Workshop on Virtual Reality Interaction and Physical Simulation VRIPHYS (2012)
J. Bender, A. Kuijper, D. W. Fellner, and É. Guérin (Editors)

Generic Spine Model with Simple Physics for Life-Like Quadrupeds and Reptiles

Ahmad Abdul Karim[1,2], Alexandre Meyer[1], Thibaut Gaudin[2], Axel Buendia[2,3] and Saida Bouakaz[1]

[1] Université de Lyon, CNRS
[1] Université Lyon 1, LIRIS, UMR5205, F-69622, France
[2] Spir.Ops Artificial Intelligence, Paris, France
[3] CNAM – CEDRIC, 292, rue St Martin, 75003 Paris, France

Figure 1: *Example of our system generated animation: from right to left, a wolf animated using the pseudo-physics and flexible spine model.*

Abstract

We propose a pseudo-physics system and a spine model that can be coupled to generate life-like locomotion animations of quadrupeds and reptiles. The pseudo-physics system uses minimalist particle-based physics and values of the gait pattern to generate the sinusoidal-like ballistic movement of the pelvis observed in nature. While the spine model uses simple geometry-based calculations and 3D Hermite curves to generate a flexible spine model, giving the animated creatures more agility. Our final system is totally controllable by the user in order to generate any desired style.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation

1. Introduction

Animated creatures in virtual worlds like arachnids, quadrupeds, reptiles *etc.* should have the ability to move freely in a convincing and a life-like way, in order to make the virtual experience more immersive. The movement of these creatures is governed essentially by their morphology, as their complex anatomies impose constraints on their way of displacement. They move in different locomotion styles based on different sets of gait patterns. Simulating the actual physics and anatomy constraints of these multi-legged characters is quite complex. But in most computer-based applications like games or multimedia virtual worlds, the most important thing is the plausibility of the generated motion [BHW96] more than the accuracy of the calculations. Even with simplified calculations, produced animations can be realistic and believable in comparison to motions generated using complex physics-based calculations. Plus, they offer a better control over the final animation.

Contribution. We propose a simple pseudo-physics system and a generic spine model that help to animate virtual creatures in a life-like way (Figure 1). The goal of the pseudo-physics system is to generate the sinusoidal-like ballistic movement of the pelvis observed in nature. To do so, we simplify calculations by using particle-based physics equations. Based on the gait pattern and Newton second laws of physics we calculate the force that each foot is exerting on the pelvis on each simulation step. We also add a flexible spine model when simulating quadrupeds and reptiles to

DOI: 10.2312/PE/vriphys/vriphys12/097-106

give them the agility observed in real-life animals. Our spine model is quite generic and can be applied on different morphologies. It uses simple geometry-based calculations and 3D Hermite curves. By coupling the spine model with the pseudo-physics system the final locomotion becomes believable and more realistic. These added components are totally controllable giving the user the ability to generate any needed locomotion style.

2. Related Work

There are many techniques for generating locomotion, many of them target human-like morphologies (bipeds). Multon *et al.* in [MFCD99] and more recently van Welbergen *et al.* in [vWvBE*10] identified two main groups: Data-Driven techniques and Procedural-Based techniques with two subcategories physics-based and kinematic-based. Skrba *et al.* in [SRH*08, SRH*09] show in their survey that same techniques are used for quadrupeds animation: Data-Driven, physics-based models, IK systems or some combination of the above.

A wide variety of input data are used to generate multi-legged characters animation. For instance, early work of McKenna *et al.* in [MZ90] uses gait patterns observed in biology and oscillatory-based dynamics to animate a cockroach model that adapts to planar and uneven terrain. More recently, Favreau *et al.* in [FRDC04] extract 3D cyclic motion of animals from video sequences using image processing techniques. Kry *et al.* in [KRFC09] animate a dog model by making a subset of its joints vibrates with specific periods and with low frequencies, which induces passive movement in other connected joints and rigid bodies. In [CKJ*11], Coros *et al.* use gait patterns and dynamic values (forces/torques) extracted from MoCap, to animate a dog capable of a wide range of locomotion on a straight line. These systems are, most of the time, morphology specific meaning that they can animate only a specific morphology like a dog in [CKJ*11], a horse in [TCHL12] or a quadruped robot (Called BigDog by Boston Dynamics ™) in [RBNP08].

We are more interested in morphology independent systems like the *PODA* system proposed by Girard *et al.* in [GM85, Gir87] as they can animate a multitude of morphologies with nearly no constraints. *PODA* is one of the earliest procedural-centric locomotion controllers. They use it to animate a wide range of multi-legged characters on planar terrain (bipeds, quadrupeds, *etc.*). The user only needs to specify the gait pattern (an example of a gait pattern is shown in Figure 4), everything else is calculated automatically. They add pseudo-physics calculations to the pelvis of the multi-legged character, making it reacts to the feet movement in the horizontal and sagittal plane in a more believable way. But in their system the user needs to fix and tweak by hand the force of each foot in order to achieve the needed effect, while in our system (Section 4) we aim for a transparent control for the user with no need for any extra settings.

Another interesting system is the one used in the game Spore ™ (by Maxis Studio ™). Where Hecker *et al.* in [HRE*08] created a system capable of animating multi-legged characters whose morphologies are unknown when creating the actual animation system. These characters can be created by the user in run-time. Their animation database contains semantically generalized keyframe data specialized in real-time based on the actual new morphology.

Finally, most of the systems discuss the importance of adding a spine-like model while animating multi-legged characters (specially quadrupeds) in order to generate more natural results [CKJ*11, CR06]. An important issue that we address in more details in Section 5.

3. Locomotion Controller

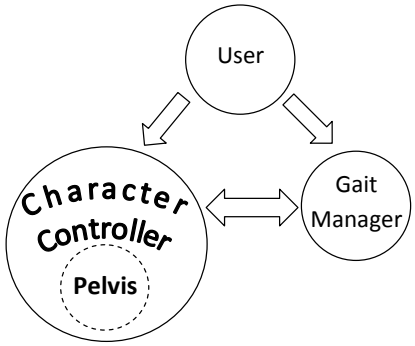

Figure 2: *Locomotion controller overview.*

Locomotion is the act of moving from one place to another. For most terrestrial animals that means putting one foot in front of the others in a successive way until reaching the designated point of interest (target). During a normal foot movement there are two main phases *Stance* and *Swing* phase [INM66]. During the *stance* phase a foot is blocked on the ground. While in *swing* phase (flight phase) the foot flies in a parabolic-like curve toward its target without any ground contact.

Our system follows these principles when generating locomotion. It is kinematic-based and inspired by the locomotion controllers found in [GM85, Gir87, AGM*12, aCT12]. The overall locomotion process is computed by two main blocks as shown in Figure 2:

- The character controller is the central main structure that manages the overall locomotion.
- The gait manager regulates the feet tempo according to the movement patterns defined by the user.

The character controller is in charge of two main tasks: managing the movement of the feet and computing the pelvis 3D movement. Concerning the movement of the feet, at each time step, the gait manager informs the character controller about the feet that are going to enter in *swing* phase. For each one of these feet, the character controller calculates a parabolic/ballistic based trajectory toward a footprint target calculated based on the current creature speed and orientation (Figure 3). This foot 3D trajectory can be more complex and incorporates environment and obstacle avoidance like in [AGM*12]. For the *stance* feet, the character controller simply blocks their 3D position on the ground.

The 2D movement of the pelvis on the ZX-plane (assuming the Y-axis is up) is calculated using only the speed and orientation. The computation of the pelvis height is more complex as will be explained in Section 4. By fixing the pelvis height based on the user preferred height only, the character controller produces an animation where the pelvis floats in un-natural way (see Figure 3).

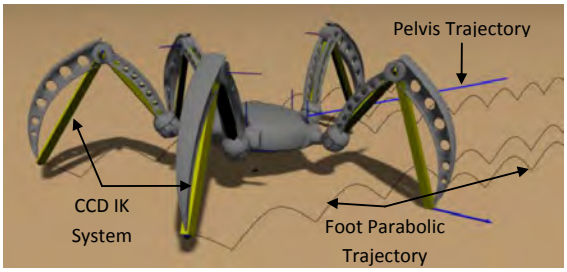

Figure 3: *The locomotion generated for a 5-legged robot. Notice the un-natural floating-like pelvis trajectory.*

The gait manager organizes and visualizes the pattern of the feet's cycle. Since locomotion is cyclic, it seemed natural to represent the gait with circles. As illustrated in Figure 4, each circle represents a foot, with the colored sectors representing the *swing* phase portion of the foot movement. The feet needle activates sectors and deactivates others based on its current position, while turning clockwise. Activating a sector means that the corresponding foot should enter its *swing* phase.

The final animation is generated based on the input parameters that the user provides and controls in real-time.

- **Gait/Tempo**: using our interface, the user designs each foot cycle (*stance* and *swing* phases). These cycles describe the tempo of the feet movement. The final gait can be symmetrical or asymmetrical.
- **Locomotion speed**: speed of the movement in *meters per second*.
- **Locomotion direction**: the needed orientation on the ZX-plane.

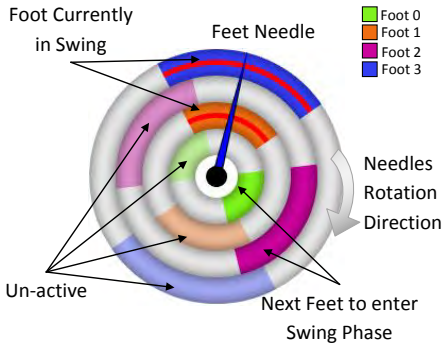

Figure 4: *An example of a four feet gait as shown by the gait manager. With this interface, the user can edit the pattern creating the needed gait.*

- **Preferred height**: the preferred height of the creature pelvis in *meter*.
- **Joints limits**: the leg and spine joints limits in *radian*.

We use the Cyclic Coordinate Descent (CCD) method proposed in [Lue84, WC91] to calculate the position and orientation of the intermediate leg joints (see Figure 3). The CCD iterates through the joints, typically starting with the one closest to the *end-effector*, and varies one joint variable at a time based on a heuristic. Unlike the **Jacobian Inverse** method, which distributes joint rotation changes equally along the chain, CCD has a preference of moving distal links first. We chose this CCD IK system thanks to its simplicity and the ability to integrate joint constraints easily.

4. Pelvis Movement Using Pseudo Physics

The previous character controller moves the pelvis based on the user needs: speed, orientation and preferred height. But by only doing so, the pelvis floats above the ground in an unnatural way, as shown in Figure 5(a).

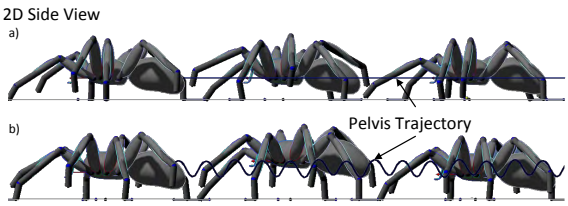

Figure 5: *Possibilities of pelvis movement for a spider model. a) A fixed pelvis movement producing a straight line. b) More realistic sinusoidal-like ballistic pelvis movement produced implicitly by the feet gait pattern. Frequency of oscillations is exaggerated on purpose in (b) to illustrate the controllability of our system.*

On the other hand, many biomechanics based studies and observations show that the pelvis trajectory in humans [INM66] (see Figure 6) and in most animals [Muy57] (see Figure 7) is sinusoidal-like. The amplitude and frequency of this movement vary based on several criteria's, like the creature morphology, the overall speed, the gait pattern, the style and so on. The goal of this section is to **implicitly** generate this sinusoidal-like ballistic pelvis movement observed in nature, as shown in Figure 5(b).

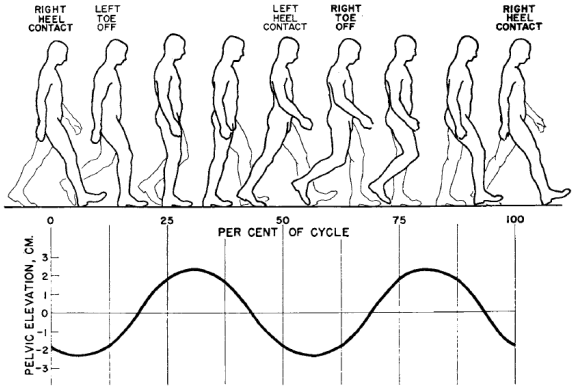

Figure 6: *The pelvis trajectory of a walking human, courtesy of [INM66].*

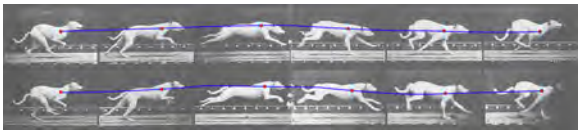

Figure 7: *The pelvis trajectory of a galloping dog, original image courtesy of [Muy57].*

We generate this movement by applying on the pelvis a pseudo particle-based physics. We have deliberately chosen this simplified approach for ease of implementation, performance and controllability. We must emphasize that more sophisticated physical approaches like in [CKJ*11] provide realistic results but with many control restrictions. This particle motion on the sagittal plane (the up Y-axis) is governed by the gravity force (pushing downward) and the feet force (pushing upward), shown in Figure 8. While in the horizontal and coronal plane (ZX-plane) the pelvis particle movement is governed by the character controller commands.

For the purpose of clarity, let us study the case of having only one foot ($n = 1$). In this case, the foot and the pelvis particle can be seen as a pogo stick†, shown in Figure 8, with the foot (leg) supporting the whole mass m of the pelvis alone. This foot pushes the pelvis particle upward with certain amount of force when the pogo stick spring is compressed. We will later discuss the case of a multi-legged character.

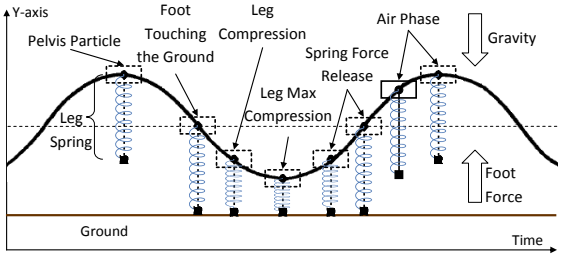

Figure 8: *The trajectory of the pelvis particle when a pogo stick is used to represent its relationship with one leg.*

To calculate the force that this foot is exerting on the pelvis particle, we use the gait pattern set by the user as follows. Each foot has two phases: *stance* and *swing* phase. In *swing* phase the foot cannot participate in pushing the pelvis, as it does not have any contact with the ground. While in *stance* phase it can push the pelvis upward. We decompose this *stance* phase into two other distinctive phases, as shown in Figure 9: *reception* phase where the foot decelerates the downward movement of the pelvis to a stop, *propulsion* phase where the foot starts pushing the pelvis upward in order to prepare for the *swing* phase. The duration of the *reception* phase is equal to the duration of the *propulsion* phase and it is half of the *stance* phase duration ($T_{reception} = T_{propulsion} = \frac{1}{2}T_{stance}$), a choice we made based on biomechanics observations [Ale96, Ale03, Muy57].

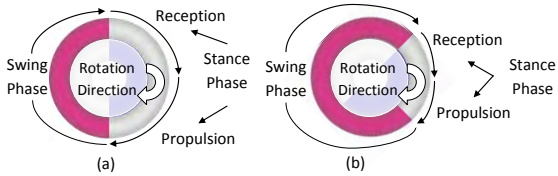

Figure 9: *Reception and propulsion phases during the foot stance phase. In (a) swing phase has the same duration of the stance phase while in (b) swing phase is longer.*

† A pogo stick is a device for jumping off the ground in a standing position with the aid of a spring, used as a toy or exercise equipment. It consists of a pole with a handle at the top and footrests near the bottom, and a spring located somewhere along the pole.

For now we are studying the case of having only one foot supporting the whole mass m of the pelvis. We use Newton second law's of motion to calculate the force which this foot is going to apply on the pelvis particle. We consider only the Y axis in our computations.

$$m \cdot a_{pelvis} = W + F_{foot} \tag{1}$$

a_{pelvis} is the pelvis acceleration, W is the weight force and F_{foot} is the foot pushing force, all on the Y axis.

$$m \cdot a_{pelvis} = -m \cdot g + m \cdot a_{foot} \tag{2a}$$

$$a_{pelvis} = -g + a_{foot} \tag{2b}$$

g is the gravity acceleration (which is negative) and a_{foot} is the foot acceleration on the Y axis. We now calculate this foot acceleration (a_{foot}) based on the gait pattern. This acceleration represents its actual upward force. We know that:

$$a_{pelvis} = \frac{v_T - v_C}{T} \tag{3}$$

With T a duration, v_T is the needed velocity to achieve at the end of that duration (T) and v_C is the current velocity. By replacing a_{pelvis} by its value from equation 2b, equation 3 becomes:

$$-g + a_{foot} = \frac{v_T - v_C}{T} \tag{4a}$$

$$a_{foot} = g + \frac{v_T - v_C}{T} \tag{4b}$$

In *reception* phase, the foot goal is to decelerate the pelvis into a stall ($v_{pelvis} = 0$). The beginning of this phase is the end of a *swing* phase, which means that the pelvis will be falling down under the gravity force. So the goal of this phase is to achieve $v_T = v_{pelvis} = 0$ with T is the *reception* phase duration. By replacing the values in equation 4b we can easily calculate the needed acceleration a_{foot} of this foot, which represents this foot pushing upward force (Figure 10).

In *propulsion* phase, the foot goal is to achieve a v_{pelvis} at the end of the *stance* phase, that ensures the ballistic movement of the pelvis during this foot *swing* phase (Figure 10). To do so, we use the following basic motion equation:

$$y_t = \frac{1}{2}at^2 + v_0 t + y_0 \tag{5a}$$

$$a = -g \tag{5b}$$

$$v_0 = \frac{y_T - y_0}{T} - \frac{1}{2}gT \tag{5c}$$

$v_0 = v_{pelvis}$ is the needed velocity (Figure 10), T is the (*propulsion* + *swing*) phase duration, y_0 is the current height and y_T is the preferred height fixed by the user. By replacing the values in equation 4b we calculate the needed acceleration a_{foot} of this foot. These calculations are done on each simulation step, giving us the punctual force (acceleration) that this foot is going to exert on the pelvis particle on each simulation step. Again, during *swing* phase the foot acceleration is null ($a_{foot} = 0$).

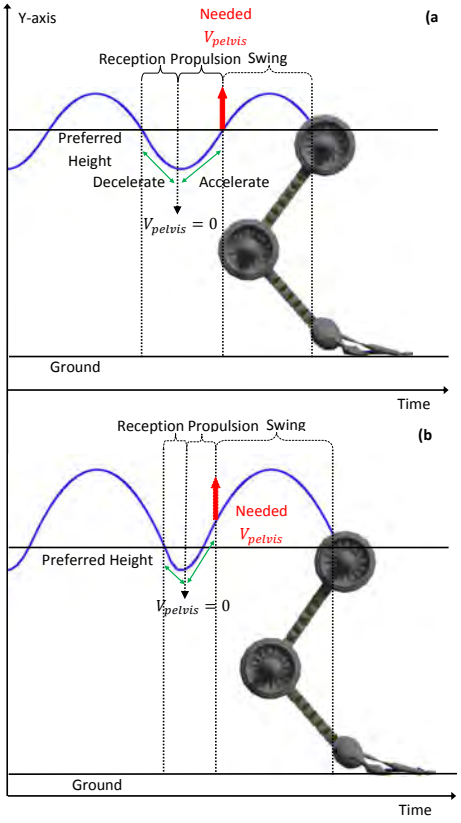

Figure 10: *Sinusoidal-like ballistic pelvis movement generated implicitly using the foot force calculated based on the reception and propulsion phases. Preferred height is the one fixed by the user. a) Trajectory is generated using the Gait Pattern (a) from Figure 9(a). b) Trajectory is generated using the Gait Pattern (b) from Figure 9(b).*

In the case of a multi-legged character $n > 1$, meaning that we have several feet interacting with the pelvis particle. Let a_i be the acceleration of the foot i calculated using the previous equations, which was denoted a_{foot} previously. This acceleration is calculated for each foot using the postulate that it is alone and using its gait pattern. So the pelvis particle needs to integrate all forces (a_i) of each foot to calculate its final a_{pelvis}. In our system we use the postulate that $m = n \times m_i$ where n is the number of feet and m_i is the share of mass that each foot supports. As if the feet share equally the mass of the pelvis particle. Thus, $a_{pelvis} = \sum_{i=1}^{n} a_i/n$ and the resulting sinusoidal-like ballistic movement of the pelvis is shown in Figure 11. A weighted average can also be used in order to give more importance to legs with more masses (heavy feet), back legs or any other user needs.

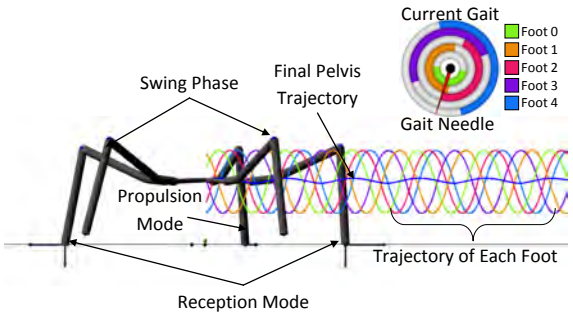

Figure 11: *The trajectory per foot and the final pelvis sinusoidal-like ballistic movement.*

If the gait pattern is symmetrical as in Figure 11 (*swing* phase duration is equal to the *stance* phase duration) the resulting pelvis movement is a sinusoid. But with other gait patterns, the pelvis movement can be totally different as phases duration can be different. Like a gait pattern with $\frac{2}{3}$ *swing* phase and $\frac{1}{3}$ *stance* phase, illustrated in Figure 9(b) with the resulting trajectory in Figure 10(b). The user has a total control over this movement through the gait pattern with no need to set any extra values. On top of that, these pseudo physics calculations are used to validate the plausibility of the gait pattern designed by the user. As with a badly designed gait pattern the forces calculated will be too large and not natural. We must note that any change in the character speed can occur only during the *propulsion* phase, as it is the only phase where a foot is virtually propelling the multi-legged character forward.

5. Spine Model

Quadruped animals like mammals (dog, horse, wolf, *etc.*) and reptiles (crocodile, lizard, gecko, *etc.*) have a flexible spine, which is an essential component for these type of animals during locomotion and other types of movements [Ale96, Ale03, Muy57]. They use the flexibility of this structure into their advantage (see Figure 12) to achieve a high variety of locomotion styles, as it gives them more agility and more Degrees of Freedom (DOF).

A variety of kinematic quadruped systems [SRH*08, SRH*09] implement a flexible spine in order to achieve more realism in the final generated animation. In [CKJ*11], Coros *et al.* show how the produced motion of their quadruped (a dog) loses its plausibility and naturalness when a rigid spine is used. By adding a flexible spine model, the animated wolf illustrated in Figure 13 looks more natural as it turns in a more realistic way.

To add this flexible spine model, we decompose the multi-legged character morphology into several virtual pelvis nodes (shoulders), seen in red in Figure 12 and in Figure 14.

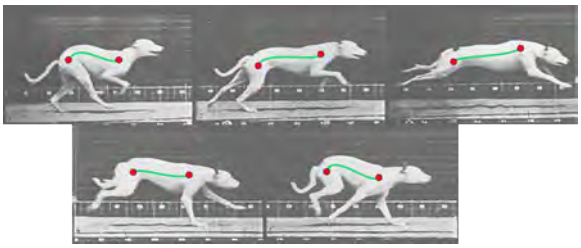

Figure 12: *The deformation of the dog flexible spine structure (in green) while galloping, original image courtesy of [Muy57].*

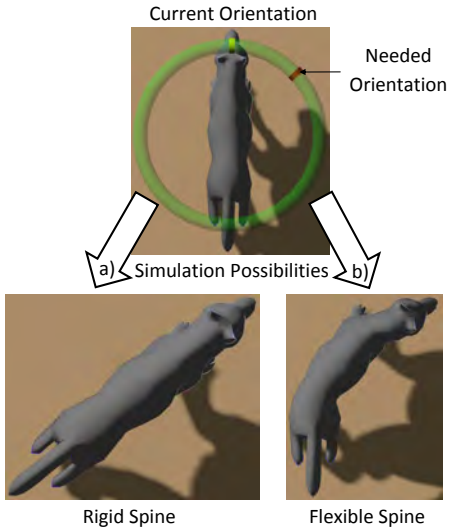

Figure 13: *The visual difference of adding a spine model when executing a change in orientation command. a) Without a spine, the wolf model turns in a rigid way. b) With a flexible spine, the wolf model turns in a more natural way.*

Each foot is connected to one of these nodes, except for the head node which has no foot connected to it. These virtual pelvis nodes (pelvis nodes for shortening) are quite independent in regard to height control, pitch control, footprint placement, *etc.* and are connected by the flexible spine model.

We calculate this spine model using four successive steps, described in Figure 15). Firstly, on the horizontal and coronal plane (ZX-plane) then on the sagittal plane (Y-axis) for simplification. On the ZX-plane we concentrate on the 2D orientation (Section 5.1) and translation (Section 5.2), while on the sagittal plane we concentrate on the elevation and pitch control (Section 5.3). In Section 5.4, we put everything together to calculate the final 3D spine model.

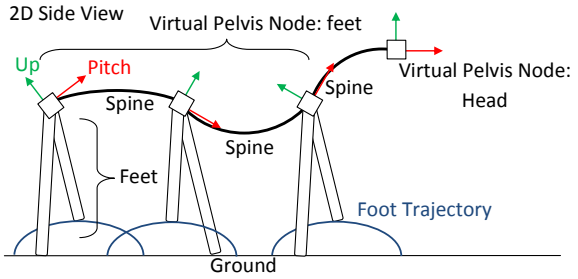

Figure 14: *The pelvis virtual nodes extracted from the morphology of a fictional 6-legged creature.*

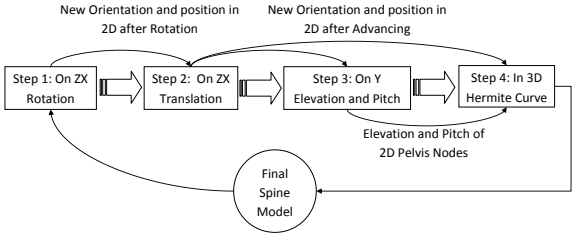

Figure 15: *A workflow explaining the different steps used to calculate the flexible spine model.*

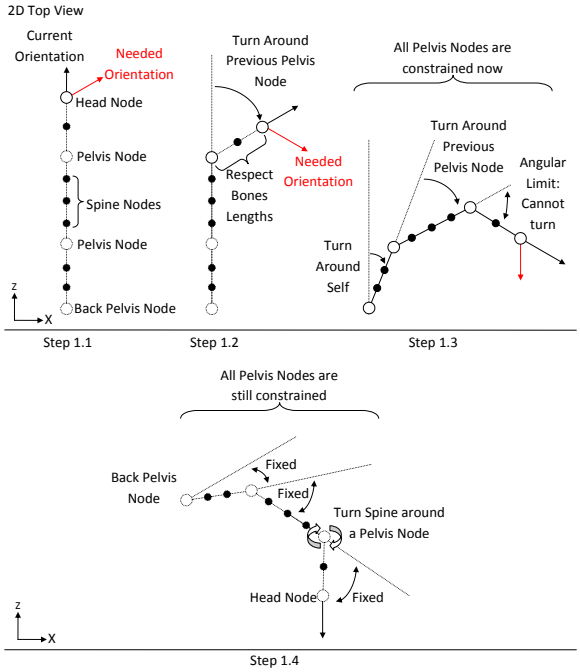

Figure 16: *A step by step illustration that shows how the change of spine orientation is computed on the ZX-plane.*

5.1. Step 1: Spine Orientation

In Figure 16, we explain how the rotation (orientation change) is achieved. The spine nodes are nodes between the virtual pelvis nodes and part of the spine model with no foot attached to them. An exception is the head node which is considered as a virtual pelvis node. We need to calculate the final position and orientation of the spine nodes as they are part of the original multi-legged character spine morphology.

When a new orientation is needed (on the *ZX*-plane), we propagate this needed orientation on the pelvis nodes from the head node toward the back pelvis node. Each pelvis node will try to satisfy its share based on its relative angular limits, sending the unsatisfied rest in the propagation direction (the joints limits are fixed by the user). When all pelvis nodes are constrained, as in Figure 16 - Step 1.3, we rotate the whole spine around one of the pelvis nodes in order to satisfy the needed orientation. In our animation system we always choose the pelvis node just before the head node, a preference that we observed in real-life animal videos. In all previous steps, bones length is always satisfied. So after calculating the position of the pelvis nodes, we place the spine nodes between them based on this bones length constraint.

We must note that a pelvis node can be constrained by joints angular limits and by the feet attached to it. For example, a pelvis node with a fully extended foot in *stance* phase cannot move without breaking the leg bone length limit. These kind of constraints are processed in the final step of our method (Section 5.4).

5.2. Step 2: Spine Advancing

In Figure 17, we explain how we translate the spine model. Based on the current orientation and the needed distance, a new head node position is calculated (the target head node in red). The needed distance is calculated using the needed pelvis speed fixed by the user. To maintain a coherent movement of the spine model, we place the virtual pelvis nodes (in blue) on the previous step spine model (in black) in a way that respects the bones length constraint. In this way, the last step spine model is considered as a support model that helps in maintaining the fluidity of the movement. In Figure 17(b), we re-calculate the relative orientation between pelvis nodes based on the new positions and the support model. For the back pelvis node, we change the relative orientation in way that relaxes the constraints (in green) in a temporal way. The duration of this relaxation is based on empirical data. At the end of this step, we obtain the position and orientation of the spine model on the *ZX*-plane and 3D calculations begin. In Figure 22 and in the accompanying video we show the animated creatures reacting to external pushes. To achieve that, we integrate these pushes in this step by varying the 2D position of a pelvis node based on the external push.

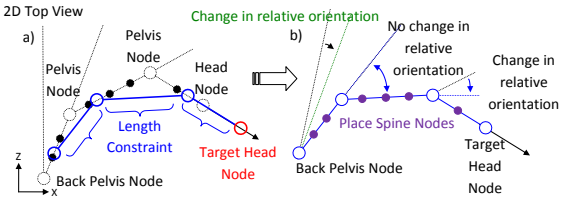

Figure 17: *Illustration on how we translate the pelvis and spine nodes, on the ZX-plane, based on the orientation and needed distance.*

5.3. Step 3: Spine Elevation and Pitch

As we decomposed the character pelvis into several virtual nodes, we decompose the height and pitch calculations into its respective nodes, as shown in Figure 18. We integrate the pseudo physics system in each virtual pelvis nodes to add realism to the spine model. By doing so, the final needed pelvis node elevation is calculated using the pseudo particle-based physics instead of the preferred height only. We must emphasize that the final visual position of each pelvis node is independent from the height calculated by the pseudo particle-based physics system. This is explained in the final step.

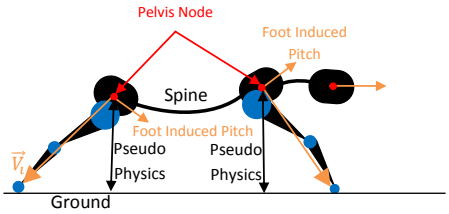

Figure 18: *Computation of each node's height using the pseudo-physics system and computation of each virtual pelvis node pitch.*

For the pitch angle of each pelvis node, we calculate it using the relative position of each foot. Let $\vec{V_i}$ be the vector that connects the current pelvis node with the foot i (in orange in Figure 18). We calculate the perpendicular vector on $\vec{V_i}$ in the direction of the creature progression. This perpendicular vector represents how much this foot affects the pitch angle of its pelvis node. We call this perpendicular vector the foot induced pitch, and the final angle of the pelvis node is an average of all the feet induced pitches.

5.4. Step 4: Final 3D Spine

Using the 2D positions calculated in **Step 1-2** (Section 5.1 & 5.2) and elevations calculated in **Step 3** (Section 5.3), we obtain a preliminarily 3D position for each virtual pelvis node. And using the 2D orientations calculated in **Step 1-2** and pitches calculated in **Step 3**, we obtain a preliminarily tangent direction for each of them. We construct a B-Spline (Hermite) curve between these 3D positions using the previous tangents data. We sample this curve using the bones length constraint in order to calculate the final pelvis and spine nodes position and 3D orientation. By doing so, the visual representation of each pelvis node can have a different 3D position from the one calculated in the previous steps. We consider the 3D positions calculated until **Step 3** as guidelines for the Hermite curve, making the pseudo particle-based physics system independent from the visual system. Sometimes, the final position of the virtual pelvis nodes can not be satisfied by the CCD IK system because of joints constraints. In this case, **Step 4** is repeated based on the closer position that the IK system can ensure from the needed one. In Figure 19, we show the final generated flexible spine model on an abstract lizard model. Its spine consists of 3 pelvis nodes and 9 spine nodes.

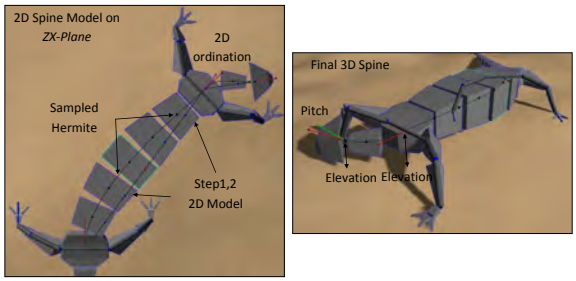

Figure 19: *Final generated flexible spine model in an abstract lizard model.*

6. Results and Conclusion

In the accompanying video we show the capabilities of our system in animating multi-legged characters in life-like way using the components presented in this article. Final simulation is real-time (30fps) even with several creatures at the same time.

In Figures 20, we show a frame by frame snapshot of a wolf running with a gallop like gait. With the integration of the pseudo physics and the flexible spine, the wolf moves in a quite natural and life-like way, relatively similar to Figure 7 and 12.

In Figures 21, we show a frame by frame snapshot of a lizard running upward. By adding a visual yaw effect to the spine model tangents, the animated lizard moves in a believable way compared to a real life lizard. We calculate the yaw value for each foot based on its current position (in white in Figures 21) and its rest position (in green). The value of the final visual yaw effect is the average yaw for all feet.

In Figures 22, we show a frame by frame snapshot of a wolf reacting to an external push. The presented spine model is quite generic and capable of integrating external forces in its own movement. These push forces are integrated in **Step 3** (Section 5.3).

We presented two totally controllable components that can be added easily to any locomotion system in order to generate believable and life-like locomotion animations for virtual creatures. The presented pseudo-physics system and spine model use the minimum calculation possible in order to generate the needed effects, without the need for complex physics or anatomic based simulations. We do not implement any balance strategies when the system reacts to external pushes. Plus, there can be some leg intersection as our character controller generates the parabolic trajectory of the feet with no special avoidance treatment. Adding these balance strategies, using more morphology specific IK system and doing 3D planning for feet trajectory to avoid leg collision are our main focus in future works, as they add more believability to the animated creatures.

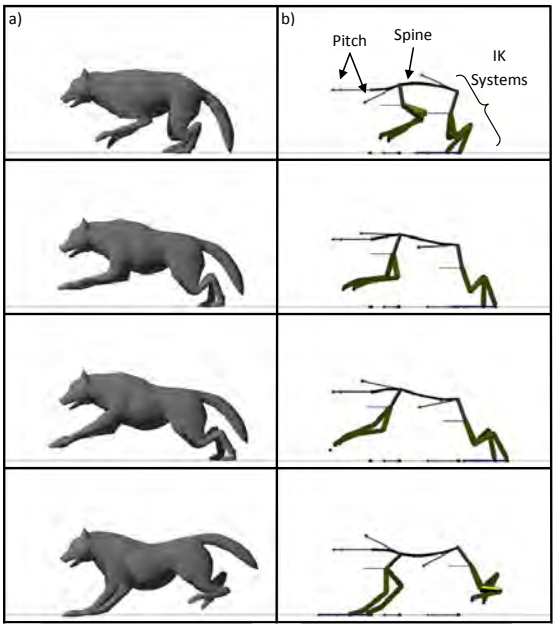

Figure 20: *A wolf running to the left with its spine model being deformed based on the pseudo physics and the pitch control. a) 3D mesh. b) spine model and IK systems*

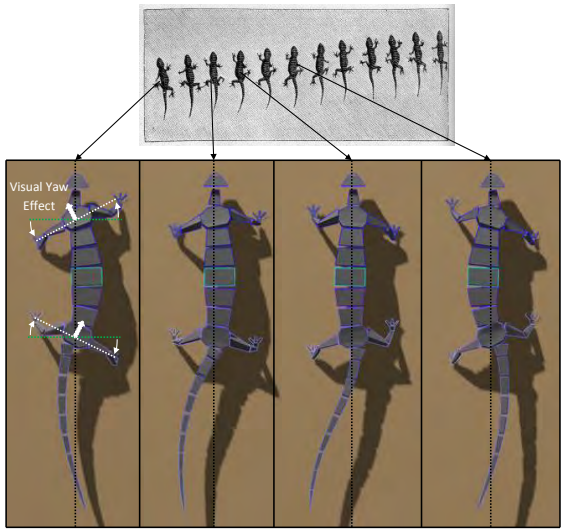

Figure 21: *A lizard running upward in a believable way compared to a real life lizard, the top image is courtesy of [Mar93]. The visual yaw effect is calculated using the current position of the feet on the ZX-Plane.*

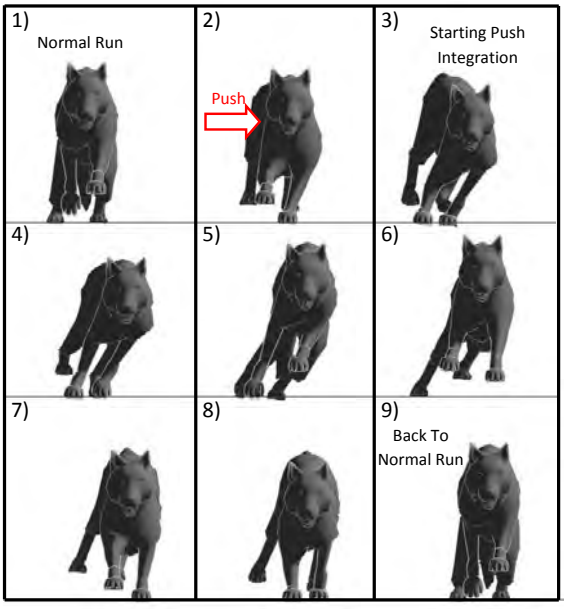

Figure 22: *A wolf reacting to an external push on the shoulder level.*

References

[aCT12] AP CENYDD L., TEAHAN B.: An embodied approach to arthropod animation. *Journal of Computer Animation and Virtual Worlds* (July 2012). 2

[AGM*12] ABDUL KARIM A., GAUDIN T., MEYER A., BUENDIA A., BOUAKAZ S.: Procedural Locomotion of Multi-Legged Characters in Dynamic Environments. *Journal of Computer Animation and Virtual Worlds* (July 2012). 2, 3

[Ale96] ALEXANDER R.: *Optima for animals*. Princeton paperbacks. Princeton University Press, 1996. 4, 6

[Ale03] ALEXANDER R.: *Principles of animal locomotion*. Princeton University Press, 2003. 4, 6

[BHW96] BARZEL R., HUGHES J. F., WOOD D. N.: Plausible motion simulation for computer graphics animation. In *Proceedings of the Eurographics workshop on Computer animation and simulation '96* (New York, NY, USA, 1996), Springer-Verlag New York, Inc., pp. 183–197. 1

[CKJ*11] COROS S., KARPATHY A., JONES B., REVERET L., VAN DE PANNE M.: Locomotion skills for simulated quadrupeds. In *ACM SIGGRAPH 2011 papers* (New York, NY, USA, 2011), SIGGRAPH '11, ACM, pp. 59:1–59:12. 2, 4, 6

[CR06] CLAUZEL G., REVÉRET L.: *Animation 3D En Temps-Réel De Quadrupèdes Par Simulation Physique Rapport*. Master's thesis, INRIA Rhône-Alpes, 2006. 2

[FRDC04] FAVREAU L., REVERET L., DEPRAZ C., CANI M.-P.: Animal gaits from video. In *ACM SIGGRAPH / Eurographics Symposium on Computer Animation, SCA* (Grenoble, France, 2004). 2

[Gir87] GIRARD M.: Interactive design of 3-d computer-animated legged animal motion. In *Proceedings of the 1986 workshop on Interactive 3D graphics* (New York, NY, USA, 1987), I3D '86, ACM, pp. 131–150. 2

[GM85] GIRARD M., MACIEJEWSKI A. A.: Computational modeling for the computer animation of legged figures. In *Proceedings of the 12th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1985), SIGGRAPH '85, ACM, pp. 263–270. 2

[HRE*08] HECKER C., RAABE B., ENSLOW R. W., DEWEESE J., MAYNARD J., VAN PROOIJEN K.: Real-time motion retargeting to highly varied user-created morphologies. *ACM Trans. Graph. 27* (August 2008), 27:1–27:11. 2

[INM66] INMAN V. T.: Human locomotion. *Canadian Medical Association* (1966). 2, 4

[KRFC09] KRY P., REVÉRET L., FAURE F., CANI M.-P.: Modal locomotion: animating virtual characters with natural vibrations. *Comput. Graph. Forum 28*, 2 (2009), 289–298. Special Issue: Eurographics 2009. 2

[Lue84] LUENBERGER D. G.: *Linear and Nonlinear Programming*. Addison-Wesley, 1984. 3

[Mar93] MAREY E.-J.: Locomotion comparée chez les différents animaux, nouvelles applications de la chronophotographie. *La Nature* (1893), 215–218. 9

[MFCD99] MULTON F., FRANCE L., CANI M.-P., DEBUNNE G.: Computer animation of human walking: a survey. *Journal of Visualization and Computer Animation (JVCA) 10* (1999), 39–54. Published under the name Marie-Paule Cani-Gascuel. 2

[Muy57] MUYBRIDGE E.: *Animals in Motion*. Dover Publications, Inc., 1957. 4, 6

[MZ90] MCKENNA M., ZELTZER D.: Dynamic simulation of autonomous legged locomotion. In *Proceedings of the 17th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1990), SIGGRAPH '90, ACM, pp. 29–38. 2

[RBNP08] RAIBERT M., BLANKESPOOR K., NELSON G., PLAYTER R.: Bigdog, the rough-terrain quadruped robot. *The 17th World Congress The International Federation of Automatic Control* (2008). 2

[SRH*08] SKRBA L., REVERET L., HÉTROY F., CANI M.-P., O'SULLIVAN C.: Quadruped animation. In *Eurographics State-of-the-Art Report* (Hersonissos, Creete, Greece, 2008), pp. 1–17. 2, 6

[SRH*09] SKRBA L., REVERET L., HÉTROY F., CANI M.-P., O'SULLIVAN C.: Animating Quadrupeds: Methods and Applications. *Computer Graphics Forum 28* (2009). 2, 6

[TCHL12] TING-CHIEH HUANG Y.-J. H., LIN W.-C.: Real-time Horse Gait Synthesis. *Journal of Computer Animation and Virtual Worlds* (July 2012). 2

[vWvBE*10] VAN WELBERGEN H., VAN BASTEN B. J. H., EGGES A., RUTTKAY Z., OVERMARS M. H.: Real time animation of virtual humans: A trade-off between naturalness and control. *Computer Graphics Forum, Eurographics 2010 29* (2010). 2

[WC91] WANG L. C. T., CHEN C. C.: A combined optimization method for solving the inverse kinematics problems of mechanical manipulators. *Robotics and Automation, IEEE Transactions on 7*, 4 (1991), 489–499. 3