Dog Code: Human to Quadruped Embodiment using Shared Codebooks

Dónal Egan
Trinity College Dublin
Dublin, Ireland
doegan@tcd.ie

Alberto Jovane
Trinity College Dublin
Dublin, Ireland
jovanea@tcd.ie

Jan Szkaradek
Trinity College Dublin
Dublin, Ireland
szkaradj@tcd.ie

George Fletcher
Bath University
Bath, UK
gf321@bath.ac.uk

Darren Cosker
Microsoft
Cambridge, UK
coskerdarren@microsoft.com

Rachel McDonnell
Trinity College Dublin
Dublin, Ireland
ramcdonn@tcd.ie

Figure 1: Example results from our novel human-to-quadruped mapping solution. The human's movements are tracked using a HMD together with two hand-controllers and three additional trackers placed on the pelvis and feet and, in an initial step, are *roughly* retargeted to the quadruped skeleton. This initial retarget is then passed through our novel neural network architecture, centred around a *shared codebook* constructed using finite scalar quantisation, to generate realistic quadruped motion that is highly synchronised with the human's movement.

Abstract

Many VR animal embodiment sytsems suffer from poor animation fidelity, typically animating the animal avatars using inverse kinematics. We address this issue, presenting a novel deep-learning method, centred around a *shared codebook*, for mapping human motion to quadruped motion. Rather than trying to directly bridge the gap from human motion to quadruped motion, a task which has proven difficult, we first use a rule-based retargeter, relying on inverse and forward kinematics, to retarget human motions to an intermediate motion domain in which the motions share the same skeleton as the quadruped. We then use *finite scalar quantization* to construct a shared latent space, or *codebook*, between this intermediate domain and the quadruped motion domain. We do this by first pre-defining a finite number of discrete latent codes and then *teaching* these codes, using unsupervised deep-learning, to represent semantically similar motions in the two domains. We incorporate our real-time human-to-quadruped motion mapping into a VR quadruped embodiment system. The output quadruped animations are natural and realistic, while also preserving the semantics of users' actions. Moreover, there is a strong synchrony between the input human motions and retargeted quadruped motions, an important factor for inducing a strong sense of VR embodiment.

MIG'24, November 21–23, 2024, Virginia, USA

ACM ISBN 979-8-4007-1090-2/24/11
https://doi.org/10.1145/3677388.3696339

CCS Concepts

• **Computing methodologies** → **Motion processing**; Machine learning.

Keywords

VR embodiment, Quadruped embodiment, Motion retargeting, Deep-learning

ACM Reference Format:
Dónal Egan, Alberto Jovane, Jan Szkaradek, George Fletcher, Darren Cosker, and Rachel McDonnell. 2024. Dog Code: Human to Quadruped Embodiment using Shared Codebooks. In *The 16th ACM SIGGRAPH Conference on Motion, Interaction, and Games (MIG '24), November 21–23, 2024, Arlington, VA, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3677388.3696339

1 Introduction

Virtual reality (VR) embodiment of quadrupedal avatars is a challenging problem. Extreme morphological differences between humans and quadrupeds, together with the complexities of their particular motions, make the construction of a real-time mapping between the human and quadruped motion domains a challenging task. Despite giving the user a high level of agency, retargeting their motion to the virtual quadruped solely using inverse kinematics (IK) can result in the quadruped's movements looking unnatural, anthropomorphic or robotic. Potentially, this can diminish the embodiment experiment [Egan et al. 2023].

Towards this, we present our novel human-to-quadruped motion mapping solution, which we incorporate into a VR quadruped embodiment system. Our mapping synthesises realistic and natural looking motion for the virtual quadruped while preserving the semantics of the users' actions. Importantly, our method preserves a strong synchrony between the human and quadruped motions, an important factor for enhancing the VR embodiment experience [Kokkinara and Slater 2014].

Directly bridging the domain gap between human and quadruped motion is difficult and many state-of-the-art deep-learning methods for human-to-human retargeting [Aberman et al. 2020; Villegas et al. 2018] do not readily extend to the human-to-quadruped case. Rather than trying to directly bridge this gap, we perform the mapping in several stages. First, we partially retarget human motions to the quadruped skeleton using a rule-based retargeter relying on inverse and forward kinematics. By *partially*, we mean that we only retarget motions between a subset of the human and quadruped skeletons' kinematic chains. Motions in this intermediate domain preserve the semantics and timing of human motion but lack the subtle details of true quadruped motion. Then, in a second step, we use unsupervised deep-learning to bridge the gap between this intermediate motion domain and the domain of true quadruped motion. This step can be viewed as enhancing the realism of the partial retarget resulting from step one. Third, we *fill in blanks* on the quadruped poses, animating those parts of the pose not dealt with in steps one and two, using both deep learning and inverse kinematics. Our key technical novelty lies in the second step where we use *finite scalar quantization* [Mentzer et al. 2024] to construct a shared latent space between the intermediate motion domain and the quadruped motion domain. This shared latent space contains a finite number of pre-defined discrete latent codes, i.e., the codes themselves are not learned but rather only their meanings are. Jointly training an autoencoder for each domain, we teach each latent code to represent semantically similar motions in both domains. Thus, our latent space can be seen as a *shared codebook*, in which the *codes* represent the semantics, or *content*, of motions and the decoders add the domain-specific subtleties, or *style*, to the motions. After applying the rule-based retargeter to a human motion, we can apply the intermediate motion domain's encoder to the result to obtain the appropriate codebook entries, and decode these using the quadruped motion decoder, thus allowing us to bridge the gap between human and quadruped motion. We evaluate our work both qualitatively and quantitatively, comparing it to a baseline system relying on IK, and through an ablation study in which we systematically omit different components of our method to examine their impact on the results.

In summary, our main contribution is a novel architecture, leveraging a shared codebook constructed using finite scalar quantization [Mentzer et al. 2024], for mapping human motion to quadruped motion. We incorporate this mapping into a new VR quadruped embodiment system in which the virtual quadruped's motions are natural and realistic, while still preserving synchrony with the user. We also incorporate dynamically switching limb-control to the VR embodiment system, in which control of the quadruped's front paws naturally shifts from the human's legs to their arms, depending on the action of the human - resulting in more intuitive control of the virtual quadruped.

2 Related Work

We highlight some related literature. In particular, we look at works on retargeting motion between humans and animals, the sense of embodiment in virtual reality, and data-driven techniques for animating avatars in virtual reality.

2.0.1 Human-to-animal retargeting. Retargeting is the problem of transferring motion from a source character to a target character [Gleicher 1998]. Extreme differences in morphology (e.g., biped, quadruped, apod) and types of motion (e.g., humans walk and run, kangaroos hop, and snakes crawl) can make retargeting between humans and animals more challenging. Many existing solutions are more akin to real-time puppetry systems with the semantics of the input human motion not necessarily being preserved. For example, Komura and Lam [2006] and Jiang et al. [2023] use human hand motions to control animal avatars, while Rhodin et al. [2015] estimate wave properties (amplitude, frequency, and phase) of human wave gestures and use these to animate virtual animals via a parametric motion graph [Heck and Gleicher 2007] in which example animal animations have been annotated with with amplitude, frequency, and phase values. We aim to have a stronger semantic relationship between the human and animal motions. For example, if the user walks or sits, the animal walks or sits. Other methods define key pose pairs for the human and animal characters and use these to construct a mapping between human and animal poses. Yamane et al. [2010] construct this mapping using a shared Gaussian process latent variable model (GPLVM) [Shon et al. 2005], while Celikcan et al. [2015] use a space-time solver. Seol et al. [2013] create a frame-by-frame correspondence by recording a reference human motion clip of equal length for each target animal animation. At run-time, the correct human motion clip is identified using a support vector machine (SVM) and the most appropriate frame in this clip is found via a nearest neighbour search. The corresponding animal pose can be played. The requirement to define key-pose pairs can limit the diversity of both the actions the user can perform and the target animal animations. In contrast, we use unsupervised deep-learning with human and animal motion capture datasets to increase the diversity of allowed input human actions and generated animal animations.

Current state-of-the-art methods for human-to-human retargeting [Aberman et al. 2020; Villegas et al. 2018] are data-driven and use unsupervised deep-learning. Aberman et al. [2020] jointly train motion autoencoders for each human skeleton structure in

an attempt to learn a common latent space in which the skeleton information and semantics of an embedded motion sequence are decoupled. Retargeting can be performed by combining the encoder for one skeleton with the decoder for another. However, these methods do not easily extend to the human-to-animal case with Villegas et al. [2018] requiring that the source and target skeletons differ only in the length of the bones and Aberman et al. [2020] requiring that they be topologically equivalent. Our work is influenced by Aberman et al. [2020]. For example, we also learn a common latent space and use adversarial training to encourage the appropriateness of retargeted motions for the target character and kinematic losses to preserve the semantics of input motions. However, there are also noticeable differences between our works. Foremost of these are our use of a rule-based retargeter to enable us to overcome the morphology gap between humans and quadrupeds and the use of a finite and discrete latent space.

Recent works have looked at applying deep-learning to the human-to-non-human retargeting problem. Kim et al. [2022] learn a mapping between human motion and quadruped robot motion in a supervised fashion. However, obtaining a paired dataset of human and animal motion is not a trivial task, making their method unsuitable for certain situations, including ours. Li et al. [2023] use adversarial training [Goodfellow et al. 2014] to train a neural network to encode human motion into the learned latent space of a pre-trained motion prior for the target animal. This motion prior can then be used to generate the target animal motion. Very recently, Li et al. [2024] introduced the vector quantized periodic autoencoder to learn a shared phase [Holden et al. 2017; Mason et al. 2022; Starke et al. 2022, 2020] manifold for human and quadruped motion. Retargeting can then be performed in the phase space rather than the pose space as happens in all of the previously discussed methods. While this method preserves the notion of phase between human and quadruped motions and produces high quality quadruped motion, it does not guarantee a synchrony between specific human and quadruped limbs - something which we believe is important for creating VR embodiment scenarios that provide a user with a strong sense of agency. Similar to our work, they also use a finite and discrete latent space. However, while they use vector quantization [van den Oord et al. 2017] for the discretization process, we use finite scalar quantization [Mentzer et al. 2024]. Reda et al. [2023] use reinforcement learning to control a physics-based simulation of a target non-human character. An advantage of their method is that they only require human motion data for training. However, they only test their method with bipedal target characters.

2.0.2 Sense of embodiment in virtual reality. Kilteni et al. [2012] describe the *sense of embodiment* (SoE) of a virtual avatar as consisting of three sub-components, namely the *sense of self-location*, the *sense of agency* and the *sense of body-ownership*. These terms refer to a user's feeling that they are located inside their virtual avatar's body, their feeling that they are the cause of their avatar's movements, and their feeling that their avatar's body is the source of their experiences sensations, respectively. While most studies into the SoE in VR focus on humanoid and extended humanoid [Egeberg et al. 2016; Hoyet et al. 2016; Steptoe et al. 2013] avatars, Krekhov et al. [2019b,a] find that the illusion of virtual body-ownership is also applicable to animal avatars. However, most studies involving VR animal embodiment animate the animal avatars using traditional inverse kinematics (IK) methods [Ahn et al. 2016; Ichikawa et al. 2022; Krekhov et al. 2019b; Oyanagi and Ohmura 2019; Pimentel and Kalyanaraman 2022; Sierra Rativa et al. 2020]. While a strong sense of agency can be achieved using IK, the resulting animal animations can be unrealistic, potentially diminishing a user's embodiment experience. Egan et al. [2023] show that animation fidelity is an important factor to consider for virtual animal embodiment.

2.0.3 Data-driven animation of VR avatars. Realistically animating VR avatars, both human and animal, is challenging due to the sparse input signals - typically a HMD and two hand controllers, and potentially a small amount of additional trackers placed on, for example, the feet and hips. Ponton et al. [2022] and Ahuja et al. [2021] retrieve animations from human motion capture databases based on the tracking data from a HMD and two hand controllers, while Jiang et al. [2022] and Du et al. [2023] use transformer and diffusion architectures, respectively, to solve the same task. Similar to our work, Ponton et al. [2023] assume six tracking sensors and use a convolutional architecture. While these works focus on humanoid avatars, very few works have explored data-driven techniques for realistically animating animal avatars. In one of the first such works, Egan et al. [2023] train a neural network to synthesise realistic quadruped motion conditioned on information such as action, trajectory, foot velocities, and ground contacts. For training, the conditioning input information is extracted from quadruped motion capture data, while at run-time it comes from the human user. In contrast to our work, no human data is used in the learning process.

3 Human-to-quadruped motion mapping

Let $\mathcal{M}^H$ and $\mathcal{M}^Q$ denote the spaces of human and quadruped motion, respectively. The goal is to construct a mapping which takes as input a human motion sequence $\mathbf{m}^H \in \mathcal{M}^H$ and outputs a quadruped motion sequence $\mathbf{m}^Q \in \mathcal{M}^Q$. The timing and semantics of $\mathbf{m}^H$ should be preserved by the mapping.

Directly bridging the gap from $\mathcal{M}^H$ to $\mathcal{M}^Q$ in a single step using deep-learning has proven a difficult challenge, with state-of-the-art human-to-human retargeting solutions not extending trivially to the human-to-quadruped case [Aberman et al. 2020]. Thus, we propose to do it in three stages. First, we partially retarget human motion sequences to the quadruped skeleton using a rule-based retargeter R, relying on forward and inverse kinematics. Defining correspondences between a subset of the human and quadruped skeletons' kinematic chains, we use R to retarget motions between these chains. These initial partial retargets, the space of which we denote by $\mathcal{M}^I$, preserve the semantics and timing of the human motions but lack the realism of quadruped motion. In a second step, we use unsupervised deep-learning to improve the realism of these initial partial retargets. Our core technical novelty lies in this step where we use *finite scalar quantization* [Mentzer et al. 2024] to construct a shared latent space between $\mathcal{M}^I$ and $\mathcal{M}^Q$. This shared latent space consists of a finite number of pre-defined discrete latent codes, each representing semantically similar motions in both $\mathcal{M}^I$ and $\mathcal{M}^Q$. By encoding and decoding in and out of this shared latent space, we can map the initial partial retargets from $\mathcal{M}^I$ to more

realistic partial retargets in $\mathcal{M}^Q$. In the third and final step, we *fill in the blanks*, predicting the motion for the remaining body parts of the target quadruped that have not been dealt with in the previous steps. We also use deep-learning for this step.

3.0.1 Design choices. The problem setting is an ambiguous one - for example, what human and quadruped body parts and motions should correspond - and several design choices must be made. One important decision is whether to assume the human adopts a bipedal or quadrupedal stance. Given our target application is VR embodiment, we assume a bipdal stance for the human, citing Krekhov et al. [2019b]'s finding that users find a quadrupedal stance significantly more exhausting for VR embodiment. Additionally, as previous research [Kokkinara and Slater 2014] suggests the embodiment experience is enhanced by synchrony between the user's and avatar's motions, we aim to preserve a synchrony between some or all of the user's and quadruped's limbs. However, there is a trade-off between this goal and our other goal of synthesising realistic quadruped motion. For human locomotion, the left and right legs move half a stride out of phase with each other. This is similar to either the front legs or the rear legs of a quadruped during a walk, trot or a pace. However, the front and rear legs of the quadruped do not move exactly in stride or exactly half a stride out of phase with each other[1]. Thus, for example, mapping all four human limbs to all four quadruped legs or each human leg to two quadruped legs could make it difficult to synthesise quadruped locomotion with realistic timings. Thus, we make the decision to map the motion of two human limbs to the quadruped's front legs. Egan et al. [2023] also employ this method, mapping the user's legs to the quadruped's front legs. While this works well for standing and locomotion, we noted that when the human transitioned to sitting in their system, lifting the legs to control the front paws looked cumbersome and not very intuitive. Therefore, we introduce a dynamically switching limb correspondence between the human and quadruped, in which control of the quadruped's front paws naturally shifts from the human's legs to their arms, depending on the action of the human (see Figure 5).

Another issue to be considered is how to transform the human trajectory into that of a quadruped. We want to preserve synchrony between the human and quadruped steps. For example, if the human takes ten steps, then the quadruped should take ten steps. However, a human and quadruped could travel significantly different distances in ten steps due to different stride lengths. Thus, maintaining a direct correspondence between trajectories would lead to noticeable foot sliding on the quadruped. To mitigate this, we apply a scale transformation to the trajectories, proportionate to the size of the characters.

3.0.2 Notation. Where relevant, we express a motion sequence as the concatenation of the motion sequences of the skeleton's individual kinematic chains. That is, we also express a human motion sequence $\mathbf{m}^H \in \mathcal{M}^H$ as $\mathbf{m}^H = [\mathbf{m}^H_{RL}, \mathbf{m}^H_{LL}, \mathbf{m}^H_{RA}, \mathbf{m}^H_{LA}, \mathbf{m}^H_H]$ where the individual components represent the motion of the kinematic chains corresponding to the right leg, left leg, right arm, left arm, and head, respectively. Similarly, for $d \in \{I, Q\}$, we express a sequence $\mathbf{m}^d \in \mathcal{M}^d$ as $\mathbf{m}^d = [\mathbf{m}^d_{RFL}, \mathbf{m}^d_{LFL}, \mathbf{m}^d_{RBL}, \mathbf{m}^d_{LBL}, \mathbf{m}^d_T, \mathbf{m}^d_H]$ where the kinematic chains correspond to the right front leg, left front leg, right back leg, left back leg, tail, and head, respectively. Where the context is understood, we can use $\mathbf{m}$ to represent the motion of either a subset, or all, of the kinematic chains. We use $\mathbf{m}^{d_1 \to d_2}$ to denote the motion sequence obtained by retargeting a motion sequence $\mathbf{m}^{d_1}$ from $\mathcal{M}^{d_1}$ to $\mathcal{M}^{d_2}$, where $d_1, d_2 \in \{H, I, Q\}$. Depending on the context, we may use $\mathbf{m}^{d_1 \to d_2}$ to represent the result of retargeting the motion of some, or all, of the kinematic chains. As we will see below, the retargeting method used depends on the values of d_1 and d_2. We use a lower case subscript to denote an individual frame of a motion sequence, for example, $\mathbf{m}^H_t$, $\mathbf{m}^Q_{LFL_t}$ or $\mathbf{m}^{H \to Q}_{RBL_t}$.

In the following sections, we explain in detail the three stages of our human to quadruped motion mapping.

3.1 Step 1: Rule-based retargeter

Given a human motion sequence $\mathbf{m}^H$, our first step is to use a rule-based retargeter R, to obtain an initial partially retargeted motion sequence $\mathbf{m}^{H \to I} = R(\mathbf{m}^H) \in \mathcal{M}^I$. Following our design choices, the employed retargeter relies on kinematic correspondence between human and dog limbs. Consequently, a target chain pose is constructed using a sequence of forward kinematics to apply the source chain shape, and inverse kinematics to adjust the end-effector position. The retargeter dynamically switches between two states depending on the action of the human. The first state is used for idling and locomotion. When the human is in a bipedal stance, the limb correspondence is between their legs and the quadruped's front legs, resulting in locomotion of the human and quadruped being synchronized. The second state is used for actions performed while sitting. In this case, the limb correspondence is between the human's arms and the quadruped's front legs. This affords users embodying the quadruped intuitive control of the paw-raising action. The retargeter dynamically switches between the states based on the height of the human's hips.

In summary, depending on the action of the human, our initial partial retarget is given by

$$\mathbf{m}^{H \to I} = [\mathbf{m}^{H \to I}_{RFL}, \mathbf{m}^{H \to I}_{LFL}] = R([\mathbf{m}^H_{RL}, \mathbf{m}^H_{LL}]) \text{ or } R([\mathbf{m}^H_{RA}, \mathbf{m}^H_{LA}]).$$

The motion $\mathbf{m}^{H \to I}$ preserves the semantics, synchrony and timing of $\mathbf{m}^H$ but is not realistic quadruped motion. Note that the second and third stages of our method (sections 3.2 and 3.3, respectively) are not necessarily dependant on R, and so in theory different design choices for R are possible.

3.2 Step 2: Shared codebook

In our second step, we use deep-learning to improve the realism of the initial partial retargets obtained using R in step one. The core technical novelty of our method is the use of *finite scalar quantization* (FSQ) [Mentzer et al. 2024] to construct a *shared codebook* between the two spaces $\mathcal{M}^I$ and $\mathcal{M}^Q$, such that each code represents semantically similar motions in both. By encoding and decoding an initial partial retarget $\mathbf{m}^I \in \mathcal{M}^I$ obtained in step one into and out of this shared codebook, we obtain a more realistic partial retarget $\mathbf{m}^{I \to Q} \in \mathcal{M}^Q$.

[1] Alexander [2003] provides a comprehensive discussion of animal locomotion.

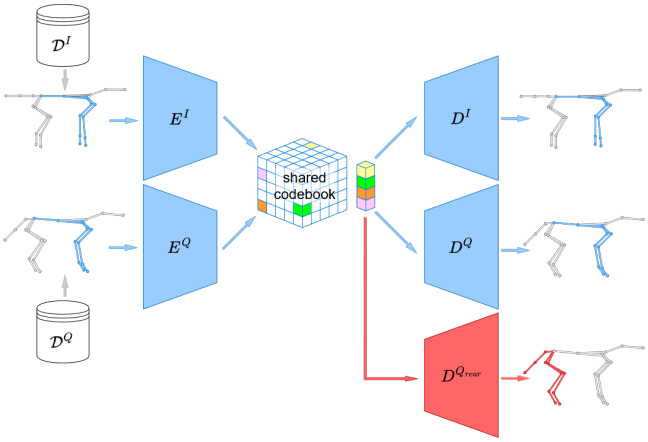

Figure 2: Training architecture: A shared codebook between $\mathcal{M}^I$ and $\mathcal{M}^Q$ is learned by jointly training two autoencoders (blue) to encode semantically similar initial partial retargets from $\mathcal{D}^I$ and real partial quadruped motions from $\mathcal{D}^Q$ to the same sequence of pre-defined discrete latent codes. Then, with the autoencoders frozen, another decoder $D^{Q_{rear}}$ is trained to generate realistic and appropriate rear motion for the virtual quadruped.

3.2.1 Finite scalar quantization. FSQ maps an n-dimensional vector $\mathbf{z} = \{z_1, z_2, \ldots, z_n\} \in \mathbb{R}^n$ to one of L^n unique possible vectors for some integer hyperparameter L. Each scalar component z_j of $\mathbf{z}$ is mapped to one of L integer values via $z_j \mapsto \text{round}\left(\lfloor \frac{L}{2} \rfloor \tanh\left(z_j\right)\right)$[2]. By applying FSQ to the encoder output of an autoencoder, the latent space becomes a codebook with a finite number, L^n, of pre-defined discrete latent codes. This is in contrast to the VQ-VAE [van den Oord et al. 2017] in which the codebook is learned (which also means that less parameters need to be learned during training with FSQ). Typically, n is also smaller for FSQ than for the VQ-VAE. For network training, gradients are back-propagated through the FSQ operator using the *straight-through estimator* [Bengio et al. 2013]. Unlike the VQ-VAE, FSQ does not require any auxiliary losses for training.

3.2.2 Construction of the shared codebook. Our network architecture is illustrated in Figure 2. We jointly train two autoencoders to learn a shared latent space between partial retargets in $\mathcal{M}^I$ and partial quadruped motions in $\mathcal{M}^Q$. Each autoencoder consists of an encoder and a decoder. We denote the encoder and decoder corresponding to $\mathcal{M}^I$ by E^I and D^I and those corresponding to $\mathcal{M}^Q$ by E^Q and D^Q . The inputs to E^I are partially retargeted sequences $[\mathbf{m}^I_{RFL}, \mathbf{m}^I_{LFL}]$, while D^I outputs reconstructions of such sequences. Similarly, the inputs to E^Q are sequences $[\mathbf{m}^Q_{RFL}, \mathbf{m}^Q_{LFL}]$ of real quadruped motion, while D^Q reconstructs such sequences.

The encoders and decoders all have convolutional architectures, employing 1D convolutions across the temporal dimension [Holden et al. 2016]. An encoder E^d (for $\{d \in I, Q\}$) contains two identical convolutional blocks for independently encoding the sequences $\mathbf{m}^d_{RFL}$ and $\mathbf{m}^d_{LFL}$. The convolutional layers of E^d encode a sequence $[\mathbf{m}^d_{RFL}, \mathbf{m}^d_{LFL}]$ of length τ as a sequence $[\mathbf{z}_{RFL}, \mathbf{z}_{RFL}]$ of length τ/s where s represents the temporal down-sampling rate (we set $s = 4$) and $\mathbf{z}_{RFL_t}, \mathbf{z}_{LFL_t} \in \mathbb{R}^n$ for each frame t.

We apply FSQ separately to $\mathbf{z}_{RFL_t}$ and $\mathbf{z}_{LFL_t}$ (for $t = 1, 2, \ldots, \tau/2$), resulting in them being replaced by two of L^n pre-defined codes. These discrete codes are concatenated along the spatial dimension to obtain the final latent code $\mathbf{z}$ - a sequence of τ/s codes, each of dimension $2n$. The decoders D^I and D^Q reconstruct partial motion sequences $[\hat{\mathbf{m}}^I_{RFL}, \hat{\mathbf{m}}^I_{LFL}] \in \mathcal{M}^I$ and $[\hat{\mathbf{m}}^Q_{RFL}, \mathbf{m}^Q_{LFL}] \in \mathcal{M}^Q$, respectively, from this latent code $\mathbf{z}$. Both decoders consist of a single 3-layer convolutional block and use linear upsampling to recover the temporal resolution.

By applying FSQ in this way as the final layer of both encoders E^Q and E^I, using the same values for the hyperparameters L and n for both, the latent spaces of the two autoencoders will, by construction, contain the same finite set of discrete latent codes. The task is to train the two autoencoders so that each of these codes learns to represent semantically similar motions in $\mathcal{M}^I$ and $\mathcal{M}^Q$, thus giving us our shared codebook C with $|C| = (L^n)^2$. We achieve this by jointly training the two autoencoders with a range of loss functions. The training process is described in detail in section 4.

Once trained, given an initial partially retargeted motion sequence $[\mathbf{m}^I_{RFL}, \mathbf{m}^I_{LFL}]$, we can combine the encoder E^I with the decoder D^Q to obtain a more realistic sequence of partial quadruped motion given by

$$[\mathbf{m}^{I\to Q}_{RFL}, \mathbf{m}^{I\to Q}_{LFL}] = D^Q\left(E^I\left([\mathbf{m}^I_{RFL}, \mathbf{m}^I_{LFL}]\right)\right) \tag{1}$$

The sequence $[\mathbf{m}^{I\to Q}_{RFL}, \mathbf{m}^{I\to Q}_{LFL}]$ has improved realism while still preserving the timing and semantics of the input partial retarget (and the human motion sequence used to generate it).

3.3 Step 3: Filling in the blanks

Given a sequence $\mathbf{z}$ of shared codebook entries, generated by encoding the front leg kinematic chains of either a real quadruped motion sequence or of an initial partial retarget, we use a separate decoder $D^{Q_{rear}}$ to decode $\mathbf{z}$ into realistic quadruped motion for the rear legs and tail kinematic chains, such that the generated motion is appropriate for the front leg motion given by $D^Q(\mathbf{z})$ (see figure 2). The decoder $D^{Q_{rear}}$ has a similar structure to D^I and D^Q. Training of $D^{Q_{rear}}$ happens after that of the two autoencoders, keeping them frozen during the process. Full details of the training procedure are given in section 4.0.2.

Finally, we animate the quadruped's head kinematic chain using IK based on the orientation of the human head. We use inverse kinematics for the head animation as it is important for the quadruped's head to accurately follow the user's head in the VR setting. However, the neural networks could be trained to also synthesise the head motion for using the mapping in an offline setting.

[2]A slight adjustment is required if L is even.

Putting the three steps together, a human motion sequence $\mathbf{m}^H$ can be mapped to a quadruped motion sequence $\mathbf{m}^{H\to Q}$ by

$$\mathbf{m}^{H\to Q} = [\underbrace{D^Q\left(E^I\left(R(\mathbf{m}^H)\right)\right)}_{\mathbf{m}^{H\to Q}_{RFL},\mathbf{m}^{H\to Q}_{LFL}}, \underbrace{D^{Q_{rear}}\left(E^I\left(R(\mathbf{m}^H)\right)\right)}_{\mathbf{m}^{H\to Q}_{RBL},\mathbf{m}^{H\to Q}_{LBL},\mathbf{m}^{H\to Q}_{T}}, \underbrace{IK\left(m^H_H\right)}_{\mathbf{m}^{H\to Q}_{H}}] \tag{2}$$

where IK denotes the inverse kinematics operator. The human sequence is partially retargeted to the quadruped skeleton using the rule based retargeter R. The resulting sequence is encoded using E^I to obtain a sequence of codes from the shared codebook, which are then decoded by D^Q and $D^{Q_{rear}}$ to generate realistic motion for the quadruped body. The pipeline for this process is illustrated in Figure 3.

4 Network training

We start with two datasets, $\mathcal{D}^H$ and $\mathcal{D}^Q$, containing sample motion sequences of length τ from $\mathcal{M}^H$ and $\mathcal{M}^Q$, respectively. We apply the rule-based retargeter R (section 3.1) to each sequence in $\mathcal{D}^H$ to obtain a dataset $\mathcal{D}^I := R(\mathcal{D}^H)$ consisting of inital partial retargets corresponding to $\mathcal{M}^I$. Note that the datasets are unpaired amd there are not corresponding sequences between $\mathcal{D}^H$ or $\mathcal{D}^I$ and $\mathcal{D}^Q$. The two autoencoders, i.e., the networks E^I, D^I, E^Q, and D^Q, are trained together first. Then, we freeze these, and train the network $D^{Q_{rear}}$. We describe the training procedures in detail in the following sections.

4.0.1 Jointly training the autoencoders. At each training iteration we sample a sequence $\mathbf{m}^I \in \mathcal{D}^I$ and a sequence $\mathbf{m}^Q \in \mathcal{D}^Q$. Note that for ease of notation, we take $\mathbf{m}^I$ and $\mathbf{m}^Q$ to represent $[\mathbf{m}^I_{RFL}, \mathbf{m}^I_{LFL}]$ and $[\mathbf{m}^Q_{RFL}, \mathbf{m}^Q_{LFL}]$ for the remainder of this subsection. Four new sequences are generated. The first two aim to reconstruct the sampled sequences and are given by $\hat{\mathbf{m}}^I = D^I\left(E^I(\mathbf{m}^I)\right)$ and $\hat{\mathbf{m}}^Q = D^Q\left(E^Q(\mathbf{m}^Q)\right)$. The second two aim to map the sequence $\mathbf{m}^I$ to $\mathcal{M}^Q$ and the sequence $\mathbf{m}^Q$ to $\mathcal{M}^I$. They are given by $\mathbf{m}^{I\to Q} = D^Q\left(E^I(\mathbf{m}^I)\right)$ and $\mathbf{m}^{Q\to I} = D^I\left(E^Q(\mathbf{m}^Q)\right)$, respectively.

We describe the loss functions used for learning the generation of $\hat{\mathbf{m}}^Q$ and $\mathbf{m}^{I\to Q}$, with equivalent losses being used for $\hat{\mathbf{m}}^I$ and $\mathbf{m}^{Q\to I}$. For $\hat{\mathbf{m}}^Q$ we use standard reconstruction losses on joint rotations and positions:

$$L_{rec}(\mathbf{m}^Q, \hat{\mathbf{m}}^Q) = ||\mathbf{m}^Q - \hat{\mathbf{m}}^Q||^2 + ||FK(\mathbf{m}^Q) - FK(\hat{\mathbf{m}}^Q)||^2. \tag{3}$$

where FK is the forward kinematics operator used to recover the joint positions from their orientations and offsets [Aberman et al. 2020].

For generating $\mathbf{m}^{I\to Q}$ we use a loss on end-effector positions, a latent consistency loss, and an adversarial loss. The positional loss on the end-effectors is given by

$$L_{ee}(\mathbf{m}^I, \mathbf{m}^{I\to Q}) = ||FK(\mathbf{m}^I)_{\mathcal{J}} - FK(\mathbf{m}^{I\to Q})_{\mathcal{J}}||^2 \tag{4}$$

where $\mathcal{J} = \{left\ front\ foot, right\ front\ foot\}$. The latent consistency loss (similar to [Aberman et al. 2020]) is given by

$$L_{lc}(\mathbf{m}^I, \mathbf{m}^{I\to Q}) = ||E^I(\mathbf{m}^I) - E^Q(\mathbf{m}^{I\to Q})||_1. \tag{5}$$

The purpose of the end-effector positional and latent consistency losses are to try and preserve the synchrony of limb motion between

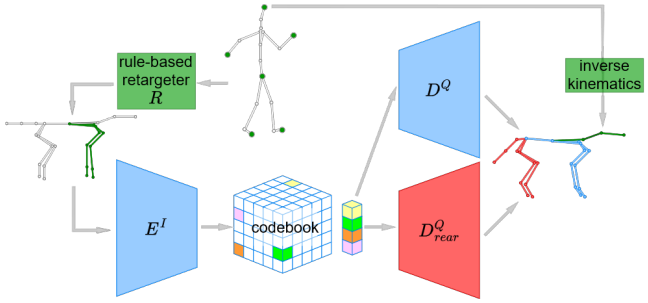

Figure 3: Run-time architecture: The human's arm or leg motion is retargeted to the front limbs of the quadruped skeleton using the rule-based retargeter R. This initial retarget is encoded into the shared codebook by E^I, with the result being decoded by D^Q and $D^{Q_{rear}}$ to generate realistic motion for the front and rear of the quadruped, respectively. The head kinematic chain is animated using IK based on the human's head orientation.

$\mathbf{m}^I$ and $\mathbf{m}^{I\to Q}$ and to encourage the codes to learn to represent semantically similar motions in both domains. We also use a standard adversarial loss [Goodfellow et al. 2014] to encourage the realism of $\mathbf{m}^{I\to Q}$:

$$L_{adv} = -\frac{1}{\tau}\sum_{t=1}^{\tau} log\left(C^Q(\mathbf{m}^{I\to Q}_t)\right) \tag{6}$$

where C^Q is a discriminator network trained to distinguish between real quadruped motion sequences $\mathbf{m}^Q$ and synthesised sequences $\mathbf{m}^{I\to Q}$. The discriminator C^Q is trained using the following loss function.

$$L_{CQ} = -\frac{1}{\tau}\sum_{t=1}^{\tau}\left(log\left(C^Q(\mathbf{m}^Q_t)\right) + log\left(1 - C^Q(\mathbf{m}^{I\to Q}_t)\right)\right) \tag{7}$$

Note that the discriminators C^Q and C^I are MLPs and act on individual frames rather than entire sequences. We also tried a convolutional architecture for the discriminators acting on entire sequences. However, we observed better results using the frame-wise discriminator. This is perhaps due to differences in the timings (for example, the length of time required to take a single stride) of motions in $\mathcal{M}^I$ and $\mathcal{M}^Q$.

4.0.2 Training $D^{Q_{rear}}$. The decoder $D^{Q_{rear}}$ is trained separately, keeping the already trained E^I, D^I, E^Q, and D^Q fixed. At each training iteration we sample a partial initial retarget $\mathbf{m}^I \in \mathcal{D}^I$ and a real quadruped motion sequence $\mathbf{m}^Q \in \mathcal{D}^Q$. Two new sequences are generated: $[\mathbf{m}^{I\to Q}_{RBL}, \mathbf{m}^{I\to Q}_{LBL}, \mathbf{m}^{I\to Q}_{T}] = D^{Q_{rear}}\left(E^I([\mathbf{m}^I_{RFL}, \mathbf{m}^I_{LFL}])\right)$ and $[\hat{\mathbf{m}}^Q_{RBL}, \hat{\mathbf{m}}^Q_{LBL}, \hat{\mathbf{m}}^Q_{T}] = D^{Q_{rear}}\left(E^Q([\mathbf{m}^Q_{RFL}, \mathbf{m}^Q_{LFL}])\right)$. We train $D^{Q_{rear}}$ using a standard reconstruction loss, similar to equation 3, and an adversarial loss (requiring a discriminator $C^{Q_{rear}}$), similar to equations 6 and 7. Note that although $D^{Q_{rear}}$ only predicts the rear leg and tail motion of the quadruped, the discriminator $C^{Q_{rear}}$ for the adversarial loss receives full poses. This is to encourage $D^{Q_{rear}}$'s predictions to be consistent with the already predicted motion for the dog's front leg kinematic chains.

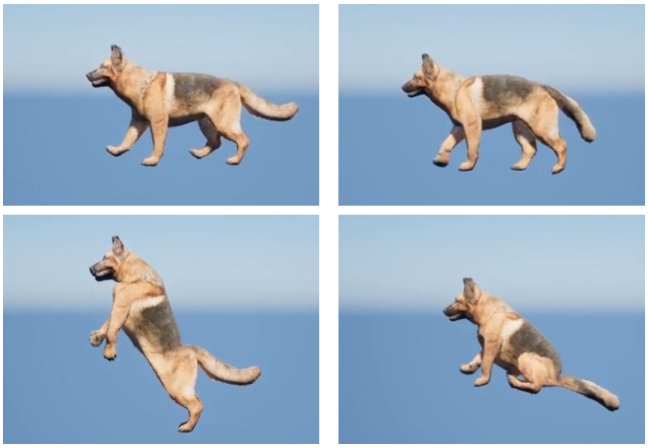

Figure 4: Left: frames generated from a human pose solely using IK have stiff limbs, right: frames generated by our method using same input pose are more natural looking.

5 Implementation details

5.0.1 Datasets. For $\mathcal{D}^Q$ we use the dataset introduced by Zhang et al. [2018]. The dataset contains approximately thirty minutes of unstructured motion capture data for a single dog. The data includes various locomotion gaits (walk, pace, trot, canter, gallop), as well as other actions such as sitting down, paw-raising, lying down, idling, and jumping. We limit ourselves to locomotion, idling, sitting-down, and paw-raising - actions which a user is likely to perform (and can do so safely) in VR.

For $\mathcal{D}^H$, we captured our own dataset of human motion capture data. As the intended final application of our method is use in a VR embodiment system, depending only on consumer-grade devices, we recorded the human motion using a HMD, two hand controllers and three additional trackers placed on the pelvis and feet. We aimed to make the structure (in terms of size and actions) of $\mathcal{D}^H$ similar to that of $\mathcal{D}^Q$ - recording locomotion at various speeds, idling, sitting-down, and hand-raising. While $\mathcal{D}^H$ only contains data for a single person, we can see in the accompanying video that our method generalises to different users with different proportions. Potentially this is due to differences in users' proportions and styles largely being accounted for by the rule-based retargeter R.

Both datasets $\mathcal{D}^I$ and $\mathcal{D}^Q$ are processed in a similar fashion. The only difference is that we slow down each sequence in $\mathcal{D}^Q$ by a factor of three after noting that quadruped strides tend to last for a significantly shorter period of time than human strides. We double the size of each dataset by mirroring the motion sequences. We then break-down the motion sequences into windows of $\tau = 60$ frames (three seconds of motion) with a ten frame overlap. We represent a pose $\mathbf{m}_t$ of a motion sequence $\mathbf{m}$ by the orientation of the skeletal joints together with the position of the hips. Both the joint orientations and the hips position are expressed in a root-centric coordinate system in which the origin is the projection of the smoothed hip position onto the ground plane, the up-axis is the global up-axis, and the forward axis is aligned with the character's facing direction. Joint orientations are parameterized using the 6D continuous representation presented in [Zhou et al. 2019].

5.0.2 Network architecture details. All encoders and decoders use 1-D convolutions in the temporal dimension with a kernel size of 11. Each encoder consists of two parallel blocks (one per kinematic chain). Each block has three convolutional layers, with the hidden layers having 128 units. The first two convolutional layers use a stride of 2 to downsample the temporal resolution by a factor of 2 and are followed by ELU activation layers. For finite scalar quantization, the hyperparameters L and n are set to 3 and 4, respectively. All decoders consist of 3 convolutional layers, with the hidden layers having 256 units. The first two convolutional layers are preceded by linear-upsampling layers, each increasing the temporal resolution by factor of 2, and are followed by ELU activation layers. The discriminators are all 4 layer MLPs, with the hidden layers having 128, 64, and 32 units and the final output layer having a single unit. The first three layers are followed by ELU activation layers and the final is followed by a sigmoid layer.

5.0.3 Neural network training. PyTorch was used for the neural network training implementations. All networks were trained using the Adam optimizer [Kingma and Ba 2017], with a learning rate of 0.0001. A mini-batch size of 64 was used. The training alternated between updating the encoders and decoders or the discriminators at each step. Training of the joint autoencoders (E^I, D^I, E^Q, D^Q)took approximately two and a half hours while training of $D^{Q_{rear}}$ took approximately one hour. Training was done using a Nvidia GeForce RTX™ 3090 GPU.

5.0.4 VR embodiment system. The run-time VR quadruped embodiment system is implemented in Unreal Engine 5. A user's motion is tracked using a VIVE Pro Eye HMD, two hand controllers and three VIVE trackers (3.0) placed on the pelvis and feet. The rule-based retargeter R is implemented using Unreal Engine's IK Rig Retargeting tool. The system is implemented on a PC with the same GPU as specified in section 5.0.3 and an AMD Ryzen 9 5950X 16-Core CPU. We target a frame rate of approximately 70 frames per second (fps) to avoid negative effects on users, such as nausea or dizziness. In order to achieve this, we perform the neural network inference at a rate of 20 hertz on a separate CPU thread, i.e., the network does not update every frame. As the networks are trained on sequences of $\tau = 60$ poses, we keep track of the previous $\tau - 1$ human poses and combine them with the current human pose at time t to obtain the input sequence for our mapping. The mapping produces a sequence of τ quadruped poses, the last of which is used as the virtual quadruped's pose at time t. The inability to *see the future* in this online setting results in the quadruped animations being slightly less smooth. Thus, we apply inertialization[3] to smooth out the results.

6 Results and evaluation

Our human to quadruped motion mapping, described in section 3, can be used in an offline setting to retarget a pre-existing human animation to a virtual quadruped character or an online setting as part of our VR animal embodiment system. Example results from both settings are illustrated in Figures 1 (online), 4 (offline), and 5

[3]https://gdcvault.com/play/1025165/Inertialization-High-Performance-Animation-Transitions

(online). Example videos for both settings are in the supplementary video.

We can see in Figures 1 and 5 that there is a strong synchrony between the human and quadruped poses. This holds for both states of our dynamically switching limb-correspondence between the human and quadruped. When the human is standing and performing locomotion, we see that their legs and the quadruped's front legs are moving synchronously. When the human is sitting, we can see that their arm movements are synchronous with the quadruped's paw raises.

All three figures show that our method produces natural looking poses for the quadruped. In particular, we see in Figure 4 that the poses generated solely using IK are stiff looking and struggle to replicate a realistic looking sitting motion. In contrast, poses generated using our method are more natural looking. For example, we see the subtle details added to the paw animation in our method. Our method also has more realistic timing between the quadruped limbs and includes tail animation.

6.1 Ablation study

In order to evaluate the importance of the different elements of our method, we carried out an ablation study. For this, we compare our method ("Ours") to three alternatives in which we omit certain components or steps of our method. These are:

- No Codebook (NoCB): We do not use FSQ to create a shared codebook as the latent space. We do this by removing the FSQ layers from the encoders E^I and E^Q. Everything else remains the same.
- No initial retarget (NoR): We no longer initially partially retarget the human motion to the quadruped skeleton using R, i.e., we omit step 1 (section 3.1). Instead, we directly input the human motion to the encoder E^I and try to create a shared codebook between the spaces of human and quadruped motions. Everything else remains the same.
- No Codebook and no initial retarget (NoRNoCB): This is a combination of the previous two, i,e, we omit both the shared codebook and the initial retargeting step.

As a baseline, we also compare all of the above to solely animating the quadruped using inverse kinematics.

Qualitatively, we observe that NoR and NoRNoCb clearly perform significantly worse than all other methods, failing to synthesise convincing looking quadruped motion at all. While NoCB does produce coherent motions, they are stiff looking and appear quite similar to the initial partially retargeted motions. Thus suggests that without the shared codebook, the model has struggled to learn a meaningful shared latent space in which latent codes represent semantically similar motions in both the intermediate domain $\mathcal{M}^I$ and the quadruped motion domain $\mathcal{M}^Q$.

6.1.1 Proposed metric. Quantitatively evaluating the quality of synthesised poses and motions is a challenging issue. Typically, we do not have a ground truth label to compare the synthesised data to. Instead, we have a dataset containing ground truth samples from the distribution of real data and we want to measure how likely it is that a generated sample comes from the same distribution. Towards this, we propose to use the following metric for evaluating the quality of generated quadruped poses: We fit a Gaussian Mixture Model (GMM) with $n = 10$ components to the real quadruped

Figure 5: Demonstration of our VR embodiment system, showing a third person view of the scene with the dog avatar and a mirror for inducing embodiment, running in Unreal Engine 5. The user is shown in the bottom left corner. In the locomotion examples, we see the user's legs are mapped to the dog's front paws, while in the sitting example, control of the front paws has dynamically switched to the user's arms.

motion data, i.e., $\mathcal{D}^Q$. Then, given a pose or sequence of poses $\mathbf{m}$, we compute the *Mahalanobis distance* between $\mathbf{m}$ and each of the Gaussians Q_i (for $i = 1, 2, \ldots, n$) in the GMM. The final *score* S for $\mathbf{m}$ is the minimum of these distances. The lower the score the better. Mathematically,

$$S(\mathbf{m}) = min_i\{d_M(\mathbf{m}, Q_i) : i = 1, 2, \ldots, n)\}. \quad (8)$$

where the Mahalanobis distance

$$d_M(\mathbf{x}, Q) = \sqrt{(\mathbf{x} - \mu)^T \Sigma^{-1} (\mathbf{x} - \mu)} \quad (9)$$

measures the distance of a point $\mathbf{x} \in \mathbb{R}^n$ from an n-dimensional probability distribution Q with mean μ and covariance matrix Σ.

We consider two options for applying the metric. In the first, we consider each individual pose $\mathbf{m}_t \in \mathcal{D}^Q$ as a separate data point when fitting the GMM. We then compute the metric for individual poses. While this measures the quality of individual poses, it does not take into account the continuous nature of motion. Thus, in an alternative fitting of the GMM, we split the sequences in $\mathcal{D}^Q$ into windows of five frames and consider each such window as a data point when fitting the GMM. We then compute the metric for sequences of five poses.

6.1.2 Quantitative results. We use the proposed metric to evaluate the different methods of our ablation study. We use an unseen test sequence of approximately 2000 frames of human motion data. The results are shown in Table 1. We evaluate the quality of the motion synthesised by only the joint autoencoders, i.e., the motion synthesised for the front limb kinematic chains of the quadruped (denoted by "Half pose" in the table), and also of the entire method (denoted by "Full pose" in the table). For both, we apply the metric on a pose-wise basis and also on windows of poses (denoted by "window" in the table). For the pose-wise setting, we calculate the score of every frame in the test data and report the average score. For the windows setting, we split the test sequence into windows of five frames, calculate the score for each window, and report the average score across all windows.

Table 1: Quantitative results using proposed metric (↓)

	IK	Ours	NoCB	NoR	NoRNoCB
Half pose	9.67	5.74	**5.58**	11.46	12.47
Half pose window	39.15	**22.12**	25.62	36.3	41.55
Full pose	12.73	**7.98**	9.21	27.59	29.82
Full pose window	58.5	**39.55**	49.19	87.76	110.65

Numbers shown are average scores across all of the test data.

Both Ours and NoCB outperform the IK baseline in all scenarios, with our method outperforming all other methods in every scenario except for the half pose scenario in which NoCB has the lowest score. This suggests our method is successful in synthesising natural looking quadruped poses. Ours outperforms NoCB in three out of four scenarios while NoR outperforms NoRNoCB in all scenarios. This suggests the importance of the shared codebook in learning a meaningful shared latent space.

Both methods (NoR and NoRNoCB) that do not use the initial retargeting step (i.e. step 1 of our method) perform significantly worse than the others (including the baseline IK), suggesting that initially bridging the morphology gap between the human and quadruped skeletons is a significant help when trying to map human motion to quadruped motion using deep-learning.

7 Discussion and limitations

In this work, we present a novel architecture for mapping human motion to quadruped motion, which we incorporate into a VR embodiment system. Importantly, our method preserves a strong synchrony between the human and quadruped motions, while still synthesising realistic and natural looking motion for the virtual quadruped. It also allows for the limb-control of the quadruped to dynamically switch between the user's arms and legs depending on their action, resulting in more intuitive control of the quadruped avatar.

We perform the mapping in several steps, the core technical novelty of which lies in the second step. Here, we leverage finite scalar quantization to construct a finite and discrete shared latent space, or *shared codebook*, between real quadruped motion and initial rough human-to-quadruped retargets that have been generated in step one of the mapping.

We evaluate our work both qualitatively and quantitatively, carrying out an ablation study to examine the importance of the different components of our method. The results of the ablation study illustrate the benefits of using the finite discrete codebook to learn a meaningful shared latent space. Constructing this shared codebook using FSQ is more straight-forward to implement, easier to train, and results in smaller models than using the VQ-VAE [van den Oord et al. 2017]. While previous research [Aberman et al. 2020] has highlighted the difficulties in directly bridging the gap from human motion to quadruped motion using deep-learning, our results show that providing a rough initial retarget of the human motion to the quadruped skeleton can significantly aid in the task. While constructing the initial rule-based retargeter R is an extra overhead, it does not need to be a very high quality retarget and can be implemented straightforwardly using off-the-shelf tools in a game engine.

Our proposed metric, combining GMMs and the Mahalanobis distance, is more straight-forward to implement than other metrics such as the Fréchet Motion Distance (FMD)[Maiorca et al. 2023] which requires the training of a neural network. However, more research is needed into the suitability of it, and other metrics in general, for evaluating motion quality. A user study in which user's ratings are compared to the metric results could be useful here.

Our work has limitations. While our method works well for symmetrical quadrupedal gaits (walk, trot, pace), it is less suited to faster asymmetrical gaits[4] such as a gallop or canter, struggling to produce realistic motion for these gaits. This is because a human jog or run is still symmetrical and so achieving synchrony between the fast symmetrical human gaits and asymmetrical quadruped gaits is a difficult challenge. A method such as the recent work by Li et al. [2024] may be more appropriate in the case of asymmetrical gaits where synchrony is more difficult to achieve. However, generally VR systems are not suited to fast motion such as running from

[4]A gait is symmetrical if the left and right legs of each pair move half a stride out of phase with each other. Otherwise it is asymmetrical.

the user and it also remains to be seen what impact the lack of synchrony would have on the sense of agency. Also, our method does not account for other actions such as jumping. It would be interesting to explore whether different choices for the rule-based retargeter R could solve these problems, for example, exploring different body-part correspondences between the human and the quadruped, or having a more complex retargeter with more states.

Motions synthesised by our method also exhibit artefacts, for example, foot sliding or artefacts while transitioning between standing and sitting. While foot sliding artefacts could be improved using an IK based clean-up, exploring solutions for improving the mapping of trajectories between characters with different morphologies is also an interesting direction for future work. This is especially important for VR animal embodiment applications where a balance needs to be struck between generating realistic animal trajectories, while not impacting negatively on the user's VR experience. Artefacts during transitions are potentially caused by the discontinuity as the rule-based retargeter R switches between states. Some type of blending between states as the switch occurs could alleviate such issues.

Since prior work has shown that a sense of embodiment can be invoked between a human and an animal avatar [Krekhov et al. 2019b] and that a stronger sense of embodiment can be invoked by synthesising more realistic animal motion than is generated by IK mappings [Egan et al. 2023], we did not include a perceptual investigation here. However, it would be interesting to carry out perceptual experiments to explore the sense of VR embodiment experienced by users of our system. In particular, to investigate if our quadruped limb-control dynamically switching between the user's arms and legs is generally considered intuitive and comfortable for users. Future work will explore the impact of dynamically switching user-to-avatar mappings on embodiment experiences in general.

Acknowledgments

This research was funded by Science Foundation Ireland under the ADAPT Centre for Digital Content Technology (Grant No. 13/RC/2106_P2) and RADICal (Grant No. 19/FFP/6409).

References

Kfir Aberman, Peizhuo Li, Dani Lischinski, Olga Sorkine-Hornung, Daniel Cohen-Or, and Baoquan Chen. 2020. Skeleton-aware networks for deep motion retargeting. *ACM Trans. Graph.* 39, 4, Article 62 (Aug 2020), 14 pages. https://doi.org/10.1145/3386569.3392462

Sun Joo (Grace) Ahn, Joshua Bostick, Elise Ogle, Kristine L. Nowak, Kara T. McGillicuddy, and Jeremy N. Bailenson. 2016. Experiencing Nature: Embodying Animals in Immersive Virtual Environments Increases Inclusion of Nature in Self and Involvement with Nature. *Journal of Computer-Mediated Communication* 21, 6 (2016), 399–419. https://doi.org/10.1111/jcc4.12173

Karan Ahuja, Eyal Ofek, Mar Gonzalez-Franco, Christian Holz, and Andrew D. Wilson. 2021. CoolMoves: User Motion Accentuation in Virtual Reality. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 5, 2, Article 52 (June 2021), 23 pages. https://doi.org/10.1145/3463499

Robert McNeill Alexander. 2003. *Principles of animal locomotion.* Princeton University Press.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. 2013. Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation. arXiv:1308.3432 [cs.LG] https://arxiv.org/abs/1308.3432

Ufuk Celikcan, Ilker O. Yaz, and Tolga Capin. 2015. Example-Based Retargeting of Human Motion to Arbitrary Mesh Models. *Comput. Graph. Forum* 34, 1 (February 2015), 216–227. https://doi.org/10.1111/cgf.12507

Yuming Du, Robin Kips, Albert Pumarola, Sebastian Starke, Ali Thabet, and Artsiom Sanakoyeu. 2023. Avatars Grow Legs: Generating Smooth Human Motion from Sparse Tracking Inputs with Diffusion Model. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).* 481–490. https://doi.org/10.1109/CVPR52729.2023.00054

Dónal Egan, Darren Cosker, and Rachel McDonnell. 2023. NeuroDog: Quadruped Embodiment using Neural Networks. *Proc. ACM Comput. Graph. Interact. Tech.* 6, 3, Article 38 (August 2023), 19 pages. https://doi.org/10.1145/3606936

Mie C. S. Egeberg, Stine L. R. Lind, Sule Serubugo, Denisa Skantarova, and Martin Kraus. 2016. Extending the human body in virtual reality: effect of sensory feedback on agency and ownership of virtual wings. In *Proceedings of the 2016 Virtual Reality International Conference* (Laval, France) *(VRIC '16).* Association for Computing Machinery, New York, NY, USA, Article 30, 4 pages. https://doi.org/10.1145/2927929.2927940

Michael Gleicher. 1998. Retargetting motion to new characters. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '98).* Association for Computing Machinery, New York, NY, USA, 33–42. https://doi.org/10.1145/280814.280820

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative Adversarial Nets. In *Advances in Neural Information Processing Systems*, Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger (Eds.), Vol. 27. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf

Rachel Heck and Michael Gleicher. 2007. Parametric motion graphs. In *Proceedings of the 2007 Symposium on Interactive 3D Graphics and Games* (Seattle, Washington) *(I3D '07).* Association for Computing Machinery, New York, NY, USA, 129–136. https://doi.org/10.1145/1230100.1230123

Daniel Holden, Taku Komura, and Jun Saito. 2017. Phase-Functioned Neural Networks for Character Control. *ACM Trans. Graph.* 36, 4, Article 42 (July 2017), 13 pages. https://doi.org/10.1145/3072959.3073663

Daniel Holden, Jun Saito, and Taku Komura. 2016. A Deep Learning Framework for Character Motion Synthesis and Editing. *ACM Trans. Graph.* 35, 4, Article 138 (July 2016), 11 pages. https://doi.org/10.1145/2897824.2925975

Ludovic Hoyet, Ferran Argelaguet, Corentin Nicole, and Anatole Lécuyer. 2016. "Wow! I Have Six Fingers!": Would You Accept Structural Changes of Your Hand in VR? *Frontiers in Robotics and AI* 3 (2016). https://doi.org/10.3389/frobt.2016.00027

Ayumi Ichikawa, Keiichi Ihara, Aoto Tanokashira, and Ikkaku Kawaguchi. 2022. Investigating the Effect of Animal Avatars on Users' Self-disclosure During Interaction in VR space. In *ICAT-EGVE 2022 - International Conference on Artificial Reality and Telexistence and Eurographics Symposium on Virtual Environments - Posters and Demos*, Theophilus Teo and Ryota Kondo (Eds.). The Eurographics Association. https://doi.org/10.2312/egve.20221308

Jiaxi Jiang, Paul Streli, Huajian Qiu, Andreas Fender, Larissa Laich, Patrick Snape, and Christian Holz. 2022. AvatarPoser: Articulated Full-Body Pose Tracking from Sparse Motion Sensing. In *Computer Vision – ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part V* (Tel Aviv, Israel). Springer-Verlag, Berlin, Heidelberg, 443–460. https://doi.org/10.1007/978-3-031-20065-6_26

Yu Jiang, Zhipeng Li, Mufei He, David Lindlbauer, and Yukang Yan. 2023. HandAvatar: Embodying Non-Humanoid Virtual Avatars through Hands. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems* (Hamburg, Germany) *(CHI '23).* Association for Computing Machinery, New York, NY, USA, Article 309, 17 pages. https://doi.org/10.1145/3544548.3581027

Konstantina Kilteni, Raphaela Groten, and Mel Slater. 2012. The Sense of Embodiment in Virtual Reality. *Presence Teleoperators & Virtual Environments* 21 (2012). https://doi.org/10.1162/PRES_a_00124

Sunwoo Kim, Maks Sorokin, Jehee Lee, and Sehoon Ha. 2022. HumanConQuad: Human Motion Control of Quadrupedal Robots using Deep Reinforcement Learning. In *SIGGRAPH Asia 2022 Emerging Technologies (SA '22).* Association for Computing Machinery, New York, NY, USA, Article 5, 2 pages. https://doi.org/10.1145/3550471.3564762

Diederik P. Kingma and Jimmy Ba. 2017. Adam: A Method for Stochastic Optimization. arXiv:1412.6980 [cs.LG]

Elena Kokkinara and Mel Slater. 2014. Measuring the Effects through Time of the Influence of Visuomotor and Visuotactile Synchronous Stimulation on a Virtual Body Ownership Illusion. *Perception* 43, 1 (2014), 43–58. https://doi.org/10.1068/p7545 arXiv:https://doi.org/10.1068/p7545 PMID: 24689131.

Taku Komura and Wai-Chun Lam. 2006. Real-time locomotion control by sensing gloves. *Computer Animation and Virtual Worlds* 17, 5 (2006), 513–525. https://doi.org/10.1002/cav.114 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/cav.114

Andrey Krekhov, Sebastian Cmentowski, Katharina Emmerich, and Jens Krüger. 2019b. Beyond Human: Animals as an Escape from Stereotype Avatars in Virtual Reality Games *(CHI PLAY '19).* Association for Computing Machinery, New York, NY, USA, 439–451. https://doi.org/10.1145/3311350.3347172

Andrey Krekhov, Sebastian Cmentowski, and Jens Krüger. 2019a. The Illusion of Animal Body Ownership and Its Potential for Virtual Reality Games. In *2019 IEEE Conference on Games (CoG).* 1–8. https://doi.org/10.1109/CIG.2019.8848005

Peizhuo Li, Sebastian Starke, Yuting Ye, and Olga Sorkine-Hornung. 2024. WalkTheDog: Cross-Morphology Motion Alignment via Phase Manifolds. In *SIGGRAPH, Technical Papers*. https://doi.org/10.1145/3641519.3657508

Tianyu Li, Jungdam Won, Alexander Clegg, Jeonghwan Kim, Akshara Rai, and Sehoon Ha. 2023. ACE: Adversarial Correspondence Embedding for Cross Morphology Motion Retargeting from Human to Nonhuman Characters. In *SIGGRAPH Asia 2023 Conference Papers* (Sydney, NSW, Australia) *(SA '23)*. Association for Computing Machinery, New York, NY, USA, Article 46, 11 pages. https://doi.org/10.1145/3610548.3618255

Antoine Maiorca, Youngwoo Yoon, and Thierry Dutoit. 2023. Validating Objective Evaluation Metric: Is Fréchet Motion Distance able to Capture Foot Skating Artifacts?. In *Proceedings of the 2023 ACM International Conference on Interactive Media Experiences* (Nantes, France) *(IMX '23)*. Association for Computing Machinery, New York, NY, USA, 242–247. https://doi.org/10.1145/3573381.3596460

Ian Mason, Sebastian Starke, and Taku Komura. 2022. Real-Time Style Modelling of Human Locomotion via Feature-Wise Transformations and Local Motion Phases. *Proc. ACM Comput. Graph. Interact. Tech.* 5, 1, Article 6 (May 2022), 18 pages. https://doi.org/10.1145/3522618

Fabian Mentzer, David Minnen, Eirikur Agustsson, and Michael Tschannen. 2024. Finite Scalar Quantization: VQ-VAE Made Simple. In *The Twelfth International Conference on Learning Representations*. https://openreview.net/forum?id=8ishA3LxN8

Akimi Oyanagi and Ren Ohmura. 2019. Transformation to a Bird: Overcoming the Height of Fear by Inducing the Proteus Effect of the Bird Avatar. In *Proceedings of the 2nd International Conference on Image and Graphics Processing*. 145–149. https://doi.org/10.1145/3313950.3313976

Daniel Pimentel and Sri Kalyanaraman. 2022. The effects of embodying wildlife in virtual reality on conservation behaviors. *Scientific Reports* 12, 1 (2022). https://doi.org/10.1038/s41598-022-10268-y

Jose Luis Ponton, Haoran Yun, Carlos Andujar, and Nuria Pelechano. 2022. Combining Motion Matching and Orientation Prediction to Animate Avatars for Consumer-Grade VR Devices. *Computer Graphics Forum* (2022). https://doi.org/10.1111/cgf.14628

Jose Luis Ponton, Haoran Yun, Andreas Aristidou, Carlos Andujar, and Nuria Pelechano. 2023. SparsePoser: Real-time Full-body Motion Reconstruction from Sparse Data. *ACM Trans. Graph.* 43, 1, Article 5 (Oct 2023), 14 pages. https://doi.org/10.1145/3625264

Daniele Reda, Jungdam Won, Yuting Ye, Michiel van de Panne, and Alexander Winkler. 2023. Physics-Based Motion Retargeting from Sparse Inputs. *Proc. ACM Comput. Graph. Interact. Tech.* 6, 3, Article 33 (August 2023), 19 pages. https://doi.org/10.1145/3606928

Helge Rhodin, James Tompkin, Kwang In Kim, Edilson de Aguiar, Hanspeter Pfister, Hans-Peter Seidel, and Christian Theobalt. 2015. Generalizing Wave Gestures from Sparse Examples for Real-Time Character Control. *ACM Trans. Graph.* 34, 6, Article 181 (Oct 2015), 12 pages. https://doi.org/10.1145/2816795.2818082

Yeongho Seol, Carol O'Sullivan, and Jehee Lee. 2013. Creature Features: Online Motion Puppetry for Non-Human Characters. In *Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Anaheim, California) *(SCA '13)*. Association for Computing Machinery, New York, NY, USA, 213–221. https://doi.org/10.1145/2485895.2485903

Aaron Shon, Keith Grochow, Aaron Hertzmann, and Rajesh PN Rao. 2005. Learning Shared Latent Structure for Image Synthesis and Robotic Imitation. In *Advances in Neural Information Processing Systems*, Vol. 18. MIT Press. https://proceedings.neurips.cc/paper/2005/file/030e65da2b1c944090548d36b244b28d-Paper.pdf

Alexandra Sierra Rativa, Marie Postma, and Menno van Zaanen. 2020. Can virtual reality act as an affective machine? The wild animal embodiment experience and the importance of appearance. In *MIT LINC 2019*, Vol. 3. 214–223. https://jwel.mit.edu/linc-2019

Sebastian Starke, Ian Mason, and Taku Komura. 2022. DeepPhase: Periodic Autoencoders for Learning Motion Phase Manifolds. *ACM Trans. Graph.* 41, 4, Article 136 (July 2022), 13 pages. https://doi.org/10.1145/3528223.3530178

Sebastian Starke, Yiwei Zhao, Taku Komura, and Kazi Zaman. 2020. Local Motion Phases for Learning Multi-Contact Character Movements. *ACM Trans. Graph.* 39, 4, Article 54 (July 2020), 14 pages. https://doi.org/10.1145/3386569.3392450

William Steptoe, Anthony Steed, and Mel Slater. 2013. Human Tails: Ownership and Control of Extended Humanoid Avatars. *IEEE Transactions on Visualization and Computer Graphics* 19, 4 (April 2013), 583–590. https://doi.org/10.1109/TVCG.2013.32

Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. 2017. Neural discrete representation learning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems* (Long Beach, California, USA) *(NIPS'17)*. Curran Associates Inc., Red Hook, NY, USA, 6309–6318.

Ruben Villegas, Jimei Yang, Duygu Ceylan, and Honglak Lee. 2018. Neural Kinematic Networks for Unsupervised Motion Retargetting. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 8639–8648. https://doi.org/10.1109/CVPR.2018.00901

Katsu Yamane, Yuka Ariki, and Jessica Hodgins. 2010. Animating Non-Humanoid Characters with Human Motion Data. In *Eurographics/ ACM SIGGRAPH Symposium on Computer Animation*. The Eurographics Association. https://doi.org/10.2312/SCA/SCA10/169-178

He Zhang, Sebastian Starke, Taku Komura, and Jun Saito. 2018. Mode-Adaptive Neural Networks for Quadruped Motion Control. *ACM Trans. Graph.* 37, 4, Article 145 (Jul 2018), 11 pages. https://doi.org/10.1145/3197517.3201366

Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. 2019. On the Continuity of Rotation Representations in Neural Networks. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 5738–5746. https://doi.org/10.1109/CVPR.2019.00589