

Controllable Video Generation through Global and Local Motion Dynamics – Supplementary Material –

Aram Davtyan[✉] and Paolo Favaro[✉]

Computer Vision Group, University of Bern, Switzerland
{[aram.davtyan](mailto:aram.davtyan@inf.unibe.ch),[paolo.favaro](mailto:paolo.favaro@inf.unibe.ch)}@inf.unibe.ch

Abstract. In the main paper we present GLASS, a method for Global and Local Action-driven Sequence Synthesis. GLASS, trained on unlabeled video sequences, allows to animate an input image at test time. The method builds a global and local action representation that is used to generate transitions of the segmented foreground sequences. Moreover, we introduced a novel dataset (W-Sprites) with a predefined action space for analysis. This supplementary material provides details and visual examples that could not be included in the main paper due to the space limitations. In section A we describe the implementation details, such as network architecture and training parameters. Section B provides details on the dataset generation protocol. In Section C we include more visual examples of the evaluation of our method. Further details, the code and example videos are available at <https://araachie.github.io/glass/>.

A Implementation

In this section we report further details regarding the implementation of GLASS.

A.1 Network architecture

In our code we mostly adopt convolutional blocks from the publicly available implementation of CADDY [5]. Those include residual blocks, upsampling and downsampling blocks. The blocks mainly incorporate Leaky-ReLu activations and Batch Normalization layers. Exceptions are the blocks that output masks (sigmoid activation), the blocks that output images (tanh activation) and the LSTM blocks (sigmoid and tanh activations) [1].

GMA. The architecture of our Global Motion Analysis (GMA) module is depicted in Fig. 1. GMA consists of 4 networks: the masking network, 2 identical shift predictors and the inpainter.

LMA. The architecture of our Local Motion Analysis (LMA) module is depicted in Fig. 2. The encoder E and the decoder D are mostly adopted from Menapace et al. [5]. However, we introduce an additional 1×1 -convolutional block C to compress the feature vector before feeding it to the RNN. This is supposed to prevent overfitting to the appearance of the agent. We also change the RNN to

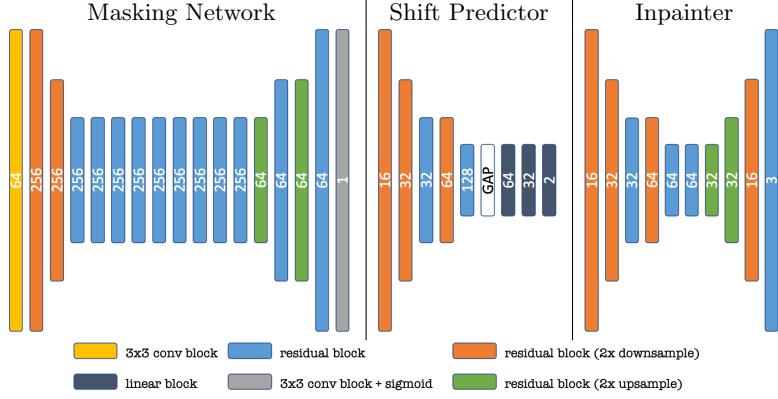


Fig. 1: The architectures of the Global Motion Analysis (GMA) module blocks. The number of output channels is indicated in the center of each block. GAP stands for Global Average Pooling.

take the action codes as input through the modulated convolution, as in StyleGAN [2]. Note, that the recurrent units leveraged in our RNN’s implementation are in fact convolutional LSTM blocks. Moreover, we upgrade the architecture of the action network A by incorporating delayed bilinear blocks and using Vector Quantization [3] for estimation of the performed action. We would also like to clarify the intuition behind using a sequence of bilinear transformations to model actions instead of the difference between ψ_{t+1} and ψ_t , as done in [5]. By using the difference as an action direction, the model only discriminates linear transitions in the latent space. This, in addition to the low dimensional action space used in [5], results in the fact that CADDY mostly discovers global 2D transformations, such as shifts. However, local actions are mostly periodic (consider an agent that rotates or walks in place). With our sequence of bilinear transformations we let the network unfold the latent space trajectories first before taking the difference between the features. Our ablation studies in the main paper suggest that this approach helps.

A.2 Training details

GMA and LMA scheduling. To start training the LMA on a valid input data, we first train the GMA module for 3k iterations to warm it up. Then, the LMA begins to train together with the GMA. But, no gradient flows back to GMA from LMA.

Loss terms coefficients. The configuration of the λ coefficients used in the linear combination of the separate loss terms is shown in Table 1. We found that this selection of λ works well across all the datasets.

Sequence length scheduling. As described in the main paper, we choose a sequence length T_f , $0 < T_f < T$, after which the encodings of the reconstructed

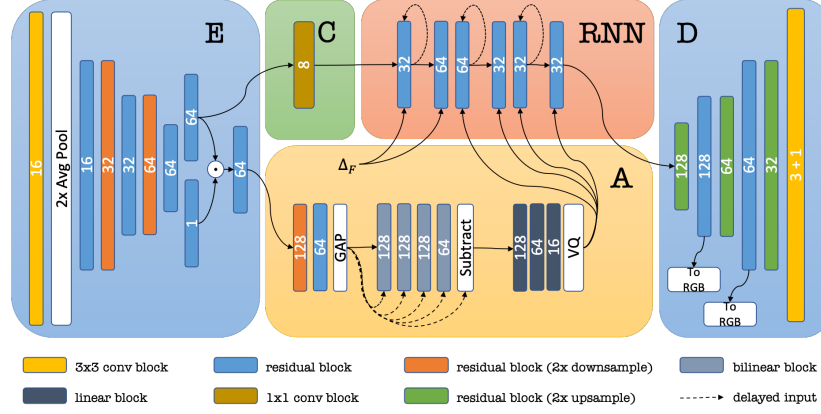


Fig. 2: The architecture of the Local Motion Analysis (LMA) module of GLASS.

Table 1: The coefficients of the loss terms used in the training of GLASS

GMA	λ_{BIN}	λ_{SIZE}	λ_{RECF}	λ_{RECB}	λ_{VGG}	λ_{RECJ}
	0.5	0.1	1.0	2.0	0.01	1.0

LMA	λ_{VQ}	λ_{RECU}	λ_{RECS}	λ_{MSK}	λ_{CYC}	$\lambda_{\text{LMA-VGG}}$
	0.25	1.0	1.0	0.2	0.2	1.0

foregrounds are fed to the RNN. For all the datasets we start from $T_f = 5, T = 6$ and gradually decrease T_f to 1 in 25000 iterations after the GMA pretraining stage has ended. On the BAIR dataset T remains also constant, while on the Tennis and on the W-Sprites datasets we gradually increase T from 6 to 12 in order to favor the quality of long generated sequences.

Optimization and Batching. As mentioned in the main paper, the models are trained using the Adam optimizer [4] with a learning rate equal to 0.0004 and weight decay 10^{-6} . We decrease the learning rate by a factor of 0.3 after 300K iterations. On W-Sprites and Tennis we used batch size equal to 4. However, on the BAIR dataset due to the high resolution of the frames, we had to decrease the batch size to 2.

B W-Sprites dataset

Here we describe how the *W-Sprites* dataset was synthesized. In particular, we provide details on the random walk used to generate the global motion of the sprite. First, a starting point (x_0, y_0) is sampled uniformly within the frame. At each step i , an action \hat{g}_i is sampled uniformly from the list of available actions: *left*, *right*, *up*, *down* and *stay* (on the edges of the image the corresponding

action is removed from the list). The transition probabilities are given by

$$p(g_i = g_{i-1} | g_{i-1}) = p_{\text{inertia}} \quad (1)$$

$$p(g_i = \hat{g}_i | g_{i-1}) = 1 - p_{\text{inertia}} \quad (2)$$

$$p(x_i = x_{i-1} + s, y_i = y_{i-1} | x_{i-1}, y_{i-1}, g_i = \text{"right"}) = 1 \quad (3)$$

$$p(x_i = x_{i-1} - s, y_i = y_{i-1} | x_{i-1}, y_{i-1}, g_i = \text{"left"}) = 1 \quad (4)$$

$$p(x_i = x_{i-1}, y_i = y_{i-1} + s | x_{i-1}, y_{i-1}, g_i = \text{"down"}) = 1 \quad (5)$$

$$p(x_i = x_{i-1}, y_i = y_{i-1} - s | x_{i-1}, y_{i-1}, g_i = \text{"up"}) = 1 \quad (6)$$

$$p(x_i = x_{i-1}, y_i = y_{i-1} | x_{i-1}, y_{i-1}, g_i = \text{"stay"}) = 1. \quad (7)$$

We set p_{inertia} to 0.9 and s to 7 pixels. The described process generates a sequence of coordinates (x_i, y_i) and global actions g_i . The global actions are further used to animate the sprite. In case of **right**, **left**, **up** and **down** global actions the corresponding walking actions are applied. The **stay** action is animated with one of **slash left**, **slash right**, **slash front**, **spellcast left**, **spellcast right** and **spellcast front** chosen at random.

The same random walk is used to generate the background motion. For the background we set $p_{\text{inertia}} = 0.95$ and $s = 2$. We also restrict the maximum background motion to 25 pixels.

The code used to generate the dataset will be made publicly available on github.

C Additional Visual Examples

In this section we provide some additional qualitative evaluation of our method, that could not be included in the main paper due to the paper length limitations.

Visual results of ablations and prior work. Fig. 3 provides some visual results for the ablations from the main paper. We observe that the full version of GLASS has a more disentangled local action space than under the other settings. We attribute this property to the use of a separate global and local action spaces (see Fig. 9). This separation provides a more efficient and richer action space than in prior work (see Fig. 4). For example, CADDY [5] trained on W-Sprites tends to learn global motion of both the agent and the background while ignoring some local variations. This means that CADDY yields action spaces of the background and the foreground that are tangled.

More reconstruction and transfer examples. We include more examples of reconstruction and motion transfer using GLASS in this section. We start from an original video, which is decoded to a sequence of global and local actions. This sequence is used for both reconstructing the original video from the first frame and transfer the motion to a different scene. The results on the *BAIR* and the *Tennis* datasets are shown in Figs. 5 and 6.

Global action space. In the main paper we included some visualizations of the global action space on the *BAIR* and *Tennis* datasets. Here we provide more videos in order to reiterate the consistency of the global actions learnt by

Plain directions

Gumbel

No modulated convs

Joint input

$\mathcal{L}_{\text{RECS}}$


GLASS 200k

Fig. 3: Visualization of local action spaces per ablation. Each row corresponds to a specific ablation. Each column depicts a certain local action being performed, global actions are set to zero-shift. To play use Acrobat Reader.

Fig. 4: Action space of CADDY [5] trained on the W-Sprites dataset. Each row corresponds to a different initial frame. Column j shows the result of applying action j multiple times in a row. To play use Acrobat Reader.

original
reconstructed
transfer

Fig. 5: Reconstruction and motion transfer examples on the *BAIR* dataset. Note the ability of GLASS to generate very diverse videos from the same initial frame. To play use Acrobat Reader.



original
reconstructed
transfer

Fig. 6: Reconstruction and motion transfer examples on the *Tennis* dataset. Note the ability of GLASS to generate very diverse videos from the same initial frame. To play use Acrobat Reader.

Fig. 7: Global action space visualization on the *BAIR* dataset. Each row starts with the same frame. Each column corresponds to one of the global actions, from left to right: **right**, **left**, **down**, **up** and **stay**. To play use Acrobat Reader.

GLASS. We sequentially feed the same global shift to the model along with a fixed local action. The resulting 8 frames long videos are shown in Figs. 7, 8 and 9.

Local action space. Here we provide some visualizations of the local action space learnt by GLASS on the different datasets. In Figs. 10, 11 and 12 we show the first frame of the video as well as the result of applying different local actions. We sequentially feed the same local action to the model along with the (0.0, 0.0) global action to keep the agent static. The 8th frame of the resulting sequence is shown. We fit 2, 4 and 6 local actions on the *BAIR*, *Tennis* and *W-Sprites* datasets respectively.

Fig. 8: Global action space visualization on the *Tennis* dataset. Each row starts with the same frame. Each column corresponds to one of the global actions, from left to right: **right**, **left**, **down**, **up** and **stay**. To play use Acrobat Reader.

Fig. 9: Global action space visualization on the *W-Sprites* dataset. Each row starts with the same frame. Each column corresponds to one of the global actions, from left to right: **right**, **left**, **down**, **up** and **stay**. To play use Acrobat Reader.

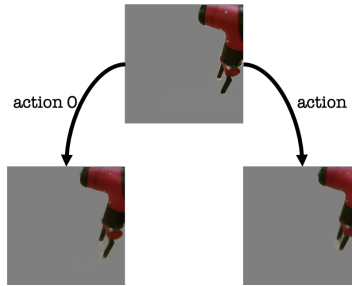


Fig. 10: Demonstration of the resulting images after applying different local actions on the *BAIR* dataset. The actions capture some local deformations of the robot arm, i.e. the state of the manipulator (**open** / **close**).

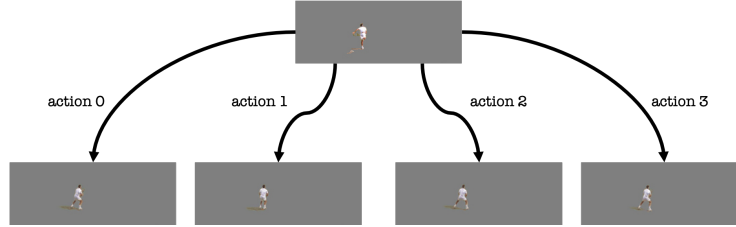


Fig. 11: Demonstration of the resulting images after applying different local actions on the *Tennis* dataset. The actions capture some small variations of the pose of the tennis player, such as rotation and the distance between the legs. This helps GLASS generate more realistic motions than CADDY and other competitors, e.g. running player (see Fig. 6)

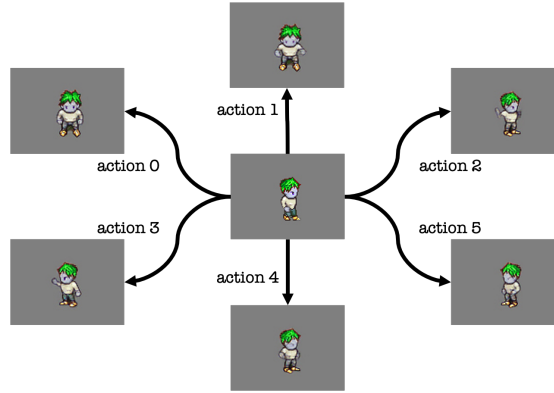


Fig. 12: Demonstration of the resulting images after applying different local actions on the *W-Sprites* dataset. The local actions learnt by the model can be interpreted as `turn front`, `slash front`, `spellcast`, `slash left`, `turn right`, `turn left`.

References

1. Hochreiter, S., Schmidhuber, J.: Long short-term memory. *Neural computation* **9**(8), 1735–1780 (1997)
2. 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)
3. 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)
4. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014)
5. 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)