Algorithm 3: DDIM Update

```
def alpha_cumprod(t, ns=0.0002, ds=0.00025):
 """cosine noise schedule"""
 n = torch.cos((t + ns) / (1 + ds)
    * math.pi / 2) ** -2
 return -torch.log(n - 1, eps=1e-5)

def ddim(map_t, map_pred, t_now, t_next):
  """
  estimate x at t_next with DDIM update rule.
  """
  $\alpha_{\text{now}}$ = alpha_cumprod(t_now)
  $\alpha_{\text{next}}$ = alpha_cumprod(t_next)
  map_enc = encoding(map_pred)
  map_enc = (sigmoid(map_enc) * 2 - 1) * scale
  eps = $\frac{1}{\sqrt{1-\alpha_{\text{now}}}}$ * (map_t - $\sqrt{\alpha_{\text{now}}}$ * map_enc)
  map_next = $\sqrt{\alpha_{\text{next}}}$ * x_pred + $\sqrt{1-\alpha_{\text{now}}}$ * eps
  return map_next
```

Algorithm 4: DDP Self-aligned Denoising

```
def train(images, maps):
  """
  images: [b, 3, h, w], maps: [b, 1, h, w]
  """
  img_enc = image_encoder(images)
  map_t = normal(mean=0, std=1)
  map_pred = map_decoder(map_t, img_enc, t=1)
  # encode map_pred
  map_enc = encoding(map_pred.detach())
  map_enc = (sigmoid(map_enc) * 2 - 1) * scale
  # corrupt the map_enc
  t, eps = uniform(0, 1), normal(mean=0, std=1)
  map_crpt = sqrt(alpha_cumprod(t)) * map_enc +
         sqrt(1 - alpha_cumprod(t)) * eps
  # predict
  map_pred = map_decoder(map_crpt, img_enc, t)
  loss = objective_func(map_pred, maps)
  return loss
```

A. Diffusion Model

A.1. Algorithm details

As a supplement to Algorithm 1 and Algorithm 2 described in the main paper, we provide the implementation details in Algorithm 3 for better clarity. Additionally, we introduce the implementation of the "*self-aligned denoising*" procedure in Algorithm 4, used in the last 5K iteration training to address the sampling drift problem (see Section 3.4). We provide an example in Figure 4 to illustrate the gap between the training and inference denoising targets.

A.2. More Discussions

As illustrated in Figure 3a, diffusion models for perceptual tasks tend to reach a saturation point within the first few steps, usually between 3-5 steps, making additional diffusion less advantageous. This is in contrast to the requirements of generative models for image generation, where multiple iterations over many steps (from 10 to 50) are often necessary. Intuitively, in generative tasks such as image generation, the goal is to produce complete and high-quality results by progressively incorporating more information at each time step, thus gradually accumulating and improving the overall result. Therefore, it may take more time steps to reach convergence in order to fully accumulate the necessary information. In perceptual tasks, such as semantic segmentation and object detection, the process from image to label is a gradual reduction of information, and critical information sufficient to make a decision needs to be obtained in only a few steps. Therefore, further diffusion has a limited role in improving the accuracy of predictions, leading to an early peak within three to five steps. In short, the diffusion process in a perception task can make decisions by accumulating the most important information. Therefore, DDP can achieve high accuracy in perception tasks with minimal computational cost.

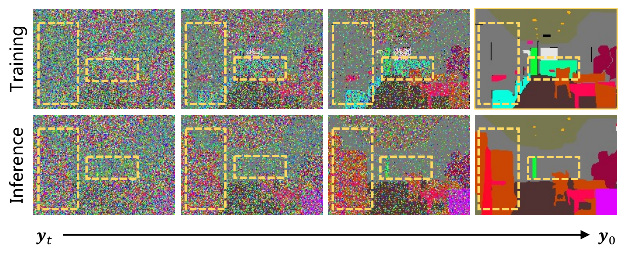

Figure 4. **Sampling drift**. Denoising targets differ from the training process and inference process.

B. Implementation Details

B.1. Semantic Segmentation

ADE20K. We conduct the experiments of ADE20K [99] semantic segmentation based on MMSegmentation [20]. In the training phase, the backbone is initialized with the ImageNet [26] pre-trained weights. We optimize our DDP models using AdamW [56] optimizer with an initial learning rate of 6×10^{-5}, and a weight decay of 0.01. The learning rate is decayed following the polynomial decay schedule with a power of 1.0. Besides, we randomly resize and crop the image to 512×512 for training, and rescale to have a shorter side of 512 pixels during testing. All models are trained for 160k iterations with a batch size of 16 and compared fairly with previous discriminative-based and non-diffusion methods.

Cityscapes. The Cityscape dataset includes 5000 high-resolution images, which contain 2,975 training images, 500 validation images, and 1525 testing samples. The images are captured from 50 different cities in Germany, cov-

Method	$\delta_1 \uparrow$	$\delta_2 \uparrow$	$\delta_3 \uparrow$	REL $\downarrow$	RMS $\downarrow$	$\log_{10} \downarrow$
Chen et al.	0.757	0.943	0.984	0.166	0.494	0.071
Yin et al. [90]	0.696	0.912	0.973	0.183	0.541	0.082
BTS [44]	0.740	0.933	0.980	0.172	0.515	0.075
AdaBins [3]	0.771	0.944	0.983	0.159	0.476	0.068
DepthFormer [48]	<u>0.815</u>	<u>0.970</u>	<u>0.993</u>	<u>0.137</u>	<u>0.408</u>	<u>0.059</u>
DDP (step 3)	**0.825**	**0.973**	**0.994**	**0.128**	**0.397**	**0.056**

Table 6. **Depth estimation on the SUN RGB-D dataset.** We report the result of the model trained on the NYU-DepthV2 dataset and tested on the SUN RGB-D dataset without fine-tuning.

ering various environments such as highways, city centers, and suburbs. Similar to ADE20K, during training, we load the ImageNet pre-trained weights and employ the AdamW optimizer. Following common practice, we randomly resize and crop the image to 512×1024 for training, and take the original images of 1024×2048 for testing. We Other hyper-parameters are kept the same as our ADE20K experiments.

B.2. BEV Map Segmentation

nuScenes. We conduct our experiments of BEV map segmentation on nuScenes [7], a large-scale multi-modal dataset for 3D detection and map segmentation. The dataset is split into 700/150/150 scenes for training/validation/testing. It contains data from multiple sensors, including six cameras, one LIDAR, and five radars. For camera inputs, each frame consists of six views of the surrounding environment at the same timestamps. We resize the input views to 256×704 and voxelize the point cloud to 0.1m. Our evaluation metrics align with [54] and report the IoU of 6 background classes, including drivable space, pedestrian crossing, walk-way, stop line, car-parking area, and lane divider, and use the mean IoU as the primary evaluation metric. We adopt the image and LiDAR data augmentation strategies from [8] for training. AdamW is utilized with a weight decay of 0.01 and a learning rate of 5e-5. We take overall 20 training epochs on 8 A100 GPUs with a batch size of 32. Other training settings are kept the same as [54] for fair comparisons.

B.3. Depth Estimation

KITTI. The KITTI depth estimation dataset is a widely used benchmark dataset for monocular depth estimation with a depth range from 0-80m. The stereo images of the dataset have a resolution of 1242×375, while the corresponding GT depth map has a low density of 3.75% to 5.0%. Following the standard Eigen training/testing split [28], we use around 26K left view images for training and 697 frames for testing. We incorporate the DDP model into the codebase developed by [48] for KITTI depth estimation experiments. We excluded the discrete label encoding module as the task requires continuous value regression All experimental settings are the same as [48] for a fair comparison.

Method	$\delta_1 \uparrow$	$\delta_2 \uparrow$	$\delta_3 \uparrow$	REL $\downarrow$	RMSE $\downarrow$	$\log_{10} \downarrow$
StructDepth [45]	0.817	0.955	0.988	0.140	0.534	0.060
MonoIndoor [41]	0.823	0.958	0.989	0.134	0.526	-
DORN [30]	0.828	0.965	0.992	0.115	0.509	0.051
BTS [44]	0.885	0.978	0.994	0.110	0.392	0.047
DAV [38]	0.882	0.980	0.996	0.108	0.412	-
TransDepth [87]	0.900	0.983	0.996	0.106	0.365	0.045
DPT-Hybrid [61]	<u>0.904</u>	0.988	**0.998**	0.110	0.357	0.045
AdaBins [3]	0.903	0.984	<u>0.997</u>	0.103	0.364	0.044
DepthFormer [48]	**0.921**	<u>0.989</u>	**0.998**	<u>0.096</u>	<u>0.339</u>	<u>0.041</u>
DDP (step 3)	**0.921**	**0.990**	**0.998**	**0.094**	**0.329**	**0.040**

Table 7. **Depth estimation on the NYU-DepthV2 val set.** We report the performance of DDP with 3 diffusion steps. The best and second-best results are bolded or underlined, respectively. ↓ means lower is better, and ↑ means higher is better.

NYU-DepthV2. The NYU-DepthV2 is an indoor scene dataset that consists of RGB and depth images captured at a resolution of 640×480 pixels. The dataset contains over 1,449 pairs of aligned indoor scenes, captured from 464 different indoor areas. We train DDP using image pairs with a resolution of 320×240 and with varying depths up to approximately 10 meters. Following previous work, we evaluate the results on the predefined center cropping by [28]. To be fair, all experimental configurations were aligned with the previous method [48].

SUN RGB-D. We use this dataset [74] to evaluate generalization. To be specific, we assess the performance of our NYU pre-trained models on the official test set, which includes 5,050 images, without any additional fine-tuning. The maximum depth is restricted to 10 meters. Please note that this dataset is solely intended for evaluation purposes and is not utilized for training.

C. Experimental Results

In Table 7, we provide the depth estimation performance of DDP on the NYU-V2 dataset, in addition, in Table 6, we provide the generalization performance results of DDP on the SUN-RGBD dataset.

D. Visualization

Figure 5 and Figure 6 visualize the "multiple inference" property of DDP on the validation sets of Cityscapes and ADE20K, respectively. These inference trajectories show that DDP can enhance its performance continuously and produce smoother segmentation maps by using more sampling steps. Figure 7 presents the BEV map segmentation results of DDP (step 3) with the ground truths and multi-view images. Figure 8 and Figure 9 compare the generated depth estimation results of DDP (step 3) with the ground truths on the validation sets of KITTI and NYU-DepthV2, respectively. These results indicate that our method can be easily generalized to most dense prediction tasks.

E. More Applications

E.1. Combine DDP with ControlNet

Setup. It has been found that compared to the previous single-shot model, DDP can achieve more continuous and semantic consistency prediction results. To demonstrate the benefits of this pixel clustering property, we combined DDP with the recently popular segmentation mask condition generation model: ControlNet. We followed the official implementation of ControlNet for all hyperparameters, including input resolution and DDIM sampling steps.

Implementation ControlNet [94] improves upon the original Stable Diffusion (SD) model by adding extra conditions, which is done by incorporating a conditioning network. In the mask-conditional ControlNet, the map generated by the segmentation model is used as input for image synthesis. The original segmentation model was adopted from Uniformaer-S [47] with UperNetHead, which has 52M parameters and achieves 47.6 mIoU (ss) on the ADE20K dataset. To make a fair comparison, we replaced the original segmentation model in the mask-conditional ControlNet with DDP using the Swin-T backbone, which has 40M parameters and achieves 47.0 mIoU (ss) on the ADE20K dataset. Note that all results were obtained with the default prompt.

Results We select images from the PEXEL website `https://www.pexels.com/` for testing in different scenarios. The results from the original ControlNet and the combination of DDP with ControlNet are shown in Figure 10. ControlNet is designed to achieve fine-grained, controllable image generation, our experiments show that DDP can produce more consistent results and has advantages in various scenarios. Moreover, when combined with DDP, ControlNet produces visually satisfying and well-composed results, surpassing those of the original ControlNet. Our experimental results suggest that DDP has great potential to improve cooperation with other types of foundation models.

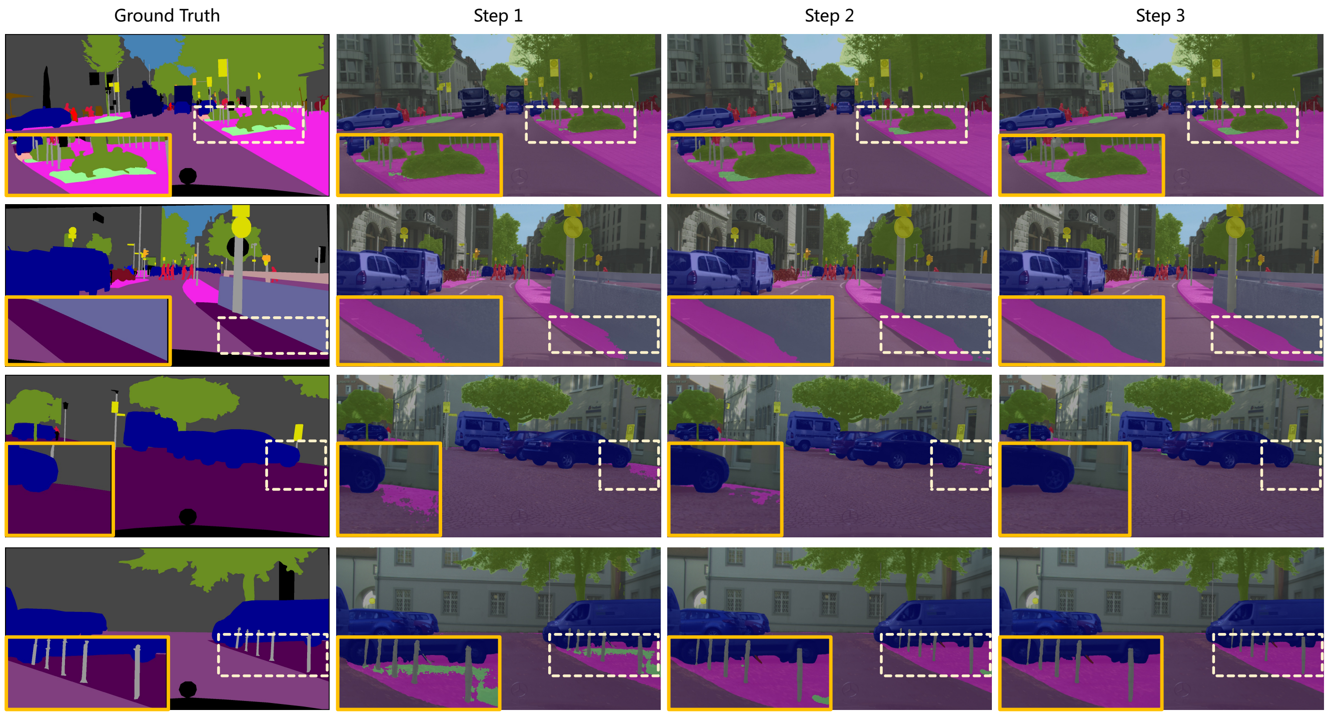

Figure 5. **Visualization of multiple inference on Cityscapes val set.**

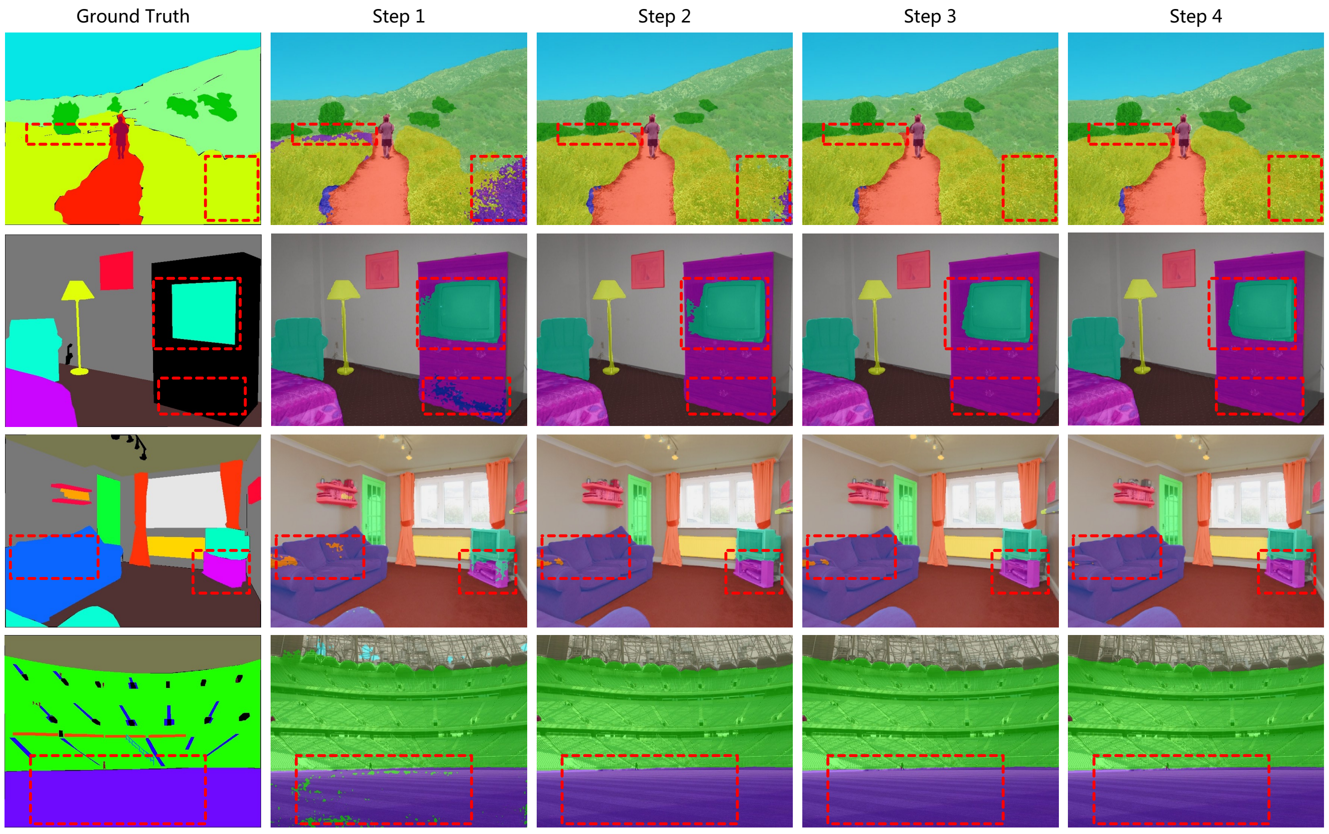

Figure 6. **Visualization of multiple inference on ADE20K val set.**

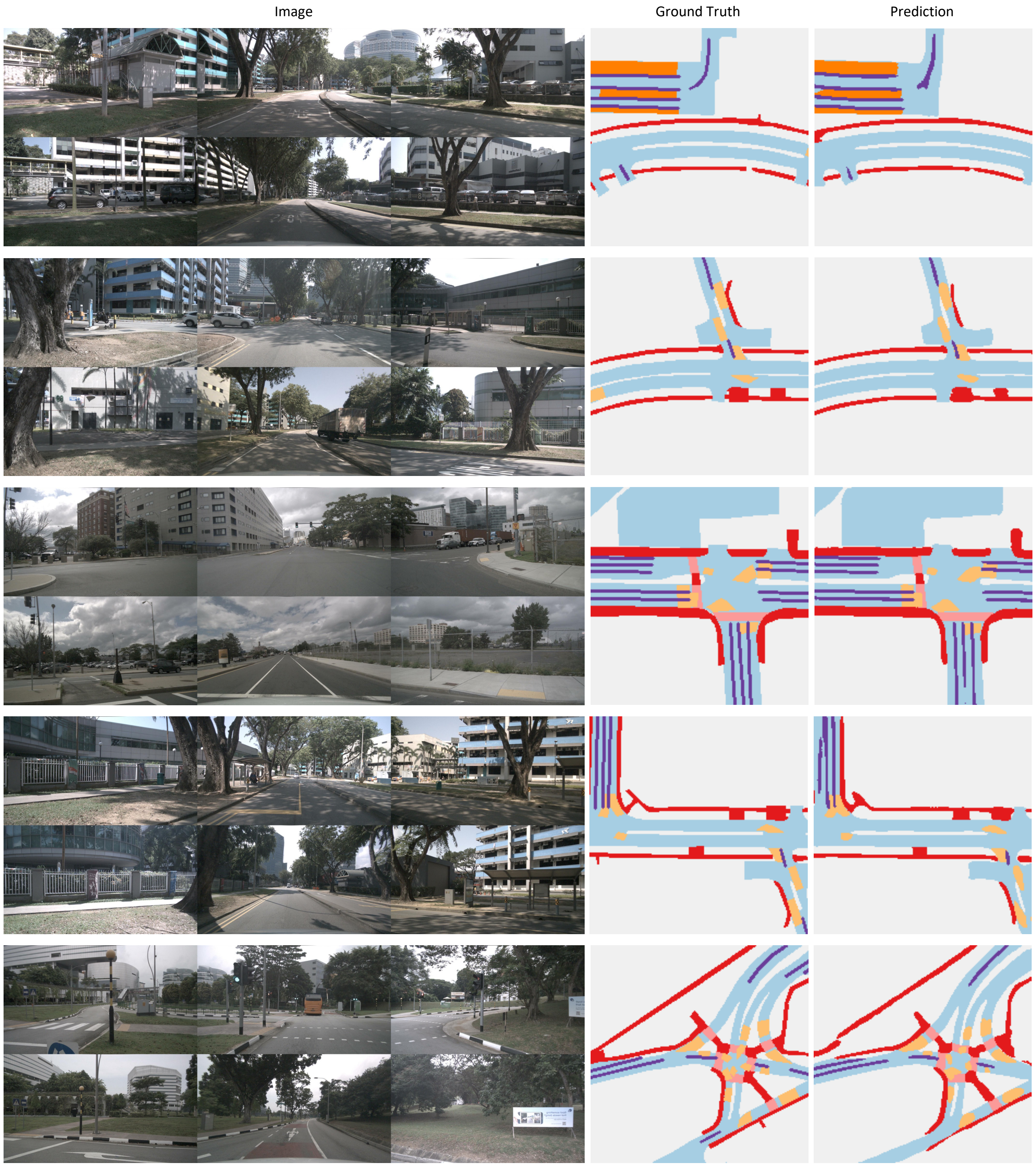

Figure 7. **Visualization of predicted BEV map segmentation results on nuScenes val set.**

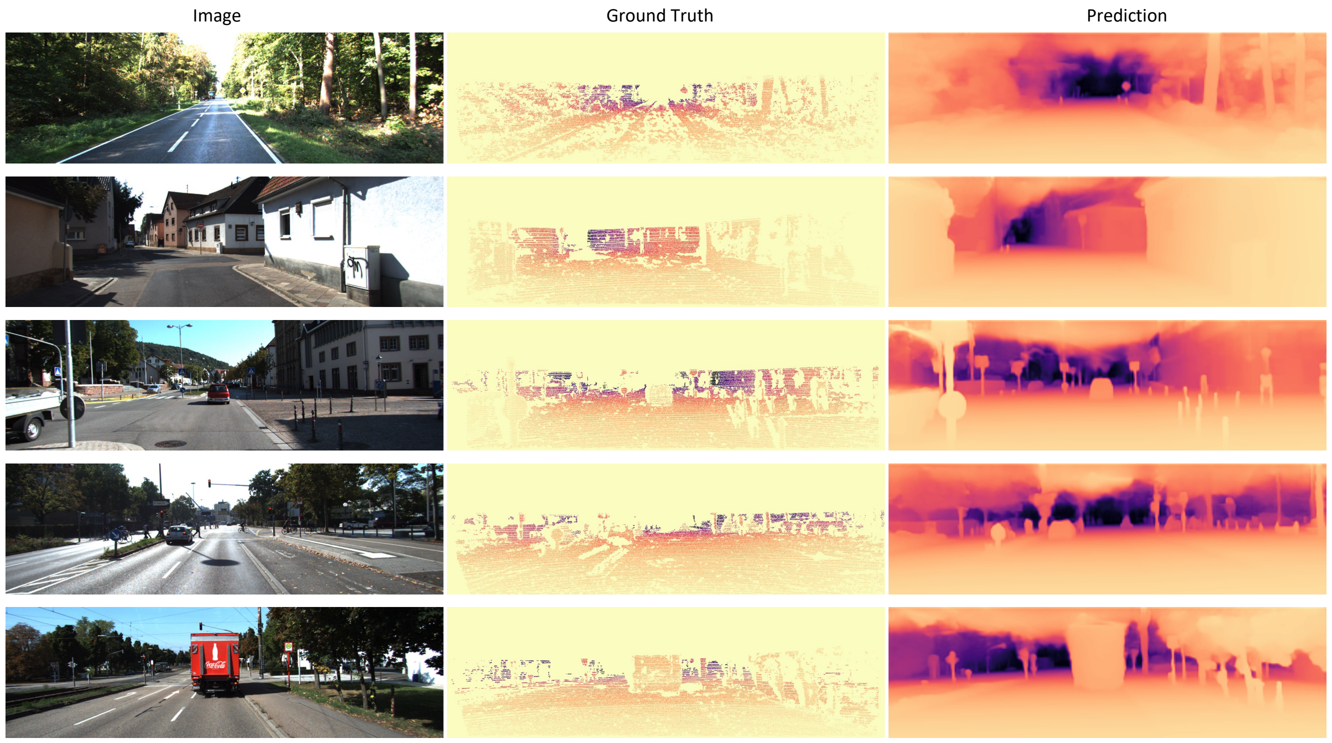

Figure 8. **Visualization of predicted depth estimation results on KITTI val set.**

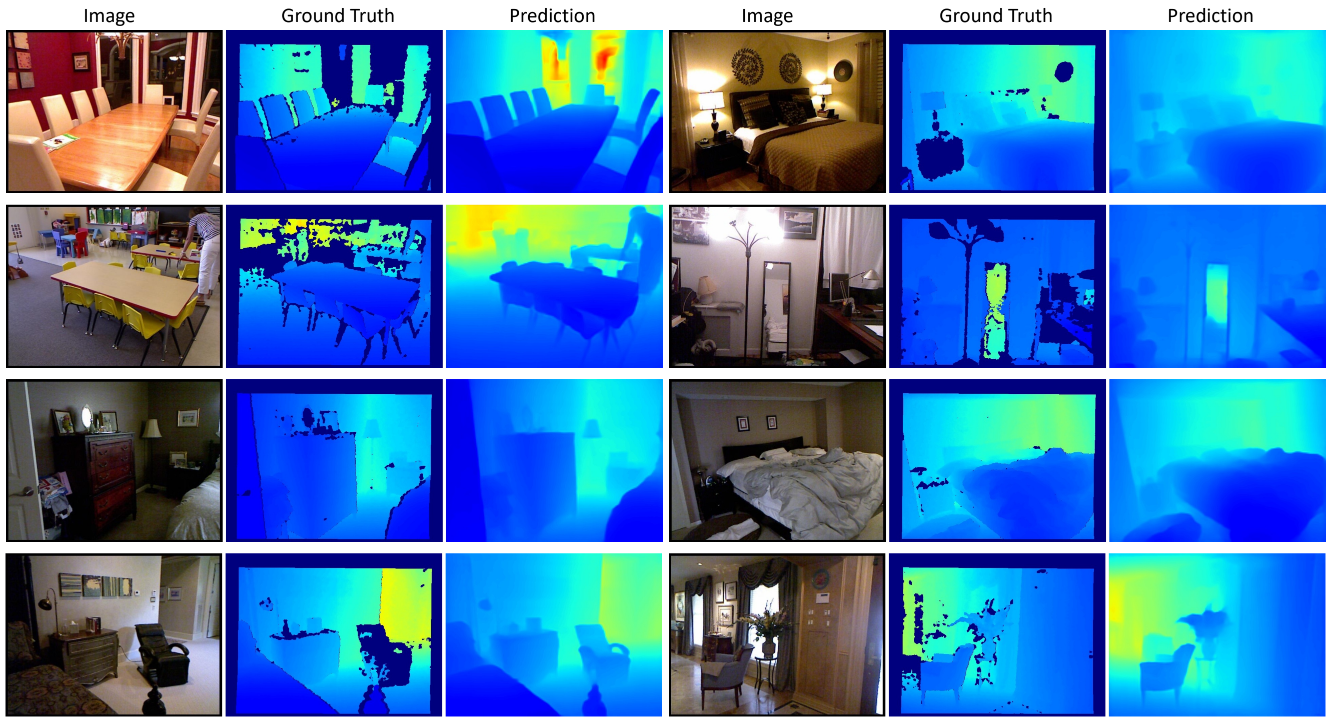

Figure 9. **Visualization of predicted depth estimation results on NYU-DepthV2 val set.**

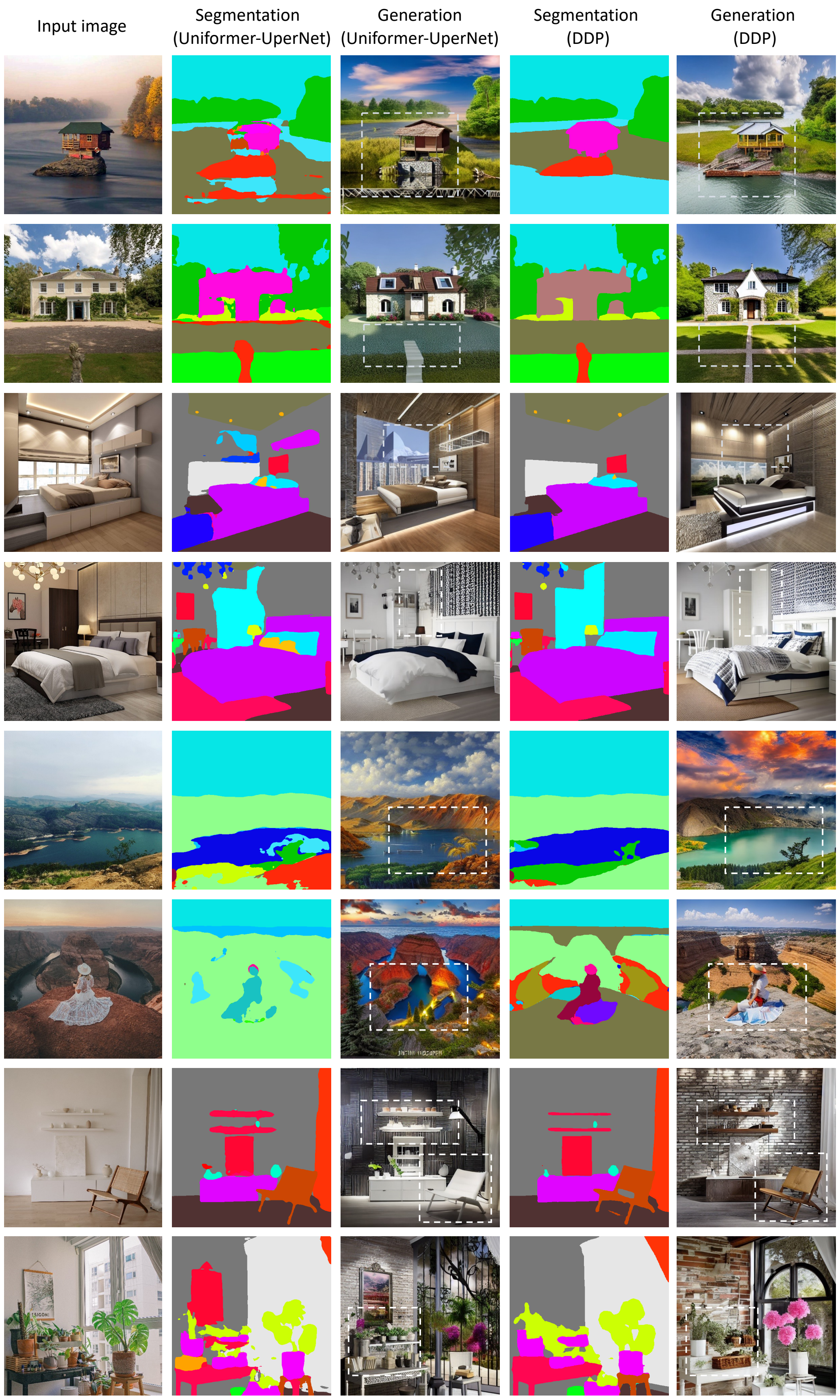
Input image
Segmentation (Uniformer-UperNet)
Generation (Uniformer-UperNet)
Segmentation (DDP)
Generation (DDP)

Figure 10. **Control Stable Diffusion with Semantic Map**, the Uniformer-UnperNet, and DDP segmentation models are used to predict segmentation maps as condition input. All results were achieved using the default prompt.