Appendix

A. Video Results

We urge readers to view the additional video generation results on the 3D-FRONT and KITTI scenes. We show our novel rendering videos in comparison to EG3D [7] and GSN [15]. The videos are best viewed by opening our project page. We note that our model consistently produces more realistic scenes that are view-consistent, whereas the generated scenes from our baselines have more artifacts. Furthermore, to showcase that our generated scenes have more accurate geometries, we visualize the corresponding depth maps for our generated scenes. We observe that in comparison to EG3D [7] our depth maps are consistently plausible, less noisy and better capture the 3D object geometries as well as the backgrounds.

More specifically, we note that the results of EG3D on the bedroom datasets suffer not only from the lower visual qualities but also from the structural errors of creating two symmetric scenes about the feature plan. Moreover, in the more challenging living room scenes with many objects, GSN and EG3D fail to generate recognizable objects. In contrast, CC3D consistently synthesizes high-quality, view-consistent scenes. Finally, on the KITTI-360 scenes, EG3D and GSN suffer from significant artifacts in both RGB and depths with large view changes.

B. Layout Consistency

We conduct an experiment to measure the layout consistency between the input semantic layouts and output top-down renderings of the scenes. To this end, we render all 5515 bedroom scenes with our trained models from top-down views. To test the effects of the layout consistency loss, as introduced in Sec. 3.3 of the main document, we obtain two generated top-down images, one rendered from the model trained with and the other one trained without the layout consistency loss. We pair the top-down renderings with ground truth semantic labels and separate the obtained datasets into train and test splits using a ratio of 0.85. Then, we train a DeepLabv3 [9] semantic segmentation model to predict the ground truth semantic segmentations from the rendered town-down images. Essentially, the more consistent the layouts and the top-down renderings the easier it becomes for the segmentation model to learn. Therefore, better test performance likely indicates higher consistencies. In Tab. 3, we indeed observe an improved IoU numbers when training our model with layout consistency loss.

In addition, we conducted another experiment where we trained Faster R-CNN [45] on top-down renderings to detect instances w.r.t. the input layout and evaluate using average precision (AP) and average recall (AR) with the COCO evaluator. The consistency metrics are generally high and we observe improved numbers (AP: 0.774, AR: 0.809) against the model without the consistency loss (AP: 0.719, AR: 0.747) in Eq. 4 of the main paper.

Table 3. **Quantitative measures of semantic consistency** – between input layouts and top-down rendering outputs on 3D-FRONT Bedroom scenes.

Method	IoU ↑
Ours	0.7182
w/o Layout Consistency loss	0.6593

C. Implementation Details

Below we share the implementation details of our training and testing pipelines. To promote reproducibility, we will release the code to the public upon acceptance.

Baselines. We use the R1 regularization [34] across all datasets and models. We replace the sphere-based camera sampling in EG3D [7] and GIRAFFE [39] to allow freely moving cameras, which are significantly challenging to learn from. As stated in the main text, we sampled the cameras by first performing the distance transform and then randomly sampling locations where the distance value is above the threshold. For bedrooms, we orient the camera toward a randomly sampled point within the bed bounding box and for living rooms, we used the largest object in the scene. For GSN [15], we do not use depth maps as supervision and train on single-view image collections in the same setup as all other models. We train all models using 3,000,000 images for all datasets and models with batch size 32.

From our qualitative evaluations in the main paper, we note that the generated scenes of GIRAFFE [39] are quite dark. We observed that the low brightness of GIRAFFE results can be prevented with higher R1 regularization, however, this led to many samples being completely black and higher FID scores. Hence, we used the regularization factor which obtains the lowest FID score for a fair comparison.

Even though we show several generated scenes using GSN [15], we would like to note that training GSN on 3D-FRONT living rooms and KITTI-360, even with hyperparameter tuning was not trivial and resulted in the model collapsing. We hypothesize that GSN requires depth supervision for better performance on these datasets.

Training Details. We build our model on top of the EG3D [7] pipeline, hence we follow their implementation protocol and hyperparameters unless otherwise stated. We use a discriminator learning rate of 0.002 and a generator learning rate of 0.0025 with Adam optimizer using $\beta_1 = 0$, $\beta_2 = 0.99$, and $\epsilon = 10^{-8}$. Our mapping network transforms a 512 latent code vector into an intermediate latent code with 2 fully connected layers of dimension 512. We do not apply

any pose conditioning to the generator or the discriminator networks. As stated above, we abandon the sphere-sampling of camera locations. Our model takes around 2 days to converge using 4 NVIDIA V100 with 32 GB memory.

Semantic Layout Details. As mentioned in the main document, we process datasets-dependent layouts $\mathbf{L}$ as conditional inputs to our model. To prepare the input $\mathbf{L}$ we discretize the provided semantic 2D floor plans onto the 2D grids. For indoor scenes, i.e., 3D-FRONT bedroom and living room scenes, we simply project the 3D bounding boxes of the scenes onto the ground plane and encode the semantic class of each pixel using a one-hot vector and a binary room layout mask. The semantic feature channels are concatenated with the local coordinates of each bounding box (origin at the left-top corner of the bounding boxes at their canonical orientations), providing orientation information to the subsequent U-Net. In total we concatenate features comprising i) a binary mask of the room layout (1), ii) local coordinates of each object (3), iii) one-hot embedding of the semantic layout (16), and iv) the global latent noise (512). This results in a feature grid with 532 channels for 3D-FRONT. We directly obtain a semantic floorplan representation for outdoor scenes by rendering the 3D semantic annotations from a top-down view. Hence, the semantic maps are pixel-based as opposed to bounding-box-based. For KITTI-360 we concatenate features comprising i) the top-down rendered semantic layouts encoded as a one-hot feature grid (59) and ii) latent noise (512). This results in a feature grid with 571 channels for KITTI-360.

C.1. Architecture Details

Our U-Net is composed of an encoder and a decoder network, each of which is composed of the building blocks of StyleGAN2. For the encoder, we use the StyleGAN2 synthesis layers except that we replace the upsampling with the max-pool downsampling operation and use style-modulation with *constant* style code (i.e., the encoder network is independent of the sampled style code $\boldsymbol{s}$). The downsampling layers are repeated to make the feature resolution 4^2. We turned off the style modulation with the global latents in order to keep the encoder deterministic and only modulate the decoder part of the U-Net. Our U-Net decoder closely follows the StyleGAN2 architecture and starts with a learnable feature with 4^2 resolution. We use skip connections to concatenate the encoder features to the intermediate features of the corresponding decoding layer. In contrast to the encoder, we modulate the decoder via FiLM with the per-layer style code that is obtained by processing the global style noise vector $\mathbf{s}$ with the StyleGAN2 mapping network.

The discriminator architecture follows that of StyleGAN2, except that we attach a decoder network that is symmetric with the encoder network. The two network components are connected via skip connections as in a typical U-Net. We use $k = 8$ for the segmentation branch.

Table 4. **Quantitative ablation studies** on 3D-FRONT living rooms. We measure the realism of generated 3D scenes without using 2D layout conditioning (i.e., unconditional version of our model) or using the layout consistency loss described in Sec. 3.3. Moreover, we swap out our 3D extrusion representation with the "floorplan" and tri-plane schemes, proving the advantage of our method.

Method	FID (↓)	KID (↓)
Ours	**40.3**	**34.5**
w/o Layout Conditioning	60.1	54.1
w/o Layout Consistency Loss	44.7	38.0
w/ GSN's Floorplan Representation	65.6	59.0
w/ EG3D's Tri-plane Representation	69.3	60.8

D. Discussions

D.1. Note on Tri-Plane Results

As discussed in the main text, we hypothesize that the tri-plane representation is conceptually not ideal for representing large-scale scenes due to weak geometric inductive bias when generating the tri-plane jointly. Moreover, as the scene gets larger, the same plane-projected features are used to describe totally different objects in a scene, which hampers the representational power of tri-planes. Indeed, in the included video websites, one can observe that the EG3D results contain artifacts where the scene contains two bedrooms, symmetric about one of the three planes (see Fig. 11). We also note that the ablated version of our conditional model using the Tri-plane representation suffers from severe layout inconsistencies. That is, we observe that the input layout is almost completely ignored and the output scenes have almost no resemblance to the input layouts, which clearly indicates that the tri-plane representation lacks geometric inductive bias in our use-case.

D.2. Note on "Floorplan" Results

We discussed in the main text how the "floorplan" representation requires a larger MLP because the vertical information needs to be decoded, or "generated" by the MLP network. Indeed, we observe that GSN [15] adopts an 11-layer MLP with 128 channels in the hidden layers. In comparison, EG3D and our representation require using a two-layer MLP with 64 channels in the hidden layer. Approximately, our MLP network size (in number of weights) is less than 20 times smaller than that of GSN's.

In our ablation study in the main text (Tab. 2), we swap our "extrusion" representation with a "floorplan" representation. Here, to make the comparison fair, we used the same two-layer MLPs for the experiment. Note that, while increas-

ing the size of the MLP might improve its performance, it comes at a significant cost of computational resources.

E. Additional Results

In Fig. 12 and Fig. 13, we show additional visualizations of our conditional generation results. Note that the generated 3D scenes generally follow the input layouts. Moreover, we sample three different global latent vectors which, when applied to the generation process, synthesize scenes with different styles. In Fig. 14, we demonstrate the object removal capability. Note how we can remove individual objects such as a chair and a coffee table. In Fig. 16 we visualize KITTI-360 layouts and renderings. While our model generates better image quality and view consistency than previous works, we acknowledge that the current model has difficulties closely following complex layouts.

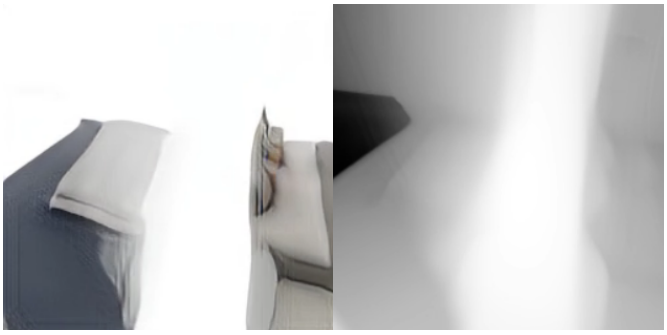

Figure 11. **Failure case of EG3D [7]** – on 3D-FRONT bedroom. We notice that the tri-plane representation induces replicating the scenes that are symmetric about one of the feature planes.

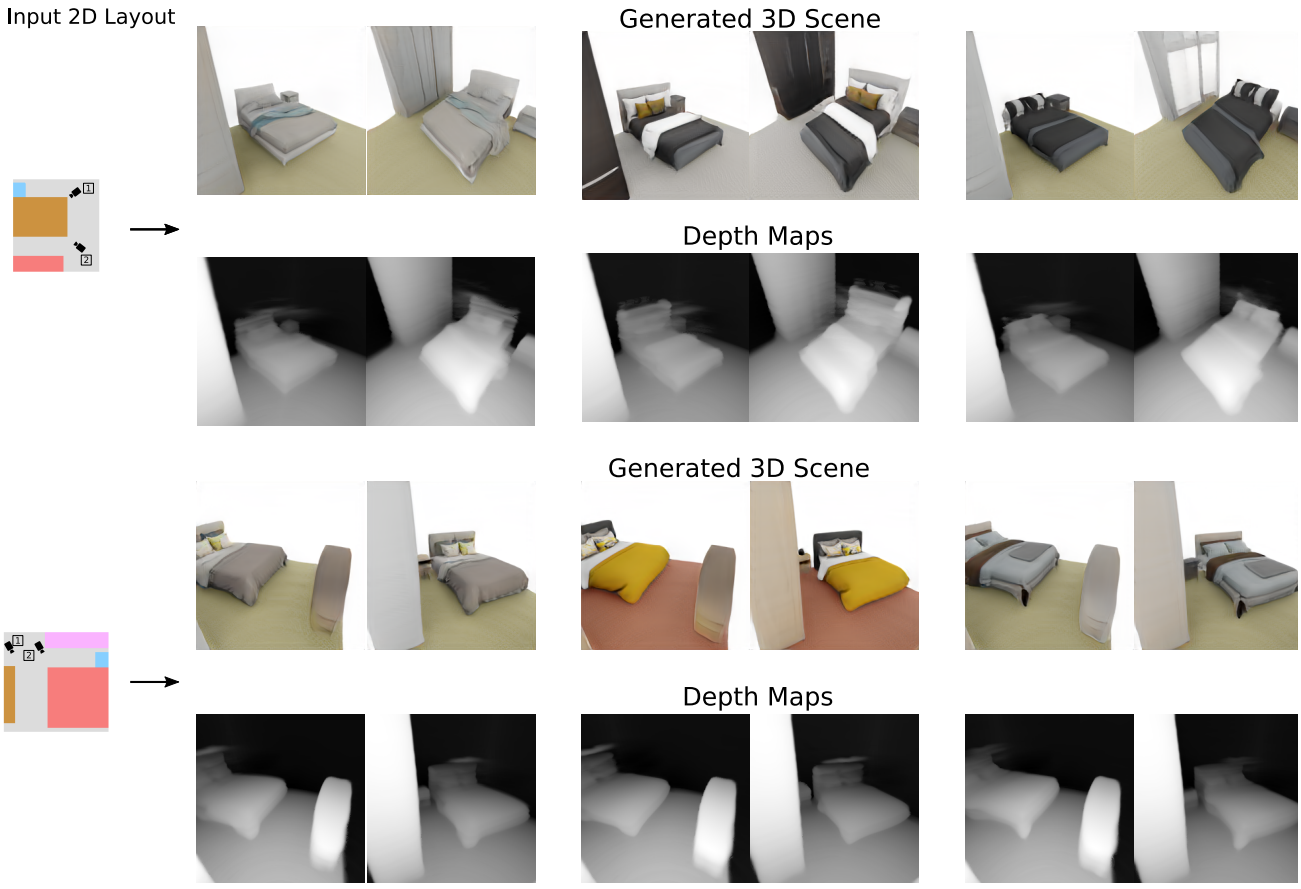

Figure 12. **Additional results on 3D-FRONT.** We visualize the conditional generation results on 3D-FRONT bedrooms with varying latent codes (shown in three styles). Note that changing the global latent codes results in a change of general styles.

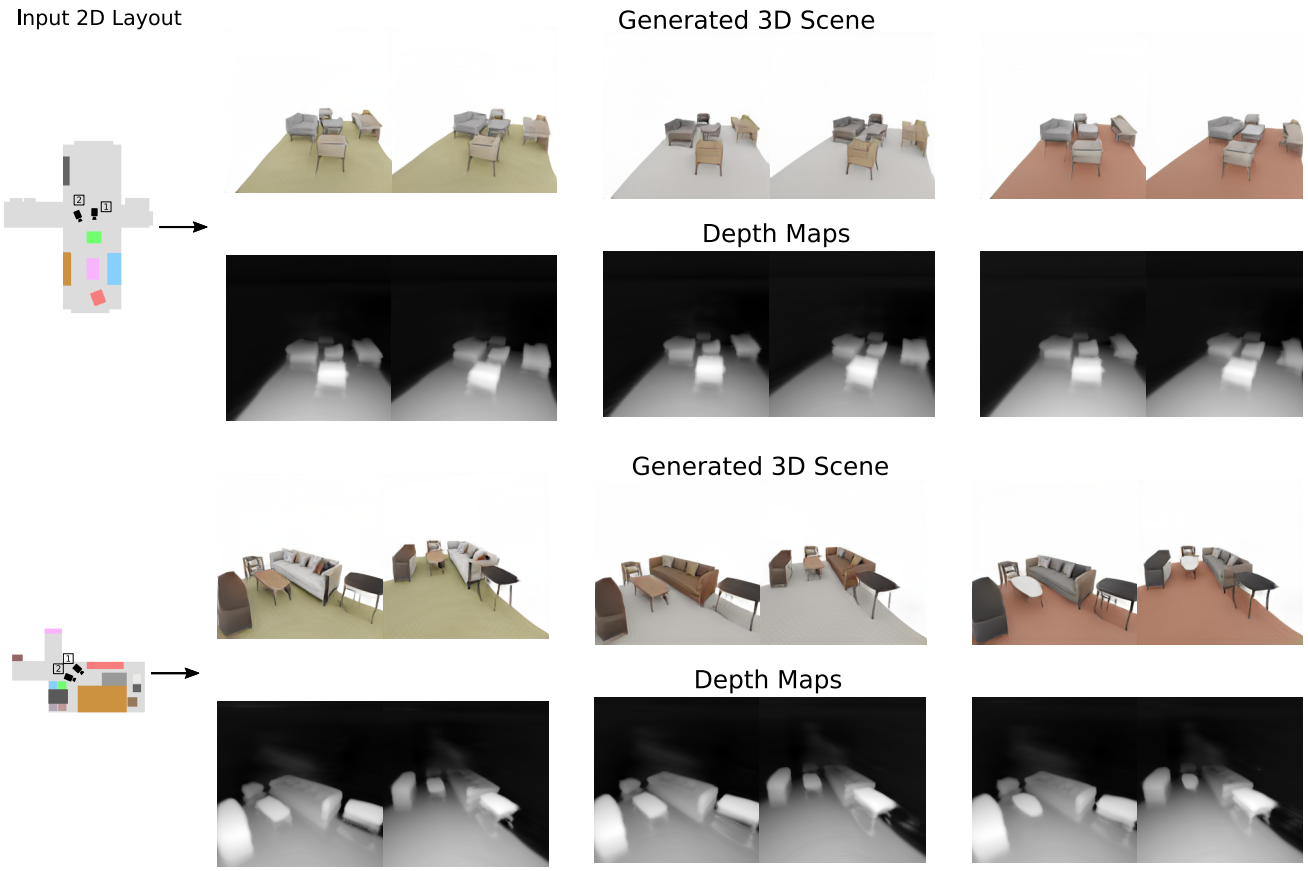

Figure 13. **Additional results on 3D-FRONT.** We visualize the conditional generation results on 3D-FRONT living rooms with varying latent codes (shown in three styles). Note that changing the global latent codes results in a change of general styles. We notice that for living room scenes the layout conditionings are not perfectly respected. For example, the big sofa bounding box of the second example is splitted into a sofa and a side table.

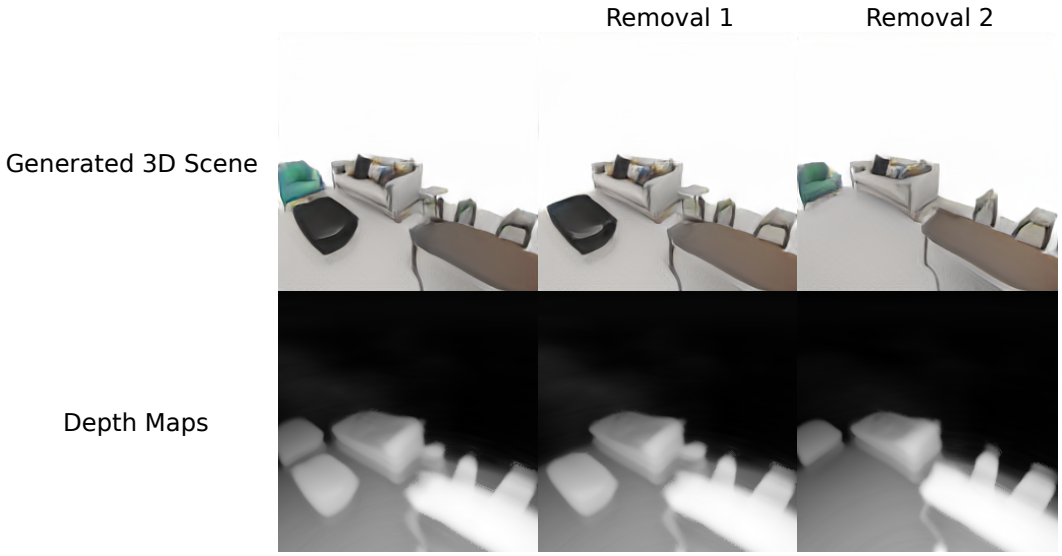

Figure 14. **Object removal experiment.** We showcase the object removal capability of our approach. Note that from the image on the leftmost column, we can remove the green sofa chair (middle column) and the black coffee table (right column).

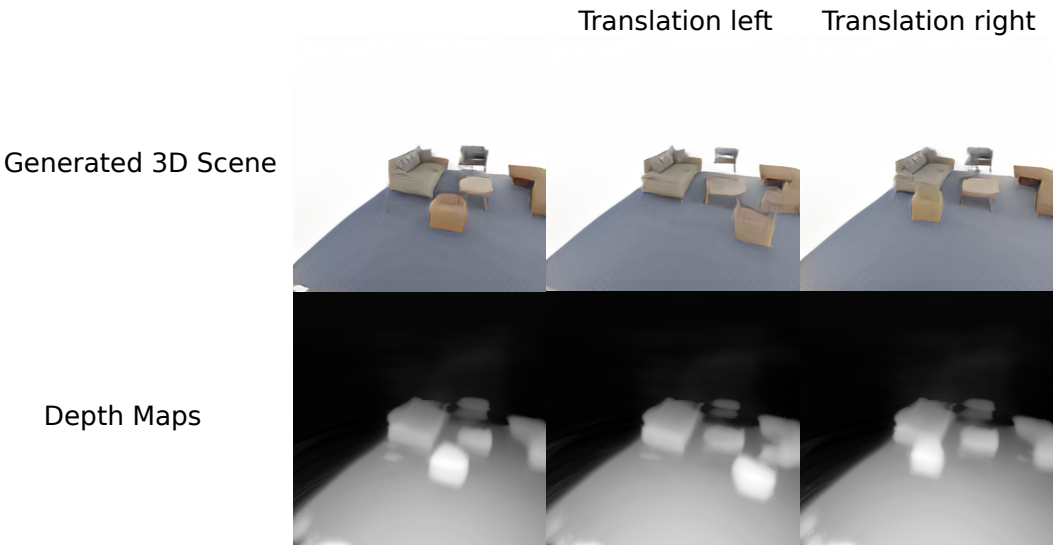

Figure 15. **Translation experiment.** We showcase the object translation capability of our approach. Note that from the image on the leftmost column, we can translate the object to the left (middle column) and right (right column).

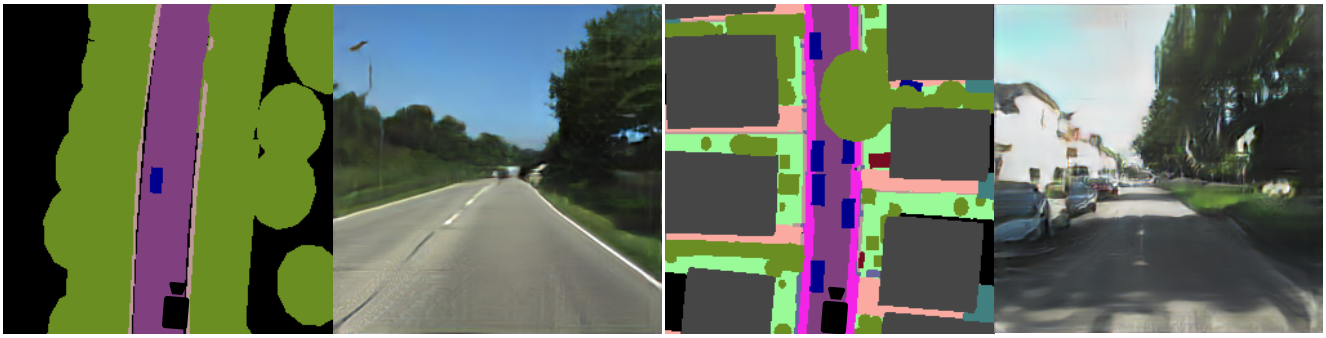

Figure 16. **KITTI-360 conditioning.** Layout inputs and generated 3D scenes for KITTI-360.