

# Node-RF: Learning Generalized Continuous Space-Time Scene Dynamics with Neural ODE-based NeRFs

## Supplementary Material

Hiran Sarkar<sup>1</sup> Liming Kuang<sup>1,2</sup> Yordanka Velikova<sup>1,2</sup> Benjamin Busam<sup>1,2</sup>

<sup>1</sup>Technical University of Munich <sup>2</sup>Munich Center for Machine Learning

hiransarkar2001@gmail.com, {liming.kuang, dani.velikova, b.busam}@tum.de

### S1. Datasets

#### S1.1. Single Sequence

1. **Pendulum:** This synthetic dataset from Hofherr et al. [2] consists of 100 frames depicting a damped pendulum motion from a fixed viewpoint. Following the previous setup, we use alternate frames (0, 2, 4, ... 94) for training and the remaining frames (1, 3, 5, ... 93) for interpolation evaluation, while the last five frames (95–99) are used to assess extrapolation performance. For scene reconstruction-based methods, we select a fixed camera pose from those provided in the D-NeRF dataset. Since SimVP [1] assumes a fixed  $\Delta t$ , it cannot perform interpolation; thus, we train it on all frames up to 94 (0–94) and evaluate on the remaining five frames (95–99) for extrapolation. Our framework leverages the static background provided by the dataset by learning a separate latent code  $z_{static}$  during the warmup stage, which is added to the decoder output to obtain the NeRF latents  $z_{t_i}$

#### S1.2. Multi Sequence Generalization

1. **Oscillating Ball:** This is a multi-view synthetic dataset featuring a ball dropped into a bowl, oscillating back and forth twice. The scene is captured using nine static cameras arranged in a  $3 \times 3$  grid. Each camera records two sequences with different initial positions of the ball, resulting in distinct trajectories. In total, the dataset contains 17 sequences, each corresponding to a unique initial position. All sequences start with zero initial velocity and consist of 90 frames. In addition to the multi-view images and camera poses, the dataset provides the 3D position of the ball at every timestep and an image of the static background. For our experiments, we subsample 25 sub-sequences from each original sequence, using the first 25 frames as the starting state. This enables the generation of sequences with varying initial velocities, computed as the

difference between consecutive positions ( $v_t = p_{t+1} - p_t$ ). We train on 16 of the original sequences, resulting in  $25 \times 16$  sub-sequences with diverse initial conditions of pose and velocity. The 17<sup>th</sup> sequence is reserved for evaluation.

2. **Bifurcating Hill:** This is a single-view synthetic dataset depicting a ball placed at the crest of a hill that eventually settles into a trough on either side. The dataset contains nine sequences, each consisting of 90 states. Similar to the previous setup, we subsample 25 sub-sequences from each original sequence, resulting in trajectories with varying initial poses and velocities. We use eight of the original sequences for training and reserve the remaining one for evaluation. For this dataset, we obtain the ground-truth 2D poses using CNOS [3]. The dataset also includes an image of the static background.

### S2. Baselines

#### S2.1. Multi-Sequence Generalization

We provide more information on the choice and implementation of the baselines used for the multi-sequence generalization experiments.

1. **D-NeRF(c):** This is a modified version of D-NeRF that we adapt to incorporate conditioning on the initial conditions. Specifically, we condition the time-deformation network on the initial state, enabling it to deform rays from the frame space to a canonical space that corresponds not only to the timestep but also to the initial conditions. This allows the model to represent multiple sequences within a single implicit representation. Formally,

$$M : (\Psi_{t,c} : (x, t, c), d) \rightarrow (c, \sigma), \quad (1)$$

where  $\Psi_{t,c}$  denotes the deformation network that warps the 3D point  $x$  from the frame space to the canonical space given time  $t$  and initial condition  $c$ . This baseline is evaluated on both the multi-view *Oscillating Ball* dataset and the single-view *Bifurcating Hill* dataset.

2. **SimVP** [1]: This is a video prediction model built entirely on a simplified CNN architecture. We include this baseline to compare our approach against a future frame prediction model in the generalization task, as it aligns with our objective. However, SimVP is restricted to videos captured from a fixed camera viewpoint and is therefore not applicable to 3D or multi-view datasets. Consequently, we evaluate this model only on the *Bifurcating Hill* dataset. For a fair comparison, we provide the first two frames as input and allow the model to predict the subsequent frames. These initial frames supply information about the starting position and velocity, ensuring consistency with our framework.
3. **Vid-ODE** [4]: This is a video generation model based on neural ODEs, designed to learn continuous-time dynamics. Since it is not equipped to handle camera poses, we evaluate it only on the 2D *Bifurcating Hill* dataset. The model takes two input frames and predicts the subsequent two frames, ensuring a fair comparison with our method. During inference, it autoregressively generates future frames starting from the two input frames.

### S3. Algorithms

**Single Sequence** Algorithm S1 provides the training steps of the *single sequence* experiments.

---

#### Algorithm S1 Single Sequence Training

---

**Require:** Images  $I_t$ ,  $t \in T$ , NeRF model  $F_\Theta$ , latents  $z \in \{z_{t_0}, z_{t_1}\}$ , ODE-RNN, nODE model  $f_\theta$ , decoder  $\mathcal{D}$ .  
Initialize latent codes  $z_{t_0}, z_{t_1}$  for the first two frames  
**while** warm-up iterations not completed **do**  
 $\hat{I} = F_\Theta(z)$   
 $\mathcal{L} \leftarrow \mathcal{L}_{\text{NeRF}}(\hat{I}_k, I_k)$ ,  $k \in \{0, 1\}$   
 $\Theta, z \leftarrow \text{Update}()$   
**end while**  
**while** joint training iterations not completed **do**  
 $(\mu, \sigma) \leftarrow \text{ODE-RNN}(z_{t_0}, z_{t_1})$   
 $z_{t_0}^{\text{dyn}} \sim \mathcal{N}(\mu, \sigma)$   
 $z_{t_i}^{\text{dyn}} = \text{ODESolve}(f_\theta, z_{t_0}^{\text{dyn}}, t_i)$   
 $z_{t_i} = \mathcal{D}(z_{t_i}^{\text{dyn}})$   
 $\hat{I}_i = F_\Theta(z_{t_i})$   
 $\mathcal{L} \leftarrow \mathcal{L}_{\text{NeRF}}(\hat{I}_i, I_i)$   
 $\Theta, \text{ODE-RNN}, f_\theta, \mathcal{D} \leftarrow \text{Update}()$   
**end while**

---

**Multi Sequence Generalization** Algorithm S2 provides the training steps of the *multi sequence generalization* experiments

### S4. Analysis

#### S4.1. Effects of the number of nODE layers

Table S1 presents the ablation study on the *Oscillating Ball* dataset, analyzing the effect of the number of layers in the

---

#### Algorithm S2 Generalized Multi-Sequence Training

---

**Require:** Images  $I_t^c$ ,  $t \in T$  of sequence  $c \in \{0, \dots, N\}$ , static background  $I_{\text{static}}$ , initial poses  $p_0^c$ , velocities  $v_0^c$ . Poses  $p_{t_i}^c$ , velocities  $v_{t_i}^c$  to predict, NeRF model  $F_\Theta$ , latents  $z_{\text{static}}, z_{\text{can}}$ , encoder  $\mathcal{E}$ , nODE model  $f_\theta$ , decoders for velocity, pose, scene:  $\mathcal{D}_v, \mathcal{D}_p, \mathcal{D}_n$ .  
Initialize  $z_{\text{static}}$  for the static background  
**while** warm-up iterations not completed **do**  
 $\hat{I} = F_\Theta(z_{\text{static}})$   
 $\mathcal{L} \leftarrow \mathcal{L}_{\text{NeRF}}(\hat{I}_{\text{static}}, I_{\text{static}})$   
 $\Theta, z_{\text{static}} \leftarrow \text{Update}()$   
**end while**  
**while** joint training iterations not completed **do**  
 $z_{t_0, c}^{\text{dyn}} = \text{concat}(z_{\text{can}}, \mathcal{E}(p_0^c), v_0^c)$ ;  
 $z_{t_i, c}^{\text{dyn}} = \text{ODESolve}(f_\theta, z_{t_0, c}^{\text{dyn}}, t_i)$   
 $z_{t_i, c} = \mathcal{D}_n(z_{t_i, c}^{\text{dyn}}) + z_{\text{static}}$   
 $\hat{p}_{t_i}^c, \hat{v}_{t_i}^c, \hat{I}_{t_i}^c \leftarrow \mathcal{D}_p(z_{t_i, c}^{\text{dyn}}, \mathcal{D}_v(z_{t_i, c}^{\text{dyn}}), F_\Theta(z_{t_i, c}))$   
 $\mathcal{L} \leftarrow \mathcal{L}_{\text{NeRF}}(\hat{I}_{t_i}^c, I_{t_i}^c)$   
 $\quad + \mathcal{L}_p(\hat{p}_{t_i}^c, p_{t_i}^c) + \mathcal{L}_v(\hat{v}_{t_i}^c, v_{t_i}^c) + \mathcal{L}_{\text{lipschitz}}$   
 $\Theta, f_\theta, \mathcal{D}_n, \mathcal{D}_p, \mathcal{D}_v, z_{\text{can}} \leftarrow \text{Update}()$   
**end while**

---

nODE MLP  $f_\theta$ . Each experiment runs for 400k iterations. We observe a performance drop in the 5-layer  $f_\theta$ , where the model struggles to generate sequences with novel initial conditions. Although the 7-layer  $f_\theta$  shows an improvement in performance compared to the 5-layer variant, it still underperforms relative to the 3-layer model. We also observe from the visual results that, on the *Bifurcating Hill* dataset, the 7-layered  $f_\theta$  completely underfits the scene and fails to learn the dynamics, whereas the 5-layered  $f_\theta$  captures the dynamics but underperforms compared to the 3-layered  $f_\theta$ .

Table S1. Ablation study of nODE layers on different evaluation metrics in *Oscillating Ball* dataset.

# nODE Layers	SSIM	LPIPS	PSNR	IoU
3	0.662	0.4364	29.091	0.3327
5	0.605	0.5171	27.580	0.1864
7	0.645	0.4721	28.858	0.3060

#### S4.2. Effects of the number of wamrup latents

Table S2 presents an ablation study on the number of latents trained during the warmup stage in the single-sequence setup. The evaluation is conducted on the *reconstruction* split of the *Bouncing Balls* dataset. Quantitatively, the results remain largely comparable across different configurations. However, qualitative observations indicate that using a higher number of warmup latents leads to more stable long-term extrapolations. We select two warmup latents as it represents the minimum number required to capture the initial dynamics of the scene.

Table S2. Ablation studies of warmup latent count on *Reconstruction* set of *Bouncing Balls*

# Count	SSIM	LPIPS	PSNR
2	0.978	0.0310	33.70
5	0.978	0.0289	32.90
10	0.979	0.0307	33.54

### S4.3. Training Time

Table S3 reports the training time for each scene. The generalization experiments involving multiple sequences take significantly longer than the single-sequence experiments, requiring roughly three days to complete. Moreover, the training time grows proportionally with the length of the input sequences, highlighting a current limitation of our method.

Dataset	Solver	Training Time
Bouncing Balls	dopri5	18 hours
Pendulum	euler	24 hours
Sear Steak	euler	36 hours
Oscillating Ball	euler	72 hours
Bifurcating Hill	euler	72 hours

Table S3. Training time per dataset.

## References

- [1] Zhangyang Gao, Cheng Tan, Lirong Wu, and Stan Z Li. Simvp: Simpler yet better video prediction. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3170–3180, 2022. [1](#), [2](#)
- [2] Florian Hofherr, Lukas Koestler, Florian Bernard, and Daniel Cremers. Neural implicit representations for physical parameter inference from a single video. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2093–2103, 2023. [1](#)
- [3] Van Nguyen Nguyen, Thibault Groueix, Georgy Ponimatin, Vincent Lepetit, and Tomas Hodan. Cnos: A strong baseline for cad-based novel object segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2134–2140, 2023. [1](#)
- [4] Sunghyun Park, Kangyeol Kim, Junsoo Lee, Jaegul Choo, Joonseok Lee, Sookyung Kim, and Edward Choi. Vid-ode: Continuous-time video generation with neural ordinary differential equation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 2412–2422, 2021. [2](#)