
Supplementary Document:

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

Anonymous Author(s)

Affiliation

Address

email

1 The supplementary document provides:

- 2 ◇ §A: Additional experimental results in the same format as figures/tables in the main paper, including
3 the full metric evaluations on the DL3DV dataset, more rendering samples, and more intermediate
4 visualizations (i.e., clustering and token-selection results).
- 5 ◇ §B: Implementation details of the latent K-means algorithm (during CLiFT construction) and the
6 token selection algorithm (during CLiFT rendering).

7 Please also refer to the accompanying video [supp.mp4](#), which shows rendered results at different
8 compression rates and compares with the baseline methods.

9 A Additional Experiment Results

10 A.1 Additional quantitative results

11 Figure 2 of the main paper reports PSNR as the primary rendering metric and shows that our method
12 consistently outperforms baselines, achieving the highest overall PSNR. Figure 1 and Figure 2
13 show SSIM and LPIPS results for RealEstate10K and DL3DV. Our method consistently outperforms
14 baselines under the same data size, while at very large data-size setting, DepthSplat [3] yields higher
15 scores on DL3DV.

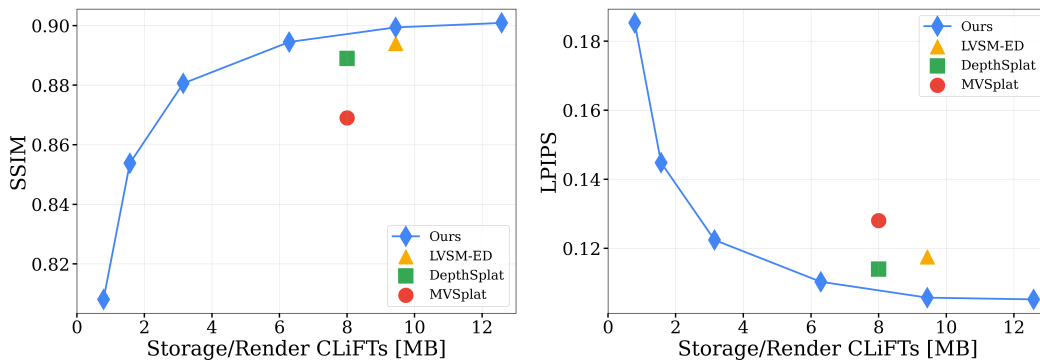


Figure 1: Evaluation results (SSIM and LPIPS) on the RealEstate10K dataset, comparing our approach with three baseline methods (LVSM-ED [2], DepthSplat [3], and MVSplat [1]). The x-axis is the data size of the scene representation.

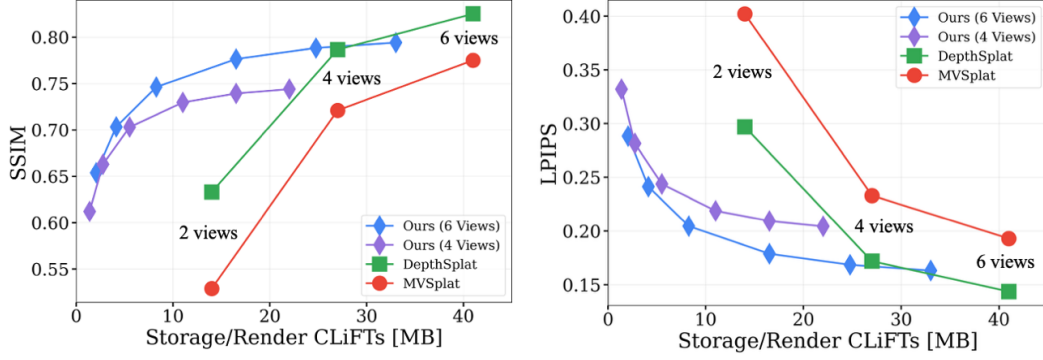


Figure 2: Evaluation results (SSIM and LPIPS) on the DL3DV dataset, comparing our approach with two baseline methods (DepthSplat [3] and MVSplat [1]). The x-axis is the data size of the scene representation.

16 A.2 Additional rendered images

17 Figure 4 of the main paper presents rendering results under varying data size constraints. We provide
 18 additional qualitative examples on RealEstate10K (see Figure 3 and Figure 4) and DL3DV (Figure 5
 19 and Figure 6). These examples cover a finer range of compression rates, extending up to 32 \times , and
 20 highlight the robustness of our method across varying levels of compression.

21 B Additional implementation details

22 B.1 Details of K-means algorithm

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

26 B.2 Details of Token Selection

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

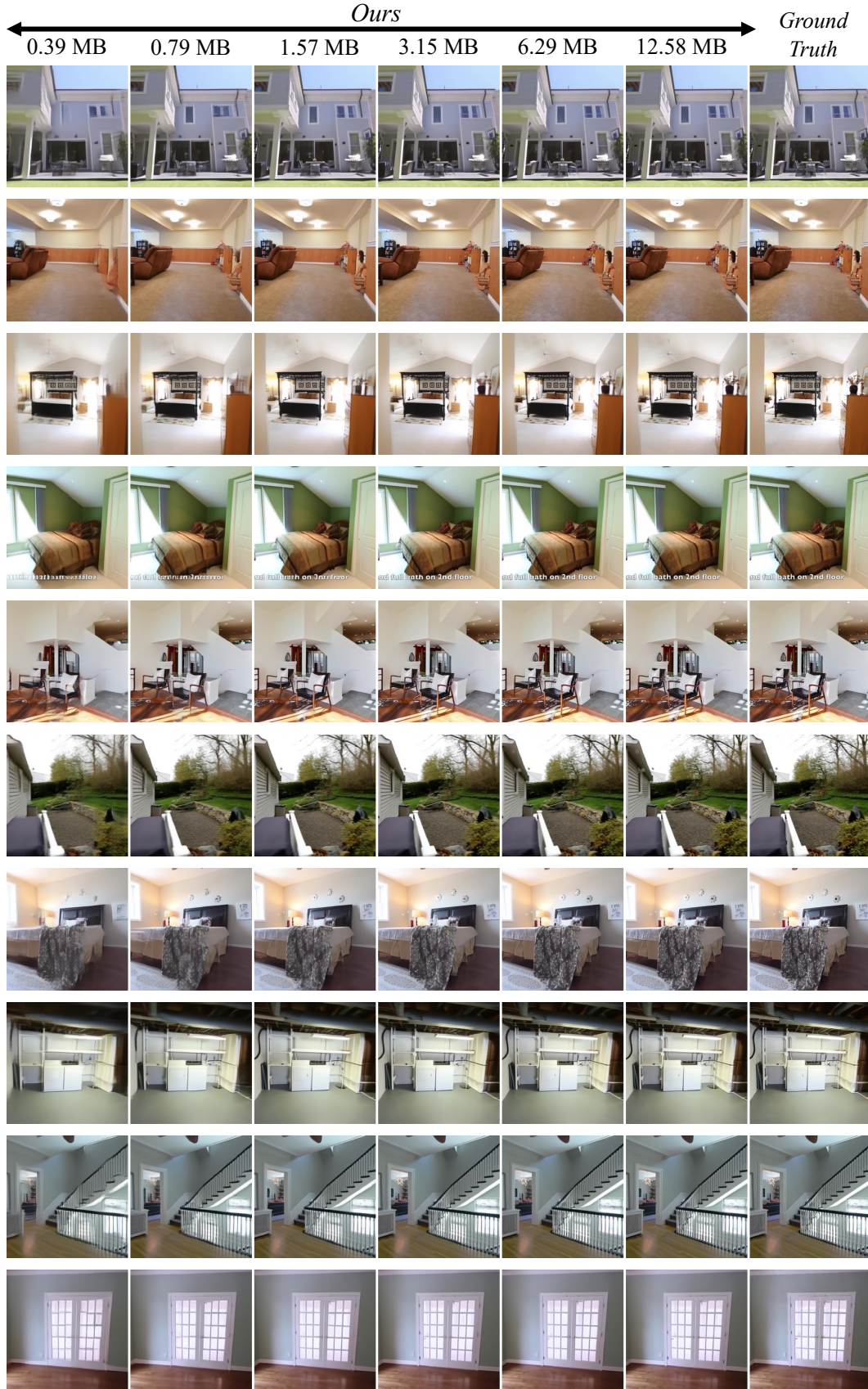


Figure 3: Additional qualitative results on the RealEstate10K dataset.

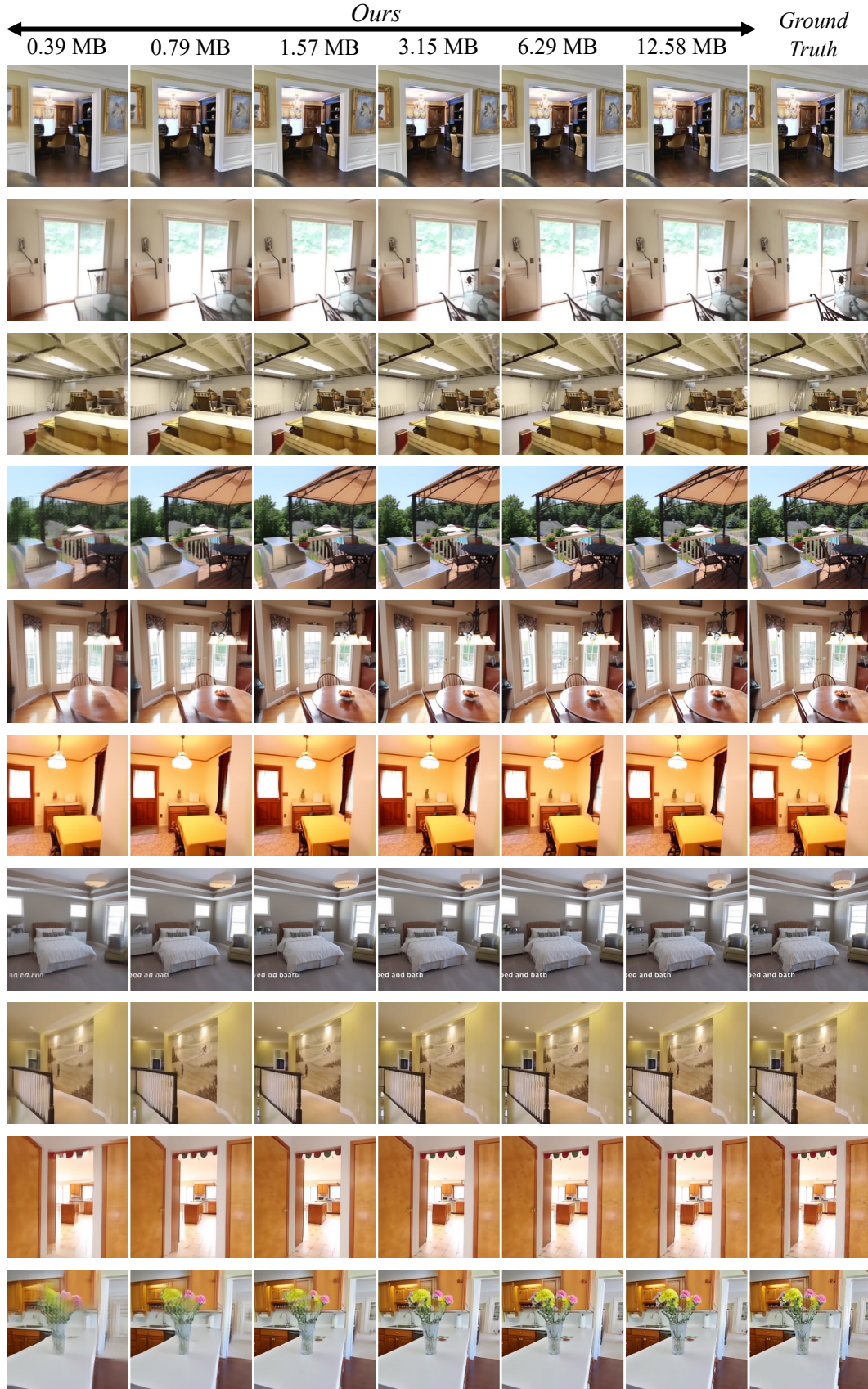


Figure 4: Additional qualitative results on the RealEstate10K dataset.

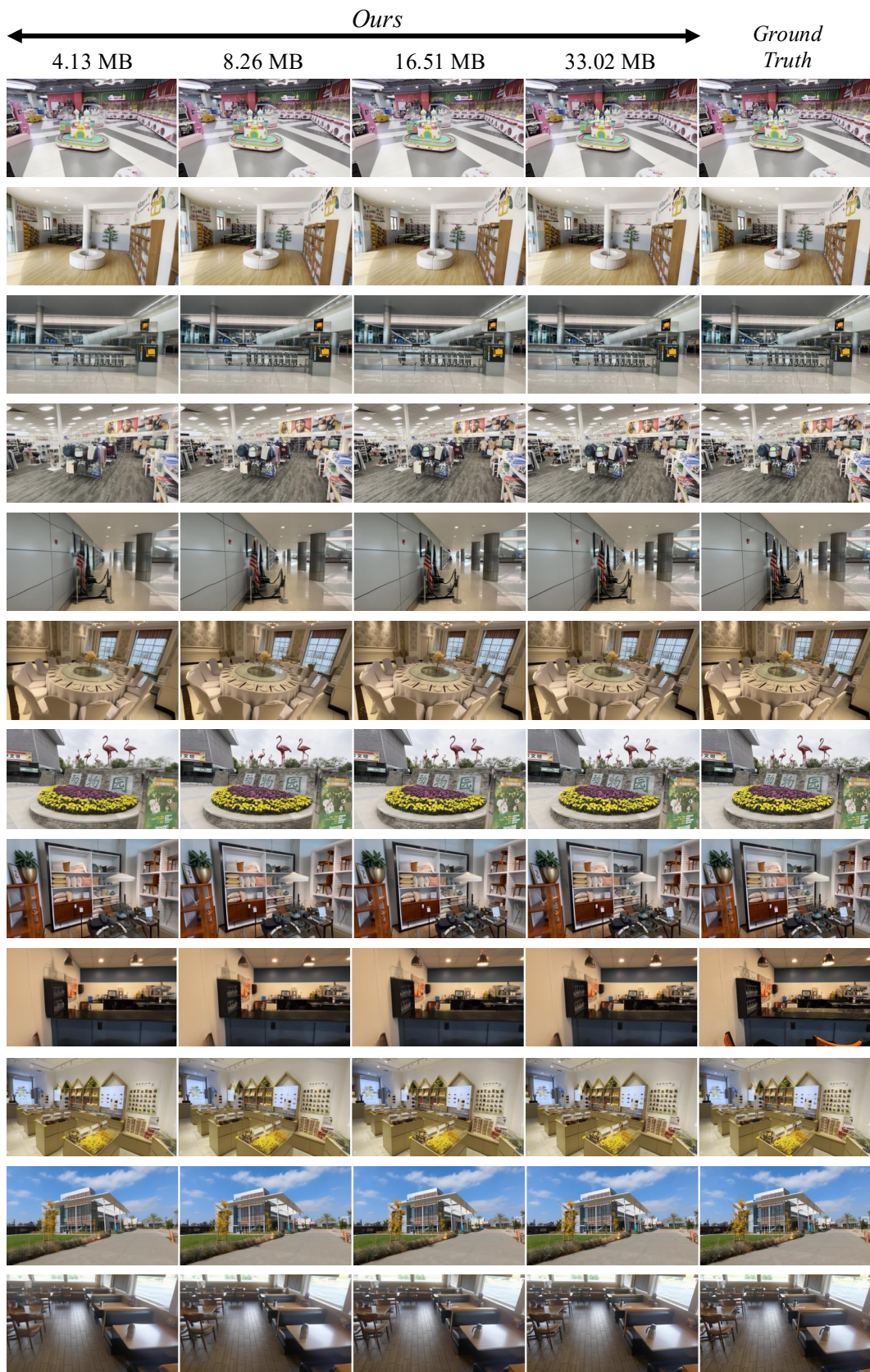


Figure 5: Additional qualitative results on the DL3DV dataset.



Figure 6: Additional qualitative results on the DL3DV dataset.

Algorithm 1 Token Selection Algorithm

```
1: Definitions:
2:    $\theta$ : Ray-angle distance between patches
3:    $\delta$ : Camera center distance,  $\delta = \|o_t - o_k\|$ 
4:    $m_k$ : Last frame token mask
5:    $P$ : Number of rays per patch (e.g.,  $P = 16 \times 16$ )
6:    $N$ : number of condition images
7:    $w_{\text{angle}} = 1.0, w_{\text{dist}} = 0.02, w_{\text{mask}} = -0.03$ 
8:   momentum factor  $\eta = 0.5$ 
9: Input:
10:  Target view ray  $(o_t, d_t)$  with  $o_t \in \mathbb{R}^3, d_t \in \mathbb{R}^{P \times 3}$ ;
11:  Condition view rays  $\{(o^k, d^k)\}_{k=1}^N$  with  $o^k \in \mathbb{R}^3, d^k \in \mathbb{R}^{P \times 3}$ ;
12:  Last frame token mask  $\{m^k\}_{k=1}^N$ 
13:  Total number of tokens to select  $T$ 
14: Output: Selected token indices  $\mathcal{I}$ 
15: Downsample target and condition view rays into  $16 \times 16$  patches
16: Expand each target patch to a  $24 \times 24$  region, including out-of-image rays, to incorporate
    edge-adjacent context and avoid boundary under-coverage
17: Set number condition rays per target patch  $n \leftarrow T // (24 \times 24)$ 
18: Initialize selected patches  $\mathcal{I}_{\text{patch}} \leftarrow \emptyset$ 
19: // Patch-wise selection
20: for each patch  $P_i$  in expanded target patches do
21:   for each condition view  $k$  from 1 to  $N$  do
22:     for each condition patch  $\bar{P}^{k,j}$  in condition view do
23:       Compute ray-angle distance  $\theta$  between rays in  $P_i$  and  $\bar{P}^{k,j}$ 
24:       Compute camera center distance between target and condition view  $\delta = \|o_t - o^k\|$ 
25:       Retrieve last frame mask  $m^k$ 
26:       Compute frame distance:  $D_i^{k,j} = w_{\text{angle}} \cdot \theta + w_{\text{dist}} \cdot \delta + w_{\text{mask}} \cdot m^k$ 
27:       Apply momentum:  $D_i^{k,j} \leftarrow (1 - \eta) \cdot D_i^{k,j} + \eta \cdot \hat{D}_i^{k,j}$ 
28:       Store previous step objective:  $\hat{D}_i^{k,j} \leftarrow D_i^{k,j}$ 
29:     end for
30:   end for
31:   Let  $\mathcal{D}_i = \{D_i^{k,j} \mid \forall k, j\}$ 
32:    $\mathcal{I}_{\text{local}}^i \leftarrow \text{Top}_n(\text{argsort}_{\text{asc}}(\mathcal{D}_i))$ 
33:    $\mathcal{I}_{\text{patch}} \leftarrow \mathcal{I}_{\text{patch}} \cup \mathcal{I}_{\text{local}}^i$ 
34: end for
35: Compute unique set:  $\mathcal{I}_{\text{unipatch}} \leftarrow \text{unique}(\mathcal{I}_{\text{patch}})$ 
36: Let  $T_{\text{rest}} \leftarrow T - |\mathcal{I}_{\text{unipatch}}|$ 
37: // Global fallback selection
38: Initialize set of fallback distances:  $\mathcal{D}_{\text{global}} \leftarrow \emptyset$ 
39: for each condition view  $k$  from 1 to  $N$  do
40:   for each condition patch  $j$  in view  $k$  do
41:     Compute  $\tilde{D}^{k,j} = \min_i D_i^{k,j}$  // best match to any target patch
42:      $\mathcal{D}_{\text{global}} \leftarrow \mathcal{D}_{\text{global}} \cup \{\tilde{D}^{k,j}\}$ 
43:   end for
44: end for
45:  $\mathcal{I}_{\text{global}} \leftarrow \text{Top}_{T_{\text{rest}}}(\text{argsort}_{\text{asc}}(\mathcal{D}_{\text{global}}))$ 
46: return  $\mathcal{I} = \mathcal{I}_{\text{patch}} \cup \mathcal{I}_{\text{global}}$ 
```

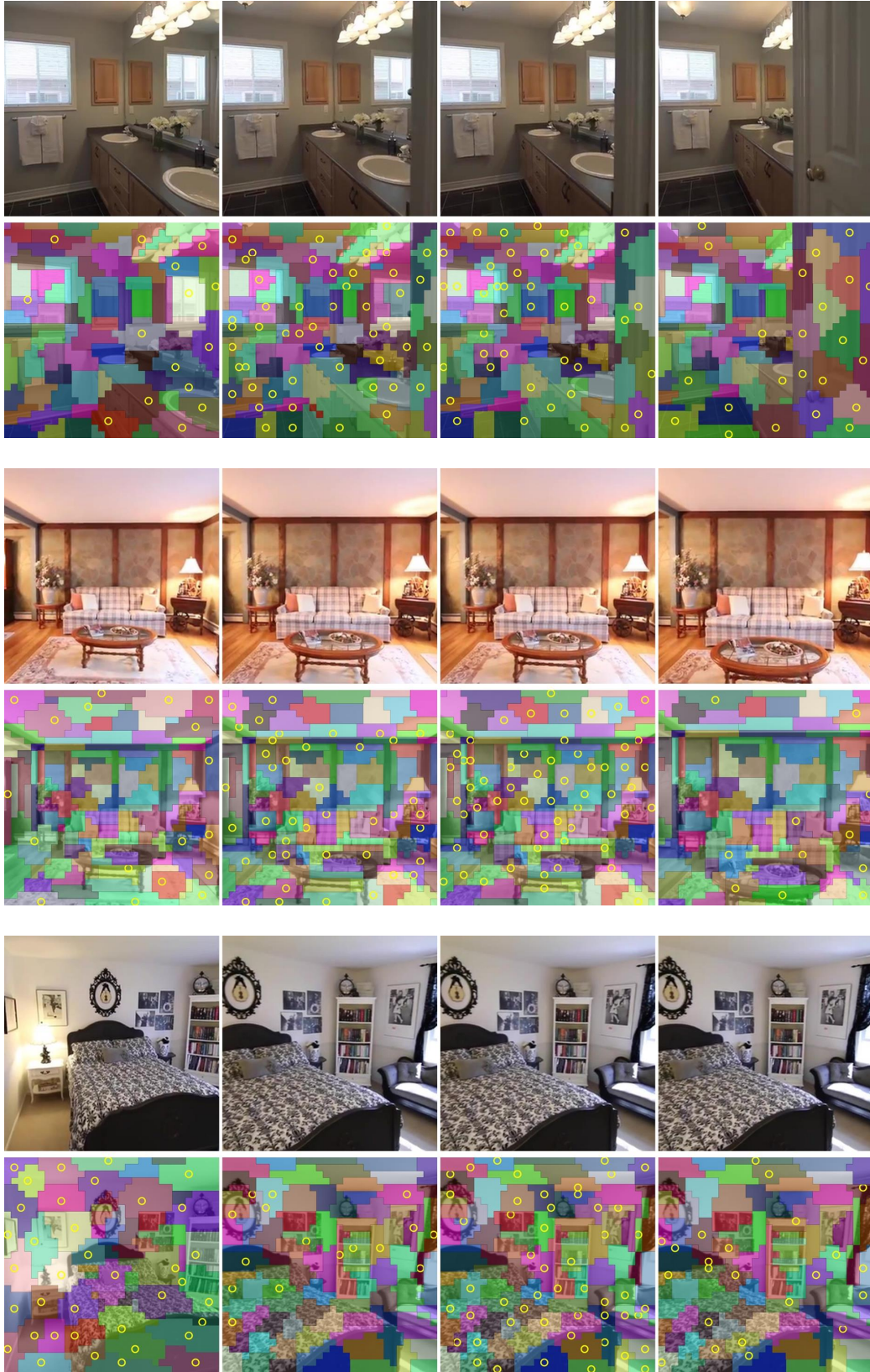


Figure 7: Additional visualization of the latent K-means algorithm for RealEstate10k dataset.



Figure 8: Additional visualization of the latent K-means algorithm for DL3DV dataset.



Figure 9: Additional visualization of the latent K-means algorithm with different values of K , which is 512, 1024, 2048, and 3072 from the top.

30 References

- 31 [1] Yuedong Chen, Haofei Xu, Chuanxia Zheng, Bohan Zhuang, Marc Pollefeys, Andreas Geiger,
32 Tat-Jen Cham, and Jianfei Cai. 2024. Mvsplat: Efficient 3d gaussian splatting from sparse
33 multi-view images. In *European Conference on Computer Vision*. Springer, 370–386.
- 34 [2] Haian Jin, Hanwen Jiang, Hao Tan, Kai Zhang, Sai Bi, Tianyuan Zhang, Fujun Luan, Noah
35 Snavely, and Zexiang Xu. 2024. Lvsm: A large view synthesis model with minimal 3d inductive
36 bias. *arXiv preprint arXiv:2410.17242* (2024).
- 37 [3] Haofei Xu, Songyou Peng, Fangjinhua Wang, Hermann Blum, Daniel Barath, Andreas Geiger,
38 and Marc Pollefeys. 2024. Depthsplat: Connecting gaussian splatting and depth. *arXiv preprint*
39 *arXiv:2410.13862* (2024).