
# Bachelor Thesis: 3D Animation of Quadruped Motion from Still

Guillaume Loranchet
Supervisors: Pierre Ecormier-Nocca, Damien Rohmer and Marie-Paule Cani

April 2020

## Abstract

This paper presents a method to go from a single herd image to a 3D animated motion. The goal is to create an animal that could be moving as part of the background of a virtual world. Therefore, they will most of the time be seen from far away with potentially trees or plants in front, which allows more freedom in the render. 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 information 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. Finally, we use billboards to display the final 3D animal.

## 1 Introduction

A lot of papers already tackled the issue of animating 2D animal images or creating 3D animals based on 2D data (pictures or videos). Most of them aimed to perfect the animation by using more data or created a realistic 3D animal. However, when creating a virtual world, you might want to have animals moving in the background without spending too much time to create and animate them manually. Our goal was to find a method that should be fairly simple with as few user inputs as possible, even if it means losing some detail. The most difficult part was to limit the user input and have something as automated as possible. The majority of papers about animation of 3D animals heavily rely on user input, with sketching or by using already existing models. It is indeed quite hard to retrieve 3D information from only 2D data.

To answer those issues, we choose to use a single herd image to reconstruct the animation and billboards for the 3D view. A single image is usually easier to analyse and should, due to the asynchronism of the different individual, offer enough information on the animation of the animal. Previous works, like Xu et al.

[1], usually order the different key frames by studying the closeness of the shapes. As our final result needed to be a 3D animal, we had to use a 3D skeleton. We used this opportunity to first map the skeleton on the shapes and then order the skeleton key positions. The skeleton gives us more local information as we know where each point should be on all positions. The second innovative idea is to use billboards to represent a 3D animal. Billboards are 2D plans placed in a 3D environment. They are usually used to represent plants or trees, as less details is needed but also allow for an easy and quick render. However, the main usage of billboard consist of rotating them such that they always face the camera. Obviously, this method would not work for animals as they need to be aligned with their moving direction.

### 1.1 Related Work

Animal motion, due to the fact that they mostly live in a wild environment, is significantly 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.

Other papers worked on animating 3D animals from a single view but most of them are based on video. For example, B.Reinert [3], uses skeletal sketching input from the user on a side view video and creates a 3D creature with a union of cylinders in order to animate it. An other example is the work from L.Favreau et al.[5]. Also based on a video, and using Principal Component Analysis, they managed to select and order key images. However, they still needed to provide 3D pose examples.

### 1.2 Approach

As said previously, the 2D part of this paper is mainly based on the work from Manon Romain. Thus, we used the data Manon Romain worked with (see shapes: Figure 1 and skeleton: 2). Furthermore, there already exists methods to extract shapes from an image so we didn't re-implemented this part. The four positions are selected from a herd image, then converted into binary images. Indeed, we are only interested in the shape of the animal. The images could be useful to prevent self occlusion, or in other words, to differentiate the left and right legs. We could for example look at the shadows made by the legs to determine which one is closer to the camera. But as both legs have the same motion, each leg position should correspond to both the left and right leg, at different moment of the movement. We can notice that the four positions come from different animals, which shouldn't be a problem as we don't focus on the global shape but only on the legs.

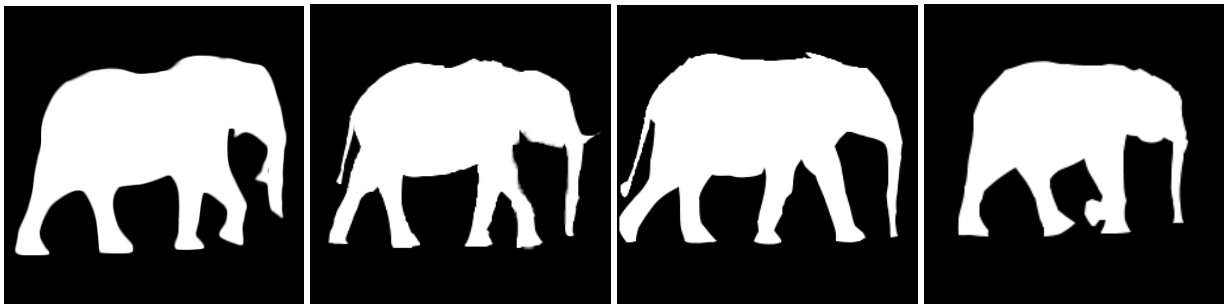


Figure 1: 4 Initial shapes

# 2 2D Animation

## 2.1 Construction of the Skeleton

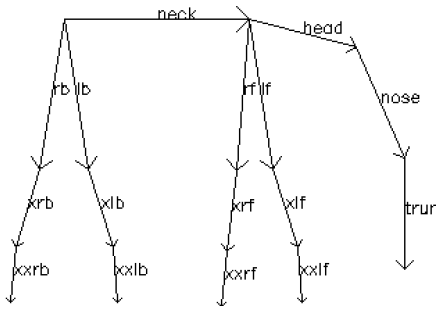



Figure 2: Initial Skeleton

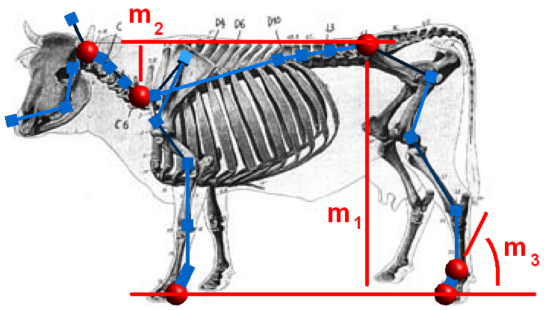



Figure 3: Morphable model (by Reveret et al. [4])

We start by constructing a 3D skeleton. The skeleton is constructed manually but could be made semi-automatically by using the method presented in the paper Morphable model of quadrupeds skeletons for animating 3D animals by Reveret et al. (2009) [4] (Figure 3). Indeed, given a database of 8 already made skeletons, and by taking 3 key measures, Reveret et al. method could create a 3D skeleton for any quadruped. 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 3), 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 image, we can guarantee that no important errors have been made op 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:

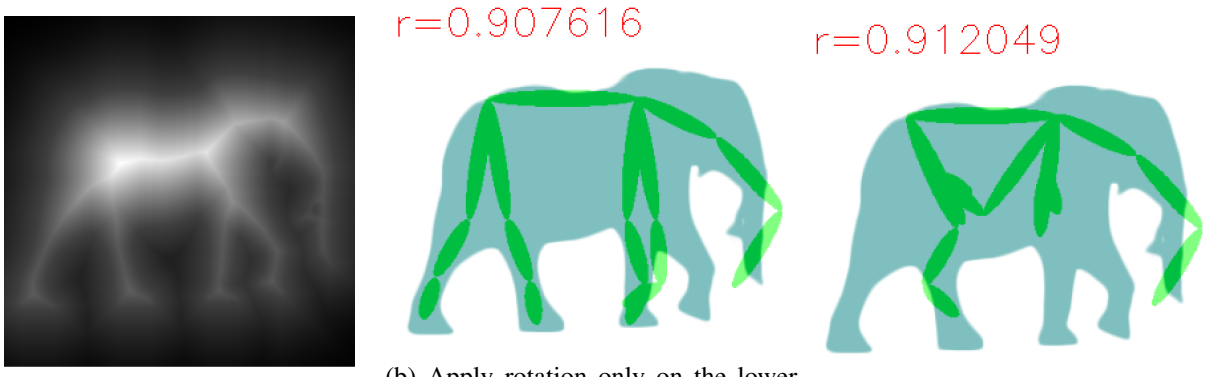
$$r = \frac{S \cap Q}{Q}$$

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.

## 2.2 Fitting the skeleton on the shapes

### 2.2.1 Distance Transform

The next step is to make the skeleton fit the shape. The first method attempted used distance transform. Each pixel of the image is attributed a value between 0 and 1 depending on the distance 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 excessively on the size of the ellipses, and the distance transform image itself isn't close enough from the correct skeleton. However, the distance transform could still be useful to compute local score more accurately.




(a) Distance transform

(b) Apply rotation only on the lower part

(c) Apply rotation on the entire leg

Figure 4: Distance Transform naive algorithm

### 2.2.2 Middle point

The goal of the second method was to find the middle of the legs (or trump). For each leg, the algorithm (see the pseudo code in the appendix) starts by the middle bone (e.g $xrf$) and finds the two borders of the leg. It first detects toward which direction the bone should rotate by looking at the local score and then rotate the bone until its 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 reaches one of the boundary, we know the direction was wrong and avoid absurd results. Furthermore, we assume that the extremity is either on the inside, left side or right side of the leg, thus one should always be able to find the two borders. Finally, given a good enough initial position, it should accurately dissociates between the two legs.

Once the lower part of the leg ($xrf$ and $xxrf$) is well placed, we can slightly improve the position of the leg by rotating the upper part ($rf$) such that the score of the leg increases (Figure 5c). We also make sure that $xxrf$ keeps the same direction as it is already well positioned. However, we can see that there is only a small improvement in the global positioning of the leg. As before, it is also heavily dependant on the size of the ellipse.


(a) Initial skeleton position

(b) Middle Point Algorithm

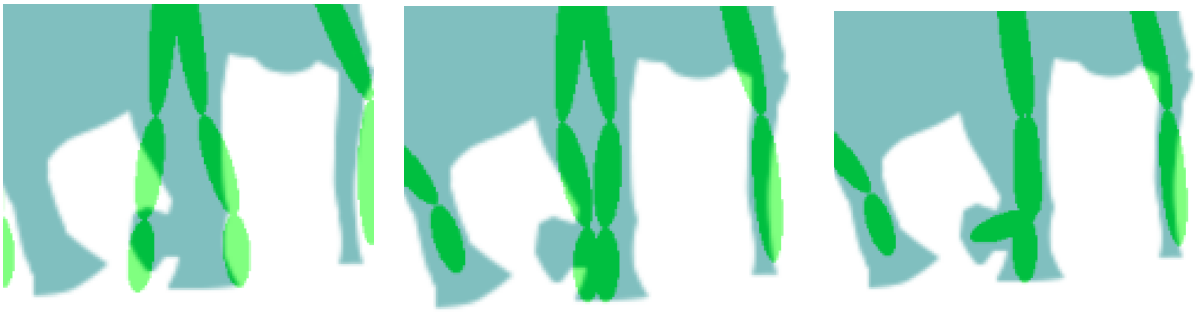
(c) Add improvement on the upper part

Figure 5: Middle Point Algorithm (gif)

While this method gives a reasonably fast and accurate mapping of the skeleton, it also has some issues. The first issue could occur if the skeleton doesn't perfectly fit the shape. Indeed, if the skeleton is bigger than the leg (Figure 6), one of our assumptions no longer holds as the extremity will either be blocked in the "corner" of the leg or will rotate until it reaches the maximum angle.

Figure 6: Skeleton size problem

The other issue concerns the position where the two legs are overlapping each other. As shown in Figure 7, the two skeletal 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 have yet to find a method to deal with 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

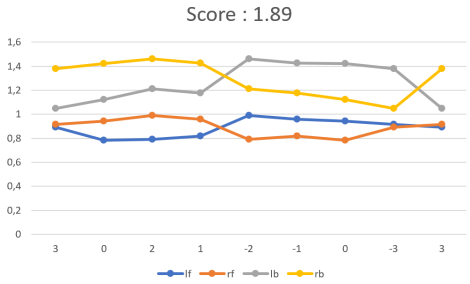
Figure 7: Middle Point Algorithm issue

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 ensure that the inversion of the front and of the back legs happens at the same time, which doesn't really replicate the movement accurately. In fact, most of the time, the back legs cross a bit before the front legs. By doubling the number of positions, there becomes a risk of creating too many similar positions in certain parts of the motion.

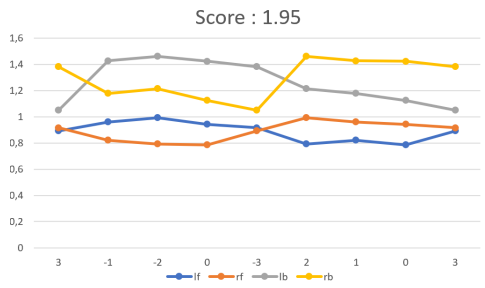
## 2.3 Ordering and animation of the Skeleton

Lastly, we need to find an optimal ordering of the $2n$ positions 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 8). We can then compute a simple score between two positions by taking the L2-norm of the difference of the two vectors. The score measures how different two consecutive positions are. We make use of Dijkstra algorithm to minimize the total score of the ordering. The algorithm starts with two initials positions and at each step, adds the node that minimizes the total score, while also 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 finds 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 leg doesn't have the same motion whether it's moving forward or backward (as, in a real movement, the leg is either moving forward or staying fixed).

Figure 8: Leg angle $\alpha$




(a) Without juxtaposition constraint




(b) With juxtaposition constraint on inverse positions

Figure 9: Optimal ordering using Dijkstra Algorithm

This ordering differs largely from the one used in the Animating Animal Motion from Still paper by Xu et al. [1]. First they ordered animal shapes while in comparison, 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 main difference is the choice of optimization method. Xu et al. use 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, Manon Romain 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 increases, as for only 10 nodes, it would need on average more than 10000 iterations. Dijkstra's algorithm should be much quicker to converge.

Finally, the other main advantage of having different key positions for the skeleton is that we can easily interpolate between each one of them to create a fluid animation. You can see the result in Figure 11 by clicking on the image. The ordering might not be the best one but the motion comes much closer to seeming 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, equivalent to the number of images we add during the interpolation, shouldn't be uniform. We tried to minimize the second derivative in Figure 9 to get similar positions closer to each other, but didn't

search more in that direction.

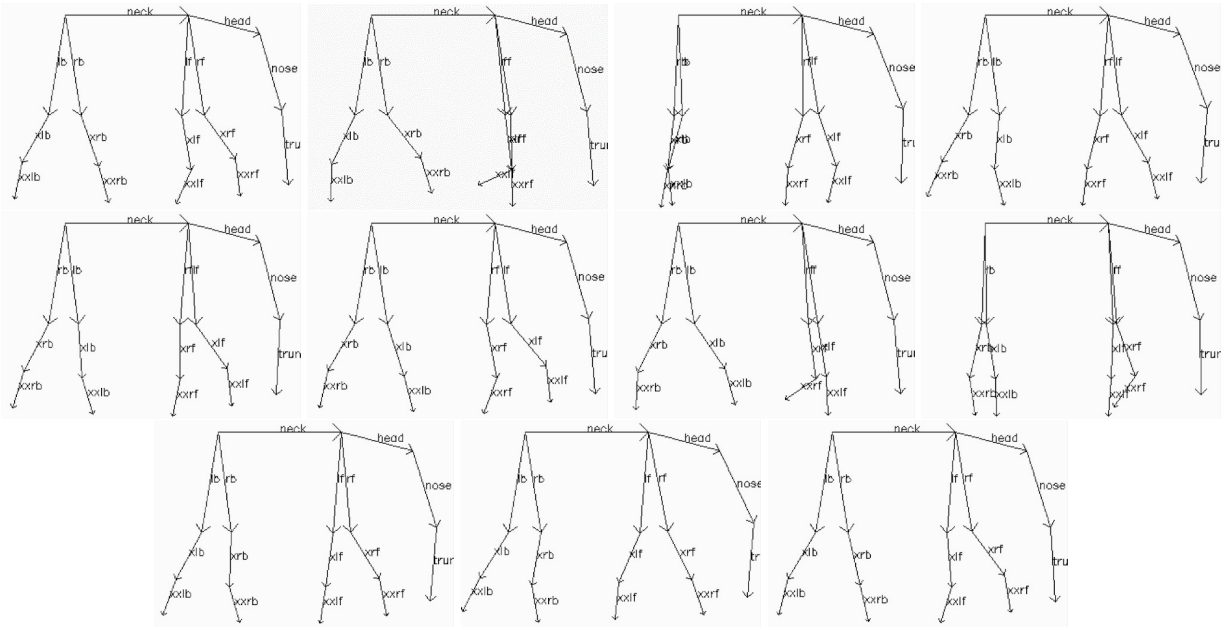



Figure 10: 2D Skeleton Animation

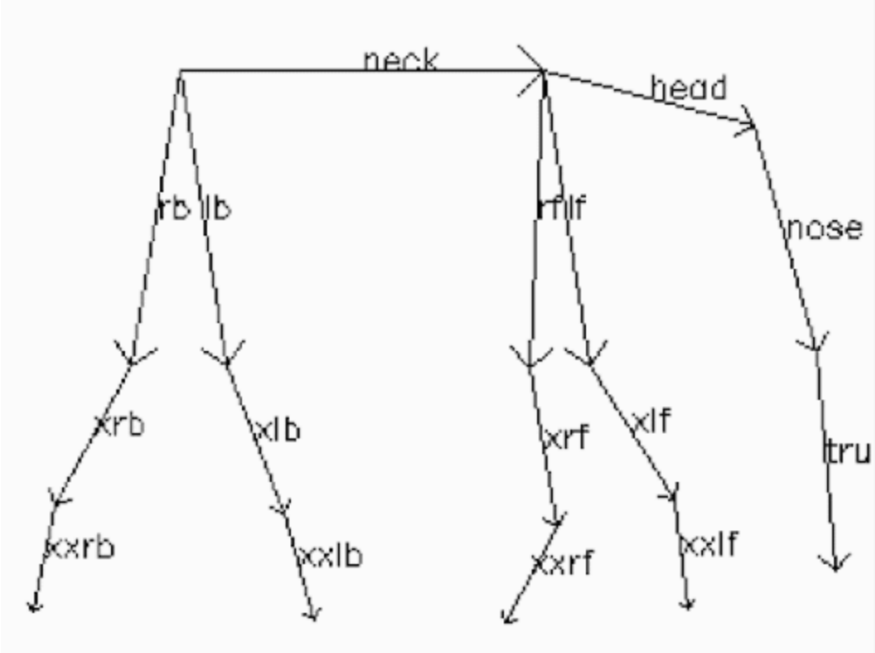


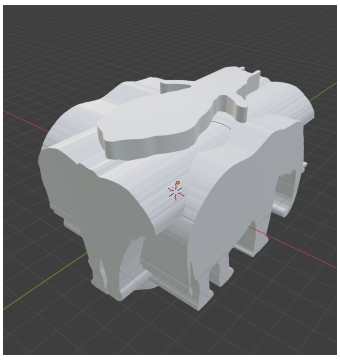

Figure 11: Final 2D Skeleton Animation (gif)

# 3 3D Render

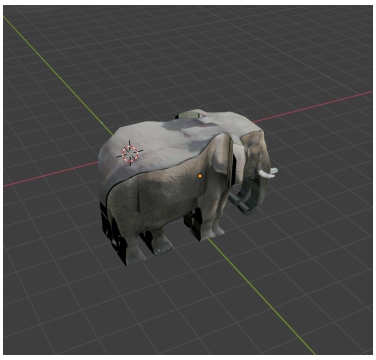
## 3.1 3D Representation

The first step was to choose how to represent the animal in 3D. The two possibilities were to either create a 3D object (Figure 12) or display 2D billboards and change their orientation depending on the position of the camera (Figure 13).

The first method consists of taking 3 different views. We take one view from the side, one from above and one either from the front or the 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 a totally unrealistic representation. This method is also computationally heavier when it comes to rendering. For these reasons, we will focus on using 2D billboards to create our 3d rendering.



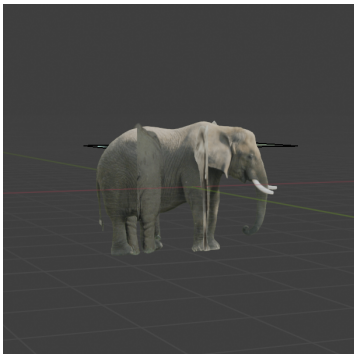
(a) Extrude Faces



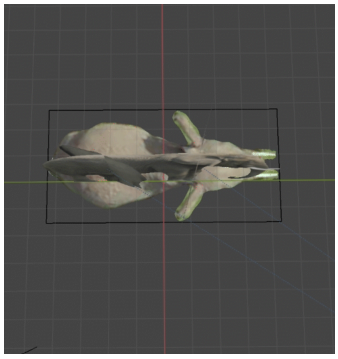
(b) Intersection with texture

Figure 12: 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 displays the part behind the side view image to avoid noticeable intersections.



(a) From side (gif)



(b) From above (gif)

Figure 13: 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 fixed with respect to the horizontal axis, by taking the lowest of the two front feet (Figure 14).

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 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 (Figure 14). We also tried to curve the 2D planes to add volume to the shape (on the back leg, Figure 15). With only the side view image, we could go from about 30 to 150 degrees horizontally and up to 60 degrees vertically (Figure 16). This could perhaps be improved by further curving the images.

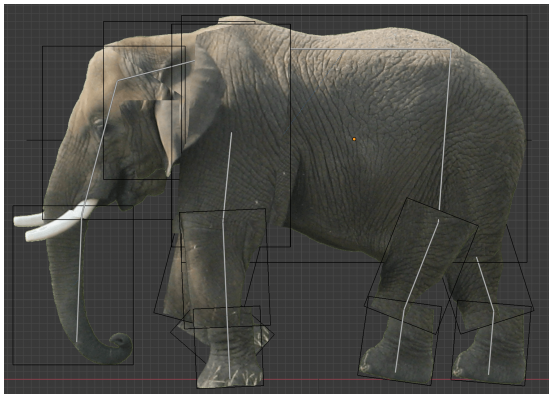
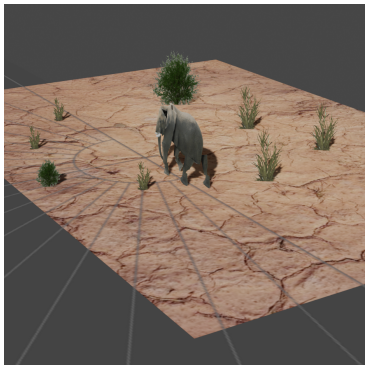


Figure 14: Fix leg animation (gif)

Figure 15: Curved images on the back leg



(a) 30 degrees (gif)

(b) 150 degree (gif)

(c) 60 degrees Vertically

Figure 16: Maximum view angles

Finally we can add the view from above. Unfortunately we didn't find an appropriate method to display it only when it is not in front of the side view, 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. However, once again, the colors are unlikely to match and would cause more complications.

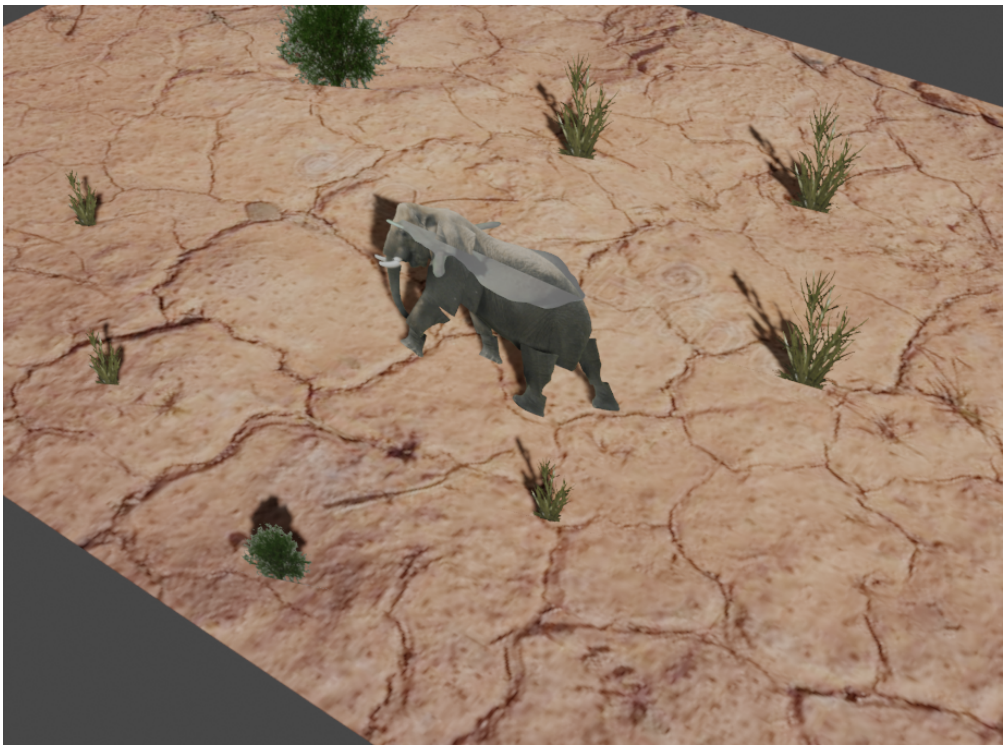


Figure 17: Final 3D Animation (gif)

## 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. However, it would need to be improved in order to be usable and applied without the need of manual corrections. The ordering using Dijkstra's algorithm is quite efficient in cases where there is few initial images but it may not be an optimal choice if there is more data. With regards to 3D rendering, we conclude that the easiest and most efficient way to represent a 3D animal is with the use of billboards. Since the produced image of the animal is supposed to be seen from far away, rather than close up, we pay less attention to some details, for example the junctions between the bones or the flatness of the form. A billboard is thus enough to cover the majority of the camera angles. However, the animation, even from far, remains quite distinguishable and shouldn't be neglected.

## Future work

When looking at the entire process of 2D animation of realistic animals, our work is far from automated. We mainly study images of elephants and our process would have to be tested with a large range of animal in order to confirm that the same methods and algorithm work as efficiently. Furthermore, the manual construction of the skeleton remains quite long and doesn't facilitate the testing on several animals. The middle point algorithm requires being adapted in order to work well with every possible position. Concerning the animation, the only way to significantly improve the motion would be to have an already animated quadruped skeleton and maybe attempt to adapt it to different animal morphology and key positions. An important part of the deconstruction and reconstruction of the side view image could also be more automatic. In addition, it would be worth investigating further methods to add volume to a 2D billboard in order to allow better results when using only the side view image. If we include different view points (from the front and above), we would need to improve the blending and the color matching. Finally, Entem et al, based on sketches, worked on the modeling of 3D animal [8] and the layering of organic shapes [7]. The later might give an easier way to deconstruct the side view image. Although drawing is quite different from real images, it may be worth seeing if some methods could also be applied to our case.

## Acknowledgements

I would like to thank my advisors, Pierre Ecormier-Nocca, Damien Rohmer and Marie-Paule Cani who helped me a lot during my three months intership at the lab and guided me in my work. I would also like to thank the entire GeoVic team at LIX.

## 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] Laurent Favreau, Lionel Reveret, Christine Depraz, Marie-Paule Cani.
*Animal gaits from video: Comparative studies*.

Graphical Models, Elsevier, 2006, 68 (2), pp.212-234. ffinria-00384231f
https://hal.inria.fr/inria-00384231

[6] 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$_{A}rticle@11e3c2463e4_{4}865ce/index.html$

[7] 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

[8] 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

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

# Appendix

**Algorithm 1:** Middle Point Algorithm

**function** `Get_border` $(Bone, Direction, Angle = 5)$**:**

// Base case

**if** $Angle < 1$ **then**

**return** Bone.angle

$Is\_inside \leftarrow (Bone\_extremity \quad is \quad inside) \quad ? \quad 1 \quad : \quad 0$

**while** $(Bone\_etremity == Is\_inside) \ \& \ (min\_angle < Bone.angle < max\_angle)$ **do**

$bone.rotation(Angle, Direction)$

$Change\ Direction$

**return** `Get_border` (***Bone,Direction,Angle/4***)

**function** `Middle_point` $(Bone, Shape)$**:**

$Direction \leftarrow (Score(bone.rotation(5\ degrees, left)) >$
$Score(bone.rotation(5\ degrees, right))) \quad ? \quad left \quad : \quad right$

// Go the direction that increases the score

$First\_angle \leftarrow$`Get_border` $(Bone, Direction)$

// After calling `Get_border()`, the extremity of the bone is located close to the border,

// on the same side as before (i.e if it was inside the leg, it's still inside)

**if** $Bone\_extremity \quad is \quad inside \quad the \quad leg$ **then**

$Change\ Direction$

// Thus, we need to change the direction only if it was inside

$bone \leftarrow bone.rotation(10, Direction)$

// Place the extremity of the bone inside the leg (even if it was already inside)

$Second\_angle \leftarrow$`Get_border` $(Bone, Direction)$

$Bone\_angle \leftarrow average(First\_angle, Second\_angle)$

`Middle_point` $(bone.child, Shape)$

**for** *Each Shape* **do**

**for** *bone* $\in [xrf, xlf, xrb, xrb, xlb, nose]$ **do**

`Middle_point` $(bone, Shape)$