Playable Environments: Video Manipulation in Space and Time

Willi Menapace*
University of Trento

Stéphane Lathuilière†
LTCI, Télécom Paris
Institut Polytechnique de Paris

Aliaksandr Siarohin
University of Trento

Christian Theobalt†
MPI for Informatics, SIC

Sergey Tulyakov†
Snap Inc.

Vladislav Golyanik†
MPI for Informatics, SIC

Elisa Ricci†
University of Trento
Fondazione Bruno Kessler

Abstract

We present Playable Environments—a new representation for interactive video generation and manipulation in space and time. With a single image at inference time, our novel framework allows the user to move objects in 3D while generating a video by providing a sequence of desired actions. The actions are learnt in an unsupervised manner. The camera can be controlled to get the desired viewpoint. Our method builds an environment state for each frame, which can be manipulated by our proposed action module and decoded back to the image space with volumetric rendering. To support diverse appearances of objects, we extend neural radiance fields with style-based modulation. Our method trains on a collection of various monocular videos requiring only the estimated camera parameters and 2D object locations. To set a challenging benchmark, we introduce two large scale video datasets with significant camera movements. As evidenced by our experiments, playable environments enable several creative applications not attainable by prior video synthesis works, including playable 3D video generation, stylization and manipulation[1].

1. Introduction

What would you change in the last tennis match you saw? The actions of the player? The style of the field, or, perhaps, the camera trajectory to observe a highlight more dramatically? To do so interactively, the geometry and the style of the field and the players need to be reconstructed. Players' actions need to be understood and the outcomes of future actions anticipated. To enable these features one needs to reconstruct the observed *environment* in 3D and provide simple and intuitive interaction, offering an experience similar to *playing* a video game. We call these representations Playable Environments (PE).

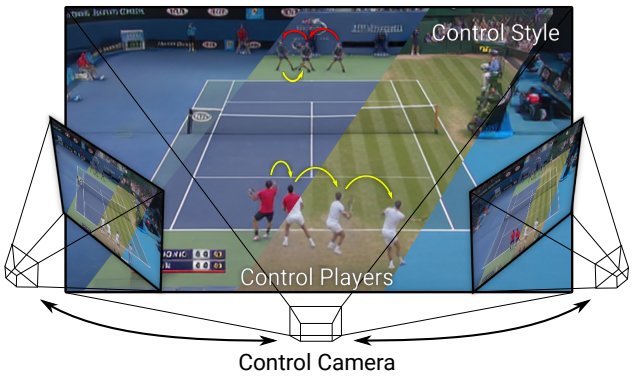

Figure 1. Given a single initial frame, our method creates *playable environments* that allow the user to interactively generate different videos by specifying discrete actions to control players, manipulating the camera trajectory and indicating the style for each object in the scene.

Such a representation enables multiple creative applications, such as 3D- and action-aware video editing, camera trajectory manipulation, changing the action sequence, the agents and their styles, or continuing the video in time, beyond the observed footage. Fig. 1 shows a playable environment for tennis matches. In it, the user specifies actions to move the players, controls the viewpoint and changes the style of the players and the field. The environment can be played, akin to a video game, but with real objects.

In this work, we propose a method to construct PEs of complex scenes that supports a large set of interactive manipulations. Trained on a dataset of monocular videos, our method presents six core characteristics listed in Tab. 1 that enable the creation of such PEs. Our framework allows the user to interactively generate videos by providing discrete actions $\langle \mathbf{1} \rangle$ and controlling the camera pose $\langle \mathbf{2} \rangle$. Further-

*This work was partially done while interning at MPI for Informatics
†Equal senior contribution
[1] willi-menapace.github.io/playable-environments-website

	Name	Description
⟨**1**⟩	*Playability*	The user can control generation with discrete actions.
⟨**2**⟩	*Camera control*	The camera pose is explicitly controlled at test time.
⟨**3**⟩	*Multi-object*	Each object is explicitly modeled.
⟨**4**⟩	*Deformable objects*	The model handles deformable object such as human bodies
⟨**5**⟩	*Appearance changes*	The model handles objects whose appearance is not constant is the training set
⟨**6**⟩	*Robustness*	The model is robust to calibration and localization errors.

Table 1. Characteristics of our method for Playable Environments. Each row is referred in the text with ⟨·⟩ symbols.

more, it can represent environments with multiple objects ⟨**3**⟩ with varying poses ⟨**4**⟩ and appearances ⟨**5**⟩ and is robust to imprecise inputs ⟨**6**⟩. In particular, we do not require ground-truth camera intrinsics and extrinsincs, but assume they can be estimated for each frame. Neither do we assume ground-truth object locations, but rely on an off-the-shelf object detector [27] to locate the agents in 2D, such as both tennis players. No other supervision is required.

Playable Environments encapsulate and extend representations built by several prior image or video manipulation methods. Novel view synthesis and volumetric rendering methods support re-rendering of static scenes. However, while some methods support moving or articulated objects [24, 26, 34, 39], it is challenging for them to handle dynamic environments and they do not allow user interaction, making them undesirable for modeling compelling environments. Video synthesis methods manipulate videos by predicting future frames [15, 16, 33, 35], animating [30–32] or playing videos [18], but environments modeled with such methods typically lack camera control and multi-object support. Consequently, these methods limit interactivity as they do not take into account the 3D nature of the environment.

Our method consists of two components. The first one is the synthesis module. It extracts the state of the environment—location, style and non-rigid pose of each object—and renders the state back to the image space. Recently introduced Neural Radiance Fields (NeRFs) [19] represent an attractive tool for their ability to render novel views. In this work, we introduce a style-based modification of NeRF to support objects of different appearances. Furthermore, we propose a compositional non-rigid volumetric rendering approach handling the rigid parts of the scene and non-rigid objects. The second component—the action module—enables playability. It takes two consecutive states of the environment and predicts an action with respect to the camera orientation. We train our framework using reconstruction losses in the image space and the state space, and a novel loss for action consistency. Finally, to improve temporal dynamics, we introduce a temporal discriminator that operates on sequences of environment states.

To thoroughly evaluate ⟨**1**−**6**⟩, we introduce two complementary large-scale datasets for the training of playable environments, a synthetic and a real one. The first is intended to evaluate ⟨**1**−**5**⟩, with a particular focus on camera control thanks to the synthetic ground truth, the second to evaluate ⟨**1**−**6**⟩, with a particular focus on ⟨**4**−**6**⟩ given the high diversity present in this dataset. We propose an extensive evaluation of our method with several baselines derived from existing NeRF and video generation methods. These experiments show that our method is able to generate high-quality videos and outperforms all baselines in terms of playability, camera control and video quality.

In summary, the primary contributions of this work are as follows: **a new framework** for the creation of compelling Playable Environments with the characteristics in Tab. 1, featuring **a new compositional NeRF** that handles deformable objects with different visual styles and an **action module** that operates in the latent space of our NeRF model; **two challenging large-scale datasets** for training and evaluating PEs to stimulate future research in this area.

2. Related Works

Video generation has seen incredible progress over past years. The video synthesis task has numerous formulations which mostly differ in the type of conditional information that is used for generation. The generation process could be conditioned on previous frames [5,16,17,35,37], on another video [30–32, 38], on the pose of the agent [2] or even be completely unconditional [28,35]. Moreover, several works proposed to condition the generation of each single frame on an action label [4, 13, 22, 23]. Still, all these methods require action supervision for training.

Playable video generation (PVG) was recently introduced in Menapace *et al.* [18]. Differently from prior works in this domain which required annotated action labels [12,13], their method, CADDY, automatically infers actions during training in a completely unsupervised manner from raw videos. This method is closely related to ours. However, CADDY assumes only a single controllable object while here we also model the camera movement, complex 3D interactions and support a variety of object appearances.

Novel view synthesis methods traditionally utilized depth maps [3, 25] or multi-view geometry [14, 29, 42] in order to reconstruct underlying 3D representation and later render new views of the corresponding scene. Recently, Neural Radiance Fields (NeRF) [19] revolutionized the field of novel view synthesis. The main idea of NeRF [19] is to model the scene as a continuous 5D function, usually represented by MLP, and directly query this function along the

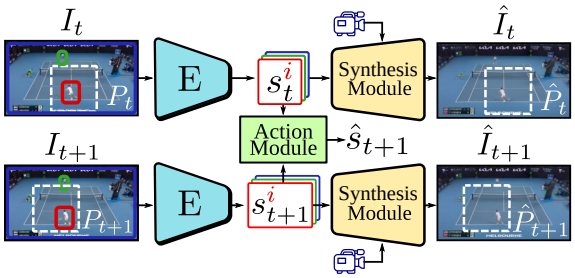

Figure 2. **Overview of our framework**. The encoder E extracts environment states for every object in the scene. The synthesis module follows a NeRF-like architecture to reconstruct the input frame and allows for camera manipulation. We introduce the action module that learns to encode state dynamics with discrete action labels. At test time, these learned action labels are provided by the user to control the generated content.

camera rays. Numerous follow-up works on [19] have been proposed. For instance, some works proposed to decompose the foreground and background [20, 40]. Other works generalised NeRF [19] to dynamic scenes [10, 24, 34, 39]. GIRAFFE [21] and GANcraft [7] proposed to utilize an internal representation that is rendered in a feature space and later decoded by a standard 2D convolutional network. However, none of these methods is able to generalize to multiple monocular videos, several moving and deforming objects, and diverse objects and scene appearances. Comparatively, our method can be trained with such data. Moreover, for enriching the interactivity of the playable environment, our method can control objects in the scene with action labels that are discovered in an unsupervised manner.

3. Method

Our framework is based on the encoder-decoder architecture shown in Fig. 2 whose design is driven by the playable environment characteristics ⟨**1**-**6**⟩ in Tab. 1. At time t, the encoder network outputs state vector s_t^i for every object i in the scene. To enable playability ⟨**1**⟩, we include an action module in the bottleneck layer that has two goals. First, it learns discrete action labels in an unsupervised manner. More precisely, we learn to discretize the transition from s_t^i to s_{t+1}^i using an action labels $a_t^i \in \{1, ..., K\}$, where the number of actions K is a hyper-parameter specified before training. Second, the action module is used at test time to condition the next frame generation on the action selected by the user. Finally, the decoder network, referred to as the synthesis module, is in charge of reconstructing the input frame combining the state of every object and the camera parameters to allow for camera control ⟨**2**⟩. The synthesis and action modules are trained in two separate phases using reconstruction as the main driving loss.

To handle environments with multiple objects ⟨**3**⟩, we adopt a compositional formulation for our encoder-decoder: we decompose the environment into a predefined set of objects. We distinguish between two object categories, namely static objects (*e.g.* background) and playable objects (*e.g.* human), where the latter are the dynamic objects the user will be able to control. We define the environment state of object i as $s_t^i = (x_t^i, w_t^i, \pi_t^i)$ where x_t^i is the position of the object in the environment, w^i is a style descriptor, and π^i is the object pose. We introduce w^i and π^i to handle deformable objects ⟨**4**⟩, such as humans, and to model appearance changes of objects in the training set ⟨**5**⟩. For every static object, we assume x_t^i to be fixed and known. For playable object i instead, given the current camera parameters and its bounding box b_t^i, we approximate x_t^i by projecting the middle point of the lower bounding box edge onto the ground plane. We then compute the style and pose descriptors using a convolutional encoder network E for each object. The encoder takes as input the image cropped at the location defined by the bounding box for each object and outputs both w_t^i and π_t^i. In the rest of the paper, we omit object indexes.

We introduce a novel synthesis module detailed in Sec 3.1. The action module is described in Sec. 3.2. The training procedures are given in Secs. 3.3 and 3.4.

3.1. Synthesis Module

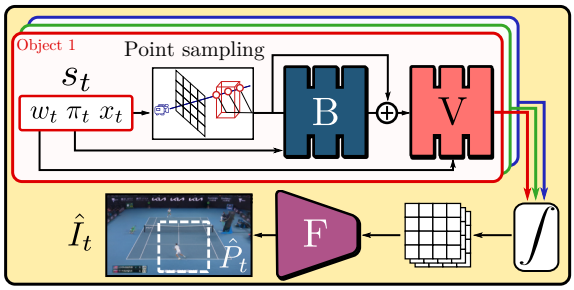

Figure 3. **The synthesis module** consists of two steps. First, non-rigid neural radiance fields with a bending network B and style modulation are used to generate a feature map. Second, the feature maps are fed to a ConvNet F.

The aim of the synthesis module is to reconstruct the input image from the camera pose and states s_t. We found NeRF [19] to be a reasonable base architecture for explicit camera control ⟨**2**⟩. Therefore, we propose a novel architecture (Fig. 3) that combines non-rigid neural radiance fields with a convolutional image generator to address ⟨**2**-**6**⟩.

Camera control ⟨**2**⟩ is achieved by employing NeRF [19] as a base architecture. Our NeRF represents scenes using a fully-connected network V, whose input is a single vector containing a point location in 3D. It outputs the volume

density σ and radiance c for the input point location. Given a desired virtual camera, 3D points are sampled along the camera ray r traced through each pixel. The color value of every pixel is computed by integration over the ray r:

$$C(r) = \int_{t_n}^{t_f} e^{-\int_{t_n}^{t} \sigma(r(s))ds} \sigma(r(t))c(r(t))dt. \quad (1)$$

Similarly to [7, 21], instead of directly predicting color values, our neural radiance fields generate feature maps for the input camera pose, while a convolutional image generator is in charge of generating realistic frames. For more details about NeRFs, please refer to the *Supp. Mat.* and [19].

Multi-object ⟨3⟩. Each object is modeled using a separate feature field parametrized as an object-specific MLP V. The field is bounded by volume β and centered at the respective object location x_t. Given a ray r, we compute its features $f(r)$ according to the following procedure. We first intersect r with each bounding volume β to compute the ingress and egress location of the ray with each object x_{in}, x_{out}. For each object, we then uniformly sample a given amount of positions $\{x_p\}_{p=1}^{N}$ between x_{in} and x_{out} and obtain the respective features f_p and opacities σ_p as $f_p, \sigma_p = V(x_p)$. $f(r)$ is obtained by integration similarly to Eq. (1).

Deformable objects ⟨4⟩. To handle deformable objects such as humans, we make use of non-rigid NeRF models, similarly to [34]. For each playable object, we introduce a ray bending network B parametrized as an MLP. Given an object pose descriptor π and position x_p on ray r, we use the bending network to regress the corresponding position $\tilde{x}_p$ on the bent ray $\tilde{r}$ as:

$$\tilde{x}_p = x_p + B(x_p, \pi_t). \quad (2)$$

We then make use of the positions on $\tilde{r}$ when sampling V. In this way, B encodes the transformation from the space of the deformed object to a canonical space and V encodes a canonical representation of the object.

Appearance changes ⟨5⟩. The appearance of each object may vary widely in the dataset. In order for each object-specific model to be able to represent the complete set of possible appearances of its object, we propose the use of a style embedding layer inspired by AdaIN [9], which we embed into V. Assuming a hidden feature h_t in V and a style code w_t, we modulate h_t as follows:

$$\tilde{h}_t = \gamma(w_t)h_t + \beta(w_t), \quad (3)$$

where γ and β are trainable linear layers. Following [19], we design V as a backbone terminated by two separate branches, one for opacity and one for features prediction. We assume that the style of an object should influence its features, but not its geometry. Therefore, we insert our modulation layer in the features prediction branch only.

Robustness ⟨6⟩ to calibration and localization errors is achieved through a Feature Renderer. Our compositional NeRF model outputs a feature map f_t corresponding to an input image patch. We employ a ConvNet F to reconstruct it. Due to the ability of ConvNets to model cross-pixel relationships, inaccuracies in the estimation of features caused by input noise can be compensated, reducing the associated blur. Note that F contains upsampling layers. It allows an important reduction in the number of rays that are to be sampled by the NeRF model since it outputs a feature map at a lower resolution than the image. Therefore, we reduce memory consumption allowing larger patches to be rendered. We also find it beneficial to use multiple input feature maps at different resolutions to capture details at different scales (see *Sup. Mat.* for details).

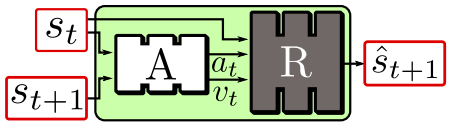

Figure 4. **The action module.** Given the states at times t and $t+1$, the action network A predicts a discrete action label a_t and action variability v_t that are combined by the dynamics network R to estimate the new environment state s_{t+1} given the old s_t.

3.2. Action Module

The action module (Fig. 4) learns the action space and enables playability ⟨1⟩. The actions of each playable object are modeled by a separate action module, consisting of the action and dynamics networks.

Action network. Given two successive environment states s_t and s_{t+1}, we use an action network A to infer a discrete representation $a_t \in \{1, ..., K\}$ of the action performed by the object in the input sequence. Following [18], to address non determinism present in the environment, we also extract an action variability embedding v_t describing the particular variation of a_t performed at time t:

$$a_t, v_t = A(s_t, s_{t+1}). \quad (4)$$

Dynamics network. The role of the dynamics network is to predict the state s_{t+1} from s_t and the action label a_t. We adopt a recurrent model R implemented as an LSTM to model the dynamics of the object. The next state prediction $\hat{s}_{t+1} = (\hat{x}_t, \hat{w}_t, \hat{\pi}_t)$ is given by:

$$\hat{s}_{t+1} = R(s_t, a_t, v_t). \quad (5)$$

In our preliminary experiments, we observe that when R directly regresses $\hat{x}_{t+1}$ as formulated in (5), the model learns actions that are independent from the current camera position. This behavior is unnatural for the user since in applications such as video games, object movements are typically expressed relatively to the camera pose. To avoid this behavior, R is instead asked to predict the object movement Δ expressed in the camera coordinate system. The estimated

position is then given by $\hat{x}_{t+1} = x_t + M\Delta$ where M is the rotation matrix expressing the orientation of the camera.

3.3. Synthesis Module Training

We train our model in two steps by first training the encoder and synthesis module until convergence, and then the action module. We train the encoder and synthesis module using the perceptual loss of Johnson *et al*. [11] that assesses image reconstruction quality in features spaces of a pretrained VGG network. The loss is computed between the ground truth and reconstructed image patches. The perceptual loss is complemented by an L2 reconstruction loss in the pixel space.

Our preliminary experiments showed that training may fail to correctly disentangle object style and pose (*i.e*. w and π respectively). Indeed, the reconstruction losses can be minimized using w alone by predicting a constant, non-deforming surface with changing style. To avoid this problem, we make the observation that the pose of an object in neighboring frames can change while the style does not. Therefore, we enforce better disentanglement by permuting the order of w codes along the temporal dimension for each sequence before feeding them to the synthesis module.

3.4. Action Module Training

In the second phase of training, we train the action module using a combination of losses. Each loss is computed separately for each playable object and then averaged to produce the final optimization objective.

Reconstruction loss. For each playable object, we reconstruct the input sequence of environment states $\{s_t\}_{t=1}^{T}$, obtained by encoding each input image using the encoder E, and impose an ℓ_2 reconstruction loss $\mathcal{L}_{\text{rec}}$ with the corresponding reconstructed sequence $\{\hat{s}_t\}_{t=1}^{T}$.

Action learning losses. We employ the information-theoretic action learning loss of [18] to foster the understanding of actions. For each playable object, the action network A produces internal estimates of action probabilities p_t and $\hat{p}_t$ for input s_t and reconstructed $\hat{s}_t$ environment states, respectively. By imposing the maximization of mutual information between these two distributions we foster the action network both to discover the K action categories, avoiding mode collapse, and to produce consistent action estimates for the input and reconstructed sequence:

$$\mathcal{L}_{\text{act}} = -\mathcal{MI}(p_t, \hat{p}_t). \tag{6}$$

In addition, to improve consistency between discrete actions and 3D movements, we propose to optimize a novel loss consisting in a soft version of the Δ Mean Squared Error (Δ-MSE) introduced in [18]. This metric is based on the idea that same actions a_t should correspond to similar object motions Δ. Assuming a batch containing J image pairs, we extract the object motion $\Delta_j, j \in \{1, ..., J\}$ and estimate the mean object motion for each action:

$$\forall k \in \{1, ..., K\}, \mu_k = \frac{\sum_{j=1}^{J} p_{jk}\Delta_j}{\sum_{j=1}^{J} p_{jk}}, \tag{7}$$

where p_{jk} denotes the probability for the image pair j to be assigned to the action k. We then minimize the mean squared distance between the motion Δ_j and the mean motion for each action:

$$\mathcal{L}_{\Delta} = \frac{1}{\text{Var}(\Delta)} \sum_{j=1}^{J}\sum_{k=1}^{K} p_{jk} \left\|\Delta_j - \mu_k\right\|_2^2, \tag{8}$$

where $\text{Var}(\Delta)$ is used as a normalization factor.

Temporal discriminator. Previous methods for playable video generation [18], tend to produce sequences where the playable objects move in the scene with unrealistic motions. We attribute this behavior to the use of reconstruction as the main training objective. Optimizing reconstruction losses does not penalize action representations that lead to temporally inconsistent videos. To address the problem, for each playable object we introduce a temporal discriminator D implemented as a 1D ConvNet over the temporal dimension. Given a sequence of environment states, the temporal discriminator is trained to classify them as real if produced by encoding the input images using E or as fake if reconstructed by the action module. We implement our adversarial training procedure using a vanilla GAN loss with loss terms $\mathcal{L}_{\text{G}}$ and $\mathcal{L}_{\text{D}}$ for the action module and temporal discriminator, respectively.

Total loss. Our optimization objective for A and R is

$$\mathcal{L} = \lambda_{\text{rec}} L_{\text{rec}} + \lambda_{\text{act}}\mathcal{L}_{\text{act}} + \lambda_{\Delta}\mathcal{L}_{\Delta} + \lambda_{\text{G}}\mathcal{L}_{\text{G}}, \tag{9}$$

where we introduce the weighting parameters λ_{rec}, λ_{act}, λ_{Δ} and λ_{G}. For training D, we minimize the adversarial objective $\mathcal{L}_{\text{D}}$ of the discriminator.

Inference. At inference time, we assume that only the first frame of the sequence is given. We use the encoder module to extract the first environment state $\hat{s}_1 = s_1$. At each timestep t, we let the user specify a discrete action for each playable object and use the dynamics network R to derive $\hat{s}_{t+1}$ in an autoregressive way. Since the action input is specified by the user, during inference we do not make use of the action network and always set $v_t = 0$. The environment states generated by the dynamics network are rendered to images using the synthesis module.

4. Experiments

Datasets. Evaluating ⟨**1**-**6**⟩ is challenging and requires video datasets featuring camera motion ⟨**2**⟩, multiple playable objects ⟨**1**,**3**⟩, deforming objects ⟨**4**⟩ and varied appearance ⟨**5**⟩. For this reason, we collect three datasets:

• *Minecraft* dataset. We collect a synthetic video dataset with duration of 1h with two sparring *Minecraft* [1] players. Wide camera movement and diverse, deforming players allow the evaluation of ⟨**1**−**5**⟩.
• *Minecraft Camera* dataset. We collect *Minecraft* [1] sequences where the camera is moved in the neighborhood of a starting position. We use these frames as a synthetic ground truth for the evaluation of camera control ⟨**2**⟩.
• *Tennis* dataset. We collect a large-scale dataset of 43 broadcast tennis matches totalling 12h of videos for the evaluation of ⟨**1**-**6**⟩. The dataset features challenging player poses ⟨**5**⟩, high variability in tennis fields and players ⟨**4**⟩ and noise in camera estimation and player localizaton ⟨**6**⟩. To allow comparison with playable video generation methods under their simplifying assumptions, we adopt the *Tennis* dataset of [18], referred to as *Static Tennis*. The dataset features limited camera movement, each video is cropped to depict only a single player, only one field is present and players have uniform appearance, thus only ⟨**1**,**4**⟩ are evaluated. The datasets are detailed in the *Supp. Mat.*.

Evaluation protocol. We perform a separate evaluation of the synthesis ⟨**2**-**6**⟩ and the action modules ⟨**1**⟩ using similar evaluation protocols. For the former, we reconstruct each test sequence by extracting the environment state of each frame and rendering the original frame back. For the action module, we follow the evaluation protocol of [18]. In particular, we consider a test sequence and extract the environment state of the first frame, then we use the action network to extract the sequence of discrete actions present in the sequence and reconstruct each frame starting from the first environment state. As video quality metrics ⟨**2**,**4**-**6**⟩ we adopt *LPIPS* [41], *FID* [8] and *FVD* [36] computed between the test sequences and the reconstructed sequences. For evaluation of the action space ⟨**1**,**3**⟩, following [18], we define Δ as the difference in position of an object between two given frames and use the following metrics:
• Δ *Mean Squared Error* (Δ-*MSE*): The expected error in terms of MSE in the regression of Δ from a discrete action. For each action, the average Δ is used as the optimal estimator. The metric is normalized by the variance of Δ.
• Δ-*based Action Accuracy* (Δ-*Acc*): The accuracy with which a discrete action can be regressed from Δ.
• *Average Detection Distance (ADD)*: The average Euclidean distance between the bounding box centers of corresponding objects in the test and reconstructed frames.
• *Missing Detection Rate (MDR)*: The portion of detections that are present in the test sequences but that are not matched by any detection in the reconstructed sequences.

4.1. Comparison on Playable Video Generation

In this section, we evaluate the action-modeling capabilities of our method by comparing against the state of the art in the related problem of Playable Video Generation (PVG) [18] where the objective is to learn a set of discrete action labels in an unsupervised fashion to condition video generation. Differently from our setting, in PVG no explicit camera control is required. Moreover, existing PVG methods assume a single user-controllable object and that camera motion is limited.

	LPIPS↓	FID↓	FVD↓	Δ-*MSE*↓	Δ-*Acc*↑	ADD↓	MDR↓
MoCoGAN [35]	0.266	132	3400	101	26.4	28.5	20.2
MoCoGAN+	0.166	56.8	1410	103	28.3	48.2	27.0
SAVP [16]	0.245	156	3270	112	19.6	10.7	19.7
SAVP+	0.104	25.2	**223**	116	33.1	13.4	19.2
CADDY [18]	0.102	**13.7**	239	72.2	45.5	**8.85**	1.01
(Ours)	**0.089**	15.3	237	**32.8**	**68.1**	9.47	**0.15**

Table 2. Comparison with PVG state of the art on the *Static Tennis* dataset of [18]. Δ-*MSE*, Δ-*Acc* and MDR in %, ADD in pixels.

To satisfy these simplifying assumptions, we adopt the *Static Tennis* dataset of [18]. Tab. 2 shows the results. Our method substantially improves the Δ-MSE and Δ-Acc action quality metrics suggesting that the learned actions are better correlated with player movement. In addition, the reduced LPIPS and MDR indicate an improvement in the quality of the generated reconstruction which is supported by a user study shown in the *Supp. Mat.*. We report qualitative results in the *Supp. Mat.*.

4.2. Comparison with Previous Methods

Baselines. We propose to build baselines for the creation of PEs from state-of-the-art methods in the related problem of PVG. We make use of the following set of versions of CADDY [18] which are modified to account for multiple playable objects and for camera motion: (i) the action network produces a distinct output for each dynamic object in the environment; (ii) (i) + the action and dynamics networks are conditioned on bounding box and camera information; (iii) (ii) + output resolution is increased to match our method; (iv) (ii) + $\mathcal{L}_\Delta$; (v) (iii) + $\mathcal{L}_\Delta$.

Playability evaluation ⟨1⟩. We evaluate player control capabilities in Tab. 3 and in the *Supp. Mat.*. On the *Tennis* dataset our model substantially improves over the baselines in the action space metrics, LPIPS and FVD, suggesting better controllability of the players. In particular, the considerably lower MDR indicates a better capacity of the model in generating players with respect to the baselines. Fig. 5 shows qualitative reconstruction results for our method. As suggested by the MDR and ADD scores, the model correctly synthesizes both players and is able to reconstruct the player movements of the ground-truth sequence using only a sequence of discrete actions. In addition, a visualization of the learned action space (see Fig. 6)

	Aux.	H.Res.	$\mathcal{L}_\Delta$	Tennis LPIPS↓	FID↓	FVD↓	Δ-*MSE*↓	Δ-*Acc*↑	ADD↓	MDR↓	Minecraft Camera LPIPS↓	FID↓	ADD↓	MDR↓
CADDY [18] (i)				0.313	61.0	877	0.901	42.6	35.1	36.9	0.747	306	11.7	95.8
CADDY [18] (ii)	✓			0.351	69.2	1109	0.592	59.6	29.0	24.8	0.762	324	44.7	92.2
CADDY [18] (iii)	✓	✓		0.213	**15.4**	727	0.693	57.5	18.7	11.7	0.669	244	29.2	82.0
CADDY [18] (iv)	✓		✓	0.445	70.3	1568	0.797	62.4	29.6	33.0	0.699	314	62.0	89.4
CADDY [18] (v)	✓	✓	✓	0.534	191	8083	0.633	73.5	20.2	60.3	0.679	337	19.1	93.6
(Ours)				**0.181**	17.4	**485**	**0.293**	**95.7**	**14.0**	**4.84**	**0.242**	**29.2**	**5.69**	**8.07**

Table 3. Playability evaluation with baselines on the *Tennis* dataset and camera control evaluation on the *Minecraft Camera* dataset. *Aux.*: use of auxiliary bounding box and camera pose information; *H.Res.*: use of the high resolution model; $\mathcal{L}_\Delta$: use of the loss for Δ-*MSE*. Δ-*MSE*, Δ-*Acc* and MDR in %, ADD in pixels.

Figure 5. Qualitative reconstruction results produced by our method on the *Tennis* and *Minecraft* datasets. In the reconstructed sequence, playable objects move according to the ground truth sequence and are rendered in realistic poses.

shows that the model learns a set of diverse discrete actions that correspond to the main movement directions.

To further evaluate the quality of the action space, we perform a user study (see *Supp. Mat.*) on the *Tennis* dataset, following the protocol of Menapace *et al.* [18]. To evaluate the consistency of learned actions, we measure user agreement using the Fleiss' kappa measure [6]. Our method achieves an agreement of 0.444, while the best baseline shows a lower agreement of 0.353.

Camera control evaluation ⟨2⟩. We evaluate the quality with which the model can synthesize novel views. We choose to perform a quantitative evaluation on the *Minecraft Camera* dataset since novel view ground truth is present.

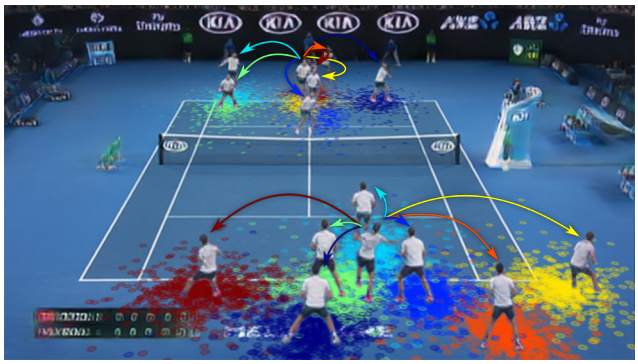

Figure 6. Action space learned by our method on the *Tennis* dataset. Each color represents a learned action and each arrow shows the effects of applying the respective action six times to the initial player. The overlay on the floor shows the distribution of possible ending positions after the application of each action.

We start from the first frame and reconstruct each sequence using the camera parameters of the novel views. Results are shown in Tab. 3. Despite the presence of auxiliary bounding box and camera pose inputs for CADDY [18], the baseline method fails in synthesizing the scene from novel perspectives. We ascribe this phenomenon to the lack of an explicit model for the camera. Our method instead can successfully synthesize the scene from novel camera perspectives.

In Fig. 7 we show qualitative camera and style manipulation results for our method on the *Tennis* dataset. Our model can synthesize the scene under novel views and correctly alter the style of the field and players to the one of a target image. We present additional camera and style manipulation results in the *Supp. Mat.*.

4.3. Ablation Studies

Synthesis module ablation study ⟨3-6⟩. In this section we evaluate the contribution of each proposed architecture component for the synthesis module: *Multi* use of multi-object modeling ⟨**3**⟩, π use of deformation modeling ⟨**4**⟩, w use of style modulation layers for appearance changes

Var.	*Multi* ⟨3⟩	π ⟨4⟩	w ⟨5⟩	F ⟨6⟩	LPIPS↓	FID↓	FVD↓	ADD↓	MDR↓
(a)					0.735	376	2548	109.1	99.9
(b)	✓				0.595	266	1617	45.4	86.4
(c)	✓	✓			0.648	301	1818	10.17	50.2
(d)	✓	✓	∼		0.361	68.6	482	7.39	31.9
(e)	✓	✓	✓		0.350	61.0	465	8.27	**31.8**
(f)	✓	✓	✓	∼	0.341	67.4	1371	88.5	88.8
Full	✓	✓	✓	✓	**0.193**	**16.5**	**289**	**5.45**	33.7

Table 4. Synthesis module ablation results on the *Minecraft* dataset. *Multi*: use of multi-object modeling, π: use of deformation, w: use of style modulation layers or of direct style encoding ($\sim$), F: use of the feature renderer or of the simplified renderer ($\sim$). ADD in pixels, MDR in %.

Figure 7. Camera and style manipulation results on the *Tennis* dataset. The original image is rendered under a novel camera perspective using varying styles for the field and players.

⟨**5**⟩, F use of the feature renderer for robustness ⟨**6**⟩. We produce the following method variations: (a) no component is used; this approach resembles NeRF [19]; (b) *Multi*; (c) *Multi* and π; this architecture is akin to NR-NeRF [34] with ⟨**3**⟩; (d) *Multi*, π, and w injected with concatenation rather than style modulation layers; (e) *Multi*, π, and w with style modulation layers; (f) *Multi*, π, w and a simplified ConvNet F that renders the complete frame from feature maps at a single resolution; from an architectural viewpoint, this feature rendering strategy resembles the one of GIRAFFE [21].

Results are shown in Tab. 4 and in the *Supp. Mat.*. (c) and (e) show that deformation and style modeling with style modulation layers are both necessary to accurately synthesize the scene, but generate blurry results due to calibration and localization errors. We recover sharpness by introducing our ConvNet feature renderer which reduces blur by modeling cross-pixel correlations. Substituting our renderer with the one of (f) leads to performance degradation due to the excessively sparse sampling of rays imposed by memory constraints when rendering the complete frame that leads to 3D consistency artifacts which are particularly apparent in the region of dynamic objects.

Var.	*Rel.*	D	$\mathcal{L}_\Delta$	$\mathcal{L}_{\text{act}}$	LPIPS↓	FID↓	FVD↓	Δ-*MSE*↓	Δ-*Acc*↑	ADD↓	MDR↓
(A)				✓	0.205	17.0	334	0.903	33.9	18.7	**33.0**
(B)	✓			✓	0.204	17.0	329	0.290	76.0	18.6	33.5
(C)	✓		✓	✓	**0.203**	16.9	340	**0.263**	**80.0**	**15.4**	34.0
(D)	✓	✓		✓	0.204	17.0	**323**	0.289	77.0	17.8	34.3
(E)	✓	✓	✓		0.204	16.9	335	0.276	77.5	17.5	34.0
Full	✓	✓	✓	✓	0.204	**16.8**	329	0.271	77.7	17.8	33.9

Table 5. Action module ablation results on the *Minecraft* dataset. *Rel.*: use of camera relative residual Δ output, D: use of the temporal discriminator, $\mathcal{L}_\Delta$: use of the loss for Δ-*MSE*, $\mathcal{L}_{\text{act}}$: use of the information-theoretic action learning loss. Δ-*MSE*, Δ-*Acc* and MDR in %, ADD in pixels.

Action module ablation study. We now evaluate the contribution of the main components of the action module by ablating the following: *Rel.* use of camera-relative object movement in the dynamics network; D use of the temporal discriminator; $\mathcal{L}_\Delta$ use of the loss on Δ-MSE; $\mathcal{L}_{\text{act}}$ use of the information-theoretic action learning loss. Results are shown in Tab. 5. Removing the temporal discriminator causes an increase in the FVD. A qualitative analysis of the results (see *Supp. Mat.*) shows that models not using D produce sequences where the players translate in the scene, but fail to realistically move their limbs. In addition, the introduction of $\mathcal{L}_\Delta$ produces a positive impact on the action space metrics. We also note that, thanks to the presence of $\mathcal{L}_\Delta$, the model learns an action space even in the absence of $\mathcal{L}_{\text{act}}$. Lastly, without camera-relative object movement in the dynamics network, the model produces movements that are independent from the current camera orientation, which is undesirable (Sec. 3.2).

5. Conclusions and Discussion

In conclusion, we present a new framework featuring a NeRF-based encoder-decoder architecture and an action module for the creation of compelling playable environments. Extensive experimental evaluation on two large-scale datasets shows that our method achieves state-of-the-art performance. We discuss the main limitations and ethical aspects of the method in the *Supp. Mat.*.

6. Acknowledgements

VG and CT were supported by the ERC consolidator grant 4DReply (770784). This project was supported by the EU H2020 project AI4Media (951911).

References

[1] Minecraft. https://www.minecraft.net. Accessed: 2021-11-12. 6

[2] Caroline Chan, Shiry Ginosar, Tinghui Zhou, and Alexei A Efros. Everybody dance now. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 5933–5942, 2019. 2

[3] Gaurav Chaurasia, Sylvain Duchêne, Olga Sorkine-Hornung, and George Drettakis. Depth synthesis and local warps for plausible image-based navigation. *ACM Transactions on Graphics*, 2013. 2

[4] Silvia Chiappa, Sébastien Racanière, Daan Wierstra, and Shakir Mohamed. Recurrent environment simulators. *CoRR*, abs/1704.02254, 2017. 2

[5] Chelsea Finn, Ian Goodfellow, and Sergey Levine. Unsupervised learning for physical interaction through video prediction. In *Advances in Neural Information Processing Systems (NIPS)*, volume 29, pages 64–72, 2016. 2

[6] Joseph L Fleiss. Measuring nominal scale agreement among many raters. *Psychological bulletin*, 76(5):378, 1971. 7

[7] Zekun Hao, Arun Mallya, Serge Belongie, and Ming-Yu Liu. GANcraft: Unsupervised 3D Neural Rendering of Minecraft Worlds. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2021. 3, 4

[8] 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 (NIPS)*, volume 30, pages 6626–6637, 2017. 6

[9] 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 (ICCV)*, 2017. 4

[10] Zhang Jiakai, Liu Xinhang, Ye Xinyi, Zhao Fuqiang, Zhang Yanshun, Wu Minye, Zhang Yingliang, Xu Lan, and Yu Jingyi. Editable free-viewpoint video using a layered neural representation. In *SIGGRAPH*, 2021. 3

[11] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *Proceedings of the European Conference of Computer Vision (ECCV)*, 2016. 5

[12] Seung Wook Kim, Jonah Philion, Antonio Torralba, and Sanja Fidler. Drivegan: Towards a controllable high-quality neural simulation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5820–5829, 2021. 2

[13] Seung Wook Kim, Yuhao Zhou, Jonah Philion, Antonio Torralba, and Sanja Fidler. Learning to Simulate Dynamic Environments with GameGAN. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2

[14] Johannes Kopf, Fabian Langguth, Daniel Scharstein, Richard Szeliski, and Michael Goesele. Image-based rendering in the gradient domain. *ACM Transactions on Graphics*, 32(6), Nov. 2013. 2

[15] Manoj Kumar, Mohammad Babaeizadeh, Dumitru Erhan, Chelsea Finn, Sergey Levine, Laurent Dinh, and Durk Kingma. Videoflow: A conditional flow-based model for stochastic video generation. In *International Conference on Learning Representations (ICLR)*, 2020. 2

[16] Alex X. Lee, Richard Zhang, Frederik Ebert, P. Abbeel, Chelsea Finn, and S. Levine. Stochastic adversarial video prediction. *ArXiv*, abs/1804.01523, 2018. 2, 6

[17] Michael Mathieu, Camille Couprie, and Yann LeCun. Deep multi-scale video prediction beyond mean square error. *arXiv preprint arXiv:1511.05440*, 2015. 2

[18] Willi Menapace, Stephane Lathuiliere, Sergey Tulyakov, Aliaksandr Siarohin, and Elisa Ricci. Playable video generation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10061–10070, 2021. 2, 4, 5, 6, 7

[19] 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 *Proceedings of the European Conference of Computer Vision (ECCV)*, 2020. 2, 3, 4, 8

[20] Michael Niemeyer and Andreas Geiger. CAMPARI: camera-aware decomposed generative neural radiance fields. In *In Proceedings of the International Conference on 3D Vision (3DV)*, 2021. 3

[21] Michael Niemeyer and Andreas Geiger. Giraffe: Representing scenes as compositional generative neural feature fields. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11453–11464, 2021. 3, 4, 8

[22] Manuel Serra Nunes, Atabak Dehban, Plinio Moreno, and José Santos-Victor. Action-conditioned benchmarking of robotic video prediction models: a comparative study. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 8316–8322, 2020. 2

[23] 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 (NIPS)*, pages 2863–2871, 2015. 2

[24] Julian Ost, Fahim Mannan, Nils Thuerey, Julian Knodt, and Felix Heide. Neural scene graphs for dynamic scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2856–2865, 2021. 2, 3

[25] Eric Penner and Li Zhang. Soft 3d reconstruction for view synthesis. 36(6), 2017. 2

[26] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-nerf: Neural radiance fields for dynamic scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10318–10327, 2021. 2

[27] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems (NIPS)*, volume 28, pages 91–99, 2015. 2

[28] Masaki Saito, Shunta Saito, Masanori Koyama, and Sosuke Kobayashi. Train sparsely, generate densely: Memory-efficient unsupervised training of high-resolution temporal gan. *International Journal of Computer Vision (IJCV)*. 2

[29] Steven M. Seitz, Brian Curless, James Diebel, Daniel Scharstein, and Richard Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 519–528, 2006. 2

[30] Aliaksandr Siarohin, Stéphane Lathuilière, Sergey Tulyakov, Elisa Ricci, and Nicu Sebe. Animating arbitrary objects via deep motion transfer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2

[31] Aliaksandr Siarohin, Stéphane Lathuilière, Sergey Tulyakov, Elisa Ricci, and Nicu Sebe. First order motion model for image animation. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2019. 2

[32] Aliaksandr Siarohin, Oliver Woodford, Jian Ren, Menglei Chai, and Sergey Tulyakov. Motion representations for articulated animation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 2

[33] Yu Tian, Jian Ren, Menglei Chai, Kyle Olszewski, Xi Peng, Dimitris N Metaxas, and Sergey Tulyakov. A good image generator is what you need for high-resolution video synthesis. In *International Conference on Learning Representations (ICLR)*, 2021. 2

[34] Edgar Tretschk, Ayush Tewari, Vladislav Golyanik, Michael Zollhöfer, Christoph Lassner, and Christian Theobalt. Non-rigid neural radiance fields: Reconstruction and novel view synthesis of a dynamic scene from monocular video. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2021. 2, 3, 4, 8

[35] 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 (CVPR)*, 2018. 2, 6

[36] 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

[37] Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. Anticipating the future by watching unlabeled video. *arXiv preprint arXiv:1504.08023*, 2, 2015. 2

[38] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Guilin Liu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. Video-to-video synthesis. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1144–1156, 2018. 2

[39] Wentao Yuan, Zhaoyang Lv, Tanner Schmidt, and Steven Lovegrove. Star: Self-supervised tracking and reconstruction of rigid objects in motion with neural rendering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 2, 3

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

[41] 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 (CVPR)*, 2018. 6

[42] C. Lawrence Zitnick, Sing Bing Kang, Matthew Uyttendaele, Simon A. J. Winder, and Richard Szeliski. High-quality video view interpolation using a layered representation. In *SIGGRAPH*, 2004. 2