RVLM: FEWER TOKENS, BETTER RESOLUTION SCALING, FINE-GRAINED SPATIAL GROUNDING

Anonymous authors
Paper under double-blind review

ABSTRACT

Vision-Language Models (VLMs) tokenize images using fixed spatial grids, so visual token count grows with pixel area rather than scene content. We propose to group neighboring patches based on their visual appearance, allowing tokens to represent coherent image regions rather than fixed spatial windows. We introduce RVLM, a VLM trained to use regions rather than patches as its visual units. RVLM partitions images using Felzenszwalb-Huttenlocher segmentation, assigns each ViT patch to a region, and pools patches within each region into a variable-length sequence of visual tokens. The patch-to-region assignment is retained to support spatially precise prediction through coarse-to-fine decoding. Across tasks spanning VQA, pointing, counting, segmentation, and depth estimation, RVLM matches or improves example-weighted task performance while using 21.5% fewer visual tokens than an identically trained patch baseline. The advantage grows with image resolution, reaching a 5.3-point gain while using 51.8% fewer visual tokens at the highest-resolution setting. The gains are largest on spatially demanding tasks, where prior token-reduction methods degrade most sharply. RVLM adds no model parameters and improves inference throughput by up to 35.9%. These results establish regions as a scalable alternative to patch-based visual representations while preserving the spatial precision required for grounded prediction.

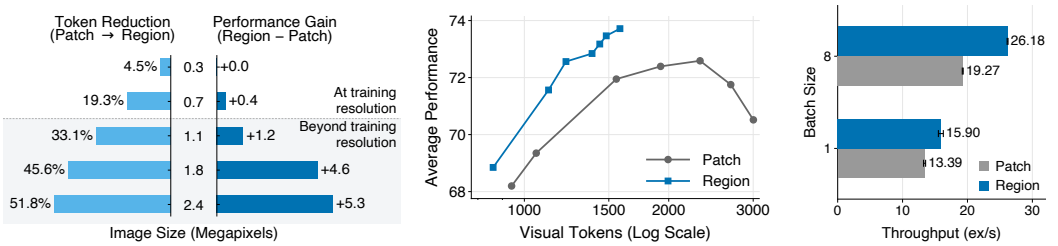
1 INTRODUCTION

Tokenizing images into a uniform grid of patches is a standard design choice in Vision-Language Models (VLMs). However, patch tokenization ties token count to pixel count rather than scene content. As a result, a high-resolution image of a single object requires more tokens than a lower-resolution image of a cluttered scene, and the blank margins of a document may consume more tokens than the text, simply because they occupy more area. This content-agnostic allocation wastes the visual token budget and limits the maximum image resolution a patch-based VLM can process within a fixed token budget.

Existing models reduce visual token count through fixed spatial pooling (Chen et al., 2023a; 2025b; Wang et al., 2024b) or by pruning and merging patch tokens (Bolya et al., 2023; Chen et al., 2024a; Yang et al., 2025b). Fixed pooling ignores image content, while pruning and merging can remove spatial coverage or obscure the correspondence between visual tokens and image locations needed for spatial grounding. Content-adaptive tokenization offers an alternative by letting visual units follow image content rather than a fixed grid (Lee et al., 2026; Chen et al., 2025a; Khosla et al., 2026). However, existing approaches rely on learned features or segmentation models, introducing additional computational costs or making region construction sensitive to VLM training (Section 4.4).

We present RVLM, a VLM trained to use regions rather than patches as its visual units. Unlike fixed spatial pooling, we group neighboring patches based on their visual appearance, forming more meaningful units for pooling visual information. We construct such regions using training-free Felzenszwalb-Huttenlocher segmentation (Felzenszwalb & Huttenlocher, 2004), assign each ViT patch to a region, and pool patch features within regions to produce a visual token sequence. RVLM maintains the patch-to-region assignment to support spatially grounded prediction. For pointing, it decodes locations coarse-to-fine, first predicting the relevant region, then a patch within that region, and finally a location within the patch. Tokenization adds no parameters over the patch baseline, and

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.



(a) Regions scale better than patches as image size increases. RVLM matches or outperforms an identically trained patch baseline while using fewer tokens. The largest gains emerge under test-time resolution scaling (Table 10).

(b) At a given token budget, RVLM can process higher-resolution images than patch-based models, resulting in better performance. The gains are strongest on resolution-sensitive tasks.

(c) RVLM's shorter sequences increase inference throughput, with gains growing from 19% at batch size 1 to 36% at batch size 8 (Table 3).

Figure 1: **Region tokens scale better with image resolution while improving inference efficiency.** See Section 4.2 for details.

the coarse-to-fine decoding preserves spatial precision even when a single token represents a large portion of the image.

We evaluate RVLM across 35 tasks and compare it with a matched patch-based VLM and similarly sized open-weight models. Our results show that regions improve resolution scaling and inference efficiency while preserving the spatial correspondence needed for grounded prediction (Figure 1). We also study the properties required for effective region construction (Section 4.4).

In summary, our main contributions are:

1. **A VLM built on region tokens.** We present RVLM, a VLM that replaces fixed spatial pooling with content-adaptive region-based tokenization. Compared to an identically trained patch-based VLM, RVLM uses 21.5% fewer visual tokens on average and improves inference throughput by up to 35.9%. It also scales more effectively to higher test-time resolutions, achieving a 5.3-point gain with 51.8% fewer tokens at the highest resolution tested. Our results show that RVLM's region tokenization and design are particularly effective for fine-grained spatial tasks, where prior methods that reduce token count degrade sharply.
2. **Requirements for effective region-based tokenization.** We investigate region construction strategies and identify five desirable properties for VLMs: complete patch coverage, spatial connectivity throughout training, no additional model parameters or forward passes, region assignments available before model execution, and independence from the vision encoder. Importantly, we find that regions need not align with complete objects, as the LLM can combine information across multiple region tokens, just as it does across patch tokens. Classical pixel-based Felzenszwalb-Huttenlocher segmentation satisfies all these requirements.

2 Related Work

Visual tokenization in VLMs. VLMs encode images into patch features using a ViT and map them into the LLM embedding space through a connector (Liu et al., 2023; Chen et al., 2023b). Since patch sequences grow rapidly with image resolution, connectors typically reduce the number of visual tokens passed to the LLM. One approach uses a fixed number of learned queries (Li et al., 2023; Alayrac et al., 2022; Bai et al., 2023), decoupling token count from image size but weakening the correspondence between tokens and image locations. A more common approach pools patch features on a fixed spatial lattice. Existing methods use pixel shuffle (Chen et al., 2024b; Laurençon et al., 2024; Marafioti et al., 2025), adaptive average pooling (Yao et al., 2024), convolutional or deformable projectors (Cha et al., 2024; Chu et al., 2024; Ye et al., 2023), and cross-attention over local patch features (Li et al., 2025; Deitke et al., 2025; Clark et al., 2026b). While these designs differ in how they aggregate features, their spatial pooling units are determined by position rather than image content. Increasing the pooling window reduces the token count but also coarsens the visual representation, potentially impairing localization (Marafioti et al., 2025). In contrast, RVLM

constructs pooling units from image content, producing region tokens of varying spatial extent while retaining explicit patch-to-token correspondence for spatial grounding.

Content-adaptive tokenization. Another line of work constructs content-adaptive visual tokens via semantic clustering of the vision encoder's patch features (Jin et al., 2024; Wu et al., 2025; Lee et al., 2026; Mao et al., 2026; Khosla et al., 2025; 2026). These approaches, however, depend on the spatial fidelity of the patch features, and we find that end-to-end VLM training destabilizes region assignments, producing spatially disconnected groups (Section 4.4). Related adaptive-length tokenizers represent images as variable-length latent sequences (Duggal et al., 2025; Yan et al., 2025), but lack explicit correspondence between tokens and image locations, limiting their use for spatial grounding. In contrast, RVLM constructs spatially connected regions directly from image pixels while retaining patch-to-region correspondence for precise spatial grounding. Prior work has also explored region-based visual tokenization in vision encoders, image autoencoders, and VLMs (Lew et al., 2025; Wang et al., 2024a; Ma et al., 2024; Yang & Gong, 2025). Closest to ours, Chen et al. (2025a) study SAM-based region tokenization and propose a lightweight learned boundary detector followed by watershed segmentation to ensure complete image coverage. However, learned detectors and segmenters introduce additional computational and memory overhead during region construction. RVLM instead uses training-free segmentation on the CPU during image loading, avoiding additional model parameters and accelerator computation while allowing preprocessing to completely overlap with VLM execution.

Post-hoc token compression. A separate line of work reduces visual tokens by pruning less important patches (Zhang et al., 2025a; Shang et al., 2025; Chen et al., 2024a; Zhang et al., 2025c; Xing et al., 2025; Yang et al., 2025b) or merging redundant features (Bolya et al., 2023; Wen et al., 2025b; Alvar et al., 2025; Zhang et al., 2025b). Although effective for image-level understanding, these methods degrade performance on spatially grounded tasks (Wen et al., 2025a; Chien et al., 2025). Pruning removes information at discarded locations, while merging can obscure the correspondence between tokens and individual patches. In contrast, RVLM constructs content-adaptive pooling units rather than compressing an existing token sequence, retaining complete patch coverage and spatial correspondence. We compare these approaches at matched per-image token budgets in Section 4.3.

3 METHOD

We introduce RVLM, a VLM trained to use regions as its visual units (Figure 2). Conventional spatial pooling summarizes patches within fixed spatial windows, regardless of their visual content. Instead, we group spatially connected patches with similar appearance and summarize each group into a visual token. These groups need not correspond to complete objects; when an object spans multiple regions, the LLM can combine information across their tokens, just as it does across patch tokens or subword tokens. Our goal is therefore to construct content-adaptive pooling units that follow local image structure while covering every pixel of the image.

3.1 MODELING

RVLM follows the standard VLM architecture, combining a pretrained vision encoder and LLM through a trainable connector. We initialize the vision encoder from SigLIP2-So400m/14 (Tschannen et al., 2025) and the LLM from Qwen3-4B-Instruct (Yang et al., 2025a). The connector is initialized from scratch, and it aggregates ViT patch features into region tokens using multi-head cross-attention. Following Molmo (Deitke et al., 2025), the vision encoder processes each image as a low-resolution overview and up to K overlapping high-resolution crops. If the image requires more than K crops, we downscale it until it fits within this budget. The low-resolution overview is processed through standard 2×2 patch pooling, producing 196 visual tokens. For the high-resolution pathway, we extract ViT features from individual crops and assemble them into a common patch lattice before applying region-based attention pooling. All spatially grounded predictions, including pointing and segmentation, operate on this high-resolution representation, while the overview is used only for global context. We use $K{=}14$ during training and increase it to $K{=}32$ at inference to study scaling to higher image resolutions.

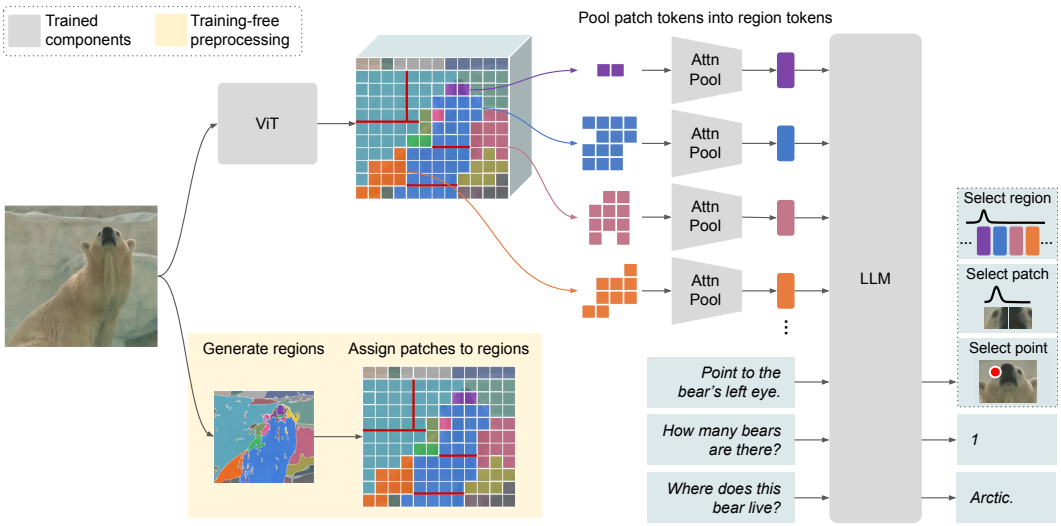


Figure 2: **Overview of RVLM.** We generate content-adaptive regions using Felzenszwalb-Huttenlocher segmentation and assign each ViT patch to a region. This is a CPU-side preprocessing that runs in parallel to the model execution. Patch features within each region are aggregated through attention pooling to produce a sequence of region tokens for the LLM. The retained patch-to-region correspondence enables coarse-to-fine spatial grounding, where the LLM selects a region, then a patch within it, and finally a precise point within the patch.

Generate regions. We use Felzenszwalb-Huttenlocher (FH) segmentation (Felzenszwalb & Huttenlocher, 2004) to partition images into spatially connected regions with complete coverage. FH merges neighboring components based on color similarity relative to their internal variation. We use a scale parameter k=100, which controls how aggressively merges are allowed; Gaussian smoothing with σ=0.8 before edge weighting, which reduces noise; and a minimum component size of 32 pixels, below which components are absorbed into neighbors. The image's longer side is capped at 512 pixels during training and 768 during inference; the resulting label map is resized using nearest-neighbor interpolation. We discuss other region generation strategies in Section 4.4.

Assign patches to regions. We assign each ViT patch to exactly one FH region. To preserve fine structures, we assign each patch to the smallest region covering at least 25% of its pixels, falling back to the region with the largest overlap if none meets this threshold. For efficient batched attention pooling, we limit each group to w=16 patches, splitting larger regions into spatially contiguous groups. This retains all patch features for spatial grounding while controlling padding and the spatial extent of each token. We study the choice of w in Appendix B.2.

Pool patches into region tokens. We convert each patch group into a visual token using a cross-attention connector. For each group, we gather features from the third-to-last and ninth-from-last layers of the vision encoder and concatenate them. To capture the spatial structure of each region, we add absolute sinusoidal position embeddings encoding the spatial coordinates (t, h, w) of each patch. Following Clark et al. (2026b), we use the mean of the valid patch features as the query for a multi-head cross-attention layer, with the individual patch features serving as keys and values. Groups smaller than w=16 are padded, and the padded positions are masked during attention. The resulting representation is projected into the LLM embedding space through an MLP, producing one visual token per group. Unlike fixed-window pooling, this connector adaptively aggregates features from spatially contiguous regions of varying sizes.

Position encoding. We position each region token in the LLM using interleaved M-RoPE (Huang et al., 2026). Specifically, we assign each token a 3D position (t, h, w), where t denotes the image's position in the input sequence and (h, w) are the coordinates of the patch closest to the region's spatial center. This places region tokens in the same spatial coordinate system as the original patch grid without requiring regions to be rectangular. Thus, M-RoPE encodes the region's position within the image for the LLM, while the spatial structure of the region is encoded using the absolute patch-level positional embeddings added before the attention pooling of patch groups into region tokens.

Table 1: **Comparison with open-weight VLMs.** Our identically trained patch and region models achieve competitive performance across image QA, pointing, and counting benchmarks. Thus, PVLM provides a strong baseline for isolating the effect of region-based visual tokenization. For TextVQA, we report validation-set results for our models because the test server is unavailable.

Model	VQAv2	AI2D	PixMo-Point	PixMo-Count	CountBench	TextVQA	DocVQA
InternVL3.5-4B	78.1	82.6	–	62.4	82.2	77.9	92.4
PLM-3B	84.4	90.9	–	63.0	87.1	84.3	93.8
Qwen3-VL-4B	81.7	84.1	43.7	58.0	85.5	81.0	**95.3**
Molmo2-4B	86.6	**95.6**	83.4	88.1	**93.9**	85.0	92.4
PVLM-4B	**87.1**	95.4	85.2	90.1	90.2	**85.5**	94.4
RVLM-4B	87.0	95.1	**86.4**	**90.4**	91.8	85.0	93.7

Coarse-to-fine spatial grounding. For pointing, we follow the coarse-to-fine decoding strategy of MolmoPoint (Clark et al., 2026a), adapted to region tokens. Instead of directly predicting image coordinates, the LLM emits a special token and uses its hidden state as a query that is scored against keys for all region tokens (dot-product retrieval); the highest-scoring region is selected. A second special token likewise retrieves a ViT patch among that region's owned patches, and a third predicts the point's location within the selected patch on a 4×4 grid. We use the patch-to-region assignment to map the predicted region, patch, and local position back to image coordinates. For segmentation, we reuse the first stage of pointing to select an anchor region for the target object. The LLM then generates a special segmentation token whose hidden state is used to predict mask scores at three levels: regions, patches within each region, and 4×4 cells within each patch. For each cell, we sum its region-, patch-, and cell-level scores to obtain a mask logit. Using the retained patch-to-region assignment, we map these logits back to their corresponding image locations, apply a sigmoid, and threshold the resulting probabilities at 0.5 to obtain a binary mask. This allows irregular region tokens to support dense, spatially precise predictions without requiring a regular pooling grid.

3.2 Training

We jointly train the vision encoder, connector, and LLM in two stages, following the Molmo2 training recipe (Clark et al., 2026b). Stage 1 consists of 31,971 steps of multimodal pretraining on captioning, pointing, counting, OCR, and text instruction data. Stage 2 consists of 24,000 steps of supervised fine-tuning, adding image QA, segmentation, and depth prediction. We optimize next-token cross-entropy together with the spatial grounding objectives of MolmoPoint (Clark et al., 2026a). Pointing uses cross-entropy at each level of coarse-to-fine decoding, while segmentation uses focal and Dice losses. Appendix A provides the full training configuration.

4 Experiments

We evaluate whether region tokens can replace patches as the visual representation in general-purpose VLMs. To isolate the effect of visual tokenization, we train PVLM, a Patch-based Vision-Language Model with the same architecture, training data, and recipe as RVLM. The two models differ only in their visual tokenization strategy. We first compare both models against similarly sized open-weight VLMs (Section 4.1). We then evaluate the effect of region tokenization across 35 tasks spanning image QA, pointing, counting, segmentation, and depth estimation (Section 4.2). Next, we compare RVLM against post-hoc token compression methods (Section 4.3). Finally, we analyze region generation strategies and identify the properties needed for effective region-based VLMs (Section 4.4).

4.1 Comparison with open-weight VLMs

Table 1 compares RVLM and PVLM with similarly sized open-weight VLMs across standard image benchmarks. Both models achieve competitive performance across image QA, pointing, and counting. Importantly, RVLM achieves this performance using content-adaptive region tokens rather than conventional patch tokens. These results also establish that PVLM is a strong baseline. Since PVLM

Table 2: **Patch vs. region tokenization.** RVLM preserves or improves overall performance over the identically trained patch baseline while using fewer visual tokens. Its main advantage, however, comes from enabling higher-resolution processing, as shown in Figure 1b. Example-weighted averages give equal weight to every example, while family-weighted averages give equal weight to each task family. Token counts are averaged across examples.

Model	Example-weighted avg.	Family-balanced avg.	Token count
PVLM	82.8	79.1	985
RVLM	83.4 (↑0.6)	79.8 (↑0.7)	773 (↓21.5%)

(a) **Reasoning / understanding.** On general Image QA, regions remain near parity with patches. On document-style VQA they are competitive on ChartQA, DocVQA, and TextVQA, with a clear drop only on InfoVQA.

Task	PVLM	RVLM
Image QA		
A-OKVQA DA	75.5	74.8
A-OKVQA MC	88.6	88.2
AI2D	95.4	95.1
OK-VQA	65.8	66.0
ScienceQA	98.2	97.8
VQAv2	87.2	87.1
ChartQA	83.7	82.9
DocVQA	93.2	92.8
InfoVQA	79.2	74.3
TextVQA	85.5	85.0
Task average	85.2	84.4
Token count	793	700

(b) **Grounding.** Regions match or beat patches in every family while using 32-41% fewer visual tokens.

Task	PVLM	RVLM
Pointing		
OSWorld-G	51.6	52.0
PixMo-Point	85.2	86.4
PointBench	63.8	63.3
SA-Co Gold	87.5	90.1
ScreenSpot Pro	29.3	26.0
ScreenSpot v2	82.7	85.0
Task average	66.7	67.1
Token count	1,793	1,065
Counting		
CountBenchQA	90.2	91.8
PixMo-Count	90.1	90.4
Task average	90.2	91.1
Token count	1,655	990
Segmentation		
PixMo-Seg	58.0	59.8
RefCOCO	82.0	82.5
RefCOCO+	78.5	78.7
RefCOCOg	80.3	81.1
SA-Co Gold	50.7	54.2
Task average	69.9	71.2
Token count	1,390	950

(c) **3D.** We report scale-invariant $\delta_{1.25}$ for point-to-depth across 12 datasets. Regions match or beat patches on average while using 43% fewer visual tokens.

Task	PVLM	RVLM
Depth		
Argoverse 2	77.9	81.8
Aria AEA	68.5	69.5
Aria Nymeria	54.6	55.6
DDAD	81.7	87.7
DIODE	92.2	94.0
EgoExo4D Ego	89.1	88.9
EgoExo4D Exo	84.0	82.9
nuScenes	84.7	88.8
NYU Depth v2	97.8	97.5
Stereo4D	89.0	88.6
SUN RGB-D	98.2	98.2
Waymo	84.1	90.8
Task average	83.5	85.4
Token count	1,617	916

and RVLM share the same architecture, training data, and recipe, our subsequent comparisons isolate the effect of visual tokenization.

4.2 Region vs. Patch Tokenization

We compare PVLM and RVLM across 35 tasks spanning VQA, pointing, counting, segmentation, and depth (Table 2). RVLM improves the family-balanced average by 0.7 points and the example-weighted average by 0.6 points while using 21.5% fewer visual tokens. This shows that regions shorten the visual sequence without sacrificing performance.

Regions improve performance on spatial tasks. Table 2 shows that the largest gains are on counting ($+0.9$), segmentation ($+1.3$), and depth ($+1.9$), with particularly strong improvements on SA-Co Gold, DDAD, Waymo, and nuScenes. These gains indicate that regions are more effective representation units for reasoning about coherent objects and surfaces. The largest drops occur on InfoVQA (-4.9), ScreenSpot Pro (-3.3), and ChartQA (-0.8), where relevant information often occupies very small areas (e.g., small text in thin fonts). Failure analysis shows that these super-fine details are sometimes absorbed into larger surrounding regions and reduce spatial precision.

Regions scale better with image size. We group examples by native resolution into five buckets to test how region and patch tokenization scale with image size. Figure 1a summarizes the results, with full breakdown reported in Table 10. As resolution increases, RVLM achieves progressively larger performance gains while using increasingly fewer visual tokens than the patch baseline. Region tokenization therefore benefits more from additional image detail without the rapid sequence-length growth of patch tokenization. This makes it particularly well suited to high-resolution settings such as documents, screens, robotics, and dense visual prediction.

Regions yield better performance at fixed token budget. We select high-resolution images from all tasks in the evaluation suite, yielding 21,273 examples ranging from 2.08-2.63 MP, and re-

Table 3: **Inference cost.** Throughput is examples processed per second of wall-clock time over 512 randomly sampled examples from PixMo-Cap, including data loading and region construction. Generation is capped at one output token, so throughput reflects prefill. Measurements use a single NVIDIA H100 after 64 warm-up batches. We report the mean and standard deviation over five runs. Shorter visual sequences increase throughput and reduce peak memory, with larger gains at larger batch sizes. Table 11 shows a finer-grained cost comparison across image resolutions.

Model	Batch size 1		Batch size 8	
	Throughput (ex/s) ↑	Peak mem. (GiB) ↓	Throughput (ex/s) ↑	Peak mem. (GiB) ↓
PVLM	13.39±0.16	19.98	19.27±0.07	26.75
RVLM	**15.90±0.38**	**19.75**	**26.18±0.07**	**24.69**
Δ	**+18.7%**	**−1.2%**	**+35.9%**	**−7.7%**

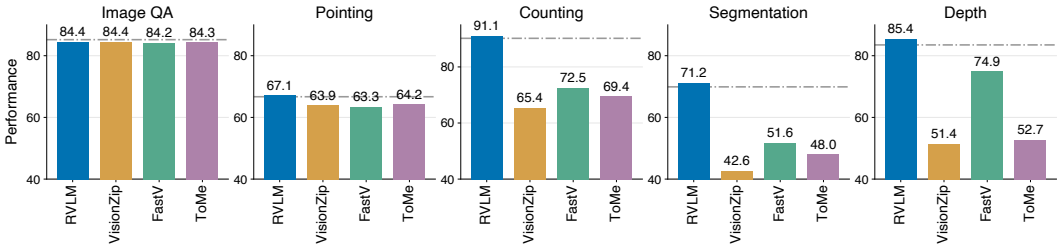


Figure 3: **Post-hoc token compression methods degrade sharply on tasks requiring spatial precision.** RVLM outperforms token pruning (FastV), token merging (ToMe), and hybrid (VisionZip) methods on pointing, segmentation, counting, and depth estimation, while all methods perform comparably on Image QA. Horizontal line denotes the PVLM baseline.

evaluate both RVLM and PVLM across seven "maximum-resolution" settings. In each setting, the selected high-resolution examples are capped at a prescribed maximum image resolution, allowing us to measure performance as progressively more image detail is made available. This produces the matched token-budget comparison shown in Figure 1b. Since regions can represent higher-resolution inputs within the same visual token budget, RVLM consistently outperforms PVLM when the LLM receives a comparable number of visual tokens. This allows RVLM to trade its token savings for higher input resolution and improve performance under a fixed LLM context budget.

Cost analysis. We report end-to-end throughput and memory measurements in Table 3. The throughput gains with region tokenization come from shorter visual sequences processed by the LLM. FH segmentation adds roughly 49 ms of CPU preprocessing per image. With eight dataloader workers, this preprocessing keeps pace with GPU execution at both batch sizes. A finer-grained analysis across image resolutions (Table 11) shows that PVLM is slightly faster at one and four crops for batch size 1, where the savings from fewer tokens do not yet offset region-pooling overhead. From eight crops onward, RVLM achieves lower latency at both batch sizes, with increasing latency and memory savings at higher resolutions. These results are measured with a 4B-parameter model; larger LLMs may yield greater efficiency gains from shorter visual sequences.

4.3 RVLM vs. Token Compression

Another way to shorten visual sequences is to compress patch tokens via pruning (FastV (Chen et al., 2024a)), merging (ToMe (Bolya et al., 2023)), or both (VisionZip (Yang et al., 2025b)). We apply these methods to PVLM and compare them with RVLM in Figure 3. For a controlled comparison, we match the visual token budget per image by setting each compression method's target token count to the number of region tokens produced by RVLM for that image. This gives the baselines a per-image signal that their own formulations do not assume, enabling them to compress dense scenes less aggressively than sparse ones.

As shown in Figure 3, post-hoc compression largely preserves performance on image understanding tasks, consistent with prior findings that visual sequences contain redundancy. However, we observe large performance drops on pointing, segmentation, and point-to-depth prediction, which is consistent with the findings of Chien et al. (2025). Pruning removes individual spatial representations,

Table 4: **Comparing region generation strategies.** Requirements are derived from the failure modes discussed in Section 4.4. Performance is reported on PixMo-Cap (Cap) and PixMo-Point (Point) after stage-1 training with Qwen-1.7B as the language model.

Strategy	Complete patch coverage	Connectivity preserved after training	No external region encoder	Encoder-independent regions	Regions at image loading	Cap ↑	Point ↑
Region encoder network							
T-REN w/ SigLIP-2	✓	✗	✓	✗	✗	52.9	77.6
T-REN w/ DINOv3	✓	✗	✓	✗	✗	51.9	80.8
T-REN w/ C-RADIOv4-H	✓	✗	✓	✗	✗	52.9	84.4
Feature-based clustering							
DiVT (Frozen ViT)	✓	✓	✓	✗	✗	51.4	79.8
DiVT (Trainable ViT)	✓	✗	✓	✗	✗	52.6	80.2
DiVT (Dual ViT)	✓	✓	✗	✗	✗	53.1	86.4
Learned segmenter							
SAM	✗	✓	✗	✓	✗	–	–
EPOC	✓	✓	✗	✓	✗	53.1	86.6
Training-free pixel clustering							
SLIC	✓	✓	✓	✓	✓	53.0	86.5
FH (ours)	✓	✓	✓	✓	✓	53.8	86.6

while merging combines information from multiple locations into shared tokens. In contrast, RVLM retains an explicit assignment from every image patch to a region token, allowing its coarse-to-fine decoder to recover spatially precise predictions. The advantage of RVLM is therefore not just fewer visual tokens, but a shorter sequence that preserves both spatial coverage and correspondence.

4.4 Constructing Regions

RVLM requires mapping image patches to regions. We evaluate multiple region generation strategies and analyze their failure cases to identify the key requirements for successful region-based VLMs. We discuss each strategy below and summarize our findings in Table 4 and Figure 4.

Region encoder network. Our first approach replaces the patch-based ViT with T-REN (Khosla et al., 2026), an efficient image encoder that directly produces region tokens. While promising as a natural starting point for region-based VLMs, this approach reveals two fundamental challenges. First, the region quality depends strongly on the choice of vision encoder. T-REN relies on the spatial structure of dense patch features to generate regions, making some vision backbones better suited than others. As shown in Table 4, we train RVLM using T-REN with SigLIP-2, DINOv3, and C-RADIOv4-H, and find that SigLIP-2-based T-REN lacks sufficient spatial precision, while DINOv3 provides better spatial features but transfers less effectively to the LLM. C-RADIOv4-H achieves the strongest overall performance among the evaluated vision encoders. More critically, with T-REN, regions lose spatial connectivity during VLM training. Even with C-RADIOv4-H, after just 1,000 training steps, $\sim$37% of region tokens become spatially disconnected in text-heavy images from PixMo-Cap. A single position ID cannot faithfully represent multiple disconnected components, making it difficult for the model to associate visual content with its location. This is particularly problematic for structured text images (Figure 5 in Appendix). Enforcing connectivity produces near-rectangular regions, sacrificing the semantic grouping that motivates region tokenization. Freezing T-REN alleviates disjoint regions but degrades performance.

Feature-based clustering. Following DiVT (Lee et al., 2026), we explore constructing regions by grouping patches with similar ViT features. We first adopt DiVT's training setup, keeping the vision encoder frozen while training the connector and fine-tuning the LLM. However, freezing the ViT limits downstream performance compared to end-to-end training. We therefore allow the ViT to train jointly with the LLM and connector, but find that as patch embeddings adapt, the resulting regions become spatially disconnected, reproducing the failure observed with T-REN. To avoid this trade-off, we decouple region generation from VLM training by using a separate frozen ViT for patch grouping while allowing the VLM's vision encoder to train freely. With C-RADIOv4-H as the frozen encoder, this variant approaches the performance of the fully trained patch baseline while

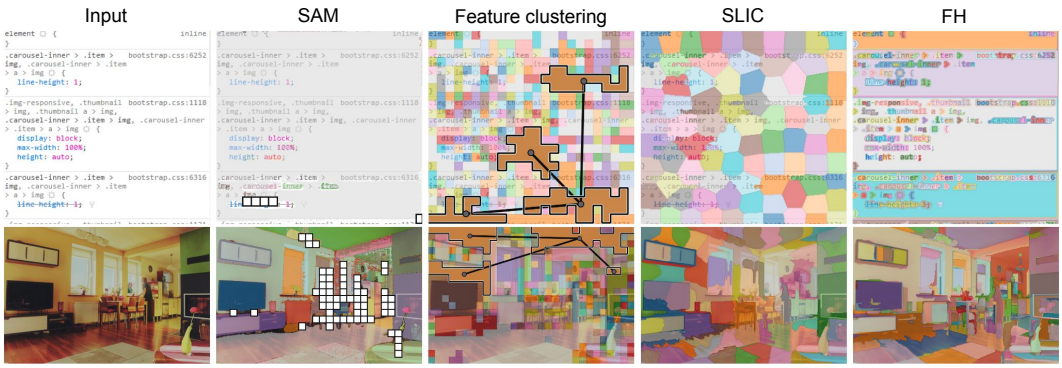


Figure 4: **Comparing region generation methods.** SAM leaves image regions uncovered (white boxes), while learned feature-based methods (DiVT, T-REN) produce disconnected regions during VLM training. SLIC often fails to isolate text, whereas FH preserves fine text structures with connected regions and complete image coverage.

using fewer visual tokens. However, it requires a second ViT forward pass for every image, adding parameters, latency, and memory overhead.

Learned segmenter. Segmentation models like SAM (Kirillov et al., 2023) can be used to obtain very precise segmentations. However, generating all regions for an image with SAM is slow and computationally expensive. More importantly, SAM does not guarantee complete image coverage, leaving some patches without a region assignment (Figure 4). This is particularly problematic for pointing, where locations excluded from the visual representation cannot be reliably predicted. EPOC (Chen et al., 2025a) addresses incomplete coverage using a learned boundary detector followed by watershed segmentation. However, its region construction introduces preprocessing overhead. For instance, with eight dataloader workers, FH achieves a segmentation throughput of 159 images/s, compared to 23 images/s for EPOC, a $\sim 7\times$ difference. FH preprocessing can be hidden behind VLM execution, whereas EPOC's accelerator-dependent pipeline limits this overlap. We also find that EPOC underperforms FH on text-heavy images, which account for much of the PixMo-Cap performance gap in Table 4.

Training-free pixel clustering. Methods such as SLIC (Achanta et al., 2012) and FH (Felzenszwalb & Huttenlocher, 2004) avoid the limitations of the previous approaches. They are training-free and operate directly on image pixels, making region quality independent of the VLM's vision encoder. They also provide spatially connected regions and complete image coverage without adding model parameters or forward passes. Although these methods tend to produce finer partitions and achieve less token reduction than T-REN or SAM, they still provide content-adaptive visual tokens while satisfying the requirements for end-to-end VLM training and spatial grounding. Importantly, regions need not correspond to complete objects because the LLM can combine information across multiple region tokens, just as it does across patch tokens. Computing region assignments in pixel space also simplifies training. Since region generation runs on CPU during data loading, both the number and spatial extent of region tokens are known before the image enters the model. Dataloader workers can therefore assign position IDs based on each region's spatial extent and pack variable-length sequences in parallel with GPU computation. In contrast, GPU-dependent methods must either run the encoder in the dataloader, stalling the input pipeline, or defer region-dependent preprocessing to the forward pass. The latter requires placeholder sequences with a fixed visual token budget, wasting computation when the budget is too large and truncating region sequences when it is too small.

5 Conclusion

We present RVLM, a VLM that replaces fixed spatial pooling with content-adaptive region tokens. Our results show that regions preserve or improve performance while using fewer visual tokens, with growing benefits at higher image resolutions. By retaining complete patch coverage and explicit spatial correspondence, RVLM supports precise grounded prediction without adding model parameters. These findings establish region-based tokenization as an efficient alternative to fixed spatial pooling for general-purpose VLMs. Future work can extend this approach to video.

AI USE STATEMENT

In this work, we used generative AI tools to improve the clarity and readability of author-written text, assist with manuscript formatting, create exploratory visualizations used only during development, and refactor and optimize existing code without changing the underlying methods. The authors manually reviewed all AI-assisted material and inspected AI-generated code. Code changes were validated by confirming that their outputs matched those of the original implementations. We also verified factual claims and citations against original sources and checked table entries against source data. We take responsibility for the final content of this work, including all text, claims, code, and artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We provide the model architecture, region construction procedure, patch-to-region assignment, and spatial grounding formulation in Section 3. Appendix A documents the training configuration, data mixtures, preprocessing parameters, task formulations, evaluation datasets, and metrics. Section 4 and Appendix A describe the experimental setups for comparisons with patch-based models, token compression methods, and alternative region construction strategies. We will release our code upon acceptance to facilitate reproduction and further research on region-based visual tokenization.

REFERENCES

Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Süsstrunk. Slic superpixels compared to state-of-the-art superpixel methods. *PAMI*, 2012.

Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katie Millican, Malcolm Reynolds, Roman Ring, Eliza Rutherford, Serkan Cabi, Tengda Han, Zhitao Gong, Sina Samangooei, Marianne Monteiro, Jacob Menick, Sebastian Borgeaud, Andrew Brock, Aida Nematzadeh, Sahand Sharifzadeh, Mikolaj Binkowski, Ricardo Barreira, Oriol Vinyals, Andrew Zisserman, and Karen Simonyan. Flamingo: a visual language model for few-shot learning. In *NeurIPS*, 2022.

Saeed Ranjbar Alvar, Gursimran Singh, Mohammad Akbari, and Yong Zhang. Divprune: Diversity-based visual token pruning for large multimodal models. In *CVPR*, 2025.

Jinze Bai, Shuai Bai, Shusheng Yang, Shijie Wang, Sinan Tan, Peng Wang, Junyang Lin, Chang Zhou, and Jingren Zhou. Qwen-vl: A versatile vision-language model for understanding, localization, text reading, and beyond. *ArXiv*, 2023.

Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. Token merging: Your ViT but faster. In *ICLR*, 2023.

Junbum Cha, Wooyoung Kang, Jonghwan Mun, and Byungseok Roh. Honeybee: Locality-enhanced projector for multimodal llm. In *CVPR*, 2024.

Delong Chen, Samuel Cahyawijaya, Jianfeng Liu, Baoyuan Wang, and Pascale Fung. Subobject-level image tokenization. In *ICML*, 2025a.

Jun Chen, Deyao Zhu, Xiaoqian Shen, Xiang Li, Zechun Liu, Pengchuan Zhang, Raghuraman Krishnamoorthi, Vikas Chandra, Yunyang Xiong, and Mohamed Elhoseiny. Minigpt-v2: large language model as a unified interface for vision-language multi-task learning. *ArXiv*, 2023a.

Liang Chen, Haozhe Zhao, Tianyu Liu, Shuai Bai, Junyang Lin, Chang Zhou, and Baobao Chang. An image is worth 1/2 tokens after layer 2: Plug-and-play inference acceleration for large vision-language models. In *ECCV*, 2024a.

Xi Chen, Xiao Wang, Soravit Changpinyo, AJ Piergiovanni, Piotr Padlewski, Daniel Salz, Sebastian Goodman, Adam Grycner, Basil Mustafa, Lucas Beyer, Alexander Kolesnikov, Joan Puigcerver, Nan Ding, Keran Rong, Hassan Akbari, Gaurav Mishra, Linting Xue, Ashish V Thapliyal, James Bradbury, Weicheng Kuo, Mojtaba Seyedhosseini, Chao Jia, Burcu Karagol Ayan, Carlos Riquelme Ruiz, Andreas Peter Steiner, Anelia Angelova, Xiaohua Zhai, Neil Houlsby, and Radu Soricut. Pali: A jointly-scaled multilingual language-image model. In *ICLR*, 2023b.

Zhe Chen, Jiannan Wu, Wenhai Wang, Weijie Su, Guo Chen, Sen Xing, Muyan Zhong, Qinglong Zhang, Xizhou Zhu, Lewei Lu, Bin Li, Ping Luo, Tong Lu, Yu Qiao, and Jifeng Dai. Internvl: Scaling up vision foundation models and aligning for generic visual-linguistic tasks. In *CVPR*, 2024b.

Zhe Chen, Weiyun Wang, Yue Cao, Yangzhou Liu, Zhangwei Gao, Erfei Cui, Jinguo Zhu, Shenglong Ye, Hao Tian, Zhaoyang Liu, Lixin Gu, Xuehui Wang, Qingyun Li, Yiming Ren, Zixuan Chen, Jiapeng Luo, Jiahao Wang, Tan Jiang, Bo Wang, Conghui He, Botian Shi, Xingcheng Zhang, Han Lv, Yi Wang, Wenqi Shao, Pei Chu, Zhongying Tu, Tong He, Zhiyong Wu, Huipeng Deng, Jiaye Ge, Kai Chen, Kaipeng Zhang, Limin Wang, Min Dou, Lewei Lu, Xizhou Zhu, Tong Lu, Dahua Lin, Yu Qiao, Jifeng Dai, and Wenhai Wang. Expanding performance boundaries of open-source multimodal models with model, data, and test-time scaling. *ArXiv*, 2025b.

Tzu-Chun Chien, Chieh-Kai Lin, Shiang-Feng Tsai, Ruei-Chi Lai, Hung-Jen Chen, and Min Sun. Grounding-aware token pruning: Recovering from drastic performance drops in visual grounding caused by pruning. *ArXiv*, 2025.

Xiangxiang Chu, Limeng Qiao, Xinyu Zhang, Shuang Xu, Fei Wei, Yang Yang, Xiaofei Sun, Yiming Hu, Xinyang Lin, Bo Zhang, and Chunhua Shen. Mobilevlm v2: Faster and stronger baseline for vision language model. *ArXiv*, 2024.

Christopher Clark, Yue Yang, Jae Sung Park, Zixian Ma, Jieyu Zhang, Rohun Tripathi, Mohammadreza Salehi, Sangho Lee, Taira Anderson, Winson Han, and Ranjay Krishna. Molmopoint: Better pointing for vlms with grounding tokens. In *ECCV*, 2026a.

Christopher Clark, Jieyu Zhang, Zixian Ma, Jae Sung Park, Mohammadreza Salehi, Rohun Tripathi, Sangho Lee, Zhongzheng Ren, Chris Dongjoo Kim, Yinuo Yang, Vincent Shao, Yue Yang, Weikai Huang, Ziqi Gao, Taira Anderson, Jianrui Zhang, Jitesh Jain, George Stoica, Winson Han, Ali Farhadi, and Ranjay Krishna. Molmo2: Open weights and data for vision-language models with video understanding and grounding. In *CVPR*, 2026b.

Matt Deitke, Christopher Clark, Sangho Lee, Rohun Tripathi, Yue Yang, Jae Sung Park, Mohammadreza Salehi, Niklas Muennighoff, Kyle Lo, Luca Soldaini, Jiasen Lu, Taira Anderson, Erin Bransom, Kiana Ehsani, Huong Ngo, YenSung Chen, Ajay Patel, Mark Yatskar, Chris Callison-Burch, Andrew Head, Rose Hendrix, Favyen Bastani, Eli VanderBilt, Nathan Lambert, Yvonne Chou, Arnavi Chheda, Jenna Sparks, Sam Skjonsberg, Michael Schmitz, Aaron Sarnat, Byron Bischoff, Pete Walsh, Chris Newell, Piper Wolters, Tanmay Gupta, Kuo-Hao Zeng, Jon Borchardt, Dirk Groeneveld, Jen Dumas, Crystal Nam, Sophie Lebrecht, Caitlin Wittlif, Carissa Schoenick, Oscar Michel, Ranjay Krishna, Luca Weihs, Noah A. Smith, Hannaneh Hajishirzi, Ross Girshick, Ali Farhadi, and Aniruddha Kembhavi. Molmo and pixmo: Open weights and open data for state-of-the-art vision-language models. In *CVPR*, 2025.

Shivam Duggal, Sanghyun Byun, William T. Freeman, Antonio Torralba, and Phillip Isola. Single-pass adaptive image tokenization for minimum program search. In *NeurIPS*, 2025.

Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Efficient graph-based image segmentation. *IJCV*, 2004.

Jie Huang, Xuejing Liu, Sibo Song, RuiBing Hou, Hong Chang, Junyang Lin, and Shuai Bai. Revisiting multimodal positional encoding in vision-language models. In *ICLR*, 2026.

Peng Jin, Ryuichi Takanobu, Wancai Zhang, Xiaochun Cao, and Li Yuan. Chat-univi: Unified visual representation empowers large language models with image and video understanding. In *CVPR*, 2024.

Savya Khosla, Sethuraman TV, Barnett Lee, Alexander Schwing, and Derek Hoiem. Ren: Fast and efficient region encodings from patch-based image encoders. In *NeurIPS*, 2025.

Savya Khosla, Sethuraman TV, Aryan Chadha, Alexander Schwing, and Derek Hoiem. T-ren: Learning text-aligned region tokens improves dense vision-language alignment and scalability. In *ECCV*, 2026.

Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloé Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross B. Girshick. Segment anything. In *ICCV*, 2023.

Hugo Laurençon, Andrés Marafioti, Victor Sanh, and Léo Tronchon. Building and better understanding vision-language models: insights and future directions. *ArXiv*, 2024.

Hyun Lee, Hyemin Jeong, Yejin Kim, Hyungwook Choi, Hyunsoo Cho, Soo Kyung Kim, and Joonseok Lee. A more word-like image tokenization for mllms. In *CVPR*, 2026.

Jaihyun Lew, Soohyuk Jang, Jaehoon Lee, Seungryong Yoo, Eunji Kim, Saehyung Lee, Jisoo Mok, Siwon Kim, and Sungroh Yoon. Superpixel tokenization for vision transformers: Preserving semantic integrity in visual tokens. *ArXiv*, 2025.

Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: bootstrapping language-image pre-training with frozen image encoders and large language models. In *ICML*, 2023.

Wentong Li, Yuqian Yuan, Jian Liu, Dongqi Tang, Song Wang, Jie Qin, Jianke Zhu, and Lei Zhang. Tokenpacker: Efficient visual projector for multimodal llm. *IJCV*, 2025.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. In *NeurIPS*, 2023.

Chuofan Ma, Yi Jiang, Jiannan Wu, Zehuan Yuan, and Xiaojuan Qi. Groma: Localized visual tokenization for grounding multimodal large language models. In *ECCV*, 2024.

Junyuan Mao, Qiankun Li, Linghao Meng, Zhicheng He, Xinliang Zhou, Kun Wang, Yang Liu, and Yueming Jin. Granulon: Awakening pixel-level visual encoders with adaptive multi-granularity semantics for mllm. In *CVPR*, 2026.

Andrés Marafioti, Orr Zohar, Miquel Farré, Merve Noyan, Elie Bakouch, Pedro Cuenca, Cyril Zakka, Loubna Ben Allal, Anton Lozhkov, Nouamane Tazi, Vaibhav Srivastav, Joshua Lochner, Hugo Larcher, Mathieu Morlon, Lewis Tunstall, Leandro von Werra, and Thomas Wolf. Smolvlm: Redefining small and efficient multimodal models. *ArXiv*, 2025.

Yuzhang Shang, Mu Cai, Bingxin Xu, Yong Jae Lee, and Yan Yan. Llava-prumerge: Adaptive token reduction for efficient large multimodal models. In *ICCV*, 2025.

Michael Tschannen, Alexey Gritsenko, Xiao Wang, Muhammad Ferjad Naeem, Ibrahim Alabdulmohsin, Nikhil Parthasarathy, Talfan Evans, Lucas Beyer, Ye Xia, Basil Mustafa, Olivier Hénaff, Jeremiah Harmsen, Andreas Steiner, and Xiaohua Zhai. Siglip 2: Multilingual vision-language encoders with improved semantic understanding, localization, and dense features. *ArXiv*, 2025.

Mengyu Wang, Yuyao Huang, Henghui Ding, Xinlong Wang, Tiejun Huang, Yao Zhao, Yunchao Wei, and Shuicheng Yan. Region-native visual tokenization. In *ECCV*, 2024a.

Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Yang Fan, Kai Dang, Mengfei Du, Xuancheng Ren, Rui Men, Dayiheng Liu, Chang Zhou, Jingren Zhou, and Junyang Lin. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution. *ArXiv*, 2024b.

Zichen Wen, Yifeng Gao, Weijia Li, Conghui He, and Linfeng Zhang. Token pruning in multimodal large language models: Are we solving the right problem? In *Findings of ACL*, 2025a.

Zichen Wen, Yifeng Gao, Shaobo Wang, Junyuan Zhang, Qintong Zhang, Weijia Li, Conghui He, and Linfeng Zhang. Stop looking for important tokens in multimodal language models: Duplication matters more. In *EMNLP*, 2025b.

Shengqiong Wu, Hao Fei, Xiangtai Li, Jiayi Ji, Hanwang Zhang, Tat-Seng Chua, and Shuicheng Yan. Towards semantic equivalence of tokenization in multimodal llm. In *ICLR*, 2025.

Long Xing, Qidong Huang, Xiaoyi Dong, Jiajie Lu, Pan Zhang, Yuhang Zang, Yuhang Cao, Conghui He, Jiaqi Wang, Feng Wu, and Dahua Lin. Conical visual concentration for efficient large vision-language models. In *CVPR*, 2025.

Wilson Yan, Volodymyr Mnih, Aleksandra Faust, Matei Zaharia, Pieter Abbeel, and Hao Liu. Elastictok: Adaptive tokenization for image and video. In *ICLR*, 2025.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report. *ArXiv*, 2025a.

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 *CVPR*, 2025b.

Xiaobo Yang and Xiaojin Gong. Re-purposing sam into efficient visual projectors for mllm-based referring image segmentation. *TOMM*, 2025.

Linli Yao, Lei Li, Shuhuai Ren, Lean Wang, Yuanxin Liu, Xu Sun, and Lu Hou. Deco: Decoupling token compression from semantic abstraction in multimodal large language models. *ArXiv*, 2024.

Jiabo Ye, Anwen Hu, Haiyang Xu, Qinghao Ye, Ming Yan, Yuhao Dan, Chenlin Zhao, Guohai Xu, Chenliang Li, Junfeng Tian, Qian Qi, Ji Zhang, and Fei Huang. mplug-docowl: Modularized multimodal large language model for document understanding. *ArXiv*, 2023.

Qizhe Zhang, Aosong Cheng, Ming Lu, Renrui Zhang, Zhiyong Zhuo, Jiajun Cao, Shaobo Guo, Qi She, and Shanghang Zhang. Beyond text-visual attention: Exploiting visual cues for effective token pruning in vlms. In *ICCV*, 2025a.

Qizhe Zhang, Mengzhen Liu, Lichen Li, Ming Lu, Yuan Zhang, Junwen Pan, Qi She, and Shanghang Zhang. Beyond attention or similarity: Maximizing conditional diversity for token pruning in mllms. In *NeurIPS*, 2025b.

Yuan Zhang, Chun-Kai Fan, Junpeng Ma, Wenzhao Zheng, Tao Huang, Kuan Cheng, Denis Gudovskiy, Tomoyuki Okuno, Yohei Nakata, Kurt Keutzer, and Shanghang Zhang. Sparsevlm: Visual token sparsification for efficient vision-language model inference. In *ICML*, 2025c.

APPENDIX

We provide training and evaluation details in Section A; additional experiments on FH parameters, connector width, and resolution scaling in Section B; and a discussion of limitations in Section C.

A IMPLEMENTATION DETAILS

A.1 ARCHITECTURE

We adopt MolmoPoint (Clark et al., 2026a) as the basis for our patch baseline (PVLM) and the starting architecture for RVLM, motivated by its strong performance among similarly sized open-weight VLMs (Table 1). We initialize the LLM from Qwen3-4B-Instruct (Yang et al., 2025a) and the ViT from SigLIP2-SO400M/14 (Tschannen et al., 2025). The LLM has a hidden dimension of 2560, 36 layers, 32 attention heads, and 8 KV heads. The vision encoder processes 378-pixel inputs with a patch size of 14. The connector concatenates ViT features from layers -3 and -9, then applies 16-head cross-attention using the mean of each patch group as the query and its constituent features as keys and values. The pooled representation is projected to the LLM embedding space through an MLP. We use adaptive overlapping crops with crop margins of $[4, 4]$.

A.2 TRAINING RECIPE

We train RVLM in two stages, jointly optimizing the vision encoder, connector, and LLM throughout. Stage 1 performs multimodal pretraining for $31{,}971$ steps. Stage 2 initializes from the Stage-1 checkpoint and performs supervised fine-tuning for another $24{,}000$ steps. Table 5 summarizes the training configuration, and Table 6 shows the data mixtures. The training recipe is adopted from concurrent work in preparation and is not a contribution of this paper.

A.3 TASK FORMULATION

We summarize how the model performs each task. The pointing formulation follows MolmoPoint (Clark et al., 2026a), while the segmentation and point-to-depth formulations are adopted from concurrent work in preparation and are not contributions of this paper.

Image QA and captioning. We formulate image question answering and captioning as autoregressive text generation. Given an image and a textual prompt, the model conditions on the region-token representation of the image and generates an answer or caption using a standard next-token objective. No task-specific prediction head or output representation is used for these tasks.

Pointing and counting. We represent an image location using a coarse-to-fine hierarchy. For each point, the model first selects a region token, then a ViT patch assigned to that region, and finally one of 4×4 spatial bins within the selected patch. The center of the predicted bin is mapped back to image coordinates using the retained patch-to-region assignment. These decisions are generated autoregressively using dedicated classification heads, with cross-entropy supervision at each level. A learned end-of-points class terminates the point sequence. For counting, the model predicts one point for each instance that matches the query using the same grounding representation. It subsequently generates the cardinality of the point set as text. Thus, counting is grounded in an explicit set of localized instances rather than being performed solely through direct scalar regression.

Segmentation. For referring segmentation, the model first selects an anchor region for the target mask, supervised using the regions with maximal target-mask coverage. Conditioned on this anchor, it predicts mask logits at three levels: regions, patches within each region, and 4×4 bins within each patch. The three scores are summed for each bin and mapped back to the image using the retained patch-to-region assignment. The resulting mask probabilities are bilinearly resized to the original image resolution and thresholded at 0.5. Multiple masks are generated by repeating the anchor-and-mask prediction sequence. Training uses region-selection supervision together with focal and Dice losses for the dense masks.

Point-to-depth. Given an image and one or more queried locations, the model predicts the depth at each point. Query points are encoded in the prompt, and the output contains a corresponding

Table 5: **Training hyperparameters for RVLM.** Both stages jointly train the vision encoder, connector, and LLM. Stage 2 initializes from the final Stage-1 checkpoint.

Setting	Stage 1	Stage 2
Training steps	31,971	24,000
Global / microbatch size	128 / 8	128 / 8
Packed sequence length	3072	3072
Maximum training crops	14	14
Maximum packed images	28	28
Optimizer	AdamW	AdamW
β_1, β_2	$(0.9, 0.95)$	$(0.9, 0.95)$
ε	10^{-6}	10^{-6}
Weight decay	0	0
ViT learning rate	1×10^{-5}	2×10^{-5}
LLM learning rate	2×10^{-5}	2×10^{-5}
Connector learning rate	2×10^{-4}	2×10^{-5}
ViT warmup	2000	200
LLM warmup	2000	200
Connector warmup	200	200
LR schedule	Cosine	Cosine
Final LR fraction	0.1	0.1
Gradient clipping	1.0	1.0
Hardware	16× NVIDIA B300	16× NVIDIA B300
Distributed training	FSDP	FSDP
Precision	bf16	bf16

coarse-to-fine point representation for each query, followed by a measurability indicator and, when measurable, its depth in meters. Depth is generated as text to a precision of two decimal places.

A.4 Evaluation

We evaluate on 35 tasks spanning image QA (including document/OCR QA), pointing, counting, referring segmentation, and point-to-depth. Table 7 lists each dataset, the metric reported in the results, and a one-line task description. Unless noted, scores are percentages, with higher scores indicating better performance.

B Additional Experiments

B.1 Ablating FH Segmentation Parameters

We study how the scale (k), the minimum region size ($\mathrm{min_region_size}$), and the working resolution ($\mathrm{max_side}$) of Felzenszwalb-Huttenlocher (FH) hyperparameters affect downstream accuracy and visual token count. All runs use the same RVLM SFT checkpoint, only inference-time FH settings change. We evaluate a six-task subset spanning general VQA, document OCR, pointing, counting, referring segmentation, and depth. The results are shown in Table 8.

General VQA and depth are robust. A-OKVQA DA stays within 74.8-75.5 and NYU $\delta_{1.25}$ within 97.4-97.6 across all settings. For these tasks, FH mainly changes token compression, not accuracy.

Text-rich understanding benefits from finer segmentation. Coarser segmentation hurts document OCR. Raising k from 100 to 400 drops InfoVQA by 2.2 points, and raising $\mathrm{min_size}$ from 32 to 128 drops it by 3.6. Conversely, a higher FH working resolution helps, with score reaching 75.2 at $\mathrm{max_side}$=1024.

Grounding degrades gently under coarsening. PixMo-Point, PixMo-Count, and RefCOCO typically move by $\leq$1.5 points when regions become coarser or the FH working resolution reduces.

Table 6: **Training data mixtures.** Patch and region models share these mixtures.

Task family	Datasets
Stage 1 (multimodal pre-training)	
Captioning	PixMo-Cap
OCR	olmOCR-mix
Pointing / counting	PixMo-Points, PixMo-Count, CoSyn-point, COCO, LVIS
Segmentation	COCO, LVIS
Depth	SUN RGB-D, NYU Depth v2, DIODE, HM3D, DDAD, Waymo, Argoverse 2, nuScenes, Aria Digital Twin, Aria Everyday Activities, Aria Nymeria, EgoExo4D, Stereo4D
Camera	Stereo4D, SUN RGB-D, NYU Depth v2, EgoExo4D, DDAD, DIODE, nuScenes, Argoverse 2, Waymo, ScanNet++, ARKitScenes, iBims-1, Aria Nymeria
Text-only NLP	Tulu-4
Stage 2 (supervised fine-tuning)	
Image QA	VQAv2, A-OKVQA (MC and DA), OK-VQA, ScienceQA, AI2D, TallyQA
Captioning	PixMo-Cap, PixMo-CapQA, PixMo-AskModelAnything
OCR	TextVQA, ChartQA, DocVQA, InfoVQA, ST-VQA, TabMWP
Pointing / counting	PixMo-Points, PixMo-Count, CoSyn-point
Segmentation	PixMo-Points, COCO, LVIS, EntitySeg
Depth	SUN RGB-D, NYU Depth v2, DIODE, HM3D, DDAD, Waymo, Argoverse 2, nuScenes, Aria Digital Twin, Aria Everyday Activities, Aria Nymeria, EgoExo4D, Stereo4D
Text-only NLP	Tulu-4

Aggressive compression (k=400 or $\mathrm{max_side}$=384) trades a small grounding loss for substantially fewer tokens.

Tokens track FH granularity. Mean visual tokens fall from 1171 at k=50 to 686 at k=400, and from 1032 at $\mathrm{min_size}$=16 to 736 at 128. Increasing $\mathrm{max_side}$ raises tokens mainly on high-resolution OCR/pointing images (InfoVQA: 1053→1669 from 384→1024), while smaller natural images saturate once $\mathrm{max_side}$ exceeds their native size.

B.2 Connector Width Ablation

The region connector aggregates at most w ViT patches into each region token. Regions containing more than w patches are split into multiple connector rows, while shorter rows are padded to width w. Increasing w can therefore reduce the number of visual tokens passed to the LLM, but also increases the size of the padded connector input. We analyze this structural trade-off to motivate our default choice of $w = 16$. Specifically, we sweep $w \in \{4, 8, 16, 32, 64, 128\}$ while keeping the FH configuration fixed at $(k, \mathrm{min_size}, \mathrm{max_side}) = (100, 32, 768)$. Our stratified sample contains 1,480 images from A-OKVQA DA, InfoVQA, PixMo-Point, PixMo-Count, RefCOCO, and NYU Depth v2. Table 9 summarizes our structural analysis.

Most regions fit within w=16. At w=4, 31.1% of FH regions require splitting, compared to only 7.8% at w=16 (Split in Table 9). Thus, the default width accommodates 92.2% of regions without splitting, leaving limited room for further reductions in visual tokens.

Wider connectors introduce substantial padding. The additional token savings come at the cost of a larger padded connector input. Increasing w from 16 to 128 reduces visual tokens by only 19.3%, while increasing the padding fraction from 62.2% to 94.1%. Consequently, the estimated bf16 connector input memory grows from 68.9 to 444.8 MiB, a 6.5× increase. This trade-off motivates $w = 16$ as our default connector width.

Visual-token reduction saturates near w=16. Table 9 shows that increasing w from 4 to 16 reduces the mean visual-token count from 1,814 to 980, a 46.0% reduction. This captures 81.5% of

Table 7: **Evaluation suite**. Metrics match those reported in the main native-resolution comparison.

Dataset	Metric	Task
	Image QA	
A-OKVQA (DA)	VQA score	Open-ended knowledge-aware visual questions (direct answer).
A-OKVQA (MC)	Accuracy	Multiple-choice knowledge-aware visual questions.
AI2D	Accuracy (transparent)	Diagram science QA with multiple-choice answers.
OK-VQA	VQA score	Open-ended visual questions requiring outside knowledge.
ScienceQA	Accuracy	Multimodal science questions with images (image subset).
VQAv2	VQA score	Open-ended natural-image visual question answering.
ChartQA	Relaxed accuracy	Questions over charts and plots.
DocVQA	ANLS	Questions over document images / scanned pages.
InfoVQA	ANLS	Questions over infographics.
TextVQA	VQA score	Questions that require reading text in natural images.
	Pointing	
OSWorld-G	Overall accuracy	GUI grounding: point to UI elements from natural-language instructions.
PixMo-Points	F1	Open-vocabulary pointing to all instances of a described object.
PointBench	Average accuracy	Multi-skill pointing (spatial, counting, reasoning, steerable, etc.).
SA-Co Gold	All-frequency F1	Long-tail concept pointing on SA-Co Gold (pointing split).
ScreenSpot Pro	Overall accuracy	Professional desktop GUI grounding across application categories.
ScreenSpot v2	Average accuracy	Mobile / desktop / web UI element grounding.
	Counting	
CountBenchQA	Exact-count accuracy	Count objects described by a natural-language question.
PixMo-Count	Exact-count accuracy	Count instances of a queried category in natural images.
	Segmentation	
PixMo-Seg	CGF1	Open-vocabulary referring segmentation (mask prediction).
RefCOCO	Overall IoU	Referring expression segmentation on COCO (RefCOCO).
RefCOCO+	Overall IoU	Referring expression segmentation without location words.
RefCOCOg	Overall IoU	Referring expression segmentation with longer expressions.
SA-Co Gold	Mean CGF1 (group average)	Long-tail concept segmentation on SA-Co Gold (segmentation split).
	Point-to-Depth	
Argoverse 2	Scale-invariant $\delta_{1.25}$	Estimate depth at queried 2D points in autonomous-driving scenes.
Aria AEA	Scale-invariant $\delta_{1.25}$	Point-to-depth on Aria Everyday Activities egocentric video.
Aria Nymeria	Scale-invariant $\delta_{1.25}$	Point-to-depth on Aria Nymeria egocentric recordings.
DDAD	Scale-invariant $\delta_{1.25}$	Point-to-depth on Dense Depth for Autonomous Driving.
DIODE	Scale-invariant $\delta_{1.25}$	Point-to-depth on Dense Indoor/Outdoor DEpth.
EgoExo4D (ego)	Scale-invariant $\delta_{1.25}$	Point-to-depth from EgoExo4D egocentric cameras.
EgoExo4D (exo)	Scale-invariant $\delta_{1.25}$	Point-to-depth from EgoExo4D exocentric cameras.
nuScenes	Scale-invariant $\delta_{1.25}$	Point-to-depth in nuScenes driving scenes.
NYU Depth v2	Scale-invariant $\delta_{1.25}$	Point-to-depth on indoor RGB-D scenes.
Stereo4D	Scale-invariant $\delta_{1.25}$	Point-to-depth on internet stereo / video-derived depth.
SUN RGB-D	Scale-invariant $\delta_{1.25}$	Point-to-depth on SUN RGB-D indoor scenes.
Waymo	Scale-invariant $\delta_{1.25}$	Point-to-depth on Waymo Open Dataset driving scenes.

the total token reduction observed between $w = 4$ and $w = 128$. Increasing the width beyond 16 yields diminishing returns.

B.3 Resolution Scaling

Performance across image resolutions. Table 10 breaks down the native-resolution evaluation into five image-size ranges. At lower resolutions, RVLM and PVLM perform comparably, but the gap widens as image size increases. In the smallest range (0.14-0.41 MP), both models achieve an example-weighted score of 85.3, with RVLM using 4.5% fewer visual tokens. In the largest range (2.08-2.63 MP), RVLM improves performance by 5.3 points while reducing the average token count by 51.8% reduction. The family-balanced gain also grows from 0.3 to 9.0 points across these ranges.

The advantage is particularly pronounced on spatial tasks. In the largest resolution range, RVLM improves the task averages by 5.6 points on pointing, 25.9 points on counting, 8.7 points on segmentation, and 5.5 points on depth estimation. Image QA remains broadly comparable. These results suggest that region tokens make better use of high-resolution visual information, particularly when predictions depend on fine spatial details. Since each resolution range contains a different set of task examples, absolute scores should not be compared directly across ranges; the relevant comparison is between PVLM and RVLM within each range.

Table 8: **FH parameter ablation.** Default settings are highlighted in blue. Token count is the mean visual-token count over the six tasks. NYUv2 reports scale-invariant δ_1 using per-image median scale alignment.

Setting	A-OKVQA	InfoVQA	PixMo-Point	PixMo-Count	RefCOCO	NYUv2	Token count
Scale k ($\mathbf{min_size = 32}$, $\mathbf{max_side = 768}$)							
k=50	75.5	74.3	86.6	91.9	82.4	97.6	1171
k=100	74.8	74.3	86.4	90.4	82.5	97.5	963
k=200	75.4	72.8	85.8	90.9	82.2	97.5	794
k=400	75.1	72.1	85.1	89.6	82.0	97.4	686
Minimum region size ($k = 100$, $\mathbf{max_side = 768}$)							
min_size=16	74.9	74.9	86.5	90.9	82.4	97.4	1032
min_size=32	74.8	74.3	86.4	90.4	82.5	97.5	963
min_size=64	75.5	72.2	85.8	90.6	82.4	97.4	860
min_size=128	75.2	70.7	85.9	91.3	82.3	97.4	736
FH working resolution ($k = 100$, $\mathbf{min_size = 32}$)							
max_side=384	75.1	71.0	85.1	90.4	82.4	97.4	746
max_side=512	75.0	71.6	85.9	91.1	82.5	97.4	830
max_side=768	74.8	74.3	86.4	90.4	82.5	97.5	963
max_side=1024	74.9	75.2	86.7	90.7	82.5	97.5	1060

Table 9: **Structural effect of connector width w.** Statistics are pooled over 1,480 images from six tasks. Input memory estimates the storage of the padded bf16 connector input for one image; it excludes model parameters and intermediate activations.

w	Visual tokens	Patches/token	Split %	Padding	Input memory (MiB)
4	1,814	3.21	31.1%	18.4%	31.9
8	1,243	4.96	16.5%	40.4%	43.7
16	980	6.63	7.8%	62.2%	68.9
32	863	7.79	3.5%	78.5%	121.3
64	812	8.43	1.5%	88.6%	228.4
128	791	8.73	0.7%	94.1%	444.8

Inference cost across image resolutions. Table 11 measures inference cost as the number of high-resolution crops increases. At one and four crops, PVLM is slightly faster at batch size 1, where the savings from shorter visual sequences do not yet offset the overhead of region pooling. At batch size 8, the models have nearly identical latency at four crops. From eight crops onward, RVLM achieves lower latency and peak memory at both batch sizes.

The efficiency gap continues to widen with resolution. At 32 crops and batch size 1, RVLM reduces generation latency from 162.1 to 105.0 ms. At batch size 8, latency falls from 144.6 to 83.7 ms, while peak GPU memory decreases from 39.57 to 30.15 GiB. These trends reflect the growing benefit of reducing the visual sequence processed by the LLM as image resolution increases, even though the vision encoder still processes the full patch grid.

C Limitations

Fine-grained visual content. RVLM underperforms PVLM on InfoVQA (-4.9 points), ScreenSpot Pro (-3.3), and ChartQA (-0.8). These tasks often require distinguishing very small text or interface elements, which may be absorbed into surrounding background regions during segmentation or patch assignment. Although complete patch coverage is retained, pooling can obscure these fine details. Our FH ablations (Appendix B.1) show that text-rich tasks are sensitive to region granularity, suggesting that adjusting min_size, working resolution, or other FH parameters could mitigate these failures. We use a fixed configuration across tasks to develop a general-purpose VLM that balances

Table 10: **PVLM vs. RVLM across five native-resolution ranges.** Task metrics are computed over all examples within each resolution range. A dash (–) indicates that the dataset contains no images in that range.

Task	0.14–0.41 MP		0.55–0.76 MP		0.83–1.38 MP		1.55–2.01 MP		2.08–2.63 MP	
	PVLM	RVLM	PVLM	RVLM	PVLM	RVLM	PVLM	RVLM	PVLM	RVLM
All tasks										
Example-weighted average	85.3	85.3 (0.0)	69.8	70.2 (↑0.4)	72.6	73.8 (↑1.2)	69.0	73.6 (↑4.6)	68.4	73.7 (↑5.3)
Family-balanced average	85.0	85.3 (↑0.3)	78.5	79.4 (↑0.9)	76.9	77.9 (↑1.0)	76.5	81.9 (↑5.4)	70.4	79.4 (↑9.0)
Token count	691	660 (↓4.5%)	987	797 (↓19.3%)	1,608	1,076 (↓33.1%)	2,499	1,358 (↓45.6%)	3,277	1,581 (↓51.8%)
Image QA										
A-OKVQA DA	75.5	74.8	–	–	–	–	–	–	–	–
A-OKVQA MC	88.6	88.2	–	–	–	–	–	–	–	–
AI2D	94.9	94.5	96.7	94.8	95.0	98.3	96.2	97.4	96.0	96.0
ChartQA	67.8	61.9	84.7	84.2	85.0	84.4	100.0	100.0	100.0	100.0
DocVQA	97.2	99.3	95.8	92.6	89.8	92.3	94.4	90.4	93.2	92.9
OK-VQA	65.8	66.0	–	–	–	–	–	–	–	–
ScienceQA	98.4	98.0	97.6	97.3	–	–	–	–	–	–
TextVQA	94.1	96.1	85.4	83.0	85.5	85.1	–	–	80.0	80.0
VQAv2	87.2	87.1	–	–	–	–	–	–	–	–
Task average	86.8	86.7	88.7	87.8	86.0	85.8	95.1	92.9	93.2	92.9
Token count	689	662	984	612	1,614	1,098	2,483	1,260	3,359	1,636
Pointing										
OSWorld-G	–	–	–	–	71.7	73.8	–	–	43.6	43.3
PixMo Point	81.2	81.6	90.6	89.5	86.0	86.6	83.1	87.3	88.1	94.0
PointBench	71.0	71.6	56.7	56.9	68.1	65.2	62.5	64.9	59.8	58.7
SA-Co Gold	91.2	91.6	88.6	89.3	90.4	91.5	87.6	90.6	78.3	87.4
ScreenSpot Pro	–	–	–	–	–	–	–	–	29.3	26.0
ScreenSpot v2	33.3	33.3	30.4	30.7	31.7	31.7	30.6	28.0	79.8	82.8
Task average	90.4	90.8	86.9	87.5	88.9	89.8	83.7	86.7	68.5	74.1
Token count	719	596	990	838	1,620	1,080	2,497	1,380	3,279	1,544
Counting										
CountBenchQA	90.8	91.8	87.2	87.2	94.9	94.9	100.0	100.0	61.5	92.3
PixMo Count	91.9	91.9	84.4	93.3	89.9	91.3	74.5	90.9	61.9	87.3
Task average	90.9	91.8	85.7	90.5	91.7	92.6	75.3	91.1	61.9	87.8
Token count	552	500	1,004	855	1,649	1,129	2,729	1,433	3,354	1,611
Segmentation										
PixMo Seg	53.9	52.5	67.8	65.0	57.1	58.1	58.9	65.9	48.6	70.8
RefCOCO	82.0	82.5	–	–	–	–	–	–	–	–
RefCOCO+	78.5	78.7	–	–	–	–	–	–	–	–
RefCOCOg	80.3	81.1	–	–	–	–	–	–	–	–
SA-Co Gold	42.9	43.1	40.9	41.9	46.0	48.4	41.3	46.7	47.6	56.2
Task average	64.3	64.6	41.9	42.7	46.5	48.9	43.9	49.6	47.6	56.3
Token count	709	669	987	843	1,610	1,077	2,499	1,386	3,209	1,596
Depth										
Argoverse 2	–	–	–	–	–	–	–	–	77.9	81.8
Aria AEA	–	–	–	–	68.5	69.5	–	–	–	–
Aria Nymeria	–	–	–	–	54.6	55.6	–	–	–	–
DDAD	–	–	–	–	–	–	–	–	81.7	87.7
DIODE	–	–	–	–	92.2	94.0	–	–	–	–
EgoExo4D Ego	89.1	88.9	–	–	–	–	–	–	–	–
EgoExo4D Exo	–	–	84.0	82.9	–	–	–	–	–	–
nuScenes	–	–	–	–	–	–	84.7	88.8	–	–
NYU Depth v2	97.8	97.5	–	–	–	–	–	–	–	–
Stereo4D	89.0	88.6	–	–	–	–	–	–	–	–
SUN RGB-D	97.5	97.7	99.0	98.8	–	–	–	–	–	–
Waymo	–	–	–	–	–	–	84.2	89.8	83.9	91.9
Task average	92.8	92.6	89.2	88.5	71.4	72.7	84.5	89.1	80.6	86.1
Token count	634	542	955	734	1,372	821	2,321	1,125	3,254	1,533

performance and token efficiency. Adapting segmentation granularity to individual images or task requirements may further improve this trade-off but remains unexplored.

Architectural scope. We develop and evaluate region tokenization within the MolmoPoint architecture (Clark et al., 2026a). Although the underlying approach is not specific to this model, we have not established its effectiveness across other VLM architectures. In particular, connectors that concatenate a fixed number of patch embeddings along the feature dimension cannot directly accommodate variable-sized regions without modification. Extending region tokenization to other connector designs and to video remains future work.

Table 11: **Inference cost across image resolutions.** Generation latency, end-to-end (E2E) latency, and peak GPU memory for PVLM and RVLM at batch sizes 1 and 8. Measurements use 512 randomly sampled PixMo-Cap examples on a single NVIDIA H100, with 64 warm-up batches and five runs. Generation is capped at one output token. Latencies are reported as mean ± standard deviation. While PVLM is slightly faster at low crop counts, RVLM achieves increasing latency and memory savings as image resolution grows.

Crops	Generation latency (ms)		E2E latency (ms)		Peak memory (GiB)	
	PVLM	RVLM	PVLM	RVLM	PVLM	RVLM
Batch size 1						
1	37.9±1.5	40.3±1.9	38.6±1.5	41.1±1.9	19.23	19.43
4	45.4±1.1	48.5±1.6	46.2±1.1	49.4±1.6	19.45	19.70
8	61.7±0.8	55.2±1.2	63.0±0.8	56.1±1.2	19.76	19.64
12	76.7±0.8	63.9±1.6	77.8±0.8	64.9±1.6	20.09	19.79
16	92.0±0.9	73.8±1.4	93.0±0.9	74.9±1.4	20.36	19.96
24	127.8±1.2	88.9±1.3	129.0±1.2	90.1±1.3	20.98	19.95
32	162.1±1.4	105.0±1.3	163.3±1.4	106.2±1.3	21.60	20.02
Batch size 8						
1	13.3±0.0	14.5±0.1	13.4±0.0	14.7±0.1	20.86	22.00
4	24.5±0.1	24.4±0.1	24.7±0.1	24.6±0.1	22.63	24.20
8	39.2±0.1	30.8±0.1	39.4±0.1	31.0±0.1	25.03	23.72
12	55.1±0.1	40.9±0.1	55.3±0.1	41.2±0.1	27.44	24.98
16	71.2±0.2	50.9±0.1	71.5±0.2	51.2±0.2	29.86	26.30
24	106.4±0.5	65.8±0.2	106.8±0.5	66.2±0.3	34.70	26.94
32	144.6±0.4	83.7±0.4	145.0±0.3	84.2±0.4	39.57	30.15

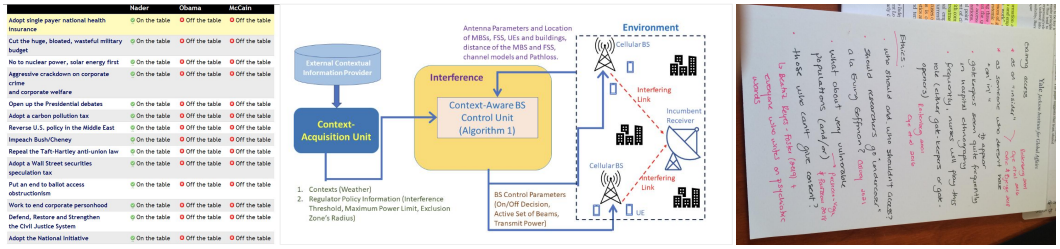


Figure 5: **Failure cases of T-REN-based RVLM.** During VLM training, region tokens become spatially disconnected, making it difficult to associate text and visual elements with their correct locations. The model recognizes individual labels but misattributes them to neighboring rows, columns, or objects, producing incorrect descriptions of the image structure. For instance, in the first example, it incorrectly describes some initiatives as "on the table" for Obama, even though the image lists none. PVLM and FH-based RVLM do not show these failures.

Dependence on photometric boundaries. FH constructs regions from local color differences, which may not align with semantic boundaries. This limitation may be particularly relevant for low-contrast or degraded imagery, including fog, camouflage, and motion blur, where meaningful structures have weak visual boundaries. We do not explicitly evaluate these conditions. Our limit of $w = 16$ patches per region bounds the amount of content pooled into a single token, but it cannot recover boundaries missed during region construction.

Connector width trade-off. We choose w=16 to balance visual-token reduction against connector padding. Although larger widths further shorten the LLM sequence, our structural ablation shows that they substantially increase the padded connector input (Appendix B.2). The chosen width, therefore, represents a practical operating point rather than a universally optimal setting. More efficient handling of variable-sized regions may improve this trade-off.