Supplementary Material for “Open-vocabulary Object Segmentation with Diffusion Models”

Ziyi Li*[1], Qinye Zhou*[1], Xiaoyun Zhang[1], Ya Zhang[1,2], Yanfeng Wang†[1,2], and Weidi Xie†[1,2]
[1]Coop. Medianet Innovation Center, Shanghai Jiao Tong University, China
[2]Shanghai AI Laboratory, China
{599lzy, zhouqinye, xiaoyun.zhang, ya_zhang, wangyanfeng, weidi}@sjtu.edu.cn
https://lipurple.github.io/Grounded_Diffusion/

In this supplementary document, we start by giving more details on the architecture of our grounding module in Section A, followed by details for generating the training dataset in Section B; then describe the additional ablation studies in Section C, as promised in the main paper; In Section D, we present additional qualitative results; Lastly, we illustrate the limitation of our method and future work in Section E.

A. Details on the Architecture of Grounding module

We show the detailed architecture of our grounding module in Fig. 1, which consists of visual encoder, text encoder, transformer decoder and MLP in the fusion module.

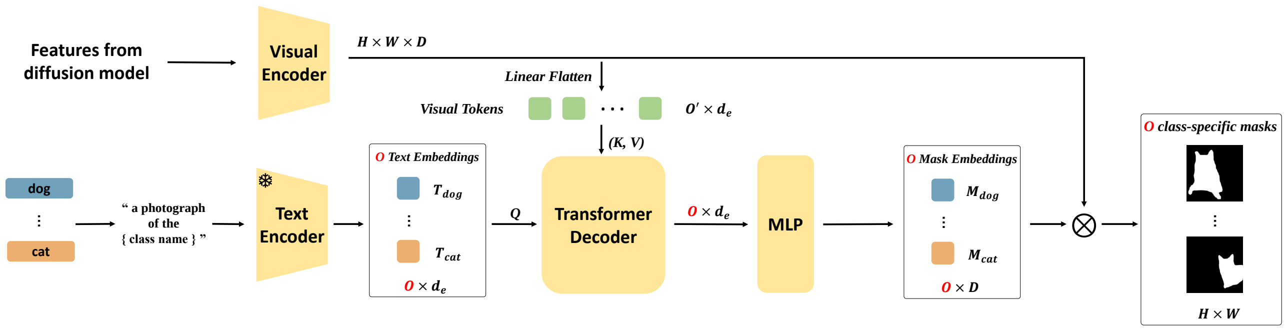


Figure 1: **Detailed architecture of our grounding module.** We first generate $\mathcal{O}$ text embeddings by injecting the class names into a prompt template and then feed them to a pre-trained text encoder. The visual encoder takes the features from Stable Diffusion as input and outputs fused visual features, which are then flattened to a sequence of visual tokens. Next, we feed the visual tokens into a transformer decoder as *Key* and *Value*, and feed text embeddings as *Query*. The outputs of transformer decoder are then fed into an MLP to obtain $\mathcal{O}$ mask embeddings. Mask embeddings are dot producted with the output features of visual encoder to generate $\mathcal{O}$ class-specific binary masks.

Visual Encoder. The input $\{f^1, \ldots, f^n\}$ are extracted from Stable Diffusion [7], and the visual encoder aims to upsample and fuse the visual feature, and output the visual feature map, $\hat{\mathcal{F}} = \Phi_{\text{v-enc}}(\{f^1, \ldots, f^n\}), \hat{\mathcal{F}} \in \mathbb{R}^{H \times W \times D}$ (here we use $H = W = 512$, and $D = 240$). Note that, the resolution of the fused visual feature is the same as the resolution of the generated image, and the segmentation masks.

Text Encoder. We adopt the pre-trained text encoder from CLIP [5], which is also used in Stable Diffusion [7]. It takes text prompt y as input and outputs the corresponding text embedding: $\mathcal{E}_{\text{object}} = \Phi_{\text{t-enc}}(y), \mathcal{E}_{\text{object}} \in \mathbb{R}^{\mathcal{O} \times d_{\text{text}}}$, where $\mathcal{O}$ is the total number of objects of interest and $d_{\text{text}} = 768$.

* Both the authors have contributed equally to this project.
† denotes corresponding author.

Transformer Decoder in Fusion Module. Similar to the operation in standard ViT architecture [2], we convert the visual features $\hat{\mathcal{F}} \in \mathbb{R}^{H \times W \times D}$ into visual tokens $\hat{\mathcal{F}}_{\text{flatten}} \in \mathbb{R}^{\mathcal{O}' \times d_{\text{visual}}}$, where $\mathcal{O}' = \frac{HW}{p^2} = 16384$ is the number of tokens, p refers to the patch size and $d_{\text{visual}} = p^2 \times D = 3840$. Then the visual tokens and text embeddings are mapped into the same dimension $d_e = 512$ with MLPs, and passed into the transformer decoder with three layers. The text embeddings are treated as `query` with dimension $\mathcal{O} \times d_e$, and the visual tokens are treated as `key` and `value` with dimension $\mathcal{O}' \times d_e$. The output of transformer decoder is of the same resolution as `query`, with dimension $\mathcal{O} \times d_e$.

MLP in Fusion Module. At last, we use an MLP to map the output of transformer decoder into mask embeddings with dimension $\mathcal{O} \times D$, which are then dot producted with the fused visual feature ($\hat{\mathcal{F}} \in \mathbb{R}^{H \times W \times D}$) to generate class-specific binary masks ($\mathcal{O} \times H \times W$), each mask is of the same spatial resolution of the generated image.

B. Details on the Synthetic Dataset

B.1. Dataset Split

Here, to train our proposed grounding module, and properly evaluate its ability for segmenting the objects that are unseen at training time, we construct the training dataset with images of only seen categories, and the test dataset consists of both seen and unseen categories. The detail of the split on PASCAL-sim and COCO-sim, *i.e.* the split of seen categories and unseen categories, is shown in Tab. 1, where PASCAL-sim has 15 seen categories and 5 unseen categories, COCO-sim has 65 seen categories and 14 unseen categories. Note that, we ignore the category: 'mouse' in the COCO-sim since the diffusion model generates 'rat' in the image for the category: 'mouse', while 'mouse' in the vocabulary of the off-the-shelf detector means mouse as a computer accessory, thus the detector fails to detect the category 'mouse' in the image generated by the diffusion model.

	Categories	seen	Unseen
PASCAL-sim	Split1	aeroplane, bicycle, bird, boat, bottle, bus, cat, chair, cow, diningtable, horse, motorbike, person, pottedplant, sheep	tvmonitor, car, dog, sofa, train
	Split2	tvmonitor, car, dog, sofa, train, aeroplane, bicycle, bird, boat, bottle, bus, cat, chair, cow, diningtable	horse, motorbike, person, pottedplant, sheep
	Split3	horse, motorbike, person, pottedplant, sheep, tvmonitor, car, dog, sofa, train, aeroplane, bicycle, bird, boat, bottle	bus, cat, chair, cow, diningtable
COCO-sim	Split1	person, bicycle, car, motorbike, bus, truck, boat, traffic light, fire hydrant, stop sign, bench, bird, dog, horse, sheep, cow, elephant, zebra, giraffe, backpack, umbrella, handbag, tie, skis, sports ball, kite, baseball bat, baseball glove, skateboard, surfboard, tennis racket, bottle, wine glass, cup, knife, spoon, bowl, banana, apple, orange, broccoli, carrot, pizza, donut, cake, chair, bench, pottedplant, bed, diningtable, tvmonitor, laptop, remote, keyboard, cell phone, microwave, oven, sink, refrigerator, book, clock, vase, scissors, teddy bear, toothbrush	aeroplane, train, parking meter, cat, bear, suitcase, frisbee, snowboard, fork, sandwich, hot dog, toilet, toaster, hair drier
	Split2	aeroplane, train, parking meter, cat, bear, suitcase, frisbee, snowboard, fork, sandwich, hot dog, toilet, toaster, hair drier, person, bicycle, car, motorbike, bus, truck, boat, traffic light, fire hydrant, stop sign, bench, bird, dog, horse, sheep, cow, elephant, zebra, giraffe, backpack, umbrella, handbag, tie, skis, sports ball, kite, baseball bat, baseball glove, skateboard, surfboard, tennis racket, bottle, wine glass, cup, knife, spoon, bowl, banana, apple, orange, broccoli, carrot, pizza, donut, cake, chair, bench, pottedplant, bed, diningtable, tvmonitor	laptop, remote, keyboard, cell phone, microwave, oven, sink, refrigerator, book, clock, vase, scissors, teddy bear, toothbrush
	Split3	laptop, remote, keyboard, cell phone, microwave, oven, sink, refrigerator, book, clock, vase, scissors, teddy bear, toothbrush, aeroplane, train, parking meter, cat, bear, suitcase, frisbee, snowboard, fork, sandwich, hot dog, toilet, toaster, hair drier, person, bicycle, car, motorbike, bus, truck, boat, traffic light, fire hydrant, stop sign, bench, bird, dog, horse, sheep, cow, elephant, zebra, giraffe, backpack, umbrella, handbag, tie, skis, sports ball, kite, baseball bat, baseball glove, skateboard, surfboard, tennis racket, bottle, wine glass, cup, knife, spoon, bowl	banana, apple, orange, broccoli, carrot, pizza, donut, cake, chair, bench, pottedplant, bed, diningtable, tvmonitor

Table 1: **The details on the split of categories on PASCAL-sim and COCO-sim.**

B.2. Dataset for Training Grounding Module

To construct the training set, (1) we first randomly select one or two categories from the seen ones, where objects tend to co-appear in natural images, based on the annotation in PASCAL VOC [3] or COCO [4], called **co-appearing category pair**, and use the prompt template to decorate these selected categories, thus we can obtain the text prompt; (2) we pass the text prompt and randomly sampled Gaussian noise to the Stable Diffusion [7] to obtain the generated image; (3) next, we pass the generated image to the off-the-shelf detector to obtain the oracle segmentation mask; (4) finally, we can construct the triplet which consists of the generated image, oracle segmentation mask, and text prompt; (5) repeat the above procedure, we can generate infinite triplets for the training set. Algorithm 1 displays the procedure for generating the training set.

To construct the test set for evaluating the grounding module, we can use a procedure similar to the training set. The differences are: (i) we use all categories, including seen and unseen categories to construct the test set. (ii) to obtain more reliable test results, we only add the triplet to the test set when the generated image and oracle segmentation mask have high quality which is checked manually, *i.e.*, the generated image by Stable Diffusion contains the recognizable objects of selected categories, and the off-the-shelf detector successfully produces the high-quality oracle segmentation mask.

In this paper, PASCAL-sim has 20 categories and 142 co-appearing category pairs. We construct 30 triplets per category and 5 triplets per co-appearing category pair for PASCAL-sim test set. In total, PASCAL-sim test set has 1310 triplets. COCO-sim has 79 categories and 1559 co-appearing category pairs. We construct 30 triplets per category and 2 triplets per co-appearing category pair for COCO-sim test set. In total, COCO-sim test set has 5488 triplets.

Algorithm 1 Constructing the dataset for training grounding module (pseudocode in PyTorch-like style).

```
# C_seen: the list of seen categories
# img_shape: the shape of expected generated image
# exp_train_size: the expected size of training set
# n: the number of selected categories, n = 1 or 2
# co-appearing_category_pair_list: a list containing all co-appearing category pairs, where objects tend to
# co-appear in natural images, based on the annotation in PASCAL VOC or COCO

D_train = [] #initialize the training set

while (len(D_train) < exp_train_size):

   y = None #initialize the text prompt

   #randomly select n categories from seen categories
   selected_class_list = random_select(C_seen, n)

   if n = 1:
      class = select_class_list[0]

      # decorate the selected category by a pre-defined prompt template, e.g., "a photograph of a [class name]"
      y = prompt_template(class)

   else if n = 2:
      class1, class2 = select_class_list[0], select_class_list[1]
      if (class1, class2) in co-appearing_category_pair_list:

         # decorate the selected categories by a pre-defined prompt template, e.g., "a photograph of a
         # [class1 name] and a [class2 name]"
         y = prompt_template(class1, class2)

   if y != None:

      #randomly sample a Gaussian noise epsilon
      epsilon=torch.randn(img_shape)

      # pass the noise and text prompt to the diffusion model to generate image I
      I = diffusion_model(epsilon, y)

      # pass the generated image to the off-the-shelf detector to obtain the oracle segmentation mask m
      m = pretrain_detector(I)

      # add the triplet (generated image, oracle segmentation mask, text prompt) to the training set
      D_train.append((I, m, y))
```

B.3. Dataset for Training Semantic Segmentation Model

As discussed in the main text, we synthesize two semantic segmentation datasets for all 20 categories in PASCAL VOC [3] and 79 categories in COCO [4], respectively. We first randomly select one or two categories (co-appearing category pair), to construct the text prompt, and then pass randomly sampled Gaussian noise and text prompt to the diffusion model to obtain the generated image, and use our proposed grounding module to get the corresponding segmentation mask. Thus, we can get the pair of generated image and segmentation mask. Repeat the above procedure, we can obtain the synthetic semantic segmentation datasets at a large scale. Algorithm 2 displays the procedure for generating the synthetic semantic segmentation dataset.

In this paper, for categories in PASCAL VOC, the synthetic semantic segmentation dataset consists of 500 images per category and 71 images per co-appearing category pair. Thus, there exist 10k images for 20 categories and 10082 images for 142 co-appearing category pairs in total. For categories in COCO, the dataset consists of 1500 images per category and 70 images per co-appearing category pair, thus there exist 118500 images for 79 categories and 109130 images for 1559 co-appearing category pairs

Algorithm 2 Pseudo-code for generating the synthetic semantic segmentation dataset in a PyTorch-like style.

```
# C: the list of all categories
# img_shape: the shape of expected generated image
# exp_dataset_size: the expected size of synthesis semantic segmentation dataset
# n: the number of selected categories, n = 1 or 2
# co-appearing_category_pair_list: a list containing all co-appearing category pairs, where objects tend to
# co-appear in natural images, based on the annotation in PASCAL VOC or COCO

D_seg = [] #initialize the synthesis semantic segmentation dataset

while (len(D_seg) < exp_dataset_size):

   y = None #initialize the text prompt

   #randomly select n categories from all categories
   selected_class_list = random_select(C, n)

   if n = 1:
      class = select_class_list[0]

      # decorate the selected category by a pre-defined prompt template, e.g., "a photograph of a [class name]"
      y = prompt_template(class)

   else if n = 2:
      class1, class2 = select_class_list[0], select_class_list[1]
      if (class1, class2) in co-appearing_category_pair_list:

         # decorate the selected categories by a pre-defined prompt template, e.g., "a photograph of a
         # [class1 name] and a [class2 name]"
         y = prompt_template(class1, class2)

   if y != None:

      #randomly sample a Gaussian noise epsilon
      epsilon=torch.randn(img_shape)

      # pass the noise and text prompt to the diffusion model with grounding module to generate image I
      # and segmentaion mask m
      I, m = diffusion_model_with_grounding(epsilon, y)

      # add the pair (generated image, generated segmentation mask) to the synthesis semantic segmentation dataset
      D_seg.append((I, m))
```

C. Additional Ablation Study

C.1. Synthetic Dataset Construction.

We explore the effect of constructing different datasets for training the grounding module, by varying the number of objects in the images. As shown in Tab. 2, training on the combination of one and two object categories gives the best results overall.

Train Set	One		Two		
# Objects	Seen	Unseen	Seen	Seen +Unseen	Unseen
single	**90.37**	**83.85**	43.89	42.33	38.91
two	88.35	82.93	**80.56**	**68.08**	56.36
mixture	90.16	83.19	78.93	66.07	**57.93**

Table 2: **Ablation on dataset construction on PASCAL-sim.** The bolded number indicates the best result. Our model achieves the best performance when training on the combination of one and two object categories.

C.2. Timesteps for Extracting Visual Representation.

We compare the performance by extracting visual representation from Stable Diffusion at different timesteps, the results on PASCAL-sim can be seen in Fig. 2, showing that as the denoising steps gradually decrease, *i.e.*, from $t = 0 \longrightarrow 50$, the performance for grounding tends to decrease in general, when $t = 5$, the best result is obtained.

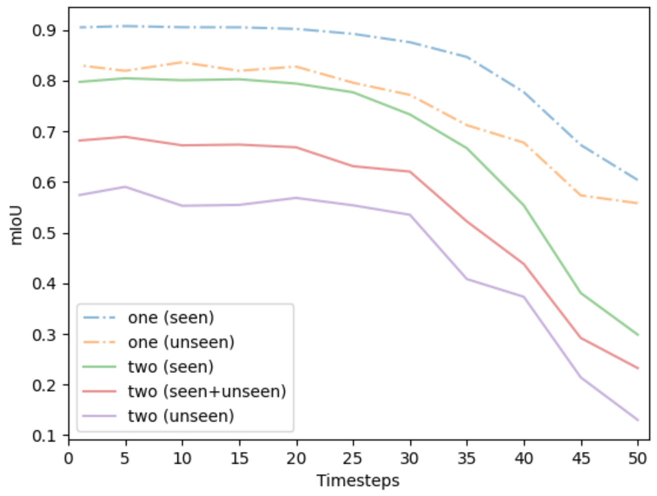


Figure 2: **Ablation on timesteps.** The mIoU is measured for the model with extracting features from Stable Diffusion in different timesteps on **PASCAL-sim**.

C.3. Dataset Construction via DDIM Inverse.

In addition to using the off-the-shelf detectors, we also consider constructing the training set by utilising the inverse process of diffusion to explicitly generate images close to those in the public dataset, for example, PASCAL VOC, and train the grounding module with the mask annotations available from the dataset.

Here, we describe an inverse procedure that enables to find a deterministic mapping from noise to images, given the sampling rule being non-Markovian, for example, Denoising Diffusion Implicit Model (DDIM) [8] with the reverse process variance to be 0. In DALL-E 2 [6], such inversion has been used to determine the noise that produces a specific image. In our considered Stable Diffusion, the image is first mapped to a latent vector z^0 by the pre-trained variational autoencoder (VAE), at each step of DDIM inversion, z^{t+1} is obtained from z^t and the predicted noise term of U-Net, that takes z^t and text prompt y as input, ending up with an inverted noise z^T eventually. In this paper, we exploit such DDIM inversion to train our grounding module with the dataset constructed from real image and segmentation masks.

In particular, the first option enables to directly inherit the segmentation mask from the public dataset, and the text prompt can be manually constructed by inserting class labels into the prompt template, for example, if the segmentation mask contains 'dog' and 'cat', the text prompt can be 'a photograph of a dog and cat'. Besides, the visual feature can be obtained by extracting the feature from the U-Net of Stable Diffusion when $t = 1$ at the inversion process.

Constructed Dataset *v.s.* Real Dataset. We explore the difference between training on constructed dataset and real dataset (PASCAL VOC) from two perspectives. *First*, we compare their performance on PASCAL-sim dataset for grounded generation in Tab. 3 (left). Though we successfully train our grounding module on real dataset, the domain gap limits its performance on grounded generation task. Considering PASCAL VOC only contains about 10k images, we adjust the construced dataset to the same magnitude and get better results. Additionally, due to the good scalability of the constructed dataset, the performance improves as the number of images increases. *Second*, we evaluate the grounding module on PASCAL VOC test dataset by DDIM inversion as shown in Tab. 3 (right). Note that, under this circumstance, our model approximates a discriminative model. On seen categories of PASCAL VOC test set, the module trained on real dataset achieves the best result, while the module trained on constructed dataset gains an advantage on unseen categories. Besides, we also train our module on both constructed dataset and real dataset, which results in great improvement on PASCAL VOC test dataset.

Dataset Type	PASCAL-sim					PASCAL-test	
	One		Two				
	Seen	Unseen	Seen	Seen+Unseen	Unseen	Seen	Unseen
real	75.67	61.26	64.08	49.23	45.14	**75.19**	34.80
sim(10k)	88.77	70.04	73.07	57.08	46.59	61.75	48.42
sim(40k)	**90.16**	**83.19**	**78.93**	**66.07**	**57.93**	64.44	53.86
sim+real	89.57	76.23	78.12	62.24	55.30	73.32	**57.14**

Table 3: **Ablation on the training dataset.** The bold numbers indicate the best results. Specifically, 'sim' and 'real' denote the constructed dataset and real dataset (PASCAL-VOC), respectively.

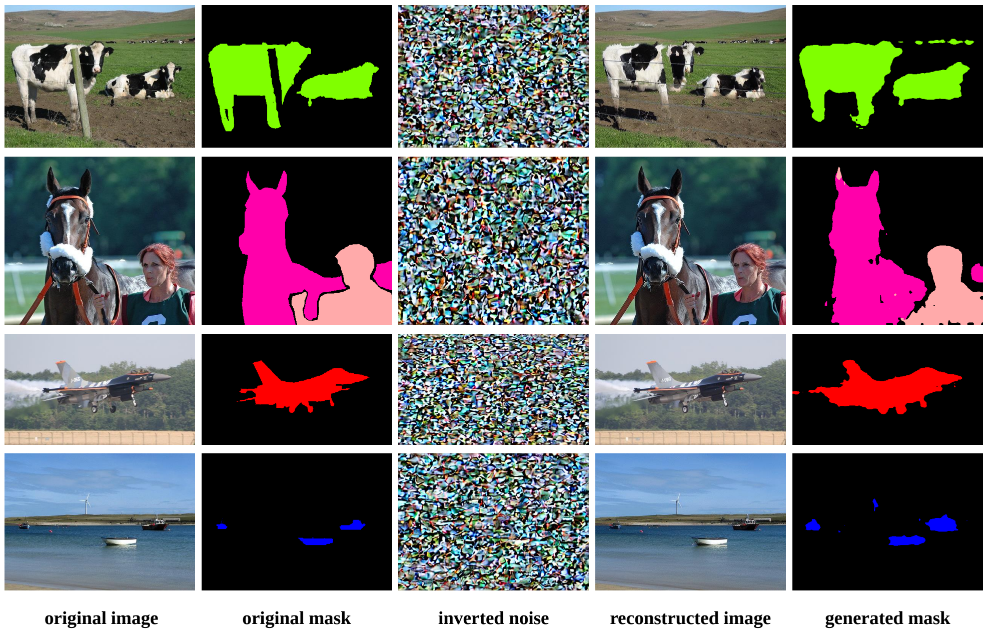


Figure 3: **Qualitative results on real dataset.** By DDIM Inversion, we firstly map the original images to corresponding noise, which are taken as input to regenerate images. And we train the grounding module to predict the masks during the regeneration process, with the supervision of the original masks.

D. More Qualitative Results

We provide more qualitative results in Fig. 4, Fig. 5, Fig. 6, and Fig. 7. Note that the images are generated from Stable Diffusion [7], and the corresponding masks are inferred from our proposed grounding module. Specifically, the generated images and their corresponding segmentation masks in Fig. 4 and Fig. 5, including common objects, *e.g.*, shark, turtle, and more unusual objects, *e.g.*, Ultraman, pterosaur, Chinese dragon, unicorn and dinosaur, shows the strong generalisability of the grounding module. In Fig. 6, we show more examples from our synthetic semantic segmentation dataset. In Fig. 7, we compare the model trained on our synthesized datasets with other ZS3 methods on PASCAL VOC dataset [3]. We can observe that the MaskFormer [1] trained on our synthetic semantic segmentation dataset can obtain accurate segmentation on both seen and unseen categories, showing that the guided text-to-image diffusion model can be used to expand the vocabulary of pre-trained detector.

E. Limitation & Future Work

In this paper, we have demonstrated the possibility for extracting the visual-language correspondence from a pre-trained text-to-image diffusion model in the form of segmentation map, that can augment the diffusion model with the ability of grounding visual objects along with generation. However, we also realise there exists certain limitation in this work, *first*, we only consider to ground the nouns that indicate visual entities, it would be interesting to ground the human-object, object-object interactions, or even verbs in the future, *second*, we are inserting the grounding module to a pre-trained text-to-image generative model, it would be interesting to co-train the two components, potentially enabling to generate images with higher quality and explainability.

References

[1] Bowen Cheng, Alex Schwing, and Alexander Kirillov. Per-pixel classification is not all you need for semantic segmentation. *NeurIPS*, 2021. 8

[2] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 2

[3] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision(IJCV)*, 2010. 4, 5, 8

[4] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Proc. ECCV*, 2014. 4, 5

[5] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *ICML*, 2021. 1

[6] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022. 6

[7] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proc. CVPR*, 2022. 1, 4, 8

[8] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020. 6

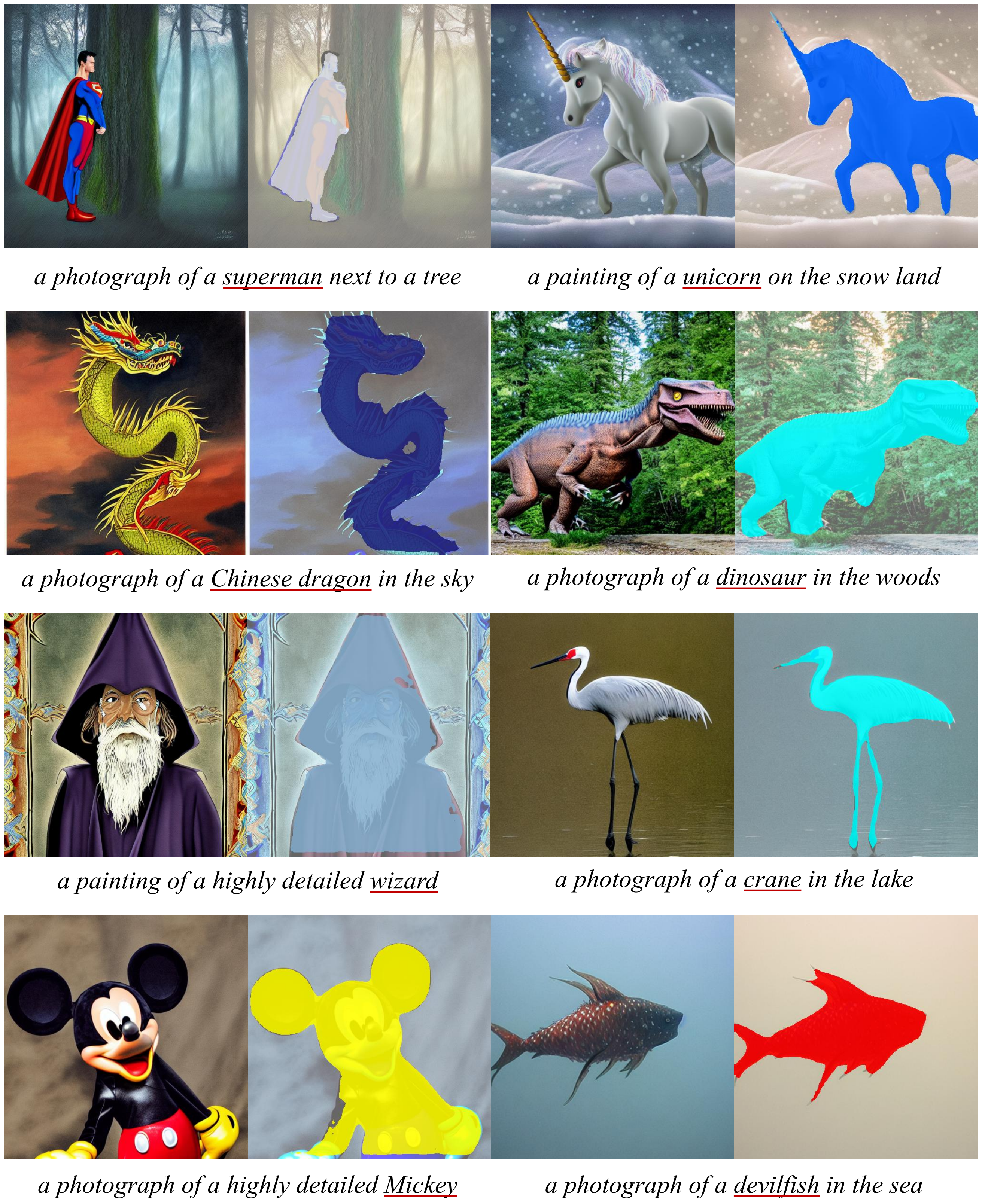
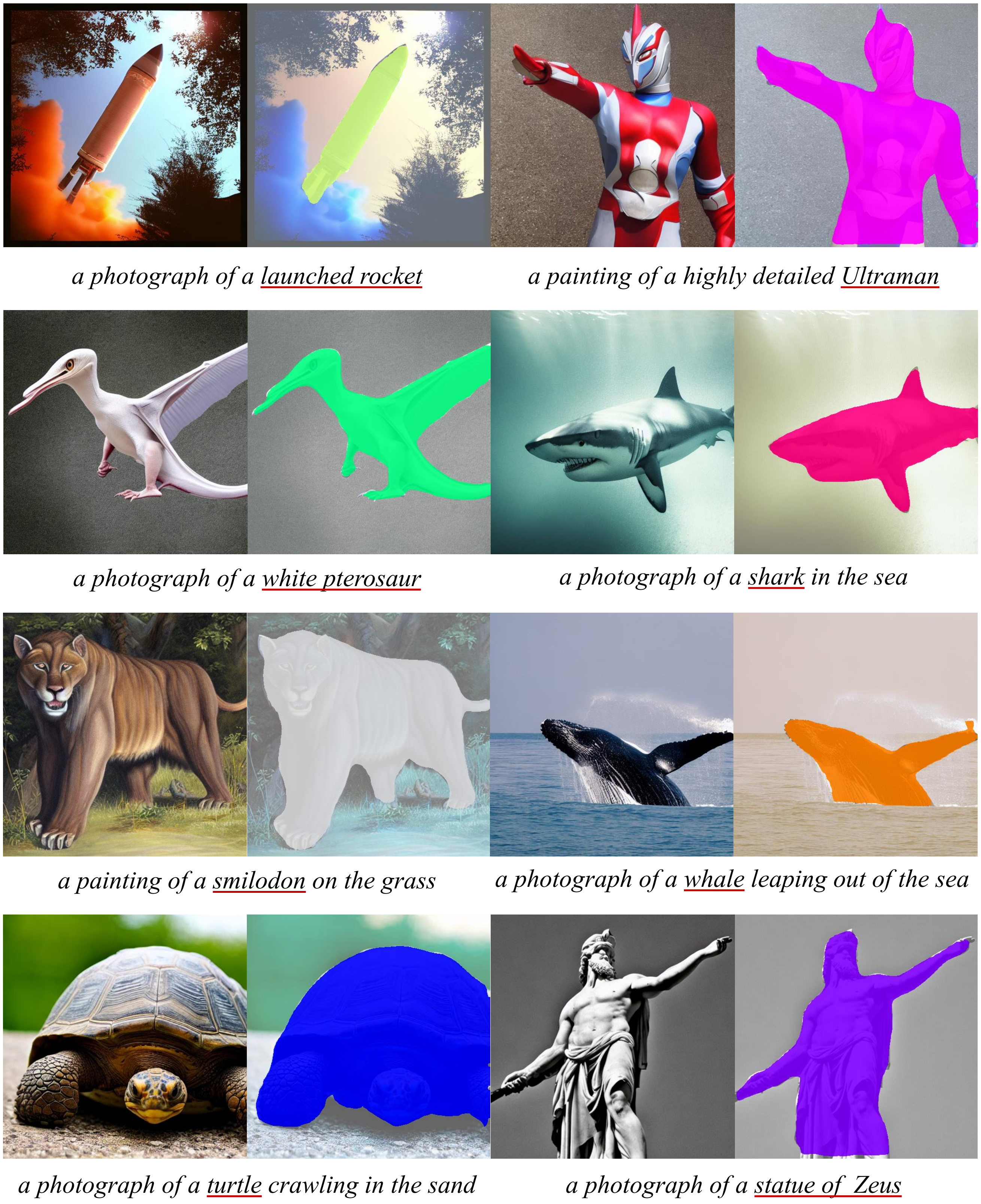

a photograph of a superman next to a tree
a painting of a unicorn on the snow land
a photograph of a Chinese dragon in the sky
a photograph of a dinosaur in the woods
a painting of a highly detailed wizard
a photograph of a crane in the lake
a photograph of a highly detailed Mickey
a photograph of a devilfish in the sea

Figure 4: **Results of grounded generation.** The segmentation mask refers to the grounding results for the object underlined.



a photograph of a launched rocket
a painting of a highly detailed Ultraman
a photograph of a white pterosaur
a photograph of a shark in the sea
a painting of a smilodon on the grass
a photograph of a whale leaping out of the sea
a photograph of a turtle crawling in the sand
a photograph of a statue of Zeus

Figure 5: **Results of grounded generation.** The segmentation mask refers to the grounding results for the object underlined.

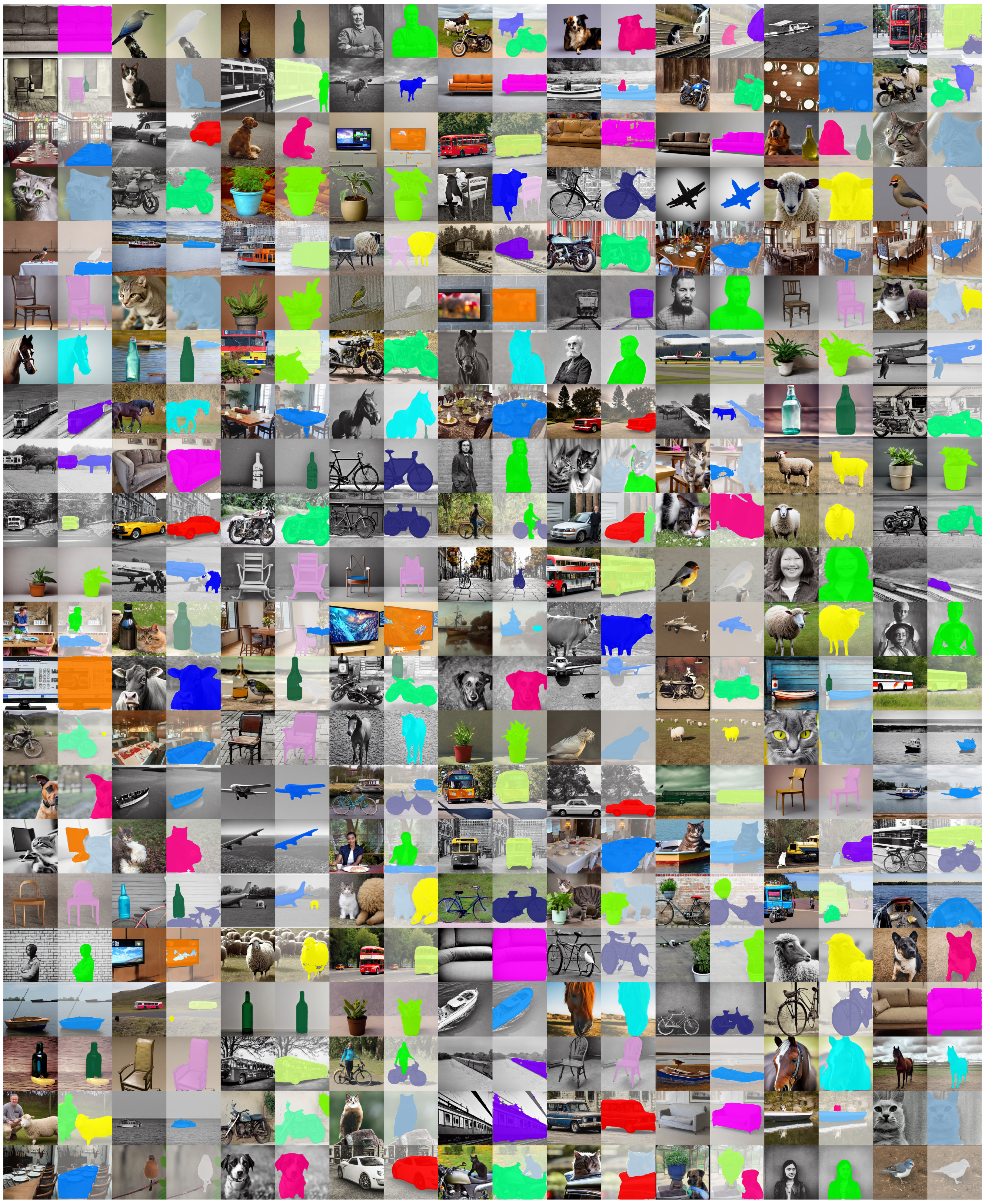


Figure 6: **Examples from our synthetic semantic segmentation dataset.**

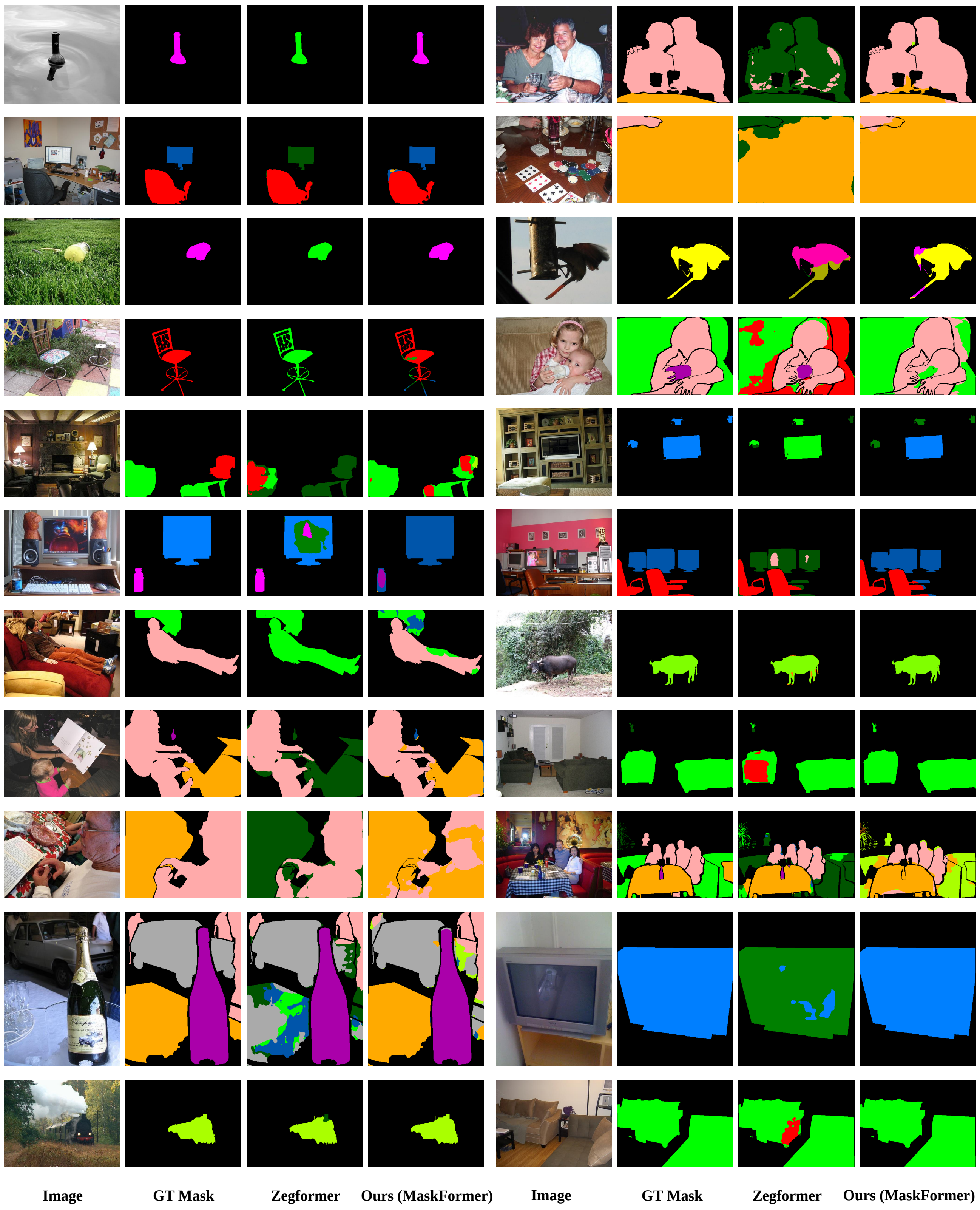


Figure 7: **More visualization of zero-shot segmentation results on Pascal VOC.**