Under-Actuated Robotics Control Via Memory-Based Machine Learning

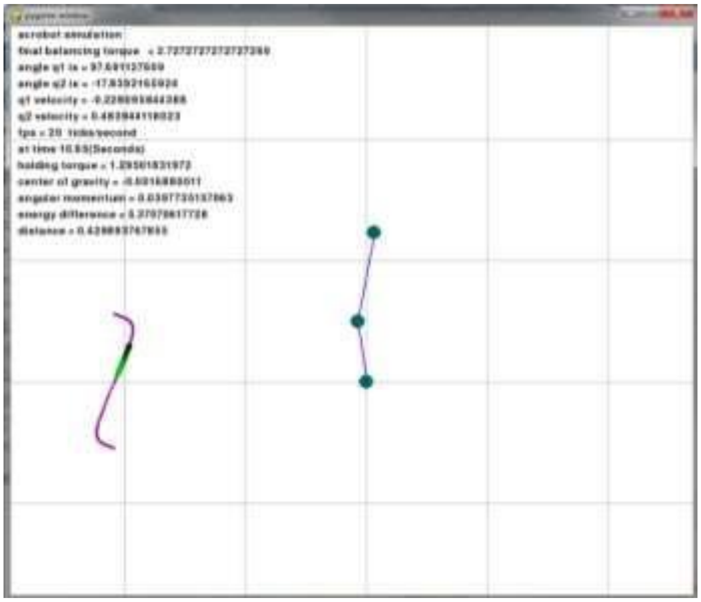
David Michelman
Professor Joshua Smith
Sensor Systems Lab, University of Washington

Oct. 23, 2015

Draft – Not yet published

Abstract

This paper explores a technique for balancing the inverted double pendulum, known as the acrobot, through memory based learning (also known as instance-based learning). The controller records state changes from an acrobot then builds a database of these state changes that can be searched to find the shortest path between any two positions. As the system records the acrobot's movements, the additional data is added to the database, providing continuously more accurate results. This approach has the benefit of being agnostic to specific hardware characteristics. While this controller has only been tested on an acrobot, the methodology behind it should be transferable to any system with a small enough state space. We have demonstrated its effectiveness in simulations and are currently incorporating it into hardware.



```
acrobot simulation
final balancing torque  = 2.7272727272727269
angle q1 is = 97.691137609
angle q2 is = -17.8352165924
q1 velocity = -0.228093844288
q2 velocity = 0.482944118023
fps = 20  ticks/second
at time 10.55(Seconds)
holding torque = 1.29561831972
center of gravity = -0.0016888011
angular momentum = 0.0307735137863
energy difference = 5.27078617726
distance = 0.429883767655
```

Figure 1: The simulation system in operation

1 Introduction

One common research vehicle for under-actuated robotics is the acrobot which has two arm segments joined by an actuated elbow joint and anchored by an unactuated shoulder joint (as in Figure 2 below). The acrobot is frequently studied since it only has two degrees of freedom but still displays highly nonlinear behavior [3]. The nonlinearity limits the usefulness of linear control methods while the low number of degrees of freedom make analytic results possible in many situations. The acrobot also has possible applications in bipedal walking. The initial challenge is to make it balance while standing straight up (an unstable equilibrium) then to have it swing up from an inverted position. Because the shoulder joint is unactuated, the acrobot must be controlled entirely through the elbow joint.

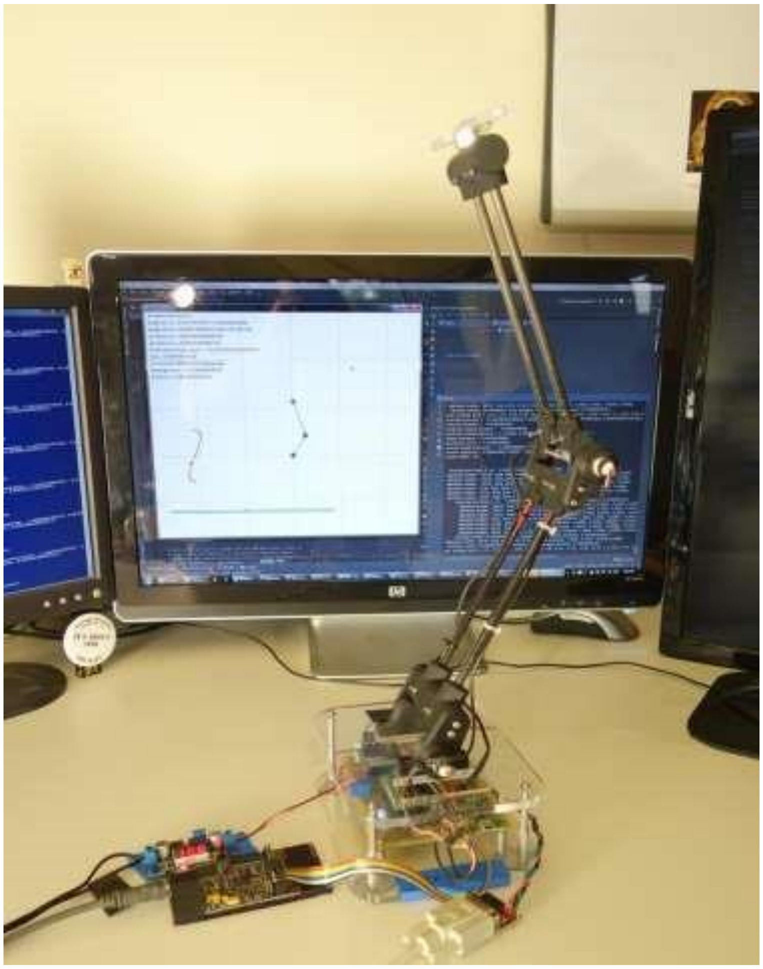


Figure 2: An acrobot posing

While research on controlling the acrobot has largely focused on adapting the Linear Quadratic Regulator (LQR) to work with the nonlinearity of the acrobot [3], we have taken an approach based on memory based learning. This entails storing observed state transitions from the acrobot then predicting its future movements by past data. We then search through the recorded data for the shortest path to the desired state. As the system records more acrobot state transitions and gains experience, the additional data is combined with previous measurements to create ever more accurate predictions.

1.1 Previous Work

There is a substantial amount of research on controlling acrobots, both in balancing them and swinging up from an inverted position. Balancing is commonly done using Linear Quadratic Regulators and previous research has pseudolinearized the acrobot with varying degrees of success [3]. Alternative controllers, such as one that attempts to minimize angular momentum [2], have also been successfully applied to the acrobot. A whole host of solutions have also been proposed for the swing up problem such as adding energy to the system until it swings all the way to a vertical state instead of trajectory planning [4]. Neural networks have also been employed [5]. Gary Boone presented a similar system to the one portrayed here in that it performs a tree search of possible end results, but it uses a mathematical model of the acrobot instead of observed results [1].

Our approach has a number of advantages and disadvantages when compared against LQR controllers:

- It's not necessary to determine the physical characteristics of the system, eg the lengths and masses of the arms, as is required by approaches based on mathematical models. In a laboratory setting this isn't so important, but in applications it can remove a hurdle and source of uncertainty.
- By incorporating actual recorded state transitions during operation, our approach offers continuous improvement through operation.
- Our approach doesn't require linearity while LQR controllers require linearized systems.
- Our approach requires very large amounts of computational power which makes operating in real time challenging. LQR controllers require significantly less computational power.
- Our approach requires experimental data about how the acrobot reacts to a range of torques over a large section of its state space. This data can be challenging to collect.

This paper is organized as follows. In section 2 we show the schematic model of an acrobot, in section 3 we describe the database design, followed by an overview of the operation in section 4. We discuss its machine learning attributes in section 5 and discuss the current status of the project in section 6.

2 Model of the Acrobot

The acrobot is a planar robot consisting of two arms that are linked together via an actuated joint. The lower arm is anchored to the ground with an unactuated joint that rotates freely. The acrobot's equations of motion will not be used in this paper, but they can be found in Nils A. Andersen, Lars Skovgaard and Ole Ravn's paper [3] or in section 3.11 of Russ Tedrake's book [4]. Figure 3 shows an acrobot with various measures marked.

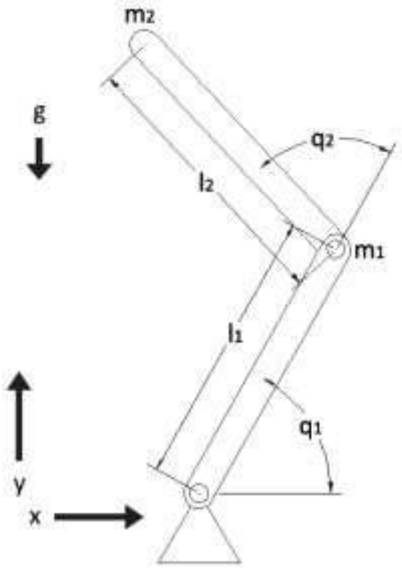


Figure 3: Acrobot schematic

3 Database Design

As the acrobot moves in response to torques, the software stores these responses in a database that is used to predict future responses.

The acrobot state is represented by a 4-tuple S = (q_1, $q_{\dot{1}}$, q_2, $q_{\dot{2}}$) including the shoulder angle and velocity and the elbow angle and velocity. The database has 5 dimensions comprised of an acrobot state and torque pair (S, T), where the torque T is a value within the range of torques that can be applied at the elbow (as determined by motor specifications). Each dimension has entries at specified intervals covering the operational range of the acrobot. For example, the shoulder angle dimension may have indices from 0 to 2π radians in 0.1 radian increments.

Each data entry stores the state that resulted from applying that torque to that starting state/position after a time delay. Looking up (S1, T1) returns the new state (S2) that that torque would result in. So for example, the entry at $q_1 = 1.4$, $q_{\dot{1}} = 0.2$, $q_2 = 0.1$, $q_{\dot{2}} = 0.4$, and $F = 0.5$ provides what the resulting q_1, $\dot{q}_1$, q_2, and $\dot{q}_2$ would be if a torque of 0.5 was applied to the acrobot while $q_1 = 1.4$, $q_{\dot{1}} = 0.2$, $q_2 = 0.1$, and $q_{\dot{2}} = 0.4$.

Note: controlling $\ddot{q}_2$instead of torque is a more common approach because it greatly simplifies the equations of motion and minimizes the effect of coulomb friction [3], but either can be used in this situation since the acrobot's equations of motion were not used and the lookup table will incorporate the effect of friction.

4 Operation Overview

The acrobot is controlled by continuously altering the torque applied by the elbow motor. We determine the next torque to apply by searching the database of pre-sampled state transitions. For the current acrobot position, an n-level tree search is performed to determine which torque sequence will get it closest to the goal position. The first torque in the selected sequence is then applied.

Candidate torques and their results are evaluated with a cost function that takes the two joint angles and velocities and compares them with where the acrobot should ideally be (which is typically a balanced position with zero velocity such as S=(π/2, 0, 0, 0)). This function then returns a rating for the candidate torque, and the torque with the highest rating is applied to the acrobot. This is the most robot-specific part of the whole system. Experimentation is generally required to find a useful cost function since there are many different ways of comparing acrobot states. The nonlinear nature of the acrobot also presents a challenge because it is not always clear which possible ending position from the tree search is closest to the target state in terms of a subsequent path.

The cost function employed in this study relies on a set of balanced positions known as the s-curve (Figure 4). The s-curve is defined as the set of acrobot states where the center of gravity is over the point of contact with the ground. One interesting property of the s-curve is that it is possible to travel along it by keeping the center of gravity of the acrobot slightly offset to the left or right. The cost function that was designed for the acrobot rates positions very highly if they are near the s-curve then slightly increases the rating if the difference in angles and velocities between the current and ideal state is small. This has the acrobot stay near the scurve and move along it until the target state is reached. This design was chosen because most stable positions of interest can be reached from the s-curve without passing through unbalanced regions. Other measurements, such as the difference in mechanical energy and difference in angular momentum between the current and target state was also considered for use in the cost function but found to be less useful.

The specific weightings used in the cost function in this study were found using a hill climbing algorithm. Choosing whether to pay more attention to elbow angle or shoulder velocity can be hard to do manually, but different sets of weights can easily be tried on simulated acrobots until successful ones are found.

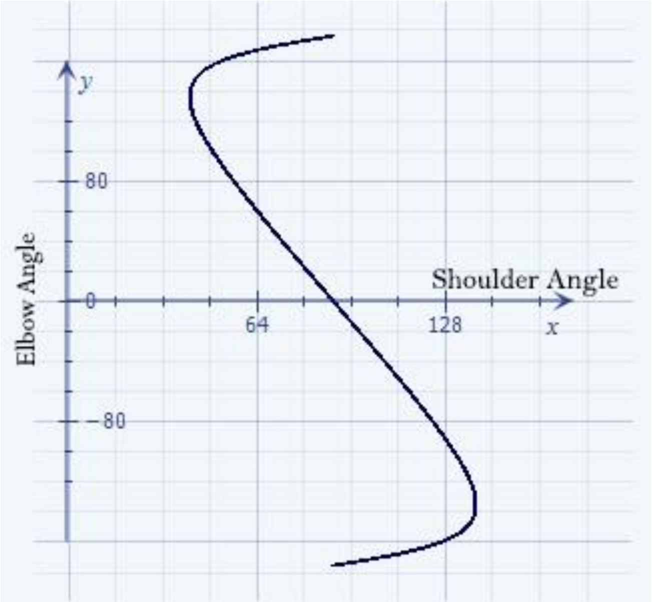


Figure 4: An example s-curve. This is the collection of points where the acrobot is balanced.

5 Lookup Table Generation and Continuous Improvement through Operational Feedback

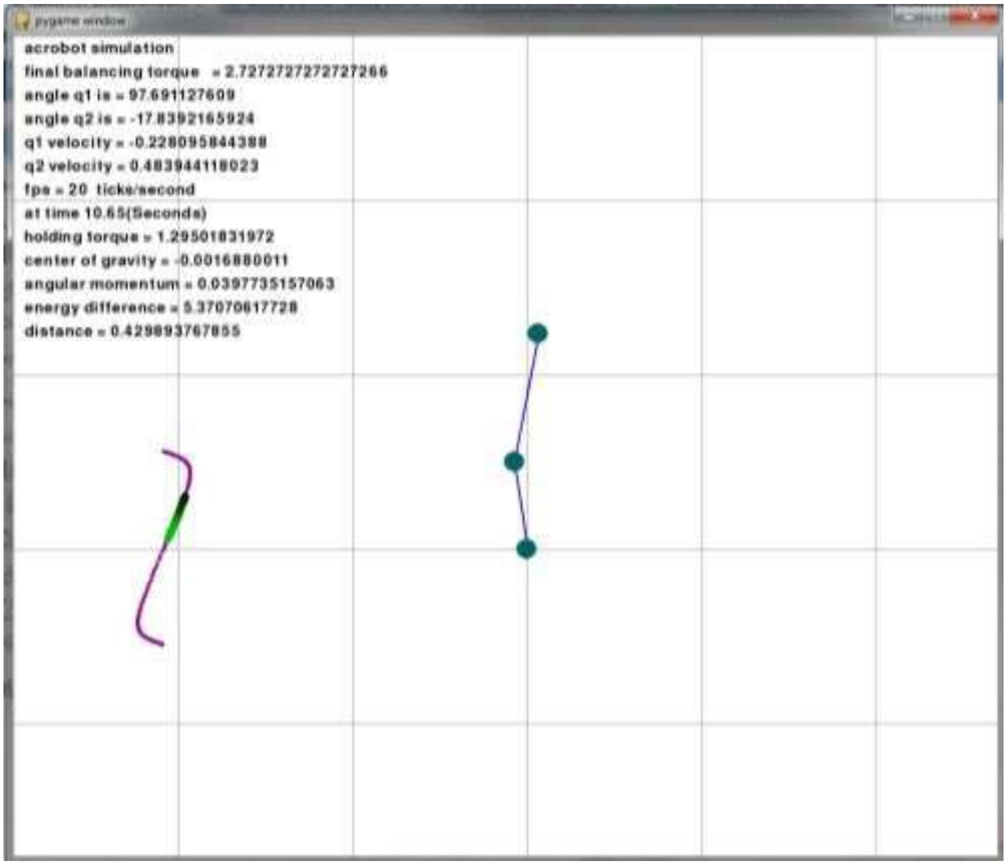
One of the most challenging parts of our approach is generating the lookup table. Generating the data required in a simulation doesn't pose a challenge because simulations can be started from arbitrary states. This means that a separate simulation can be run and recorded for each data point in the lookup table. Physical robots pose a significant challenge because manually positioning an acrobot for each index that needs to be filled is time prohibitive, especially when non-zero initial velocities are needed. The approach we took instead was to record the acrobot's movements then process that data to approximate values for the lookup table indices. After each torque is applied, the resultant position is added to the already stored data for that state/torque. This provides continuously more accurate results as the stored data incorporates increasing numbers of observed results. This also means that if the physical properties of the acrobot change then the current lookup table will be updated without human intervention.

However, incorporating state changes into the database is not a trivial task. Simply averaging the actual result data coming from the hardware is unadvised since the reported positions are continuous and must be quantized before being stored. This is a problem because incoming data isn’t uniformly distributed across the state space and so simple averaging will return skewed results. This doesn’t pose an issue while working in simulations because as noted initial positioning can be specified at will, but lookup tables generated from actual robots can only be created from response data. We plan on storing all previous responses then generating lookup table indices by interpolating between nearby collected data points, but this system has not been implemented yet.

6 Current Status and On-going Work

As of writing, the controller has been successfully implemented and can balance a simulated acrobot in many stable positions, move between stable positions, and move to a stable position from an unstable one (Figure 1). We are currently working on interfacing the controller to actual acrobot hardware (Figure 2). This has the range of expected challenges such as signal noise, sensor response time, and operating in real-time, but should not require significant change to the controller itself. The control system is the heart of the effort and is common between the simulator and the hardware.

In addition, on-going development is focusing on improving the cost function for points not on the s-curve and further optimizing the code so that more extensive searches can be performed in real time. We are also considering dynamically changing the weights used in the cost function depending on the acrobot’s current state.



```
acrobot simulation
final balancing torque  = 2.7272727272727266
angle q1 is = 97.691127609
angle q2 is = -17.8392165924
q1 velocity = -0.228095844388
q2 velocity = 0.483944118023
fps = 20  ticks/second
at time 10.65(Seconds)
holding torque = 1.29501831972
center of gravity = -0.0016880011
angular momentum = 0.0397735157063
energy difference = 5.37070617728
distance = 0.429893767855
```

Figure 1 (repeated): The simulation system in operation

7 References

[1] Boone, G., "Minimum-time control of the Acrobot," in *Robotics and Automation, 1997. Proceedings., 1997 IEEE International Conference on* , vol.4, no., pp.3281-3287 vol.4, 2025 Apr 1997

[2] Morteza Azad, and Roy Featherstone. "Angular Momentum Based Controller for Balancing an Inverted Double Pendulum." Romansy 19 - Robot Design, Dynamics and Control 544th ser. (2013): 251-58. Springer. Web. 6 July 2014.

[3] Nils A. Andersen, Lars Skovgaard, Ole Ravn, Daniela Rus, and Sanjiv Singh. "Control of an under Actuated Unstable Nonlinear Object." Vol. 271. N.p.: Springer Berlin Heidelberg, 2001. 481-490. Lecture Notes in Control and Information Sciences. Web. 3 Nov. 2015.

[4] Russ Tedrake. Underactuated Robotics: Algorithms for Walking, Running, Swimming, Flying, and Manipulation (Course Notes for MIT 6.832). Downloaded in Fall, 2014 from http://people.csail.mit.edu/russt/underactuated/

[5] Wiklendt, Lukasz, Stephan Chalup, and Rick Middleton. "A Small Spiking Neural Network With LQR Control Applied To The Acrobot." Neural Computing & Applications 18.4 (2009): 369. Advanced Placement Source. Web. 30 Oct. 2015.