arXiv:2012.09855v2 [cs.CV] 18 Dec 2020

Infinite Nature: Perpetual View Generation of Natural Scenes from a Single Image

Andrew Liu* Richard Tucker* Varun Jampani
Ameesh Makadia Noah Snavely Angjoo Kanazawa
Google Research

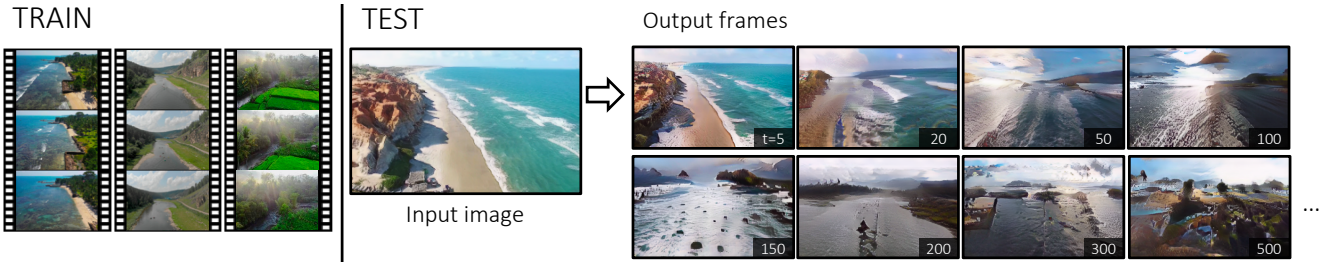

Figure 1. **Perpetual View Generation.** From a collection of aerial videos of nature scenes (left), we propose a method that can perpetually generate novel views for a camera trajectory covering a long distance from a single image (right). Our method can successfully generate hundreds of frames of an aerial video from a single input image (up to 500 shown here).

Abstract

We introduce the problem of perpetual view generation—long-range generation of novel views corresponding to an arbitrarily long camera trajectory given a single image. This is a challenging problem that goes far beyond the capabilities of current view synthesis methods, which work for a limited range of viewpoints and quickly degenerate when presented with a large camera motion. Methods designed for video generation also have limited ability to produce long video sequences and are often agnostic to scene geometry. We take a hybrid approach that integrates both geometry and image synthesis in an iterative 'render, refine *and* repeat' *framework, allowing for long-range generation that cover large distances over hundreds of frames. Our approach can be trained from a set of monocular video sequences without any manual annotation. We propose a dataset of aerial footage of natural coastal scenes, and compare our method with recent view synthesis and conditional video generation baselines, showing that it can generate plausible scenes for much longer time horizons over camera trajectories covering a large distance compared to existing methods. Please visit our project page at* https://infinite-nature.github.io/.

* indicates equal contribution

1. Introduction

Consider the input image of a coastline in Fig. 1. Imagine flying through this scene on a small airplane. Initially, we would see objects grow in our field of view as we approach them. Beyond, we might find a wide ocean or new islands. At the shore, we might see cliffs or beaches, while inland there could be mountains or forests. As humans, we are good at imagining a plausible world from a single picture, based on our own experiences. How can we build a system that can similarly imagine the world beyond the edges of a single image?

We introduce the problem of *perpetual view generation*: given a single image of a scene, the goal is to continually generate new views of the scene corresponding to an arbitrarily long camera trajectory, giving the effect of flying through an imaginary world generated from the given image. Solving this problem requires extrapolating new content for unseen regions and synthesizing new details in existing regions as the camera approaches them. Building an infinitely generative scene model has applications in content creation, novel photo interactions, and methods that use learned world models like model-based reinforcement learning.

However, generating a long video from a static image is an extremely challenging problem. Two active areas of research, video synthesis and view synthesis, both fail to scale to this problem for different reasons. Recent video synthe-

sis methods generate limited numbers of novel frames (e.g., 25 [40] or 48 frames [9]), even when trained with massive compute resources. These methods take advantage of recent developments in image synthesis [20] and apply them to the temporal domain or rely on recurrent models [10]. However, they often neglect an important element of the video's structure—a video is a function of both the underlying scene *and* camera geometry. Proper geometry imposes constraints on how a video can evolve and is crucial for synthesizing moving camera sequences such as the ones we tackle.

In contrast, many view synthesis methods do take advantage of geometry to synthesize high-quality novel views. However, these approaches can only operate within a limited range of camera motions. As shown in Fig. 6, if the camera strays too far, such methods fail catastrophically. To successfully generate distant views, an algorithm will need to inpaint disoccluded regions, extrapolate (i.e. outpaint) unseen regions beyond the boundary of the previous frames, and add detail to (super-resolve) areas that approach the camera over time.

We propose a hybrid framework that takes advantage of both geometry and image synthesis techniques to address these challenges. Specifically, we use disparity maps to encode the scene geometry, and decompose the perpetual view generation task into the framework of *render-refine*-and-*repeat*. First, we **render** the current frame from a new viewpoint, using disparity to ensure that scene content moves in a geometrically correct manner. Then, we **refine** the resulting image and geometry. This step adds detail and synthesizes new content in areas that require inpainting, outpainting, and super-resolution. Because we refine both the image and disparity, the whole process can be **repeated** in an auto-regressive manner, allowing for perpetual generation of novel views.

To train our system, we curated a large dataset of drone footage of nature and coastal scenes from over 700 videos, spanning 2 million frames. We run a structure from motion pipeline to recover 3D camera trajectories, and refer to this as the Aerial Coastline Imagery Dataset (ACID), which we have released publicly. Our trained model can generate sequences of hundreds of frames while maintaining the aesthetic feel of an aerial coastal video, even though after just a few frames, the camera has moved beyond the edges of the original image.

Our experiments show that the render-refine-repeat structure of our framework is key to tackling this problem. Compared to recent view synthesis and video generation baselines, we show that our approach can produce plausible frames for much longer time horizons than prior methods. While there is more to be done on this immensely challenging task, our work shows the potential of merging geometry and generative models, and we hope it inspires more research in this direction.

2. Related Work

Image extrapolation. Our work is inspired by the seminal work of Kaneva *et al*. [19], which proposed a non-parametric approach for generating 'infinite' images by means of stitching 2D-transformed images, and by patch-based non-parametric approaches for image extension [29, 1]. We revisit the 'infinite images' concept in a learning framework that also reasons about the 3D geometry behind each image. Also related to our work are recent deep learning approaches to the problem of *outpainting*, i.e., inferring unseen content outside image boundaries [43, 45, 36], as well as *inpainting*, the task of filling in missing content within an image [15, 47]. These approaches use adversarial frameworks and semantic information for in/outpainting. Our problem also incorporates aspects of super-resolution [14, 23]. Image-specific GAN methods also demonstrate a form of image extrapolation and super-resolution of textures and natural images [50, 34, 30, 33]. In contrast to the above methods, we reason about the 3D geometry behind each image and study image extrapolation in the context of temporal image sequence generation.

View synthesis. Many view synthesis methods operate by interpolating between multiple views of a scene [24, 3, 25, 12, 7], although recent work can generate new views from just a single input image, as in our work [5, 38, 26, 37, 31, 6]. However, in both settings, most methods only allow for a very limited range of output viewpoints. Even methods that explicitly allow for view extrapolation (not just interpolation) typically restrict the camera motion to small regions around a reference view [49, 35, 8].

One factor that limits camera motion is that many methods construct a static scene representation, such as a layered depth image [38, 32], multiplane image [49, 37], or point cloud [26, 44], and inpaint disoccluded regions. Such representations can allow for fast rendering, but the range of viable new camera positions is limited by the finite bounds of the scene representation. Some methods augment this scene representation paradigm, enabling a limited increase in the range of output views. Niklaus *et al*. perform inpainting *after* rendering (then project back into a point cloud) [26], while SynSin uses a post-rendering refinement network to produce realistic images from feature point-clouds [44]. We take inspiration from these methods by rendering and then refining our output. In contrast, however, our system does not construct a single 3D representation of a scene. Instead we proceed iteratively, generating each output view from the previous one, and producing a geometric scene representation in the form of a disparity map for each frame.

Some methods use video as training data. Monocular depth can be learned from 3D movie left-right camera pairs [22] or from video sequences analysed with structure-from-motion techniques [4]. Video can also be directly used for

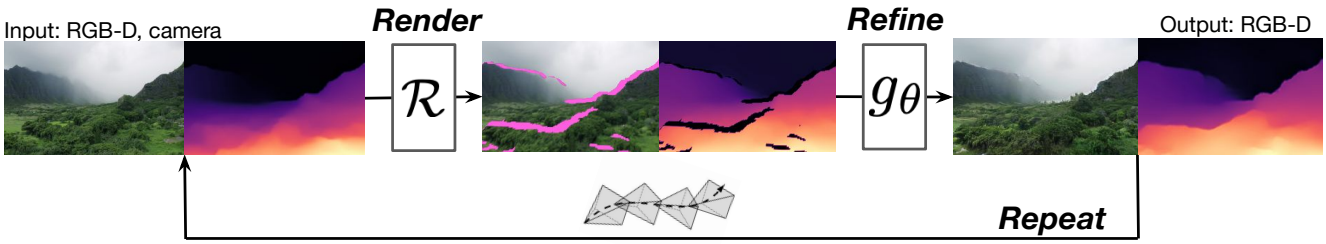

Figure 2. **Overview.** We first *render* an input image to a new camera view using the disparity. We then *refine* the image, synthesizing and super-resolving missing content. As we output both RGB and geometry, this process can be *repeated* for perpetual view generation.

view synthesis [37, 44]. These methods use pairs of images, whereas our model is trained on sequences of several widely-spaced frames since we want to generate long-range video.

Video synthesis. Our work is related to methods that generate a video sequence from one or more images [41, 11, 42, 10, 39, 46]. Many such approaches have focused on predicting the future of dynamic objects with a static camera, often using simple videos of humans walking [2] or robot arms [11]. In contrast, we focus on mostly static scenes with a moving camera, using real aerial videos of nature. Some recent research addresses video synthesis from in-the-wild videos with moving cameras [9, 40], but without taking geometry explicitly into account, and with strict limits on the the length of the generated video. By accounting for geometry, the task of video prediction becomes conceptually easier as the movement of pixels from camera motion can be explicitly modeled using 3D geometry. Our work takes a step towards jointly modeling the scene geometry and camera motion for video generation.

3. Perpetual View Generation

We introduce *perpetual view generation*, the task of continually generating novel views of a scene corresponding to an arbitrary long camera trajectory. Specifically, at test time, given an RGB image I_0 and a camera trajectory $\{P_0, P_1, P_2, \dots\}$ of arbitrary length, the task is to output a new image sequence $\{I_0, I_1, I_2, \dots\}$ that forms a video depicting a flythrough of the scene captured by the initial view. The trajectory is a series of 3D camera poses $P_t = \left(\begin{smallmatrix} R^{3\times3} & \mathbf{t}^{3\times1} \\ 0 & 1 \end{smallmatrix}\right)$, where R and $\mathbf{t}$ are 3D rotations and translations, respectively. In addition, each camera has an intrinsic matrix K. At test time the camera trajectory may be pre-specified or controlled by an auto-flight algorithm. At training time camera data is obtained from video clips via structure-from-motion as in [49].

3.1. Approach: Render, Refine, Repeat

We decompose *perpetual view generation* into three steps, as illustrated in Fig. 2:

1. **Render** a new view from an old view, by warping the image according to a disparity map using a differentiable renderer,
2. **Refine** the rendered view and geometry to fill in missing content and add detail where necessary,
3. **Repeat** this process, generating each future view from the previous one.

Our approach has several desirable characteristics. *Representing geometry* with a disparity map allows much of the heavy-lifting of moving pixels from one frame to the next to be handled by differentiable rendering, ensuring local temporal consistency. The synthesis task is then reduced to one of *image refinement*, which comprises: 1) inpainting disoccluded regions 2) outpainting of new image regions and 3) super-resolving image content. Here, techniques from recent image synthesis and generative models can be applied to produce realistic images. Because every step is *fully differentiable*, we can train our refinement network by backpropagating through several view generation iterations. And because the framework is *auto-regressive*, novel views may be infinitely generated with explicit view control, even though training data is finite in length.

More formally, for an image I_t at camera P_t we have an associated disparity map (i.e., an inverse depth map) $D_t \in \mathbb{R}^{H\times W}$, and we compute the next frame I_{t+1} and its disparity D_{t+1} as

$$\hat{I}_{t+1}, \hat{D}_{t+1}, \hat{M}_{t+1} = \mathcal{R}(I_t, D_t, P_t, P_{t+1}), \quad (1)$$

$$I_{t+1}, D_{t+1} = g_\theta(\hat{I}_{t+1}, \hat{D}_{t+1}, \hat{M}_{t+1}). \quad (2)$$

Here, $\hat{I}_{t+1}$ and $\hat{D}_{t+1}$ are the result of rendering the image I_t and disparity D_t from the new camera P_{t+1}, using a differentiable renderer $\mathcal{R}$ [13]. This function also returns a mask $\hat{M}_{t+1}$ which indicates which regions of the image are missing and need to be filled in. The refinement network g_θ then inpaints, outpaints and super-resolves these inputs to produce the next frame I_{t+1} and its disparity D_{t+1}. The process is repeated iteratively for T steps during training, and at test time for an arbitrary length camera trajectory. Next we discuss each step in detail.

Geometry and Rendering. Our render step $\mathcal{R}$ uses a differentiable mesh renderer [13]. First, we convert each pixel coordinate (u, v) in I_t and its corresponding disparity d in D_t into a 3D point in the camera coordinate system: $(x, y, z) = K^{-1}(u, v, 1)/d$. We then convert the image into a 3D triangular mesh where each pixel is treated as a vertex connected to its neighbors, ready for rendering.

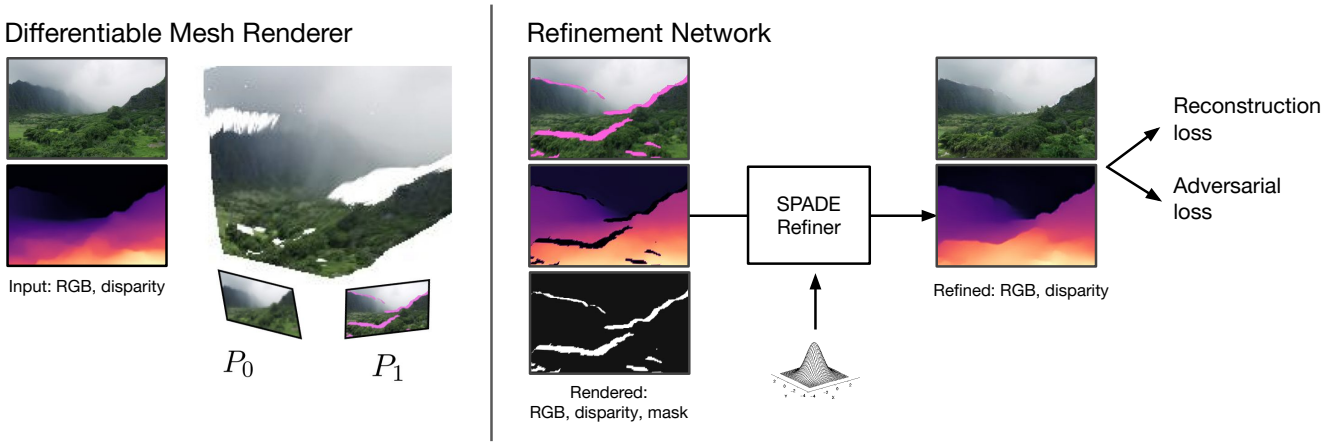

Figure 3. **Illustration of the rendering and refinement steps.** (Left): Our differentiable rendering stage takes a paired RGB image and disparity map from viewpoint P_0 and creates a textured mesh representation, which we render from a new viewpoint P_1, warping the textures, adjusting disparities, and returning a binary mask representing regions to fill in. (Right) The refinement network takes the output of the renderer and uses SPADE [27] as our network architecture to fill in holes and add details. The output is a new RGB image and disparity map that can be supervised with reconstruction and adversarial losses.

To avoid stretched triangle artefacts at depth discontinuities, and to aid our refinement network by identifying regions to be completed, we compute a per-pixel binary mask $M_t \in \mathbb{R}^{H \times W}$ by thresholding the gradient of the disparity image $\nabla \hat{D}_t$, computed with a a Sobel filter:

$$M_t = \begin{cases} 0 & \text{where } ||\nabla \hat{D}_t|| > \alpha, \\ 1 & \text{otherwise.} \end{cases} \quad (3)$$

The 3D mesh, textured with the image I_t and mask M_t, is then rendered from the new view P_{t+1}, and the rendered image is multiplied element-wise by the rendered mask to give $\hat{I}_{t+1}$. The renderer also outputs a depth map as seen from the new camera, which we invert and multiply by the rendered mask to obtain $\hat{D}_{t+1}$. This use of the mask ensures that any regions in $\hat{I}_{t+1}$ and $\hat{D}_{t+1}$ that were occluded in I_t are masked out and set to zero (along with regions that were outside the field of view of the previous camera). These areas are ones that the refinement step will have to inpaint (or outpaint). See Fig. 2 and Fig. 3 for examples of missing regions shown in pink.

Refinement and Synthesis. Given the rendered image $\hat{I}_{t+1}$, its disparity $\hat{D}_{t+1}$ and its mask M_{t+1}, our next task is to refine this image, which encapsulates inpainting, outpainting, and super-resolution. For this refinement stage we want a model that is both generative and conditional, so we adopt the state-of-the-art conditional image synthesis framework of Park *et al*., which uses spatially-adaptive normalization (SPADE) to condition the image synthesis process at multiple scales [27]. While the original SPADE approach generated an image from a semantic segmentation input, in our work the input is the rendered image, disparity, and mask. The generator output is a 4-channel image comprising RGB and disparity channels. We also train a single encoder that encodes the initial input image I_0 to compute the latent noise. This module, shown on right in Fig. 3, is trained with both reconstruction and adversarial losses.

Rinse and Repeat. A crucial part of our approach is to not just refine the RGB pixels, but also the disparity as well. Together the geometry (represented by a disparity) and RGB texture provide the necessary information for our renderer to produce the next view. This insight is key for allowing our approach to repeat itself perpetually, as each frame and disparity depend only on the previous one.

Because our render-refinement steps are entirely self-contained, there is no global scene representation—indeed the only representation is the image and disparity output from the refinement network at each step. While this is advantageous because a global representation is expensive to store perpetually, it means that our render-refine-repeat loop is memory-less and as a result there is no guarantee of global consistency across multiple iterations.

Geometric Grounding to Prevent Drift. A notorious challenge in long generation of sequences is dealing with the accumulation of errors [28]. In a system where current prediction affects future outputs, subtle deviations in each iteration may compound, eventually generating predictions that are unseen during training and causing unexpected behaviors. Repeating the generation loop in the training process and feeding the network with its own output ameliorates the drift as can be seen in the *'No Repeat'* ablation study (Section 6). However, we notice that the output, particularly the disparity, can still drift at test time. Therefore we propose an explicit geometric re-grounding of the disparity maps.

Specifically, we take advantage of the fact that the rendering process provides the correct range of disparity from a new viewpoint $\hat{D}_{t+1}$ for visible regions of the previous

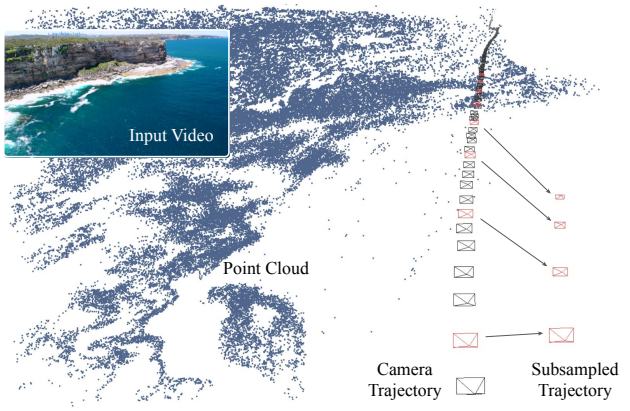

Figure 4. **Processing video for ACID.** We run structure from motion on coastline drone footage collected from YouTube to create the Aerial Coastline Imagery Dataset (ACID). See Section 4.

frame. The refinement network may modify these values as it refines the holes and blurry regions, which can lead to drift as the overall disparity gradually may change. However, we can geometrically correct this by rescaling the refined disparity map to the correct range by computing a scale factor γ via solving

$$\min_{\gamma} ||M \odot (\log(\gamma D_{t+1}) - \log(\hat{D}_{t+1}))||. \quad (4)$$

By scaling the refined disparity by γ, our approach ensures that the disparity map stays at a consistent scale, which significantly reduces drift at test time as shown in Section 6.2.

4. Aerial Coastline Imagery Dataset (ACID)

Learning to generate long sequences requires real image sequences for training. Many existing datasets for view synthesis do not use sequences, but only a set of views from slightly different camera positions. Those that do have sequences are limited in length: RealEstate10K, for example, has primarily indoor scenes with limited camera movement [49]. To obtain long sequences with a moving camera and few dynamic objects, we turn to aerial footage of beautiful nature scenes that are available on the Internet. Nature scenes are a good starting point for attempting our challenging problem, as GANs have shown promising results on nature textures [30, 33]. We collected 765 videos using keywords such as 'coastal' and 'aerial footage', and processed these videos with SLAM and structure-from-motion following the approach of Zhou *et al*. [49], yielding over 13,000 sequences with a total of 2.1 million frames. We make the list of videos and the SfM camera trajectories available. See Fig. 4 for an illustrative example of our SfM pipeline running on a coastline video.

Disparity We use the off-the-shelf MiDaS single-view depth prediction method [22] to obtain disparity maps for every frame. We find that MiDaS is quite robust and produces sufficiently accurate disparity maps for our method. Because MiDaS disparity is only predicted up to scale and shift, it must first be rescaled to match our data. To achieve this, we use the sparse point-cloud computed for each scene during structure from motion. For each frame we consider only the points that were tracked in that frame, and apply least-squares to compute the optimal scale and shift which minimize the disparity error on these points. We apply this scale and shift to the MiDaS output to obtain disparity maps $\{D_i\}$ which are scale-consistent with the SfM camera trajectories $\{P_i\}$ for each sequence.

Aligning Camera Speed. The speed of camera motion varies widely in our collected videos, so we compute a proxy of camera speed in order to normalize the amount of motion present in training image sequences. We use the translation magnitude of the estimated camera poses between frames after scale-normalizing the video as in Zhou *et al*. [49] to determine a range of rates at which each sequence can be subsampled in order to obtain a camera speed within a desired target range. We randomly select frame rates within this range to subsample videos. We picked a target speed range for training sequences that varies by up to 30% and, on average, leaves 90% of an image's content visible in the next sampled frame. Fig. 4 shows an example of subsampling.

5. Experimental Setup

Losses. We train our approach on a collection of image sequences $\{I_t\}_{t=0}^{T}$ with corresponding camera poses $\{P_t\}_{t=0}^{T}$ and disparity maps for each frame $\{D_t\}_{t=0}^{T}$. Following the literature on conditional generative models, we use an L1 reconstruction loss on RGB and disparity, a VGG perceptual loss on RGB [18] and a hinge-based adversarial loss with a discriminator [27] for the T frames that we synthesize during training. We also use a KL-divergence loss [21] on our encoder $\mathcal{L}_{\text{KLD}} = \mathcal{D}_{\text{KL}}(q(z|x)||\mathcal{N}(0,1))$. Our complete loss function is

$$\mathcal{L} = \mathcal{L}_{\text{reconst}} + \mathcal{L}_{\text{perceptual}} + \mathcal{L}_{\text{adversarial}} + \mathcal{L}_{\text{KLD}} \quad (5)$$

The loss is computed over all iterations and over all samples in the mini-batch.

Metrics. Evaluating the quality of the generated images in a way that correlates with human judgement is a challenge. We use the Fréchet inception distance (FID), a common metric used in evaluating generative models of images. FID computes the difference between the mean and covariance of the embedding of real and fake images through a pretrained Inception network [17] to measure the realism of the generated images as well as their diversity. We precompute real statistics using 20k real image samples from our dataset. To measure changes in generated quality over time, we report FID over a sliding window: we write FID-w at t to indicate

Method	LPIPS ↓	MSE ↓	FID ↓
SVG-LP [10]	0.60	0.020	135.9
SynSin [44]	0.32	**0.018**	98.1
SynSin-Iter	0.40	0.021	143.6
MPI [37]	0.35	0.019	65.0
MPI-Iter	0.47	0.020	201.2
3D Photos [32]	**0.30**	0.020	123.6
Ours (no-repeat)	**0.30**	0.022	95.4
Ours	0.32	0.020	**50.6**

Table 1. **Quantitative evaluation.** For LPIPS and MSE we compute the error over ten frames of ground truth. We report FID-50 computed over all 50 frames generated from an input test images. See Section 6.1.

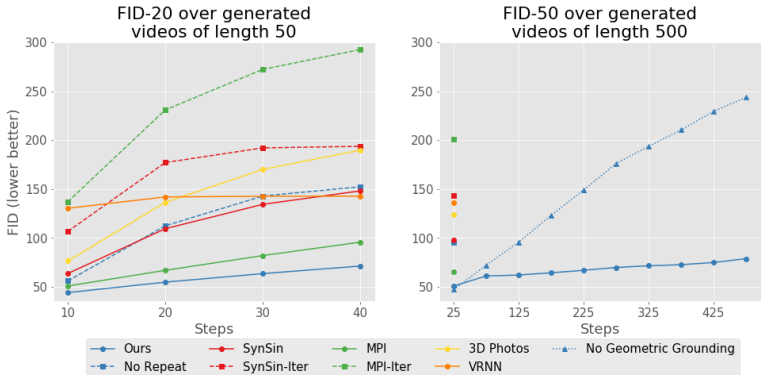

Figure 5. **FID over time.** Left: FID-20 over time for 50 frames generated by each method. Right: FID-50 over *500* frames generated by our method using autopilot. For comparison, we plot FID-50 for the baselines on the first 50 steps. Despite generating sequences an order of magnitude longer, our FID-50 is still lower than that of the baselines. See Sections 6.1, 6.2.

a FID value computed over all image outputs within a temporal window of width w centered at time t, i.e. $\{I_i\}$ for $t - w/2 < i \leq t + w/2$. For short-range generations where ground truth images are available, we report mean squared error (MSE) and LPIPS [48], a perceptual similarity metric that correlates better with human perceptual judgments than traditional metrics such as PSNR and SSIM.

Implementation Details. We train our model with $T = 5$ steps of render-refine-repeat at an image resolution of 160 $\times$ 256 (as most aerial videos have a 16:9 aspect ratio). The choice of T is limited by both memory and available training sequence lengths. The refinement network architecture is the same as that of SPADE generator in [27], and we also employ the same multi-scale discriminator. We implement our models in TensorFlow, and train with a batch size of 4 over 10 GPUs for 7M iterations, which takes about 8 days. We then identify the model checkpoint with the best FID score over a validation set.

6. Evaluation

We compare our approach with three recent state-of-the-art single-image view synthesis methods—the 3D Photography method of Shih *et al*. (henceforward '3D Photos') [32], SynSin [44], and single-view MPIs [37]—as well as the SVG-LP video synthesis method [10]. We retrain each method on our ACID training data, with the exception of 3D Photos which is trained on in-the-wild imagery and, like our method, takes MiDaS disparity as an input. SynSin and single-view MPI were trained at a resolution of 256 $\times$ 256. SVG-LP takes two input frames for context, and operates at a lower resolution of 128 $\times$ 128.

The view synthesis methods were not designed for long camera trajectories, so we also consider iterative variants (SynSin-Iter, MPI-Iter) of these methods in which instead of synthesizing all output views from the initial input image, the next viewpoint is produced by using the previously generated output as the new input at test time. We omit this for 3D photos as iterating is unfortunately prohibitively slow.

6.1. Short-to-medium range view synthesis

To evaluate short-to-medium-range synthesis, we select sequences from our ACID test data with an input frame and 10 subsequent ground truth frames (subsampling as described in Section 5), with the camera moving forwards at an angle of up to 45°. Although our method is trained on all types of camera motions, this forward motion is appropriate for comparison with view synthesis methods which are not designed to handle extreme camera movements.

We then extrapolate the camera motion from the last two frames of each sequence to extend the trajectory for an additional 40 frames. To avoid the camera colliding with the scene, we check the final camera position against the disparity map of the last ground-truth frame, and discard sequences in which the final pose is outside the image or at a depth large enough to be occluded by the scene.

This yields a set of 279 sequences with camera trajectories of 50 steps and ground truth images for the first 10 steps. For short-range evaluation, we compare to ground truth on the first 10 steps. For medium-range evaluation, we compute FID scores over all 50 frames.

We apply each method to these sequences to generate novel views corresponding to the camera poses in each sequence (SVG-LP is the exception in that it does not take account of camera pose.) See results in Table 1. While our goal is perpetual view generation, we find that our approach is competitive with recent view synthesis approaches for short-range synthesis on LPIPS and MSE metrics. For mid-range evaluation, we report FID-50 over all generated 50 frames. Our approach has a dramatically lower FID-50 score

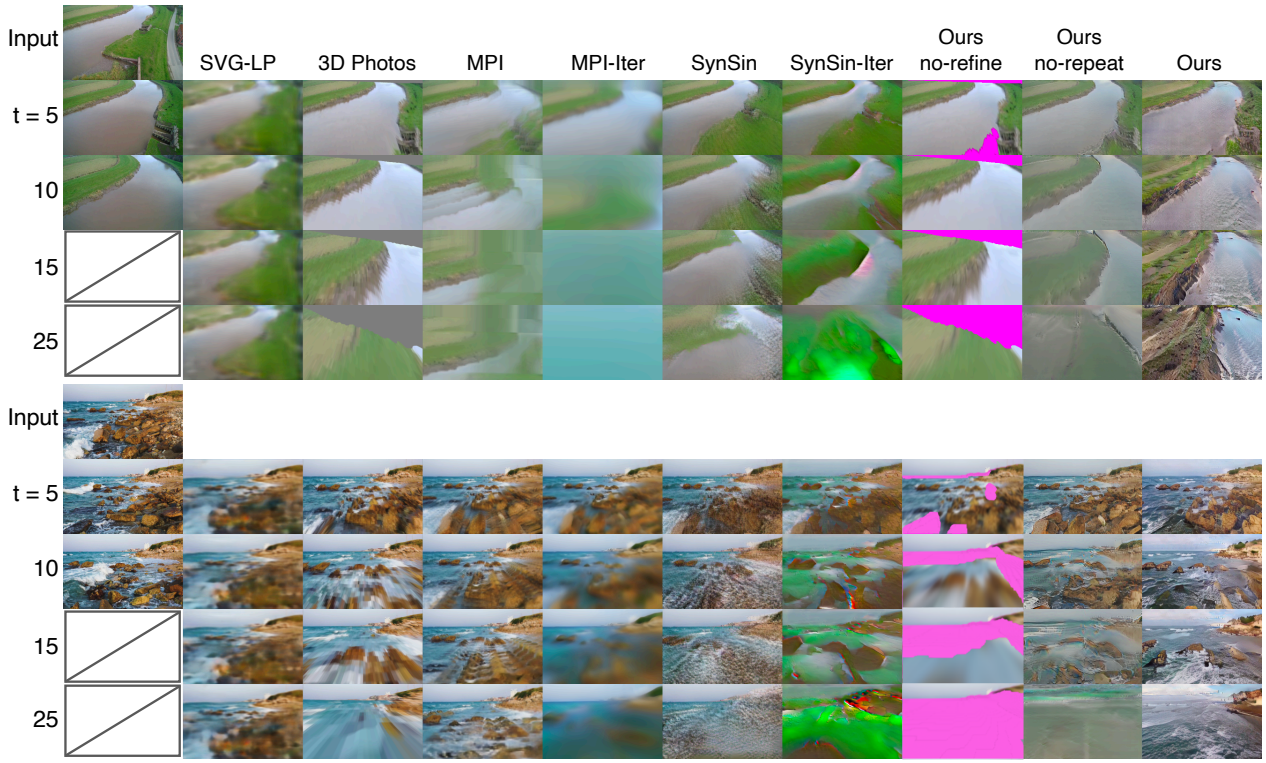

Figure 6. **Qualitative comparison over time.** We show generated sequence results for each method at different time steps. Note that we only have ground truth images for 10 frames; the subsequent frames are generated using an extrapolated trajectory. Pink region in Ours no-refine indicate missing content uncovered by the moving camera.

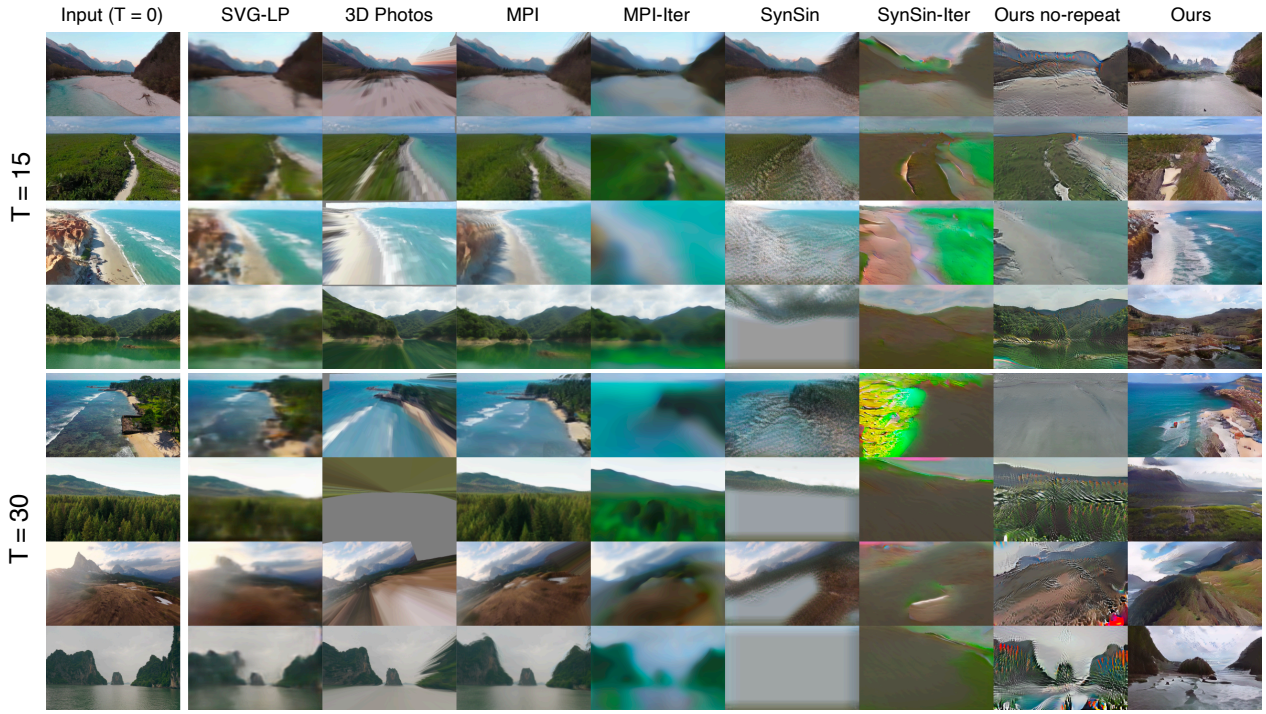

Figure 7. **Qualitative comparison.** We show the diversity and quality of many generated results for each method on the t=15 and 30 frame generation. Competing approaches result in missing or unrealistic frames. Our approach is able to generate plausible views of the scene.

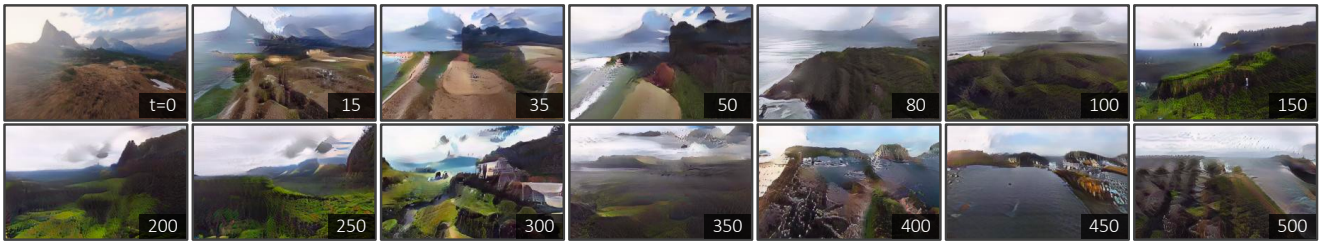

Figure 8. **Long trajectory generation**. From a single image, our approach can generate 500 frames of video without suffering visually. Please see the supplementary video for the full effect.

than other methods, reflecting the more naturalistic look of its output. To quantify the degradation of each method over time, we report a sliding window FID-20 computed from $t = 10$ to 40. As shown in Fig. 5 (left), the image quality (measured by FID-20) of the baseline methods deteriorates quicker with increasing t compared to our approach.

Qualitative comparisons of these methods are shown in Fig. 6 and our supplementary video, which illustrates how the quality of each method's output changes over time. Notable here are SVG-LP's blurriness and inability to predict any camera motion at all; the increasingly stretched textures of 3D Photos' output; and the way the MPI-based method's individual layers become noticeable. SynSin does the best job of generating plausible texture, but still produces holes after a while and does not add new detail. Neither SynSin nor MPI benefits from being applied iteratively. These results are not surprising, but illustrate that none of these systems were designed for more than short-range range synthesis. Fig. 7 shows additional qualitative output of generating 15 and 30 frames for each method on a variety of inputs.

Ablations. We investigate the benefit of training over multiple iterations of our *render-refine-repeat* loop by also training a *'No Repeat'* variant of our model with $T = 1$. That is, at training time this variant predicts only one frame ahead instead of the 5 predicted in our full model. At inference time, the performance on short-range generation, as measured in LPIPS and MSE, is similar to our full model. But when we look at FID, we observe that this method generates lower quality images (Table 1), and that they get substantially worse with increasing t (Fig. 5). This shows the importance of auto-regressive training to our method.

We next illustrate the contribution of our *refine* step, via a comparison with a version of our method which omits this step completely, shown as *'Ours no-refine'* in Fig. 6. For clarity, in this figure we set masked pixels to pink at each step. Note that with increasing t, a larger and larger portion of the image consists of such pixels. In the full model, this region would have been inpainted or outpainted by our refinement network in one of its preceding steps. Note also that even non-masked areas of the image are much blurrier when the refinement step is omitted. This shows the need of our refinement network in super-resolving image content.

6.2. Perpetual view generation

We also evaluate the ability of our model to perform perpetual view generation by synthesizing videos of 500 frames, using an *auto-pilot* algorithm to create an online camera trajectory that avoids flying directly into the ground, sky or obstacles such as mountains. This algorithm works iteratively in tandem with image generation to control the camera based on heuristics which measure the proportion of sky and of foreground obstacles in the scene. See the supplementary for details. We generate 500 frames for each of our test sequences and compute their FID-50 over time, as shown in Fig. 5. Our performance on this metric is robust: even after 500 frames, the FID is lower than that of all the baseline methods over 50 frames. Fig. 5 also shows the benefit of our proposed Geometric Grounding—when it's omitted, the image quality gradually deteriorates, indicating that drift is an important issue to resolve.

Fig. 8 shows a qualititative example of long sequence generation. Despite the challenging problem, our approach retains some aesthetic look of a coastline, generating new islands, rocks, beaches, and waves as it flies through the world. The auto-pilot algorithm can receive additional inputs (such as a user-specified trajectory or random elements), allowing us to generate diverse videos from a single image. Please see the supplementary video for more examples and the full effect of these generated fly-through videos.

7. Discussion

We introduce a new problem of perpetual view generation and present a novel framework that combines both geometric and generative techniques to tackle it. Our system can generate video sequences covering over hundreds of steps, which to our knowledge has not been shown for prior video or view synthesis methods. Our results indicate that our hybrid approach is a promising step—nevertheless, there remain many exciting challenges in this domain. First, the refinement network, like other GANs, can produce images that seem realistic but not recognizable [16]. Further advancement in image and video synthesis generation methods that incorporate geometry would be an interesting future direction. Second, we have modeled scene generation as a Markov process. While local temporal consistency is pro-

vided by the rendering process, the memory-less property leads to a system that does not have global consistency. After some steps, a field may gradually turn into an ocean as we fly over it. Incorporating memory in this system would be an exciting direction. Lastly, we do not model dynamic scenes (although our model does a reasonable job in generating plausible-looking ocean waves): combining our geometry-aware approach with methods that can reason about object dynamics is an exciting direction for future work.

References

[1] Connelly Barnes, Eli Shechtman, Adam Finkelstein, and Dan B Goldman. PatchMatch: A randomized correspondence algorithm for structural image editing. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 28(3), Aug. 2009. 2

[2] Moshe Blank, Lena Gorelick, Eli Shechtman, Michal Irani, and Ronen Basri. Actions as space-time shapes. In *ICCV*, pages 1395–1402. IEEE, 2005. 3

[3] Gaurav Chaurasia, Sylvain Duchêne, Olga Sorkine-Hornung, and George Drettakis. Depth synthesis and local warps for plausible image-based navigation. *Trans. on Graphics*, 32:30:1–30:12, 2013. 2

[4] Weifeng Chen, Shengyi Qian, and Jia Deng. Learning single-image depth from videos using quality assessment networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 2

[5] Xu Chen, Jie Song, and Otmar Hilliges. Monocular neural image based rendering with continuous view control. In *ICCV*, 2019. 2

[6] Xu Chen, Jie Song, and Otmar Hilliges. Monocular neural image based rendering with continuous view control. In *ICCV*, pages 4090–4100, 2019. 2

[7] Inchang Choi, Orazio Gallo, Alejandro Troccoli, Min H Kim, and Jan Kautz. Extreme view synthesis. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 7781–7790, 2019. 2

[8] Inchang Choi, Orazio Gallo, Alejandro Troccoli, Min H Kim, and Jan Kautz. Extreme view synthesis. In *ICCV*, pages 7781–7790, 2019. 2

[9] Aidan Clark, Jeff Donahue, and Karen Simonyan. Efficient video generation on complex datasets. *arXiv preprint arXiv:1907.06571*, 2019. 2, 3

[10] Emily Denton and Rob Fergus. Stochastic video generation with a learned prior. *arXiv preprint arXiv:1802.07687*, 2018. 2, 3, 6

[11] Chelsea Finn, Ian Goodfellow, and Sergey Levine. Unsupervised learning for physical interaction through video prediction. In *NeurIPS*, pages 64–72, 2016. 3

[12] John Flynn, Michael Broxton, Paul Debevec, Matthew DuVall, Graham Fyffe, Ryan Overbeck, Noah Snavely, and Richard Tucker. Deepview: View synthesis with learned gradient descent. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 2

[13] Kyle Genova, Forrester Cole, Aaron Maschinot, Aaron Sarna, Daniel Vlasic, and William T. Freeman. Unsupervised training for 3d morphable model regression. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 3

[14] Daniel Glasner, Shai Bagon, and Michal Irani. Super-resolution from a single image. In *ICCV*, pages 349–356, 2009. 2

[15] James Hays and Alexei A Efros. Scene completion using millions of photographs. *ACM Transactions on Graphics (TOG)*, 26(3):4–es, 2007. 2

[16] Aaron Hertzmann. Visual indeterminacy in generative neural art. *arXiv preprint arXiv:1910.04639*, 2019. 8

[17] 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

[18] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European conference on computer vision*, pages 694–711. Springer, 2016. 5

[19] Biliana Kaneva, Josef Sivic, Antonio Torralba, Shai Avidan, and William T. Freeman. Infinite images: Creating and exploring a large photorealistic virtual space. In *Proceedings of the IEEE*, 2010. 2

[20] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4401–4410, 2019. 2

[21] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. 5

[22] Katrin Lasinger, René Ranftl, Konrad Schindler, and Vladlen Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *arXiv preprint arXiv:1907.01341*, 2019. 2, 5

[23] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew P Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In *CVPR*, 2017. 2

[24] Marc Levoy and Pat Hanrahan. Light field rendering. In *Proceedings of SIGGRAPH 96*, Annual Conference Series, 1996. 2

[25] Ben Mildenhall, Pratul P. Srinivasan, Rodrigo Ortiz-Cayon, Nima Khademi Kalantari, Ravi Ramamoorthi, Ren Ng, and Abhishek Kar. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. *ACM Transactions on Graphics (TOG)*, 2019. 2

[26] Simon Niklaus, Long Mai, Jimei Yang, and Feng Liu. 3D Ken Burns effect from a single image. *ACM Transactions on Graphics (TOG)*, 2019. 2

[27] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019. 4, 5, 6

[28] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635, 2011. 4

[29] Arno Schödl, Richard Szeliski, David H Salesin, and Irfan Essa. Video textures. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 489–498, 2000. 2

[30] Tamar Rott Shaham, Tali Dekel, and Tomer Michaeli. Singan: Learning a generative model from a single natural image. In *ICCV*, pages 4570–4580, 2019. 2, 5

[31] Lixin Shi, Haitham Hassanieh, Abe Davis, Dina Katabi, and Fredo Durand. Light field reconstruction using sparsity in the continuous fourier domain. *Trans. on Graphics*, 34(1):12:1–12:13, Dec. 2014. 2

[32] Meng-Li Shih, Shih-Yang Su, Johannes Kopf, and Jia-Bin Huang. 3d photography using context-aware layered depth inpainting. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2, 6

[33] Assaf Shocher, Shai Bagon, Phillip Isola, and Michal Irani. Ingan: Capturing and remapping the" dna" of a natural image. *arXiv preprint arXiv:1812.00231*, 2018. 2, 5

[34] Assaf Shocher, Nadav Cohen, and Michal Irani. "zero-shot" super-resolution using deep internal learning. In *CVPR*, pages 3118–3126, 2018. 2

[35] Pratul P. Srinivasan, Richard Tucker, Jonathan T. Barron, Ravi Ramamoorthi, Ren Ng, and Noah Snavely. Pushing the boundaries of view extrapolation with multiplane images. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 2

[36] Piotr Teterwak, Aaron Sarna, Dilip Krishnan, Aaron Maschinot, David Belanger, Ce Liu, and William T Freeman. Boundless: Generative adversarial networks for image extension. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 10521–10530, 2019. 2

[37] Richard Tucker and Noah Snavely. Single-view view synthesis with multiplane images. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 2, 3, 6

[38] Shubham Tulsiani, Richard Tucker, and Noah Snavely. Layer-structured 3D scene inference via view synthesis. In *The European Conference on Computer Vision (ECCV)*, September 2018. 2

[39] Sergey Tulyakov, Ming-Yu Liu, Xiaodong Yang, and Jan Kautz. Mocogan: Decomposing motion and content for video generation. In *CVPR*, pages 1526–1535, 2018. 3

[40] Ruben Villegas, Arkanath Pathak, Harini Kannan, Dumitru Erhan, Quoc V Le, and Honglak Lee. High fidelity video prediction with large stochastic recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 81–91, 2019. 2, 3

[41] Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. Generating videos with scene dynamics. In *NeurIPS*, pages 613–621, 2016. 3

[42] Carl Vondrick and Antonio Torralba. Generating the future with adversarial transformers. In *CVPR*, pages 1020–1028, 2017. 3

[43] Yi Wang, Xin Tao, Xiaoyong Shen, and Jiaya Jia. Wide-context semantic image extrapolation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1399–1408, 2019. 2

[44] Olivia Wiles, Georgia Gkioxari, Richard Szeliski, and Justin Johnson. SynSin: End-to-end view synthesis from a single image. In *CVPR*, 2020. 2, 3, 6

[45] Zongxin Yang, Jian Dong, Ping Liu, Yi Yang, and Shuicheng Yan. Very long natural scenery image prediction by outpainting. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 10561–10570, 2019. 2

[46] Yufei Ye, Maneesh Singh, Abhinav Gupta, and Shubham Tulsiani. Compositional video prediction. In *ICCV*, 2019. 3

[47] Jiahui Yu, Zhe Lin, Jimei Yang, Xiaohui Shen, Xin Lu, and Thomas S Huang. Free-form image inpainting with gated convolution. In *ICCV*, 2019. 2

[48] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018. 6

[49] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: Learning view synthesis using multiplane images. *ACM Trans. Graph.*, 37(4):65:1–65:12, 2018. 2, 3, 5

[50] Yang Zhou, Zhen Zhu, Xiang Bai, Dani Lischinski, Daniel Cohen-Or, and Hui Huang. Non-stationary texture synthesis by adversarial expansion. *arXiv preprint arXiv:1805.04487*, 2018. 2

arXiv:2012.09855v2 [cs.CV] 18 Dec 2020

Supplementary Material for Infinite Nature: Perpetual View Generation of Natural Scenes from a Single Image

Andrew Liu* Richard Tucker* Varun Jampani
Ameesh Makadia Noah Snavely Angjoo Kanazawa
Google Research

1. Implementation Details

1.1. ACID Collection and Processing

We started by identifying over 150 proper nouns of coastline and island locations such as *Big Sur*, *Half Moon Bay*, *Moloka'i*, *Shi Shi Beach*, *Waimea bay* etc. Then, we combine each proper noun with a set of keywords $\{aerial, drone, dji, mavic\}$ and use the combinations to make YouTube queries.

We take the top 10 video ids for each query as the candidate videos for our dataset. We process all the videos through a SLAM and SfM pipeline as in Zhou *et al*. [1]. This returns the the camera poses of the input video trajectory and 3D keypoints. We manually identify and remove videos that are not aerial, have static camera, and those that have scenes with too many people or man-made structures. In an effort to limit the potential privacy concerns of our work, we run the state of the art object detection network [2] to identify any humans present in the frames. If detected humans occupy more than 10% of a given frame, we discard the frame. The above filtering steps are applied to identify high quality video sequences for training with limited privacy implications and form the basis of our dataset.

Many videos, especially drone footage, are shot with cinematic horizontal borders like letterboxes. We pre-process every frame to remove detected letterboxes and appropriately adjust the camera intrinsics to reflect this crop operation.

From the remaining set of sequences, we run the MiDaS system [3] on every frame to get dense disparity (inverse depth). MiDaS predicts disparity only up to an unknown scale and shift, so we use the 3D keypoints produced by running SfM to compute scale and shift parameters for each frame that best fit the MiDaS disparity values to the 3D keypoints visible in that frame, so that the disparity images align with the SfM camera trajectories during training. More specifically, the scale a and shift b are calculated via least-squares as:

$$\underset{a,b}{\operatorname{argmin}} \sum_{(x,y,z)\in\mathcal{K}} \left(a\hat{D}_{xyz} + b - z^{-1}\right)^2 \tag{1}$$

where $\mathcal{K}$ is the set of visible 3D keypoints from the local frame's camera viewpoint, $\hat{D}$ is the disparity map predicted by MiDaS on the given frame, and $\hat{D}_{xyz}$ is the disparity value sampled from that map at texture coordinates corresponding to the projection of the point (x, y, z) with the camera intrinsics. The disparity map D we use during training and rendering is then $D = a\hat{D} + b$.

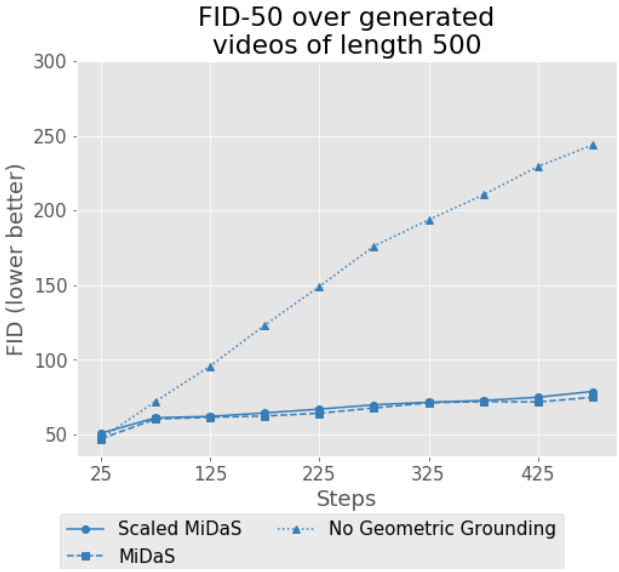

Figure 1. **Scaled MiDaS vs MiDaS.** We scale the MiDaS disparity maps to be consistent with the camera poses estimated by SfM during training. At test-time our approach only requires a single image with disparity. Here we show results of FID-50 long generation using the original MiDaS output vs the scaled-and-shifted MiDaS. Despite being only trained on scaled disparity, our model still performs competitively with the unscaled MiDaS output.

*Equal Contribution

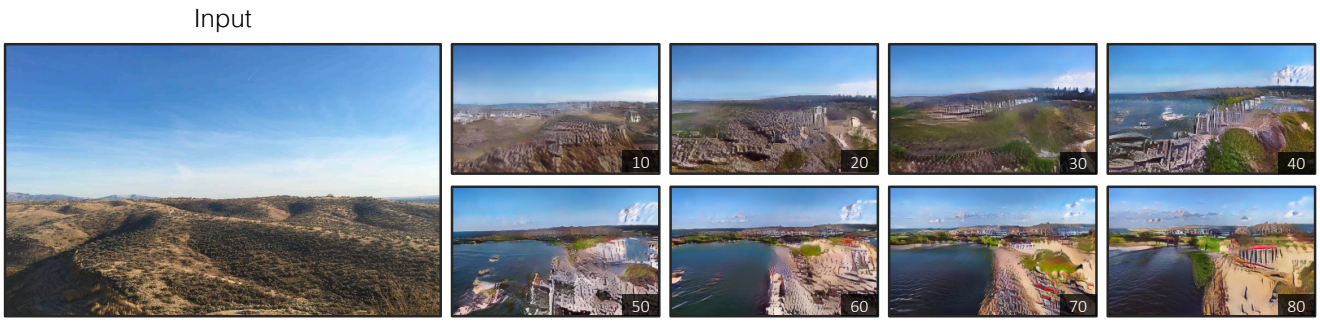

Figure 2. **Generation from smartphone photo.** Our perpetual view generation applied to a photo captured by the authors on a smartphone. We use MiDaS for the initial disparity, and assume a field of view of 90°.

1.2. Inference without Disparity Scaling

Scaling and shifting the disparity as described above requires a sparse point cloud, which is generated from SfM and in turn requires video or multi-view imagery. At test-time, however, scaling and shifting the disparity is only necessary if we seek to compare generated frames at target poses against ground truth. Just to generate sequences, we can equally well use the original MiDaS disparity predictions. Fig. 1 compares long generation using scaled and original MiDaS outputs, and shows that there is negligible effect on the FID scores. Fig. 2 gives an example of a long sequence generated in this way from a photo taken on a smartphone, demonstrating that our framework runs well on a single test image using original MiDaS disparity.

1.3. Data Source for Qualitative Illustrations

Note that for license reasons, we do not show generated qualitative figures and results on ACID. Instead, we collect input images with open source licenses from [4] and show the corresponding qualitative results in the paper and the supplemental video. The quantitative results are computed on ACID test set.

1.4. Auto-pilot View Control

We use an auto-pilot view control algorithm when generating long sequences from a single input RGB-D image. This algorithm must generate the camera trajectory in tandem with the image generation, so that it can avoid crashing into the ground or obstacles in the scene. Our basic approach works as follows: at each step we take the current disparity image and categorize all points with disparity below a certain threshold as *sky* and all points with disparity above a second, higher threshold as *near*. (In our experiments these thresholds are set to 0.05 and 0.5.) Then we apply three simple heuristics for view-control: (1) look up or down so that a given percentage (typically 30%) of the image is *sky*, (2) look left or right, towards whichever side has more *sky*, (3) If more than 20% of the image is *near*, move up (and if less, down), otherwise move towards a horizontally-centered point 30% of the way from the top of the image. These heuristics determine a (camera-relative) target look direction and target movement direction. To ensure smooth camera movement, we interpolate the actual look and movement directions only a small fraction (0.05) of the way to the target directions at each frame. The next camera pose is then produced by moving a set distance in the move direction while looking in the look direction. To generate a wider variety of camera trajectories, we can add an offset to the target look direction that varies over time: a horizontal sinusoidal variation in the look direction, for example, generates a meandering trajectory.

This approach generates somewhat reasonable trajectories, but an exciting future direction would be to train a model that learns how to choose each successive camera pose, using the camera poses in our training data.

1.5. Additional Frame Interpolation

For the purposes of presenting a very smooth and cinematic video with a high frame rate, we can additionally interpolate between frames generated by our model. Since our system produces not just RGB images but also disparity, and since we have camera poses for each frame, we can use this information to aid the interpolation. For each pair of frames (P_t, I_t, D_t) and $(P_{t+1}, I_{t+1}, D_{t+1})$ we proceed as follows:

First, we create additional camera poses (as many as desired) by linearly interpolating position and look-direction between P_t and P_{t+1}. Then, for each new pose P a fraction λ of the way between P_t and P_{t+1}, we use the differentiable renderer $\mathcal{R}$ to rerender I_t and I_{t+1} from that viewpoint, and blend between the two resulting images:

$$\begin{aligned} I'_t &= \mathcal{R}(I_t, D_t, P_t, P), \\ I'_{t+1} &= \mathcal{R}(I_{t+1}, D_{t+1}, P_{t+1}, P), \\ I &= (1-\lambda) I'_t + \lambda I'_{t+1}, \end{aligned} \tag{2}$$

Note: we apply this interpolation to the long trajectory sequences in the supplementary video only, adding four new frames between each pair in the sequence. However, all

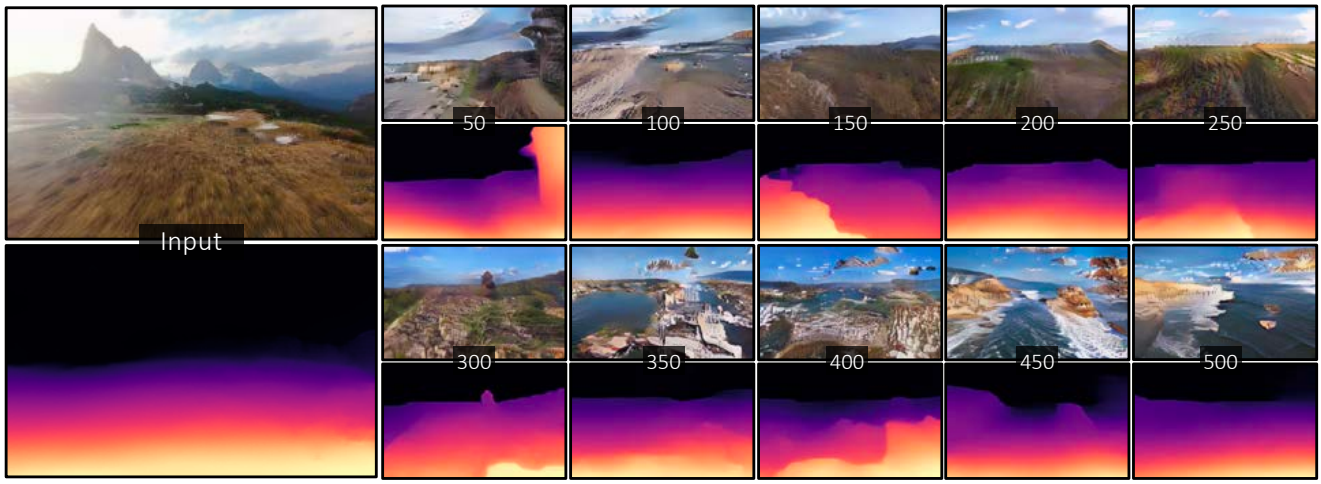

Figure 3. **Long Generation with Disparity.** We show generation of a long sequence with its corresponding disparity output. Our render-refine-repeat approach enables refinement of both geometry and RGB textures.

short-to-mid range comparisons and all figures and metrics in the paper are computed on raw outputs without any interpolation.

1.6. SynSin training

We first trained Synsin [5] on our nature dataset with the default training settings (i.e. the presets used for the KITTI model). We then modified the default settings by changing the camera stride in order to train the Synsin to perform better for the task of longer-range view synthesis. Specifically, we employ the same motion-based sampling for selecting pairs of images as described in the main paper at Section 4 (Aligning Camera Speed). However, here we increase the upper end of the desired motion range by a factor of 5, which allow the network to train with longer camera strides. This obtains a better performance than the default setting, and we use this model for all Synsin evaluations. We found no improvement going beyond $5X$ camera motion range. We also implemented an exhaustive search for desirable image pairs within a sequence to maximize the training data. We also experimented with synthesizing long videos by applying SynSin in an auto-regressive fashion at test time. But this performed worse than the direct long-range synthesis.

2. Additional Analysis of Results

2.1. Limitations

As discussed in the main paper, our approach is essentially a memory-less Markov process that does not guarantee global consistency across multiple iterations. This manifests in two ways: First on the geometry, *i.e.* when you look back, there is no guarantee that the same geometric structure that was observed in the past will be there. Second, there is also no global consistency enforced on the appearance——the appearance of the scene may change in short range, such as sunny coastline turning into a cloudy coastline after several iterations. Similarly, after hundreds of steps, two different input images may end up in a scene that has similar stylistic appearance, although never exactly the same set of frames. Adding global memory to a system like ours and ensuring more control over what will happen in the long range synthesis is an exciting future direction.

2.2. Disparity Map

In addition to showing the RGB texture, we can also visualize the refined disparity to show the geometry. In Fig. 3, we show the long generation as well as its visualized disparity map. Note that the disparity maps look plausible as well because we train our discriminator over RGB and disparity concatenated. Please also see our results in the supplementary videeo.

2.3. Effect of Disabling Geometric Grounding

We use geometric grounding as a technique to avoid drift. In particular we found that without this grounding, over a time period of many frames the render-refine-repeat loop gradually pushes disparity to very small (i.e. distant) values. Fig. 4 shows an example of this drifting disparity: the sequence begins plausibly but before frame 150 is reached, the disparity (here shown unnormalized) has become very small. It is notable that once this happens the RGB images then begin to deteriorate, drifting further away from the space of plausible scenes. Note that this is a test-time difference only: the results in Fig. 4 were generated using the same model checkpoint as our other results, but with geometric grounding disabled at test time.

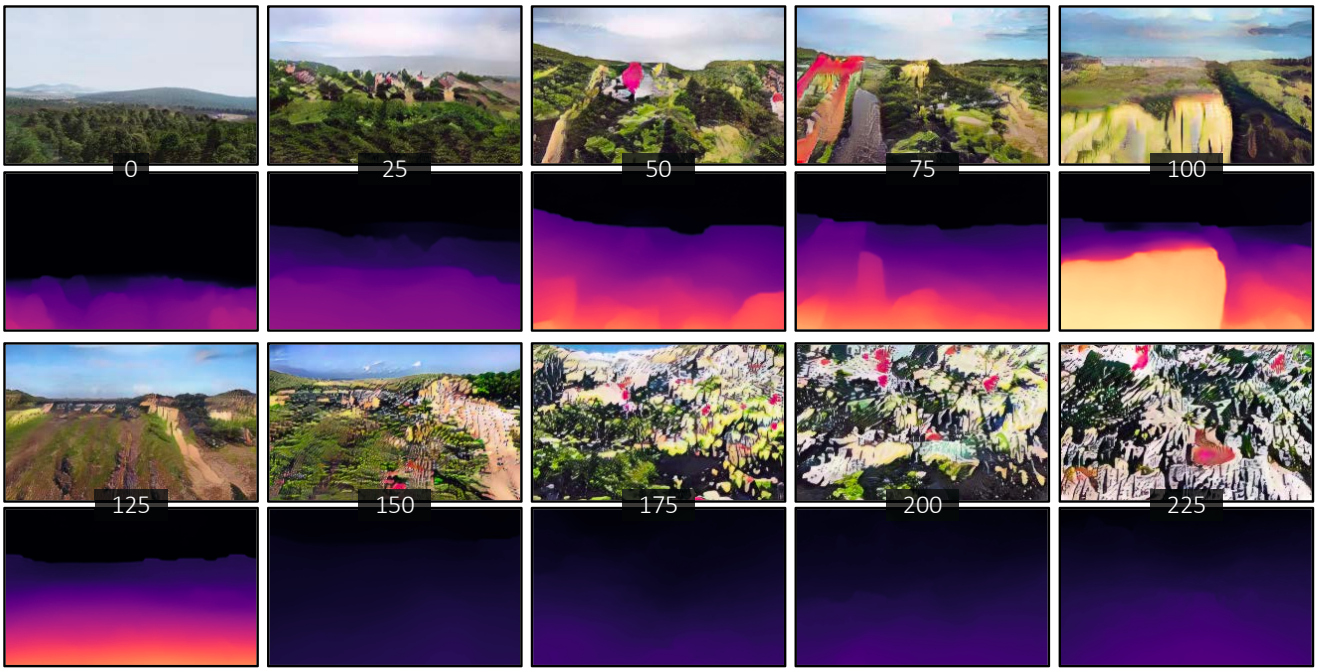

Figure 4. **Geometric Grounding Ablation.** We show our pretrained checkpoint without using the geometric ground on the task of long generation. The disparity maps are visualized using an *unnormalized* color scale. Note that by the 150th frame, the disparity map has drifted very far away. Subsequently the RGB frame drifts after the 175th frame. However prior to drifting, the network produces plausible video sequences.

References

[1] T. Zhou, R. Tucker, J. Flynn, G. Fyffe, and N. Snavely, "Stereo magnification: Learning view synthesis using multiplane images," *ACM Trans. Graph.*, vol. 37, no. 4, pp. 65:1–65:12, 2018.

[2] M. Tan, R. Pang, and Q. V. Le, "Efficientdet: Scalable and efficient object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10781–10790, 2020.

[3] K. Lasinger, R. Ranftl, K. Schindler, and V. Koltun, "Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer," *arXiv preprint arXiv:1907.01341*, 2019.

[4] *Pexels.* Pexels provides high quality and completely free stock photos licensed under the Creative Commons Zero (CC0) license. All photos are tagged, searchable and easy to discover .

[5] O. Wiles, G. Gkioxari, R. Szeliski, and J. Johnson, "SynSin: End-to-end view synthesis from a single image," in *CVPR*, 2020.