Video Autoencoder: self-supervised disentanglement of static 3D structure and motion

Zihang Lai
Carnegie Mellon University

Sifei Liu
NVIDIA

Alexei A. Efros
UC Berkeley

Xiaolong Wang
UC San Diego

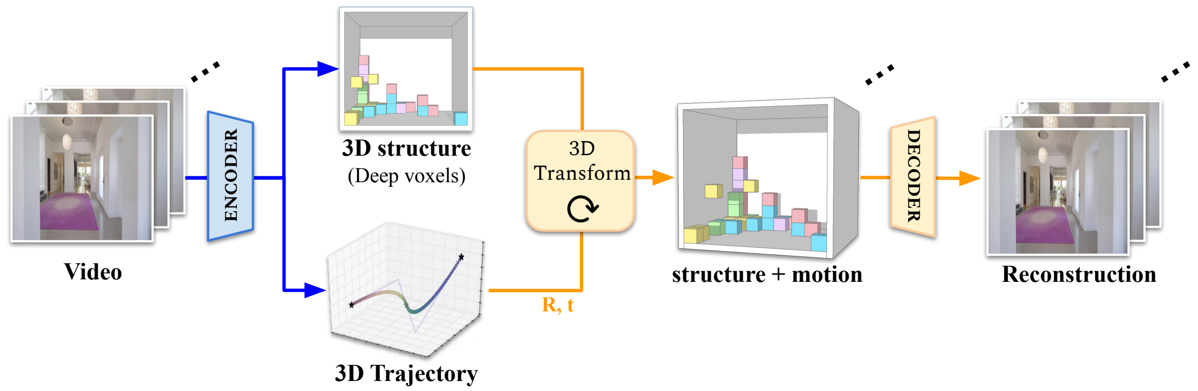

Figure 1: **Video Autoencoder:** raw input video is automatically disentangled into 3D scene structure and camera trajectory. To reconstruct the original video, the camera transformation is applied to the 3D structure feature and then decoded back to pixels. Without any fine-tuning, the model generalizes to unseen videos and enables tasks such as novel view synthesis, pose estimation, and "video following".

Abstract

A video autoencoder is proposed for learning disentangled representations of 3D structure and camera pose from videos in a self-supervised manner. Relying on temporal continuity in videos, our work assumes that the 3D scene structure in nearby video frames remains static. Given a sequence of video frames as input, the video autoencoder extracts a disentangled representation of the scene including: (i) a temporally-consistent deep voxel feature to represent the 3D structure and (ii) a 3D trajectory of camera pose for each frame. These two representations will then be re-entangled for rendering the input video frames. This video autoencoder can be trained directly using a pixel reconstruction loss, without any ground truth 3D or camera pose annotations. The disentangled representation can be applied to a range of tasks, including novel view synthesis, camera pose estimation, and video generation by motion following. We evaluate our method on several large-scale natural video datasets, and show generalization results on out-of-domain images. Project page with code: https://zlai0.github.io/VideoAutoencoder.

1. Introduction

The visual world arrives at a human eye as a streaming, entangled mess of colors and patterns. The art of seeing, to a large extent, is in our ability to disentangle this mess into physically and geometrically coherent factors: persistent solid structures, illumination, texture, movement, change of viewpoint, etc. From its very beginnings, computer vision has been concerned with acquiring this impressive human ability, including such classics as Barrow and Tenebaum's Intrinsic Image decomposition [5] in the 1970s, or Tomasi-Kanade factorization [64] in the 1990s. In the modern deep era, learning a disentangled visual representation has been a hot topic of research, often taking the form of an autoencoder [40, 24, 37, 20, 42, 49, 23, 3, 51]. However, almost all prior work has focused on disentanglement within the 2D image plane using datasets of still images.

In this work, we propose a method that learns a disentangled 3D scene representation, separating the static 3D scene structure from the camera motion. Importantly, we employ videos as training data (as opposed to a dataset of stills), using the temporal continuity within a video as a source of training signal for self-supervised disentanglement. We make the assumption that a local snippet of video is capturing a static scene, so the changes in appearance must be due to camera motion. This leads to our *Video Autoencoder* formulation, shown on Figure 1: an input video is encoded into two codes, one for 3D scene structure (which is forced to remain fixed cross frames) and the other for the camera trajectory (updated for every frame). The 3D structure is represented by 3D deep voxels (similar to [47, 58]) and the camera pose with a 6-dimension rotation and translation vector. To reconstruct the original video, we simply apply the camera transformation to the 3D structure features and then decode back to pixels.

A key advantage of our framework is that it provides 3D representations readily integrated modern neural rendering methods, which typically requires 3D and/or camera pose ground truth annotation at training time [14, 48, 4, 72, 63]. This usually implies a 2-stage process with running Structure-from-Motion (SfM) as precursor to training. In our work, we are working towards a way of training on completely unstructured datasets, removing the need for running SfM as preprocessing.

At test time, the features obtained using our Video Autoencoder can be used for several downstream tasks, including novel view synthesis (Section 4.3), pose estimation in video (Section 4.2), and video following (Section 4.4). For novel view synthesis, given a single input image, we first encode it as a 3D scene feature, and then render to a novel view by providing a new camera pose. We show results on large-scale video datasets including RealEstate10K [80], Matterport3D [6], and Replica [61]. Our method not only achieves better view synthesis results than state-of-the-art view synthesis approach [72] that requires stronger camera supervision on RealEstate10K, but also generalizes better when applied to out-of-domain data. As another application, we show that our method could be used to implicitly factorize structure from motion in novel videos, by evaluating the estimated camera pose against SfM baseline. Finally, we show that by swapping the 3D structure and camera trajectory codes between a pair of videos, we can achieve Video Following, where a scene from one video is "following" the motion from the other video.

2. Related Work

Learning disentangled representations. Disentangled representations learned from unlabeled data not only provide a better understanding of the data, but also produce more generalizable features for different downstream applications. Popular ways to learn such representations include generative models (e.g. GANs) [8, 70, 82, 28, 36, 47, 57, 50, 41] and autoencoders [40, 24, 37, 20, 42, 49, 23, 3, 51]. For example, Kulkarni et al. [40] proposed to learn disentangled representations of pose, light, and shape for human faces using a Variational Autoencoder (VAE) [39]. However, almost all these works are modeling still 2D images, inherently limiting the data available for disentanglement. In order to learn disentangled representations related to motion and dynamics, researchers have been looking at video data [9, 75, 30, 73, 67, 26, 44, 74]. For example, Denton et al. [9] proposed to learn the disentangled representation which factorizes each video frame into a stationary component and a temporally varying component. Beyond learning latent features, Tomas et al. [30] designed a video frame reconstruction method for disentangling human pose skeleton from frame appearance. While these results are encouraging, they are not able to capture the 3D structure of generic scenes from video.

Learning 3D representations. 3D representation learning from video or 2D image sets is a long-standing problem in computer vision. Traditional approaches typically rely on multi-view geometry [21] to understand real-world 3D structures. Based on geometric principles, 3D structure and camera motion can be jointly optimized in Structure-from-Motion (SfM) pipelines and has yielded great success in a wide range of domains [1, 56, 60]. To better generalize to diverse environments, learning-based approaches are proposed to learn 3D representations using 2D supervision [33, 79, 66, 35, 72, 27, 53]. For instance, Wiles et al. [72] proposed to utilize the point cloud as an intermediate representation for novel view synthesis. However, it requires camera poses computed from SfM for training, and point cloud estimation can be inaccurate when the test image is out of distribution. Instead of using point clouds, neural 3D representations, including implicit function [43, 59] and the deep voxels [13, 19, 58, 47, 46] have shown impressive reconstruction and synthesis results. Our work is closely related to the approach proposed by Tung et al. [13], which leverages view prediction for learning latent 3D voxel structure of the scene. However, camera pose is still required to provide supervision. Our work is also highly inspired by Nguyen-Phuoc et al. [47], who proposed inserting the voxel representation into Generative Adversarial Networks, enabling the disentanglement of 3D structure, style, and pose. Finally, our work is related to plenty of downstream tasks that leverages a learned 3D deep voxels, such as 3D object detection [13], 3D object tracking [18], 3D motion estimation [19] and few-shot concept learning [52].

Self-supervised learning on video. Our work is related to self-supervised learning of visual representations from video [2, 69, 29, 31, 32, 45, 79, 73, 71, 13, 17]. For example, Wei et al. [71] proposed to learn from the arrow of time and obtain a representation that is sensitive to temporal changes; it can then be used for action recognition. Instead of fine-tuning the learned representation for recognition, our work is more focused on the 3D structure of the representation itself, and we can directly adopt our representation for multiple applications without fine-tuning. In this regard, our work is more related to Zhou et al. [79], who perform joint estimation of image depth and camera pose in a self-supervised manner. However, their approach is restricted to specific domains, such as scenes from self-driving cars, while our model allows generalization to a wider range of real-world videos. Instead of predicting depth, our method uses a voxel representation, which can be applied to downstream tasks, such as novel view synthesis.

3. Approach

The proposed Video Autoencoder is a conceptually simple method for encoding a video into a 3D representation and a trajectory in a completely self-supervised manner (**no 3D labels** are required). Figure 2 shows a schematic layout of our Video Autoencoder. Like other auto-encoders, we *encode* data into a deep representation and *decode* the representation back to reconstruct the original input, relying on the consistency between the input and the reconstruction to learn sub-modules of multiple neural networks. In our case, the goal is encoding a video into two *disentangled* components: a static 3D representation and a dynamic camera trajectory. By assuming that the input video clip shows a static scene that remains unchanged in the video clip, we can construct a single 3D structure (represented by deep voxels) and apply camera transformations on the structure to reconstruct corresponding video frames. Unlike other existing methods [72, 65, 43, 81], our model does not need ground truth camera poses. Instead, we use another network to predict the camera motion, which is then jointly optimized with the 3D structure. By doing so, we find that the 3D motion and structure can automatically emerge from the auto-encoding process.

Training. At training time, the *first frame* of an N-frame video clip (we use $N = 6$) passes through the 3D Encoder (*blue box* in Fig. 2), which computes a voxel grid of deep features representing the 3D scene. At the same time, the trajectory encoder (*red box* in Fig. 2) uses the same video clip to produce a short trajectory of three points. This trajectory estimates the camera pose of each frame w.r.t. the first frame. Next, we *re-entangle* the camera trajectory and the 3D structure and reconstruct the input video clip. First, we use the estimated camera pose to transform the encoded 3D deep voxel. Because we assume that the scene is static, the transformed 3D deep voxel should align with the corresponding frame if both the voxel representation and the camera pose are accurately estimated. We then use a decoder network to render N 2D images from the set of N camera pose transformed 3D voxels. The reconstruction loss encourages the disentanglement between the static 3D scene and the camera pose. We also adopt a consistency loss to enforce the 3D deep voxels extracted from different frames to be the same, which facilitates training.

Inference for View Synthesis. The procedure during test time is similar. First, the 3D Encoder estimates the 3D voxel representation from a single input image. The trajectory can be of arbitrary length and pre-computed. Next, we compute the transformed 3D deep voxels according to the given camera trajectory. These trajectory-guided deep voxels are then fed into the decoder which renders each frame of the video as outputs. While this approach works well for nearby frames where the motion is not too large, we found that voxels can fail to render clear images if the applied transformation is too large. Therefore, when we need to generate long videos at test time, we employ a simple heuristic which *reinitializes* the deep voxels to the current frame every K frames ($K = 12$ in our implementation). Here, the *reinitialize* operation re-encodes the previous prediction into a new 3D voxel to replace the existing 3D voxel.

We describe each individual component of our architecture in the following sections. In section 3.1, we give details of how we obtain the 3D latent voxel representation from a single image. In section 3.2, we describe the method for predicting trajectories for a particular video. In section 3.3, the decoder which *re-entangles* camera motion and 3D structure back into image space is presented. In section 3.4, we discuss the loss function used for training the auto-encoder.

3.1. 3D Encoder

The 3D encoder $\mathcal{F}_{\text{3D}}$ encodes an image input into a 3D deep voxels that represents the same scene as the output,

$$z = \mathcal{F}_{\text{3D}}(I)$$

Figure 3 (a) illustrates the structure of 3D Encoder in detail. Taking an image as input, we first use a 2D encoder (a pretrained ResNet-50 [22] in our implementation) to compute a set of 2D feature maps (*Resnet-50* in Figure 3 (a)). Next, to obtain a 3D representation of the image, we reshape these 2D feature maps into 3D feature grids, which are also referred to as deep voxels. Here, the *reshape* operation is performed on the feature dimension: if the 2D feature maps have dimension $H \times W \times C$, the reshaped tensor is four-dimensional and has size $H \times W \times D \times (C/D)$, where D refers to depth dimension. This ensures that the spatial arrangement is not perturbed, *e.g.* the top right corner of the deep voxels corresponds to the top right corner of the input image. Because the 2D feature extractor repeatedly downsamples the input image, the spatial resolution of the deep voxels is actually small (only $1/16$ of the original image). In order to reconstruct images of high fidelity, we upsample and refine this 3D structure as a final step. This is done by a set of strided 3D deconvolutions (*3D Conv* in Figure 3 (a)).

Note while there are other possible 3D representations such as point cloud and polygon mesh, none of these methods are as easy to work with as voxels, which allows reshaping and could be applied with convolutions easily. We use voxel representation to keep things simple, but other representations could potentially work as well.

3.2. Trajectory Encoder

The trajectory encoder $\mathcal{F}_{\text{traj}}$ estimates trajectory from input videos. Specifically, the encoder computes the camera pose (6-D, rotation and translation) w.r.t. the first frame for each image in a sequence. We call this output sequence of

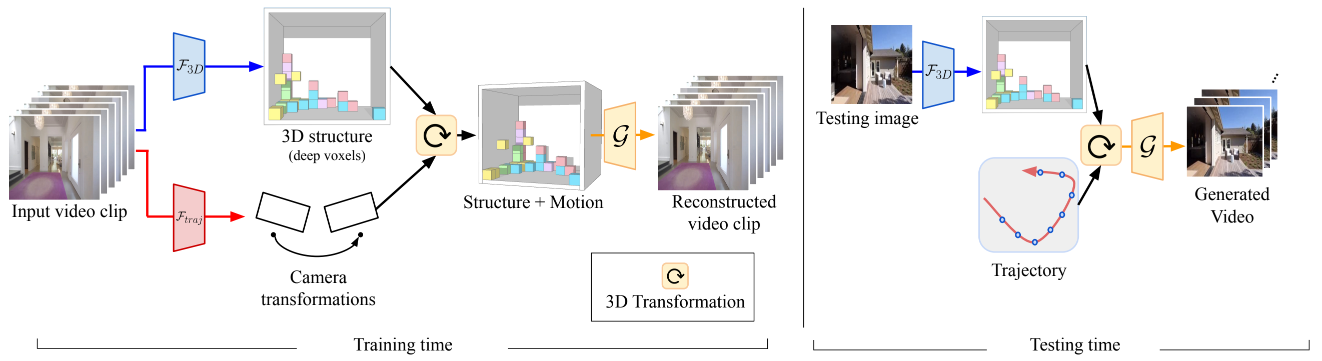

Figure 2: **Training and testing procedure of Video Autoencoder:** During training, we use short clips extracted from videos. The first frame of the clip is used to predict the 3D structure of the scene. Then, the subsequent frames are used to compute poses relative to the first frame. We apply these predicted poses as affine transformations to the 3D voxel and use a decoder to reconstruct the input video clip. Once the autoencoder is trained, we can get the 3D representation for downstream tasks using only one image as input.

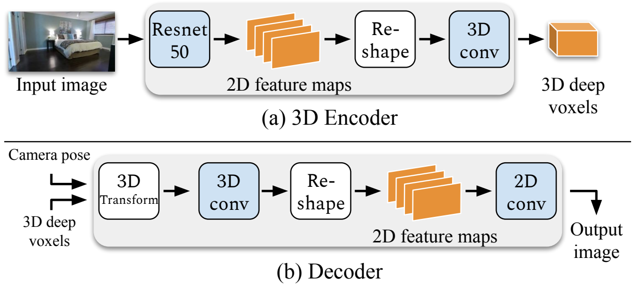

Figure 3: **Structure of 3D Encoder and Decoder:** The 3D encoder takes an image as input and estimates a 3D deep voxel corresponding to the same scene. Conversely, the decoder takes the 3D voxel and a camera pose as input and renders a 2D image as output.

camera poses the *trajectory*. Because we compute camera pose for every video frame, the length of the trajectory is the same as the number of frames. To compute the relative pose between a particular target image and the reference image (*i.e.,* the first frame), we make use of a simple ConvNet $\mathcal{H}$. The network takes as input both the target and reference image, stacked along the channel dimension (*i.e.,* the input channel is 6), and computes a 6-dim vector through a series of seven 2D convolutions. We use this vector as the 3D rotation and translation between the reference image and the target image, and it is used for transforming the deep voxels (Sec. 3.3). Overall, we obtain the trajectory as

$$E = \mathcal{F}_{\text{traj}}(V) = \{\mathcal{H}(I_1, I_i)\}_{i=1}^{N}$$

3.3. Decoder

The decoder $\mathcal{G}$ is very similar to an inverse process of the 3D encoder: it renders a 3D deep voxel representation back into image space with a given camera transformation.

$$\hat{I} = \mathcal{G}(Z, e),\ \ e \in E$$

Fig. 3 (b) illustrates the architecture of the decoder. Taking camera pose and 3D deep voxels as input, the decoder first applies the 3D transformation (*3D Transform* in Figure 3 (b)) of the camera pose on the deep voxels. If the camera pose and the 3D representation are both correct, the transformed 3D voxels should match the frame corresponds to the camera pose. Specifically, we warp the grid such that the voxel at location $p = (i, j, k)^T$ will be warped to $\hat{p}$, which is computed as

$$\hat{p} = Rp + t$$

where R, t is the 3×3 rotation matrix and translation vector corresponding to the camera pose. In our implementation, the warp is performed inversely and the value at fractional grid location is trilinearly sampled. Due to the coarseness of the voxel representation, there could be misaligned voxels during the sampling procedure. We thus apply two 3D convolutions to refine and correct these mismatches (*3D Transform* in Figure 3 (b)). The refined voxels are then reshaped back into the 2D feature maps. To align with the similar *reshape* process in 3D Encoder, we concatenate the feature dimension and the depth dimension. That is, if the input 3D deep voxel has dimension $H \times W \times D \times C$, the reshaped tensor will be of size $H \times W \times (D * C)$. This is a set of 2D feature maps which we then use several layers of 2D convolutions to map them back to an image. The output image has the original resolution. In our implementation, $H = 64, W = 64, D = 32, C = 32$.

3.4. Training Loss

We apply a reconstruction loss between reconstructed video clips and the original video clips. The loss is defined as,

$$L_{\text{recon}}(\hat{I}_t, I_t) = \lambda_{L1}||\hat{I}_t - I_t||_1 + \lambda_{\text{perc}} L_{\text{perc}}(\hat{I}_t, I_t)$$

where I_t is the original video frame at time t and $\hat{I}_t$ is the reconstructed frame at time t; L_{perc} denotes the VGG-16 perceptual loss [34, 10]. In our experiments, $\lambda_{L1} = 10$, $\lambda_{\text{perc}} = 0.1$. To enhance the image quality of reconstructed images, we also apply a WGAN-GP [16] adversarial loss on each output frame in addition to the reconstruction loss. This adversarial loss comes from a separate critic function $\mathcal{F}_D$ which learns to decide if the image looks realistic. Formally, the WGAN-GP minimizes the value function,

$$\min_{G} \max_{\mathcal{F}_D \in \mathcal{D}} \mathop{\mathbb{E}}_{I \in \mathbb{P}_r} [\mathcal{F}_D(I)] - \mathop{\mathbb{E}}_{\hat{I} \in \mathbb{P}_g} [\mathcal{F}_D(\hat{I})]$$

and thereby minimizes the Wasserstein distance between the data distribution defined by the training set and the model distribution induced by Video Autoencoder. Here G is our model, $\mathcal{D}$ is the set of 1-Lipschitz functions, $\mathbb{P}_r$ is data distribution and $\mathbb{P}_g$ is the model distribution. The loss L_{GAN} is thus defined as the negated critic score for each image:

$$L_{\text{GAN}}(\hat{I}) = -\mathcal{F}_D(\hat{I})$$

Finally, in order to ensure that a *single* 3D structure is used to represent different frames of the same scene, we apply a consistency loss between the 3D voxels extracted from different frames. Specifically, we want to ensure that any pairs of images from the same video, I_{t1}, I_{t2}, should be encoded into the same 3D representation, after rotating by the relative camera motion. Formally, we apply consistency loss

$$L_{\text{cons}}(I_{t1}, I_{t2}) = ||\mathcal{R}(\mathcal{F}_{\text{3D}}(I_{t1}), \mathcal{H}(I_{t1}, I_{t2})) - \mathcal{F}_{\text{3D}}(I_{t2})||_1$$

where we enforce that the 3D deep voxel encoded from frame I_{t1}, after transformed by the relative pose between I_{t1} and I_{t2}, should be consistent with the 3D deep voxel encoded from frame I_{t2}. $\mathcal{F}_{\text{3D}}$ and $\mathcal{H}$ are described in Sec. 3.1 and 3.2, respectively. $\mathcal{R}$ is the 3D transformation function described in Sec. 3.3.

Overall, our final loss is:

$$L = \sum_t L_{\text{recon}}(I_t, \hat{I}_t) + \lambda_{\text{GAN}} L_{\text{GAN}}(\hat{I}_t) + \lambda_{\text{cons}} L_{\text{cons}}(I_0, I_t)$$

In our experiments, we use $\lambda_{\text{cons}} = 1$, $\lambda_{\text{GAN}} = 0.01$.

4. Experiments

In this section, we empirically evaluate our method and compare it to existing approaches on three different tasks: camera pose estimation, single image novel view synthesis, and video following. We show that, although our method is quite simple, it performs surprisingly well against more complex existing methods.

4.1. Implementation Details

As preprocessing, we resize all images into a resolution of 256×256. During training, the training video clip consists of 6 frames. When we train on the RealEstate10K dataset [80], these 6 frames are sampled at a frame-rate of 4 fps so that the motion is sufficiently large. For training on Matterport3D [6], we do not sample with intervals because the motion between frames is already large. The depth dimension of the 3D deep voxels is set to $D = 32$ in our implementation. We train our model end-to-end using a batch size of 4 for 200K iterations with an Adam optimizer [38]. The initial learning rate is set to $2e^{-4}$ and is halved at 80K, 120K and 160K iterations. The training time is about 2 days on 2 Tesla V100 GPUs.

4.2. Camera Pose Estimation

We evaluate our pose estimation results (i.e., the predicted trajectory) qualitative and quantitatively. Specifically, we use 30-frame video clips from the RealEstate10K testing set, which consists of videos unseen during training. For each video clip, we estimate the relative pose between every two video frames and chain them together to get the full trajectory. Because this estimated transformation is in the coordinate space of deep voxels, we apply Umeyama alignment [68] to align the predicted camera trajectory with an SfM trajectory provided in the dataset.

We evaluate the Absolute Trajectory Error (ATE) on the RealEstate10K dataset and compare our performance with the state-of-the-art self-supervised viewpoint estimation method SSV [46] and a structurally similar method SfMLearner [79]. Additionally, we also compare with P^2-Net [76] (a.k.a Indoor SfMlearner), an improved version of [79] that is optimized for indoor environment. We use 1000 30-frame (2.5-sec) video sequences and measure the difference between the trajectory estimated by each method and the trajectory obtained from SfM. Results are shown in Table 1. Our result drastically reduces the error rate of the learning-based baseline method [79] with about 69% less in mean error and 72% less in maximum error, suggesting that our approach learns much better viewpoint representations. Comparing to the Structure from Motion pipeline COLMAP [56], our method can obtain higher accuracy under the 30-frame testing setup. We also looked at the subset of clips on which COLMAP fails (12.0% of all clips): for these, our method achieves even better results (mean error $= 0.004$, max error $= 0.009$). Inspecting the failed videos, most have either very small motion, or pure rotations. These cases are hard/impossible for SfM but are potentially easy to learn. Finally, note COLMAP takes much longer to process a video compared to our method (71.53 secs versus our 0.004 secs).

A further application of our camera trajectory is *camera stabilization*. We show that we can warp future frames back into the viewpoint of the first frame by using the estimated relative pose between these two frames. Please see our website for details.

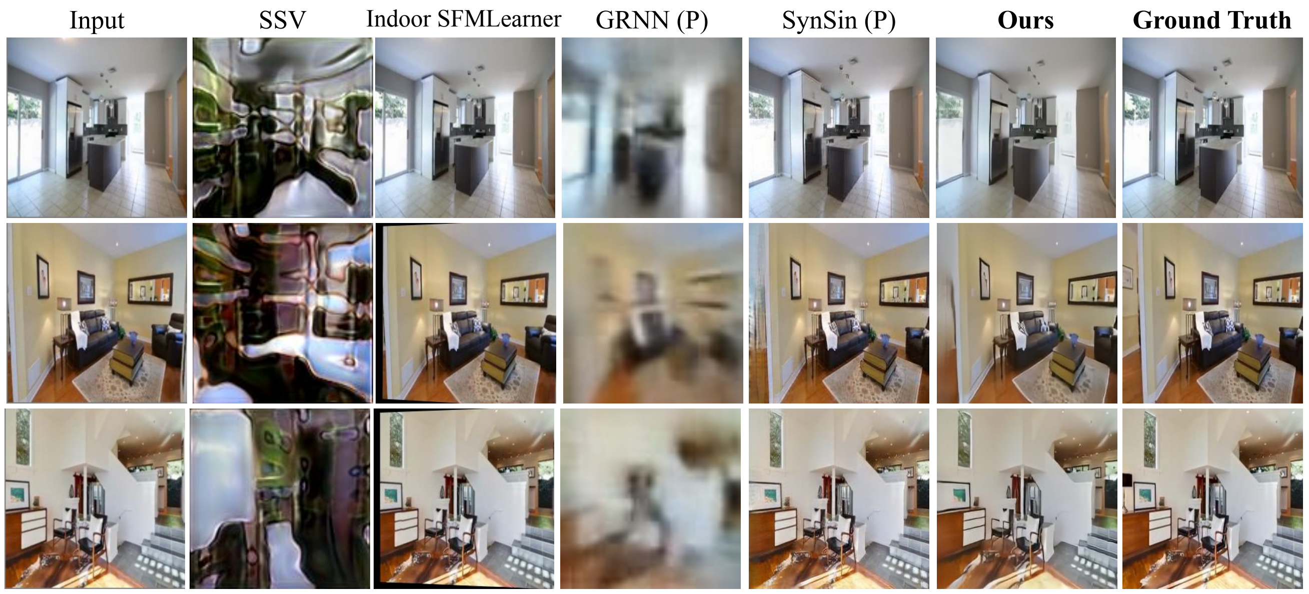

Figure 4: **Novel View Synthesis (our method vs. previous methods):** Other methods show systematic errors either in rendering or pose estimation. Similar to our work, SSV [46] make use of the least supervision signals. However, the view synthesis results are quite unsatisfactory. Indoor SFMLearner [76] (SFMLearner optimized for indoor scenes) warps the image with predicted depth and pose. However, this warping operation could also cause large blank areas where no corresponding original pixels could be found. GRNN [13] shares the most similar representation with ours, but it fails to generate clear images for reasons including their model could only handle 2 dof of camera transformation. Synsin [72] shows the most competitive performance as their model trained with much stronger supervision. See Fig. 5 for a detailed comparison. (P) denotes model trained with camera pose.

Method	Mean↓	RMSE↓	Max err.↓	Density↑
SSV [46]	0.142	0.175	0.365	100.0%
P^2-Net [76]	0.059	0.068	0.1475	100.0%
SFMLearner [79]	0.048	0.055	0.1105	100.0%
COLMAP [56]	0.024	0.030	0.0765	88.0%
Ours	**0.017**	**0.019**	**0.0410**	100.0%

Table 1: Absolute Trajectory Error (ATE) on RealEstate10K [80] dataset. We evaluate on 1000 30-frame video clips and take an average across all clips.

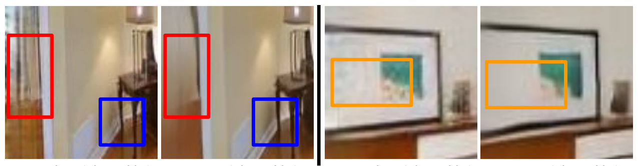

Figure 5: **Extrapolating into unseen areas (details)**: Both Synsin and our model exhibit artifacts when extrapolating into unseen areas, but our model is able to produce more smoothed results (as shown in three colorful rectangles).

4.3. Novel View Synthesis

Creating a 3D walk-though from a single still image has been a classic computer graphics task [25, 55], more recently known as single image Novel View Synthesis [62, 81, 79, 72, 65]. Given an image and a desired viewpoint, the aim is to synthesize the same scene from that new viewpoint. We should that our video autoencoder can be effectively utilized for this task. We report results on two public datasets: RealEstate10K [80] and Matterport3D [6]. Additionally, we benchmark the generalization ability of our approach and the baselines on an additional dataset: Replica dataset [61] . We use the Replica datasets to test the out-of-domain accuracy of our model because they are not used during training.

Metrics. We compare against other methods using PSNR (Peak Signal-to-Noise Ratio), SSIM (Structural Similarity), and LPIPS (Perceptual Similarity). PSNR measures pixel-wise differences between two images, and SSIM measures luminance, contrast, and structure changes and aims to better reflect the human perceptual quality. A higher number indicates better results. LPIPS measures the distance in deep feature space, and is shown to be a good indicator of perceptual similarity. Lower values are better.

4.3.1 Novel View Synthesis on RealEstate10K

The RealEstate10K dataset consists of footages of real estates (both indoor and outdoor) and is mostly static. We use 10000 videos for training and 5000 videos for testing. In Table 2, we compare our method with previous approaches on the real dataset RealEstate10K. As seen in the table, we compare favorably to most single-image view synthesis algorithms, even though our method does not train on camera pose ground-truths while other methods do. Our method is able to achieve better results in both PSNR and SSIM than Synsin [72], which is a recent approach using point clouds as the intermediate representation and ground-truth cameras

Method	Cam_{in}	Cam_{ex}	PSNR↑	SSIM↑	LPIPS↓
methods without any camera supervision:					
SSV[46]	×	×	7.95	0.19	4.12
Ours	×	×	**23.21**	**0.73**	**1.54**
methods trained with camera intrinsics:					
SfMLearner[79]	✓	×	15.82	0.46	2.39
MonoDepth2[15]	✓	×	17.15	0.55	2.08
P^2-Net[76]	✓	×	17.77	0.56	1.96
methods trained with camera intrinsics and extrinsics:					
Dosovitsky et al [11]	✓	✓	11.35	0.33	3.95
GQN [12]	✓	✓	16.94	0.56	3.33
Appearance Flow [81]	✓	✓	17.05	0.56	2.19
GRNN [13]	✓	✓	19.13	0.63	2.83
3DPaper [72]	✓	✓	21.88	0.66	1.52
SynSin (w/ voxel) [72]	✓	✓	21.88	0.71	1.30
SynSin [72]	✓	✓	22.31	0.74	**1.18**
Single-view MPIs [65]	✓	✓	23.70	0.80	-
StereoMag† [80]	✓	✓	**25.34**	**0.82**	1.19

Table 2: Novel view synthesis task with RealEstate10K [80]. We follow the standard metrics of PSNR, SSIM and LPIPS [77]. For PSNR and SSIM, higher numbers are better. For LPIPS, lower numbers are better. We use implementation of [72] to compute LPIPS. † StereoMag makes use of 2 images as input.

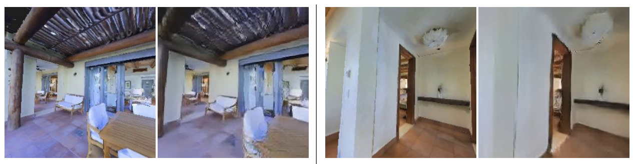

Figure 6: Our model can also be applied on Matterport3D [6], using images rendered from 3D models in the dataset.

during training. This indicates learning the disentangled deep voxel and camera pose jointly can lead to better 3D representations. Our method also easily outperforms methods trained without using camera poses. SSV [46], which similarly uses no camera information during training, fails to generate meaningful content. SfMLearner [79] and the subsequent P^2-Net [76] (a.k.a. Indoor SfMLearner) are structurally similar to ours. We warp the image input with the predicted depth and pose to test their view synthesis results. However, these methods are not optimized for view synthesis and fail to achieve the image quality as ours.

Figure 4 shows the qualitative comparisons. Given a *single image* and a specified motion trajectory as inputs, our method is able to generate photorealistic results with correct motion. While Synsin [72] achieves competitive results, it requires true camera poses for training. Our model also shows reasonable extrapolation into unseen areas. Both row 2 and 3 involve extrapolating into unknown areas to the left. Fig. 5 shows detailed comparison in unseen areas between our model and [72]. Our model produces more smoothed results and [72] shows stronger artifacts.

4.3.2 Novel View Synthesis on Matterport3D

We evaluate the Video Autoencoder on the Matterport3D dataset [6]. The Matterport3D dataset is a collection of 3D

Method	Pose	PSNR↑	SSIM↑	LPIPS↓
methods without any camera supervision:				
Ours	×	20.58	0.64	2.44
methods trained with camera intrinsics and extrinsics:				
Dosovitsky et al [11]	✓	14.79	0.57	3.73
Appearance Flow [81]	✓	15.87	0.53	2.99
Synsin (w/ voxel) [72]	✓	20.62	0.70	1.97
Synsin [72]	✓	20.91	0.72	1.68

Table 3: Novel view synthesis with Matterport3D [6]. We follow the standard metrics of PSNR and SSIM. Higher values are better.

Method	Pose	PSNR↑	SSIM↑	LPIPS↓
methods without any camera supervision:				
Ours	×	21.72	0.77	2.21
methods trained with camera intrinsics and extrinsics:				
Dosovitsky et al [11]	✓	14.36	0.68	3.36
Appearance Flow [81]	✓	17.42	0.66	2.29
Synsin (w/ voxel) [72]	✓	19.77	0.75	2.24
Synsin [72]	✓	21.94	0.81	1.55

Table 4: Novel view synthesis task with Replica dataset [61].

models of scanned and reconstructed properties. It consists of 61 training scenes and 18 testing scenes. We use a navigation agent in the Habitat simulator [54] to render around 100 episodes per scene as videos. These videos show an agent navigating from one point in the scene to another point. A total of around 6000 videos is generated for training and 800 videos for testing. For experiments, we rendered an additional 60K image pairs related by a random rotation and translation for fine-tuning the model. In this way, the training data includes rotations in all three axes.

Table 3 shows the numerical results for Video Autoencoder on the Matterport3D. Without training on any camera information, our model is able to perform comparably to methods supervised with pose across all three metrics. Figure 6 visualizes the view synthesis results of our method using data from the Matterport3D dataset. Although the visual appearance rendered from 3D models is usually flawed due to incomplete point clouds, occlusions, and other possible artifacts, our model still produces satisfactory results.

4.3.3 Generalization Ability of Novel View Synthesis

We benchmark Video Autoencoder on an out-of-domain dataset, Replica [61], a set of 3D reconstructions of indoor spaces, to evaluate the generalization ability of our model, without any further finetuning. We rendered 200 image pairs on each of the 5 episodes with Habitat simulator [54] as the test set. Table 4 shows numerical results compared to other methods. All methods use Matterport3D as the training dataset and only test on Replica without any finetuning on it. We observe the same trend as in other datasets, with our method perform comparably to existing methods that require camera supervision. This indicates that our method is able to consistently outperform the baseline methods on generalization to out-of-domain data.

Resolution	SSIM↑	PSNR↑	LPIPS↓
128×128	0.70	22.39	1.63
64×64	**0.73**	**23.21**	**1.54**
32×32	0.61	20.23	2.77

(a) **3D deep voxel spatial resolution**: Our final model with a resolution of 64 × 64 offers the best performance.

Depth	SSIM↑	PSNR↑	LPIPS↓
D=64	0.71	22.05	1.61
D=32	**0.73**	**23.21**	**1.54**
D=16	0.70	21.92	1.86

(b) **3D deep voxel depth resolution**: A depth dimension of 32 shows superior results compared to other variants.

# Frame	SSIM↑	PSNR↑	LPIPS↓
3 frames	0.72	22.52	**1.52**
6 frames	**0.73**	**23.21**	1.54
10 frames	0.69	21.54	1.73

(c) **Video clip for training - number of frame**: Training with a clip of 6 frames can offer better overall performance .

Table 5: **Ablations** on RealEstate10K view synthesis results. We show SSIM, PSNR and LPIPS performance on testing set.

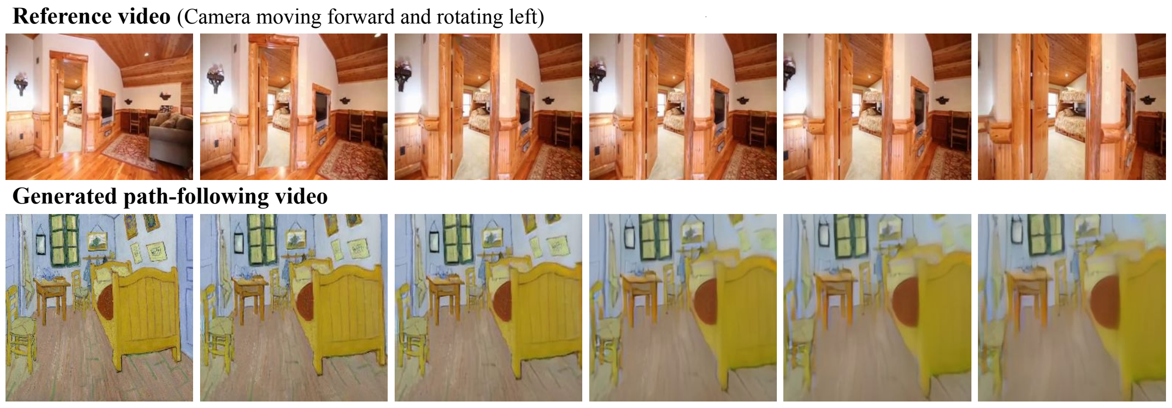
Reference video (Camera moving forward and rotating left)
Generated path-following video

Figure 7: **Following trajectories of other videos:** by using trajectories of other videos, we can even animate oil paintings.

4.3.4 Ablation Studies

To quantitatively evaluate the impact of various components of the Video Autoencoder, we conduct a set of ablation studies using variants of our model in Table 5. The best model with 6 input frames, 64 × 64 spatial resolution and 32 depth resolution in the 3D voxel corresponds to our model reported in previous sections.

Resolution of 3D voxels: We compare our default voxel resolution with variants that modify the encoder's output spatial resolution and depth. As shown in Table 5a, the errors increase drastically when the spatial resolution is changed. Comparing to models with depth changed (Table 5b), although the other variants are capable of attaining reasonable performance, our default model achieves the best performance.

Video clip for training: We modify the training video clip by changing the clip length. As seen in Table 5c, the performance improved when the clip length is increase. We conjecture that a clip too short could provide an appropriate scale of motion, which is crucial for training. However, if we further expand the clip length to 10 frames, the performance drops by about 5%. We hypothesize that predicting the last frame from the first frame becomes too difficult under such a setting, which is undesirable for training. This suggests that 6-frame clips are more effective training data.

4.4. Video Following

Finally, we evaluate Video Autoencoder on the task of animating a single image with the motion trajectories from different videos. Specifically, we obtain a 3D deep voxels representation from our desired image and trajectory from a different video. We then combine the trajectory and the 3D structure for the decoder to render a new video.

Figure 7 visualizes a video predicted from an out-of-domain image. Training only once on the RealEstate10K dataset, our model can adapt to a diverse set of images. The shown frames are generated from the painting *Bedroom in Arles*. Although the painting has a texture that is quite different from the training dataset, our method still models it reasonably well. Adapting from the reference video, the generated sequence shows a trajectory as if we are walking into Vincent van Gogh's bedroom in Arles.

5. Conclusion

We present Video Autoencoder that encodes videos into disentangled representations of 3D structure and camera pose. The model is trained with only raw videos without using any explicit 3D supervision or camera pose. We show that our representation enables tasks such as camera pose estimation, novel view synthesis and video generation by motion following. Our model demonstrates superior generalization ability on all tasks and achieves state-of-the-art results on self-supervised camera pose estimation. Our model also achieves on par results on novel view synthesis comapred to approaches using ground-truth camera in training.

Acknowledgements. This work was supported, in part, by grants from DARPA MCS and LwLL, NSF 1730158 CI-New: Cognitive Hardware and Software Ecosystem Community Infrastructure, NSF ACI-1541349 CC*DNI Pacific Research Platform, and gifts from Qualcomm, TuSimple and Picsart. We thank Taesung Park and Bill Peebles for valuable comments.

References

[1] Sameer Agarwal, Noah Snavely, Ian Simon, Steven M Seitz, and Richard Szeliski. Building rome in a day. In *Proc. ICCV*, 2009. 2

[2] Pulkit Agrawal, Joao Carreira, and Jitendra Malik. Learning to see by moving. In *Proceedings of the IEEE international conference on computer vision*, pages 37–45, 2015. 2

[3] Ivan Anokhin, Pavel Solovev, Denis Korzhenkov, Alexey Kharlamov, Taras Khakhulin, Aleksei Silvestrov, Sergey Nikolenko, Victor Lempitsky, and Gleb Sterkin. High-resolution daytime translation without domain labels. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7488–7497, 2020. 1, 2

[4] Aayush Bansal, Minh Vo, Yaser Sheikh, Deva Ramanan, and Srinivasa Narasimhan. 4d visualization of dynamic events from unconstrained multi-view videos. In *Proc. CVPR*, 2020. 2

[5] Harry G. Barrow and J.M. Tenenbaum. Recovering intrinsic scene characteristics. *Comput. Vis. Syst*, 2(3-26):2, 1978. 1

[6] 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. *3*. 2, 5, 6, 7

[7] Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. ShapeNet: An Information-Rich 3D Model Repository. Technical Report arXiv:1512.03012 [cs.GR], Stanford University — Princeton University — Toyota Technological Institute at Chicago, 2015. 16

[8] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in neural information processing systems*, pages 2172–2180, 2016. 2

[9] Emily Denton and Vighnesh Birodkar. Unsupervised learning of disentangled representations from video. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 4417–4426, 2017. 2

[10] Alexey Dosovitskiy and Thomas Brox. Inverting visual representations with convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4829–4837, 2016. 5

[11] Alexey Dosovitskiy, Jost Tobias Springenberg, and Thomas Brox. Learning to generate chairs with convolutional neural networks. In *Proc. CVPR*, 2015. 7, 12

[12] SM Ali Eslami, Danilo Jimenez Rezende, Frederic Besse, Fabio Viola, Ari S Morcos, Marta Garnelo, Avraham Ruderman, Andrei A Rusu, Ivo Danihelka, Karol Gregor, et al. Neural scene representation and rendering. *Science*, 360(6394):1204–1210, 2018. 7, 12

[13] Hsiao-Yu Fish Tung, Ricson Cheng, and Katerina Fragkiadaki. Learning spatial common sense with geometry-aware recurrent networks. In *cvpr*. 2, 6, 7, 12

[14] J. Flynn, I. Neulander, J. Philbin, and N. Snavely. Deep stereo: Learning to predict new views from the world's imagery. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5515–5524, 2016. 2

[15] Clément Godard, Oisin Mac Aodha, Michael Firman, and Gabriel J Brostow. Digging into self-supervised monocular depth estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3828–3838, 2019. 7

[16] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In *NerIPS*, 2017. 5

[17] Tengda Han, Weidi Xie, and Andrew Zisserman. Video representation learning by dense predictive coding. In *Proc. ICCV*, 2019. 2

[18] Adam W Harley, Shrinidhi Kowshika Lakshmikanth, Paul Schydlo, and Katerina Fragkiadaki. Tracking emerges by looking around static scenes, with neural 3d mapping. In *eccv*, 2020. 2

[19] Adam W. Harley, Fangyu Li, Shrinidhi K. Lakshmikanth, Xian Zhou, Hsiao-Yu Fish Tung, and Katerina Fragkiadaki. Learning from unlabelled videos using contrastive predictive neural 3d mapping. In *ICLR*, 2019. 2

[20] Ananya Harsh Jha, Saket Anand, Maneesh Singh, and VSR Veeravasarapu. Disentangling factors of variation with cycle-consistent variational auto-encoders. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 805–820, 2018. 1, 2

[21] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003. 2

[22] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. CVPR*, 2016. 3

[23] Ari Heljakka, Yuxin Hou, Juho Kannala, and Arno Solin. Deep automodulators. *Advances in Neural Information Processing Systems*, 33, 2020. 1, 2

[24] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. 2016. 1, 2

[25] Derek Hoiem, Alexei A Efros, and Martial Hebert. Automatic photo pop-up. In *ACM SIGGRAPH 2005 Papers*. 2005. 6

[26] Jun-Ting Hsieh, Bingbin Liu, De-An Huang, Li F Fei-Fei, and Juan Carlos Niebles. Learning to decompose and disentangle representations for video prediction. In *Advances in Neural Information Processing Systems*, pages 517–526, 2018. 2

[27] Ronghang Hu, Nikhila Ravi, Alexander C. Berg, and Deepak Pathak. Worldsheet: Wrapping the world in a 3d sheet for view synthesis from a single image. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2021. 2

[28] Xun Huang, Ming-Yu Liu, Serge Belongie, and Jan Kautz. Multimodal unsupervised image-to-image translation. In *ECCV*, 2018. 2

[29] Phillip Isola, Daniel Zoran, Dilip Krishnan, and Edward H. Adelson. Learning visual groups from co-occurrences in space and time. In *Proc. ICLR*, 2015. 2

[30] Tomas Jakab, Ankush Gupta, Hakan Bilen, and Andrea Vedaldi. Unsupervised learning of object landmarks through conditional image generation. In *Advances in neural information processing systems*, pages 4016–4027, 2018. 2

[31] Dinesh Jayaraman and Kristen Grauman. Learning image representations tied to ego-motion. In *Proc. ICCV*, 2015. 2

[32] Dinesh Jayaraman and Kristen Grauman. Slow and steady feature analysis: higher order temporal coherence in video. In *Proc. CVPR*, 2016. 2

[33] Danilo Jimenez Rezende, SM Eslami, Shakir Mohamed, Peter Battaglia, Max Jaderberg, and Nicolas Heess. Unsupervised learning of 3d structure from images. *Advances in neural information processing systems*, 29:4996–5004, 2016. 2

[34] 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

[35] Angjoo Kanazawa, Shubham Tulsiani, Alexei A Efros, and Jitendra Malik. Learning category-specific mesh reconstruction from image collections. In *Proc. ECCV*, 2018. 2

[36] 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

[37] Hyunjik Kim and Andriy Mnih. Disentangling by factorising. In *ICML*, 2018. 1, 2

[38] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 5

[39] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. 2

[40] Tejas D Kulkarni, William F Whitney, Pushmeet Kohli, and Josh Tenenbaum. Deep convolutional inverse graphics network. In *Advances in neural information processing systems*, pages 2539–2547, 2015. 1, 2

[41] Hsin-Ying Lee, Hung-Yu Tseng, Qi Mao, Jia-Bin Huang, Yu-Ding Lu, Maneesh Singh, and Ming-Hsuan Yang. Drit++: Diverse image-to-image translation via disentangled representations. *International Journal of Computer Vision*, pages 1–16, 2020. 2

[42] Andrew Liu, Shiry Ginosar, Tinghui Zhou, Alexei A. Efros, and Noah Snavely. Learning to factorize and relight a city. In *ECCV*, 2020. 1, 2

[43] 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 *Proc. ECCV*, 2020. 2, 3

[44] Matthias Minderer, Chen Sun, Ruben Villegas, Forrester Cole, Kevin P Murphy, and Honglak Lee. Unsupervised learning of object structure and dynamics from videos. In *Advances in Neural Information Processing Systems*, pages 92–102, 2019. 2

[45] Ishan Misra, C. Lawrence Zitnick, and Martial Hebert. Shuffle and learn: Unsupervised learning using temporal order verification. In *Proc. ECCV*, 2016. 2

[46] Siva Karthik Mustikovela, Varun Jampani, Shalini De Mello, Sifei Liu, Umar Iqbal, Carsten Rother, and Jan Kautz. Self-supervised viewpoint learning from image collections. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3971–3981, 2020. 2, 5, 6, 7, 12

[47] Thu Nguyen-Phuoc, Chuan Li, Lucas Theis, Christian Richardt, and Yong-Liang Yang. Hologan: Unsupervised learning of 3d representations from natural images. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 7588–7597, 2019. 2

[48] Simon Niklaus, Long Mai, Jimei Yang, and Feng Liu. 3d ken burns effect from a single image. *ACM Trans. Gr.*, 38(6), 2019. 2

[49] Taesung Park, Jun-Yan Zhu, Oliver Wang, Jingwan Lu, Eli Shechtman, Alexei Efros, and Richard Zhang. Swapping autoencoder for deep image manipulation. *Advances in Neural Information Processing Systems*, 33, 2020. 1, 2

[50] William Peebles, John Peebles, Jun-Yan Zhu, Alexei A. Efros, and Antonio Torralba. The hessian penalty: A weak prior for unsupervised disentanglement. In *Proceedings of European Conference on Computer Vision (ECCV)*, 2020. 2

[51] Stanislav Pidhorskyi, Donald A Adjeroh, and Gianfranco Doretto. Adversarial latent autoencoders. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14104–14113, 2020. 1, 2

[52] Mihir Prabhudesai, Shamit Lal, Darshan Patil, Hsiao-Yu Tung, Adam W Harley, and Katerina Fragkiadaki. Disentangling 3d prototypical networks for few-shot concept learning. In *iclr*, 2021. 2

[53] Chris Rockwell, David F. Fouhey, and Justin Johnson. Pixelsynth: Generating a 3d-consistent experience from a single image. In *ICCV*, 2021. 2

[54] Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, Devi Parikh, and Dhruv Batra. Habitat: A Platform for Embodied AI Research. In *Proc. ICCV*, 2019. 7

[55] Ashutosh Saxena, Min Sun, and Andrew Y Ng. Make3d: Learning 3d scene structure from a single still image. *IEEE transactions on pattern analysis and machine intelligence*, 31(5):824–840, 2008. 6

[56] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Proc. CVPR*, 2016. 2, 5, 6

[57] Yujun Shen, Jinjin Gu, Xiaoou Tang, and Bolei Zhou. Interpreting the latent space of gans for semantic face editing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9243–9252, 2020. 2

[58] Vincent Sitzmann, Justus Thies, Felix Heide, Matthias Nießner, Gordon Wetzstein, and Michael Zollhofer. Deepvoxels: Learning persistent 3d feature embeddings. In *Proc. CVPR*, 2019. 2

[59] Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. Scene representation networks: Continuous 3d-structure-aware neural scene representations. In *NerIPS*, 2019. 2

[60] Keith N Snavely. *Scene reconstruction and visualization from internet photo collections*. University of Washington USA, 2008. 2

[61] Julian Straub, Thomas Whelan, Lingni Ma, Yufan Chen, Erik Wijmans, Simon Green, Jakob J. Engel, Raul Mur-Artal, Carl Ren, Shobhit Verma, Anton Clarkson, Mingfei Yan, Brian Budge, Yajie Yan, Xiaqing Pan, June Yon, Yuyang Zou, Kimberly Leon, Nigel Carter, Jesus Briales, Tyler Gillingham, Elias Mueggler, Luis Pesqueira, Manolis Savva, Dhruv Batra, Hauke M. Strasdat, Renzo De Nardi, Michael Goesele, Steven Lovegrove, and Richard Newcombe. The Replica dataset: A digital replica of indoor spaces. *arXiv preprint arXiv:1906.05797*, 2019. 2, 6, 7

[62] Maxim Tatarchenko, Alexey Dosovitskiy, and Thomas Brox. Multi-view 3d models from single images with a convolutional network. In *European Conference on Computer Vision*, pages 322–337. Springer, 2016. 6, 16

[63] Ayush Tewari, Ohad Fried, Justus Thies, Vincent Sitzmann, Stephen Lombardi, Kalyan Sunkavalli, Ricardo Martin-Brualla, Tomas Simon, Jason Saragih, Matthias Nießner, et al. State of the art on neural rendering. In *Computer Graphics Forum*, volume 39, pages 701–727. Wiley Online Library, 2020. 2

[64] Carlo Tomasi and Takeo Kanade. Shape and motion from image streams under orthography: a factorization method. *International journal of computer vision*, 9(2):137–154, 1992. 1

[65] Richard Tucker and Noah Snavely. Single-view view synthesis with multiplane images. In *Proc. CVPR*, 2020. 3, 6, 7

[66] Shubham Tulsiani, Tinghui Zhou, Alexei A Efros, and Jitendra Malik. Multi-view supervision for single-view reconstruction via differentiable ray consistency. In *Proc. CVPR*, 2017. 2

[67] Sergey Tulyakov, Ming-Yu Liu, Xiaodong Yang, and Jan Kautz. Mocogan: Decomposing motion and content for video generation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1526–1535, 2018. 2

[68] Shinji Umeyama. Least-squares estimation of transformation parameters between two point patterns. *IEEE PAMI*. 5, 17

[69] Xiaolong Wang and Abhinav Gupta. Unsupervised learning of visual representations using videos. In *Proc. ICCV*, 2015. 2

[70] Xiaolong Wang and Abhinav Gupta. Generative image modeling using style and structure adversarial networks. In *ECCV*, 2016. 2

[71] Donglai Wei, Joseph Lim, Andrew Zisserman, and William T. Freeman. Learning and using the arrow of time. In *Proc. CVPR*, 2018. 2

[72] Olivia Wiles, Georgia Gkioxari, Richard Szeliski, and Justin Johnson. Synsin: End-to-end view synthesis from a single image. In *Proc. CVPR*, 2020. 2, 3, 6, 7, 12, 16

[73] O. Wiles, A.S. Koepke, and A. Zisserman. Self-supervised learning of a facial attribute embedding from video. In *British Machine Vision Conference*, 2018. 2

[74] Tianfan Xue, Jiajun Wu, Katherine Bouman, and Bill Freeman. Visual dynamics: Probabilistic future frame synthesis via cross convolutional networks. In *Advances in neural information processing systems*, pages 91–99, 2016. 2

[75] Tian Ye, Xiaolong Wang, James Davidson, and Abhinav Gupta. Interpretable intuitive physics model. In *ECCV*, 2018. 2

[76] Zehao Yu, Lei Jin, and Shenghua Gao. P^{2}net: Patch-match and plane-regularization for unsupervised indoor depth estimation. In *ECCV*, 2020. 5, 6, 7, 12

[77] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018. 7

[78] Junsheng Zhou, Yuwang Wang, Kaihuai Qin, and Wenjun Zeng. Moving indoor: Unsupervised video depth learning in challenging environments. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8618–8627, 2019. 16

[79] Tinghui Zhou, Matthew Brown, Noah Snavely, and David G Lowe. Unsupervised learning of depth and ego-motion from video. In *Proc. CVPR*, 2017. 2, 5, 6, 7, 12

[80] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: Learning view synthesis using multiplane images. *Proc. ACM SIGGRAPH*, 2018. 2, 5, 6, 7, 12

[81] Tinghui Zhou, Shubham Tulsiani, Weilun Sun, Jitendra Malik, and Alexei A Efros. View synthesis by appearance flow. In *Proc. ECCV*, 2016. 3, 6, 7, 12

[82] Jun-Yan Zhu, Zhoutong Zhang, Chengkai Zhang, Jiajun Wu, Antonio Torralba, Joshua B. Tenenbaum, and William T. Freeman. Visual object networks: Image generation with disentangled 3D representations. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018. 2

Supplementary Material

A. Additional View Synthesis Results

A.1. Our Qualitative Results

Figure 8, 9 and 10 provide additional novel view synthesis results. Our method is able to synthesize high quality view with correct camera transformation.

A.2. Qualitative Comparison with Previous Methods

Figure 11 and 12 provide additional comparison with previous methods. In Fig. 13, we show additional detailed comparison with the strong competitor, Synsin [72]. Our method is able to generate more clear and accurate results, even though our method is trained without any camera supervision.

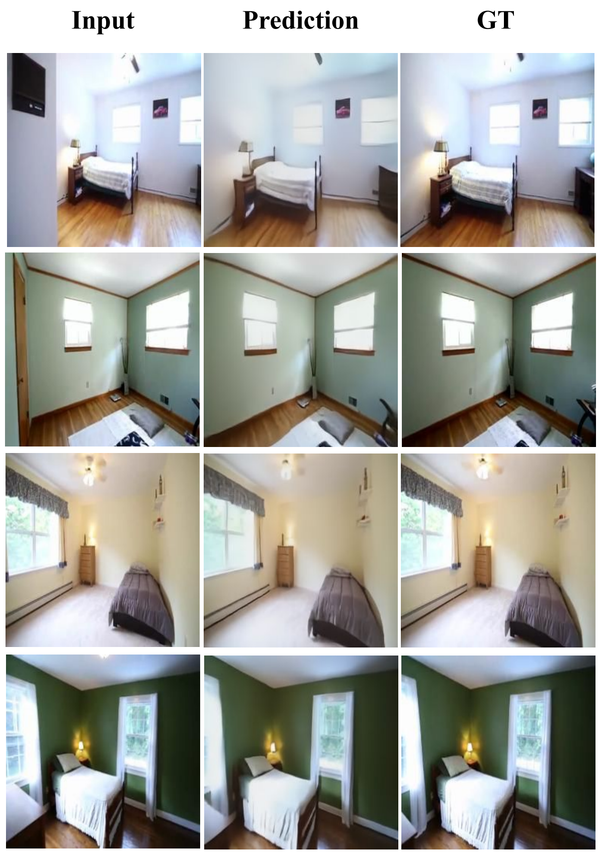

Figure 8: **Novel View Synthesis:** Our method is able to generate accurate results that correspond to the ground truths. Our method works for both camera translation (row 1-2) and rotation (row 3-4).

B. Network structures

In Table 6,7,8, we give additional information about the network strucutres of the subcomponents of our model. There is a total of 29M trainable parameters, mainly in the Resnet-50 feature extractor.

C. Details of baselines

In this section, we describe the baselines we used to compare our method with. Specifically, we compared with Dosovitsky et al. [11], Appearance Flow [81], StereoMag [80], Synsin [72], SFMLearner [79], Indoor SFMLearner [76], GQN [12], GRNN [13] and SSV [46].

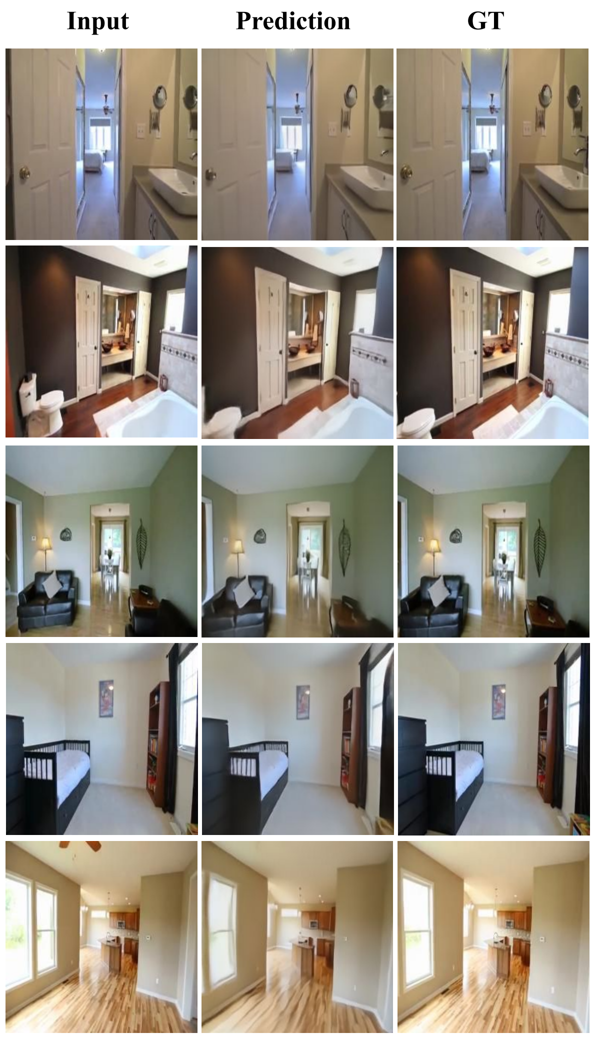

Figure 9: **Novel View Synthesis (continued)**

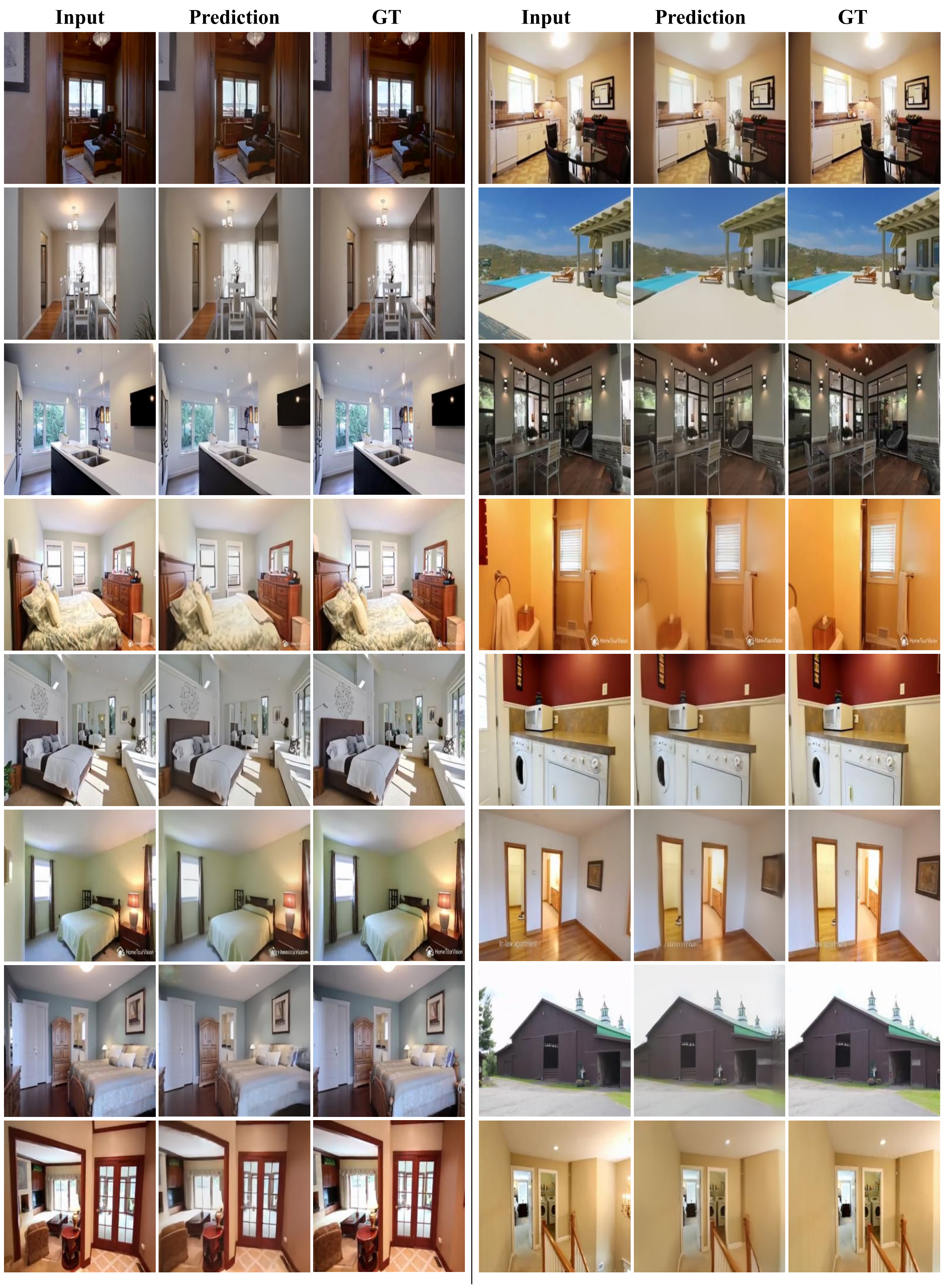

Figure 10: **Novel View Synthesis (continued)**

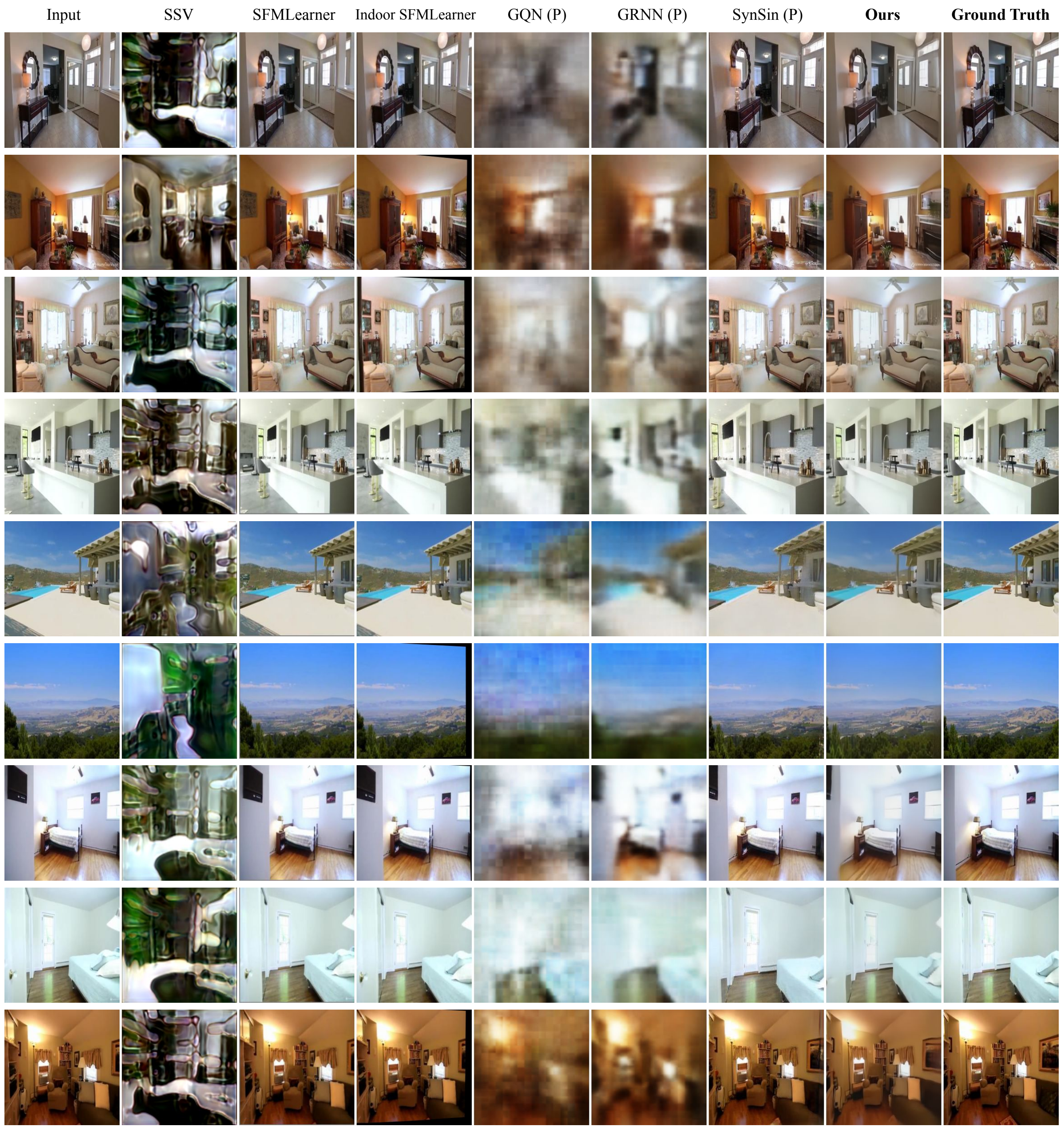

Figure 11: **Novel view synthesis results compared with previous methods:** Other methods show systematic errors either in rendering or pose estimation. Our method is able to outperform other unsupervised methods visually by large margin. (P) indicates methods supervised by true camera transformations.

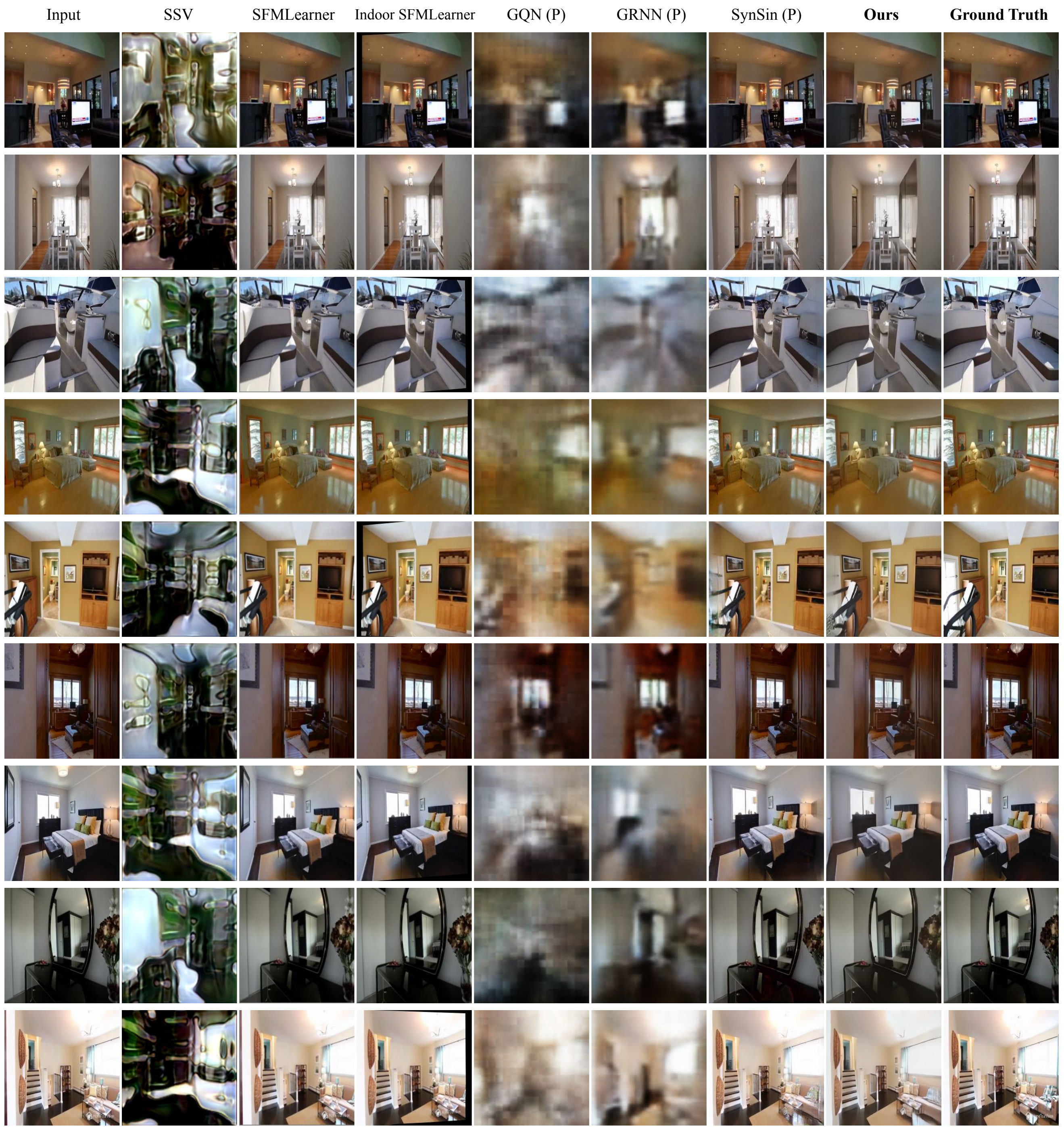

Figure 12: **Novel view synthesis results compared with previous methods (continued)**

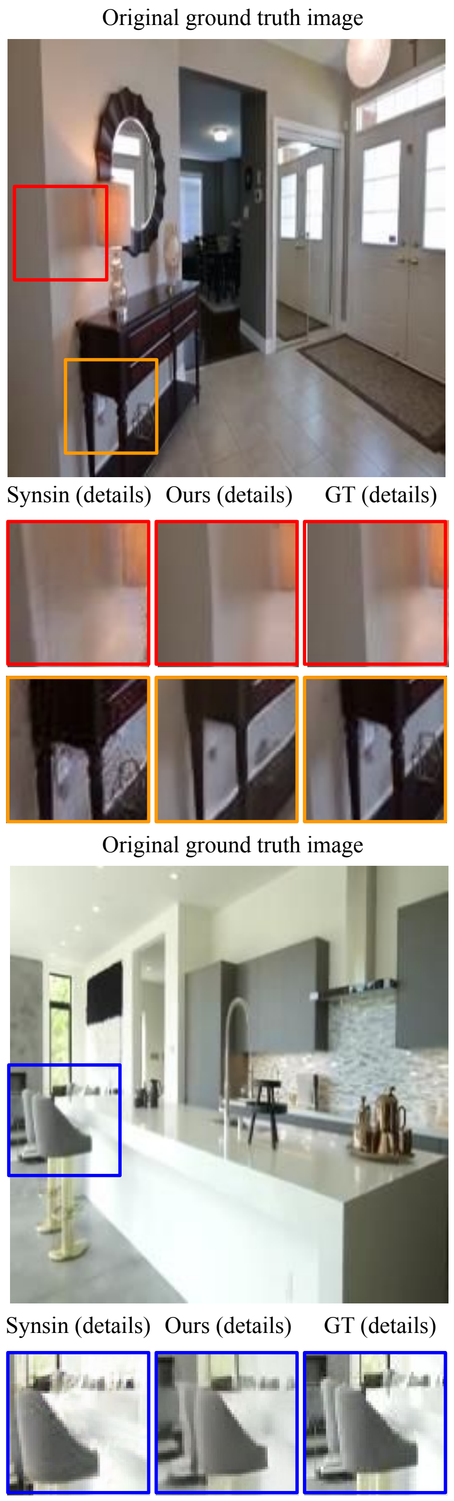

Figure 13: **Novel view synthesis results compared with [72]** Our method synthesizes novel views with less artifacts and noises. See Fig. 11 for uncropped full results of our model and [72].

Dosovitsky et al. This method infers a novel view of a given input with a neural network. The model feeds an image and the desired viewpoint into the network, which directly estimates an image corresponding to the given view. This method was only tested on images of a single object, which are considerably easier than the real scene data we used.

Appearance Flow. Appearance flow predicts a flow field that warps the original image into a novel view. This flow field is computed from a ConvNet which takes the original image and the viewpoint transformation as input. This method is shown to work very well on synthetic object datasets (*e.g.* ShapeNet [7]). It has also shown that the performance on real-world scenes surpasses direct pixel prediction [62].

SynSin. Synsin is a novel view synthesis algorithm that aims to synthesize new viewpoints of a given single image. The algorithm makes use of an intermediate representation: a point cloud where each point is a feature vector. Taking a single image as input, Synsin first computes a depth map and a set of 2D feature maps. The 2D feature maps are projected back into 3D points with the depth map. Next, a camera transformation is given and the point cloud is rendered with the new camera pose. There are two major differences in the high-level philosophy between our model and Synsin. First, we do not require true camera transformation during training time. This means that Video Autoencoder works on raw videos and Synsin can only be trained on datasets with camera pose obtained from sensors or precomputed from SfM. Second, Synsin leverages explicit 3D representation (*i.e.* the depth map) whereas the Video Autoencoder does not. As shown in the paper, explicit 3D representation could fail in out-of-domain data whereas our method generalizes better into unseen images.

StereoMag. StereoMag constructs multiplane images (MPIs) from two input views. The multiplane images represent a 3D structure as a set of fronto-parallel planes at fixed depths. It can be rendered at a given viewpoint by *blending* planes with a set of blending weights. These blending weights are predicted from one image, and the other image is used as the plane sweep volume (*i.e.* the value of the planes). Apart from the representation, one major difference between our method and StereoMag is that StereoMag makes use of two images as input, which greatly simplifies the problem.

SFMLearner. SFMLearner disentangle a short video clip into depth maps and camera trajectory. The method is structurally similar to our method. However, there are several major differences. (i) SFMLearner (and similarly, GeoNet, etc.) results were shown on the relatively simple KITTI dataset, but work poorly on more complex data [78]. (ii) SFMLearner requires ground truth camera intrinsics, which makes it more difficult to train on raw videos. (iii) SFMLearner cannot produce satisfactory view synthesis results

Stage	Configuration	Output
0	Input image	$H \times W \times 3$
2D feature extraction		
1	Extract feature with Resnet-50	$\frac{H}{16} \times \frac{W}{16} \times 2048$
Reshaping 2D to 3D		
2	Reshape feature dimension to 256	$\frac{H}{16} \times \frac{W}{16} \times 8 \times 256$
3D Convolutions		
3	3D-Deconvolution (4^3 kernel, 128 filters, stride 2)	$\frac{H}{8} \times \frac{W}{8} \times 16 \times 128$
4	3D-Deconvolution (4^3 kernel, 32 filters, stride 2)	$\frac{H}{4} \times \frac{W}{4} \times 32 \times 32$

Table 6: 3D Encoder ($\mathcal{F}_{3D}$) architecture.

Stage	Configuration	Output
0	Two concatenated input image	$H \times W \times 6$
2D feature extraction		
1	2D-convolution (3^2 kernel, 16 filters, stride 2)	$\frac{H}{2} \times \frac{W}{2} \times 16$
2	2D-convolution (3^2 kernel, 32 filters, stride 2)	$\frac{H}{4} \times \frac{W}{4} \times 32$
3	2D-convolution (3^2 kernel, 64 filters, stride 2)	$\frac{H}{8} \times \frac{W}{8} \times 64$
4	2D-convolution (3^2 kernel, 128 filters, stride 2)	$\frac{H}{16} \times \frac{W}{16} \times 128$
5	2D-convolution (3^2 kernel, 256 filters, stride 2)	$\frac{H}{32} \times \frac{W}{32} \times 256$
6	2D-convolution (3^2 kernel, 256 filters, stride 2)	$\frac{H}{64} \times \frac{W}{64} \times 256$
7	2D-convolution (3^2 kernel, 256 filters, stride 2)	$\frac{H}{128} \times \frac{W}{128} \times 256$
8	2D-convolution (1^2 kernel, 6 filters, stride 1)	$\frac{H}{128} \times \frac{W}{128} \times 6$
9	Mean pooling	6
10	Multiply by 0.01	6

Table 7: Architecture of ConvNet ($\mathcal{H}$) for Trajectory Encoder ($\mathcal{F}_{Traj}$). The last step could stabilize the training process.

because its spatial representation is a 2.5D depth map. We use the predicted depth and camera transformation to warp the first frame into the target frame.

Indoor SFMLearner. P^2-Net, or the Indoor SFM-Learner, proposes to use a patch-based loss to overcome the optimization problem of SFMLearner, and shows better results in indoor scenes. Similar to SFMLearner, we use predicted depth and camera to warp the first frame into the target frame. Because not all pixels in the target frame have a corresponding pixel in the first frame, this method could produce large blank areas (as seen in Fig. 11 and 12).

GQN Generative Query Network combines 2D features of multiple input images into a 2D *Neural Scene Representation*. This 2D representation is then decoded with an LSTM network conditioned on a query viewpoint. Although GQN is shown to yield good results on toy datasets, we find it struggles to generate clear results for real-world scenes.

Stage	Configuration	Output
0	Input 3D deep voxels	$\frac{H}{4} \times \frac{W}{4} \times 32 \times 32$
1	Input 3D transformation	6
Rotating 3D deep voxels		
2	Rotate deep voxels with input transf.	$\frac{H}{4} \times \frac{W}{4} \times 32 \times 32$
3	3D-Convolution (3^3 kernel, 64 filters, stride 1)	$\frac{H}{4} \times \frac{W}{4} \times 32 \times 64$
4	3D-Convolution (3^3 kernel, 64 filters, stride 1)	$\frac{H}{4} \times \frac{W}{4} \times 32 \times 64$
Reshape 3D to 2D		
5	Concatenate feature and depth dim.	$\frac{H}{4} \times \frac{W}{4} \times 2048$
2D Convolutions (neural network renderer)		
6	2D-convolution (1^2 kernel, 512 filters, stride 1)	$\frac{H}{4} \times \frac{W}{4} \times 512$
7	2D-Deconvolution (4^2 kernel, 64 filters, stride 2)	$\frac{H}{2} \times \frac{W}{2} \times 64$
8	2D-Deconvolution (4^2 kernel, 32 filters, stride 2)	$H \times W \times 32$
9	2D-Deconvolution (3^2 kernel, 3 filters, stride 1)	$H \times W \times 3$

Table 8: Decoder ($\mathcal{G}$) architecture.

GRNN GRNN constructs an RNN-aggregated 3D voxel from input images as scene representation. This 3D voxel could be projected back into a set of 2D feature maps with a specific query viewpoint and decoded into a query view. While GRNN also makes use of a 3D deep voxel as representation, several significant differences between our model and GRNN include: (i) GRNN only support 2 degrees of freedom for camera transformations (yaw and pitch) whereas we support a full 6 DOF camera transformation (ii) GRNN is trained with ground truth cameras whereas we do not make use camera supervision. (iii) GRNN requires a complicated matching process during testing. As shown in the main text and Fig. 11 and 12, GRNN fails to produce satisfactory results on RealEstate10K.

SSV. SSV is the state-of-the-art self-supervised viewpoint estimation algorithm. Leveraging multiple constraints such as cycle consistencies, SSV learns from image collections to predict camera poses of these images. We retrained SSV on the same training dataset, treating video frames as individual images. The output of SSV is a rotation angle of a single image. To obtain the relative pose, we concatenate pose predictions of the reference frame and current frame. We then fit a linear regression model to predict the true camera transformation using about 600 true poses, similar to SSV's original testing procedure. For fair comparisons, we also applied the Umeyama alignment [68].