Bachelor Thesis: 3D Animation of Animal Motion from Still

Guillaume Loranchet

April 2020

Abstract

The goal of this paper is to present a method to go from a single herd image to a 3D animated motion. The first main idea is that a herd image gives enough different positions to infer a full cycle motion. The second one is that most of the 3D informations of an animal can be captured with a side view image. We first map a 3D skeleton on several 2D shapes, then we order them to create the animation. We finally use billboards to display the final 3D animal.

1 Introduction

1.1 Related Work

Animal motion, due to the fact that they mostly live in a wild environment, is way less documented than human motion. The limited data is an important constraint when studying animal motion, which motivated Xu et al. [1] to first try to animate animal motion from a single image. Our paper is mainly based on the work of Xu et al. and Manon Romain [2] who searched for a method to go from 2D animation to 3D. Her main contribution was to use a skeleton that is way easier to manipulate in a 3D environment.

1.2 Approach

2 2D Animation

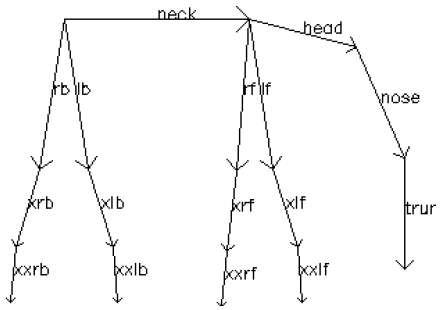


Figure 1: Initial Skeleton

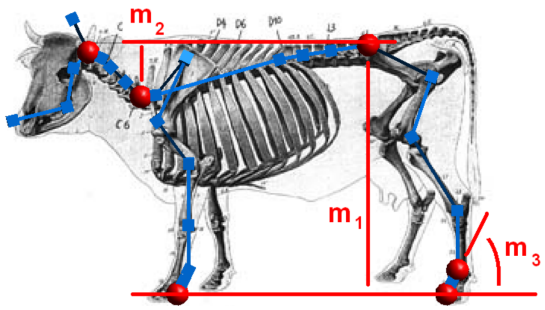


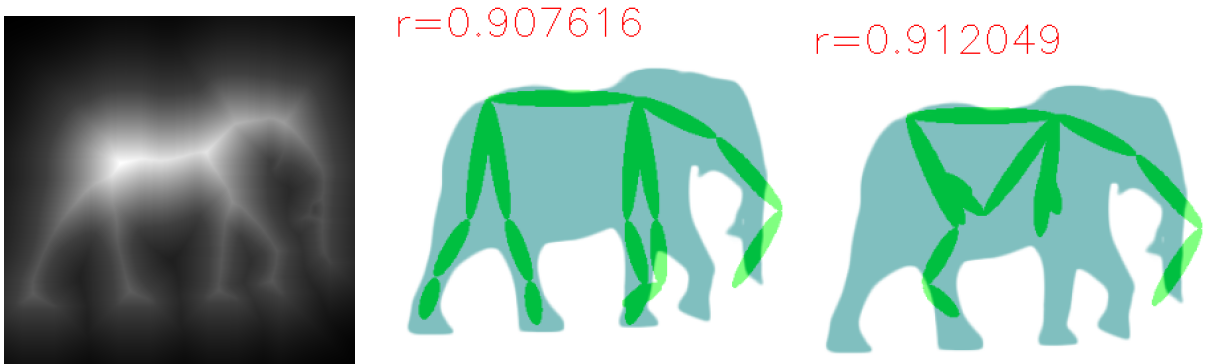
Figure 2: Morphable model (by Reveret et al. [4])

We start by constructing a 3D skeleton. The skeleton is constructed manually but could be made quicker by using the method presented in the paper Morphable model of quadrupeds skeletons for animating by Reveret et al. (2009) [4] (Figure 2). Indeed, with a database of 8 already made skeletons, and by taking 3 key measures, they can create a skeleton for any quadrupeds. Those three measures could also be useful for the second step, that is the placement of the skeleton on the different shapes. We could, for each image, select the hip (equivalent to m_1 on Figure 2), resize the image and translated it so that all images have the same m_1. The first two steps could probably be more automated but a small error at this point could compromise the rest of the process. With only three initial points to create the skeleton, and one additional point for each images, we'd make sure that there will not be any important mistake up to this point. Finally, for each bone, we add an ellipse that will allow us, for a reference surface S and a query surface Q, to define a score r as follow:

$$r = \frac{S \cap Q}{Q}$$

In other words, r will be maximum (equal to 1) when the entire ellipse is inside the animal shape. In her paper, Manon Romain [2] uses the Intersection over Union (IOU) which measure how much the skeleton fills up the entire shape. But as the ellipses are not constructed based on the shape of the animal, it would be more accurate to measure only how much the ellipse is inside the shape.

The next step in to make to skeleton fit the shape. The first method we tried was using distance transform. Each pixel of the image is attributed a value between 0 and 1 depending how far it is from the closest boundary. We firstly tried a naive algorithm by simply rotating each bone such that the local score is maximum. However, this method depends too much on the size of the ellipses and the distance transform image itself is isn't close enough from the correct skeleton.



(a) Distance transform

(b) Apply rotation only on the lower part

(c) Apply rotation on the entire leg

Figure 3: Distance Transform naive algorithm

With the second method, we tried to find the middle of the legs (or trump). For each leg, the algorithm starts by the middle bone (e.g xrf) and finds the two border of the leg. It first detects toward which direction the bone should rotates by looking at the local score and then rotates the bone until the extremity changes of surface (i.e going from outside to inside the shape). We can average the two positions to find a good approximation of the middle of the leg. We also add constraints on the angle of each bone such that if the angle reach one of the boundary, we know the direction was wrong and avoid absurd results. Furthermore, we assume that the extremity is either on the left side of the leg, inside or on the right side, thus one should always be able to find the two borders. Finally, given a good enough initial position, it should accurately dissociates the two legs.

Once the lower part of the leg (xrf and $xxrf$) are well placed, we can improve a bit the position of the leg by rotating the upper part (rf) such that the score of the leg increases (Figure 4c). We also make sure that $xxrf$ keeps the same direction as it is already well positioned. We can see that it doesn't improve a lot the global position of the leg. As before, it is also very dependant on the size of the ellipse.

(a) Initial skeleton position

(b) Middle Point Algorithm

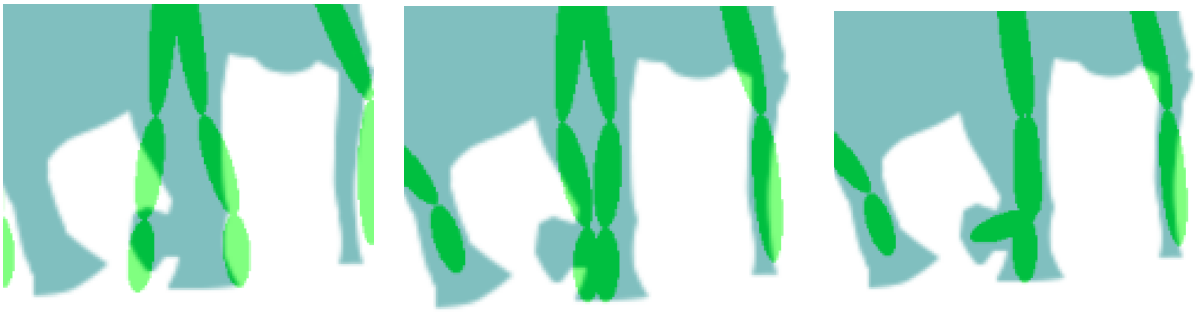
(c) Add improvement on the upper part

Figure 4: Middle Point Algorithm (gif)

While this method gives an reasonably fast and accurate mapping of the skeleton, it also has some issues. The first could occur if the skeleton doesn't perfectly fit the shape. Indeed if the skeleton is bigger than the leg (Figure 5), one of our assumption doesn't hold anymore and the extremity will either be block in the "corner" of the leg or rotate until it reaches the maximum angle.

Figure 5: Skeleton size problem

The other issue concerns the position where the two legs are overlapping each other. As shown in Figure 6, the two front legs are mapped to the same leg. This problem could also be due to the imprecise construction of the skeleton or resizing. While we haven't found yet how to fix this specific case, one solution could be to add another constraint on the leg. For example, in a walking movement, $xxrf$ shouldn't be able to be directed toward the front while being behind the left leg, and thus conclude that it should rotate 90 degrees in the other direction.



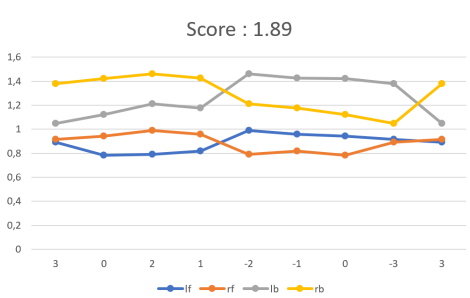
(a) Initial skeleton position (b) Middle Point Algorithm (c) Manual placement

Figure 6: Middle Point Algorithm issue

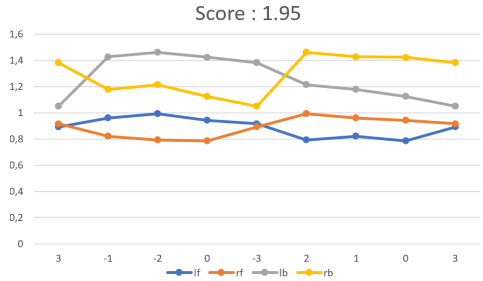
We now have n different positions for the skeleton. As, by construction, the left leg is always in front of the right one, we also create n new positions by inverting the order of the legs. However, by doing this, we insure that the inversion of the front and of the back legs happen at the same time, which is usually not exactly true. Indeed, most of the time, the back legs cross a bit before the front legs.

We finally need to find an optimal ordering of the $2n$ positions in order to make a realistic movement. For each position, we associate a vector of dimension 4 containing the angle between each leg and the horizontal axis (Figure 7). We can then compute a simple score between two positions by taking the norm of the difference of the two vector. The score measure how different two consecutive positions are. We use a Dijkstra algorithm to minimize the total score of the ordering. The algorithm start with two initials positions and at each step, add the node that minimizes the total score, while keeping in memory the score of the other possible paths. In the worse case, this algorithm has the same complexity as a naive algorithm but most of the time find a minimum before exploring every path. Furthermore, it is guaranteed to find the smallest score. We also make sure that the last position is also the first one to make a full cycle. We add a last constraint which is that two opposite positions (same position with an inversion of the legs) can't be juxtaposed. Indeed, the motion is not symmetric whether the leg is moving forward or backward.

Figure 7: Leg angle α



(a) Without juxtaposition constraint



(b) With juxtaposition constraint on inverse positions

Figure 8: Optimal ordering using Dijkstra Algorithm

This ordering is quite different from the one used in the Animating Animal Motion from Still paper by Xu et al. [1]. First they ordered animal shapes while, thanks to the skeleton, we have more local information on the position of the legs, which is the most important part to take into account when ordering animal positions. The second difference concerns the optimization method as they used simulated annealing. However, Manon Romain [2] pointed that this could be an error as "This randomness makes no sense, there would be no benefit to select a worse configuration with a small probability, as one would not be stuck in a local minimum". Instead, she chose to generate a random path a certain number of times. Even if it is quite efficient for a small number of nodes, it quickly increase as for only 10 nodes, it would need on average more than 10000 iterations. The Dijkstra algorithm should be much quicker to converge.

You can see the result in Figure 9 by clicking of the image. The ordering might not be the best one but the motion seems realistic. Unfortunately, the few number of images doesn't give enough information on the back legs. This could presumably not be solved without more prior data on the movement. Finally, the time between two key positions shouldn't be uniform. We tried to minimize the second derivative in Figure 8 to get similar positions closer to each others, but didn't search more in that direction.

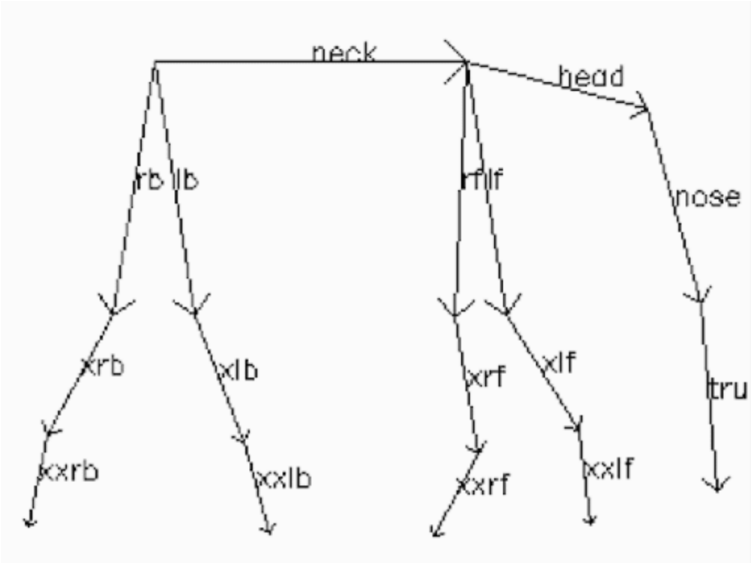


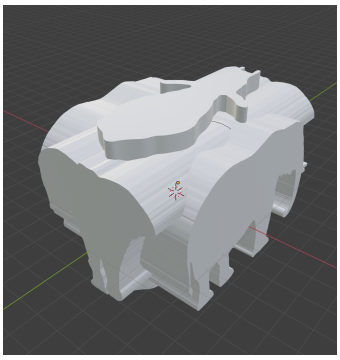
Figure 9: Final 2D Skeleton Animation (gif)

3 3D Render

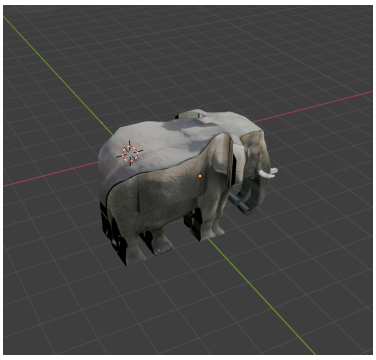
3.1 3D Representation

The first step was to choose how to represent the animal in 3D. The two possibilities were to either create a 3D object (Figure 10) or display 2D billboards and change their orientation depending on the position of the camera (Figure 11).

The first method consist in taking different 3 views, from the side, from above and from the front / back. We can then extrude them, take the intersection and apply the images as texture. The main advantage of this method is that we obtain a 3D object that has a volume and feels more real. On the other hand, it's often hard to find perfect lateral views of an animal and will result most of the time in totally unrealistic representation. It is also way heavier in term of rendering. For those reasons, we'll prefer using billboards.



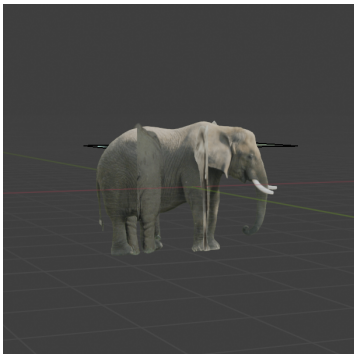
(a) Extrude Faces



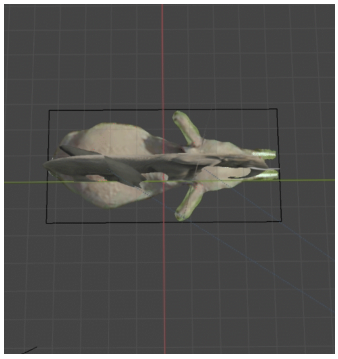
(b) Intersection with texture

Figure 10: 3D Object

To represent a static animal with billboards, we first place the side view image as this one alone already covers most of the camera angles. We then add two images from the back and the front that rotate to face the camera. Finally, we add a picture from above that only display the part behind the side view image to avoid noticeable intersections.



(a) From side (gif)



(b) From above (gif)

Figure 11: Billboards

3.2 Animation

The first step is to animate the skeleton. We already have the $2n$ ordered key positions but we still need to make the skeleton move forward. To obtain a more realistic motion, we determine the pivot leg, i.e the foot that stays fix with respect to the horizontal axis, by taking the lowest of the two front feet.

In order to animate a still image from the side, we first need to deconstruct it into several pieces that we can attach to each bone. We already have n side view images from the initial herd picture and the n corresponding skeleton positions. However, the quality of the texture and the lightning of the animal is rarely usable. Furthermore, an automatic deconstruction would need to find the front leg with respect to the camera position to avoid strange shadows. For those reasons, we choose to use a different side view images and do the deconstruction manually. The automatic placement on the skeleton works well for the legs but we still had to adjust manually for the head and body. We also tried to curve the 2D planes to add volume to the shape (on the back leg, Figure 13). With only the side view image, we could go from about 30 to 150 degrees horizontally and up to 60 degrees vertically (Figure 14). This could probably be improved by curving even more the images.

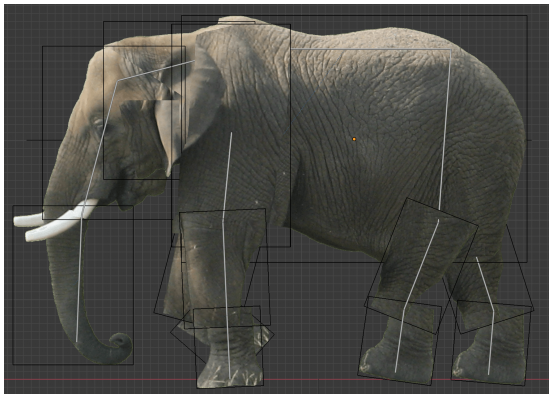
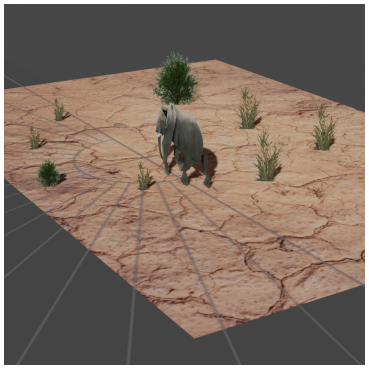


Figure 12: Images attach to the skeleton

Figure 13: Curved images on the back leg



(a) 30 degrees (gif)

(b) 150 degree (gif)

(c) 60 degrees Vertically

Figure 14: Maximum view angles

Finally we can add the view from above. We didn't managed to make it transparent which result in a more obvious intersection. Due to the movement, the views from the front and back are too noticeable to be included without modifications. A solution could be to deconstruct them like we did for the side view image and place them perpendicular to the first one. But once again, the colors will probably not match.

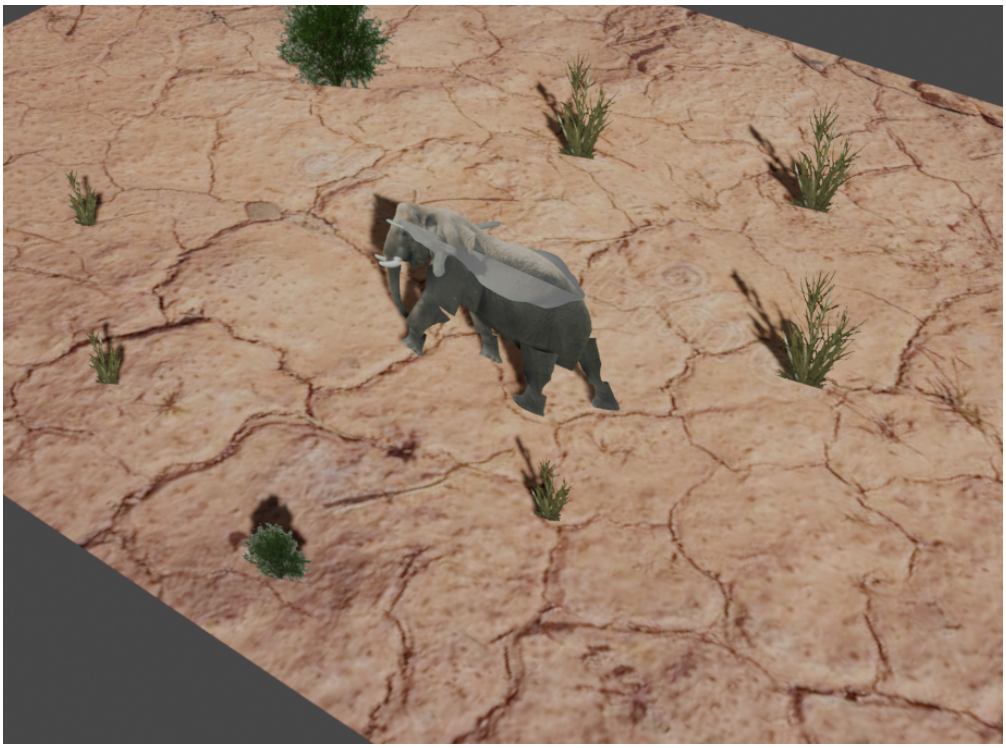


Figure 15: Final 3D Animation (gif)

4 Conclusion

Starting from a single elephant herd image, we managed, with a bit of manual correction, to create and animate an almost 3D animal. The middle point algorithm works well for most of the positions but still needs to be improved to be completely usable. The ordering using Dijkstra algorithm is quite efficient in this case, as we don't have too much initial images but might not be optimal if there is more images. The easiest way to represent a 3D animal, especially if he's seen from far, clearly seemed to be with billboards. We then deconstructed a side view image to animate each bone individually, which is enough to cover the majority of the camera angles from the side.

5 Future work

The all process is far from being fully automated. The middle point algorithm still need to be fix to work for every positions. Concerning the animation, the only way to significantly improve the motion would be have an already animated quadruped skeleton and maybe try to adapt it to the different animal morphology and key positions. An important part of the deconstruction and reconstruction of the side view image could also be more automatic. It could also be worth investigate more on how to add more volume to a 2D billboard to fit an animal shape if we want to only use the side view image, or if we add the view from the front / back / above, improve the blending and the color matching. Finally, the skeleton could still be useful to rig an already existing 3D animal model and gives a starting point for the animation.

References

[1] Xuemiao Xu, Liang Wan, Xiaopei Liu, Tien-Tsin Wong, Liansheng Wang and Chi-Sing Leung *Animating Animal Motion from Still*. ACM Transactions on Graphics (SIGGRAPH Asia 2008 issue), Vol. 27, No. 5, December 2008, pp. 117:1-117:8.
http://www.cse.cuhk.edu.hk/ ttwong/papers/flock/flock.html

[2] Manon Romain, 2018 *P3A - Research Project*
https://github.com/guillaume-lrt/Thesis-animal-animation/blob/master/Documents/p3a-research-project.pdf

[3] Bernhard Reinert, Tobias Ritschel, Hans-Peter Seidel *Animated 3D Creatures from Single-view Video by Skeletal Sketching*. 2016.
http://resources.mpi-inf.mpg.de/DeformableObjectExtraction/AnimatedCreaturesCompressed.pdf

[4] Lionel Reveret, Laurent Favreau, Christine Depraz, Marie-Paule Cani *Morphable model of quadrupeds skeletons for animating 3D animals*. ACM SIGGRAPH / Eurographics Symposium on Computer Animation, SCA'05, Jul 2005.
https://hal.inria.fr/inria-00389338

[5] Philippe Decaudin and Fabrice Neyret *Volumetric Billboards*. ACM SIGGRAPH / Eurographics Symposium on Computer Animation, SCA'05, Jul 2005.
http://www-ljk.imag.fr/Publications/Basilic/com.lmc.publi.PUBLI$_{A}rticle@11e3c2463e4_{4}865ce/index.html$

[6] Even Entem, Amal Parakkat, Marie-Paule Cani, Loïc Barthe *Structuring and layering contour drawings of organic shapes*. Joint Symposium on Computational Aesthetics and Sketch-Based Interfaces and

Modeling and Non-Photorealistic Animation and Rendering, Aug 2018.
https://hal.inria.fr/hal-01853410

[7] Even Entem, Loïc Barthe, Marie-Paule Cani, Frederic Cordier, Michiel van de Panne *Modeling 3D animals from a side-view sketch*. Computers and Graphics, Elsevier, 2015, 46, pp.221-230. ff10.1016/j.cag.2014.09.037ff. ffhal-01073059.
https://hal.inria.fr/hal-01073059

[8] Ljiljana Skrba, Lionel Reveret, Franck Hétroy, Marie-Paule Cani, Carol O'Sullivan *Quadruped Animation*. Eurographics '2008 - 29th annual conference of the European Association for Computer Graphics, Apr 2008, Hersonissos, Creete, Greece. pp.7-23. ffinria-00331715v2f
https://hal.inria.fr/inria-00331715v2

https://github.com/guillaume-lrt/Thesis-animal-animation