DOI: 10.1111/cgf.14632
ACM SIGGRAPH / Eurographics Symposium on Computer Animation 2022
D. L. Michels and S. Pirk
(Guest Editors)

Volume 41 (2022), Number 8

Pose Representations for Deep Skeletal Animation

N. Andreou[†][1,2], A. Aristidou[‡][1,2], and Y. Chrysanthou[1,2]

[1]University of Cyprus, Nicosia, Cyprus, [2]CYENS Centre of Excellence, Nicosia, Cyprus

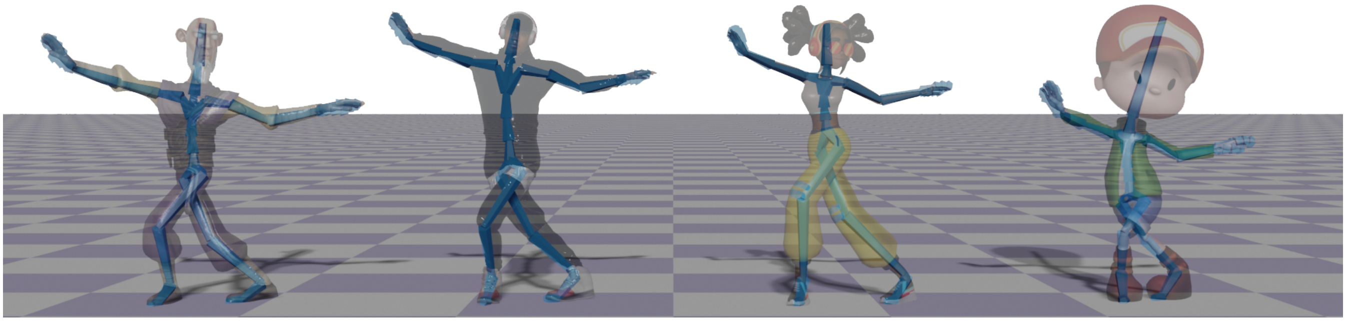

Figure 1: *A fundamental component of motion modeling with deep learning is the pose parameterization. A suitable parameterization is one that holistically encodes the rotational and positional components. The dual quaternion formulation proposed in this work can encode these two components enabling a rich encoding that implicitly preserves the nuances and subtle variations in the motion of different characters.*

Abstract

Data-driven skeletal animation relies on the existence of a suitable learning scheme, which can capture the rich context of motion. However, commonly used motion representations often fail to accurately encode the full articulation of motion, or present artifacts. In this work, we address the fundamental problem of finding a robust pose representation for motion, suitable for deep skeletal animation, one that can better constrain poses and faithfully capture nuances correlated with skeletal characteristics. Our representation is based on dual quaternions, the mathematical abstractions with well-defined operations, which simultaneously encode rotational and positional orientation, enabling a rich encoding, centered around the root. We demonstrate that our representation overcomes common motion artifacts, and assess its performance compared to other popular representations. We conduct an ablation study to evaluate the impact of various losses that can be incorporated during learning. Leveraging the fact that our representation implicitly encodes skeletal motion attributes, we train a network on a dataset comprising of skeletons with different proportions, without the need to retarget them first to a universal skeleton, which causes subtle motion elements to be missed. Qualitative results demonstrate the usefulness of the parameterization in skeleton-specific synthesis.

CCS Concepts

• ***Computing methodologies*** → ***Motion processing; Animation;*** *Learning paradigms;*

1. Introduction

Motion modeling is a fundamental task on the cornerstone of computer graphics and vision. A well established motion representation enables the synthesis of realistic human motion, satisfying constraints such as motion smoothness, continuity, and naturalism. Yet, computationally synthesizing human motion remains challenging, since motion is a highly stochastic process, governed by both intrinsic and extrinsic factors of which a harmonic balance should be achieved. Recent advances of machine learning and neural networks, along with the release of high-quality, large, diverse, and highly realistic motion capture databases, have shown promising results in reconstructing human motion from videos with high expressiveness, in synthesizing arbitrary movement with realism, and controlling articulated characters.

The quality of the generated motion is highly influenced by two main factors: (a) the network architecture/design and learn-

[†] nefeliandreou@outlook.com
[‡] a.aristidou@ieee.org

ing scheme, and (b) the input pose representation. Over the last few years an enormous amount of research has been devoted to the architecture design and learning schema [HSK16; FLFM15; BKL17]. However, less effort has been dedicated to the motion parameterization, even though it is of equal significance [ZBL*19; XL20; PGG*20]. Previous research has found that keeping the architecture fixed and experimenting with different representations affects the quality of the results [PGA18; SAA*20]. In addition, it has been observed that common rotational representations still occasionally produce large errors when used for rotation regression tasks. This becomes more evident when the mapping between the network's embedding space and the original data is discontinuous, thus making it hard for the network to learn properly [ZBL*19]. As highlighted in a recent survey by Mourot *et al.* [MHL*21], a good pose representation for human motion modeling should accurately reflect the visual outcome while being suitable for optimisation (avoiding the error accumulation along the skeleton). It should capture both spatial and temporal correlations, leading to the extraction of informative patterns that serve a variety of tasks, such as rigging and skinning. The learnt patterns lay the grounds for inference, and determine the generalization ability of the model to different motions and skeletons.

In early deep neural approaches, pose is represented using the Cartesian locations of joints, to ensure the continuity of the reconstructed motion [HSK16; HKS17; CHS*18; ZLX*18]. However, joint locations can only describe a small set of the full human motion articulation, resulting in ambiguities in motion reconstruction, related to the rotation on the roll axis. In addition, it is required to enforce priors (e.g., an Inverse Kinematics solver) about bone lengths to limit potential rig violations. In some later work, motion is represented as local joint rotations using 3D or 4D representations, such as Euler angles, axis-angle (exponential maps), or quaternions [ZBL*19]. The space of the 3D/4D rotational representations, though, is not continuous causing singularities. Prediction error in crucial joints of the rig hierarchy is not properly encoded, resulting in error accentuation as we move to the end-effectors. To benefit from the trade-off between rotational and positional representations, recent works, such as [PGA18; ALL*20; SAA*20], model motion using joint local quaternions and employ a forward kinematics (FK) layer to recover or revise the corresponding joint positions. However, networks trained in such a way only rely on the positional information for supervision, hindering the rotational, which is a particular component in animation.

In addition, when training various models for motion related tasks, previous works suggest to retarget all motions in the database into a universal skeleton configuration, in order to allow for generalizable character and setup agnostic experiments. This is because databases consist of motion captured using actors, where each performer's body structure varies in terms of bone lengths and proportions, shaping their motion traits. However, such a requirement limits the ability of networks to support expressive motion attributes and nuances that were originally performed by actors with different skeletal structure, and were lost in the retargeting process.

In this paper, we tackle the fundamental problem of representing poses, using a hierarchy-aware representation that uses dual quaternions as the mathematical framework. The main premise of our method is that dual quaternions allow us to define poses in a root-centered manner, implicitly encoding skeletal information that can be leveraged by the network. Dual quaternions provide a unified, elegant, and compact representation that incorporates rotational and translational information in the form of orthogonal quaternions [Ken12]. They carefully encode the aforementioned elements of human motion in one component, and parse it as an input to the network, so that the model benefits from both parameters, thus leading to more constrained learning. In this way we can directly extract the positional information from the representation avoiding the need for external kinematic operations such as FK. Then we can use this information as a loss, similar to recent works e.g., [SAA*20; PGA18; HYNP20].

Our main contribution is the exploitation of a hierarchy-aware pose motion representation based on the properties of dual quaternions, as well as the assessment of currently used inputs for deep skeletal animation. The proposed representation allows to infer both rotational and positional information directly, while it encodes the correlations between joints and limbs along the structure of the rig. We integrate several constraints, in the form of training losses, to better penalize the errors in crucial joints of the hierarchy; the losses operate directly on the representation, eliminating the need for external kinematic calculations during training. To the best of our knowledge, the dual quaternion properties have not been explored before in the context of human motion modeling with deep neural frameworks. In particular, we hypothesize that learning from such a representation may enable the network to better exploit skeletal-nuances of motion, enabling the generation of stable motion on with less training. This representation can be used for a variety of tasks, such as motion prediction or synthesis, motion retargeting, motion reconstruction, and it is not tuned towards specific motion patterns. We demonstrate the effectiveness and practicality of the proposed representation using two well-known network architectures which focus on the fundamental application of motion synthesis: the auto-conditioned Recurrent Neural Network (acRNN) of Zhou *et al.* [ZLX*18], and the QuaterNet developed by Pavllo *et al.* [PGA18]. In particular, we perform both short-term prediction and long-term synthesis, we examine the relevance of the proposed losses for each architecture through an ablation study, and show that depending on the task, each loss serves a purpose. Finally, we show that the proposed parameterization can be used for skeleton-specific synthesis, where the characters during training can have varying proportions (see Figure 1).

2. Related Work

In recent years, various deep learning models have been developed to accomplish complex tasks in character animation, including motion reconstruction [CHS*18; SAA*20], action recognition [DWL15], motion synthesis [HSK16; HKS17; ZLX*18], prediction [FLFM15; WGLM18], style transfer [AWL*20; DAS*20; SCNW19], motion retargeting [ALL*20; DHS*19] etc. Over the last few years, a number of different architectures and learning schemes have been introduced to model human articulation, including convolutional [HSK16; BBKK17], recurrent [FLFM15; ZLX*18; MBR17; JZSS16; WHSZ21], phase-functioned [HKS17; SZKZ20], or adversarial [BKL17; WGLM18; WCX21] networks. Furthermore, generative models based on normalizing flows [HAB20] and VAEs [LZCV20] are becoming popular since they can generate diverse motions, and have proven to be effective for a variety of tasks such as control or planning. However, despite the progress made on the design of advanced networks to model the high frequencies of motion, the proposed architectures still depend on simplified pose representations (e.g., joint locations or local rotations) which cannot encode the full articulation of human motion. Even with abundant weights and extensive training, they still occasionally produce big errors. In previous work, the efficiency of deep learning approaches has been found to be highly dependent upon the quality of the dataset and the representation used [XL20; ZBL*19; MYGY19], due to the fact that the network learns motion patterns from the data. A rich encoding of motion, based on a concrete mathematical model, encourages the network to properly disentangle complex characteristics of motion which are present in the data.

2.1. Motion parameterization

Human motion is often modelled as a sequence of skeletal states expressed in terms of the 3D orientation of the bones (angular representations) or the 3D coordinates of joints (positional representations). Each parameterization has its own benefits and limitations, which we briefly outline below. As pointed out in a recent survey, the parameterization of input and output guide the network to retain specific features [MHL*21]. We argue that a representation that unifies positional and rotational information can achieve the maximum potential in character animation.

Positional Pose Representations: In early machine learning approaches, large amounts of work have been devoted to the development of deep learning networks that use 3D joint positions [HSK16; FLFM15; ZLX*18], since minimizing the 3D position errors ensures prediction of correct joint locations, and maintains that mispredictions on crucial joints are taken into consideration. Positional data, however, comes at several costs. First, for a pair of fixed consecutive 3D positions there exist multiple limb rotations, which can be recovered using Inverse Kinematics (IK). This leads to ambiguities on the rotation of each limb on its roll axis. In addition, such representation lacks the benefit of the parameterized skeleton. If the generated motion is to be applied on a different skeleton, post-processing is required to secure that the skeleton constraints are satisfied, i.e., bone constraints or motions within the articulation range. Consequently, positional data fails to describe the full range of human motion articulation, thus does not suffice to uniquely recover human characters. Considering the aforementioned reasons, positional representations are less suitable for graphics and animation applications.

Angular Pose Representations: Rotational representations belong to the rotation group $SO(3)$, and can be expressed using various parameterizations e.g. Euler angles, axis-angle, quaternions, to mention a few. The main benefit of rotational representations over positional is that they allow for a parameterized skeleton [ALL*20; SAA*20], thus avoid prediction errors related to bone stretching or motion outside the articulation range. Euler angles are the most intuitive representation, and have been widely explored in previous research [ACH*18; ZBL*19]. They can be used to describe the orientation of a rigid body as successive rotations relative to a fixed coordinate system x, y, z. However, Euler angles come with many limitations: they are discontinuous and non-unique in the sense that for a particular orientation, θ and $\theta + 2k\pi$ for $k \in \mathbb{Z}$ represents the exact same rotation, causing learning problems. In addition, they are prone to the Gimbal lock effect. To mitigate the issues caused by Euler angles, exponential maps (axis-angle) representations have been adopted [MBR17; WGLM18; HAB20]. Axis-angle representations have a major drawback, that is they cannot express composition of rotations [Gra98], thus cannot be used for hierarchical modeling. Another way of overcoming the discontinuity issue of Euler angles is to represent each angle θ using a 2D vector, $[\cos\theta, \sin\theta]$, or equivalently a unit complex number $a + bi$. To perform smooth regression the constraint $a^2 + b^2 = 1$ should be maintained. This approach, doubles the number of parameters that need to be predicted, and does not benefit from incorporating positional information.

On the other hand, many works use unit quaternions to encode joint rotations when training deep networks in character animation, demonstrating satisfactory results in several tasks, such as motion reconstruction from video [SAA*20], style transfer [DAP*17], motion retargeting [VYCL18], motion synthesis and prediction [PGA18; AYA*22]. However, despite the growing popularity of unit quaternions, Zhou *et al.* [ZBL*19] and Xiang *et al.* [XL20] have recently demonstrated experimentally that the quaternion's space is still not continuous. More specifically, they showed that motion continuity cannot be achieved for a space with less than four dimensions ($\mathbb{R}^4$), and proved that neural networks can learn better from continuous representations. Zhou *et al.* [ZBL*19] pointed out that a rotation representation can be made continuous using the identity mapping, which would result in $n \times n$ sized matrices (n is the dimension of rotations), which may not only be excessive, but also still require orthogonalization in mapping from the representation to the original space. Thus, they proposed to perform an orthogonalization in the representation itself, resulting in a 6 dimensional parameterization (ortho6D). This representation has been widely adopted in recent works [ZSKS18; LZCV20; PBV21]. However, it does not encode positional information, while a hierarchical modeling with FK would require conversion to transformation matrices, in order to perform the sequential encoding of orientations along the rig.

In recent works, networks which rely on rotational parameterizations often integrate an additional FK layer, and are paired with a positional loss to further constrain motion [PGA18; AWL*20]. It has been observed that this is a necessary step for designing accurate learning schemes since small prediction errors in certain joint's rotations fail to be properly encoded in losses which average local rotation errors, yet drastically impact the positional error. In practice, the prediction error is accumulated along the hierarchy, as illustrated in Figure 2.

Hybrid Representations: Several works proposed the use of hybrid representations in order to get the best of both worlds. It has been demonstrated that additional information in the parameterization, such as velocities, benefits the learning process. Some hybrid representations include joint orientations with joint positions [LLL18], positions and velocities [HKS17; ZSKS18; SZKZ20; SZKS19], and joint positions with linear and angular ve-

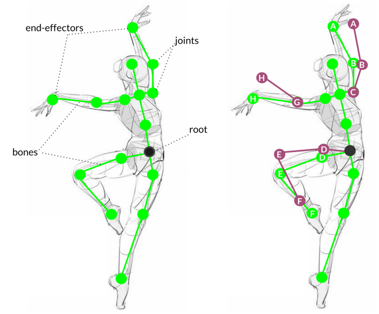

Figure 2: *Error accumulation along kinematic chain. We can see that a small local rotation error in either joint C or D drastically affects the position of joints A,B,E,F. Errors on joint G are less crucial as it is not at the base of the hierarchical chain and its transformation only affects joint H. Taking a weighted average of the local joint rotation errors does not reflect that a wrong prediction in joint D has a larger impact on the resulting motion, than joint G.*

locities [HKPP20; LYC*20]. In contrast, our parameterization is equally expressive and combines the benefits of positional and angular information in a unified entity, but does not include redundant information.

2.2. Dual Quaternions in Character Animation

Dual quaternions are powerful algebraic constructs which have been successfully used in a number of domains. Their power lies in the fact that they can efficiently represent rotations and translations in a unified way, while their well-defined algebraic operations such as multiplication, make them appealing for hierarchical modeling. They have been used extensively in computer vision and robotics [GA98; Dan99; TRA11], and more recently have been adopted in character animation. For instance, Kavan *et al.* [KCŽO08] apply dual quaternions for skeletal skinning and achieve faster execution times than more traditional skinning methods. Kenwright [Ken12] demonstrates the superiority of dual quaternions over traditionally used transformation matrices, while Vemulapalli *et al.* [VAC16] examine various skeletal representations, for the purpose of human action recognition. Comparisons between representations that use joint rotations and joint translations separately, indicate that the newly proposed family of skeletal representations achieve higher performance for the task of action recognition on a wide range of datasets. It is worth noting that the work of Kenwright [Ken12] demonstrates that the well formulated algebraic operations of dual quaternion make them ideal candidates to represent rigid transformations of hierarchical chains, with both performance gains compared to rotation matrices and computational benefits when calculating angular and linear differences.

In this paper, dual quaternions are used for the first time to parameterize skeletal motion in a deep learning framework. In contrast to previous works that have designed a neural network architecture where each node corresponds to a dual quaternion [SPN21], we use dual quaternions to model complex movement of the skeleton which has multiple connected joints. To handle the skeletal structure and introduce an inductive bias in the learning process, we express the representation of each joint in current coordinates. Thus, in our work, dual quaternions encode the hierarchical nature of a rig, enabling the integration of losses that take into consideration the rotation and displacement components. This formulation assists the learning, leading to more realistic synthesis.

3. Mathematical Representation

A dual quaternion $\underline{\mathbf{q}}$ can be represented as an ensemble of ordinary quaternions $\mathbf{q}_r$ and $\mathbf{q}_d$, in the form $\mathbf{q}_r + \mathbf{q}_d\varepsilon$, where ε is the dual unit, satisfying the relation $\varepsilon^2 = 0$. The first quaternion describes the rotation. The second quaternion, $\mathbf{q}_d$ encodes translational information which we will define in Section 5. Each quaternion is characterized by 4 DOF making the dual quaternion an 8D representation, which can be considered as an 8-tuple of real values [KŽ05]. A unit dual quaternion can be interpreted as a manifold in the 8-dimensional Euclidean space [KCŽO08].

Dual quaternions allow for convenient mappings from and to other representations which are currently used in the literature, allowing for effortless integration into current architectures. To reduce ambiguity we establish the following notation which will be used throughout this paper:

$\mathbf{q}$ quaternion	$\underline{\mathbf{q}}$ dual quaternion
$\hat{\mathbf{q}}$ unit quaternion	$\hat{\underline{\mathbf{q}}}$ unit dual quaternion
$\mathbf{q}^*$ quaternion conjugate	$\underline{\mathbf{q}}^*$ dual quaternion conjugate

3.1. Basic Algebraic Properties

Dual quaternions have well-defined algebraic operations which we list below. To put in context, in the following sections we elaborate on the construction of dual quaternions from other representations, that is, dual quaternion to and from rotation and 3D position.

Multiplication: Suppose that $\underline{\mathbf{q}}_1, \underline{\mathbf{q}}_2$ are dual quaternions of the form $\underline{\mathbf{q}}_1 = \mathbf{q}_{r1} + \mathbf{q}_{d1}\varepsilon$, $\underline{\mathbf{q}}_2 = \mathbf{q}_{r2} + \mathbf{q}_{d2}\varepsilon$ then the multiplication operation is defined as:

$$\underline{\mathbf{q}}_1\underline{\mathbf{q}}_2 = \mathbf{q}_{r1}\mathbf{q}_{r2} + (\mathbf{q}_{r1}\mathbf{q}_{d2} + \mathbf{q}_{d1}\mathbf{q}_{r2})\varepsilon. \tag{1}$$

Conjugate: The conjugate of $\underline{\mathbf{q}}$, i.e. $\underline{\mathbf{q}}^* = \mathbf{q}_r^* + \mathbf{q}_d^*\varepsilon$

Magnitude: The magnitude of a dual quaternion is given by:

$$||\underline{\mathbf{q}}|| = \sqrt{\underline{\mathbf{q}}\underline{\mathbf{q}}^*} = ||\mathbf{q_r}|| + \varepsilon\frac{<\mathbf{q}_r, \mathbf{q}_d>}{||\mathbf{q}_r||}. \tag{2}$$

Here $<a,b>$ denotes the dot product between the real valued vectors a and b.

Unitary condition: A unit dual quaternion should satisfy the following two conditions:

1. the real part $\mathbf{q}_r$ must be a unit quaternion, i.e. $||\mathbf{q}_r|| = 1$
2. the real part must be orthogonal to the dual part, i.e.

$$\mathbf{q}_r^*\mathbf{q}_d + \mathbf{q}_d^*\mathbf{q}_r = 0 \tag{3}$$

For long multiplication chains, it might be necessary to renormalize a dual quaternion to mend drift. To make a dual quaternion unit, we can divide it by its magnitude.

Inverse: The inverse of a unit dual quaternion is equal to its conjugate, i.e. $\underline{\hat{\mathbf{q}}}^{\mathbf{-1}} = \underline{\hat{\mathbf{q}}}^*$.

Dual quaternions can represent transformations. A pure rotation can be represented by a unit dual quaternion with zero dual part, namely $\underline{\mathbf{q}} = (w_r + x_r\mathbf{i} + y_r\mathbf{j} + z_r\mathbf{k}) + \varepsilon(0 + 0\mathbf{i} + 0\mathbf{j} + 0\mathbf{k})$, whereas a pure displacement by x, y, z units on each of the unit axes respectively can be represented by a dual quaternion with the identity quaternion as real part, i.e. $\underline{\mathbf{q}} = (1 + 0\mathbf{i} + 0\mathbf{j} + 0\mathbf{k}) + \varepsilon(0 + \frac{x}{2}\mathbf{i} + \frac{y}{2}\mathbf{j} + \frac{z}{2}\mathbf{k})$. The operation $\underline{\hat{\mathbf{q}}}\mathbf{p}\underline{\hat{\mathbf{q}}}^*$ can be used to transform a point $\mathbf{p}$ inserted in a quaternion.

3.2. Dual Quaternion from 3D rotation and 3D position

The rotation quaternion can be defined as the combination of Euler rotations along each of the x, y, z axes. The unit quaternion that corresponds to a rotation of angle θ around axis $\mathbf{n}$ is given by:

$$\hat{\mathbf{q}}_r = (\cos\frac{\theta}{2}, \mathbf{n}\sin\frac{\theta}{2}) \tag{4}$$

Therefore, a rotation of α radians along the x-axis corresponds to the quaternion $\hat{\mathbf{q}}_{\mathbf{rx}} = \cos\frac{\alpha}{2} + \sin\frac{\alpha}{2}\mathbf{i} + 0\mathbf{j} + 0\mathbf{k}$. Similarly for rotation β along the y-axis and γ along the z-axis the corresponding quaternions are $\hat{\mathbf{q}}_{\mathbf{ry}} = \cos\frac{\alpha}{2} + 0\mathbf{i} + \sin\frac{\beta}{2}\mathbf{j} + 0\mathbf{k}$ and $\hat{\mathbf{q}}_{\mathbf{rz}} = \cos\frac{\alpha}{2} + 0\mathbf{i} + 0\mathbf{j} + \sin\frac{\gamma}{2}\mathbf{k}$. Then, we multiply the quaternions based on the order of rotation. For example, if the order of rotation is zyx, the quaternion $\hat{\mathbf{q}}_{\mathbf{r}}$ describing the resulting orientation can be obtained by:

$$\hat{\mathbf{q}}_r = \hat{\mathbf{q}}_{rz}\hat{\mathbf{q}}_{ry}\hat{\mathbf{q}}_{rx}. \tag{5}$$

The dual part, $\mathbf{q}_d$ can be constructed as:

$$\mathbf{q}_d = \frac{1}{2}\mathbf{q}_t\hat{\mathbf{q}}_r \tag{6}$$

where $\mathbf{q}_t = 0 + x\mathbf{i} + y\mathbf{j} + z\mathbf{k}$, where x, y, z denotes 3D displacement in Cartesian coordinates.

3.3. Recovering the 3D rotation and translation in Cartesian coordinates

Apart from constructing the dual quaternions, we make sure that we can recover the rotational component in Euler angles, which we then use to construct a BVH file. Given a unit dual quaternion $\underline{\hat{\mathbf{q}}}$, we can recover the rotation from the rotational part. For example, assuming that the order of rotation is zyx we can obtain angles α, β and γ, which correspond to rotations along the x, y, z axes by:

$$\begin{aligned} \alpha &= \arctan\left(\frac{2(w_r x_r + y_r z_r)}{1 - 2(x_r^2 + y_r^2)}\right) \\ \beta &= \arcsin\left(2(w_r y_r - z_r x_r)\right) \\ \gamma &= \arctan\left(\frac{2(w_r z_r + x_r y_r)}{1 - 2(y_r^2 + z_r^2)}\right) \end{aligned} \tag{7}$$

The translation component in Cartesian coordinates can be obtained using the displacement quaternion,

$$2\mathbf{q}_d\hat{\mathbf{q}}_r^*, \tag{8}$$

where $\hat{\mathbf{q}}_r^*$ denotes the unit quaternion conjugate of the rotational quaternion. The Cartesian coordinates x, y, z are the coefficients of the unit vectors $\mathbf{i}, \mathbf{j}, \mathbf{k}$.

For more details on quaternions and dual quaternions the reader is referred to the works of Dam *et al.* [DKL98] or Kenwright [Ken12].

4. Motion Representation

In skeletal animation, the rig is represented as a hierarchical graph consisting of interconnected bones (edges), usually referred to as joints and end-effectors. Joints are connected with a parent/child relation. Each joint can be subject to an affine transformation, i.e. translation, rotation and scaling. Motion is created by capturing the transformations of the set of joints for a number of points in time.

In previous work, and in common motion file formats (e.g., BVH), the orientation of each joint is expressed based on the coordinate axes of its parent, i.e., in local coordinates. Since the human body is characterized by strong joint hierarchies, it can be treated as a kinematic chain starting from the root. In fact, it has been found that the use of some sort of relative coordinates improves the stability of the network and makes the learnt frames reusable in the 3D space [VAC16]. In this paper, we adopt a similar configuration: we treat the hip as the root joint and express each joint's position and rotation according to the root. Thus, we predict the orientation and displacement of each joint with respect to the root, which we refer to as *current coordinate system*. We model the root displacement as a separate component. We can express the current transformation of the root joint using local homogeneous coordinates of the form:

$$M_{curr,root} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & 0 \\ r_{21} & r_{22} & r_{23} & 0 \\ r_{31} & r_{32} & r_{33} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{9}$$

Then following the tree hierarchy and using the local homogeneous coordinates of each joint j,

$$M_{loc,j} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & \text{offset}_x \\ r_{21} & r_{22} & r_{23} & \text{offset}_y \\ r_{31} & r_{32} & r_{33} & \text{offset}_z \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{10}$$

we can compute the current homogeneous representation for each joint using:

$$M_{curr,j} = M_{curr,(j-1)} \times M_{loc,j} \tag{11}$$

Obtaining the local rotations w.r.t. each joint's parent in the tree architecture, as well as the offsets is straightforward when animation files are used. Intuitively, we can recover the local rotation of joint j using the inverse procedure:

$$M_{loc,j} = M^{-1}_{curr,(j-1)}M_{curr,j} \tag{12}$$

This representation allows us to proceed similarly to traditional techniques, in order to produce the animation in the desired format.

Joint current configuration offers many advantages over the commonly used local representation. That is because during inference we estimate the orientation of each joint independently - relative to a root joint - making our predictions less vulnerable to accumulated errors from local orientations. This process is illustrated

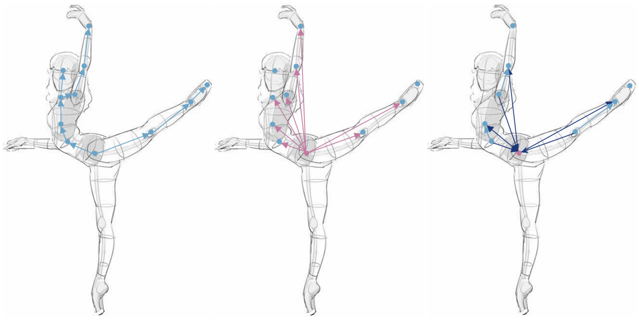

Figure 3: *From left to right: local coordinate system, current coordinate system (purple), the procedure of retrieving the local orientation given the current orientations (dark blue). In our representation, we encode the rotation and translation components in current coordinates, i.e. with respect to the root joint.*

in Figure 3. Our encoding, which is untangled in the next section, handily allows us to recover local coordinates and orientations.

5. Dual Quaternion Motion Representation

An ideal representation would be one which incorporates prior skeletal information, and allows the extraction of precise information on the orientation of each joint. The use of transformation matrices would result in a 4×4 representation for each joint of the skeleton, leading to a high-dimensional representation which may contain excessive information. Instead, such information can be encoded in a dual quaternion.

5.1. Dual Quaternion Encoding

Setting the dual quaternion for the root equal to $\underline{\mathbf{q}} = (w_r + x_r\mathbf{i} + y_r\mathbf{j} + z_r\mathbf{k}) + \varepsilon(0 + 0\mathbf{i} + 0\mathbf{j} + 0\mathbf{k})$ and the dual quaternion of each joint as $\underline{\mathbf{q}} = (w_r + x_r\mathbf{i} + y_r\mathbf{j} + z_r\mathbf{k}) + \varepsilon(w_d + x_d\mathbf{i} + y_d\mathbf{j} + z_d\mathbf{k})$ we can construct the hierarchical representation using the dual quaternion operations, equivalent to those of transformation matrices. The rotation coefficients w_r, x_r, y_r, z_r can be obtained by conversion of local Euler angles to quaternions (see Eq. 7), while the displacement coefficients w_d, x_d, y_d, z_d can be obtained using Eq. 6.

Dual quaternions, just like ordinary quaternions, are known to exhibit the antipodal property. That is, $\hat{\underline{\mathbf{q}}}$ and $-\hat{\underline{\mathbf{q}}}$ represent the same rigid transformation. We tackle this phenomenon by adopting, in preprocessing, the technique employed in Kavan *et al.* [KCŽO08] and Pavllo *et al.* [PGA18]. Among $\hat{\underline{\mathbf{q}}}$ and $-\hat{\underline{\mathbf{q}}}$, we choose the representation with the lowest Euclidean distance from the representation of the previous frame. We apply this step for both dual quaternions as well as quaternions. As shown in Figure 4, bypassing this step led to uneven interpolation, of the form of instantaneous flickering and global shaking in the generated motion. To make sure that the generated dual quaternions remain meaningful and retain their dual nature, we explicitly normalize them using the following formula:

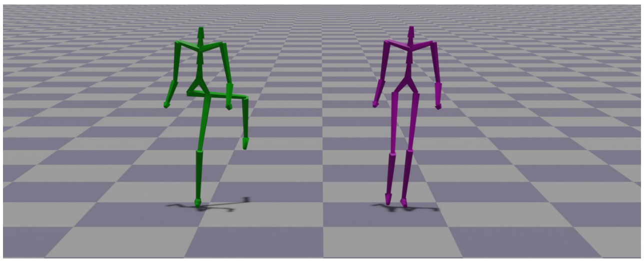

Figure 4: *Abnormal motion generated with data for which we do not correct for the antipodal property during preprocessing (green) compared to the expected motion (purple).*

$$\begin{aligned}\hat{\underline{\mathbf{q}}} = \frac{\underline{\mathbf{q}}}{||\underline{\mathbf{q}}||} &= \frac{\mathbf{q}_r}{||\mathbf{q}_r||} + \varepsilon\left[\frac{\mathbf{q}_d}{||\mathbf{q}_r||} - \frac{\mathbf{q}_r}{||\mathbf{q}_r||}\frac{<\mathbf{q}_r, \mathbf{q}_d>}{||\mathbf{q}_r||^2}\right] \\ &= \hat{\mathbf{q}}_r + \varepsilon\left[\frac{\mathbf{q}_d}{||\mathbf{q}_r||} - \hat{\mathbf{q}}_r\frac{<\mathbf{q}_r, \mathbf{q}_d>}{||\mathbf{q}_r||^2}\right] \end{aligned} \tag{13}$$

5.2. Dual Quaternion Learning

Dual quaternions not only form a unified and compact representation, but also eliminate the need for a forward kinematics layer which is commonly used to recover positions and employ a corresponding loss [PGA18; ZBL*19; SAA*20]. We have carefully defined the dual component to encode the 3D orientation of each joint relative to the root (current coordinates), and incorporated the following losses:

1. If we consider a dual quaternion as an 8-dimensional vector, then we can employ the Mean Squared Error (MSE) between the predicted and generated dual quaternions, given by:
$$\mathcal{L}_{\text{mse}} = ||\mathbf{dq} - \tilde{\mathbf{dq}}||_2^2 \tag{14}$$
where $\mathbf{dq}, \tilde{\mathbf{dq}} \in \mathbb{R}^8$ denote the predicted and true dual quaternion for one joint.
2. **Rotational Loss** is the error between predicted and ground-truth rotations. The loss can be calculated in Euler space or quaternion space, similar to Kavan *et al.* [KCŽO08]. We can measure this loss on the local rotations. To recover the local rotations we perform the relevant operations along the hierarchy to recover the local dual quaternions, and then extract the local quaternions. We can measure the error using the following equation:
$$\mathcal{L}_{\text{quat}} = 1 - \mathbf{q} \cdot \tilde{\mathbf{q}} \tag{15}$$
where $\cdot$ denotes the dot product between ground-truth and predicted quaternions $\mathbf{q}$ and $\tilde{\mathbf{q}}$, respectively.
3. **Positional Loss:** The main benefit of the dual quaternion representation is the capability to directly extract the local position of each joint ($\mathbf{p} \in \mathbb{R}^3$). Since we have adopted the current coordinate system, and utilizing the algebra of dual quaternions we can get the position of each joint using Eq. 8.For each joint the loss $\mathcal{L}_{\text{pos}}$ is given by computing the Euclidean distance between predicted and ground-truth positions , i.e. $\mathcal{L}_{\text{pos}} = ||\mathbf{p} - \tilde{\mathbf{p}}||_2$ where $\tilde{\mathbf{p}}$ is the predicted joint positions.

4. **Offset Constraint:** To maintain that no skeletal violations occur, we explicitly pose skeletal restrictions on the joint offsets ($\mathcal{L}_{\text{offset}}$). This can be achieved by recovering the offset of each joint as the translation encoded in a local dual quaternion. To get that information from our representation we first convert the current dual quaternions to local using 12 and then extract the positional information (which corresponds to the offset $\mathbf{o} \in \mathbb{R}^3$ in the local coordinate system) using Equation 8. The loss is measured as the Euclidean distance between the predicted and ground truth offset, i.e. $\mathcal{L}_{\text{offset}} = \|\mathbf{o} - \tilde{\mathbf{o}}\|_2$ where $\tilde{\mathbf{o}}$ is offset extracted from the predictions.
5. **Penalty Loss (Regularization Term):** To stabilize the learning process we include a regularisation term in the loss ($\mathcal{L}_{\text{reg}}$), responsible for preserving the unity conditions of the dual quaternion (similarly applied for quaternions). It is calculated prior to normalization of the generated frames to reflect the fact that only unit dual quaternions represent a valid transformation. This loss is based on the unitary conditions (Equation 3), i.e.

$$\begin{aligned}\mathcal{L}_{\text{reg}} = (w_r^2 + x_r^2 + y_r^2 + z_r^2 - 1)^2 \\ + (w_r w_d + x_r x_d + y_r y_d + z_r z_d)^2\end{aligned} \tag{16}$$

6. Experimental Setup

In this section, we explain the experimental setup used to evaluate the performance of our pose representation. In our experiments we utilize two popular architectures, the acRNN by Zhou *et al.* [ZLX*18], and QuaterNet by Pavllo *et al.* [PGA18]; we chose these networkssince they are simple recurrent networks which process sequential data, but do not involve complex structures and inductive biases which might influence the performance. The proposed representation encourages the learning of both spatial as well as temporal correlations in recurrent architectures, i.e. while the RNN focuses on the temporal context, the representation implicitly encodes the spatial context. Keeping the networks fixed, we assess the performance of different pose representations. We design our experiments around the prediction task, since we want to assess the quality of the pose representation, without influences from other controls such as end-effector constraints, terrain information, etc. It is worth mentioning that QuaterNet, apart from motion prediction, allows for motion synthesis given a trajectory control. These architectures allow us to effectively show that the examined hierarchy-aware encoding can not only replace the FK layer, but also lead to more stable long-term synthesis via the encoding of prior skeletal information in the representation itself. It is important to note that our representation is model-agnostic, and can be incorporated in other architectures, assuming that the kinematic tree (hierarchy and offsets) are known.

Our experiments have been conducted on a PC with NVIDIA GeForce 2080 Ti GPU, and Intel Core i9-9980HK at 2.4 GHz CPU with 32GB RAM. The deep learning frameworks used are implemented using PyTorch. We used data taken from the Carnegie Mellon University motion capture database [CMU21] (subjects 61, 86) for the acRNN network. Motion files were originally captured at 120 frames per second, but were subsampled to 30 frames per second without much loss of temporal information. We split the dataset in training, validation, and testing sets using the 75/10/15 % configuration. The exact details, as well as the weights used for different losses, can be found in the Appendix. For the QuaterNet we use the exact same dataset and configuration as in the original paper. Training the acRNN network takes about 6.5 hours (around 1500 epochs) when using positional input, 9 hours when the quaternions, quaternions-positions and ortho6D are used, and 11 hours for our representation and ortho6D-positions. On the other hand, training QuaterNet short-term takes 20 minutes for 3000 epochs using quaternions, and 45 minutes for dual quaternions, and QuaterNet long-term takes 2 hours for 4000 epochs using quaternions, and 3.5 hours for dual quaternions. All visualisations have been created using the publicly available rendering engines of Aberman *et al.* [ALL*20]. The code used for the transformations between quaternions and dual quaternions, local and current dual quaternions, as well as all the code related to algebraic operations and losses of dual quaternions can be found on our project page.

In our experiments, human motion is represented as a combination of J joint coordinates for which we describe the orientation using various parameterizations. In addition, the global translation is modelled using the Cartesian coordinates of the root joint.

6.1. acRNN

This network is capable of synthesizing complex human motion with the use of recurrent neural networks. The model consists of 3 LSTM layers with a memory size of 1024, followed by a linear layer. During training, the network receives as input a sequence of 100 frames, consisting of a mixture of ground truth frames from the database and frames predicted by the model, and then generates the consecutive 100 frames. During test time, it receives a seed motion of 10 frames with the goal of producing extended sequences of complex human motion. Training is performed in batches 32, optimized using Adam [KB15] with learning rate 0.0001.

In the original implementation of acRNN, human motion is represented with joint positions relative to the root, and training is performed using the MSE loss that measures the similarity between the predicted and ground-truth frames. We modify the original network to receive as input: (a) local quaternions, (b) the ortho6D representation [ZBL*19], and (c) current dual quaternions. The quaternion and ortho6D representations are used to parameterize local rotations, while the dual quaternions operate using the current rotations and positions. Note that, similarly to our representation, quaternions can also be further constrained (in addition to the MSE loss) with the rotational ($\mathcal{L}_{quat}$) and positional losses ($\mathcal{L}_{pos}$). Recall that, the latter requires an additional FK layer, which converts local rotations to positions; we refer to this combination as quaternions-FK. Finally, to distinguish whether the performance gain in the learning is a result of the use of dual quaternions, or the combination of positions and rotations, we assess two more input formats: (a) local quaternions with current positions, appended for each joint, and (b) the ortho6D representation with the current positions. In that case, we employ only the MSE loss. In the following sections, we will refer to these two representations as quaternions-positions and ortho6D-positions.

The network receives as input the motion sequence $\mathbf{m}_t \in \mathbb{R}^{3+DJ}$, where the first 3 entries of each sequence contain the root translation, and D is the dimension of each representation ($J = 31$, $D = 3$

for positions, $D = 4$ for quaternions, $D = 6$ for the ortho6D, $D = 8$ for dual quaternions, $D = 7$ for quaternions with positions, and $D = 9$ for ortho6D with positions). Note that, during training we normalize the features of the acRNN network by subtracting the mean and dividing by the standard deviation of motion of the training set to account for different scales in the feature space and speed up the optimisation.

6.2. QuaterNet

We decided to experiment with QuaterNet since it accomplishes both short- and long-term motion modeling, and uses local quaternions as the pose representation. The short-term task concerns predictions over small time intervals of the future. Instead of predicting absolute rotations, at each frame, quaternion multiplication is used to obtain rotation deltas, while learning is based on the angle error between the predicted and ground-truth rotations. On the other hand, the long-term task is defined as the generation of motion that is conditioned on control variables, which specify the trajectory and speed. In this case, future motion is predicted using absolute rotations, and learning is achieved by integrating a positional loss that requires FK to convert the rotations to positions.

In this experiment, we modify the original QuaterNet architecture to additionally take as input dual quaternions. For a fair comparison, we train the original and adapted networks using the exact same datasets, losses, and number of epochs. In particular, for training the short-term task we use the local rotational loss on the Euclidean space. For the long-term task, we use the positional ($\mathcal{L}_{\text{pos}}$), offset ($\mathcal{L}_{\text{offset}}$), and rotational ($\mathcal{L}_{\text{quat}}$) losses. Instead of employing FK on the rotations, we derive the predicted positions directly from the output representation, circumventing the need for additional kinematic operations. Note that, long-term motion generation is highly uncertain and difficult to evaluate quantitatively as no ground-truth exists. For that reason, we assess the visual quality of the generated motion compared to that generated with ordinary quaternions. Finally, for both short- and long-term tasks, the global displacement of the root is controlled through the trajectory and speed parameters, inferred by the so-called pace network, which is a simple recurrent network.

7. Results and Evaluation

In the following section, we present our findings for each of the two networks. Note that, motion shaking cannot be illustrated in snapshots, thus refer to our supplementary video for animated results.

7.1. acRNN

7.1.1. Ablation Study

We begin our experiments by evaluating the variants of our representations in the acRNN architecture, using two motion subjects from the CMU dataset. The details for the train/validation/test configuration can be found in Table 3. Through an ablation study, we are able to evaluate the impact and the effectiveness of each of the proposed losses. The first experiment consists of the motion of subject 86, which consists of locomotion and other simple motions. The second experiment is performed on more complex motion, i.e. salsa dancing (subject 60-61).

First, we examine the loss used in the original implementation by Zhou *et al.* [ZLX*18], namely the $\mathcal{L}_{\text{mse}}$. We found that $\mathcal{L}_{\text{mse}}$ dominates over all other losses proposed when used simultaneously. Therefore, to examine the effect of the remaining loss components, which are specific for our representation, we removed the MSE loss in this experiment. We neglect the hip position and only focus on the quality of generated poses. We notice that the use of the positional loss alone ($\mathcal{L}_{\text{pos}}$) leads to ambiguity in the learning. This is reflected in abnormal generated poses. However, by additionally incorporating the offset loss ($\mathcal{L}_{\text{offset}}$), we observe that the poses are corrected, with minor artifacts on the hands. Finally, by incorporating the rotational loss, $\mathcal{L}_{\text{quat}}$, we observe that the generated motion remains natural and realistic on longer duration, with no artifacts. Motion does not freeze, however, it might get repetitive from a certain point onward; we believe that is because of the nature of the architecture, and not the representation itself, and can be resolved with more training.

We observed that when the motion is simple, the MSE loss, $\mathcal{L}_{\text{mse}}$, is sufficient to produce similar results compared to those obtained by including all other losses together. However, on data with more complex and dynamic movements, the addition of the remaining losses ($\mathcal{L}_{\text{quat}}$, $\mathcal{L}_{\text{offset}}$, $\mathcal{L}_{\text{pos}}$), can further improve the quality of motion, eliminating minor artifacts present. Incorporating various losses, though, requires investigation to adjust the weights assigned to each loss. However, for our representation the MSE loss is sufficient for smooth motion to be produced, even for 300000 frames.

7.1.2. Results and Evaluation

In terms of qualitative experiments, firstly, we evaluate the performance of the different quaternion-based representations: the quaternions with MSE only, the quaternions-FK, and the quaternions-positions. For this experiment we use the CMU dataset, subject 86. Among the three representations, we find that the quaternions-FK is more stable, with less shaking, especially on the feet; our findings are inline with previous works [PGA18; VYCL18; ALL*20]. We also observed that appending the positions with quaternions does not improve the generated motion, leading to increased shaking. We believe that this happens since the two components are learnt independently, and there is no constraint forcing them to match.

In a similar setup, we compare the ortho6D representation with ortho6D-positions. In this case, we were unable to clearly distinguish among the two, since abnormalities were only minor in both cases. We believe that appending the current positions in the input did not provide improvements, while increasing the complexity of the model (see Table 4 of the Appendix).

Our next experiment focuses on the learning and convergence. Figure 5 shows the training loss for the positional, ortho6D, quaternion, with their variants, and our representation. It can be seen that quaternion, and its variants, have the slowest convergence. We believe that this is due to the well-studied quaternion interpolation problem [ZSKS18; ZBL*19]. Directly regressing on positions is indeed effective and converges fast, at the cost of ambiguity when recovering the rotations. On the other hand, ours and the ortho6D rep-

Table 1: *Quantitative evaluation: (a) NPSS (↓), (b) Euclidean Loss (↓) on joint positions, and (c) the acceleration.*

	NPSS (↓)			Euclidean (↓)			Acceleration (↓)
	Duration in μs			Duration in μs			Duration in μs
representation	100	400	1000	100	400	1000	1×10^4
quaternions (MSE)	0.49	0.70	1.19	2.29	2.28	2.41	0.18
quaternions-FK	0.38	0.52	1.03	1.96	1.98	2.19	0.10
quaternions-positions (MSE)	0.50	0.69	1.15	2.09	2.09	2.22	0.25
ortho6D (MSE)	0.42	0.55	0.86	1.93	1.87	2.05	0.13
ortho6D-positions (MSE)	0.40	0.48	0.81	1.57	1.57	1.79	0.11
dual-quaternions (MSE)	0.44	0.53	0.87	1.62	1.63	1.86	0.10
dual-quaternions (all)	**0.30**	**0.36**	**0.70**	**1.36**	**1.46**	**1.78**	**0.08**

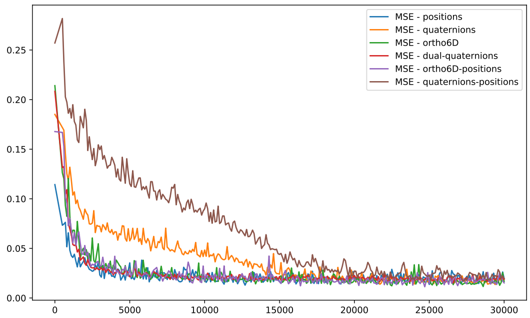

Figure 5: *Training and validation losses for each representation. Loss curves converge faster for dual quaternion and positional compared to quaternions.*

resentations have similar convergence: ortho6D avoids the quaternion interpolation problems by using the forward and upward vectors, while our hierarchy-aware encoding is richer in information, thus facilitating learning.

We qualitatively assess the generated motion at various learning stages. In this setup, we only use the MSE loss, for the sake of fair comparisons, among all three representations: the local quaternions, ortho6D, and ours. We generate results after training for 30K, as well as, 150K iterations. With 30K training iterations, we observe that the results generated using the quaternion and ortho6D representations are shaky and exhibit artifacts at certain times, while with our representation the generated motion is stable. With 150K iterations, the artifacts observed using the other two representations are reduced, yet, minor shaking is still evident in the generated motion. On top of that, we compare our representation, incorporating all losses, with the best variant of quaternions (quaternions-FK), and the ortho6D representations. Again, we train each network for 150K iterations, and observe that those minor artifacts, even though reduced, are still present when using quaternions-FK and ortho6D, but are not evident when using ours. Please refer to the video for an animated visual illustration.

Finally, we perform an additional experiment on the CMU salsa dataset, to assess the ability of our representation in synthesizing long-term sequences of complex motion (up to 30K frames). The generated motion does not freeze, however, gets repetitive from a certain point onward; that is because of the nature of the architecture, and not the representation itself. We compared our results with the corresponding quaternions and ortho6D, and observed that the quality of motion generated using our representation is more stable.

We employ several metrics to measure the quantitative performance of our representation in comparison to the baselines. In particular, we measure the frame-wise Euclidean distance and the Normalized Power Spectrum Similarity (NPSS) [GMK*19]; NPSS measures the similarity of power spectrum between ground truth and the corresponding generated sequence. Both metrics are calculated on joint positions, which are computed using Forward Kinematics. During this calculation, the global root displacement is ignored (zero) since we want to focus our evaluation on the pose. We evaluate all models at 150000 iterations. Following the original evaluation [ZLX*18] we sample 400 seed motions (10 frames) with a sliding window of 7 frames from the test set and generate motions of different intervals $\{100, 400, 1000\}\mu s$. As demonstrated in Table 1, our pose representation achieves the lowest scores.

Furthermore, we calculate the acceleration as an indication for the amount of jitter. We expect the mean acceleration of the predictions to be as close as possible to the real data. All models are evaluated at 150000 iterations and the root is ignored. We sample 400 seed motions (10 frames) with a sliding window of 7 frames from the test set and generate 300 frames, corresponding to 10 seconds of motion. As can be seen in Table 1, our representation constrained with all losses has the lowest acceleration error.

7.2. QuaterNet

7.2.1. Ablation Study

We perform a separate ablation study for the QuaterNet architecture. In this setup, we evaluate the contribution of $\mathcal{L}_{\text{pos}}$, $\mathcal{L}_{\text{offset}}$, and $\mathcal{L}_{\text{quat}}$ on local and current rotations, by removing one loss at a time and re-training the model. Note that, in this experiment we do not examine the contribution of the MSE loss, since it is not part of the original implementation. We observe that the use of either the $\mathcal{L}_{\text{quat}}$, local or current, allows for the generation of smooth motion when applied on the current dual quaternion.

Removing $\mathcal{L}_{\text{quat}}$, does not significantly affect the quality of the overall motion. However, certain artifacts occur in the end-effectors, particularly on hands. We believe that they occur be-

Table 2: *Average errors over all actions on test set in Euler space for short-term task on subject 5.*

Time (ms)	Quaternions	Dual Quaternions
80	0.3871	0.4091
160	0.6766	0.7250
320	1.0117	1.1086
400	1.1452	1.2612
600	1.4327	1.5686
800	1.7983	1.8111
1000	2.3004	**2.0584**
2000	6.0033	**2.8334**
3000	10.7987	**3.8045**
4000	13.1485	**4.7124**

cause the positional loss alone does not explicitly constrain the end-effector's orientation.

Furthermore, we re-train our network without integrating $\mathcal{L}_{\text{offset}}$. In this scenario, we found that the positional loss, $\mathcal{L}_{\text{pos}}$, alone is not sufficient to guarantee that the generated motion remains realistic. In the original implementation, the positional loss alone was sufficient since FK were used to retrieve the positional data given the ground-truth offsets. In contrast, with our representation we need to explicitly ensure that no skeletal violations occur. This can be achieved by additionally integrating the offset loss $\mathcal{L}_{\text{offset}}$. In fact, we conclude that $\mathcal{L}_{\text{offset}}$ is a significant component in the hierarchy-aware encoding. Note that, we observe that the use of bone length constraint (i.e. incorporating a loss that measures the differences in the ground truth and predicted bone lengths), instead of the proposed offset loss, fails to preserve the nature of skeletal structure.

7.2.2. Results and Evaluation

QuaterNet is used to quantitatively compare the results generated using the original quaternion and dual quaternion representation for short-term tasks. The numerical evaluation reveals the superiority of our hierarchy-aware encoding for longer time predictions. As seen in Table 2, our representation obtains lower rotational error, up to 65% for predictions of duration more than 1000ms, whereas we achieve a marginally higher error for predictions on shorter time intervals. The better performance on longer duration indicates that our implicit hierarchical encoding learns elements of the skeletal hierarchy, which prevent the motion from diverging far from a reference point. It is important to clarify that, while joint rotation accuracy is an evidence that the model can generate natural motions, our major evaluation is based on the naturalness of the generated motion as can be seen later in the long-term setup.

Similarly to Pavllo *et al.* [PGA18], we measure the performance of our pose representation in terms of the quality of the generated motion using the long-term setup. We verify the findings of Pavllo *et al.* [PGA18], by training the QuaterNet using quaternions, with and without employing a positional loss. In this latter setup, the generated motion is uncontrolled and suffers from perceptual abnormalities at certain points. That is because averaging the rotation errors by assigning equal weights in all joints, fails to embed that errors in crucial joints of the hierarchy, significantly affect the resulting pose.

When incorporating the positional loss, our observations are inline with the findings of Pavllo *et al.* [PGA18], namely, that the motion produced remains smooth and perceptually correct. This is due to the fact that the positional loss better constrains the motion, even though it allows for occasional mistakes on rotations which are not visible to the eye. Our representation produces similar results to the quaternion with the FK layer, since it encodes positional information within. Please refer to the supplementary video for animated experimental results.

7.3. Applications

One of the main advantages of our method is that it encodes features dependent on the skeletal proportions, allowing training on skeletal-variant datasets, without requiring that they are a priory retargeted onto a universal model. We leverage upon the fact that the network can learn correlations between motion patterns and skeletal attributes through the hierarchy-aware encoding. We examine whether these can be transferred onto unseen motions and skeletons. To do so, we use the acRNN architecture and the MIXAMO dataset. We compare our representation with 2 other input representations: (a) quaternions with offsets appended, and (b) ortho6D with offsets appended. We train the models for approximately 170 epochs. We compare with those two representations which explicitly take into consideration skeletal aspects of the different characters. For all representations, we use the MSE loss for the fairness of comparison. Our aim is to highlight that the skeletal features are implicitly incorporated into the representation and assist learning.

We limit our experiments to training on 4 distinct skeletons (Abe, Michelle, Timmy, and James), which have similar skeletal topology, but different proportions. We split the motion for each character into training/validation and testing, and consider the following 2 motion prediction scenarios: (a) seen character, but unseen motion; and (b) unseen character, but seen motion. We use a different character (Mousey) as the unseen skeleton. Finally, to further test the capabilities of our representation, we test it for the scenario of unseen character and unseen motion. Note that unseen motion refers to motion which was not directly used during training, but a contextually similar one could have.

With the original evaluation setup, i.e. feeding 10 motion frames as seed to initialize the generation, and various skeletons present in the training database, we have observed that the network's capabilities in synthesizing long-term motion decrease. Thus, for qualitative comparisons, we slightly adapt the evaluation protocol. We feed a sequence of 100 ground-truth frames ($f_t : f_{t+100}$), and generate a total of 250 frames: the first 100, ($f_{t+1} : f_{t+101}$) are based on ground-truth input frames, and the rest are generated based on the network's outputs. This allows us to have a reference motion for visual comparisons. Note that due to sub-sampling, some motions end up having less than 100 frames. In such cases, motion is generated based on ground-truth frames, for the maximum number of available frames. Once these are exhausted, predictions are generated based on the network's predictions.

In all the examined scenarios, our representation shows the ability to satisfy perceptually sensitive constraints by encoding prior information about the hierarchy and skeletal topology. This results

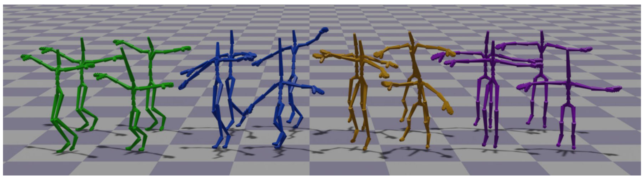

Figure 6: *Static screenshot of motion generated with unseen seed motion on seen characters. The green group depicts the seed motion, the blue represents motion generated using quaternions-offsets, the orange represents motion generated using ortho6D-offsets, and the purple motion generated with our representation.*

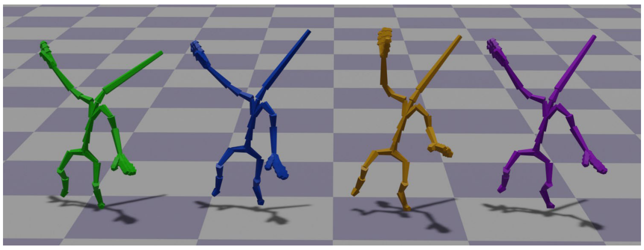

Figure 7: *The seen motion, shown in green, is transferred to an unseen character. The pose of the blue character (quaternion-offsets) and orange character (ortho6D-offsets) are abnormal, whereas the purple (ours) is closer to the reference motion.*

to motion prediction that more closely resembles the reference motion of each character. In contrast, when motion is represented using quaternions-offsets and ortho6D-offsets, the predicted motion is shaky, with jittering on several joints. Note that, in our visualizations we neglect the root translation, that is not part of our representation, and needs to be treated differently for various skeletons. Figure 6 shows the generated motion on characters from the training set, initialized with unseen seed motion. Figure 7 illustrates motion generated on an unseen character, with seed seen motion. As can be seen in the accompanying video, the motion generated with our representation is the most robust. Our representation is capable of producing smoother motion, even in the extreme scenario of unseen motion on unseen character. Our findings give a promising direction for the field of deep skeletal animation, revealing that the encoding of skeletal information into the representation, lays the grounds for training on skeletons with varying proportions. Please refer to the accompanying video for the animated results.

8. Conclusion

We have presented a skeletal pose representation, well-suited for deep skeletal animation. Our parameterization, which is the first to employ dual quaternions as the input to neural networks for motion modelling, learns to infer both rotational and positional information directly from the training data, resulting in a more constrained synthesis with improved realism. Results, using two different network architectures, demonstrate that our representation, which encodes the correlations between joints and limbs along the rig, enables the prediction of coherent and stable longer-term motion sequences. It also supports training on various skeletons, contrary to most works which assume a unified skeleton. Thus, the representation preserves motion attributes and nuances that are usually lost when retargeting motion onto a common skeleton for training purposes.

One limitation of our method relates to the way that the root translation is handled, when training with different skeletons. In this work, this information is being ignored and is corrected manually for the visualization. Future work could focus on handling the root translation when training with different characters. Furthermore, the encoding of both positional and rotational information in a unified representation comes at a small cost, computational complexity. This is attributed to two factors. Firstly, during training, the network has to learn an 8-D representation for each joint compared to 3, 4, or 6 dimensions of other representations. Secondly, even though the conversion from local to current coordinates and construction of dual quaternions is performed during pre-processing, some calculations performed during training such as conversion from local to current coordinate frame or vice versa, the extraction of the positions and normalization result in marginally increased training times. However, the increased complexity is not a limiting factor since the synthesis/prediction is still achieved in real-time. In the future, our implementation could be further optimized to reduce the computational complexity. Finally, we have shown the applicability of the representation in motion prediction using recurrent architectures. Future work can exploit the performance of representation in other types of architectures (feed-forward, convolutional), other applications (motion retargeting or reconstruction), controlled environments, or interactions.

Acknowledgements

The authors would like to thank Anastasios Yiannakidis for his help in rendering the results and helping with the implementation of the acRNN model, as well as Victoria Fernández Abrevaya for fruitful discussions. This work has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 860768 (CLIPE project). This project has also received funding from the University of Cyprus, and the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement No 739578 and the Government of the Republic of Cyprus through the Deputy Ministry of Research, Innovation and Digital Policy.

References

[ACH*18] Aristidou, Andreas, Cohen-Or, Daniel, Hodgins, Jessica K., et al. "Deep Motifs and Motion Signatures". *ACM Trans. Graph.* 37.6 (Dec. 2018), 187:1–187:13. doi: 10.1145/3272127.3275038 3.

[ALL*20] Aberman, Kfir, Li, Peizhuo, Lischinski, Dani, et al. "Skeleton-Aware Networks for Deep Motion Retargeting". *ACM Trans. Graph.* 39.4 (July 2020). issn: 0730-0301. doi: 10.1145/3386569.3392462 2, 3, 7, 8.

[AWL*20] Aberman, Kfir, Weng, Yijia, Lischinski, Dani, et al. "Unpaired Motion Style Transfer from Video to Animation". *ACM Trans. Graph.* 39.4 (July 2020). issn: 0730-0301 2, 3.

[AYA*22] Aristidou, Andreas, Yiannakidis, Anastasios, Aberman, Kfir, et al. "Rhythm is a Dancer: Music-Driven Motion Synthesis with Global Structure". *IEEE Transactions on Visualization and Computer Graphics* (2022). doi: 10.1109/TVCG.2022.3163676 3.

[BBKK17] Bütepage, Judith, Black, Michael, Kragic, Danica, and Kjellström, Hedvig. "Deep representation learning for human motion prediction and classification". *Proceedings IEEE Conference on Computer Vision and Pattern Recognition*. CVPR'17. Piscataway, NJ, USA: IEEE, July 2017 2.

[BKL17] Barsoum, Emad, Kender, John, and Liu, Zicheng. "HP-GAN: Probabilistic 3D human motion prediction via GAN". *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. CVPR'17 (2017) 2.

[CHS*18] Cao, Zhe, Hidalgo, Gines, Simon, Tomas, et al. "OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields". *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. CVPR '18. Washington, DC, USA: IEEE Computer Society, 2018 2.

[CMU21] CMU. *Carnegie Mellon University MoCap Database: http://mocap.cs.cmu.edu/*. [Online; Retrieved May, 2021]. 2021 7.

[Dan99] Daniilidis, Konstantinos. "Hand-Eye Calibration Using Dual Quaternions". *The International Journal of Robotics Research* 18.3 (1999), 286–298. doi: 10.1177/02783649922066213 4.

[DAP*17] Dong, Yuzhu, Aloba, Aishat, Paryani, Sachin, et al. "Adult2Child: Dynamic Scaling Laws to Create Child-like Motion". *Proceedings of the 10th International Conference on Motion in Games*. MIG '17. Barcelona, Spain: ACM, 2017. isbn: 9781450355414. doi: 10.1145/3136457.3136460 3.

[DAS*20] Dong, Yuzhu, Aristidou, Andreas, Shamir, Ariel, et al. "Adult2Child: Motion Style Transfer using CycleGANs". *Proceedings of the ACM SIGGRAPH Conference on Motion, Interaction, and Games*. MIG '20. Charleston, South Carolina, USA: ACM, 2020, 1–11. doi: 10.1145/3424636.3426909 2.

[DHS*19] Du, Han, Herrmann, Erik, Sprenger, Janis, et al. "Stylistic Locomotion Modeling with Conditional Variational Autoencoder". *40th Annual Conference of the European Association for Computer Graphics, Eurographics 2019*. Ed. by Cignoni, Paolo and Miguel, Eder. Genoa, Italy: EG Association, 2019, 9–12 2.

[DKL98] Dam, Erik B, Koch, Martin, and Lillholm, Martin. *Quaternions, interpolation and animation*. Tech. rep. DIKU-TR-98/5. Department of Computer Science, University of Copenhagen, July 1998 5.

[DWL15] Du, Yong, Wei, Wang, and Liang, Wang. "Hierarchical recurrent neural network for skeleton based action recognition". *Proceedings IEEE Conference on Computer Vision and Pattern Recognition*. CVPR'15. IEEE, 2015, 1110–1118. doi: 10.1109/CVPR.2015.7298714 2.

[FLFM15] Fragkiadaki, Katerina, Levine, Sergey, Felsen, Panna, and Malik, Jitendra. "Recurrent Network Models for Human Dynamics". *Proceedings of the 2015 IEEE International Conference on Computer Vision*. ICCV '15. USA: IEEE Computer Society, 2015, 4346–4354. isbn: 9781467383912 2, 3.

[GA98] Goddard, J. S. and Abidi, Mongi A. "Pose and motion estimation using dual quaternion-based extended Kalman filtering". *Three-Dimensional Image Capture and Applications*. Ed. by Ellson, Richard N. and Nurre, Joseph H. Vol. 3313. International Society for Optics and Photonics. SPIE, 1998, 189–200. doi: 10.1117/12.302453 4.

[GMK*19] Gopalakrishnan, Anand, Mali, Ankur, Kifer, Dan, et al. "A Neural Temporal Model for Human Motion Prediction". *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. CVPR'19. June 2019 9.

[Gra98] Grassia, F. Sebastin. "Practical Parameterization of Rotations Using the Exponential Map". *J. Graph. Tools* 3.3 (Mar. 1998), 29–48. issn: 1086-7651 3.

[HAB20] Henter, Gustav Eje, Alexanderson, Simon, and Beskow, Jonas. "MoGlow: Probabilistic and controllable motion synthesis using normalising flows". *ACM Transactions on Graphics* 39.4 (2020), 236:1–236:14. doi: 10.1145/3414685.3417836 3.

[HKPP20] Holden, Daniel, Kanoun, Oussama, Perepichka, Maksym, and Popa, Tiberiu. "Learned Motion Matching". *ACM Trans. Graph.* 39.4 (July 2020). issn: 0730-0301. doi: 10.1145/3386569.3392440 4.

[HKS17] Holden, Daniel, Komura, Taku, and Saito, Jun. "Phase-Functioned Neural Networks for Character Control". *ACM Trans. Graph.* 36.4 (July 2017). issn: 0730-0301. doi: 10.1145/3072959.3073663 2, 3.

[HSK16] Holden, Daniel, Saito, Jun, and Komura, Taku. "A Deep Learning Framework for Character Motion Synthesis and Editing". *ACM Trans. Graph.* 35.4 (July 2016). issn: 0730-0301. doi: 10.1145/2897824.2925975 2, 3.

[HYNP20] Harvey, Félix G., Yurick, Mike, Nowrouzezahrai, Derek, and Pal, Christopher. "Robust Motion In-Betweening". *ACM Trans. Graph.* 39.4 (July 2020). issn: 0730-0301. doi: 10.1145/3386569.3392480 2.

[JZSS16] Jain, Ashesh, Zamir, Amir Roshan, Savarese, Silvio, and Saxena, Ashutosh. "Structural-RNN: Deep Learning on Spatio-Temporal Graphs." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. CVPR '16. IEEE, 2016, 5308–5317 2.

[KB15] Kingma, Diederik P. and Ba, Jimmy. "Adam: A Method for Stochastic Optimization". *Proceedings of the International Conference on Learning Representations*. Ed. by Bengio, Yoshua and LeCun, Yann. ICLR'15. San Diego, CA, USA, 2015 7.

[KCŽO08] Kavan, Ladislav, Collins, Steven, Žára, Jiří, and O'Sullivan, Carol. "Geometric Skinning with Approximate Dual Quaternion Blending". *ACM Trans. Graph.* 27.4 (Nov. 2008). issn: 0730-0301. doi: 10.1145/1409625.1409627 4, 6.

[Ken12] Kenwright, Ben. "A Beginners Guide to Dual-Quaternions: What They Are, How They Work, and How to Use Them for 3D Character Hierarchies". *Proceedings of the 20th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision*. WSCG'12. 2012 2, 4, 5.

[KŽ05] Kavan, Ladislav and Žára, Jiří. "Spherical Blend Skinning: A Real-Time Deformation of Articulated Models". *Proceedings of the 2005 Symposium on Interactive 3D Graphics and Games*. I3D '05. Washington, District of Columbia: Association for Computing Machinery, 2005, 9–16. isbn: 1595930132. doi: 10.1145/1053427.1053429 4.

[LLL18] Lee, Kyungho, Lee, Seyoung, and Lee, Jehee. "Interactive Character Animation by Learning Multi-Objective Control". *ACM Trans. Graph.* 37.6 (Dec. 2018). issn: 0730-0301. doi: 10.1145/3272127.3275071 3.

[LYC*20] Li, Jiaman, Yin, Yihang, Chu, Hang, et al. "Learning to Generate Diverse Dance Motions with Transformer". *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. CVPR'20. 2020 4.

[LZCV20] Ling, Hung Yu, Zinno, Fabio, Cheng, George, and Van De Panne, Michiel. "Character Controllers Using Motion VAEs". *ACM Trans. Graph.* 39.4 (July 2020). issn: 0730-0301. doi: 10.1145/3386569.3392422 3.

[MBR17] Martinez, Julieta, Black, Michael J., and Romero, Javier. "On human motion prediction using recurrent neural networks". *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. CVPR'17. 2017 2, 3.

[MHL*21] Mourot, Lucas, Hoyet, Ludovic, Le Clerc, François, et al. "A Survey on Deep Learning for Skeleton-Based Human Animation". *Computer Graphics Forum* 41.1 (Nov. 2021), 1–32. doi: 10.1111/cgf.14426 2, 3.

[MYGY19] Ma, Li-Ke, Yang, Zeshi, Guo, Baining, and Yin, KangKang. "Towards Robust Direction Invariance in Character Animation". *Computer Graphics Forum* 38.7 (2019), 235–242. doi: 10.1111/cgf.13832 3.

[PBV21] Petrovich, Mathis, Black, Michael J., and Varol, Gül. "Action-Conditioned 3D Human Motion Synthesis with Transformer VAE". *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. Oct. 2021, 10985–10995 3.

[PGA18] Pavllo, Dario, Grangier, David, and Auli, Michael. "QuaterNet: A Quaternion-based Recurrent Model for Human Motion". *Proceedings of the British Machine Vision Conference*. BMVC '18. Newcastle upon Tyne, UK, 2018 2, 3, 6–8, 10.

[PGG*20] Peretroukhin, Valentin, Giamou, Matthew, Greene, W. Nicholas, et al. "A Smooth Representation of Belief over SO(3) for Deep Rotation Learning with Uncertainty". *Proceedings of Robotics: Science and Systems*. Corvalis, Oregon, USA, July 2020. doi: 10.15607/RSS.2020.XVI.007 2.

[SAA*20] Shi, Mingyi, Aberman, Kfir, Aristidou, Andreas, et al. "MotioNet: 3D Human Motion Reconstruction from Monocular Video with Skeleton Consistency". *ACM Trans. Graph.* 40.1 (Sept. 2020). issn: 0730-0301. doi: 10.1145/3407659 2, 3, 6.

[SCNW19] Smith, Harrison Jesse, Cao, Chen, Neff, Michael, and Wang, Yingying. "Efficient Neural Networks for Real-Time Motion Style Transfer". *Proc. ACM Comput. Graph. Interact. Tech.* 2.2 (July 2019) 2.

[SPN21] Schwung, Andreas, Pöppelbaum, Johannes, and Nutakki, Pradeep C. "Rigid Body Movement Prediction Using Dual Quaternion Recurrent Neural Networks". *2021 22nd IEEE International Conference on Industrial Technology (ICIT)*. Vol. 1. 2021, 756–761. doi: 10.1109/ICIT46573.2021.9453587 4.

[SZKS19] Starke, Sebastian, Zhang, He, Komura, Taku, and Saito, Jun. "Neural State Machine for Character-Scene Interactions". *ACM Trans. Graph.* 38.6 (Nov. 2019). issn: 0730-0301. doi: 10.1145/3355089.3356505 3.

[SZKZ20] Starke, Sebastian, Zhao, Yiwei, Komura, Taku, and Zaman, Kazi. "Local Motion Phases for Learning Multi-Contact Character Movements". *ACM Trans. Graph.* 39.4 (July 2020). issn: 0730-0301. doi: 10.1145/3386569.3392450 2, 3.

[TRA11] Torsello, Andrea, Rodolà, Emanuele, and Albarelli, Andrea. "Multiview registration via graph diffusion of dual quaternions". *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2011, 2441–2448. doi: 10.1109/CVPR.2011.5995565 4.

[VAC16] Vemulapalli, Raviteja, Arrate, Felipe, and Chellappa, Rama. "R3DG features: Relative 3D geometry-based skeletal representations for human action recognition". *Computer Vision and Image Understanding* 152 (2016), 155–166. issn: 1077-3142. doi: 10.1016/j.cviu.2016.04.005 4, 5.

[VYCL18] Villegas, Ruben, Yang, Jimei, Ceylan, Duygu, and Lee, Honglak. "Neural Kinematic Networks for Unsupervised Motion Retargetting". *Proceedings of the Conference of Computer Vision and Pattern Recognition* abs/1804.05653 (2018). doi: 10.1109/CVPR.2018.00901. eprint: 1804.05653 3, 8.

[WCX21] Wang, Zhiyong, Chai, Jinxiang, and Xia, Shihong. "Combining Recurrent Neural Networks and Adversarial Training for Human Motion Synthesis and Control". *IEEE Transactions on Visualization and Computer Graphics* 27.1 (Jan. 2021), 14–28. issn: 1077-2626. doi: 10.1109/TVCG.2019.2938520 2.

[WGLM18] Wang, Yuxiong, Gui, Liang-Yan, Liang, Xiaodan, and Moura, Jose M. F. "Adversarial Geometry-Aware Human Motion Prediction". *Proceedings of (ECCV) European Conference on Computer Vision*. Springer, Sept. 2018 2, 3.

[WHSZ21] Wang, He, Ho, Edmond S. L., Shum, Hubert P. H., and Zhu, Zhanxing. "Spatio-temporal Manifold Learning for Human Motions via Long-horizon Modeling". *IEEE Transactions on Visualization and Computer Graphics* 27.1 (2021), 216–227. doi: 10.1109/TVCG.2019.2936810 2.

[XL20] Xiang, Sitao and Li, Hao. *Revisiting the Continuity of Rotation Representations in Neural Networks*. 2020. arXiv: 2006.06234 [math.OC] 2, 3.

[ZBL*19] Zhou, Yi, Barnes, Connelly, Lu, Jingwan, et al. "On the continuity of rotation representations in neural networks". *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. CVPR '19. Washington, DC, USA: IEEE CS, 2019 2, 3, 6–8.

[ZLX*18] Zhou, Yi, Li, Zimo, Xiao, Shuangjiu, et al. "Auto-Conditioned Recurrent Networks for Extended Complex Human Motion Synthesis". *Proceedings of the International Conference on Learning Representations*. ICLR'18. 2018 2, 3, 7–9, 13.

[ZSKS18] Zhang, He, Starke, Sebastian, Komura, Taku, and Saito, Jun. "Mode-Adaptive Neural Networks for Quadruped Motion Control". *ACM Trans. Graph.* 37.4 (July 2018). issn: 0730-0301. doi: 10.1145/3197517.3201366 3, 8.

Appendix

A. Training data for acRNN

Table 3: *Train/Validation/Test split configuration for experiments with the acRNN architecture [ZLX*18]*

CMU Data Subject	Train	Validation	Test
Subject 86	1,2,4,5,7,8,10,12,13-15	6	3,9
Subject 61	1-12	13	14,15

B. Loss Weights

For the acRNN experiments, $\mathcal{L}_{\text{pos}}$ and $\mathcal{L}_{\text{quat}}$ is weighted by $\frac{1}{3}$, FK by $\frac{1}{4}$ and $\mathcal{L}_{\text{quat}}$ by 1. When only the MSE is used, the features are not weighted. The weight of the regularizer which is used to penalize the generation of un-normalized quaternions and dual quaternions, $\mathcal{L}_{\text{reg}}$, is set to 0.01.

For the QuaterNet experiments, $\mathcal{L}_{\text{reg}}$ is weighted by 0.01, and all other losses ($\mathcal{L}_{\text{pos}}$, $\mathcal{L}_{\text{off}}$, rotational loss) by 1.

C. Model Complexity

Table 4: *Number of total parameters of the acRNN model for each representation*

Representation	**Parameters**
Positional	21579890
Euler angles	21487712
Quaternions (CMU)	21646463
Quaternions (Mixamo)	21523559
Ortho6D (CMU)	21963965
Ortho6D (Mixamo)	21779606
Dual quaternions (CMU)	22281467
Quaternions+positional (CMU)	22122716
Ortho6D+positional (CMU)	22440218
Dual quaternions (Mixamo)	22035659
Quaternions+positional/offset (Mixamo)	21907634
Ortho6D+positional/offset (Mixamo)	22163684