GANcraft: Unsupervised 3D Neural Rendering of Minecraft Worlds

Zekun Hao*†, Arun Mallya*, Serge Belongie†, Ming-Yu Liu*
*NVIDIA, †Cornell University
{hz472, sjb344}@cornell.edu, {amallya, mingyul}@nvidia.com

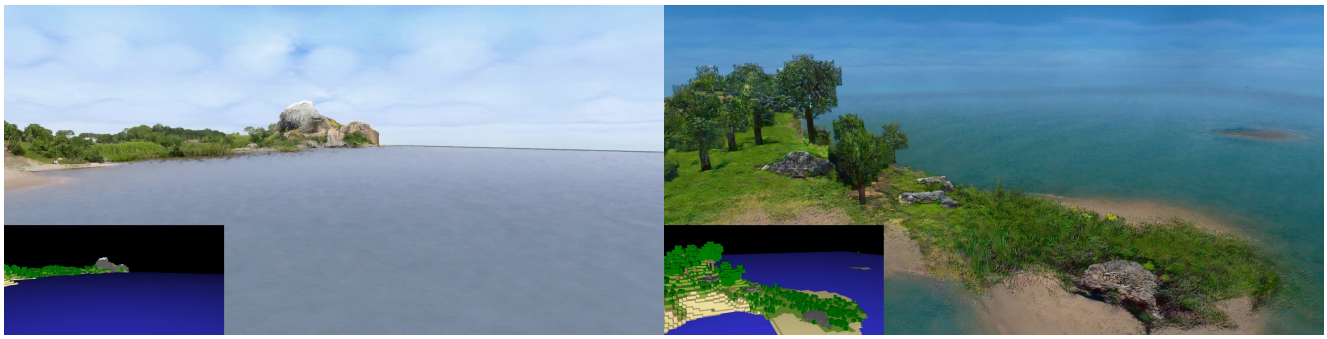

Figure 1: Given a semantically-labeled block world as input (insets), GANCraft generates high-resolution view-consistent realistic outputs. It unsupervisedly learns to translate the input world to a realistic-looking world in the absence of paired training data across these two worlds. *Click on image to play video in web browser.*

Abstract

We present GANcraft, an unsupervised neural rendering framework for generating photorealistic images of large 3D block worlds such as those created in Minecraft. Our method takes a semantic block world as input, where each block is assigned a semantic label such as dirt, grass, or water. We represent the world as a continuous volumetric function and train our model to render view-consistent photorealistic images for a user-controlled camera. In the absence of paired ground truth real images for the block world, we devise a training technique based on pseudo-ground truth and adversarial training. This stands in contrast to prior work on neural rendering for view synthesis, which requires ground truth images to estimate scene geometry and view-dependent appearance. In addition to camera trajectory, GANcraft allows user control over both scene semantics and output style. Experimental results with comparison to strong baselines show the effectiveness of GANcraft on this novel task of photorealistic 3D block world synthesis. The project website is available at https://nvlabs.github.io/GANcraft/.

1. Introduction

Imagine a world where every Minecrafter is a 3D painter!

Advances in 2D image-to-image translation [3, 22, 49] have enabled users to *paint* photorealistic images by drawing simple sketches similar to those created in Microsoft Paint. Despite these innovations, creating a realistic 3D scene remains a painstaking task, out of the reach of most people. It requires years of expertise, professional software, a library of digital assets, and a lot of development time. In contrast, building 3D worlds with blocks, say physical LEGOs or their digital counterpart, is so easy and intuitive that even a toddler can do it. Wouldn't it be great if we could build a simple 3D world made of blocks representing various materials (like Fig. 1 (insets)), feed it to an algorithm, and receive a realistic looking 3D world featuring tall green trees, ice-capped mountains, and the blue sea (like Fig. 1)? With such a method, we could perform *world-to-world* translation to convert the worlds of our imagination to reality. Needless to say, such an ability would have many applications, from entertainment and education, to rapid prototyping for artists.

In this paper, we propose GANcraft, a method that produces realistic renderings of semantically-labeled 3D block worlds, such as those from Minecraft (www.minecraft.net). Minecraft, the best-selling video game of all time with over 200 million copies sold and over 120 million monthly users [2], is a sandbox video game in which a user can explore a procedurally-generated 3D world made up of blocks arranged on a regular grid, while modifying and building structures with blocks. Minecraft provides blocks representing various building materials—grass, dirt, water, sand, snow, *etc*. Each block is assigned a simple texture, and the game is known for its distinctive cartoonish look. While one might discount Minecraft as a simple game with simple mechanics, Minecraft is, in fact, a very popular 3D content

Representative method	3D view consistent?	Across worlds?	W/o paired data?
CycleGAN [71], MUNIT [21]	✗	✓	✓
pix2pix [22], SPADE [49]	✗	✓	✗
wc-vid2vid [36]	✓	✓	✗
NeRF [39], NSVF [31]	✓	✗	✗
GANcraft (ours)	✓	✓	✓

Table 1: Given a Minecraft world, our goal is to train a neural renderer that can convert any camera trajectory in the Minecraft world to a sequence of view-consistent images in the real world, at test time. The training needs to be achieved without paired Minecraft–real data as it does not exist. Among prior work, only unsupervised image-to-image translation methods such as CycleGAN [71] and MUNIT [21] can work in this setting. However, they do not generate 3D view-consistent outputs. Neural radiance field-based methods like NeRF [39] and NSVF [31] are suited for novel view synthesis. They cannot handle the Minecraft–real domain gap. All other prior works require paired training data unavailable in our setting. GANcraft, our proposed method, can generate 3D view-consistent Minecraft-to-real synthesis results without paired Minecraft–real training data.

creation tool. Minecrafters have faithfully recreated large cities and famous landmarks including the Eiffel Tower! The block world representations are intuitive to manipulate and this makes it well-suited as the medium for our world-to-world translation task. We focus on generating natural landscapes, which was also studied in several prior work in image-to-image translation [3, 49].

At first glance, generating a 3D photorealistic world from a semantic block world seems to be a task of translating a sequence of projected 2D segmentation maps of the 3D block world, and is a direct application of image-to-image translation . This approach, however, immediately runs into several serious issues. First, obtaining paired ground truth training data of the 3D block world, segmentation labels, and corresponding real images is extremely costly if not impossible. Second, existing image-to-image translation models [21, 49, 61, 71] do not generate consistent views [36]. Each image is translated independent of the others.

While the recent world-consistent vid2vid work [36] overcomes the issue of view-consistency, it requires paired ground truth 3D training data. Even the most recent neural rendering approaches based on neural radiance fields such as NeRF [39], NSVF [31], and NeRF-W [37], require real images of a scene and associated camera parameters, and are best suited for view interpolation. As there is no paired 3D and ground truth real image data, as summarized in Table 1, none of the existing techniques can be used to solve this new task. This requires us to employ ad hoc adaptations to make our problem setting as similar to these methods' requirements as possible, *e.g*. training them on real segmentations instead of Minecraft segmentations.

In the absence of ground truth data, we propose a framework to train our model using *pseudo*-ground truth photorealistic images for sampled camera views. Our framework uses ideas from image-to-image translation and improves upon work in 3D view synthesis to produce view-consistent photorealistic renderings of input Minecraft worlds as shown in Fig. 1. Although we demonstrate our results using Minecraft, our method works with other 3D block world representations, such as voxels. We chose Minecraft because it is a popular platform available to a wide audience.

Our key contributions include:

- The novel task of producing view-consistent photorealistic renderings of user-created 3D semantic block worlds, or *world-to-world* translation, a 3D extension of image-to-image translation.
- A framework for training neural renderers in the absence of ground truth data. This is enabled by using *pseudo*-ground truth images generated by a pretrained image synthesis model (Section 3.1).
- A new neural rendering network architecture trained with adversarial losses (Section 3.2), that extends recent work in 2D and 3D neural rendering [20, 31, 37, 39, 44] to produce state-of-the-art results which can be conditioned on a style image (Section 4).

2. Related Work

2D image-to-image translation. The GAN framework [16] has enabled multiple methods to successfully map an image from one domain to another with high fidelity, *e.g*., from input segmentation maps to photorealistic images. This task can be performed in the supervised setting [22, 26, 35, 49, 61, 72], where example pairs of corresponding images are available, as well as the unsupervised setting [14, 21, 32, 33, 35, 53, 71], where only two sets of images are available. Methods operating in the supervised setting use stronger losses such as the L_1 or perceptual loss [23], in conjunction with the adversarial loss. As paired data is unavailable in the unsupervised setting, works typically rely on a shared-latent space assumption [32] or cycle-consistency losses [71]. For a comprehensive overview of image-to-image translation methods, please refer to the survey of Liu *et al*. [34].

Our problem setting naturally falls into the unsupervised setting as we do not possess real-world images corresponding to the Minecraft 3D world. To facilitate learning a view-consistent mapping, we employ *pseudo*-ground truths during training, which are predicted by a pretrained supervised image-to-image translation method.

Pseudo-ground truths were first explored in prior work on self-training, or bootstrap learning [38, 66][1]. More recently, this technique has been adopted in several unsupervised

[1] See https://ruder.io/semi-supervised/ for an overview.

domain adaptation works [13, 27, 55, 60, 64, 69, 73]. They use a deep learning model trained on the 'source' domain to obtain predictions on the new 'target' domain, treat these predictions as ground truth labels, or *pseudo labels*, and finetune the deep learning model on such self-labeled data.

In our problem setting, we have segmentation maps obtained from the Minecraft world but do not possess the corresponding real image. We use SPADE [49], a conditional GAN model, trained for generating landscape images from input segmentation maps to generate pseudo ground truth images. This yields the pseudo pair: input Minecraft segmentation mask and the corresponding pseudo ground truth image. The pseudo pairs enable us to use stronger supervision such as L_1, L_2, and perceptual [23] losses in our world-to-world translation framework, resulting in improved output image quality. This idea of using pretrained GAN models for generating training data has also been explored in the very recent works of Pan *et al*. [47] and Zhang *et al*. [70], which use a pretrained StyleGAN [24, 25] as a multi-view data generator to train an inverse graphics model.

3D neural rendering. A number of works have explored combining the strengths of the traditional graphics pipeline, such as 3D-aware projection, with the synthesis capabilities of neural networks to produce view-consistent outputs. By introducing differentiable 3D projection and using trainable layers that operate in the 3D and 2D feature space, several recent methods [4, 18, 42, 43, 56, 62] are able to model the geometry and appearance of 3D scenes from 2D images. Some works have successfully combined neural rendering with adversarial training [18, 42, 43, 44, 54], thereby removing the constraint of training images having to be posed and from the same scene. However, the under-constrained nature of the problem limited the application of these methods to single objects, synthetic data, or small-scale simple scenes. As shown later in Section 4, we find that adversarial training alone is not enough to produce good results in our setting. This is because our input scenes are larger and more complex, the available training data is highly diverse, and there are considerable gaps in the scene composition and camera pose distribution between the block world and the real images.

Most recently, NeRF [39] demonstrated state-of-the-art results in novel view synthesis by encoding the scene in the weights of a neural network that produces the volume density and view-dependent radiance at every spatial location. Follow-up works have tried to improve the output quality [31, 68], make it faster to train and evaluate [30, 31, 41, 51, 59], extend it to deformable objects [15, 28, 48, 50, 63], account for lighting [9, 6, 37, 57] and compositionality [17, 44, 46, 67], as well as add generative capabilities [11, 54, 44].

Most relevant to our work are NSVF [31], NeRF-W [37], and GIRAFFE [44]. NSVF [31] reduces the computational cost of NeRF by representing the scene as a set of voxel-bounded implicit fields organized in a sparse voxel octree, which is obtained by pruning an initially dense cuboid made of voxels. NeRF-W [37] learns image-dependent appearance embeddings allowing it to learn from unstructured photo collections, and produce style-conditioned outputs. These works on novel view synthesis learn the geometry and appearance of scenes given ground truth posed images. In our setting, the problem is inverted — we are given coarse voxel geometry and segmentation labels as input, without any corresponding real images.

Similar to NSVF [31], we assign learnable features to each corner of the voxels to encode geometry and appearance. In contrast, we do not learn the 3D voxel structure of the scene from scratch, but instead implicitly refine the provided coarse input geometry (*e.g*. shape and opacity of trees represented by blocky voxels) during the course of training. Prior work by Riegler *et al*. [52] also used a mesh obtained by multi-view stereo as a coarse input geometry. Similar to NeRF-W [37], we use a style-conditioned network. This allows us to learn consistent geometry while accounting for the view inconsistency of SPADE [49]. Like neural point-based graphics [4] and GIRAFFE [44], we use differentiable projection to obtain features for image pixels, and then use a CNN to convert the 2D feature grid to an image. Like GIRAFFE [44], we use an adversarial loss in training. We, however, learn on large, complex scenes and produce higher-resolution outputs (1024×2048 original image size in Fig. 1, v/s 64×64 or 256×256 pixels in GIRAFFE), in which case adversarial loss alone fails to produce good results.

3. Neural Rendering of Minecraft Worlds

Our goal is to convert a scene represented by semantically-labeled blocks (or voxels), such as the maps from Minecraft, to a photorealistic 3D scene that can be consistently rendered from arbitrary viewpoints (as shown in Fig. 1). In this paper, we focus on landscape scenes that are orders of magnitude larger than single objects or scenes typically used in the training and evaluation of previous neural rendering works. In all of our experiments, we use voxel grids of $512\times512\times256$ blocks (512×512 blocks horizontally, 256 blocks tall vertically). Given that each Minecraft block is considered to have a size of 1 cubic meter [1], each scene covers an area equivalent to 262,144 m^2 (65 acres, or the size of 32 soccer fields) in real life. At the same time, our model needs to learn details that are much finer than a single block, such as tree leaves, flowers, and grass, that too without supervision. As the input voxels and their labels already define the coarse geometry and semantic arrangement of the scene, it is necessary to respect and incorporate this prior information into the model. We first describe how we overcome the lack of paired training data by using pseudo-ground truths. Then, we present our novel sparse voxel-based neural renderer.

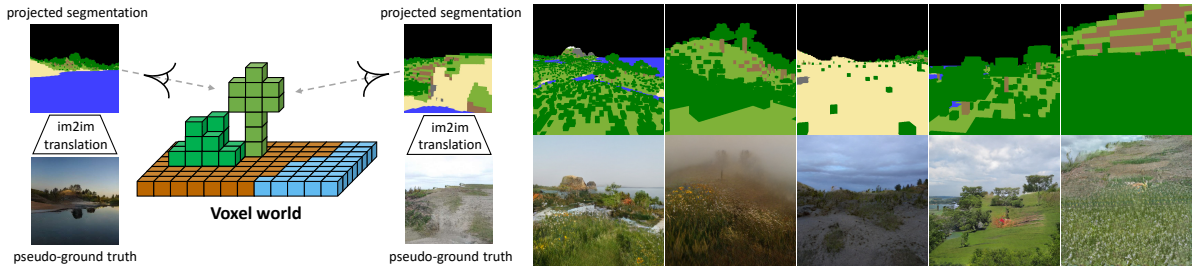

Figure 2: **Pseudo-ground truth generation.** Left: We use a pretrained image-to-image translation model (SPADE [49]) to convert projected segmentation maps to images. Right: Sample input segmentation maps showing different labels (grass, trees, water, sand, sky) and SPADE outputs for different style codes. Note that some generated outputs can look unrealistic due to domain gap of the blocky segmentations and sampled camera poses, with the real image data used to train SPADE. Our method is designed to be robust to noise, varying styles, and inconsistencies present in these generated pseudo-ground truth images.

3.1. Generating pseudo-ground truth training data

The most straightforward way of training a neural rendering model is to utilize ground truth images with known camera poses. A simple L_2 reconstruction loss is sufficient to produce good results in this case [31, 37, 39, 45, 65]. However, in our setting, ground truth real images are simply unavailable for user-generated block worlds from Minecraft.

An alternative route is to train our model in an unpaired, unsupervised fashion like CycleGAN [71], or MUNIT [21]. This would use an adversarial loss and regularization terms to translate Minecraft segmentations to real images. However, as shown in the ablation studies in Section 4, this setting does not produce good results, for both prior methods, and neural renderers. This can be attributed to the large domain gap between blocky Minecraft and the real world, as well as the label distribution shift between worlds.

To bridge the domain gap between the voxel world and our world, we supplement the training data with *pseudo*-ground truth that is generated on-the-fly. For each training iteration, we randomly sample camera poses from the upper hemisphere and randomly choose a focal length. We then project the semantic labels of the voxels to the camera view to obtain a 2D semantic segmentation mask. The segmentation mask, as well as a randomly sampled style code, is fed to a pretrained image-to-image translation network, SPADE [49] in our case, to obtain a photorealistic pseudo-ground truth image that has the same semantic layout as the camera view, as shown in the left part of Fig. 2. This enables us to apply reconstruction losses, such as L_2, and the perceptual loss [23], between the pseudo-ground truth and the rendered output from the same camera view, in additional to the adversarial loss. This significantly improves the result.

The generalizability of the SPADE model trained on large-scale datasets, combined with its photorealistic generation capability helps reduce both, the domain gap and the label distribution mismatch. Sample pseudo-pairs are shown in the right part of Fig. 2. While this provides effective supervision, it is not perfect. This can be seen especially in the last two columns in the right part of Fig. 2. The blockiness of Minecraft can produce unrealistic images with sharp geometry. Certain camera poses and style code combinations can also produce images with artifacts. We thus have to be careful to balance reconstruction and adversarial losses to ensure successful training of the neural renderer.

3.2. Sparse voxel-based volumetric neural renderer

Voxel-bounded neural radiance fields. Let K be the number of occupied blocks in a Minecraft world, which can also be represented by a sparse voxel grid with K non-empty voxels given by $\mathcal{V} = \{V_1, ..., V_K\}$. Each voxel is assigned a semantic label $\{l_1, ..., l_K\}$. We learn a neural radiance field per voxel. The Minecraft world is then represented by the union of voxel-bounded neural radiance fields given by

$$F(\mathbf{p}, \mathbf{z}) = \begin{cases} F_i(\mathbf{p}, \mathbf{z}), & \text{if } \mathbf{p} \in V_i, \;\; i \in \{1, \cdots, K\} \\ (\mathbf{0}, 0), & \text{otherwise} \end{cases}, \quad (1)$$

where F is the radiance field of the whole scene and F_i is the radiance field bounded by V_i. Querying a location in the neural radiance field returns a feature vector (or color in prior work [31, 37, 39]) and a density value. At the location where a block does not exist, we have the null feature vector $\mathbf{0}$ and zero density 0. To model diversified appearance of the same scene, *e.g.* day and night, the radiance fields are conditioned on style code $\mathbf{z}$.

The voxel-bounded neural radiance field F_i is given by

$$F_i(\mathbf{p}, \mathbf{z}) = G_\theta(g_i(\mathbf{p}), l_i, \mathbf{z}) = (\mathbf{c}(\mathbf{p}, l(\mathbf{p}), \mathbf{z}), \sigma(\mathbf{p}, l(\mathbf{p}))) \,,$$

where $g_i(\mathbf{p})$ is the location code at $\mathbf{p}$ and $l_i \equiv l(\mathbf{p})$ is a short-hand for the label of the voxel that $\mathbf{p}$ belongs to. The multi-layer perceptron (MLP) G_θ is used to predict the feature $\mathbf{c}$, and volume density σ at the location $\mathbf{p}$. We note that G_θ is shared amongst all voxels. Inspired by NeRF-W [37], $\mathbf{c}$ additionally depends on the style code, while the density σ does not. To obtain the location code, we first assign a learnable feature vector to each of the eight vertices of a voxel V_i. The location code at $\mathbf{p}$, $g_i(\mathbf{p})$, is then derived through

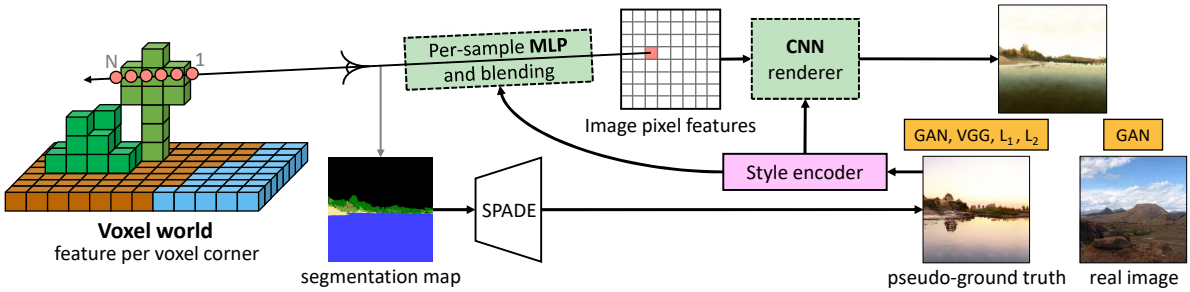

Figure 3: **Overview of GANcraft.** Given an input voxel world with segmentation labels, we first assign features to every voxel corner. For arbitrarily sampled camera viewpoints, we obtain the trilinearly interpolated voxel features at the point of ray-voxel intersections, process with an MLP, and blend the output features to obtain the image pixel features. These features are fed to an image-space CNN renderer. Both the MLP and the CNN are conditioned on the style code of the pseudo-ground truth for the chosen camera view. Our method is trained with an adversarial loss with real images, and a combination of adversarial, pixel-wise, and VGG perceptual losses on the pseudo-ground truths. After training, we can render the world in a photorealistic manner, controlling the style of the output images by conditioning on an input style code or image.

trilinear interpolation. Here, we assume that each voxel has a shape of $1\times1\times1$, and the coordinate axes are aligned to the voxel grid axes. Vertices and their feature vectors are shared for adjacent voxels. This allows for a smooth transition of features when crossing the voxel boundaries, preventing discontinuities in the output. We compute Fourier features from $g_i(\mathbf{p})$, similar to NSVF [31], and also append the voxel class label. Our method can be interpreted as a generalization of NSVF [31] to use a style and semantic label conditioning.

Neural sky dome. The sky is an indispensable part of photorealistic landscape scenes. However, as it is physically located much farther away from the other objects, it is inefficient to represent it with a layer of voxels. In GANcraft, we assume that the sky is located infinitely far away (no parallax). Thus, its appearance is only dependent on the viewing direction. The same assumption is commonly used in computer graphics techniques such as environment mapping [8]. We use an MLP H_ϕ, to map ray direction $\mathbf{v}$ to sky color, or feature, $\mathbf{c}_{\text{sky}} \equiv H_\phi(\mathbf{v}, \mathbf{z})$, conditioned on style code $\mathbf{z}$, This representation can be viewed as covering the whole scene with an infinitely large sky dome.

Volumetric rendering. Here, we describe how a scene represented by the above-mentioned neural radiance fields and sky dome can be converted to 2D feature maps via volumetric rendering. Under a perspective camera model, each pixel in the image corresponds to a camera ray $\mathbf{r}(t) = \mathbf{o} + t\mathbf{v}$, originating from the center of projection $\mathbf{o}$ and advances in direction $\mathbf{v}$. The ray travels through the radiance field while accumulating features and transmittance,

$$C(\mathbf{r},\mathbf{z}) = \int_0^{+\infty} T(t)\sigma\big(\mathbf{r}(t), l(\mathbf{r}(t))\big)\mathbf{c}\big(\mathbf{r}(t), l(\mathbf{r}(t)), \mathbf{z}\big)dt + T(+\infty)\mathbf{c}_{\text{sky}}(\mathbf{v},\mathbf{z}), \quad (2)$$

$$\text{where } T(t) = \exp\left(-\int_0^t \sigma(\mathbf{r}(s))ds\right). \quad (3)$$

$C(\mathbf{r},\mathbf{z})$ denotes the accumulated feature of ray $\mathbf{r}$, and $T(t)$ denotes the accumulated transmittance when the ray travels a distance of t. As the radiance field is bounded by a finite number of voxels, the ray will eventually exit the voxels and hit the sky dome. We thus consider the sky dome as the last data point on the ray, which is totally opaque. This is realized by the last term in Eq. 2. The above integral can be approximated using discrete samples and the quadrature rule, a technique popularized by NeRF [39]. Please refer to NeRF [39] or our supplementary for the full equations.

We use the stratified sampling technique from NSVF [31] to randomly sample valid (voxel bounded) points along the ray. To improve efficiency, we truncate the ray so that it will stop after a certain accumulated distance through the valid region is reached. We regularize the truncated rays to encourage their accumulated opacities to saturate before reaching the maximum distance. We adopt a modified Bresenham method [5] for sampling valid points, which has a very low complexity of $O(N)$, where N is the longest dimension of the voxel grid. Details are in the supplementary.

Hybrid neural rendering architecture. Prior works [31, 37, 39] directly produce images by accumulating colors using the volumetric rendering scheme described above instead of accumulating features. Unlike them, we divide rendering into two parts: 1) We perform volumetric rendering with an MLP to produce a feature vector per pixel instead of an RGB image, and 2) We employ a CNN to convert the per-pixel feature map to a final RGB image of the same size. The overall framework is shown in Fig. 3. We perform activation modulation [20, 49] conditioned on the input style code for both the MLP and CNN. The individual networks are described in detail in the supplementary.

Apart from improving the output image quality as shown in Section 4, this two-stage design also helps reduce the computational and memory footprint of rendering. The MLP modeling the 3D radiance field is evaluated on a per-sample

basis, while the image-space CNN is only evaluated after multiple samples along a ray are merged into a single pixel. The number of samples to the MLP scales linearly with the output height, width, and number of points sampled per ray (24 in our case), while the size of the feature map only depends on output height and width. However, unlike MLPs that operate pre-blending, the image-space CNN is not intrinsically view consistent. We thus use a shallow CNN with a receptive field of only 9×9 pixels to constrain its scope to local manipulations. A similar idea of combining volumetric rendering and image-space rendering has been used in GIRAFFE [44]. Unlike us, they also rely on the CNN to upsample a low-resolution 16×16 feature map.

Losses and regularizers. We train our model with both reconstruction and adversarial losses. The reconstruction loss is applied between the predicted images and the corresponding pseudo-ground truth images. We use a combination of the perceptual [23], L_1, and L_2 losses. For the GAN loss, we treat the predicted images as 'fake', and both the real images, and the pseudo-ground truth images as 'real'. We use a discriminator conditioned on the semantic segmentation maps, based on Liu *et al*. [35] and Schönfeld *et al*. [53]. We use the hinge loss [29] as the GAN training objective. Following previous works on multimodal image synthesis [3, 21, 72], we also include a style encoder which predicts the posterior distribution of the style code given a pseudo-ground truth image. The reconstruction loss, in conjunction with the style encoder, makes it possible to control the appearance of the output image with a style image.

As mentioned earlier, we truncate the ray during volumetric rendering. To avoid artifacts due to the truncation, we apply an opacity regularization term on the truncated ray, $\mathcal{L}_{\text{opacity}} = \sum_{\mathbf{r}\in\mathbf{R}_{\text{trunc}}} T_{\text{out}}(\mathbf{r})$. This discourages leftover transmittance after a ray reaches the truncation distance.

4. Experiments

The previous section described how we obtain pseudo-ground truths in the absence of paired Minecraft–real training data, and the architecture of our neural renderer. Here, we validate our framework by comparing with prior work on multiple diverse large Minecraft worlds.

Datasets. We collected a dataset of $\sim$1M landscape images from the internet with a minimum side of at least 512 pixels. For each image, we obtained 182-class COCO-Stuff [10] segmentation labels by using DeepLabV2 [12, 40]. This formed our training set of paired real segmentation maps and images. We set aside 5000 images as a test set. We generated 5 different Minecraft worlds of $512\times512\times256$ blocks each. We sampled worlds with various compositions of water, sand, forests, and snow, to show that our method works correctly under significant label distribution shifts.

Baselines. We compare against the following, which are representative methods under different data availability regimes.

- MUNIT [21]. This is an image-to-image translation method trainable in the unpaired, or unsupervised setting. Unlike CycleGAN [71] and UNIT [32], MUNIT can learn multimodal translations. We learn to translate Minecraft segmentation maps to real images.
- SPADE [49]. This is an image-to-image translation method that is trained in the paired ground truth, or supervised setting. We train this by translating real segmentation maps to corresponding images and test it on Minecraft segmentations.
- wc-vid2vid [36]. Unlike the above two methods, this can generate a sequence of images that are view-consistent. wc-vid2vid projects the pixels from previous frames to the next frame to generate a guidance map. This serves as a form of memory of the previously generated frames. This method also requires paired ground truth data, as well as the 3D point clouds for each output frame. We train this to translate real segmentation maps to real images, while using the block world voxel surfaces as the 3D geometry.
- NSVF-W [31, 37]. We combine the strengths of two recent works on neural rendering, NSVF [31], and NeRF-W [37], to create a strong baseline. NSVF represents the world as voxel-bounded radiance fields, and can be modified to accept an input voxel world, just like our method. NeRF-W is able to learn from unstructured image collections with variations in color, lighting, and occlusions, making it well-suited for learning from our pseudo-ground truths. Combining the style-conditioned MLP generator from NeRF-W with the voxel-based input representation of NSVF, we obtain NSVF-W. This resembles the neural renderer used by us, with the omission of the image-space CNN. As these methods also require paired ground truth, we train NSVF-W using pseudo-ground truths generated by the pretrained SPADE model.

MUNIT, SPADE, and wc-vid2vid use perceptual and adversarial losses during training, while NSVF, NeRF-W, and thus NSVF-W use the L_2 loss. Details are in the supplementary.

Implementation details. We train our model at an output resolution of 256×256 pixels. Each model is trained on 8 NVIDIA V100 GPUs with 32GB of memory each. This enables us to use a batch size of 8 with 24 points sampled per camera ray. Each model is trained for 250k iterations, which takes approximately 4 days. All baselines are also trained for an equivalent amount of time. Additional details are available in the supplementary.

Evaluation metrics. We use both quantitative and qualitative metrics to measure the quality of our outputs.

- Fréchet Inception Distance [19] (FID) and Kernel Inception Distance [7] (KID). We use FID and KID to measure the distance between the distributions of the generated and real images, using Inception-v3 [58]. We generate

Method	FID ↓	KID ↓
SPADE [49]	58.90	0.027
MUNIT [21]	78.42	0.047
NSVF-W [31, 37]	84.53	0.052
GANcraft (ours)	**61.33**	**0.033**

Table 2: **Automated image quality metrics.** We compare baselines against GANcraft on all 5 block worlds. SPADE sets the lower bound on FID and KID as it is a strong photo-realistic image generator, although it is not view-consistent. Our view-consistent method achieves values close to SPADE, beating MUNIT and NSVF-W.

1000 images for each of the 5 worlds from arbitrarily sampled camera view points using different style codes, for a total of 5000 images. We then generate outputs from each method for the same pair of view points and style code for a fair comparison. We use a held-out set of 5000 real landscape images to compute the metrics. For both metrics, a lower value indicates better image quality.

- Human preference score. Using Amazon Mechanical Turk (AMT), we perform a subjective visual test to gauge the relative quality of generated videos with top-qualified turkers. We ask turkers to choose 1) the more temporally consistent video, and 2) the video with overall better realism. For each of the two questions, a turker is shown two videos synthesized by two different methods and asked to choose the superior one according to the criteria. We generate 64 videos per world, total of 320 per method, and each comparison is evaluated by 3 workers.

Main results. Fig. 4 shows output videos generated by different methods. Each row is a unique world, generated using the same style-conditioning image for all methods. We can observe that our outputs are more realistic and view-consistent when compared to baselines. MUNIT [21] and SPADE [49] demonstrate a lot of flickering as they generate one image at a time, without any memory of past outputs. Further, MUNIT also fails to learn the correct mapping of segmentation labels to textures as it does not use paired supervision. While wc-vid2vid [36] is more view-consistent, it fails for large motions as it incrementally inpaints newly explored parts of the world. NSVF-W [31, 37] and GANcraft are both inherently view-consistent due to their use of volumetric rendering. However, due to the lack of a CNN renderer and the use of the L_2 loss, NSVF-W produces dull and unrealistic outputs with artifacts. The use of an adversarial loss is key to ensuring vivid and realistic results, and this is further reinforced in the ablations presented below. Our method is also capable of generating higher resolution outputs as shown in Fig. 1, by sampling more rays.

We sample novel camera views from each world and compute the FID and KID against a set of held-out real images. As seen in Table 2, our method achieves FID and KID close to that of SPADE, which is a very strong image-to-image translation method, while beating other baselines. Note that wc-vid2vid uses SPADE to generate the output for first camera view in a sequence and is thus ignored in this comparison. Further, as summarized in Table 3, users consistently preferred our method and chose its predictions as the more view-consistent and realistic videos. More high-resolution results and comparisons as well as some failure cases are available in the supplementary.

Comparison	Human preference	
	Consistency ↑	Realism ↑
MUNIT [21] / GANcraft	30.1/**69.9**	37.5/**62.5**
SPADE [49] / GANcraft	29.7/**70.3**	37.2/**62.8**
wc-vid2vid [36] / GANcraft	47.0/**53.0**	16.2/**83.8**
NSVF-W [31, 37] / GANcraft	46.6/**53.4**	31.4/**68.6**

Table 3: **Human preference scores.** We compare videos generated by different methods on all 5 block worlds. Users chose GANcraft as more temporally consistent and realistic.

Ablations. We train ablated versions of our full model on one Minecraft world due to computational constraints. We show example outputs from them in Fig. 5. Using no pseudo-ground truth at all and training with just the GAN loss produces unrealistic outputs, similar to MUNIT [21]. Directly producing images from volumetric rendering, without using a CNN, results in a lack of fine detail. Compared to the full model, skipping the GAN loss on real images produces duller images, and skipping the GAN loss altogether produces duller, blurrier images resembling NSVF-W outputs. Qualitative analysis is available in the supplementary.

5. Discussion

We introduced the novel task of *world-to-world* translation and proposed GANcraft, a method to convert block worlds to realistic-looking worlds. We showed that *pseudo*-ground truths generated by a 2D image-to-image translation network provide effective means of supervision in the absence of real paired data. Our hybrid neural renderer trained with both real landscape images and pseudo-ground truths, and adversarial losses, outperformed strong baselines.

There still remain a few exciting avenues for improvements, including learning a smoother geometry in spite of coarse input geometry, and using non-voxel inputs such as meshes and point clouds. While our method is currently trained on a per-world basis, we hope future work can enable feed-forward generation on novel worlds.

Acknowledgements. We thank Rev Lebaredian for challenging us to work on this interesting problem. We thank Jan Kautz, Sanja Fidler, Ting-Chun Wang, Xun Huang, Xihui Liu, Guandao Yang, and Eric Haines for their feedback during the course of developing the method.

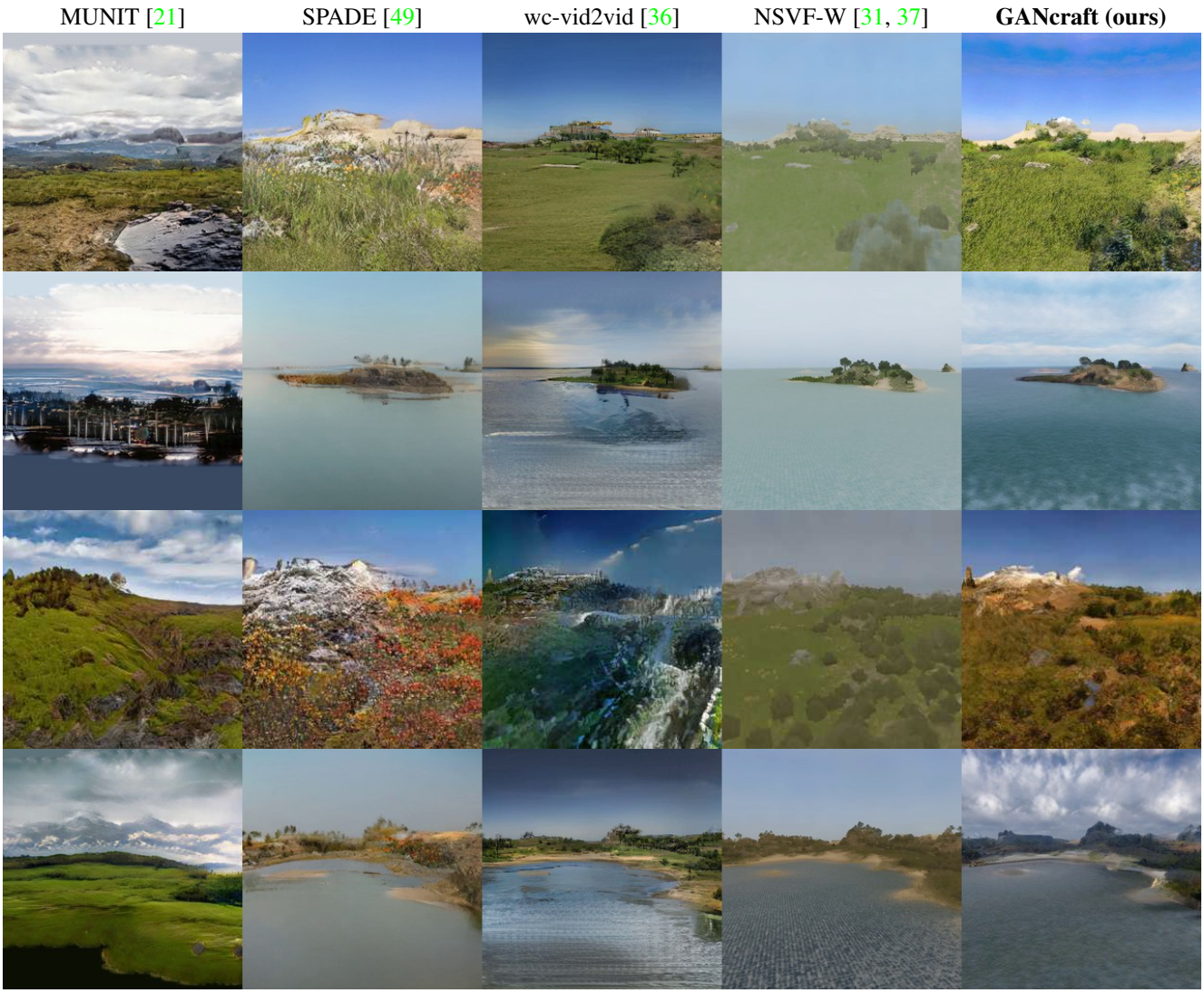

Figure 4: **Output video comparison.** Each row is a unique world, and each column is a different method. For a given world, all methods use the same style-conditioning image. GANcraft produces more view-consistent and more realistic outputs compared to all baselines. *Click any row to play video in web browser.*

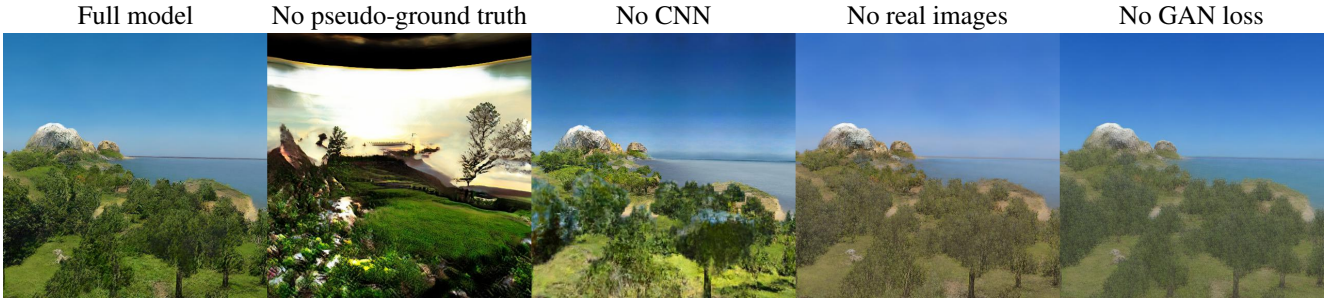

Figure 5: **Ablated model outputs.** Using only the GAN loss with no pseudo-ground truths produces unrealistic images. Not using a CNN produces outputs that lack detail and contain artifacts. Excluding the GAN loss on real images results in dull colors, and no GAN loss at all produces dull and blurry outputs, when compared to the full model.

References

[1] Minecraft units of measure. https://minecraft.gamepedia.com/Tutorials/Units_of_measure#Distance. 3

[2] Minecraft user statistics. https://bit.ly/3sZkaMF. 1

[3] NVIDIA GauGAN. https://blogs.nvidia.com/blog/2019/07/30/gaugan-ai-painting/. 1, 2, 6

[4] Kara-Ali Aliev, Artem Sevastopolsky, Maria Kolos, Dmitry Ulyanov, and Victor Lempitsky. Neural point-based graphics. In *ECCV*, 2020. 3

[5] John Amanatides and Andrew Woo. A fast voxel traversal algorithm for ray tracing. In *Eurographics*, 1987. 5

[6] Sai Bi, Zexiang Xu, Pratul Srinivasan, Ben Mildenhall, Kalyan Sunkavalli, Miloš Hašan, Yannick Hold-Geoffroy, David Kriegman, and Ravi Ramamoorthi. Neural reflectance fields for appearance acquisition. *arXiv preprint arXiv:2008.03824*, 2020. 3

[7] Mikołaj Bińkowski, Dougal J. Sutherland, Michael Arbel, and Arthur Gretton. Demystifying MMD GANs. In *ICLR*, 2018. 6

[8] James F Blinn and Martin E Newell. Texture and reflection in computer generated images. *Comm. of the ACM*, 1976. 5

[9] Mark Boss, Raphael Braun, Varun Jampani, Jonathan T Barron, Ce Liu, and Hendrik Lensch. NeRD: Neural reflectance decomposition from image collections. *arXiv preprint arXiv:2012.03918*, 2020. 3

[10] Holger Caesar, Jasper Uijlings, and Vittorio Ferrari. COCO-Stuff: Thing and stuff classes in context. In *CVPR*, 2018. 6

[11] Eric R Chan, Marco Monteiro, Petr Kellnhofer, Jiajun Wu, and Gordon Wetzstein. pi-GAN: Periodic implicit generative adversarial networks for 3d-aware image synthesis. In *CVPR*, 2021. 3

[12] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE TPAMI*, 2017. 6

[13] Jaehoon Choi, Minki Jeong, Taekyung Kim, and Changick Kim. Pseudo-labeling curriculum for unsupervised domain adaptation. In *BMVC*, 2019. 3

[14] Yunjey Choi, Minje Choi, Munyoung Kim, Jung-Woo Ha, Sunghun Kim, and Jaegul Choo. StarGAN: Unified generative adversarial networks for multi-domain image-to-image translation. In *CVPR*, 2018. 2

[15] Yilun Du, Yinan Zhang, Hong-Xing Yu, Joshua B. Tenenbaum, and Jiajun Wu. Neural radiance flow for 4d view synthesis and video processing. In *ICCV*, 2021. 3

[16] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. In *NeurIPS*, 2014. 2

[17] Michelle Guo, Alireza Fathi, Jiajun Wu, and Thomas Funkhouser. Object-centric neural scene rendering. *arXiv preprint arXiv:2012.08503*, 2020. 3

[18] Philipp Henzler, Niloy J Mitra, and Tobias Ritschel. Escaping Plato's cave: 3d shape from adversarial rendering. In *CVPR*, 2019. 3

[19] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs trained by a two time-scale update rule converge to a local Nash equilibrium. In *NeurIPS*, 2017. 6

[20] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *ICCV*, 2017. 2, 5

[21] Xun Huang, Ming-Yu Liu, Serge Belongie, and Jan Kautz. Multimodal unsupervised image-to-image translation. *ECCV*, 2018. 2, 4, 6, 7, 8

[22] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *CVPR*, 2017. 1, 2

[23] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *ECCV*, 2016. 2, 3, 4, 6

[24] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *CVPR*, 2019. 3

[25] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of StyleGAN. In *CVPR*, 2020. 3

[26] Cheng-Han Lee, Ziwei Liu, Lingyun Wu, and Ping Luo. MaskGAN: Towards diverse and interactive facial image manipulation. In *CVPR*, 2020. 2

[27] Dong-Hyun Lee. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on challenges in representation learning, ICML*, 2013. 3

[28] Zhengqi Li, Simon Niklaus, Noah Snavely, and Oliver Wang. Neural scene flow fields for space-time view synthesis of dynamic scenes. In *CVPR*, 2021. 3

[29] Jae Hyun Lim and Jong Chul Ye. Geometric GAN. *arXiv preprint arXiv:1705.02894*, 2017. 6

[30] David B Lindell, Julien NP Martel, and Gordon Wetzstein. AutoInt: Automatic integration for fast neural volume rendering. In *CVPR*, 2021. 3

[31] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields. In *NeurIPS*, 2020. 2, 3, 4, 5, 6, 7, 8

[32] Ming-Yu Liu, Thomas Breuel, and Jan Kautz. Unsupervised image-to-image translation networks. In *NeurIPS*, 2017. 2, 6

[33] Ming-Yu Liu, Xun Huang, Arun Mallya, Tero Karras, Timo Aila, Jaakko Lehtinen, and Jan Kautz. Few-shot unsupervised image-to-image translation. In *ICCV*, 2019. 2

[34] Ming-Yu Liu, Xun Huang, Jiahui Yu, Ting-Chun Wang, and Arun Mallya. Generative adversarial networks for image and video synthesis: Algorithms and applications. *Proc. of the IEEE*, 2021. 2

[35] Xihui Liu, Guojun Yin, Jing Shao, Xiaogang Wang, et al. Learning to predict layout-to-image conditional convolutions for semantic image synthesis. In *NeurIPS*, 2019. 2, 6

[36] Arun Mallya, Ting-Chun Wang, Karan Sapra, and Ming-Yu Liu. World-consistent video-to-video synthesis. In *ECCV*, 2020. 2, 6, 7, 8

[37] Ricardo Martin-Brualla, Noha Radwan, Mehdi SM Sajjadi, Jonathan T Barron, Alexey Dosovitskiy, and Daniel Duck-

worth. NeRF in the Wild: Neural radiance fields for unconstrained photo collections. In *CVPR*, 2021. 2, 3, 4, 5, 6, 7, 8

[38] David McClosky, Eugene Charniak, and Mark Johnson. Effective self-training for parsing. In *NAACL*, 2006. 2

[39] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020. 2, 3, 4, 5

[40] Kazuto Nakashima. DeepLab with PyTorch. https://github.com/kazuto1011/deeplab-pytorch. 6

[41] Thomas Neff, Pascal Stadlbauer, Mathias Parger, Andreas Kurz, Chakravarty R. Alla Chaitanya, Anton Kaplanyan, and Markus Steinberger. DONeRF: Towards real-time rendering of neural radiance fields using depth oracle networks. *Comput. Graph. Forum*, 2021. 3

[42] Thu Nguyen-Phuoc, Chuan Li, Lucas Theis, Christian Richardt, and Yong-Liang Yang. HoloGAN: Unsupervised learning of 3d representations from natural images. In *CVPR*, 2019. 3

[43] Thu Nguyen-Phuoc, Christian Richardt, Long Mai, Yong-Liang Yang, and Niloy Mitra. BlockGAN: Learning 3d object-aware scene representations from unlabelled images. In *NeurIPS*, 2020. 3

[44] Michael Niemeyer and Andreas Geiger. GIRAFFE: Representing scenes as compositional generative neural feature fields. In *CVPR*, 2021. 2, 3, 6

[45] Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision. In *CVPR*, 2020. 4

[46] Julian Ost, Fahim Mannan, Nils Thuerey, Julian Knodt, and Felix Heide. Neural scene graphs for dynamic scenes. In *CVPR*, 2021. 3

[47] Xingang Pan, Bo Dai, Ziwei Liu, Chen Change Loy, and Ping Luo. Do 2d GANs know 3d shape? Unsupervised 3d shape reconstruction from 2d image GANs. In *ICLR*, 2021. 3

[48] Keunhong Park, Utkarsh Sinha, Jonathan T Barron, Sofien Bouaziz, Dan B Goldman, Steven M Seitz, and Ricardo-Martin Brualla. Deformable neural radiance fields. In *ICCV*, 2021. 3

[49] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. In *CVPR*, 2019. 1, 2, 3, 4, 5, 6, 7, 8

[50] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-NeRF: Neural radiance fields for dynamic scenes. In *CVPR*, 2021. 3

[51] Daniel Rebain, Wei Jiang, Soroosh Yazdani, Ke Li, Kwang Moo Yi, and Andrea Tagliasacchi. DeRF: Decomposed radiance fields. In *CVPR*, 2021. 3

[52] Gernot Riegler and Vladlen Koltun. Free view synthesis. In *ECCV*, 2020. 3

[53] Edgar Schönfeld, Vadim Sushko, Dan Zhang, Juergen Gall, Bernt Schiele, and Anna Khoreva. You only need adversarial supervision for semantic image synthesis. In *ICLR*, 2021. 2, 6

[54] Katja Schwarz, Yiyi Liao, Michael Niemeyer, and Andreas Geiger. GRAF: Generative radiance fields for 3d-aware image synthesis. In *NeurIPS*, 2020. 3

[55] Rui Shu, Hung H Bui, Hirokazu Narui, and Stefano Ermon. A DIRT-T approach to unsupervised domain adaptation. In *ICLR*, 2018. 3

[56] Vincent Sitzmann, Justus Thies, Felix Heide, Matthias Nießner, Gordon Wetzstein, and Michael Zollhofer. DeepVoxels: Learning persistent 3d feature embeddings. In *CVPR*, 2019. 3

[57] Pratul P Srinivasan, Boyang Deng, Xiuming Zhang, Matthew Tancik, Ben Mildenhall, and Jonathan T Barron. NeRV: Neural reflectance and visibility fields for relighting and view synthesis. In *CVPR*, 2021. 3

[58] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *CVPR*, 2016. 6

[59] Matthew Tancik, Ben Mildenhall, Terrance Wang, Divi Schmidt, Pratul P Srinivasan, Jonathan T Barron, and Ren Ng. Learned initializations for optimizing coordinate-based neural representations. In *CVPR*, 2020. 3

[60] Kevin Tang, Vignesh Ramanathan, Li Fei-Fei, and Daphne Koller. Shifting weights: Adapting object detectors from image to video. In *NeurIPS*, 2012. 3

[61] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional GANs. In *CVPR*, 2018. 2

[62] Olivia Wiles, Georgia Gkioxari, Richard Szeliski, and Justin Johnson. SynSin: End-to-end view synthesis from a single image. In *CVPR*, 2020. 3

[63] Wenqi Xian, Jia-Bin Huang, Johannes Kopf, and Changil Kim. Space-time neural irradiance fields for free-viewpoint video. In *CVPR*, 2020. 3

[64] Shaoan Xie, Zibin Zheng, Liang Chen, and Chuan Chen. Learning semantic representations for unsupervised domain adaptation. In *ICML*, 2018. 3

[65] Lior Yariv, Yoni Kasten, Dror Moran, Meirav Galun, Matan Atzmon, Basri Ronen, and Yaron Lipman. Multiview neural surface reconstruction by disentangling geometry and appearance. In *NeurIPS*, 2020. 4

[66] David Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In *ACL*, 1995. 2

[67] Wentao Yuan, Zhaoyang Lv, Tanner Schmidt, and Steven Lovegrove. STaR: Self-supervised tracking and reconstruction of rigid objects in motion with neural rendering. In *CVPR*, 2021. 3

[68] Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. NeRF++: Analyzing and improving neural radiance fields. *arXiv preprint arXiv:2010.07492*, 2020. 3

[69] Weichen Zhang, Wanli Ouyang, Wen Li, and Dong Xu. Collaborative and adversarial network for unsupervised domain adaptation. In *CVPR*, 2018. 3

[70] Yuxuan Zhang, Wenzheng Chen, Huan Ling, Jun Gao, Yinan Zhang, Antonio Torralba, and Sanja Fidler. Image GANs meet differentiable rendering for inverse graphics and interpretable 3d neural rendering. In *ICLR*, 2021. 3

[71] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *ICCV*, 2017. 2, 4, 6

[72] Jun-Yan Zhu, Richard Zhang, Deepak Pathak, Trevor Darrell, Alexei A Efros, Oliver Wang, and Eli Shechtman. Toward multimodal image-to-image translation. In *NeurIPS*, 2017. 2, 6

[73] Yang Zou, Zhiding Yu, BVK Vijaya Kumar, and Jinsong Wang. Unsupervised domain adaptation for semantic segmentation via class-balanced self-training. In *ECCV*, 2018. 3