UniGS: Unified Representation for Image Generation and Segmentation Supplementary File

This supplementary material document provides more visualization results and training/inference details on Multi-class multi-region inpainting, image synthesis, referring segmentation, and entity segmentation. The supplementary material is organized as follows:

- Training/inference details on four tasks.
- More visualization results, including an example illustration of decoding colormap and the generation results of three tasks.

Also, please check *the recorded video* on our project page to obtain a brief description of our paper.

1. Training/inference Details

Multi-class multi-region inpainting In our research, we train our model on the COCO and Open-Images dataset, initializing it with the stable diffusion inpainting model v1.5. For each image, we sample a maximum of four objects and then. Based on the ground truth mask, we have two schemes to make the coarse mask: 1. simulate a more coarse mask using the curve. 2. direct. The process follows in Algorithm 1 that is used in Paint-by-Example. This coarse mask is then cropped out from the original image. The cropped image and coarse mask are concatenated and fed into a UNet, with the expected output being an inpainted image and a separate color mask. Under this setting, the model was trained for 48 epochs, resulting in the development of our model. During inference, we sample a maximum of three objects. Similarly, we sample out coarse masks. And utilize randomly initialized noise for DDIM denoising, a total of 200 steps.

We have compared the number of parameters and the inference speed with the original stable diffusion inpainting model in Table 1. We have only added a few channels to the input and output, keeping the parameter count almost consistent with that of stable diffusion. The inference speed also remains nearly the same. Without incurring additional computational costs, we have achieved better output results.

Method	Parameters	Speed
Stable Diffusion	859.54 M	14.48
UniGS (Ours)	859.56 M	14.40

Table 1. **Comparison of parameters and inference speed between Stable Diffusion and our UniGS.** The inference speed is tested by *the seconds per image*. We use the DDIM sampling strategy for both methods. We do not use any accelerating techniques for a fair comparison. And we note that those techniques used in Stable Diffusion also work in our UniGS.

Image Synthesis During image synthesis, we use a colormap as our conditional input instead of an image, and the input for the coarse mask is a mask entirely filled with ones, indicating that the area of interest is the entire image. We conducted a full training over 48 epochs on the COCO and Open-Images dataset, with the maximum number of entity samples set to four. The inference process is consistent with the training but with a maximum sample number of three. The input includes a text condition, colormap, and an all-one coarse mask to produce the synthesized image.

Referring segmentation In the task of referring segmentation, the method of sampling coarse mask remains consistent with the inpainting approach, but the coarse mask region is not excised from the original image. Instead, the original image and the coarse mask are concatenated and then jointly input into the model. Similarly, the model was trained on the COCO and Open-Images dataset for 48 epochs to yield the final model. The original image, coarse mask, and text prompt are input during inference. Post-processing is then performed on the output colormap to obtain the final segmentation mask

Entity segmentation For entity segmentation, unprocessed original images are input along with an all-one coarse mask, indicating that the entire image area is subject to segmentation. During training, we no longer sample entities but directly use all entities from the COCO and EntitySeg datasets, encoding them into a colormap as input for the framework. In inference, the output colormap undergoes post-processing, but the background is not removed. Instead, all clusters are retained as individual entities.

Algorithm 1 Pseudocode (Python-like) of the Coarse Mask Sampling Method: Curve and Bounding Box

```
prob=random.uniform(0, 1)
## random or bounding box mask

if prob<self.arbitrary_mask_percent:
   mask_img = Image.new('RGB', (W, H), (255, 255, 255))
   bbox_mask=copy.copy(bbox)
   extended_bbox_mask=copy.copy(extended_bbox)
   top_nodes = np.asfortranarray([
               [bbox_mask[0], (bbox_mask[0]+bbox_mask[2])/2 , bbox_mask[2]],
               [bbox_mask[1], extended_bbox_mask[1], bbox_mask[1]],
            ])
   down_nodes = np.asfortranarray([
         [bbox_mask[2],(bbox_mask[0]+bbox_mask[2])/2 , bbox_mask[0]],
         [bbox_mask[3], extended_bbox_mask[3], bbox_mask[3]],
      ])
   left_nodes = np.asfortranarray([
         [bbox_mask[0],extended_bbox_mask[0] , bbox_mask[0]],
         [bbox_mask[3], (bbox_mask[1]+bbox_mask[3])/2, bbox_mask[1]],
      ])
   right_nodes = np.asfortranarray([
         [bbox_mask[2],extended_bbox_mask[2] , bbox_mask[2]],
         [bbox_mask[1], (bbox_mask[1]+bbox_mask[3])/2, bbox_mask[3]],
      ])
   top_curve = bezier.Curve(top_nodes,degree=2)
   right_curve = bezier.Curve(right_nodes,degree=2)
   down_curve = bezier.Curve(down_nodes,degree=2)
   left_curve = bezier.Curve(left_nodes,degree=2)
   curve_list=[top_curve,right_curve,down_curve,left_curve]
   pt_list=[]
   random_width=5
   for curve in curve_list:
      x_list=[]
      y_list=[]
      for i in range(1,19):
         if (curve.evaluate(i*0.05)[0][0]) not in x_list and (curve.evaluate(i*0.05)[1][0] not in y_list):
            pt_list.append((curve.evaluate(i*0.05)[0][0]+random.randint(-random_width,random_width),curve.
                evaluate(i*0.05)[1][0]+random.randint(-random_width,random_width)))
            x_list.append(curve.evaluate(i*0.05)[0][0])
            y_list.append(curve.evaluate(i*0.05)[1][0])
   mask_img_draw=ImageDraw.Draw(mask_img)
   mask_img_draw.polygon(pt_list,fill=(0,0,0))
   mask_tensor=get_tensor(normalize=False, toTensor=True)(mask_img)[0].unsqueeze(0)

else:
   mask_img=np.zeros((H,W))
   mask_img[extended_bbox[1]:extended_bbox[3],extended_bbox[0]:extended_bbox[2]]=1
   mask_img=Image.fromarray(mask_img)
   mask_tensor=1-get_tensor(normalize=False, toTensor=True)(mask_img)
```

One-to-Many Joint Model We train our single model for the four tasks within the COCO, Open-Images, and EntitySeg datasets. Unlike training a single model, we add the task embedding for the multi-task training. Furthermore, the sample ratios of four tasks, including multi-class multi-region inpainting, image synthesis, referring segmentation, and entity segmentation, are 0.3, 0.3, 0.2, and 0.2, respectively. We train the single model for multi-tasks in 96 epochs. Finally, this model performs comparably to each model of a single task, indicating the great potential of our unified representation for image generation and segmentation.

2. More Visualization Results

Progressive Dichotomy Module. Figure 1 shows an example of our progressive dichotomy module to decode the generated colormap into several explicit entity masks. We can see that our decoding process does not require assigning cluster numbers in a depth-first search manner.

Visualization Results Figure 2, 3 and 4 shows more visualization results of multi-class multi-region inpainting and image synthesis with our UniGS framework. And Figure 5 shows the entity segmentation results of our UniGS.

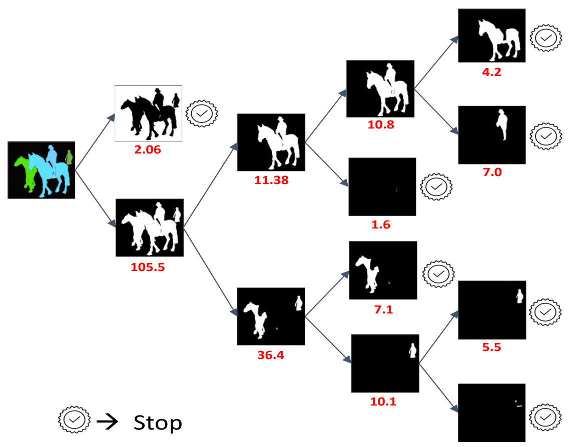

Figure 1. **An example illustration of our progressive dichotomy module at each clustering iteration.** The red color indicates the average distance to its cluster center for all the pixels in the cluster.

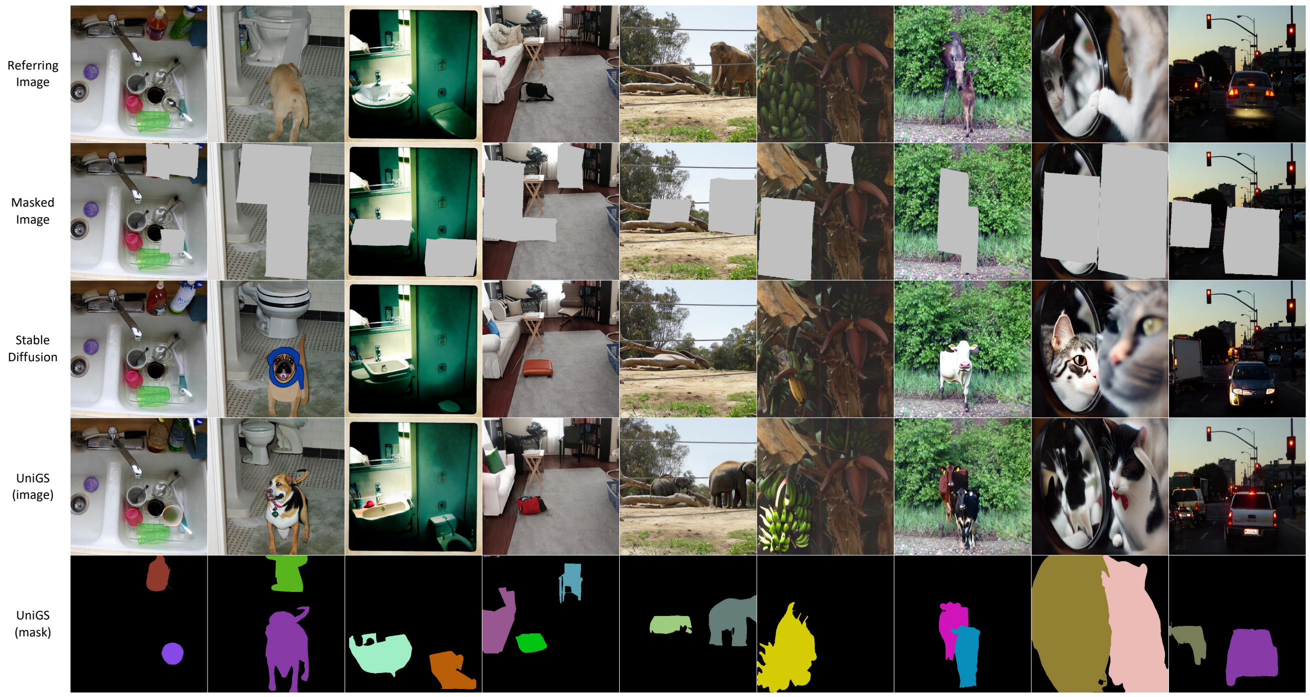

Figure 2. **More visualization results of our UniGS in multi-class multi-region inpainting.**

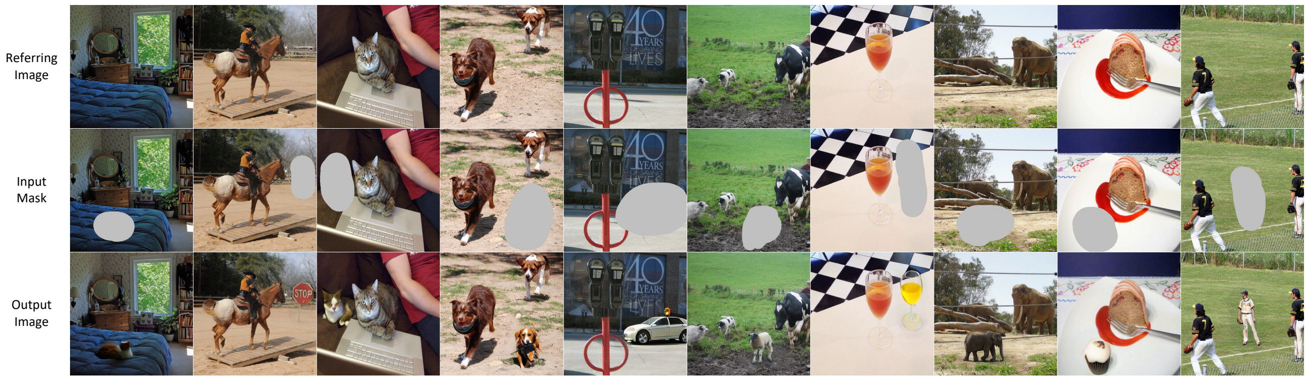

Figure 3. **More visualization results of our UniGS in the real world.** We generate the coarse masks used in inpainting by brush strokes from Gradio. We identified some interesting observations, particularly the appearance of shadows from the third to sixth columns.

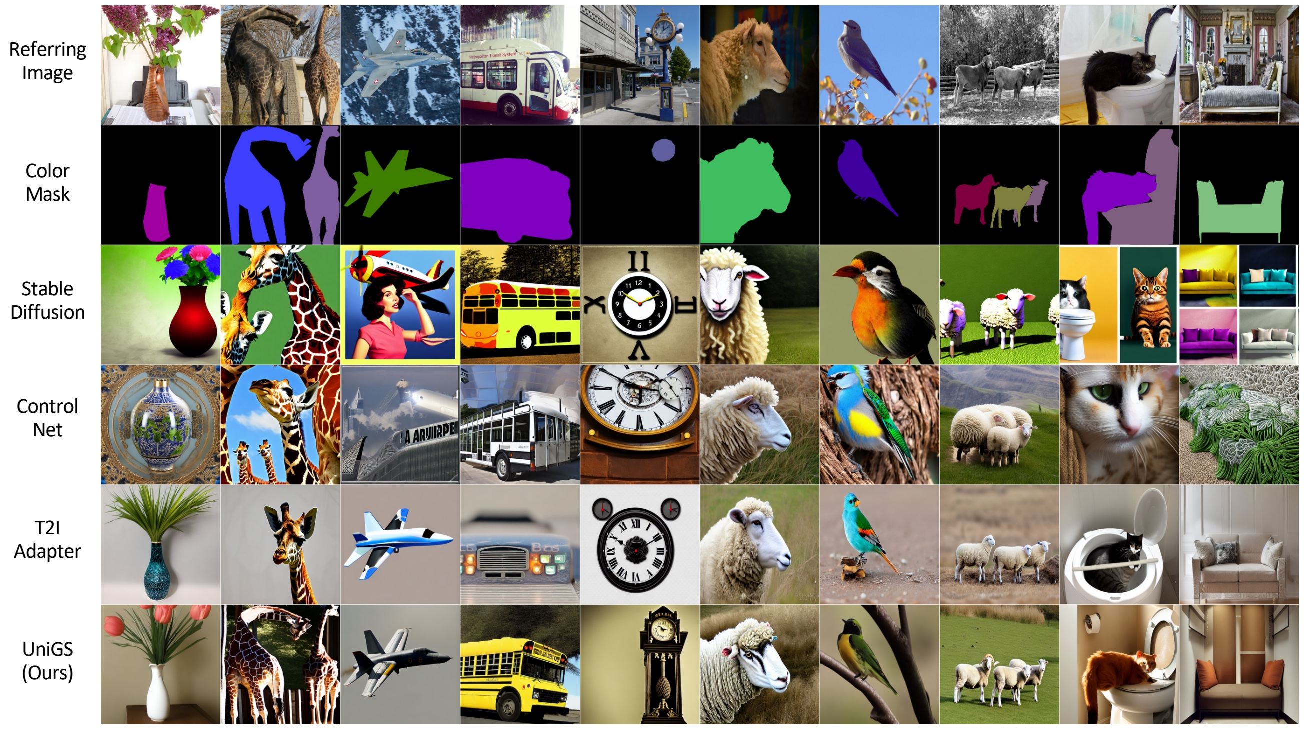

Figure 4. **More visualization results of our UniGS in image synthesis**

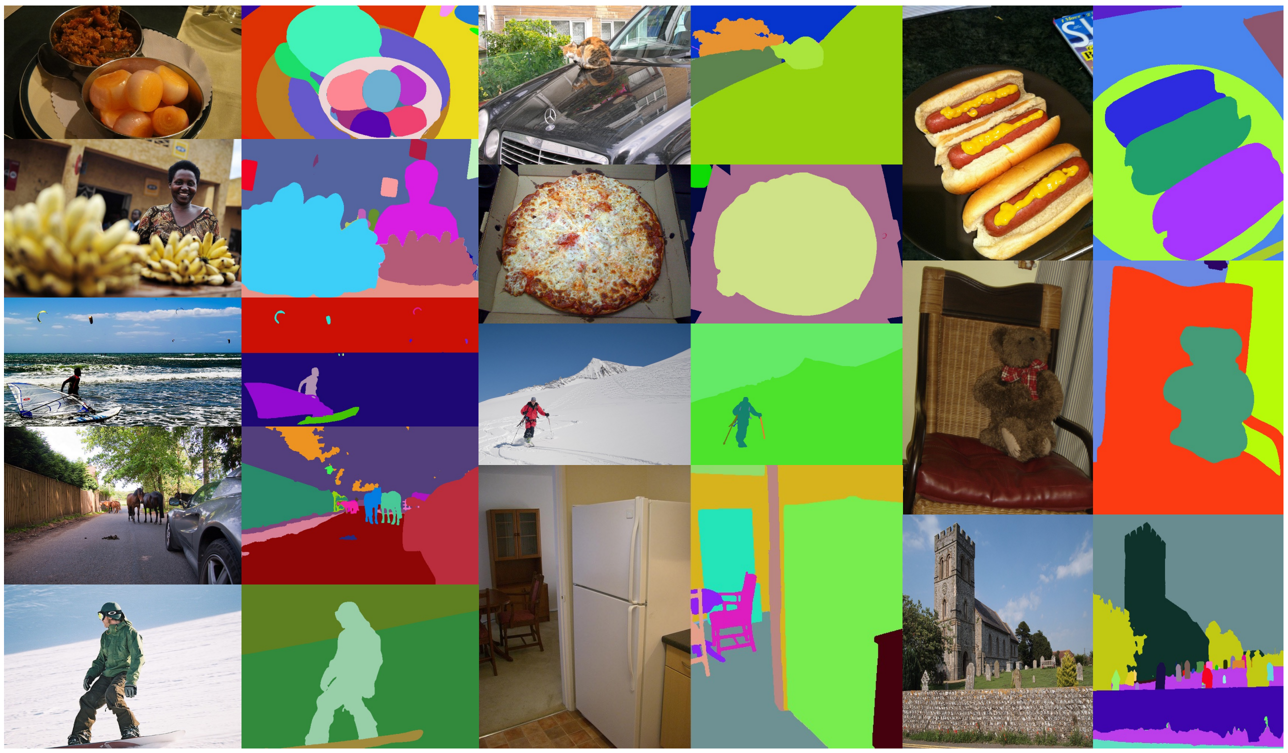

Figure 5. **More visualization results of our UniGS in Entity Segmentation.**