A. Architecture details and hyperparameters

The inverse renderer is composed of 3 submodels: a 2D convolutional network mapping images to 2D features, an inverse projection layer mapping 2D features to 3D features and a 3D convolutional network mapping 3D features to the scene representation. Each subnetwork is described in detail in the tables below. The renderer is simply the transpose of the inverse renderer with a sigmoid activation at the ouput layer to ensure pixel values are in $[0, 1]$.

Note that every layer is followed by a GroupNorm layer and a LeakyReLU activation (except the final scene and image layers). Each ResBlock is composed of a sequence of 1×1, 3×3, 1×1 convolutions added to the identity.

To ensure rotations of the scene representation do not exit the bounds of the voxel grid we apply a spherical mask to the scene representation before performing rotations.

The full model has 11.5 million trainable parameters.

INPUT SHAPE	OUTPUT SHAPE	OPERATION
(3, 128, 128)	(64, 128, 128)	1X1 CONV
(64, 128, 128)	(64, 128, 128)	2X RESBLOCK
(64, 128, 128)	(128, 64, 64)	4X4 CONV, STRIDE 2
(128, 64, 64)	(128, 64, 64)	1X RESBLOCK
(128, 64, 64)	(128, 32, 32)	4X4 CONV, STRIDE 2
(128, 32, 32)	(128, 32, 32)	1X RESBLOCK
(128, 32, 32)	(256, 16, 16)	4X4 CONV, STRIDE 2
(256, 16, 16)	(256, 16, 16)	1X RESBLOCK
(256, 16, 16)	(128, 32, 32)	4X4 CONV.T, STRIDE 2
(128, 32, 32)	(128, 32, 32)	2X RESBLOCK

Table 1. Architecture of 2D subnetwork.

INPUT SHAPE	OUTPUT SHAPE	OPERATION
(128, 32, 32)	(256, 32, 32)	1X1 CONV
(256, 32, 32)	(512, 32, 32)	1X1 CONV
(512, 32, 32)	(1024, 32, 32)	1X1 CONV
(1024, 32, 32)	(32, 32, 32, 32)	RESHAPE

Table 2. Architecture of inverse projection network from 2D to 3D.

INPUT SHAPE	OUTPUT SHAPE	OPERATION
(32, 32, 32, 32)	(32, 32, 32, 32)	1X1 CONV
(32, 32, 32, 32)	(32, 32, 32, 32)	2X RESBLOCK
(32, 32, 32, 32)	(128, 16, 16, 16)	4X4 CONV, STRIDE 2
(128, 16, 16, 16)	(128, 16, 16, 16)	2X RESBLOCK
(128, 16, 16, 16)	(64, 32, 32, 32)	4X4 CONV.T, STRIDE 2
(64, 32, 32, 32)	(64, 32, 32, 32)	2X RESBLOCK

Table 3. Architecture of 3D subnetwork.

Hyperparameters. When training with $\ell_1 + \text{SSIM}$ loss, we set the weight of the SSIM loss to 0.05.

Training. We train each model for 100 epochs on all datasets, although most models converge much earlier than this (around 60 epochs). When training on a single GPU we use a batch size of 16 and when training on 8 GPUs we use a batch size of 112.

Optimizer. We use Adam with a learning rate of 2e-4.

Losses. We use the ℓ_2 loss for quantitative comparisons as PSNR is inversely proportional to ℓ_2. Indeed, the baselines we compare against (except TCO) all directly optimize ℓ_2, making comparisons fairer. We generally found that $\ell_1 + \text{SSIM}$ produces more visually pleasing samples and therefore use this loss for qualitative comparisons and novel view synthesis.

B. Dataset descriptions

Detailed descriptions of the ShapeNet chairs and cars dataset can be found in the appendix of Sitzmann et al..

B.1. MugsHQ

The MugsHQ data set was rendered with a branch of the Mitsuba Renderer (Jakob, 2010) adapted to import ShapeNet geometry (Shi, 2014). Every scene was rendered with the same environment map (lighting conditions) and checkerboard disk platform. ShapeNet objects were scaled by their largest bounding box dimension, centered, and placed on the platform. The object's material is a two-sided plastic designed to highlight glossy reflections and the diffuse reflectance color was randomly sampled. For each object, 150 viewpoints were uniformly sampled over the upper hemisphere. Each viewpoint was rendered to a 256×256 high dynamic range image, and then resized and tone-mapped to a linear RGB image.

B.2. 3D Mountains

We created the 3D mountains dataset by first scraping the height, latitude and longitude of the 559 highest mountains in the Alps. We then used Apple Maps to render 50 images of each mountain. Specifically, the camera was placed on a sphere of radius 600m centered on the latitude, longitude and height - 100m of the mountain. We then fixed the elevation angle to be 55 degrees (or a pitch of 35 degrees) and randomly sampled the azimuth angle between 0 and 360 degrees to capture multiple views of each mountain.

C. Train/validation/test splits

For each dataset we train a model and choose hyperparameters based on the lowest validation loss. All images and quantitative measurements are then made on a held out test set which is only seen after everything else has been fixed.

C.1. Chairs

The chairs dataset consists of 6591 scenes, with the training and validation set each having 50 views per scene and the test set having 251 views per scene, for a total of 594,267 images. The train/validation/test splits are:

- Train: 4612 scenes (230,600 images)
- Validation: 662 scenes (33,100 images)
- Test: 1317 scenes (330,567 images)

C.2. Cars

The cars dataset consists of 3514 scenes, with the training and validation set each having 50 views per scene and the test set having 251 views per scene, for a total of 317,204 images. The train/validation/test splits are:

- Train: 2458 scenes (122,900 images)
- Validation: 352 scenes (17,600 images)
- Test: 704 scenes (176,704 images)

C.3. MugsHQ

The MugsHQ dataset consists of 214 scenes, each with 150 views for a total of 32,100 images. The train/validation/test splits are:

- Train: 186 scenes (27,900 images)
- Validation: 14 scenes (2,100 images)
- Test: 14 scenes (2,100 images)

C.4. 3D mountains

The 3D mountains dataset consists of 559 scenes, each with 50 views for a total of 27,950 images. The train/validation/test splits are:

- Train: 478 scenes (23,900 images)
- Validation: 26 scenes (1,300 images)
- Test: 55 scenes (2,750 images)

D. Runtimes

D.1. Training time

Training time for all datasets are shown in Table 4. When training on a single V100 GPU we use a batch size of 16, whereas we use a batch size of 112 when training on 8 V100s.

DATASET	V100	8 V100S
CHAIRS	9.7 DAYS	2.2 DAYS
CARS	5.5 DAYS	1.3 DAYS
MUGSHQ	1.2 DAYS	6 HRS
MOUNTAINS	1 DAY	5 HRS

Table 4. Training times.

D.2. Inference time

We measured inference time with a trained model on the cars dataset running on a single Tesla V100 GPU. We took the mean and standard deviation over 1000 iterations (using 100 warmup steps).

Single image: $\mathbf{21.9 \pm 0.3}$ **ms**

Batch of 128 images: $\mathbf{1578.6 \pm 10.2}$ **ms** (**12.3 ms** per image)

Note that for a single image this corresponds to a framerate of 45 fps, allowing for real time inference.

E. Things that didn't work

We experimented with several things which we found did not improve performance.

- We experimented with partitioning the latent space (across channels) into a viewpoint invariant and equivariant part. We hypothesized this might help in learning complex textures and create something akin to a global texture map, but found that this did not decrease (nor increase) the loss in practice.
- When rotating the voxels we use trilinear interpolation to calculate the value of points that do not align with the grid. While rotations on the grid will always suffer from aliasing we hypothesized that using nearest neighbor interpolation (instead of trilinear) could help model performance. We also tried using shear rotations as these have been shown to reduce aliasing in certain cases (Paeth, 1986). In practice we found that this did not make a big difference.
- The latent space we use in our model has shape $64 \times 32 \times 32 \times 32$. We hypothesized that increasing the spatial resolution might help improve performance. We therefore tried a latent space of size $8 \times 64 \times 64 \times 64$ but found that this performed the same as the original latent space, but was much slower to train.

F. Samples from datasets

We include random ground truth samples from the MugsHQ and 3D mountains dataset.

Figure 1. Random samples from the MugsHQ dataset.

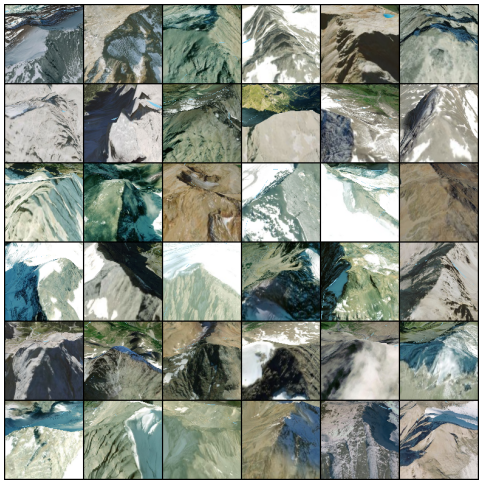

Figure 2. Random samples from the 3D mountains dataset.

G. Random samples from model

We include random novel view synthesis samples on all datasets.

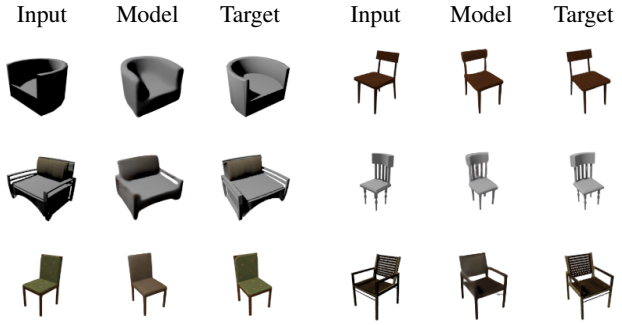

Figure 3. Random single shot novel view synthesis samples on chairs.

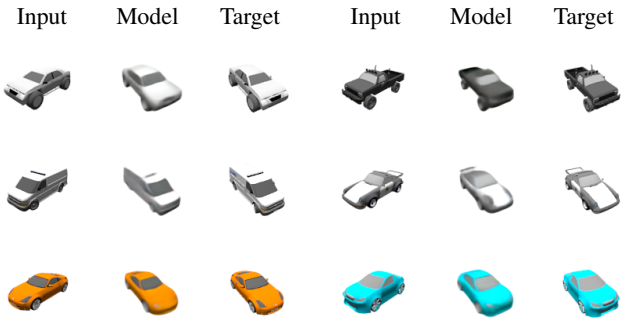

Figure 4. Random single shot novel view synthesis samples on cars.

References

Jakob, W. Mitsuba renderer, 2010. http://www.mitsuba-renderer.org.

Paeth, A. A fast algorithm for general raster rotation. In *Proceedings on Graphics Interface'86/Vision Interface'86*, pp. 77–81. Canadian Information Processing Society, 1986.

Shi, J. Mitsuba for shapenet, 2014. https://github.com/shi-jian/mitsuba-shapenet.

Sitzmann, V., Zollhöfer, M., and Wetzstein, G. Scene representation networks: Continuous 3d-structure-aware neural scene representations. *arXiv preprint arXiv:1906.01618*, 2019.

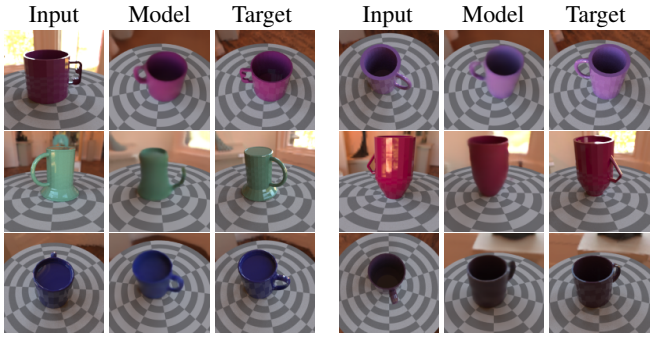

Figure 5. Random single shot novel view synthesis samples on MugsHQ.

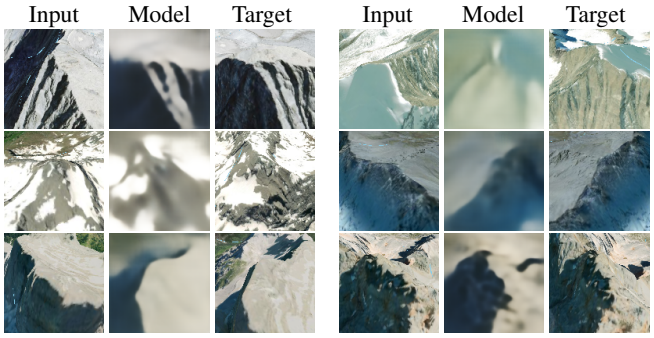

Figure 6. Random single shot novel view synthesis samples on 3D mountains.