

PixelSynth: Generating a 3D-Consistent Experience from a Single Image

Chris Rockwell

David F. Fouhey
University of Michigan

Justin Johnson

Abstract

Recent advancements in differentiable rendering and 3D reasoning have driven exciting results in novel view synthesis from a single image. Despite realistic results, methods are limited to relatively small view change. In order to synthesize immersive scenes, models must also be able to extrapolate. We present an approach that fuses 3D reasoning with autoregressive modeling to outpaint large view changes in a 3D-consistent manner, enabling scene synthesis. We demonstrate considerable improvement in single-image large-angle view synthesis results compared to a variety of methods and possible variants across simulated and real datasets. In addition, we show increased 3D consistency compared to alternative accumulation methods.

1. Introduction

Imagine that you walk into the office shown in Figure 1. What will you see if you turn right? Is there a door onto a patio? What if you step backward then look left? While the image itself does not contain this information, you can imagine a rich world behind the image due to your experience of other rooms. This task of *single-image scene synthesis* promises to bring arbitrary photos to life, but requires solving several key challenges. First, handling large view changes involves *extrapolation* far beyond the input pixels. Second, generating multiple outputs from the same input requires *consistency*: turning left by 10° or 20° should reveal progressively more of a single underlying world. Finally, modeling view changes requires *3D-awareness* to properly capture perspective changes.

Prior methods for view synthesis fall short of these goals. There has been great progress at *interpolating* between many input views of a single scene [29, 28, 43, 44, 48, 50]; while these 3D-aware methods generate consistent outputs, they do not attempt to extrapolate beyond their input views. Prior approaches to single-image view synthesis [15, 47, 53] can extrapolate to small rotations and translations, but fail to model viewpoint changes at this scale. For example, we show that naïvely retraining SynSin [53]

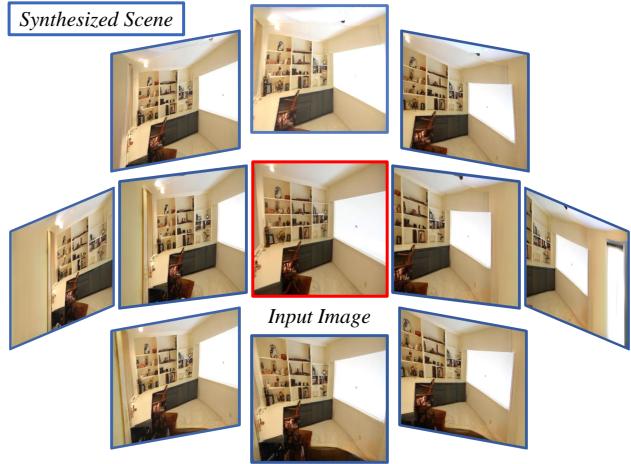


Figure 1: Single-Image Scene Synthesis. Our framework fuses the complementary strengths of 3D reasoning and autoregressive modeling to create an immersive scene from a single image.

for larger angles leads to collapse.

In parallel, autoregressive models have shown impressive results for image generation and completion [4, 30, 35, 40, 49]. These methods are very successful at extrapolating far beyond the boundaries of an input image; however they make no attempt to explicitly model a consistent 3D world behind their generated images.

In this paper we present an approach for single-image scene synthesis that addresses these challenges by fusing the complementary strengths of 3D reasoning and autoregressive modeling. We achieve *extrapolation* using an autoregressive model to complete images when faced with large view changes. Generating all output views independently would give inconsistent outputs. Instead, we identify a *support set* at the extremes of views to be generated (shown at the boundaries of Figure 1). Generated images for the support set are then lifted to 3D and added to a consistent scene representation. Intermediate views can then be re-rendered from the scene representation instead of generated from scratch, ensuring *consistency* among all outputs.

Producing a system that can both do extreme view syn-

thesis and lift the results to 3D without requiring auxiliary data poses a challenge. Our approach, described in Section 3, builds upon insights from both the view synthesis and autoregressive modeling communities. Each image and new viewpoint yields a large and custom region to be filled in, which we approach by adapting VQVAE2 [35] and expanding Locally Masked Convolutions [18] to learn image-specific orderings for outpainting. Once filled, we obtain 3D using techniques from SynSin [53]. This system can outpaint large and diverse regions, accumulate outpainting in 3D, and can be trained without any supervision beyond images and 6DOF relative pose.

We evaluate our approach as well as a variety of alternative methods and competing approaches on standard datasets, Matterport 3D+Habitat [3, 41] and RealEstate10K [63], using substantially larger angle changes ($6 \times$ larger than [53]). Throughout the experiments in Section 4, we evaluate with the standard metrics of human judgments, PSNR, Perceptual Similarity, and FID. Our experimental results suggest that: (1) Our proposed approach produces meaningfully better results compared to training existing methods on our larger viewpoints. In particular, users select our approach 73% of the time vs. the best variant of multiple SynSin ablations. (2) Our approach of re-rendering support sets outperforms alternate iterative approaches, being more consistent an average of 72% of images.

2. Related Work

Both novel view synthesis and image completion have recently seen rapid progress. While novel view synthesis work has approached large view change, it typically uses multiple input images. Given only a single input image, completion becomes highly relevant for outpainting.

Novel view synthesis. If multiple views are available as input, 3D information can be inferred to synthesize new viewpoints. Classical methods often use multi-view geometry [6, 9, 11, 22, 42, 65]. Deep networks use a learned approach, and have shown impressive results with fewer input views and less additional information. They represent 3D in a variety of ways including depth images [1, 28, 37, 56], multi-plane images [47, 63], point clouds [53], voxels [25, 44], meshes [15, 20, 26], multi-layer meshes [13, 43], and radiance fields [29, 50, 60]. We use a point cloud representation.

Given only a single input image, CNNs have also achieved success [8, 23, 48, 57], owing largely to progress in generative modeling [2, 5, 17, 19, 21, 32, 51, 59]. Nevertheless, single-image work has been limited to small angle change [45, 47, 64]. Methods such as SynSin [53] treat outpainting the same as inpainting, which struggles beyond a margin. Our goal is to synthesize a scene from an image, which requires large outpainting. We thus outpaint explicitly using a completion-based approach.

Concurrent work approaches similar, though distinct, problems. Liu *et al.* [23] move forward on camera trajectories through nature scenes. In contrast, our focus is on indoor scenes and we handle outpainting. We show in this setting, a completion-based approach produces better results than a similar approach to [23]. Hu and Pathak [15] use a mesh representation which undertakes twice the rotation of SynSin, but still treats outpainting as interpolation. In contrast, our completion-based approach outpaints explicitly, which we show beats inpainting-based methods in large angles. Rombach *et al.* [38] approach large angle change on images, but do not learn a 3D representation. On scenes, we show our approach of accumulating 3D information across views is critical for consistency.

Image Completion and Outpainting. Recent work in inpainting takes an adversarial approach [16, 24, 54, 58], and has been used in novel view synthesis refinement [23, 44, 47, 53]. However, inpainting is not suitable for synthesizing large angle change, which yields large missing regions. Methods targeting outpainting [46, 52, 55] improve extrapolation, but are not flexible to arbitrary missing regions that may occur in view synthesis.

Our work adopts techniques from the literature on deep autoregressive models. These works use masking with RNNs [30], CNNs [27, 30, 36, 40, 49], and Transformers [4] to predict individual pixels sequentially. While sequential generation is slower than feed-forward methods, it enables flexible ordering and state-of-the-art performance [4, 7, 27, 31, 35]. Yet, autoregressive methods by themselves do not enable 3D consistent outpainting. Thus, we build upon this literature in conjunction with 3D view synthesis to produce a set of *3D consistent* views. Making this 3D fusion work requires building upon several recent developments: we adapt the masked convolution approach of Jain *et al.* [18] to handle custom, per-image, regions to outpaint; also, like VQVAE2 [35] and Dall-E [33], we find that selecting from a set of completions aids realism.

3. Approach

Our goal is to input a single image and synthesize a consistent set of images showing the surrounding scene. This requires generating high-quality images even under large transformations, and ensuring 3D consistency in the results. We propose an approach to this task which uses deep autoregressive modeling to facilitate high-quality extrapolations in conjunction with 3D modeling to ensure consistency.

The two critical insights of the method are the order in which image data is produced and the 3D nature of the approach. As illustrated in Figure 2, our system generates data on extremal *support views* first and operates on point clouds. We outpaint these support views with an autoregressive outpainting module that handles most of the generation. As we

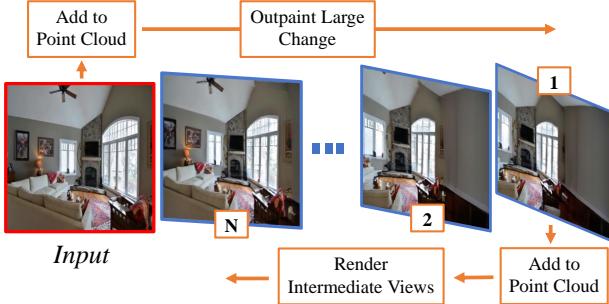


Figure 2: Consistent Scene Synthesis. The model first generates extremal support views, from which intermediate views can be generated. This allows the model to outpaint once and re-render many times, which improves 3D consistency.

reproject intermediate views, we touch up the results with a refinement module. Throughout, we translate from images to point clouds with a self-supervised depth module and back with a differentiable renderer.

3.1. 3D and Synthesis Modules

We begin by introducing each of the modules used by our system, which are pictured on the right of Figure 3. We first describe the two modules that map to and from point clouds, followed by the models that generate and refine pixels. With the exception of the projection module, all are learnable functions that are represented by deep neural networks. Full descriptions of each are in the supplement.

Depth Module D : Given an image I , we can convert it into a colored point cloud $C = D(I)$ using a learned depth prediction system. Specifically, the per-pixel depth is inferred using a U-Net [39] and the pixel is mapped to 3D using known intrinsics. In our work, we learn D via end-to-end training on reprojection losses.

Projector π : Given a colored point cloud C , and 6DOF pose p , we can project it to an image $I = \pi(C, p)$ using Pytorch3D [34]’s differentiable renderer. This renderer uses a soft z-buffer which blends nearby points.

Outpainter O : When the viewpoint changes dramatically, large missing regions come into the field of view and must be outpainted. The specific regions depend on both the viewpoint shift and the image content. We perform per-image outpainting on the latent space of a VQ-VAE [31, 35]. Our particular model autoencodes $256 \times 256 \times 3$ inputs through a discrete $32 \times 32 \times 1$ -dimensional embedding space $\{Z\}$, $Z_{i,j,1} \in \mathbb{Z}_1^{512}$. Using discrete values encourages the model to choose more divergent completions.

We outpaint in this 32×32 latent space using an autoregressive model [31, 33, 10]. In our particular case we predict pixel embeddings with a PixelCNN++ [40] architecture, using a 512-way classification head to predict a distri-

bution over embeddings. We use Locally Masked Convolution [18] blocks to implement image-specific custom-pixel orderings. We show examples of the orders used in Figure 4, which outpaint pixels close to the visible region, followed by those farther away.

Refinement Module R : Outpainting returns images that are sensible, but often lack details or have inconsistencies due to imperfect depth. We therefore use an adversarially-trained refinement module to correct local errors. This module blends the reprojection of the original and outpainted pixels and predicts a residual to its input. Our generator architecture is similar to [53] and uses 8 ResNet [12] blocks containing Batch Normalization injected with noise like [2]. We adopt the discriminator from [51].

3.2. Inference

At inference time, we compose the modules to generate the full set of images in two phases: support view outpainting, followed by intermediate view rendering. The process overview is shown on the left of Figure 3, and can be reused to synthesize multiple viewing directions.

Support View Outpainting and Refinement: Given a single input image and support view p_1 , our goal is to create an updated point cloud that includes a set of pixels that might be seen in view p_1 . We achieve this by outpainting in the support view: first, we estimate what can be seen in the support view by projecting the point cloud inferred from the input, or $I' = \pi(D(I), p_1)$. This projection usually has a large and image-specific gap (Figure 4). Our second step composes the outpainting and refinement module, or $I_1 = R(O(I'))$. Finally, the resulting large-view synthesis itself is lifted to a point cloud by applying D , or $C_1 = D(I_1)$.

Intermediate View Rendering and Refinement: Once the input and outpainted support views have converted to point clouds, we can quickly render any intermediate view p_i by applying the projection and refinement modules. Specifically, if $C = D(I)$ is the input point cloud and C_1 is the support view point cloud, we simply apply the refinement to their combined projection, or $R(\pi([C, C_1], p_i))$.

3.3. Training and Implementation Details

End-to-end training of the model is difficult because the Outpainter requires ground truth input and output, which breaks gradient flow. We therefore perform training of the Depth and Refinement Modules, the Outpainter VQ-VAE, and Outpainter autoregressive model separately. In all cases, batch size is chosen to maximize GPU space, and training stops when validation loss plateaus.

We first train the latent VQ-VAE space of the Outpainter O , which is then frozen and used by the other modules during training. Training takes $\sim 30k$ iterations with batch size

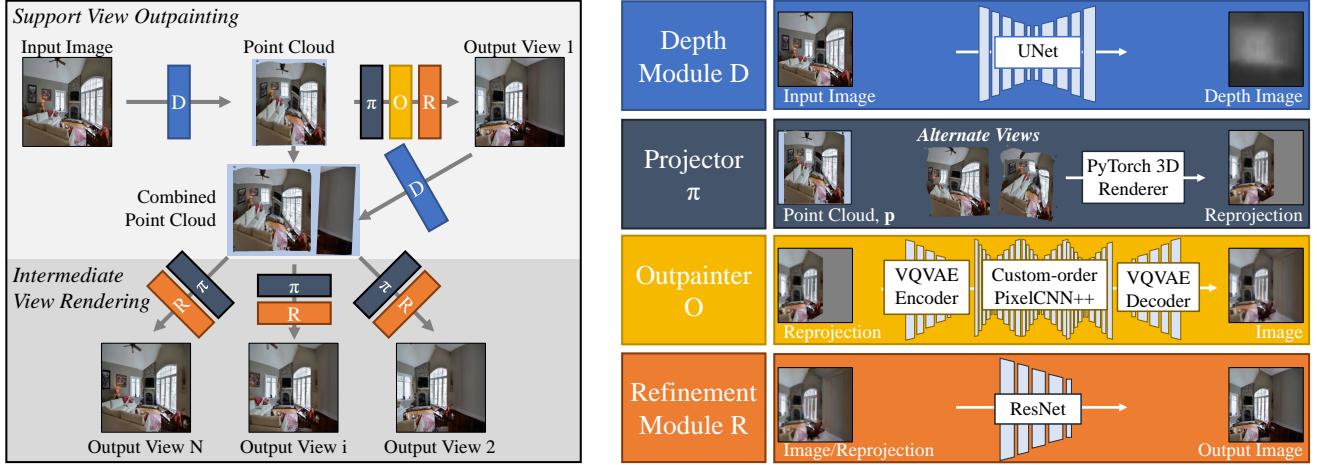


Figure 3: Approach Overview. At inference, the model first outpaints an extremal support view, then renders intermediate views (left). Both steps rely on the Depth Module to lift images to point clouds, the Projector to render in a novel view, and the Refinement Module to smooth outputs (right). The Outpainter fills in missing information in the target view during outpainting.

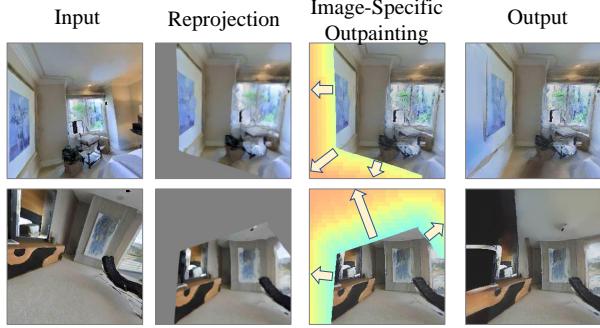


Figure 4: Autoregressive Outpainting. We outpaint using image-specific orderings, which begin with pixels adjacent to the visible region and move outward. Our model outpaints a vector-quantized embedding space.

of 120 and uses the losses from [35]: an L_2 reprojection loss and an embedding commitment loss.

Next, we train the Depth and Refinement Modules jointly. Ground truth is used in place of missing pixels to be outpainted to avoid having to sample from the Outpainter during training. The composition of Depth and Refinement is trained with an L_1 pixel loss, a content loss [61], and a multi-scale discriminator loss [51]. The discriminator in the Refinement Module is trained at multiple scales with a feature-matching loss. In the process, the Depth Module is implicitly learned. We train for 125k iterations (200k on Matterport) with batch size 12.

Finally, the autoregressive model in O is trained. It is trained upon the learned VQ-VAE latent space using custom outpainting orderings. Orders move outward from reprojections (Figure 4); reprojections are a function of depths

predicted by the Depth Module. Training takes ~ 75 k iterations using a batch size of 60 and cross-entropy loss.

Curriculum Learning. The Depth and Refinement Modules are trained via a curriculum. They first learn to synthesize small view changes, then generalize to larger angles. For the first 25k iterations, we train at the same rotation as [53]. For Matterport, this is 20° in each Euclidean direction, for RealEstate10K it is 5° total. Next, we increase maximum rotation by this amount, and repeat this increase every 25k iterations until reaching our target rotation.

Outpainting Inference Details: Outpainting produces diverse completions, which is a dual-edged sword: some are good, but many will be inconsistent with the input. We thus produce multiple samples and select the best. Selection uses the complementary signals of classifier entropy and discriminator losses – samples that are consistent with inputs usually have less entropy over model classes (we use a Places [62] classifier), and detailed images usually have higher discriminator loss. Full details are in supplement.

Computational Reduction. We perform aggressive yet efficient pruning to the aggregate model, which becomes heavy otherwise. The Outpainter is most critical to speed. We reduce the depth of the autoregressive model by 60% and reduce width by 50%, and use 32×32 completions compared to VQVAE2’s of both 32×32 and 64×64 completions. We find that detail from 64×64 completions can instead be generated by pairing a 32×32 completions with the refinement module.

In total, our changes improve our inference speed by $10\times$ for 50 completions ($500\times$ for one completion), compared to using a full VQVAE2 and PixelCNN++ setup. One completion takes ≈ 1 minute using 50 samples, or 1 second with 1 sample. Training takes ≈ 5 days on 4 2080 Ti GPUs.



Figure 5: Consistent, High Quality Scenes. Given a single image, the proposed method generates images across large viewpoint changes. It both continues content (e.g. wall, bottom right) and invents consistent content (e.g. door, top left). Results shown on RealEstate10K.

4. Experiments

The goal of our experiments is to identify how well our proposed method can synthesize new *scenes* from a single image. We do this on standard datasets and compare with the state of the art (Sec. 4.1). Our task requires not only creating plausible new content, but also ensuring the created content is 3D consistent. We evaluate these two goals separately. We test the generated views for quality by independently evaluating each generated view (Sec. 4.2); we measure consistency by evaluating consistency across a *set* of overlapping views (Sec. 4.3).

4.1. Experimental setup

We evaluate throughout on standard datasets, using standard metrics. We compare our approach with baselines from the state of the art, as well as ablations that test alternate scene generation or view synthesis strategies.

Datasets. Following [53], we evaluate on Matterport3D [3] and RealEstate10K [63]. These enable the generation of pairs of views for training and evaluation. For consistency with past work, we follow a similar selection setup as [53], except we increase rotations; making corresponding changes in sampling to do so. Full details appear in the supplement.

Matterport: Image selection is done by an embodied agent doing randomized navigation in Habitat [41]. We increase the limits of [53] angle selection from 20° in each direction to 120° .

RealEstate10K: RealEstate10K is a collection of videos and image collection consists of selecting frames from a clip. SynSin selects pairs with angle changes of $\geq 5^\circ$ with maximum frame difference of 30. Increasing the angle change is not straightforward since $\geq 30^\circ$ changes are infrequent and can correspond to far away frames from different rooms.

We therefore select pairs of between 20° and 60° apart and $\leq 1\text{m}$ away. The average angle is $\approx 30^\circ$, roughly $8\times$ larger than SynSin. SynSin average angle is less than 5° because it sometimes re-samples; see [53] for details.

Evaluation Metrics. We evaluate content quality and consistency using human judgments as well as a set of automated metrics.

Human A/B Judgments: We evaluate image quality by asking annotators to compare generated images and consistency by asking annotators to compare image pairs. In both cases, we ask humans to make pairwise comparisons and report average preference rate compared to the proposed method: a method is worse than the proposed method if it is below 50%. Automatic evaluation of synthesis is known to be difficult, and we find that human judgments correlate with our own judgments more than automated systems.

Fréchet Inception Distance (FID) [14]: We evaluate how well generation images match on a distribution level using FID, which measures similarity by comparing distributions of activations from an Inception network. It has been shown to correlate well with human judgments [14], and we find it is the best automated measure of image quality.

PSNR and Perceptual Similarity [61]: PSNR and Perc Sim are standard metrics for comparing images. They are excellent measures of *consistency*, which is a unimodal task. Prior work [46, 52, 54] suggests that they are poor measures for conditional image generation since there are many modes of the output. We report them only for consistency with past work.

Baselines. We compare with existing work in the space of synthesizing unseen parts of rooms, as well as ablations that test components of our system (which are introduced when used). Our primary point of comparison is SynSin [53] since it is state of the art, although we evaluate other stan-

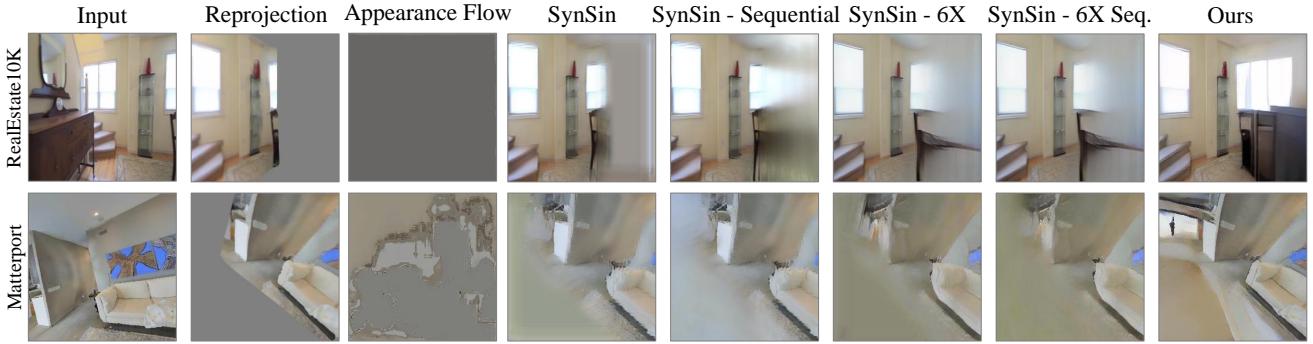


Figure 6: View Synthesis Ablations. Prior work is not capable of synthesizing large angle change, even with additional training and sequential generation. This typically leads to collapse. Explicit outpainting instead creates realistic and consistent content.

dard baselines [45, 47, 64]. In addition to standard SynSin, we evaluate many approaches to extending SynSin to handle the large rotations in our dataset.

SynSin [53]: One primary baseline is SynSin as described in [53] with no adaptation for extreme view change. We also evaluate the following extensions: (*SynSin - Sequential*) an autoregressive SynSin that breaks the transformation into 6 smaller transforms, accumulating 3D information; (*SynSin - 6X*) a SynSin model trained on larger view change; (*SynSin - 6X, Sequential*) a SynSin model trained on larger view changes and evaluated sequentially.

Other Baselines: We compare with a number of other view synthesis approaches, which tests whether any difficulties are specific to SynSin. In particular, we use: *Appearance Flow* [64]; *Tatarchenko et al. (Multi-View 3D from Single Image)* [45]; and *Single-View MPI* [47], which is only available on RealEstate10K.

4.2. Evaluating Quality

We begin by measuring the generated image quality. Being able to synthesize realistic images well beyond inputs is critical to generate an immersive scene.

Qualitative Results. Figure 5 shows that the proposed method can produce high-quality, 3D-consistent images across large angle changes. The images suggest the method is capable of continuing visible scene information realistically, including an entirely new door that is consistent with the original image content on top left and the continuation of the textured wall on bottom right.

Comparisons with prior work in Figure 6 show that the baselines struggle with large angle changes. Straightforward solutions like sequential generation or training on large angle changes do not succeed. While SynSin - 6X generates some results, it mainly repeats visible pixels. Our approach can extend visible information where appropriate, but also creates new objects like desks, windows, and tables.

Table 1: Image Quality as measured by A/B testing (preference frequency for a method compared to ours) as well as FID. In A/B tests, workers select the synthesized image better matching image reprojections choosing from the alternate method and ours. All baselines are preferred less often than our approach, and our approach better matches the true distribution as measured by FID. Single-View MPI [47] is not available on Matterport.

Method	Matterport A/B ↑ FID ↓	RealEstate A/B ↑ FID ↓
Tatarchenko <i>et al.</i> [45]	0.0% 427.0	0.0% 256.6
Appearance Flow [64]	19.8% 95.8	1.9% 248.3
Single-View MPI [47]	- -	2.7% 74.8
SynSin [53]	14.8% 72.0	5.8% 34.7
SynSin - Sequential	19.5% 77.8	11.5% 34.9
SynSin - 6X	27.3% 70.4	22.0% 27.9
SynSin - 6X, Sequential	21.2% 79.3	14.4% 33.1
Ours	- 56.4	- 25.5

Quantitative Results. Quantitative results in Table 1 are largely consistent with qualitative results from Figures 6. On Matterport, our explicit outpainting does substantially better across metrics compared to baselines including SynSin. Alternative baselines to SynSin perform worse, showing this is not a failing specific to SynSin. Training on larger rotation and applying sequential generation to SynSin help, but do not close the gap to our method.

On RealEstate10K, the gap is even larger for human judgment. Interestingly, SynSin - 6X does well on FID on RealEstate10K despite often producing repeated and mean colors. This is in part because RealEstate10K contains a high frequency of images looking through doorways. In these cases, the target view often includes the wall next to the doorway, which typically consists of bland and repeated colors. Thus, repeated colors become reasonable at a distribution level, even if the difference is clear to humans.

To follow past work, we report PSNR and Perceptual

Table 2: Traditional metrics such as PSNR are poor measures for extrapolation tasks [46, 52, 54], but are reported for reference.

Method	Matterport		RealEstate10K	
	PSNR ↑	Perc Sim ↓	PSNR ↑	Perc Sim ↓
Tatarchenko <i>et al.</i> [45]	13.72	3.82	10.63	3.98
Appearance Flow [64]	13.16	3.68	11.95	3.95
Single-View MPI [47]	-	-	12.73	3.45
SynSin - 6X, Sequential	15.61	3.17	14.21	2.73
Ours	14.60	3.17	13.10	2.88

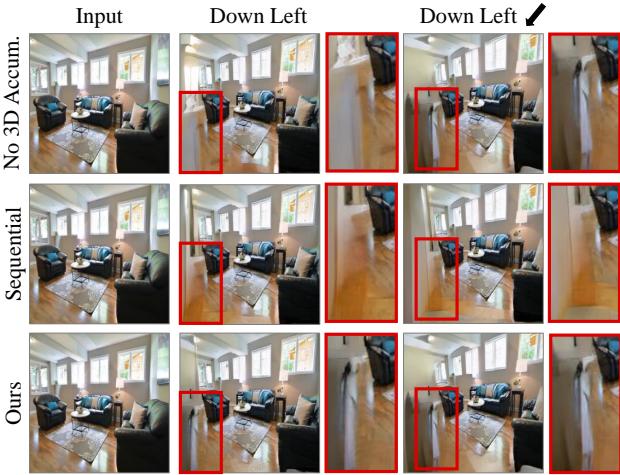


Figure 7: Consistency Ablations. The proposed method generates a consistent scene across views. Without 3D accumulation, outpainted regions are completely inconsistent. Sequential outpainting yields artifacts, due to using autoregressive completions in multiple views.

Similarity metrics in Table 2 for best performing methods (see supplement for all). These automated metrics, especially PSNR, are poor measures for extrapolation tasks [46, 52, 54], so A/B testing is the primary measure of success. The results of Appearance Flow in Figure 6 is evidence of this phenomenon. This method often produces entirely gray images in RealEstate10K, and loses to our method 98.1% of the time in A/B testing. Yet, its PSNR is competitive with other methods.

4.3. Evaluating Consistency

Having evaluated the quality of individual images, we next evaluate consistency. We note that consistency only matters if results are of high quality – producing a constant value is consistent. We therefore focus only on our approach and alternate accumulation strategies for our method. We evaluate consistency between a pair of generated results, one extreme view and an intermediate view. The setup follows view synthesis, with two exceptions: we

Table 3: Scene Consistency. A/B comparison of consistency. Workers select the most consistent pair of overlapping synthesized images (e.g. right two full images in Figure 7). All scores below 50 indicate the proposed method beats all ablations, on average. Consistency is lowest without 3D accumulation. Sequential order generation is less consistent than ours due to repeated outpainting.

Method	A/B vs. Ours ↑	
	Matterport	RealEstate10K
No 3D Accumulation	22.6%	7.5%
Sequential Generation	44.0%	36.2%
Ours	-	-

pick a large view change ($\sim 35^\circ$ horizontal, $\sim 17.5^\circ$ vertical) to ensure enough change to check consistency, and we use only horizontal and vertical rotation since camera roll makes judging consistency difficult. Full details are in supplement.

Alternate Strategies For Scene Synthesis. Throughout, we use our base model but compare with alternate strategies for scene synthesis. Specifically, we try:

Ours - No 3D Accumulation: We apply the method without accumulating the point cloud across generated images. This means outpainting takes place for each synthesized view, and outpainting is independent across views.

Ours - Sequential Generation: We apply the proposed 3D accumulation using the reverse order: this outpaints the missing region for the nearest image, then repeats outward. This results in outpainting in each new view, compared to our method which outpaints only one extremal view.

Qualitative Results. We show two outputs for an image in Figure 7. Without accumulation, one gets two wildly different results for each of the views (top row). Adding the accumulation helps resolve this, but doing it sequentially in two stages (middle row) produces visible artifacts. By generating a single large change first, our approach (bottom row) produces more consistent results.

Quantitative Results. A/B testing shown in Table 3 supports the qualitative findings: On RealEstate10K, independent generation is chosen only 7.5% of the time. Sequential generation accumulates a 3D representation, and performs better than the naïve method, but is less consistent than the proposed method.

We quantitatively validate these results using PSNR and Perceptual Similarity in a controlled setting. We use the same setup, but apply pure rotations to images, which means the resulting images are related by a homography. We apply this on RealEstate10K and use homographies to warp extreme to intermediate views and to warp intermediate to extreme views. We then calculate consistency using



Figure 8: Improving Sample Selection. Selection is crucial since the Outpainter creates a diversity of completions. Using classifier entropy yields completions consistent with inputs, while the trained discriminator provides more detail. Combined selection produces both consistent and detailed generations.

PSNR on overlapping regions and Perc Sim on warped images with non-overlapping regions masked. Without 3D accumulation does poorly, with Perc Sim/PSNR 0.606/13.6; sequential generation with 3D accumulation improves results tremendously to 0.456/17.9. The full method improves further to 0.419/18.6.

4.4. Ablations

Finally, we report some ablations of the method. These test the contribution of our latent space, the use of multiple samples, and the mechanism used to select the samples.

Ablations. We compare the proposed Outpainting Module and Sampling to alternatives.

Ours - RGB Autoregressive: We compare with using a RGB space to test the value of our latent space. Similar to prior work [40], we only consider a single completion for RGB. As opposed to VQ-VAE-based models, multiple completions is less helpful empirically.

Ours - 1 Completion: We evaluate effectiveness of our method with just one completion, which is more efficient but often less effective.

Ours - Classifier Selection: We apply our proposed method without using a discriminator for selection.

Ours - Discriminator Selection: We apply our proposed method without classifier included for selection.

Qualitative Results. An autoregressive approach alone does not fully explain our success. As we examine in Figure 8, the variety of autoregressive completions means sample selection is critical. While classifier entropy [35] selects sensible completions, they tend to lack detailed texture (left). In contrast, a discriminator selects completions with realistic textures, but they may not make sense with the entire scene (middle). We find the selection methods are complimentary. Combined they select sensible completions with realistic detail (right).

Quantitative Results. Table 4 confirms the qualitative results. On RealEstate10K, the baseline classifier and trained discriminator perform about as good or better than a single

Table 4: Synthesis Ablations. Comparison of autoregressive model and selection criteria. Our method beats all ablations on RealEstate10K. On Matterport, the same selection trends are true. However, Matterport’s scanned environments exhibit homogeneous lighting, so a single completion is sufficient to maximize autoregressive performance.

Method	Matterport		RealEstate10K	
	A/B vs. Ours ↑	FID ↓	A/B vs. Ours ↑	FID ↓
RGB Autoregressive	41.3%	60.73	29.6%	31.90
1 Completion	52.3%	55.46	38.4%	28.04
Classifier Selection	47.7%	59.78	44.9%	28.71
Discriminator Selection	47.9%	56.49	47.7%	26.30
Ours	-	56.36	-	25.53

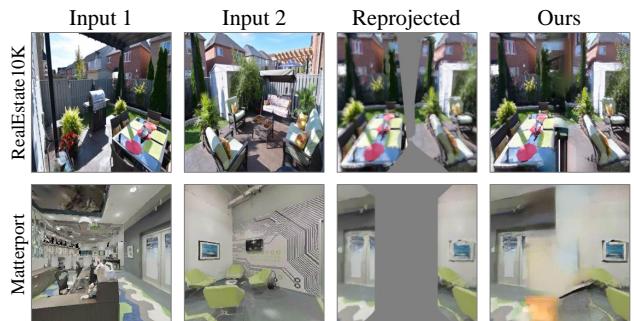


Figure 9: Two Input Synthesis. The proposed method can readily generalize to two input images due to building on point clouds.

completion. Again, combining the discriminator and classifier yields the best selections. Combining is also helpful on Matterport. However, its scanned environments tend to result in more homogeneous lighting, in comparison with the reflection effects of light in real images. As a result, a single completion is typically sufficient to maximize autoregressive performance. Finally, in the table, we confirm outpainting in a VQ-VAE space is superior to using RGB.

5. Discussion

We see synthesizing a rich, full world from a single image as a steep new challenge. Requiring only a single input opens up new experiences, but even with a single image, we see 3D awareness as important for good results and generality. For instance, our model’s 3D awareness enables the application of our system to two views as shown in Figure 9 by ingesting two point clouds.

Acknowledgments. We thank Angel Chang, Richard Tucker, and Noah Snavely for allowing us to share frames from their datasets, and Olivia Wiles and Ajay Jain for easily extended code. Thanks Shengyi Qian, Linyi Jin, Karan Desai and Nilesh Kulkarni for the helpful discussions.

A. Appendix

Video results available on the paper website give a thorough sense of model quality and consistency. As stated in the paper, the proposed method tends to produce high-quality, consistent *scenes*. In contrast, baselines such as SynSin – 6X are unable to create content, and ablations such as No 3D Accumulation are wildly inconsistent. These results are best seen in video; and are available at this URL: https://crockwell.github.io/pixelsynth/#video_results

The pdf portion of the supplemental material shows: detailed descriptions of model architectures (Section A.1); implementation details (Section A.2); details about the experimental setup (Section A.3); additional results (Section A.4); and additional and A/B testing details (Section A.5).

A.1. Model Architecture

As stated in the paper, a forward pass of the model takes in a single image and produces a consistent image in a novel view \mathbf{p} . This involves multiple components: a depth module D that maps images to depthmaps (producing point clouds); a projector π that projects a point cloud to a novel view; an outpainter O that can outpaint missing pixels by autoregressive modeling on the latent space of a VQ-VAE; and a refinement R module that adds details and corrects mistakes on a full image. We now provide more architectural details for each component; source code is available at <https://github.com/crockwell/pixelsynth>.

Depth Module D : The Depth Module takes in a $256 \times 256 \times 3$ image and predicts a depth for each pixel, yielding a $256 \times 256 \times 1$ depthmap. For fair comparison, we follow the U-Net used in SynSin [53], which consists of 8 encoding blocks that are mirrored by 8 decoding blocks.

Encoder: Each encoder block consists of a convolution (size: 4×4 / stride 2 / padding 1) followed by BatchNorm and leaky ReLU (negative slope of 0.2). Each block halves the width and height. The first convolution has 32 filters (mapping 3 channels to 32 channels); filter counts double at each block until reaching 256; they then remain constant.

Decoder: The decoder mirrors the encoder. Each block consists of a ReLU, $2 \times$ bilinear upsampling, convolution (3×3 / stride 1 / padding 1), and BatchNorm (except the last layer). Mirroring the encoder, filter counts remain the same (256) until the feature map has been upsampled to $\frac{1}{16}$ th of the input size. Filter counts then start halving. The last block does not have BatchNorm and has a final tanh at the top of the network.

Projector π : The Projector takes a colored point cloud and pose \mathbf{p} and projects it as if seen at \mathbf{p} . This produces a $256 \times 256 \times 3$ image along with an indication that of which pixels were projected to and which need to be outpainted. We implement the projection with the point cloud rendering

functions from Pytorch3D [34]. Our design decisions follow SynSin [53] for fair comparison: We alpha-composite points in the z-buffer and accumulate within a radius of 4 pixels.

We find that one change is important for autoregressive outpainting: we do not consider reprojected pixels at the edge of the visible region. Pixels that fall just outside the projected point cloud’s silhouette can be non-zero: each rendered pixel is a function of the projected points within a radius, and points just outside the silhouette are the result of interpolating some points and the missing regions. If we do not remove this, autoregressive outpainting begins with a border that is the mean color, which it tends to continue. We prevent this by treating border pixels as background/to-be-outpainted.

Outpainter O : The Outpainter takes as input the 256×256 reprojection from the Projector, possibly including large missing regions, and outpaints to a full image. This consists of performing autoregressive outpainting on the latent space of an autoencoder. Crucially, the autoregressive model follows an image-specific order because each image and new pose yields different missing regions. We describe the autoencoder, followed by the autoregressive model, then the autoregressive order. Our design decisions aim to make lightweight versions of VQVAE2 [35] for the autoencoder and the PixelCNN++ [40] used in Locally Masked Convolutions [18] for the autoregressive model.

Autoencoder: We follow a lightweight adaptation of VQ-VAE2 [35] that maps a $256 \times 256 \times 3$ input to a $32 \times 32 \times 1$ quantized embedding space Z and back. The encoder consists of first 3 convolution blocks followed by 2 ResNet blocks, then 2 convolution blocks and 2 ResNet blocks. The encoder produces a $32 \times 32 \times 64$ continuous output; each pixel in the encoded continuous space is quantized to an embedding $Z_{i,j,1} \in \mathbb{Z}_1^{512}$. The decoder first upsamples using a transpose convolution followed by a convolution. Then, it mirrors the encoder with 2 ResNet blocks followed by 2 transpose convolution blocks to produce a $256 \times 256 \times 3$ output.

Autoregressive model: The autoregressive model is a lightweight PixelCNN++ [40] that produces, as output, a distribution over the 512 possible quantized embedding values. Every convolution in the network uses locally-masked convolutions [18] for custom completion ordering. We follow the general design used by Locally Masked Convolutions [18] on CIFAR-10 consisting of 30 Gated ResNet blocks with 160 filters. However, we reduce its computational cost by using 12 Gated ResNet blocks with 80 filters, keeping everything else constant. For more details, we refer the reader to [18].

Autoregressive ordering: Autoregressive outpainting follows an image-specific order. We must use this custom or-

der because outpainting works best when one predicts adjacent pixels in a sequence using as much known data as possible. In some scenarios (e.g., extending a center crop), this can be achieved with a fixed order. However, in our case the particular points that must be outpainted depend on the depthmap as seen in the first image and the new pose from which it is projected.

Our order (Figure 4 of main paper) aims to go from closest out, following a spiral pattern. We achieve this by sorting the background/to-be-outpainted pixels in ascending distance to the center of mass of the foreground/projected-to pixels. We start with the closest pixel and add the closest adjacent point not in the generation order. We repeat this process until the entire image is ordering; ties due to pixels having equal distance are broken using a spiral pattern outwards from the center of mass.

Refinement Module R : The Refinement Module takes in a full $256 \times 256 \times 3$ outpainted image and produces the final, refined output of the same size $256 \times 256 \times 3$. This is trained adversarially and so it consists of a generator and discriminator.

Generator: The generator follows BigGAN [2] and SynSin [53] and consists of 8 ResNet blocks capped with a tanh. Following [2], there is an added downsampling block and with noise injection into BatchNorm throughout. Specifically, each ResNet block follows the following structure, consisting of two paths which are added. The first path consists of: a linear layer that injects noise followed by BatchNorm; ReLU; convolution (size: 3×3 / stride 1 / padding 1); a linear layer to inject noise followed by BatchNorm; ReLU; and convolution (size 3×3 / stride 1 / padding 1). The second path consists as a convolution (size 1×1 / stride 1 / padding 0), which is added to the input. Whenever a ResNet block downsamples, it uses average pooling to downsample the input during residual connection; whenever it upsamples, it uses bilinear upsampling.

Discriminator: The discriminator consists of 2 discriminator modules at different scales. Each discriminator contains 5 convolution blocks. Each block contains a convolution (size 4×4 / stride 2), followed by a Leaky ReLU (negative slope 0.2). The middle three blocks additionally contain an instance normalization layer between the conv and leaky ReLU.

A.2. Implementation Details

Outpainting Inference. The Outpainter’s autoregressive model produces its outputs by sampling. The forward pass produces a probability distribution over the vector embedded classes for every missing pixel in the 32×32 image. We find that best results are obtained by generating a set of full completions, followed by selection, and by adjusting the sampling temperature used during inference to balance

detail and error.

Sample selection. For each image, we generate 50 completions and select the best. We use a combination of the discriminator loss from R and a classifier entropy. The classifier is trained on MIT Places 365 [62]. Selection uses the average of ranks obtained by: (1) ranking in descending order of discriminator loss (since higher loss tends to correspond to issues with details); (2) ranking in ascending order of entropy (since sensible completions tend to be confident predictions of the classifier).

Sampling Temperature. Sampling temperature is important for balancing diversity and error. On Matterport, we use a sampling temperature of 0.5, which we find reduces strange completions but is still detailed. On RealEstate10K, we also use 0.5 temperature for the 1-completion model. Again, we see this temperature best balances realism with detailed completions. On RealEstate10K’s 50-completion model, we find we can increase sampling temperature to 0.7. While this means more completions are not sensible, we can select the best using an automated method. Therefore, the final outputs are more detailed and are still realistic.

Generating Multiple Viewing Directions. To create scenes as approximated in Figure 1 and seen in the supplemental video, support views are synthesized in eight directions: up, left, down, right, up-left, up-right, down-left, and down-right. These directions are selected to give a sense of all directions, and can be used to synthesize interior views without additional outpainting thereafter. When using multiple support views, we accumulate in a similar manner to the first support view: we lift existing information to 3D, reproject into the new support view, and outpaint as needed. In other words, we do not outpaint the same region again.

A.3. Experimental Setup

As detailed in the paper, we use two datasets to evaluate. Matterport uses embodied agent navigation to select paired views, while RealEstate10K selects paired frames from real video clips. We therefore use different processes to achieve a shared goal of selecting image pairs with large angle change.

Matterport: Matterport selection is straightforward because of embodiment. This consists of randomly drawing angle change in an embodied agent with a maximum of 120° in each direction. We use a dataset of 3.6k pairs.

RealEstate10K: On RealEstate10K, it is harder to select large angle changes because we are instead selecting from pairs of images in real videos. We select pairs of images such that the pairs have at least 20 degrees angle change. In order to attain such pairs, we allow sampling from anywhere in video clips, which can be over 270 frames apart. To minimize the number of view changes so extreme that input and target view are in different rooms, we limit translation at 1.0

meters, and angle at 60 degrees. We only consider pairs that meet this criteria, rather than resampling. Our filtered test set chooses 3600 pairs from over 3.5 million possible pairs across over 2.4k video clips. All selected evaluation pairs are cached for replicability.

A.4. Additional Results

We report additional results, including video predictions, which provide the best simultaneous display of quality and consistency.

Additional Qualitative Results: We first report additional qualitative results. Video results give a thorough sense of model quality and consistency, and are in the project webpage. As stated in the paper, the proposed method tends to produce high-quality, consistent scenes. In contrast, baselines such as SynSin - 6X are unable to create content, and ablations such as No 3D Accumulation are wildly inconsistent. Additional frame-level generated images are available in Figures 10 and 11.

Additional Quantitative Results: We report more extensive results for generated image quality. This is an expansion on Table 2 in the paper. Although these automated metrics are poor measures for extrapolation, we present in more detail in Table 5 for completeness. Ground truth depth is available in Matterport, meaning visible and non-visible regions can be attained, similar to SynSin. The same is not true of RealEstate10K, which contains real videos for which extensive ground truth labeling is prohibitive.

We reiterate the caution from the paper that PSNR has poor correlation with perceived quality when there are multiple possible completions (for instance during outpainting): Appearance Flow is competitive with other methods on PSNR but loses to our proposed method 98% of the time in A/B testing.

Limitations: Our primary limitation is outpainting both consistent and detailed content, especially on large view changes. There is a trade-off between the two, as a greater diversity of samples is required for detail, but can result in inaccurate content. While the approach of rejection sampling can improve outpainting errors, they remain a challenge, particularly on Matterport. In fact, on Matterport we reduce sampling temperature to minimize inconsistent completions, which can result in less detail. For instance, in Supp. Figure 11 row 3 column 1, notice missing content tends to repeat visible content rather than ending visible objects and creating new ones.

A.5. Additional and A/B Testing Details

A/B testing is the primary measure for success throughout experiments. This is common in work on extrapolation, as automated metrics tend to struggle. We detail our A/B testing framework below.

Table 5: Full PSNR and Perc Sim: Traditional metrics such as PSNR are poor measures for extrapolation tasks, but are reported for reference.

Method	Matterport			RealEstate10K		
	Both	PSNR ↑ InVis	Vis	Perc Sim ↓	PSNR ↑	Perc Sim ↓
Tatarchenko <i>et al.</i>	13.72	13.59	15.24	3.82	10.63	3.98
Appearance Flow	13.16	13.11	14.75	3.68	11.95	3.95
Single-View MPI	-	-	-	-	12.73	3.45
SynSin	15.05	14.35	17.86	3.13	13.92	2.77
SynSin - Sequential	14.31	13.36	17.65	3.14	13.30	2.78
SynSin - 6X	15.52	14.94	17.98	3.16	14.17	2.78
SynSin - 6X, Sequential	15.61	15.07	17.92	3.17	14.21	2.73
Ours	14.60	13.58	18.08	3.17	13.10	2.88

All A/B testing follows a standard A/B testing paradigm where human workers are shown images and are asked which is preferred given an input image. Workers were given instructions and example images and labels. They then had to pass a qualifier assessing whether they understood the task. Workers were also monitored by gold standard sentinel labels. All annotations were gathered using thehive.ai, a website similar to Amazon Mechanical Turk. Tasks are detailed below, and we share worker instructions.

Evaluating Quality via A/B: For comparisons of quality, we use novel view synthesis. A/B testing presents workers with the input image reprojected into a new view, and asks them to select the final image that makes more sense given this image. The reason we use reprojections as worker input instead of input images is it makes the A/B testing much clearer for workers. Using an input and rotation are very difficult to visualize, and thus difficult to compare across models. We also do not compare to ground truth output, as final images can vary drastically from ground truth and still be highly reasonable.

Instructions are shared in Figure 12. Ties are not allowed; final selection requires agreement of at least two workers. Reprojections use the learned depth from our model, which is effective on large view changes and therefore tend to be accurate, compared to ground truth images.

Evaluating Consistency via A/B: For consistency A/B testing, we use a similar novel view synthesis setup. However, instead of predicting one image, the model predicts two images such that the second generated image is half the rotation and translation of the first.

Workers are then asked to compare generated image pairs across methods. We ask them to do so by showing them a pair of images generated by two competing methods, pairs being stacked vertically. We do not use the input or ground truth images as the goal is not to judge quality, but only to judge consistency. Thus, even if one pair looks less realistic, it should be selected. Full instructions are displayed in Figure 13.

Consistency can be difficult for workers to judge as it re-

quires attending to small regions of each pair of images that may be slightly different. We therefore take several steps to maximize worker success. We use fixed rotations of large size ($\sim 35^\circ$ horizontal, $\sim 17.5^\circ$ vertical) to ensure angle change is apparent. We also constrain the rotation so that it must be in the horizontal and vertical directions, as additionally using roll rotations can make transformations confusing. Movement is also limited to that related to embodied rotation, since movement opposing rotation can make consistency difficult to evaluate. Finally, rotation is randomly selected for each image from one of eight possible directions seen in Figure 1: up, left, down, right, up-left, up-right, down-left, and down-right. This allows us to explicitly specify rotation direction of each image pair to help workers attend to specific regions of image pairs.

Evaluating Consistency via Homography: We validate A/B consistency using PSNR and Perceptual Similarity via homography. We use the same setup as in A/B testing, but instead apply only pure rotations to images. This enables homographies to warp across generated images. We do so in each pair both from intermediate to extreme images and from extreme to intermediate images. PSNR is then calculated on overlapping regions, while Perc Sim is calculated on warped images with non-overlapping regions masked. Scores are averaged across both directions in each pair.

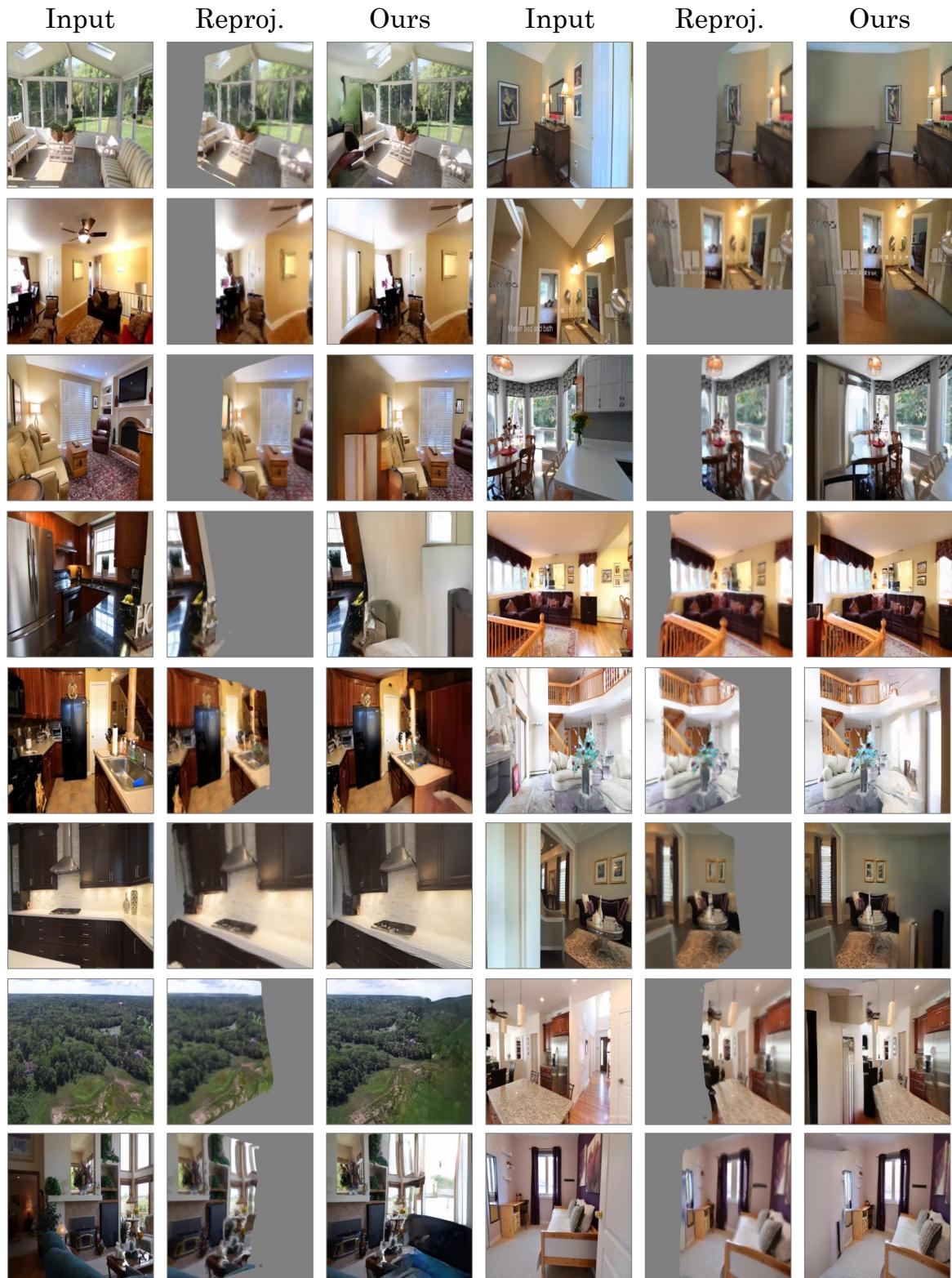


Figure 10: Additional Results on RealEstate10K.

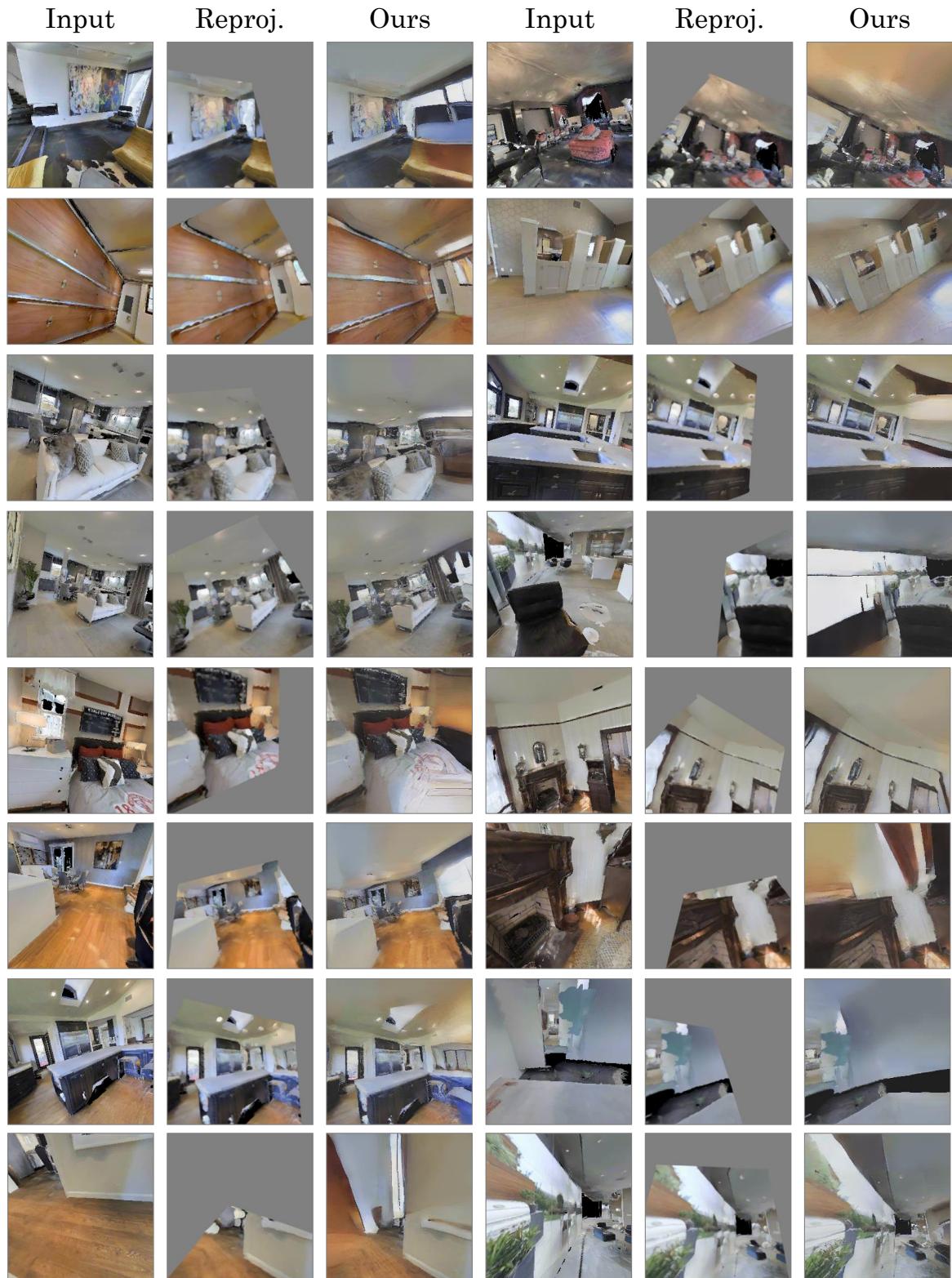


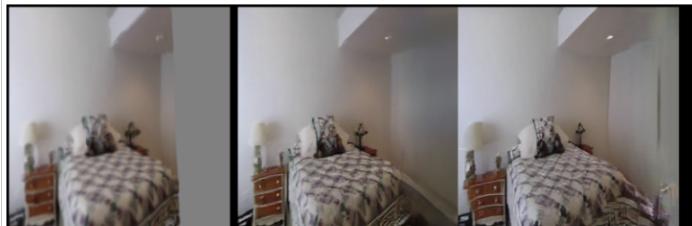
Figure 11: Additional Results on Matterport.

We have a system that is trying to generate images from part of the image. We'd like to identify cases where one version of the system is better than another version. Please select which of two output images is better given an input of part of the image.

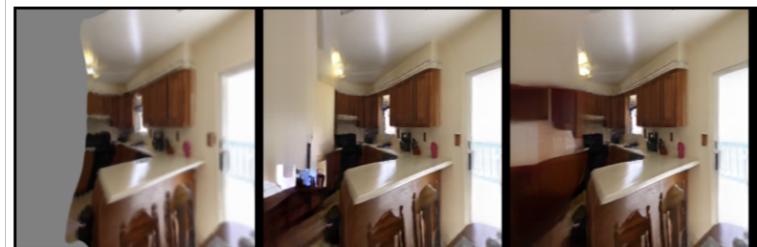
Below are some examples.



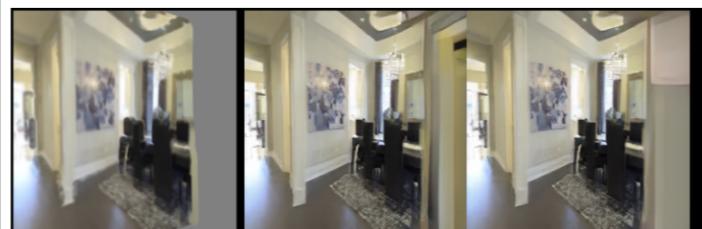
[Middle] This time, the image in the middle has has a straighter wall than the one on the right. Also the model on the right has blurring



[Right] Sometimes, neither completion is great. However, the one in the middle seems to make less sense than the one on the right, which continues cabinets and feels more like the same room.



[middle] Both generations seem pretty reasonable. However, the one in the middle is more interesting than the one on the right. There is a door on the right side instead of a wall with a strange box near the top.



[middle] While more detail is preferred, here the detail does make much sense, and the middle example is better.

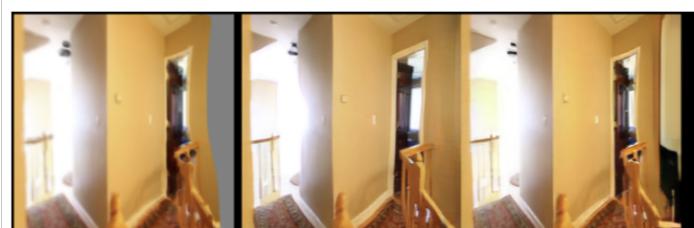


Figure 12: Quality A/B Worker Instructions.

We have a system that is trying to generate consistent images across small image rotations. We'd like to identify cases where one version of the system is better than another version. Please select which of two output pairs is more consistent.
Below are some examples.

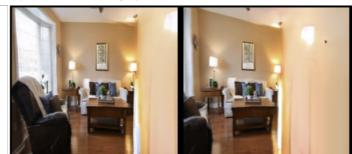
[Bottom] The left image is more consistent with the right image on the bottom, compared to the top. The top left image is not consistent with the top right. Note we don't care about quality of each image, only the consistency of the pair (top, or bottom).



[Bottom] Here, the bottom left image is more consistent with the bottom right, compared to the top left and top right. The top left has some strange blue area in the left image that does not appear in the right image.



[Top] Look on the right side of both images. The bottom image on the right looks very different than the bottom image on the left. The left image has yellow and black area on its right side, while the one on the right has a pink area to the right of the yellow and black. The top pair have a consistent orange / yellow in the same area.



[Top] Look at the bottom pair of images. The right image has a different shading on the very left side, as opposed to the left image, which has a different shade. So these images are not consistent. Meanwhile, the top pair of images continues a smooth brown (wall) across the pair.



Figure 13: Consistency A/B Worker Instructions.

References

- [1] Kara-Ali Aliev, Artem Sevastopolsky, Maria Kolos, Dmitry Ulyanov, and Victor Lempitsky. Neural point-based graphics. *arXiv preprint arXiv:1906.08240v3*, 2020. [2](#)
- [2] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale GAN training for high fidelity natural image synthesis. In *ICLR*, 2019. [2](#), [3](#), [10](#)
- [3] Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niessner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3d: Learning from rgb-d data in indoor environments. *3DV*, 2017. [2](#), [5](#)
- [4] Mark Chen, Alec Radford, Rewon Child, Jeffrey Wu, Heewoo Jun, David Luan, and Ilya Sutskever. Generative pretraining from pixels. In *ICML*, pages 1691–1703, 2020. [1](#), [2](#)
- [5] Qifeng Chen and Vladlen Koltun. Photographic image synthesis with cascaded refinement networks. In *ICCV*, pages 1511–1520, 2017. [2](#)
- [6] Shenchang Eric Chen and Lance Williams. View interpolation for image synthesis. In *SIGGRAPH*, pages 279–288, 1993. [2](#)
- [7] Xi Chen, Nikhil Mishra, Mostafa Rohaninejad, and Pieter Abbeel. Pixelsnail: An improved autoregressive generative model. In *ICML*, pages 864–872, 2018. [2](#)
- [8] Xu Chen, Jie Song, and Otmar Hilliges. Monocular neural image based rendering with continuous view control. In *ICCV*, pages 4090–4100, 2019. [2](#)
- [9] Paul E Debevec, Camillo J Taylor, and Jitendra Malik. Modeling and rendering architecture from photographs: A hybrid geometry-and image-based approach. In *SIGGRAPH*, pages 11–20, 1996. [2](#)
- [10] Patrick Esser, Robin Rombach, and Björn Ommer. Taming transformers for high-resolution image synthesis. *arXiv preprint arXiv:2012.09841*, 2020. [3](#)
- [11] Steven J Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F Cohen. The lumigraph. In *SIGGRAPH*, pages 43–54, 1996. [2](#)
- [12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016. [3](#)
- [13] Peter Hedman and Johannes Kopf. Instant 3d photography. *TOG*, 37(4):1–12, 2018. [2](#)
- [14] 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*, pages 6626–6637, 2017. [5](#)
- [15] Ronghang Hu and Deepak Pathak. Worldsheets: Wrapping the world in a 3d sheet for view synthesis from a single image. *arXiv preprint arXiv:2012.09854*, 2020. [1](#), [2](#)
- [16] Satoshi Iizuka, Edgar Simo-Serra, and Hiroshi Ishikawa. Globally and locally consistent image completion. *ToG*, 36(4):1–14, 2017. [2](#)
- [17] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *CVPR*, pages 1125–1134, 2017. [2](#)
- [18] Ajay Jain, Pieter Abbeel, and Deepak Pathak. Locally masked convolution for autoregressive models. In *UAI*, pages 1358–1367, 2020. [2](#), [3](#), [9](#)
- [19] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *CVPR*, pages 8110–8119, 2020. [2](#)
- [20] Hiroharu Kato, Yoshitaka Ushiku, and Tatsuya Harada. Neural 3d mesh renderer. In *CVPR*, pages 3907–3916, 2018. [2](#)
- [21] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi. Photo-realistic single image super-resolution using a generative adversarial network. In *CVPR*, pages 105–114, 2017. [2](#)
- [22] Marc Levoy and Pat Hanrahan. Light field rendering. In *SIGGRAPH*, pages 31–42, 1996. [2](#)
- [23] Andrew Liu, Richard Tucker, Varun Jampani, Ameesh Makadia, Noah Snavely, and Angjoo Kanazawa. Infinite nature: Perpetual view generation of natural scenes from a single image. *arXiv preprint arXiv:2012.09855*, 2020. [2](#)
- [24] Guilin Liu, Fitzsum A Reda, Kevin J Shih, Ting-Chun Wang, Andrew Tao, and Bryan Catanzaro. Image inpainting for irregular holes using partial convolutions. In *ECCV*, pages 85–100, 2018. [2](#)
- [25] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields. *arXiv preprint arXiv:2007.11571*, 2020. [2](#)
- [26] Shichen Liu, Weikai Chen, Tianye Li, and Hao Li. Soft rasterizer: Differentiable rendering for unsupervised single-view mesh reconstruction. In *ICCV*, 2019. [2](#)
- [27] Jacob Menick and Nal Kalchbrenner. Generating high fidelity images with subscale pixel networks and multidimensional upscaling. In *ICLR*, 2019. [2](#)
- [28] Moustafa Meshry, Dan B Goldman, Sameh Khamis, Hugues Hoppe, Rohit Pandey, Noah Snavely, and Ricardo Martin-Brualla. Neural rerendering in the wild. In *CVPR*, pages 6878–6887, 2019. [1](#), [2](#)

- [29] 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. [1](#), [2](#)
- [30] Aaron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. In *ICML*, 2016. [1](#), [2](#)
- [31] Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. In *NeurIPS*, 2017. [2](#), [3](#)
- [32] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. In *CVPR*, pages 2337–2346, 2019. [2](#)
- [33] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. *arXiv preprint arXiv:2102.12092*, 2021. [2](#), [3](#)
- [34] Nikhila Ravi, Jeremy Reizenstein, David Novotny, Taylor Gordon, Wan-Yen Lo, Justin Johnson, and Georgia Gkioxari. Accelerating 3d deep learning with pytorch3d. *arXiv preprint arXiv:2007.08501*, 2020. [3](#), [9](#)
- [35] Ali Razavi, Aaron van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with vq-vae-2. In *NeurIPS*, pages 14866–14876, 2019. [1](#), [2](#), [3](#), [4](#), [8](#), [9](#)
- [36] Scott Reed, Aäron van den Oord, Nal Kalchbrenner, Sergio Gómez Colmenarejo, Ziyu Wang, Dan Belov, and Nando De Freitas. Parallel multiscale autoregressive density estimation. In *ICML*, 2017. [2](#)
- [37] Gernot Riegler and Vladlen Koltun. Free view synthesis. In *ECCV*, pages 623–640. Springer, 2020. [2](#)
- [38] Robin Rombach, Patrick Esser, and Björn Ommer. Geometry-free view synthesis: Transformers and no 3d priors. *arXiv preprint arXiv:2104.07652*, 2021. [2](#)
- [39] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, pages 234–241. Springer, 2015. [3](#)
- [40] Tim Salimans, Andrej Karpathy, Xi Chen, and Diederik P. Kingma. Pixelcnn++: A pixelcnn implementation with discretized logistic mixture likelihood and other modifications. In *ICLR*, 2017. [1](#), [2](#), [3](#), [8](#), [9](#)
- [41] Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, et al. Habitat: A platform for embodied ai research. In *ICCV*, pages 9339–9347, 2019. [2](#), [5](#)
- [42] S. M. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. In *CVPR*, volume 1, pages 519–528, 2006. [2](#)
- [43] Meng-Li Shih, Shih-Yang Su, Johannes Kopf, and Jia-Bin Huang. 3d photography using context-aware layered depth inpainting. In *CVPR*, pages 8028–8038, 2020. [1](#), [2](#)
- [44] Vincent Sitzmann, Justus Thies, Felix Heide, Matthias Nießner, Gordon Wetzstein, and Michael Zollhofer. Deepvoxels: Learning persistent 3d feature embeddings. In *CVPR*, pages 2437–2446, 2019. [1](#), [2](#)
- [45] Maxim Tatarchenko, Alexey Dosovitskiy, and Thomas Brox. Multi-view 3d models from single images with a convolutional network. In *ECCV*, pages 322–337. Springer, 2016. [2](#), [6](#), [7](#)
- [46] Piotr Teterwak, Aaron Sarna, Dilip Krishnan, Aaron Maschinot, David Belanger, Ce Liu, and William T Freeman. Boundless: Generative adversarial networks for image extension. In *ICCV*, pages 10521–10530, 2019. [2](#), [5](#), [7](#)
- [47] Richard Tucker and Noah Snavely. Single-view view synthesis with multiplane images. In *CVPR*, 2020. [1](#), [2](#), [6](#), [7](#)
- [48] Shubham Tulsiani, Richard Tucker, and Noah Snavely. Layer-structured 3d scene inference via view synthesis. In *ECCV*, pages 302–317, 2018. [1](#), [2](#)
- [49] Aaron Van den Oord, Nal Kalchbrenner, Oriol Vinyals, Lasse Espeholt, Alex Graves, and Koray Kavukcuoglu. Conditional image generation with pixelcnn decoders. In *NeurIPS*, volume 29, pages 4790–4798, 2016. [1](#), [2](#)
- [50] Qianqian Wang, Zhicheng Wang, Kyle Genova, Pratul Srinivasan, Howard Zhou, Jonathan T. Barron, Ricardo Martin-Brualla, Noah Snavely, and Thomas Funkhouser. Ibrnet: Learning multi-view image-based rendering. In *CVPR*, 2021. [1](#), [2](#)
- [51] 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*, pages 8798–8807, 2018. [2](#), [3](#), [4](#)
- [52] Yi Wang, Xin Tao, Xiaoyong Shen, and Jiaya Jia. Wide-context semantic image extrapolation. In *CVPR*, pages 1399–1408, 2019. [2](#), [5](#), [7](#)
- [53] Olivia Wiles, Georgia Gkioxari, Richard Szeliski, and Justin Johnson. Synsin: End-to-end view synthesis from a single image. In *CVPR*, pages 7467–7477, 2020. [1](#), [2](#), [3](#), [4](#), [5](#), [6](#), [9](#), [10](#)
- [54] Chao Yang, Xin Lu, Zhe Lin, Eli Shechtman, Oliver Wang, and Hao Li. High-resolution image inpainting

- using multi-scale neural patch synthesis. In *CVPR*, pages 6721–6729, 2017. 2, 5, 7
- [55] Zongxin Yang, Jian Dong, Ping Liu, Yi Yang, and Shuicheng Yan. Very long natural scenery image prediction by outpainting. In *ICCV*, pages 10561–10570, 2019. 2
- [56] Jae Shin Yoon, Kihwan Kim, Orazio Gallo, Hyun Soo Park, and Jan Kautz. Novel view synthesis of dynamic scenes with globally coherent depths from a monocular camera. In *CVPR*, pages 5336–5345, 2020. 2
- [57] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelnerf: Neural radiance fields from one or few images, 2020. 2
- [58] Jiahui Yu, Zhe Lin, Jimei Yang, Xiaohui Shen, Xin Lu, and Thomas S Huang. Generative image inpainting with contextual attention. In *CVPR*, pages 5505–5514, 2018. 2
- [59] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-attention generative adversarial networks. In *ICML*, pages 7354–7363, 2019. 2
- [60] Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. Nerf++: Analyzing and improving neural radiance fields. *arXiv preprint arXiv:2010.07492*, 2020. 2
- [61] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, pages 586–595, 2018. 4, 5
- [62] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. *TPAMI*, 2017. 4, 10
- [63] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: Learning view synthesis using multiplane images. In *SIGGRAPH*, 2018. 2, 5
- [64] Tinghui Zhou, Shubham Tulsiani, Weilun Sun, Jitendra Malik, and Alexei A Efros. View synthesis by appearance flow. In *ECCV*, pages 286–301. Springer, 2016. 2, 6, 7
- [65] C Lawrence Zitnick, Sing Bing Kang, Matthew Uyttendaele, Simon Winder, and Richard Szeliski. High-quality video view interpolation using a layered representation. *TOG*, 23(3):600–608, 2004. 2