OV-Stitcher: A Global Context-Aware Framework for Training-Free Open-Vocabulary Semantic Segmentation

Supplementary Material

A. Additional Results on Varying Resolutions.

To complement the results presented in the main paper, we provide additional evaluations on the same resolution settings using both graphical summaries (Fig. 8) and numerical results (Tab. 4). In addition to VOC21 [16], Context60 [38], and COCO Object [49], this section includes results on ADE20K [65] and COCO Stuff [5]. Following the same setup, we fix the window size and stride to 336 and 112, respectively, and vary the shorter side from 336 to 448, 560, and 672, increasing the number of sub-image crops. Under identical conditions, we compare OV-Stitcher with CorrCLIP [61] across all datasets.

As shown in Tab. 4 and Fig. 8, OV-Stitcher consistently shows smaller drops in performance compared to CorrCLIP as resolution increases. While CorrCLIP generally achieves its best results at the lowest resolution across datasets, OV-Stitcher attains its peak performance at higher resolutions, demonstrating the effectiveness of the stitching mechanism in leveraging high-resolution inputs.

B. Computational Analysis.

While computational efficiency is not the primary focus of our method, Stitch Attention introduces full token-to-token interaction across sub-images, which makes it necessary to examine how inference cost scales with input size. Our proposed Stitch Attention enables attention over all tokens across sub-images, allowing the model to capture global context effectively. However, this also introduces a dependency of inference cost on both input resolution and the total number of tokens, which varies per image. Therefore, we evaluate the computational cost at fixed resolutions of 336×336, 448×448, and 560×560. For these experiments, we adopt a sliding-window configuration with a window size of 336 and a stride of 112, allowing us to measure how inference cost increases sequentially with input size.

As expected, higher resolutions lead to a larger number of tokens and consequently higher computation, as shown in Table 5. In particular, increasing the resolution results in more crops being processed, which further amplifies the computational load. Nonetheless, our method effectively leverages higher-resolution inputs to yield improved segmentation performance, as reported in Table 4, making this additional computation a reasonable trade-off and highlighting the advantage of maintaining global interactions even at large input scales.

Moreover, to examine the practical feasibility of our approach, we apply Flash Attention[13] to the Stitch Attention module by replacing the attention computation, while keeping the rest of the framework unchanged. As shown in Table 6, this consistently reduces both latency and peak memory across all resolutions, with larger gains at higher resolutions (e.g., 52.0% latency and 77.7% memory reduction at 672×672). These results indicate that the additional cost

Resolution	V21	C60	Obj.	Stf.	ADE	avg.
OV-Stitcher						
$336^{(336-112)}$	71.74	41.81	40.80	30.14	22.69	41.43
$448^{(336-112)}$	73.13	**42.42**	**41.42**	**30.76**	23.61	**42.27**
$560^{(336-112)}$	**73.15**	42.40	40.79	30.40	23.66	42.08
$672^{(336-112)}$	72.19	41.75	39.06	29.50	**23.67**	41.23
CorrCLIP						
$336^{(336-112)}$	**69.19**	40.01	**39.80**	**29.59**	21.74	**40.07**
$448^{(336-112)}$	68.66	**40.09**	39.08	29.56	22.40	39.96
$560^{(336-112)}$	66.38	39.23	37.71	28.88	**22.43**	38.93
$672^{(336-112)}$	62.44	37.52	35.29	27.68	21.99	36.98

Table 4. **Ablation on resolution robustness.** Comparison between CorrCLIP and WeaveCLIP under varying input resolutions without post-processing to clearly show the effect of the proposed framework. Each resolution is denoted as **shorter side**$^{(\text{window size - stride})}$.

Input Res.	# Crops	# Params. (M)	Mem. (MB)	Thru. (img/sec)
Precomputed Masks				
336 × 336	1	235	1435	6.98
448 × 448	4	235	1450	4.72
560 × 560	9	235	2040	3.12
672 × 672	16	235	3198	2.12
Masks Generated On-the-Fly				
336 × 336	1	458	2627	1.58
448 × 448	4	458	2651	1.47
560 × 560	9	458	2691	1.25
672 × 672	16	458	3717	1.03

Table 5. **Computational costs on RTX 4090 with FP16.** We separate cases where SAM2 masks for highlighting the attention map and post-processing are precomputed from those where they are generated on-the-fly.

Input Res.	**Naive Ver.** Stitch Attention		**Flash Ver.** Stitch Attention	
	Latency (ms)	Memory (MB)	Latency (ms)	Memory (MB)
336×336	0.25	293	0.21 (16.0% ↓)	220 (24.9% ↓)
448×448	0.72	454	0.44 (38.9% ↓)	241 (46.9% ↓)
560×560	1.65	799	0.81 (50.9% ↓)	273 (65.8% ↓)
672×672	3.08	1423	1.48 (52.0% ↓)	317 (77.7% ↓)

Table 6. **Computational costs on RTX 4090 with FP16.** Latency and peak CUDA memory of StitchAttention with naive attention and Flash Attention at different resolutions.

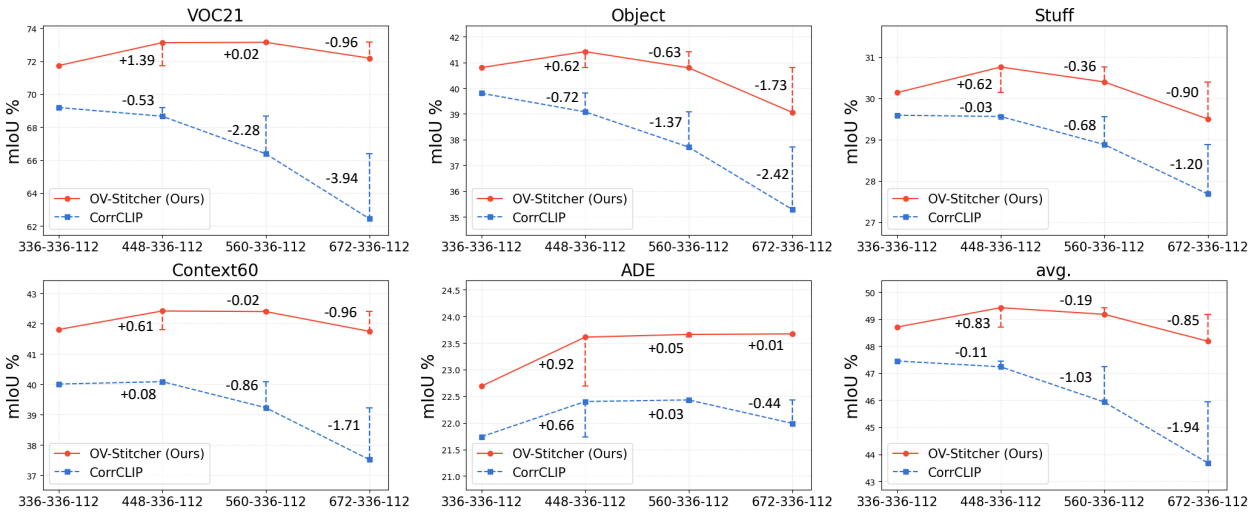

Figure 8. **Ablation on resolution robustness.** Post-processing is excluded to clearly show the effect of the proposed framework. The x-axis represents the settings in the format **shorter side – window size – stride**.

introduced by global token interactions can be effectively mitigated using standard efficient attention implementations, supporting the practical applicability of our method.

To generate Class-Biased Prompts, we employ a Large Language Model (LLM). Empirically, generating 15 descriptions per class required an average of approximately 5 seconds, which can pose a computational burden. However, this computational burden can be alleviated either by reducing the number of descriptions per class or by precomputing prompts—even if this slightly deviates from the fully open-vocabulary scope of OVSS—for a massive set of classes with extensive vocabularies (e.g., ImageNet-21K [42], Open Images Dataset [29]).

C. Evaluation on Diverse CLIP Variants.

We additionally evaluate our method using various CLIP backbones, including OpenCLIP [36], MetaCLIP [21], and DFNCLIP [17] with both ViT-B/16 and ViT-L/14. As shown in Tab. 7, results vary noticeably across CLIP variants: MetaCLIP still delivers the highest overall performance, but its improvement is less pronounced compared to the substantial jump observed from the Base models, whereas OpenCLIP and DFNCLIP exhibit more moderate gains across datasets.

Interestingly, higher zero-shot classification accuracy does not necessarily translate into stronger segmentation performance, likely because segmentation relies more on spatial detail and local region consistency than on the global semantic discrimination emphasized during CLIP pretraining. Since our method directly leverages CLIP's value features, it would be valuable for future work to explore how these value representations could retain or enhance spatial information, potentially improving segmentation robustness across diverse datasets.

Type	Size	Acc.	V21	C60	Obj.	Stf.	City	ADE
OpenCLIP	ViT-B/16	70.2%	72.92	40.54	42.27	31.20	51.07	28.01
MetaCLIP		72.1%	**76.37**	**43.92**	**44.59**	**32.10**	**52.26**	**27.80**
DFNCLIP		76.2%	72.26	41.82	43.93	32.16	52.09	27.78
OpenCLIP	ViT-L/14	75.3%	73.10	41.55	42.16	31.02	52.82	28.10
MetaCLIP		79.2%	**76.47**	**45.50**	**49.75**	**34.47**	**53.01**	**30.59**
DFNCLIP		81.4%	74.58	43.39	43.13	33.77	51.77	28.63

Table 7. **Comparison of different CLIP variants used as vision–language backbones.** Acc. denotes the zero-shot classification accuracy of each CLIP model on ImageNet-1K [15]

Type	Size	V21	C60	Obj.	Stf.	City	ADE
DINO	ViT-S/8	**75.84**	43.72	**42.63**	**31.86**	47.05	24.61
	ViT-B/8	75.72	**43.85**	42.55	31.83	**48.06**	**24.72**
	ViT-B/16	75.47	43.68	41.77	31.69	46.58	24.26
DINOv2	ViT-B/14	75.42	43.41	42.14	31.53	45.23	24.31
	ViT-L/14	75.28	43.21	42.54	31.45	44.14	24.36

Table 8. **Evaluation of various feature extractors.**

D. Evaluation on Diverse Featrue Extractor.

We evaluate our approach using a range of self-supervised feature extractors, including DINO [6] and DINOv2 [39] variants with different backbone sizes and patch resolutions. As shown in Tab. 8, models with smaller patch sizes (e.g., ViT-S/8 and ViT-B/8) consistently deliver higher segmentation quality than architectures with larger patch sizes, even when those models are stronger or larger overall. This highlights the importance of preserving detailed spatial information, which is more naturally retained with finer patch granularity.

Although DINO-B/8 slightly outperforms its smaller counterpart, DINO-S/8, the gap remains relatively modest. Considering the increased computational cost of larger mod-

els, this suggests a practical trade-off: lightweight models with small patch sizes—such as DINO-S/8—can offer competitive segmentation performance while improving inference speed and reducing memory consumption.

E. Class-Biased Prompts Construction.

To obtain class-biased prompts, we used an LLM to generate fine-grained visual descriptions tailored to each category. In addition to conventional ImageNet-style templates (e.g., "a photo of {class}"), we designed a set of instructions that guide the model to produce diverse, visually grounded sentences highlighting typical and distinctive attributes of the target class.

The LLM was instructed to produce 15 concise descriptions (5–15 words) for each class, highlighting features such as shape, surface appearance, material, structural components, and typical visual contexts, and to describe each class from multiple visual perspectives—for example, by emphasizing form, texture, surrounding environment, or characteristic parts.

A simplified version of the instruction used is:

- *"Generate 15 concise visual descriptions of a* {class}*, focusing only on typical, observable features such as shape, material, or context."*

This prompt design leads to more detailed and discriminative text representations than conventional template-based prompts and provides richer cues for vision–language alignment. A qualitative comparison between using CBP and not using CBP is presented in Fig. 9, and a pseudo-code illustrating how CBP is applied is shown in Algorithm 1. Representative examples of the generated descriptions are provided in Tab. 9.

F. Additional Visualization Results.

We provide additional qualitative comparisons. Fig. 10 shows a comparison between our method and the baseline CorrCLIP without post-processing, clearly illustrating the effectiveness of our approach. From Fig. 11 onward, we present the main qualitative results that include post-processing, comparing our method against previous approaches such as SCLIP [50], ProxyCLIP [31], Trident [46], and CorrCLIP [61] across various datasets, including VOC21 [16], Context60 [38], Cityscapes [11], ADE20K [65], COCO Stuff [5], and COCO Object [49].

Algorithm 1 PyTorch-Like Code for Text Embeddings Generation

```
# cls_list: list of class names to embed
# CBP: dict that stores class-biased prompts for some
    classes
# CBP_generator: function that generates biased
    prompts when missing
def generate_text_embeddings(cls_list, CBP,
    CBP_generator)
  text_embeddings = []

  # (1) get class-biased prompts
  for cls in cls_list:
    if cls in CBP.keys():
      biased_prompt = CBP[cls]
    else:
      biased_prompt = CBP_generator(cls)

  # (2) build full prompt set: ImageNet templates +
    biased prompts
    prompts = [temp.format(cls) for temp in
     imagenet_temp] + biased_prompt
    query = tokenizer(prompts)

  # (3) encode prompts with CLIP text encoder
    feature = clip.encode_text(query)
    feature /= feature.norm(dim=-1, keepdim=True)
    feature = feature.mean(dim=0)
    feature /= feature.norm()

  # (4) store class embedding
    text_embeddings.append(feature.unsqueeze(0))

  # (5) stack all class embeddings
  text_embeddings = torch.cat(query_features, dim=0)

  return text_embeddings
```

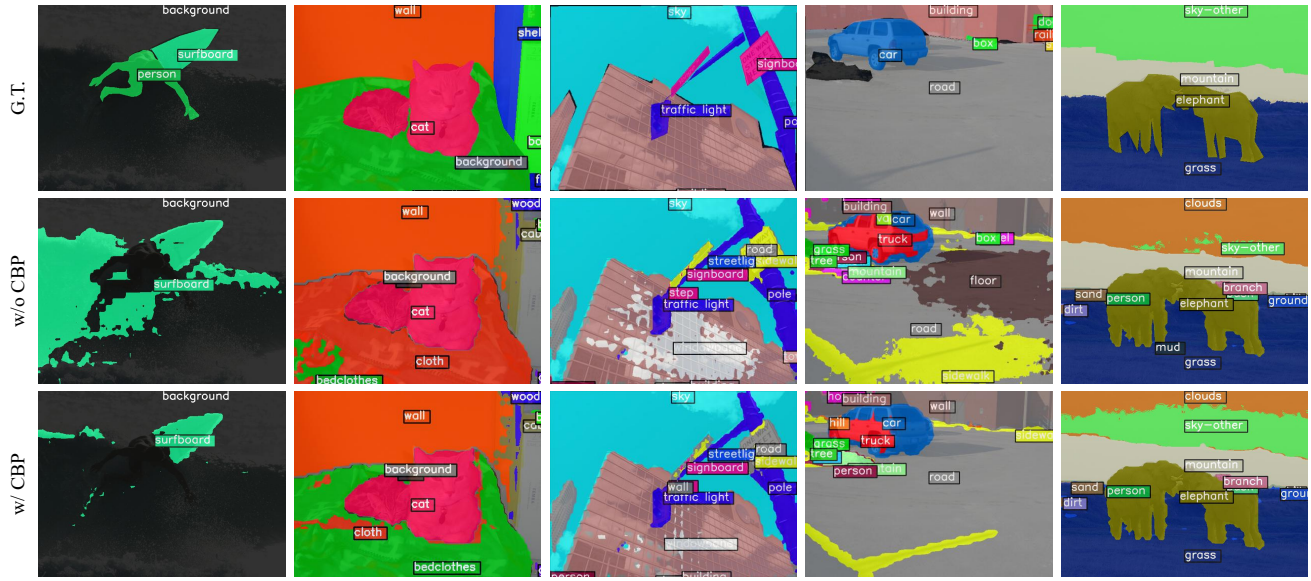

Figure 9. **Qualitative comparison showing the effect of CBP.** Qualitative comparison showing the effect of CBP. To enable a more explicit comparison, post-processing is removed; while higher feature coherence can cause larger regions to be assigned to the wrong class, CBP reduces class ambiguity and helps maintain correct labeling.

Class	Class-Biased Prompts Examples
goldfish	"a small, rounded body covered in shimmering scales" "a fish with a flat tail and vertical fins" "a small, orange fish with a white belly" "a gold-colored fish with a transparent tail" "a small, slender fish with a rounded head" "a fish with a long, flowing fins" "a small, yellow-gold fish with a black spot" "a fish with a flat, broad head" "a small, streamlined fish for swimming" "a fish with a bright orange dorsal fin" "a small, rounded fish with a horizontal stripe" "a fish with transparent scales reflecting light" "a small, gold-colored fish with a pointed snout" "a fish with a long, pointed fin on its back" "a small, slender fish with a distinctive pattern"
salad	"a mix of leafy greens and colorful vegetables", "a bowl of fresh greens and vegetables arranged", "a tossed salad with mixed vegetables and greens", "a colorful salad with edible flowers", "a crunchy salad with crispy vegetables and nuts", "a fresh mix of lettuce and other leafy greens", "a salad with a variety of textures and colors", "a salad bowl filled with mixed greens and toppings", "a simple salad of mixed greens and cherry tomatoes", "a large salad bowl with multiple layers", "a colorful salad with fruits and vegetables", "a green salad with croutons and cheese", "a mixed salad with crunchy and soft ingredients", "a salad with a variety of leafy greens and vegetables", "a refreshing salad with lettuce, tomatoes, and cucumbers"

Table 9. **Representative examples of class-biased prompts generated for each category.**

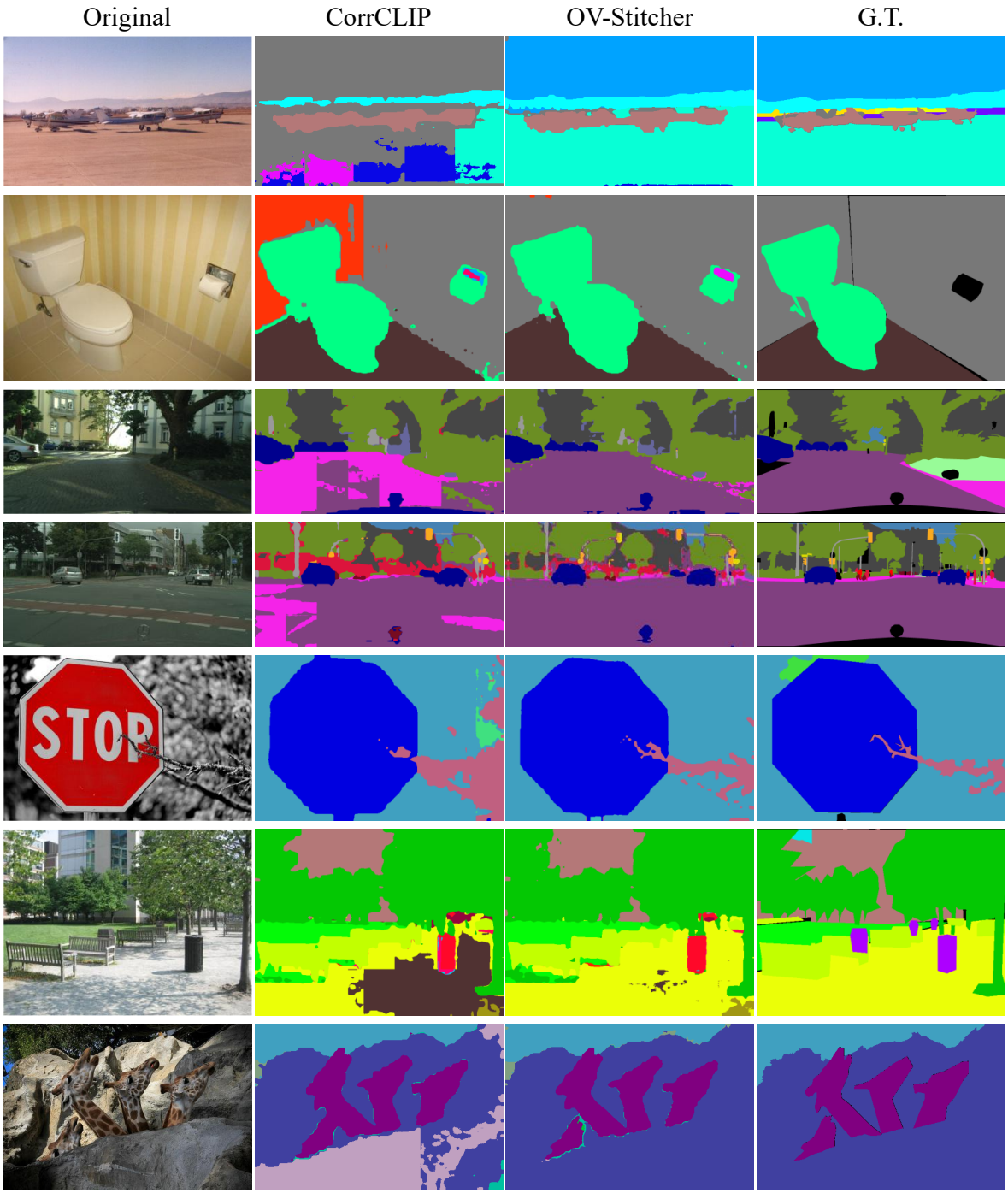

Figure 10. **Qualitative comparison without post-processing.** By removing post-processing, it becomes clear that our method produces more spatially and semantically feature-coherent results than the baseline CorrCLIP.

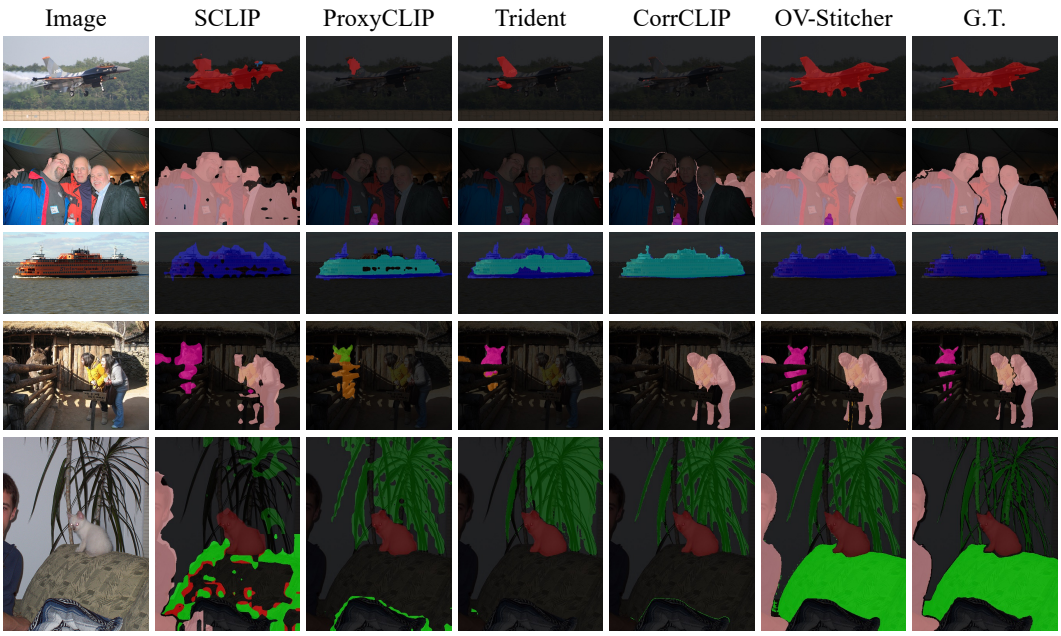

Figure 11. **Additional qualitative comparison on VOC21 [16].**

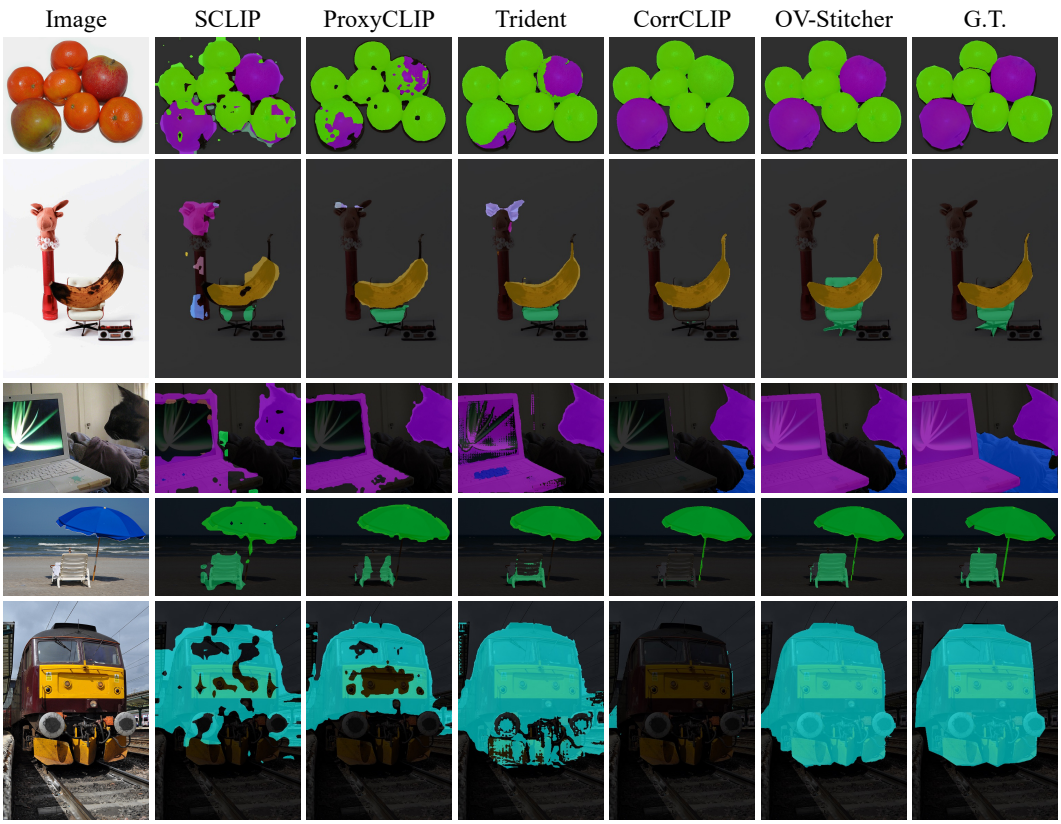

Figure 12. **Additional qualitative comparison on COCO Object [5].**

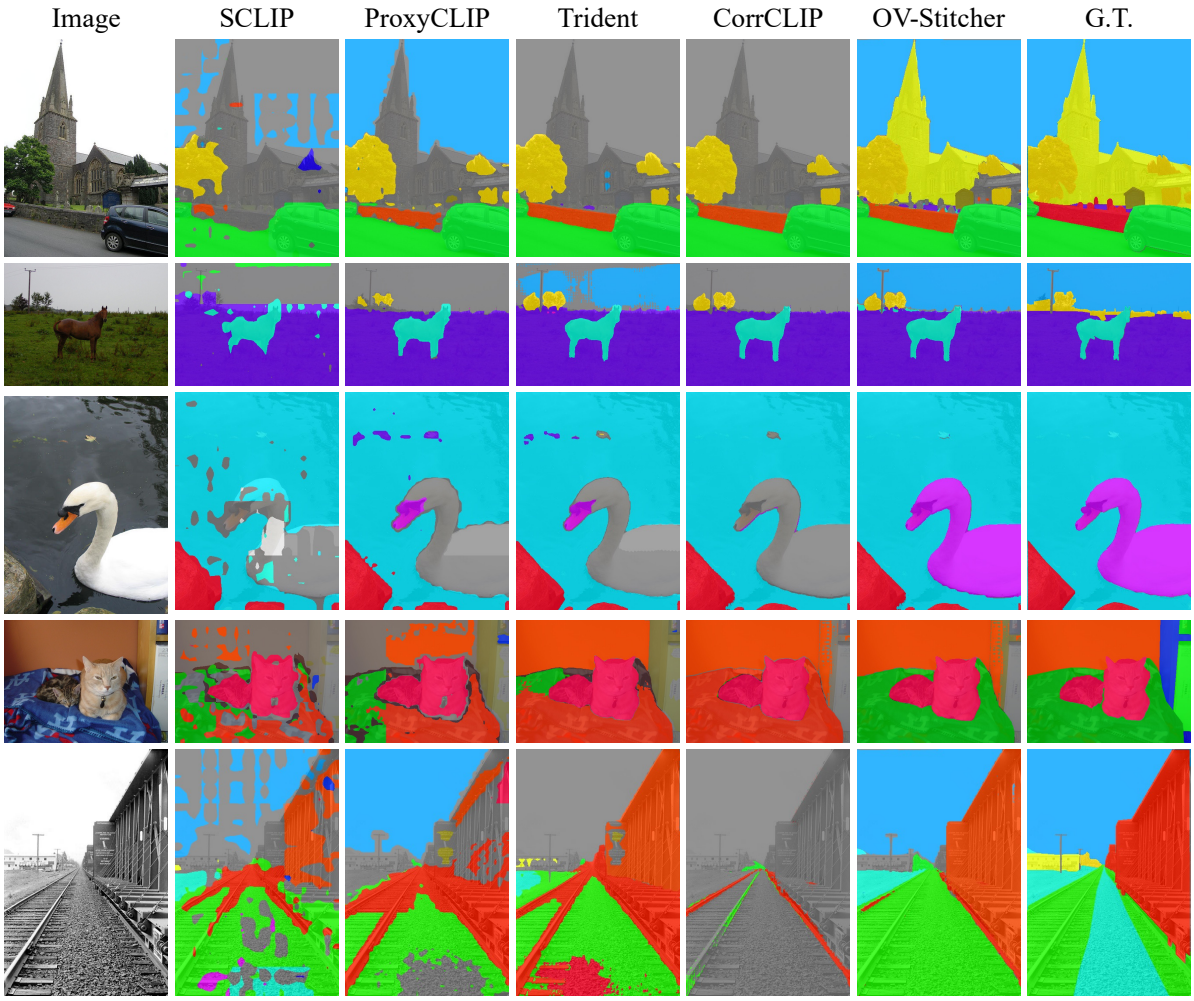

Figure 13. **Additional qualitative comparison on Context60 [38].**

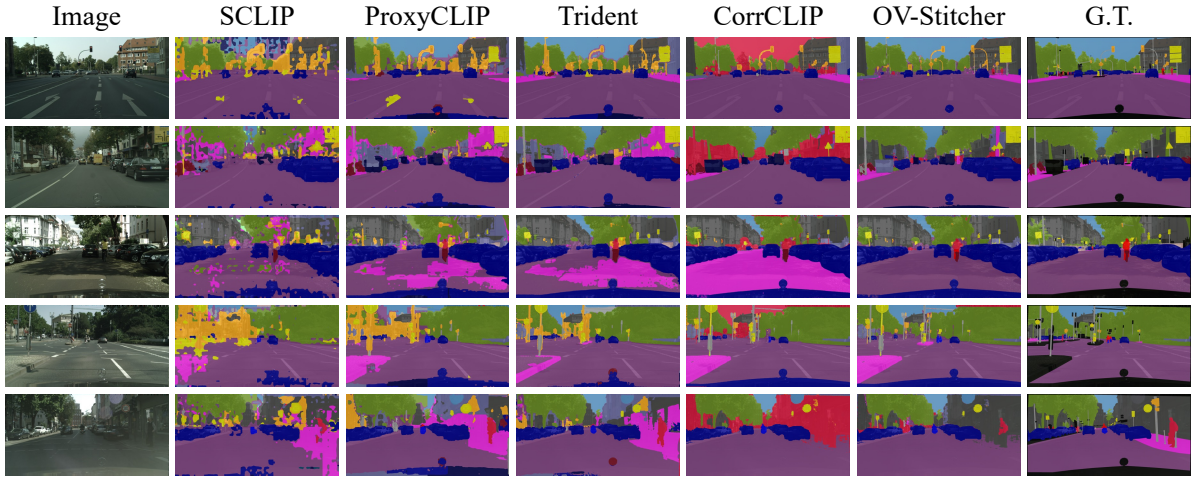

Figure 14. **Additional qualitative comparison on Cityscapes [11]**

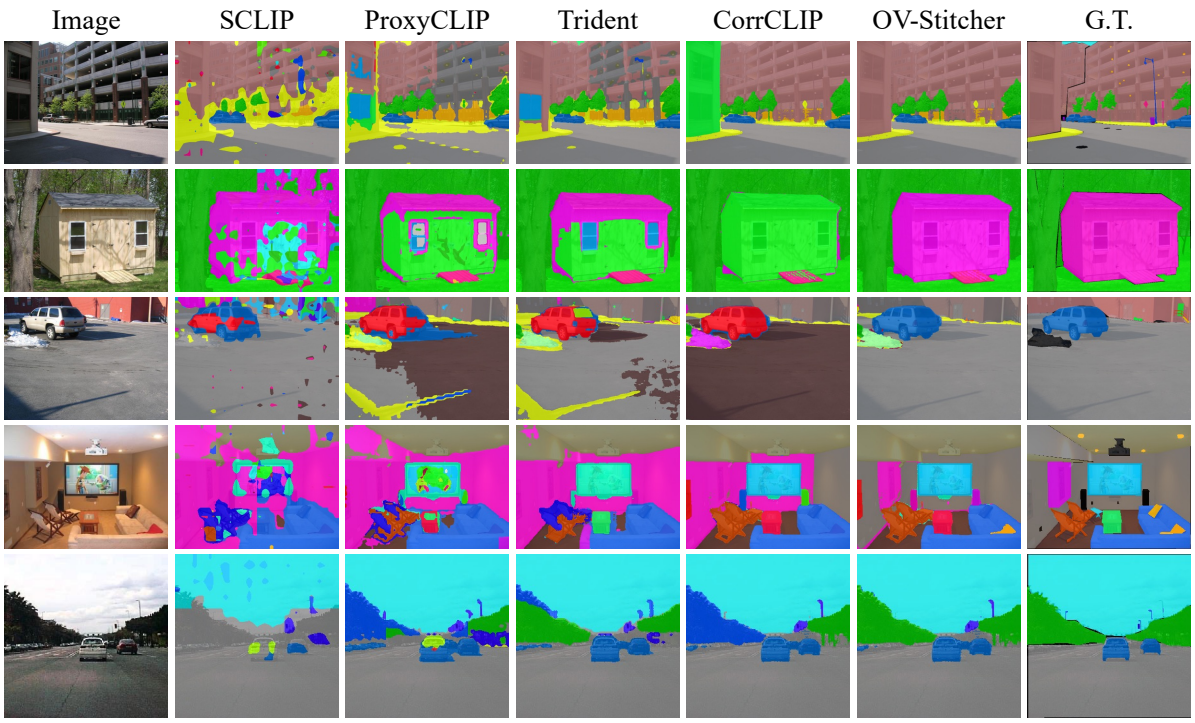

Figure 15. **Additional qualitative comparison on ADE20K [65]**

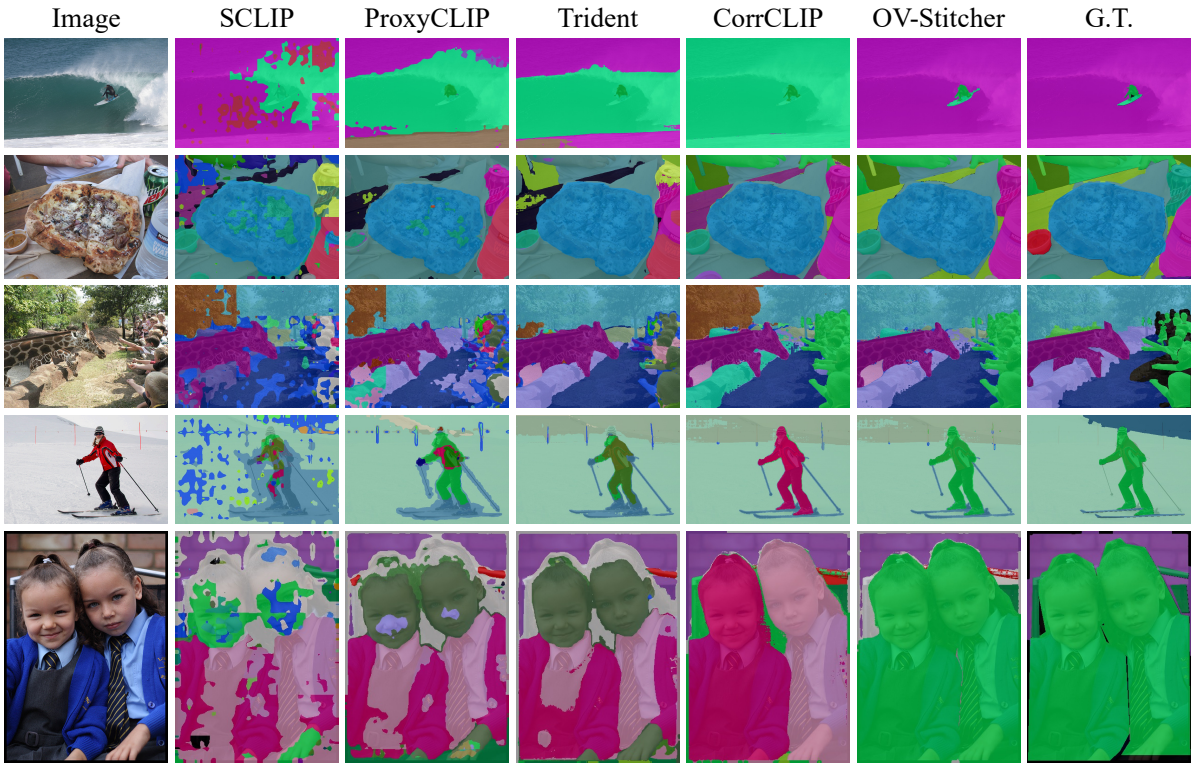

Figure 16. **Additional qualitative comparison on COCO Stuff [5]**