Robotic Unicycle

Final Report

Harvey Hughes

Supervisor:Professor Carl Rasmussen

May 13, 2020

Technical Abstract

The application of machine learning to calculate a control policy opposed to using control theory could prove to be highly useful when its not possible to determine system dynamics. However, success outside of simulations has been slow. A robotic unicycle is considered in this project with tests into roll limit, sampling time, phone connection and initial orientation being tested first in order to brooch the gap between simulation and real success.

a

Contents

I. Introduction

i. Project aims

The aim of this project is to create a controller that is able to balance a robotic unicycle. The controller starts off with no information about the system, and is therefore required to slowly learn what are good actions to make. This mimics the way in which humans learn new tasks as children by trying new actions and seeing if they're beneficial.

This is an ongoing project from previous years where success has been shown in simulations yet has so far been hard to achieve in real life. A major outcome of this project is the build upon previous years work and fix existing issues, in addition to solving new ones in order to bring the success of a real unicycle closer to simulations.

This project aims to demonstrate the effectiveness of implementing the PILCO algorithm [1] on more complex dynamic systems. PILCO has previously been shown to tackle simpler problems such as a cart-pole system with less data and high speeds compared to other algorithms.

A unicycle is a suitable example to test on due to the complex nature of the problem, even for human riders. Balancing on the

(a) *1m tall 1st model [6]* **(b)** *20cm tall 2nd model [8]*

Figure 1: *Previous unicycle iterations*

spot is additionally a less linear and therefore harder problem to solve then simply moving forward. This complexity allows the full extremes of the PILCO algorithm to be explored and therefore demonstrate all flaws and benefits in PILCO.

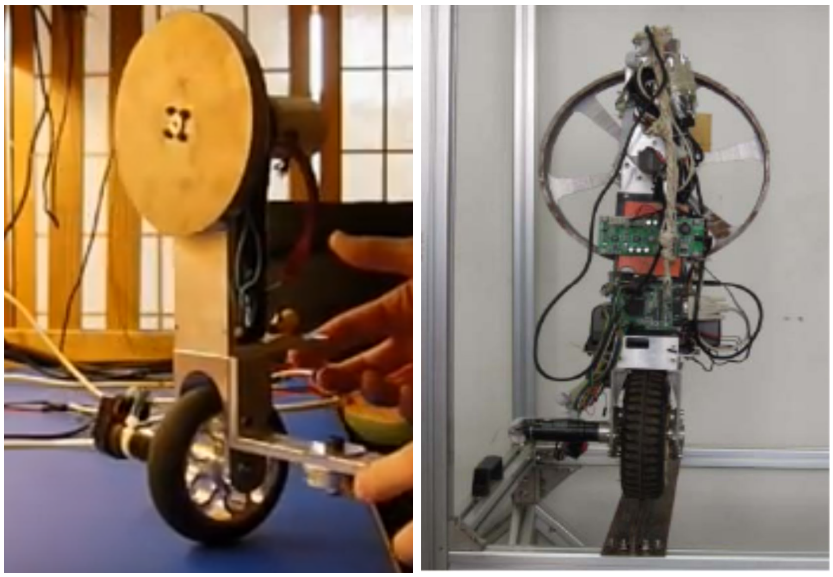
ii. Previous work

Work has been conducted on robotic unicycles within the engineering Department on and off since 2005 [2]. This unicycle was full sized at 1m, and weighed 35kg shown in figure 1a. Built by Mellors and Lamb the unicycle involved a wheel motor and horizontal flywheel motor to mimic the rotation arms can create. Having the flywheel in the horizontal plane increases the complexity over other designs with in in the vertical plane, as it couples the two motors together when a roll angle is corrected.

The problem was originally being solved using control theory approaches. This involved complex analysis of the dynamics in 2D and 3D conducted in 2007-2009 by D'Souza-Mathew [3] and Forster [4]. Some success was demonstrated at balancing the unicycle. However, simplifications in the analysis resulted in non-linearities present in real life to not be considered accurately. In 2010 Mchutchon [5] applied the RMLC (Reinforced Model Learnt Control) algorithm developed by Rasmussen and Deisenroth to the unicycle, and acived balancing of a unicycle with stabilisers. In 2011 building upon Mchutcons work Queiro [6] and Douglass [7] were able to modify the unicycle to balance unrestrained for up to 5 seconds.

The size of the unicycle posed serious safety risks, both in terms of replacing large components and the need to limit the unicycle during tests. One way this was done was by attaching to bike racks [3] or suspending the unicycle from the department atrium [6]. These methods while reducing the risk didn't eliminate it, and caused unnecessary simplifications or disturbances to the unicycle dynamics.

To increase safety a small model unicycle was constructed as in figure 1b. PILCO was also implemented on the unicycle [1]. The increased safety of the smaller unicycle allowed unconstrained motion to be tested more easily. Work began on the miniature unicycle in 2015 by Tukiainen [**?**]. Problems in balancing the unicycle were encountered at this point, with concerns about identify-



(a) *[11]* **(b)** *[12]*

Figure 2: *Other Unicycle designs*

ing the cause for problems. In 2017 an overhaul of the software was done by Wieser [8], and in 2019 an electrical overhaul by Harris [9]. This was to simplify the system and allow for more accurate troubleshooting.

iii. Examples of other unicycles

There are numerous examples of other work conducted on balancing robotic unicycles. Many of these involved a large vertical flywheel and use of control theory as seen in figure 2. Orientating the disk this way improves stability and was therefore quite successful at balancing. The improved stability is due to uncoupling the two torques, meaning that to correct a roll error just the flywheel can be used, and to correct a pitch error the wheel only needs rotating. A combination of both actions is required to fix roll errors on a unicycle with horizontal flywheel.

There has been work done on unicycles set-up with a horizontal flywheel. [13]. Vos and Von Flotow analysed a large unicycle and were able to balance to some success. This required multi region control policies to work, and therefore was very complicated.

The robotic unicycles discussed so far have

all been developed using traditional control theory. Very few unicycles have been controlled with a machine learning approach. Simulations of a 2D unicycles were successfully balanced [14] using standard keras and Tensorflow libraries. However, this approach simplifies the problem back to a much easier inverted pendulum, and didn't deal with the problems associated with real models. The lack of previous success at this particular task highlights its difficulty.

II. Theory

i. Coordinate System

In order to fully define the unicycle state at each time step seven coordinates and their derivatives are required. These coordinates are shown in figure 3 giving the state as

$$\mathbf{x} = [\dot{x}\ \dot{y}\ \dot{\theta}\ \dot{\phi}\ \dot{\psi}_f\ \dot{\psi}_w\ \dot{\psi}_t\ x\ y\ \theta\ \phi\ \psi_f\ \psi_w\ \psi_t]^T$$

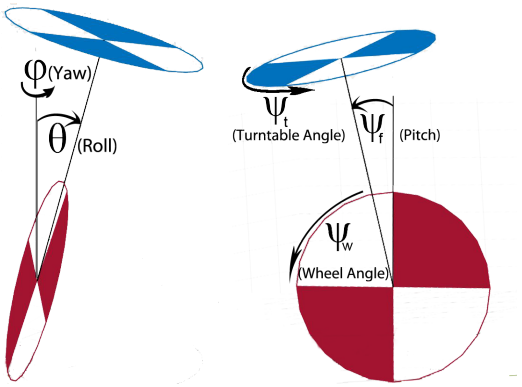


Figure 3: *Angles used to define position*

In order to define rotations in 3D to represent θ, ϕ, ψ_f correctly care must be taken to the order the rotations are applied. This is because rotations are non commutative in 3D unlike in 2D. The convention of Euler angles is used to solve this problem. [8] There are 12 orders of rotation which can be used. XYZ or 123 convention is used on the unicycle as in equation 1. Where $R_A(\alpha)$ represents a right hand rotation about axis A of α.

$$R_{XYZ}(\psi_f, \theta, \phi)\mathbf{x} = R_X(\psi_f)R_Y(\theta)R_Z(\phi)\mathbf{x} \tag{1}$$

ii. Quaternions

As discussed previously rotations in 3D are difficult to combine due to their non-commutative nature. A handy way of dealing with consecutive rotations is to use quaternions. These are an extension of complex numbers to include three imaginary parts and are written as:

$$a + b\mathbf{i} + c\mathbf{j} + d\mathbf{k}$$

The following properties allow quaternions obey the same rules as 3D rotations:

$$\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = \mathbf{ijk} = -1$$

A rotation about axis (x,y,z) by α can be defined using the following unit quaternion [**?**]

$$\mathbf{q}_{rot} = cos(\frac{\alpha}{2}) + sin(\frac{\alpha}{2})x\mathbf{i} + sin(\frac{\alpha}{2})y\mathbf{j} + sin(\frac{\alpha}{2})z\mathbf{j}$$

$$\mathbf{q}_{new} = \mathbf{q}_{old} \cdot \mathbf{q}_{rot}$$

iii. Optimal Control

iv. PILCO

In policy optimisation an algorithm called PILCO [1] will be used. This method will optimise a policy to decide the best action $\mathbf{u}$ given the unicycles state. To determine optimum actions the expected loss over a horizon is evaluated under the current policy, as in equation 2.

$$\begin{aligned} \pi^*(\mathbf{x}) &= argmin \sum_{0}^{t} c(\mathbf{x}^{(i)}) \\ \mathbf{u}^{(i)} &= \pi(\mathbf{x}^{(i)}) \end{aligned} \tag{2}$$

v. Gaussian Processes

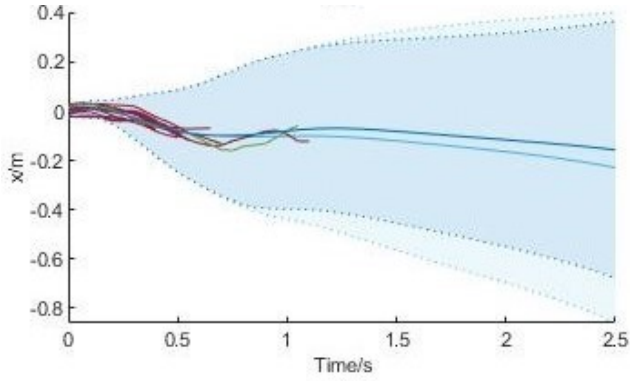
A Gaussian process is used to model the dynamics and gives a range of possible functions which fit observed data in a non parametric way as to not introduce model bias. A Gaussian process is a generalisation of a multivariate Gaussian to infinitely many variables. As a result a mean and covariance function are used to define it as in equation 3.

$$\begin{aligned} f(\mathbf{x}) &\sim \mathcal{GP}(m(\mathbf{x}, k(\mathbf{x}, \mathbf{x'})) \\ m(\mathbf{x}) &= \mathbb{E}[f(\mathbf{x})] \\ k(\mathbf{x}, \mathbf{x'}) &= \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x'}) - m(\mathbf{x'}))] \end{aligned} \tag{3}$$

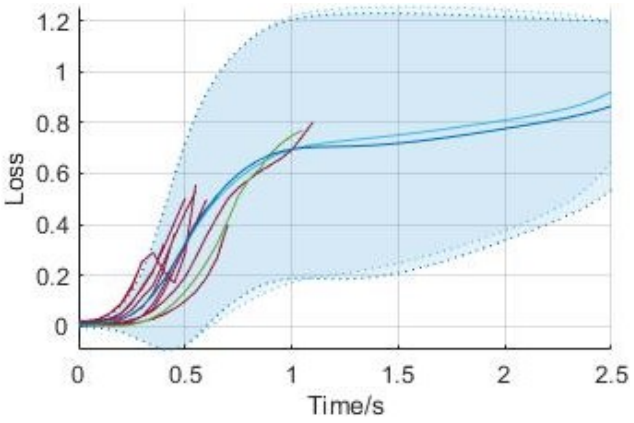
A major advantage of PILCO is the ability to retain uncertainty throughout this process. This means an initial state distribution can be propagated forward to generate a state distribution at each time step. These distributions are used to calculate the average instantaneous cost in policy evaluation. The uncertainty grows over time and comes from an initial variation in start position, measurement noise, and process noise.

Figure 4a shows predicted x position, with predictions up to 0.5s showing low uncertainty. After this the model loses the ability to predict the state accurately and the state distribution widens. A similar trend is observed for the predicted loss at each time step in figure 4b.

PILCO uses an iterative approach to optimising policy. Initially a random controller is used to get sufficient data for a dynamic model. Policy optimisation and rollouts under the new policies can then occur until the task is either learnt or terminated.



(a) *x Prediction*



(b) *Loss prediction*

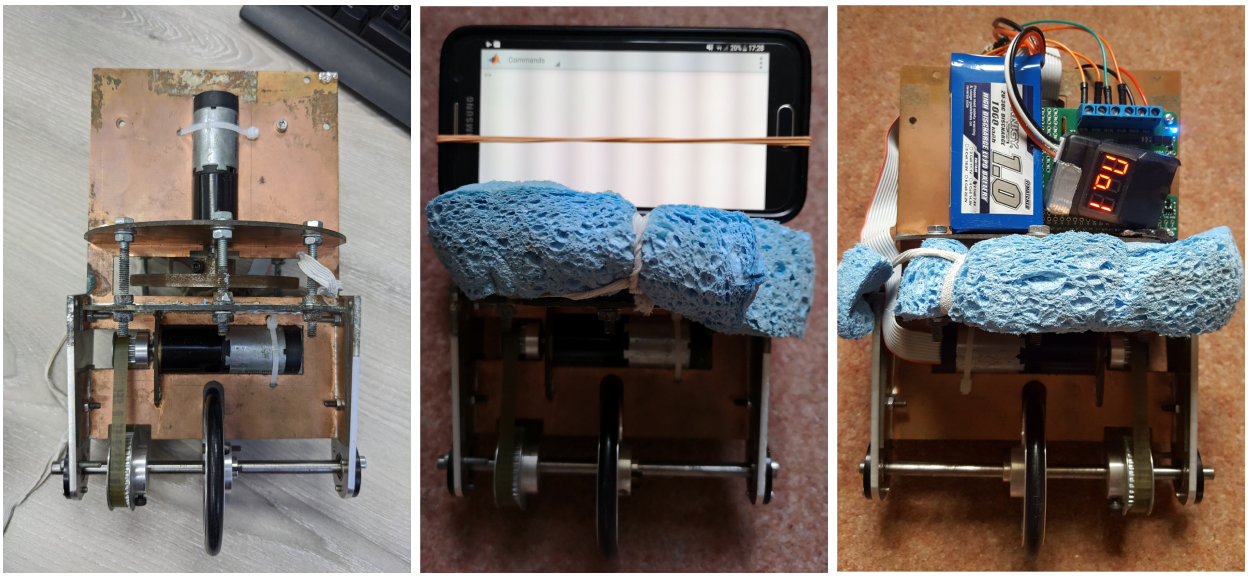
Figure 4: *Predictions from PILCO approach*

from the starting position.

$$c(\mathbf{x}) = 1 - \mathcal{R}(\mathbf{x})$$
$$\mathcal{R}(\mathbf{x}) = exp(-\frac{a}{2h^2}d(\mathbf{x})^2 - \frac{\phi^2}{2(4\pi)^2} - \frac{\theta^2}{\theta_{max}^2}) \quad (4)$$

vi. Cost Function

A cost function is required to evaluate the performance at each timestep. This takes the form of equation 4. Where a, h and θ_{max} represent positive constants, and d(**x**) is the geometric distance of the unicycle tip



(a) *Frame* **(b)** *Front [9]* **(c)** *Back [9]*

Figure 5: *Current set-up of unicycle*

III. Current Unicycle Set-up

i. Mechanical

ii. Electrical

iii. Software

An Android phone is used as the gyroscope and accelerometer for reading the current state. The sensor data is transmitted from the phone to a computer running the policy script before the required action is transmitted to a raspberry pi to controls the motors.

IV. Initial Simulation Results

During previous years the Unicycles performance on the real model was relatively poor. A mechanical redesign was considered to fix some of these issues. In order to evaluate the performance of this redesign and additional factors simulations were conducted as they allowed a quick and easy way to alter variables.

i. Test Procedure

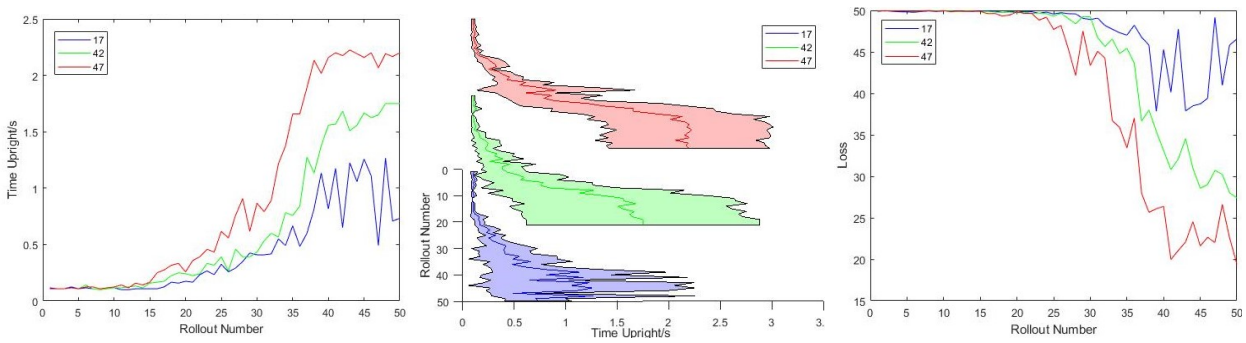
Two different measures will be used to determine success. The first being time upright as more rollouts occur. This is to determine any success at balancing the model has. The second is total rollout loss in order to see if success is achieved by staying near the origin instead of staying upright.

ii. Roll Limit

The first factor that would determine the benefit of a large mechanical redesign is the roll limit. In the unicycle model the roll angle cannot exceed 17 degrees. This is thought to inhibit learning as initially the system can't balance and will hit this limit within very few time steps. This could mean the model has insufficient data to successfully learn a policy. Futhermore as the last measurment is removed in real trials due to high accelerations the data available is reduced more. This reduction will have a larger effect on small roll limts and can cause failure to appear to happen at a low cost. Which could in turn inhibit learning more.

Several designs were sketched to change the roll limit. 42 degrees could be achieved with a new chassis and the same gears, and 47 degrees by changing an additional bracket. Simulations of these can be seen in figure 6.

After 10 random rollouts the roll limit made little difference as in figure 6a. However, when policy optimisation started then a higher roll limit improved success greatly. The change between 42 and 47 degrees appears significant which shows more progress is likely to be made by increasing roll limit further. The plateau observed at 47 degress is due to some tests consistently staying upright for the whole horizon time. When looking at rollout total loss in figure 6c the same pattern is observed with a higher roll limit reducing the



(a) *Average of six tests* **(b)** *With* $\pm\sigma$ **(c)** *Comparing against rollout loss*

Figure 6: *Simulating the effect of roll limit* θ_{max}

overall rollout loss.

The variation between simulations can be seen in figure 6b and shows greater reliability at higher roll limits, with a standard deviation of spread at 47 degrees close to being higher then the mean at 42 degrees. The spread is very important to create a reliable policy training method.

Higher roll limits could be achieved by changing other parts. 53 degrees would require a 5mm larger wheel. This change would require evaluating the effect wheel size had on performance as well as roll limit.

iii. Sampling Time

Sampling time is another important factor to consider when optimising the training situation of the real unicycle. A faster sampling time effects the following:

- Dynamics change slower and more linearly between readings, decreasing modelling complexity and improving predictions.
- A zero order hold controller is used on the motors, with torque calculated at the start of a step. As the unicycle moves this action is non optimum, therefore an error is introduced.
- More data points are collected, this can help build better models but at the expense extra computational time.
- Every policy call injects noise into the predictions from process and measurement noise. This causes a more uncertain trajectory prediction over the horizon time.

Not enough tests have been simulated

yet to fully determine the effect. However, new learning behaviours which prioritised distance to origin over just being upright were observed. Therefore showing promise that sampling time can greatly effect learning.

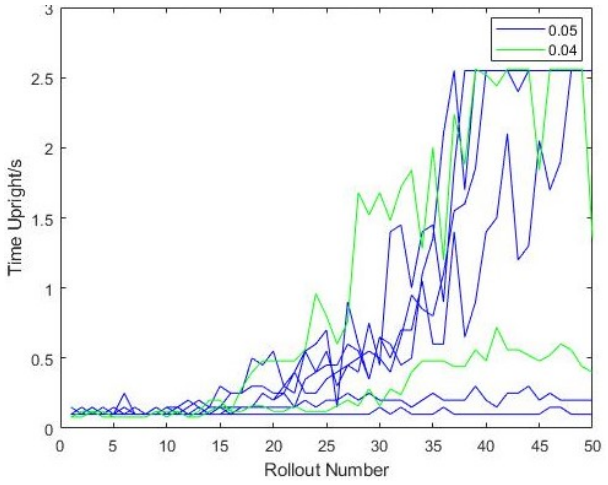


Figure 7: *Effect of sampling time change*

Figure 7 shows the results of changing timestep for a roll limit of 42 degrees. Not enough tests have been conducted yet due to the increased simulation time meaning no conclusions can be drawn yet. The lower performing trial at 0.04s did show promise as it learnt to balance in a different way to all other trials, it prioritised staying at the origin over staying upright. This shows that there is a possibility of the timestep greatly effecting learning and should be pursued further.

iv. Wheel Size

v. Future Simulations

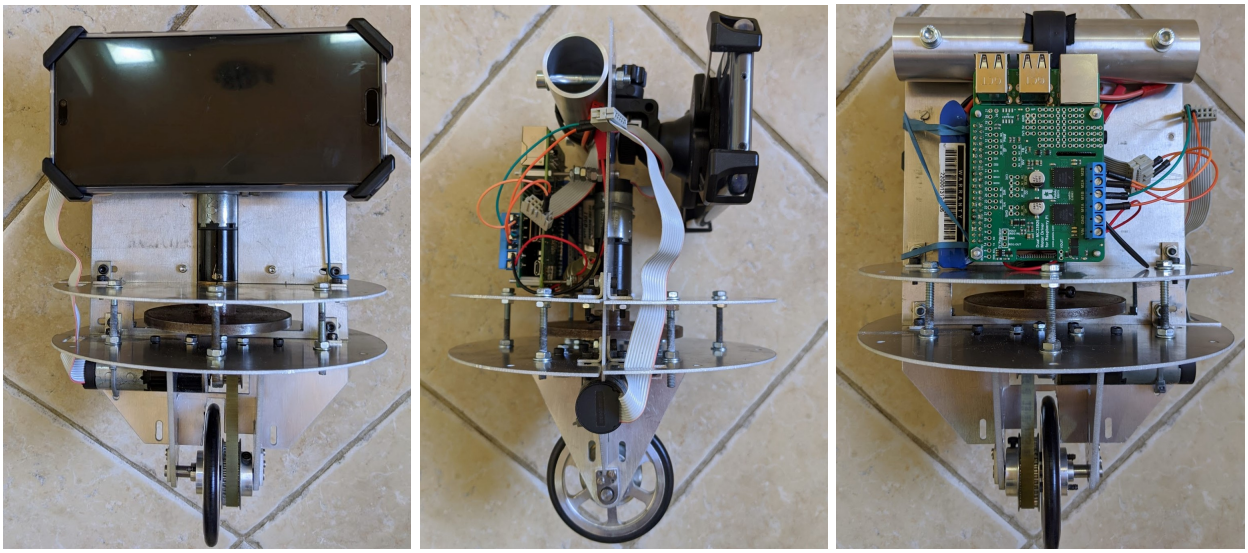
Additional factors are to be simulated to determine the variables that effect learning rate the most. These are the following:

- Higher roll limits with larger wheels.
- More sampling times.
- Strength of the cost function.
- Motor torques.

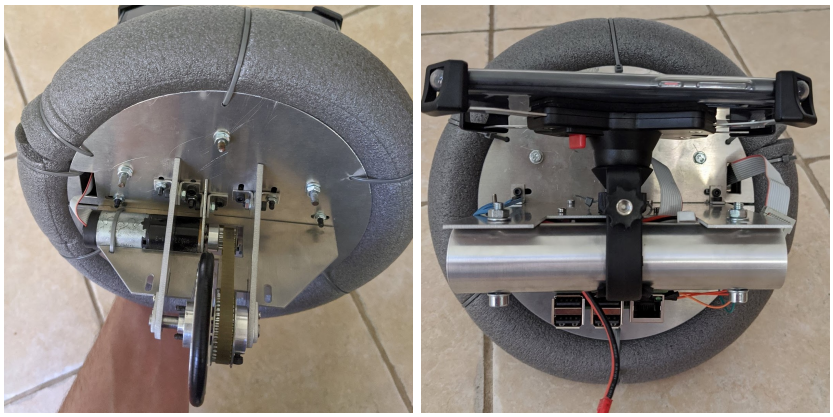
V. Mechanical redesign

i. Changes

ii. Concerns

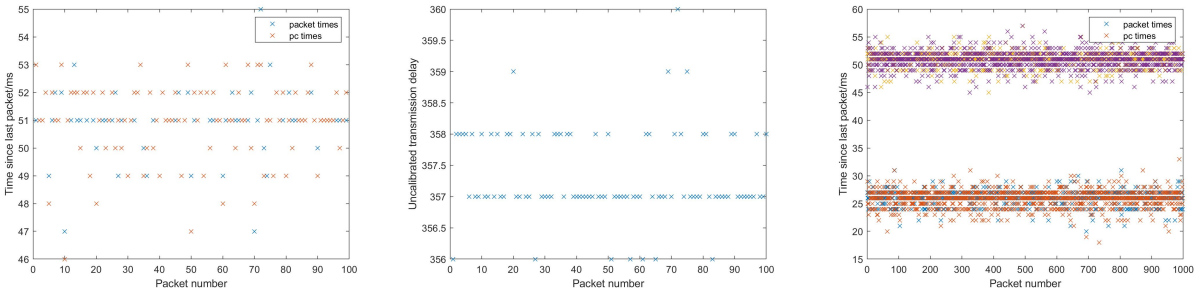


(a) *Front* **(b)** *Side* **(c)** *Back*



(d) *Bottom* **(e)** *Top*

Figure 8: *New set-up of unicycle*



(a) *Time between measurements* **(b)** *Uncalibrated delay* **(c)** *Different sampling times*

Figure 9: *Testing connecting by USB*

VI. Results

i. Phone

i.1 Connection

Previously the phone connected to the Matlab scrip on the computer using Matlab mobile and wifi. However, this feature has since been removed from Matlab mobile and delays caused by using wifi were believed to occur occasionally. This could introduce extra noise and incorrect state measurements decreasing learning ability. HyperIMU [10] was used to replace Matlab mobile. This allowed connection using wifi and USB. The transmission protocol followed was UDP using the packet layout:

$$[t, a_x, a_y, a_z, \dot{\alpha}_x, \dot{\alpha}_y, \dot{\alpha}_z \#]$$

UDP was chosen over serial communication protocols to take advantage of existing software and allow for wifi to easily be used again in the future if desired.

Tests were conducted to determine the reliability of this method. Figure 9a shows the time between consecutive measurements on both the phone side and computer side. A mean of 50.89ms and standard deviation of 1.27ms was recorded over 1000 measurements. Variation is undesirable but this is small and times are known so won't prove problematic. A similar spread was observed at lower sampling times as shown in figure 9c where a 25ms sampling time resulted in a mean of 25.88ms and standard deviation of 1.38ms. This shows that this software is suitable if faster sampling times are required.

Due to different clocks on the computer and phone true transmission delay wasn't

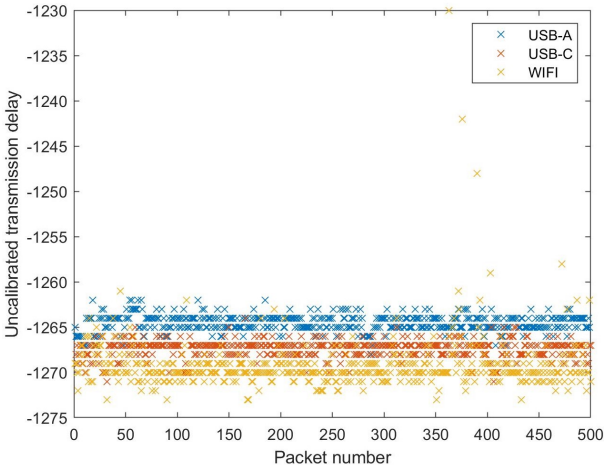


Figure 10: *Effect of changing transmission medium*

possible to measure. Figure 9b shows this uncalibrated delay. The majority of packets take one of two times to transmit with a few outliers taking $\pm$2ms. A low variation is desirable as it means actions can be applied for the correct length of time, as to not alter the trajectory away from the policy prediction.

Test	USB-A	USB-C	WIFI
1	0.99	1.05	3.02
2	1.24	1.26	1.25
3	1.10	1.02	1.84

Table 1: *Transmission delay σ in ms*

The delay over Three transmission mediums were compared in figure 10. A smaller number on these graphs corresponds to a faster transmission. It was noticed that the difference between mediums was too large, this was explained when conducting the tests in different orders. The two clock times drifted apart after each test causing the exact delays to be incomparable, therefore flipping test order resulted in the opposite order to the figure. However, the variation in delay is what is most important when deciding method as this is what can cause unexpectedly long applied actions. Wifi showed some delays of nearly a whole time step in figure 10. The standard deviation was calculated in Table 1. In test two the variation was almost identical amongst methods, but in the other tests wifi again showed greater variation. For this reason either USB connection is a better choice for use on the unicycle. The extra noise introduced by an extra cable effecting dynamics however may outweigh the transmission delay advantage and will need investigating.

i.2 Orientation

Imperfect starting positions in real rollouts can cause a constant offset in position throughout tests unless its determined at the start or learnt. Learning the offset is

possible when training however due to the offline nature of rollouts can't be currently done on new runs. Measuring the initial orientation is therefore required, which also allows accurate uncertainty measurements ensuring predictions incorporate as much information as possible.

$$\delta \mathbf{a} = \mathbf{a}_0 - R\hat{\mathbf{a}}$$

$$\begin{bmatrix} \theta \\ \psi_f \end{bmatrix} = 2 \begin{bmatrix} tan^{-1}(\frac{\delta \mathbf{a}_x}{\sqrt{\delta \mathbf{a}_y^2 + \delta \mathbf{a}_z^2}}) \\ -tan^{-1}(\frac{\delta \mathbf{a}_y}{\sqrt{\delta \mathbf{a}_x^2 + \delta \mathbf{a}_z^2}}) \end{bmatrix} \quad (5)$$

To calculate the initial orientation the roll and pitch angles between transformed accelerometer reading and $\mathbf{a}_0 = [0, 0, -1]^T$ needs to be determined. A transformation is required from the raw sensor acceleration to convert between phone and unicycle co-ordinates, this transformation being:

$$\mathbf{a} = \begin{bmatrix} 0 & -1 & 0 \\ 0 & 0 & -1 \\ 1 & 0 & 0 \end{bmatrix} \mathbf{a}$$

The unicycle has only two angular DoF initially, this means that one angle can be set to 0. For this we choose yaw as this angle only depends on the starting position and is penalised when higher. The angles are measured for two seconds before initiating the rollout to estimate initial uncertainty.

Equation 5 calculates the initial euler orientation. This can then be easily converted to quaternion form with two quaternions rotations : $(1 + 0\mathbf{i} + 0\mathbf{j} + 0\mathbf{k})R_Y(\theta)R_X(\psi_f)$. Quaternions extend complex numbers to 4D and are required to accuracy keep track of repetitive rotations due to following the same mathematical rules. [9] Quaternion form is then used for all subsequent rotations calculated using gyroscope readings. A rotation about axis (x,y,z) by α can be defined using the following unit quaternion.

$$\mathbf{q} = cos(\frac{\alpha}{2}) + sin(\frac{\alpha}{2})(x\mathbf{i} + y\mathbf{j} + z\mathbf{j})$$

$$\mathbf{q}_{new} = \mathbf{q}_{old} \cdot \mathbf{q}$$

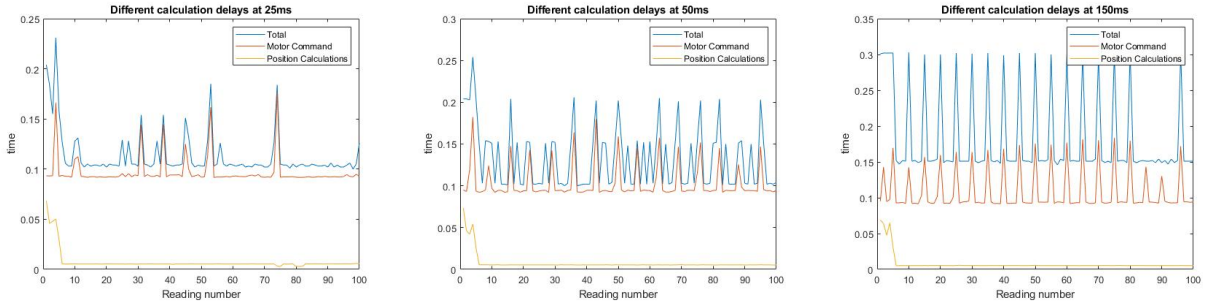
ii. Rasberry Pi

ii.1 Connection

iii. Motor

maybe something about calculating righting angles? and stiff

iv. Rollouts



(a) *25ms* **(b)** *50ms* **(c)** *150ms*

Figure 11: *Testing connecting from Matlab to Pi using Matlab system command*

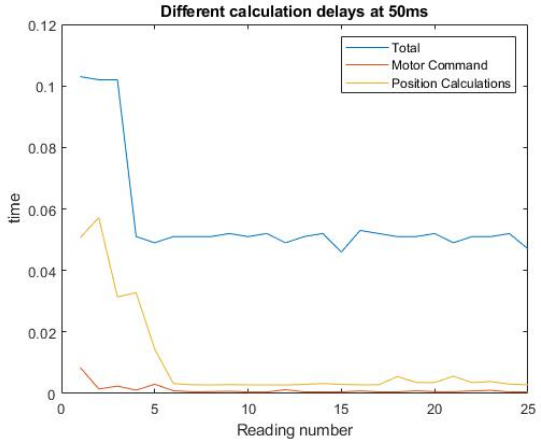
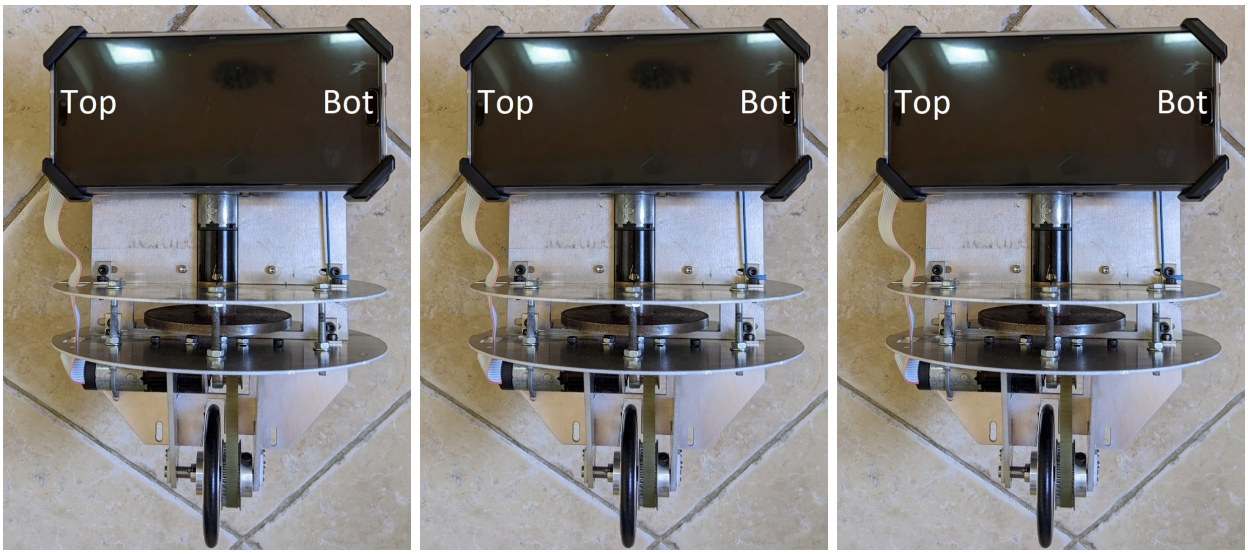


Figure 12: *Testing connection to Pi sending torques by UDP*



(a) *Phone set-up* **(b)** *HyperIMU settings* **(c)** *Release position*

Figure 13: *Steps in set-up*

iv.1 Test Procedure

Mechanical Set-Up

- Attach phone as in figure 13a with the top of the phone facing to the left of the unicycle. Ensure the phone attachment is vertical and secure
- Attach the Raspberry Pi using an Ethernet cable to the computer running Matlab
- Connect phone to computer using a USB-A/C cable
- Connect Raspberry Pi to 5V power source, and attach on board battery

Software Set-Up

- Turn on USB tethering in the phones settings (settings/network and internet/hotspot and tethering). This often works more reliably with the phone in aeroplane mode
- Find the ipv4 address of the attached computer. on windows this is found by typing ipconfig into a command window
- Open HyperIMU on the phone and go to settings and enter the ipv4 address, port 5555 and desired sample rate(this is often 1ms too slow) as figure 13b
- Save these settings and click start logging from HyperIMU

Rollouts

- In an open flat space position the unicycle, run "doitreal.m" script from the connected computer
- After initialisation a key press is required to initiate a rollout
- Make sure that the unicycle is being held still and vertical at this point, holding onto the string improves repeatability and release smoothness as in figure 13c
- The orientation is then measured for a few seconds before the motors start, at this point let go
- After the unicycle has fallen the motors should terminate, and confirmation of a successful rollout is required in Matlab
- Policy optimisation and dynamics learning then ensues, the unicycle could be disconnected during this step
- this is repeated until the desired rollouts are completed

VII. Conclusions

i. Progress

ii. Future Work

After the further simulations has been performed and evaluated the following changes will be considered in the future:

- Redesign unicycle to allow for higher roll limit, different motors, greater modularity, robustness to falls and secure phone mounting.
- Effect of independence between state variables on real rollouts.
- Adjustments to cost function.

iii. Additional rollouts

iv. Encoder

v. Exploration

vi. Mass adjustment

References

[1] M. P. Deisenroth and C. E. Rasmussen. "PILCO: A Model-based and Data-Efficient Approach to Policy Search". 2011. http://www.icml-2011.org/papers/323_icmlpaper.pdf.

[2] Mark Mellors, Andrew Lamb and J Maciejowski. 2005 roboticunicycle.info

[3] Neil D'Souza-Mathew. "Balancing of a Robotic Unicycle". MEng Thesis. CUED. 2008 roboticunicycle.info/documents/MyFinalReport.pdf

[4] David Forster. "Robotic Unicycle" MEng Thesis. CUED. 2009

[5] Andrew McHutchon, "Machine learning for Control". MEng Thesis. CUED. 2010

[6] Rodrigo Queiro. "Machine Learning for Control". MEng Thesis. CUED. 2011 https://github.com/drigz/IIB-Project/blob/master/iibproject.pdf

[7] Alan Douglass. "Machine Learning for Control". MEng Thesis. CUED. 2011

[8] Eric Weiser. "Robotic Unicycle" . MEng Thesis. CUED. 2017

[9] Arsalan Harris. "Robotic Unicycle" . MEng Thesis. CUED. 2019

[10] Ianovir: HyperIMU app ianovir.com/works/mobile/hyperimu/

[11] mhexrobot. 2012. https://edgetriggered.wordpress.com/category/unicycle-robot/

[12] G. Daoxiong, P. Qi, Z. Guoyu, and L. Xinghui, "Lqr control for a self-balancing unicycle robot on inclined plane". 2012.

[13] D. W. Vos and A. H. Von Flotow, "Dynamics and nonlinear adaptive control of an autonomous unicycle: Theory and experiment," in 29th IEEE Conference on Decision and Control, pp. 182187, IEEE, 1990.

[14] Matthew Davis. 2019 https://www.mdavis.xyz/unicycle/

VIII. Appendix

i. Risk Assessment Review