Learning Riemannian Metrics for Interpolating Animations

Sarah Kushner[1,2], Vismay Modi[1], and Nina Miolane[2]

[1] University of Toronto, Ontario, Canada {sak,vismay}@cs.toronto.edu
[2] UC Santa Barbara, California, USA ninamiolane@ucsb.edu

Abstract. We leverage a family of Riemannian metrics to upsample low frame rate animations for creative design and compression applications in computer graphics. Our method interpolates animated characters' bone orientations along various geodesics from a family of invariant Riemannian metrics on a product of $SO(3)$ manifolds. For compression, an optimization step selects the best-fitting metric. We show that our approach outperforms existing techniques.

1 Introduction

Upsampling for Creative Design Character animation in 3D graphics involves posing a skeletal rig—an articulated hierarchy of bones—across a sequence of frames. To this end, animators define a sparse set of keyframes, specifying the position and orientation of bones at selected times, and then interpolate to generate the full motion [6]. However, standard interpolation methods such as linear interpolation, SLERP [12] or Squad [13] degrade in quality when keyframes are sparse, requiring manual corrections or additional keyframes. Recent learning-based approaches improve interpolation fidelity but depend on large datasets and do not generalize well across rig structures [11, 15]. These limitations motivate the search for alternative interpolation strategies that provide higher-quality results under sparse sampling while remaining data-efficient and rig-agnostic.

Upsampling for Compression Upsampling is also critical for compression, as animations often demand significant storage. Compressing animations into a small number of keyframes, with interpolation recovering the full sequence, is an effective strategy – but hinges on the quality of the interpolation. Upsampling beyond the original frame rate allows animators to then create an animation curve with an arbitrarily high frame rate. Thus, there is a motivation for researching new upsampling techniques that can achieve high accuracy in recovering original animations and even enhance them by increasing their original frame rate.

Contributions We apply geodesic interpolation techniques to animation upsampling and compression. We model the animated character's pose space as the Lie group $SO(3)^B$, with B the number of bones, and equip it with invariant Riemannian metrics. We explore how varying the metric influences interpolation quality and motion characteristics and how this can be used for creative control and data compression.

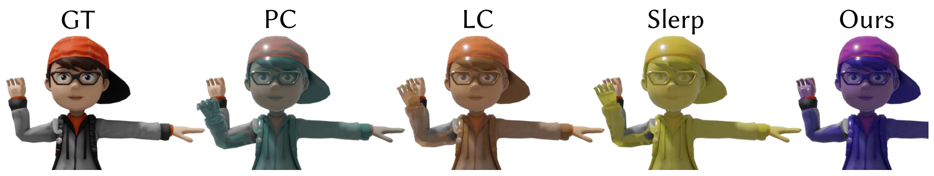


Fig. 1: Comparison of traditional interpolation techniques with the proposed geodesic interpolations. From left to right: ground truth animation, piecewise constant, linear (cartesian), spherical linear (slerp), and our geodesic interpolation. In this frame our geodesic interpolation most closely matches the original.

2 Methods

We introduce elements of Riemannian geometry and Lie groups.

2.1 Manifolds and Riemannian Metrics

Lie theory and Riemannian geometry provide mathematics to precisely define the poses of animation characters, specifically the rotation of each joint of a character. We refer the reader to [5] for mathematical details. We will represent the space of possible animated character poses as a Lie group equipped with a Riemannian metric. We define these concepts here.

Definition 1 (Riemannian metric). *Let $\mathcal{M}$ be a d-dimensional smooth connected manifold and $T_p\mathcal{M}$ be its tangent space at point $p \in \mathcal{M}$. A Riemannian metric $<,>$ on M is a collection of inner products $<,>_p$: $T_p\mathcal{M} \times T_p\mathcal{M} \rightarrow \mathbb{R}$ on each tangent space $T_p\mathcal{M}$ that vary smoothly with p. A manifold $\mathcal{M}$ equipped with a Riemannian metric $<,>$ is called a Riemannian manifold.*

A Riemmanian metric $<,>$ provides a notion of *geodesic distance* `dist` on $\mathcal{M}$. Let $\gamma : [0,1] \rightarrow \mathcal{M}$ be a smooth parameterized curve on $\mathcal{M}$ with velocity vector at $t \in [0,1]$ denoted as $\dot{\gamma}_t \in T_{\gamma(t)}\mathcal{M}$. The length of γ is defined as $L_\gamma = \int_0^1 \sqrt{<\dot{\gamma}_t, \dot{\gamma}_t>_{\gamma_t}} dt$ and the distance between any two points $p, q \in \mathcal{M}$ is: $\mathrm{dist}(p,q) = \inf_{\gamma:\gamma(0)=p,\gamma(1)=q} L_\gamma$. The Riemannian metric also provides the notion of geodesic.

Definition 2 (Geodesic). *A geodesic between two points p, q is defined as a curve which minimizes the energy functional:*

$$E(\gamma) = \frac{1}{2}\int_0^1 < \dot{\gamma}(t), \dot{\gamma}(t)) >_{\gamma(t)} dt. \tag{1}$$

Curves minimizing the energy E also minimize the length L: geodesics are locally distance-minimizing paths on the manifold $\mathcal{M}$.

Intuitively, a geodesic is the generalization of straight lines from vector spaces to manifolds, see Figure 2a. We note that the notion of geodesic depends on the notion of geodesic distance, and thus on the choice of Riemannian metric on the manifold $\mathcal{M}$. Different Riemannian metrics yield different geodesics between two given points.

2.2 Lie Groups and Metrics

In the context of animation interpolations, we consider specific manifolds: Lie groups.

Definition 3 (Lie group). *A Lie group is a group $(G, \cdot)$ such that G is also a finite dimensional smooth manifold, and the group and differential structures are compatible, in the sense that the group law $\cdot$ and the inverse map $g \mapsto g^{-1}$ are smooth. Let e denote the neutral element, or identity of G. The tangent space T_eG of a Lie group at the identity element $e \in G$ is called the Lie algebra of G.*

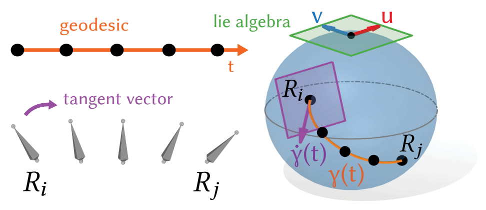
The set of all 3D rotations forms a Lie group. This group is referred to as $SO(3)$, the special orthogonal group in three dimensions. It is defined as: $SO(3) = \{R \in M_3(\mathbb{R}) | R^T R = I_3, \det(R) = 1\}$, where each element is a 3D rotation matrix R. Its The Lie algebra is a vector space of dimension 3, which is also called the dimension of the Lie group $SO(3)$.

Consider an animation of a character with B bones. Each bone is associated with a joint that has some 3D orientation, represented as a rotation matrix $R \in SO(3)$. The set of all possible poses of this character is the power Lie group $SO(3)^B = SO(3) \times \cdots \times SO(3)$, which we call the *pose Lie group* (see Figure 2b).

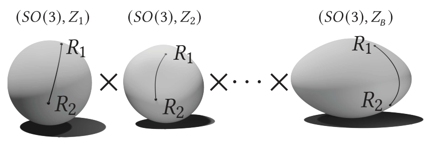
We equip this Lie group with a *Riemannian metric* $<, >$, that is by a collection of inner-products on the tangent spaces that varies smoothly with the base point. We do so by equipping each of the $SO(3)$s with a different invariant Riemannian metric. An invariant metric on a connected Lie group is fully described by the matrix Z of its inner-product on its Lie algebra, and whether it is a left- or a right- invariant metric. Once the pose Lie group is equipped with a Riemannian metric, we can consider geodesics on it, that is, the generalization of straight lines from vector spaces to manifolds, see Figure 2a. We refer to [9] for additional details on Riemannian geometry on Lie groups.

Goal: Consider a *ground truth animation* A_G, see Figure 3 (left) with F frames, represented as a sequence of F poses on the pose Lie group, *i.e.*, $A_G(t) \in SO(3)^B$ for each time $t \in [t_1, t_F]$. Our goal is to learn the Riemannian metric $<, >$ on $SO(3)^B$ that best describes the animated character's motion in A_G, in the following sense: the animation A_G can be downsampled (compressed) to a lower frame rate F', such that the geodesic interpolation with metric $<, >$ brings it back to its original (higher) frame rate F with the highest accuracy. Once $<, >$ is learned, it can be used for creative design in digital creation, including extracting perceptual insights on the character's motion in A_G, or for compression.

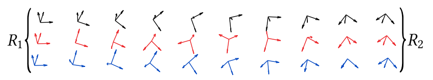
Notations: The metric $<, >$ is the result of an optimization problem that depends on A_G and F', for which we introduce notations. Consider a sampling rate $0 < s < 1$. We call *initial animation*, and denote it A_I, the animation obtained after uniformly downsampling the ground truth animation A_G of frame rate F to the lower frame rate $F' = sF$, see Figure 3 (frames colored pink). For example, if we have a ground truth animation of $F = 60$ frames, and a sampling rate of $s = 0.2$, the initial animation will have $F' = 12$ frames, each 5 frames apart in the ground truth. We call *interpolated* or *upsampled animation*, and denote it A_U, the animation obtained by upsampling the initial animation A_I back up to the ground truth frame rate F, see Figure 3 (frames colored purple). We note that A_U depends on the interpolation technique used: in particular, in the case of a geodesic interpolation, A_U depends on the choice of metric $<,>$. Reformulated using these notations, our goal is to learn the metric $<,>$ so that A_U is as close as possible to A_G, according to a quality score Q. Figure 3 shows our pipeline.



(a) For ease of explanation, we represent $SO(3)$ as a sphere. At the identity of the group, we define an inner-product for all vectors u, v in the tangent space (green). The vector $\dot{\gamma}(t)$ in the tangent space at R_i (purple) is the velocity of the parameterized curve going to R_j. The geodesic curve $\gamma(t)$ (orange) is the shortest path between two rotations R_i and R_j. In this example we interpolate 3 in-between rotations along the geodesic (black dots).



(b) Changing α, β values can be thought of as deforming the group. The distance between the same 2 rotations changes. Our Pose Lie group is a product of manifolds, one for each bone.



(c) We verify experimentally that updating α, β leads to different geodesics, and thus different trajectories despite the same start and end states.

Fig. 2: Explaining how a geodesic on a manifold can interpolate trajectories.

2.3 Riemannian Metric Learning

We propose to learn the metric $<,>$ that most accurately describes the motion of a given animated character. We restrict our optimization to a set of invariant Riemannian metrics on $SO(3)^B$, which provides a convenient parameterization of $<,>$.

Metric Parameterization Consider one $SO(3)$ within the power Lie group $SO(3)^B$. We can parameterize a Riemannian metric on the Lie group $SO(3)$ by an inner product matrix Z on its Lie algebra.

The matrix Z must be symmetric positive definite, meaning it can be decomposed as $Z = P^T DP$, where D is a diagonal matrix with strictly positive values and P is orthogonal. In our work, we restrict Z to a diagonal form:

$$Z = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \alpha & 0 \\ 0 & 0 & \beta \end{bmatrix}, \qquad \text{with } \alpha, \beta > 0, \tag{2}$$

for each component $SO(3)$ within the Pose Lie group $SO(3)^B$. This corresponds to penalizing displacements along the standard basis directions of the Lie algebra, with weights 1, α, and β. A more general symmetric positive definite matrix Z would allow penalizing arbitrary directions (*i.e.*, different orthonormal bases), but in practice we

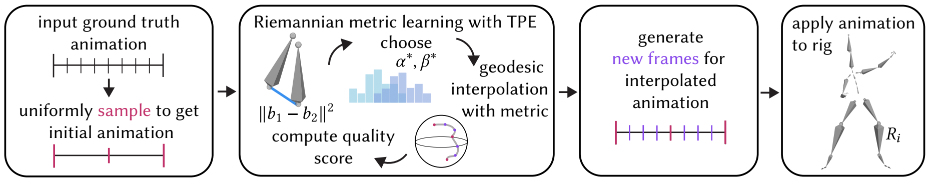

input ground truth animation
uniformly sample to get initial animation
Riemannian metric learning with TPE
choose
α^*, β^*
geodesic interpolation with metric
$\|b_1 - b_2\|^2$
compute quality score
generate new frames for interpolated animation
apply animation to rig
R_i

Fig. 3: Pipeline of our method. Given an animation, we downsample it, and upsample new in-between frames using a search sweep for optimal parameters. Then we apply the animation to the rig and quantitatively and qualitatively analyse the results.

found that restricting to a diagonal form yielded comparable performance. Additional motivation, including computational considerations and software stability, is discussed in the implementation section.

Our metric on $SO(3)^B$ is parameterized by the set: $\{\alpha_1, \beta_1, ..., \alpha_B, \beta_B\}$, written $\{\alpha, \beta\}$ for short. We also add a categorical parameter, called `inv`, which indicates whether whether we propagate the inner-product Z with left or right translations: *i.e.*, whether the resulting metric $<,>$ is left- or right- invariant. This parameterization does not cover every metric on $SO(3)^B$; yet, it encodes a $4B$-dimensional family of metrics where we can perform metric learning.

Geodesic Interpolation Consider a bone b and two frames i, j that are consecutive in the initial animation A_I and $j - i + 1$ frames apart in the ground-truth animation A_G, *i.e.*, $A_I(b, i) = A_G(b, i) = R_i \in SO(3)$ and $A_I(b, j) = A_G(b, j) = R_j \in SO(3)$. Given a metric $<,>$, we compute the geodesic γ on $SO(3)$ such that $\gamma(0) = R_i$ and $\gamma(1) = R_j$ and the energy $E(\gamma)$ measured with $<,>$ is minimal according to the definition of a geodesic. The main challenge is to compute the initial tangent vector $u_0 = \dot{\gamma}(0)$ required to shoot from $\gamma(0)$ to $\gamma(1)$. This requires to numerically invert the Exp map defined in the previous section, *i.e.*, solving the optimization problem:

$$u_0 = \underset{u \in T_{R_i} SO(3)}{\arg\min} \|\text{Exp}_{R_i}(u) - R_j\|^2. \tag{3}$$

The tangent vector u_0 then yields values of A_U between frames i and j as: $A_U(b, t) = \text{Exp}_{R_i}(t.u_0)$ for $t \in [0, 1]$. We observe that we do not have a closed form expression for the interpolating geodesic, which is instead computed via numerical integration and optimization.

Optimization Criteria: Quality Metrics The upsampled animation A_U is obtained by geodesic interpolation, which depends on the invariant Riemannian metric $<,>$ that is itself parameterized by α, β and `inv`. Thus, we write A_U as a function of $\alpha, \beta, \texttt{inv}$: $A_U(\alpha, \beta, \texttt{inv})$. We detail here how we find the optimal parameters $\alpha, \beta, \texttt{inv}$ and thus the optimal Riemannian metric $<,>$ for digital animations, see Figure 3 (center). Consider a quality metric Q that denotes how close the interpolated animation $A_U(\alpha, \beta, \texttt{inv})$ is from the ground truth animation A_G. We get:

$$\alpha^*, \beta^*, \texttt{inv}^* = \underset{\alpha, \beta, \texttt{inv}}{\arg\min}\, Q\left(A_U(\alpha, \beta, \texttt{inv}), A_G\right), \tag{4}$$

for $\alpha, \beta \in (\mathbb{R}_+^*)^B$ and `inv` in {`left`, `right`}. We will experiment with various quality metrics Q within this optimization criterion.

Quality metrics compare the differences between bones in the ground truth animation A_G and the corresponding bones in the upsampled animation A_U. Our first quality metric quantifies the difference in position between two bones' endpoints:

$$Q_{loc}(b_1, b_2) = \|b_1 - b_2\|^2, \tag{5}$$

where b_1 and b_2 are the endpoint positions of bones 1 and 2.

Our second quality metric quantifies the angle difference in rotation between two bones:

$$Q_{rot}(b_1, b_2) = \arccos\left[\frac{tr(b_1 b_2^T) - 1}{2}\right], \tag{6}$$

where in this case b_1 and b_2 are the rotation matrices of bones 1 and 2.

Our third quality metric Q_{hyb} is a weighted sum of $Q_{loc}(b_1, b_2)$ and $Q_{rot}(b_1, b_2)$. Each of these three quality metrics is defined for a given bone of the rig, at a given frame. To get the quality scores Q across bones and frames, we sum across the bones $b = 1, \dots, B$ with or without a weight $w_b > 0$ corresponding to the depth of that bone in the rig, and we average over all frames in the ground truth animation [14]. Thus, the total quality metric between a pose in the ground truth animation A_G and the upsampled animation A_U is:

$$Q = \frac{1}{F}\sum_{t=1}^{F}\sum_{b=1}^{B} w_b \tilde{Q}(A_U(b,t), A_G(b,t)), \tag{7}$$

and $\tilde{Q}$ equal to Q_{loc}, Q_{rot} or Q_{hyb}. The dependency on $\alpha, \beta,$ `inv` is within the bone $b_{t,i}^U$ of the upsampled animation A_U. For all results we use Q_{hyb}.

Optimization Method: Gradient-Free We introduce the optimization method chosen to minimize the criterion of Eq. 4 and learn α^*, β^* and `inv` *. This criterion does not have a closed form as a function of α, β and `inv`. Thus, we cannot compute its gradient, nor leverage any gradient-based optimization methods. Consequently, we propose to rely on a gradient-free optimization methods: the Tree-Structured Parzen Estimator (TPE). Tree-Structured Parzen Estimator algorithm [2] is designed to find parameters that optimize a given criterion whose gradient is not available. TPE is an iterative process that uses history of evaluated parameters $\alpha, \beta,$ `inv` to create a probabilistic model, which is used to suggest the next set of parameters $\alpha, \beta,$ `inv` to evaluate, until the optimal set $\alpha^*, \beta^*,$ `inv`* is reached.

Implementation Our *ground truth animations* are motion capture sequences downloaded from Adobe Mixamo at 30 frames per second [1]. All animations are imported to Blender, which we use to visualize, render, and export animation data [4]. Blender provides functionality to manipulate and access animation data such as rig structures, animation curves, and keyframes using Python scripting, which allows us to automate importing the initial animation, downsampling it, and exporting the downsampled keyframes for processing. After saving the keyframes in NumPy files [7], we load the bone locations and rotation matrices into a script which computes the interpolations. File sizes are calculated as the sum of the sizes (in bytes) of the exported NumPy files.

For cartesian linear interpolation, we linearly interpolate the locations as well as the rotations in the form of component-wise quaternion interpolation. Blender's quaternion interpolation was once implemented this way but was problematic since it can yield invalid (non-unit) quaternions. Blender has since updated to using a version of spherical linear interpolation (slerp), which we also compare to.

Riemannian metric learning with TPE is performed using `HyperOpt` [3] and `Tune` [8]. See the supplementary material for details on the TPE algorithm. During geodesic interpolation on $SO(3)$, we generate new rotation matrices representing the orientation of each bone at a frame using the implementation of invariant Riemannian metrics parameterized by $\alpha, \beta,$ `inv` and available through the Geomstats library [10]. In implementation, we restricted Z to a diagonal form primarily for computational and numerical reasons. Although a full symmetric positive definite matrix could in principle allow finer control by penalizing arbitrary directions, earlier experiments using such matrices did not yield significant improvements in reconstruction quality. Moreover, we encountered numerical instabilities when working with full matrices in Geomstats at the time.

To compute quality metrics, we need to recover the new bone positions b at each frame given the interpolated orientations $R \in SO(3)$ and the position of the root bone. To do so, we start from the root bone of the rig (*e.g.* hips) and traverse the tree breadth first, applying each new rotation to the bones on that "level" of the tree, computing the new positions, iteratively until we have leaf node (*e.g.* fingertips) positions.

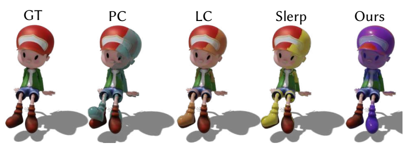
Once we have the interpolated frames for all interpolation schemes also saved in NumPy files, we then load them back into Blender to be applied to copies of the downsampled animation.

3 Results

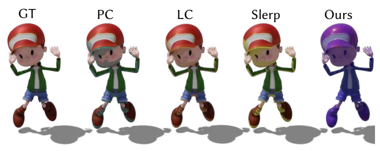
We compare our geodesic interpolations (purple) to the three most commonly used schemes: piecewise constant (PC, teal), linear cartesian (LC, orange), spherical linear (slerp, yellow), on 5 different increasingly complex Mixamo animations: `Pitching`, `Rolling`, `Punching`, `Jumping`, and `Sitting`. Our supplemental video contains the full animations.

Perceptual Accuracy We visualize and qualitatively compare the accuracy of each interpolation scheme. We present this comparison using a sampling rate of $s = 0.3$ in Figs. 4a-4d, while the corresponding figures for other sampling rates can be found in the supplemental materials. Our visualizations show the ground truth animation, with the interpolation methods layered transparently over to highlight where the interpolation deviates from the original.

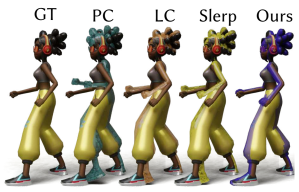
The `Pitching` animation in Fig. 1 has 24 bones and shows our method working with animations with a fixed root node. `Sitting` in Fig. 4a is an example where the fixed node is in the middle of the armature. `Jumping` contains vertical motion and rotations in the legs that are far apart, *i.e.*, differ by a large angle close to pi. `Punching` animation in Fig. 4c shows horizontal translations with contacts. For example, it would



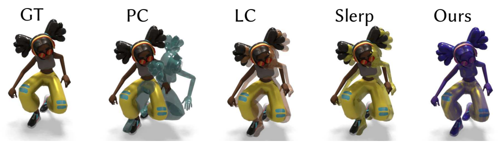
(a)



(b)



(c)



(d)

Fig. 4: Figure 4a shows our geodesic almost perfectly recreating the pose in frame 24 of the `Sitting` animation. Figure 4b shows the entire purple overlay for the `Jumping` animation which indicates a high quality reconstruction. Figure 4c shows extremities like hands are captured more accurately in our method for the `Punching` animation. In Figure 4d, we capture the fast `Rolling` motion in frame 44.

be undesirable for an interpolation to miss frames where her feet touch the floor to create an illusion of floating. Our approach outperforms traditional techniques, as it most accurately interpolates characters within this diversity of animations: displaying a larger purple overlay in Figs. 4a-4b, effectively capturing extremities (hands and feet) in Fig. 4c as well as fast motions in Fig. 4d. The `Rolling` animation is difficult because it has the complexity of all previous animations. Bones rotations are large and flip upside down (see Fig. 4d). In this difficult setting, visual inspection shows that our interpolation performs particularly well.

Quantitative Accuracy and Compression In addition to these perceptual comparisons, we compare the interpolations' accuracies using the weighted error $Q_{hyb} = 0.5\, Q_{loc} + 0.5\, Q_{rot}$ and present it in Fig. 5 for the `Rolling` animation. The supplementary materials show these plots for the 4 other animations. Our approach presents the lowest error just in front of slerp's. Despite the seemingly small quantitative difference between these two, we note that Fig. 4d shows significant perceptual differences. Fig. 5 also allows us to evaluate our method in terms of compression: we require a lower sampling rate s to achieve a given interpolation error (or accuracy). Consequently, this method can decrease the memory required to store animations: Our compressed animation is a factor of s smaller than the ground truth, plus the $B\alpha$ and $B\beta$ float values. The supplemental materials provide additional details on compression and exact file size.

4 Conclusion and Future Work

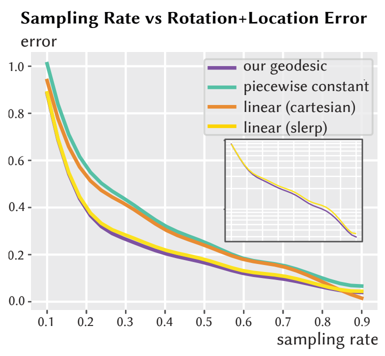


Fig. 5: As the sampling rate for the `Pitching` animation increases, the error metric Q_{hyb} decreases.

We presented a method for animation interpolation using geodesics on Riemannian manifolds where we learn the optimal metric. To our knowledge, this is the first time that Riemannian metric learning is proposed for computer graphics. We hope that these ideas will inspire other applications in this field. We showed that our method interpolates animations with high accuracy (both perceptually and quantitatively) on a variety of different motion capture sequences. Because we are able to accurately represent a high frame rate animation with very few frames, we achieve a compression rate that requires digital animators to pose fewer keyframes during the creation process.

Future work will involve a deeper analysis of the metric parameters α and β to better understand how they influence the perceived qualities of motion. By studying the optimal values α^*, β^* learned across different animations, we hope to uncover patterns that reflect stylistic choices or emotional intent—*e.g.*., whether higher α values in a joint correlate with faster, more expressive movement. These insights could not only offer semantic intuition into motion design but also inform tools that give animators direct, real-time control over interpolation styles. To that end, we plan to integrate our geodesic interpolation framework into animation software, allowing users to experiment interactively with different parameter settings. We are also interested in conducting perceptual studies to evaluate which interpolations are most appealing or expressive to animators and viewers.

One can also explore how choice of keyframes impacts interpolation and compression results. Our experiments uniformly downsample the ground-truth animation. Yet, with an extremely low sampling rate, the downsampled animation consists of very few frames which might not capture all important actions. One can explore how a smart downsampling of the animation improves interpolation quality by ensuring that the most important frames are kept.

5 Acknowledgments

Sarah Kushner and Nina Miolane acknowledge funding from the NSF Career 2240158 and the NSF grant 2134241.

Bibliography

[1] Adobe: Mixamo (2023), URL `https://www.mixamo.com/`

[2] Bergstra, J., Bardenet, R., Bengio, Y., Kégl, B.: Algorithms for hyper-parameter optimization. In: Shawe-Taylor, J., Zemel, R., Bartlett, P., Pereira, F., Weinberger, K. (eds.) Advances in Neural Information Processing Systems, vol. 24, Curran Associates, Inc. (2011)

[3] Bergstra, J., Yamins, D., Cox, D.D., et al.: Hyperopt: A python library for optimizing the hyperparameters of machine learning algorithms (2013)

[4] Community, B.O.: The Free and Open Source 3D Creation Suite. Blender Foundation, Stichting Blender Foundation, Amsterdam (2023), URL `http://www.blender.org`

[5] Guigui, N., Miolane, N., Pennec, X.: Introduction to riemannian geometry and geomtric statistics: from basic theory to implementation with geomstats. Foundations and Trends in Machine Learning (2022)

[6] Haarbach, A., Birdal, T., Ilic, S.: Survey of higher order rigid body motion interpolation methods for keyframe animation and continuous-time trajectory estimation. In: 2018 International Conference on 3D Vision (3DV), pp. 381–389, IEEE (2018)

[7] Harris, C.R.: Array programming with NumPy. Nature **585**(7825), 357–362 (Sep 2020)

[8] Liaw, R., Liang, E., Nishihara, R., Moritz, P., Gonzalez, J.E., Stoica, I.: Tune: A research platform for distributed model selection and training. arXiv preprint arXiv:1807.05118 (2018)

[9] Milnor, J.: Curvatures of left invariant metrics on lie groups (1976)

[10] Miolane, N., Guigui, N., Brigant, A.L., Mathe, J., Hou, B., Thanwerdas, Y., Heyder, S., Peltre, O., Koep, N., Zaatiti, H., Hajri, H., Cabanes, Y., Gerald, T., Chauchat, P., Shewmake, C., Brooks, D., Kainz, B., Donnat, C., Holmes, S., Pennec, X.: Geomstats: A python package for riemannian geometry in machine learning. Journal of Machine Learning Research (2020)

[11] Oreshkin, B.N., Valkanas, A., Harvey, F.G., Ménard, L.S., Bocquelet, F., Coates, M.J.: Motion in-betweening via deep δ-interpolator. IEEE Transactions on Visualization and Computer Graphics pp. 1–12 (2023)

[12] Shoemake, K.: Animating rotation with quaternion curves. In: Proceedings of the 12th annual conference on Computer graphics and interactive techniques, pp. 245–254 (1985)

[13] Shoemake, K.: Quaternion calculus and fast animation, computer animation: 3-d motion specification and control. Siggraph (1987)

[14] Wang, J., Tan, S., Zhen, X., Xu, S., Zheng, F., He, Z., Shao, L.: Deep 3d human pose estimation: A review. Computer Vision and Image Understanding **210**, 103225 (2021)

[15] Zhang, X., van de Panne, M.: Data-driven autocompletion for keyframe animation. In: Proceedings of the 11th Annual International Conference on Motion, Interaction, and Games, pp. 1–11 (2018)