Technical Milestone Report

F-CER54-3: Using machine learning to control an inverted double pendulum using camera feedback

Nicholas Capel (nrjc2)
Supervisor: Prof. Carl Edward Rasmussen

January 18, 2017

Abstract

The aim of this project is to implement the control of an inverted double pendulum in the PILCO framework using camera feedback. Work to date has involved MATLAB computer simulation experiments to understand how added delay and uncertainty about states can adversely affect training of the model and controller parameters in the PILCO framework. Additional work was done to determine an upper bound on the system delay introduced by processing times and video lag. Going forward, a physical system of the double pendulum will be implemented. Furthermore, experiments will be conducted on a SIMULINK model to better understand the problem from a control perspective, and the camera experiments will be refined.

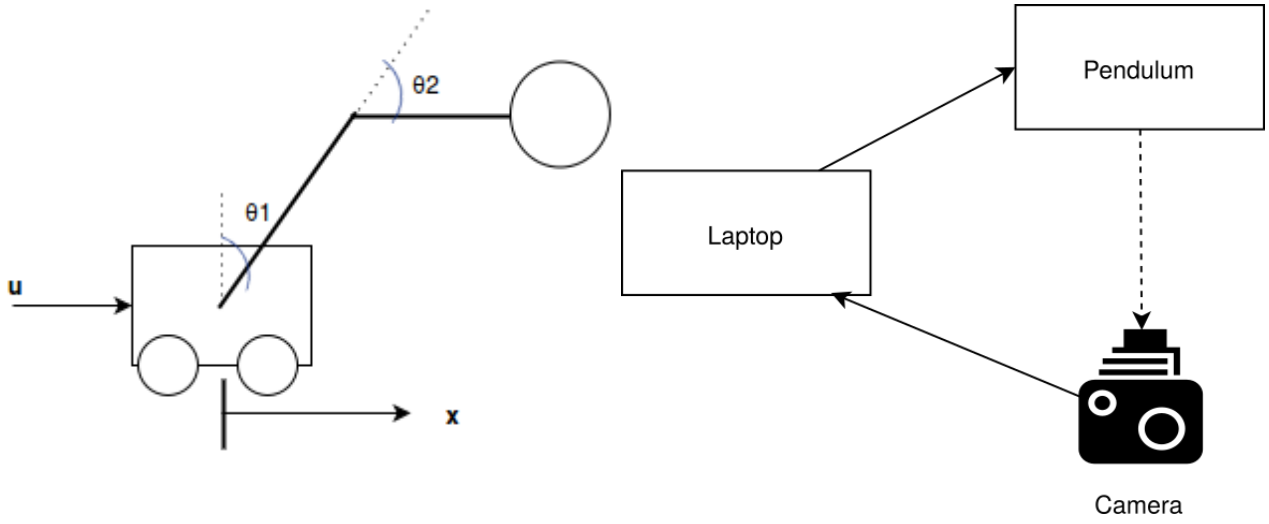
1 Introduction

The inverted double pendulum problem is a control problem.
ope

Figure 1: Diagram of setups

(a) Problem setup

(b) System setup



The state of the system is described by x.

$$x = [\dot{x_t}, x_t, \dot{\theta_{1(t)}}, \theta_{1(t)}, \dot{\theta_{2(t)}}, \theta_{2(t)}]^T$$

The objective of the control problem is to discover a controller to minimize the expected cost function over an infinite time horizon. In this project, the PILCO framework is used to both discover a model of the system, and to design a controller to stabilize the system [1].

In reality, this corresponds to the pendulum performing two tasks:

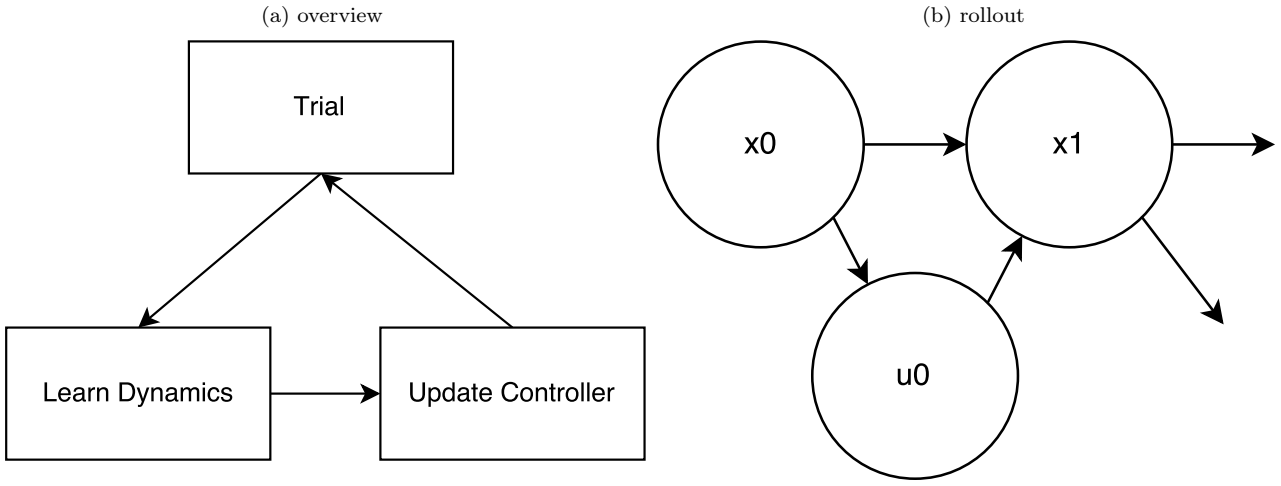
1. Swing up: The pendulum is initialised at the position $\theta_1 = \pi$, $\theta_2 = 0$. The control system has to swing the pendulum up such that $\theta_1 = 0$
2. Stabilize: The controller has to control the system state such that $\theta_1 = 0$, $\theta_2 = 0$

In this project, another layer of difficulty is added. The current state vector of the system, x, cannot be directly read from sensors, but must be inferred from a video feed. Thus, the apparatus is set up as shown in Figure 1b.

2 Motivation and Theory

The classical approach to the Inverted Double Pendulum problem involves writing down the Newtonian equations and making a first order approximation about the equilibrium point. However, this model-based approach fails when the linear approximation breaks down, and it is difficult to exactly characterise this point of failure. Thus, successful approaches to the problem of both swing up and stabilization involve defining multiple equilibrium states that can be linearized, and designing a controller to successfully transition between these states[2].

Figure 2: PILCO algorithm



The PILCO approach is to use a data-based approach to learn a Gaussian Process model that describes the transition from $x_t \rightarrow x_{t+1}$. The Gaussian Process can model both the transition between the two states, as well as give an uncertainty estimate about the transition. By propagating the state forward in time, as in Figure 2b, a cost function can be made:

$$c = \sum_{t=0}^{T} E[f(x_t)]$$

We can thus perform gradient descent over the cost function, which is parameterized by the controller parameters, to obtain an optimal controller to perform the stabilization task.
In each iteration of the PILCO algorithm (Figure 2a), a GP dynamics model is learned using all previous data, and a cost function is minimized to discover the optimal controller policy. After which, the controller policy is applied to a real system (a rollout), and the algorithm repeats.
Stabilization of the inverted double pendulum has been attempted before, without success [3]. This is becasue noise and uncertainty about the pendulum state has significantly affected the ability of the PILCO algorithm to learn both the controller and system dynamics.

3 Progress and Results

3.1 Computer Simulations: Stabilization About Equilibrium Point

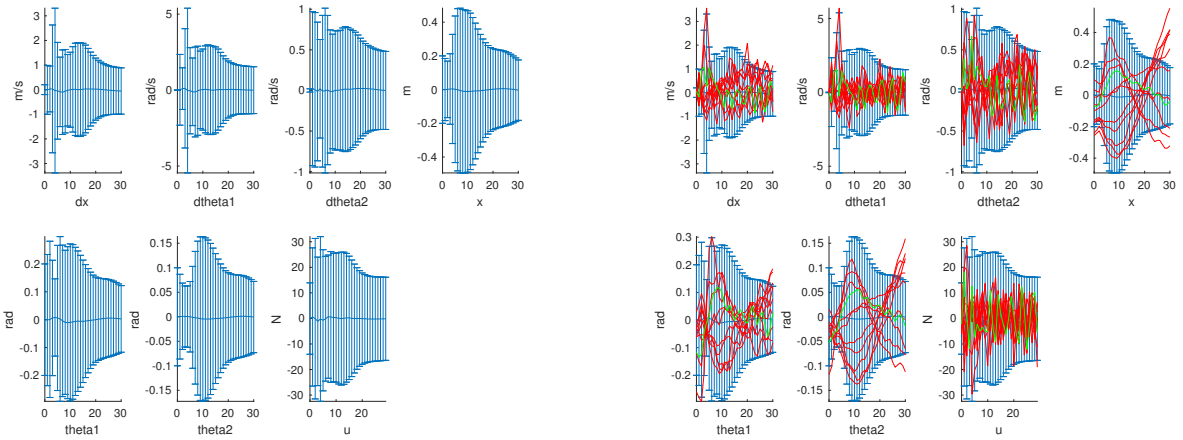
The first objective is to discover how delay and noise affect the tractability of the inverted double pendulum problem. In this series of experiments, a linear policy model was chosen for the control, and the initial state of the system is set such that $\theta_1 \approx 0$, $\theta_2 \approx 0$.

A computer model of the double pendulum was built by simulating Newton's equations in MATLAB. PILCO is then used on the simulated double pendulum, initialized with varying levels of noise and delay, and the results are analysed to understand if the PILCO control problem is tractable.
In previous experiments [3] with the PILCO algorithm, the controller was able to perform swingup, but was unable to stabilize the penduli about the equilibria. Hence, it is appropriate to investigate the failure of the PILCO algorithm about the $\theta_1 \approx 0$, $\theta_2 \approx 0$ point.
A linear policy model was selected because it is the simplest controller that can achieve the task, and is used in traditional control theory approaches [2].
The Figure below represents a typical result of an attempted PILCO stabilization:

Figure 3: GP predictions

(a) Base

(b) With rollouts



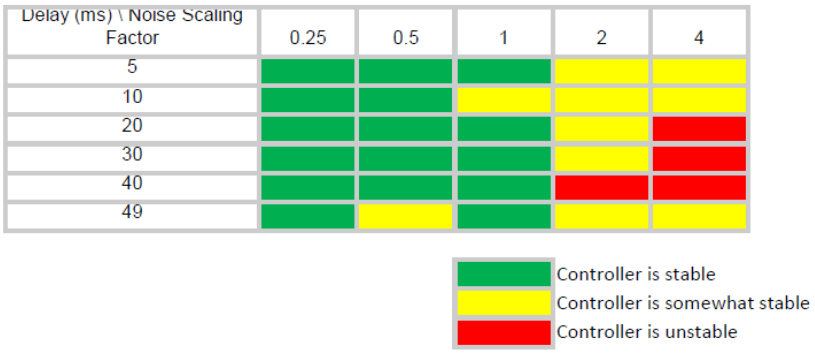
Figures 3a and 3b show how the value of a given state variable is predicted to vary with time.
The blue mean line (Figure 3a) represents the predicted trajectory of the inverted double pendulum given the controller.
The blue bars represent the 95% confidence interval of the trajectory as predicted by the model.
The green line (Figure 3b) represents a rollout that is observed by the PILCO framework.
The red lines (Figure 3b) represent simulated rollouts. These rollouts are drawn for display purposes, to evaluate the performance of the PILCO framework, but the results obtained are not fed in to the training of the GP.
After plotting the results of the PILCO experiments, the next step is to find a method to evaluate the performance of the PILCO framework. The PILCO framework can fail in two ways:

1. Model error: The framework learns an incorrect dynamics model.

2. Control error: The framework learns the correct dynamics model, but cannot discover a controller to control the unstable system.

If the control problem is tractable, two things should be observed. Firstly, as the number of steps increases, the predicted trajectory should be centered on the targeted state (0, in all cases). Furthermore, the 95% confidence interval should shrink as time increases. This demonstrates that the model predicts that it can keep the state of the double pendulum bound. Hence, there is no control error.
Secondly, the trajectories of the simulated and actual rollouts should be well described by the PILCO framework.
In an initial series of experiments, single trials were conducted for various delays and noise scaling factors. To model random noise, the computer setup corrupts the observation of each state by sampling from a Gaussian centered about the true value with a variance. The noise scaling factor scales this variance to model different magnitudes of noise.

Figure 4: Stability



The results are promising, suggesting that the PILCO algorithm is more strongly affected by noise than by delays. However, the analysis of the stability is crude: manual observation and classification of the trajectories is used to determine whether a controller is stable, somewhat stable, or unstable.
The experiment was then repeated 4 times, and a simple algorithm was used to quantify the stability of a single PILCO trial. A trial is initially assigned a stability rating of 0 (the most stable). When certain conditions are fulfilled, penalty points are added to the stability rating.

Table 1: Stability Rating

Condition (evaluated for state variable θ_2)	Penalty	Reasoning
$\sigma(T_{max}) > \sigma(T_{max-1}) > \sigma(T_{max-2})$	1	System state unbound $\longrightarrow$control error
$\theta_2(actual) > \theta_2(pred) + 2\sigma$ or $\theta_2(actual) < \theta_2(pred) - 2\sigma$ for $> 3/10$, $< 7/10$runs	1	Rollouts not well described by model $\longrightarrow$model error
$\theta_2(actual) > \theta_2(pred) + 2\sigma$ or $\theta_2(actual) < \theta_2(pred) - 2\sigma$ for $> 7/10$runs	2	

Thus, an average stability rating can be computed for all 5 runs.

Figure 5: Average stability rating over 5 runs

Delay (ms) \ Noise Scaling Factor	0.25	0.5	1	2	4
5	0.75	0	0.5	0.75	0.75
10	0	0	0	0.25	1
20	0.25	0.25	0.5	1	1.25
30	0.25	0	0.75	1	1
40	1	0.75	1	1	1.75
49	0.5	1	1	1.5	1.25

Once again, the trend identified in the original experiment does not disappear: the PILCO algorithm seems to be more strongly affected by noise than by delays.

3.2 Camera Experiment

An experiment was conducted using a program written in Java to estimate the magnitude of the camera delay.

1. Camera and LED are placed in dark room.
2. LED is turned on, timing begins.
3. When camera observes an increase in brightness, stop timing.

This experiment aims to discover the minimum time taken for an output of the system to propagate to the input. Hence, the time recorded in this experiment should serve as an upper bound on the size of the delay. Repeating the experiment, it appears that:

$$t_{delay}(max) = (307 \pm 60)ms$$

Originally, it was hypothesized that the delay is on the order of 30ms. However, the delays obtained in this experiment were a full order of magnitude higher. This is most likely because the Java code that implements the experiment invokes a very expensive system call which can be removed. Hence, the actual delay will probably be far lower.

4 Future Plans

Figure 6: Project Timeline

	Lent				Easter Holiday		Easter			
	1-2	3-4	5-6	7-8	First half	Second half	1-2	3-4	5-6	7-8
Camera Experients										
Control Theory										
Variance Experiments										
Write OpenCV image filter										
Write physical controller										
Create matlab-C interface										
Exam										
Implement Stabilization Algorithm										

4.1 Control Theory Perspective

A model will be created in SIMULINK to model an optimal linear controller for the inverted double pendulum as well as the system itself. Experiments will be conducted with varying time delays and noise levels in this system to understand if the control problem is tractable using the best methods under noisy conditions with delay.

4.2 Further Camera Experiments

The Java code implementing the camera experiments will be rewritten to remove the expensive system call. More experiments will be conducted with the camera system to obtain a more accurate delay. If the delay still remains extremely high, the experiment will be redone in C++ / C (which the actual system will be written in), to increase the speed of processing and to reduce the size of the delay.

4.3 Variance Experiments

Initial experiments suggest that the PILCO algorithm is more strongly affected by noise than by delays. Given that, further experiments will be conducted to discover which of the 7 state variables x will most severely affect the stability of the PILCO algorithm. To this end, the simulation experiment will be conducted. However, unlike the original, instead of introducing a noise scaling term to scale the noise corrupting all 6 state variables simultaneously, the total variance will instead be kept constant and only the distribution amongst the 6 state variables will vary.

4.4 Physical Implementation

After this is done, the C++ code will be written, calling on OpenCV libraries in order to implement the PILCO algorithm for a physical system. If it is discovered that delays or noise levels are too high to implement a system controlled using visual feedback, the camera will be replaced in favour of either a faster camera, or for actual sensors on the joints of the physical system.

References

[1] Deisenroth, Marc, and Carl E. Rasmussen. "PILCO: A model-based and data-efficient approach to policy search." Proceedings of the 28th International Conference on machine learning (ICML-11). 2011.

[2] Yamakita, M. A. S. A. K. I., et al. "Robust swing up control of double pendulum." American Control Conference, Proceedings of the 1995. Vol. 1. IEEE, 1995.

[3] Kukla, M. M. (n.d.). Learning to control double inverted pendulum with vision feedback.