P3A - Research project

Manon Romain

January 23, 2018

1 Introduction

Reconstructing animal motion can have various applications, one of them being to recreate virtual environment for entertainment as accurately as possible. While human motion reconstruction is well studied, animals were not documented as well. There are several reasons for this, of which noncompliance of animals and lack of researchers interest. As a result, less data is available.

The main premise of this work is that animals in a herd are similar enough to be used as different poses of the same individual. Hence, it is possible to find motion embedded in a still picture of an herd. After recreating this motion cycle on various still pictures on the same animals, we made use of an a priori on the animal to fit a template skeleton and produce photo realistic 2.5D sprites of the animal. Although this technique doesn't return a template for interactive animation, it provides the user with quick results with very few work.

1.1 Previous work

Animating animals from still My work largely depends on [5], which reconstructs animated animals from a single 2D picture of an herd, reconstructing the motion path of a single individual using all poses present in the herd (see Section 2). The implementation of the paper led to questioning I will detail there. Rather than an exhaustive implementation, this work can be seen as a continuation with Sections 3 and 4 explaining my personal contribution, departing from the initial work.

3D animation Several papers have tried to animate animal motion from wild-life documentaries: both [1] and [3] relied on user intervention on a few key-frames to display a 3D animated animal. Instead, I decided to limit user intervention to a minimum therefore only limiting the results to a 2.5D final rendering.

Skeleton fitting Without any a priori, [3] have managed to recreate a 3D skeleton from

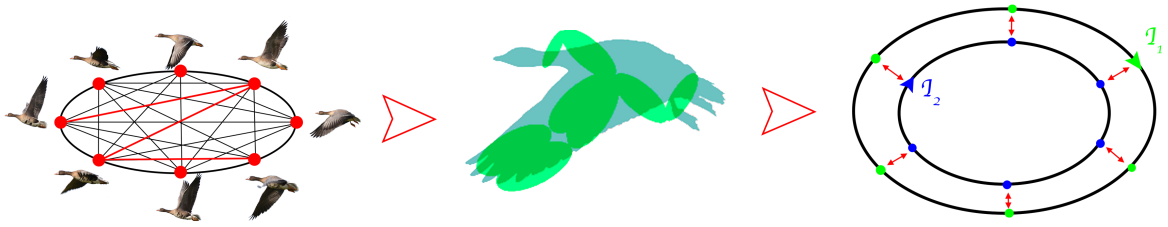


Figure 1: Overview of the method

a single view video, but requires the user to sketch the skeleton on a few key frames.

[4] have constructed a morphable model of quadrupeds skeletons from statistical analysis and proved that a set of three parameters (deduced from PCA) can be sufficient to efficiently fit the skeleton to a new animal. This approach fits the morphable skeleton on a preexisting 3D mesh, which could be an input of our approach. Creating the actual skeleton was not really our issue here, as we focused on fitting it to 2D view. Moreover, we don't actually use the skeleton for animation, but rather as a intermediary step to go from the output of [5] to a 2.5D animated animal.

1.2 Overview

Our method takes a still picture of an herd of a known species in a known orientation. We will assume it contains sufficient snapshots to create the animation. As you can see in 1, our pipeline consists of three main steps:

1. Ordering the 2D snapshots: this is the main interest of [5]
2. Fitting a template 3D skeleton to each of the selected poses
3. Unifying various orientations in sprites thanks to the 3D skeleton computed in step 2

This allows to produce 2.5D animated animals. The following three sections discuss each part separately.

2 Ordering 2D images

To order the snapshots and create a 2D animated animal from still pictures, I reimplemented the main contribution of [5].

The idea is to build a complete graph of snapshots, in which we will find a *motion path*.

2.1 Snapshot graph

The *snapshot graph* is a complete graph in which each node is a snapshot and the weight of an edge is a measure of the similarity between two snapshots.

2.1.1 Distance

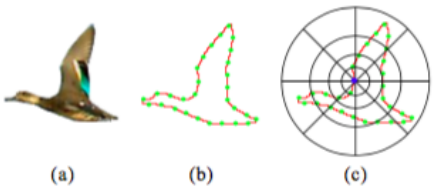
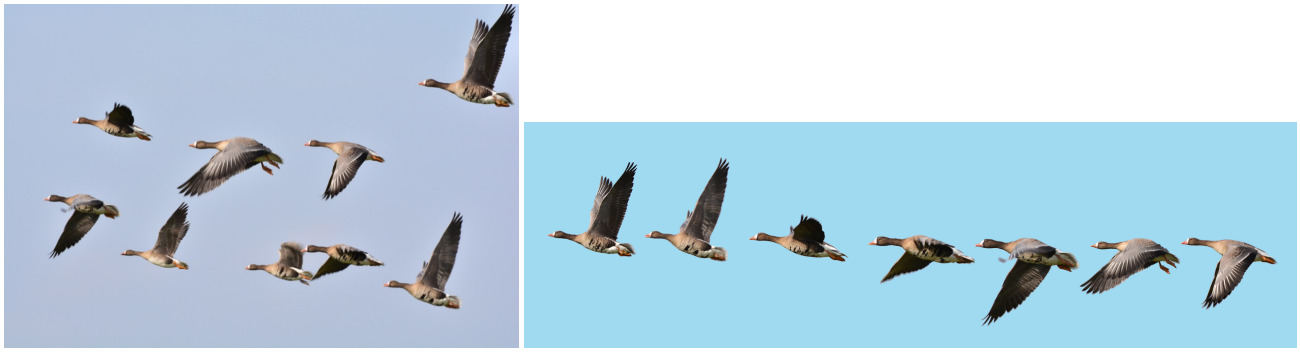


Figure 3: Taken from [5]. Shape feature extraction. (a) snapshot image (b) sampled contour points (c) *shape context* with log-polar histogram that computes the relative position of other contour points.

Measuring the similarity between two snapshots is done using shape features. It means we disregard the color information - which may vary from individual to individual - and work on binary images. For this project, we did the background subtraction manually, as it wasn't our purpose but several techniques (GrabCut, Mixture of Gaussians, ML, etc.) would automatize this step quite easily, although it may prove challenging for animals that blend with their environment. The shape of the snapshot is hence represented by a set of discrete points uniformly sampled on the contour.

The distance is then the shape context distance as introduced by [2]. This descriptor is invariant to translation, scaling, rotation, and even robust under small geometrical distortion, occlusion and outliers. For each contour point, shape context describes the distribution of the relative positions of all the other points in a spatial histogram. As shown in Figure 3(c), we construct bins that are uniformly distributed in log-polar space, and the number of contour points falling into each bin is one corresponding component in the resulting shape feature vector. The feature vectors of all contour points are then combined together to describe the shape.

Implementation point OpenCV extracts contour points and provides an implementation of [2]. However, I had to make sure that they were uniformly sampled: to do so, I set a minimum distance between two contour points as advised in the paper.



(a) Initial image. (b) Manual ordering of snapshot

Figure 2: Ordering of a flock of gooses.

Figure 4: OpenCV contour sampling: red dots are selected contour points, whereas green ones are redundant.

The distance between snapshot shapes S_k and S_l is then:

$$D(S_k, S_l) = \frac{1}{M_k} \sum_{i \in S_k} ||\mathbf{f}_i - \mathbf{h}_{j^*}|| + \frac{1}{M_l} \sum_{j \in S_l} ||\mathbf{f}_{i^*} - \mathbf{h}_j|| \quad (1)$$

where M_k is the number of points of S_k (resp. for S_l); f_i is the shape vector of the i-th contour points of S_k; the j^*-th contour point of S_l is the nearest neighbor of the i-th contour point of S_k ie $j^* = \text{argmin}_j ||\mathbf{f}_i - \mathbf{h}_j||$.

Xu and al. discuss the impact of self-occlusion of limbs that may induce incorrect corresponding points. However, I didn't tackle this issue as their solution involved some user intervention I wanted to avoid. Note that this issue affects just a few species like elephants.

2.2 Path extraction

After having computed the graph, the question is on how to extract a consistent motion cycle ie a path in the graph that would recreate the motion of a single individual. We assumed that the snapshot graph contains sufficient snapshots to form the motion cycle. Two adjacent snapshots in the path need to be similar enough to reproduce smoothness, but the whole path need to provide us with enough distinction to form a believable motion cycle. We also want to sample uniformly across the snapshot to reconstruct a believable motion. Note that there is no need to select all snapshots as some might be too similar and some outliers.

2.2.1 Cost function

Then, the required qualities of the path are local similarity, global distinction and sampling uniformity. To find a satisfying path, Xu and al. try to minimize the following cost function.

Let's denote the total number of snapshots as N. The path in the snapshot graph is $\mathcal{I} = \{I_i\}_{i=1...L}$ where $L <= N$ is the length of the selected path.

A measure for local similarity (smoothness) is defined as the mean of all shape distance between adjacent nodes in the path:

$$C_s = \alpha \cdot \text{E}(\{D(I_i, I_{i+1})\}) \quad (2)$$

where α is a normalization factor defined as the maximum distance between two nodes in the graph ie $\alpha = \max_{k,l} D(S_k, S_l)$. The measure for sampling uniformity is the variance of the same set of distances:

$$C_u = \text{Var}(\{D(I_i, I_{i+1})\}) \quad (3)$$

Finally, global distinction can be measured as the sum of all distances between any two snap-

shots in the path, ie:

$$C_d = \alpha \cdot \frac{\sum_i \sum_{j,j \neq i} D(I_i, I_j)}{L \cdot (L-1)} \qquad (4)$$

Therefore, the optimal path $\mathcal{I}$ will be found by minimizing the following cost function (maximizing local similarity, sampling uniformity and global distinction) formulated as:

$$C(\mathcal{I}) = C_s + \lambda_1 C_u + \lambda_2 (1 - C_d) \qquad (5)$$

The coefficients were empirically set as $\lambda_1 = 2.5$ and $\lambda_2 = 0.5$ in the original paper, we didn't conduct an other analysis to confirm their results.

2.2.2 Optimization scheme

To find the optimal path, Xu and al. argue that they use simulated annealing as described in Appendix A of their paper (see fig.5). However, I believe this is a mistake. Indeed, they argue they *randomly generate* a new path at each iteration, whereas simulated annealing requires to switch between neighboring configurations - this might be hard to define because the path length varies. This *randomness* makes no sense as there would be no benefit to select a worse configuration with a small probability, as one would not be stuck in a local minimum.

Note that there is also a minor indentation error on the last line on the pseudo code of figure 5.

Therefore I chose to get rid of the simulated annealing scheme and rather generate a random path at each iteration for a certain number of times. I computed the probability of reaching the optimal solution in Appendix A.

2.2.3 Half and full cycles

Xu and al. explain that:

> motion cycles of animal can be classified in two categories: half-cycle motion and full-cycle motion. Full-cycle motion contains completely distinct states in the cycle. The walking of mammals usually belongs to this type. [...] In contrast, if the motions of two half cycles are mirrored to each other, we call this type of motion the half-cycle motion. One typical example is the flapping of birds, where the motion of up-swing half cycle is mirrored to the motion of down-swing half cycle. So a full-cycle motion corresponds to a closed path in the graph while the half-cycle motion corresponds to an open path.

Hence, we constrained the path to pass through two extremal nodes, the further away in the graph. To avoid selecting outliers at this step, we considered, as advised by Xu and al., that distances in the graph should conform a Gaussian model $\mathcal{N}(\mu, \sigma^2)$, so we exclude the nodes which minimal distances to others nodes in the graph exceeds the threshold $\delta = \mu + c * \sigma$. They experimentally set $c = 2.0$ and I didn't try to verify this result.

2.3 A partial implementation

I didn't re-implement every single result of the initial paper as I considered some of them to be refinements and not the core of the article. However, the adjustments would have to be implemented to end up with a more realistic result. For instance, pose, morphology and appearance consistency would have to be refined. Although the individuals in a herd share very similar features, changes in scale, orientation, morphology or color might affect quite strongly the end result. The animation might also suffer from unrealistic speed or jolting images. The speed is to be set by the user, and interpolation performed between snapshots.

3 Fitting the 3D skeleton on 2D binary images

The second step of our pipeline towards creating 2.5D animated animal is fitting a template skeleton to our extracted snapshots.

Appendix A: Motion Cycle Optimization

Algorithm 1 presents the pseudo-code of motion-path optimization. The motion path contains at least two extremal nodes in snapshot graph. During each iteration, we randomly generate a new path in the graph, which is associated with a valid path length L. Based on the energy function (Equation 5), the new path is accepted or rejected according to an annealing strategy [Pepper et al. 2002]. To guarantee a stable convergence, a certain number of iterations (K) is set at temperature T before the next annealing process. In our implementation, we set $T_0 = 50,000$, $K = 500$, $AnnealFactor = 0.992$, and $limit = 0.01$. It takes less than 35 seconds to search the motion path even for the bird example with 30 snapshots (Figure 1(a)).

Algorithm 1: Pseudo-code of motion cycle optimization

choose a path length L
generate an initial path $\mathcal{I}_\mathcal{L}$ with length L
choose a temperature $T = T_0 > 0$
C_{old} = C($\mathcal{I}_L$)
Loop ($T >$ limit)
  Loop (K times)
    choose a new path length L
    generate a new path $\mathcal{I}_\mathcal{L}$
    C_{new} = C($\mathcal{I}_L$)
    $\Delta C = C_{new} - C_{old}$
    if $\Delta C \leq 0$, accept new path
    else if $exp(-\Delta C/T) \leq rand(0,1)$, accept new path
    else reject new path
  $T = T \times AnnealFactor$

Figure 5: Extract from [5]. The highlighted parts are discussed in Section 2.2.2

3.1 Skeletons

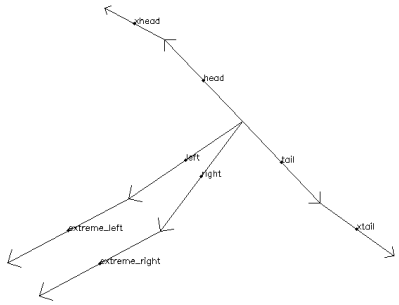


Figure 6: A bird skeleton

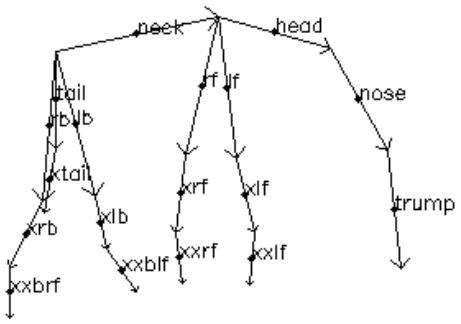


Figure 7: An elephant skeleton

3.1.1 Representation

We use a hierarchical representation for our skeletons, meaning that a skeleton is represented as a tree where joints are nodes and bones edges. We used the standard convention of setting the pelvis as root of the tree.

[4] investigates the impact of parametrization when fitting their skeleton to 3D meshes: Euler angles vs. Quaternion for rotations and Local vs. Global configuration for edge lengths. According to their analysis, it seems that Quaternion (that avoid gimbal lock) and global configuration (that avoid accumulating errors along edges) is the best setup.

However, we decided to fix edge length and only optimize rotation, hence the choice of local vs. global was of little importance, but we used quaternions to parametrize rotations.

3.1.2 Creation of the template skeleton

For now, the template skeleton of the two species (goose and elephants) consists of very few joints as they were designed manually, without any help from a more advanced designing software, like in figures 6 and 7

Our template skeleton could be the morphable skeleton computed in [4] adapted to our animal, a skeleton created by an expert animator, or from anatomical sketches. In all cases, the template skeleton should be at rest and the type of animal specified by the user.

3.2 Fitting process

3.2.1 Hypothesis

Let's recall that our optimization only tries to find optimal rotations for each joint but keeps edges (bones) lengths constant.

To fit the 3D-skeleton onto a 2D shape, we first need to project the skeleton onto a plane. We chose an orthonormal projection along a user-defined axis. Indeed, in wild-life documentaries, animals are often seen from far away, an orthonormal projection would not significantly distort the view. Moreover, the user can easily specify from which angle the herd is seen. The constraint is that all animals in the herd are supposed to have the same orientation.

The projected skeleton has itself a tree structure but no intrinsic surface. Our goal is to match the shape of our animal (a binary image) to this skeleton, so we need to give it some volume. We artificially plotted ellipses along edges. The length l of those ellipses is the actual length of the edge while its width varies in relation with $l^{3/2}$.

Finally, our last hypothesis is that the rotations will be done in the projection plane, as we have absolutely no information on the geometry of the third dimension.

3.2.2 Error metric

Figure 8: Fitting the skeleton, in blue is the target and in green the skeleton projection, r is the current iou value

As we can see in figure 8, we used a cost function that is well known in object detection: *Intersection over Union* (IOU) also known as the *Jaccard Index*. Actually, we want to maximize this objective function. For a reference surface S and a query surface Q, this metric is defined as:

$$J_S(Q) = \frac{|S \cap Q|}{|S \cup Q|} \tag{6}$$

J_S varies between 0 and 1.

3.2.3 Optimization scheme

We experimented with various optimization schemes in order to fit our skeleton to our shapes. The one I went with is simply selecting the best configuration among randomly picked configurations. At each iteration, each joint is rotated by a random angle. I also gave a try to simulated annealing but I realized it was ill-advised as we are actually looking for a local minimum, not too far from the resting pose.

Overlapping The issue with the metric we chose is that overlapping limbs at an extreme, might result in a better score that a plausible configuration. This is why we should look for a local optimum rather than a global one.

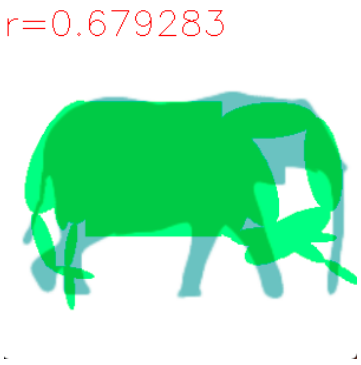


Figure 9: Example of overlapping limbs

Under constrained We didn't specify any morphological constraints for the skeletons, which might result in unintuitive results. For instance, birds' wings should have a larger rotation amplitude than elephants' knee.

4 From 2D to 2.5D

4.1 Linking the 2D paths of 2 distinct orientations

We apply the two previous steps on two different images of the same animal, with different orientations eg. from the top and the side. After matching every snapshot in the motion paths with their corresponding skeletons, we then have two cycles $\mathcal{I}_1$ and $\mathcal{I}_2$ (half or full) of snapshots and associated skeletons.

The goal is to find how to make the two cycles coincide. That is where we make use of the skeletons computed before.

We will assume that both cycles have the same length. This is a pretty strong assumption but it is necessary to reconstruct a smooth animation, as both sprites must be animated in sync. We will also assume that both motion paths are sampled uniformly across the cycles, as should guarantee the optimization of the C_s term in Eq.5.

First of all we need to restore full cycles only: as explained in Section 2.2.3, the down-swing half cycle is mirrored to the motion of the up-swing half cycle.

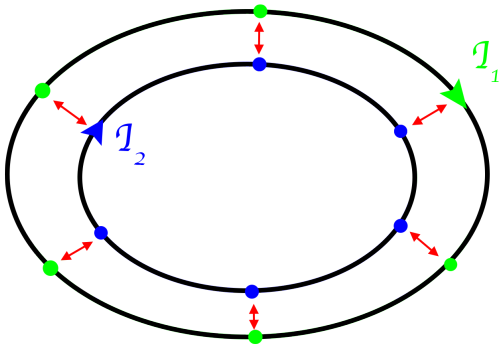


Figure 10: Adjusting cycles $\mathcal{I}_1$ and $\mathcal{I}_2$

Then to fit both cycles, our approach is greedy but quite simple. For every cyclic permutation $\mathcal{P}$, let's denote as A_i the i-th element of $\mathcal{I}_1$ and as B_i the i-th element of $\mathcal{P}(\mathcal{I}_2)$, hence we compute a cost defined as:

$$C_{\mathcal{I}_1,\mathcal{I}_2}(\mathcal{P}) = \sum_{i=0}^{l-1} \sum_{\substack{\mathbf{q}\in A_i^s \\ \mathbf{q}'\in B_i^s}} d(\mathbf{q},\mathbf{q}') \qquad (7)$$

where A_i^s is the vector of quaternions representing the skeleton associated to the node A_i in the cycle. For two quaternions $\mathbf{q}$ and $\mathbf{q}'$, the distance is defined as:

$$d(\mathbf{q},\mathbf{q}') = 1 - \langle \mathbf{q},\mathbf{q}' \rangle^2 \qquad (8)$$

This gives a rough estimate of the distance. In particular, it gives 0 whenever the quaternions represent the same orientation, and it gives 1 whenever the two orientations are 180 apart.

Minimizing this cost function is done in quadratic time, trying out all possible rotations.

4.2 Displaying

The display of our final animation is done through a small JavaScript applet, kindly provided by Damien Rohmer.

5 Results and Ameliorations

As creating photo realistic sprites from still pictures is not a well-researched application, it is been hard finding working examples to compare this work to.

A lot of details have been left out to get to the core of the method, but those should be reimplemented to give the user a truly satisfying result. We can still see inconsistencies in 11

Besides, to continue this work, we should try to give the user some leverage, so that he/she could refine results of the method to his/her tastes.

Appendix A Convergence

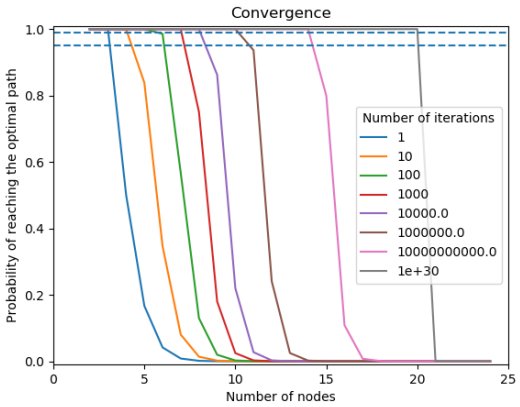


Figure 12: Probability of reaching the optimal path

N denotes the number of nodes in the graph.
To pick a random path in the graph, we first pick its length uniformly between 2 (or 3 if the path is supposed to represent full-cycle motion) and N. Then, we uniformly choose $L-2$ nodes (as the start and stop nodes are fixed) in the remaining $N-2$ nodes. Let's denote as $\{\mathbf{X}_i\}$ the random variable of the path at iteration i, $\{L_i\}$ the length of the path at iteration i and T

(a) From the side (b) From the top (c) Diagonal view

Figure 11: Final result in the JavaScript applet

the random variable of the iteration at which the optimal path is chosen.

$$(\mathbf{X}_i = \mathbf{x} \Rightarrow L_i = |\mathbf{x}|) \Rightarrow \mathbb{P}(L_i = |\mathbf{x}||\mathbf{X}_i = \mathbf{x}) = 1 \tag{9}$$

$$\mathbb{P}(\mathbf{X}_i = \mathbf{x}) = \mathbb{P}(\mathbf{X}_i = \mathbf{x}|L_i = |\mathbf{x}|)\mathbb{P}(L_i = |\mathbf{x}|) \tag{10}$$

As L_i is chosen uniformly in $[2, N]$:

$$\mathbb{P}(L_i = |\mathbf{x}|) = \frac{1}{N-1} \tag{11}$$

And given L_i, $\mathbf{X}_i$ is a path chosen uniformly.

$$\mathbb{P}(\mathbf{X}_i = \mathbf{x}|L_i = |\mathbf{x}|) = \frac{(N - |\mathbf{x}|)!}{(N-2)!} \tag{12}$$

Let's compute the probability of reaching the optimal path $\mathbf{x}^*$ (we can assume it's unique) before the end of iterations:

$$\mathbb{P}(T < p) = 1 - \mathbb{P}\left(\bigcap_{i=1}^{p} \{\mathbf{X}_i \neq \mathbf{x}^*\}\right) \tag{13}$$

As the $\mathbf{X}_i$ are independent, we have:

$$\mathbb{P}(T < p) = 1 - \left(1 - \frac{(N - l^*)!}{(N-2)!}\right)^p \tag{14}$$

$$\mathbb{P}(T < p) \geq 1 - \left(1 - \frac{1}{(N-2)!}\right)^p \tag{15}$$

We thus have a lower bound for the probability of reaching the optimal path, and can deduce a sufficient number of iterations depending on the number of snapshots in the graph.

References

[1] Favreau, L., Reveret, L., Depraz, C., and Cani, M.-P. Animal Gaits from Video. Research Report RR-5170, INRIA, 2004.

[2] Mori, G., Belongie, S., and Malik, J. Efficient shape matching using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence 27*, 11 (2005), 1832–1837.

[3] Reinert, B., Ritschel, T., and Seidel, H.-P. Animated 3d creatures from single-view video by skeletal sketching. In *GI '16: Proceedings of the 42st Graphics Interface Conference* (2016).

[4] Reveret, L., Favreau, L., Depraz, C., and Cani, M.-P. Morphable model of quadrupeds skeletons for animating 3D animals. In *ACM SIGGRAPH / Eurographics Symposium on Computer Animation, SCA'05* (Los Angeles, United States, July 2005).

[5] Xu, X., Wan, L., Liu, X., Wong, T.-T., Wang, L., and Leung, C.-S. Animating animal motion from still. *ACM Transactions on Graphics (SIGGRAPH Asia 2008 issue) 27*, 5 (December 2008), 117:1–117:8.