Machine Learning for Control

by

Rodrigo Queiro (DOW)

Fourth-year undergraduate project in
Group F, 2010/2011

I hereby declare that, except where specifically indicated, the work submitted herein is my own original work.

Signed: ______________________________ Date: _______________

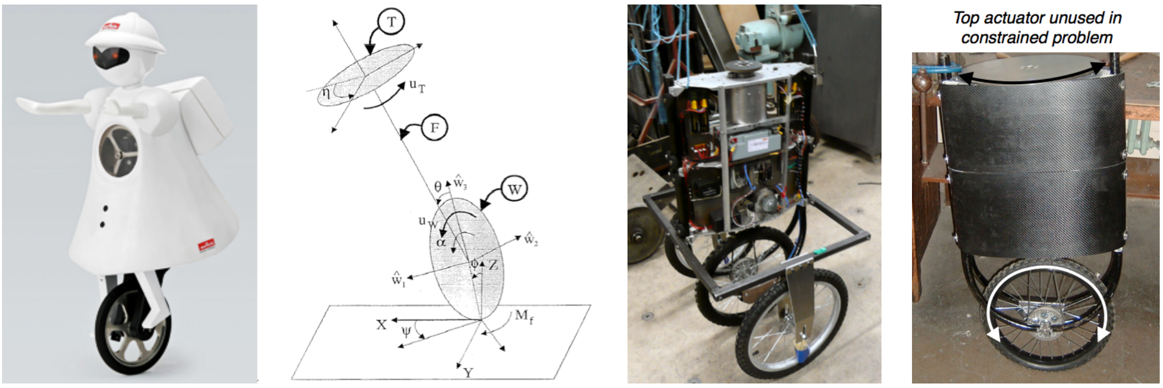
Technical Abstract

technical abstract...

Contents

1 Introduction

Balancing a unicycle is a very challenging task for a human rider. Many attempts have been made to achieve this task, using a variety of models for the action of the rider. Some represent the rider as a flywheel or pendulum in the coronal plane, allowing direct compensation of falling to the side [1, 2], as in Figure 1(a). Other use a more realistic (and challenging) model of a flywheel in the horizontal plane [3, 4], as in Figures 1(b), but none of these have reliably balanced a real unicycle.



(a) Murata Girl (b) Naveh's Model (c) 2D Problem (d) 3D Problem

Figure 1: Different balance problems

In 2004/2005 Mellors and Lamb [5, 6] built a robotic unicycle, shown in Figure 1(d), intending to design a controller to balance it. However, they were only able to complete the construction of the unicycle. In 2007/2008, D'Souza-Mathew resumed work, replacing a wheel sensor and attempting to design a controller. He simplified the problem by removing the ability to fall to the side, reducing it to 2D dynamic control: the inverted pendulum (from now on referred to as the 2D system). This is shown in Figure 1(c). He was unable to balance the unicycle due to hardware problems.

Next, in 2008/2009 Forster analysed the dynamics of both the 2D problem and the unrestricted 3D unicycle [7]. Again, hardware problems prevented him from balancing the 2D system, and although he designed a controller for the 3D unicycle, it was not even tested in simulation. Given the simplicity of his approach compared to those of Vos and Naveh [3, 4], it appears unlikely to work.

One thing all these approaches have in common is that their first step is a series of simplifying assumptions about the dynamic system. They ignore the non-linearities like motor dead-zones and wheel friction that are present in any real-world system, and attempt to design a controller to stabilise the idealised system. In many cases, this approach is very successful. However, D'Souza-Mathew and Forster found that their model was invalid since the unicycle's motor drive didn't react faster enough. Vos and Naveh had to use complex, approximate techniques to model the unicycle.

An alternative “intelligent” approach to control involves learning the dynamics of the system directly, instead of relying on assumptions and mechanical analysis. Various methods for this have been used, but many require prohibitively large amounts of data from the system. One method due to Rasmussen and Deisenroth, known here as Reinforced Model Learnt Control (RMLC), achieves unprecedented data efficiency, and has been successfully used to stabilise a computer simulation of a 3D unicycle [8, 9].

In 2009/2010, McHutchon successfully applied RMLC to the 2D system [10]. However, since he had to make significant changes to the unicycle hardware and software to achieve this, he did not have time to attempt to balance the 3D unicycle.

The principle objective of this project is to apply RMLC to the unrestricted 3D unicycle. This includes the solution of problems identified by McHutchon, as well as other problems identified during the project.

2 Reinforced Model Learnt Control

The main technique in this project is Reinforced Model Learnt Control, diagrammed in Figure 2. At its core, it assumes that the system (in this case, the unicycle) can be modelled in discrete time as:

$$\boldsymbol{q}[n+1] = \boldsymbol{f}(\boldsymbol{q}[n], \boldsymbol{u}[n])$$

In this equation, $\boldsymbol{q}[n]$ is the state of the system at time n, and $\boldsymbol{u}[n]$ is control input at time n. In the case of the unicycle, $\boldsymbol{q}$ consists of angles and angular velocities of the components of the unicycle, and the position of the unicycle. $\boldsymbol{u}$ consists of the commands sent to the wheel and flywheel motors.

This function, $\boldsymbol{f}$, is modelled as a Gaussian Process (GP). By using Gaussian Process Regression (GPR), we can estimate any continuous function from sampled inputs and outputs. When the unicycle runs, we get a series of states and control inputs that can be converted to samples of $\boldsymbol{f}$, and this allows us to use GPR to estimate $\boldsymbol{f}$ at any point. This estimated $\boldsymbol{f}$ is referred to as the **dynamics model**.

By successively applying $\boldsymbol{f}$ to an initial distribution of possible starting states, we can estimate, with confidence bounds, a distribution of states over some finite horizon. This is referred to as **simulation** of the system. Then, a **loss function** is applied to the state distributions—this might find, for example, the expected distance between the top of the unicycle and the upright position. Summing these losses over the horizon gives a numerical score that rates how well the dynamics model believes a given controller will balance the unicycle. This loss score penalises uncertainty as well as falling.

The gradient of the loss with respect to the controller parameters can be calculated, and this allows standard gradient descent optimisation methods to be used to find a locally optimal controller (for the estimated dynamics model). This process is shown as the right hand "Simulation & Controller Optimisation" box in Figure 2, and is referred to as **training** a controller.

Once a optimal controller has been trained on the simulated system, a **rollout** is performed on the real system ("Real World Testing" in Figure 2). This generates a log of states and control inputs, which can be converted into more samples of $\boldsymbol{f}$, improving the quality of the dynamics model and allowing a better controller to be trained.

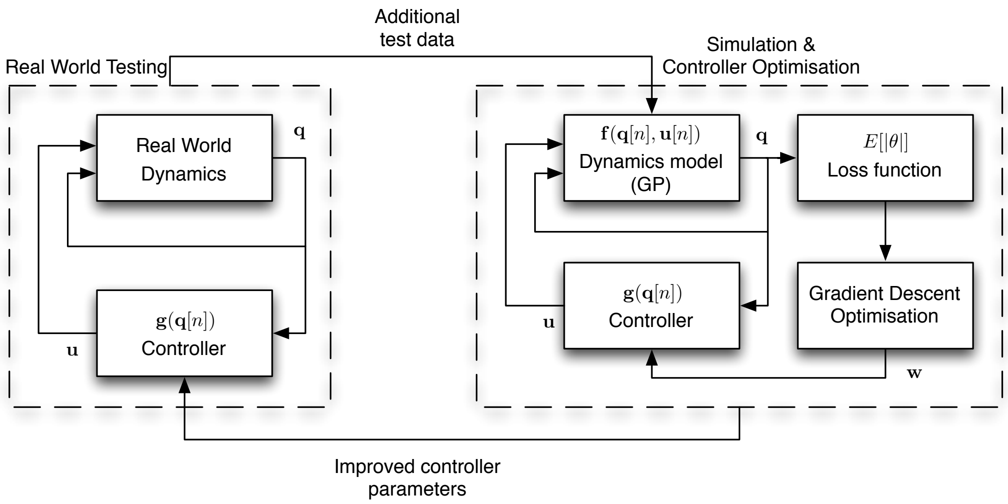


Figure 2: Reinforced Model Learnt Control

3 Apparatus and Experimental Results

4 Results and Discussion

5 Conclusions

References

[1] D. Zenkov et al. The Lyapunov-Malkin theorem and stabilization of the unicycle with rider. *Systems and Control Letters*, 46:293–302, 2002.

[2] Murata Manufacturing Co. Development of the unicycle-riding robot: Murata girl. `http://www.murata.com/new/news_release/2008/0923.html`, 2008.

[3] D. Vos. Nonlinear control of an autonomous unicycle robot: Practical issues. MIT PhD Thesis, 1992.

[4] Y. Naveh et al. Nonlinear modelling and control of a unicycle. *Dynamics and Control*, 9:279–296, 1999.

[5] M. Mellors. Robotic unicycle: Mechanics & control. CUED Master's Project, 2005.

[6] A. Lamb. Robotic unicycle: Mechanics & control. CUED Master's Project, 2005.

[7] D. Forster. Robotic unicycle. CUED Master's Project, 2009.

[8] Carl Edward Rasmussen and Marc Peter Deisenroth. Recent advances in reinforcement learning. chapter Probabilistic Inference for Fast Learning in Control, pages 229–242. Springer-Verlag, Berlin, Heidelberg, 2008.

[9] Marc P. Deisenroth and Carl E. Rasmussen. PILCO: A Model-Based and Data-Efficient Approach to Policy Search. In L. Getoor and T. Scheffer, editors, *Proceedings of the 28th International Conference on Machine Learning*, Bellevue, WA, USA, June 2011.

[10] A. McHutchon. Machine learning for control. CUED Master's Project, 2010.

A Extra Stuff