Controllable Video Generation through Global and Local Motion Dynamics

Aram Davtyan and Paolo Favaro

Computer Vision Group, University of Bern, Switzerland
{aram.davtyan, paolo.favaro}@inf.unibe.ch

Abstract. We present GLASS, a method for Global and Local Action-driven Sequence Synthesis. GLASS is a generative model that is trained on video sequences in an unsupervised manner and that can animate an input image at test time. The method learns to segment frames into foreground-background layers and to generate transitions of the foregrounds over time through a global and local action representation. Global actions are explicitly related to 2D shifts, while local actions are instead related to (both geometric and photometric) local deformations. GLASS uses a recurrent neural network to transition between frames and is trained through a reconstruction loss. We also introduce W-Sprites (Walking Sprites), a novel synthetic dataset with a predefined action space. We evaluate our method on both W-Sprites and real datasets, and find that GLASS is able to generate realistic video sequences from a single input image and to successfully learn a more advanced action space than in prior work. Further details, the code and example videos are available at https://araachie.github.io/glass/.

Keywords: video generation; unsupervised action discovery; controllable generation

1 Introduction

A long-standing objective in machine learning and computer vision is to build agents that can learn how to operate in an environment through visual data [12]. A successful approach to do so is to use supervised learning, *i.e.*, to train a model on a large, manually annotated dataset [25]. However, if we take inspiration from how infants learn to move, we are brought to conclude that they may not rely on extensive guidance. In fact, while supervision from adults might come through language [30], the signal is certainly not detailed enough to fully define the locomotion dynamics. One approach that does not require direct supervision is to learn just through direct scrutiny of other agents, *i.e.*, through passive imitation. In fact, infants have an abundance of sensory exposure to the activities of adults before they themselves learn how to perform them [29].

The first step for an observing agent to learn how to operate in an environment through passive imitation and without explicit supervision is to build a model that: 1) separates an agent from its environment, 2) captures the appearance of the agent and its environment, and 3) builds a description of the agent's

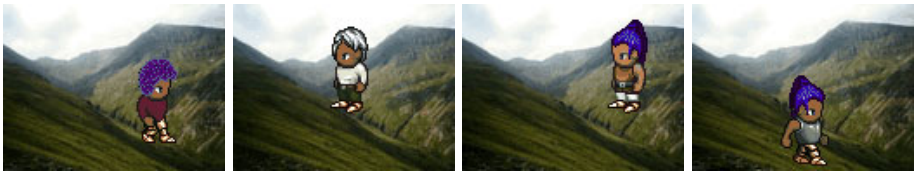

Fig. 1: *W-Sprites* dataset sample videos. To play them use Acrobat Reader.

dynamics. The first requirement implies that the model incorporates some segmentation capability, and it allows to explain transitions over time more easily. The second requirement is dictated by the fact that we exploit the reconstruction of visual observations as our indirect supervision signal. Thus, our model also relates to the video generation literature. Finally, the third requirement is that the model includes an *action space*, which serves two purposes: i) it allows the model to decode a video into a sequence of actions (which is a representation of the agent's dynamics) and ii) it allows the model to control the generation of videos by editing the action sequence.

We introduce GLASS, a method for Global and Local Action-driven Sequence Synthesis. As shown in Fig. 2, GLASS first learns to segment each frame of a video into foreground and background layers. A basic principle to do that is to use motion as a cue, *i.e.*, the fact that agents exhibit, on average, a distinct motion flow compared to the environment. Motion-based segmentation could be achieved through background subtraction, which is however restricted to stationary backgrounds, or instead, more in general, via optical flow. For simplicity, we propose to use an explicit foreground-background motion segmentation based on 2D shifts. Then, GLASS regresses the relative shift between the foregrounds of two subsequent frames, which we call the *global action*, and between the backgrounds (see Fig. 3). The *local actions* are learned only from the foregrounds. The decomposition of the agent's motion into global and local components provides a computationally efficient representation of the video. In contrast, a global action space would require the much larger Cartesian product of the local and global action spaces. In practice, given an action space of finite size, global models tend to learn only global motions and to ignore local deformations. We train an RNN to predict, through a decoder, the next foreground by using an encoding of a foreground, the previous state, and an encoding of the local and global actions as input. All networks are trained via reconstruction losses.

We evaluate GLASS on both synthetic and real data. As synthetic data we introduce *W-Sprites* (Walking Sprites [23,1,2]) (see Fig. 1), a dataset with a pre-defined action space, and where the action labels between pairs of frames (as well as the agent segmentation and location, and the background shift) are known. We find that GLASS learns a robust representation of both global and local dynamics on W-Sprites. Moreover, GLASS is able to decode videos into sequences of actions that strongly correlate with the ground truth action sequences. Finally, users can generate novel sequences by controlling the input

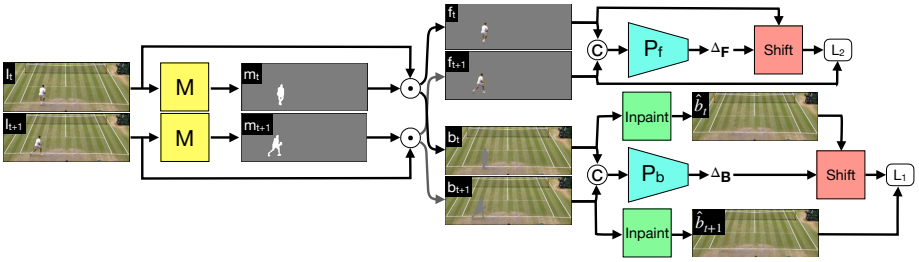

Fig. 2: GLASS Global Motion Analysis. Two input frames I_t and I_{t+1} are fed (separately) to a segmentation network M to output the foreground masks m_t and m_{t+1} respectively. The masks are used to separate the foregrounds f_t and f_{t+1} from the backgrounds b_t and b_{t+1}. The concatenated foregrounds are fed to the network P_f to predict their relative shift Δ_F. We use Δ_F to shift f_t and match it to f_{t+1} via an L_2 loss (foregrounds may not match exactly and this loss does not penalize small errors). In the case of the backgrounds we also train an inpainting network before shifting them with the predicted Δ_B and matching them with an L_1 loss (unlike foregrounds, we can expect backgrounds to match).

action sequences to GLASS. On real data, we find that GLASS can also generate realistic sequences by controlling the actions between frames.

Contributions: i) We introduce GLASS, a novel generative model with a global and local action space; the shifts estimated and generated through the global actions have an accuracy comparable to or higher than SotA; moreover, local actions allow a fine-grained modeling of dynamics that is not available in prior work; ii) We introduce W-Sprites, a novel dataset for the evaluation of action identification and generation; iii) We demonstrate GLASS on both synthetic and real datasets and show that it can: 1) segment an agent from its environment and estimate its global shift over time; 2) learn a disentangled action space that is consistent across agents; 3) decode videos into sequences of actions; 4) synthesize realistic videos under the guidance of a novel action policy.

2 Prior work

Video generation. Because GLASS is trained based on reconstruction losses, and it is built as a generative model, it relates to the generation of videos. Recent success in deep generative models for images [10,17,28] has aroused renewed interest in video generation. Several formulations tackling the problem of video generation exploit adversarial losses [3,4,9,22,32,35,36,37], autoregressive models [39] and use a wide range of architectures from RNNs [31] to transformers [40].

Controllable video generation. Video generation models can also differ in how they apply conditioning. While some prior work uses per-video class labels [19,38], *e.g.*, actions performed in a short sequence of frames, others, as in

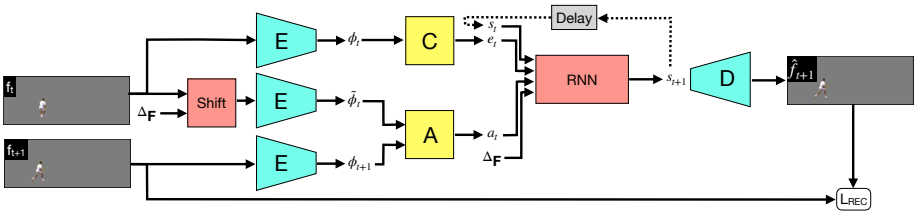

Fig. 3: GLASS Local Motion Analysis. We feed the segmented foreground f_t, its shifted version and f_{t+1} separately as inputs to an encoder network E to obtain features ϕ_t, $\tilde{\phi}_t$ and ϕ_{t+1} respectively. The latter two features are then mapped to an action a_t by the action network A. A further encoding of ϕ_t into e_t, the previous state s_t, and the local action a_t and global action Δ_F are fed as input to the RNN to predict the next state s_{t+1}. Finally, a decoder maps the state s_{t+1} to the next foreground $\hat{f}_{t+1}$, which is matched to the original foreground f_{t+1} via the reconstruction loss.

GLASS, use conditioning at each step [7,12,18,26,27]. For instance, in [12] the authors train a model to simulate the behavior of a robotic arm given the performed actions. Kim et al. [18] introduce GameGAN, a powerful generative model that can replace a game engine. It is trained to render the next frame given the current frame and the pressed keyboard action. One limitation of these methods is that they require knowledge of the ground truth actions and hence are restricted to synthetic data, such as video games. To become applicable to real data, several recent methods that learn an action space of the agent from raw videos without fine-grained annotations have been proposed. For instance, Rybkin et al. [29] propose a continuous latent space for the actions. They introduce arithmetical structure into their action space by exploiting the fact that two actions can be composed to get another action that would lead to the same result as when applying the original actions sequentially. [13] generates high-quality videos of moving agents with controllable actions. However, it builds an autoregressive model on top of pose maps obtained from supervised training, while our method is completely unsupervised. In [24] the continuous action space is replaced by a finite set. This allows a simpler control (playability) of the generated videos and favors interpretability of the learned actions. More recent work by Huang et al. [15] explicitly separates the foreground from the background and trains a network to predict the next frame given the current frame and the next segmentation mask. GLASS relates to this last family of methods as it also does not require any supervision signal.

Unsupervised learning of structured representations. In GLASS we propose to learn the global and local actions from video frames. While the global ones are defined as foreground 2D shifts, the local ones are represented as a discrete set of action codes. This leads to a latent clustering problem. In GLASS, we propose to solve it through variational inference [21]. Some recent work learns structured representations from raw input data [6,5]. The VQ-VAE [34] formula-

tion instead uses a discrete latent space and assumes a uniform distribution over the latent features. Recent advances in image and video generation has shown that such VQ-VAE based models have a remarkable performance [28,40] and this has encouraged us to adopt this approach.

3 Training GLASS

GLASS consists of two stages: One is the Global Motion Analysis (GMA) (shown in Fig. 2) and the other is the Local Motion Analysis (LMA) (shown in Fig. 3). GMA aims to separate the foreground agent from the background and to also regress the 2D shifts between foregrounds and backgrounds. LMA aims to learn a representation for local actions that can describe deformations other than 2D shifts. Towards this purpose it uses a Recurrent Neural Network (RNN) and a feature encoding of a frame and of the global and local actions as input. Both GMA and LMA stages are jointly trained in an unsupervised manner.

3.1 Global Motion Analysis

Let us denote a video as a sequence of T frames $I_t \in \mathbb{R}^{3\times H\times W}$, where $t = 1,\ldots,T$, and 3, H and W denote the number of color channels, the height and the width of the frame. Although GLASS is trained with video sequences, we can illustrate all the training losses with a single pair (I_t, I_{t+1}) of frames. Each frame is fed to a mask network M to output masks m_t and m_{t+1}. The masks can take values between 0 and 1 (a sigmoid is used at the output), but are encouraged to take the extreme values through the following binarization loss

$$\mathcal{L}_{\text{BIN}} = \textstyle\sum_t \min\{m_t, 1-m_t\}. \tag{1}$$

We also discourage the mask from being empty or covering the whole frame by using a mask size loss

$$\mathcal{L}_{\text{SIZE}} = \textstyle\sum_t |\mathbb{E}[m_t] - \theta|, \tag{2}$$

where $\mathbb{E}[\cdot]$ denotes the average over all pixels and $\theta \in [0,1]$ is a tuning parameter (the percentage of image pixels covered by a mask on average). The masks are then used to extract the foregrounds $f_t = I_t \odot m_t$ and $f_{t+1} = I_{t+1} \odot m_{t+1}$ and the backgrounds $b_t = I_t \odot (1-m_t)$ and $b_{t+1} = I_{t+1} \odot (1-m_{t+1})$ ($\odot$ denotes the element-wise product). We assume that the foregrounds are approximately matching up to a relative shift $\bar{\Delta}_F$, *i.e.*, that $f_{t+1}[p] \simeq \left(f_t \circ \bar{\Delta}_F\right)[p] \doteq f_t[p+\bar{\Delta}_F]$, for all pixel coordinates $p \in \Omega \subset \mathbb{R}^2$. We then concatenate the foregrounds and feed them as input to the pose network P_f to regress the relative shift $\Delta_F = P_f([f_t, f_{t+1}])$ between f_t and f_{t+1}. Since we do not have the ground truth shift $\bar{\Delta}_F$, we cannot train P_f via supervised learning. In alternative, we rely on the modeling assumption and define a reconstruction loss for the foreground by applying the estimated shift Δ_F to f_t and by matching it to the frame f_{t+1} in the L_2 norm (to allow for some error tolerance), *i.e.*,

$$\mathcal{L}_{\text{RECF}} = \textstyle\sum_t \left\| f_{t+1} - f_t \circ \Delta_F \right\|_2^2. \tag{3}$$

A similar derivation pertains to the backgrounds. We concatenate the backgrounds and feed them as input to the pose network P_b to regress the relative shift $\Delta_B = P_b([b_t, b_{t+1}])$ between b_t and b_{t+1}. However, because of the holes left by the masks, learning the relative shift via a direct matching of the backgrounds would not work. Therefore, we also introduce an inpainting network N. To indicate the masked region to N we simply fill it with a value out of the image range (we use [-1.1,-1.1,-1.1] as RGB values at the masked pixels). The inpainted regions are then copied to the corresponding backgrounds so that we obtain $\hat{b}_j = b_j \odot (1 - m_j) + \mathrm{N}(b_j) \odot m_j$, with $j = \{t, t+1\}$. The background reconstructions are then matched with both an L_1 norm and a perceptual loss $\mathcal{L}_{\mathrm{VGG}}$ based on VGG features[1] [16]

$$\mathcal{L}_{\mathrm{RECB}} = \sum_t \left\| \hat{b}_{t+1} - \hat{b}_t \circ \Delta_B \right\|_1 + \lambda_{\mathrm{VGG}} \mathcal{L}_{\mathrm{VGG}} \left(\hat{b}_{t+1}, \hat{b}_t \circ \Delta_B \right). \tag{4}$$

Finally, we also have a joint reconstruction loss where we compose the foreground with the estimated foreground shift Δ_F and the inpainted background with the estimated background shift Δ_B

$$\mathcal{L}_{\mathrm{RECJ}} = \sum_t \left\| (f_t \odot m_t) \circ \Delta_F + (\hat{b}_t \circ \Delta_B) \odot (1 - m_t \circ \Delta_F) - I_{t+1} \right\|_1. \tag{5}$$

These losses are all we use to train the mask network M, the inpainting network N and the pose estimation networks P_f and P_b. The inpainting network and the other networks could be further improved, but we find that the choices above are sufficient to obtain accurate segmentation masks and good shift estimates.

3.2 Local Motion Analysis

The LMA stage works directly on the foreground frames f_t and f_{t+1}. It first shifts f_t with Δ_F. This is done to remove the global shift information from the input frames and to make the action network focus on the local variations. It further encodes the foreground frames with a convolutional neural network E and obtains $\phi_t = \mathrm{E}(f_t)$, $\tilde{\phi}_t = \mathrm{E}(f_t \circ \Delta_F)$ and similarly for $\phi_{t+1} = \mathrm{E}(f_{t+1})$. The convolutional feature ϕ_t is then projected via C to give $e_t = \mathrm{C}(\phi_t)$.

In the action network A there are a few pre-processing steps. First, both feature maps $\tilde{\phi}_t$ and ϕ_{t+1} are fed to a CNN and flat features ψ_t and ψ_{t+1} are obtained from the resulting feature maps through global average pooling. In CADDY [24] the actions are determined through a direct difference between Gaussian samples around ψ_t and ψ_{t+1}. On average this means that the difference between features of images with the same action must align with the same direction. Although this works very well for CADDY, we find that this may be restrictive, especially if one wants to represent periodic motion (*e.g.*, in our case, an agent walking in place). Thus, we propose to learn a modified mapping of ψ_{t+1} conditioned on ψ_t. We compute $\psi^i_{t+1} = \mathrm{T}^i(\psi_t, \psi^{i-1}_{t+1})$ with $i = 1, \ldots, P$,

[1] VGG was obtained through supervised training, but GLASS can be trained equally well by replacing VGG with a similar network trained in a self-supervised manner.

T^i are bilinear transformations, $\psi^0_{t+1} = \psi_{t+1}$, and we choose $P = 4$. We then compute the action direction $d_t = \psi^P_{t+1} - \psi_t$. Finally, the action a_t is predicted through vector quantization after one additional MLP U to give $a_t = \mathrm{VQ}[\mathrm{U}(d_t)]$. The vector quantization VQ relies on K learnable prototype vectors c_k, with $k = 1, \ldots, K$. The method identifies the prototype c_q closest in L_2 norm to $\mathrm{U}(d_t)$, *i.e.*, $q = \arg\min_k \|c_k - \mathrm{U}(d_t)\|_2^2$, and uses that as the quantized action $\mathrm{VQ}[\mathrm{U}(d_t)] = c_q$. To train the VQ prototypes, we use the following loss [34]

$$\mathcal{L}_{VQ} = \|\mathrm{sg}[c_q] - U(d_t)\|_2^2 + \lambda_{\mathrm{VQ}} \|c_q - \mathrm{sg}[U(d_t)]\|_2^2, \tag{6}$$

where $\lambda_{\mathrm{VQ}} > 0$ is a tuning parameter and $\mathrm{sg}[\cdot]$ denotes stop-gradient. Vector quantization allows us to obtain latent space clustering in a simpler way compared to the Gaussian priors and the explicit tracking of cluster centroids as done in CADDY [24]. Moreover, our ablation studies show that VQ works better than the Gumbel-softmax trick (see Section 6 and Table 3).

Now, we have all the inputs needed for the RNN. We introduce an RNN state s_t and feed it together with the encoding e_t as input. Our RNN is split into 6 blocks as in CADDY [24]. Both the global action Δ_F and the local action a_t are first mapped to embeddings of the same size and then fed to the modulated convolutional layers of the RNN similarly to StyleGAN [17]. To differentiate the roles of Δ_F and a_t we feed the embeddings of Δ_F to the first two blocks of the RNN and that of a_t to the remaining four blocks. The rationale is that early blocks correlate more with global changes, such as translations, and the later blocks correlate more with local deformations.

Finally, the decoder D takes the RNN prediction s_{t+1} as input and outputs the frame $\hat{f}_{t+1} = D_f(s_{t+1})$ and the predicted mask $\hat{m}_{t+1} = D_m(s_{t+1})$. Moreover, the decoder predicts frames at 3 different scales (as also done in CADDY [24]). We introduce a reconstruction loss for each scale

$$\mathcal{L}_{\mathrm{RECU}} = \textstyle\sum_t \left\| \mathrm{sg}[\omega_{\mathrm{UNS}}] \odot \left(f_{t+1} - \hat{f}_{t+1} \right) \right\|_1, \tag{7}$$

where $\forall p \in \Omega$, $\omega_{\mathrm{UNS}}[p] = \|f_t[p] - f_{t+1}[p]\|_1 + \|\hat{f}_t[p] - \hat{f}_{t+1}[p]\|_1$ are weights that enhance the minimization at pixels where the input and predicted foregrounds differ, and also a perceptual loss

$$\mathcal{L}_{\text{LMA-VGG}} = \mathcal{L}_{\mathrm{VGG}}(f_{t+1}, \hat{f}_{t+1}). \tag{8}$$

To better learn local deformations, we also introduce a reconstruction loss that focuses on the differences between the foregrounds after aligning them with the estimated relative shifts, *i.e.*,

$$\mathcal{L}_{\mathrm{RECS}} = \textstyle\sum_t \left\| \mathrm{sg}[\omega_{\mathrm{ALIGN}}] \odot \left(f_{t+1} - \hat{f}_{t+1} \right) \right\|_1, \tag{9}$$

where $\omega_{\mathrm{ALIGN}}[p] = \|f_t \circ \Delta_F[p] - f_{t+1}[p]\|_1 + \|\hat{f}_{t+1}[p] - f_{t+1}[p]\|_1$. To encourage the consistency between the predicted mask $\hat{m}_{t+1}$ and the mask m_{t+1} obtained from I_{t+1}, we also minimize

$$\mathcal{L}_{\mathrm{MSK}} = \textstyle\sum_t \|\hat{m}_{t+1} - m_{t+1}\|_1. \tag{10}$$

Moreover, we encourage a cyclic consistency between the encoded features via

$$\mathcal{L}_{\text{CYC}} = \sum_t \|\text{sg}[\phi_{t+1}] - \text{E}(\hat{f}_{t+1})\|_1. \tag{11}$$

Our final loss consists of a linear combination of all the above losses (both from the GMA and LMA) through corresponding positive scalars λ_{VQ}, $\lambda_{\text{LMA-VGG}}$, λ_{RECU}, λ_{RECS}, λ_{MSK}, λ_{CYC}, λ_{RECF}, λ_{RECB}, λ_{RECJ}, λ_{BIN}, and λ_{SIZE}.

4 Implementation details

At inference time, GLASS can generate a sequence of frames given only the first one. This setting is slightly different from training, where the model only predicts the next frame given the previous one. In order to prepare the model for test time, we adopt the mixed training procedure (Teacher Forcing) also used in [24]. That is, we select a video duration T_f, $0 < T_f < T$, and if $t \leq T_f$ we feed the encodings of the real frames to the RNN, otherwise if $t > T_f$ we use the encodings of the reconstructed frames. During the training we gradually decrease T_f to 1 and increase T to adapt the network to the generation of longer sequences. To speed up the convergence, we pretrain the GMA component for 3000 iterations. The coefficients before the loss terms are estimated on the training set. We found that the selected configuration works well across all datasets. The models are trained using the Adam optimizer [20] with a learning rate equal to 0.0004 and weight decay 10^{-6}. For more details, see the supplementary material.

5 W-Sprites dataset

In order to assess and ablate the components of GLASS, we build a synthetic video dataset of cartoon characters acting on a moving background. We call the dataset W-Sprites (for Walking Sprites). Each sequence is generated via the following procedure. First, one of 1296 different characters is sampled from the Sprites dataset [1,23,2]. This character is then animated in two stages. A random walk module produces a sequence of global coordinates of the sprite within a 96×128 resolution frame. We then sample one of 9 local actions conditioned on the shift induced by the global motion component. Those actions include: `walk front`, `walk left`, `walk right`, `spellcast front`, `spellcast left`, `spellcast right`, `slash front`, `slash left`, and `slash right`. The intuition under conditioning is that the global actions and the local ones should be correlated for more realism. For instance, when the global action module dictates a right shift, the only possible local action should be `walk right`. Analogously, the left shift induces the `walk left` action. The up and down shifts are animated with the `walk front` action. The remaining actions are used to animate the static sprite. To incorporate more generality and to reduce the gap with real data, we apply an independent random walk to the background image (this simulates camera motion). We use a single background image sampled

Table 1: Global Motion Analysis (GMA). mIoU evaluations

Configuration	$\not{\mathcal{L}}_{\text{RECB}}$	$\not{\mathcal{L}}_{\text{RECF}}$	$\not{\mathcal{L}}_{\text{SIZE}}$	$\not{\mathcal{L}}_{\text{RECJ}}$	$\not{\mathcal{L}}_{\text{BIN}}$	$\not{\mathcal{L}}_{\text{VGG}}$	GLASS
mIoU	0.01	0.08	0.08	0.08	0.87	**0.89**	0.88

Table 2: Global Motion Analysis (GMA). Shift error estimation

Configuration	Background Shift Error				Foreground Shift Error			
	mean	min	max	$\measuredangle$-ACC	mean	min	max	$\measuredangle$-ACC
$\not{\mathcal{L}}_{\text{BIN}}$	0.55	0.01	1.16	**1.00**	4.46	0.05	8.50	**1.00**
$\not{\mathcal{L}}_{\text{VGG}}$	0.52	0.02	0.90	**1.00**	4.38	**0.01**	8.51	**1.00**
GLASS	**0.51**	**0.00**	**0.86**	**1.00**	**4.34**	0.02	**8.32**	**1.00**

from the "valleys" class of ImageNet [8]. Each video in the W-Sprites dataset is annotated with per frame actions (*i.e.*, global shifts and local action identifiers), background shifts and character masks. We show sequence samples from our dataset in Fig. 1 (to play the videos view the pdf with Acrobat Reader). The dataset contains videos with 10 to 90 frames per sprite. For testing purposes, we split the dataset into training (about 8/9 th) and validation sets (about 1/9 th). The validation set contains sprites never seen during training. For more details, see the supplementary material.

6 Ablations

In this section we separately ablate the global and local components of GLASS. We run the ablations on **W-Sprites**, which has been introduced in section 5.

GMA ablations. For the global motion analysis, we assess the impact of each loss function. Different loss terms are sequentially switched off and the performance of the model trained without those terms is reported. Given that W-Sprites is fully annotated, we propose several metrics to evaluate the training. First, we calculate the mean intersection over union (mIoU) between the ground truth and the predicted segmentation masks. Table 1 shows that the VGG loss seems to slightly hurt the segmentation performance. However, as shown in Table 2 the VGG loss benefits the shift estimation. Notice that in Table 2 we report only the cases where the masks are good enough (mIoU > 0.8). For the shift errors we show the L_2 norm of the difference between the ground truth foreground/background shift and the predicted one (in pixels). We also show the accuracy of the predicted foreground/background shift directions ($\measuredangle$-ACC). The direction is considered to be correctly predicted if the angle between the ground truth and the predicted shifts is less than 45°. Each model is trained for 60K iterations with a batch size of 4. The results are calculated on the validation set.

LMA ablations. For the local motion analysis module we specifically design 5 cases that differ from GLASS in its essential components and show the results in Table 3. First, we evaluate swapping the modified mapping T of the

Table 3: Local Motion Analysis (LMA). Component ablation results

Configuration	LPIPS↓	FID↓	FVD↓	mIoU_{RE} ↑	NMI_{G} ↑	NMI_{S} ↑	CON ↓
Plain directions	0.402	12.8	204	0.83	0.14	0.17	0.03
Gumbel	0.402	16.8	327	0.84	0.00	0.02	0.02
No modulated convs	**0.398**	10.4	172	**0.89**	0.35	0.38	0.03
Joint input	0.399	**9.8**	**146**	**0.89**	0.34	0.37	0.03
$\not{\mathcal{L}}_{\text{RECS}}$	0.400	11.3	203	**0.89**	0.29	0.30	**0.01**
GLASS 200K	0.399	10.5	175	0.88	**0.39**	**0.41**	0.02
CADDY [24]	0.404	**6.8**	153	-	-	-	-
GLASS 470K	**0.398**	8.2	**129**	0.93	0.39	0.40	0.01

features ψ_{t+1} for the direct difference between the features $\psi_{t+1} - \psi_t$ (as done in CADDY [24]). We refer to this configuration as "Plain directions". Second, we replace the vector quantization with an MLP that predicts the distribution over actions followed by the Gumbel-Softmax trick to sample a discrete action identifier. We name this model "Gumbel". We also ablate the impact of using modulated convolutional layers by feeding the action embeddings as normal inputs to common convolutional blocks. This cases is referred to as "No modulated convs". Also we consider the case where we feed the global and local action embeddings jointly to all RNN blocks instead of separate ones. We refer to this case as "Joint input". The last case that we evaluate for the ablations is the model trained without $\mathcal{L}_{\text{RECS}}$. All the models are trained for 200K iterations with a batch size of 4. Additionally we report the metrics of GLASS trained for 470K iterations. As a reference, we also show the performance of CADDY [24] trained on the W-Sprites dataset with the same configuration from the original paper for the Tennis dataset. For visual results, please, refer to the supplementary material.

Following CADDY [24], we generate the sequences from the first frames of the validation videos conditioned on the actions inferred from the remaining frames. We measure FID [14], FVD [33] and LPIPS [41] scores on the generated sequences to asses the quality of the generated videos. Additionally we segment the reconstructed sequences and report the mean IoU with the ground truth masks to asses the ability of the RNN to condition on the input global and local action embeddings. However, the most important aspect of our (and prior) work on controllable models is the identification of the action space. Thus, one needs a metric to asses the quality of the learned action space. For this purpose, we propose to use the normalized mutual information score (NMI) between the ground truth and inferred local actions

$$\text{NMI}(X, Y) = \frac{2I(X,Y)}{H(X)+H(Y)}, \tag{12}$$

where $I(X, Y)$ is the mutual information between X and Y and $H(X)$ is the entropy of X. In our formulation, X and Y correspond to the predicted and ground truth actions respectively. One appealing advantage of NMI for GLASS is that NMI is invariant to permutations of the labels. Another advantage of using

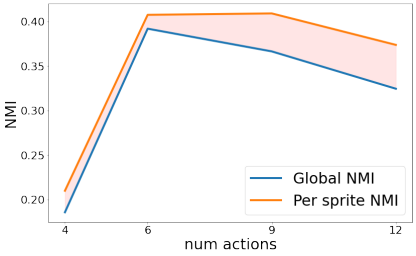

Fig. 4: Ablation of the number of actions fitted during the training of GLASS on the *W-Sprites* dataset.

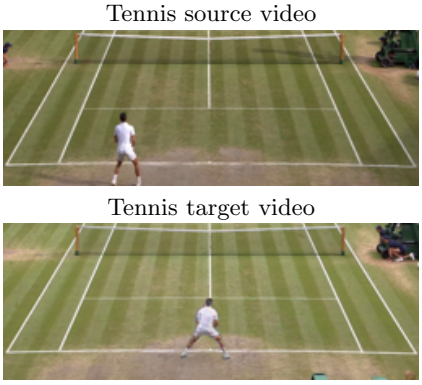

Fig. 5: Example of transferring an action sequence decoded from the source video to the target. To play view the paper in Acrobat Reader.

NMI is that NMI does not require the distributions to have the same number of actions. Thus, even with a given known number of ground truth actions, the model can be trained and assessed with a different number of actions. Indeed, the decomposition of a sequence into actions is not unique. For instance the `walk left` action can be decomposed into `turn left` and `walk`. We introduce two different protocols of measuring NMI. First, we use the trained model to map all the pairs of successive frames into actions. Then, the global NMI_G is computed between the ground truth actions and those predictions. Additionally, we average the per sprite NMI scores to obtain NMI_S. Normally $\text{NMI}_\text{S} > \text{NMI}_\text{G}$. However, if the gap is large enough, this indicates the overfitting and the lack of consistency of the learned actions across different sprites. Therefore, we also report the consistency metric $\text{CON} = \text{NMI}_\text{S} - \text{NMI}_\text{G}$. As a reference we use the NMI_RAND, that is the NMI measured between the ground truth actions and random actions. The results are provided in Table 3. Given that $\text{NMI}_\text{RAND} = 0.02$ on the W-Sprites test set, the full GLASS configuration with an NMI of 0.41 shows that the model estimates action sequences with a high correlation to the ground truth actions. Note that since our main focus is learning the action space of the agent, the NMI metric is of higher importance than FID/FVD. However, since the model is based on reconstruction as a supervision signal, it is also important that the FID/FVD are within the high-quality range. Furthermore, we ablate the number of actions K used to train GLASS. In Fig. 4 one can see that $K = 6$ is optimal in both NMI and CON.

7 Experiments on real data

We evaluate GLASS on 3 datasets. For synthetic data we use **W-Sprites**. For real data we use: 1) the **Tennis Dataset** and 2) the **BAIR Robot Pushing**

Table 4: *BAIR* dataset evaluation

Method	LPIPS↓	FID↓	FVD↓
MoCoGAN [32]	0.466	198	1380
MoCoGAN+ [24]	0.201	66.1	849
SAVP [22]	0.433	220	1720
SAVP+ [24]	<u>0.154</u>	<u>27.2</u>	<u>303</u>
Huang et al. [15] w/ *non-param* control	0.176	29.3	293
CADDY [24]	0.202	35.9	423
Huang et al. [15] w/ *positional* control	0.202	28.5	333
Huang et al. [15] w/ *affine* control	0.201	30.1	**292**
GLASS	**0.118**	**18.7**	411

Table 5: *Tennis* dataset evaluation

Method	LPIPS↓	FID↓	FVD↓	ADD↓	MDR↓
MoCoGAN [32]	0.266	132	3400	28.5	20.2
MoCoGAN+ [24]	0.166	56.8	1410	48.2	27.0
SAVP [22]	0.245	156	3270	10.7	19.7
SAVP+ [24]	0.104	25.2	223	13.4	19.2
Huang et al. [15] w/ *non-param* control	0.100	8.68	204	1.76	0.306
CADDY [24]	<u>0.102</u>	13.7	239	8.85	1.01
Huang et al. [15] w/ *positional* control	0.122	<u>10.1</u>	<u>215</u>	4.30	<u>0.300</u>
Huang et al. [15] w/ *affine* control	0.115	11.2	**207**	<u>3.40</u>	0.317
GLASS	**0.046**	**7.37**	257	**2.00**	**0.214**

Dataset. The Tennis Dataset was introduced in [24] and contains ~ 900 videos extracted from 2 Tennis matches from YouTube at 96×256 pixel resolution. The videos are cropped to contain only one half of the court, so that only one player is visible. The BAIR Robot Pushing Dataset [11] contains around 44K clips of a robot arm pushing toys on a flat square table at 256×256 pixel resolution.

Baselines. We compare to CADDY [24], because it allows frame-level playable control, and to Huang et al. [15]. However, the comparison to [15] is not fair, since it requires a prior knowledge of the future agent masks and also it lacks the ability to control the agent through discrete actions (playability). We also report the metrics on other conditional video generation models such as MoCoGAN [32], SAVP [22] and their large scale versions introduced in [24].

Quantitative analysis. Following [24] we evaluate GLASS on the video reconstruction task. Given a test video, we use GMA to predict the global shifts and LMA to estimate the discrete actions performed along the video. Further, the agent is segmented using the masking network and the foreground is animated and pasted back to the shifted background using both global and local actions to reconstruct the whole sequence from the first frame. We report FID, FVD and LPIPS scores on the generated videos. On the Tennis dataset we additionally report the Average Detection Distance (ADD) and the Missing Detection Rate

Fig. 6: A sequence generated with GLASS trained on the *W-Sprites* dataset. Note that the level of control provided by GLASS allows to generate unseen motion such as *jump*. Use Acrobat Reader to play the first frame.

(MDR) suggested in [24]. Those metrics are supposed to assess the action space quality by detecting the tennis player with a pretrained human detector and by comparing the locations of the detected agents in the ground truth and generated sequences. On BAIR (see Table 4) our model performs almost 40% better in terms of frame-level quality, but lacks in FVD compared to [15]. However, it is still slightly better than CADDY. On the Tennis dataset (see Table 5) GLASS is around 50% better than the closest competitor in LPIPS, almost 30% better in FID, but loses in FVD. However, GLASS provides finer control over the agent according to ADD and MDR. It is worth noting, that the FVD is largely affected by the background modeling choices. We found that even different interpolation methods had a major impact on the FVD. However, since the main focus of this work is the action space of the foreground agent, we chose the simplest background model that yielded a performance on par (or better) with the SotA.

Qualitative analysis. A trained GLASS allows a detailed control of the agent. On W-Sprites, we find that the LMA discovers such actions as `turn right`, `turn left`, `turn front`, `spellcast` and `slash`. Note that despite the difference between the discovered set of actions and the ground truth, all videos in the training set can be generated with this reduced set of actions (see Fig. 6). On Tennis we found that the local actions mostly correspond to some leg movements. On BAIR the LMA component discovers some small local deformations such as the state of the manipulator (closed or open).

In Fig. 7, we provide visual examples of the GLASS global action space. Given two different starting foregrounds from the BAIR and Tennis datasets (shown in the green channel), we show the generated foregrounds (in the red channel) after applying the `right`, `left`, `down`, `up` and `no motion` global shifts. We can also see that global actions apply consistently across different initial foregrounds. To show that the learned action space is consistent across different agents also in their fine-grained dynamics we use GLASS to transfer (both global and local) motion from one video to another. We first extract the sequence of actions in the first video using the GMA and LMA components of GLASS and then sequentially apply these actions to the first frame of the second video. In Fig. 5, we demonstrate it on the Tennis dataset.

Finally, in Fig. 8 we provide some sample outputs from our GMA module on test images from all three datasets. Given an input image, we can see that

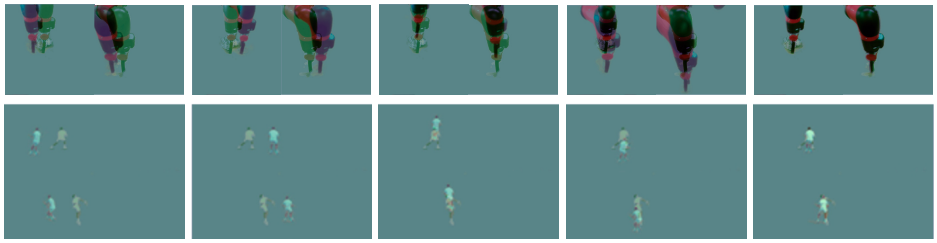

Fig. 7: Learned global actions on the *BAIR* and *Tennis* datasets (foreground-only generation). We use the green channel for the given initial foreground and the red channel for the foreground generated with the selected action. For both datasets we show 2 examples to demonstrate the action consistency.

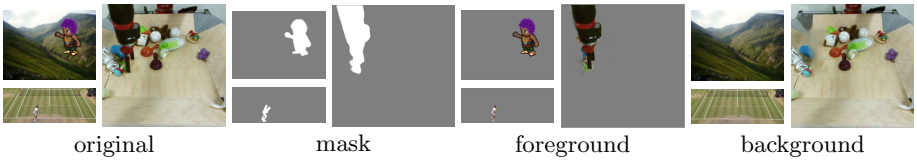

Fig. 8: Sample outputs from our GMA module. From left to right: original image, predicted segmentation, foreground, and inpainted background.

the segmentation network learns to extract accurate masks with which one can obtain high quality foreground images. These are necessary to model local dynamics. The inpainting of the background is sufficiently accurate to separate the two layers. For more visual examples, please see the supplementary material.

8 Conclusions and limitations

GLASS is a novel generative model with a global and local action space that enables a fine-grained modeling and control of dynamics not available in prior work. GLASS is trained in a completely unsupervised manner. We also introduce W-Sprites, a novel dataset for the evaluation of action identification and generation. Our experimental evaluation shows that GLASS learns consistent, and thus transferrable, action representations and is able to synthesize realistic videos with arbitrary action policies. One limitation of GLASS is that it works only on data where the background dynamics are 2D shifts. This does not capture, for example, the motion of the background objects in the BAIR dataset. GLASS is capable of rendering multiple agents within the same frame, but does not learn the separate action spaces of multiple agents. This remains a challenging task, which we plan to tackle in future work.

Acknowledgements This work was supported by grant 188690 of the Swiss National Science Foundation.

References

1. Liberated pixel cup. https://lpc.opengameart.org, accessed on 2022-03-07
2. Universal lpc sprite sheet. https://github.com/makrohn/Universal-LPC-spritesheet/tree/7040e2fe85d2cb1e8154ec5fce382589d369bdb8, accessed on 2022-03-07
3. Acharya, D., Huang, Z., Paudel, D.P., Van Gool, L.: Towards high resolution video generation with progressive growing of sliced wasserstein gans. arXiv preprint arXiv:1810.02419 (2018)
4. Babaeizadeh, M., Finn, C., Erhan, D., Campbell, R.H., Levine, S.: Stochastic variational video prediction. arXiv preprint arXiv:1710.11252 (2017)
5. Burgess, C.P., Higgins, I., Pal, A., Matthey, L., Watters, N., Desjardins, G., Lerchner, A.: Understanding disentangling in β-vae. arXiv preprint arXiv:1804.03599 (2018)
6. Caron, M., Bojanowski, P., Joulin, A., Douze, M.: Deep clustering for unsupervised learning of visual features. In: Proceedings of the European conference on computer vision (ECCV). pp. 132–149 (2018)
7. Chiappa, S., Racaniere, S., Wierstra, D., Mohamed, S.: Recurrent environment simulators. arXiv preprint arXiv:1704.02254 (2017)
8. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: 2009 IEEE conference on computer vision and pattern recognition. pp. 248–255. Ieee (2009)
9. Denton, E., Fergus, R.: Stochastic video generation with a learned prior. In: International conference on machine learning. pp. 1174–1183. PMLR (2018)
10. Dhariwal, P., Nichol, A.: Diffusion models beat gans on image synthesis. Advances in Neural Information Processing Systems **34** (2021)
11. Ebert, F., Finn, C., Lee, A.X., Levine, S.: Self-supervised visual planning with temporal skip connections. In: CoRL. pp. 344–356 (2017)
12. Finn, C., Goodfellow, I., Levine, S.: Unsupervised learning for physical interaction through video prediction. Advances in neural information processing systems **29** (2016)
13. Gafni, O., Wolf, L., Taigman, Y.: Vid2game: Controllable characters extracted from real-world videos. arXiv preprint arXiv:1904.08379 (2019)
14. Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., Hochreiter, S.: Gans trained by a two time-scale update rule converge to a local nash equilibrium. Advances in neural information processing systems **30** (2017)
15. Huang, J., Jin, Y., Yi, K.M., Sigal, L.: Layered controllable video generation. arXiv preprint arXiv:2111.12747 (2021)
16. Johnson, J., Alahi, A., Fei-Fei, L.: Perceptual losses for real-time style transfer and super-resolution. In: European conference on computer vision. pp. 694–711. Springer (2016)
17. Karras, T., Aittala, M., Laine, S., Härkönen, E., Hellsten, J., Lehtinen, J., Aila, T.: Alias-free generative adversarial networks. Advances in Neural Information Processing Systems **34** (2021)
18. Kim, S.W., Zhou, Y., Philion, J., Torralba, A., Fidler, S.: Learning to simulate dynamic environments with gamegan. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 1231–1240 (2020)
19. Kim, Y., Nam, S., Cho, I., Kim, S.J.: Unsupervised keypoint learning for guiding class-conditional video prediction. Advances in neural information processing systems **32** (2019)

20. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
21. Kingma, D.P., Welling, M.: Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114 (2013)
22. Lee, A.X., Zhang, R., Ebert, F., Abbeel, P., Finn, C., Levine, S.: Stochastic adversarial video prediction. arXiv preprint arXiv:1804.01523 (2018)
23. Li, Y., Mandt, S.: Disentangled sequential autoencoder. In: International Conference on Machine Learning (2018)
24. Menapace, W., Lathuilière, S., Tulyakov, S., Siarohin, A., Ricci, E.: Playable video generation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 10061–10070 (2021)
25. Mottaghi, R., Bagherinezhad, H., Rastegari, M., Farhadi, A.: Newtonian scene understanding: Unfolding the dynamics of objects in static images. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 3521–3529 (2016)
26. Nunes, M.S., Dehban, A., Moreno, P., Santos-Victor, J.: Action-conditioned benchmarking of robotic video prediction models: a comparative study. In: 2020 IEEE International Conference on Robotics and Automation (ICRA). pp. 8316–8322. IEEE (2020)
27. Oh, J., Guo, X., Lee, H., Lewis, R.L., Singh, S.: Action-conditional video prediction using deep networks in atari games. Advances in neural information processing systems **28** (2015)
28. Razavi, A., Van den Oord, A., Vinyals, O.: Generating diverse high-fidelity images with vq-vae-2. Advances in neural information processing systems **32** (2019)
29. Rybkin, O., Pertsch, K., Derpanis, K.G., Daniilidis, K., Jaegle, A.: Learning what you can do before doing anything. arXiv preprint arXiv:1806.09655 (2018)
30. Smith, L., Gasser, M.: The development of embodied cognition: Six lessons from babies. Artificial life **11**(1-2), 13–29 (2005)
31. Srivastava, N., Mansimov, E., Salakhudinov, R.: Unsupervised learning of video representations using lstms. In: International conference on machine learning. pp. 843–852. PMLR (2015)
32. Tulyakov, S., Liu, M.Y., Yang, X., Kautz, J.: Mocogan: Decomposing motion and content for video generation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1526–1535 (2018)
33. Unterthiner, T., van Steenkiste, S., Kurach, K., Marinier, R., Michalski, M., Gelly, S.: Towards accurate generative models of video: A new metric & challenges. arXiv preprint arXiv:1812.01717 (2018)
34. Van Den Oord, A., Vinyals, O., et al.: Neural discrete representation learning. Advances in neural information processing systems **30** (2017)
35. Vondrick, C., Pirsiavash, H., Torralba, A.: Anticipating the future by watching unlabeled video. arXiv preprint arXiv:1504.08023 **2**, 2 (2015)
36. Vondrick, C., Pirsiavash, H., Torralba, A.: Generating videos with scene dynamics. Advances in neural information processing systems **29** (2016)
37. Wang, Y., Bilinski, P., Bremond, F., Dantcheva, A.: G3an: Disentangling appearance and motion for video generation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5264–5273 (2020)
38. Wang, Y., Bilinski, P., Bremond, F., Dantcheva, A.: Imaginator: Conditional spatio-temporal gan for video generation. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. pp. 1160–1169 (2020)
39. Weissenborn, D., Täckström, O., Uszkoreit, J.: Scaling autoregressive video models. arXiv preprint arXiv:1906.02634 (2019)

40. Yan, W., Zhang, Y., Abbeel, P., Srinivas, A.: Videogpt: Video generation using vq-vae and transformers. arXiv preprint arXiv:2104.10157 (2021)
41. Zhang, R., Isola, P., Efros, A.A., Shechtman, E., Wang, O.: The unreasonable effectiveness of deep features as a perceptual metric. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 586–595 (2018)