Geometric Neural Distance Fields for Learning Human Motion Priors

Supplementary Material

A. Geometry of Articulated Motions

For the sake of a self-contained manuscript, we now provide a complete treatment of the Riemannian geometry of rotations that we use in this work. For completeness, we also recall the definitions we provide in the main paper.

Riemannian manifolds. Following [2, 3, 5], we define an m-dimensional *Riemannian manifold*, embedded in an ambient Euclidean space $\mathcal{X} = \mathbb{R}^d$ and endowed with a *Riemannian metric* $\mathbf{G} \triangleq (\mathbf{G}_\mathbf{x})_{\mathbf{x}\in\mathcal{M}}$ to be a smooth curved space $(\mathcal{M}, \mathbf{G})$. A vector $\mathbf{v} \in \mathcal{X}$ is said to be *tangent* to $\mathcal{M}$ at $\mathbf{x}$ *iff* there exists a smooth curve $\gamma : [0, 1] \to \mathcal{M}$ s.t. $\gamma(0) = \mathbf{x}$ and $\dot\gamma(0) = \mathbf{v}$. The velocities of all such curves through $\mathbf{x}$ form the *tangent space* $\mathcal{T}_\mathbf{x}\mathcal{M} = \{\dot\gamma(0) \,|\, \gamma : \mathbb{R} \to \mathcal{M} \text{ is smooth around } 0 \text{ and } \gamma(0) = \mathbf{x}\}$. We will also work with a product of K manifolds, $\mathcal{M}_{1:K} := \mathcal{M}_1 \times \mathcal{M}_2 \times \cdots \times \mathcal{M}_K$, For identical manifolds, *i.e.* $\mathcal{M}_i \equiv \mathcal{M}_j$, we recover the *power manifold*, $\mathcal{M}^K := \mathcal{M}_{1:K}$. In this work, we will specifically work on the differentiable manifold of a product of K rotations, $\mathrm{SO}(3)^K$.

Definition 1 ($\mathrm{SO}(3)$)**.** *The manifold of 3D rotations admit the structure:*

$$\mathrm{SO}(3) = \left\{\mathbf{R} \in \mathbb{R}^{3\times3} : \mathbf{R}^\top\mathbf{R} = \mathbf{I}, \det(\mathbf{R}) = 1\right\}. \quad (1)$$

To characterize this manifold, we start by deriving its tangent space:

Definition 2 (Tangent space of $\mathrm{SO}(3)$)**.** *The tangent space of* $\mathrm{SO}(3)$ *at* $\mathbf{R}$ *is defined throughout as the* left-invariant vector fields*:*

$$\mathcal{T}_\mathbf{R}\mathrm{SO}(3) = \{\mathbf{R}\boldsymbol{\Omega} \,:\, \boldsymbol{\Omega} \in \mathfrak{so}(3)\}, \quad (2)$$

where $\mathfrak{so}(3)$ *is given by the set of* skew-symmetric *tensors:*

$$\mathfrak{so}(3) = \left\{\boldsymbol{\Omega} \in \mathbb{R}^{3\times3} : \boldsymbol{\Omega}^\top = -\boldsymbol{\Omega}\right\}, \quad (3)$$

and $\boldsymbol{\Omega}\mathbf{h} = \boldsymbol{\omega} \times \mathbf{h}$ *for any* $\mathbf{h} \in \mathbb{R}^3$*.* $\boldsymbol{\omega}$ *is called the* axial *or* rotation vector.

Proof. Differentiating the constraints that specify $\mathrm{SO}(3)$ leads us to the definition of skew-symmetric matrices:

$$\frac{\mathrm{d}(\mathbf{R}^\top\mathbf{R})}{\mathrm{d}t} = \dot{\mathbf{R}}^\top\mathbf{R} + \mathbf{R}^\top\dot{\mathbf{R}} = 0 \implies \boldsymbol{\Omega} = -\boldsymbol{\Omega}^\top, \quad (4)$$

where $\boldsymbol{\Omega} = \mathbf{R}^\top\dot{\mathbf{R}}$ is a skew symmetric matrix and the vector $\dot{\mathbf{R}} = \mathbf{R}\boldsymbol{\Omega}$ is tangent to $\mathbf{R}$. □

To work with functions on this manifold, for example in optimization, we will further introduce the notions of differentials and gradients that makes intrinsic sense.

Definition 3 (Riemannian gradient for $\mathrm{SO}(3)$)**.** *The Riemannian (or intrinsic) gradient of a function defined on* $\mathrm{SO}(3)$ *is the unique tangent vector satisfying:*

$$\langle \mathrm{grad} f(\mathbf{R}), \xi\rangle_R = Df(\mathbf{R})[\boldsymbol{\xi}] = \langle \nabla f(\mathbf{R}), \boldsymbol{\xi}\rangle \quad (5)$$

for all $\boldsymbol{\xi} \in \mathcal{T}_\mathbf{R}\mathrm{SO}(3)$.

The intrinsic gradient is the only part of the gradient that makes sense to a local citizen. For $\mathrm{SO}(3)$, given the Euclidean gradient, it can be obtained via the $\mathrm{egrad2rgrad}$ operator:

Proposition 1 ($\mathrm{egrad2rgrad}$ for $\mathrm{SO}(3)$)**.** *The projection of the Euclidean gradient* $\nabla f(\mathbf{R})$ *onto the tangent space of* $\mathbf{R}$ *yields the Riemannian gradient:*

$$\begin{aligned}\mathrm{grad} f(\mathbf{R}) &\triangleq \Pi_\mathbf{R}\big(\nabla f(\mathbf{R})\big) && (6)\\ &= \mathbf{R}\,\mathrm{skew}\big(\mathbf{R}^\top\nabla f\,(\mathbf{R})\big) && (7)\\ &= \frac{1}{2}\mathbf{R}\left(\mathbf{R}^\top\nabla f(\mathbf{R}) - \nabla f(\mathbf{R})^\top\mathbf{R}\right), && (8)\end{aligned}$$

where $\Pi_\mathbf{R} : \mathcal{X} \to \mathcal{T}_\mathbf{R}\mathrm{SO}(3)$ *is an orthogonal projector onto the tangent space of* $\mathbf{R}$*. This is famously known as the* $\mathrm{egrad2rgrad}$ *in various packages including ManOpt [28].*

Proof. In a differential geometric setting, any vector can be written down as a linear combination of normal and tangential components. To project the Euclidean gradient onto the tangent space $\mathcal{T}_\mathbf{R}\mathrm{SO}(3) = \{\mathbf{R}\boldsymbol{\Omega} : \boldsymbol{\Omega} \in \mathfrak{so}(3)\}$, our aim will be to *kill* the normal component, leaving only the tangential one. Let us start by writing:

$$\nabla f\,(\mathbf{R}) = \mathbf{R}\left(\mathbf{R}^\top\nabla f\,(\mathbf{R})\right). \quad (9)$$

Since any square matrix can uniquely be written as the sum of a symmetric and a skew-symmetric matrix, we can write:

$$\nabla f\,(\mathbf{R}) = \mathbf{R}\,\mathrm{sym}\big(\mathbf{R}^\top\nabla f\,(\mathbf{R})\big) + \mathbf{R}\,\mathrm{skew}\big(\mathbf{R}^\top\nabla f\,(\mathbf{R})\big).$$

By definition of the tangent space, $\mathbf{R}\,\mathrm{skew}(\mathbf{v}) \in \mathcal{T}_\mathbf{R}\mathrm{SO}(3)$ for any $\mathbf{v} \in \mathbb{R}^3$. Also note that $\mathbf{R}\,\mathrm{sym}\big(\mathbf{R}^\top\nabla f\,(\mathbf{R})\big)$ lives in the *normal space*, *i.e.*, it is orthogonal to all tangent vectors with respect to the Frobenius inner product. To see this, let $\mathbf{S} = \mathrm{sym}\big(\mathbf{R}^\top\nabla f\,(\mathbf{R})\big)$ and write:

$$\begin{aligned}\langle\mathbf{R}\mathbf{S}, \mathbf{R}\boldsymbol{\Omega}\rangle &= \mathrm{Tr}\left((\mathbf{R}\mathbf{S})^\top(\mathbf{R}\boldsymbol{\Omega})\right) && (10)\\ &= \mathrm{Tr}\left(\mathbf{S}^\top\mathbf{R}^\top\mathbf{R}\boldsymbol{\Omega}\right) && (11)\\ &= \mathrm{Tr}\left(\mathbf{S}^\top\boldsymbol{\Omega}\right) && (12)\\ &= 0. && (13)\end{aligned}$$

Note once again that $\mathbf{R}\boldsymbol{\Omega} \in \mathcal{T}_{\mathbf{R}}\mathrm{SO}(3)$ by definition. To see the last equality, consider

$$\mathrm{Tr}(\mathbf{S}\boldsymbol{\Omega}) = \mathrm{Tr}\left((\mathbf{S}\boldsymbol{\Omega})^\top\right) = \mathrm{Tr}((-\boldsymbol{\Omega})\mathbf{S}) = -\mathrm{Tr}(\mathbf{S}\boldsymbol{\Omega}).$$

Observe that $\mathrm{Tr}(\mathbf{S}\boldsymbol{\Omega}) = -\mathrm{Tr}(\mathbf{S}\boldsymbol{\Omega}) \implies \mathrm{Tr}(\mathbf{S}\boldsymbol{\Omega}) = 0$. Therefore, because $\mathbf{R}\,\mathrm{sym}\left(\mathbf{R}^\top \nabla f\left(\mathbf{R}\right)\right)$ is normal, it vanishes under projection onto the tangent space and we are left with the skew component, $\mathbf{R}\,\mathrm{skew}\left(\mathbf{R}^\top \nabla f\left(\mathbf{R}\right)\right)$, leading to the following orthogonal projection:

$$\begin{aligned}\mathrm{grad} f(\mathbf{R}) &\triangleq \Pi_{\mathbf{R}}\left(\nabla f(\mathbf{R})\right) && (14)\\ &= \mathbf{R}\,\mathrm{skew}\left(\mathbf{R}^\top \nabla f\left(\mathbf{R}\right)\right). && (15)\end{aligned}$$

This concludes the proof. □

Finally, we will introduce the exponential and logarithmic maps that are required to *walk* on $\mathrm{SO}(3)$. Most of the proofs, which we will omit for brevity, are based on the properties of the *matrix exponential*.

Definition 4 (Exponential Map ((Exp)))**.** *Let $\boldsymbol{\xi} \in \mathcal{T}_{\mathbf{R}}\mathrm{SO}(3) = \mathbf{R}\boldsymbol{\Omega}$ denote any tangent vector with $\boldsymbol{\Omega} = [\boldsymbol{\omega}]_\times$. The exponential map at $\mathbf{R}$ is defined as a map $\mathrm{Exp} : \mathrm{SO}(3) \times \mathcal{T}_{\mathbf{R}}\mathrm{SO}(3) \to \mathrm{SO}(3)$:*

$$\begin{aligned}\mathrm{Exp}_{\mathbf{R}}(\boldsymbol{\Omega}) &= \mathbf{R}\mathrm{Exp}(\boldsymbol{\Omega}) && (16)\\ &= \mathbf{R}\left(\mathbf{I} + \frac{\sin(\theta)}{\theta}\boldsymbol{\Omega} + \frac{1-\cos(\theta)}{\theta}\boldsymbol{\Omega}^2\right), && (17)\end{aligned}$$

*where $\theta = \|\omega\|$ and $\mathrm{Exp}(\boldsymbol{\Omega})$ (right hand side) is known as the celebrated **Rodrigues' formula**.*

Definition 5 (Logarithmic Map ((Log)))**.** *The inverse of Exp is given by the logarithmic map $\mathrm{Log} : \mathrm{SO}(3) \times \mathrm{SO}(3) \to \mathcal{T}_{\mathbf{R}}\mathrm{SO}(3)$, which expresses any rotation $\mathbf{R}_1$ near $\mathbf{R}$ in the tangent space $\mathcal{T}_{\mathbf{R}}\mathrm{SO}(3)$:*

$$\begin{aligned}\boldsymbol{\Omega} = \mathrm{Log}_{\mathbf{R}}(\mathbf{R}_1) &= \mathrm{Log}(\mathbf{R}^\top\mathbf{R}_1) && (18)\\ &= \begin{cases} \frac{\theta}{2\sin(\theta)}\left(\mathbf{R}^\top\mathbf{R}_1 - \mathbf{R}_1^\top\mathbf{R}\right), & \theta \in (\pi,\pi)\setminus\{0\} \\ \frac{1}{2}\left(\mathbf{R}^\top\mathbf{R}_1 - \mathbf{R}_1^\top\mathbf{R}\right), & \theta = 0, \end{cases}\end{aligned}$$

where

$$\theta := d_{\mathrm{SO}(3)}(\mathbf{R},\mathbf{R}_1) = \cos^{-1}\left(\frac{\mathrm{Tr}\left(\mathbf{R}^\top\mathbf{R}_1\right) - 1}{2}\right), \quad (19)$$

*is known as the **geodesic distance** between two rotations.*

Our approach benefits from the dynamics on $\mathrm{SO}(3)$ through the angular velocity and acceleration, which are the backbone of geometric mechanics [12, 13, 17]. We will derive the basic notions below and refer the reader these references for a broader exposition.

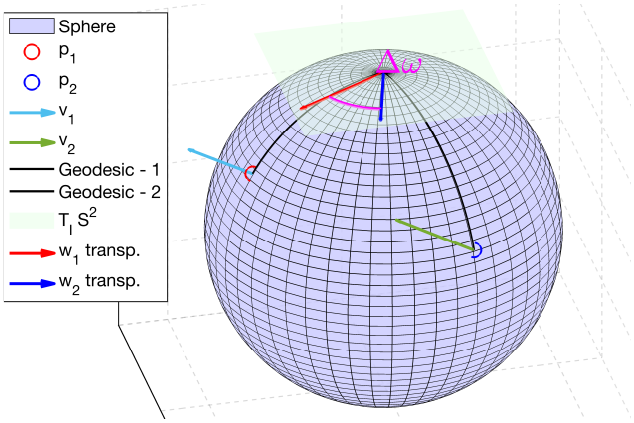

Figure 1. Illustration of angular velocity and accelaration for two points on a manifold. Given two points on the manifold ($\mathbf{p}_i$), the velocities are vectors in the tangent planes at those points. In practice, to compute an angular acceleration, they are *parallel transported* to the tangent space of the identity and then compared. If the points are close enough, the variation of the tangent spaces are ignored, and the computation is carried out by directly comparing vectors. *Sphere is chosen for illustration purposes whereas the manifold or articulated bodies is higher dimensional.*

Definition 6 (Angular velocity)**.** *While any tangent vector can be interpreted as representing an* angular velocity[1]*, it is useful to represent the velocity of the geodesic motion on $\mathrm{SO}(3)$ as a time (t) dependent curve $R(t)$:*

$$\Omega(t) = R^\top(t)\dot{R}(t) \in \mathfrak{so}(3), \quad (20)$$

and the corresponding angular velocity matrix $\boldsymbol{\Omega} := \Omega(t)$ *is skew-symmetric. Since $\mathfrak{so}(3)$ is isomorphic to $\mathbb{R}^3$, we instead store the **angular velocity vector**, $\boldsymbol{\omega} \in \mathbb{R}^3$, corresponding to the skew–symmetric $\boldsymbol{\Omega}$ via the **hat-operator**: $\widehat{\boldsymbol{\omega}} = \boldsymbol{\Omega}$ or $\widehat{\omega(t)} = \Omega(t)$ and equivalently $[\boldsymbol{\omega}]_x = \boldsymbol{\Omega}$.*

Proposition 2 (Angular acceleration)**.** *The time-derivative of angular velocity defines the angular acceleration:*

$$\frac{\mathrm{d}[\boldsymbol{\omega}_t]_x}{\mathrm{d}t} := [\dot{\boldsymbol{\omega}}_t]_x := \ddot{\mathbf{R}}_t = \mathbf{R}_t\left(\boldsymbol{\Omega}_t^2 + \dot{\boldsymbol{\Omega}}_t\right), \quad (21)$$

where $\boldsymbol{\Omega}_t^2 + \dot{\boldsymbol{\Omega}}_t$ is also skew-symmetric. The additional $\boldsymbol{\Omega}^2$ term accounts for the curvature of the configuration space when transforming to the inertial frame.

Proof. We start by differentiating the angular velocity in Eq. (20) to obtain:

$$\ddot{R}(t) = \dot{R}(t)\Omega(t) + R(t)\dot{\Omega}(t). \quad (22)$$

Substituting $\dot{R}(t) = R(t)\Omega(t)$ (by Eq. (20)), we obtain:

[1] with an appropriate choice of the body-fixed frame

$$\ddot{R}(t) = R(t)\Omega^2(t) + R(t)\dot{\Omega}(t) \tag{23}$$
$$= R(t)\left(\Omega^2(t) + \dot{\Omega}(t)\right). \tag{24}$$

Left multiplying this with $R(t)^\top$ yields:

$$R(t)^\top \ddot{R}(t) = \Omega^2(t) + \dot{\Omega}(t). \tag{25}$$

Since the square of any skew–symmetric matrix is symmetric, this is a natural decomposition of a matrix into its symmetric and skew-symmetric parts. Therefore to recover $\dot{\Omega}(t)$ as the angular acceleration, we take the skew:

$$\widehat{\dot{\boldsymbol{\omega}}(t)} = \dot{\Omega}(t) = \text{skew}\left(\Omega^2(t) + \dot{\Omega}(t)\right) \tag{26}$$
$$= \text{skew}\left(R(t)^\top \ddot{R}(t)\right), \tag{27}$$

where the last equality follows from Eq. (25). In matrix form, $[\dot{\boldsymbol{\omega}}]_x = \boldsymbol{\Omega}^2 + \dot{\boldsymbol{\Omega}} = \mathbf{R}^\top \ddot{\mathbf{R}}$ where $\ddot{\mathbf{R}}$ denotes the second derivative in *ambient space*. □

Remark 1 (On Angular Acceleration)**.** *In practice, we use a second-order finite differencing scheme for angular acceleration. While being more efficient than using the logarithmic maps to compare different velocities in the same tangent frame (through parallel transport as illustrated in Fig. 1), it might also be less accurate. Thus, we compare the two discretization schemes in Fig. 2, revealing that the two schemes are quite close in practice. This motivates us to leverage Euclidean central differencing, after estimating the angular velocities.*

Extension to power manifolds. As articulated bodies are represented as a collection of rotations, we consider the power manifold (product of identical manifolds):

Definition 7 (Geodesic distances of articulated bodies)**.** *Endowed with the L_p* product metric*, the geodesic distances of articulated bodies can be written as:*

$$d_{\text{SO}(3)^K}(\boldsymbol{\theta}, \boldsymbol{\theta}') \tag{28}$$
$$= \|d(\mathbf{R}_1, \mathbf{R}'_1), d(\mathbf{R}_2, \mathbf{R}'_2), \ldots, d(\mathbf{R}_K, \mathbf{R}'_K)\|_p,$$

where $\mathbf{R}_i \in \boldsymbol{\theta} \in \text{SO}(3)^K$ and $\mathbf{R}'_i \in \boldsymbol{\theta}' \in \text{SO}(3)^K$. In this work, we use $p = 1$ and $d \equiv d_{\text{SO}(3)}$.

Definition 8 (Exp / Log map on power manifold)**.** *The natural isomorphism allows us to write its exponential map $\text{Exp}_{\boldsymbol{\theta}} : \mathcal{T}_{\boldsymbol{\theta}}\text{SO}(3)^K \to \text{SO}(3)^K$ component-wise:*

$$\text{Exp}_{\boldsymbol{\theta}} = \left(\text{Exp}_{\mathbf{R}_1}, \text{Exp}_{\mathbf{R}_2}, \ldots, \text{Exp}_{\mathbf{R}_K}\right).$$

Akin to this, is the logarithmic map, $\text{Log}_{\boldsymbol{\theta}}$.

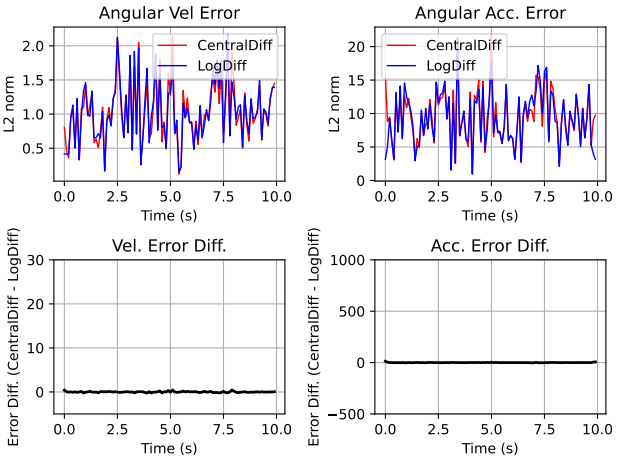

Figure 2. Comparing log-central differencing to classical central differencing for angular acceleration estimation. The attained error between the two (last row) is relatively low indicating that simple central differences are good estimators of angular acceleration, given geometrically meaningful angular velocities.

Definition 9 (Riemannian gradient for an articulated pose)**.** *Since the tangent spaces and therefore $\Pi_{\boldsymbol{\theta}}$ are replicas, the gradient of a smooth function $f : \text{SO}(3)^K \to \mathbb{R}$ w.r.t. $\boldsymbol{\theta}$ is also the Cartesian product of the individual gradients:*

$$\text{grad}_{\boldsymbol{\theta}} f(\boldsymbol{\theta}) = \left(\text{grad}_{\mathbf{R}_1} f(\boldsymbol{\theta}), \ldots, \text{grad}_{\mathbf{R}_K} f(\boldsymbol{\theta})\right). \tag{29}$$

B. Dataset details

In our experiments, we leverage the following datasets:

1. **AMASS** [16] is a large-scale human motion database comprising of high-quality and diverse motion captures. For training and evaluation, we subsample the dataset at a rate of 30 Hz and follow the split used in [11, 19, 27]. We extract transitions (velocities) and corresponding accelerations by sampling 80% of each motion sequence.
2. **3DPW** [29] is a popular benchmark for 3D human pose and shape estimation in challenging, real-world settings. Provides high-quality 3D pose annotations captured through synchronized IMUs and motion capture systems, along with RGB videos recorded in diverse outdoor environments.
3. **i3DB** [29] is an RGB dataset of human-scene interactions, providing 3D joint annotations along with 3D scene reconstructions.
4. **EgoBody** [30] is a large-scale dataset of 3D human motion captures during social interactions in 3D environments, with one subject wearing a head-mounted device recording egocentric multi-modal data, including eye gaze tracking, synchronized multi-view RGBD video, and full-body 3D motion reconstruction.
5. **PROX** [9] is an RGB-D dataset that provides 3D human pose and shape data in 3D real-world indoor environ-

Algorithm 1 RMF Integrator

Input: An initial pose $\boldsymbol{\theta}_0$ and velocity $\dot{\boldsymbol{\theta}}_0$, a possibly noisy acceleration sequence $\{\ddot{\boldsymbol{\theta}}_t\}_t$ and all components of the trained network f_Γ (and hence the projectors Π),

Output: Plausible motion $\{\mathbf{X}_t\}_t$

for $t \leftarrow 1, \ldots, T$ **do** ▷ Euler integrate of velocities
  $\dot{\boldsymbol{\theta}}_t \leftarrow \dot{\boldsymbol{\theta}}_{t-1} + \lambda_t \ddot{\boldsymbol{\theta}}_{t-1}$
  $\dot{\boldsymbol{\theta}}_t \leftarrow \Pi^{\omega}(\dot{\boldsymbol{\theta}}_t)$
end for
for $t \leftarrow 1, \ldots, T$ **do** ▷ Update poses
  $\boldsymbol{\theta}_t \leftarrow \mathrm{Exp}_{\boldsymbol{\theta}_{t-1}}(\alpha_t\, [\dot{\boldsymbol{\theta}}_{t-1}]_x)$
  $\boldsymbol{\theta}_t \leftarrow \Pi^{R}(\boldsymbol{\theta}_t)$
end for
Compose $\{\mathbf{X}_t\}_t$ from individual estimates $\{\ddot{\boldsymbol{\theta}}_t, \dot{\boldsymbol{\theta}}_t, \boldsymbol{\theta}_t\}_t$

ments with interactions between humans and surroundings. We pre-process them to obtain the observations in the same way as i3DB.

C. Algorithms, Architecture & Details

Pseudocode of RMF-Integrator. We provide the pseudocode for integration of motion via porjection onto the learned manifold in Alg. 1.

Parametric model. We employ the SMPL+H body model [22], consistent with the AMASS dataset [16] on which our work is based. Since our focus is exclusively on modeling body motion, we follow HuMoR [21, 24, 31] tp exclude hand joints from consideration, retaining only the 22 body joints (including the root). Incorporating hand joints into the optimization alongside body motion is straightforward, which lies outside the scope of our present study but can be a future work.

Test-time optimization. We implement our optimizer and optimization pipeline with Pytorch [18]. We provide full details of our HuMoR [21]-inspired test-time-optimization in this section. During runtime, given a sequence of 2D or 3D data observations, *e.g.* joint detections, point clouds, described as $\mathbf{O}_{0:T}$, our goal is to recover the motion sequence $\{\mathbf{X}_t\}_{t=0}^{T}$ that aligns with the observations while being *likely* under the NRFM. Our mutlti-stage optimization starts from a proper initialization using the pose prior, by minimizing the following combined objective:

$$E_I(\mathbf{t}_\mathrm{r}, \boldsymbol{\theta}, \boldsymbol{\beta}) = \mathcal{L}_\mathrm{data} + \lambda_{\boldsymbol{\beta}}\mathcal{L}_{\boldsymbol{\beta}} + \lambda_{\boldsymbol{\theta}}\mathcal{L}_{\boldsymbol{\theta}} + \lambda_{reg}\mathcal{L}_\mathrm{reg} \quad (30)$$

where $\mathcal{L}_{\boldsymbol{\beta}} = \|\boldsymbol{\beta}\|^2$ is the shape prior term. $\mathcal{L}_{\boldsymbol{\theta}} = f_\Phi^R(\boldsymbol{\theta}_i)$ is our pose prior term predicting the distance to the nearest plausible pose[2]. Moreover, we apply additional regularization terms where $\mathcal{L}_\mathrm{reg} = \lambda_\mathrm{smooth}\mathcal{L}_\mathrm{smooth} + \lambda_\mathrm{bl}\mathcal{L}_\mathrm{bl}$:

$$\mathcal{L}_\mathrm{smooth} = \sum_{t=0}^{T} \|\boldsymbol{J}_{t+1} - \boldsymbol{J}_t\|^2, \quad \mathcal{L}_\mathrm{bl} = \sum_{t=0}^{T} \|\mathbf{B}_{t+1} - \mathbf{B}_t\|^2$$

where $\mathbf{B}$ are the bone length of the human body. These regularization terms essentially enforces the skeleton to be consistent over time and prevent large jitters. Finally, $\mathcal{L}_\mathrm{data}$ is the task-dependent data terms and will be different when given different types of inputs. Specifically, in the task of fitting to 2D joints the objective function is written as:

$$\mathcal{L}_\mathrm{data}^\mathrm{2D} = \lambda_\mathrm{data} \sum_{t=0}^{T}\sum_{j=1}^{J} \sigma_t^j \rho(\Pi(\boldsymbol{J}_t^j) - \mathbf{O}_t^j) \quad (13)$$

where $\mathbf{O}_t^j$ is the 2D joint observations with σ^j as the confidence score for 2D joint observations. ρ is the robust Geman-McClure function [4, 7] and Π is the 2D reprojection with the 3D joints $\boldsymbol{J}$ extracted from human meshes. When fitting 3D joints, the objective becomes:

$$\mathcal{L}_\mathrm{data}^\mathrm{3D} = \lambda_\mathrm{data} \sum_{t=0}^{T}\sum_{j=1}^{J} \|\boldsymbol{J}_t^j - \mathbf{O}_t^j\|^2 \quad (12)$$

where the $\mathbf{O}_t^j$ is the 3D joint observations. $\mathcal{L}_\mathrm{data}$ can be further extended to various observation representations such as 3D point clouds:

$$\mathcal{L}_\mathrm{data}^\mathrm{PC3D} = \lambda_\mathrm{data} \sum_{t=0}^{T}\sum_{i=1}^{N_t} w_\mathrm{bs} \min_{\boldsymbol{V}_t \in V_t} \left\|\boldsymbol{V}_t - \mathbf{O}_t^i\right\|^2. \quad (31)$$

where $\mathbf{O}$ now represents 3D human point cloud observations extracted from depth images with person segmentation masks obtained from [6] and w_bs a bisquare weight [1] based on the Chamfer distance.

While optimizing E_I yields refined estimates $\mathbf{X}$, it cannot combat motion ambiguities since the pose prior only captures 0^{th}-order distribution. After initialization, we factor in our transition and acceleration priors as $\mathcal{L}_{\dot{\boldsymbol{\theta}}}$ and $\mathcal{L}_{\ddot{\boldsymbol{\theta}}}$:

$$\mathcal{L}_{\dot{\boldsymbol{\theta}}} := f_\Psi^\omega(\dot{\boldsymbol{\theta}}_i) \quad \mathcal{L}_{\ddot{\boldsymbol{\theta}}} := f_\Xi^{\dot{\omega}}(\ddot{\boldsymbol{\theta}}_i) \quad (32)$$

Disentangle and model global motion representation. For our motion state representation $\mathbf{X}$, we explicitly disentangle global motion (relative to the current state) $\mathbf{t}_\mathrm{r} \in \mathbb{R}^3$ from the local pose configuration for each state. Specifically, we achieve this by multiplying the inverse root transform with each pose, mapping all poses into a local coordinate frame aligned to a canonical facing direction. This canonicalization simplifies the learning process by removing unnecessary global variation. Inspired by prior works [10, 21], we predict $\{\mathbf{t}_\mathrm{r}\,, c_t^i\}$, through the sub-module network integrated within the transition prior framework based

[2] Note that for VPoser [20] $\mathcal{L}_{\boldsymbol{\theta}} = \|\mathbf{z}\|_2^2$ where z is the latent code.

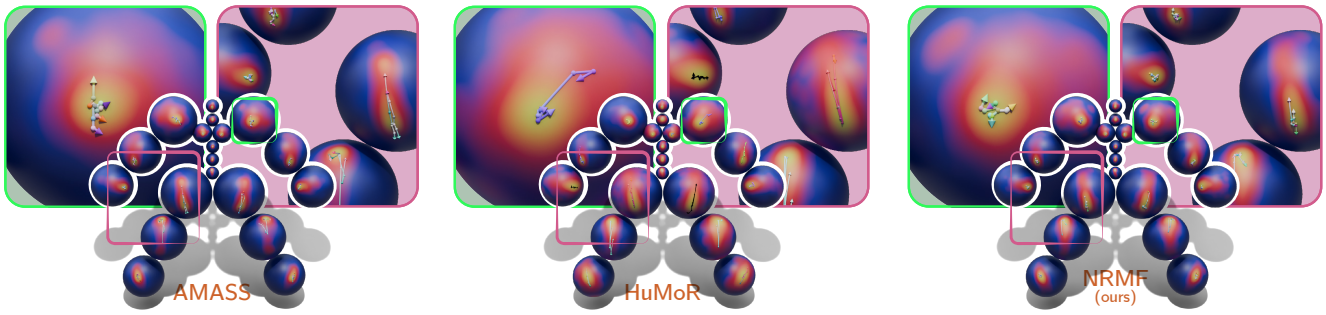

Figure 3. **Transitions and accelerations overlaid onto the pose distributions.** Four motions originating from the most common body pose in the AMASS dataset are represented as colored arrows on the surface of the spheres depicting the per-joint pose distributions comprising $p(\boldsymbol{\theta})$. Spheres here are oriented to maximize visibility of the high-probability region of the distribution rather than showcasing the range of motions through skeletal alignment. Each transition arrow is split into different segments representing the transition between consecutive time-steps.The color of the arrows maps the three components of the angular acceleration to RGB space.

on the same rich contextual input following [21]. During optimization, it is constrained by data terms during the optimization. Additionally, we expand $\mathcal{L}_{\text{reg}}$ to account also for floor contact physics constraint terms (contact loss $\mathcal{L}_{\text{contact}}$ and contact height loss $\mathcal{L}_{\text{ch}}$):

$$\mathcal{L}_{\text{contact}} = \sum_{t=1}^{T} \sum_{i=1}^{N_{contact}} \lambda_{\text{cj}} c_t^i \left\| \mathbf{j}_t^i - \mathbf{j}_{t-1}^i \right\|^2 + \lambda_{\text{cv}} c_t^i \left\| \dot{\boldsymbol{\theta}}_t^i \right\|^2 \tag{33}$$

$$\mathcal{L}_{\text{ch}} = \sum_{t=1}^{T} \sum_{i=1}^{N_{contact}} \lambda_{\text{ch}} c_t^i \max(|\mathbf{j}_{z,t}^i| - \delta, 0), \tag{34}$$

where c_t^i is the contact probability predicted by the transition prior network specifically for joint i at timestep t, and the contact height loss will further constrain the z-component of $\mathbf{j}^i$. The updated objective function then becomes:

$$\begin{aligned} E_I'(\mathbf{t}_{\text{r}}, \boldsymbol{\theta}, \boldsymbol{\beta}) = \mathcal{L}_{\text{data}} + \lambda_{\boldsymbol{\beta}} \mathcal{L}_{\boldsymbol{\beta}} + \lambda_{\boldsymbol{\theta}} \mathcal{L}_{\boldsymbol{\theta}} \\ + \lambda_{\text{reg}} \mathcal{L}_{\text{reg}} + \lambda_{\dot{\boldsymbol{\theta}}} \mathcal{L}_{\dot{\boldsymbol{\theta}}} + \lambda_{\ddot{\boldsymbol{\theta}}} \mathcal{L}_{\ddot{\boldsymbol{\theta}}} \end{aligned} \tag{35}$$

Moreover, we apply the rollout in Alg. 1 within each iteration to update the motion state $\mathbf{X}$ for drift correction and rebuild the consistent state. During the test-time optimization of motion estimation, we balance the loss terms and prior with the following coefficients: $\lambda_{\text{2d}} = 1e^{-3}, \lambda_{\text{smooth}} = 10, \lambda_{\boldsymbol{\beta}} = 8e^{-2}, \lambda_{\boldsymbol{\theta}} = 8e^{-2}, \lambda_{\dot{\boldsymbol{\theta}}} = 1.0, \lambda_{\ddot{\boldsymbol{\theta}}} = 5e^{-2}$. The latter terms influence the trust in each prior.

Training data generation. We now provide more details of our training process, including the data preparation. For training dataset, we follow [11, 15, 27] to use the same split and clip the motion sequence to extract the middle 80% and downsample the dataset to 30 fps. We then sample the negative training samples with an adopted multi-step mechanism for querying the nearest neighbours using FAISS [14] to speed up the NN process. To accelerate nearest-neighbor (NN) search, we adopt a two-stage retrieval scheme similar to [11, 27]. In the first stage, we use FAISS [14] to retrieve k' coarse candidates for each noisy transition or acceleration. In the second stage, we refine these candidates by computing the distance and selecting the closest k neighbors. In our setup, we set $k' = 1000$ and $k = 1$. Unlike Pose-NDF, which averages the distances to the top 5 nearest neighbors, we determine the final match using only the single closest neighbor. To generate negative samples, we adopt a mixed sampling strategy with fixed ratios for three types of negatives: distance-targeted half-Gaussian distribution based perturbations (60%) following the scheme in [11], random-swap candidates (30%), and fully random candidates (10%). Specifically, for the first category, we perturb the ground-truth transition or acceleration by applying a small random rotation sampled from a zero-mean Gaussian distribution, thereby introducing controlled manifold noise. For the second category, we randomly select transitions from the clean motion dataset without ensuring semantic proximity to the query. For the third category, we generate fully random quaternions and normalize them to unit length. All negative candidate are concatenated and passed through a single FAISS-based nearest neighbor search to compute their distances to the training set.

Training details. In our experiments, we introduce our baselines HuMoR [21], RoHM [31], and MDM [26] which were trained on the same split as ours for a fair comparison. Specifically, the original MDM essentially requires clean motion because it would replace denoised joints with visible input joints at each timestep of denoising and was not trained on AMASS. To make it comparable on AMASS, we train MDM with the same split and with quaternion as the rotation representation instead of the original hml_{vec}, which is a high-dimensional composition vector of motion states. For PhaseMP [24], we use the official code for evaluation and training. Finally, our prior modules (T-NRDF and

A-NRDF) are pre-trained independently in the first stage and jointly optimized in the second stage with learning rate of 3e-5. For training the global motion prediction layers, we follow a simple scheme to train it jointly with the network in our experiments. We choose the checkpoints by selecting the best-performing model on validation set.

Details on joint distribution visualizations. Although poses are defined in the product manifold of $\mathrm{SO}(3)$ ($\boldsymbol{\theta} \in \mathrm{SO}(3)^{N_J-1}$), for the purpose of visualization, we separately analyze each joint angle space. We observe that joint rotations can be plotted onto the surface of a sphere by rotating skeletal bones in their rest position. Precisely, given a unit vector representing the direction of the axis between two consecutive joints in the kinematic tree, we rotate it using the Rodrigues' rotation formula and obtain a rotated unit vector. Repeating this procedure for a set of joint poses, we obtain a set of points on a sphere. We thus fit these points using a spherical kernel density estimation [8], which replaces the traditional Gaussian function used in kernel density estimation with the Von Mises-Fisher distribution. We repeat the same procedure for both $\boldsymbol{\theta}$ and $\dot{\boldsymbol{\theta}}$, as $\dot{\boldsymbol{\theta}}$ is also a relative angle determing the transition between two poses.

All spherical kernel density estimations are performed on a set of $10,000$ rotations using a fixed bandwidth $h = 0.08$. The log-probability density estimates computed at the vertices of each sphere are subject to a power normalization with exponent γ. For ease of visualization, we set $\gamma = 60$ for the poses of AMASS, $\gamma = 50$ for NRMF, and $\gamma = 20$ for NRDF, HuMoR, and all the transition densities.

Poses and transitions are randomly selected from AMASS, and generated with all the other methods. For NRMF, we select $10,000$ samples from 195 150-frames-long motion sequences generated starting from initial poses of AMASS motions.

In addition to the evaluation of the pose $p(\boldsymbol{\theta})$ and transition $p(\dot{\boldsymbol{\theta}})$ distributions, we also explicitly visualize a selected subsets of motion sequences and their full states $\mathbf{X}_t$. We choose the most common pose in AMASS, denoted by $\boldsymbol{\theta}_c$, as the starting point for all trajectories and methods. We then perform a k-nearest neighbor (kNN) search using $d^{N_J}_{\mathrm{SO}(3)}(\boldsymbol{\theta}_c, \boldsymbol{\theta}_i)$ to geodesically measure distances and correctly identify the 150 closest poses to $\boldsymbol{\theta}_c$. These poses can occur at any timestep of a motion sequence as long as there are enough timesteps to complete a 20 frames motion. To promote diversity among the motions to represent, we also perform a $d^{N_J}_{\mathrm{SO}(3)}$-based farthest point sampling with respect to $\boldsymbol{\theta}_c$ and all the poses at $t+20$ extracted from the motion sequences identified with kNN. This allows us to identify the 4 trajectories of 20 timesteps originating as close as possible to $\boldsymbol{\theta}_c$ and terminating the farthest apart from each other –and their origin.

The trajectories are geodesically traced onto the spheres using FlipOut, the edge flipping algorithm introduced in [23] to find shortest geodesics on polyhedral surfaces. Note that we rely on a discrete method for geodesic computations because the spheres are effectively discretized into ico-spheres for visualization purposes. We break the trajectory in 3 segments each containing the same number of timesteps. Therefore, longer segments equate to faster transitions. Accelerations are finally represented mapping the three components of $\ddot{\boldsymbol{\theta}}$ to the RGB space.

As it can be observed in Fig. 7 of main paper, HuMoR tends to overestimate the range of motion of each joint, often enabling poses which are never or rarely represented in the training data. NRDF significantly mitigates this issue, closely mimicking the behavior of AMASS on most joints. Nevertheless, NRDF appears the best at capturing the true pose distribution. A similar trend is observable for transitions, which not only appear slightly improved in terms of distributions in Fig. 7 of main paper, but also in terms of individual examples reported in Fig. 3. In fact, the arrow orientations of NRMF more closely align with those of AMASS. Most notably, when comparing accelerations, we note that while NRMF can properly capture their variability, HuMoR seems to erroneously produce transitions with almost-fixed angular accelerations. These results prove once again the superiority of our method compared to the state of the art as well as the necessity of adopting a motion model.

D. Additional Evaluations

As we can observed from Fig. 7, our method outperforms the previous SotA significantly. In this section we provide more qualitative results for our motion estimation. As shown in Fig. 5 and Fig. 6, we present additional qualitative comparisons on motion estimation and in-betweening. Our method consistently outperforms existing approaches such as HuMoR [21], RoHM [31], and PhaseMP [24], producing more realistic and physically plausible motions. In scenarios with partial or noisy observations, our predictions exhibit improved temporal consistency and anatomically accurate transitions, especially in challenging regions like occluded body parts. For motion in-betweening and motion generation, NRMF generates smooth and coherent transitions between sparse keyframes, capturing natural dynamics and avoiding artifacts or discontinuities often seen in baseline methods. These results further highlight the benefit of incorporating

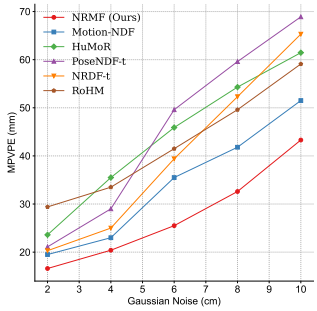

Figure 8. Behavior under increasing Gaussian noise.

Figure 4. Qualitative results on in-the-wild fitting on PROX [9] dataset. We compare our method agains [21, 24, 31], which demonstrates the robustness of our method under challenging occlusion.

higher-order priors and respecting the underlying geometry of articulated motion. Moreover, Fig. 8 demonstrates the effect of gradual (Gaussian) noising on MPVPE, where our method consistently performs better and behaves more stable under increased noise.

Motion In-betweening. To further evaluate NRMF's performance in reconstructing plausible motion under sparse supervision, we follow the sparse keyframe in-betweening protocol introduced in [10]. Specifically, we provide a small subset of keyframes as observations, either the two endpoints or 20%, 35%, and 50% of the total frames, and evaluate the model's ability to recover the full motion sequence. As shown in Table 1, NRMF consistently improves the baseline methods and achieves lower FID scores, indicating higher visual fidelity of the generated motions. In addition, we report the Acc Err as a measure of temporal smoothness and physical realism. Our method outperforms both NeMF across all sparsity levels, achieving the

Table 1. **Sparse keyframe in-betweening**. We report FID_m and Acc Err for different sequence lengths. The best and second-best results are shown in **bold** and underlined, respectively.

Method	**FID**$_m$ ↓				**Acc Err** (m/s^2) ↓			
	End points	10%	20%	30%	End points	10%	20%	30%
SLERP [25]	3.418	0.321	0.031	0.024	19.357	17.226	9.207	5.371
HuMoR [21]	2.572	0.695	0.106	0.078	13.356	11.579	6.601	3.378
NeMF [10]	2.044	0.315	0.081	0.063	12.864	10.194	6.535	3.563
Ours (SLERP)	2.238	0.308	**0.029**	**0.019**	10.767	6.478	3.768	2.319
Ours (NeMF)	**1.891**	**0.301**	0.045	0.032	**7.135**	**5.245**	**3.394**	**1.983**

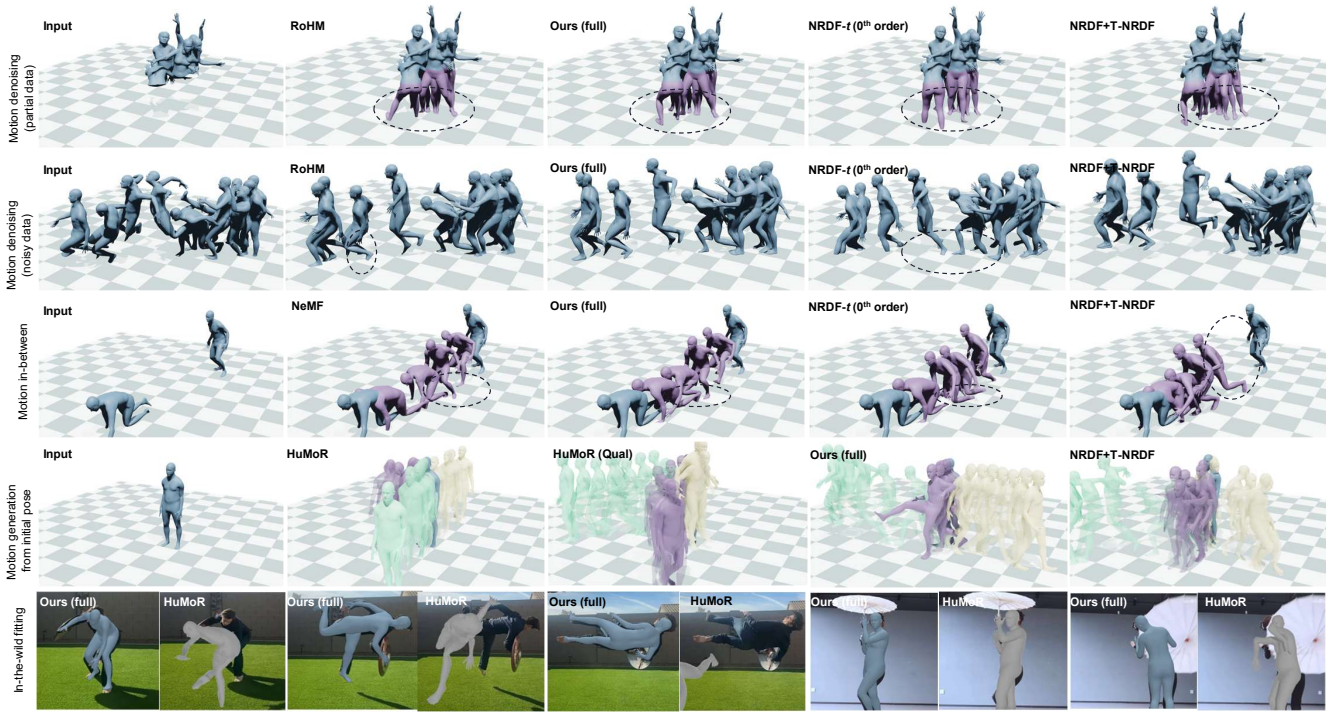

Figure 5. Qualitative results on downstream applications. The **mesh and body parts** are the observation inputs and we show the **output or optimization results**. For in-the-wild fitting, we conduct experiments on both in-door datasets (bottom row) and online videos (upper row).

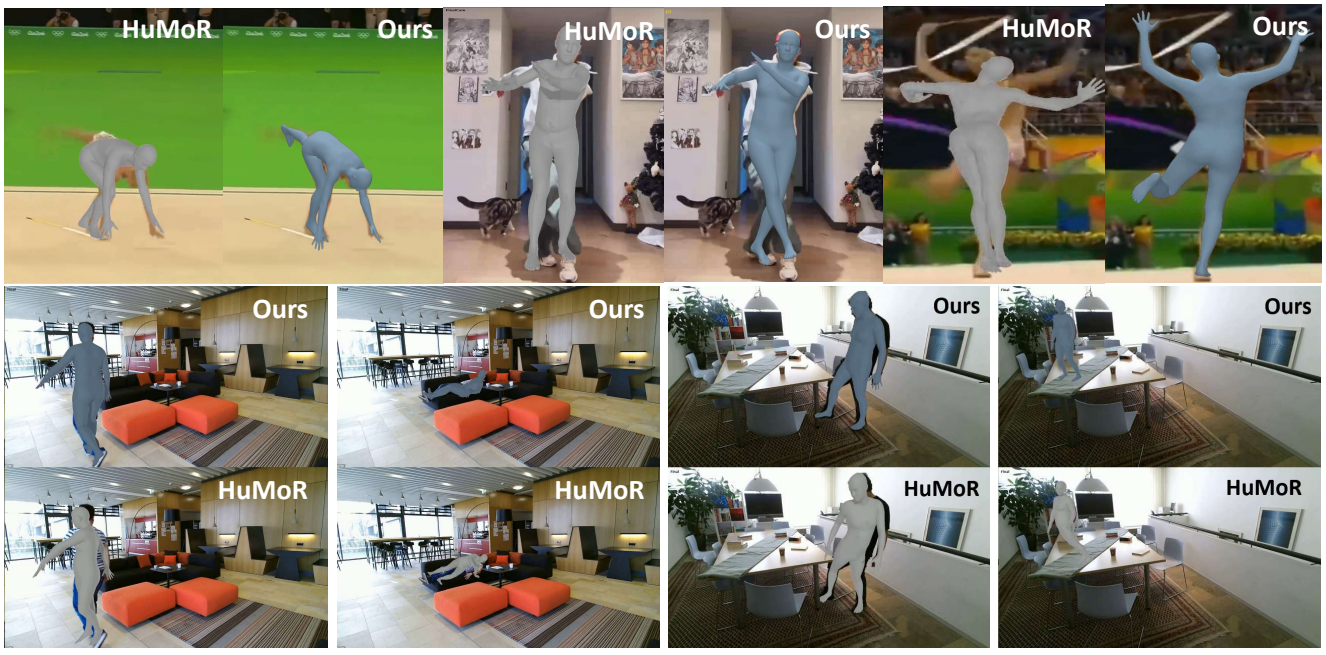

Figure 6. In-the-wild motion estimation. We show qualitative comparisons on real-world videos with challenging conditions such as occlusion, background clutter. Our method recovers more plausible, consistent, and anatomically accurate motions.

best trade-off between accuracy and dynamical consistency. These results highlight NRMF's ability to robustly interpolate missing frames by leveraging high-order dynamics.

Computational Cost. Excluding the time to obtain initial observations, our optimizer will take approximately 7min. to process a video of 10sec. at 30 fps on an NVIDIA A100 GPU. We also report the processing time for a 30 fps 10-

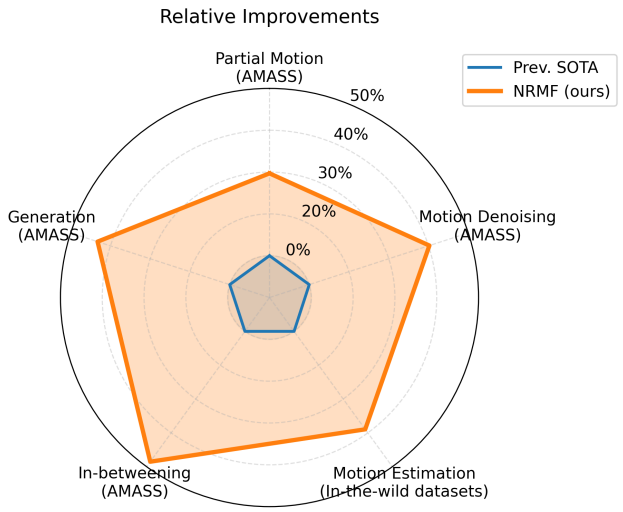

Figure 7. Improvements on different benchmarks.

second RGB video. As seen, our NRMF reduces the turnaround for 10 times, which is an order of magnitude lighter than the widely used HuMoR.

Table 2. **Runtime for a 10-second RGB video (minutes).**

	HuMoR	Stage I	Stage II	Stage III	**Total (Ours)**
Runtime (min.)	61.56	0.62	2.18	3.23	6.03

Evaluating A-NRDF. Tab. 3 includes an explicit comparison with A-NRDF, which consistently improves over pose-only NRDF, confirming the complementary benefits of acceleration. Yet, A-NRDF alone remains inferior to the full NRMF across all evaluated tasks: jointly constraining all components of the dynamics, integrating pose, velocity, and acceleration, is essential for accurate motion modeling.

Table 3. **Evaluation of A-NRDF on motion estimation.**

Experiment	**Method**	**All** (↓)	**Legs** (↓)	**Acc. Err**(↓)
Motion estimation (noisy)	NRDF + A-NRDF	17.9	19.3	2.34
	NRDF + T-NRDF	16.7	17.5	2.97
	NRMF (full)	**16.4**	**17.1**	**2.25**
Motion denoising (partial)	NRDF + A-NRDF	54.7	95.2	2.38
	NRDF + T-NRDF	48.5	84.8	3.01
	NRMF (full)	**46.3**	**81.6**	**2.31**

Failure cases. Sec. D below shows failure cases across some out-of-distribution samples and ambiguous observations. We included more examples in the revision and thank the reviewer for bringing up this important point.

Figure 9. Failure case of NRMF.

References

[1] Albert E Beaton and John W Tukey. The fitting of power series, meaning polynomials, illustrated on band-spectroscopic data. *Technometrics*, 16(2):147–185, 1974. 4

[2] Tolga Birdal and Umut Simsekli. Probabilistic permutation synchronization using the riemannian structure of the birkhoff polytope. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11105–11116, 2019. 1

[3] Tolga Birdal, Umut Simsekli, Mustafa Onur Eken, and Slobodan Ilic. Bayesian pose graph optimization via bingham distributions and tempered geodesic mcmc. *Advances in Neural Information Processing Systems*, 31, 2018. 1

[4] Federica Bogo, Angjoo Kanazawa, Christoph Lassner, Peter Gehler, Javier Romero, and Michael J Black. Keep it smpl: Automatic estimation of 3d human pose and shape from a single image. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part V 14*, pages 561–578. Springer, 2016. 4

[5] Jiayi Chen, Yingda Yin, Tolga Birdal, Baoquan Chen, Leonidas J Guibas, and He Wang. Projective manifold gradient layer for deep rotation regression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6646–6655, 2022. 1

[6] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017. 4

[7] Stuart Geman. Statistical methods for tomographic image restoration. *Bull. Internat. Statist. Inst.*, 52:5–21, 1987. 4

[8] Will Handley. williamjameshandley/spherical_kde 0.1.1, 2020. 6

[9] Mohamed Hassan, Vasileios Choutas, Dimitrios Tzionas, and Michael J. Black. Resolving 3D human pose ambiguities with 3D scene constraints. In *International Conference on Computer Vision*, pages 2282–2292, 2019. 3, 7

[10] Chengan He, Jun Saito, James Zachary, Holly Rushmeier, and Yi Zhou. Nemf: Neural motion fields for kinematic animation. *Advances in Neural Information Processing Systems*, 35:4244–4256, 2022. 4, 7

[11] Yannan He, Garvita Tiwari, Tolga Birdal, Jan Eric Lenssen, and Gerard Pons-Moll. Nrdf: Neural riemannian distance fields for learning articulated pose priors. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. 3, 5

[12] Darryl D Holm. *Geometric mechanics-Part I: Dynamics and symmetry*. World Scientific Publishing Company, 2011. 2

[13] Darryl D Holm. *Geometric mechanics-part II: rotating, translating and rolling*. World Scientific, 2011. 2

[14] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*, 7(3):535–547, 2019. 5

[15] Junzhe Lu, Jing Lin, Hongkun Dou, Yulun Zhang, Yue Deng, and Haoqian Wang. Dposer: Diffusion model as robust 3d human pose prior. *arXiv preprint arXiv:2312.05541*, 2023. 5

[16] Naureen Mahmood, Nima Ghorbani, Nikolaus F. Troje, Gerard Pons-Moll, and Michael J. Black. AMASS: Archive of motion capture as surface shapes. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5441–5450, 2019. 3, 4

[17] Jerrold E Marsden and Tudor S Ratiu. *Introduction to mechanics and symmetry: a basic exposition of classical mechanical systems*. Springer Science & Business Media, 2013. 2

[18] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019. 4

[19] Georgios Pavlakos, Vasileios Choutas, Nima Ghorbani, Timo Bolkart, Ahmed AA Osman, Dimitrios Tzionas, and Michael J Black. Expressive body capture: 3d hands, face, and body from a single image. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10975–10985, 2019. 3

[20] Georgios Pavlakos, Vasileios Choutas, Nima Ghorbani, Timo Bolkart, Ahmed A. A. Osman, Dimitrios Tzionas, and Michael J. Black. Expressive body capture: 3D hands, face, and body from a single image. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 10975–10985, 2019. 4

[21] Davis Rempe, Tolga Birdal, Aaron Hertzmann, Jimei Yang, Srinath Sridhar, and Leonidas J. Guibas. Humor: 3d human motion model for robust pose estimation. In *International Conference on Computer Vision (ICCV)*, 2021. 4, 5, 6, 7

[22] Javier Romero, Dimitrios Tzionas, and Michael J. Black. Embodied hands: Modeling and capturing hands and bodies together. *ACM Transactions on Graphics, (Proc. SIGGRAPH Asia)*, 36(6), 2017. 4

[23] Nicholas Sharp and Keenan Crane. You can find geodesic paths in triangle meshes by just flipping edges. *ACM Trans. Graph.*, 39(6), 2020. 6

[24] Mingyi Shi, Sebastian Starke, Yuting Ye, Taku Komura, and Jungdam Won. Phasemp: Robust 3d pose estimation via phase-conditioned human motion prior. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14725–14737, 2023. 4, 5, 6, 7

[25] Ken Shoemake. Animating rotation with quaternion curves. In *Proceedings of the 12th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, pages 245–254. ACM, 1985. 7

[26] Guy Tevet, Sigal Raab, Brian Gordon, Yoni Shafir, Daniel Cohen-or, and Amit Haim Bermano. Human motion diffusion model. In *The Eleventh International Conference on Learning Representations*, 2023. 5

[27] Garvita Tiwari, Dimitrije Antic, Jan Eric Lenssen, Nikolaos Sarafianos, Tony Tung, and Gerard Pons-Moll. Pose-ndf: Modeling human pose manifolds with neural distance fields. In *European Conference on Computer Vision (ECCV)*, 2022. 3, 5

[28] J. Townsend, N. Koep, and S. Weichwald. PyManopt: a Python toolbox for optimization on manifolds using automatic differentiation. *Journal of Machine Learning Research*, 17(137):1–5, 2016. 1

[29] Timo von Marcard, Roberto Henschel, Michael Black, Bodo Rosenhahn, and Gerard Pons-Moll. Recovering accurate 3d human pose in the wild using imus and a moving camera. In *European Conference on Computer Vision (ECCV)*, 2018. 3

[30] Siwei Zhang, Qianli Ma, Yan Zhang, Zhiyin Qian, Taein Kwon, Marc Pollefeys, Federica Bogo, and Siyu Tang. Egobody: Human body shape and motion of interacting people from head-mounted devices. In *European conference on computer vision*, pages 180–200. Springer, 2022. 3

[31] Siwei Zhang, Bharat Lal Bhatnagar, Yuanlu Xu, Alexander Winkler, Petr Kadlecek, Siyu Tang, and Federica Bogo. Rohm: Robust human motion reconstruction via diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14606–14617, 2024. 4, 5, 6, 7