DriveGAN: Towards a Controllable High-Quality Neural Simulation

Seung Wook Kim[1,2,3] Jonah Philion[1,2,3] Antonio Torralba[4] Sanja Fidler[1,2,3]
[1]NVIDIA [2]University of Toronto [3]Vector Institute [4] MIT
{seungwookk,jphilion,sfidler}@nvidia.com torralba@mit.edu

Abstract

Realistic simulators are critical for training and verifying robotics systems. While most of the contemporary simulators are hand-crafted, a scaleable way to build simulators is to use machine learning to learn how the environment behaves in response to an action, directly from data. In this work, we aim to learn to simulate a dynamic environment directly in pixel-space, by watching unannotated sequences of frames and their associated actions. We introduce a novel high-quality neural simulator referred to as DriveGAN that achieves controllability by disentangling different components without supervision. In addition to steering controls, it also includes controls for sampling features of a scene, such as the weather as well as the location of non-player objects. Since DriveGAN is a fully differentiable simulator, it further allows for re-simulation of a given video sequence, offering an agent to drive through a recorded scene again, possibly taking different actions. We train DriveGAN on multiple datasets, including 160 hours of real-world driving data. We showcase that our approach greatly surpasses the performance of previous data-driven simulators, and allows for new key features not explored before.

1. Introduction

The ability to *simulate* is a key component of intelligence. Consider how animals make thousands of decisions each day. Some of the decisions are critical for survival, such as deciding to step away from an approaching car. Mentally simulating the future given the current situation is key in planning successfully. In robotic applications such as autonomous driving, simulation is also a scaleable, robust and safe way of testing self-driving vehicles in safety-critical scenarios before deploying them in the real world. Simulation further allows for a fair comparison of different autonomous driving systems since one has control over the repeatability of the scenarios.

Desired properties of a good robotic simulator include accepting an action from an agent and generating a plausible next world state, allowing for user control over the scene elements, and the ability to re-simulate an observed scenario

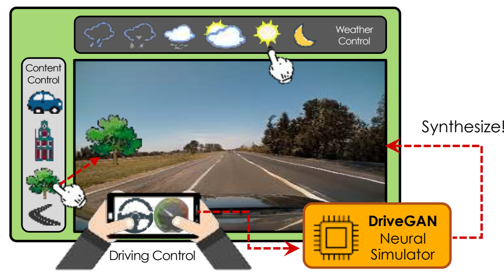

Figure 1: We aim to learn a controllable neural simulator that can generate high-fidelity real-world scenes. DriveGAN takes user controls (*e.g.* steering weel, speed) as input and renders the next screen. It allows users to control different aspects of the scene, such as weather and objects.

with plausible variations. This is no easy feat as the world is incredibly rich in situations one can encounter. Most of the existing simulators [9, 41, 30, 46] are hand-designed in a game engine, which involves significant effort in content creation, and designing complex behavior models to control non-player objects. Grand Theft Auto, one of the most realistic driving games to date, set in a virtual replica of Los Angeles, took several years to create and involved hundreds of artists and engineers. In this paper, we advocate for data-driven simulation as a way to achieve scaleability.

Data-driven simulation has recently gained attention. LidarSim [36] used a catalog of annotated 3D scenes to sample layouts into which reconstructed objects obtained from a large number of recorded drives are placed, in the quest to achieve diversity for training and testing a LIDAR-based perception system. [24, 8, 43], on the other hand, learn to synthesize road-scene 3D layouts directly from images without supervision. These works do not model the dynamics of the environment and object behaviors.

As a more daring alternative, recent works attempted to create neural simulators [27, 14] that learn to simulate the environment in response to the agent's actions directly in pixel-space by digesting large amounts of video data along with actions. This line of work provides a scaleable way to simulation, as we do not rely on any human-provided annotations, except for the agent's actions which are cheap to obtain from odometry sensors. It is also a more challenging way, since the complexity of the world and the dynamic agents acting inside it, needs to be learned in a high-resolution camera view. In this paper, we follow this route.

We introduce DriveGAN, a neural simulator that learns from sequences of video footage and associated actions taken by an ego-agent in an environment. DriveGAN leverages Variational-Auto Encoder [29] and Generative Adversarial Networks [13] to learn a latent space for images on which a dynamics engine learns the transitions within the latent space. The key aspects of DriveGAN are its disentangled latent space and high-resolution and high-fidelity frame synthesis conditioned on the agent's actions. The disentanglement property of DriveGAN gives users additional control over the environment, such as changing the weather and locations of non-player objects. Furthermore, since DriveGAN is an end-to-end differentiable simulator, we are able to re-create the scenarios observed from real video footage allowing the agent to drive again through the recorded scene but taking different actions. This property makes DriveGAN the first neural driving simulator of its kind. By learning on 160 hours of real driving data, we showcase DriveGAN to learn high-fidelity simulation, surpassing all existing neural simulators by a significant margin, and allowing for the control over the environment not possible previously.

2. Related Work

2.1. Video Generation and Prediction

As in image generation, the standard architectures for video generation are VAEs [6, 19], auto-regressive models [42, 48, 23, 55], flow-based models [31], and GANs [37, 53, 44, 45, 4, 51]. For a generator to sample videos, it must be able to generate realistic looking frames as well as realistic transitions between frames. Video prediction models [39, 34, 11, 38, 1, 33, 57] learn to produce future frames given a reference frame, and they share many similarities to video generation models. Similar architectures can be applied to the task of conditional video generation in which information such as semantic segmentation is given as input to the model [54, 35]. In this work, we use a VAE-GAN [32] based on StyleGAN [25] to learn a latent space of natural images, then train a dynamics model within the space.

2.2. Data-driven Simulation and Model-based RL

The goal of data-driven simulation is to learn simulators given observations from the environment to be simulated. Meta-Sim [24, 8] learns to produce scene parameters in a synthetic scene. LiDARSim [36] leverages deep learning and physics engine to produce LiDAR point clouds. In this work, we focus on data-driven simulators that produce future frames given controls. World Model [14] use a VAE [29] and LSTM [18] to model transition dynamics

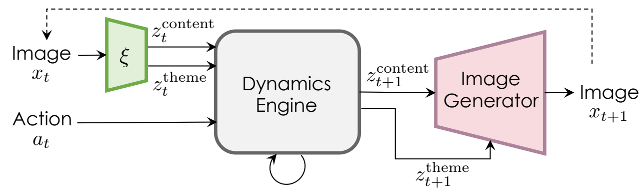

Figure 2: DriveGAN takes an image x_t and action a_t as input at time t. With encoder ξ, x_t is encoded into disentangled latent codes z_t^{theme} and z_t^{content}. Dynamics Engine learns the transition function for the latent codes given a_t. Image Generator produces x_{t+1}, which is fed to the next time step, autoregressively.

and rendering functionality. In GameGAN [27], a GAN and a memory module are used to mimic the engine behind games such as Pacman and VizDoom. Model-based RL [49, 5, 15, 22, 14] also aims at learning a dynamics model of some environment which agents can utilize to plan their actions. While prior work has applied neural simulation to simple environments [2, 50] in which a ground-truth simulator is already known, we also apply our model to real-world driving data and focus on improving the quality of simulations. Furthermore, we show how users can interatively edit scenes to create diverse simulation environments.

3. Methodology

Our objective is to learn a high-quality controllable neural simulator by watching sequences of video frames and their associated actions. We aim to achieve controllability in two aspects: 1) We assume there is an egocentric agent that can be controlled by a given action. 2) We want to control different aspects of the current scene, for example, by modifying an object or changing the background color.

Let us denote the video frame at time t as x_t and the continuous action as a_t. We learn to produce the next frame x_{t+1} given the previous frames $x_{1:t}$ and actions $a_{1:t}$. Fig 2 provides an overview of our model. Image encoder ξ produces the disentangled latent codes z^{theme} and z^{content} for x in an unsupervised manner. We define *theme* as information that does not depend on pixel locations such as the background color or weather of the scene, and *content* as spatial content (Fig 4). Dynamics Engine, a recurrent neural network, learns to produce the next latent codes z_{t+1}^{theme}, z_{t+1}^{content} given z_t^{theme}, z_t^{content}, and a_t. z_{t+1}^{theme} and z_{t+1}^{content} go through an image decoder that generates the output image.

Generating high-quality temporally-consistent image sequences is a challenging problem [31, 37, 4, 51, 54, 35]. Rather than generating a sequence of frames directly, we split the learning process into two steps, motivated by World Model [14]. Sec 3.1 introduces our encoder-decoder architecture that is pre-trained to produce the latent space for images. We propose a novel architecture that disentangles themes and content while achieving high-quality generation by leveraging a Variational Auto-Encoder (VAE) and Gen-

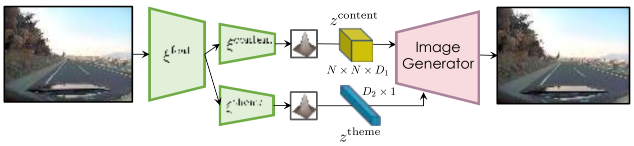

Figure 3: Pretraining stage learns the encoder and decoder for images. The encoder ξ produces z^{content} and z^{theme} which comprise the disentangled latent space that the dynamics engine trains on. The gaussian blocks represent reparameterization steps [29].

erative Adversarial Networks (GAN). Sec 3.2 describes the Dynamics Engine that learns the latent space dynamics. We also show how the Dynamics Engine further disentangles action-dependent and action-independent content.

3.1. Pre-trained Latent Space

We build our image decoder on top of the popular StyleGAN [25, 26], but make several modifications that allow for theme-content disentanglement. Since extracting the GAN's latent code that corresponds to an input image is not trivial, we introduce an encoder ξ that maps an image x into its latent code z. We utilize the VAE formulation, particularly the β-VAE [17] to control the KL term better. Therefore, on top of the adversarial losses from StyleGAN, we add the following loss at each step of generator training:

$$L_{VAE} = E_{z\sim q(z|x)}[log(p(x|z))] + \beta KL(q(z|x)||p(z))$$

where $p(z)$ is the standard normal prior distribution, $q(z|x)$ is the approximate posterior from the encoder ξ, and KL is the Kullback-Leibler divergence. For the reconstruction term, we reduce the perceptual distance [58] between the input and output images rather than the pixel-wise distance.

This form of combining VAE and GAN has been explored before [32]. To achieve our goal of controllable simulation, we introduce several novel modifications to the encoder and decoder. Firstly, we disentangle the *theme* and *content* of the input image. Our encoder ξ is composed of a feature extractor ξ^{feat} and two encoding heads ξ^{content} and ξ^{theme} (Figure 3). ξ^{feat} takes an image x as input and consists of several convolution layers whose output is passed to the two heads. ξ^{content} produces $z^{\text{content}} \in \mathbb{R}^{N\times N\times D_1}$ which has $N \times N$ spatial dimension. On the other hand, ξ^{theme} produces $z^{\text{theme}} \in \mathbb{R}^{D_2}$, a single vector, which controls the theme of the output image. Let us denote $z = \{z^{\text{content}}, z^{\text{theme}}\}$. Note that z^{content} and z^{theme} are matched to be from the standard normal prior by the reparametrization and training of VAE. We feed z into the StyleGAN decoder. StyleGAN controls the appearance of generated images with adaptive instance normalization ($AdaIN$) [10, 20, 12] layers after each convolution layer of its generator. $AdaIN$ applies the same scaling and bias to each spatial location of a normalized feature map:

$$AdaIN(\mathbf{m}, \alpha, \gamma) = \mathcal{A}(\mathbf{m}, \alpha, \gamma) = \alpha\frac{\mathbf{m} - \mu(\mathbf{m})}{\sigma(\mathbf{m})} + \gamma \quad (1)$$

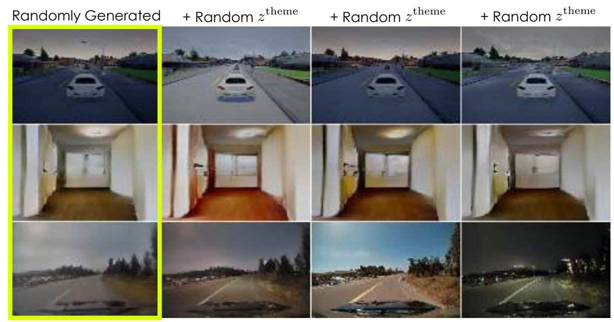

Figure 4: Left column shows randomly generated images from different environments. By sampling z^{theme}, we can change theme information such as weather while keeping the content consistent.

where $\mathbf{m} \in \mathbb{R}^{N\times N\times 1}$ is a feature map with $N \times N$ spatial dimension and α, γ are scalars for scaling and bias. Thus, $AdaIN$ layers are perfect candidates for inserting *theme* information. We pass z^{theme} through an MLP to get the scaling and bias values for each $AdaIN$ layer. Now, because of the shape of z^{content}, it naturally encodes the content information from the corresponding $N \times N$ grid locations. Rather than having a constant block as the input to the first layer as in StyleGAN, we pass z^{content} as the input. Furthermore, we can sample a new vector $v \in \mathbb{R}^{1\times 1\times D_1}$ from the normal prior distribution to swap out the content of some grid location. Preliminary experiments showed that encoding information only using the plain StyleGAN decoder is not adequate for capturing the details of scenes with multiple objects because the generator must recover spatial information from the inputs to $AdaIN$ layers, which apply the same scaling and bias to all spatial locations. We use the multi-scale multi-patch discriminator architecture [54, 21, 47], which results in higher quality images for complex scenes. We use the same adversarial losses L_{GAN} from StyleGAN, and the final loss function is $L_{pretrain} = L_{VAE} + L_{GAN}$.

We observe that balancing the KL loss with suitable β in L_{VAE} is essential. Smaller β gives better reconstruction quality, but the learned latent space could be far away from the prior, in which case the dynamics model (Sec.3.2) had a harder time learning the dynamics. This causes z to be overfit to x, and it becomes more challenging to learn the transitions between frames in the overfitted latent space.

3.2. Dynamics Engine

With the pre-trained encoder and decoder, the Dynamics Engine learns the transition between latent codes from one time step to the next given an action a_t. We fix the parameters of the encoder and decoder, and only learn the parameters of the engine. This allows us to pre-extract latent codes for a dataset before training. The training process becomes faster and significantly easier than directly working with images, as latent codes typically have dimensionality much smaller than the input. In addition, we further disentangle

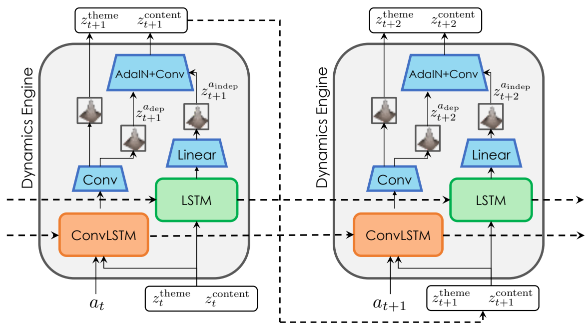

Figure 5: Dynamics Engine produces the next latent codes, given an action and previous latent codes. It disentangles content information into action-dependent and action-independent features with its two separate LSTMs. Dashed lines correspond to temporal connections. Gaussian blocks indicate reparameterization steps.

content information from z^{content} into *action-dependent* and *action-independent* features without supervision.

In a 3D environment, the view-point shifts as the ego agent moves. This shifting naturally happens spatially, so we employ a convolutional LSTM module (Figure 5) to learn the spatial transition between each time step:

$$v_t = \mathcal{F}(\mathcal{H}(h^{\text{conv}}_{t-1}, a_t, z^{\text{content}}_t, z^{\text{theme}}_t)) \tag{2}$$

$$i_t, f_t, o_t = \sigma(v^i_t), \sigma(v^f_t), \sigma(v^o_t) \tag{3}$$

$$c^{\text{conv}}_t = f_t \odot c^{\text{conv}}_{t-1} + i_t \odot \tanh(v^g_t) \tag{4}$$

$$h^{\text{conv}}_t = o_t \odot \tanh(c^{\text{conv}}_t) \tag{5}$$

where $h^{\text{conv}}_t, c^{\text{conv}}_t$ are the hidden and cell state of the convLSTM module, and i_t, f_t, o_t are the input, forget, output gates, respectively. $\mathcal{H}$ replicates a_t and z^{theme}_t spatially to match the $N \times N$ spatial dimension of z^{content}_t. It fuses all inputs by concatenating and running through a 1×1 convolution layer. $\mathcal{F}$ is composed of two 3×3 convolution layers. v_t is split into intermediate variables $v^i_t, v^f_t, v^o_t, v^g_t$. All state and intermediate variables have the same size $\mathbb{R}^{N \times N \times D_{\text{conv}}}$. The hidden state h^{conv}_t goes through two separate convolution layers to produce z^{theme}_{t+1} and $z^{a_{\text{dep}}}_{t+1}$. The action dependent feature $z^{a_{\text{dep}}}_{t+1}$ is used to produce z^{content}_{t+1}, along with $z^{a_{\text{indep}}}_{t+1}$.

We also add a plain LSTM [18] module that only takes z_t as input. Therefore, this module is responsible for information that does not depend on the action a_t. The input z_t is flattened into a vector, and all variables inside this module have size $\mathbb{R}^{D_{\text{linear}}}$. The hidden state goes through a linear layer that outputs $z^{a_{\text{indep}}}_{t+1}$. Finally, $z^{a_{\text{dep}}}_{t+1}$ and $z^{a_{\text{indep}}}_{t+1}$ are used as inputs to two $AdaIN$ + Conv blocks.

$$\boldsymbol{\alpha}, \boldsymbol{\beta} = MLP(z^{a_{\text{indep}}}_{t+1}) \tag{6}$$

$$z^{\text{content}}_{t+1} = \mathcal{C}(\mathcal{A}(\mathcal{C}(\mathcal{A}(z^{a_{\text{dep}}}_{t+1}, \boldsymbol{\alpha}, \boldsymbol{\beta})), \boldsymbol{\alpha}, \boldsymbol{\beta})) \tag{7}$$

where we denote convolution and $AdaIN$ layers as $\mathcal{C}$ and $\mathcal{A}$, respectively. An MLP is used to produce $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$. We reparameterize $z^{a_{\text{dep}}}$,$z^{a_{\text{indep}}}$, z^{theme} into the standard normal distribution $N(0, I)$ which allows sampling at test time:

$$z = \mu + \epsilon\sigma, \quad \epsilon \sim N(0, I) \tag{8}$$

where μ and σ are the intermediate variables for the mean and standard deviation for each reparameterization step.

Intuitively, $z^{a_{\text{indep}}}$ is used as $style$ for the spatial tensor $z^{a_{\text{dep}}}$ through $AdaIN$ layers. $z^{a_{\text{indep}}}$ does not get action information, so it alone cannot learn to generate plausible next frames. This architecture thus allows disentangling action-dependent features such as the layout of a scene from action-independent features such as object types. Note that the engine could ignore $z^{a_{\text{indep}}}$ and only use $z^{a_{\text{dep}}}$ to learn dynamics. If we keep the model size small and use a high KL penalty on the reparameterized variables, it will utilize full model capacity and make use of $z^{a_{\text{indep}}}$. We can also enforce disentanglement between $z^{a_{\text{indep}}}$ and $z^{a_{\text{dep}}}$ using an adversarial loss [7]. In practice, we found that our model was able to disentangle information well without such a loss.

Training: We extend the training procedure of GameGAN [27] in latent space to train our model with adversarial and VAE losses. Our adversarial losses L_{adv} come from two networks: 1) single latent discriminator, and 2) temporal action-conditioned discriminator. We first flatten z_t into a vector with size $\mathbb{R}^{N^2 D_1 + D_2}$. The single latent discriminator is an MLP that tries to discriminate produced z_t from the real latent codes. The temporal action-conditioned discriminator is implemented as a temporal convolution network such that we apply filters in the temporal dimension [28] where the actions a_t are fused to the temporal dimension. We also sample negative actions $\bar{a}_t$, and the job of the discriminator is to figure out if the given sequence of latent codes is realistic and faithful to the given action sequences. We use the temporal discriminator features to reconstruct the input action sequence and reduce the action reconstruction loss L_{action} to help the dynamics engine to be faithful to the given actions. Finally, we add latent code reconstruction loss L_{latent} so that the generated z_t matches the ground truth latent codes, and reduce the KL penalty L_{KL} for $z^{a_{\text{dep}}}_t$,$z^{a_{\text{indep}}}_t$, z^{theme}_t. The final loss function is $L_{DE} = L_{adv} + L_{latent} + L_{action} + L_{KL}$. Our model is trained with 32 time-steps with a warm-up phase similar to GameGAN. Further details are provided in the Appendix.

3.3. Differentiable Simulation

One compelling aspect of DriveGAN is that it can create an editable simulation environment from a real video. As DriveGAN is fully differentiable, it allows for recovering the scene and scenario by discovering the underlying factors of variations that comprise a video, while also recovering the actions that the agent took, if these are not provided. We refer to this as *differentiable simulation*. Once these parameters are discovered, the agent can use DriveGAN to

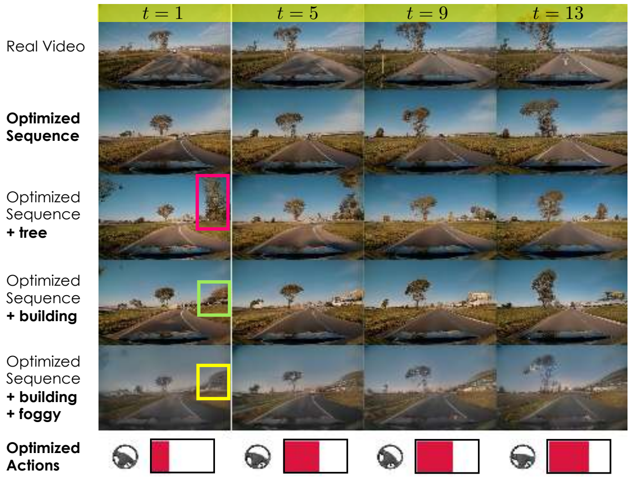

Figure 6: **Differentiable simulation**: We can first optimize for the underlying sequence of inputs that can reproduce a real video. With its controllability, we can replay the same scenario with modified content or scene condition.

re-simulate the scene and take different actions. DriveGAN further allows sampling and modification of various components of a scene, thus testing the agent in the same scenario under different weather conditions or objects.

First, note that reparametrization steps (Eq. 8) involve a stochastic variable ϵ which gives stochasticity in a simulation to produce diverse future scenarios. Given a sequence of frames from a real video $x_0, ..., x_T$, our model can be used to find the underlying $a_0, ..., a_{T-1}, \epsilon_0, ...\epsilon_{T-1}$:

$$\underset{a_{0..T-1}, \epsilon_{0..T-1}}{\text{minimize}} \sum_{t=1}^{T} ||z_t - \hat{z}_t|| + \lambda_1 ||a_t - a_{t-1}|| + \lambda_2 ||\epsilon_t|| \quad (9)$$

where z_t is the output of our model, $\hat{z}_t$ is the encoding of x_t with the encoder, and λ_1, λ_2 are hyperparameters for regularizers. We add action regularization assuming the action space is continuous and a_t does not differ significantly from a_{t-1}. To prevent the model from utilizing ϵ_t to explain all differences between frames, we also add the ϵ regularizer.

4. Experiments

We perform thorough quantitative (Sec 4.1) and qualitative (Sec 4.2) experiments on the following datasets.

Carla [9] simulator is an open-source simulator for autonomous driving research. We use five towns in Carla to generate the dataset. The ego-agent and other vehicles are randomly placed and use random policy to drive in the environment. Each sequence has a randomly sampled weather condition and consists of 80 frames sampled at 4Hz. 48K sequences are extracted, and 43K are used for training.

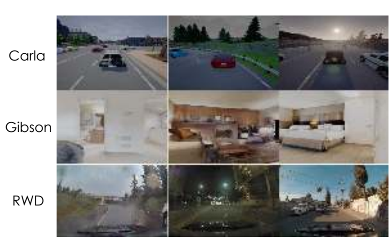

Figure 7: Image samples from three datasets studied in this work, for simulated and real-world driving, and indoor navigation.

Gibson environment [56] virtualizes real-world indoor buildings and has an integrated physics engine with which virtual agents can be controlled. We first train a reinforcement learning agent that can navigate towards a given destination coordinate. In each sequence, we randomly place the agent in a building and sample a destination. 85K sequences each with 30 frames are extracted from 100 indoor environments, and 76K sequences are used for training.

Real World Driving (RWD) data consists of real-world recordings of human driving on multiple different highways and cities. It was collected in a variety of different weather and times. RWD is composed of 128K sequences each with 36 frames extracted at 8Hz. It corresponds to $\sim$ 160 hours of driving, and we use 125K sequences for training.

Figure 7 illustrates scenes from the datasets. Each sequence consists of the extracted frames (256$\times$256) and the actions the ego agent takes at each time step. The 2-dim actions consist of the agent's speed and angular velocity.

4.1. Quantitative Results

The quality of simulators needs to be evaluated in two aspects. The generated videos from simulators have to look realistic, and their distribution should match the distribution of the real videos. They also need to be faithful to the action sequences used to produce them. This is essential to be useful for downstream tasks, such as training a robot. Therefore, we use two automatic metrics to measure the performance of models. The experiments are carried out by using the first frames and action sequences of the test set. The remaining frames are generated autoregressively.

We compare with four baseline models: Action-RNN [3] is a simple action-conditioned RNN model trained with reconstruction loss on the pixel space, Stochastic Adversarial Video Prediction (SAVP) [33] and GameGAN [27] are trained with adversarial loss along with reconstruction loss on the pixel space, World Model [14] trains a vision model based on VAE and an RNN based on mixture density networks (MDN-RNN). World Model is similar to our model as they first extract latent codes and learn MDN-RNN on top of the learned latent space. However, their VAE is not powerful enough to model the complexities of the datasets studied in this work. Fig 9 shows how a simple VAE cannot reconstruct the inputs; thus, the plain World Model cannot produce realistic video sequences by default. Therefore, we include a variant, denoted as World Model*, that uses our proposed latent space to train the MDN-RNN component.

We also conduct human evaluations with Amazon Mechanical Turk. For 300 generated sequences from each

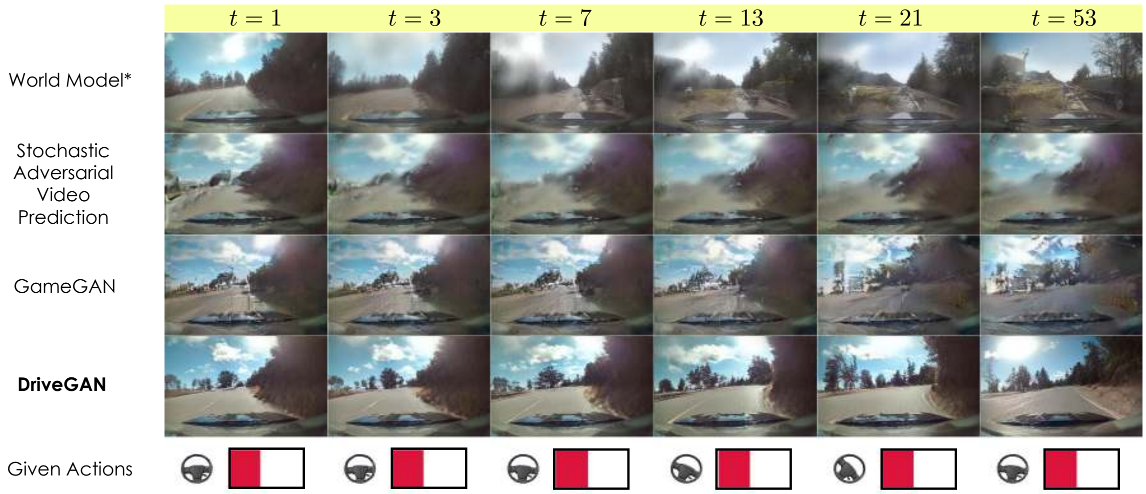

Figure 8: Comparison of baseline models. All models are given the same initial screen and sequence of actions. Our model can produce a high-quality temporally consistent simulation that conforms to the action sequence.

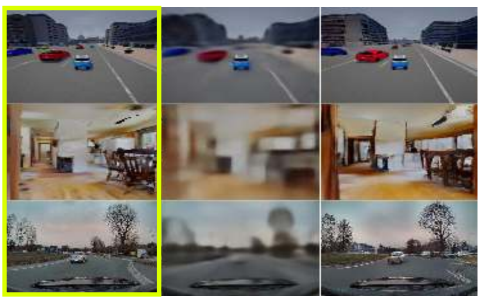

Figure 9: **Left:** original images, **Middle:** reconstructed images from VAE, **Right:** reconstructed images from our encoder-decoder model.

dataset, we show one video from our model and one video from a baseline model for the same test data. The workers are asked to mark their preferences on ours versus the baseline model on visual qulity and action consistency (Fig 10).

Video Quality: Tab 1 shows the result on Fréchet Video Distance (FVD) [52]. FVD measures the distance between the distributions of the ground truth and generated video sequences. FVD is an extension of FID [16] for videos and is suitable for measuring the quality of generated videos. Our model achieves lower FVD than all baseline models except for GameGAN on Gibson. The primary reason we suspect is that our model on Gibson sometimes slightly changes the brightness. In contrast, GameGAN, being a model directly learned on pixel space, produced more consistent brightness. Human evaluation of visual quality (Fig 10) shows that subjects strongly prefer our model, even for Gibson.

Action Consistency: We measure if generated sequences conform to the input action sequences. We train a CNN model that takes two images from real videos as input and predicts the action that caused the transition between them. The model is trained by reducing the mean-squared error loss between the predicted and input actions. The trained model can be applied to the generated sequences from simulator models to evaluate action consistency. Table 2 and human evaluation (Fig 10) show that our model achieves the best performance on all datasets.

Model	Frechet Video Distance ↓ Carla	Gibson	RWD
Action-RNN	1523.3	1109.2	2560.7
World Model	1663.0	1212.0	2795.6
World Model*	1138.6	561.1	591.7
SAVP	1018.2	470.7	977.9
GameGAN	739.5	**311.4**	801.0
Ours	**281.9**	360.0	**518.0**

Table 1: Results on FVD [52]. Lower is better.

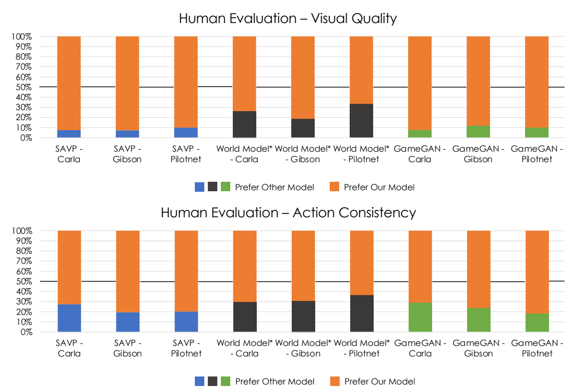

Figure 10: **Human evaluation:** Our model outperforms baseline models on both criteria.

4.2. Controllability and Differentiable Simulation

DriveGAN learns to disentangle factors comprising a scene without supervision, and it naturally allows controllability on all zs as $z^{a_{\text{dep}}}$, $z^{a_{\text{indep}}}$, z^{content} and z^{theme} can be sampled from the prior distribution. Fig 4 demonstrates how we can change the background color or weather condition by sampling and swapping z^{theme}. Fig 12 shows how sampling different $z^{a_{\text{indep}}}$ modifies the interior parts, such as ob-

Model	Action Prediction Loss ↓ Carla	Gibson	RWD
Action-RNN	4.850	0.062	0.586
World Model	5.310	0.167	0.721
World Model*	17.384	0.082	0.885
SAVP	3.178	0.070	0.645
GameGAN	2.341	0.065	0.638
Ours	**1.686**	**0.045**	**0.412**
Real Data	0.370	0.005	0.159

Table 2: Results on Action Prediction. Lower is better.

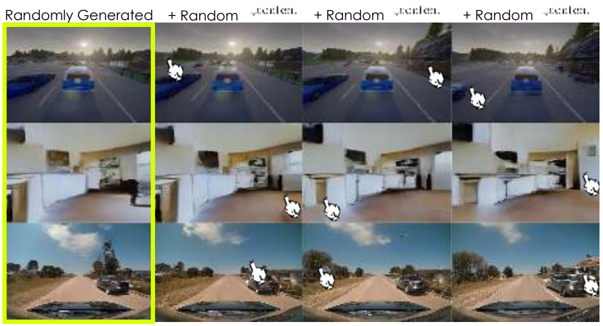

Figure 11: Users can randomly sample a vector for a grid cell in z^{theme} to change the cell's content. The white figner corresponds to the locations a user clicked to modify.

Figure 12: Swapping $z^{a_{\text{indep}}}$ modifies objects in a scene while keeping layout, such as the shape of the road, consistent. **Top:** right turn, **Middle:** road for slight left, **Bottom:** straight road.

ject shapes, while keeping the layout and theme consistent. This allows users to sample various scenarios for specific layout shapes. As z^{content} is a spatial tensor, we can sample each grid cell to change the content of the cell. In the bottom row of Fig 11, a user clicks specific locations to erase a tree, add a tree, and add a building.

We also record the sampled zs corresponding to specific content and build an editable neural simulator, as in Fig 1. This editing procedure lets users create unique simulation scenarios and selectively focus on the ones they want. Note that we can even sample the first screen, unlike some previous works such as GameGAN [27].

Differentiable Simulation: Sec 3.3 introduces how we can create an editable simulation environment from a real video by recovering the underlying actions a and stochastic

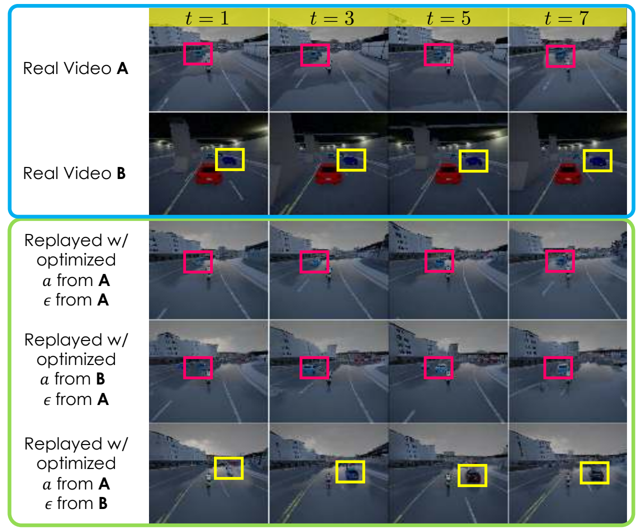

Figure 13: We optimize action ($a^A_{0..T-1}, a^B_{0..T-1}$) and stochastic variable sequences ($\epsilon^A_{0..T-1}, \epsilon^B_{0..T-1}$) for real videos A and B. Let z^A_0 be the latent code of A's initial frame. We show re-played sequences using (z^A_0, a^A, ϵ^A), (z^A_0, a^B, ϵ^A) and (z^A_0, a^A, ϵ^B).

variables ϵ with Eq.(9). Fig 13 illustrates the result of differentiable simulation. The third row exhibits how we can recover the original video A by running DriveGAN with optimized a and ϵ. To verify we have recovered a successfully and not just overfitted using ϵ, we evaluate the quality of optimized a from test data using the Action Prediction loss from Tab 2. Optimized a results in a loss of 1.91 and 0.57 for Carla and RWD, respectively. These numbers are comparable to Tab 2 and much lower than the baseline performances of 3.64 and 1.01, calculated with the mean of actions from the training data, demonstrating that DriveGAN can recover unobserved actions successfully. We can even recover a and ϵ for non-existing intermediate frames. That is, we can do *frame interpolation* to discover in-between frames given a reference and a future frame. If the time between the two frames is small, even a naive linear interpolation could work. However, for a large gap ($\geq$ 1 second), it is necessary to reason about the environment's dynamics to properly interpolate objects in a scene. We modify Eq.(9) to minimize the reconstruction term for the last frame z_T only, and add a regularization $||z_t - z_{t-1}||$ on the intermediate zs. Fig 14 shows the result. Top row, which shows interpolation in the latent space, produces reasonable in-between frames, but if inspected closely, we can see the transition is unnatural (*e.g.* a tree appears out of nowhere). On the contrary, with differentiable simulation, we can see how it learns to utilize the dynamics of DriveGAN to produce plausible transitions between frames. In Fig 15, we calculate the action prediction loss with optimized actions from frame interpolation. We discover optimized actions that follow the ground-truth actions closely when we interpolate frames one second apart. As the interpolation interval becomes larger, the loss increases since many possible action

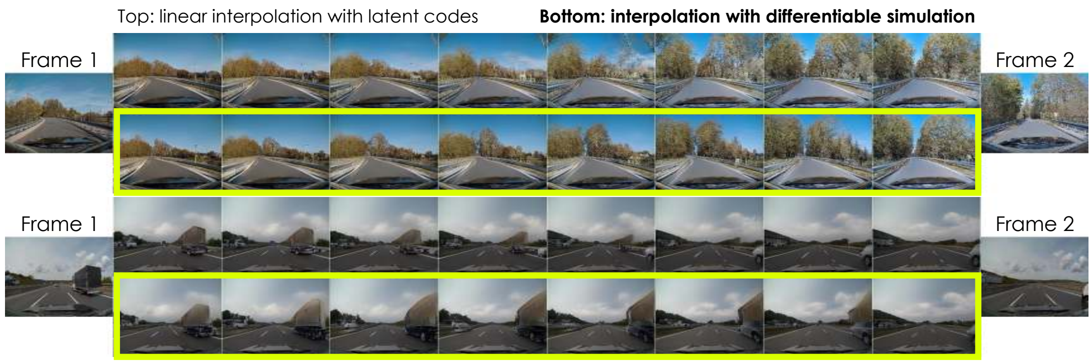

Figure 14: **Frame Interpolation** We run differentiable simulation to produce Frame 2 given Frame 1. **Top:** Linear interpolation in latent space does not account for transition dynamics correctly. **Bottom:** DriveGAN keeps dynamics consistent with respect to the environment.

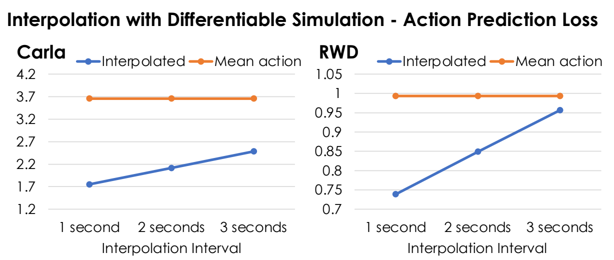

Figure 15: Optimized actions from frame interpolation discovers in-between actions. Mean action measures action prediction loss when the mean of actions from the training dataset is used as input.

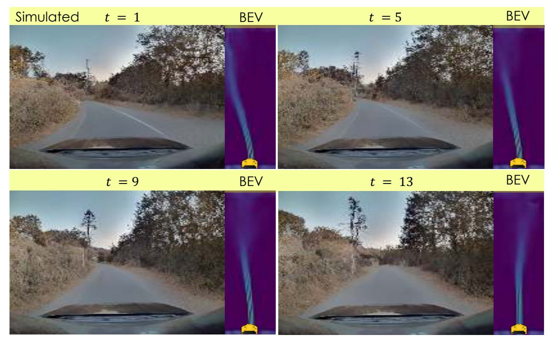

Figure 16: Bird's-Eve-View (BEV) lane prediction with Lift-Splat [40] model on generated scenes.

sequences lead to the same resulting frame. This shows the possibility of using differentiable simulation for video compression as it can decode missing intermediate frames.

Differentiable simulation also allows replaying the same scenario with different inputs. In Fig 13, we get optimized a^A, ϵ^A and a^B, ϵ^B for two driving videos, A and B. We replay starting with the encoded first frame z_0^A of A. On the fourth row, ran with (a^B,ϵ^A), we see that a vehicle is placed at the same location as A, but since we use the slightly-left action sequence a^B, the ego agent changes the lane and slides toward the vehicle. The fifth row, replayed with (a^A,ϵ^B), shows the same ego-agent's trajectory as A, but it puts a vehicle at the same location as B due to ϵ^B. This effectively shows that we can *blend-in* two different scenarios together. Furthermore, we can modify the content and run a simulation with the environment inferred from a video. In Fig 6, we create a simulation environment from a RWD test data, and replay with modified objects and weather.

4.3. Additional Experiments

LiftSplat [40] proposed a model for producing the Bird's-Eye-View (BEV) representation of a scene from camera images. We use LiftSplat to get BEV lane predictions from a simulated sequence from DriveGAN (Fig 16). Simulated scenes are realistic enough for LiftSplat to produce accurate predictions. This shows the potential of DriveGAN being used with other perception models to be useful for downstream tasks such as training an autonomous driving agent. Furthermore, in real-time driving, LiftSplat can potentially employ DriveGAN's simulated frames as a safety measure to be robust to sudden camera drop-outs.

Plain StyleGAN latent space: StyleGAN [26] proposes an optimization scheme to project images into their latent codes without an encoder. The projection process optimizes each image and requires significant time (∼19200 GPU hours for Gibson). Therefore, we use 25% of Gibson data to compare with the projection approach. We train the same dynamics model on top of the projected and proposed latent spaces. The projection approach resulted in FVD of **636.8** with the action prediction loss of **0.225**, whereas ours achieved **411.9** (FVD) and **0.050** (action prediction loss).

5. Conclusion

We proposed DriveGAN for a controllable high-quality simulation. DriveGAN leverages a novel encoder and an image GAN to produce a latent space on which the proposed dynamics engine learns the transitions between frames. DriveGAN allows sampling and disentangling of different components of a scene without supervision. This lets users interactively edit scenes during a simulation and produce unique scenarios. We showcased *differentiable simulation* which opens up promising ways for utilizing real-world videos to discover the underlying factors of variations and train robots in the re-created environments.

References

[1] Mohammad Babaeizadeh, Chelsea Finn, Dumitru Erhan, Roy H. Campbell, and Sergey Levine. Stochastic variational video prediction. *CoRR*, abs/1710.11252, 2017. 2

[2] Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013. 2

[3] Silvia Chiappa, Sébastien Racaniere, Daan Wierstra, and Shakir Mohamed. Recurrent environment simulators. *arXiv preprint arXiv:1704.02254*, 2017. 5

[4] Aidan Clark, Jeff Donahue, and Karen Simonyan. Efficient video generation on complex datasets. *CoRR*, abs/1907.06571, 2019. 2

[5] Marc Deisenroth and Carl E Rasmussen. Pilco: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on machine learning (ICML-11)*, pages 465–472, 2011. 2

[6] Emily Denton and Rob Fergus. Stochastic video generation with a learned prior. *CoRR*, abs/1802.07687, 2018. 2

[7] Emily L Denton et al. Unsupervised learning of disentangled representations from video. In *Advances in neural information processing systems*, pages 4414–4423, 2017. 4

[8] Jeevan Devaranjan, Amlan Kar, and Sanja Fidler. Meta-sim2: Unsupervised learning of scene structure for synthetic data generation. *arXiv preprint arXiv:2008.09092*, 2020. 1, 2

[9] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16, 2017. 1, 5

[10] Vincent Dumoulin, Jonathon Shlens, and Manjunath Kudlur. A learned representation for artistic style. *arXiv preprint arXiv:1610.07629*, 2016. 3

[11] Chelsea Finn, Ian Goodfellow, and Sergey Levine. Unsupervised learning for physical interaction through video prediction. In *Advances in neural information processing systems*, pages 64–72, 2016. 2

[12] Golnaz Ghiasi, Honglak Lee, Manjunath Kudlur, Vincent Dumoulin, and Jonathon Shlens. Exploring the structure of a real-time, arbitrary neural artistic stylization network. *arXiv preprint arXiv:1705.06830*, 2017. 3

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

[14] David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. In *Advances in Neural Information Processing Systems*, pages 2450–2462, 2018. 1, 2, 5

[15] Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *International Conference on Machine Learning*, pages 2555–2565. PMLR, 2019. 2

[16] 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 *Advances in neural information processing systems*, pages 6626–6637, 2017. 6

[17] 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. 3

[18] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 2, 4

[19] Jun-Ting Hsieh, Bingbin Liu, De-An Huang, Li Fei-Fei, and Juan Carlos Niebles. Learning to decompose and disentangle representations for video prediction. *CoRR*, abs/1806.04166, 2018. 2

[20] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1501–1510, 2017. 3

[21] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017. 3

[22] Lukasz Kaiser, Mohammad Babaeizadeh, Piotr Milos, Blazej Osinski, Roy H Campbell, Konrad Czechowski, Dumitru Erhan, Chelsea Finn, Piotr Kozakowski, Sergey Levine, et al. Model-based reinforcement learning for atari. *arXiv preprint arXiv:1903.00374*, 2019. 2

[23] Nal Kalchbrenner, Aäron van den Oord, Karen Simonyan, Ivo Danihelka, Oriol Vinyals, Alex Graves, and Koray Kavukcuoglu. Video pixel networks. *CoRR*, abs/1610.00527, 2016. 2

[24] Amlan Kar, Aayush Prakash, Ming-Yu Liu, Eric Cameracci, Justin Yuan, Matt Rusiniak, David Acuna, Antonio Torralba, and Sanja Fidler. Meta-sim: Learning to generate synthetic datasets. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4551–4560, 2019. 1, 2

[25] 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, 3

[26] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8110–8119, 2020. 3, 8

[27] Seung Wook Kim, Yuhao Zhou, Jonah Philion, Antonio Torralba, and Sanja Fidler. Learning to simulate dynamic environments with gamegan. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1231–1240, 2020. 1, 2, 4, 5, 7

[28] Yoon Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014. 4

[29] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2014. 2, 3

[30] Eric Kolve, Roozbeh Mottaghi, Daniel Gordon, Yuke Zhu, Abhinav Gupta, and Ali Farhadi. Ai2-thor: An interactive 3d environment for visual ai. In *arXiv:1712.05474*, 2017. 1

[31] Manoj Kumar, Mohammad Babaeizadeh, Dumitru Erhan, Chelsea Finn, Sergey Levine, Laurent Dinh, and Durk Kingma. Videoflow: A flow-based generative model for video. *CoRR*, abs/1903.01434, 2019. 2

[32] Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, and Ole Winther. Autoencoding beyond pixels using a learned similarity metric. *CoRR*, abs/1512.09300, 2015. 2, 3

[33] Alex X Lee, Richard Zhang, Frederik Ebert, Pieter Abbeel, Chelsea Finn, and Sergey Levine. Stochastic adversarial video prediction. *arXiv preprint arXiv:1804.01523*, 2018. 2, 5

[34] William Lotter, Gabriel Kreiman, and David Cox. Deep predictive coding networks for video prediction and unsupervised learning. *arXiv preprint arXiv:1605.08104*, 2016. 2

[35] Arun Mallya, Ting-Chun Wang, Karan Sapra, and Ming-Yu Liu. World-consistent video-to-video synthesis, 2020. 2

[36] Sivabalan Manivasagam, Shenlong Wang, Kelvin Wong, Wenyuan Zeng, Mikita Sazanovich, Shuhan Tan, Bin Yang, Wei-Chiu Ma, and Raquel Urtasun. Lidarsim: Realistic lidar simulation by leveraging the real world. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11167–11176, 2020. 1, 2

[37] Michael Mathieu, Camille Couprie, and Yann LeCun. Deep multi-scale video prediction beyond mean square error, 2016. 2

[38] 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

[39] Junhyuk Oh, Xiaoxiao Guo, Honglak Lee, Richard L Lewis, and Satinder Singh. Action-conditional video prediction using deep networks in atari games. In *Advances in neural information processing systems*, pages 2863–2871, 2015. 2

[40] Jonah Philion and Sanja Fidler. Lift, splat, shoot: Encoding images from arbitrary camera rigs by implicitly unprojecting to 3d. *arXiv preprint arXiv:2008.05711*, 2020. 8

[41] Xavier Puig, Kevin Ra, Marko Boben, Jiaman Li, Tingwu Wang, Sanja Fidler, and Antonio Torralba. Virtualhome: Simulating household activities via programs. In *CVPR*, 2018. 1

[42] Marc'Aurelio Ranzato, Arthur Szlam, Joan Bruna, Michaël Mathieu, Ronan Collobert, and Sumit Chopra. Video (language) modeling: a baseline for generative models of natural videos. *CoRR*, abs/1412.6604, 2014. 2

[43] Nataniel Ruiz, Samuel Schulter, and Manmohan Chandraker. Learning to simulate. *arXiv preprint arXiv:1810.02513*, 2018. 1

[44] Masaki Saito, Eiichi Matsumoto, and Shunta Saito. Temporal generative adversarial nets with singular value clipping, 2017. 2

[45] M. Saito and Shunta Saito. Tganv2: Efficient training of large models for video generation with multiple subsampling layers. *ArXiv*, abs/1811.09245, 2018. 2

[46] Shital Shah, Debadeepta Dey, Chris Lovett, and Ashish Kapoor. Aerial Informatics and Robotics platform. Technical Report MSR-TR-2017-9, Microsoft Research, 2017. 1

[47] Tamar Rott Shaham, Tali Dekel, and Tomer Michaeli. Singan: Learning a generative model from a single natural image. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4570–4580, 2019. 3

[48] Nitish Srivastava, Elman Mansimov, and Ruslan Salakhutdinov. Unsupervised learning of video representations using lstms. *CoRR*, abs/1502.04681, 2015. 2

[49] Richard S Sutton. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Machine learning proceedings 1990*, pages 216–224. Elsevier, 1990. 2

[50] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012. 2

[51] Sergey Tulyakov, Ming-Yu Liu, Xiaodong Yang, and Jan Kautz. Mocogan: Decomposing motion and content for video generation. *CoRR*, abs/1707.04993, 2017. 2

[52] Thomas Unterthiner, Sjoerd van Steenkiste, Karol Kurach, Raphael Marinier, Marcin Michalski, and Sylvain Gelly. Towards accurate generative models of video: A new metric & challenges. *arXiv preprint arXiv:1812.01717*, 2018. 6

[53] Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. Generating videos with scene dynamics, 2016. 2

[54] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Guilin Liu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. Video-to-video synthesis. *CoRR*, abs/1808.06601, 2018. 2, 3

[55] Dirk Weissenborn, Oscar Täckström, and Jakob Uszkoreit. Scaling autoregressive video models. *CoRR*, abs/1906.02634, 2019. 2

[56] Fei Xia, Amir R. Zamir, Zhi-Yang He, Alexander Sax, Jitendra Malik, and Silvio Savarese. Gibson env: real-world perception for embodied agents. In *Computer Vision and Pattern Recognition (CVPR), 2018 IEEE Conference on*. IEEE, 2018. 5

[57] Wei Yu, Yichao Lu, Steve Easterbrook, and Sanja Fidler. Efficient and information-preserving future frame prediction and beyond. In *ICLR*, 2020. 2

[58] 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. 3