Temporal Residual Jacobians for Rig-free Motion Transfer

Sanjeev Muralikrishnan[1], Niladri Dutt[1], Siddhartha Chaudhuri[2], Noam Aigerman[3], Vladimir Kim[2], Matthew Fisher[2], and Niloy J. Mitra[1,2]

[1]University College London, [2]Adobe Research, [3]University Of Montreal

Abstract. We introduce Temporal Residual Jacobians as a novel representation to enable data-driven motion transfer. Our approach does not assume access to any rigging or intermediate shape keyframes, produces geometrically and temporally consistent motions, and can be used to transfer long motion sequences. Central to our approach are two coupled neural networks that individually predict local geometric and temporal changes that are subsequently integrated, spatially and temporally, to produce the final animated meshes. The two networks are jointly trained, complement each other in producing spatial and temporal signals, and are supervised directly with 3D positional information. During inference, in the absence of keyframes, our method essentially solves a motion extrapolation problem. We test our setup on diverse meshes (synthetic and scanned shapes) to demonstrate its superiority in generating realistic and natural-looking animations on unseen body shapes against SoTA alternatives. Supplemental video and code are available at `https://temporaljacobians.github.io/`.

1 Introduction

A major challenge in character animation is to transfer the motion of a source (skeletal) system to a diverse range of target characters in a realistic manner. The traditional approach to achieve this involves using a *rig*, which connects a skeleton to the character's surface and manages the surface motion through a variety of constraints and parameters. Target movement is then conveyed from the skeleton to the surface by either simulating the physics of the muscles and fat or by using tailored sets of skinning weights. These weights can be manually created by skilled artists or derived from pre-existing rigged models [36, 37]. Despite its simplicity, rigging can be time consuming to set up and complicated to transfer to new target shapes; it may also fail to accurately capture dynamics.

There has been growing interest in developing surface deformation techniques that are more flexible and efficient than traditional rigging-based approaches and can utilize available volumes of full-body motion capture data. One possibility is to train data-driven methods to learn a low-dimensional parameterization, such as a morphable humanoid template [20] or a neural space deformation [38], to provide controllable handles for shape and pose-aware manipulations. However, such methods do not capture continuity over time and can overlook subtle

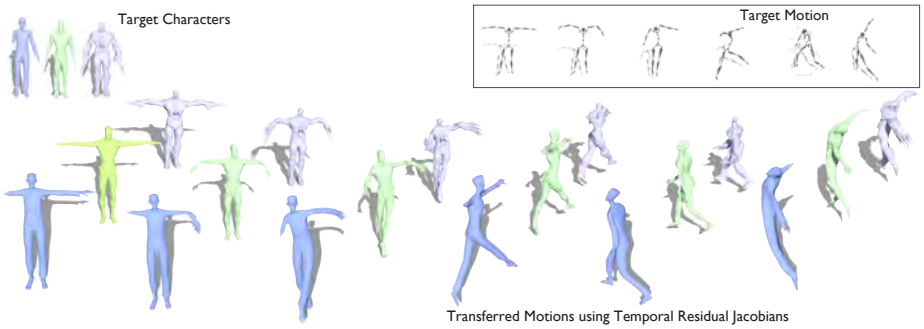

Fig. 1: Given a stick figure dance motion (top-right), *Temporal Residual Jacobians* retarget the animation to unseen, unrigged meshes (top-left) across time, producing realistic motion dynamics. Please refer to the supplemental webpage for videos. Our method can be trained on limited data, does not require rigged models or skinning weights during training or inference, and does not assume paired sequences or registration to any canonical template mesh. The method was trained on other bodyshapes: no target characters were seen during training. All results in the paper and supplemental material were obtained with automatic feature correspondences and without any postprocessing or smoothing applied.

motion dynamics essential for enhancing the realism of the generated motion sequences. To add time-dependent effects, corrective vertex deformations, similar to DMPLs [20] and SoftSMPL [31], have been introduced in multistage workflows. Unfortunately, such methods do not account for elementwise temporal inter-relations and have limited generalization to unseen characters.

We aim to transfer a source motion, expressed as joint angles on a stick figure, onto a target shape, specified as a (rig-free) mesh. We want to do so without access to any rigging on the target shape, and we also want to avoid any fixed template or morphable shapes. A desirable solution should address several challenges: (i) handle rig-free meshes and/or scans with arbitrary topology; (ii) produce plausible transfers to diverse shapes; (iii) achieve continuity over space and time and thus avoid artifacts (e.g., broken meshes, jittery motion); and (iv) work with long motion sequences without significant drift. The first two problems are partially handled by rig-free pose transfer (i.e., transferring a pose to a target mesh) methods [2,19,34]. While such methods produce plausible single frames, they lead to jittery motion transfer and motion-induced geometric artifacts like shearing.

We propose a novel approach that learns local spatio-temporal changes to produce natural-looking motion transfers. Technically, we achieve this via a new representation in the form of *Temporal Residual Jacobians* that temporally links spatial predictions and is directly supervised using example motion sequences. We jointly train two neural networks to individually predict local spatial and temporal changes. They are coupled by spatial integration with a differentiable Poisson solve, and temporal integration with a neural ODE. A **key technical insight** is that instead of having the neural ODE predict per-frame mesh de-

formations, it is more effective to predict initial deformations independently of time via a base posing model (we use Neural Jacobian Fields [2]), and then have the neural ODE predict *residual deformations*, linked over time, as corrective factors that improve temporal coherence. Figure 1 shows how a stick-figure control motion produces target motions for different characters, without requiring registration to any standard template or character rigs.

We evaluate the effectiveness of our method for motion generation to different character bodies (e.g., humanoids, animals, Mixamo characters, scans) and different motions (walk, run, jump, punch, dance). We compare our approach to alternatives, when available. We provide qualitative and quantitative results using the AMASS [21], COP3D [32], and 4DComplete [17] datasets.

In summary, our primary contributions are: (i) a novel method that enables motion transfer via Temporal Residual Jacobians and can be trained directly using positional data; (ii) local predictors that can be integrated in space and time to create natural looking character animations; and (iii) a robust pathway to transfer realistic character motion without the need for explicit rigging or learning a parameterization using any canonical template shape.

2 Related Work

Parametric Shape Deformation. These methods express 2D or 3D shapes as a known function of a set of common parameters, and model deformations as variations of these parameters. Such methods include cages, explicit [14] or neural [38], blendshapes [15], skinned skeletons [13], Laplacian eigenfunctions [30], and several other variations. Linking the parameters to the shape's surface often requires manual annotation of weights (commonly known as weight painting) in 3D authoring tools. Alternately, given sufficient data (i.e., meshes, rigs, skinning weights), end-to-end training can produce realistic neural rigs, as demonstrated by Pinocchio [4], RigNet framework [36], skinning-based human motion retargeting [22], and skeletal articulations with neural blend shapes [16]. Unsupervised shape and pose disentanglement [42] proposes to learn a disentangled latent representation for shape and pose, which can be further used to transfer motion using shape codes. This requires meshes to be registered and have the same connectivity. To animate these parametrized shapes through time, the parameters are varied over time and the dynamic weights animate the mesh. These methods require access to body templates and/or rigs and can produce results that are jittery due to loose coupling of the individual frame predictions.

Dynamic Motion. It is possible to model temporal surface effects by simulating the underlying soft tissues using finite element methods (FEM) [6, 9]. This direct simulation is typically slow and requires artists to design the underlying bone and muscle structure [1]. Approaches have been developed to overcome the stiffness problem in FEM to accelerate simulating these systems [23] or to solve the problem in a lower-dimensional subspace [25]. For the specific case of rigged human characters, Santesteban et al. [31] add soft-tissue deformation as an additive per-vertex bump map on top of a primary motion model; AMASS [21]

imparts secondary motion using the blending coefficients of the DMPL shape space [20]; and Dyna [26] learns a data-driven model of soft-tissue deformations using a linear PCA subspace. These efforts, however, assume access to primary motion via a skeleton rig and are restricted to humans, registered to a canonical template. Complementary dynamics [5, 40] models physically-based secondary motion in the subspace complementary to that spanned by an animation rig. This approach adds automatic secondary motion to arbitrary animated objects, but requires the target shape to be rigged, is not designed for deformation transfer (the base animation is part of the input), and is tied to a specific hand-coded secondary physics model. DeepEmulator [41] achieves a similar effect using a local-patch-based neural network to learn the secondary behavior, but again requires the primary motion as input and does not support deformation transfer.

Discrete Time Motion Models. Given their ability to model time, deep recurrent neural networks have also been used to model shape sequences. Fragkiadaki et al. [10] use LSTMs to predict short human joint motions given initial frames. Harvey et al. [11] leverage LSTMs for in-betweening to predict intermediate joint motions. He et al. [12] learn a motion field for joints through time. In all these methods, the mesh itself is deformed via rigging, and since joint motion does not encode bodyshape, they do not sufficiently represent secondary motion. Also, being discrete time representations, these approaches must train on large datasets of joint motion. Qiao et al. [27] instead use mesh convolutions with LSTMs to deform vertices through time. In our work, we use neural ODEs as they can model time continuously instead of discretizing time and modeling the sequence using LSTMs. Further, vertex-based deformation models are susceptible to artifacts like normal-inversion as we demonstrate in our evaluation section.

3 Approach

Given an unrigged, triangulated mesh of a 3D character, we aim to animate it by motion transfer from an available motion described by relative joint angles (stick figure motion) at each time step. The relative joint angles are represented as Euler angles and are defined at each joint with respect to its hierarchy in SMPL's kinematic tree (cf. [12]). We obtain these joint-based motion representations from the AMASS dataset [21]. Since we aim to animate the mesh itself from the joints' motion, we seek to learn a mapping from the joint representation to the positions of the given mesh's vertices, and to do so at each time step while ensuring we generate a smooth and artifact-free mesh animation. We supervise our setup with ground-truth meshes from the AMASS dataset [21].

We parameterize such a character $X \in \mathbb{R}^{N\times 3}$ as assigning positions to each of its N vertices of the underlying mesh. Thus to impart a motion to a mesh, we perform this assignment at every time step. Given a shape X_0 as a triangulated mesh, in its initial pose configuration, along with per-frame pose configuration M_t that describes the target pose at time t, we aim to predict the shape X_t at each time t generating the full motion sequence; essentially learning a mapping from relative joint orientations to mesh deformations (typically, from 30-50 joints

to 50-100k vertices). We aim for a data-driven method that generalizes to new characters and does *not* rely on shape rigging or intermediate shape keyframes. Another desirable property is to do so with limited data (e.g., working with 5-10 motion examples), as obtaining full-body 3D motion data is non-trivial.

Our key observation is that we can robustly train neural networks to predict changes *local* in both space and time, which can then be integrated across space, using a differentiable Poisson solve; and across time, using Euler equations to handle an ODE numerically. Integrating across space helps maintain plausibility of the entire shape, while integrating across time helps model a realistic, time-coherent motion. Further, local predictions help higher generalization capability to unseen body shape; while the spatio-temporal integration ensures smoothness of the sequence, without jitters or undesirable shearing artifacts. Our end-to-end differentiable formulation allows training the networks directly using example training sequences, without requiring factorized spatial and temporal motion signals. For spatial handling, we extend the representation from local deformation encoding [33] and affine mapping framework Neural Jacobian Fields (NJF) [2], which learns an affine transformation field that is sampled at each mesh face and integrated into vertex positions via a Poisson solve. For temporal coherence and consistency in these predicted Jacobians, we couple temporal signals across the character motions using a neural ODE framework [7] via a novel Temporal Residual Jacobian representation. Predicting *residual deformations* that correct the predictions of a base model for time-coherence turns out to be more effective than trying to make the base model itself time-coherent.

3.1 Overview

Training: Since our goal is to map temporally varying joint angles to full mesh animation, during training we assume this mapping is available i.e., we have one-to-one mapping across time between joint orientations and the corresponding meshes. Thus, we use the motion sequences in AMASS [21] as our training dataset for humanoid shapes. Thus, during training, our algorithm takes in the joint angles at each time step, the mesh itself at time $t = 0$ in its starting pose, and the full mesh sequence is used for supervising our neural networks.

Inference: At inference time, our algorithm simply takes in an unrigged character provided as a triangulated mesh. Our trained method is then used to animate this mesh from joint-sequences of motions sampled from AMASS. These meshes can be obtained in-the-wild or from character datasets like Mixamo. For shapes in the AMASS dataset since we have the ground-truth motions, we evaluate our framework quantiatively. We only show qualitative motion generation for other shapes in the work, as there is no ground-truth solution.

Motion and Shape parameters: The motion sequences in AMASS were obtained from live captures of subjects performing different motions; these were then parametrized in SMPL's [20] shape and pose space, thus providing a mapping from joint orientations to 3D mesh pose - i.e., vertices of the mesh oriented

to match the pose represented by the provided joint orientations. We use this mapping to supervise our framework. Specifically, this data framework allows us to sample motions across both shapes and poses, by varying the respective low-dimensional parameters β and α. In our work, the α translate to joint angles at each time step and we use the source live-capture subject's β as is, and pass it as a conditioning variable to our method.

3.2 Preliminaries

Our framework uses two components – one to learn spatial mesh deformations (re-posing) and the other to learn temporal signals to generate a temporally coherent mesh motion. We describe these below.

Neural Jacobian fields, as introduced in [2], encode local deformations by defining a field via map ϕ, defined over space, that can be sampled at the N vertices of the surface mesh. Specifically, given ϕ, we compute Jacobians in the basis of the triangles of the surface mesh as,

$$J_i = \phi \nabla_i^T \tag{1}$$

where ∇_i is defined as the gradient of triangle t_i in its basis B_i defined at the triangle's centroid. Thus, given a learned map, we obtain each triangle's (estimated) affine transformations J_i. Recovering the vertex positions from this per-triangle assignment of affine transformations is then done via a least-squares formulation that reduces to solving a Poisson system given by (cf. [39]),

$$\phi^* = L^{-1} \rho \nabla^T J \tag{2}$$

where ρ is the mesh's mass matrix, $L = \nabla^T \rho \nabla$ is the cotangent Laplacian, and J is a stack of (estimated) Jacobians J_i. This solution gives a unique mesh up to a translation, which we fix using X_0, the mesh at $t = 0$. Mesh positions directly supervise the neural map ϕ via a differentiable Poisson solver [24].

Thus, NJF allows us to learn affine deformations of a given mesh. In our work, the map ϕ is a trainable neural network that predicts local deformations of the mesh's triangles. Given conditioning parameters, the trained map can then be used to predict the Jacobians of each triangle in the mesh at each time step, generating a time-coherent animated 3D mesh.

Neural ODEs Chen et al. [7] represent differential equations with neural networks to model the dynamics of time-varying systems (motion sequences in our setting). Specifically, given a function $f(t; \theta) := \frac{\partial J}{\partial t}$, where f is a neural network with parameters θ and t being time, the variable J can be integrated out at t as

$$J(t) = \int_0^t \frac{\partial J}{\partial t} dt = \int_0^t f(t; \theta) dt + J_0. \tag{3}$$

The neural network can be directly supervised with J values as the network output utilizes a black box differential equation solver without explicitly

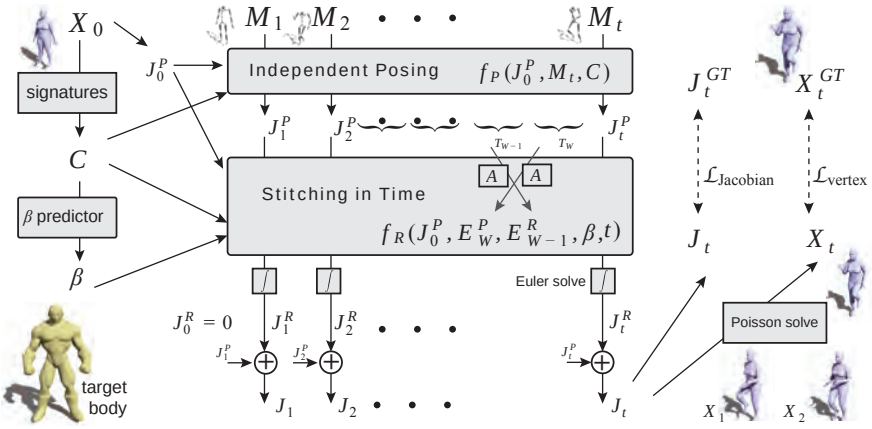

Fig. 2: Method overview. Starting from input stick figure motion ($\{M_i\}$) and a target body shape (X_0), Residual Temporal Jacobians makes local predictions, using primary f_P and residual f_R MLPs, to predict spatial and temporal changes to per-triangle Jacobians. These are then integrated in space, via a Poisson solve, and in time, via numerical Euler stepping, to predict motion dynamics at time frame t. These two learnable modules are trained simultaneously with only direct object-level supervision using a combination of positional and Jacobian losses. We use the ground-truth meshes from AMASS to supervise the predictions. The time t input is positionally encoded.

discretizing the dynamic system. This leads to better estimation, benefits from constant memory updates (as opposed to explicit hidden states in RNNs, LSTMs, etc.), and trades numerical precision for speed.

Our work seeks to learn how the triangle Jacobians move through time. Thus, our neural ODE operates in the Jacobian space, with Jacobians being the variables to integrate across time.

3.3 Motion Transfer with Space-time Integration

Without keyframes, we have to solve a temporal extrapolation problem. Our goal is to produce shape-preserving realistic motion sequences, as well as to learn such a motion model from a very sparse dataset. Not surprisingly, if each frame were independently predicted, the resulting motion would not be consistent across time, resulting in a jittery and artifact-ridden sequence (see Section 4). We, therefore, propose a novel framework wherein independent frame predictions are improved by providing temporal signals from a neural ODE. The two networks work in lockstep, each boosting the other's predictions, and are trained together.

Independent posing: We estimate each frame's Jacobians independent of temporal information. Given a shape X_0, pose configuration (relative joint angles with respect to SMPL's kinematic tree) M_t and features describing each mesh face,

we predict J_t^P as the primary Jacobians at time t. Specifically,

$$\Delta J_t^P = f_P(J_0^P, M_t, C; \theta_P) \tag{4}$$

$$J_t^P = J_0^P + \Delta J_t^P \tag{5}$$

where J_0^P is the first frame Jacobian computed from X_0, M_t is the pose configuration given as relative joint angles, and C are per-face features defined at the centroid of the mesh faces, of the mesh at $t = 0$. In our tests, as features C, we jointly learned PointNet features on the centroids and normals of X_0 and augmented them with pre-computed Wave Kernel Signatures [3]. Our feature network (a shallow PointNet) learns geometric features that enable mapping to an unseen shape via correspondence in this learned feature space. We note that conditioning the posing network on J_0^P and adding the predicted delta via a residual connection significantly improves pose prediction and generalization to unseen pose configurations. We additionally note that the Jacobians are computed in the local basis defined at each face's centroid. Thus, J_0^P is the Jacobian of the identity deformation in the local basis (a rotation). Henceforth, we do not update the basis in the sequence and express all Jacobians on this chosen basis.

In practice, we analytically compute the triangle Jacobians of only the first frame X_0 in the sequence. We then predict the Jacobians at each time instance t in the coordinate frame of X_0 and solve Equation 2 to obtain the shape X_t at time t. Importantly, we augment NJF with temporal learning signals, by linking these independent per-frame predictions via a neural ODE, as described next.

Stitching across time: Our key observation is that learning *local* changes in time generalizes better to unseen shapes. Central to our method is a neural ODE that provides temporal training signals to the primary NJF and integrates across time to learn a smooth, arbitrary-length motion sequence. We found independent per-frame predictions are prone to artifacts and do not generalize well (see Section 4). We also observe that a neural ODE, in isolation, cannot predict the entire sequence due to the drift problem inherent to estimating functions using numerical ODE methods. Specifically, given an initial state and per-frame control parameters (in our case, joint angles), predicting the mesh sequence is an extrapolation problem. As such, ODEs are prone to drifting away from the underlying function. This problem is exacerbated as the length of the motion increases – the longer the motion, the larger the accumulated drift.

As a solution, we propose a novel formulation to address both the incoherence of per-frame predictions and the drift problem. Specifically, we direct the neural ODE to learn only *Residual Jacobians* at each time step conditioned on the predictions from Eq 5 and on a window of past Jacobians. The Residual Jacobians are corrective factors which are directly added to the per-frame Jacobians. We predict Residual Jacobians local in time as outputs of a Neural ODE to ensure temporal coherence. This allows us to handle much longer motions, spanning 1-3k frames, without noticeable drift. We describe our method below.

ODE formulation: To handle arbitrary length sequences, in the interest of memory and training speed, we train and infer a given sequence in windows of consec-

utive frames. A given sequence is broken into fixed window sizes. We initialize J_0 in Eq 3 as the first frame's Residual Jacobian. Since the first frame is stationary and given as input, we set the first frame's Residual Jacobian as,

$$J_0^R = \mathbf{0} \in \mathbb{R}^{3\times 3}. \tag{6}$$

We then task the neural ODE to learn Residual Jacobian J_t^R at each t, by extrapolating from J_0^R using Euler's integration. We then extract the final Jacobians in terms of the base Jacobians corrected by the Residual Jacobians as

$$J_t = J_t^P + J_t^R. \tag{7}$$

We predict the residuals J_t^R by integrating over time the bodyshape-specific local changes predicted by f_R which is defined as,

$$\frac{\partial J_t^R}{\partial t} = f_R(J_0^P, E_W^P, E_{W-1}^R, \beta, t; \theta_R) \tag{8}$$

where β is the shape signature that defines the given body shape, and J_0^P, as defined previously, are the Jacobians of the first frame; $E_W^P$and $E_{W-1}^R$are attention encodings of current-window pose predictions and previous-window residual predictions. We integrate the local changes (see Eq 8) over time using Euler's method to obtain J_t^R at each t as,

$$J_t^R = \int_0^t \frac{\partial J_t^R}{\partial t} dt + J_0^R = \int_0^t f_R(J_0^P, E_W^P, E_{W-1}^R, \beta, t; \theta_R) dt. \tag{9}$$

Our jointly trained attention encoders are defined as,

$$E_W^P = A^P(J_W^P, T_W) \tag{10}$$

$$E_{W-1}^R = A^R(J_{W-1}^R, T_{W-1}) \tag{11}$$

where A^P and A^R are multi-head attention networks, J_W^P and J_{W-1}^R are a block of sequential Jacobians in the current window W and past window $W-1$, respectively; T_W and T_{W-1} are correspondingly blocks of time instances in these windows and are positionally encoded using time. In all our experiments we use a window size of 32 frames.

Note that we use the attention networks to encode a window of Jacobians to a single encoding. Since the encoding sizes are thus constant, we can handle arbitrary window/sequence lengths without overflowing memory. These encoders distill the current posed Jacobian predictions from Eq 5 and previously predicted Residual Jacobians obtained from Eq 9.

We pass the output of the attention networks as conditioning to Eq 8 to integrate and obtain the residuals in Eq 9. The predicted residuals are then added to the posed Jacobians in Eq 7. Finally, we spatially integrate the predicted Jacobians J_t using a differentiable Poisson solve [24], in the coordinate frame of the first frame, to obtain the predicted shape X_t at t.

Fig. 3: Generalization across bodyshapes. We show results of different motion transfers on meshes found in-the-wild (blue), FAUST scans (pink) and Mixamo characters (green). We observe a smooth motion consistent with the target geometry in each case. Please see supplemental materials.

Loss function: Our pipeline is trained end-to-end using only a shape loss over vertices of X_t and a Jacobian loss. Thus, our final objective function is simply

$$\mathcal{L}_{\text{vertex}} = \|X_t - X_t^{GT}\|^2 \quad \text{and} \quad \mathcal{L}_{\text{Jacobian}} = \|J_t - J_t^{GT}\|^2, \tag{12}$$

$$\mathcal{L} = \mathcal{L}_{\text{vertex}} + \alpha \mathcal{L}_{\text{Jacobian}}. \tag{13}$$

We use $\alpha = 0.05$ in our tests.

4 Evaluation

Motion datasets. We train and evaluate our method on motion sequences from three different datasets. First, the AMASS dataset [21] for humanoid motions, which is based on the SMPL body model [20] that allows us to vary body-shapes by changing its shape parameter β, for a given sequence. Thus, we train our model for a given motion category (like running) to predict the sequences on varying body-shapes (indicated as the β-predictor in Figure 2). During training, each sequence in the category is trained with only one body-shape. During inference, given an unknown test shape X_0 and a motion $\{M_t\}$, we synthesize the full body motion for the target shape. Second, we also tested on a synthetic DeformingThings4D dataset [18], which provides animal 4D meshes as deforming sequences. Finally, we also present results on motions extracted from video recordings of animals in the COP3D dataset [32] where our inputs are meshes fitted to the video recordings to train Temporal Residual Jacobians.

Target shapes. We evaluate our method on various forms of target shapes, namely sampled SMPL models, scans from the FAUST dataset, characters from

Fig. 4: Generalization across shapes with very sparse training sets. Here, we show motion transfer from two animal sequences (in yellow) sampled from the DeformingThings4D dataset [18], to animal meshes found in the wild (in blue). Our method was trained on only two sequences from this dataset and yet generates plausible motion transfer to unseen shapes. Rigs were *not* available to our algorithm at training and/or test time. (Note: blue sequences have been slightly globally rotated for visibility.)

the Mixamo library, various models from online 3D repositories (e.g., wolf, triceratop, monster, hole-man, etc.). Non-manifold inputs were converted to manifold meshes before running our algorithm. All correspondences were as inferred by the signature module (i.e., combination of PointNet features on centroids and normals of faces along with WaveKernel Signatures, previously defined as C).

Implementation details. All the networks we use are shallow MLPs to aid training speed. For our independent posing network, we use a 4-layer MLP with ReLU activation, with the final layer being linear. For the residual Jacobian prediction, we use a 3-layer MLP similarly with intermediate ReLU activations and a final linear layer. Both our attention networks follow the same architecture – we use two attention heads, with a 32-neuron wide feed forward layer and 32-dimensional features for the keys and values. Our PointNet network from which we obtain per-face features similarly has 3 ReLU layers followed by a linear layer at the end. Our method and the baselines are trained until $\mathcal{L}_{\text{vertex}}$ converges to less than 3e-4 or upto 300 iterations. On a 12 GB TitanX our typical training time is 6-8 hours, varying by the lengths of the sequences.
For the `walking` and `dancing` motions, we set the root orientation in AMASS to zero, so all subjects are front facing, making it easier to train (similar to [12]), as our Jacobian-based formulation cannot, on its own, infer global rotation and/or translation. We do not, however, predict global transformations; we obtain the global transforms from the source AMASS sequences and apply them to our final outputs at inference after appropriate scaling according to the target's height.

Qualitative results. We show various examples of deformation transfer by our method in Figures 1, 4, and 5. Please refer to the videos in the supplemental webpage. Our method generalizes to new shapes of varied body types, including non-humans *completely unseen during training*, while preserving the source motion. We show examples on monsters, a 4-armed character, etc. Additionally, as different humans perform the same motion in different manners, we capture those varying dynamics in the deformation of the new shape as well. We can also learn and apply motions from animals, both from synthetically generated mesh sequences as well as motions extracted from monocular video recordings.

Fig. 5: Motion transfer from COP3D dataset. We train on only four sequences of dogs obtained from the COP3D dataset [32], which are monocular video recordings of animal motion, and transfer the observed regressed motion (in yellow) to creature meshes found in the wild (in blue).

Comparisons. Our method Temporal Residual Jacobians, due to the space-time coupled formulation, is designed to produce natural-looking motion retargeting. Although we are unaware of any other method that performs the specific motion extrapolation problem (i.e., *without* access to keyframes), we compare with possible baseline design variations. First, *VertexODE*, where an MLP predicts the rate of change of vertex positions (velocity) at time t, followed by a Neural ODE [7] that takes numerical steps by Euler's method to integrate to the vertex positions at t. This baseline directly displaces the vertices (i.e., without triangle Jacobians). As seen in Figure 6, this method produces significant artifacts and leads to degenerate shapes. It also loses the intended motion after some time, as seen in the videos. Second, *NJF(M_t)*, where we extend the original NJF framework to be additionally conditioned on per-frame relative joint orientations M_t, to predict the Jacobians at each time step. Framewise prediction leads this approach to suffer from jitters (please refer to supplemental videos) and a tendency to overfit to training data. Although the individual frames are mostly plausible, they occasionally suffer from frame-level artifacts as shown in Figure 6. The primary limitation, however, is jittery output as seen in the video results since the frames are independently generated. Note that a method without spatial or temporal derivatives wherein a simple MLP is trained to predict vertex deformations given the initial geometry and time t performs poorly, with numerous artefacts due to vertex-level predictions.

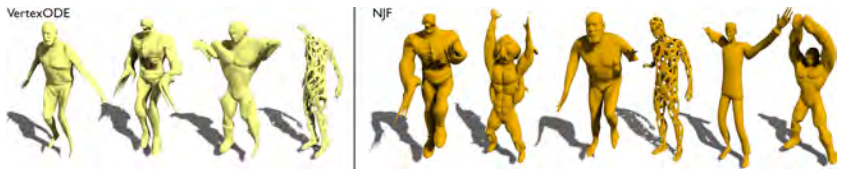

Fig. 6: Observed artifacts in baselines. For each motion, we show results from an intermediate frame of the motion transfer for our baselines. VertexODE (left, yellow) completely distorts the shape, while not following the target motion. NJF (right, brown) suffers from temporal discontinuity resulting in motion-driven geometric artifacts – extended or shrunken parts as it tends to a linear path in later time steps – and jitter.

Quantitative comparison: We evaluate our method on several types of sequences and compare ours against other baselines in Table 1 by computing the residual error with respect to the ground-truth sequences. We compare five different motion categories from the AMASS dataset - `running`, `jumping jacks`, `walking`, `hopping`, and `punching`. We achieve the best on all metrics.

Human motion to non-humanoid characters. We show results on human motion applied to non-humanoid characters (e.g., 4-armed monster, alien-reptile) in supplemental videos. Please note the quality of automatic transfer without manual landmark specification.

Table 1: Quantitative evaluation. Average vertex-to-vertex error in cm, L2 error of predicted Jacobians and angular error of normals in degrees, measured against ground truth sequences, for different motion categories and averaged over multiple sequences within the same target motion category. Here we compare against neural ODE [7] and an extended version of NJF [2]. Lower values indicate better generalization.

Method	Jump			Run			Punch			Walk			Dance		
	L2-V	L2-J	L2-N	L2-V	L2-J	L2-N	L2-V	L2-J	L2-N	L2-V	L2-J	L2-N	L2-V	L2-J	L2-N
VertexODE	22.59	1.22	48.21	14.23	0.83	42.11	17.66	0.65	40.04	23.92	1.01	46.12	26.15	1.26	50.17
NJF(M_t)	5.52	0.41	9.66	3.61	0.32	7.38	4.95	0.38	7.42	6.85	0.34	8.12	4.86	0.44	11.24
TRJ (Ours)	**2.64**	**0.28**	**7.31**	**1.73**	**0.24**	**5.63**	**2.86**	**0.26**	**6.65**	**1.48**	**0.22**	**6.33**	**3.96**	**0.37**	**9.88**

Handling long sequences. One advantage of our space-time coupled formulation is the ability to handle long motion sequences. We show examples of motion transfer from a few hundred to a couple of thousand frames (dance and walk) in the supplemental.

Design variation. A seemingly possible variation of ours is to choose a residual Jacobian representation, but expressed in terms of the previous predicted frame. While this seems attractive, we found it very slow to train as we have to back-propagate through time, and convergence is very slow (when using moderate computational resources). Hence, we found this approach to be infeasible with the current memory requirements for attention (and transformer) modules.

Alternatives to Euler Solve. We experimented with higher-order ODE solvers, namely Runge-Kutta and Runge-Kutta-Fehlberg. However, we noticed no significant improvements, even with the noticeably increased training time. We use the simpler Euler's method for temporal integration for our tests.

5 Conclusion

We have presented Temporal Residual Jacobians, a spatially-coupled, neural ODE-based, motion transfer framework conditioned on body type and target motion to produce local Jacobians that are subsequently integrated across space and time to deform the target geometry. The resultant motions are robust, realistic, and generalize to different body types. We extensively tested our method on both synthetic and real data captures, and enabled generic motion transfers to an extent which is not possible using existing methods.

Our method has limitations. (i) We do not impose physics constraints, therefore, our animation can have self-intersections (see webpage videos). An interesting direction is to incorporate constraints for collision detection, e.g., via subspace-based contact handling [29]. We want to incorporate such an approach directly into the method, possibly via an attention mechanism, as motion dynamics are affected by earlier collisions. (ii) Although our Temporal Residual Jacobians, along with windowed attention modules, keep drifts low, error still

accumulates over long motion sequences. A possible solution is to couple our method with a keyframe-based workflow [35]. (iii) Finally, our current formulation implicitly establishes correspondences and uses them to infer temporal Jacobians for surface triangles. This step may incur errors as shown in project webpage (e.g., flapping ear of the mouse). These spurious correspondences can be overridden by artists, possibly directly by "paintbrushing" correspondences or materials (e.g., indicating that the ear of the bunny model should be floppy) or using semantic features learned from untextured meshes [8, 28].

Acknowledgement This project has received funding from the UCL AI Center, gifts from Adobe Research, and EU Marie Skłodowska-Curie grant agreement No 956585.

References

1. Abdrashitov, R., Bang, S., Levin, D., Singh, K., Jacobson, A.: Interactive modelling of volumetric musculoskeletal anatomy. ACM Transactions on Graphics **40**(4) (2021)
2. Aigerman, N., Gupta, K., Kim, V.G., Chaudhuri, S., Saito, J., Groueix, T.: Neural jacobian fields: Learning intrinsic mappings of arbitrary meshes. SIGGRAPH (2022)
3. Aubry, M., Schlickewei, U., Cremers, D.: The wave kernel signature: A quantum mechanical approach to shape analysis. In: 2011 IEEE international conference on computer vision workshops (ICCV workshops). pp. 1626–1633. IEEE (2011)
4. Baran, I., Popović, J.: Automatic rigging and animation of 3D characters. ACM Transactions on graphics (TOG) **26**(3), 72–es (2007)
5. Benchekroun, O., Zhang, J.E., Chaudhuri, S., Grinspun, E., Zhou, Y., Jacobson, A.: Fast complementary dynamics via skinning eigenmodes. arXiv preprint arXiv:2303.11886 (2023)
6. Chadwick, J.E., Haumann, D.R., Parent, R.E.: Layered construction for deformable animated characters. ACM Siggraph Computer Graphics **23**(3), 243–252 (1989)
7. Chen, R.T.Q., Rubanova, Y., Bettencourt, J., Duvenaud, D.K.: Neural ordinary differential equations. In: Advances in Neural Information Processing Systems. vol. 31 (2018)
8. Dutt, N.S., Muralikrishnan, S., Mitra, N.J.: Diffusion 3d features (diff3f): Decorating untextured shapes with distilled semantic features (2023)
9. Fan, Y., Litven, J., Pai, D.K.: Active volumetric musculoskeletal systems. ACM Transactions on Graphics (TOG) **33**(4), 1–9 (2014)
10. Fragkiadaki, K., Levine, S., Felsen, P., Malik, J.: Recurrent network models for human dynamics. In: ICCV (2015)
11. Harvey, F.G., Yurick, M., Nowrouzezahrai, D., Pal, C.: Robust motion inbetweening **39**(4) (2020)
12. He, C., Saito, J., Zachary, J., Rushmeier, H., Zhou, Y.: Nemf: Neural motion fields for kinematic animation. In: NeurIPS (2022)
13. Jacobson, A., Deng, Z., Kavan, L., Lewis, J.P.: Skinning: Real-time shape deformation. In: SIGGRAPH Courses (2014)
14. Ju, T., Schaefer, S., Warren, J.: Mean value coordinates for closed triangular meshes. In: SIGGRAPH (2005)
15. Lewis, J.P., Anjyo, K., Rhee, T., Zhang, M., Pighin, F., Deng, Z.: Practice and theory of blendshape facial models. Eurographics State of the Art Reports (2014)

16. Li, P., Aberman, K., Hanocka, R., Liu, L., Sorkine-Hornung, O., Chen, B.: Learning skeletal articulations with neural blend shapes. ACM Transactions on Graphics (TOG) **40**(4), 1–15 (2021)
17. Li, Y., Takehara, H., Taketomi, T., Zheng, B., Nießner, M.: 4dcomplete: Non-rigid motion estimation beyond the observable surface. IEEE International Conference on Computer Vision (ICCV) (2021)
18. Li, Y., Takehara, H., Taketomi, T., Zheng, B., Nießner, M.: 4dcomplete: Non-rigid motion estimation beyond the observable surface. (2021)
19. Liao, Z., Yang, J., Saito, J., Pons-Moll, G., Zhou, Y.: Skeleton-free pose transfer for stylized 3d characters. In: European Conference on Computer Vision (ECCV). Springer (October 2022)
20. Loper, M., Mahmood, N., Romero, J., Pons-Moll, G., Black, M.J.: SMPL: A skinned multi-person linear model. ACM Trans. Graphics (Proc. SIGGRAPH Asia) **34**(6), 248:1–248:16 (Oct 2015)
21. Mahmood, N., Ghorbani, N., Troje, N.F., Pons-Moll, G., Black, M.J.: AMASS: Archive of motion capture as surface shapes. In: International Conference on Computer Vision. pp. 5442–5451 (Oct 2019)
22. Marsot, M., Rekik, R., Wuhrer, S., Franco, J.S., Olivier, A.H.: Correspondence-free online human motion retargeting (2023)
23. Modi, V., Fulton, L., Jacobson, A., Sueda, S., Levin, D.: Emu: Efficient muscle simulation in deformation space. Computer Graphics Forum (Dec 2020). `https://doi.org/10.1111/cgf.14185`, `http://dx.doi.org/10.1111/cgf.14185`
24. Nicolet, B., Jacobson, A., Jakob, W.: Large steps in inverse rendering of geometry. ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia) **40**(6) (Dec 2021). `https://doi.org/10.1145/3478513.3480501`, `https://rgl.epfl.ch/publications/Nicolet2021Large`
25. Park, S.I., Hodgins, J.K.: Data-driven modeling of skin and muscle deformation. In: ACM SIGGRAPH 2008 papers, pp. 1–6 (2008)
26. Pons-Moll, G., Romero, J., Mahmood, N., Black, M.J.: Dyna: A model of dynamic human shape in motion. ACM Transactions on Graphics (TOG) **34**(4), 1–14 (2015)
27. Qiao, Y.L., Gao, L., Lai, Y.K., Xia, S.: Learning bidirectional lstm networks for synthesizing 3d mesh animation sequences (2018)
28. Richardson, E., Metzer, G., Alaluf, Y., Giryes, R., Cohen-Or, D.: Texture: Text-guided texturing of 3d shapes. In: ACM SIGGRAPH 2023 Conference Proceedings (2023). `https://doi.org/10.1145/3588432.3591503`
29. Romero, C., Casas, D., Pérez, J., Otaduy, M.: Learning contact corrections for handle-based subspace dynamics. ACM Trans. Graph. **40**(4) (jul 2021)
30. Rong, G., Cao, Y., Guo, X.: Spectral mesh deformation. The Visual Computer **24** (2008)
31. Santesteban, I., Garces, E., Otaduy, M.A., Casas, D.: SoftSMPL: Data-driven Modeling of Nonlinear Soft-tissue Dynamics for Parametric Humans. Computer Graphics Forum (Proc. Eurographics) (2020)
32. Sinha, S., Shapovalov, R., Reizenstein, J., Rocco, I., Neverova, N., Vedaldi, A., Novotny, D.: Common pets in 3d: Dynamic new-view synthesis of real-life deformable categories. CVPR (2023)
33. Sumner, R.W., Popović, J.: Deformation transfer for triangle meshes. ACM Trans. Graph. **23**(3), 399–405 (2004)
34. Wang, J., Li, X., Liu, S., Mello, S.D., Gallo, O., Wang, X., Kautz, J.: Zero-shot pose transfer for unrigged stylized 3d characters (2023)
35. Wang, Y., Shao, T., Fu, K., Mitra, N.: Learning an intrinsic garment space for interactive authoring of garment animation. ACM Trans. Graph. **38**(6) (2019)

36. Xu, Z., Zhou, Y., Kalogerakis, E., Landreth, C., Singh, K.: Rignet: Neural rigging for articulated characters. ACM Trans. on Graphics **39** (2020)
37. Xu, Z., Zhou, Y., Kalogerakis, E., Singh, K.: Predicting animation skeletons for 3d articulated models via volumetric nets. In: 2019 International Conference on 3D Vision (3DV) (2019)
38. Yifan, W., Aigerman, N., Kim, V.G., Chaudhuri, S., Sorkine-Hornung, O.: Neural cages for detail-preserving 3d deformations. In: CVPR (2020)
39. Yu, Y., Zhou, K., Xu, D., Shi, X., Bao, H., Guo, B., Shum, H.Y.: Mesh editing with poisson-based gradient field manipulation. ACM Trans. Graph. **23**(3), 644–651 (aug 2004)
40. Zhang, J.E., Bang, S., Levin, D.I.W., Jacobson, A.: Complementary dynamics. In: SIGGRAPH Asia (2020)
41. Zheng, M., Zhou, Y., Ceylan, D., Barbic, J.: A deep emulator for secondary motion of 3d characters. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5932–5940 (2021)
42. Zhou, K., Bhatnagar, B.L., Pons-Moll, G.: Unsupervised shape and pose disentanglement for 3d meshes. In: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXII 16. pp. 341–357. Springer (2020)