Merge3D: Efficient 3D Multimodal LLMs via Joint 2D-3D Token Merging
Supplementary Material

Tianbo Pan[1] Xingyi Yang[2] Xinchao Wang[1*]

[1]National University of Singapore [2]The Hong Kong Polytechnic University

pantianbo@u.nus.edu, xingyi.yang@polyu.edu.hk, xinchao@nus.edu.sg

Method	Retention ratio	Token	Total Time↓	Δ ↓	Scan2Cap [1]	CV-Bench [3]	BLINK [2]
Baseline	-	1623	9.75s	-	52.6	82.1	68.4
randomzip	30%	564	8.40s	1.16×	49.7	77.3	59.9
Visionzip-3D			7.33s	1.33×	50.4	75.3	62.2
Visionzip-2D			9.46s	1.03×	50.2	79.5	65.9
Merge3D*			9.52s	1.02×	50.6	79.6	67.5
randomzip	10%	261	6.99s	1.39×	45.7	69.5	56.3
Visionzip-3D			7.14s	1.36×	46.5	68.7	56.6
Visionzip-2D			7.00s	1.39×	46.0	74.1	62.9
Merge3D*			7.03s	1.38×	48.2	77.5	64.1
randomzip	5%	186	5.77s	1.69×	43.2	65.1	56.0
Visionzip-3D			5.72s	1.70×	44.4	63.6	56.3
Visionzip-2D			5.80s	1.68×	42.5	68.9	58.0
Merge3D*			5.85s	1.67×	45.3	74.8	60.5

Table 1. Comparison of methods based on token count, total inference time, and speedup factor and accuracy on different benchmarks (3D grounding-Scan2Cap [1]; Spatial reasoning: CV-bench [3] and BLINK [2]).

1. Quantitative comparison of effieciency and accuracy

To provide a more comprehensive picture of the trade-off between efficiency and accuracy, we first average the scores of different metrics over all three benchmarks (Scan2Cap, CV-Bench and BLINK) under each retention ratio. Across all settings (30%, 10% and 5%), our Merge3D consistently achieves the best performance among all token-compression baselines, showing that jointly leveraging 2D semantics and 3D geometry leads to a strictly better accuracy–compression trade-off. We further compare the efficiency of different methods in terms of token count and wall-clock inference time, as summarized in Tab. 1. While Merge3D generally exhibits comparable or better efficiency than other compression baselines at the same retention ratio, the relative speedup over the non-compressed baseline is smaller here than in Tab.5 of the main paper. This is because Tab.5 uses vanilla attention in the backbone, under which Merge3D can reach up to 2.5× speedup, whereas Tab. 1 adopts FlashAttention for the baseline and we need to additionally materialize full attention maps for token merging, introducing extra overhead. As a result, the gain at higher retention ratios (e.g., 30%) is modest (1.02×), but under more aggressive compression (5% retention) Merge3D still achieves a 1.66× speedup while maintaining state-of-the-art accuracy across all datasets.

2. Baselines for Token Merging

In our experiments, we compare Merge3D against three token-merging baselines adapted from VisionZip [4] to the dual-encoder 3D VLM setting: VisionZip-2D, VisionZip-3D, and RandomZip. These baselines are designed to disentangle the roles of 2D semantic cues and 3D geometric cues in token selection and merging. All methods operate on the same backbone and share the initial feature extraction and fusion steps: given multi-view frames, we obtain attention maps, attention keys, and per-token features from the 2D semantic encoder and the 3D geometry encoder, and then compute the fused features by element-wise addition. For a fair comparison, all baselines and Merge3D keep the same number of tokens $K = r \times N$ for a

```
# Extract from encoders
attn_2d, attn_key_2d, features_2d = run_2d_encoder(frames)
attn_3d, attn_key_3d, features_3d, spatial_sim_map = run_3d_encoder(frames) #
    Includes vggt_sim_map
features_fused = features_2d + features_3d # Fuse features

# Compute attention scores (mean over queries and heads)
attn_scores_2d = attn_2d.mean(dim=[queries, heads])

# Select top-K target indices based on 2D scores (K = retain_ratio * total_tokens)
target_idx = topk(attn_scores_2d, K)

# Mask out targets for merges (all remaining are merges)
merges_mask = ~in(target_idx)
merges_attn_key = attn_key_2d[merges_mask] # Use 2D keys for semantic sim
norm_merges_key = normalize(merges_attn_key) # For cosine sim
merges_features = features_fused[merges_mask] # For actual merging

# Targets keys and features
targets_key = attn_key_2d[target_idx]
norm_targets_key = normalize(targets_key)
targets_fused = features_fused[target_idx]

# Compute semantic similarity (using 2D keys)
sem_sim = dot(norm_merges_key, norm_targets_key.T)

# Extract and normalize 3D spatial weights (submatrix for merges-to-targets)
merges_idx = nonzero(merges_mask)
spatial_weight = spatial_sim_map[merges_idx[:, None], target_idx[None, :]]
norm_spatial_weight = (spatial_weight - min(spatial_weight)) / (max(spatial_weight
    ) - min(spatial_weight) + epsilon)

# Fuse similarities: hybrid sim = semantic * spatial
hybrid_sim = sem_sim * norm_spatial_weight

# Assign merges to targets
assign_idx = argmax(hybrid_sim, dim=targets_key)

# Average assigned merges per target (using fused features)
avg_merged = group_avg(merges_fused, assign_idx)

# Contextual tokens: targets_fused + avg_merged
context_tokens = targets_fused + avg_merged
```

Figure 1. Pseudo for Merge3D.

given retention ratio r, where N is the number of input tokens. The differences lie in how dominant/target tokens are chosen and how the remaining contextual tokens are merged. Below we describe each baseline and then summarize our Merge3D design.

Merge3D. integrates both 2D semantic guidance and 3D geometric guidance in a unified token-merging strategy tailored for dual-encoder 3D VLMs. As illustrated in Fig. 1, we first compute 2D attention scores by averaging the 2D attention map across queries and heads, and select the top-K tokens as targets (i.e., the tokens to be retained after compression). All remaining tokens are treated as merge tokens. For similarity computation, we use two complementary signals: (i) semantic similarity obtained from the cosine similarity between ℓ_2-normalized 2D attention keys of merge and target tokens, and (ii) geometric affinity derived from a 3D spatial similarity map produced by the VGGT encoder. We then form a hybrid similarity by element-wise multiplication of the semantic similarity and the normalized spatial affinity, so that only tokens that are both semantically related and spatially close are strongly linked. Each merge token is assigned to the target with the highest hybrid similarity, and we compute the average fused feature of the merges assigned to each target. The final contextual token for each target is obtained by adding this averaged merge feature to the target's fused feature. The set of updated targets constitutes the compressed visual token sequence. Compared to the VisionZip-based baselines, Merge3D uses 2D attention only to decide where to focus while explicitly constraining merging with 3D geometry, leading to better preservation of both

semantic details and 3D spatial structure under the same token budget.

VisionZip-2D is a semantic-only variant that uses information from the 2D visual encoder (e.g., CLIP or SigLIP) for both dominant token selection and contextual token merging. We first compute token-wise attention scores by averaging the 2D attention map over query positions and heads, and select the top-K tokens with the highest scores as dominant tokens. The remaining tokens are candidates for contextual merging. From these remaining tokens, we uniformly sample a subset of M targets using strided sampling to ensure an even spatial and semantic coverage. Their 2D attention keys are ℓ_2-normalized and used to compute cosine similarities with the normalized keys of the other remaining tokens (merges). Each merge token is assigned to the most similar target based on 2D key similarity. Aggregation is then performed in the fused feature space: for each target, we average the fused features of its assigned merges and add this average to the target's fused feature. Finally, we keep only the dominant tokens and the updated targets as the compressed token set, and update the corresponding masks and positions before feeding them into the LLM.

VisionZip-3D is a geometry-only variant that mirrors VisionZip-2D but uses information from the 3D geometry encoder (e.g., VGGT) instead. Concretely, we compute token-wise attention scores from the 3D attention map, again by averaging over queries and heads, and select the top-K tokens as dominant tokens, favoring tokens that are spatially important according to the 3D encoder. Among the remaining tokens, we perform the same strided sampling to choose M targets, and use their ℓ_2-normalized 3D attention keys to compute cosine similarities with the keys of the remaining merge tokens. Assignment of merges to targets is based on this 3D-key similarity, while aggregation is still carried out on the fused features to retain both semantic and geometric information. By relying solely on 3D attention and keys for selection and grouping, this baseline isolates the effect of geometry-driven compression, which tends to preserve spatial coherence but can be suboptimal on benchmarks that require fine-grained semantics.

RandomZip serves as a naive control baseline in which selection is largely random. We first randomly sample K dominant token indices from all fused tokens, without using any encoder-specific scores. From the remaining tokens, we randomly sample M targets, and treat the others as merge tokens. To keep the merging stage comparable to VisionZip-2D, we follow its similarity-based assignment scheme: we use the ℓ_2-normalized 2D attention keys of remaining tokens to compute cosine similarities between merges and targets, assign each merge to its most similar target, and then average the fused features of assigned merges for each target. The aggregated features are added to the targets' fused features, and we keep the dominant tokens together with the updated targets as the final compressed token set. This baseline isolates the effect of informed token selection; because both dominant and target tokens are chosen randomly, it typically yields the lowest performance among all methods.

References

[1] Zhenyu Chen, Ali Gholami, Matthias Nießner, and Angel X Chang. Scan2cap: Context-aware dense captioning in rgb-d scans. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3193–3203, 2021. 1

[2] Xingyu Fu, Yushi Hu, Bangzheng Li, Yu Feng, Haoyu Wang, Xudong Lin, Dan Roth, Noah A Smith, Wei-Chiu Ma, and Ranjay Krishna. Blink: Multimodal large language models can see but not perceive. In *European Conference on Computer Vision*, pages 148–166. Springer, 2024. 1

[3] Peter Tong, Ellis Brown, Penghao Wu, Sanghyun Woo, Adithya Jairam Vedagiri IYER, Sai Charitha Akula, Shusheng Yang, Jihan Yang, Manoj Middepogu, Ziteng Wang, et al. Cambrian-1: A fully open, vision-centric exploration of multimodal llms. *Advances in Neural Information Processing Systems*, 37:87310–87356, 2024. 1

[4] Senqiao Yang, Yukang Chen, Zhuotao Tian, Chengyao Wang, Jingyao Li, Bei Yu, and Jiaya Jia. Visionzip: Longer is better but not necessary in vision language models. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 19792–19802, 2025. 1