Learning to Jump Higher with Muscles

Duo Li*
University of British Columbia

Abstract

We present a complete system to generate jumping movement for a given musculoskeletal model. The two main components for our method are a forward musculoskeletal simulation and an optimization of the muscle activation levels. We exploit and extend the strand model into Hill type muscle model to handle the sliding constraints between muscle and skeleton, and incorporate this muscle model into our dynamics system. The activation control signals have been modeled as a natural spline, and our optimization automatically generates suitable muscle activation sequences to achieve a maximum-height jumping movement, without any prior knowledge. We test our system with three different musculoskeltal models. The results agrees with our common knowledge about jumping.

Keywords: Jumping, Muscle, Strand, CMA

1 Introduction

Recently, muscle based character animation has been draws attentions. Compared with joint torque based approach, using muscles give the added benefit to animation in terms of robustness due to the muscle properties [van Soest and Bobbert 1993], extra inertias [Pai 2010], and subtle skin deformations [Sueda et al. 2008]. This motivates our use of the muscle in this paper.

Although using muscle provides significant advantages, we have to pay the price in modeling, simulation and control aspects. Due to the redundancies in musculoskeletal systems, modeling them are tedious. Moreover, since muscle is strongly connected with skeleton, handling the highly constrained dynamical system itself is not a easy task. Finally, compared with joint based approach, the muscle activation based controller not only maintains a relatively high dimensionality which makes the control space too large to be directly solved, but it also provides no direct or intuitive mapping to the desired movement such as using the PD control method.

Maximum-height jumping is a good example to demonstrate both the advantages and disadvantages of muscle based animation. On one side, compared with high gain joint torque based approach, the extra inertia of muscles reduce the magnitude of input signals of the system, which enhances the robustness of the system. On the other side, jumping introduces several diffculties: 1) muscles used for jumping sometimes involve more than two insertion points. For example, the vasti muscle in human leg slides around the knee3. Purely Lagrangian based muscle model can not handle this constraint easily. 2) The jumping behavior is ridled with local minima. For instance, squatting a little or even lifting the heel generate a small upward acceleration, but maximal height jumping needs a deep enough squatting stage, though it is not a greedy choice.

In this project, we simulate and control the maximum-height jumping. Our system consists of two main components: simulating motion and optimizing control strategies. We simulate the musculoskeletal system with rigid segments and strand model in maximal coordinates. The muscle activation level signals drive our system forwardly. The second component provides an automatic way to discover the best activation sequences for jumping. We empoly an optimization technique called Covariance Matrix Adaptation (CMA) to explore the domian of possible activation curves. Finally, we apply three different musculoskeletal models into our system, and observe the generated jumping gaits.

*e-mail: duoli@cs.ubc.ca

2 Jumping Simulator

2.1 Skeleton Model

We model the skeleton with several planar articulated rigid cuboids. Each two adjacent segments are connected with a hinge joint. To mimic the friction bewteen the ground and the lowest linkage (ususally the foot), we also place a fake friction joint, which constrains the lowest linkage from sliding on the ground, but allowing it to move upward and rotate along y axis (see Figure 3). After the skeleton jumps off the ground, we cancel this 'friciton' joint.

2.2 Muscle Model

Linkages are connected by a group of muscles. Any simple movement is the result of the coordinated activation of all the muscles. Purely Lagrangian methods (e.g. mass spring systems, FEM) cannot capture the sliding between muscle and bones easily (e.g. Figure 3, vasti muscle). Instead, we employ the strand model [Sueda et al. 2011], which combines both the Lagrangian method and Eulerian method to handle this muscle-bone sliding problem. Usually, a curve can be represented with a bunch of control points and intermediate segments. In Lagrangian model, the control point q is defined by its world position, $q = [x, y, z]$. In contrast, for Eulerain model, the control point q need to be defined as its material coordinate, that is $q = [s]$. Strand model incorprates both the Lagrangian part and Eulerian part into the state vector, $q = [x, y, z, s]$. This representation allows the material to flow around control points. Details of this method can be found in [Sueda et al. 2011].

We still need several modifications to exploit strand model for simulating the Hill type muscle [Hill 1938]. Generally, the Hill type muscle can be written as

$$f = f_l(\epsilon) f_v(\dot{\epsilon}) f_{active} + f_{passive} \tag{1}$$

$$\tag{2}$$

ϵ is the strain on the strand. In this project, we use Cauchy linear strain $\epsilon = \frac{l}{s_0} - 1$, where l is the current length of the segment, s_0 is the initial Eulerian distance and therefore the rest length of this segment. f_l provides the coefficent of the active force with respect to the current strain, which is also proportional to the current segment length. f_v is the coefficent for current strain rate, which

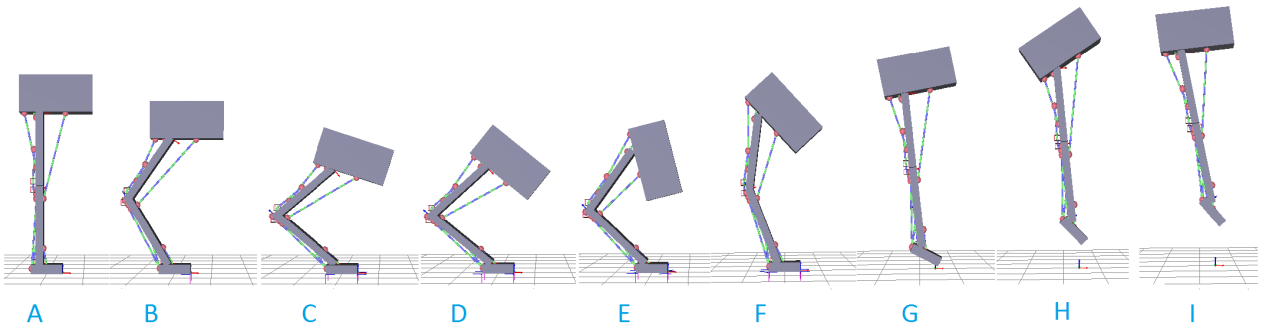

Figure 1: *Ostrich jumping: maximizing the top-middle point of the trunk*

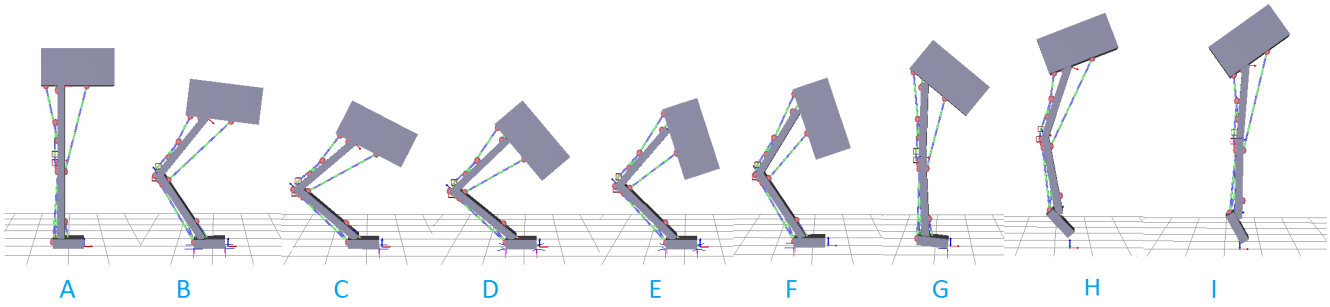

Figure 2: *Ostrich jumping: maximizing the top-right point of the trunk*

corresponds to the contraction velocity. Active force f_{active} is linear with strain. Passive force $f_{passive}$ is linear when the strand has been stretched, but it will be cancelled out during contraction. Our derivation of strain ϵ and strain rate $\dot{\epsilon}$ with respect to the specifically mixed Lagrangrian-Eulerain state q can be found in Appendex A.

$$f_{active} = k_p \epsilon \tag{3}$$

$$f_{passive} = \begin{cases} k_p \epsilon, & \text{if } \epsilon \geq 0 \\ 0, & \text{if } \epsilon < 0 \end{cases} \tag{4}$$

2.3 Overall Equation of Motion

Finally, we incorprate both the muscle and the skeleton into a quadratic programming equation:

$$M\dot{q}_t = M\dot{q_{t-1}} + hf_{ext}, \tag{5}$$

with equality constraints (joint)

$$G_i \dot{q}_t = 0, \tag{6}$$

and inequality constraints (joint limits, strain limits, contact constraints)

$$G_e \dot{q}_t \geq 0 \tag{7}$$

M is the total mass matrix (it is not diagnal due to the strand). G_e and G_i are the equality and inequality constraint matrices. Because the constraints are solved at velocity level, the system may drift away from the constraint manifold over time. To compensate for this drift, we add a post-stabiliztion step [Cline and Pai 2003] after taking a position step to get a correction vector δq.Finally, the generalized positions are updated as $q = q_0 + \delta q$.

3 Optimization

Each muscle has different functionality. We suppose that the character has no prior knowledge about this, and allow it learn how to jump automatically.

3.1 Control Signal Respentation

Usually the simulation takes several hundred time steps. Encoding all the control signals for all the time steps incurs the curse of dimensionality. Instead, we uniformly sample the time axis to obtain several points in time. Therefore, activation levels on these points of time for each muscle are the actual control signals which we are applying into our system. All intermediate activation levels will be interpolated with a natural spline accroding to the current time step. The signals will be clamed to $[0, 1]$ if it exceeds this range.

3.2 Objective Function

The objective function in our system mainly tries to maximize the vertical distance of jumping, yet takes into account the amount of energy consumed.

$$u = \arg\min w_1 H + w_2 \sum_{i,j} (a_{ij}), \tag{8}$$

H is the maximal height during this whole jumping movement. a_ij is the activation level for muscle j at time step i. It is generated by the sampled control signal u via the natural spline. w_1 and w_2 are the weights. Because we mainly want to measure a one-time jumping, we stop the simulation when the model starts to decline, or a second collision happens between foot and ground.

3.3 Optimization

Although our objective function is very simple, it prones to local minima due to sub-optimal jumping gaits, contact with ground and etc. Many previous works shows that Covariance Matrix Adaptation (CMA) is a suitable choice for this stituation. CMA evaluates the objective function value with a population of samples over the parameter space. Samples with low objective function value will be discarded. Then CMA updates its mean and variance matrices to build a new sampling distribtuion upon the previous population of samples with high objective function value. After many iterations, it converges to a optimal solution with low covariance. More details of CMA method can be found in [Hansen and Kern 2004]

4 Experiments and Results

We tested our system with three different models: a human(Figure 3), ostrich(Figure 4), and fish(Figure 5) model. The wire-frame box stands for bone segments, and the green-violet line represents the muscles. The solid sphere stands for the Lagrangrian node that the material coordinate is fixed on at this point, while the wire-frame box node denotes the Eulerian node which permits the changes on the material coordinate. The red-green-blue coordinate diagram represents the joint axis. Since we mainly consider the jumping in 2D, all joints on the skeleton are hinge joint that only rotation around y axis (green arrow) is permitted.

4.1 Models

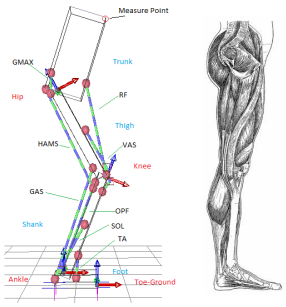

Figure 3: *Human model*

Following [Marcus G. Pandy and Levine 1992], we model the human body (Figure 3) with four segments (foot, shank, thigh, trunk), with three hinge joints to connect the adjacent segments (ankle, knee, hip). Toe-Ground joint is the fake friction joint. Eight muscles connect these four segments: soleus (SOL), gastrocnemius (GAS), platarflexors (OPF), tibialis anterior (TA), vasti (VAS), rectus femoris (RF), hamstrings (HAMS), and gluteus maximus (GMAX).

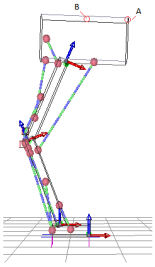

Figure 4: *Ostrich model*

Although real ostrich contains four links on its leg, we model the ostrich (Figure 4) by removing the short knee joint, as well as flipping the RF, HAMS, and VAS muscles from human model to mimic this backward-leg creature since we can not get the real anatomical data.

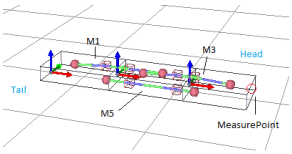

Figure 5: *Simplified fish model. Note that actually the muscle distribution is symmetric. For the sake of convenience, we only show M1, M3 on the upper side and M5 on the downside. Their counterpart muscles M2, M4, M6 reflect on the other side*

Finally, we also applied our system into a non-leg model: a three-segment fish lies on the ground (Figure 5). The right shorter link is its head and the left longer link is its tail. We hope that the fish can learn to jump off the ground to maximize the height of its head. Again the left joint is the joint used to mimic friction. It constrains the tail from sliding on the ground surface.

4.2 Optimization Results

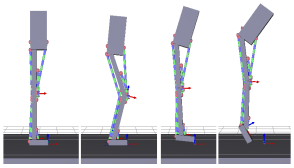

Figure 6: *Human jumping for 1st iteration*

Let us first consider the human model. We place the measurement point on the top left corner of trunk segment (Figure 3). In the first few iterations, the model learns to use RF, HAMS and GAS to squat shallowly, and then strectch VAS to generate a larger upward force, yet in a very inefficient way (Figure 6).

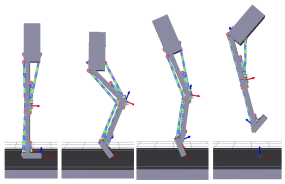

Figure 7: *Human jumping for 15th iteration*

After 15 iterations, it learns to lift its body around ankle joint with OPF and SOL muscles, yet it still do not know to gain more scores by raising its trunk (Figure 7).

Finally, after 50 iterations, The human model is able to coordinate its inertia distribution, and jumps very efficiently (Figure 8).

For the ostrich model, we first place the measurement point on the top-left corner of the trunk (Point A in Figure 4). It has a similar convergence pattern as the human model. The model learns to crouch first with proper muscles, and then jumps with another group of muscles. Moreover, During jumping, it learns to lift its right side

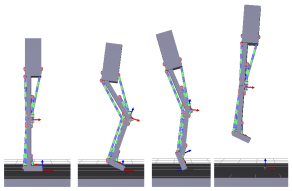

Figure 8: *Human jumping for 50th iteration*

of the trunk to obtain a higher gain (I in Figure 2). An interesting finding is that when we adjust the terminal condition to allow the model to fall back on the floor several times, CMA returns us a pattern where the ostrich learns to adjust its posture by several small hopping steps. It crouches to get the inertia in a suitable direction, and then jump higher than previous experiment configuration.

We then place the measurement point on the top-middle of the highest segment (Point B in Figure 1). CMA returns us a different optimal solution that instead of tilting up the upper block, the upper block remains close to horizontal (Figure I in Figure 1) at the optimal height configuration. These cases demonstrates that our system can handle muscle coordination successfully and robustly.

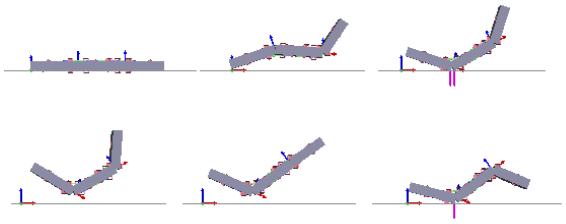

Figure 9: *Fish jumping*

The last experiment is to let the fish learn to jump off the ground, while maximizing the height of its head. Surprisingly, after only one iteration with 40 samples, the fish can find a nice pattern that lift its head using its tail to accelerate the whole body (Figure 9). The following iterations only improve this solution a little bit (Figure 10). However, since we use a very simple muscle model with only three links, the fish cannot actually flips its head and then jumps with its tail as we expect. A better anatomical model should improve this experiment significantly.

4.3 Convergence

Figure 10 plots the convergence behaviours for our four test cases. We can observe that although the mean of objective function value decreases after many iterations, only with less than 10 iterations, we can sample several nice candidate solutions which are very close to the final optimal one. Therefore, we can memorize the best solution for every iteration. If the objective value converges to a certain envelope, we can assume that the whole optimization has converged already.

Finally, thanks to the muscle property that penalizes the elongation and fast contraction, with different muscle activation level patterns, we can achieve similar jumping sequences. The advantage of this is that it demonstrates the robustness of muscle based control model. However, it also makes the optimization more difficult to solve since it is a highly multi-root system.

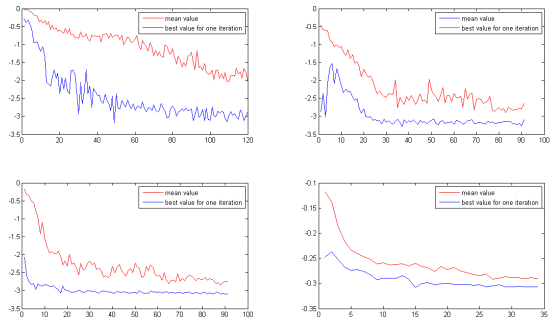

Figure 10: *Convergence: x axis is the iterations, y axis is the objective function values. Upper left figure for human model; upper right figure for ostrich model with measurement point A; lower left figure for ostrich model with measurement point B; lower right figure for fish model*

5 Implementaion and Performance

Since we need to use the novel strand model, we developed our own forward physics engine including a rigid body dynamics system, a strand simulator, OBB based collision detection, and contact handling. For the sparse matrices computation, we exploit the CSparse library. For the QP solver, we use a dense solver called QL. We also employ Boost library for seriliazing and unseriliazing to reload the simulator for optimization. Finally, our optimization is based on the open source CMA package. We run our system on an Intel i5 core machine with 4GB memeory. The forward simulation usually takes only about 1-3 seconds. But the optimization takes about a couple of hours to converge. However, good candidate solution shows up after only 3-10 iterations.

6 Conclusions and Future Work

In conclusion, we have developed a complete system that automatically generates optimal jumping movement with input musculoskeletal model, even for a model with no legs. It learns the coordinated activation of muscles successfully, albeit slowly. Due the time limitation, we have not tested our system for a 3D case, or with full skeleton model. Incorprating more parts of the musculoskeletal model such as arms and bendable spine for human model should improve the realism, but the increased Dofs for control signals may impede the optimization convergence. Also, in this project we exploit the simple spline interpolation for control signals without futher disscussions. Although this seems to work, a better parametrization apporach should improve our system to a large extent. Finally, we also want to extend our system to learn forward jumping.

7 Gains and Challenges for this project

I was always curious about how the animation system works. With this project, I walked through the whole pipeline of the animation system from modeling to simulation, and finally control. I learned lots of specific techniques such as CMA, strand model and etc. In addition, extending strand model into muscle model is a nice mathematic practice for me. The main challenge I encountered during this project is how to specify the parameters for CMA. It takes a long time to run a whole optimization, so I need to analyze very carefully before I apply any changes of the parameters into the CMA. Besides, tuning parameters for muscle properties is also a tedious

work, which costs me lots of time either.

Acknowledgements

We would like to thank Professor Michiel van de Panne and Professor Dinesh Pai for their valuable sugguestions and guidance on this project. We also want to thank Shinjiro Sueda for his help on the strand simulation.

Appendix A

This appendix shows how to calcuate strain ϵ and strain rate $\dot{\epsilon}$. Suppose we have two points $q_0 = [x_0, s_0]$, $q_1 = [x_1, s_1]$, $q = [x_0, x_1, s_0, s_1]^T$, $\dot{q} = [\dot{x_0}, \dot{x_1}, \dot{s_0}, \dot{s_1}]$

$$E = \frac{1}{2} Y \epsilon^2 V_0 = \frac{1}{2} Y \epsilon^2 \Delta s A \tag{9}$$

$$= \frac{1}{2} k_p \epsilon^2 \Delta s \tag{10}$$

$$\epsilon = (\frac{\sqrt{\Delta x^T \Delta x}}{\Delta s} - 1) \tag{11}$$

$$\Delta s = s_1 - s_0 \tag{12}$$

$$\Delta x = x_1 - x_0 \tag{13}$$

E is the elasticity energy. Young's modular coefficent Y times area of this segment A forms a coefficient k_p

With chain rule, we get

$$\frac{\partial E}{\partial (\Delta x)} = k_p \epsilon \frac{\Delta x}{\sqrt{\Delta x^T \Delta x} \Delta s} \tag{14}$$

$$\frac{\partial E}{\partial (\Delta s)} = \frac{1}{2} k_p \epsilon (\epsilon + 2) \tag{15}$$

Therefore,

$$\frac{\partial E}{\partial \dot{q}} = [k_p \epsilon \frac{\Delta x}{\sqrt{\Delta x^T \Delta x} \Delta s} (-I, I), \frac{1}{2} k_p \epsilon (\epsilon + 2)(-1, 1)] \dot{q} \tag{16}$$

References

CLINE, M. B., AND PAI, D. K. 2003. Post-stabilization for rigid body simulation with contact and constraints. In *ICRA*, 3744–3751.

HANSEN, N., AND KERN, S. 2004. Evaluating the cma evolution strategy on multimodal test functions. In *PPSN*, 282–291.

HILL, A. 1938. The heat of shortening and dynamics constants of muscles. *Proc. R. Soc. Lond. B (London: Royal Society) 126*, 843 (Oct).

MARCUS G. PANDY, FELIX E. ZAJAC, E. S., AND LEVINE, W. S. 1992. An optimal control model for maximum-height human jumping. *J. Biomechanics 25*, 2 (Feb), 207–9.

PAI, D. K. 2010. Muscle mass in musculoskeletal models. *J. Biomechanics 43*, 11 (Aug), 2093–2098.

SUEDA, S., KAUFMAN, A., AND PAI, D. K. 2008. Musculotendon simulation for hand animation. *ACM Trans. Graph. 27* (August), 83:1–83:8.

SUEDA, S., JONES, G. L., LEVIN, D. I. W., AND PAI, D. K. 2011. Large-scale dynamic simulation of highly constrained strands. *ACM Trans. Graph. 30* (August), 39:1–39:10.

VAN SOEST, A. J., AND BOBBERT, M. F. 1993. The contribution of muscle properties in the control of explosive movements. *Biological Cybernetics*.