Simple Quadruped Simulation

Suwen Wang

CPSC526 Project, 2010
University of British Columbia

Abstract

In this project, I aim to design and experiment with a controller for quadruped walking gait in a physics based simulation. A simplified and articulated dog model was designed with realistic dimensions. The controller is based on a finite state machine, where each state represents a target pose in the gait cycle. A simple balancing strategy was implemented.

1 Introduction

Physics based simulation has been gradually taking on a more important role in articulated character motion synthesis. This approach can ensure the generated motion to be physically realistic, and also provide relatively easy control over the motion by parameter tuning. While there are a lot of research for biped locomotion, there are less emphasis on simulating quadruped motion.

Comparing to biped locomotion, quadruped model has its unique complexities. For the four legs, the sequences of alternating swinging and striking need to both keep balance and propel the torso forwards. This requires a more complex pose control finite state machine. Coordinating them while maintaining balance is also challenging, since reversed pendulum model [Raibert and Hodgins 1991] does not apply here. I attempted to figure out the relations between the swinging torque and the compensating torques required to maintain balance.

2 Related Work

People have been fascinated by the quadruped locomotion for quite a long time. For more details and other research works, Skrba et. al. [Skrba et al. 2008] has a comprehensive survey around the topic of quadruped animation.

Studies in biomechanics and zoology provided valuable knowledge regrading quadruped locomotion, which is essential in order to construct adequate quadruped models for physics simulation. In seminal paper [Alexander 1984] by Alexander , he presented a detailed classification of different animal gaits, and set the standard terminology for describing gaits. In my project, I focused on the simplest walking gait, where duty factor of each leg is 0.75. The quadruped model is static stable during the motion.

The framework described in [van de Panne 1996] is composed of an articulated character and a 'pose-control graph representing the gait cycle. The pose control graph is a finite state machine, where each state represent a target pose of the articulated character. In physics simulation, the character is driven to the desired posed stored in the pose states by actuating PD controlled output torques:

$$\tau = k_p(\theta_d - \theta) - k_d\omega$$

where θ_d is the desired joint position, and θ, ω are the current joint angle and velocity.

Another challenge for actuated characters in physics simulations is how to keep balance. In their work in Yin. et. al. described in [Yin et al. 2007], they proposed two main balancing control strategies designed for biped walking motion. First, to maintain the position of the torso, a virtual PD controller is used to calculate a net torque τ_{torso}, depending on how much the torso is deviating from the upright position. The swing leg torque τ_{swing} is calculated by PD controller towards target position. As shown in Figure 1 To keep the net torque on the biped hip zero, the torque on the stance hip was assigned as $-(\tau_{torso} + \tau_{swing})$.

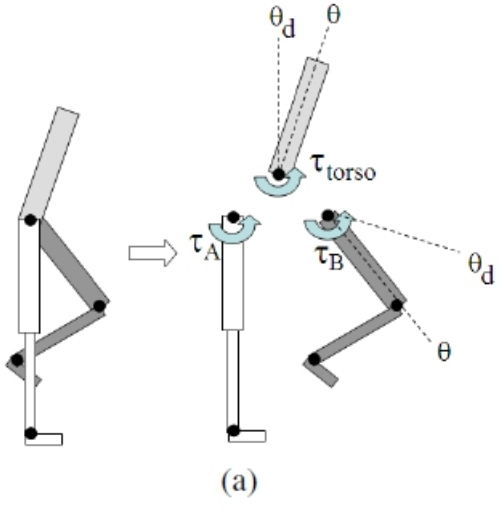

Figure 1: Balance strategy used in SIMBICON

I attempted to modify and apply this balance strategy to the quadruped simulation, with limited success. More De-

tails are provided in Implementation section. The second in SIMBICON is to track the position and velocity of the center of mass(CoM), and dynamically update the swinging target angle. This method is not included in my project.

3 Implementation

3.1 Quadruped Model design

For this project, I created the quadruped model from scratch. I referred to images of dog skeleton, as shown in Figure 2. Note that the height of the front hips are slightly higher than that of the rear hips, and the front legs are longer than the rear ones. Since only the leg movements are of interests to us, I decided to imitate the relative positions of the hip and knee joints, and simplify over the other details.

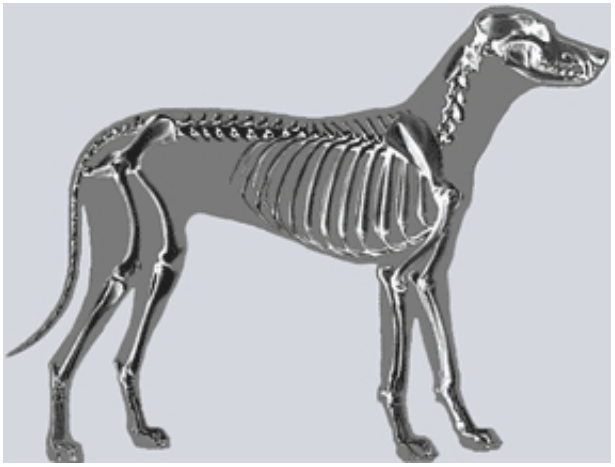

Figure 2: Dog Skeleton

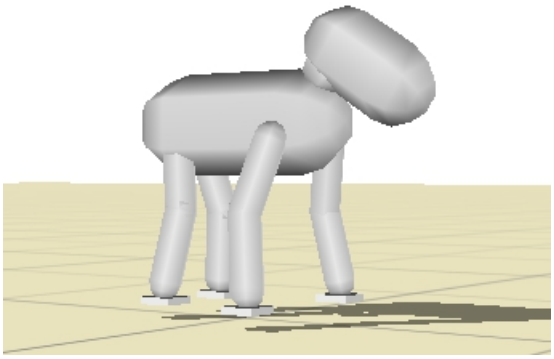

Figure 3: Simplified quadruped model

Figure 3 shows the simplified dog model I created. The simulation is built on top of Open Dynamic Engine(ODE), an open source physics simulation library[Smith 2006]. Therefore, the joint types used in my model are chosen from the joint types specified in ODE. Each of the four hips is a ball-in-socket joint, with 3 degrees of freedom. And the knees are modeled using hinge joints, with one degree of freedom. And the ankles are modeled using a ODE joint type called Universal joint having 2 degrees of freedom, as seen in Figure 4.

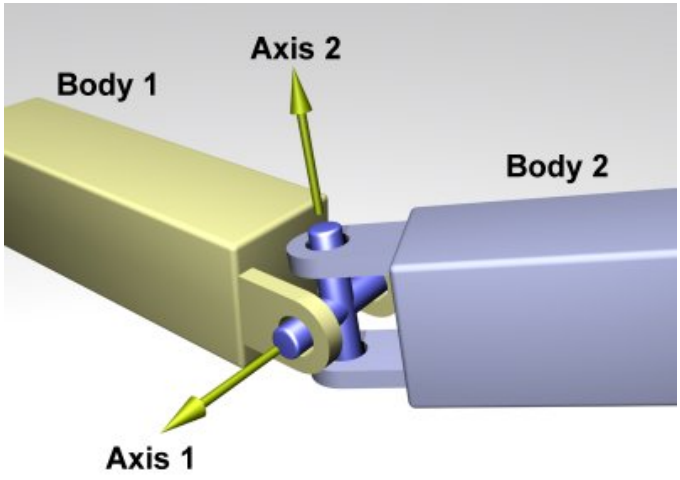

Figure 4: ODE Universal joint type

3.2 Standing up with PD

After constructing the model, the natural next step is to let it stand on its legs in the physics simulation. I implemented the PD function to compute the torque output from the difference of joint angles from the standard position. At first the model will stand with the knees bending for a while, before it pitched forwards and fell down (See Figure 5). Soon I realized it was actually collapsing under gravity and only stopped by the knee joint limits, and the falling down may be due to some small numerical disturbance, and the PD gain was too small to stop it.

After increasing the proportional gain coefficient of the PD controller, the quadruped model could then stand up on the legs with applied torque, and any slight torso pitch was corrected by updating the torques on the four hips. And these torques are necessary to support the weight of the quadruped before any additional torques is applied for any locomotion.

3.3 Pose control FSM and Controller Structure

For the pose control graph, I implemented a finite state machine very similar to Yin. et. al. described in [Yin et al. 2007]. Each state is composed of a full set of target angles for all the joints, and some semantic labellings the controller uses to interpret the graph. The state transitions model the

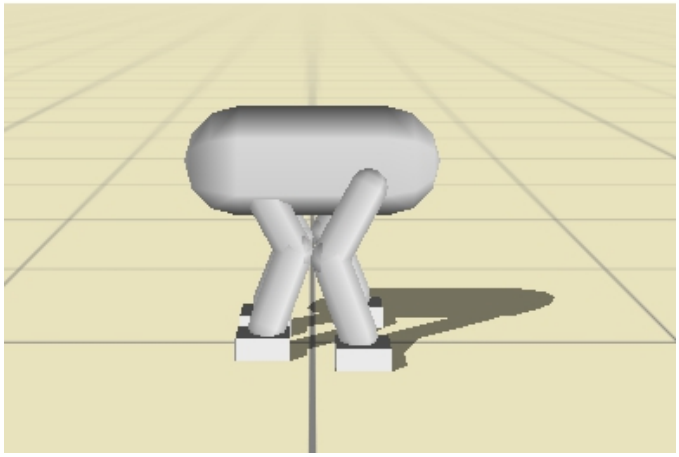

Figure 5: Collapsing without PD control

quadruped walking gait, which has been well studied. Each walk cycle follows the alternative swing of the legs:

$$\textbf{FL} \rightarrow \textbf{RR} \rightarrow \textbf{FR} \rightarrow \textbf{RL},$$

where FL means front-left leg, and RR means rear-right leg, so on. With each leg going through a swinging state and a striking state, the whole pose control graph would have had 8 states. However, some initial experiment with the quadruped model showed that using only two states for front leg swing often failed. This is because of the backward bending knees on the front legs. During the forward swing, the hip and the knee both try to rotate in the same direction, often push the foot down to the ground so much that the fiction jams the leg from forward motion. To solve this problem, I decided to adding a retracting state before the swinging states for the front legs. It gives the front knees some time and space to curling upwards, clearing the way for the forward swing. Therefore, the resulting pose control graph has 10 nodes, with each front leg having 3 states and each rear leg has 2 states. The structure of the graph is shown in Figure 6.

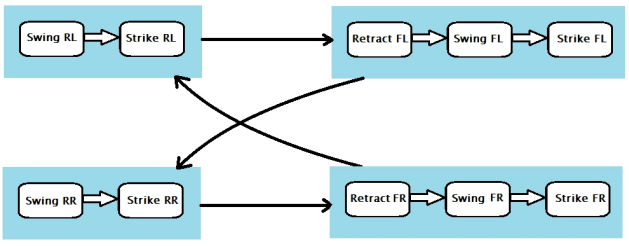

Figure 6: Pose control graph

3.4 Balance Control

As mentioned above, the simple balancing strategies I experimented with the quadruped model were inspired by the work in SIMBICON paper. One of the first things I noticed is that the compensating torque on the stance hip is essential to make the quadruped walk. Without that, the quadruped will simply tilt towards the side of the lifted leg, and lose balance and fall. The stance torque will temporarily support the extra weight offloaded from the swinging leg, and causes an implicit lateral shift of the center of mass(CoM). Therefore, the CoM was kept inside the support triangle formed by the three stance feet, and static balance is maintained.

The main difference when balancing a quadruped is that the CoM is roughly in the middle between the front hips and rear hips, not right above the single pair of hips, as in a biped. Therefore, the CoM may not fall in front of the stance foot during the swinging of the other foot. I experimented with dividing the counter-swing torque among the three stance hips, but found this only works in very specific cases. So I tried instead to only apply the entire torque to the lateral counterpart of the swinging hip. For example, if the front left hip is swinging, the front right hip will have the compensating torque.

While experimenting with the simulation, I realized a necessary condition for the walking gait to persist the speed resulting from the swings of front feet must be equal to that of the rear feet. Although the statement itself is intuitive, it can be easily overlooked while considering the target poses. Given the time limit, I manually adjust the target angles of the leg swingings.

4 Results

With a lot of manual tweaking, the quadruped model was finally able to walk steadily without falling down. Figure 7 contains the parameters giving a good walking cycle for my model.

I also tried to increase the walking speed of the quadruped by increasing the swinging target angles. I did this by controlling a coefficient used to increment the angles. But it appears the robustness is very limited, only small increase of the swinging angles yield successful walk. Figure 9 are some screen shots of a full walking cycle.

5 Discussion and Conclusion

In this project, I constructed a simple quadruped model mimicking skeletal structure of a dog, implemented a FSM based pose control graph to generate the walking gait, and applied some naive balancing strategies. I manually tuned the parameters to produce a stable walking cycle, and experimented with this controller under increased speeds.

Creating a walking dog model in physics simulation turned out to be much more difficult than I expected, and

Target Joint Angles	Front leg	Retract	Hip	-60
			Knee	230
		Swing	Hip	15
			Knee	140
		strike	Hip	-100
			Knee	10
	Rear leg	Swing	Hip	95
			Knee	-120
		strike	Hip	-15
			Knee	-25
PD coefficients	K_p			40
	K_d			3
Swing states transition time				0.1s

All target angles are in degrees, with regard to unit vector (0,0,1).

Figure 7: List of parameters used in walk gait

there are a lot issues I have not explored or solved robustly. In retrospect, the asymmetry of the front and rear limbs may have complicated things unnecessarily. Since the configurations for the front legs are slightly different from that of the rear, the control parameters need to be adjust accordingly. Since I was interested in creating general walking gaits and balance control strategies, a further simplified model would allow me to get a walking motion quickly, and spend more time working on balance.

My current implementation of the pose control graph has many limitations. All the non-striking states in the pose control graph currently last the same amount of time, this is a gross simplification. It will be interesting to see how specifying a varying time length for each state affects the resulting gaits.

Instead of manually adjusting the parameters to fit one particular model, it will be much more useful if the controller can be trained with an input model. For example, given the configuration of the four legs, let it optimize for the relation between the swinging angles for front and rear legs, so the linear velocities are consistent.

6 Acknowledgements

I would like to thank Michiel van de Panne, Kai Ding and Stelian Coros for offering guidance and advice. And thanks to Stelian's awesome ODE+OpenGL platform. This work was supported in part by TA payments.

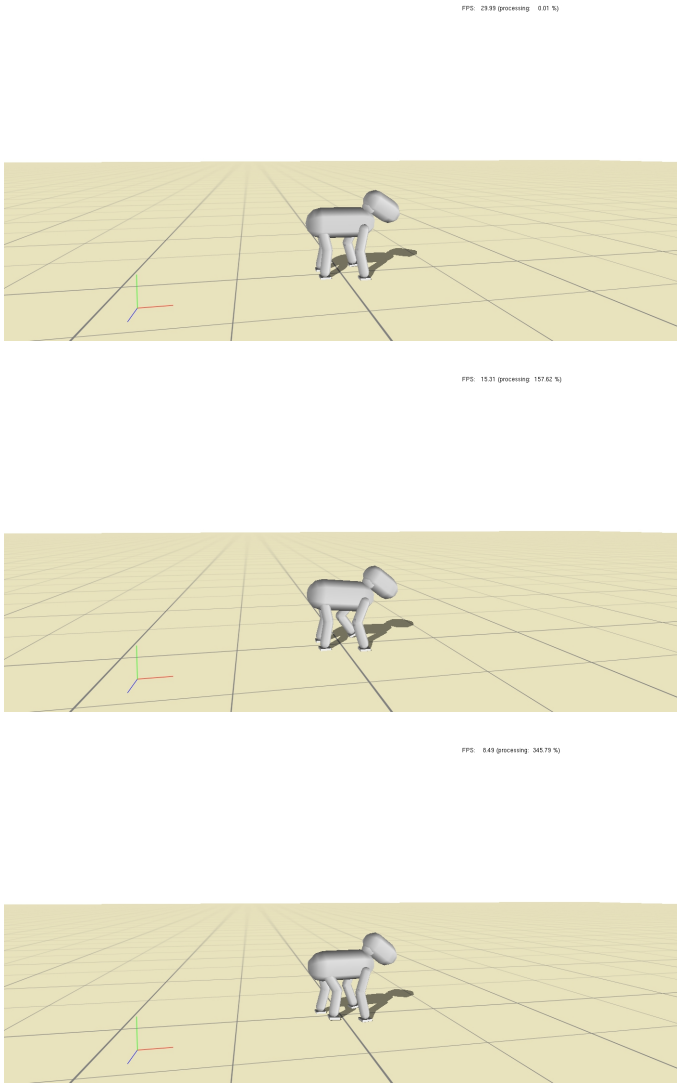

Figure 8: Snapshots of a full walk cycle

References

ALEXANDER, R. M. 1984. The Gaits of Bipedal and Quadrupedal Animals. *The International Journal of Robotics Research 3*, 2, 49–59.

RAIBERT, M. H., AND HODGINS, J. K. 1991. Animation of dynamic legged locomotion. In *SIGGRAPH '91: Proceedings of the 18th annual conference on Computer graphics and interactive techniques*, ACM, New York, NY, USA, 349–358.

SKRBA, L., REVERET, L., HETROY, F., CANI, M.-P., AND O'SULLIVAN, C. 2008. Quadruped animation. In *Eurographics State-of-the-Art Report (EG-STAR)*, Eurographics Association, Eurographics Association.

SMITH, R. 2006. *Open Dynamics Engine User Guide*, 0.5 ed.

VAN DE PANNE, M. 1996. Parameterized gait synthesis. *Computer Graphics and Applications, IEEE 16*, 2 (mar), 40 –49.

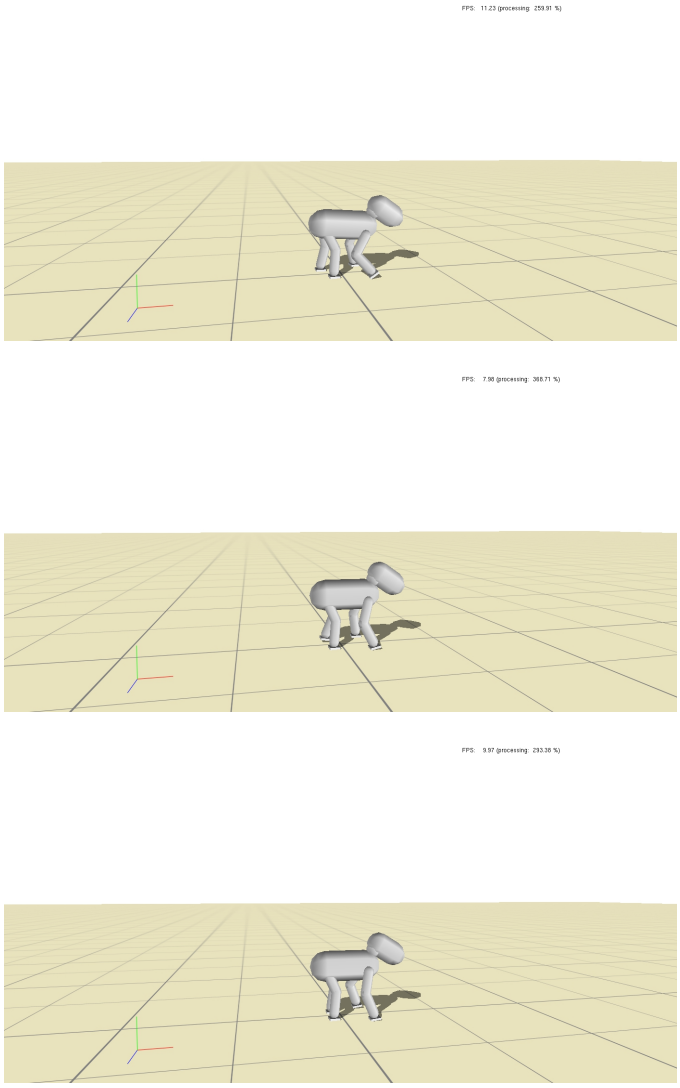

Figure 9: Snapshots of a full walk cycle continued

YIN, K., LOKEN, K., AND VAN DE PANNE, M. 2007. Simbicon: simple biped locomotion control. In *SIGGRAPH '07: ACM SIGGRAPH 2007 papers*, ACM, New York, NY, USA, 105.