clusters. Furthermore, clusters tend to grow larger in texture-homogeneous regions.

Effectiveness of K-means clustering. We also evaluate the effectiveness of latent K-means clustering by comparing it against a simple patch-wise grouping baseline. The baseline divides the image into non-overlapping local patches, assigning tokens uniformly across regions, whereas latent K-means adaptively clusters tokens in latent space. As shown in Table 1, latent K-means consistently outperforms patch-wise clustering across both token budgets, with especially large gains under stronger compression (e.g., +2.24 PSNR at 256 tokens with condensation). This improvement arises because patch-wise clustering distributes capacity evenly, while latent K-means allocates more tokens to informative regions. As illustrated in Figure 11 and Figure 12, this adaptive allocation leads to better visual quality when budgets are tight.

Table 1: Ablation study comparing latent K-means clustering with a simple patch-wise grouping baseline. We report PSNR under two token budget settings (256 and 1024), with and without the condenser.

Method	256 tokens	1024 tokens
Patch-wise	22.64	27.48
Patch-wise + Condenser	22.97	27.55
K-means	24.46	28.17
K-means + Condenser	**25.21**	**28.41**

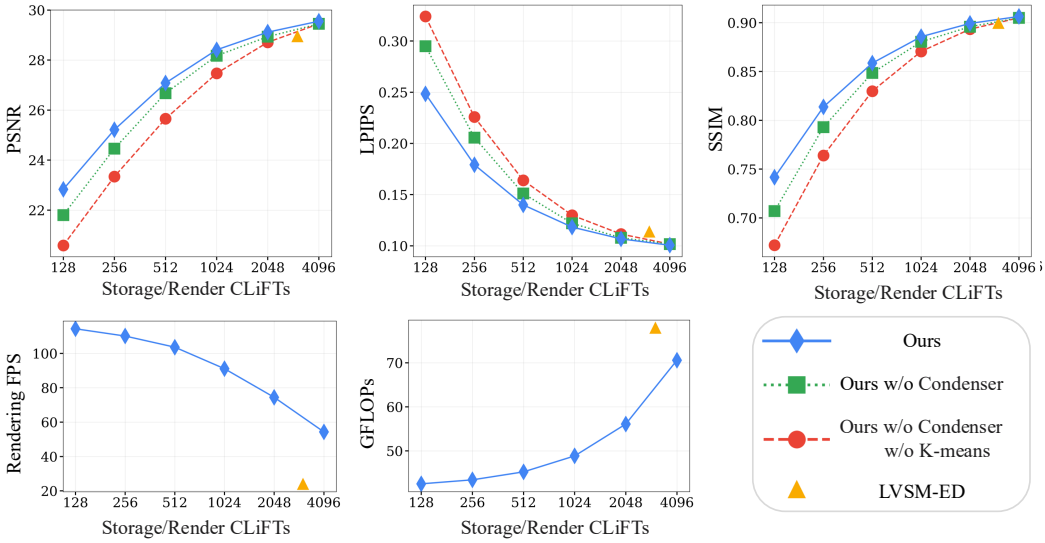

Figure 3: Ablation studies on our individual components, in particular, latent K-means and neural condensation. The plots compare three variants of our system by dropping latent K-means and neural condensation one by one from the system, while varying the data size. Specifically, the x-axis is the size of the scene representation. The y-axis is rendering quality (PSNR, LPIPS, and SSIM), rendering speed (FPS), or rendering cost (FLOPs), measured on an NVIDIA RTX A6000 GPU.

CLiFT Rendering. A CLiFT token is associated with a specific ray, encoding localized geometry and appearance. This ray-based design enables intuitive and efficient token selection heuristics at inference time (§3.3), allowing the renderer to control the trade-off between quality and speed on the fly. For example, in large scenes with multiple rooms, rendering a view within one room can be done without using tokens from unrelated areas. In contrast, methods like LVSM represent the entire scene using a fixed set of global tokens, which limits their ability to selectively render parts of the scene or adjust computation dynamically.

We validate this ability on large scenes from the RealEstate10K dataset. Specifically, we construct a large-scene test set by filtering for scenes with more than 200 frames, as higher frame counts typically indicate broader camera motion. We use 50 such scenes for evaluation, with 8 input views per scene to ensure sufficient coverage. As shown in Table 2, our method supports on-the-fly adjustment of rendering cost by varying the number of tokens used, achieving lower FLOPs and higher FPS when needed, or allocating more tokens for higher quality.

5 Conclusions and Future Challenges

We introduced CLiFT, a compressive light-field token representation paired with a compute-adaptive transformer renderer. Experiments on RealEstate10K and DL3DV demonstrate that CLiFT achieves higher data reduction rate with comparable rendering quality and the highest overall rendering score, while capable of controlling the trade-offs between data size, rendering quality, and rendering speed. Our current system exhibits two typical failure modes. The first occurs when camera motions deviate significantly from the training distribution. As shown on the left of Figure 6, RealEstate10K training data primarily consists of smooth translations with minor rotations, making it difficult to generalize to more complex motions. The second failure mode arises in large scenes where target views are not adequately covered by the input views, resulting in blurry renderings. This is illustrated on the right of Figure 6 with examples from DL3DV. A promising future direction is to extend our framework by incorporating generative priors to improve rendering quality in unseen or occluded areas.

Broader Impacts. On the positive side, our method has potential applications in digital media content consumption, enabling more efficient and flexible rendering for immersive experiences. However, like generative models, it raises concerns around misuse—particularly in the creation of deep-fakes. We emphasize the importance of responsible deployment and clear content provenance.

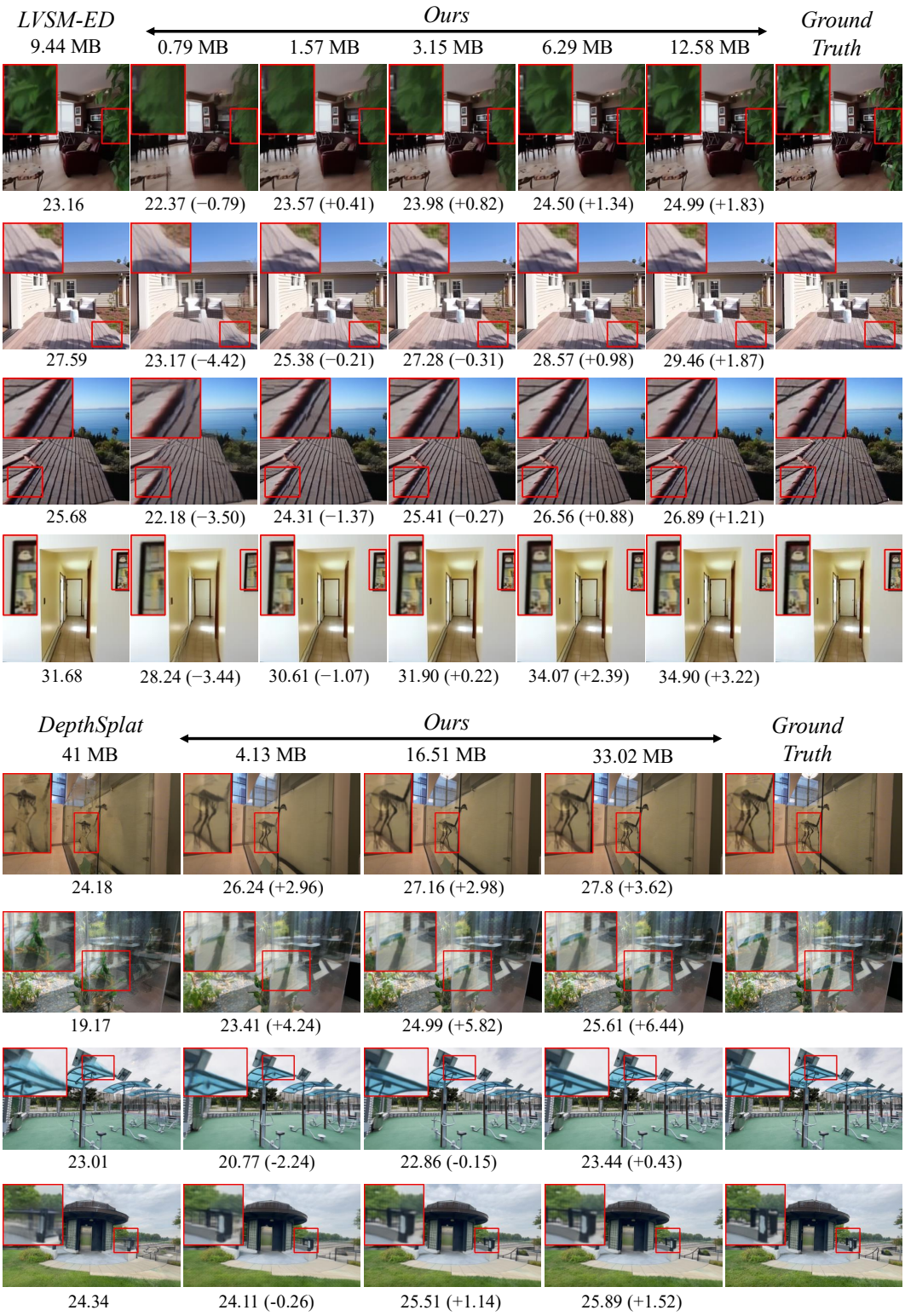

Figure 4: Qualitative rendering results of the baselines and ours with different data size (i.e., the number of CLiFT tokens for ours). **Top**: Ours vs. LVSM [12] on RealEstate10K. **Bottom**: Ours vs. DepthSplat [34] on DL3DV. The PSNR value is recorded under each rendering.

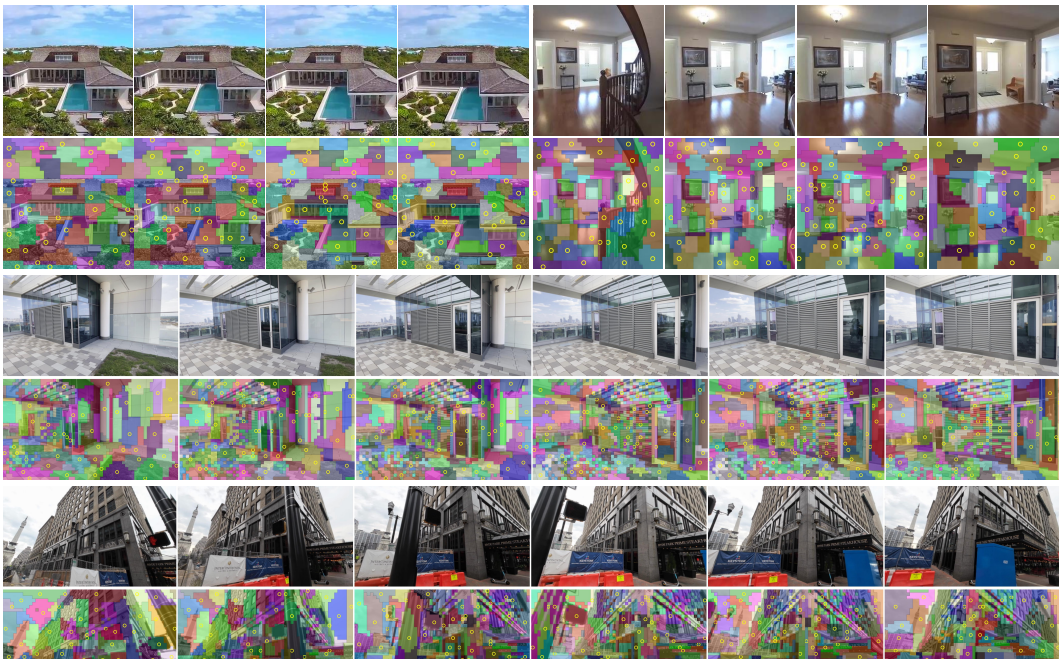

Figure 5: Visualization of the latent K-means clustering, where K=N_s=128. Each color represents a cluster, and the yellow ring indicates the centroid token. Note that clustering is performed across multiple views, so a single cluster can span multiple images. As a result, some clusters may not have a visible centroid in a given image.

Figure 6: Typical failure cases are (Left) Camera motions deviating too much from the training distributions in RealEstate10K; and (Right) Target views not covered by the input images in DL3DV. The top row shows our results, and the bottom row shows the ground truth.

Acknowledgments

We thank Haian Jin for helpful discussions on reproducing LVSM and training on the DL3DV dataset. We thank Jiaqi Tan for assistance with the project page and demo design. This research is partially supported by NSERC Discovery Grants, NSERC Alliance Grants, and John R. Evans Leaders Fund (JELF). We thank the Digital Research Alliance of Canada and BC DRI Group for providing computational resources.

References

[1] Emmanuel J Candès, Justin Romberg, and Terence Tao. 2006. Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. *IEEE Transactions on Information Theory* (2006).

[2] David Charatan, Sizhe Lester Li, Andrea Tagliasacchi, and Vincent Sitzmann. 2024. pixelsplat: 3d gaussian splats from image pairs for scalable generalizable 3d reconstruction. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 19457–19467.

[3] Yuedong Chen, Haofei Xu, Chuanxia Zheng, Bohan Zhuang, Marc Pollefeys, Andreas Geiger, Tat-Jen Cham, and Jianfei Cai. 2024. Mvsplat: Efficient 3d gaussian splatting from sparse multi-view images. In *European Conference on Computer Vision*. Springer, 370–386.

[4] David L Donoho. 2006. Compressed sensing. *IEEE Transactions on Information Theory* (2006).

[5] Yilun Du, Cameron Smith, Ayush Tewari, and Vincent Sitzmann. 2023. Learning to render novel views from wide-baseline stereo pairs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 4970–4980.

[6] Marco F Duarte, Mark A Davenport, D Takhar, Jason N Laska, Ting Sun, Keith F Kelly, and Richard G Baraniuk. 2008. Single-pixel imaging via compressive sampling. *IEEE Signal Processing Magazine* (2008).

[7] Lue Fan, Hao Zhang, Qitai Wang, Hongsheng Li, and Zhaoxiang Zhang. 2024. FreeSim: Toward Free-viewpoint Camera Simulation in Driving Scenes. *arXiv preprint arXiv:2412.03566* (2024).

[8] Yasutaka Furukawa and Jean Ponce. 2009. Accurate, dense, and robust multiview stereopsis. *IEEE transactions on pattern analysis and machine intelligence* (2009).

[9] Stephan J Garbin, Marek Kowalski, Matthew Johnson, Jamie Shotton, and Julien Valentin. 2021. Fastnerf: High-fidelity neural rendering at 200fps. In *Proceedings of the IEEE/CVF international conference on computer vision*. 14346–14355.

[10] Steven J Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F Cohen. 1996. The lumigraph. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*. 43–54.

[11] Jiatao Gu, Alex Trevithick, Kai-En Lin, Joshua M Susskind, Christian Theobalt, Lingjie Liu, and Ravi Ramamoorthi. 2023. Nerfdiff: Single-image view synthesis with nerf-guided distillation from 3d-aware diffusion. In *International Conference on Machine Learning*. PMLR, 11808–11826.

[12] Haian Jin, Hanwen Jiang, Hao Tan, Kai Zhang, Sai Bi, Tianyuan Zhang, Fujun Luan, Noah Snavely, and Zexiang Xu. 2024. Lvsm: A large view synthesis model with minimal 3d inductive bias. *arXiv preprint arXiv:2410.17242* (2024).

[13] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 2023. 3d gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.* (2023).

[14] Xin Kong, Shikun Liu, Xiaoyang Lyu, Marwan Taher, Xiaojuan Qi, and Andrew J Davison. 2024. Eschernet: A generative model for scalable view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 9503–9513.

[15] Marc Levoy and Pat Hanrahan. 1996. Light field rendering. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*. 31–42.

[16] Lu Ling, Yichen Sheng, Zhi Tu, Wentian Zhao, Cheng Xin, Kun Wan, Lantao Yu, Qianyu Guo, Zixun Yu, Yawen Lu, et al. 2024. Dl3dv-10k: A large-scale scene dataset for deep learning-based 3d vision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 22160–22169.

[17] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. 2020. Neural sparse voxel fields. *Advances in Neural Information Processing Systems* 33 (2020), 15651–15663.

[18] Yuan Liu, Cheng Lin, Zijiao Zeng, Xiaoxiao Long, Lingjie Liu, Taku Komura, and Wenping Wang. 2023. Syncdreamer: Generating multiview-consistent images from a single-view image. *arXiv preprint arXiv:2309.03453* (2023).

[19] Xiaoxiao Long, Yuan-Chen Guo, Cheng Lin, Yuan Liu, Zhiyang Dou, Lingjie Liu, Yuexin Ma, Song-Hai Zhang, Marc Habermann, Christian Theobalt, et al. 2024. Wonder3d: Single image to 3d using cross-domain diffusion. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 9970–9980.

[20] Kshitij Marwah, Gordon Wetzstein, Yosuke Bando, and Ramesh Raskar. 2013. Compressive light field photography using overcomplete dictionaries and optimized projections. In *ACM Transactions on Graphics (TOG)*. ACM.

[21] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. 2021. Nerf: Representing scenes as neural radiance fields for view synthesis. *Commun. ACM* (2021).

[22] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. 2022. Instant neural graphics primitives with a multiresolution hash encoding. *ACM transactions on graphics (TOG)* 41, 4 (2022), 1–15.

[23] Ren Ng, Marc Levoy, Mathieu Brédif, Gene Duval, Mark Horowitz, and Pat Hanrahan. 2005. *Light field photography with a hand-held plenoptic camera*. Ph. D. Dissertation. Stanford university.

[24] Christian Reiser, Songyou Peng, Yiyi Liao, and Andreas Geiger. 2021. Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps. In *Proceedings of the IEEE/CVF international conference on computer vision*. 14335–14345.

[25] Mehdi SM Sajjadi, Henning Meyer, Etienne Pot, Urs Bergmann, Klaus Greff, Noha Radwan, Suhani Vora, Mario Lučić, Daniel Duckworth, Alexey Dosovitskiy, et al. 2022. Scene representation transformer: Geometry-free novel view synthesis through set-latent scene representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

[26] Vincent Sitzmann, Semon Rezchikov, Bill Freeman, Josh Tenenbaum, and Fredo Durand. 2021. Light field networks: Neural scene representations with single-evaluation rendering. *Advances in Neural Information Processing Systems* 34 (2021), 19313–19325.

[27] Noah Snavely, Steven M Seitz, and Richard Szeliski. 2006. Photo tourism: exploring photo collections in 3D. In *ACM Transactions on Graphics (TOG)*. ACM.

[28] Mohammed Suhail, Carlos Esteves, Leonid Sigal, and Ameesh Makadia. 2022. Generalizable patch-based neural rendering. In *European Conference on Computer Vision*. Springer, 156–174.

[29] Mohammed Suhail, Carlos Esteves, Leonid Sigal, and Ameesh Makadia. 2022. Light field neural rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 8269–8279.

[30] Shitao Tang, Jiacheng Chen, Dilin Wang, Chengzhou Tang, Fuyang Zhang, Yuchen Fan, Vikas Chandra, Yasutaka Furukawa, and Rakesh Ranjan. 2024. Mvdiffusion++: A dense high-resolution multi-view diffusion model for single or sparse-view 3d object reconstruction. In *European Conference on Computer Vision*. Springer.

[31] Qitai Wang, Lue Fan, Yuqi Wang, Yuntao Chen, and Zhaoxiang Zhang. 2024. Freevs: Generative view synthesis on free driving trajectory. *arXiv preprint arXiv:2410.18079* (2024).

[32] Bennett Wilburn, Neel Joshi, Vaibhav Vaish, Eino-Ville Talvala, Emilio Antunez, Adam Barth, Andrew Adams, Mark Horowitz, and Marc Levoy. 2005. High performance imaging using large camera arrays. In *ACM Transactions on Graphics (TOG)*.

[33] Rundi Wu, Ben Mildenhall, Philipp Henzler, Keunhong Park, Ruiqi Gao, Daniel Watson, Pratul P Srinivasan, Dor Verbin, Jonathan T Barron, Ben Poole, et al. 2024. Reconfusion: 3d reconstruction with diffusion priors. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 21551–21561.

[34] Haofei Xu, Songyou Peng, Fangjinhua Wang, Hermann Blum, Daniel Barath, Andreas Geiger, and Marc Pollefeys. 2024. Depthsplat: Connecting gaussian splatting and depth. *arXiv preprint arXiv:2410.13862* (2024).

[35] Alex Yu, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, and Angjoo Kanazawa. 2021. Plenoctrees for real-time rendering of neural radiance fields. In *Proceedings of the IEEE/CVF international conference on computer vision*. 5752–5761.

[36] Shiran Yuan and Hao Zhao. 2024. Slimmerf: Slimmable radiance fields. In *2024 International Conference on 3D Vision (3DV)*. IEEE, 64–74.

[37] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. 2018. Stereo magnification: Learning view synthesis using multiplane images. *arXiv preprint arXiv:1805.09817* (2018).

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Our main contributions are summarized in the abstract and introduction.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss the limitation in the section 5

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

Justification: The paper does not include theoretical results.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: All necessary details to reproduce the main experimental results are provided in section 4

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [No]

Justification: We will release the code and model in the future.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: We include the training and test details in section 4.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: We follow the official benchmark to conduct the evaluation to be consistent with prior works.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We provide the computer resource information in the section 4

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research conducted in this paper completely conforms with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We discuss broader impacts in section 5.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: he paper does not have such risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We cite every asset we use and we follow the license.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: The paper does not release new assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification: Our method does not involve LLMs in any part of the core methodology, model design, or experimentation.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.

Appendix

The appendix provides:

- ◇ §A: Additional more rendering results and intermediate visualizations (i.e., clusteirng and token-selection results).
- ◇ §B: Implementation details of the latent K-means algorithm (during CLiFT construction) and the token selection algorithm (during CLiFT rendering).

Please check our project page at `https://clift-nvs.github.io`, which shows rendering results at different compression rates and compares with the baseline methods.

A Additional Rendered Images

Figure 4 of the main paper presents rendering results under varying data size constraints. We provide additional qualitative examples on RealEstate10K (see Figure 7 and Figure 8) and DL3DV (Figure 9 and Figure 10). These examples cover a finer range of compression rates, extending up to 32×, and highlight the robustness of our method across varying levels of compression. We also show more K-means clustering results in Figure 11 and Figure 12.

B Additional implementation details

B.1 Details of K-means algorithm

During training, we precompute cluster assignments after the multi-view encoder training and before the condensation training, using *faiss.Kmeans* which supports GPU acceleration. At test time, we use *sklearn.cluster.KMeans* for better accuracy.

B.2 Details of Token Selection

Rendering a specific region within a large scene does not require all tokens. Using only the tokens essential to the target view improves FPS and reduces FLOPs, with minimal impact on PSNR. Algorithm 1 is our token selection algorithm.

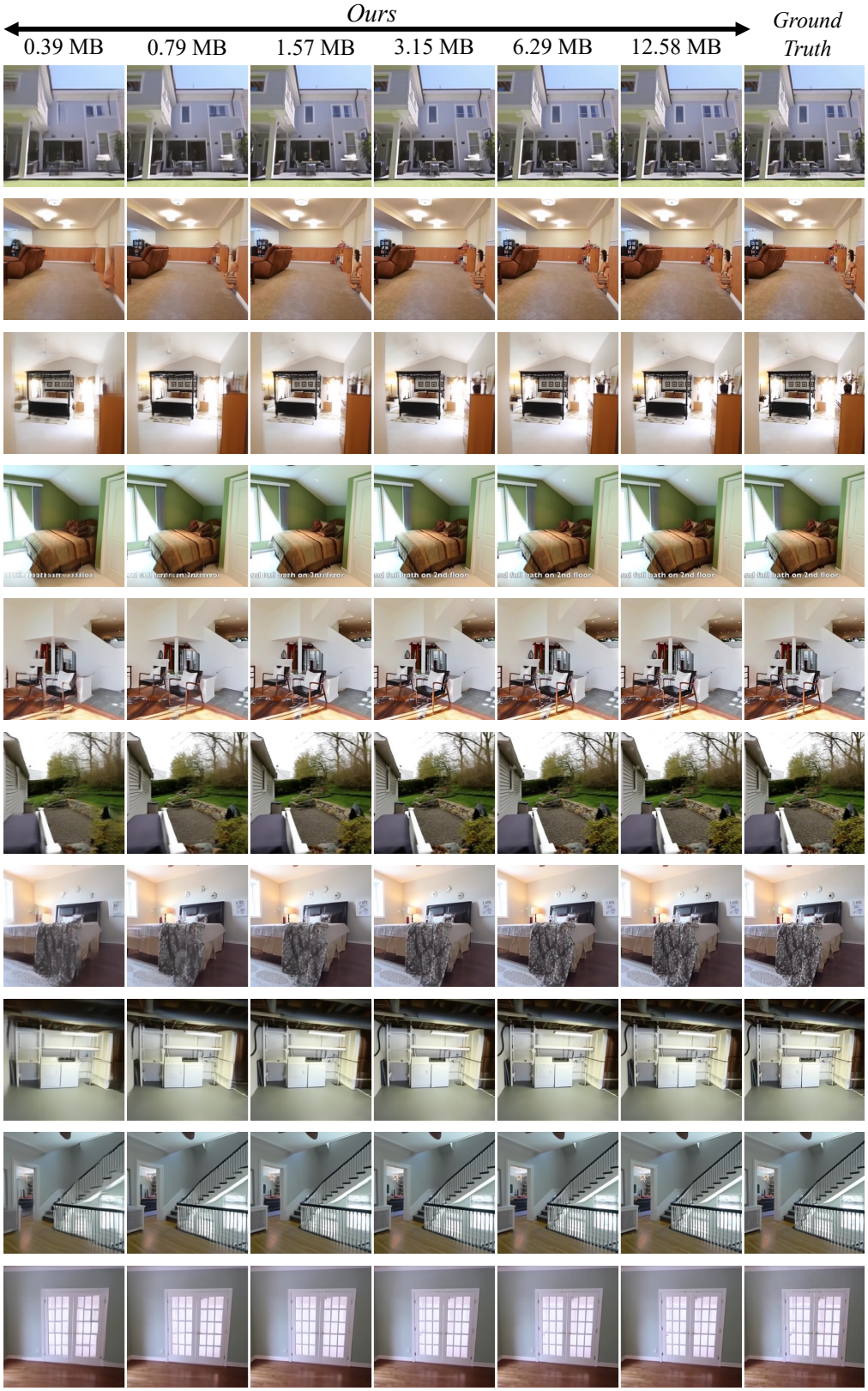

Figure 7: Additional qualitative results on the RealEstate10K dataset.

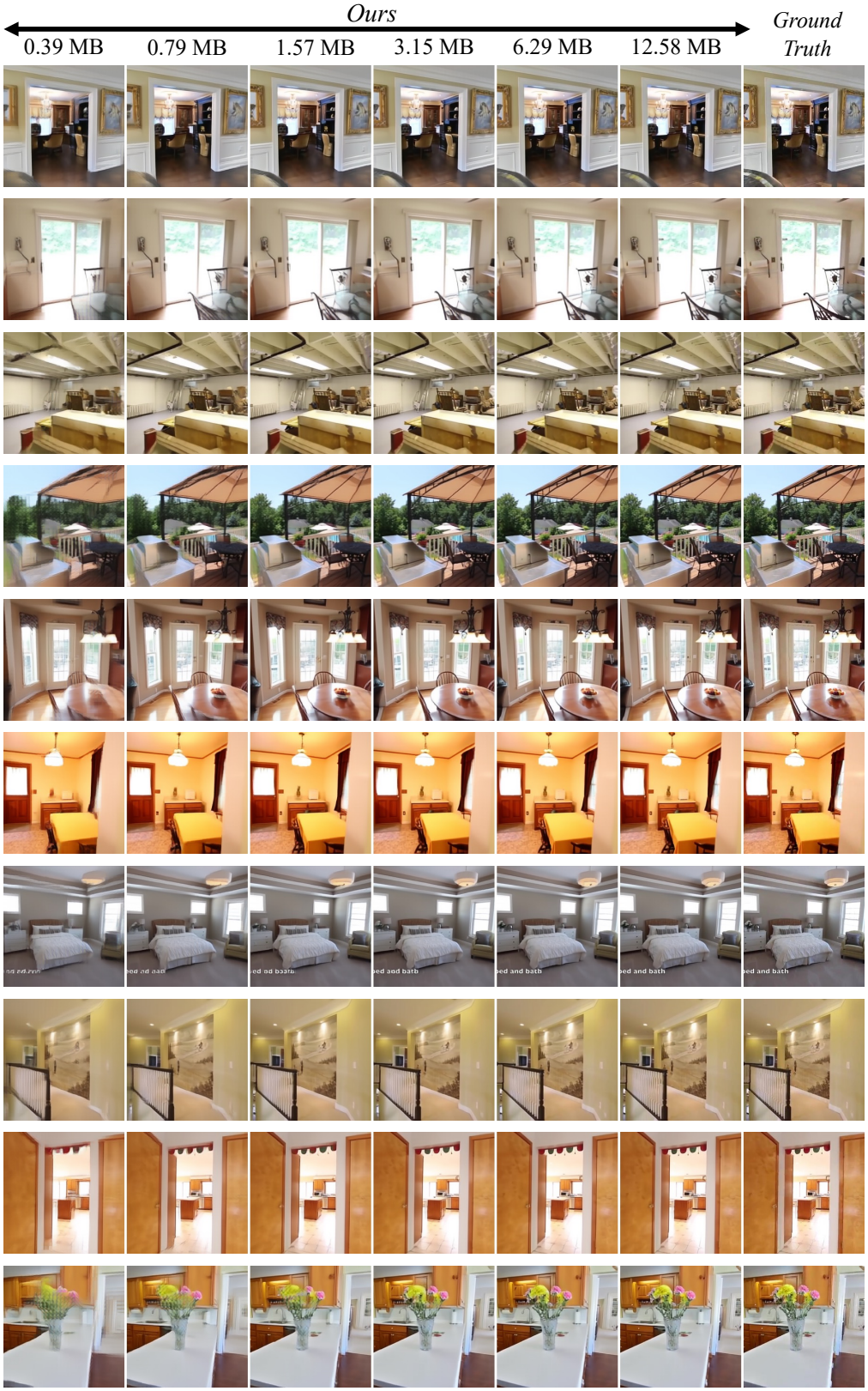

Figure 8: Additional qualitative results on the RealEstate10K dataset.

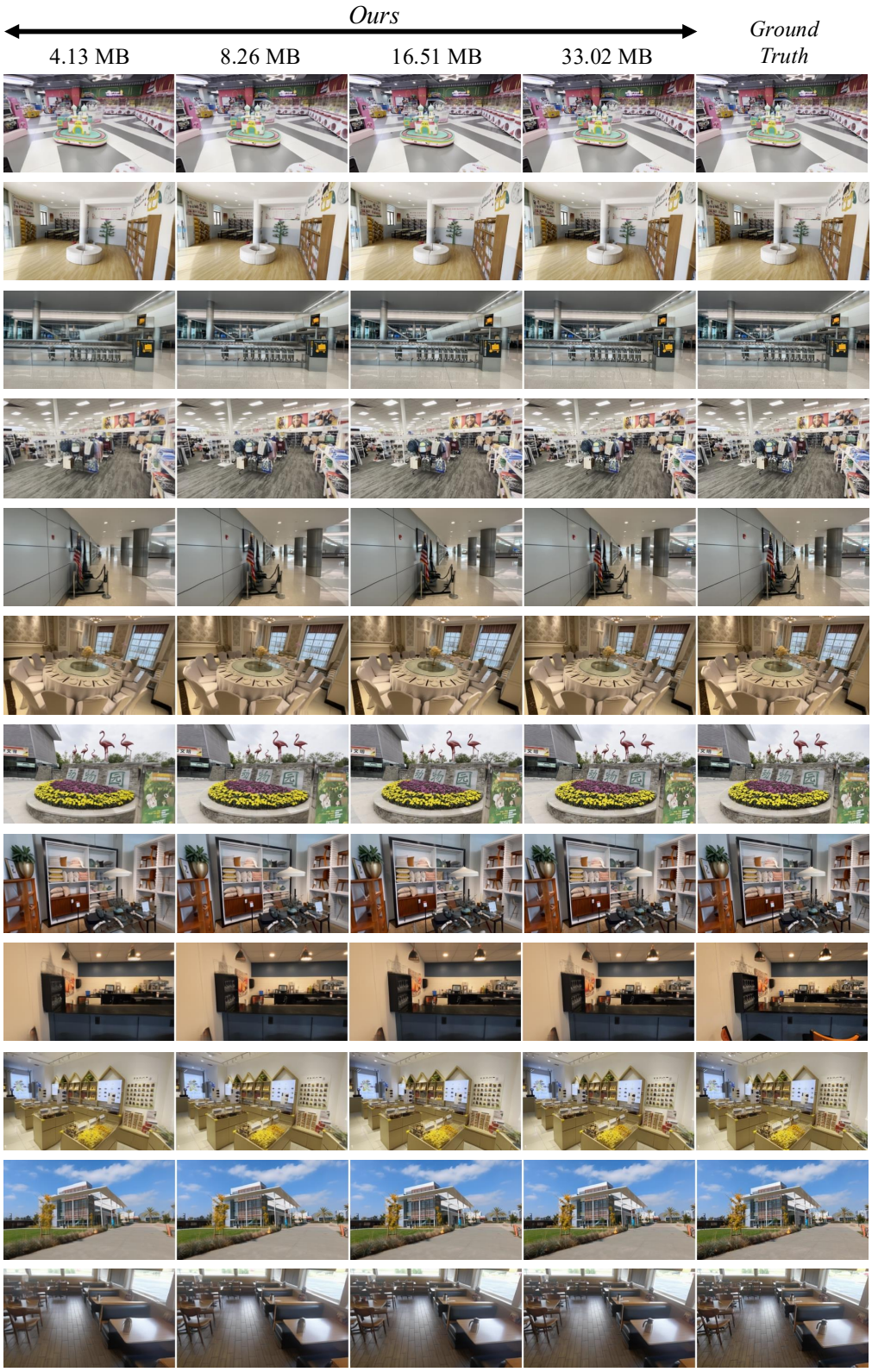

Figure 9: Additional qualitative results on the DL3DV dataset.

Figure 10: Additional qualitative results on the DL3DV dataset.

Algorithm 1 Token Selection Algorithm

Definitions:
  θ: Ray-angle distance between patches
  δ: Camera center distance, $\delta = \|o_t - o_k\|$
  m_k: Last frame token mask
  P: Number of rays per patch (e.g., $P = 16 \times 16$)
  N: number of condition images
  $w_{\text{angle}} = 1.0$, $w_{\text{dist}} = 0.02$, $w_{\text{mask}} = -0.03$
  momentum factor $\eta = 0.5$
Input:
  Target view ray (o_t, d_t) with $o_t \in \mathbb{R}^3$, $d_t \in \mathbb{R}^{P\times 3}$;
  Condition view rays $\{(o^k, d^k)\}_{k=1}^N$ with $o^k \in \mathbb{R}^3$, $d^k \in \mathbb{R}^{P\times 3}$;
  Last frame token mask $\{m^k\}_{k=1}^N$
  Total number of tokens to select T
Output: Selected token indices $\mathcal{I}$
Downsample target and condition view rays into 16×16 patches
Expand each target patch to a 24×24 region, including out-of-image rays, to incorporate edge-adjacent context and avoid boundary under-coverage
Set number condition rays per target patch $n \leftarrow T \,//\, (24 \times 24)$
Initialize selected patches $\mathcal{I}_{\text{patch}} \leftarrow \emptyset$
// Patch-wise selection
for each patch P_i in expanded target patches **do**
  for each condition view k from 1 to N **do**
    for each condition patch $\bar{P}^{k,j}$ in condition view **do**
      Compute ray-angle distance θ between rays in P_i and $\bar{P}^{k,j}$
      Compute camera center distance between target and condition view $\delta = \|o_t - o^k\|$
      Retrieve last frame mask m^k
      Compute frame distance: $D_k^{i,j} = w_{\text{angle}} \cdot \theta + w_{\text{dist}} \cdot \delta + w_{\text{mask}} \cdot m^k$
      Apply momentum: $D_i^{k,j} \leftarrow (1-\eta) \cdot D_i^{k,j} + \eta \cdot \hat{D}_i^{k,j}$
      Store previous step objective: $\hat{D}_i^{k,j} \leftarrow D_i^{k,j}$
    end for
  end for
  Let $\mathcal{D}_i = \{D_i^{k,j} \mid \forall k, j\}$
  $\mathcal{I}_{\text{local}}^i \leftarrow \text{Top}_n\big(\text{argsort}_{\text{asc}}(\mathcal{D}_i)\big)$
  $\mathcal{I}_{\text{patch}} \leftarrow \mathcal{I}_{\text{patch}} \cup \mathcal{I}_{\text{local}}^i$
end for
Compute unique set: $\mathcal{I}_{\text{unipatch}} \leftarrow \text{unique}(\mathcal{I}_{\text{patch}})$
Let $T_{\text{rest}} \leftarrow T - |\mathcal{I}_{\text{unipatch}}|$
// Global fallback selection
Initialize set of fallback distances: $\mathcal{D}_{\text{global}} \leftarrow \emptyset$
for each condition view k from 1 to N **do**
  for each condition patch j in view k **do**
    Compute $\tilde{D}^{k,j} = \min_i D_i^{k,j}$ // best match to any target patch
    $\mathcal{D}_{\text{global}} \leftarrow \mathcal{D}_{\text{global}} \cup \{\tilde{D}^{k,j}\}$
  end for
end for
$\mathcal{I}_{\text{global}} \leftarrow \text{Top}_n\left(\text{argsort}_{\text{asc}}(\mathcal{D}_{\text{global}})\right)$
return $\mathcal{I} = \mathcal{I}_{\text{patch}} \cup \mathcal{I}_{\text{global}}$

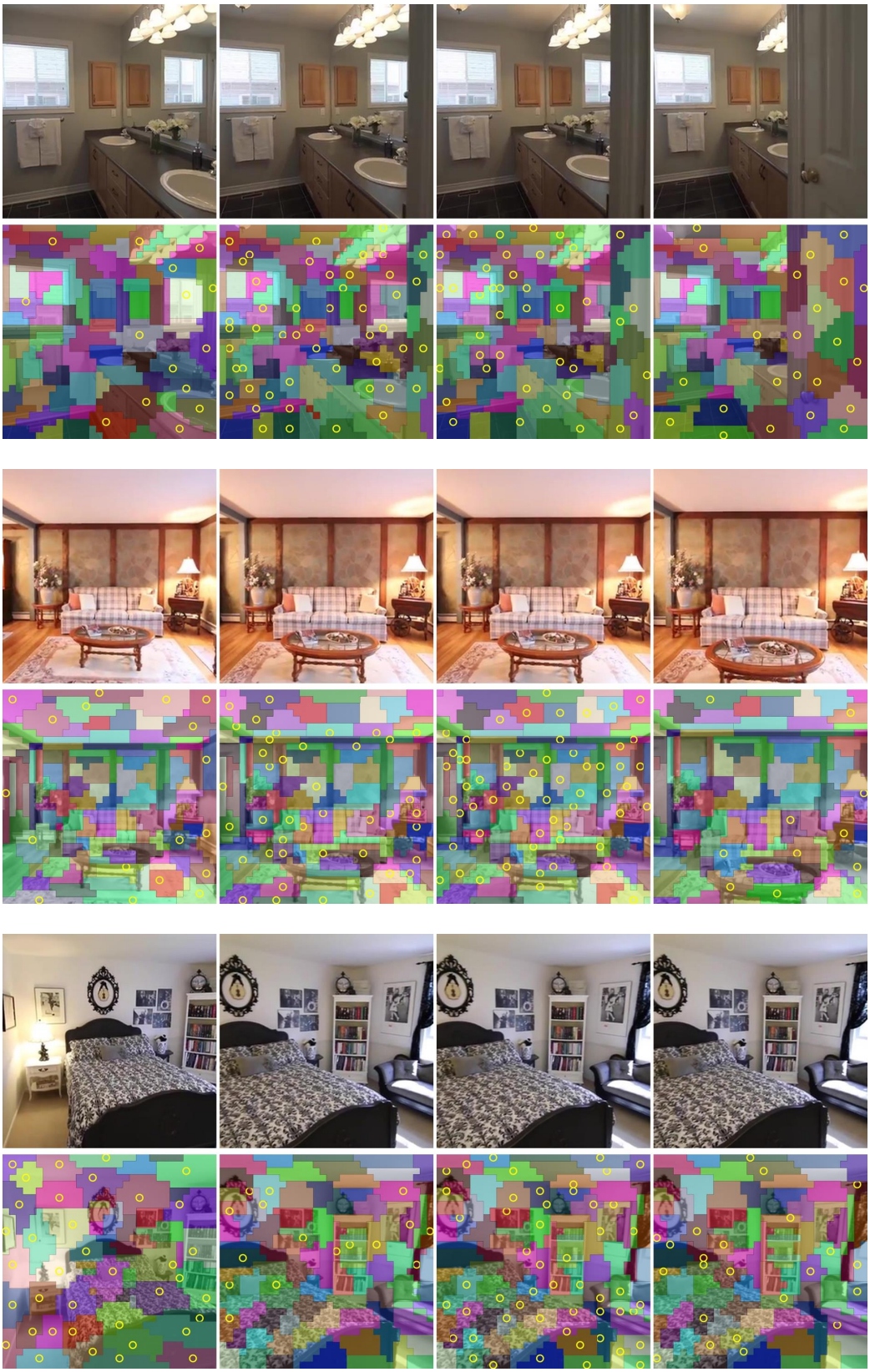

Figure 11: Additional visualization of the latent K-means algorithm for the RealEstate10k dataset.

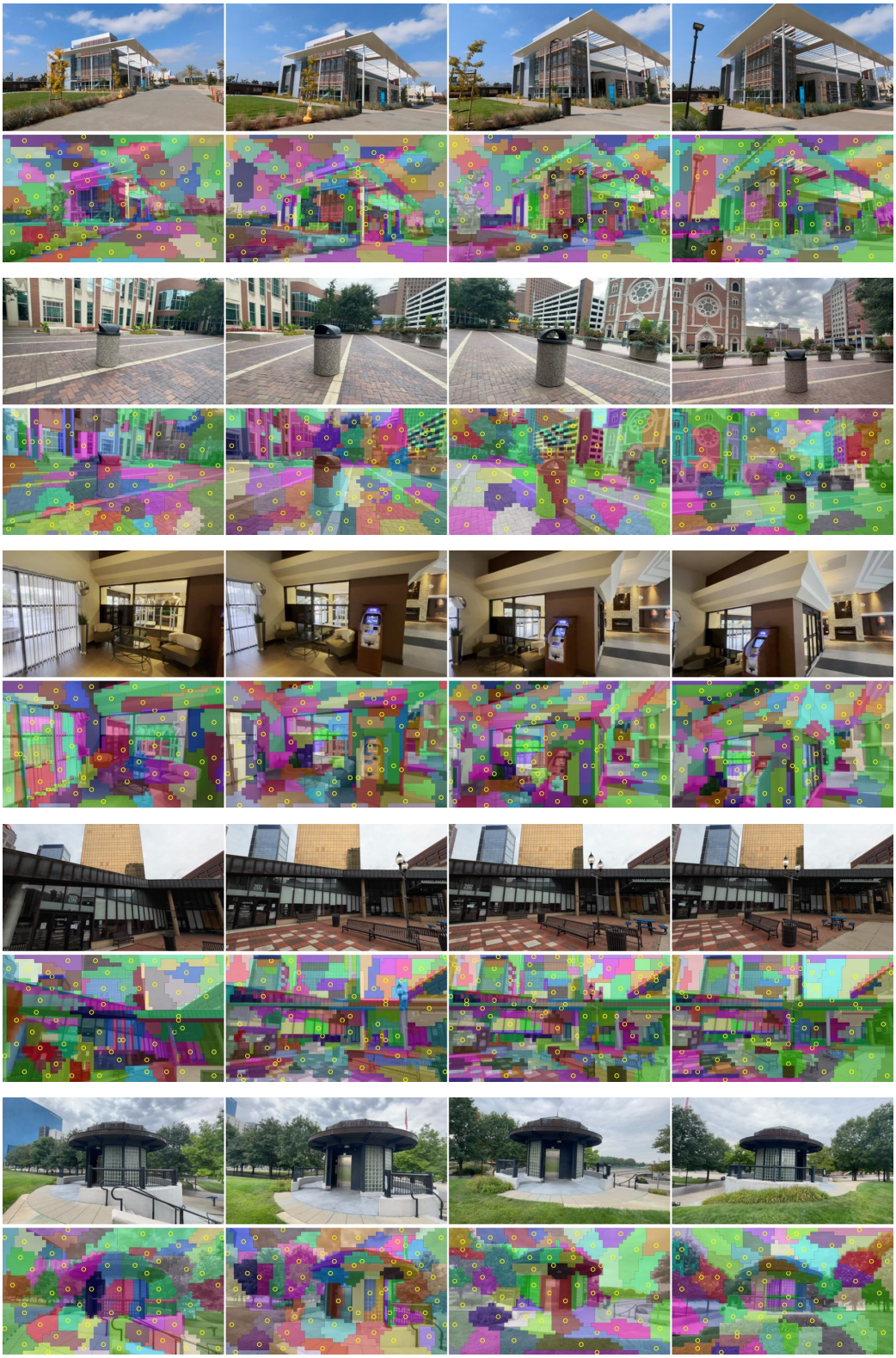

Figure 12: Additional visualization of the latent K-means algorithm for the DL3DV dataset.

Figure 13: Additional visualization of the latent K-means algorithm with different values of K, which is 512, 1024, 2048, and 3072 from the top.