LEARNING OBJECT-CENTRIC DYNAMIC MODES FROM VIDEO AND EMERGING PROPERTIES

Anonymous authors
Paper under double-blind review

ABSTRACT

One of the long-term objectives of Artificial Intelligence is to endow machines with the capacity of structuring and interpreting the world as we do. Towards this goal, recent methods have successfully decomposed and disentangled video sequences into their composing objects, attributes and dynamics, in a self-supervised fashion. However, there have been scarce efforts to propose useful decompositions of the dynamics in a scene. We propose a method to decompose a video into moving objects, their attributes and the dynamic modes of their trajectories. We model the objects' dynamics with linear system identification tools, by means of a Koopman mapping and the Koopman operator $\mathcal{K}$. This allows user access and interpretation of the dynamics in the scene. We test our framework in a variety of datasets, while illustrating the novel features that emerge from our dynamic modes decomposition: temporal super-resolution, backwards forecasting, model reduction and video dynamics interpretation and manipulation at test-time. We successfully forecast challenging object trajectories from pixels, achieving competitive performance while drawing useful insights.

1 INTRODUCTION

Unsupervised learning of symbolic representations from high dimensional data poses a great challenge to current machine intelligence. As humans, our cognitive model of the world is based on segregation of reality into abstract categories, or symbols. We construct novel behaviours by dynamic reuse of familiar symbols Greff et al. (2020). In visual scenes, we can think of objects and their attributes as one valid set of symbolic representations.

There have been numerous efforts to model visual scenes from an object-centric perspective without supervision Eslami et al. (2016); Greff et al. (2019); Locatello et al. (2020); Burgess et al. (2019); Lin et al. (2020b). These works use inductive biases encoded in their architecture to decompose a static scene into its objects and their attributes.

As a natural extension, recent research has tackled unsupervised video decomposition Kosiorek et al. (2018); Denton & Birodkar (2017); Comas et al. (2020); Hsieh et al. (2018); Kossen et al. (2020); He et al. (2019); Jiang et al. (2020); Kipf et al. (2020); Jaques et al. (2021); Watter et al. (2015); Karl et al. (2016). A key challenge here is tracking the decomposed objects across time while learning to define what these objects are. Many of these works also attempt to model the intrinsic dynamics of the objects in the scene, and thus how the dynamic scene will look like in the future. While attempting to decompose the video into symbolic representations, these works rely on black-box neural methods to model dynamics. The usual choices are Recurrent Neural Network (RNN) or Graph Neural Netowrk (GNN) architectures. Interpreting dynamics from RNN models is not feasible. While GNNs are principled and object-centric, their formulation still relies on opaque architectures for the dynamical model. Thus, while their frame predictions are often accurate, the used dynamical models lack a user-interpretable formulation: even though a user can potentially manipulate objects, and their attributes in these models, there is no apparent controllable way to manipulate their dynamics or to understand their underlying propagation mechanism.

In this work, we advocate for decomposition not only from a visual perspective, but also from a dynamics perspective. We argue that decomposition leads to user interpretability. We want a user to be able to manipulate object dynamic behaviour in a controlled manner, directly on the pixel domain.

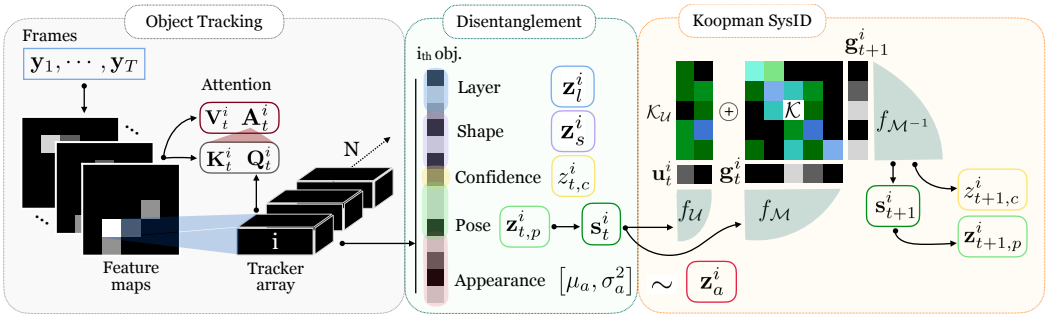

Figure 1: Architecture of OKID. Left: an attention-based recurrent tracker decomposes the scene into its composing objects. Center: the object representations are disentangled into Confidence, Layer, Shape, Appearance and Pose. Right: the dynamic features of Pose are modeled and forecasted by using a Koopman embedding. Latent representations are later used to reconstruct or predict frames.

With this objective, we propose to combine neural network-based architectures with Koopman operator theory, which is based on the insight that a finite-dimensional nonlinear system can be transformed into an infinite-dimensional linear dynamical system, with a linear operator $\mathcal{K}$. In this framework, the dynamical system can be understood as a composition of first and second order impulse responses, i.e. "Dynamic Modes" (DM), which can be extracted from a linear (Koopman) operator through spectral decomposition.

We introduce Object-centric Koopman-based Interpretable Decomposition (OKID), which is posed as a step forward towards dynamics interpretability through decomposition. Our method is self-supervised and (**1**) uses an attention-based tracking method to learn video representations, factorized into moving objects and their attributes; (**2**) decomposes the discovered trajectories into linear dynamic modes by finding a mapping to the Koopman space; (**3**) learns a global Koopman operator that characterizes the underlying dynamics of the training data; and hence (**4**) performs video decomposition, prediction and interpolation.

Our main contributions are:

(i) **We propose OKID, that jointly discovers object representations and performs attribute and dynamics decomposition in a self-supervised setting**. OKID uses a novel approach to learn a modular dynamical system from data by means of Koopman theory.
While Koopman theory has been employed for the dynamics and interaction prediction of multiple object trajectories, we propose a novel end-to-end method that can do so from pixels.

(ii) **Proposing simple video manipulation techniques that unveil novel properties for a video prediction model**. We illustrate this in our experiments. Temporal super-resolution, backwards forecasting, model reduction and video dynamics interpretation and manipulation are easily performed at test-time. While we do not beat the state of the art in the task of video prediction we remain competitive, generating visual results of good quality and plausible object trajectories.

2 Background

Consider a time-invariant dynamical system of a single object on $\Re^n$ of the form $\mathbf{s}_{t+1} = \Phi(\mathbf{s}_t)$. $\mathbf{s}_t \in \Re^n$ is the state of the system at time t. $\Phi : \Re^n \to \Re^n$ is a potentially non-linear function that defines the temporal transition of the states.

The fundamental insight of Koopman operator theory is that finite-dimensional nonlinear dynamics can be transformed to an infinite-dimensional linear dynamical system by appropriately chosing a Hilbert space of observables $\mathbf{g}$ Koopman (1931); Mezic (2005) and seeking an operator $\mathcal{K}$ that propagates $\mathbf{g}$ one step ahead:

$$\mathbf{g}_t = f_\psi(\mathbf{s}_t), \quad \mathbf{g}_{t+1} = \mathcal{K}\mathbf{g}_t, \tag{1}$$

$$f_\psi \circ \Phi(\mathbf{s}_t) = \mathcal{K} f_\psi(\mathbf{s}_t), \tag{2}$$

where f_ψ is the mapping from the state space to the observable space $\mathbf{g}_t$ and $\circ$ denotes the composition operator. While Koopman operator theory applies for an infinite dimensional Hilbert observable space, here we make a simplified assumption that it holds for some finite dimensional subspace, $\Re^m$ of observables, *i.e.,* $\mathbf{g}_t \in \Re^m$, $f_\psi : \Re^n \to \Re^m$, $\mathcal{K} : \Re^m \to \Re^m$. The eigenfunctions $\psi_j(\mathbf{s}_t) : \Re^n \to \Re^m$ of the Koopman operator satisfy $\mathcal{K}\psi_j(\mathbf{s}_t) = \lambda_j \psi_j(\mathbf{s}_t)$, where $\lambda_j \in \mathbb{C}$ is the corresponding eigenvalue. Eigenfunctions are hard to find, and some algorithms have been proposed to tackle the challenge. The most widely used are the dynamic mode decomposition (DMD) Schmid (2008) and its extension to nonlinear observables, the extended DMD (EDMD) Williams et al. (2015).

Previous research used hand-crafted functions to model the observable space. Currently, some approaches use deep neural networks to represent the observable space Lusch et al. (2018); Morton et al. (2019); Li et al. (2020); Xiao et al. (2021); Azencot et al. (2020). Neural networks have the advantage of being universal approximators, and are effective in finding the Koopman invariant subspace. Koopman methodology is data-driven, model-free and can discover the underlying dynamics and control of a given system from data alone Proctor et al. (2018). The operator $\mathcal{K}$ is usually found by linear regression given historical data or by end-to-end gradient-descent-based optimization.

In some cases, Koopman is employed in presence of control inputs. There are different approaches to introducing inputs to Koopman (*e.g.* Proctor et al. (2018); Li et al. (2020)). In our case, inputs model forces external to an object, originated from object-environment interactions. Therefore, inputs will depend both on the state of the object and the environment's geometry. Thus, the counterpart of equation 1 is:

$$\mathbf{g}_t = f_\psi(\mathbf{s}_t), \quad \mathbf{u}_t = f_{\mathcal{U}}(\mathbf{s}_t), \tag{3}$$

$$\mathbf{g}_{t+1} = \mathcal{K}\mathbf{g}_t + \mathcal{K}_{\mathcal{U}}\mathbf{u}_t \tag{4}$$

Here, $\mathbf{u}_t$ depends on the current state $\mathbf{s}_t$ (closed-loop control). It is expected to be sparse, and low-dimensional. $\mathcal{K}_{\mathcal{U}} : \Re^v \to \Re^m$ is the input Koopman operator. We usually define dimension v such that $v << m$.

3 Related Work

Video decomposition. DRNETDenton & Birodkar (2017) is an early work that decomposes video into a static component (content) and a dynamic component (pose). This is a key idea adopted by several subsequent works, such as DDPAEHsieh et al. (2018) and SQAIRKosiorek et al. (2018) Those methods are not designed to be scalable or explicitly model interactions. Hence, SCALOR Jiang et al. (2020) emerges to target scalability by massive parallelization, which can handle hundreds of objects in a scene. STOVE Kossen et al. (2020) adds on by modeling interactions with a graph neural network. G-SWM Lin et al. (2020a) unifies the abilities of previous models in a principled framework. It is scalable and handles interactions. ODDN Tang et al. (2022), distills object dynamics and models their interaction in an unsupervised way using clustering type approach. Finally, TBA He et al. (2019) presents a simple method that tracks by decomposition, disentanglement and generation of objects.

Koopman Operator. Koopman Operator theory has been successfully used to disentangle the dynamic modes in complex dynamical systems using dynamic mode decomposition techniques like DMD Schmid (2008), and Extended DMD (EDMD)Williams et al. (2015). By leveraging the fact that Koopman operator based methods require a rich family of functions to generate a mapping, many modern works have used deep neural networks to approximate the eigenvectors associated to the Koopman operator Lusch et al. (2018); Azencot et al. (2020); Morton et al. (2019); Li et al. (2020); Xiao et al. (2021); Otto & Rowley (2019). This idea has recently found applications in fluid dynamics Morton et al. (2018); Azencot et al. (2020), atomic and molecular scale dynamics Xie et al. (2019); Mardt et al. (2017), chaotic systems Brunton et al. (2017) and traffic dynamics Xie et al. (2019). However, to our knowledge, video sequences have not still been targeted. Proctor et al. (2018) generalized Koopman theory for control inputs, which gives rise to other methods like Li et al. (2020), that have used Koopman theory to model object dynamics and interactions from coordinates.

4 METHOD

Video often presents multiple objects in motion that generate a complex dynamical scene. In the pixel space, dynamics are highly complex. But object trajectories are often simpler. For our approach, we decompose the scene into its moving objects. We track and identify N objects across input frames, and assign a set of 5 variables to each one of them. Each object representation will be disentangled into the following categories:

- **Pose** $\mathbf{z}_{t,p} \in (-1, 1)^4$: Indicates the parameters for an 2D affine spatial transformation of an object; x and y centroid coordinates , scale and ratio, scaled to the range $(-1, 1)$ with a $\texttt{Tanh}(\cdot)$ activation.
- **Appearance** $\mathbf{z}_{t,a} \in \Re^A$: A vector containing information about an object's appearance.
- **Shape** $\mathbf{z}_{t,s} \in (0, 1)^S$: Binary object shape mask.
- **Confidence** $z_{t,c} \in (0, 1)$: Probability indicating the certainty of an object being correctly modeled.
- **Layer** $\mathbf{z}_{t,l} \in (0, 1)^L$: Indicator of the relative scene depth of an object with respect to the others.

As shown in Fig.1, the architecture consists of: 1) an object tracking block, 2) a Koopman SysID block and 3) a rendering block (not illustrated). We first track objects in a video by means of attention. The tracking block estimates the above attributes from pixels. A state $\mathbf{s}$ based on the pose is then embedded into the observable space with a Koopman mapping. The Koopman operator $\mathcal{K}$ is then used to propagate the trajectory in time. We decode the Koopman predictions back into the attribute space and render a new scene with the estimated set of object attributes. We reconstruct and predict video scenes, and use the ground-truth video to train the entire model end-to-end. We assume that there is no background. Next, we describe in more detail the main modules that form our model.

4.1 ARCHITECTURE

Tracking. We leverage the technique developed by He et al. (2019) to track by decomposition, with some modifications that allows forecasting, stochastic appearance modeling and object interactions. We encode each video frame $(\mathbf{y}_1, \cdots, \mathbf{y}_T)$ with a convolutional encoder, and obtain a feature map as $\mathbf{H}_{t,y} = f_{\texttt{enc}}([\mathbf{y}_t, \text{PE}])$, where PE is a positional encoding. We then track objects across features by using an array of trackers. N trackers are initialized, where N is an *upper-bound* on the expected number of objects in every scene. We define the tracker recurrent updates as:

$$\begin{aligned} &\mathbf{h}^i_{t,tr} = f_{\texttt{tr}}\left(\mathbf{h}^i_{t-1,tr}, \mathbf{H}_{t,y}\right), \quad \left[z^i_{t,c}, \mathbf{z}^i_{t,l}, \mathbf{z}^i_{t,p}, \mathbf{z}^i_{t,s}, \mathbf{d}^i_{t,a}\right] = \texttt{FC}\left(\mathbf{h}^i_{t,tr}\right) \\ &\mathbf{z}^i_{t,a} \sim \mathcal{N}(\mu_a, \sigma^2_a), \quad \left[\mu_a, \sigma^2_a\right] = \texttt{FC}(\mathbf{d}^i_{t,a}), \end{aligned} \tag{5}$$

where $f_{\texttt{tr}}(\cdot)$ is an attention-based tracking function described in Equation equation 6. We implement the appearance latent vector $\mathbf{z}^i_{t,a}$ to be the only stochastic variable in this setup, given its inherent complexity. It is sampled from a Gaussian distribution and trained as in a VAE framework. $z^i_{t,c}$ and $\mathbf{z}^i_{t,l}$ are softly binarized with $\texttt{Sigmoid}(\cdot)$ and $\texttt{Softmax}(\cdot)$ functions respectively.

Tracker updates. The tracking function, $f_{\texttt{tr}}(\cdot)$ is based on recurrent updates:

$$\mathbf{h}^i_{t,tr} = \texttt{GRU}_{\texttt{tr}}\left(\mathbf{h}^i_{t-1,tr}, \mathbf{u}^i_t\right), \mathbf{u}^i_t = \mathbf{A}^i_t \mathbf{V}_t, \quad \mathbf{A}^i_t = \texttt{Softmax}\left(\beta^i_t \mathbf{Q}^i_t \mathbf{K}^T_t\right) \tag{6}$$

where a $\texttt{GRU}$ cell is used to update the hidden state of each tracker $\mathbf{h}^i_{t,tr}$ and $\mathbf{u}^i_t$ is obtained by soft-attention mechanism , relying on the Query, Key, Value triad. Note that $\mathbf{V}^i_t = \mathbf{K}^i_t = \mathbf{H}_{t,y}$, the Query is obtained from $\mathbf{h}^i_{t-1,tr}$ using a fully connected layer. See appendix for more details.

We need a mechanism to provide information across trackers, while preserving the identity of each object through time. Trackers interact through external memory by using interface variables. We describe the memory module in detail in Section B.1 of the Supp. Material.

Koopman Embedding. We employ Koopman theory as an alternative to modelling dynamics. We argue that finding dynamic modes will introduce benefits to our model, as we discuss in Section 5. We define the state as a concatenation of T_S delayed instances (from now on referred to as delayed coordinates) of the pose vector $\mathbf{z}^i_p$ for the i_{th} object:

$$\mathbf{s}^i_t = \left[\mathbf{z}^i_{t,p}, \mathbf{z}^i_{t-1,p}, \ldots, \mathbf{z}^i_{t-T_S+1,p}\right] \tag{7}$$

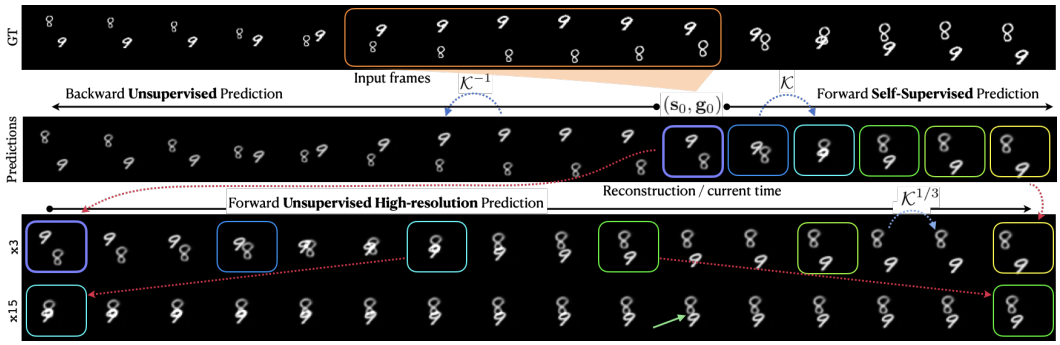

Figure 2: Visualization of OKID's unsupervised **temporal super-resolution** and **backward predictions**. Top row: Ground Truth sequence, in orange we highlight the frames from which we estimate the current dynamic state $\mathbf{s}_0$. Middle row: Forward prediction (right) at the input sample rate; Backward unsupervised prediction (left) by applying the inverse Koopman operator $\mathcal{K}^{-1}$. Note that although the input frames are encoded into $\mathbf{s}_0$, $\mathbf{s}_{[-1:-5]}$ are also predictions from $\mathbf{s}_0$. Bottom rows: Unsupervised interpolation of the future frames. The green arrow indicates the exact instant where the objects in the predicted scene do not superpose anymore. Only with high temporal resolution not present in the input data could we find that instant.

Therefore, $\mathbf{s}_t^i \in \mathbb{R}^{4T_s}$. T_s indicates our prior belief on the number of time-steps needed to model dynamics with an Auto-Regressive (AR) approach.

The observables $\mathbf{g}_t^i$ are obtained through the Koopman mapping $f_{\mathcal{M}} : \Re^{4\times T_s} \rightarrow \Re^{\mathcal{M}}$ (equivalent to Eq. equation 1). $\mathcal{M}$ and $\mathcal{U}$ make reference to the spaces of observables and inputs, respectively. We recover the original state by approximating the inverse function $f_{\mathcal{M}^{-1}} \approx f_{\mathcal{M}}^{-1}$ with a deterministic Auto-Encoder (AE) architecture. We realize those functions differently depending on whether we expect interactions in our scene. In the absence of interactions, we use a simple MLP to map states to observables in the Koopman space, and vice-versa. Otherwise, we use a graph neural network as described in the next paragraph. If external forces are present, we will also model inputs $\mathbf{u}_t$ as a non-linear mapping $f_{\mathcal{U}} : \Re^{4T_s} \rightarrow (-1, 1)^{\mathcal{U}}$ from the state space. Note that we set $f_{\mathcal{U}} : \Re^{4T_s} \rightarrow \{0\}^{\mathcal{U}}$ if we assume that the objects' dynamics are not affected by the environment:

$$\hat{\mathbf{s}}_{t+1}^i = f_{\mathcal{M}^{-1}} \circ \left(\mathcal{K} \circ f_{\mathcal{M}}\left(\mathbf{s}_t^i\right) + \mathcal{K}_{\mathcal{U}} \circ f_{\mathcal{U}}\left(\mathbf{s}_t^i\right)\right) = f_{\mathcal{M}^{-1}}\left(\mathcal{K}\mathbf{g}_t^i + \mathcal{K}_{\mathcal{U}}\mathbf{u}_t^i\right). \tag{8}$$

We refer to the estimated states as $\hat{\mathbf{s}}_T^i$. We define the Koopman operators $\mathcal{K} : \Re^{\mathcal{M}} \rightarrow \Re^{\mathcal{M}}$ and $\mathcal{K}_{\mathcal{U}} : \Re^{\mathcal{U}} \rightarrow \Re^{\mathcal{M}}$ as parameter matrices. We provide a discussion of the role of $\mathbf{u}_t$ in the Appendix. The pose vector $\hat{\mathbf{z}}_{t+1,p}^i$ is recovered by keeping the first stacked coordinates of the estimated state $\hat{\mathbf{s}}_{t+1}^i$, and limited to the range $(-1, 1)$. Once trained, the eigendecomposition of the Koopman operator $\mathcal{K}$ provides us with insights of the scene dynamics.

Interactions in the Koopman space. Object dynamics in the scene might encompass interactions. We introduce the use of a fully connected graph neural network to model interactions across objects. The mapping $f_{\mathcal{M}}$ is parametrized by a network reminiscent of Interaction Networks Battaglia et al. (2016). We represent the N vertices of the graph as the estimated object states with a binary identifier of the tracker, and learn edge features. Similarly to Li et al. (2020), we apply this graph network both as the Koopman mapping $\mathbf{g}_t^i = f_{\mathcal{M}}(\mathbf{s}_t^i, \mathbf{s}_t^{N\setminus i})$ and inverse mapping $\mathbf{s}_t^i = f_{\mathcal{M}^{-1}}(\mathbf{g}_t^i, \mathbf{g}_t^{N\setminus i})$. We learn the interaction types by assuming that object identities and roles in each scene will remain the same across training and testing data. In Section B.2 in the Supp. Material, we describe the graph and message passing in more detail.

4.2 Training

Given the video sequence $(\mathbf{y}_1, \cdots, \mathbf{y}_T)$, its generative distribution is given by:

$$p(\mathbf{y}_{1:T}|\mathbf{z}_{1:T}) = \prod_{i=1}^{N} p_{rec}(\mathbf{y}_{1:T}^i|\mathbf{z}_{1:T}^i) \prod_{i=1}^{N} p_{pred}(\mathbf{y}_{1:T}^i|\mathbf{z}_{1:K}^i), \tag{9}$$

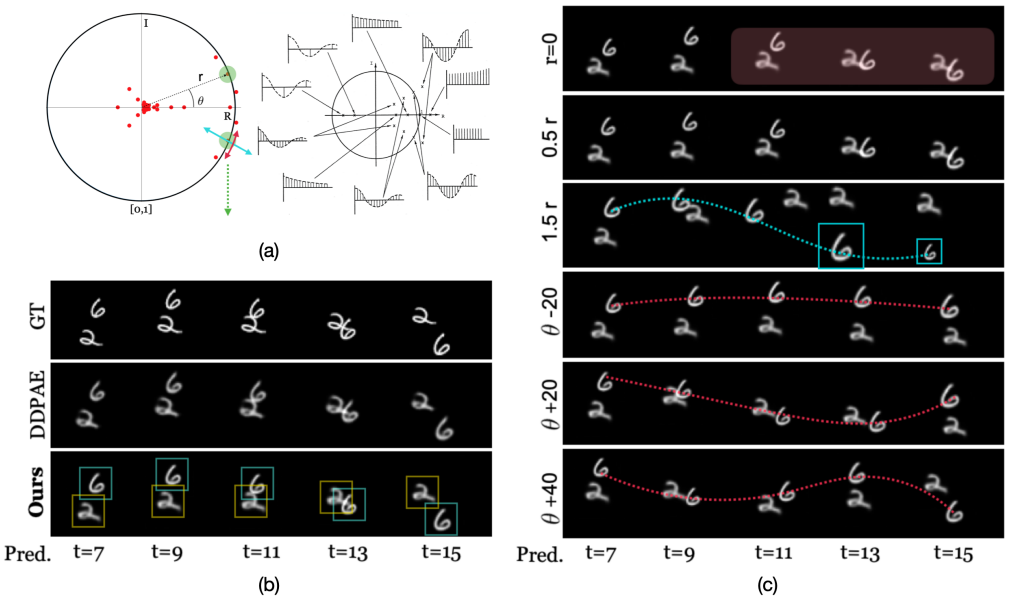

Figure 3: **(a)** right: Illustration of the eigenvalue behaviors of matrix $\mathcal{K}$ Phillips & Nagle (2007) visualized on the complex-plane, left: Learned Koopman matrix eigenvalues for 3D to 2D projection experiment. In green we highlight the eigenvalue pair that we will modify in (c) to visualize the effect of manipulations. **(b)**: Predictions: Ground Truth; our main baseline DDPAE; OKID with the object decomposition bounding boxes; **(c)** by rows: 2 variations to the radius of the highlighted eigenvalue, 3 variations to the angle. Blue line shows the effect of changing the radius and red line shows that of changing the angle.

where $\mathbf{z}_t^i = \left[z_{t,c}^i, \mathbf{z}_{t,l}^i, \mathbf{z}_{t,p}^i, \mathbf{z}_{t,s}^i, \mathbf{z}_{t,a}^i\right]$. Note that we both reconstruct the whole sequence with length T and predict it from K initial frames ($K < T$). For our experiments, K will coincide with the number of delayed coordinates T_s. Given the inferred latent variables, we reconstruct and predict $\mathbf{y}_i^t$ for each object sequentially. We first generate the object in the center with resolution $R = C \times h \times w$, given the appearance $\mathbf{z}_{t,a}^i$. The decoder $f_{\texttt{dec}} : \Re^A \rightarrow \Re^R$ is a CNN. We then apply a spatial transformer $\mathcal{T}$ to rescale and place the object according to the pose $\mathbf{z}_{t,p}^i$. For each object, the generative model is: $p(\mathbf{y}_t^i|\mathbf{z}_{t,a}^i) = \mathcal{T}(f_{\texttt{dec}}(\mathbf{z}_{t,a}^i); \mathbf{z}_{t,p}^i) \circ z_{t,c}^i$.

As in He et al. (2019), the final scene is a layer-by-layer composition of the decoded objects, according to their estimated shape $\mathbf{z}_{t,s}^i$ and layers $\mathbf{z}_{t,l}^i$. Similarly to the VAE framework, we train the model by maximizing the evidence lower bound (ELBO).

We use self-supervision for reconstructing the input $\mathbf{y}_{1:T}$ and predicting that same input from few initial conditions $(1:K)$. While the ELBO is enough for the performing prediction, we find that adding regularizers helps achieving some of the desired features. Our objective $\mathcal{J}_\omega$ with respect to the trainable weights ω is:

$$\mathcal{J}_\omega = \min_\omega \left[-\text{ELBO} + \lambda L_{\texttt{Reg}}\right] \tag{10}$$

$L_{\texttt{Reg}}$ is an autoencoding loss for the state space $\mathbf{s}$ and observable space $\mathbf{g}$, which enforces the desired behaviour in the Koopman embedding space. In case of expected sparse inputs, we enforce sparsity with the ℓ_1 loss of the estimated inputs $\hat{\mathbf{u}}$. Details about the objective and the frame generation description can be found in the Supp. material (Section B.4).

5 Experiments

We tested OKID on two main datasets with simple visual scenes but complex objects dynamics. We evaluated the fundamental properties of our method while modeling moderately complex appearances by using Moving MNIST. We evaluated OKID's performance in the presence of interactions, with realistic dynamics, by performing experiments with data from a real Double Pendulum.

Our baselines are established state-of-the-art methods for decomposed self-supervised video generation: DDPAE Hsieh et al. (2018) (the closest to OKID in terms of architecture), DRNET Denton & Birodkar (2017) and SCALOR Jiang et al. (2020). For scenarios with interactions, we compare to models that specifically model them: DDPAE, G-SWM Lin et al. (2020a), built on top of SCALOR, and STOVE Kossen et al. (2020). All baselines model dynamics with LSTM-like modules.

Evaluation Metrics. Our quantitative results will be measured in terms of per frame Mean Square Error (MSE), Mean Absolute Error (MAE), Structural Similarity (SSIM) and the Perceptual Similarity Metric (LPIPS) Zhang et al. (2018).

5.1 MOVING MNIST EXPERIMENTS

Moving MNIST Srivastava et al. (2015) is a synthetic dataset consisting of two digits with size 28×28 moving independently in a 64×64 frame. Each training sequence is generated on-the-fly by sampling MNIST digits and synthesizing trajectories according to a definition of the motion. Our model is 10k samples for training, 1k for validation and 2k for testing. We simulate 4 scenarios:

- ***Circular motion***: We generate a fairly simple dataset, for which we know the expected results. We sample randomly initial coordinates (x_0, y_0), radius R and the angular step length. We generate the motion with equation: $x = R\cos(t) + x_0$, $y = R\sin(t) + y_0$, where t increases linearly with a slope given by the angular step length. Finally, we constrain the motion to the dimensions of the frame. From $T = 3$ input frames, we predict 17.
- ***Cropped circular motion***: We mask the 29 top rows of the circular motion case, simulating a partially cropped frame. Note that an object of size 28×28 can be completely occluded.
- ***Inelastic/Super-elastic collisions***: We sample initial coordinates (x_0, y_0) and angle θ and let the object collide against the frame limits with fixed velocity. We increase the complexity of the case by simulating an inelastic response from the left and top limits $(\times 0.8)$ and a super-elastic response from the right and bottom limits $(\times 1.25)$. We generate chunks of $T = 13$ frames as input. From the first 3 frames, we predict 10.
- ***3D to 2D motion projection***: Finally, we generate trajectories that lay withing the cube $[-1, 1]^3$. Those are generated randomly following the steps described in the Supp. Material (Section C). We project the trajectory to the $xy-$ axis, and simulate depth with the relative sizes of the objects. From 6 input frames, we predict 10.

Quantitative Results. A quantitative general overview of the experiments can be seen in Table 1. Looking at the perceptual metrics, OKID often outperforms all baselines in terms of LPIPS, and is on par with DDPAE's SSIM. DDPAE is the closest to ours in terms of decomposition and frame generation. The key difference is the dynamics modeling. DDPAE uses a concatenation of LSTMs for reconstructing and predicting the pose. On the absolute MSE and MAE metrics, OKID does not outperform DDPAE. However, its performance remains competitive while providing a novel set of useful features:

Model Reduction. We propose to test our architecture after removing all but n eigenvalues of $\mathcal{K}$. n was chosen to ensure we do not discard eigenvalues with module higher than 0.5. As we see in Table 1, the results are close to the prediction with the full matrix $\mathcal{K}$. This indicates that our model was able to capture the motion of the digits with 3 to 4 conjugate pairs of eigenvalues for the studied cases.

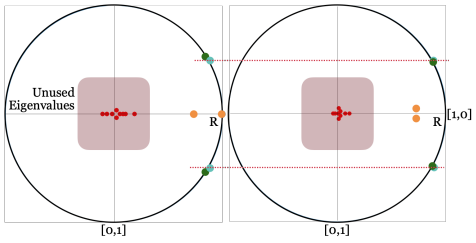

Figure 4: **Eigenvalues of the learned matrix $\mathcal{K}$ in a complex plane** for both *circular* experiments. Left: Complete image, right: Cropped image. The learned eigenvalues are very similar, while one of them has been trained with corrupted data. Yellow points in real axis are associated to non-oscillating dynamic modes. In red, eigenvalues close to 0 have very weak effect on the dynamics.

Interpreting $\mathcal{K}$: Circular Motion Experiments. Figure 4 depicts the eigendecomposition of the learned $\mathcal{K}$ for the *Circular Motion Experiments*. We know that the motion is expected to be sinusoidal in both x and y. Therefore, a correct Koopman mapping would not need to be non-linear to capture the dynamics in this case. A sinusoid can be modelled by a linear operator with a complex pair of eigenvectors in the unit circle. We can see that pair, together with real eigenvalues that present no

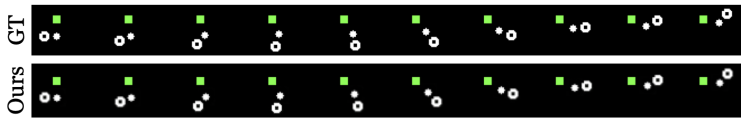

Figure 5: Visualization of the Double Pendulum dataset, together with OKID's 10 forward predictions. Frames depict the pendulum state at $(1/5)_{\text{th}}$ of the original sampling rate.

oscillation. We observe also that the learned model is stable as no eigenvalue has greater radius than 1. The model has learned a very similar operator $\mathcal{K}$ for both the cropped and the complete version. This is an indicator that OKID is learning consistent dynamics across datasets when they share the same motion. It also suggests that the model is able to impute a trajectory when the data is missing. Qualitative results can be seen in the Supp. Material (Section D).

Interpreting $\mathcal{K}$: 3D to 2D Projection Experiments. Figure 3 shows the qualitative performance in terms of prediction of OKID and the result of several interventions. It illustrates how single or conjugated pairs of eigenvalues impact on the dynamics, for the *3D to 2D Projection Experiments*. This dataset is challenging because it entangles linear motion across dimensions by means of projection. It also encompasses digit size variations. We can see that the predictions are accurate and sharp and the learned dynamics are correct. The model correctly disentangles the two objects that appear in the scene, and models their dynamics.

We manipulate the eigenvalues of $\mathcal{K}$ highlighted in green (Fig. 3(a)) by changing their module r or their angle θ. Shadowed in red, we see that this particular eigenvalue pair has higher effect in the latter part of the trajectory. If we increase its module above 1, we observe an increase on the intensity of the variations, leading to strong oscillation at the end of the sequence (see size of the object). This happens because the system is now unstable. If we vary the angle of the eigenvalue pair with respect to the real axis, we see variations in terms of frequency. When we subtract 20 degrees to the angle, it becomes almost 0 resulting into an almost flat trajectory. When we increase that angle, we see digits oscillating with higher frequency.

Inputs: Collision Experiments. For the *Collision Experiments* scenario, the challenge is the use of inputs $\mathbf{u}_t$. Every collision against the frame limits applies a force to the object, that modifies its dynamics. Therefore, we model the effect of the environment as in Equation 8, allowing the inputs to be non-zero, and forcing them to be sparse and low-dimensional ($\mathbf{u}_t \in \Re^4$) to avoid overfitting. This generates sharp objects and captures correctly the dynamics.

Time reversal and Super-Resolution. An additional property of OKID, is its ability to easily modify the discrete time resolution or direction. For simplicity, we will look at the cases where no input $\mathbf{u}_t$ is present. As described in Eq. 8, we advance one step in time by using once the Koopman operator $\mathcal{K}$. Thus, OKID is learning to model discrete time at the sampling frequency of the training data. To find the state midway between state t and $t+1$, we can simply operate on an observable $\mathbf{g}_t^i$ with $\mathcal{K}^{1/2}$. In general, to interpolate m states in between, we can operate with $\mathcal{K}^{1/m}$. Interestingly, we can also reverse the dynamics by taking a negative power of the operator, i.e. $\mathcal{K}^{-1}$. Note that for all cases, $\mathcal{K}$ is a square matrix. To handle noise amplification during inversion, we suppress the effect of inverting eigenvalues close to zero or negative. In our experiments, we found that small eigenvalues have almost no effect on the dynamics. Figure 2 shows how we can increase the temporal resolution by a factor of 3 or 15 or reverse time prediction by manipulating $\mathcal{K}$. Note that the model has only been trained to predict forward at a specific time resolution. SeeSupp. Material for technical details.

5.2 Double Pendulum Experiments

Next, we propose to learn object-centric dynamic modes from video of a double pendulum in motion. Here we wish to demonstrate the ability of OKID to model object interactions. A double pendulum exhibits a rich dynamic behavior with a strong sensitivity to initial conditions and noises in the environment, despite being a simple physical system. We employ the Double Pendulum Chaotic (DPC) dataset Asseman et al. (2018) to generate video samples.

In this dataset, 21 individual runs of a real double pendulum were recorded. Each of the recorded sequences lasts around 40s and consists of around 17,500 frames. Three pairs of marker positions (x, y) were extracted, representing the coordinates of the anchor, extreme of first arm and extreme of second arm. We constructed a dataset by locating three objects with different appearances at the coordinate locations provided by the DPC dataset (see Fig. 5).

Table 1: Quantitative comparison of all methods for the four scenarios. We evaluate reconstruction, prediction, and prediction with model reduction (ne denotes that we keep only the top n eigenvalues). Top to bottom shows results of different methods on several datasets. Our method performs similarly to the best RNN-based baseline and outperforms the rest of baselines in prediction.

Model	MSE ↓			MAE ↓			SSIM ↑			LPIPS ↓		
Circular	Rec	Pred	Pred(6e)	Rec	Pred	Pred(6e)	Rec	Pred	Pred(6e)	Rec	Pred	Pred(6e)
DDPAE	**40.13**	**71.96**	/	**123.94**	**162.64**	/	**0.87**	**0.82**	/	0.19	0.21	/
OKID	59.97	84.23	84.64	139.92	168.83	169.96	0.86	**0.82**	0.82	**0.15**	**0.17**	0.17
Cropped circular	Rec	Pred	Pred(6e)	Rec	Pred	Pred(6e)	Rec	Pred	Pred(6e)	Rec	Pred	Pred(6e)
DDPAE	**35.04**	**54.95**	/	**98.48**	**118.90**	/	**0.88**	**0.85**	/	0.21	0.23	/
OKID	47.80	59.26	59.32	108.28	120.04	120.15	**0.88**	0.86	0.86	**0.17**	**0.19**	0.19
Collision	Rec	Pred	Pred(6e)	Rec	Pred	Pred(6e)	Rec	Pred	Pred(6e)	Rec	Pred	Pred(6e)
DRNET	109.01	214.49	/	218.14	339.58	/	0.75	0.60	/	0.30	0.40	/
SCALOR	**13.24**	329.63	/	**50.02**	494.41	/	**0.95**	0.28	/	0.02	0.48	/
DDPAE	49.53	**93.52**	/	146.65	**199.57**	/	0.84	0.76	/	0.23	0.25	/
OKID	59.53	103.63	110	155.07	205.31	212.38	0.83	**0.77**	0.76	**0.19**	**0.21**	0.22
3D to 2D proj	Rec	Pred	Pred(8e)	Rec	Pred	Pred(8e)	Rec	Pred	Pred(8e)	Rec	Pred	Pred(8e)
DRNET	80.31	136.06	/	172.43	248.11	/	0.77	0.66	/	0.32	0.41	/
SCALOR	**7.58**	233.18	/	**32.05**	377.34	/	**0.96**	0.32	/	**0.02**	0.45	/
DDPAE	20.99	**43.72**	/	57.08	**86.87**	/	0.94	**0.89**	/	0.11	**0.14**	/
OKID	36.32	45.63	49.48	78.27	93.57	101.10	0.89	0.87	0.86	0.17	0.17	0.20

For this experiment, the task is to predict 10 frames from an input of 5, at a sample rate $5\times$ lower than the original. Our setup encompasses a total of approximately 280k frames for training, 70k for validation and 9k for test, which are provided separately by the original dataset. We train OKID for 400 epochs and choose the best according to validation.

Results. The correct modeling of the three moving objects dynamics is not possible by treating them as independent, exchangeable entities. Hence, each scene object must be assigned to a particular tracker. Not only across time frames, but across video instances.

Table 2: Results of 10 frame prediction in the Double Pendulum dataset. **SR** denotes Super-Resolution: we predict the sequence at the original sampling rate. It predicts 54 frames instead of 10.

Model	MSE ↓	MAE ↓	SSIM ↑	LPIPS ↓
DDPAE	171.72	339.02	0.85	0.119
STOVE	438.98	532.78	0.83	0.150
G-SWM	**112.45**	**137.74**	**0.95**	**0.019**
OKID	120.82	162.44	**0.95**	0.026
OKID (SR)	133.06	174.69	**0.95**	0.029

Table 2 shows the quantitative evaluation of our method. OKID outperforms the baselines except for G-SWM. In that case performance is fairly similar [1]. With OKID (SR) we predict 54 frames at the original sampling rate. Here we interpolate the missing prediction steps. Performance drops, but remains fairly close. Figure 5 illustrates our model's qualitative performance. We show 10 predictions. Objects look sharp and dynamics are correctly modeled. More visual results can be found in Section D of the Supp. Material.

6 Conclusion

We propose a self-supervised method for video prediction by means of scene decomposition into objects, their attributes and scene dynamic modes. We embed the dynamic object-centric attributes into a space where dynamics behave linearly, by means of a Koopman embedding. This enables the use of linear algebra techniques to interpret and manipulate the scene dynamics. Through careful experiments we show that decomposition into dynamic modes is indeed possible and carries predictive power. We include examples with inputs and object interactions. We also provide insights into the dynamics of objects (*interpretability*) through the analysis of eigenvalues of the learned Koopman operator. Finally, we show how to manipulate the model to obtain arbitrarily higher temporal resolution, backward prediction, model reduction and alter the dynamic behavior of a targeted object.

Our model has room for improvement. OKID's current implementation doesn't support background modeling and an upper-bound of objects (number of trackers) N must be defined *a priori*. This is detrimental to the performance, as the number of trackers used has a considerable effect on the iteration time and memory requirements. We intend to address these limitations in the future work.

[1] Following indications from their paper, STOVE pre-processes and evaluates data in grayscale.

REFERENCES

Alexis Asseman, Tomasz Kornuta, and Ahmet Ozcan. Learning beyond simulated physics. In *Modeling and Decision-making in the Spatiotemporal Domain Workshop*, 2018. URL `https://openreview.net/forum?id=HylajWsRF7`.

Omri Azencot, N. Benjamin Erichson, Vanessa Lin, and Michael Mahoney. Forecasting sequential data using consistent koopman autoencoders. In *International Conference on Machine Learning (ICML)*, 2020.

Peter Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, and koray kavukcuoglu. Interaction networks for learning about objects, relations and physics. In *Advances in Neural Information Processing Systems*, 2016.

S. Brunton, B. W. Brunton, J. Proctor, E. Kaiser, and J. N. Kutz. Chaos as an intermittently forced linear system. *Nature Communications*, 8, 2017.

C. Burgess, Loïc Matthey, Nicholas Watters, Rishabh Kabra, I. Higgins, M. Botvinick, and Alexander Lerchner. Monet: Unsupervised scene decomposition and representation. *a Computer Research Repository (CoRR)*, 2019.

Armand Comas, Chi Zhang, Zlatan Feric, Octavia Camps, and Rose Yu. Learning disentangled representations of videos with missing data. *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Emily L Denton and vighnesh Birodkar. Unsupervised learning of disentangled representations from video. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

S. M. Ali Eslami, Nicolas Heess, Theophane Weber, Yuval Tassa, David Szepesvari, koray kavukcuoglu, and Geoffrey E Hinton. Attend, infer, repeat: Fast scene understanding with generative models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2016.

A. Graves, Greg Wayne, M. Reynolds, T. Harley, Ivo Danihelka, Agnieszka Grabska-Barwinska, Sergio Gomez Colmenarejo, Edward Grefenstette, Tiago Ramalho, J. Agapiou, Adrià Puigdomènech Badia, K. Hermann, Yori Zwols, Georg Ostrovski, A. Cain, H. King, C. Summerfield, P. Blunsom, K. Kavukcuoglu, and Demis Hassabis. Hybrid computing using a neural network with dynamic external memory. *Nature*, 538:471–476, 2016.

Klaus Greff, Raphaël Lopez Kaufman, Rishabh Kabra, Nick Watters, Christopher Burgess, Daniel Zoran, Loic Matthey, Matthew Botvinick, and Alexander Lerchner. Multi-object representation learning with iterative variational inference. In *International Conference on Machine Learning (ICML)*, 2019.

Klaus Greff, Sjoerd van Steenkiste, and Jürgen Schmidhuber. On the binding problem in artificial neural networks. *ArXiv*, abs/2012.05208, 2020.

Z. He, J. Li, Daxue Liu, Hangen He, and D. Barber. Tracking by animation: Unsupervised learning of multi-object attentive trackers. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1318–1327, 2019.

Jun-Ting Hsieh, Bingbin Liu, De-An Huang, Li F Fei-Fei, and Juan Carlos Niebles. Learning to decompose and disentangle representations for video prediction. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

Miguel Jaques, Michael Burke, and Timothy M Hospedales. Newtonianvae: Proportional control and goal identification from pixels via physical latent spaces. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4454–4463, 2021.

Jindong Jiang, Sepehr Janghorbani, Gerard De Melo, and Sungjin Ahn. Scalor: Generative world models with scalable object representations. In *International Conference on Learning Representations (ICLR)*, 2020.

Maximilian Karl, Maximilian Soelch, Justin Bayer, and Patrick Van der Smagt. Deep variational bayes filters: Unsupervised learning of state space models from raw data. *arXiv preprint arXiv:1605.06432*, 2016.

Thomas Kipf, Elise van der Pol, and Max Welling. Contrastive learning of structured world models. In *International Conference on Learning Representations (ICLR)*, 2020.

B. O. Koopman. Hamiltonian systems and transformation in hilbert space. *Proceedings of the National Academy of Sciences of the United States of America*, 17 5:315–8, 1931.

Adam Kosiorek, Hyunjik Kim, Yee Whye Teh, and Ingmar Posner. Sequential attend, infer, repeat: Generative modelling of moving objects. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

Jannik Kossen, Karl Stelzner, Marcel Hussing, Claas Voelcker, and Kristian Kersting. Structured object-aware physics prediction for video modeling and planning. In *International Conference on Learning Representations (ICLR)*, 2020. URL https://openreview.net/forum?id=B1e-kxSKDH.

Yunzhu Li, Hao He, Jiajun Wu, Dina Katabi, and Antonio Torralba. Learning compositional koopman operators for model-based control. In *International Conference on Learning Representations (ICLR)*, 2020.

Zhixuan Lin, Yi-Fu Wu, Skand Vishwanath Peri, Bofeng Fu, Jindong Jiang, and Sungjin Ahn. Improving generative imagination in object-centric world models. In *ICML*, 2020a.

Zhixuan Lin, Yi-Fu Wu, Skand Vishwanath Peri, Weihao Sun, Gautam Singh, Fei Deng, Jindong Jiang, and Sungjin Ahn. Space: Unsupervised object-oriented scene representation via spatial attention and decomposition. In *International Conference on Learning Representations (ICLR)*, 2020b.

Francesco Locatello, Dirk Weissenborn, Thomas Unterthiner, Aravindh Mahendran, Georg Heigold, Jakob Uszkoreit, Alexey Dosovitskiy, and Thomas Kipf. Object-centric learning with slot attention. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Bethany Lusch, J. N. Kutz, and S. Brunton. Deep learning for universal linear embeddings of nonlinear dynamics. *Nature Communications*, 9, 2018.

Andreas Mardt, L. Pasquali, Hao Wu, and F. Noé. Vampnets for deep learning of molecular kinetics. *Nature Communications*, 9, 2017.

I. Mezic. Spectral properties of dynamical systems, model reduction and decompositions. *Nonlinear Dynamics*, 41:309–325, 2005.

Jeremy Morton, Antony Jameson, Mykel J Kochenderfer, and Freddie Witherden. Deep dynamical modeling and control of unsteady fluid flows. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

Jeremy Morton, F. Witherden, and Mykel J. Kochenderfer. Deep variational koopman models: Inferring koopman observations for uncertainty-aware dynamics modeling and control. In *International Joint Conferences on Artificial Intelligence (IJCAI)*, 2019.

Samuel E. Otto and Clarence W. Rowley. Linearly recurrent autoencoder networks for learning dynamics. *SIAM Journal on Applied Dynamical Systems*, 18(1):558–593, 2019. doi: 10.1137/18M1177846.

Charles L Phillips and H Troy Nagle. *Digital control system analysis and design*. Prentice Hall Press, 2007.

J. Proctor, S. Brunton, and J. N. Kutz. Generalizing koopman theory to allow for inputs and control. *SIAM J. Appl. Dyn. Syst.*, 17:909–930, 2018.

P. Schmid. Dynamic mode decomposition of numerical and experimental data. *Journal of Fluid Mechanics*, 656:5–28, 2008.

Nitish Srivastava, Elman Mansimov, and R. Salakhutdinov. Unsupervised learning of video representations using lstms. In *ICML*, 2015.

Qu Tang, Xiangyu Zhu, Zhen Lei, and Zhaoxiang Zhang. OBJECT DYNAMICS DISTILLATION FOR SCENE DECOMPOSITION AND REPRESENTATION. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=oJGDYQFKL3i.

Manuel Watter, Jost Springenberg, Joschka Boedecker, and Martin Riedmiller. Embed to control: A locally linear latent dynamics model for control from raw images. *Advances in neural information processing systems*, 28, 2015.

M. Williams, I. Kevrekidis, and C. Rowley. A data–driven approximation of the koopman operator: Extending dynamic mode decomposition. *Journal of Nonlinear Science*, 25:1307–1346, 2015.

Yongqian Xiao, Xin Xu, and Qianli Lin. Cknet: A convolutional neural network based on koopman operator for modeling latent dynamics from pixels. *ArXiv*, 2021.

Tian Xie, A. France-Lanord, Yanming Wang, Y. Shao-Horn, and J. Grossman. Graph dynamical networks for unsupervised learning of atomic scale dynamics in materials. *Nature Communications*, 10, 2019.

Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

SUPPLEMENTAL MATERIAL

For a better understanding of this work, **please see the slideshow given as part of the supplemental material.**

In this appendix, we start by describing the implementation details in Section A. Following, we describe the architecture in Section B, together with a description of the objective (Section B.4) and technical details of the Koopman module (Section B.5). We give further details of the 3D to 2D motion projection experiment in Section C. We provide additional experiments in Section D. In Section E we discuss about the roles of inputs in our experiments by means of an example. In Section F briefly show the robustness of our model to different seeds. Finally, in Sections G and H we discuss limitations and broader impact of our work.

A MODEL IMPLEMENTATION DETAILS

A.1 BASELINE MODELS

SCALOR The SCALORJiang et al. (2020) model is an unsupervised model for learning scalable object oriented representations. The model can track a large amount of objects with a dynamic background. However, although SCALOR can predict future frames from previous frames, its model is not specifically designed for it. In any case, prediction is reported in their work. The baseline was chosen as an alternative for parallel object decomposition in an attention-based tracking. Our experiments show that while SCALOR is very good at reconstruction, it predicts poorly. Despite following the instructions in their paper and tried different configurations, we were not able to reach a reasonable performance in terms of prediction for the studied dynamical scenes.

For our experiments we used the implementation available in `https://github.com/JindongJiang/SCALOR.git`. In order to obtain the best performance with SCALOR on our data set, we trained the model for 625 epochs. However, we found that the loss reached a plateau for the prediction task in the early stages. At that point of training, SCALOR is unable to generate predictions. In order to tackle this phenomenon, we kept the MNIST digits stationary and trained SCALOR until it could generate predictions. From that point, we observed that after 625 epochs of training, we got some good prediction frames. Nevertheless, predictions are often blurry and inaccurate, and obvious artifacts appear in the scene.

DRNET The original version of the DRNETDenton & Birodkar (2017) model only uses the first four frames as input for training. For our experiments, we need three/six input frames. We changed the scene discriminator in DRNET to train on all frames in the sequences. The rest of the model was kept exactly the same as the authors' implementation for better reproduction of results. As mentioned in their github repository (`https://github.com/ap229997/DRNET.git`), the main network and the LSTM in DRNET were trained separately. Firstly, we trained the main base network, then we trained it again with skip connection, and finally we trained the LSTM part. Therefore, the performance of the the LSTM part is determined by the main network. The scene discriminator was trained with BCE loss. The main network and LSTM were trained with MSE loss; and we trained the main model and LSTM with 4 RTX 2080 Ti GPU with 12.8GB memory each. For more details about DRNET, please refer to their github.

DDPAE We used the code provided by the authors. The hyperparameters that they use in the public version were kept unchanged. Also, we followed the instructions in their github repository (`https://github.com/jthsieh/DDPAE-video-prediction`) for the Moving MNIST experiment. For Moving MNIST, the model was trained for 250 epochs, 50 more than `OKID`. For the Double Pendulum experiments, we trained for 400 epochs.

DDPAE has an option to enable interactions between detected objects. We activated that option for the Double Pendulum experiments.

STOVE For the pendulum experiments we compare our model with the baselines using the same input/output configuration. To train STOVE we reduced the number of visible frames per sequence from 8 to 5. During testing we increased the number of rollout frames to predict 10 frames. The

image size of our dataset is of 64 pixels compared to 32 of mostly used datasets to train STOVE. We trained the model for 400 epochs as specified as default in the official implementation (`https://github.com/jlko/STOVE`). We used the video prediction version of the model.

In their experiments, they use grayscale data with objects of identical appearance. Although it is not explicitly mentioned, STOVE could be unable to reconstruct different shapes for their experimental setup. That would explain their failure to reconstruct the scenes in Double Pendulum dataset. We would also like to highlight that the dataset contains different object shapes because each one of them has a different relation to the other two. Hence, the models need to identify them by leveraging their appearance, as no additional information is provided as input.

G-SWM In the code available in `https://github.com/jlko/STOVE` it uses 10 conditional steps to generate new frames, so we reduced it to 5 steps. To compare fairly the baseline with our model we increased the number of supervised frames in training. STOVE uses curriculum learning increasing the difficulty of the task by predicting more frames. We modified the number of supervised frames each 10,000 steps as in its original version. We increased the number of predictions in training as follows: From [2, 4, 6, 8, 9] to [2, 6, 9, 12, 15].

A.2 Our Model

OKID The main latent variables have the following dimensions: $\mathbf{z}^i_{t,a} \in \mathbb{R}^{50}$ for Moving MNIST experiments and $\mathbf{z}^i_{t,a} \in \mathbb{R}^{30}$ for the Double Pendulum. $\mathbf{z}^i_{t,c} \in [0, 1)$ and $\mathbf{z}^i_{t,p} \in \mathbb{R}^4$. The latter's four dimensions correspond to $[x, y, \Delta s, \Delta r]$, where x and y are the coordinates of the centroid of an object; Δs is the increment of the size of the object with respect to the decoded 32×32 appearance, which would be of size $s = 1$; and Δr is the increment of the ratio s_x/s_y, which by default is 1. The increments are weighted by a scalar $w \in (0, 1]$ that regulates their effect. All components of the pose are bounded between -1 and 1. The number of delayed coordinates per state $\mathbf{s}^i_t$, T_s, corresponds to the number of input frames in our experiments.

Following the VAE framework, we implemented $\mathbf{z}^i_{t,a}$ to be a sample of a learned posterior distribution $q\left(\mathbf{z}^i_{t,a}|\mathbf{y}_t\right) = \mathbb{N}(\mu_a, \sigma_a)$, with the reparameterization trick. As usual, we regularized our training by adding a KL divergence term between our posterior and a Gaussian prior with $\mu = 0$ and $\sigma = 1$.

We implemented the convolutional features with dimensionality $\mathbf{H}^i_{t,y} \in \mathbb{R}^{(96\times4\times4)}$ and the tracker hidden states as $\mathbf{h}^i_{t,tr} \in \mathbb{R}^{288}$. The dimensions were chosen after a manual sweep of hyperparameters range. Particularly, the dimensionality of $\mathbf{h}^i_{t,tr}$ was chosen from the range $[96, 192, 288, 384]$; $\mathbf{z}^i_{t,a}$ from $[40, 50, 70]$; and $\mathbf{H}^i_{t,y}$ from $[96, 128]$. The simplest Koopman mapping and inverse are parameterized by a multi-layer perceptron with 4 layers and hidden state dimensionality of 40. For cases with interactions, our graph neural network (based on Interaction Networks) has intermediate layers of size 30 for both the edge and node effects. The Koopman operator $\mathcal{K}$ is initialized as a matrix of 0s, and the input operator $\mathcal{K}_\mathcal{U}$ with Xavier initialization (same as other layers of the Koopman embedding). $\mathbf{H}^i_{t,y}$ was obtained by leveraging the input frames $\mathbf{y}_t$ and a positional encoding PE. The latter has dimensionality 4 and values that indicate the distance from the 4 frame edges in terms, normalized in $[0, 1]$. With the exception of these details, the implementation of the attention and memory follows the guidelines of He et al. (2019).

We trained the model in all Moving MNIST scenarios for 200 epochs and 10k iterations per epoch. We used a batch size of 40, and a prior number of objects $N = 2$. For the Double Pendulum, we trained for 400 epochs and picked the best model. We set the batch size to 100 and the number of objects to $N = 3$. Our model has the potential to set redundant components to be empty by reducing the confidence value to 0. We show qualitative examples of this capability in D.2. The learning rate was set to $1e{-}3$ (DP) and $5e{-}4$ (MM) and reduced by a factor of 0.7 on plateau of the validation loss. We used Adam as our optimizer with parameters $\beta = [0.9, 0.999]$ and weight decay regularization $1e{-}4$.

Next, we describe variations for each experiment. For the *circular motion* experiment, we have an observable space of dimension $\mathbf{g}^i_t \in \Re^{15}$ and inputs are set to 0. The same is done for the *cropped circular motion* experiment, but in this case the loss is evaluated only in the visible part of the frame. Note that this is also done for the baselines. The loss weights are increased linearly, according to the values that will be provided in the codebase.

We also use $\mathbf{g}_t^i \in \Re^{15}$ for the *Inelastic/Superelastic Collision* experiment. In this case, the input dimensionality is $\mathbf{u}_t^i \in \Re^4$. We keep it low dimensional so it does not absorb the free dynamics of the object.

For the *3D to 2D motion projection* experiment, we expect higher order dynamics in the Koopman manifold, given the apparent complexity of the dataset. Therefore, we set an observable space of $\mathbf{g}_t^i \in \Re^{30}$.

For all Moving MNIST experiments, objects are decoded to a size of $1 \times 32 \times 32$ and located in the 64×64 frame.

Finally, for the Double Pendulum experiments we set the observable space to $\mathbf{g}_t^i \in \Re^{30}$, while $\mathcal{K}$ will have dimensionality $N \times 30$. We let $\mathcal{K}$ learn freely the dynamics for each of the distinct object, without repeating blocks. Objects are decoded to a size of $1 \times 16 \times 16$ and located in the 64×64 frame following Equation 14 We introduce the concept of curriculum learning for these experiments. We start predicting 7 frames, which are increased to 9 and 10 in epochs 20 and 40 respectively.

Further details can be found in the codebase that will be provided together with the final version of the work.

Software We implemented this method using Ubuntu 18.04, Python 3.6, Pytorch 1.2.0 and Cuda 10.0.

Hardware For each of our experiments we used a single GPU RTX 2080 Ti (Blower Edition) with 12.8GB of memory.

B Model Architecture Details

Given the space restrictions, some of the architecture details are discussed here.

B.1 Memory Module

We need a mechanism to provide information across trackers, while preserving the identity of each object through time. Based on He et al. (2019); Graves et al. (2016), trackers interact through external memory by using interface variables. We use the frame convolutional features $\mathbf{H}_{t,y}$ as external memory, and implement read and write operations. When iterating through trackers, the input $\mathbf{H}_{t,y}$ will be updated as follows:

$$\mathbf{e}_t^i = \texttt{Sigmoid}\left(\hat{\mathbf{e}}_t^i\right), \quad \left[\hat{\mathbf{e}}_t^i, \mathbf{w}_t^i\right] = \texttt{FC}\left(\mathbf{h}_{t,tr}^i\right) \tag{11}$$

$$\mathbf{H}_{t,y}^{i+1} = \left(\mathbf{1} - \mathbf{A}_t^i \otimes \mathbf{e}_t^i\right) \otimes \mathbf{H}_{t,y}^i + \mathbf{A}_t^i \otimes \mathbf{w}_t^i. \tag{12}$$

Here, $\mathbf{e}_t^i$ and $\mathbf{w}_t^i$ are the erasing and writing vector, respectively. The feature map is updated in the spatial locations indicated by the attention matrix $\mathbf{A}_t^i$ of the previous iteration. The operator $\otimes$ denotes a element-wise product in the channel dimension of $\mathbf{H}_{t,y}^i$.

B.2 Koopman Interaction Network

We represent the vertices of the graph as the object states with a binary identifier of the tracker $\mathbf{v}_t^i = f_{\text{V}}\left(\left[\mathbf{s}_t^i, \texttt{ID}^i\right]\right)$. Relations are represented as the identifiers and the difference of the associated vertices $\mathbf{r}_t^{(k,l)} = f_{\text{R}}\left(\left[\texttt{ID}^k, \texttt{ID}^l, \mathbf{v}_t^k - \mathbf{v}_t^l\right]\right), \forall (k,l) \in [1, \ldots, N]$. The embedding space is the result of a message passing procedure, where we compute an edge effect $\mathbf{e}_t^{(k,l)} = f_{\text{EE}}\left(\left[\mathbf{v}_t^l, \mathbf{v}_t^k, \mathbf{r}_t^{(k,l)}\right]\right), \forall (k,l) \in [1, \ldots, N]$ and a node effect $\mathbf{g}_t^i = f_{\text{NE}}\left(\left[\mathbf{v}_t^i, \sum_{n=1}^N \mathbf{e}_t^{(i,n)}\right]\right)$. All functions f are implemented as MLPs. Similarly to Li et al. (2020), we apply this graph network both as the Koopman mapping $\mathbf{g}_t^i = f_{\mathcal{M}}(\mathbf{s}_t^i, \mathbf{s}_t^{N \setminus i})$ and inverse mapping $\mathbf{s}_t^i = f_{\mathcal{M}^{-1}}(\mathbf{g}_t^i, \mathbf{g}_t^{N \setminus i})$. We do not indicate the nature of interactions supervisedly. Instead, we assume that object identities in each scene will remain the same across training and testing data, as well as the nature of their interactions.

For this case, and ignoring $\mathbf{u}_t^i$, we re-write Equation 8 as:

$$\hat{\mathbf{s}}_{t+1}^i = f_{\mathcal{M}^{-1}}\left(\sum_{n=1}^{N}\mathcal{K}^{(n,i)}\mathbf{g}_t^n, \sum_{n=1}^{N}\mathcal{K}^{(n,N\setminus i)}\mathbf{g}_t^n\right). \tag{13}$$

The global matrix $\mathcal{K}$ is a composition of submatrices $\mathcal{K}^{(l,k)}$ associated to a sender object l and a receiver object k. Thus, the interactions are also linear in the Koopman space.

B.3 Scene Rendering

In Section 4.2 we describe a simplification of the rendering process, by which we are simply allocating the objects in the predicted location. However, to account for object occlusions and sharp edges we will leverage $\mathbf{z}_{t,l} = [z_{t,l,1}, \ldots, z_{t,l,K}]$ and $\mathbf{z}_{t,s}$. We sum the contributions of each object to the final frame as:

$$\begin{aligned}
&p(\mathbf{y}_{t,a}^i|\mathbf{z}_{t,a}^i, \mathbf{z}_{t,p}^i, z_{t,c}^i) = \mathcal{T}(f_{\texttt{dec,a}}(\mathbf{z}_{t,a}^i); \mathbf{z}_{t,p}^i) \circ z_{t,c}^i \\
&p(\mathbf{y}_{t,s}^i|\mathbf{z}_{t,s}^i, \mathbf{z}_{t,p}^i, z_{t,c}^i) = \min\left(1, \mathcal{T}(f_{\texttt{dec,s}}(\mathbf{z}_{t,s}^i); \mathbf{z}_{t,p}^i) \circ z_{t,c}^i\right) \\
&p(\mathbf{y}_{t,a,k}|\mathbf{y}_{t,a}^i, \mathbf{y}_{t,s}^i, \mathbf{z}_{t,l}^i) = \sum_i z_{t,l,k}^i \circ \mathbf{y}_{t,s}^i \circ \mathbf{y}_{t,a}^i \\
&p(\mathbf{y}_{t,s,k}|\mathbf{y}_{t,s}^i, \mathbf{z}_{t,l}^i) = \sum_i z_{t,l,k}^i \circ \mathbf{y}_{t,s}^i
\end{aligned} \tag{14}$$

Where $\circ$ indicates an element-wise multiplication and K is the number of layers, which coincides with the number of objects N. To generate the final frame, we follow an iterative method. For $k = [1, \ldots, K]$

$$p(\mathbf{y}^{t,k}|\mathbf{y}_{t,a,k}, \mathbf{y}_{t,s,k}) = (\mathbf{1} - \mathbf{y}_{t,s,k}) \circ \mathbf{y}^{t,(k-1)} + \mathbf{y}_{t,a,k}.$$

Where we initialize $\mathbf{y}^{t,(k-1)}$ to be a vector of zeroes $\mathbf{y}^{t,0} = \mathbf{0}$. The last iteration gives the final rendered frame $\mathbf{y}^{t,K}$. This process is required only by the Double Pendulum experiments, as objects might occlude each other. Moving MNIST is simpler in terms of rendering. Pixel values of the ground truth have a single channel with a binary $(0, 1)$ value. Therefore, we can render the whole frame by summing up all the decoded objects: $p(\mathbf{y}^t|\mathbf{y}_{t,a}^i) = \sum_i \mathbf{y}_{t,a}^i$ and clipping the maximum value to 1. The layer vector $\mathbf{y}_{t,l}^i$ is neglected as we assume that all objects are in the same plane.

B.4 Objective

In Section 4.2 we describe the objective as the sum of ELBO and regularization terms. The ELBO is described following:

$$\begin{aligned}
&\log p_\theta(\mathbf{y}_{1:T}) \geq \mathbb{E}_q\big[\log p_\theta\left(\mathbf{y}_{1:T}|\mathbf{z}_{1:T}^{1:N}\right) + \\
&\log p_\theta\left(\mathbf{y}_{K+1:T}|\hat{\mathbf{z}}_{K+1:T|1:K}^{1:N}\right) - \\
&\text{KL}\left(q_\phi\left(\mathbf{z}_{1:K,a}^{1:N}\right)||p\left(\mathbf{z}_{1:K,a}^{1:N}\right)\right)\big],
\end{aligned} \tag{15}$$

where the first term $\log p_\theta\left(\mathbf{y}_{K+1:T}|\hat{\mathbf{z}}_{K+1:T|1:K}^{1:N}\right)$ is the reconstruction error and $\text{KL}\left(q_\phi\left(\mathbf{z}_{1:K,a}^{1:N}\right)||p\left(\mathbf{z}_{1:K,a}^{1:N}\right)\right)$ stands for the Kullback-Leibler divergence between the approximate posterior q_ϕ and a Gaussian prior p.

In order to achieve the desired properties in the Koopman space, we enforce the observables $\mathbf{g}$ and states $\mathbf{s}$ to be consistent before and after the Koopman embedding. This is done through an autoencoding loss. We might also expect the presence of sparse inputs $\mathbf{u}$, as in the Collision case of Moving MNIST experiments. In that case, we enforce sparsity with its convex surrogate, the ℓ_1 norm. The full regularization term is given by:

$$L_{\text{reg}} = \lambda_{\texttt{fit}} L_{\texttt{fit}} + \lambda_{\texttt{AE}} L_{\texttt{AE}} + \lambda_{\mathcal{U}} L_{\mathcal{U}} \tag{16}$$

$$L_{\texttt{AE}} = \left\|\mathbf{s}_{2:T}^{1:N} - \hat{\mathbf{s}}_{2:T|1:T-1}^{1:N}\right\|_1, L_{\texttt{fit}} = \left\|\mathbf{g}_{2:T}^{1:N} - \hat{\mathbf{g}}_{2:T|1:T-1}^{1:N}\right\|_2^2, L_{\mathcal{U}} = \left\|\hat{\mathbf{u}}_{1:T}^{1:N}\right\|_1,$$

Here, $\|\cdot\|_1$ and $\|\cdot\|_2^2$ indicate the ℓ_1 and ℓ_2 losses respectively. $\|\cdot\|_1$ is used to enforce sparsity in $\hat{\mathbf{u}}_{1:T}^{1:N}$. Finally, the λs are the weights applied to each one of the loss terms.

B.5 CONDITIONING THE KOOPMAN OPERATOR

With the aim of interpreting the square matrix $\mathcal{K}$, we have proposed to do an eigendecomposition so that $\mathcal{K} = Q\Lambda Q^{-1}$. Assuming $\mathcal{K}$ has dimensionality $n \times n$, Λ is a diagonal matrix with the complex eigenvalues in the diagonal and Q is chosen to be the matrix with the n eigenvectors as columns. This decomposition exists if and only if $\mathcal{K}$ has n eigenvectors which are a basis for $\mathbb{C}^n$. As discussed in Section 5.1 (Time reversal and Super-Resolution paragraph), we propose to achieve an arbitrarily high temporal resolution by manipulating the Koopman operator $\mathcal{K}$. We thus would operate with $\mathcal{K}^{1/m}$ to increase resolution of our predictions by a factor of m. Thus we can express $\mathcal{K}^{1/m} = Q\Lambda^{1/m}Q^{-1}$, raising each complex element of the diagonal to the power $1/m$. For the simplest case with $m = 2$:

$$\begin{aligned}\left(Q\Lambda^{1/2}Q^{-1}\right)^2 &= Q\Lambda^{1/2}\left(Q^{-1}Q\right)\Lambda^{1/2}Q^{-1} = \\ &= Q\Lambda Q^{-1} = \mathcal{K}\end{aligned} \tag{17}$$

This can be generalized to negative values of m. In that case, we would be predicting backwards in time by applying the modified operator. In our experiments, we evaluate the qualitative performance for $m = -1$.

Some engineering decisions must be made to properly condition matrix $\mathcal{K}$. The operator is initialized to very small values, although we make sure that no eigenvalue is 0 so that the matrix can be safely inverted. We expect it to learn the needed eigenvalues (or poles) by optimization during training. Therefore, at the beginning of training, $\mathcal{K}$ will map any value of $\mathbf{g}$ to a vector of 0 (or very small values). Some of those eigenvalues will remain practically static during training, as they might not be needed to model the observed object dynamics. For instance, in Table 1 we can see how small eigenvalues have a very low effect on the prediction accuracy. This suggests that those can be safely ignored when manipulating $\mathcal{K}$, at the expense of a low accuracy drop.

Applying a negative power to the operator $\mathcal{K}$ would greatly amplify the eigenvalues close to 0, creating an unstable system, highly sensitive to noise. We avoid their effect by clipping the eigenvalues to a maximum module of 1.5 after applying the negative power to $\mathcal{K}$.

Finally, having a negative real eigenvalue can also originate problems when applying an arbitrary power to $\mathcal{K}^{(1/m)}$. However, in principle only frequency components at the Nyquist sampling rate in the training data could lead to having an eigenvalue in such location. In our experiments we observe that our sampling rates are much higher than the maximum frequency in the dynamics. Hence, we assume that a negative real eigenvalue will be an artifact of training with no observable effect on the object dynamics, therefore we suppress its effect.

C 3D TO 2D MOTION PROJECTION DESCRIPTION

We discuss in more detail some of this particular case of the Moving MNIST dataset. This motion is created parametrically following $z = cos(v_z\theta + \phi_{z_0})$, $R = \sqrt{1 - z^2}$, $x = R\cos(v_x\theta + \phi_{x_0})$, $y = R\sin(v_y\theta)$, with coordinates (x, y, z) and angular velocities (v_x, v_y, v_z). This parameterization constrains it to lay within the cube $[-1, 1]^3$. We then rotate the trajectory with an angle $\pi/8$ and project a random portion of the full trajectory with size $T = 16$ (6 in 10 out) to the $xy-$ axis. The offset phase ϕ_{z_0} and ϕ_{x_0} are also randomly generated. Using perspective projection, the objects are resized according to their depth in the z axis, after being projected. A graphic depiction of the process can be seen in Figure C.1.

D MORE EXAMPLES AND FAILURE CASES

In this section, we provide examples for the four studied scenarios, including failure cases for `OKID`. We will also show more examples of our ablation studies. For the latter, we modify the eigenvalues of the learned Koopman operator $\mathcal{K}$ to study and visualize the variations in the dynamics of the objects in the scene.

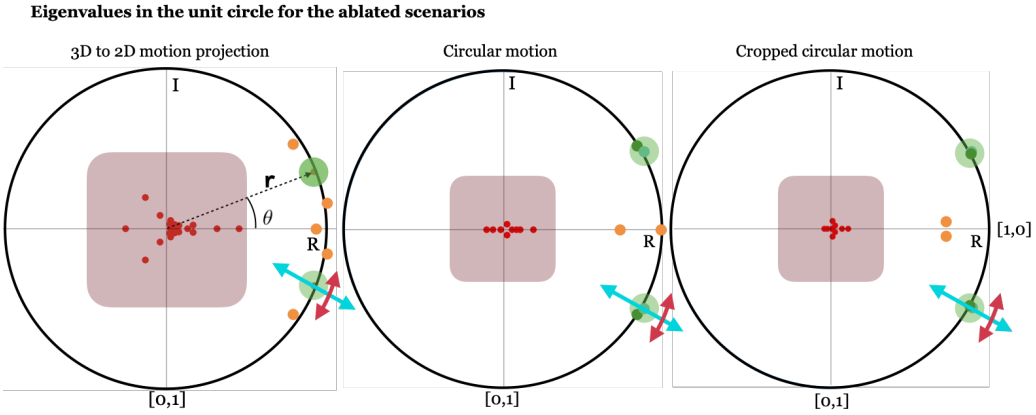

Figure B.1: Eigenvalues in the unit circle for the Koopman operator $\mathcal{K}$ for the three ablated scenarios. The axes are the imaginary (I) and real (R) components of the eigenvalues. Shadowed in red, the area where the eigenvalues have negligible effect on the trajectories, according to the results shown in Table 1.The following indicators are linked to the results shown in Figures H.1, H.2, and H.3: Highlighted in green, the chosen eigenvalue pair for the ablation study. The blue and red arrows indicate the kind of manipulation that the chosen eigenvalue pair will undergo. r and θ are the magnitude and phase of the uppermost eigenvalue with respect to the real axis.

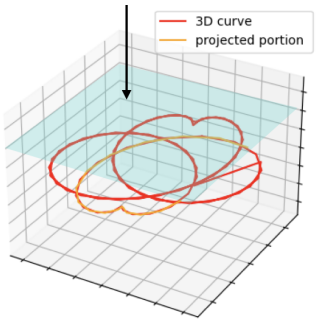

Figure C.1: Example of a projected trajectory for *3D to 2D motion projection* experiment. In orange the selected chunk.

D.1 Moving MNIST Experiments

Figure B.1 gives an overview of the learned eigenvalues for three of the studied scenarios: *Circular motion, Cropped circular motion* and *3D to 2D motion projection*.

Figure B.1 shows shadowed in red the eigenvalues that have been empirically proven to have low effect on the performance in Table 1. For all cases, 6 to 8 eigenvalues (usually 3 to 4 complex conjugate pairs) are enough to generate the behaviour seen in the dynamics of the Koopman manifold, with a low error margin. We chose an eigenvalue pair (highlighted in green) and changed its radius r and angle θ to study the effects on the scene dynamics.

In Figure H.1, we see the case of *3D to 2D motion projection*. Here, we display the two best-performing RNN-based baselines (DDPAE and DRNET) together along OKID. Qualitatively, we see that they perform similarly or worse than OKID in this case. In this particular case, DDPAE shows slightly worse reconstruction in terms of appearance. However, if we pay attention to the size and location of the digits, DDPAE is slightly more accurate than OKID. This is a trend in our evaluation setting. OKIDsacrifices part of its performance in order to gain new capabilities. Instead, DDPA uses a RNN as their dynamics forecasting module. This is effective for non-interacting scenes, but it can't be intervened or interpreted by an external observer in. In Fig. , we can see a similar behavior. DDPAE is slightly better when it comes to locating the objects. Appearance is very similar. In Fig.

we illustrate a failure case of DDPAE, which is unable to model the object when this disappears from the scene. However, the general behavior of DDPAE is still better in terms of quantitative evaluation, except in case of perceptual quality (LPIPS). We can see how the model identifies and predicts independently each one of the digits. The blue dotted line shows behaviors due to changes in magnitude of the eigenvalue pair. When the eigenvalues are outside of the unit circle, the system they model is unstable. This can be observed in the figure by looking at the behaviour for $\hat{r} = 1.5r$. The digits start showing an unstable behaviour by changing progressively it size and the amplitude of their oscillation. At a certain point, the digit gets stuck in the bottom frame limit, reaching the constraint of the pose vector. For variations in the phase angle θ, it is interesting to note that it has a direct link to the frequency of the digit's oscillation. When $\hat{\theta} = \theta - 20$, $\hat{\theta}$ is close to 0. This has a clear impact on the vertical component of the object's trajectory. When we increase θ, the vertical oscillation frequency increases with it. These behaviors are as expected given the illustration in Figure 3 (top-left).

We see a very similar behavior in Figures H.2 and H.3. For $\hat{r} > 1$, Figure H.2 shows an oscillating behavior of the digit sizes that increase with t. Again, the behavior seems unstable. Figure H.3 illustrates a familiar saturation behavior for $\hat{r} > 1$. Also, it shows how OKID is able to find the unseen dynamics of a digit in a partially visible frame. If we observe the third row of Figure H.3, we can see how the digit "2" has the expected oscillation, even when it was unseen neither in the input data or the self-supervision. This is an indicator of the ability of OKID to discover the true dynamics of a system. It also exhibits better reconstruction and prediction that the strongest baseline DDPAE.

Figure H.4 shows three particular cases of the *Inelastic/Superelastic collision*. The first example is a success, the second one is a partial success and the third one is a failure. In all cases we see a similar behaviour for the baselines. DDPAE is the closest to our model in capturing the dynamics for prediction, with the difference that it leverages RNNs. SCALOR has a very good reconstruction, but its prediction is not comparable to the other tested architectures. DRNET seems to decompose the scene and capture partially the dynamics and appearance. However, its performance is poor. For the top case, OKID correctly decomposes the scene and predicts the inelastic and superelastic collisions with the result of an accurate and sharp prediction. For the center case, the performance is similar. However, we can observe that the final digits are a bit off from their actual trajectory. This is likely due to a bad modelling of the collisions. In the bottom example all methods fail to decompose the scene into its composing objects, as they appear overlapped.

Figure D.3 illustrates a failure case for the *Cropped circular motion* scenario. In this case, the input frames are heavily occluded. One of the digits (6) is fully visible but the other is not. In this case, as expected, none of the methods can model the occluded digit. However, given the uncertainty, OKID has a sharper and more accurate prediction than DDPAE.

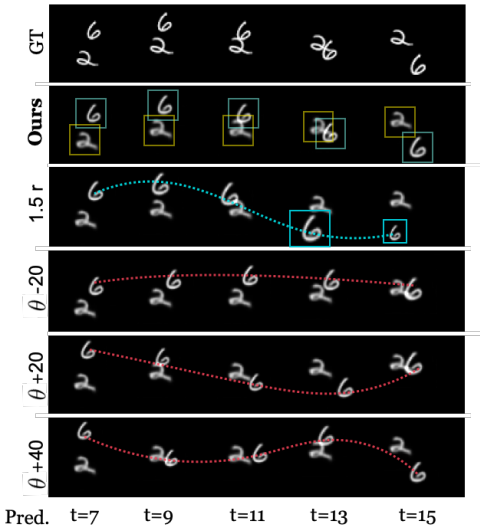

Figure D.1: We manipulate $\mathcal{K}$ for a single object ("6") while leaving "2" as it is. This corresponds to rows 6 and 9 of Figure 3.

Single Object Manipulation. Figure D.1 shows the ability of our model to manipulate a single object instead of the whole scene dynamics. In this case, we just need to identify the object-tracker correspondence, and manipulate $\mathcal{K}$ only for the selected object.

More Trackers than Objects in the Scene. Finally, we show in Figure D.2 the qualitative performance of OKID when there are more object slots (trackers) than actual objects in the scene. In the top, we trained our model with 3 trackers. In the bottom, with 4. In both cases, the number of objects in the scene is 2. We see that the quality of the reconstruction is not affected by having more trackers than objects in the scene.

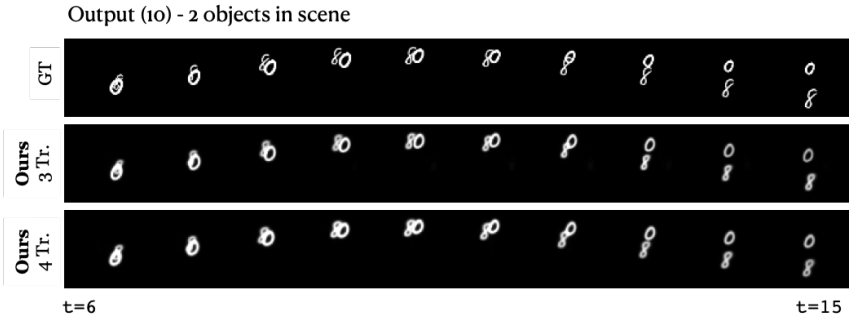

Figure D.2: Qualitative display of OKID's performance for the cases where the number of trackers N is higher than the number of objects in the scene. In the middle row, 3 trackers were employed for training. In the bottom row 4 trackers. The figure shows that the model is capable of ignore some of the initialized trackers if they are not needed, while correctly generating the predicted scene.

D.2 Double Pendulum Experiments

Figure H.5 shows two success cases and one failure for OKID, together with the baselines performance. We can see how OKID correctly identifies the objects, and generates them with high accuracy. In two of the cases, OKID and G-SWM predict almost perfectly the trajectories and generate accurate frames. We also show a case where G-SWM, which is the strongest baseline, outperforms OKID by predicting more accurately the last frames of the predicted sequence. If inspected carefully, G-SWM often fails to predict the exact trajectory (see middle and bottom cases). However, it is by a low margin. DDPAE and STOVE fail to generate sharp objects, or even to correctly distinguish them. They also fail to correctly predict the trajectories in all cases. Particularly, STOVE is unable to model different appearances.

Figure H.6 shows the qualitative performance of OKID when performing super-resolution. The figure depicts 50 predictions of OKID at 5 times faster sampling than the training data. The qualitative results show sharp objects and correct trajectory estimation.

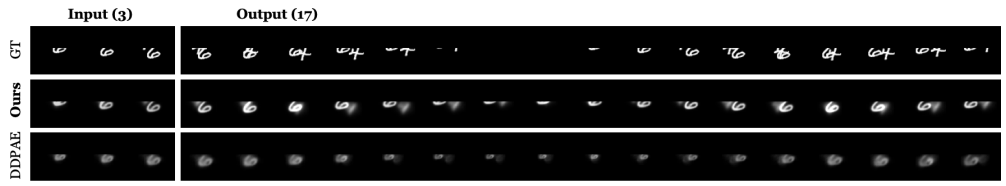

Figure D.3: Failure case for the *cropped circular motion* scenario. In this case, one of the numbers is never seen in the initial conditions, and therefore OKID can't reconstruct its appearance. Note that DDPAE is also unable to reproduce it, and the results are qualitatively worse.

E DISCUSSION: ESTIMATION AND ROLE OF INPUTS $\mathbf{u}_t$

External forces $\mathbf{u}_t$ model the interaction of objects with their environment. To illustrate this point, consider a particle moving with constant velocity (e.g. acceleration $= 0$) that collides elastically with a vertical wall at time T. The equations of motion in the horizontal direction $\mathbf{x}$ are given by:

$$\mathbf{z}_{k+1} = \begin{bmatrix} 2 & -1 \\ 0 & 1 \end{bmatrix} \mathbf{z}_k + \begin{bmatrix} -2 & 0 \\ 0 & 0 \end{bmatrix} \mathbf{u}_k \text{ where}$$
$$\mathbf{z}_k \doteq \begin{bmatrix} \mathbf{x}_k \\ \mathbf{x}_{k-1} \end{bmatrix}, \ \mathbf{u}_k = \mathbf{z}_k \delta_{k,T} \tag{18}$$

where $\mathbf{u}_t$ models the force exerted by the wall. In free space, the equations above simply reduce to the discrete time version of $\ddot{\mathbf{x}} = 0$. When a collision takes place, the wall exerts a force proportional to the velocity of the impact. This is the scenario modelled by Eq. 8. Since we expect these collisions to be infrequent, the input $\mathbf{u}_t$ should be as sparse as possible. Note that the collision time T is a function of the state of the particle and the geometry of the scene. Thus, both the mapping and the force $\mathbf{u}_t$ must be learned from states $\mathbf{s}_t$ (to account for instance for scenarios more complex than simple elastic collisions). Inputs are restricted with low dimensionality and the mentioned sparsity penalty.

F MULTIPLE RUNS

To explore the effect of randomness in initialization of neural network parameters, we ran experiments on spherical moving-MNIST dataset 4 times under different seeds and report the mean and standard deviation of multiple runs. The results are summarized in Table 3.

Table 3: Train the OKIDs with four different seed in circular motion dataset.

Model	MSE		MAE		SSIM		LPIPS	
name	Rec	Pred	Rec	Pred	Rec	Pred	Rec	Pred
OKID	34.09 ± 1.75	47.1 ± 1.08	78.27 ± 3.58	96.92 ± 2.92	0.84 ± 0.008	0.81 ± 0.006	0.177 ± 0.02	0.18 ± 0.018

G LIMITATIONS OF OUR MODEL

For the sake of transparency, we discuss the assumptions, restrictions and limitations of our model. OKID's current implementation doesn't support background modeling and an upper-bound of objects (number of trackers) N must be defined a priori. Also, the number of trackers have a considerable effect on the iteration time and memory requirements (limitations shared with Hsieh et al. (2018); Denton & Birodkar (2017); Kossen et al. (2020); He et al. (2019)). However, qualitative results reveal that a small margin between the number of trackers and the actual number of objects often has a very low effect on the accuracy. OKID has not been tested for sparse interactions such as collisions of objects. We consider this to be beyond the scope of this paper. There are challenges in terms of training. OKID is sensitive to some of the hyper-parameters. More specifically, the size of the convolutional map $\mathbf{H}_{t,y}$ showed to perform best with resolution 4×4 or 8×8. However, the depth of the frame encoder, decoder, the Koopman mappings or the size of the tracker hidden state do not show to be sensitive. Loss terms involving long-term prediction supervision (pixel-level and Koopman mapping supervision after several predictions) can impede convergence if activated at the initial stages of training. Therefore, they are softly introduced after around 3k iterations. For similar reasons, for the Double Pendulum experiments we make use of curricular learning, increasing the number of predicted frames along training.

H BROADER IMPACT

Our primary objective is video physical understanding and interpretability which can have broader positive impact in numerous applications from medical images to self driving. However, this also comes with potentially negative element of manipulating videos for personal gain or with ill intent. Therefore, we urge the users to take such negative impacts into account while using our work in real world applications.

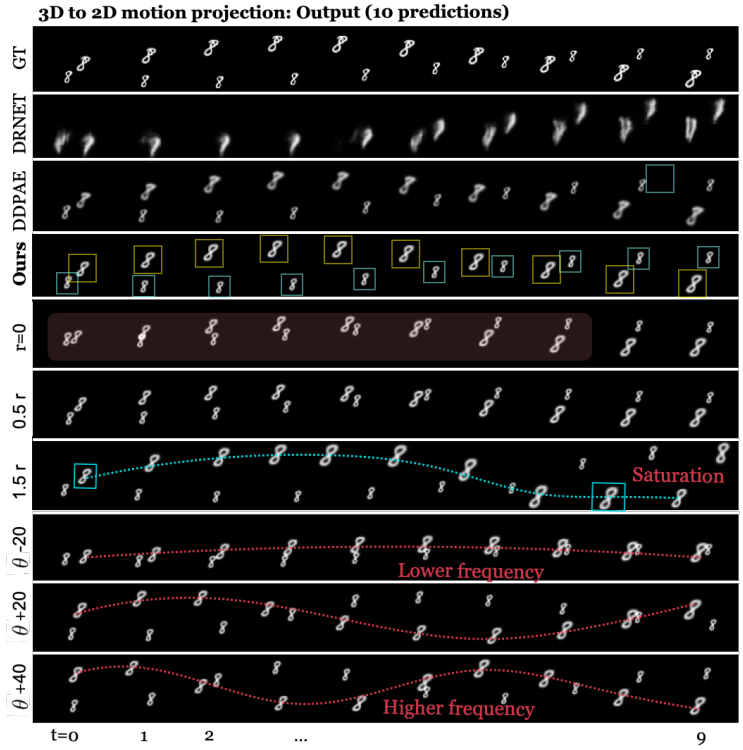

Figure H.1: Ablation study and qualitative results of the *3D to 2D motion projection* scenario. The labels in the Y axis indicate the variations suffered by a selected eigenvalue of the learned $\mathcal{K}$. Written in red, we indicate the behaviors we perceive. We show the $T = 10$ predictions of our model and the strongest baseline, DDPAE. This setup will be the same as in the rest of ablation studies shown (Figures H.2 and H.3). We can see how the model decomposes the scene into its composing objects and predicts accurately their trajectory. We vary one of the eigenvalues (Figure B.1), chosen so that the visualization is clear. We see how variations in angle θ (red dotted line) have an effect in the frequency of oscillation of the trajectory. Variations in the radius r (blue dotted line) show unstable behaviours when the eigenvalue is greater than 1, and smoothing effects when it's smaller than 1.

Figure H.2: Ablation study and qualitative results of the *Circular motion* scenario. The labels in the Y axis indicate the variations suffered by a selected eigenvalue of the learned $\mathcal{K}$. Written in red, we indicate the behaviors we perceive.

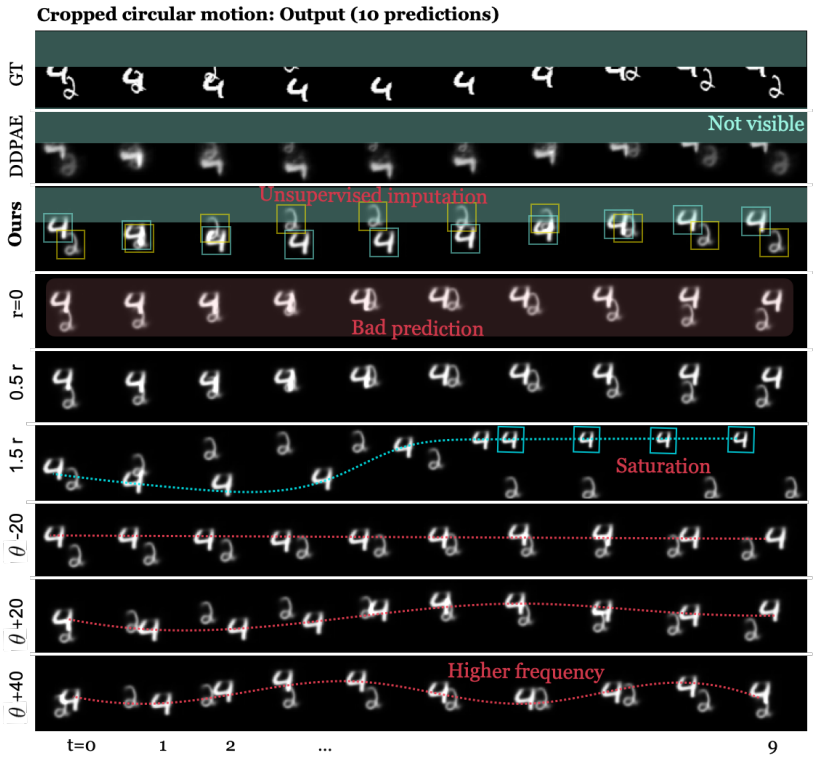

Figure H.3: Ablation study and qualitative results of the *Cropped circular motion* scenario. In transparent blue, we show the occluded area of the frame. The labels in the Y axis indicate the variations suffered by a selected eigenvalue of the learned $\mathcal{K}$. Written in red, we indicate the behaviors we perceive. In this particular case it's interesting to highlight the unsupervised imputation of the trajectory that OKID discovers in an unsupervised fashion. This indicates that the model learns the correct dynamics. DDPAE fails to reconstruct the objects properly.

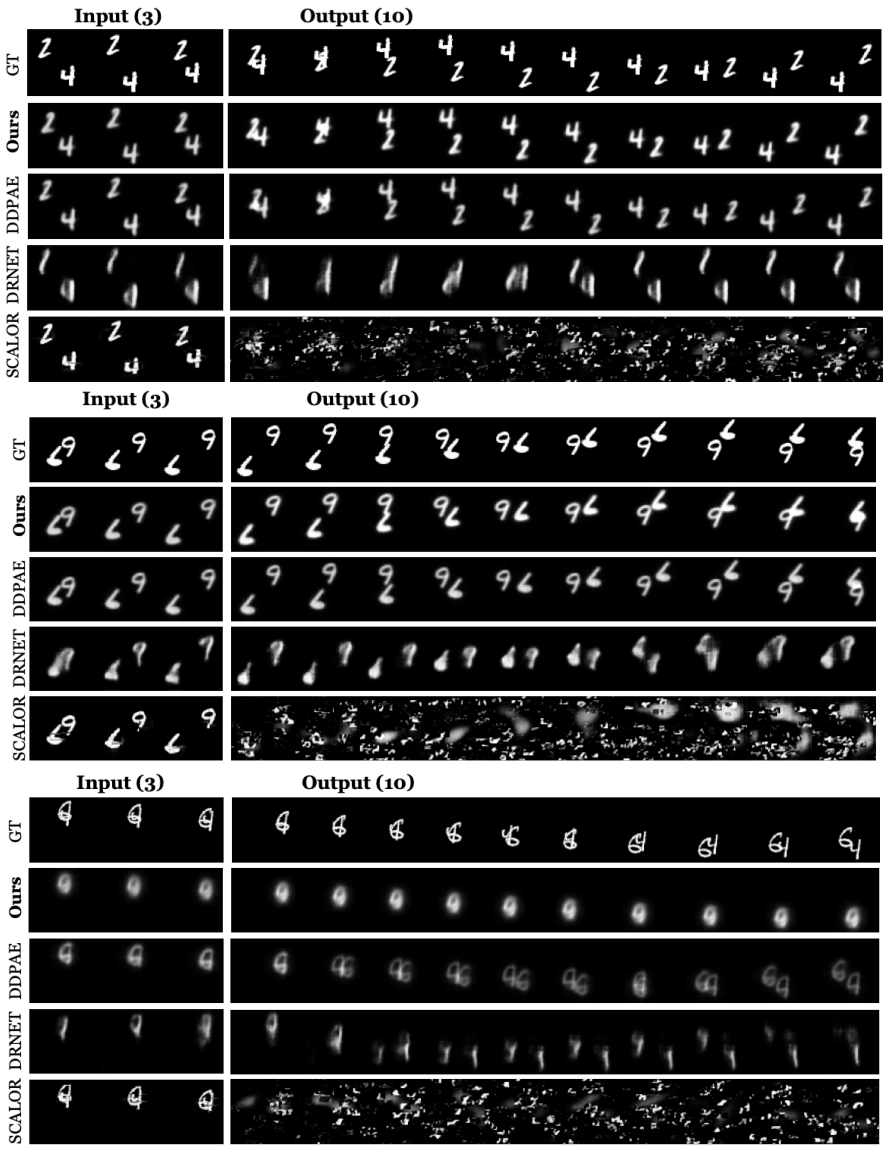

Figure H.4: Qualitative comparison (with failure cases) of OKID against the baselines for *Inelastic/Superelastic collision* case. In the top, success case where the inelastic collision is properly modelled by OKID. The best baseline, DDPAE, also reconstructs the trajectory successfully. In the center, failure case where the decomposition and reconstruction of the scene is correct, but the dynamics are slightly off for OKID, from the points of collision. In the bottom, failure case in which OKID is unable to decompose the scene. Note that SCALOR provides a very poor prediction although it has the best reconstruction.

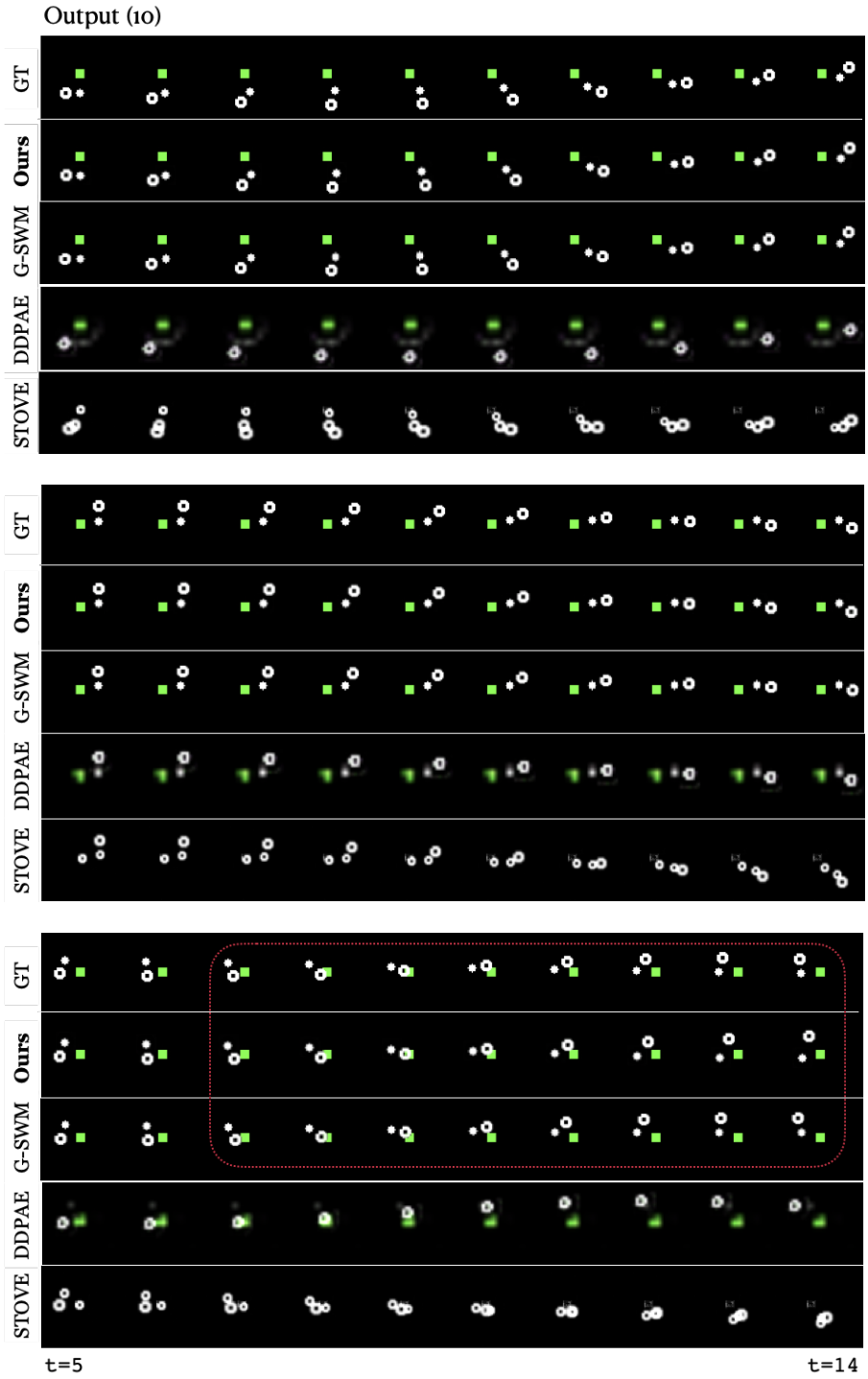

Figure H.5: Qualitative comparison of OKID against the baselines for the Double Pendulum experiments. In the top and middle, success cases where both OKID and G-SWM generate very accurate frames. In the bottom, clear failure case of OKID in the last frames of the prediction. If observed carefully, G-SWM fails in the last 8 frames. However, it is by a lower margin than OKID. DDPAE and STOVE fail both to generate sharp and accurate objects, and to correctly predict their trajectory.

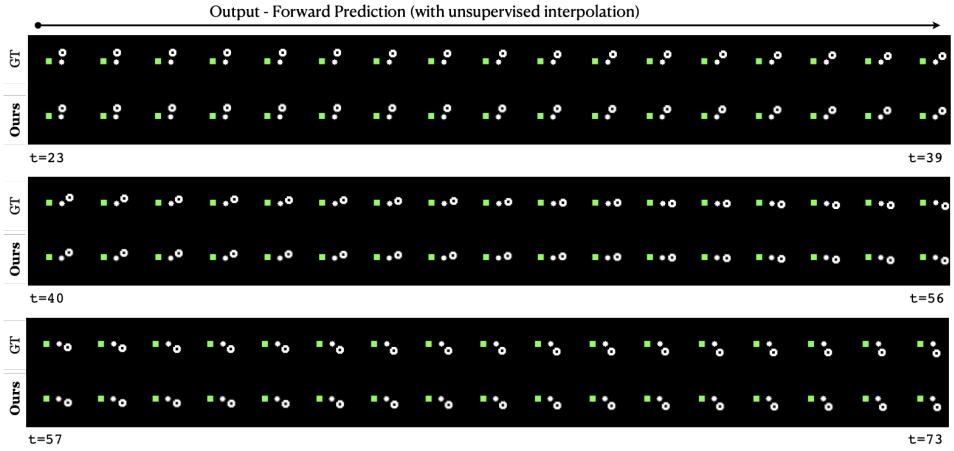

Figure H.6: Qualitative results of our model's prediction at super-resolution, evaluated in Table 2. This is a success case of 5-frames interpolation for future predictions. We can see very sharp objects located at the correct location. OKID has been trained to predict at a frame rate $5\times$ lower than the original dataset. However, at test time, it can accurately predict at the original frame rate by applying $\mathcal{K}^{(1/5)}$