Layered Controllable Video Generation

Supplementary Material

This supplementary material is structured as follows. In Section A, we present our architectural details of our models, and the implementation details of our training/ testing procedures. In Section B, we show visual results of the frames generated by our model under different conditions. In Section C, we show additional ablation results to prove the effectiveness of our two staged training setup. In Section D, we show soft masks could lead to degenerated results. Our video results can be found at this website.

A Implementation Details

Architecture details. Our model consists of four main modules: the *Mask Network* $\mathcal{M}$, the *Encoder* $\mathcal{E}$, the learnable discrete code book $\mathcal{Z}$, and the *Decoder* $\mathcal{D}$. We summarize the details in Table 3. In total, our model has 47.8 M parameters, and takes about 191.2 MB to store on disk.

Training Details. In all our experiments, we use the Adam Optimizer with a fixed learning rate of 4.5×10^{-6}, which we found empirically.

- *Initial training for mask-based generation:* For this stage, we train all models for $10,000$ iterations. For experiments on the *BAIR* dataset, we initialize the ratio between $\lambda_{\text{bg}} : \lambda_{\text{fg}}$ to be $80 : 1$, and reduce it by a factor of 2 for every $1,000$ iterations until this ratio becomes $5 : 1$. For experiments on the *Tennis* dataset, this ratio is initialized to $10 : 1$ and reduced by half after $5,000$ iterations.
- *Fine-tuning for controllability:* In this stage of training, we drop all losses related to the mask regularization as their gradient becomes zero (see Section 3.2), except for $\mathcal{L}_{\text{bg}}$. On the *BAIR* dataset, λ_{bg} is set to 5, and on the *Tennis* dataset, it's set to 0.5.
- *Finding the "ground-truth" control signal:* In Equation. 19, we use the fully differentiable transformation operation $\mathcal{T}(\theta^t)$ to approximate the mask of the ground truth next frame using the mask of the current frame. For each frame, we optimize the affine transformation matrix that's been used to transform $\mathbf{m}^t$ into $\mathbf{m}_c^t$, the optimization is performed by the ADAM optimizer for $1,000$ iterations with the learning rate of 0.1.
  On the *BAIR* dataset, our model requires 80GB GPU memory and 40 hours to train on 4× RTX-6000 (24GB V-RAM per GPU) GPUs, and on the *Tennis* daaset, the model takes 22GB GPU memory and 23 hours to train on a single RTX-3090 (24GB V-RAM).

Table 3. Architecture details of the main modules of our model. The high-level design follows the architecture presented in [15]. K denotes the number of entries in the code book, n_z is the dimension of each entry, $h, w = (H, W)/4$ and $h', w' = (H, W)/16$. For the discriminator $\mathcal{C}$, we use the vanilla PatchGAN discriminator as described in [25].

Mask Network $\mathcal{M}$
$x \in \mathbb{R}^{H \times W \times 3}$
Conv2D $\rightarrow \mathbb{R}^{H \times W \times 64}$
2×{Residual Blocks + Downsample} $\rightarrow \mathbb{R}^{h \times w \times 256}$
9 × Residual Blocks $\rightarrow \mathbb{R}^{h \times w \times 256}$
2×{Upsample + Residual Blocks }$\rightarrow \mathbb{R}^{h \times w \times 64}$
Conv2D + Sigmoid$\rightarrow \mathbb{R}^{H \times W \times 1}$
Encoder $\mathcal{E}$
$x \in \mathbb{R}^{H \times W \times 3}$
Conv2D $\rightarrow \mathbb{R}^{H \times W \times 256}$
4×{Residual Blocks + DownSample} $\rightarrow \mathbb{R}^{h' \times w' \times 64}$
Residual Block $\rightarrow \mathbb{R}^{h' \times w' \times 64}$
Attention Block $\rightarrow \mathbb{R}^{h' \times w' \times 64}$
Residual Block $\rightarrow \mathbb{R}^{h' \times w' \times 64}$
GroupNorm, SiLu, Conv2D $\rightarrow \mathbb{R}^{h' \times w' \times n_z}$
Code book $\mathcal{Z}$
$K = 1024$
$n_z = 256$
Decoder $\mathcal{D}$
$\mathbf{q}(\mathbf{z}) \in \mathbb{R}^{h' \times w' \times n_z}$
Conv2D $\rightarrow \mathbb{R}^{h' \times w' \times 64}$
Residual Block $\rightarrow \mathbb{R}^{h' \times w' \times 64}$
Attention Block $\rightarrow \mathbb{R}^{h' \times w' \times 64}$
Residual Block $\rightarrow \mathbb{R}^{h' \times w' \times 64}$
4×{Residual Blocks + DownSample} $\rightarrow \mathbb{R}^{H \times W \times 256}$
GroupNorm, SiLu, Conv2D $\rightarrow \mathbb{R}^{H \times W \times 3}$

B Additional qualitative results

Comparing with other baselines. In Figure 9 and Figure 10, we show the comparison of the three variants of our model against other baselines that we mentioned in the main text, on the *BAIR* and *Tennis* datasets respectively. Noticeably on both datsets, our model achieves the most precise control over the generated video (see how the generated motion corresponds to the ground truth sequences), meanwhile generating one of the best frame-quality videos. For example, CADDY fails to control the robot arm, SAVP+ controls it up to some degree, but as shown in frames t=10 and t=15, their control is not as precise, whereas all of our variants provide highly accurate control.

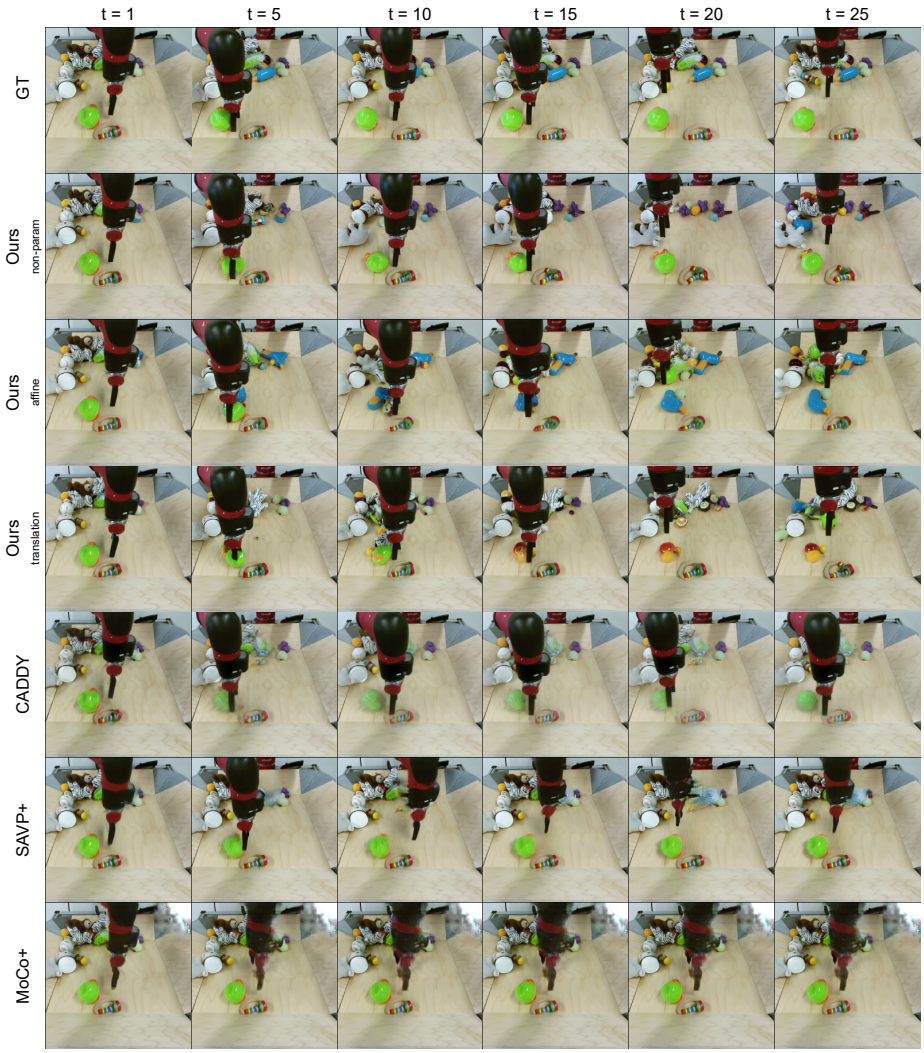

Fig. 9. Comparing with other baselines on the *BAIR* dataset.

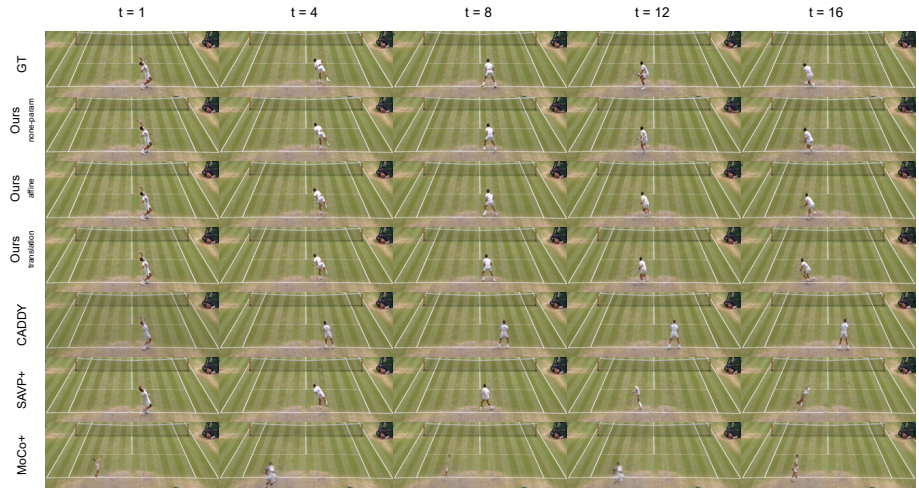

Fig. 10. Comparing with other baselines on the *Tennis* dataset.

Generating videos conditioned on user inputs. In Figure 11 and Figure 12, we show how the video generated by our model responds to the user inputs, on the *BAIR* and *Tennis* datasets respectively. Here, for each sample, we keep applying the same control signal (moving left/right/up/down) to the model so that it can generate video sequences with single consistent motions.

Action mimicking. In Figure 13, we show that our method can be used to extract the motion from a driving sequence and then apply onto different appearances (staring frames).

Control of Multiple Objects. In Figure 15, our method is able to generate videos with multiple moving objects by overlaying individually controlled mask sequences.

Video results. For more video results, please refer to the attached readme.html, where we demonstrate all above mentioned results in video format, as well as other applications of our method, such as generating videos of multi players, real-time user controllable generation demo, and animating a single frame with different motions.

C Additional Ablation results - Single Stage Training

As mentioned in our main text, breaking down our training procedure into two stages is a crucial design in our setup. In stage I, our model focuses on generating a FG/BG segmentation mask, and in stage II, we make edits to this mask to finetune the network for controlled generation. Training the model in one single stage wouldn't allow us to introduce controllability, without stage II finetuning, the model is extremely vulnerable to any changes to the mask. As shown on Figure 14, on the top row, the model is tested using ground truth masks to

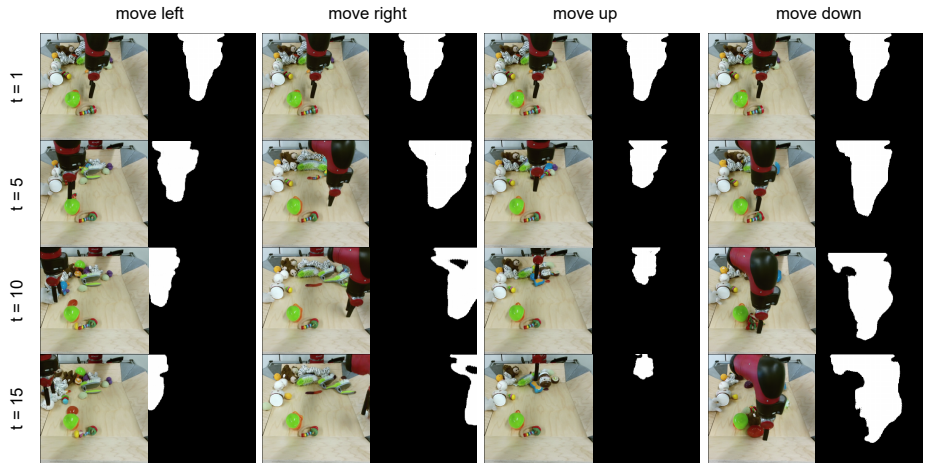

Fig. 11. Generating videos conditioned on user inputs, *BAIR* dataset.

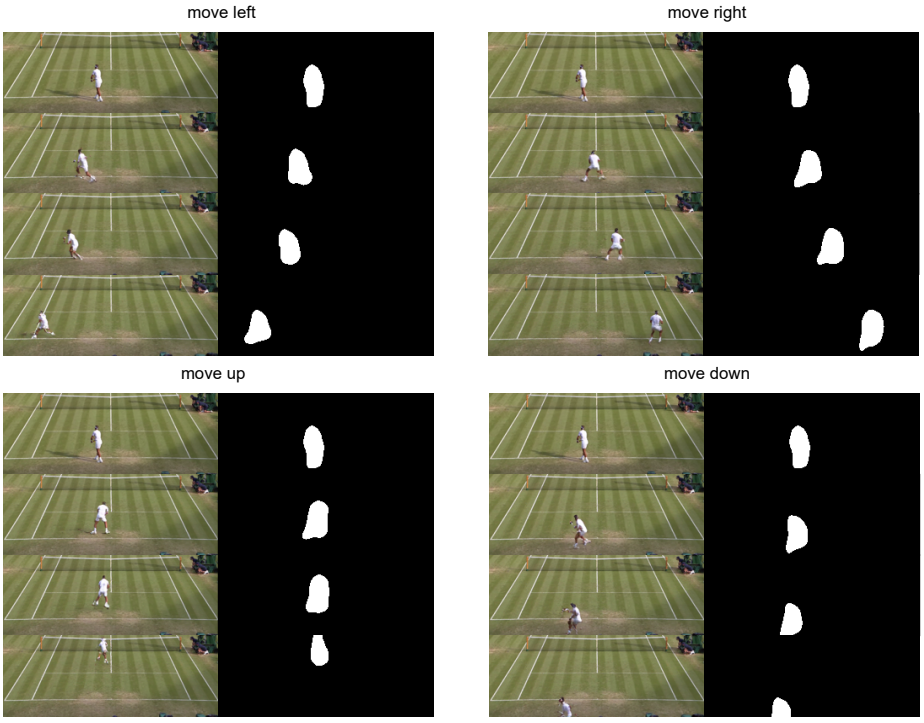

Fig. 12. Generating videos conditioned on user inputs, *Tennis* dataset.

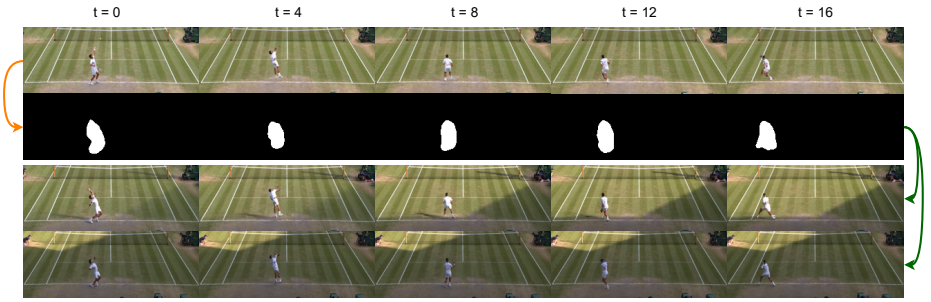

Fig. 13. "Mimicking". Our model can be used to extract the motion from a driving sequence and then apply onto different appearances (staring frames). The mask sequence is extracted from the driving video (first row), then applied onto two other staring frames (third, fourth row).

generate future frames (same as single stage training setup), and the model performed well. However, on the bottom row, when we try to use pseudo control signals to transform the mask, then use the transformed mask to condition the generation (same as real-world testing for controlled video generation), the model performs poorly.

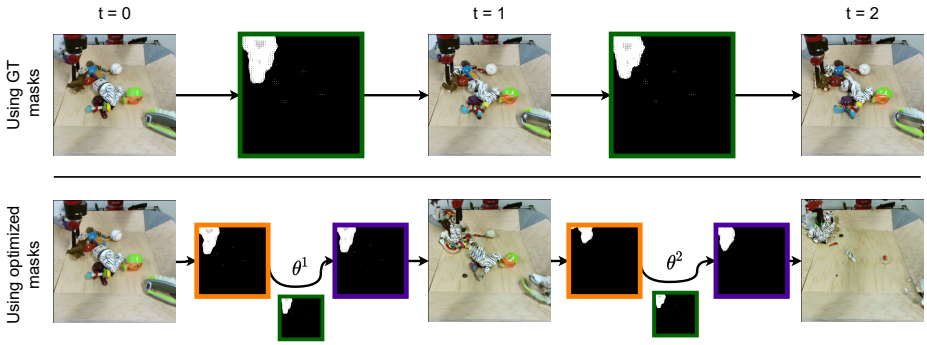

Fig. 14. Single stage training results in poor testing performance when control is introduced. Green represents GT next frame mask, orange represents generated current frame's mask, and purple represents transformed mask. On the top row, the model is tested using ground truth masks to generate future frames (same as how the model was trained). On the bottom row, the model is tested using pseudo control signals (same as real-world testing). The performance of the single stage trained model drops significantly when there are edits to the mask.

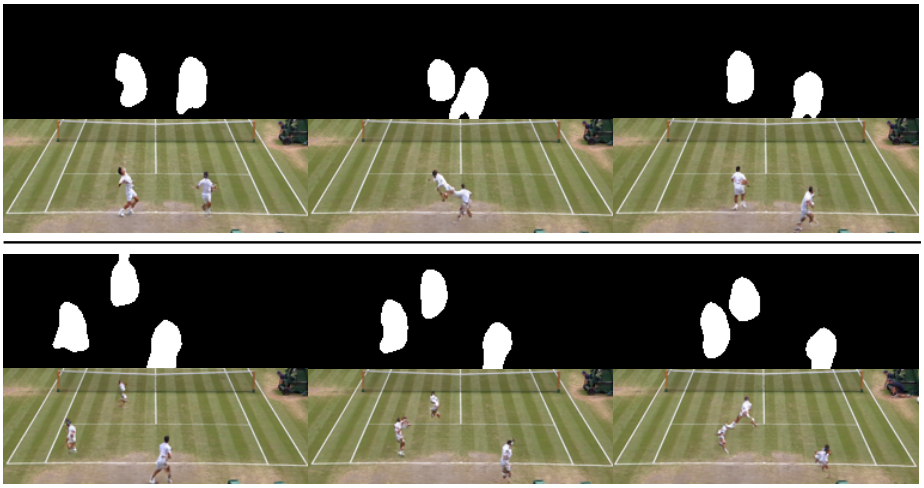

Fig. 15. Generating videos of multiple players, *Tennis* dataset.

D Additional Ablation results - Soft Masks

As shown in Fig. 16, without mask binarization, information is leaked through the soft masks, shifting the mask will not only affect the foreground, but the entire scene, which results in degeneration of the frame.

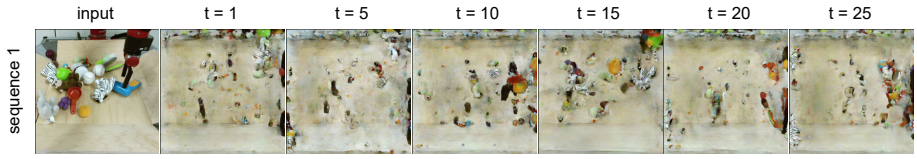

Fig. 16. Soft masks result in degenerated frames.