Learn the Force We Can: Enabling Sparse Motion Control in Multi-Object Video Generation

Aram Davtyan, Paolo Favaro

Computer Vision Group, Institute of Informatics, University of Bern, Switzerland
aram.davtyan@unibe.ch, paolo.favaro@unibe.ch

Abstract

We propose a novel unsupervised method to autoregressively generate videos from a single frame and a sparse motion input. Our trained model can generate unseen realistic object-to-object interactions. Although our model has never been given the explicit segmentation and motion of each object in the scene during training, it is able to implicitly separate their dynamics and extents. Key components in our method are the randomized conditioning scheme, the encoding of the input motion control, and the randomized and sparse sampling to enable generalization to out of distribution but realistic correlations. Our model, which we call YODA, has therefore the ability to move objects without physically touching them. Through extensive qualitative and quantitative evaluations on several datasets, we show that YODA is on par with or better than state of the art video generation prior work in terms of both controllability and video quality.

Introduction

Imagining the potential outcomes of own and others’ actions is considered the most powerful among the three levels of cognitive abilities that an intelligent system must master according to (Pearl and Mackenzie 2018). Imagining is particularly valuable. It allows organisms to understand and analyse possible futures, which enables them to plan ahead and significantly increases their chances of survival. Thus, including the capability of imagining seems a natural choice for the design of intelligent systems.

In this work, we aim at building models for video generation that learn to imagine the outcome of sparse control inputs in multi-object scenes. Learning is allowed only through the passive observation of a video dataset, i.e., without the ability to explicitly interact with the scene, and without any manual annotation. We aim to build a model that can show some degree of generalization from the training data, i.e., that can generate data that was not observed before, that is directly related to the control input, and that is plausible, as shown in Figure 1. One possible choice of controls expressed in natural language could take the form “Move this object at this location”. However, such controls assume that the objects in an image are already known.

Since we do not use annotations, it is quite difficult to obtain an explicit scene decomposition into objects. Therefore, we opt for representing the controls as 2D shifts at pixels and let the model implicitly learn the extents of the objects during training by observing correlated motion of the pixels in real videos. Thus, the control inputs we consider correspond instead to sentences of this form: “Move this pixel (and the corresponding underlying object) at this location”. The model is fed with the current frame, context frames (a subset of the past frames to enable memory), and a set of control inputs. The output is a generated subsequent frame that shows how the input frame would change under the specified motion. We train our model without specifying what objects are (i.e., we do not use information about object categories), or where they are (i.e., we do not rely on bounding boxes, landmarks/point annotations or segmentation masks), or how they interact (i.e., we do not make use of information on object relationships or action categories or textural descriptions of the scenes). Given two subsequent frames (the current and following one) from a video in our training data, we obtain the motion control input by sampling the estimated optical flow at a few (typically 5) locations. To generate frames we use flow matching (Lipman et al. 2022; Liu, Gong, and Liu 2022; Albergo and Vanden-Eijnden 2022), where the model is conditioned on the current and past frames (Davtyan, Sameni, and Favaro 2023), and feed the motion control input through cross-attention layers in a transformer architecture.

As shown in Figure 1, an emerging property of our proposed approach is that the model learns the physical extent of objects in the scene without ever requiring explicit supervision for it. For example, although the control is applied to the handle of the brush in the second row, the motion is applied correctly to the whole brush, and to the whole brush only. A second learned property can be observed on the first row: When the robot arm is driven towards other objects, it interacts with them realistically. In the second row, we can observe a third remarkable capability that the model has learned. The generated video shows that we can directly rotate a single object, without using the robot arm to do so. This is a video that has never been observed in the dataset (all objects are moved directly by the arm or indirectly via other objects). Because of this ability of moving objects without touching them, we call our approach YODA.
It does demonstrate empirically that the model has the ability to imagine novel plausible outcomes when the reality is modified in ways that were not observed before.

Our contributions can be summarized as follows

1. We introduce a model for controllable video synthesis that is trained in a completely unsupervised fashion, is not domain-specific, and can scale up to large datasets;
2. We introduce an effective way to embed motion information and to feed it to the model, and show analysis to understand the impact of sampling and the use of sparsity of the motion field. We conclude that all these components are crucial to the correct learning of object interactions and the disentanglement of object motion control;
3. We demonstrate for the first time multi-object interactions in the unsupervised setting on real data, which has not been shown in other state of the art methods (Menapace et al. 2021; Blattmann et al. 2021b,a).

Prior Work

Video generation. An increased interest in video generation has followed the success of generative models for images (Karras et al. 2021; Esser, Rombach, and Ommer 2021; Dhariwal and Nichol 2021; Rombach et al. 2022). In contrast to image generation, video generation is plagued by problems such as rendering realistic motion, capturing diversity (i.e., modeling the stochasticity of the future outcomes) and, most importantly, managing the high computational and storage requirements. Conventional approaches to video generation are autoregressive RNN-based models (Babaeizadeh et al. 2021; Blattmann et al. 2021b). Other models instead directly generate a predefined number of frames (Ho et al. 2022; Blattmann et al. 2021a; Singer et al. 2022). Variability (stochasticity) of the generated sequences has been tackled with GANs (Clark, Donahue, and Simonyan 2019; Luc et al. 2020), variational approaches (Babaeizadeh et al. 2018; Denton and Fergus 2018; Babaeizadeh et al. 2021; Lee et al. 2021), Transformers (Le Moing, Ponce, and Schmid 2021; Rakhimov et al. 2020; Gupta et al. 2022; Yan et al. 2021) and diffusion-based approaches (Ho et al. 2022; Voleti, Jolicoeur-Martineau, and Pal 2022; Höppe et al. 2022; Harvey et al. 2022). The recently proposed autoregressive method RIVER (Davtyan, Sameni, and Favaro 2023) deals with the stochastic nature of the generative process through flow matching (Lipman et al. 2022). Among all above approaches, YODA uses RIVER as a backbone, because of its efficiency and ease of training.

Controllable video generation mostly differs in the nature of the control signals. Control can be defined per frame (Chiappa et al. 2017; Finn, Goodfellow, and Levine 2016; Kim et al. 2020; Nunes et al. 2020; Oh et al. 2015; Hu et al. 2021), or as a global label (Wang et al. 2020; Singer et al. 2022). Some of them are obtained via supervision (Chiappa et al. 2017; Finn, Goodfellow, and Levine 2016; Kim et al. 2020; Nunes et al. 2020; Oh et al. 2015), or discovered in an unsupervised manner (Menapace et al. 2021; Davtyan and Favaro 2022; Huang et al. 2022; Rybkin et al. 2018). For instance, CADDY (Menapace et al. 2021) learns a discrete action code of the agent that moves in the videos. Another model, GLASS (Davtyan and Favaro 2022) decouples the actions into global and local ones, where global actions, as in YODA, are represented with 2D shifts, while local actions are discrete action codes, as in (Menapace et al. 2021). In (Huang et al. 2022) the authors explicitly separate the foreground agent from the background and condition the generation on the transformations of the segmentation mask. However, all these models are restricted to single agent videos, while YODA successfully models multiple objects and their interactions. (Hu et al. 2021; Hao, Huang, and Belongie 2018; Blattmann et al. 2021b,a) are the most similar works to ours as they specify motion control at the pixel-level. However, (Hu et al. 2021) leverages a pre-trained object detector to obtain the ground truth control. (Hao, Huang, and Belongie 2018) is based on warping and therefore does not incorporate memory to model long-range consequences of actions. II2V (Blattmann et al. 2021b) uses a hierarchical RNN to allow modeling higher-order details, but focuses on deterministic prediction. iPOKE (Blattmann et al. 2021a) aims to model stochasticity via a conditional invertible neu-
ral network, but has to sacrifice the ability to generate long videos and to intervene into the generation process at any timestamp. None of those works has demonstrated controllable video generation on multi-object real scenes.

**Multi-object scenes and interactions.** Modeling multiple objects in videos and especially their interactions is an extremely difficult task. It either requires expensive human annotations (Hu, Luo, and Chen 2022) or is still limited to simple synthetic scenes (Wu, Yoon, and Ahn 2021; Schmeckpeper, Georgakis, and Daniilidis 2021; Janner et al. 2018). (Menapace et al. 2022; Yan et al. 2022) allow for multi-agent control, but leverage ground truth bounding boxes during the training. YODA in turn is an autoregressive generative model for controllable video generation from sparse motion controls that i) efficiently takes memory into account to simulate long-range outcomes of the actions, ii) models stochasticity of the future, iii) does not require human annotation for obtaining the control signal(s), and iv) demonstrates controllability on a complex multi-object real dataset.

**Training YODA**

We denote with \( x = \{x^1, \ldots, x^N\} \) an RGB video that contains \( N \) frames, where \( x^i \in \mathbb{R}^{3 \times H \times W} \), \( i = 1, \ldots, N \), and \( H \) and \( W \) are the height and the width of the frames respectively. The goal is to build a controllable video prediction method that allows us to manipulate separate objects in the scene. We formulate this goal as that of approximating a sampler from the following conditional distribution

\[
p(x^{k+1}|x^k, x^{k-1}, \ldots, x^1, a^k),
\]

for \( k < N \) and where \( a^k \) denotes the motion control input. \( a^k \) specifies the desired shifts at a set of pixels (including the special cases with a single pixel or none). Our ultimate objective is to ensure through training that this control implicitly defines the shift(s) for the object(s) containing the selected pixel(s). The conditioning in eq. (1) allows an autoregressive generative process at inference time, where the next frame \( x^{k+1} \) in a generated video is sampled conditioned on the current frame \( x^k \), the previously generated frames \( x^{k-1}, \ldots, x^1 \) and the current control \( a^k \).

To model the conditional distribution in eq. (1), we use RIVER (Davtyan, Sameni, and Favaro 2023). This is a recently proposed video prediction method based on conditional flow matching (Lipman et al. 2022). We chose RIVER due to its simplicity and training efficiency compared to conventional RNNs (Babaieizadeh et al. 2021; Hochreiter and Schmidhuber 1997; Babaieizadeh et al. 2018; Denton and Fergus 2018; Lee et al. 2018) and Transformers (Weissenborn, Täckström, and Uszkoreit 2019; Le Moing, Ponce, and Schmid 2021; Rakhimov et al. 2020; Gupta et al. 2022; Seo et al. 2022) for video prediction. For completeness, we briefly introduce Flow matching and RIVER in section . In section we show how the latter is adapted to handle control. In section , we focus on how the control signals are obtained and encoded.

**Preliminaries: Flow Matching and RIVER**

Flow matching (Lipman et al. 2022; Liu, Gong, and Liu 2022; Albergo and Vanden-Eijnden 2022) was introduced as a simpler, more general and more efficient alternative to diffusion models (Ho, Jain, and Abbeel 2020). The goal is to build an approximate sampler from the unknown data distribution \( q(y) \), given a training set of samples of \( y \). This is formalized as a continuous normalizing flow (Chen et al. 2018) via the following ordinary differential equation

\[
\dot{\phi}_t(y) = v_t(\phi_t(y)), \quad \phi_0(y) = y.
\]

Eq. (2) defines a flow \( \phi_t(y) : [0, 1] \times \mathbb{R}^d \rightarrow \mathbb{R}^d \) that pushes \( p_t(y) = N(y | 0, 1) \) towards the distribution \( p_1(y) \approx q(y) \) along the vector field \( v_t(y) : [0, 1] \times \mathbb{R}^d \rightarrow \mathbb{R}^d \). Remarkably, (Lipman et al. 2022) shows that one can obtain \( v_t(y) \) by solving

\[
\min_{v_t} \mathbb{E}_{t, p_t(y \mid y_1), q(y_1)} ||v_t(y) - u_t(y \mid y_1)||^2,
\]

where one can explicitly define the vector field \( u_t(y \mid y_1) \) and its corresponding probability density path \( p_t(y \mid y_1) \), with \( y_1 \sim q(y) \). A particularly simple choice (Lipman et al. 2022) is the Gaussian probability path \( p_t(y \mid y_1) = \mathcal{N}(y | \mu_t(y_1), \sigma_t^2(y_1)) \), with \( \mu_0(y_1) = 0, \mu_1(y_1) = y_1, \sigma_0(y_1) = 1, \sigma_1(y_1) = \sigma_{\text{min}} \). The corresponding target vector field is then given by

\[
u_t(y \mid y_1) = \frac{y_1 - (1 - \sigma_{\text{min}})y}{1 - (1 - \sigma_{\text{min}})t} \].

Sampling from the model that was trained to optimize (3) can be obtained by first sampling \( y_0 \sim \mathcal{N}(y \mid 0, 1) \) and then by numerically solving eq. (2) to obtain \( y_1 = \phi_t(y_0) \).

**RIVER** (Davtyan, Sameni, and Favaro 2023) is an extension of the above procedure to the video prediction task with a computationally efficient conditioning scheme on past frames. The training objective of RIVER is given by

\[
L_\theta(y) = ||v_t(x \mid x^{\tau-1}, x^c, \tau - c; \theta) - u_t(x \mid x^{\tau})||^2,
\]

where \( v_t \) is a network with parameters \( \theta \), \( x^\tau \) is a frame randomly sampled from the training video, \( c \) is an index randomly sampled uniformly in the range \( \{1, \ldots, \tau - 2\} \) and \( u_t \) is calculated with eq. (4). An additional information provided to \( v_t \) is the time interval \( \tau - c \) between the target frame \( x^{\tau} \) and the past frame \( x^c \), which we call the context frame.

At test time, during the integration of eq. (2), a new context frame \( x^c \) is sampled at each step \( t \). This procedure enables to condition the generation of the next frame on the whole past. To further speed up the training and enable high-resolution video synthesis, RIVER works in the latent space of a pretrained VQGAN (Esser, Rombach, and Ommer 2021). That is, instead of \( x^\tau, x^{\tau-1}, x^c \) in eq. (5) one should write \( z^\tau, z^{\tau-1}, z^c \), where \( z^k \) is the VQ latent code of the \( i \)-th frame (Davtyan, Sameni, and Favaro 2023). Since the use of VQGAN encoding is an optional and separate procedure, we simply use \( x \) in our notation.

**Learning to Master the Force**

We now show how to incorporate control into eq. (5) to build a sampler for the conditioning probability in eq. (1). To do
so, we make \( v_\tau \) depend on \( a^{\tau-1} \), which is the motion control at time \( \tau - 1 \), as shown in the following objective

\[
L_{\text{c}}(\theta) = \| v_\tau(x | x^{\tau-1}, x^c, \tau - c, a^{\tau-1}; \theta) - u_\tau(x | x^\tau) \|^2.
\]

In practice, we implement this conditioning by substituting the bottleneck in the self-attention layers of the U-ViT (Bao et al. 2022) architecture of RIVER with cross-attention blocks (for detailed description of the architecture, see supplementary material). The control inputs \( a^{\tau-1} \) are obtained by splitting the image domain into a grid of tiles, so that a motion control can be specified in each tile via a code, and then be fed as keys and values to the cross-attention layer. More details on \( a^{\tau-1} \) are provided in the next section.

Inspired by the classifier-free guidance for diffusion models (Ho and Salimans 2022), we propose to switch off the conditioning on both the context and motion control at every iteration, with some probability \( \pi \) (see Algorithm 1). This is done by using noise as the code corresponding to a switched off context or a motion control. A typical value for \( \pi \) in our experiments is 0.5. This procedure yields two important outcomes: 1) a model that makes use of the conditioning signals and 2) a model that can generate a video by starting from a single frame. As evidence of the outcome 1), note that the conditioning on both the context frame \( x^c \) and the control \( a^{\tau-1} \) is often redundant, because in many instances the future frame can be reliably predicted given only one of the two conditioning signals. In these cases, the model could learn to use only one of them. By randomly switching off the conditions, we force the model to always use both of them, as it does not know which one might be missing. In addition, this training procedure enables the model to change control on the fly. To see 2), consider that when the model generates the first predicted frame, there is no valid context frame and our procedure allows us to replace the context frame with noise. In alternative, one would have to duplicate the first frame, for example, but this would result in an undesirable sampling bias.

**Force Embeddings**

Ideally, \( a^{\tau-1} \) could encode detailed motion information for the objects in the scene. For instance, \( a^{\tau-1} \) could describe that an object is rotating or pressing against another object or walking (in the case of a person). Such supervision could potentially provide the ability to control the video generation in detail and to generalize well to unseen object motion combinations. However, obtaining such ground truth control signals requires costly large-scale manual annotation. Similarly to (Blattmann et al. 2021a), we avoid such costs by leveraging optical flow. Essentially, instead of using a costly and detailed motion representation, we use a simpler one that can be computed automatically and at a large scale.

Given an optical flow \( w^\tau \in \mathbb{R}^{2 \times H \times W} \) between the frames \( x^\tau \) and \( x^{\tau+1} \) (obtained with a pretrained RAFT (Teed and Deng 2020)), we define a probability density function \( p(i, j) \propto ||w^\tau_{ij}||^2 \) over the image domain \( \Omega \), with \( (i, j) \in \Omega \). Then, we randomly sample a sparse set \( S \subset \Omega \) of \( n_c = |S| \) pixel locations from \( p \). This distribution makes it more likely that pixels of moving objects will be selected. However, in contrast to (Blattmann et al. 2021a) we do not introduce additional restrictions to the sampling or explicitly define the background. This is an essential difference, because in multi-object scenes, objects that belong to the background in one video might be moving in another. Thus, in our case, one cannot use the magnitude of the optical flow to separate objects from the background in each video.

To condition the video generation on the selected optical flow vectors, we introduce an encoding procedure. First, we construct a binary mask \( m \in \{0,1\}^{3 \times H \times W} \) such that \( m_{ij} = 1 \), \( \forall (i, j) \in S \) and \( m_{ij} = 0 \) otherwise. This mask is concatenated in the channel dimension with \( m \odot w^\tau \) to form a tensor \( \hat{w}^\tau \) of shape \( (3, H, W) \), which we refer to as the *sparse optical flow*. \( \hat{w}^\tau \) is further tiled into a \( 16 \times 16 \) grid. Each of these tiles is independently projected through an MLP and augmented with a learnable positional encoding to output a code (see Figure 2). This particular design of the optical flow encoder is a trade-off between having a restricted receptive field (because each tile is processed independently) and efficiency. A small receptive field is needed to ensure that separate controls minimally interact before being fed to the cross-attention layers. We found that this is crucial to enable the independent manipulation of objects.

![Sparse optical flow (OF) encoder. The dense OF is sampled and tiled in a 16 x 16 grid. Each OF tile is fed independently to the MLP, and then combined with a learnable positional encoding into a code.](image-url)
Figure 3: Violin plots of the local error distributions. Top: the distribution of the errors when using a convolutional encoder of the motion controls (i.e., with a large receptive field). Bottom: the distribution with our encoder (i.e., with a limited receptive field). Our encoder not only reduces the overall average error but also tends to have more errors of smaller magnitude when compared to the convolutional encoder. In contrast, the convolutional encoder shows a distinct long tail in errors of larger magnitude.

Experiments

In this section, we evaluate YODA to see how controllable the video generation is, i.e., how much the generated object motion correlates with the input motion control (see next section), and to assess the image and sequence quality on three datasets with different scene and texture complexities, as well as different object dynamics. We report several metrics, such as FVD (Unterthiner et al. 2018) and average LPIPS (Zhang et al. 2018), PSNR, SSIM (Wang et al. 2004) and FID (Parmar, Zhang, and Zhu 2022). Implementation and training details are in the supplementary material.

Evaluation of Intended vs Generated Motion

The objective of our training is to build a model to generate videos that can be controlled by specifying motion through \( \pi \) as a set of shifts at some user-chosen pixels. To evaluate how much the trained model follows the intended control, we introduce the following metrics: Local and global errors. To compute them, we sample an image from the test videos. Then, we randomly select one object in the scene and apply a random motion control input to a pixel of that object. Because the generated images are of high-quality, we can use a pre-trained optical flow estimation model (Teed and Deng 2020) to calculate the optical flow between the first image and the generated one. In principle, one could measure the discrepancy between the input motion shift and the generated one at the same pixel. However, since single pixel measurements are too noisy to use, we assume that all pixels within a small neighborhood around the selected pixel move in the same way, and then average the estimated optical flow within that neighborhood to compare it to the input control vector (depending on what we focus on, we use the relative \( L_2 \) norm or a cosine similarity). We call this metric the local error (see Figure 4 on the left in blue). Notice that this metric is quite coarse. For example, in some cases the model could use the motion input to rotate an object, instead of translating it. Also, the chosen neighborhood (whose size is fixed) may not fully cover just the object of interest. Nonetheless, this metric still provides a useful approximation of the average response of the trained model to the control inputs. We also assume that the motion generated far away from where the motion control is applied should be zero on average (although in general a local motion could cause another motion far away through a chain reaction). To assess this, we calculate the average \( L_2 \) norm of the estimated optical flow outside some neighborhood of the controlled pixel. We call this metric the global error (see Figure 4 on the right in orange).

Datasets

We evaluate YODA on the following three datasets:

**CLEVRER** (Yi et al. 2020) is a dataset containing 10K training and 1000 test videos capturing a synthetic scene with multiple simple objects interacting with each other through collisions. We cropped and downsampled the videos to 128 \( \times \) 128 resolution. On CLEVRER we show the ability of our model to model complex cascading interactions and also to learn the control of long-term motions (i.e., motions that once started at one frame can last for several frames).

**BAIR** (Ebert et al. 2017) is a real dataset containing around 44K 256 \( \times \) 256 resolution videos of a robot arm pushing toys on a flat square table. The visual complexity of BAIR is much higher than that of CLEVRER. The objects on the table are diverse and include non-rigid objects, such as stuffed toys, that have different physical properties. However, in contrast to CLEVRER, its interactions are simpler and do not require modeling long-term dynamics.

**iPER** (Liu et al. 2019) captures 30 humans with diverse styles performing various movements. The official train/test split separates the dataset into 164 training and 42 test clips. Although our main focus is to work with multi-object datasets, we use this dataset for two reasons: 1) We can test how YODA learns to control articulated objects, such as humans; 2) We can compare to iPOKE (Blattmann et al. 2021a), which has already been tested on this dataset (and is not designed for multi-object datasets).

Ablations

**Sparse optical flow encoder.** First, we show the importance of using a sparse optical flow encoder with a restricted receptive field. We observe that such an encoder is essential for the model to learn to independently control different objects in the scene. We manually annotated 128 images from the test set of the BAIR dataset. For each image we store a list of pixel coordinates that belong to objects in the scene, 1-2 pixels per object, 3 pixels for the robot arm. We use the local error to compare our encoder with a convolutional sparse optical flow encoder from (Blattmann et al. 2021a). In Figure 3, we show that our restricted receptive field optical flow encoder outperforms the convolutional one. For qualitative comparisons, see supplementary materials.

**Randomized conditioning.** In Table 1, we demonstrate the importance of our randomized conditioning scheme, where we randomly switch off the conditioning with respect to both the past frame and the control input with probability \( \pi \). The comparisons on the CLEVRER dataset show that random-

11726
The effect of the number of control vectors when training on the BAIR dataset. The local error (in blue and on the left-hand side) is the error of the optical flow in the neighborhood of the controlled pixel, while the global error (in orange and on the right-hand side) is the average optical flow vector outside of a circle around the interacted pixel. A smaller local error indicates a better response of the model to the control, while a low global error ensures that only the object of interest moves. The two boxes at the bottom of the figure show the regions across which the error is averaged (left - local error, right - global error). Best viewed in color.

<table>
<thead>
<tr>
<th>$n_c$</th>
<th>$\pi$</th>
<th>LPIPS↓</th>
<th>PSNR↑</th>
<th>SSIM↑</th>
<th>FID↓</th>
<th>FVD↓</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>0.0</td>
<td>0.288</td>
<td>24.88</td>
<td>0.86</td>
<td>9.9</td>
<td>401</td>
</tr>
<tr>
<td>5</td>
<td>0.25</td>
<td>0.123</td>
<td>30.11</td>
<td>0.93</td>
<td>6.0</td>
<td>86</td>
</tr>
<tr>
<td>5</td>
<td>0.5</td>
<td>0.126</td>
<td>30.07</td>
<td>0.93</td>
<td>5.7</td>
<td>70</td>
</tr>
<tr>
<td>5</td>
<td>0.75</td>
<td>0.133</td>
<td>29.41</td>
<td>0.92</td>
<td>6.1</td>
<td>83</td>
</tr>
<tr>
<td>1</td>
<td>0.5</td>
<td>0.144</td>
<td>28.78</td>
<td>0.92</td>
<td>6.3</td>
<td>83</td>
</tr>
</tbody>
</table>

Table 1: Evaluation of the generated videos on CLEVRER under different input sparsity ($n_c$) and randomization ($\pi$).

Figure 5: Ablation of the number of control vectors $n_c$ during training. The videos in each column start from the same initial frame and are generated with the same sequence of control vectors. Notice, however, that for $n_c = 100$ one has to use more controls at inference to bridge the gap between training and test settings. With too many control vectors during training the model demonstrates decent control over background objects, but struggles modelling interactions. With too few control vectors the interactions are modelled well, while the model lacks control over background objects. With the optimal $n_c = 5$ we get the best of the two worlds. Use Acrobat Reader to play the videos.
In this section we provide some visual examples of the generated sequence. On the BAIR dataset, we highlight the capability of YODA to move, rotate and deform separate objects without the robot arm physically touching them. Figure 1 shows different object manipulations on the BAIR test set. Notice how the model has learned also the 3D representations and interactions of the objects. Figure 7 shows the diversity of generated videos that share the same initial frame, but use different control signals. Notice the high correlation between the intended and generated motions and the ability of the model to correctly predict the interactions between the colliding objects. More qualitative results, including sequences on the iPER and CLEVERER datasets, as well as controllability evaluation and possible applications of YODA, are in the supplementary materials.

**Qualitative Results**

In this section we provide some visual examples of the generated sequence. On the BAIR dataset, we highlight the capability of YODA to move, rotate and deform separate objects. The Thirty-Eighth AAAI Conference on Artificial Intelligence (AAAI-24)
Acknowledgements
This work was supported by grant 188690 of the Swiss National Science Foundation.

References


