PoseD-Flow: Versatile and Guided Flow Matching Model of Human Pose

Supplementary Material

A. Intuition

We believe that an intuitive understanding of our contributions is essential for appreciating their impact and value. A human skeleton / pose is best explained not by a set of real numbers, but by subset of numbers in a $\#\text{ joints} \times \dim(\text{parameterization})$-dimensional space. This subset exactly corresponds to the *product manifold of rotations*, parameterized as $\#$joints amount of rotations. This is not a mere technicality, but a fundamental way to incorporate the structure in the problem into the solution / method. In our case, the inclusion of geometry transforms the model from a purely statistical generator into a structure-aware dynamical system. By operating on the manifold of articulated rotations, the flow respects the intrinsic constraints of human pose, ensuring that both learning and inference unfold along physically and mathematically meaningful trajectories (through curvature and covariance). This is the implicit bias toward realistic, stable, and data-consistent poses. Riemannian geometry is the key enabler of this, which we will review next. Note that, while we provide our code, most of the ideas presented here can be implemented for different purposes via libraries such as geomstats [49], geoopt [34], manopt [11], or others [63].

B. Geometry of Human Poses

We follow NRDF [26] as well as [5, 6, 13] to provide a detailed treatise regarding the product Riemannian manifold on which an articulated human pose lives.

Riemannian geometry. We define an m-dimensional *Riemannian manifold*, embedded in an ambient Euclidean space $\mathcal{X} = \mathbb{R}^d$ and endowed with a *Riemannian metric* $G \triangleq (G_x)_{x\in\mathcal{M}}$ to be a smooth curved space $(\mathcal{M}, G)$. A vector $v \in \mathcal{X}$ is said to be *tangent* to $\mathcal{M}$ at x *iff* there exists a smooth curve $\gamma : [0,1] \to \mathcal{M}$ s.t. $\gamma(0) = x$ and $\dot\gamma(0) = v$. The velocities of all such curves through x form the *tangent space* $\mathcal{T}_x\mathcal{M} = \{\dot\gamma(0) \,|\, \gamma : \mathbb{R} \to \mathcal{M} \text{ is smooth around } 0 \text{ and } \gamma(0) = x\}$, whose union is called the *tangent bundle*: $\mathcal{TM} = \bigcup_x \mathcal{T}_x\mathcal{M} = \{(x, v) \mid x \in \mathcal{M}, v \in \mathcal{T}_x\mathcal{M}\}$. The Riemannian metric $G(\cdot)$ equips each point x with an inner product in the tangent space $\mathcal{T}_x\mathcal{M}$, $\langle \mathbf{u}, v\rangle_x = \mathbf{u}^T G_x v$. 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*, $\mathcal{M}^K := \mathcal{M}_{1:K}$, whose tangent bundle admits the *natural isomorphism*, $\mathcal{TM}^K \simeq (\mathcal{TM} \times \cdots \times \mathcal{TM})$. We now define the operators required for our algorithm.

Definition 1 (Riemannian Gradient)**.** *For a smooth function $f : \mathcal{M} \to \mathbb{R}$ and $\forall (x, v) \in \mathcal{TM}$, we define the* Riemannian gradient *of f as the unique vector field* $\operatorname{grad} f$ *satisfying [10]:*

$$\mathrm{D}f(x)[v] = \langle v, \operatorname{grad} f(x)\rangle_x \tag{B.1}$$

where $\mathrm{D}f(x)[v]$ is the derivation of f by v. It can further be shown that an expression for $\operatorname{grad} f$ *can be obtained through the projection of the* Euclidean *gradient orthogonally onto the tangent space*

$$\operatorname{grad} f(x) = \nabla f(x)_{\|} = \Pi_x\big(\nabla f(x)\big). \tag{B.2}$$

where $\Pi_x : \mathcal{X} \to \mathcal{T}_x\mathcal{M} \subseteq \mathcal{X}$ is an orthogonal projector with respect to $\langle\cdot,\cdot\rangle_x$.

In most packages such as ManOpt [61], Eq. (B.2) is known as the *egrad2rgrad*.

Definition 2 (Riemannian Optimization)**.** *We consider gradient descent to solve the problems of* $\min_{x\in\mathcal{M}} f(x)$*. For a local minimizer or a* stationary point $x^\star$ *of f, the Riemannian gradient vanishes* $\operatorname{grad} f(x^\star) = 0$ *enabling a simple algorithm,* Riemannian Gradient Descent *(RGD):*

$$x_{k+1} = R_{x_k}(-\tau_k \operatorname{grad} f(x_k)) \tag{B.3}$$

where τ_k is the step size at iteration k and R_{x_k} is the retraction *usually chosen related to the exponential map. Note that both RGD and its stochastic variant [8] are practically convergent [8, 10, 51, 62, 75]. Though, only in rare cases is τ_k analytically computable. Therefore, most minimizers use either Armijo or Wolfe line-search [1].*

$\mathbf{SO(3)}$. We now explaing the space of a single joint. A rotation R is an element of the $\mathrm{SO}(3)$ group:

Definition 3 ($\mathrm{SO}(3)$)**.** *Rotations are elements of the special orthogonal group defined as:*

$$\mathrm{SO}(3) = \left\{R \in \mathbb{R}^{3\times3} : R^\top R = \mathbf{I}, \det(R) = 1\right\}. \tag{B.4}$$

Definition 4 ($\mathcal{T}\mathrm{SO}(3)$)**.** *Differentiating the constraint gives the tangent space at identity (Lie algebra):*

$$\mathcal{T}\mathrm{SO}(3) := \mathfrak{so}(3) = \left\{\Omega \in \mathbb{R}^{3\times3} \mid \Omega^\top = -\Omega\right\}. \tag{B.5}$$

Every $\Omega \in \mathfrak{so}(3)$ can be written using the hat operator:

$$\Omega = \hat\omega = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix}, \quad \omega \in \mathbb{R}^3 \tag{B.6}$$

The inverse map (vee) satisfies $(\hat\omega)^\vee = \omega$. Standard choice (from embedding in $\mathbb{R}^{3\times3}$) gives the notion of an inner product:

Definition 5 (Bi-invariant Riemannian metric & distance)**.** $\mathrm{SO}(3)$ *is a compact Lie group with bi-invariant metric*

$$\langle \Omega_1, \Omega_2 \rangle_R = \frac{1}{2} \operatorname{Tr}\left(\Omega_1^\top \Omega_2\right) = \omega_1^\top \omega_2. \tag{B.7}$$

Because the metric is bi-invariant:

$$d\left(R_1, R_2\right) = \left\| \operatorname{Log}_{R_1}\left(R_2\right)^\vee \right\|_2 = \cos^{-1}\left(\frac{\operatorname{Tr}\left(R_1^\top . R_2\right) - 1}{2}\right).$$

This equals the rotation angle between them.

Definition 6 (Exp / Log maps)**.** *At identity the exponential maps is given by the Rodrigues formula and the logarithmic map follows from its inverse:*

$$\operatorname{Exp}_I(\hat{\omega}) = \exp(\hat{\omega}) = I + \frac{\sin\theta}{\theta}\hat{\omega} + \frac{1-\cos\theta}{\theta^2}\hat{\omega}^2 \tag{B.8}$$

$$\operatorname{Log}_I(R) = \frac{\theta}{2\sin\theta}\left(R - R^\top\right), \theta = \cos^{-1}\left(\frac{\operatorname{Tr}(R) - 1}{2}\right),$$

where $\theta = \|\omega\|$*. At a general point* R *:*

$$\operatorname{Exp}_R(\hat{\omega}) = R\exp(\hat{\omega}) \tag{B.9}$$

$$\operatorname{Log}_R(Q) = \operatorname{Log}_I\left(R^\top Q\right) = \log\left(R^\top Q\right) \tag{B.10}$$

Manifold of human poses ($\mathcal{M} := \mathrm{SO}(3)^K$)**.** We parameterize the pose of a 3D articulated body composed of K joints, $x := \{R_i \in \mathrm{SO}(3)\}_{i=1}^K$, on the power manifold of rotations $\mathcal{M} := \mathrm{SO}(3)^K = \mathrm{SO}(3) \times \cdots \times \mathrm{SO}(3)$.

Definition 7 (Geometry of 3D articulated poses)**.** $\mathrm{SO}(3)^K$ *turns into a Riemannian manifold* $(\mathrm{SO}(3)^K, G^K)$ *when endowed with the* L_p product metric $d_{\mathrm{SO}(3)^K} : \mathrm{SO}(3)^K \times \mathrm{SO}(3)^K \to \mathbb{R}$*:*

$$d_{\mathcal{M}}(x, x') = \|d(R_1, R_1'), d(R_2, R_2'), \ldots, d(R_K, R_K')\|_p,$$

where $R \in x \in \mathrm{SO}(3)^K$ *and* $R' \in x' \in \mathrm{SO}(3)^K$*. In this work, we use* $p = 1$*. The natural isomorphism further allows us to write its exponential map* $\operatorname{Exp}_x : \mathcal{T}\mathrm{SO}(3)^K \to \mathrm{SO}(3)^K$ *component-wise:* $\operatorname{Exp}_x = \left(\operatorname{Exp}_{R_1}, \operatorname{Exp}_{R_2}, \ldots, \operatorname{Exp}_{R_K}\right)$*. Akin to this, is the logarithmic map,* Log_x*. Since the tangent spaces and therefore* Π_x *are replicas, the gradient of a smooth function* $f : \mathrm{SO}(3)^K \to \mathbb{R}$ *w.r.t.* x *is also the Cartesian product of the individual gradients:*

$$\operatorname{grad}_x f(x) = \left(\operatorname{grad}_{R_1} f(x), \ldots, \operatorname{grad}_{R_K} f(x)\right). \tag{B.11}$$

C. Omitted Proofs

We now provide the proofs that are excluded from the main paper. Whenever necessary, we will recall the definitions / theorems from the main paper for the sake of a self-contained exposition.

Proof that $\mathcal{L}_{\mathrm{traj}}$ *promotes small rotations.* Let $R \in \mathrm{SO}(3)$ represent a rotation by angle θ about some axis. By Rodrigues' rotation formula:

$$\operatorname{tr}(R) = 1 + 2\cos\theta \tag{C.1}$$

$$\min\ 3 - \operatorname{tr}(R) = \min\ 2(1 - \cos\theta) \tag{C.2}$$

$$\arg\min_\theta\ 2(1 - \cos\theta) = 0 \tag{C.3}$$

Hence minimizing $\mathcal{L}_{\mathrm{traj}}$ prefers small angle solutions. □

Theorem 1 (Tangent denoiser)**.** *For the data distribution* $x_1 \sim p_1$ *on* $\mathcal{M}$*, define at any* $x \in \mathcal{M}$ *the tangent random variable* $\xi_x := \operatorname{Log}_x(x_1) \in \mathcal{T}_x\mathcal{M}$*. The tangent denoiser* μ *at time* t *given by:*

$$\mu(x) := \mu_{1|t}(x) = \mathbb{E}\left[\xi_x \mid x(t) = x\right], \tag{C.4}$$

is the unique minimizer of the Riemannian flow matching loss:

$$\mathcal{L}(v) := \mathbb{E}\left[\|\xi - v\|_{g_x}^2 \mid x_t = x\right]. \tag{C.5}$$

Proof of Thm. 5. The proof follows from the observation that in an inner-product space the best single-vector predictor (in mean squared error) of a random vector is its conditional expectation. This idea has been key to developing RFM [14]. We start by expanding the RFM loss (in the tangent space) as:

$$\mathcal{L}(v) = \mathbb{E}\left[\|\xi - v\|^2 \mid x\right] = \mathbb{E}[\langle \xi - v, \xi - v\rangle \mid x] \tag{C.6}$$

$$= \mathbb{E}\left[\|\xi\|^2 \mid x\right] - 2\langle \mathbb{E}[\xi \mid x], v\rangle + \|v\|^2. \tag{C.7}$$

Observing that the first term does not depend on v, we write:

$$\arg\min_v \mathcal{L}(v) = \arg\min_v \|v\|^2 - 2\langle \mu, v\rangle = \|v - \mu\|^2. \tag{C.8}$$

The unique minimum is obtained at $v = \mu$. □

Theorem 2 (Covariant derivative & covariance)**.** *The covariant derivative* $(\nabla\mu)_v(x) : \mathcal{T}_x\mathcal{M} \to \mathcal{T}_x\mathcal{M}$ *of the tangent denoiser is given by:*

$$(\nabla\mu)_v(x) = A_x[C(x)v] + R_x[v], \tag{C.9}$$

where $A_x[v]$ *is a linear operator,* $C(x) := \mathrm{C}_{1|t}(x)$ *is a self-adjoint, positive semidefinite linear map* $\mathcal{T}_x\mathcal{M} \to \mathcal{T}_x\mathcal{M}$*, representing the covariance under the conditional distribution, and* $R_x[v]$ *is a* Riemannian remainder*:*

$$C(x) = \mathbb{E}\left[\left(\xi_x - \mu_{1|t}\right) \otimes \left(\xi_x - \mu_{1|t}\right) \mid x(t) = x\right] \tag{C.10}$$

$$R_x[v] = \mathbb{E}\left[\nabla_v^{(x)}\xi \mid x\right]. \tag{C.11}$$

As $t \to 1$*,* $C(x)$ *approaches the local data covariance under variance scheduling of geodesic kernels:* $\sigma \to 0$*.*

Proof of Thm. 6. By definition of the denoiser, we have:

$$\mu(x) = \int_{\mathcal{T}_x\mathcal{M}} \xi p(\xi \mid x) d\xi. \tag{C.12}$$

We then take the covariant derivative:

$$(\nabla_v \mu)(x) = \int \nabla_v^{(x)}(\xi) p(\xi \mid x) d\xi + \int \xi \nabla_v^{(x)} p(\xi \mid x) d\xi.$$

The gradient in the first term is the *base-point derivative* whereas the second gradient measures how the posterior weight changes when we move x. Using the score identity $\nabla_v p = p \nabla_v \log p$, we re-write the second integral as:

$$\int \xi \nabla_v p(\xi \mid x) d\xi = \int \xi p(\xi \mid x) \nabla_v \log p(\xi \mid x) d\xi \tag{C.13}$$

$$= \mathbb{E}\left[\xi \nabla_v \log p(\xi \mid x) \mid x\right]. \tag{C.14}$$

By the definition of expectation, we have:

$$\mathbb{E}\left[\nabla_v^{(x)} \xi \mid x\right] := \int \nabla_v^{(x)}(\xi) p(\xi \mid x) d\xi. \tag{C.15}$$

Combining this with Eq. (C.14) and plugging into the definition of the covariant derivative, we write:

$$(\nabla_v \mu)(x) = \mathbb{E}\left[\nabla_v^{(x)} \xi \mid x\right] + \mathbb{E}\left[\xi \nabla_v \log p(\xi \mid x) \mid x\right].$$

We now make the substitution $\xi = \mu(x) + (\xi - \mu(x))$ into Eq. (C.14) and write:

$$\mathbb{E}\left[\xi \nabla_v \log p(\xi \mid x) \mid x\right] = \mathbb{E}\left[(\xi - \mu) \nabla_v \log p \mid x\right] \tag{C.16}$$

since $\mathbb{E}\left[\nabla_v \log p(\xi \mid x) \mid x\right] = 0$ (score is 0-mean). This yields:

$$(\nabla_v \mu)(x) = \mathbb{E}\left[\nabla_v^{(x)} \xi \mid x\right] + \mathbb{E}\left[(\xi - \mu) \nabla_v \log p \mid x\right].$$

Next, we decompose the score function into linear (best-fit) and residual, non-linear (orthogonal) terms as follows:

$$\nabla_v \log p(\xi \mid x) = \langle A_x[v], (\xi - \mu) \rangle + s(\xi, x, v), \tag{C.17}$$

where $A_x : \mathcal{T}_x\mathcal{M} \to \mathcal{T}_x\mathcal{M}$ is a linear map [1], $A_x[v]$ a tangent vector, and $\mathbb{E}[(\xi - \mu) s(\xi, x, v) \mid x] = 0$ from orthogonality. This leads to:

$$\mathbb{E}\left[(\xi - \mu) \nabla_v \log p \mid x\right] = \mathbb{E}\left[(\xi - \mu) \langle A_x[v], (\xi - \mu) \rangle \mid x\right] + \underbrace{\mathbb{E}\left[(\xi - \mu) s(\xi, x, v) \mid x\right]}_{=0} \tag{C.18}$$

$$= (A_x \circ C(x))[v]. \tag{C.19}$$

[1] best-fit linear coefficient in the least-squares sense

The last equality follows from the definition of Riemannian covariance $C(x)$ in Eq. (C.10). We now collect the terms and re-write the covariant derivative:

$$(\nabla_v \mu)(x) = \underbrace{\mathbb{E}\left[\nabla_v^{(x)} \xi \mid x\right]}_{(Eq.\ (C.15))} + \underbrace{(A_x \circ C(x))[v]}_{(Eq.\ (C.19))}. \tag{C.20}$$

Calling the first term $R_x[v] := \mathbb{E}\left[\nabla_v^{(x)} \xi \mid x\right]$:

$$(\nabla \mu)(x)[v] = A_x[v] \circ C(x) + R_x[v]. \tag{C.21}$$

We can also write this in operator-style revealing:

$$(\nabla \mu)(x) = A_x \circ C(x) + R_x. \tag{C.22}$$

□

Corollary 1. *The covariant derivative of the marginal velocity field satisfies ($\circ$ denotes composition):*

$$\nabla u_t(x) = \frac{1}{1-t}\left(C_{1|t}(x) \circ A_x + R_x\right). \tag{C.23}$$

This is the drift of the adjoint ODE, determining the Riemannian adjoint, as we explain next.

Proof of Corollary 2. Pointwise, the velocity of the geodesic interpolation equals the tangent vector from the current position to the final endpoint, scaled by the inverse remaining time:

$$u_t(x) := \mathbb{E}\left[\dot{X}_t \mid x_t = x\right] \tag{C.24}$$

$$= \mathbb{E}\left[\frac{1}{1-t} \mathrm{Log}_x(x_1) \,\middle|\, x_t = x\right] \tag{C.25}$$

$$= \frac{1}{1-t} \mathbb{E}\left[\xi \mid x_t = x\right] \tag{C.26}$$

$$= \frac{1}{1-t} \mu(x). \tag{C.27}$$

Plugging Eq. (C.9) in Thm. 6 into Eq. (C.27):

$$\nabla u_t(x) = \frac{1}{1-t}(\nabla \mu)(x) = \frac{1}{1-t}\left(C_{1|t}(x) \circ A_x + R_x\right).$$

□

While the next theorem is a standard result in Riemannian geometry, we nevertheless prove it, showing backpropagation through the continuous Riemannian flow is pulling the gradient at $t = 1$ back to $t = 0$ via the Riemannian adjoint of the flow map.

Theorem 3 (Riemannian adjoint)**.** *Let $\Psi : \mathcal{M} \to \mathcal{M}$ denote the flow such that $x_1 = \Psi(x_0)$. Then the Riemannian gradients at the start and end points are related by the* Riemannian adjoint *(pullback map) $D_{x_0}\Psi(x_0)^*$:*

$$\mathrm{grad}_{x_0} \mathcal{L}(x_1) = D\Psi(x_0)^* \left[\mathrm{grad}_{x_1} \mathcal{L}(x_1)\right]. \tag{C.28}$$

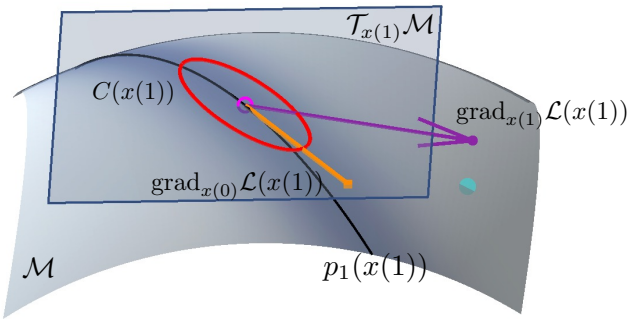


Figure C.1. Additional illustration of the implicit bias in differentiating through the solver. A infinitesimally small change in the source point causes the projection the intrinsic gradient at the end point onto the flow directions up to a curvature term.

Proof of Thm. 7: Relating Riemannian gradients. Let $\Psi : \mathcal{M} \to \mathcal{M}$ denote the flow such that $x_1 = \Psi(x_0)$. Let us write the composite objective as $F(x_0) := \mathcal{L} \circ \Psi(x_0) = \mathcal{L}(\Psi(x_0))$. By the defining property of the Riemannian gradient, for all $\xi_0 \in T_{x_0}\mathcal{M}$, we have

$$dF_{x_0}(\xi_0) = \left\langle \text{grad}_{x_0} F,\ \xi_0 \right\rangle_{g_{x_0}}, \tag{C.29}$$

where g is the Riemannian metric. Applying the chain rule to F yields:

$$dF_{x_0} = d\mathcal{L}_{x_1} \circ D\Psi(x_0), \tag{C.30}$$

where $x_1 = \Psi(x_0)$. Evaluating this on ξ_0 yields

$$\begin{aligned} dF_{x_0}(\xi_0) &= d\mathcal{L}_{x_1}(D\Psi(x_0)[\xi_0]) && \text{(C.31)} \\ &= \left\langle \text{grad}_{x_1} \mathcal{L},\ D\Psi(x_0)[\xi_0] \right\rangle_{g_{x_1}}, && \text{(C.32)} \end{aligned}$$

where the last equality again uses the definition of the gradient. The Riemannian adjoint $D\Psi(x_0)^*$ is defined implicitly by

$$\langle v_1,\ D\Psi(x_0)[\xi_0] \rangle_{g_{x_1}} = \langle D\Psi(x_0)^*[v_1],\ \xi_0 \rangle_{g_{x_0}}, \tag{C.33}$$

$\forall\, v_1 \in T_{x_1}\mathcal{M},\ \xi_0 \in T_{x_0}\mathcal{M}$. Applying this definition with $v_1 = \text{grad}_{x_1} \mathcal{L}$ converts Eq. (C.31) into

$$dF_{x_0}(\xi_0) = \left\langle D\Psi(x_0)^*[\text{grad}_{x_1} \mathcal{L}],\ \xi_0 \right\rangle_{g_{x_0}}. \tag{C.34}$$

Comparing this with Eq. (C.29), and using the non-degeneracy of g_{x_0}, we identify the unique tangent vector that reproduces the linear functional dF_{x_0}, namely

$$\text{grad}_{x_0} F = D\Psi(x_0)^*[\text{grad}_{x_1} \mathcal{L}(x_1)], \tag{C.35}$$

which is precisely Eq. (C.28). This completes the proof. □

Theorem 4 (Implicit bias in endpoint update)**.** *Consider a small optimization step updating the optimized source variable:* $x_0 \longrightarrow \text{Exp}_{x_0}\left(-\eta\, \text{grad}_{x_0} \mathcal{L}\left(x_1\right)\right)$ *where* $\text{grad}_{x_0} \mathcal{L}\left(x_1\right)$ *is given in Eq.* (C.28)*. As* $\eta \to 0$ *(infinitesimal change), the variation of the end-point* $x_1 = \Psi(x_0)$*, denoted* δx_1 *reads:*

$$\delta x(1) = -\eta \underbrace{\left(D_{x_0}\Psi\left(x_0\right) D_{x_0}\Psi\left(x_0\right)^*\right)}_{\mathcal{K} \text{ self-adjoint, PSD on } \mathcal{T}_{x_1}\mathcal{M}} \text{grad}_{x(1)} \mathcal{L}(x(1)),$$

where $\mathcal{K} = D_{x_0}\Psi\left(x_0\right) D_{x_0}\Psi\left(x_0\right)^*$ *resembles a local covariance on the endpoint gradient, explaining why the update is biased toward directions of high density.*

Proof of Thm. 8. Assume a small gradient step:

$$x_0 \longrightarrow x_0' = \text{Exp}_{x_0}\left(-\eta\, \text{grad}_{x_0} \mathcal{L}\left(x_1\right)\right) \tag{C.36}$$

For infinitesimal step size $\eta \to 0$, the first-order variation is

$$\begin{aligned} \delta x_0 &= -\eta\, \text{grad}_{x_0} \mathcal{L}\left(x_1\right) \in \mathcal{T}_{x_0}\mathcal{M} && \text{(C.37)} \\ &= -\eta d\Psi\left(x_0\right)^* \left[\text{grad}_{x_1} \mathcal{L}\left(x_1\right)\right], && \text{(C.38)} \end{aligned}$$

where the last equality follows from plugging Eq. (C.28). The end-point is affected through the flow as: $x_1 = \Psi(x_0)$ and $x_1' = \Psi(x_0 + \delta x_0)$. Assuming an infinitesimally small change, this yields:

$$\delta x_1 := x_1' - x_1 = D\Psi\left(x_0\right)\left[\delta x_0\right] \in \mathcal{T}_{x_1}\mathcal{M}. \tag{C.39}$$

Plugging Eq. (C.37) into Eq. (C.39) (substituting δx_0), gives the tangent vector at x_1 describing how the endpoint moves in response to the gradient step at x_0:

$$\delta x(1) = -\eta \underbrace{\left(D_{x_0}\Psi\left(x_0\right) D_{x_0}\Psi\left(x_0\right)^*\right)}_{\mathcal{K} \text{ self-adjoint, PSD on } \mathcal{T}_{x_1}\mathcal{M}} \text{grad}_{x(1)} \mathcal{L}(x(1)),$$

where $\mathcal{K}$ is a projection onto the reachable subspace of the endpoint tangent space. When the flow generates the data distribution, this operator becomes the local covariance of the data manifold. This completes the proof. □

D. Implementation Details

In this section, we detail the implementation setup used in our experiments.

Training. We train our models using the training split of AMASS [47]. We preprocess the dataset by trimming the first and last 10% of all sequences and sampling them at 30 Hz, resulting in approximately 18 million poses. We train our model for 50000 steps, with a runtime of 8 hours on a single NVIDIA A30 GPU.

Inverse problems. Here, we expand on the implementation details of our optimization algorithm. For initialization,

Table D.1. Hyperparameters for various inverse tasks

Task	LR	Solver	NFE	Num. of Iters	Blending α	Loss weights
Pose Completion	0.1	Euler	100	200 / 300	0.25	$\lambda_{\text{data}} = 1,\ \lambda_{\text{traj}} = 1e^{-3}$
Motion Denoising	0.25	Euler	100	300 / 400	0.75	$\lambda_{\text{data}} = 1,\ \lambda_{\text{temp}} = 1e^{-1}/1e^{-2},\ \lambda_{\text{traj}} = 1e^{-5}$
Human Mesh Recovery	0.1	Euler	100	400	0.75	$\lambda_{\text{data}} = 1,\ \lambda_{\alpha} = \lambda_{\beta} = \lambda_{\text{traj}} = 50$

we use the linear blend strategy from D-Flow [4] for better convergence. Specifically, we initialize x_0 with a blend of a sample from the source distribution p and the backward ODE solution of x^{obs} from $t = 1$ to $t = 0$:

$$x_0 = \sqrt{\alpha} \cdot x_0^{\text{obs}} + \sqrt{1-\alpha} \cdot x_0' \quad x_0' \sim p, \tag{D.1}$$

where

$$x_0^{\text{obs}} = \int_1^0 v_w(x^{\text{obs}}, t)\, dt \tag{D.2}$$

x^{obs} varies by task. For pose completion, it corresponds to the partially observed pose, with the occluded joints filled with the mean pose. For human mesh recovery, x^{obs} is given by the output of CLIFF [36]. Using this initialization, the optimization proceeds as described in Alg. 2. The hyperparameters used in all experiments are presented in Tab. D.1.

For pose completion, we use 300 iterations for the arms occluded case, and 200 iterations for all other cases. For motion denoising, we run 400 iterations when denoising the HPS [25] dataset and when the noise standard deviation is set to 0.1, along with setting $\lambda_{\text{temp}} = 1e^{-1}$ for the latter.

For pose completion, we evaluate on the AMASS test split with a sampling rate of 10. We generate 10 hypotheses for each partial ground truth. For motion denoising, we use the HumanEva split from AMASS and the HPS [25] dataset. Both datasets undergo the same preprocessing: we sample motions at 30 Hz and segment them into 60-frame chunks. This yields 190 sequences from HumanEva and 6,359 from HPS. To limit HPS size, we randomly select 50 sequences per subject with a seed of 42, resulting in a final set of 350 sequences. For human mesh recovery, we use the EHF [53] dataset, which contains 100 images. We extract 2D keypoints using ViTPose-H [69]. When initializing with CLIFF [36], we predict the poses using the "hr48-PA53.7_MJE91.4_MVE110.0_agora_val.pt" checkpoint.

Additional details on the teaser figure. Fig. 1 depicts how the noise distribution $p(x_0)$ flows towards the data distribution $p(x_1)$ with PoseRFM. We project the distributions onto a 2D surface embedded in 3D space. While the extrinsic shape of this surface is arbitrarily chosen for illustrative purposes, the distributions mapped onto it are rigorously obtained by dimensionally reducing real flow trajectories. In particular, we randomly sample $10,000$ poses on $\mathcal{M} = \text{SO}(3)^K$: $\{x_0^{(i)}\}_{i=1}^{10\text{k}} \sim p(x_0)$ and propagate them using the learned PoseRFM, $v_w(x, t)$. We record samples positions on the manifold $\{x_t^{(i)}\}_{i=1}^{10\text{k}}$ for multiple time-steps t. Then we collect all these samples and learn a transformation to a common two-dimensional manifold using PaCMAP [67] with its default hyperparameter selection, but replacing the available distances with the correct geodesic distance, $d_{\text{SO}(3)^K}$. The learned transformation is used to project the samples $(\{x_t^{(i)}\}_{i=1}^{10\text{k}})$ at $t = [0.0, 0.2, 0.4, 0.6, 0.8, 1.0]$, which are then used to estimate the distributions via kernel density estimation. The same transformation is also used to draw the optimization trajectories of Riemannian D-Flow on $p(x_0)$ and $p(x_1)$.

Further details. **(i) How is β obtained?** For IK from images, we use CLIFF [36] to predict human pose and shape β, and then refine these predictions via optimization. For all other tasks, we fix $\beta = 0$ and optimize for the pose. **(ii) FID for pose.** As in NRDF [26], we compute the FID using the Frechet distance between the 3D positions of the generated and real body joints, both obtained through SMPL model. **(iii) Dimension of u_t.** Our flow field lives in the tangent space of K articulated joints. Hence, $\dim(u_t) = K \times 3$.

E. Additional Results

Ablation studies. First, we conduct an ablation study on the sampling strategy for unconditional pose generation. We evaluate the impact of different ODE solvers, retraction methods, and number of function evaluations (NFE), aiming to balance generation quality, diversity, and runtime. The results are summarized in Tab. E.1, with the best configuration highlighted.

Using the midpoint ODE solver increases sample diversity with only a small runtime overhead, while integrating for 1000 steps offers no meaningful gains. For retraction, projecting back to the manifold after every step is prohibitively slow and provides no benefit, making it impractical. Retraction only at the final step or using the Exp map are both viable; with the former being faster, while the latter yielding higher diversity.

We perform the same analysis for the pose completion inverse task. Guided by the previous findings, we exclude per-step retraction and the 1000 NFE setting. We evaluate accuracy, diversity, and runtime for the remaining choices, as shown in Tab. E.2.

In this setting, using the midpoint solver becomes more

Table E.1. Ablation results on sampling strategies for pose generation

Solver	Retraction	100 steps				1000 steps			
		FID ↓	APD ↑	d_{NN} ↓	Time ↓	FID ↓	APD ↑	d_{NN} ↓	Time ↓
Euler	Last step	0.013	14.984	0.066	0.614	0.013	15.401	0.069	2.028
	Every step	0.014	15.256	0.068	28.406	0.014	15.431	0.069	284.602
	Exp map	0.014	15.252	0.068	1.294	0.014	15.431	0.069	10.734
Midpoint	Last step	0.014	15.447	0.070	0.773	0.014	15.451	0.070	3.570
	Every step	0.014	15.448	0.070	28.401	0.014	15.451	0.070	284.082
	Exp map	0.015	15.786	0.071	1.383	0.014	15.483	0.070	12.350

Table E.2. Ablation results on sampling strategies for pose completion.

Solver	Retraction	Occ. left leg			Occ. legs		
		MPVPE ↓	APD ↑	Time ↓	MPVPE ↓	APD ↑	Time ↓
Euler	Last step	83.81	6.02	100.71	95.00	7.23	106.10
	Exp map	82.89	5.94	483.72	94.20	7.18	475.08
Midpoint	Last step	83.85	6.12	140.55	95.82	7.37	146.80
	Exp map	84.01	6.12	523.92	95.90	7.38	517.94

Table E.3. Runtime comparison across various tasks. Timings are measured in seconds.

Task	Batch Size	DPoser [46]	PoseFM	PoseRFM
Pose Generation	500	1.055	0.118	0.773
Pose Completion	500	0.82	20.91	100.71
Motion Denoising	60	5.07	32.49	80.85
HMR (IK)	100	17.05	47.90	120.13

expensive as it requires two passes through the model per step, unlike Euler's single pass. Likewise, using the Exp map significantly slows optimization with only modest accuracy gains. We adopt the highlighted configuration for this and all other inverse problems.

Runtime comparison. We benchmark the runtime for each task over 10 runs and report the median in Tab. E.3. All measurements were performed on a NVIDIA A100 GPU.

The ODE formulation of flow models enables fast pose generation relative to SotA diffusion methods. Unfortunately, the benefits stop there: PoseRFM remains orders of magnitude slower than diffusion-based approaches in downstream tasks. Each iteration of Riemannian D-Flow requires backpropagating all the way to the source point, which is substantially slower than the single-step denoising used in DPoser [46]. In addition, Riemannian D-Flow requires differentiating through geometric components such as geodesics and Exp map, introducing further computational overhead. Developing an optimized version of our algorithm is left for future work.

Convergence analysis. Thus far, we have focused on the final output of our inverse algorithm. In this section, we will examine the behavior of x_1 throughout optimization.

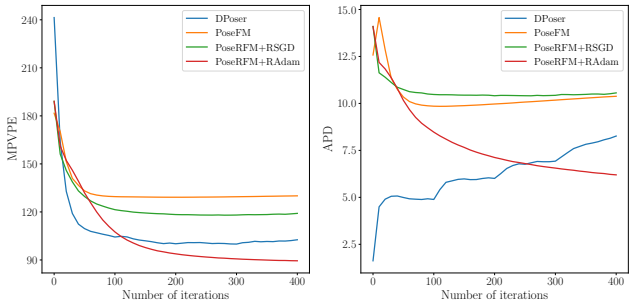
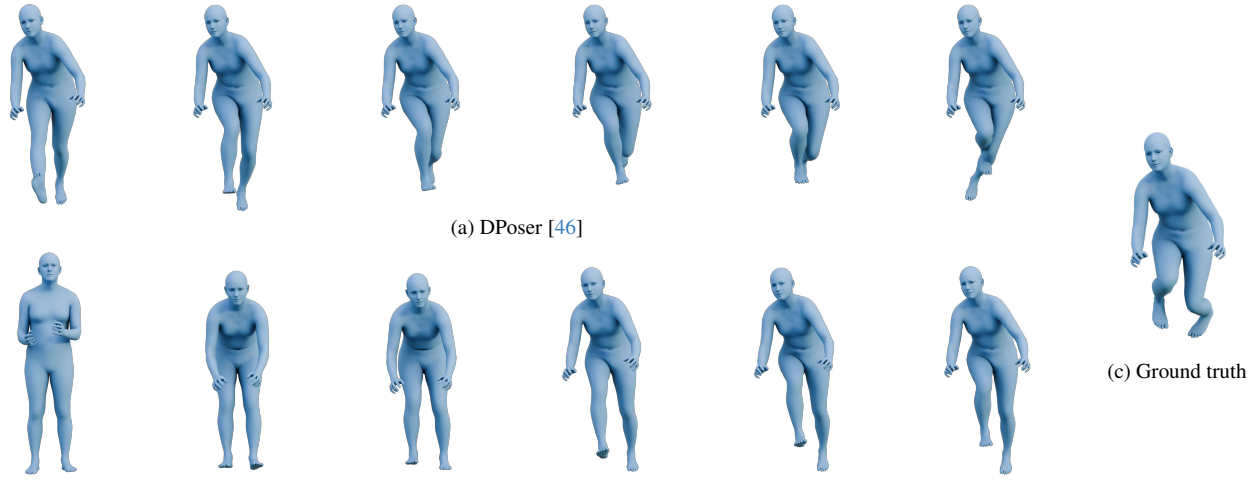


Figure E.1. Convergence comparison for various methods on pose completion with legs occluded.

We first plot the convergence curves in Fig. E.1, showing how accuracy and diversity evolve over the optimization steps. For this analysis, we use the pose completion task with legs occluded, and compare four methods: the diffusion baseline DPoser [46], PoseFM, PoseRFM with RiemannianSGD, and PoseRFM with RiemannianAdam.

For MPVPE, both PoseFM and PoseRFM with RSGD stop converging after a few iterations, leading to their poor accuracy. Meanwhile, PoseRFM with RAdam continues to improve up to 400 iterations. APD follows a similar pattern, except that flow and diffusion models converge differently. DPoser [46] begins with similar poses and gradually introduces diversity, while PoseRFM starts from diverse poses and progressively converges toward poses resembling the ground truth. To further illustrate this behavior, we plot intermediate poses at various iterations K for the same inverse problem in Fig. E.2. A similar visualization for human mesh recovery is shown in Fig. E.3. In this case, both DPoser [46] and PoseRFM converge rapidly, with later iterations introducing only minor refinements.

Metrics for unoccluded regions. Since unoccluded joints are *observed* as ground truth in the data, their fitting error is 0 by construction. To better understand the fitting, we remove this replacement step and report results for both the visible regions and the full pose in Tab. E.4. Although the geodesic loss is geometrically accurate, it is harder to optimize than simple MSE. Nevertheless, our method general-

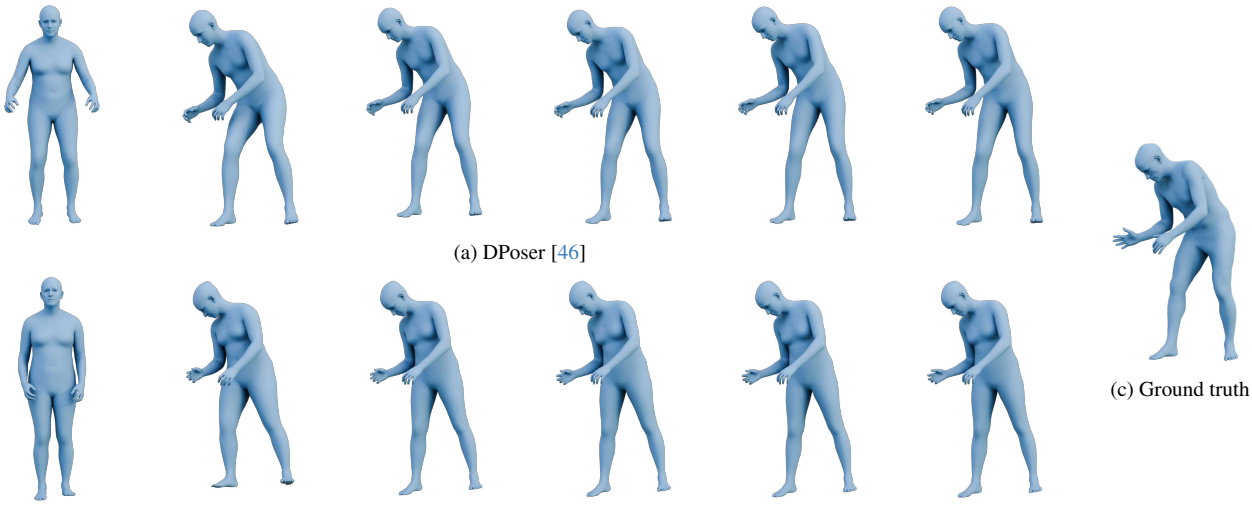


(a) DPoser [46]

(b) PoseRFM

(c) Ground truth

Figure E.2. Intermediate results at iterations K for pose completion with occluded legs. Left to right: $(K = 0, 40, 80, 120, 160, 200)$.



(a) DPoser [46]

(b) PoseRFM

(c) Ground truth

Figure E.3. Intermediate results at iterations K for inverse kinematics. Left to right: $(K = 0, 100, 200, 300, 400, 500)$.

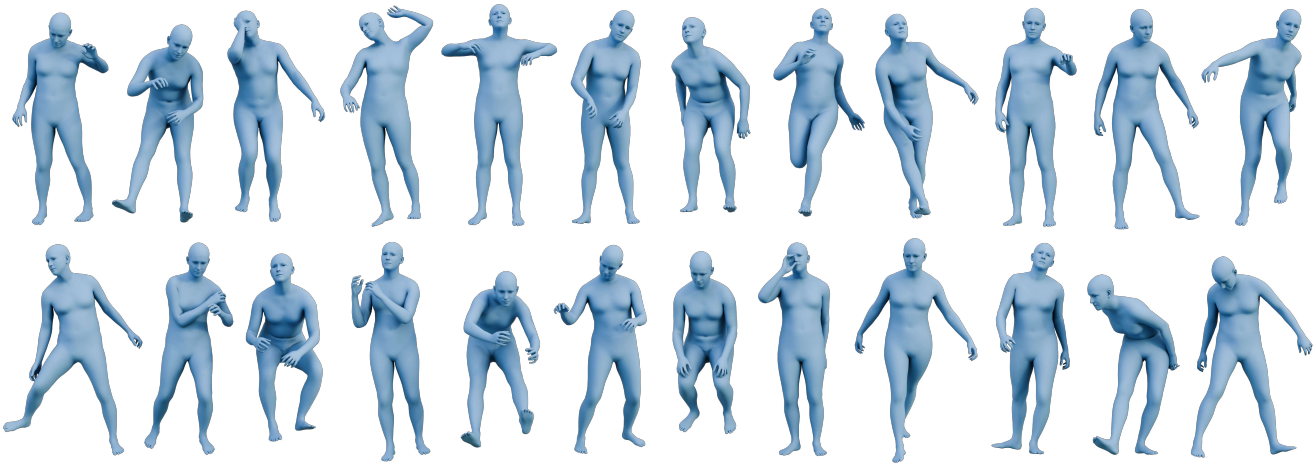


Figure E.4. More samples from unconditional generation using PoseRFM.

Table E.4. Additional Pose Completion metrics

Method	Occ. part	MPVPE ↓			APD ↑		
		Vis.	Occ.	Full	Vis.	Occ.	Full
DPoser [46]	Left leg	0.38	78.31	6.59	0.01	6.53	1.22
Ours	Left leg	10.02	83.81	16.03	0.81	6.02	1.76
DPoser [46]	Legs	0.60	102.46	17.26	0.01	7.75	2.80
Ours	Legs	8.57	95.00	22.66	0.74	7.23	3.10
DPoser [46]	Arms	19.19	104.94	28.54	0.01	5.69	2.09
Ours	Arms	26.56	107.36	36.46	0.60	5.72	2.55

izes well to occluded regions despite potential underfitting.

Qualitative comparison. We provide additional qualitative examples for pose generation (Fig. E.4), pose completion (Fig. E.5), motion denoising (Fig. E.6), and human mesh recovery (Fig. E.7). Comments on each comparison are included in the corresponding figure captions.

Beyond human pose. Our versatile Riemannian D-Flow framework can be used across different data domains and tasks. To illustrate, we now provide an application in **Earth science**. In this context, we no longer operate on the power manifold of rotations, but on the surface of the Earth, which is approximated by a sphere $\mathcal{S}^2$ embedded in the ambient 3D Euclidean space $\mathbb{R}^3$. Samples on this manifold are now points on $\mathcal{S}^2$.

We leverage the pre-trained RFM models of [14] as priors over the distributions of: volcanic activity (since $4,360$ BC), earthquakes (since $2,150$ BC), major floods (since 1985), and recent wildfires. Each data point represents the spatial occurrence of an event, without currently factoring in temporal or auxiliary meteorological data (e.g., temperature, wind, pressure, etc.). With each of these models we formulate an inpainting problem on a portion of the Earth's surface and use our Riemannian D-Flow framework to infer the x_0 producing the most accurate results outside the region to be in-painted. Guidance is still provided in the form of Riemannian source-point optimization following Eq. (10):

$$\min_{x_0 \in \mathcal{S}^2} \big(\mathcal{L}(x(1)) := \mathcal{L}_{\text{data}}(x(1)) + \mathcal{R}(x_0, u)\big).$$

$\mathcal{L}_{\text{data}}(x(1))$ is conceptually similar to Eq. (20), but uses the geodesic distance on $\mathcal{S}^2$

$$d_{\mathcal{S}^2}(x_a, x_b) = r \ \arccos\left(\frac{x_a \cdot x_b}{r^2}\right),$$

and the mask is applied on a subset $\Omega \subset \mathcal{S}^2$ of $\mathcal{S}^2$ rather than on the state itself. Assuming to operate on a unit-sphere, the data loss can be written as:

$$\mathcal{L}_{\text{data}}(x_1) = \sum_{i \in (\mathcal{S}^2 \setminus \Omega)} \arccos\left(x_1^{(i)} \cdot x_{\text{obs}}^{(i)}\right),$$

Table E.5. Dataset specifications for the climate experiments

Event	Masked region	GT size	Total Test size
Volcano	Japan & nearby	16	82
Earthquake	Central & South America	104	612
Flood	South & South-East Asia	182	487
Fire	Africa	499	1280

Table E.6. Negative log-likelihood (NLL) measured on the inpainted region on Earth Climate dataset.

Method	Volcano	Earthquake	Flood	Fire
D-Flow [4]	-4.669	0.445	-0.171	-0.472
Riemannian D-Flow	**-6.337**	**-0.641**	**-0.451**	**-1.174**

where $x_{\text{obs}}^{(i)}$ is the i-th observed data point (ground truth) outside the masked region. By minimizing the full objective using this data term, the prior allows us to infer plausible points also within the masked region Ω.

We use the test split of each dataset as our ground truth and select regions across the Earth where the corresponding natural event has commonly occurred. Details of the selected regions are provided in Tab. E.5. We compare 2 methods, D-Flow [4] and Riemannian D-Flow, for recovering the missing points in the masked regions. We evaluate performance using the negative log-likelihood (NLL) of the predicted points within the masked area, where lower values indicate a higher likelihood of the event occurring at that point. Results for all four datasets are presented in Tab. E.6. Riemannian D-Flow achieves the best performance across every event, indicating its generalizability to different manifolds. To visualize the results, we plot the masked regions, along with the ground truth and predicted points in Fig. E.8.

References

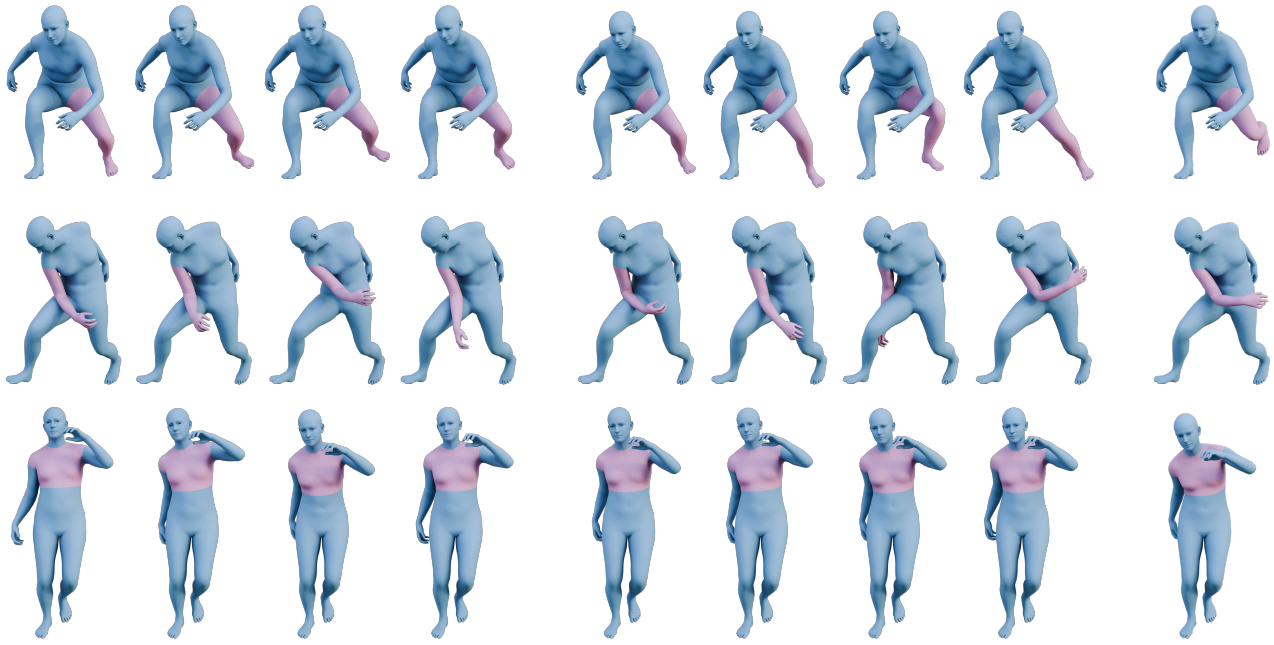
[1] P-A Absil and Kyle A Gallivan. Accelerated line-search and trust-region methods. *SIAM Journal on Numerical Analysis*, 47(2):997–1018, 2009. 1

[4] Heli Ben-Hamu, Omri Puny, Itai Gat, Brian Karrer, Uriel Singer, and Yaron Lipman. D-flow: differentiating through flows for controlled generation. In *Proceedings of the 41st International Conference on Machine Learning*, pages 3462–3483, 2024. 5, 8

[5] 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

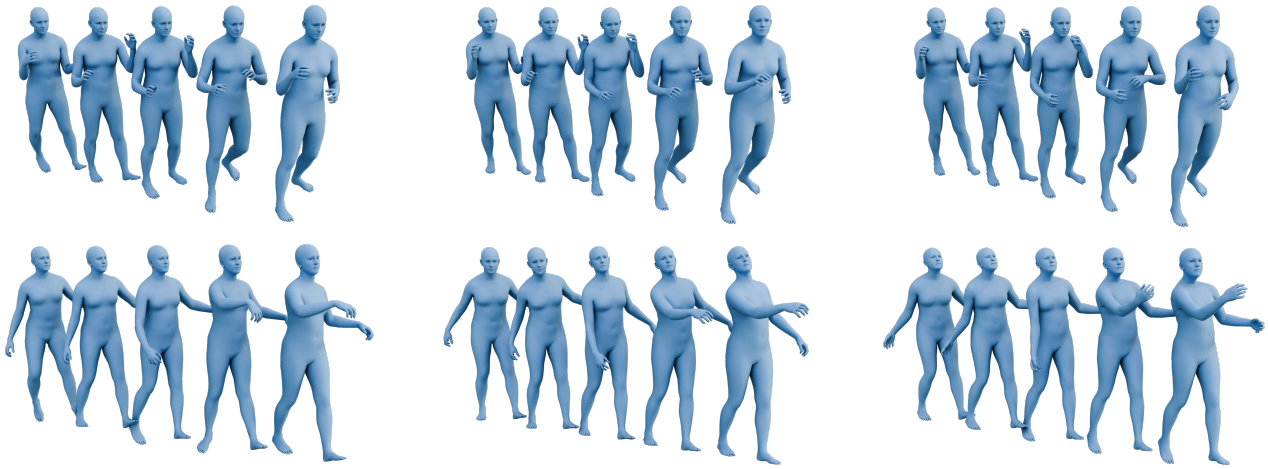
[6] Tolga Birdal, Umut Simsekli, Mustafa Onur Eken, and Slobodan Ilic. Bayesian pose graph optimization via bingham distributions and tempered geodesic mcmc. *Advances in Neural Information Processing Systems*, 31, 2018. 1

[8] Silvere Bonnabel. Stochastic gradient descent on riemannian manifolds. *IEEE Transactions on Automatic Control*, 58(9): 2217–2229, 2013. 1



(a) DPoser [46] (b) PoseRFM (ours) (c) Ground truth (GT)

Figure E.5. Completed poses with left leg (top), right arm (middle) and torso (bottom) occluded. We show both visible and occluded joints. PoseRFM continues to generate realistic and diverse completions, except in the torso-occlusion case, where the limited diversity is clearly noticeable.



(a) DPoser [46] (b) PoseRFM (ours) (c) Ground truth (GT)

Figure E.6. Motion denoising with standard deviations of 40mm (top row) and 100mm (bottom row).

[10] Nicolas Boumal. An introduction to optimization on smooth manifolds. *Available online, May*, 2020. 1

[11] Nicolas Boumal, Bamdev Mishra, P-A Absil, and Rodolphe Sepulchre. Manopt, a matlab toolbox for optimization on manifolds. *The Journal of Machine Learning Research*, 15 (1):1455–1459, 2014. 1

[13] 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

[14] Ricky T. Q. Chen and Yaron Lipman. Flow matching on general geometries. In *The Twelfth International Conference on Learning Representations*, 2024. 2, 8

[25] Vladimir Guzov, Aymen Mir, Torsten Sattler, and Gerard Pons-Moll. Human poseitioning system (hps): 3d human pose estimation and self-localization in large scenes from body-mounted sensors. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021. 5

[26] Yannan He, Garvita Tiwari, Tolga Birdal, Jan Eric Lenssen,

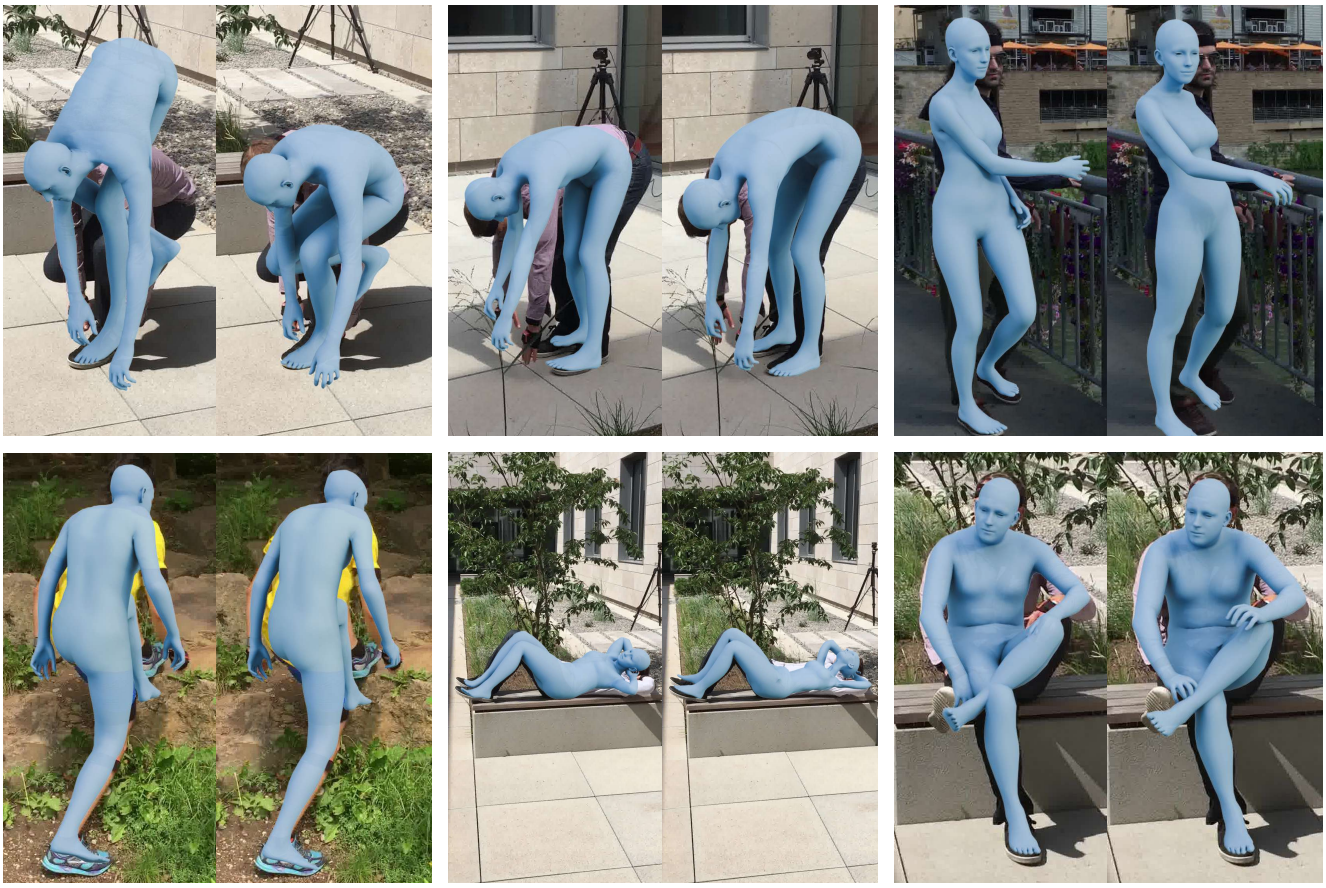


Figure E.7. Additional results of HMR on in-the-wild images from 3DPW [64]. Fitting from scratch (top) and initialization using CLIFF [36] (bottom). These results highlight the strength of PoseRFM on a challenging non-linear problem. DPoser [46] fails to optimize in two cases, and predicts poses with self-intersections.

and Gerard Pons-Moll. Nrdf: Neural riemannian distance fields for learning articulated pose priors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1661–1671, 2024. 1, 5

[34] Max Kochurov, Rasul Karimov, and Serge Kozlukov. Geoopt: Riemannian optimization in pytorch, 2020. 1

[36] Zhihao Li, Jianzhuang Liu, Zhensong Zhang, Songcen Xu, and Youliang Yan. Cliff: Carrying location information in full frames into human pose and shape estimation. In *European Conference on Computer Vision*, pages 590–606. Springer, 2022. 5, 10

[46] Junzhe Lu, Jing Lin, Hongkun Dou, Ailing Zeng, Yue Deng, Xian Liu, Zhongang Cai, Lei Yang, Yulun Zhang, Haoqian Wang, et al. Dposer-x: Diffusion model as robust 3d whole-body human pose prior. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9988–9997, 2025. 6, 7, 8, 9, 10

[47] Naureen Mahmood, Nima Ghorbani, Nikolaus F Troje, Gerard Pons-Moll, and Michael J Black. Amass: Archive of motion capture as surface shapes. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 5442–5451, 2019. 4

[49] Nina Miolane, Nicolas Guigui, Alice Le Brigant, Johan Mathe, Benjamin Hou, Yann Thanwerdas, Stefan Heyder, Olivier Peltre, Niklas Koep, Hadi Zaatiti, Hatem Hajri, Yann Cabanes, Thomas Gerald, Paul Chauchat, Christian Shewmake, Daniel Brooks, Bernhard Kainz, Claire Donnat, Susan Holmes, and Xavier Pennec. Geomstats: A python package for riemannian geometry in machine learning. *Journal of Machine Learning Research*, 21(223):1–9, 2020. 1

[51] Julien Munier. Steepest descent method on a riemannian manifold: the convex case. *Balkan Journal of Geometry & Its Applications*, 12(2), 2007. 1

[53] 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. 5

[61] 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

[62] Nilesh Tripuraneni, Nicolas Flammarion, Francis Bach, and

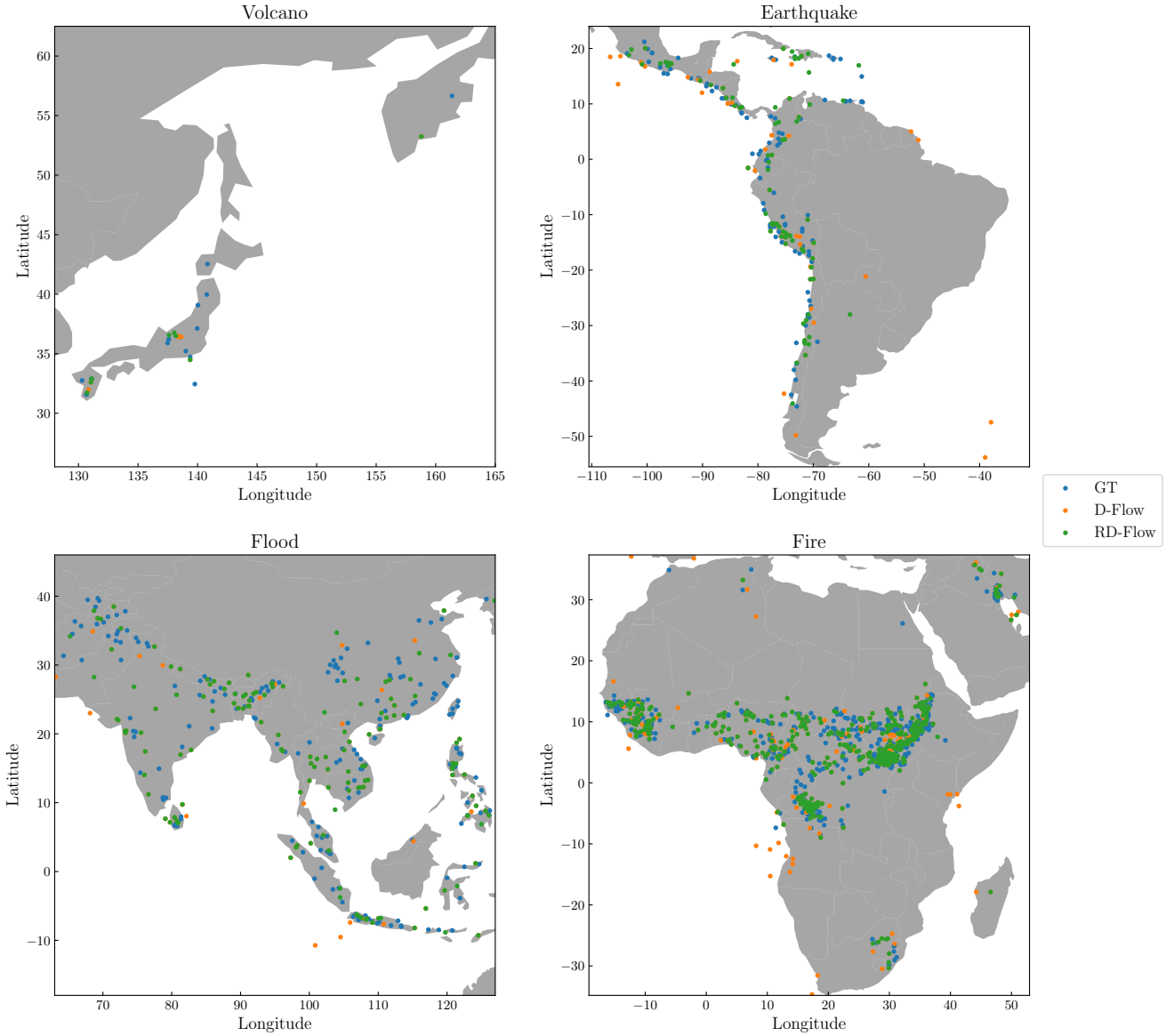


Figure E.8. Results on the climate dataset. Each visualization depicts the prediction over the region to be inpainted Ω. Riemannian D-Flow generates geographically consistent and likely points across diverse hotspots. In contrast, D-Flow often produces unlikely predictions, such as wildfires appearing in the ocean, leading to its poor NLL.

Michael I Jordan. Averaging stochastic gradient descent on riemannian manifolds. In *Conference On Learning Theory*, pages 650–687. PMLR, 2018. 1

[63] Hippolyte Verninas, Caner Korkmaz, Stefanos Zafeiriou, Tolga Birdal, and Simone Foti. Parallelised differentiable straightest geodesics for 3d meshes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2026. 1

[64] Timo Von Marcard, Roberto Henschel, Michael J Black, Bodo Rosenhahn, and Gerard Pons-Moll. Recovering accurate 3d human pose in the wild using imus and a moving camera. In *Proceedings of the European conference on computer vision (ECCV)*, pages 601–617, 2018. 10

[67] Yingfan Wang, Haiyang Huang, Cynthia Rudin, and Yaron Shaposhnik. Understanding how dimension reduction tools work: An empirical approach to deciphering t-sne, umap, trimap, and pacmap for data visualization. *Journal of Machine Learning Research*, 22(201):1–73, 2021. 5

[69] Yufei Xu, Jing Zhang, Qiming Zhang, and Dacheng Tao. Vitpose: Simple vision transformer baselines for human pose estimation. *Advances in neural information processing systems*, 35:38571–38584, 2022. 5

[75] Hongyi Zhang and Suvrit Sra. First-order methods for geodesically convex optimization. In *Conference on Learning Theory*, pages 1617–1638. PMLR, 2016. 1