UnCommon Objects in 3D

Supplementary Material

A. CAT3D-like model details

This section provides more details for the CAT3D-like model used in Sec. 4.2. CAT3D [19] is a diffusion model which takes as input a set of N_{tgt} target cameras and a set of N_{src} source views with cameras, and aims at generating the views associated with the target cameras. Since the code is not available, we follow the implementation details of [19] to reproduce a model with similar capabilities. Specifically, starting from a pre-trained text-to-image latent diffusion model similar to [48], we first modify all 2D self-attention layers in the decoder part of the denoising UNet such that 2D self-attention is performed across all the views in the batch. First proposed by [53], this cross-view attention allows each image token to attend to tokens of all views in the batch, thus improving multi-view consistency. Then, we modify the architecture with zero-initialized channel expansion such that it can take as input the latent features concatenated with mask maps indicating source views and camera maps in the form of Plücker rays. Different from [19], we use the v-prediction / v-loss parametrization [49] and the zero terminal SNR noise scheduling recommended by [37] as we found it to work better than the original CAT3D recipe. For all experiments, we use $N_{\text{src}} = 3$, $N_{\text{tgt}} = 5$ and train for 100k iterations using Adam [30] optimizer with a constant learning rate of $1e^{-5}$ and a global batch size of 64.

For evaluation, we follow standard practices [19, 67] and report results on common out-of-distribution NVS datasets (RealEstate10K, LLFF, DTU, Mip-NeRF 360) using the same test splits. We evaluate novel-view synthesis in the 3-view input setting and report LPIPS and PSNR metrics.

B. Text-to-3D model details

The text-to-image stage of the Instant3D-like model from Sec. 4.3 is based on an internal text-to-image model architecturally similar to Emu [11]. Starting from a model pre-trained on a dataset of image-caption pairs, we fine-tune the model on 4-view canonical-render grids of uCO3D objects. In all our experiments we use the Adam optimizer [31] with a batch size of 160 and a constant learning rate of $1e^{-5}$. We distribute the training across 32 NVIDIA A100 GPUs for a total of 20k steps. During inference, we use a Diffusion Probabilistic Model (DPM) Sampler [40] and denoise over 60 steps. The 4-view-to-3D stage of the Instant3D-like model is based on LightplaneLRM [6]. For the LightplaneLRM model which reconstructs both the central object and the scene background, we use the coordinate contraction of MERF [46] which non-linearly maps the distant parts of the scene so they always fall into the [-1,1] bounding cube of the utilizied triplane representation. The rest of the training procedure follows the LightplaneLRM protocol described in Sec. 4.1.

We train three different versions of both the Instant3D-like model and LightplaneLRM, each version corresponding to a different dataset. Specifically, we train on a dataset of synthetic assets similar to Objaverse [13], and on two versions of uCO3D, one that contains background and one where the background information is masked. For evaluation, we report the FID metric [25] for the models trained on datasets without background information (Tab. 4). The evaluation sets corresponding to the **Surreal** and **Real** prompts are created by randomly selecting 50 image frames for every scene/prompt pair. We center the objects of the uCO3D images using the per-frame mask information for consistent evaluation across all datasets. The generated 3D objects of the text-to-3D model are rendered from sampled cameras drawn from the camera distribution of each individual evaluation set. The qualitative results presented in Fig. 8 are extracted using the model variants trained with background information.

C. Rigid scene alignment

In Sec. 3, we described a procedure that estimates a rigid transform for each object to align it to dataset-wide object-centric reference. Here, we provide additional details.

We start with finding the gravity axis by making sure the roll of the cameras is close to 0, following [57]. Then, we translate and scale the densified point cloud ((b) in Fig. 3) so that the median locations along the horizontal axes are 0, and so that the STD of its points' coordinates is 1. Then, we normalize the 2D rotation in the horizontal plane by aligning the principal components, and, finally, shift the object vertically to make the ground plane's elevation zero. As shown in the experiments, this normalization allows rendering each object from 4 canonical viewpoints defined in the object-centric reference, which eventually enables the Instant3D-like text-to-3D model training.