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 in under different conditions. In Section C, we discuss some potential social implications of our method. We also include a readme.html to summarize our work and to show some video results.

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 3.3), 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. 5, 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 $4\times$ 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).

Mask Network $\mathcal{M}$
$x \in \mathbb{R}^{H\times W\times 3}$
Conv2D $\rightarrow \mathbb{R}^{H\times W\times 64}$
$2\times\{$Residual Blocks + Downsample$\} \rightarrow \mathbb{R}^{h\times w\times 256}$
$9\times$ Residual Blocks $\rightarrow \mathbb{R}^{h\times w\times 256}$
$2\times\{$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 $\rightarrow \mathbb{R}^{H\times W\times 256}$
$4\times\{$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 $\mathcal{D}$
$\mathsf{q}(\mathbf{z}) \in \mathbb{R}^{h'\times w'\times n_z}$
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 $\rightarrow \mathbb{R}^{h'\times w'\times 64}$
$4\times\{$Residual Blocks + DownSample$\} \rightarrow \mathbb{R}^{H\times W\times 256}$
GroupNorm, SiLu, Conv2D $\rightarrow \mathbb{R}^{H\times W\times 3}$

Table 3. Architecture details of the main modules of our model. The high-level design follows the architecture presented in [14]. 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].

B. Additional qualitative results

Comparing with other baselines. In Figure 8 and Figure 9, 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.

Generating videos conditioned on user inputs. In Figure 10 and Figure 11, 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 12, we show that our method can be used to extract the motion from a driving sequence and then apply onto different appearances (staring frames).

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. Broader Impact

Our work enables generating videos conditioned on the on-the-fly user inputs at the frame-level.

Since it is also an unsupervised method, unlike other works on conditioned video generation, it does not require any kind of data annotations while training. Thus our model can be applied to a large variety of raw video datasets. In other words, our method has the potential to widen the application area of conditional video generation, which may increase the impact this line of research has.

Like other generative models, our method shares similar benefits and risks, which have been discussed extensively in [7, 50].

We acknowledge that our method can be used to generate videos that involve humans, as shown in our Tennis example, that may have potential for misuse. While it is unlikely that our method can be used create videos of completely novel scenes and objects, such as in the case of deep fakes, it can be used to generate a video of action combinations that did not happen in reality. At the same time, we see the potential benefits of having controllable generative models, for example, allowing simulations without the need of an explicit model, or to be used as a data augmentation tool that providing both frame data and action annotations. As concluding remark, we would like to emphasize the importance of research on detecting videos and images generated from deep models [20, 59] to further mitigate any potential misuse.

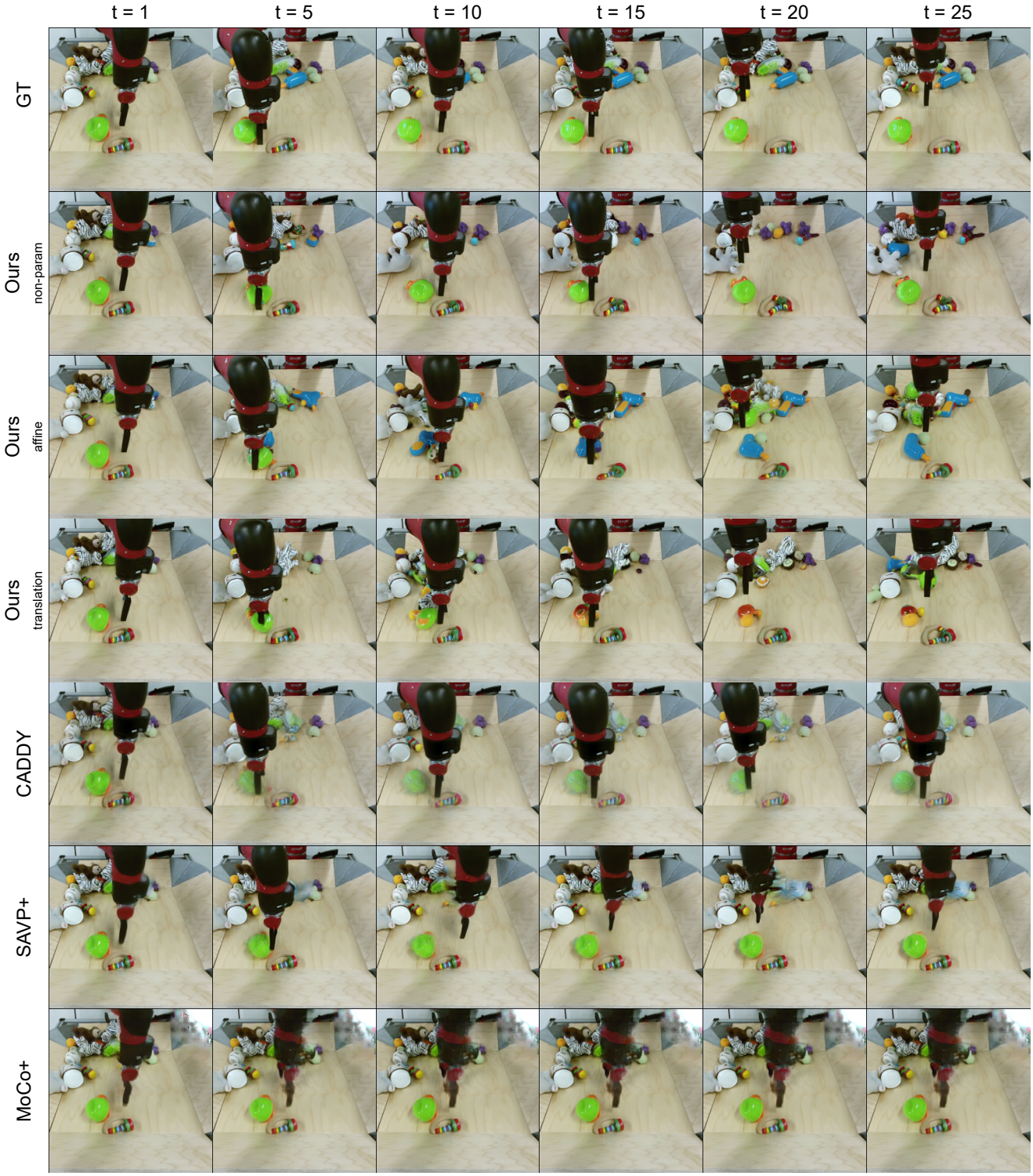

Figure 8. Comparing with other baselines on the *BAIR* dataset.

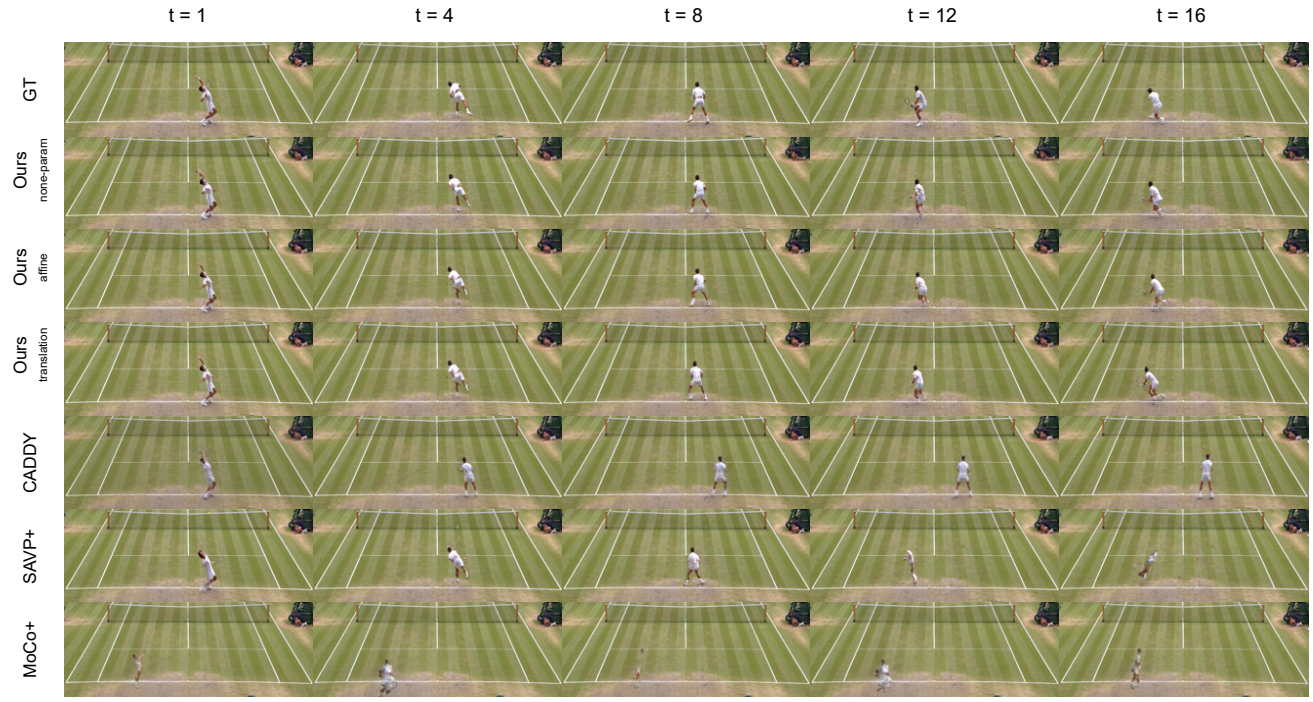

Figure 9. Comparing with other baselines on the *Tennis* dataset.

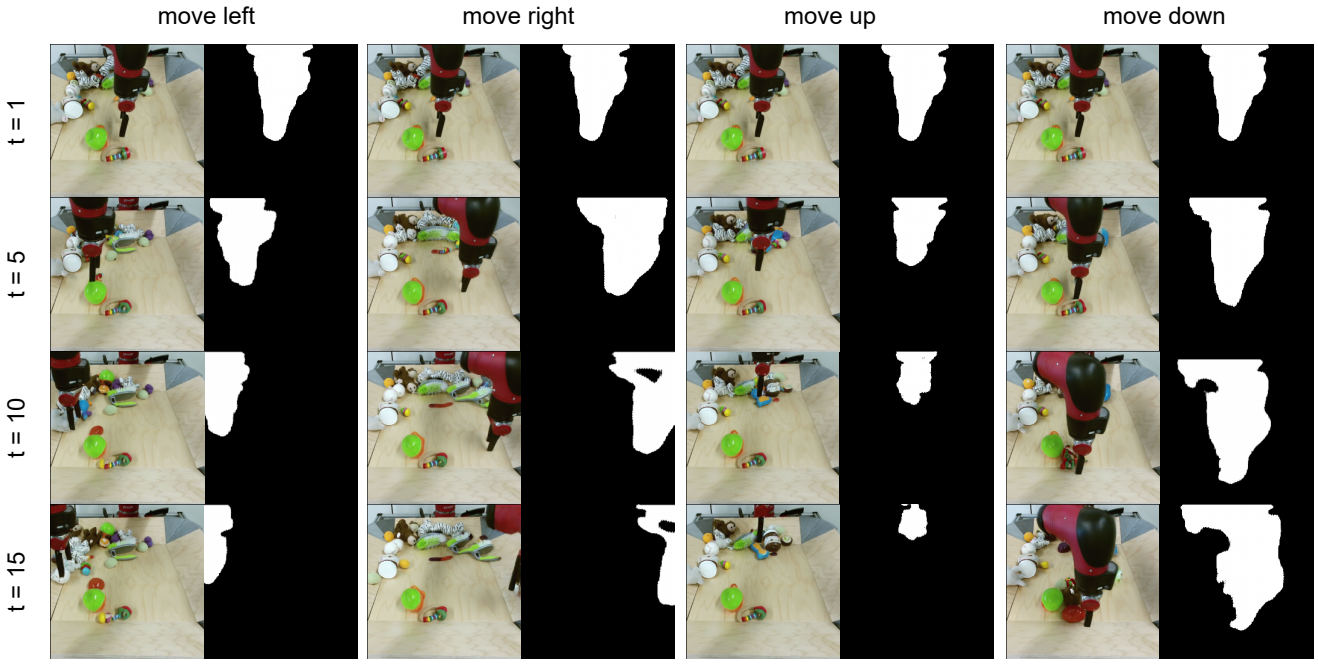

Figure 10. Generating videos conditioned on user inputs, *BAIR* dataset.

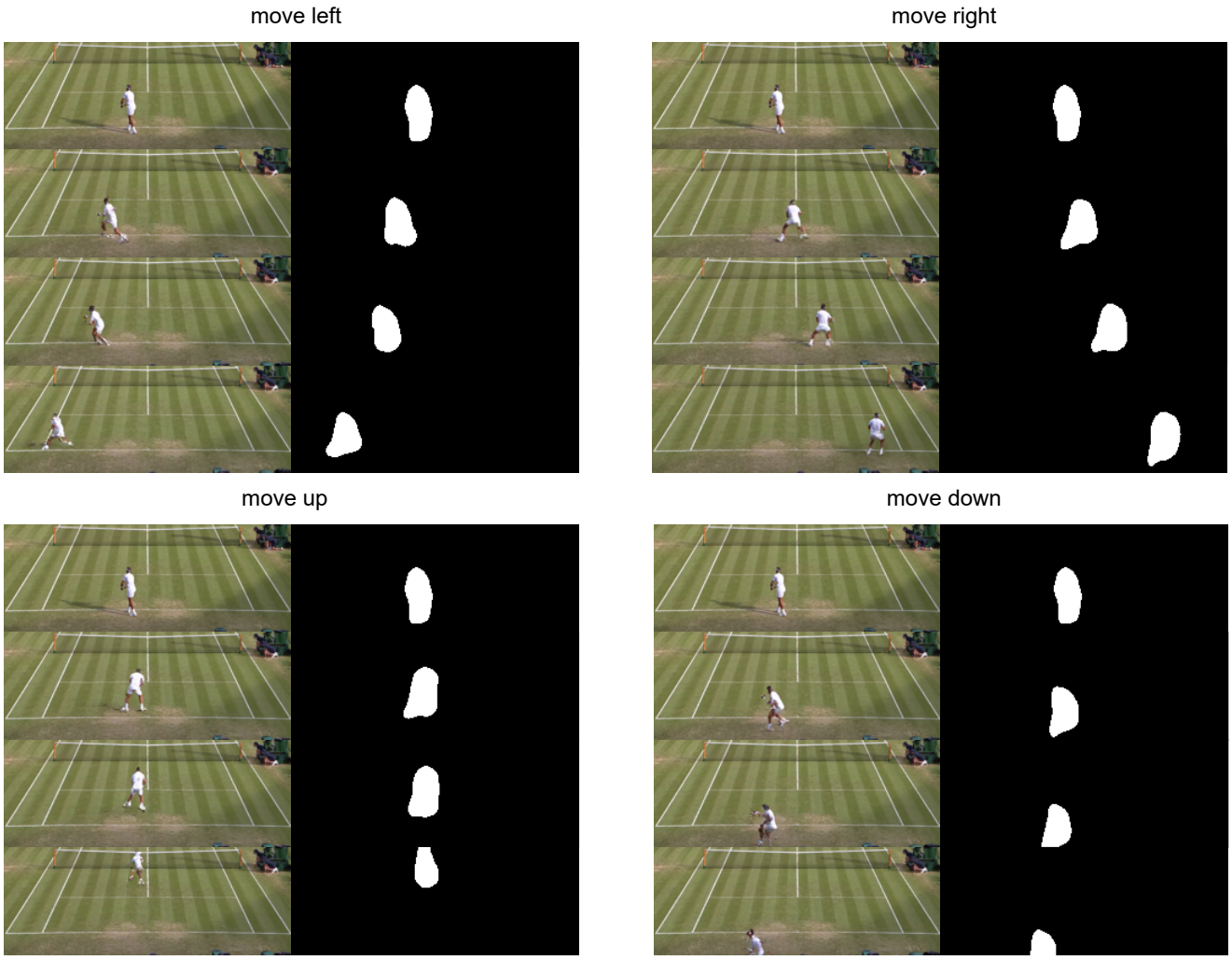

Figure 11. Generating videos conditioned on user inputs, *Tennis* dataset.

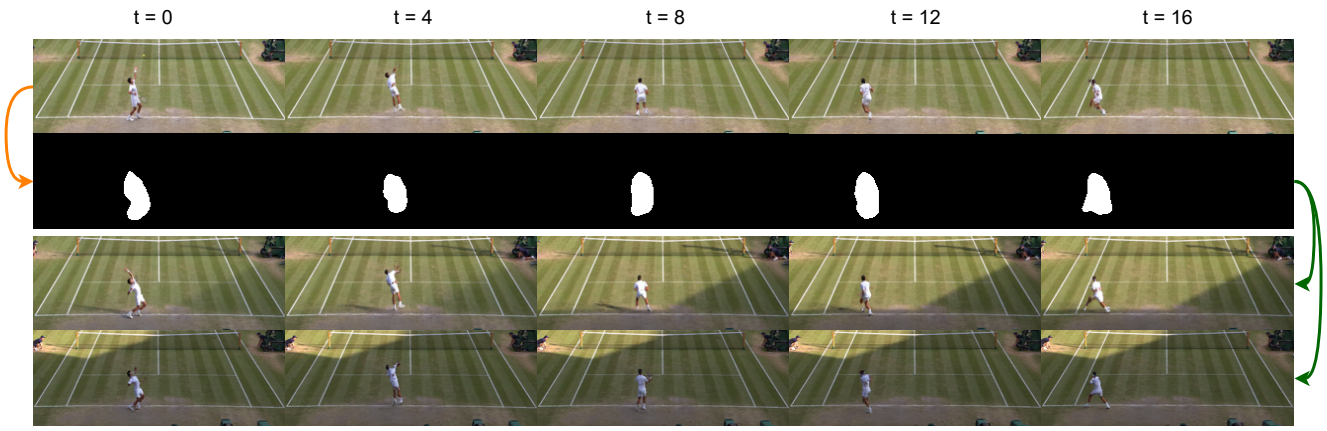

Figure 12. "Mimicking". Our model can be used to extract the motion from a driving sequence and then apply onto different appearances (staring frames). The mask sequence is extracted from the driving video (first row), then applied onto two other staring frames (third, fourth row).