Learning Muscle Based Arm Controllers

David Brown

Abstract

We present a system for automatically creating control policies for a simulated biologically based arm performing basic tasks such as reaching for an object. The arm is driven by forces from a set of Hill-type musculotendon units (MTU). Each MTU is controlled by an excitation parameter that represents the overall activation of the muscle by the motor neurons. The control policies use splines to drive simple PD-controllers which in turn generate the muscle excitation signals which drive the MTUs. Spline control points and PD-controller parameters are learned for a specific task using a reenforcement learning based optimization which rewards how well the arm completes the task and penalizes energy usage. We use this general system to generate controllers for specific tasks.

1 Introduction

Getting a robot or animated character to perform a task both efficiently and skillfully can be quite difficult. Some robots, such as the Barret WAM arm allow you to guide the robot through a task and it can then repeat those motions. However most tasks involve not only following a specific motion but applying a certain amount of force in order to be successful. The forces required for a task cannot be as easily demonstrated as a simple motion. Also robots are often controlled using positional servos so even if you do know the force it is not always clear how to achieve that force. If you do have a controller that allows a specific robot to perform a task it may not be the most efficient or skillful way of performing that task. It can be time consuming and challenging to hand design a controller for a specific task and robots often require a whole repertoire of basic tasks to achieve some larger goal.

We address these issues with a system that learns control policies for a simulated robot arm based on some objective function. Furthermore our simulated arm controlled by forces generated using Hill-type musculotendon units (MTUs) [9]. This allows us to control how much force is used to perform a specific task. The muscle force is controlled using the biologically inspired control functions from Wang et al. [7]. The benefit of force based control instead of direct position control is it allows the arm to more or less compliant as needed by its current task. The goal of specifically using biologically inspired actuators is to enable the system to emulate the smooth and energy efficient arm control of animals.

Hand designing control algorithms for a muscle controlled arm is not likely to be any easier than for other actuation methods so we use a reinforcement learning based optimization to dome up with control policies for performing specific tasks. The control policy will be based on a set of splines specifying the target value of each degree of freedom of the arm. The muscle excitation value of each MTU will be controlled using PD-controller based on the current and target joint angles of the arm. The parameters of the PD-controllers and the joint angle splines will be evaluated using the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) stochastic optimization method [1]. This technique is very similar to several recent robotics works which use Path Integral Policy Improvement (PI^2) and Dynamic Motion Primitives (DMPs) to learn control robot control strategies [3–5]. We are simply use CMA-ES instead of PI and splines instead of DMPs. The similarities between CMA-ES and PI^2 have been shown by Stulp and Sigaud [6] and splines are essentially similar to DMPs.

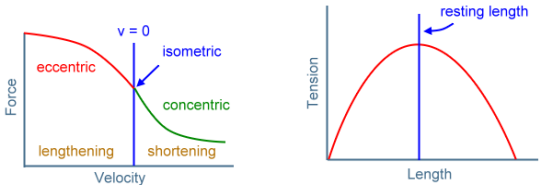

Figure 1: Muscle tension-length and velocity-force curves

2 Muscle Model

Our MTU model is based off the one presented by Wang et al. [7]. The MTU is made up of a contractile element (CE), and a parallel-elastic element (PE) in parallel connected to a serial-elastic element (SE) in series. The CE and PE represent the active and passive forces generated by the muscle fibers while the SE represents the force from the tendon. The CE is parameterized by an optimal length l^{opt} and a maximum isometric force F^0. The force it generates depends on these parameters as well as the current muscle activation a, CE length l^{CE}, and velocity v^{CE}

$$F^{CE} = aF^0 f_l(\frac{l^{CE}}{l^{opt}}) f_v(\frac{v^{CE}}{l^{opt}}) \quad (1)$$

where f_l and f_v are the force-length and force velocity curves as in Figure 1.

The force generated by the PE is passive and depends only on the current length of the CE. The total force generated by the MTU can be represented as the sum of the force from the CE and PE.

$$F^{MTU} = F^{CE} + F^{PE} \quad (2)$$

Since the SE is in series with the PE and CE, the force in the SE is also equal to the force generated by the MTU

$$F^{MTU} = F^{SE} \quad (3)$$

The SE is also passive and its force depends only on the ratio of it's current length (l^{SE}) to its slack length (l^{slk}) which is a property of the MTU. Using the equation

$$F^{SE} = F^{CE} + F^{PE} \tag{4}$$

we can compute the current velocity of the CE

$$v^{CE} = l^{opt} f_v^{-1} \left(\frac{F^{SE} - F^{PE}}{aF^0 f_l(l^{CE}/l^{opt})} \right) \tag{5}$$

We can then numerically integrate v^{CE} to update l^{CE}. Equations for F^{PE} and F^{SE} can be found in [7] and its supplementary material.

Finally the muscle activation value a is not directly controlled but is instead modelled by a first order differential equation which is integrated using $a_{t+1} = 100h(u_t - a_t) + a_t$ where u_t is the excitation signal and h is the timestep.

2.1 Muscle Energy

We not only want to get the arm to perform a task, but we also want it to do it in as energy efficient a way as possible. This means we need to be able to measure how much power a muscle is using at any given time. We again use metabolic energy expenditure described in [7] summarized below.

The total metabolic power $\dot{E}$ used by a muscle is a combination of several terms

$$\dot{E} = \dot{A} + \dot{M} + \dot{S} + \dot{W} \tag{6}$$

where $\dot{A}$ is the muscle activation power, $\dot{M}$ is the muscle maintenance power, $\dot{S}$ is the muscle shortening power and $\dot{W}$ is the mechanical power of the muscle.

The terms $\dot{A}$ and $\dot{M}$ represent the amount of heat generated by the excitation and activation of the muscle respectively. They both depend on the mass of the muscle since larger muscles produce more heat. In our examples we set the mass of the muscle to be directly proportional to how much force it can exert. The muscle shortening power $\dot{S}$ represents the heat generated by the shortening of muscle fibers and is thus a function of the total force F^{MTU} and the velocity of the contractile element v^{CE}. Lastly $\dot{W}$ is the mechanical power and thus depends on the force and velocity of the contractile element (F^{CE} and v^{CE}) since that is the only component that is actively generating force.

3 Task Controller

The task controller we use is very general with parameters that allow it to be adapted to perform a wide variety of tasks. It is based on PD controllers which calculate excitation signal for each MTU based on the current joint angle and a desired angle of the joint that MTU influences.

$$\begin{aligned} u_m = \max \Big\{ & k_m \left(\theta\left(t - \Delta t_m\right) - \hat{\theta}_m \right) \\ & + d_m \left(\dot{\theta}\left(t - \Delta t_m\right) \hat{\dot{\theta}}_m \right), 0 \Big\} \end{aligned} \tag{7}$$

where $\hat{\theta}_m$ and $\hat{\dot{\theta}}_m$ are the desired joint angle and velocity, and Δt_m is the muscle dependant time delay for the propagation of the excitation signal. We use only positive u_m because a negative excitation doesn't make sense since the MTUs can only apply a contractive force.

Each PD controller has free parameters k_m, d_m, $\hat{\theta}$ and $\hat{\dot{\theta}}$. For non-trivial tasks we will need to vary these parameters over time making them functions of t: $k_m(t)$, $d_m(t)$, $\hat{\theta}_m(t)$ and $\hat{\dot{\theta}}_m(t)$. We define these functions as piecewise cubic splines parameterized by n control points $\boldsymbol{y} \in \mathbb{R}^n$ and duration t [8]. Using a cubic spline also gives us the additional parameters n and t. For our experiments we fix n and t. The remaining parameters can then be chosen using the reinforcement learning method described in section 4.

4 Controller Optimization

To learn controller parameters for a specific task we use the stochastic optimization method CMA-ES with a task specific cost function plus the energy term described in Section 2.1. The general form of our optimization function is

$$f(\boldsymbol{x}) = \sum_i w_i f_i(\boldsymbol{x}) + w_E \sum_{t=0}^{T} \frac{\dot{E}_t}{T} \tag{8}$$

where f_i is some task objective function with weight w_i. The second term is the energy penalty with weight w_E. Its value is the sum of the energy of the system at a given time $\dot{E}_t$ summed over the duration of the task T.

This optimization is very general to make it possible to optimize a variety of tasks as long as you have an objective function for that task. We test our system with a very simple objective of simply reaching a target given by

$$f_{reach}(\boldsymbol{x}) = \min_t \left\| p_t^{hand} - p^{target} \right\| \tag{9}$$

where p_t^{hand} is the position of the arm's end effector at time t and p^{target} is the reaching target position. We do not specify a desired time for the hand to reach the target so that we can specify multiple targets for a single task and the system can determine the best order to perform them.

5 Results

5.1 Muscle Model

To start we ran some basic experiments on our muscle model to see how it compares with experimental data from real muscles. To do this we created a sample muscle 10 centimeters in length with a maximum isometric force of 100 newtons. We set the muscle's excitation to its maximum value and looked at the force it produced for different lengths. Figure 2 shows the results of this experiment. The full excitation force vs length curve is similar to results obtained from experiments on individual sarcomere. It has a similar peak at the optimal length of the muscle followed by a small valley and then an exponential increase. The one major difference is there is a second peak just after the peak at the muscle's optimal length. However it is reasonably close given the simplicity of this muscle model.

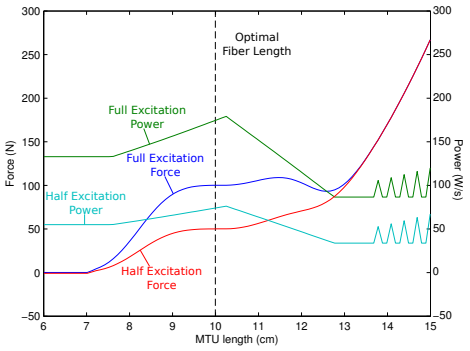

Figure 2: Graph of how the MTU force and metabolic power change with both length and excitation.

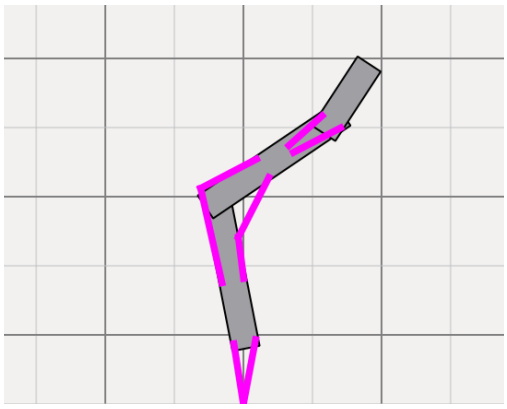

Figure 3: Arm model showing links (grey) and attached muscles (magenta)

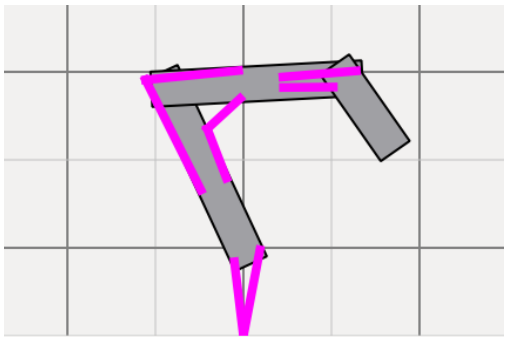

Figure 4: Optimized rest state of the arm

We also looked at the force when the muscle is only at half it's maximum excitation. According to the full excitation curve, when the muscle is stretched beyond it's optimal length it will become unstable since the longer it gets, the smaller the force it can apply. However this is only the case if the muscle is at it's maximum excitation. At lower excitations we can simply increase its excitation to get a larger force. Additionally, at lower excitations the force valley after the optimal muscle length is much less pronounced which helps alleviate the instability problem. When at half the maximum excitation as shown in Figure 2, the valley is actually non-existent.

Additionally since the length of the Contractile Element (CE) in the MTU does not change immediately when the MTU is stretched. So when the muscle is stretched it is first the Serial Element (SE) that is lengthen in our model and since the SE is essentially a spring it will apply more force and hopefully prevent the MTU from going unstable.

Figure 2 also shows the energy used by the muscle at different lengths. We did not have any experimental reference to compare this to, however we would expect the energy used to be roughly proportional to the force applied by the CE since that is what actively generates force in our muscle model. This is what we see in the length vs energy curves however there are some odd spikes in the curve as the length increases. We believe this is due to integration errors in calculating the length of the CE as we found their shape changed when the timestep was changed. Since it is rare for the muscle to be stretched to the point where the spikes are we did not find that they caused much of an issue.

5.2 Arm Control

To test our arm control system we used a simple two-dimensional 3-link arm with two muscles per arm as shown in Figure 3. The root and the forearm (second link) both weight 5 kg and the hand weighs 1 kg. The two muscles on the root and the hand are identical other than their positions. The muscle on the right side of the forearm is shorter and has a lower internal attachment point than the muscle on the left. Because of this the natural pose of the forearm is to be pointing to the right with respect to the root. Because we only have two muscles per arm the joint limits of each link is roughly 170 degrees in either direction. We expressed these limits as box constraints in the optimization. The arm is simulated using Open Dynamics Engine (ODE) rigid body simulator. For our optimization we use the Shark library's CMA implementation [2]. We ran each optimization for 3000 iterations though often they achieved reasonable convergence much sooner.

For our first test of the arm control we had our controller try to find a pose for the arm that required the least energy to maintain. This could be as a "rest pose" for the arm when it is inactive. To do this we had the optimization find a single pose (3 degrees of freedom, so the state space is $\mathbb{R}^3$) with the objective function being just the power expended by the arm. Figure 4 shows the optimized rest pose. The arm is in a configuration such that the muscles that are needed to counteract the effects of gravity are stretched slightly so that their passive force can contribute as much as possible to maintaining the pose. The center of mass is also roughly above the root joint so little energy is needed to keep the arm upright.

To test the arm in actually performing a task we gave it the objective of getting its hand as close as possible to two targets while also minimizing energy usage with a much lower weight for the energy objective. We used the parameters of a cubic interpolation spline with four evenly spaced control points as our objective function. We also fixed the final control point so the arm would

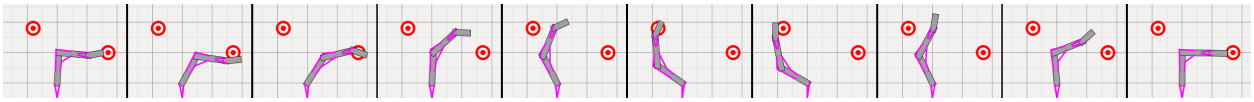

Figure 5: The optimized control of the arm reaching for two targets

return to some fixed pose. This gives us a state space of $\mathbb{R}^9$ (3 free control points time 3 degrees of freedom in the pose). Figure 5 shows the resulting motion of the arm. The arm gets within a centimeter of each target and tries to minimize the energy usage by going under the right-hand target and swinging up and slightly past the second target before returning to the fixed final pose. If needed one could also specify an objective that the arm have zero velocity when it reaches a target, however we simply had it touch the target at any velocity.

6 Conclusion

We have presented a method for creating an open loop control of a robotic arm using a simple muscle model to control the arm. We have also demonstrated that although the muscle model is simple, it does produce similar results to what has been seen in real muscles. Although we have only demonstrated this with two examples the system should be capable of producing much more. Since we treat the optimization to find our control parameters as a black box this system can come up with control for a variety of tasks as long as we can come up with an objective function that adequately characterizes that task.

6.1 Limitations

Although our system is general, it is limited by the control parameterization and the nature of whatever objective function we give it. We use cubic interpolation splines with relatively few control points to limit the size of our optimization space. More complex tasks might require more freedom which will cause finding a good result to take much longer. In addition, although CMA is relatively robust it will still have difficulty with very noisy problems that have lots of local minima. CMA also cannot be used on problems that have more than simple box constraints. The optimization can also be quite slow since we are performing a global optimization where the objective function is running a rigid body simulation for several seconds.

References

[1] Nikolaus Hansen. The CMA evolution strategy: a comparing review. *Towards a new evolutionary computation*, 102(2006):75–102, 2006.

[2] Christian Igel, Tobias Glasmachers, and Verena Heidrich-Meisner. Shark. *Journal of Machine Learning Research*, 9:993–996, 2008.

[3] Mrinal Kalakrishnan, Ludovic Righetti, Peter Pastor, and Stefan Schaal. Learning force control policies for compliant manipulation. *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4639–4644, September 2011.

[4] Petar Kormushev, Sylvain Calinon, and Darwin G Caldwell. Robot motor skill coordination with EM-based Reinforcement Learning. *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3232–3237, October 2010.

[5] Eric Rombokas, Evangelos Theodorou, Mark Malhotra, Emo Todorov, and Yoky Matsuoka. Tendon-driven control of biomechanical and robotic systems: A path integral reinforcement learning approach. *2012 IEEE International Conference on Robotics and Automation*, pages 208–214, May 2012.

[6] Freek Stulp and Olivier Sigaud. Path Integral Policy Improvement with Covariance Matrix Adaptation. *arXiv preprint arXiv:1206.4621*, June 2012.

[7] Jack M. Wang, Samuel R. Hamner, Scott L. Delp, and Vladlen Koltun. Optimizing locomotion controllers using biologically-based actuators and objectives. *ACM Transactions on Graphics*, 31(4):1–11, July 2012.

[8] Eric W. Weisstein. *Cubic Spline*. From MathWorld–A Wolfram Web Resource. `http://mathworld.wolfram.com/CubicSpline.html` Visited on 8/11/2012.

[9] F E Zajac. Muscle and tendon: properties, models, scaling, and application to biomechanics and motor control. *Critical reviews in biomedical engineering*, 17(4):359–411, January 1989.