How to train your dog: Neural enhancement of quadruped animations

Donal Egan
Trinity College Dublin
Ireland

George Fletcher
Yiguo Qiao
Darren Cosker
University of Bath
United Kingdom

Rachel McDonnell
Trinity College Dublin
Ireland

Figure 1: Blue: frames from initial animation lacking the subtleties of true canine motion and containing small errors. Green: corresponding frames from ground truth canine motion capture dataset. Red: Output after passing the initial (blue) animation through our quadruped animation enhancement neural network.

ABSTRACT

Creating realistic quadruped animations is challenging. Producing realistic animations using methods such as key-framing is time consuming and requires much artistic expertise. Alternatively, motion capture methods have their own challenges (getting the animal into a studio, attaching motion capture markers, and getting the animal to put on the desired performance) and the resulting animation will still most likely require cleaning up. It would be useful if an animator could provide an initial rough animation and in return be given a corresponding high quality realistic one. To this end, we present a deep-learning approach for the automatic enhancement of quadruped animations. Given an initial animation, possibly lacking the subtle details of true quadruped motion and/or containing small errors, our results show that it is possible for a neural network to

learn how to add these subtleties and correct errors to produce an enhanced animation while preserving the semantics and context of the initial animation. Our work also has potential uses in other applications, for example, its ability to be used in real-time means it could form part of a quadruped embodiment system.

CCS CONCEPTS

• **Computing methodologies** → **Motion processing**; **Neural networks**.

KEYWORDS

Quadruped Animation, Motion Capture, Deep-Learning

ACM Reference Format:

Donal Egan, George Fletcher, Yiguo Qiao, Darren Cosker, and Rachel McDonnell. 2021. How to train your dog: Neural enhancement of quadruped animations. In *Motion, Interaction and Games (MIG '21)*, November 10–12, 2021, Virtual Event, Switzerland. ACM, New York, NY, USA, 7 pages. <https://doi.org/10.1145/3487983.3488293>

This work is licensed under a [Creative Commons Attribution International 4.0 License](https://creativecommons.org/licenses/by/4.0/).

MIG '21, November 10–12, 2021, Virtual Event, Switzerland
© 2021 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-9131-3/21/11.
<https://doi.org/10.1145/3487983.3488293>

1 INTRODUCTION

Modern movies and video games are increasingly demanding higher quality animations, at higher volumes and on shorter time scales.

Animals are no exception, and many movies and video games show-case animals in key roles. In modern pipelines, animals – like other characters – are animated either using key-framing, motion capture, or a combination of both methods. In the former case (key-framing) we rely on expensive and scarce artistic talent to create realistic movements from reference motions. In the latter case (motion-capture using human actors) we still typically rely on hand cleaning of animations to achieve high quality animal-like motions.

In this paper we explore the automatic enhancement and improvement of quadruped animations. To this end, we present a deep-learning approach trained from a large dataset of quadruped motion capture – in our case canines. A feed-forward neural network takes an existing quadruped animation, on a per-pose basis, and adds corrections or subtle modifications such that it matches more closely to the exemplar motion for that quadruped (the motion capture training basis). Using this approach, we show that input animations are improved in terms of their accuracy and faithfulness to the quadruped’s expected movement without changing the semantics and context of the input motion (i.e. its timing and gait).

For training our deep-learning approach we first need to create data pairs. We do this using a simple quadruped inverse kinematics (IK) solver to create corresponding initial animations for each sequence in the ground truth motion capture dataset. The IK solver takes as constraints the positions/orientations from the ground truth data of a small subset of the skeletal joints and solves for the orientations of the remaining joints. The resulting animations contain errors and often lack the subtleties of true quadruped motion. The feed-forward neural network learns the difference between these rough IK generated animations and the ground truth motion sequences. We also show that our system has the potential to be adapted to alternative quadruped motion capture datasets and models.

While the focus of this paper is on the improvement of existing animations, the ideas explored here have potential uses in many other applications. Our per-pose approach to the enhancement means that our system could be used in real-time applications, for example, as part of a quadruped embodiment system for use in the acting or gaming industries. Our work could also potentially be used for marker reduction when capturing quadruped motion capture data, making it easier to obtain.

2 RELATED WORK

Motion capture is a data-driven approach used for capturing real human and animal motion, with applications in entertainment (film and games), sport and medicine. While human motion capture data is relatively easy to obtain [Ghorbani et al. 2021; Mahmood et al. 2019; Smith 2017], motion capture for animals is more challenging – from getting the animals into a motion capture studio and attaching markers to them, to getting the animals to perform the desired motions. Nevertheless, quadruped motion capture datasets do exist. For practicality reasons these are usually for canines. Kearney et al. [2020] introduce a dataset containing motion capture for five dogs. The starting point for our work is a large canine motion capture dataset introduced by Zhang et al. [2018].

Several existing works use **deep-learning for predicting and generating human motion**. Fragkiadaki et al. [2015] and Ghorbani et al. [2020] use recurrent neural network (RNN) architectures for the task, while Holden et al. [2017] introduce *phase-functioned neural networks* – a novel neural network architecture allowing for the real-time control of human motion. Pavllo et al. [2020] introduce the idea of quaternion-based neural networks for human motion prediction with *QuaterNet*. In this paper we use a quaternion-based network to model quadruped motion rather than human motion. While less common, deep-learning has also been used for **predicting and generating quadruped motion**. Zhang et al. [2018] propose *mode-adaptive neural networks* – a network architecture for controlling and generating realistic quadruped motion in real-time. Luo et al. [2020] use deep reinforcement learning to develop a physics-based controller for quadruped locomotion. Also related to our work are previous works using deep neural networks to **correct corrupted animation data**. In [Holden 2018] a feed forward neural network is trained to predict skeletal poses in the form of joint rotations from possibly corrupted motion capture marker data. Our approach is similar in that we also produce the network inputs for training from the ground truth data and then task the network with reconstructing this original ground truth. However, instead of artificial noise/occlusion mimicking real motion capture noise/occlusion as in [Holden 2018], we generate rough IK poses using ground truth IK constraints to mimic IK poses which could be produced by other processes (for example, key-framing or partial motion capture solves). Holden et al. [2016, 2015] show that a convolutional autoencoder can be used for correcting corrupted animation sequences.

Also relevant are existing works on **predicting motion from sparse motion capture input signals**. While we focus on recovering full-body quadruped motion, several existing works approach the problem for human motion. Similar to us, these approaches are often data driven. Chai and Hodgins [2005] use the positions of a small set of markers placed on the actor’s body as control signals for constructing full-body animation. At run-time, given the control signals for the actor’s current pose, a nearest-neighbour search is used to find the K closest example poses in a large motion capture database. These examples are then used to learn a simple local linear model which can then be used to construct the full body pose from the low dimensional control signals. Liu et al. [2006] present an algorithm for finding an optimal subset of motion capture markers from which it is still possible to generate plausible full-body human motion, while Dagnes et al. [2019] present an automatic technique, using k-nearest neighbours classification, for computing minimum optimized marker layouts for facial motion capture. Dissimilar to us, these approaches do not rely on deep-learning techniques. However, deep-learning approaches have been used for recovering full-body motion from sparse inputs. For example, Huang et al. [2018] train a RNN to recover full human body pose using the signals from just 6 inertial measurement units (IMUs) as input, while Yang et al. [2021] train a deep neural network to predict lower body pose when only sparse tracking data for the upper body is available, for example, with commercial virtual reality systems.

Both **Forward Kinematics (FK)** and **Inverse Kinematics (IK)** are relevant to our work. Human and animal bodies can be viewed as hierarchical structures of kinematic chains composed of joints

Figure 2: The skeleton used for the dog model. It is composed of 27 joints, including a root and six end-effectors. The red dots highlight the reduced marker configurations used in the IK generated animations.

and bones. If the joint rotations are known, FK is the integration of these along the kinematic chains to calculate the joint and end-effector positions. Conversely, if the end-effector or joint positions are known, inverse kinematics can be used to calculate the corresponding joint rotations. IK is commonly used in traditional motion retargeting to enforce constraints [Bindiganavale and Badler 1998; Lee and Shin 1999; Monzani et al. 2000]. Many researchers [Aberman et al. 2020; Pavlo et al. 2020; Villegas et al. 2018] use FK layers within their neural network architectures – something which we adopt in this paper.

3 METHOD

We present a system for the automatic enhancement or improvement of quadruped animations. We use a deep-learning approach to create a network which, working on a per-pose basis, takes as input an existing quadruped animation and outputs a tidied-up or corrected animation more closely matching what would be the expected true motion of the quadruped. Our starting point is a large set of quadruped motion capture data for a single canine. To create the data pairs for our learning, we use inverse kinematics to create corresponding *rough* animations for each ground truth motion sequence in the motion capture dataset. In subsection 3.1 we describe the preparation of the data and in subsection 3.2 we provide details of the network and its training. An outline of our pipeline is shown in figure 3.

3.1 Data Preparation

We use the dataset introduced in Zhang et al. [2018]. The data consists of unstructured motion capture data for a single dog. It contains various modes of locomotion (walk, pace, trot, canter) as well as other motions such as sitting, standing, idling, lying, and jumping. A skeleton fitted to the motion capture data contains 27 joints, including a root and 6 end-effectors (Figure 2).

For each ground truth motion sequence in the data, we use inverse kinematics (IK) to create corresponding *rough* animations. To do this, an IK solver for the dog skeleton was set up. This solver takes the positions and/or orientations for a small subset of skeletal joints from the ground truth motion sequences as constraints and solves for the orientations (and hence positions) of the remaining joints.

The IK solver works in local space, i.e. the position and orientation of the root is fixed to be stationary and all other joint positions/orientations are calculated relative to this. Seven other joints are also used for constraints (figure 2): the position and global

orientation of the top of the spine, the positions of the nose and tail end, and four constraints for the limbs. Two possibilities are considered for the four limb constraints: the positions of the four feet (i.e. the four limb end-effectors) or the positions of the four ankles (i.e. the second last joints in the limbs). The different possibilities cause different (in)accuracies in the *rough* animations in comparison to the ground truth motion capture data. For example, using the foot positions as constraints promotes their correct placement in the IK generated animations but causes inaccuracies further back along the kinematic chains, while using the ankle positions leads to a more accurate solve further back along the kinematic chains but also means that foot placements/orientations will most likely be incorrect in the IK generated animations.

The ground truth data contains 147541 frames of motion or poses. Applying the IK solver to each of these, using both choices of joints for constraints, gives 295082 (147541×2) pairs of poses, with each pair consisting of a *rough* IK solved pose, $\mathbf{x}$, together with a corresponding ground truth pose, $\mathbf{y}$. We train a simple feed-forward neural network to learn the differences between these IK generated poses and the ground truth poses. Thus, once trained, the network is able to take a pre-existing animation, for example, a rough animation created using inverse kinematics, and augment it so as to more closely match what would be the dog’s corresponding true motion as defined by the ground truth motion capture data. This network is described in the following sub section (3.2).

3.2 Network Architecture

A skeletal pose can be represented by the 3D positions of the joints [Holden et al. 2016, 2015] or by the 3D rotations of the joints relative to their parents [Aberman et al. 2020; Fragkiadaki et al. 2015; Ghorbani et al. 2020; Pavlo et al. 2020]. In the latter case, the joint positions can be retrieved from the rotations using forward kinematics. When predicting joint positions, the skeletal constraints (i.e. bone lengths) may not be respected by the network. Thus, these constraints must be explicitly accounted for [Holden et al. 2016]. The same issue does not arise when predicting joint rotations. For this reason, we use the joint rotation representation for poses. The rotations can be parameterized using Euler angles, axis-angle representation, rotation matrices or unit quaternions. Pavlo et al. [2020] provides a detailed discussion of the strengths and weaknesses of each. We use the unit quaternion representation.

Thus the input to our network is a vector $\mathbf{x} \in \mathbb{R}^{27 \times 4}$ representing a pose to be enhanced by the network. The output is a vector $f_{\mathbf{w}}(\mathbf{x}) \in \mathbb{R}^{27 \times 4}$ representing the enhanced pose. The function $f_{\mathbf{w}}$ is a simple feed-forward neural network parameterized by weights $\mathbf{w}$. The network $f_{\mathbf{w}}$ consists of 5 fully-connected layers. The first 4 of these each have 1024 units and are each followed by a ReLU activation layer. The final output fully connected layer has $108 = 27 \times 4$ units. To ensure that the network outputs valid rotations, i.e. unit quaternions, a final normalisation layer is added at the end of the network (similar to [Ghorbani et al. 2020; Pavlo et al. 2020]).

The network is trained, i.e. values for the parameters $\mathbf{w}$ are determined, by minimising the following loss function

$$\mathcal{L}_{total}(f_{\mathbf{w}}(\mathbf{x}), \mathbf{y}) = \alpha \mathcal{L}_{pos}(f_{\mathbf{w}}(\mathbf{x}), \mathbf{y}) + \beta \mathcal{L}_{rot}(f_{\mathbf{w}}(\mathbf{x}), \mathbf{y}) + \gamma \mathcal{L}_{unit}(f_{\mathbf{w}}(\mathbf{x})). \quad (1)$$

Figure 3: Given ground truth skeletal poses y , a subset of the skeletal joints and inverse kinematics are used to produce corresponding rough poses x . These pairs are used to train a feed-forward neural network to automatically enhance an input pose.

where y is the corresponding ground truth pose for input pose x . The term $\mathcal{L}_{unit}$ is a penalty $(w^2 + x^2 + y^2 + z^2 - 1)^2$ applied to each output quaternion $q = w + xi + yj + zk$ prior to the normalisation layer. It encourages the network to produce unit quaternions so that the normalisation layer is only needed for small corrections [Pavlo et al. 2020]. We next explain the two reconstruction loss terms $\mathcal{L}_{pos}$ and $\mathcal{L}_{rot}$ in more detail.

3.2.1 Predicted joint position error. The function $\mathcal{L}_{pos}$ is a reconstruction loss over joint positions defined by

$$\mathcal{L}_{pos}(x, y) = \frac{1}{27} \sum_{i=1}^{27} ||FK(x)_i - FK(y)_i|| \quad (2)$$

where $FK(x)_i$ and $FK(y)_i$ are the 3D joint positions for joint i , obtained via forward kinematics, for two poses x and y , respectively. The forward kinematics operator is differentiable with respect to joint rotations, making $\mathcal{L}_{pos}$ a valid loss for training the network [Aberman et al. 2020; Pavlo et al. 2020].

3.2.2 Predicted joint rotation error. The geodesic distance between two rotations (unit quaternions) is given by:

$$d_1(q_1, q_2) = 2\arccos(|q_1 \cdot q_2|) \quad (3)$$

It can be interpreted as the amount of rotation required to align q_1 with q_2 . Thus, if q_2 represents the target ground truth rotation, then $d_1(q_1, q_2)$ can be interpreted as an error measure on q_1 . Rather than using d_1 , we can use the computationally less expensive approximation [Huyh 2009]

$$d_2(q_1, q_2) = 1 - |q_1 \cdot q_2|. \quad (4)$$

Simply applying this loss (equation 4) to each joint and summing to get the total error for a pose gives equal weight to each joint in

the hierarchical skeletal structure. No allowance is made for the fact that errors propagate down the kinematic chains and so an error in a joint’s rotation has a more significant impact on the final pose than an error of the same magnitude in any of its descendants. To account for this, we use the hierarchical joint angle loss proposed by [Ghorbani et al. 2020] to measure the difference between two poses x and y :

$$\mathcal{L}_{rot}(x, y) = \sum_{i=1}^{27} \omega_i d_2(q_i, p_i) \quad (5)$$

where q_i and p_i are the rotations for joint i in poses x and y , respectively. The weight ω_i is defined to be the length of the longest kinematic chain from the joint i to an end-effector:

$$\omega_i = \max_j (\omega_j + l_{i-j}) \text{ s.t. } j \text{ is a child of } i \quad (6)$$

where l_{i-j} is the length of the bone connecting joints i and j .

We experimented with using only one of $\mathcal{L}_{pos}$ and $\mathcal{L}_{rot}$. However, we observed better results when using both. For α , β , and γ we use values of 1.75, 1.0, and 0.01, respectively.

3.3 Implementation

Three entire motion sequences, containing a range of motions and totalling 5448 frames (and hence 10896 data pairs due to the doubling of the data using the IK solver, section 3.1), were removed from the data and kept as test material. The remaining data (284186 pairs of poses) was randomly split into training (85%), validation (10%), and test (5%) subsets. While the data in the test subset allows the performance of the network on previously unseen poses to be assessed, the network may have seen very similar poses (for example, the immediately preceding or following poses from the motion sequence) in the training subset. Thus, removing three entire motion sequences from the data allows the performance of the network on entirely unseen motion sequences to be assessed.

The IK solver for creating the input animations was implemented using the Klamp Python API¹ which performs Newton-Raphson root solving with line search (never diverges). PyTorch² was used for the deep-learning implementation. The network was trained using the Adam optimizer [Kingma and Ba 2017] with a learning rate of 0.0001 and a mini-batch size of 256. Training was continued until the error $\mathcal{L}_{total}$ (equation 1) for the validation set converged (approximately 250 epochs). Training was done using a Nvidia GeForce RTX™ 3090 GPU and took approximately five hours.

4 EVALUATION

We use four unseen test motion sequences for our evaluation. Sequence A (856 frames) consists of idling and walking in a straight line. Sequence B (800 frames) consists of fast locomotion with turns. Sequence C (250 frames) consists of sitting down. Sequence D (200 frames) consists of locomotion with jumping. For each test sequence, we compare both a *rough* initial animation generated using the IK solver as described in subsection 3.1 (poses $(x_1, x_2, \dots, x_N)$), and the corresponding enhanced animation produced by the network (poses $(f_w(x_1), f_w(x_2), \dots, f_w(x_N))$), to the ground truth motion capture sequence (poses $(y_1, y_2, \dots, y_N)$).

¹<http://motion.cs.illinois.edu/software/klamp/latest/pyklamp/docs/>

²<https://pytorch.org/>

Figure 4: Sample poses from the test sequences. The blue poses were generated using inverse kinematics. The red poses are the blue poses following automatic enhancement by the network. The green poses are the ground truth poses from the motion capture dataset. For each test sequence we can see that the network has made subtle enhancements and corrections to the input pose.

4.0.1 Qualitative evaluation. Figure 4 shows a single frame from each test sequence. For each, we show the IK generated pose (blue), this IK generated pose following automatic enhancement by our network (red), and the ground truth pose from the motion capture data (green). It is clear that semantics and context (for example, the current phase of the gait cycle) are preserved by the enhancement process. In each, we see that poses generated using our simple IK solver fail to replicate certain subtleties of the dog motion, for example, the correct orientation of the paws at different phases of the gait cycle (test materials A, B, and D). These subtle enhancements are added to the motions when passed through the network. The network is also able to correct unnatural joint rotations, for example, the unnaturally rotated outwards front right leg in test material C. In some cases the network does make possibly unwanted changes, for example, in figure 4c where the orientation of the head has been altered. The entire test sequences can be seen in the supplemental movie.

4.0.2 Quantitative evaluation. In Table 1 we examine joint rotation errors in both the IK generated animations and their network enhanced versions in comparison to the ground truth motion capture sequences. For the analysis, we split the skeletal joints into five subgroups as illustrated in the table. Joint rotation errors are measured using the geodesic distance approximation from equation 4. For each test sequence and joint grouping, we report both

the mean error per-joint per-frame and also the variance of these errors. In almost all cases (only excluding the tail grouping for two test sequences), both the mean error and variance are significantly reduced. Thus, both the magnitude and spread of joint rotation errors in an initial animation are significantly reduced following passage through the enhancement network, suggesting the enhanced animations more closely match the expected true dog motion as defined by the ground truth motion capture data.

We also looked at the 3D joint position errors for the four test sequences. Following passage of the initial IK animations through the enhancement network, the mean positional error per-joint per-frame decreased for test material A (1.06 to 0.99), but increased for test materials B (1.68 to 1.70), C (1.37 to 2.2), and D (1.33 to 2.26). However, we suggest that comparing 3D joint position errors between the initial IK generated animations and the enhanced animations is not necessarily a fair comparison. The constraints used by the IK solver to make the data pairs are a subset of the ground truth joint positions. Thus, assuming the solver succeeds in satisfying these constraints, joints in this subset will have (close to) zero positional error when compared to the ground truth data. It follows that any movement of these joints by the enhancement network will lead to a significant proportional increase in positional error. The joint rotation errors, together with our qualitative results, suggest that despite these positional error values, our network enhanced animations are improvements of the initial IK animations. Interestingly, the two motion types for which the positional error significantly increased, i.e. sit and jump, make up a significantly smaller proportion of the initial motion capture dataset than locomotion. Sit and jump are approximately 11% and 0.7% of the initial dataset, respectively [Zhang et al. 2018]. This could be a contributing factor to the increase in positional error values.

5 ALTERNATIVE DATASET AND CANINE MODEL

We also tested our method using a dog from the dataset released in [Kearney et al. 2020]. The dataset contains motion capture data for five different dogs. However, the amount of data and range of motions per dog is significantly smaller than found in [Zhang et al. 2018]. The skeleton model fitted to the motion capture data is more complex than that used in [Zhang et al. 2018] (43 joints vs 27 joints) (figures 5 and 2).

The input IK generated animations we created using the ground truth positions and/or orientations from the motion capture data for a small subset of joints (figure 5) similar to the method described in 3.1. The network architecture is unchanged from that described in 3.2 except for the size of the input and output layers. Training takes approximately 15 minutes using the same GPU.

5.1 Results

There are 2638 frames of data for our chosen dog composed of 6 motion sequences - walk×2, trot, jump, jump onto/from a table, and maneuvering over poles. Due to the small amount of data we trained the network twice, leaving out a different test sequence each time. First, we left out a walk sequence (602 frames). Second, we left out the jump sequence (148 frames). Some qualitative results are shown in figure 1. The three leftmost frames are from the walk

Table 1: Rotation errors: Comparison of both IK generated and network enhanced animations with ground truth motion capture sequences by joint groupings. For each test material and joint grouping, we look at both the mean (radians) and variance (radians²) of the geodesic distance joint errors per-joint per-frame. The final column shows the percentage reduction in the mean errors and variance following enhancement by the network.

	IK vs GT mean / var	Enhanced vs GT mean / var	% decrease mean / var
Sequence A			
Spine	0.2386 / 0.0609	0.0001 / 0.0000	100% / 100%
Rear legs	0.0131 / 0.0002	0.0008 / 0.0000	94% / 100%
Front legs	0.0440 / 0.0034	0.0016 / 0.0000	96% / 99%
Head	0.0844 / 0.0030	0.0026 / 0.0000	97% / 100%
Tail	0.0690 / 0.0032	0.0524 / 0.0043	24% / -35%
Sequence B			
Spine	0.2095 / 0.0492	0.0002 / 0.0000	100% / 100%
Rear legs	0.0310 / 0.0016	0.0020 / 0.0000	93% / 99%
Front legs	0.0534 / 0.0062	0.0062 / 0.0003	88% / 96%
Head	0.2100 / 0.0146	0.0039 / 0.0000	98% / 100%
Tail	0.3969 / 0.0748	0.0684 / 0.0216	83% / 71%
Sequence C			
Spine	0.2036 / 0.0403	0.0003 / 0.0000	100% / 100%
Rear legs	0.0434 / 0.0016	0.0150 / 0.0007	65% / 56%
Front legs	0.0398 / 0.0041	0.0035 / 0.0001	91% / 98%
Head	0.2963 / 0.0223	0.0045 / 0.0000	98% / 100%
Tail	0.2885 / 0.0694	0.2801 / 0.0942	3% / -36%
Sequence D			
Spine	0.2112 / 0.0553	0.0003 / 0.0000	100% / 100%
Rear legs	0.0184 / 0.0003	0.0041 / 0.0000	77% / 91%
Front legs	0.0659 / 0.0125	0.0139 / 0.0013	79% / 90%
Head	0.3359 / 0.0579	0.0040 / 0.0000	99% / 100%
Tail	0.3696 / 0.0687	0.1015 / 0.0276	73% / 60%

Figure 5: Skeleton used for canine model in dataset from [Kearney et al. 2020]. The red dots highlight the reduced marker configuration used for the IK generated animations.

test sequence, while the four rightmost frames are from the jump test sequence. In all cases it can be seen that subtle corrections have been made by the network. In the jump sequence we can also see in places where the network failed to correct the initial animation. However, it is worth noting that there were no other jump sequences in the training data. The entire test sequences are visible in the supplemental movie. Following passage of the initial IK generated animations through the networks, the mean

(radians)/variance (radians²) of the rotation errors per-joint per-frame decreased from 0.0158/0.0006 to 0.0002/0.0000 on the walk test sequence and from 0.0246/0.0021 to 0.0050/0.0002 on the jump test sequence. While these networks are less generalizable than that trained on the data from [Zhang et al. 2018] due to the smaller initial dataset and range of motions in it, these qualitative and quantitative results suggest that our method can also extend to other datasets and canine models. It also allowed us to view some results on a skinned dog model.

6 DISCUSSION

We present a deep-learning approach for the automatic enhancement of quadruped animations. Using a simple IK solver, taking constraints from a large canine motion capture dataset for only a small subset of skeletal joints, we create canine animations covering a range of motions but lacking the subtle details of true canine motion and containing errors. Using these IK generated animations along with the motion capture dataset as the ground truth, we train a feed-forward neural network, on a per-pose basis, to learn the difference between the IK generated animations and the corresponding ground truth motions. Our results show that it is possible for a deep neural network to learn how to improve existing quadruped animations by making subtle enhancements and small corrections so as to more closely match what would be the expected true quadruped motion, all the while preserving the semantics and context of the initial animation. We also showed that our method can extend to other datasets and canine models.

While our simple feed-forward neural network architecture produces plausible animations, it is just a starting point and has limitations. Future work includes experimenting with more complex or novel network architectures designed to explicitly account for the special characteristics of quadruped motion, for example, the hierarchical skeletal structure or the phases of the different gait cycles. Our system works in the character’s local coordinate system, i.e. it makes no allowance for the root position and orientation. It follows that an accurate route trajectory must be supplied with the input animation. Future work also includes adding the ability to correct the route trajectory to our animation enhancement system.

Our system could be used as a basis for various applications, for example, refining animation sequences. Typically, animation production goes through a number of creation stages – rough blocking of the animation, followed by a refining pass, followed by a final polish where the clean-up is done. We believe that our system could help an animator go from the blocking stage to the refining stage automatically, thus speeding up the animation process, while regaining control for the final finesse, which is something that we will investigate in future work.

Another potential application of our system could be reducing the number of markers needed to be placed on a canine during motion capture. Markers can be difficult to apply and uncomfortable for the animal. For this application to work, the dog would have to have similar properties to the training data (same breed etc.), or the studio could create a training set of ‘high quality’ animations on the first day of production with a regular marker-set, and then reduce on subsequent capture sessions. Development of a system that generalises to all dogs would be also be useful for this application.

Our enhancement system works on a per-pose basis. This could potentially lead to jittering in the output animations. However, any such jitter could be removed using a filter to remove high-frequency noise [Holden 2018]. Working on a per-pose basis can also be advantageous. It allows for the use of such a system in real-time applications, for example, our system could form part of a quadruped embodiment system for humans in VR. By extracting constraints for our quadruped IK solver from a human actor attempting to behave as a quadruped, using the IK solver to produce an initial pose using these constraints, and using our enhancement network to produce a final realistic quadruped pose, a real-time quadruped embodiment system could be developed. While current animal embodiment systems are limited to IK reconstructions ([Krekhov et al. 2019]), incorporating our animation enhancement method could potentially allow for improved experiences of embodiment. This is something which we plan to investigate in future work.

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 Transactions on Graphics (TOG)* 39, 4 (2020), 62.
- Rama Bindiganavale and Norman I Badler. 1998. Motion abstraction and mapping with spatial constraints. In *International Workshop on Capture Techniques for Virtual Environments*. Springer, 70–82.
- Jinxiang Chai and Jessica K. Hodgins. 2005. Performance Animation from Low-Dimensional Control Signals. *ACM Trans. Graph.* 24, 3 (July 2005), 686–696. <https://doi.org/10.1145/1073204.1073248>
- Nicole Dagnes, Federica Marcolin, Enrico Vezzetti, François-Régis Sarhan, Stéphanie Dakpé, Frédéric Marin, Francesca Nonis, and Khalil Ben Mansour. 2019. Optimal marker set assessment for motion capture of 3D mimic facial movements. *Journal of Biomechanics* 93 (2019), 86–93. <https://doi.org/10.1016/j.jbiomech.2019.06.012>
- Katerina Fragkiadaki, Sergey Levine, Panna Felsen, and Jitendra Malik. 2015. Recurrent Network Models for Human Dynamics. In *2015 IEEE International Conference on Computer Vision (ICCV)*. 4346–4354. <https://doi.org/10.1109/ICCV.2015.494>
- Saeed Ghorbani, Kimia Mahdavian, Anne Thaler, Konrad Kording, Douglas James Cook, Gunnar Blohm, and Nikolaus F. Troje. 2021. MoVi: A large multi-purpose human motion and video dataset. *PLOS ONE* 16, 6 (June 2021), e0253157. <https://doi.org/10.1371/journal.pone.0253157> Publisher: Public Library of Science.
- Saeed Ghorbani, Calden Wloka, Ali Etemad, Marcus A. Brubaker, and Niko Troje. 2020. Probabilistic Character Motion Synthesis Using a Hierarchical Deep Latent Variable Model. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation (Virtual Event, Canada) (SCA '20)*. Eurographics Association, Goslar, DEU, Article 21, 15 pages. <https://doi.org/10.1111/cgf.14116>
- Daniel Holden. 2018. Robust Solving of Optical Motion Capture Data by Denoising. *ACM Trans. Graph.* 37, 4, Article 165 (July 2018), 12 pages. <https://doi.org/10.1145/3197517.3201302>
- 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 (2016), Article 138. <https://doi.org/10.1145/2897824.2925975>
- Daniel Holden, Jun Saito, Taku Komura, and Thomas Joyce. 2015. Learning Motion Manifolds with Convolutional Autoencoders. In *SIGGRAPH Asia 2015 Technical Briefs (Kobe, Japan) (SA '15)*. Association for Computing Machinery, New York, NY, USA, Article 18, 4 pages. <https://doi.org/10.1145/2820903.2820918>
- Yinghao Huang, Manuel Kaufmann, Emre Aksan, Michael J. Black, Otmar Hilliges, and Gerard Pons-Moll. 2018. Deep Inertial Poser: Learning to Reconstruct Human Pose from Sparse Inertial Measurements in Real Time. *ACM Trans. Graph.* 37, 6, Article 185 (Dec. 2018), 15 pages. <https://doi.org/10.1145/3272127.3275108>
- Du Q. Huynh. 2009. Metrics for 3D Rotations: Comparison and Analysis. *Journal of Mathematical Imaging and Vision* 35, 2 (2009), 155–164. <https://doi.org/10.1007/s10851-009-0161-2>
- Sinead Kearney, Wenbin Li, Martin Parsons, Kwang In Kim, and Darren Cosker. 2020. RGBD-Dog: Predicting Canine Pose from RGBD Sensors. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 8333–8342. <https://doi.org/10.1109/CVPR42600.2020.00836>
- Diederik P. Kingma and Jimmy Ba. 2017. Adam: A Method for Stochastic Optimization. arXiv:1412.6980 [cs.LG]
- Andrey Krekhov, Sebastian Cmentowski, and Jens Krüger. 2019. 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>
- Jehee Lee and Sung Yong Shin. 1999. A hierarchical approach to interactive motion editing for human-like figures. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*. 39–48.
- Guodong Liu, Jingdan Zhang, Wei Wang, and Leonard McMillan. 2006. Human Motion Estimation from a Reduced Marker Set. In *Proceedings of the 2006 Symposium on Interactive 3D Graphics and Games (Redwood City, California) (I3D '06)*. Association for Computing Machinery, New York, NY, USA, 35–42. <https://doi.org/10.1145/1111411.1111418>
- Ying-Sheng Luo, Jonathan Hans Soeseno, Trista Pei-Chun Chen, and Wei-Chao Chen. 2020. CARL: Controllable Agent with Reinforcement Learning for Quadruped Locomotion. *ACM Trans. Graph.* 39, 4, Article 38 (July 2020), 10 pages. <https://doi.org/10.1145/3386569.3392433>
- Naureen Mahmood, Nima Ghorbani, Nikolaus F. Troje, Gerard Pons-Moll, and Michael Black. 2019. AMASS: Archive of Motion Capture As Surface Shapes. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE, Seoul, Korea (South), 5441–5450. <https://doi.org/10.1109/ICCV.2019.00554>
- Jean-Sébastien Monzani, Paolo Baerlocher, Ronan Boulic, and Daniel Thalmann. 2000. Using an Intermediate Skeleton and Inverse Kinematics for Motion Retargeting. *Computer Graphics Forum* 19, 3 (2000), 11–19. <https://doi.org/10.1111/1467-8659.00393> arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/1467-8659.00393
- Dario Pavlo, Christoph Feichtenhofer, Michael Auli, and David Grangier. 2020. Modeling Human Motion with Quaternion-Based Neural Networks. *International Journal of Computer Vision* 128, 4 (2020), 855–872. <https://doi.org/10.1007/s11263-019-01245-6>
- Jonathan Smith. 2017. REP3D: 3D Human Motion Capture Dataset for Athletic Movement. *undefined* (2017). <https://www.semanticscholar.org/paper/REP3D%3A-3D-Human-Motion-Capture-Dataset-for-Athletic-Smith/189d9e826124ba7abcd79943eb85de21b2f10bfb>
- Ruben Villegas, Jimei Yang, Duygu Ceylan, and Honglak Lee. 2018. Neural kinematic networks for unsupervised motion retargeting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 8639–8648.
- Dongseok Yang, Doyeon Kim, and Sung-Hee Lee. 2021. LoBSTr: Real-time Lower-body Pose Prediction from Sparse Upper-body Tracking Signals. *Computer Graphics Forum* 40, 2 (May 2021), 265–275. <https://doi.org/10.1111/cgf.142631>
- 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 (July 2018), 11 pages. <https://doi.org/10.1145/3197517.3201366>