Forecasting Continuous Non-Conservative Dynamical Systems in SO(3)

Appendix

Lennart Bastian[1,2*] **Mohammad Rashed**[1,2*] **Nassir Navab**[1,2] **Tolga Birdal**[3]

[1] **Technical University of Munich** [2] **Munich Center of Machine Learning** [3] **Imperial College London**

Contents

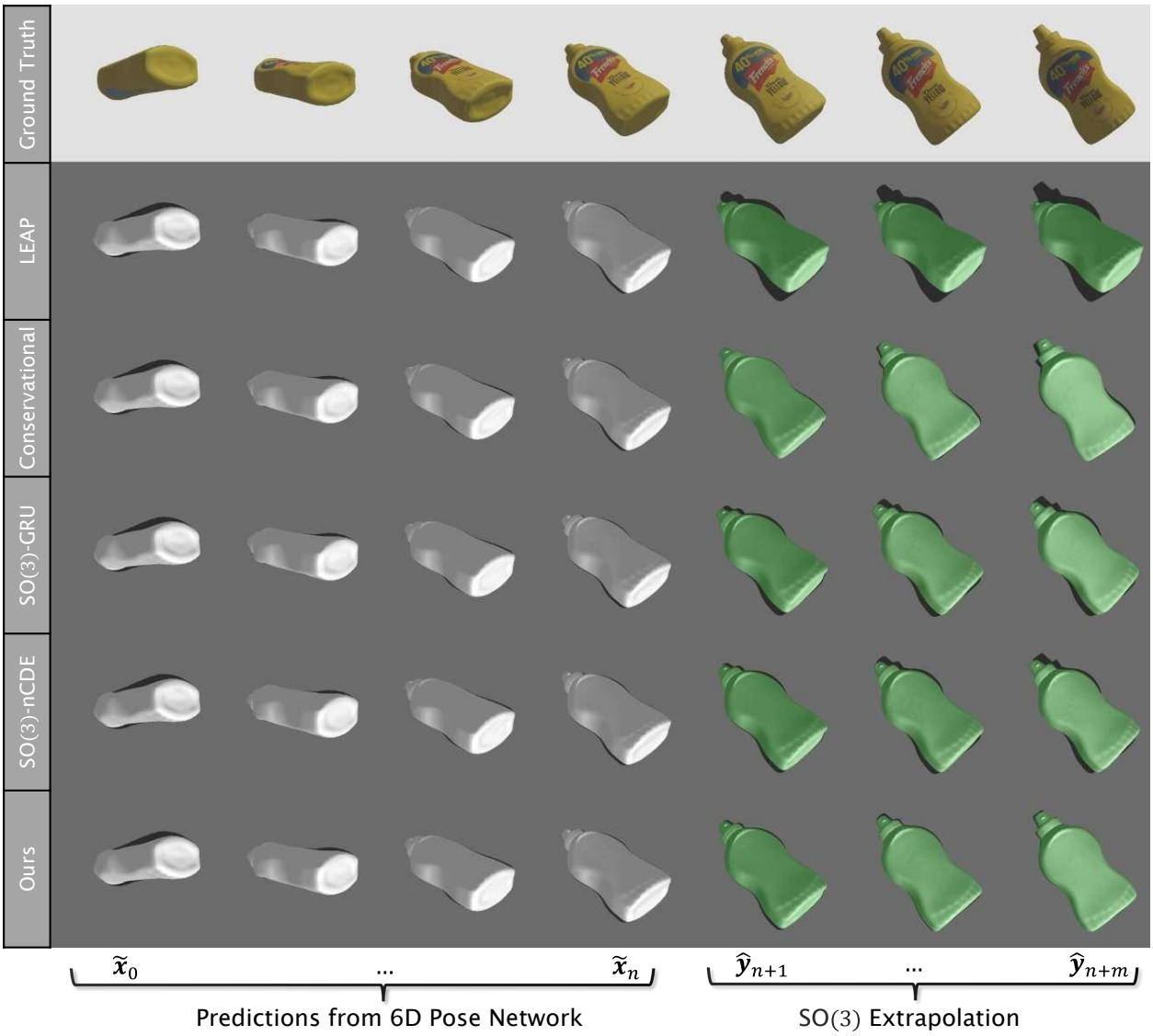

Figure 1. Application: 6D object pose estimation. We compare our method with the baselines by applying all models on the pose prediction outputs of GDRNet [17]. 6D pose predictions (bottom left) are used as input to the $\mathrm{SO}(3)$extrapolation methods. Extrapolated outputs are depicted in green. Our method is able to predict more robust future states based on noisy pose estimates.

A. Experiments and Ablations

A.1. 6D Pose Estimation

In this section, we provide additional details regarding the 6D Pose estimation application introduced in the main paper. A total of 25k images are generated using BlenderProc2 [4]. The synthetic scenes featured a single object (a mustard bottle) from the YCB-Video dataset [18], with the object's orientation randomly sampled. We then train GDRNet [17] on individual (non-sequential) RGB images. The publicly available pre-trained weights are fine-tuned for one training epoch, without data augmentations.

Figure 1 illustrates simulated trajectories of the object for the configuration-dependent torque scenario. In this experiment, image sequences are given as input to the pose estimator, producing noisy $\mathrm{SO}(3)$ rotation trajectories. These are then used as input for the $\mathrm{SO}(3)$ extrapolation methods. We compare our method to the baselines LEAP, Conservational, $\mathrm{SO}(3)$-GRU, and $\mathrm{SO}(3)$-nCDE.

As shown in Figure 1, the LEAP baseline struggles to handle the rotational behavior effectively. The conservational

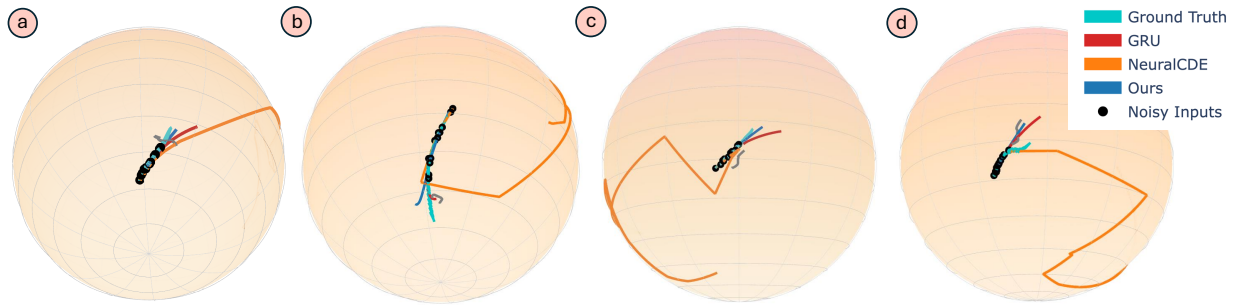

Figure 3. We evaluate the best-performing models' extrapolation capabilities under irregular sampling from noisy samples fused from multiple sensors. The trajectories are visualized on the unit sphere $\mathbb{S}^2$.

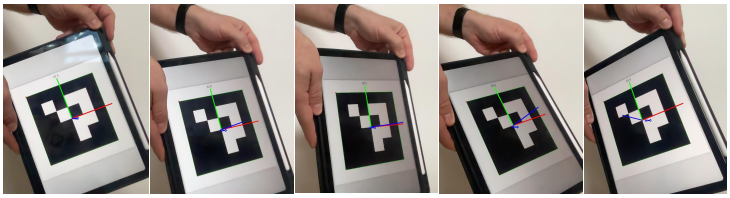

Figure 4. Evaluating the robustness of the proposed extrapolation methods for irregularly sampled extrapolation in the wild. The IMU signal from a tablet is synchronized with an external camera which captures the pose of a displayed ArUco marker. The resulting noisy and irregular trajectories are used to evaluate the models' extrapolation capabilities.

Table 1. Comparison of prediction errors across different models from irregularly sampled noisy sensor measurements. Results show mean ± standard deviation in rotational geodesic error (RGE).

Model	Prediction Error
LEAP	18.30 ± 5.45
$\mathrm{SO}(3)$-GRU	11.39 ± 0.73
$\mathrm{SO}(3)$-nCDE	75.48 ± 9.39
Ours	$\mathbf{8.11 \pm 1.59}$

approach effectively predicts rotation within this time horizon but does not capture the damping behavior and overshoots slightly. While the $\mathrm{SO}(3)$-GRU and $\mathrm{SO}(3)$-nCDE baselines perform better, managing to account for the damped patterns, the extrapolated poses produced by these methods exhibited higher errors than our approach.

A.2. Application: Irregularly Sampled Sensor Fusion

To evaluate the capabilities of the proposed method in real-world scenarios, we study the case of extrapolation in a sensor fusion application. The data rig consists of a tablet depicting an ArUco [5] marker that streams the onboard inertial measurement unit (IMU) sampled at 100Hz sensor signal to a system that simultaneously captures the 6D pose of the tablet with a camera (sampled at 30Hz) as shown in figure Fig. 4. The sensor is then moved manually and tracked, a particularly challenging (and potentially ill-defined) scenario as stochastic external torques can influence the device during the extrapolation period. Sensors are calibrated spatiotemporally using a Levenberg-Marquardt optimization, after which we perform outlier rejection based on a simple median-absolute-deviation heuristic. We then capture 24 trajectories with the sensor rig to evaluate the models trained on the simulated *combined external torque* scenarios. The trajectories from the two sensors are naively merged according to the registered timestamps and then used as input to the models.

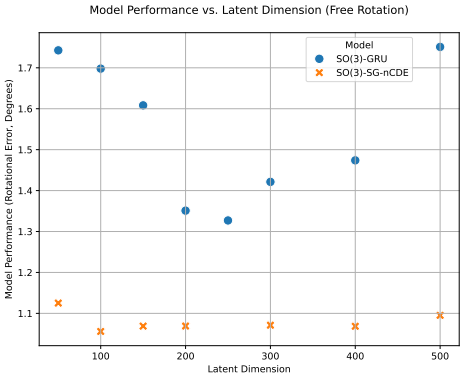

Figure 2. Ablating latent dimension vs. prediction performance in GRU vs. neural CDEs on the *freely rotating* scenario with $\delta = 0.05$ and a prediction horizon of $t = 0.8s$.

Results The results are depicted in Tab. 1, with visualizations in Fig. 3. Our method outperforms the baselines LEAP and $\mathrm{SO}(3)$-nCDE by a large margin. In contrast to the simulated scenarios, the latter cannot accommodate the irregular, noisy signals and creates errant extrapolations. The GRU performs competitively, despite the irregular sampling, and produces reasonable estimates. Fig. 3 (d) depicts an example where the applied torque changes in a stochastic manner during the extrapolation horizon, which the models are not

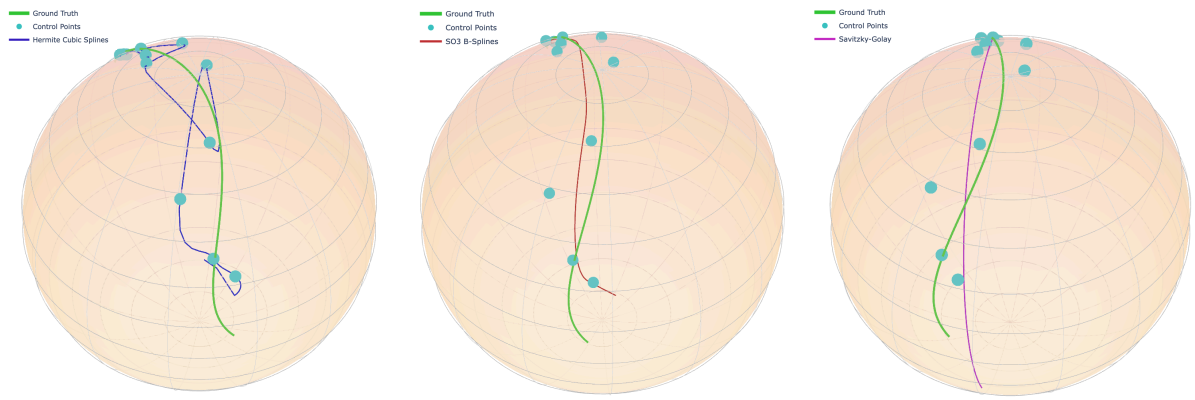

Figure 5. Comparison of interpolation methods on a particularly complex and noisy trajectory from the *combined external force* scenario. From left to right: Hermite cubic splines with backward differences [10], $\mathrm{SO}(3)$ B-splines [14], and $\mathrm{SO}(3)$ Savitzky Golay Filtering [7]. Trajectories represent interpolation and $t = 0.2s$ extrapolation into the future.

able to capture as they have no knowledge of such events.

A.3. Hyperparameters

As the latent dimension significantly impacts model performance for both the GRU and nCDE-based methods, we provide an additional ablation over the latent dimension (see Fig. 2).

While increasing the latent dimension of the GRU leads to overfitting, $\mathrm{SO}(3)$-SG-nCDEs are robust to changes in latent dimension and overfitting. $\mathrm{SO}(3)$-nCDEs and $\mathrm{SO}(3)$-SG-nCDEs are trained with a latent dimension of 125, while GRUs are trained with a latent dimension of 250 and 3 hidden units. LEAP uses a latent dimension of 50; increasing the latent dimension for this method significantly increases runtime and VRAM but not performance.

A.4. Choosing Integration Paths

Various control signals for neural CDEs have been proposed; most interpolation methods use cubic or higher order splines [8, 10]. Kidger et al. [8] construct the integration path $X \in \mathcal{C}^2$ as a natural cubic spline over the input sequence. While this guarantees smoothness of the first derivative, Hermite cubic splines with backward differences are preferred for CDE integration due to local control [10]. We observe that these choices do not obey the geometric structure of $\mathrm{SO}(3)$; furthermore, they are not robust with respect to sensor noise interpolating the points erratically (see Fig. 5, left). In general, such higher-order polynomials tend to diverge near the interpolation endpoints and must be extrapolated with some assumption, for instance, constant velocity [11]. However, when dynamics are complicated, this assumption is overly simplifying, even more so than assuming conservation of energy as in [9].

Efficient derivative calculations have been proposed for robust $\mathrm{SO}(3)$ B-spline interpolation of trajectories in $\mathrm{SO}(3)$ [14]. These can be made suitable for extrapolation with, e.g., constant velocity extrapolation at the endpoint by constructing a nonlinear optimization to minimize residuals to the measurements, with additional smoothness near the endpoint. We implement this in Ceres [1], Basalt [16], and Sophus [15] to integrate Lie group constraints into the manifold optimization. We use the efficient derivative computation of [14] and an additional first-order Taylor expansion using the computed Jacobians to include a constant velocity endpoint constraint defining the splines beyond their support interval. While the splines nicely interpolate the noisy trajectory and can be used to extrapolate (see Fig. 5, center), they still exhibit bias towards endpoints. Moreover, this approach is too computationally expensive to use during network training. On a 24-Core Intel(R) Xeon(R) CPU, a single batch of 1000 trajectories takes ≈ 22 seconds when maximally parallelized, a fraction of the trajectories we use for training.

On the other hand, the Savitzky-Golay filter provides a reasonable approximation at a low cost. As it merely requires solving a modest linear system, this means it can be differentiated through to learn a robust weighting suitable for extrapolation (see Fig. 5, right).

A.5. Model performance vs Input Noise

In Fig. 6, we further evaluate the quality of the predictions over varying time horizons (1, 4, 6, 8) and level of noise $[0.02, 0.03, 0.04, 0.05]\pi$ on the *velocity damping* scenario. The results are depicted in Fig. 6. Notably, all methods

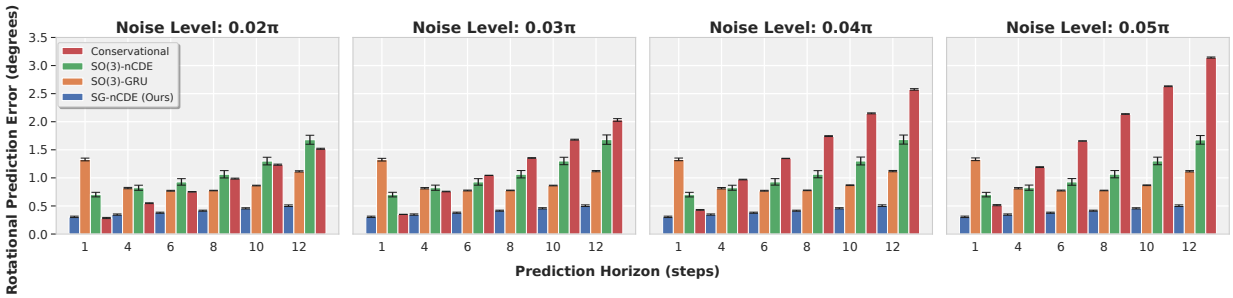

Figure 6. Rotational geodesic error for different input noise levels on the velocity damping scenario. We compare the performance of the baseline neural CDE and our best performing method (weighted, 2nd-order) with different amounts of simulated noise added to the simulated dynamical trajectories. Noise is sampled from $\mathcal{N}(0, \delta)$ with $\delta \in [0.02, 0.05]\pi$.

suffer from increased prediction errors as noise increases, but our method consistently outperforms the baselines.

B. Implementation Details

We provide further details regarding the five simulated experimental scenarios described in the main paper.

B.1. Experimental Design

For each experiment, we sample an initial orientation according to the strategy in [13], and an initial angular velocity is sampled according to a truncated normal distribution to avoid degenerate cases with a small angular velocity.

We sample via rejection sampling:

1. Sample $X \sim \mathcal{N}(0, \sigma)$
2. Accept if $|X| > \eta$, otherwise repeat step 1,

with $\sigma = 0.3$ and $\eta = 0.1$

B.2. Moment of Inertia Distributions for Simulation

To evaluate model generalization across different rigid body properties, we simulate trajectories using four distinct moment of inertia (MOI) distributions. Each distribution is characterized by a base diagonal MOI tensor and additive noise:

$$\mathbf{J} = \text{diag}(\mathbf{J}_{\text{base}}) + \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(0, \sigma^2 \mathbf{I}_3) \tag{1}$$

where $\mathbf{J}_{\text{base}}$ is the base MOI vector and $\sigma = 0.2$ is the noise standard deviation. The four distributions use the following base MOI configurations:

$$\mathbf{J}_{\text{base}}^{(1)} = [1.0, 2.0, 3.0] \tag{2}$$
$$\mathbf{J}_{\text{base}}^{(2)} = [3.0, 1.0, 2.0] \tag{3}$$
$$\mathbf{J}_{\text{base}}^{(3)} = [3.0, 2.0, 1.0] \tag{4}$$
$$\mathbf{J}_{\text{base}}^{(4)} = [2.0, 3.0, 1.0] \tag{5}$$

These distributions represent different permutations of the same eigenvalues, creating rigid bodies with varied principal axis orientations. This approach challenges models to generalize across different rotational behaviors resulting from the same underlying physics but with different inertial configurations.

B.3. Simulation Scenarios

Freely Rotating. Freely rotating objects are governed by kinematic equations. Without any external torques, they reduce to the following:

$$\mathbf{J}\dot{\boldsymbol{\omega}} = -\boldsymbol{\omega} \times \mathbf{J}\boldsymbol{\omega} \tag{6}$$

While this equation is simplified and maintains energy conservation, the gyroscopic torque defined through the cross-product is highly non-linear and can lead to chaotic motion depending on the mass distribution.

Linear Control. In the case of having an identity moment of inertia matrix, the gyroscopic torque can be neglected, and the kinematic equations

$$\mathbf{J}\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times \mathbf{J}\boldsymbol{\omega} = \tau_{\text{ext}} \tag{7}$$

reduce to:

$$\mathbf{J}\dot{\boldsymbol{\omega}} = \tau_{\text{ext}} \tag{8}$$

To this end, we simulate a linear controller where all acceleration is *driven* by an external torque which changes according to the linear control law $\tau_{\text{ext}} = \mathbf{J}(\mathbf{A}\boldsymbol{\omega} + \mathbf{b})$ where $A \in \mathbb{R}^{3\times3}$ and $b \in \mathbb{R}^3$.

By defining τ_{ext} in this manner, we see that the angular velocity changes according to the external control signal directly as a function of orientation: $\dot{\boldsymbol{\omega}} = (\mathbf{A}\boldsymbol{\omega} + \mathbf{b})$, which is a useful simplified external torque without highly non-linear gyroscopic effects. Notably, the baseline $\mathrm{SO}(3)$-nCDE and $\mathrm{SO}(3)$-GRU also perform competitively in this scenario due to it's simplicity.

Configuration Dependent Torque. We further consider scenarios where the external torque depends explicitly on the current orientation, leading to

$$\mathbf{J}\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times \mathbf{J}\boldsymbol{\omega} = \tau_{\text{ext}}(R) \tag{9}$$

where $R \in \mathrm{SO}(3)$ is the current orientation. This class of torques arises in various physical systems where the interaction with external fields generates configuration-dependent forcing, such as magnetic dipoles in uniform fields or gravitational gradients.

Unlike the linear control case, the rotational motion is coupled directly to the orientation state in a global reference frame, potentially leading to multiple equilibria and complex trajectories. The external torque takes the general form $\tau_{\text{ext}}(R) = f(R\mathbf{v})$ where $\mathbf{v} \in \mathbb{R}^3$ represents some body-fixed vector and $f : \mathbb{R}^3 \to \mathbb{R}^3$ is a nonlinear function that describes the physical interaction in the world frame. We observe this scenario to be most challenging for the models as the trajectories can exhibit erratic motion.

To limit chaotic behavior, we additionally apply a damping signal (e.g., simulating friction or other dissipative forces). We initialize the damping matrix $\mathbf{D}$ to be negative definite such that $\lim_{t\to\infty} \|\boldsymbol{\omega}(t)\| = 0$ while the object is also under the effect of the internal gyroscopic motion:

$$\mathbf{D} = \begin{bmatrix} -0.2 & 0 & 0 \\ 0 & -0.2 & 0 \\ 0 & 0 & -0.2 \end{bmatrix} \tag{10}$$

These are combined for the following total external torque:

$$\tau_{\text{ext}} = w_1\tau_{\text{config}}(R) + w_2\tau_{\text{damp}}(\boldsymbol{\omega}) \tag{11}$$

where the weights w_i allow for scaling individual contributions. Despite combining multiple effects, these scenarios often exhibit more predictable behavior than pure configuration-dependent cases, as damping terms provide a stabilizing effect by dissipating energy from the system.

B.4. Baseline Model Design

We provide additional details regarding the baseline models. Specifically, how we adapted the $\mathrm{SO}(3)$-GRU and $\mathrm{SO}(3)$-nCDE baselines to predict rotations in $\mathrm{SO}(3)$. We also detail the essential components of the conservation approach [9] that are used to predict rotations directly without confounding pose estimation from images with object dynamics.

$\mathrm{SO}(3)$**-nCDE**. We adapt the neural CDE baseline from [8] to predict rotations in $\mathrm{SO}(3)$. Hermit cubic coefficients are directly constructed on the input 9D rotation representation [19] via backward differences. Following [8], the method encodes an initial value z_0, which is then integrated forward in time using *torchdiffeq* [3] and Dormand-Prince 4/5 with respect to the constructed spline. The latent representation is then decoded into the 6D rotation representation [6, 19], and transformed to a rotation matrix with Gram-Schmidt orthonormalization (GSO).

Table 2. Comparison of prediction errors with the conservational approach [9] using a ground truth momentum estimate across different experimental conditions. Results show mean ± standard deviation in rotational geodesic error (RGE). First and second best results are emphasized.

Extrapolation Horizon	Method	Freely Rotating	Linear Control	Velocity Damping	Configuration-Dependent Torque	Variable Dynamics
$t = 0.8s$	Conservational ([9])	1.15 ± 0.00	1.25 ± 0.00	1.14 ± 0.00	1.26 ± 0.00	1.43 ± 0.00
	SG-nCDE (Ours)	$\mathbf{0.87 \pm 0.05}$	$\mathbf{0.49 \pm 0.03}$	$\mathbf{0.42 \pm 0.02}$	$\mathbf{0.58 \pm 0.03}$	$\mathbf{0.89 \pm 0.07}$
$t = 1.2s$	Conservational ([9])	$\mathbf{1.15 \pm 0.00}$	2.05 ± 0.00	1.87 ± 0.00	2.05 ± 0.00	2.44 ± 0.01
	SG-nCDE (Ours)	1.28 ± 0.08	$\mathbf{0.64 \pm 0.03}$	$\mathbf{0.50 \pm 0.03}$	$\mathbf{0.74 \pm 0.04}$	$\mathbf{1.31 \pm 0.14}$

$\mathrm{SO}(3)$**-GRU**. The GRU baseline consists of a multi-layer gated recurrent unit (GRU) RNN with three recurrent units. The GRU is similarly applied sequentially on each 9D rotation representation, yielding a 6D prediction converted to an element in $\mathrm{SO}(3)$ via GSO.

Conservational [9]. The conservation of energy approach of [9] is designed to learn a rotation representation directly from images. A sequence of images is used to estimate the object's momentum, upon which they integrate the kinematics equations (the authors equivalently use the Lie–Poisson formulation) forward in time and obtain estimates of the $\mathrm{SO}(3)$ state change under energy conservation. They then reconstruct the predicted image sequence, and estimate predicted rotation accuracy based on image reconstruction.

We observe that the dynamics component of this pipeline is identical to how we simulate the *Free Rotation* scenario in Appendix B.3. However, for forecasting rotations from a trajectory, this approach has several limitations:

1. by solving an initial value problem, they can only condition the solve with a single value
2. high sensitivity to the initial momentum estimate
3. cannot handle non-conservative systems ($\tau_{ext} \neq 0$)

To study this method more generally, we provide a momentum estimate and similarly integrate the equations of motion forward in time, evaluating the accuracy directly via rotational geodesic error (RGE) instead of for a downstream image reconstruction as in [9]. In this way, we decouple the image reconstruction task and the associated image quality metrics (the authors use pixel mean-squared error) from dynamics estimation. We emphasize that our problem setting encompasses this task; any dynamics module can be plugged directly into an object pose estimator to forecast rotation.

For a fair comparison, we additionally provide results with a ground-truth momentum estimate, observing that our method outperforms [9] in all cases but conservational, where the trajectories are parallel to the ground-truth trajectories (they are offset by the noise of the last observation). The results can be seen in table Tab. 2.

In practice, obtaining an accurate momentum estimate of an unknown object would be challenging to obtain from a single trajectory as it essentially requires estimating the velocity and mass distribution. The authors demonstrate this on a limited set of synthetically generated and well-cropped images; however, such an approach does not generalize to unknown mass distributions, let alone objects, limiting applicability in practice.

C. Additional Background

C.1. Moment of Inertia and Diagonalization

The moment of inertia (MOI) tensor $\mathbf{J}$ is defined as the integral of the mass distribution that characterizes a body's resistance to rotational acceleration about any axis. For a continuous mass distribution with density $\eta(\mathbf{r})$:

$$J_{ij} = \begin{bmatrix} J_{xx} & J_{xy} & J_{xz} \\ J_{xy} & J_{yy} & J_{yz} \\ J_{xz} & J_{yz} & J_{zz} \end{bmatrix} \tag{12}$$

with components:

$$
\begin{aligned}
J_{xx} &= \int \eta(\mathbf{r})(y^2 + z^2)\, dV \\
J_{yy} &= \int \eta(\mathbf{r})(x^2 + z^2)\, dV \\
J_{zz} &= \int \eta(\mathbf{r})(x^2 + y^2)\, dV \\
J_{xy} = J_{yx} &= -\int \eta(\mathbf{r}) xy\, dV \\
J_{xz} = J_{zx} &= -\int \eta(\mathbf{r}) xz\, dV \\
J_{yz} = J_{zy} &= -\int \eta(\mathbf{r}) yz\, dV
\end{aligned}
$$

where $\mathbf{r} = (x, y, z)$ and $\mathbf{r}^2 = x^2 + y^2 + z^2$. Diagonalization yields the principal moments:

$$
\mathbf{P}^{-1}\mathbf{J}\mathbf{P} = \begin{bmatrix} I_1 & 0 & 0 \\ 0 & I_2 & 0 \\ 0 & 0 & I_3 \end{bmatrix} \tag{13}
$$

where I_k are eigenvalues of $\mathbf{J}$ and $\mathbf{P}$ contains the corresponding eigenvectors as columns.

As any MOI tensor can be diagonalized, we consider only diagonal uniform and non-uniform MOI in our numerical simulations, with objects rotating in the canonicalized coordinate system. We note that [9] also consider several fixed non-diagonal MOIs, but these reduce to diagonal cases [2]. Instead, we sample from distributions around non-uniform MOI tensors, which are varied during training, validation, and testing, forcing models to generalize.

C.2. Exponential and Logarithmic Maps

Definition 1 (Exponential Map on SO(3))**.** *The exponential map* $\mathrm{Exp} : \mathfrak{so}(3) \to \mathrm{SO}(3)$ *is a surjective mapping from the Lie algebra* $\mathfrak{so}(3)$ *to the Lie group* $\mathrm{SO}(3)$ *defined as:*

$$
\mathrm{Exp}(\boldsymbol{\xi}) = \sum_{n=0}^{\infty} \frac{1}{n!}\boldsymbol{\xi}^n = \mathbf{I} + \boldsymbol{\xi} + \frac{1}{2!}\boldsymbol{\xi}^2 + \frac{1}{3!}\boldsymbol{\xi}^3 + \cdots \tag{14}
$$

where $\boldsymbol{\xi} \in \mathfrak{so}(3)$ *is a skew-symmetric matrix.*

Geometrically, if $\boldsymbol{\xi} = \hat{\mathbf{v}}$ *for some* $\mathbf{v} \in \mathbb{R}^3$ *with* $\|\mathbf{v}\| = \theta$*, then* $\mathrm{Exp}(\boldsymbol{\xi})$ *represents a rotation by angle* θ *about the axis* $\mathbf{v}/\|\mathbf{v}\|$.

Definition 2 (Logarithmic Map on SO(3))**.** *The logarithmic map* $\mathrm{Log} : \mathrm{SO}(3) \to \mathfrak{so}(3)$ *is the local inverse of the exponential map, retrieving the corresponding Lie algebra element from a rotation matrix:*

$$
\mathrm{Log}(\mathbf{R}) = \boldsymbol{\xi} \in \mathfrak{so}(3) \tag{15}
$$

such that $\mathrm{Exp}(\boldsymbol{\xi}) = \mathbf{R}$*. This mapping is well-defined for all* $\mathbf{R} \in \mathrm{SO}(3)$ *where the rotation angle* θ *satisfies* $0 \leq \theta < \pi$.

The logarithmic map Log is not uniquely defined for rotations of angle $\theta = \pi$, as rotations by π radians about antipodal axes $\mathbf{n}$ and $-\mathbf{n}$ yield identical rotation matrices, yielding a singularity.

C.3. SO(3) Savitzky-Golay Filtering [7]

The Savitzky-Golay filter on SO(3) solves a least-squares problem to estimate polynomial coefficients that best fit the noisy rotational data in the Lie algebra, as described in Theorem 10.

The design matrix $\mathbf{A} \in \mathbb{R}^{3(2n+1)\times 3(p+1)}$ takes the form of a Vandermonde matrix with time-shifted polynomial coefficients. It is expanded with the Kroeneker product $\otimes$ and $\mathbf{I}_3$, the identity matrix in $\mathbb{R}^3$ such that $\mathbf{A} = \hat{A} \otimes \mathbf{I}_3$. $\hat{A} \in \mathbb{R}^{(2n+1)\times(p+1)}$ is defined as:

$$\hat{A} = \begin{bmatrix} 1 & (t_{-n} - t_k) & \cdots & \frac{1}{p}(t_{-n} - t_k)^p \\ 1 & (t_{-n+1} - t_k) & \cdots & \frac{1}{p}(t_{-n+1} - t_k)^p \\ \vdots & \vdots & \vdots & \vdots \\ 1 & (t_n - t_k) & \cdots & \frac{1}{p}(t_n - t_k)^p \end{bmatrix}$$

$\mathbf{b}$ is constructed by rotational differences in the lie algebra and has the form:

$$\mathbf{b} = \begin{bmatrix} \mathrm{Log}(\tilde{\mathbf{x}}_{k-n}\tilde{\mathbf{x}}_k^{-1})^\vee \\ \vdots \\ \mathrm{Log}(\tilde{\mathbf{x}}_{k+n}\tilde{\mathbf{x}}_k^{-1})^\vee \end{bmatrix}$$

$(\cdot)^\vee : \mathfrak{so}(3) \to \mathbb{R}^3$ is the inverse of the hat operator that maps skew-symmetric matrices to vectors.

This formulation brings several advantages over conventional interpolation with splines. It filters robustly in the region of a point (with an arbitrary support window) and can be solved without iterative optimization. Moreover, the derivatives are smooth up to the order of the polynomial, which we leverage in the next section to demonstrate universal approximation based on the results from [10].

D. Proof of Proposition 10

We demonstrate our modified Savitzky-Golay filter satisfies the three properties of a suitable control path for the neural CDE as described by Morrill et al. [10]. We first restate the proposition regarding the suitability of the control path.

Proposition (10)**.** *Let $\varphi(t)$ be the control path constructed in Def. (7), with polynomial coefficients $\boldsymbol{\rho}$ based on solving the optimization problem in Thm. (8). If the maximum rotational difference $\delta_n = \max_k \|\mathrm{Log}(\tilde{\mathbf{x}}_k\tilde{\mathbf{x}}_0^{-1})\|$ within the window is bounded, then:*

1. *$\varphi(t)$ is analytic and twice differentiable,*
2. *The derivatives $\varphi'(t)$ and $\varphi''(t)$ are bounded,*
3. *$\varphi(t)$ uniquely minimizes the problem in Thm. (8).*

Proof. **Smoothness**. We begin by demonstrating sufficient smoothness of the map $\varphi(t)$. We observe that $\tilde{\mathbf{x}}_k$ is constant, and the term $\mathrm{Exp}\left(\mathbf{p}\left(t - t_k; \boldsymbol{\rho}_k\right)\right)$ is a composition of a polynomial and the exponential map.

We note that the logarithmic map $\mathrm{Log} : \mathrm{SO}(3) \to \mathfrak{so}(3)$ is not uniquely defined for rotations of angle π. Specifically, for $\|\xi\| = \pi$ the axis of rotation becomes non-unique, as rotations by π about $\mathbf{n}$ and $-\mathbf{n}$ yield the same rotation matrix. This requires constraints or assumptions on the magnitude of rotations within any given filter window to ensure smoothness and injectivity of the maps.

Since the exponential map $\mathrm{Exp} : \mathfrak{so}(3) \to \mathrm{SO}(3)$ is smooth and analytic on $\mathfrak{so}(3)$ [12], and the polynomial $\mathbf{p}\left(t - t_k; \boldsymbol{\rho}_k\right) \in C^\infty$ by definition, we have that $\varphi(t) \in C^\infty$ provided that $\|\mathbf{p}(t - t_k; \boldsymbol{\rho}_k)\| < \pi$ for all t.

In practice, the magnitude of rotational differences encountered in the filtering process is small, especially when dealing with high-frequency sampling and smooth motion trajectories. Therefore, the norm $\|\mathbf{p}(t - t_k; \boldsymbol{\rho}_k)\| < \pi$ holds in most cases. However, to rigorously ensure the smoothness and differentiability of $\varphi(t)$, we impose a constraint that $\|\mathbf{p}(t - t_k; \boldsymbol{\rho}_k)\| < \pi$ for all t within the filter window as the exponential map is injective on this domain. This can be achieved by appropriately selecting the filter window size and ensuring the sampling rate is sufficiently high with respect to the angular velocity.

Next, we consider smoothness of the derivatives $\frac{d}{dt}\varphi(t)$ and $\frac{d^2}{dt^2}\varphi(t)$ of $\varphi(t)$. We observe that they coincide with the angular velocity and acceleration, which are defined by [7] as:

$$\begin{aligned} \widehat{\boldsymbol{\omega}}(t) &= D\mathrm{Exp}\left(\boldsymbol{\varphi}(t)^\wedge\right)\dot{\boldsymbol{\varphi}}(t), \\ \widehat{\dot{\boldsymbol{\omega}}}(t) &= D^2\mathrm{Exp}\left(\boldsymbol{\varphi}(t)^\wedge\right)[\dot{\boldsymbol{\varphi}}(t), \dot{\boldsymbol{\varphi}}(t)] + \\ &\quad D\mathrm{Exp}\left(\boldsymbol{\varphi}(t)^\wedge\right)\ddot{\boldsymbol{\varphi}}(t) \end{aligned}$$

where $D^k f(\cdot)$ denotes the k-th order directional derivative with respect to f. These are, again, compositions of analytic functions and the polynomial derivatives, which are bounded and continuous and, therefore, continuously differentiable. We refer to [7] for a complete derivation of the derivatives.

Boundedness. To show the polynomial coefficients $\boldsymbol{\rho}$ are bounded, we observe that they are calculated via least squares with $\mathbf{A}$ and $\mathbf{b}$ defined as in Appendix C.3. We can see that $\hat{A}$ has full column rank as long as $2n+1 \geq p+1$. Therefore the rank of $\mathbf{A}$ can be obtained via: rank($\mathbf{A}$) = rank($\hat{A}$) $\cdot$ rank($\mathbf{I}_3$). Since both these matrices have full column rank, so does $\mathbf{A}$, and therefore, the matrix $\mathbf{A}^T\mathbf{A}$ is symmetric and positive definite. The inverse $(\mathbf{A}^T\mathbf{A})^{-1}$ therefore exists and is bounded.

The data vector $\mathbf{b}$ is constructed from the residuals $\boldsymbol{\delta}_m$, which are bounded by $\|\boldsymbol{\delta}_m\| \leq \theta_{\max} < \pi$. The entries of $\mathbf{A}^\top\mathbf{b}$ are sums of the form:

$$[\mathbf{A}^\top\mathbf{b}]_i = \sum_{m=k-n}^{k+n} \frac{(t_m - t_k)^i}{i!}\boldsymbol{\delta}_m,$$

which are bounded as $\boldsymbol{\delta}_k$, and the time intervals are bounded.

Combining the above, the polynomial coefficients $\boldsymbol{\rho}$ satisfy:

$$\|\boldsymbol{\rho}\| \leq \left\|(\mathbf{A}^\top\mathbf{A})^{-1}\right\| \left\|\mathbf{A}^\top\mathbf{b}\right\|$$

Finally, we note that as long as our learned weight matrix $\mathbf{W}$ is bounded, this also holds for the weighted adaptation. $\mathbf{W} \in \mathbb{R}^{3(2n-1)\times 3(2n-1)}$ is a (similarly expanded with $\otimes\mathbf{I}_3$) diagonal matrix with a weight coinciding to each point in the trajectory window. Then, the bound can be adapted to:

$$\|\boldsymbol{\rho}\| \leq \left\|(\mathbf{A}^\top\mathbf{W}\mathbf{A})^{-1}\right\| \left\|\mathbf{A}^\top\mathbf{W}\mathbf{b}\right\|$$

In practice, we find that we do not need to impose any constraints on $\mathbf{W}$ and that the network learns to construct the regression weights that are well-behaved.

Uniqueness. [10] additionally requires the control signal X to be unique. For regular sampling, this follows as cubic splines pass directly through the control points.

In our case, the polynomial $\mathbf{p}\,(t - t_k; \boldsymbol{\rho}_k)$ does not pass through the control points. However, we observe that because the matrix $\mathbf{A}$ has full column rank, the reduced optimization problem in Thm. (8) is strictly convex, and thus the least squares problem attains a unique solution. In the case of irregular sampling, however, this does not hold [10, B.2.1]. As a simple counterexample, consider that a point placed on the estimated trajectory $\varphi(t)$ would not alter the solution due to the strict convexity of the cost function in Thm (8). In practice, we find that the method also behaves well for irregularly sampled measurements. □

References

[1] Sameer Agarwal, Keir Mierle, and The Ceres Solver Team. Ceres Solver, 2023. 4

[2] Lennart Bastian, Yizheng Xie, Nassir Navab, and Zorah Lähner. Hybrid functional maps for crease-aware non-isometric shape matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3313–3323, 2024. 8

[3] Ricky T.Q. Chen. torchdiffeq, 2018. 6

[4] Maximilian Denninger, Dominik Winkelbauer, Martin Sundermeyer, Wout Boerdijk, Markus Knauer, Klaus H. Strobl, Matthias Humt, and Rudolph Triebel. Blenderproc2: A procedural pipeline for photorealistic rendering. *Journal of Open Source Software*, 8(82):4901, 2023. 2

[5] Sergio Garrido-Jurado, Rafael Muñoz-Salinas, Francisco José Madrid-Cuevas, and Manuel Jesús Marín-Jiménez. Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition*, 47(6):2280–2292, 2014. 3

[6] Andreas René Geist, Jonas Frey, Mikel Zhobro, Anna Levina, and Georg Martius. Learning with 3d rotations, a hitchhiker's guide to so (3). In *Int. Conf. Mach. Lear.*, 2024. 6

[7] Maarten Jongeneel and Alessandro Saccon. Geometric savitzky-golay filtering of noisy rotations on so (3) with simultaneous angular velocity and acceleration estimation. In *Int. Conf. Intell. Robot. and Syst.*, pages 2962–2968. IEEE, 2022. 1, 4, 8, 9

[8] Patrick Kidger, James Morrill, James Foster, and Terry Lyons. Neural controlled differential equations for irregular time series. 33:6696–6707, 2020. 4, 6

[9] Justice J Mason, Christine Allen-Blanchette, Nicholas Zolman, Elizabeth Davison, and Naomi Ehrich Leonard. Learning to predict 3d rotational dynamics from images of a rigid body with unknown mass distribution. *Aerospace*, 10(11):921, 2023. 4, 6, 7, 8

[10] James Morrill, Patrick Kidger, Lingyi Yang, and Terry Lyons. On the choice of interpolation scheme for neural cdes. *Trans. Mach. Lear. Resea.*, 2022(9), 2022. 4, 9, 10

[11] Mikael Persson, Gustav Häger, Hannes Ovrén, and Per-Erik Forssén. Practical pose trajectory splines with explicit regularization. In *Int. Conf. 3D Vision*, pages 156–165. IEEE, 2021. 4

[12] Ramona-Andreea Rohan. Some remarks on the exponential map on the groups so(n) and se(n). In *International Conference on Geometry, Integrability and Quantization*, pages 160–175. Avangard Prima, 2013. 9

[13] Ken Shoemake. Uniform random rotations. In *Graphics Gems III (IBM Version)*, pages 124–132. Elsevier, 1992. 5

[14] Christiane Sommer, Vladyslav Usenko, David Schubert, Nikolaus Demmel, and Daniel Cremers. Efficient derivative computation for cumulative b-splines on lie groups. In *CVPR*, 2020. 4

[15] Hauke Strasdat. Sophus: C++ implementation of lie groups using eigen. https://github.com/strasdat/Sophus, 2011. 4

[16] Vladyslav Usenko, Nikolaus Demmel, and Daniel Cremers. The double sphere camera model. In *2018 International Conference on 3D Vision (3DV)*, pages 552–560. IEEE, 2018. 4

[17] Gu Wang, Fabian Manhardt, Federico Tombari, and Xiangyang Ji. Gdr-net: Geometry-guided direct regression network for monocular 6d object pose estimation. In *CVPR*, pages 16611–16621, 2021. 2

[18] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. In *Robotics: Science and Systems (RSS)*. Robotics: Science and Systems Foundation, 2018. 2

[19] Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. On the continuity of rotation representations in neural networks. In *CVPR*, pages 5745–5753, 2019. 6