A Deep Reinforcement Learning Approach for Robotic Bicycle Stabilization

by

Shubham Turakhia

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

Approved November 2020 by the
Graduate Supervisory Committee:

Wenlong Zhang, Chair
Sze Zheng Yong
Yi Ren

ARIZONA STATE UNIVERSITY

December 2020

ABSTRACT

Bicycle stabilization has become a popular topic because of its complex dynamic behavior and the large body of bicycle modeling research. Riding a bicycle requires accurately performing several tasks, such as balancing and navigation which may be difficult for disabled people. Their problems could be partially reduced by providing steering assistance. For stabilization of these highly maneuverable and efficient machines, many control techniques have been applied – achieving interesting results, but with some limitations which includes strict environmental requirements.

This thesis expands on the work of Randlov and Alstrom, using reinforcement learning for bicycle self-stabilization with robotic steering. This thesis applies the deep deterministic policy gradient algorithm, which can handle continuous action spaces which is not possible for Q-learning technique. The research involved algorithm training on virtual environments followed by simulations to assess its results. Furthermore, hardware testing was also conducted on Arizona State University's RISE lab Smart bicycle platform for testing its self-balancing performance. Detailed analysis of the bicycle trial runs are presented. Validation of testing was done by plotting the real-time states and actions collected during the outdoor testing which included the roll angle of bicycle. Further improvements in regard to model training and hardware testing are also presented.

ACKNOWLEDGMENTS

I express my gratitude to the members of Robotics and Intelligent Systems (RISE) laboratory at Arizona State University, Polytechnic Campus for continuous support during the Bicycle project. Special thanks to Dr. Wenlong Zhang for supporting and guiding me since the beginning of my Master's degree and giving me an opportunity to be a part of the laboratory to work on the end to end development of the bicycle. I would also like to thank Dr. Yi Ren and Dr. Sze Zheng Yong for serving as committee members for the thesis project.

I would like to thank my parents for showing constant support and providing funding for my academia. This journey would not be possible without their backing. Furthermore, I would like to thank Jonathan Bush and Hansol Moon for their constant help during the development of the bike system.

TABLE OF CONTENTS

Page

CHAPTER Page

APPENDIX Page

LIST OF FIGURES

Figure Page

Figure Page

LIST OF SYMBOLS / NOMENCLATURE

1. ω Roll angle of bicycle
2. $\dot{\omega}$ Roll velocity of bicycle
3. $\ddot{\omega}$ Roll acceleration of bicycle
4. CMG Control moment Gyroscope
5. CM Center of Mass of bicycle (including the CMG)
6. c Horizontal distance between front tire and CM
7. h Height of CM over the ground
8. l Distance between front and back tire
9. M Mass of complete bicycle system
10. g Acceleration due to gravity
11. M_d Mass of tire
12. r_f Radius for the front tire
13. r_b Radius for the back tire
14. r_{CM} Radius for the Center of mass of bicycle
15. I_{dc} Axis of Moment of Inertia through the center of tire
16. I_{dl} Axis of Moment of Inertia along the handle bar
17. I_{dv} Axis of Moment of Inertia along the length of bicycle
18. $\dot{\sigma}$ Angular velocity of tire = v / r
19. v Velocity of bicycle
20. T Torque applied

CHAPTER 1

INTRODUCTION

1.1 Motivation

Two wheelers (bicycles and E-bikes) are one of the efficient means of transportation in highly crowded areas, offer better accessibility and save time of travel. People travelling via bicycles are offered great benefits and have dedicated bicycle lanes for safe travel on roads. Also, taking into consideration of the disabled and handicapped people who are unable to ride on them, there should be some means by which it could auto-stabilize itself. This also has advantages of helping kids and teenagers learning to ride a bicycle or motorbike.

But why bicycle? As the world continues to urbanize, people in cities will have to face new challenges which requires new innovative solutions for infrastructure, services and mobility. New mobility demands for a commute solution which is efficient, easy, inexpensive and reliable which would ensure faster and easier means for flow of people and goods around cities and thus minimize locomotion problems. In these scenarios, bicycles are the best option for short commute distances or as first or last mile solutions in connection to mass population transit.

Though bicycles offer great benefits but at the same time it is a pain for starters learning to ride it especially when it comes to controlling and balancing. An increased consequence of this leads to major disasters on roads and deaths. As per the Pedestrian and

Bicycle Information Centre (PBIC), most of the bike accidents and collisions happen during turning/maneuverability of bicycle. Also, as per the report of Chicago Department of Transportation "Chicago's Street for cycling plan 2020" which shows that there has been an increase in the bicycle crashes with increase in bicycle usage. Another report provided by NHTSA showed that there were 783 bicyclists killed in motor vehicle crashes.

In addition, with the advent of autonomous vehicles which is the emerging future of transportation offering V2V and I2V communication, the bicycle models can be leveraged in order to perform communications with vehicles (B2V communication) and hence increase the safety margin and reduce the collision rate. With the help of deep learning and computer vision techniques used with appropriate sensors, bicycles can be made smarter and independent to conduct its own decision-making process. One of the key aspects which makes the bike autonomous is self-balancing and it is important that while commuting the bike has maximum stability by overcoming the obstacles.

Bicycle control has been achieved before using the classical control methods and reinforcement learning techniques as shown in Cam *et al*. (2013) and Fawaz *et al.* (2019). These techniques either require robust hardware systems or cannot achieve results due to learning methods such as Deep Q learning and SARSA algorithm use discrete action spaces. Therefore, work on an improved algorithm which can support continuous control required for bicycle stabilization was required.

1.2 Focus

This thesis focuses on achieving bicycle stabilization using robotic steering mechanism which is actuated via deep deterministic reinforcement learning algorithm. This was mainly to achieve velocity balancing of bike and also serves as an extended work to previous research which achieved stationary balancing using Control Moment Gyroscope in Arizona State University's RISE Laboratory.

The implementation of algorithm was studied in detail and tested on the custom bike environment which served as a part of software testing. It was also tested on the physical bike with usage of appropriate sensors and actuators as a part of hardware testing. Detailed study of algorithm and using complex reward function will create variations to the bicycle hardware system.

1.3 Existing Work

Reinforcement Learning for balancing of bicycle was first introduced by Randlov and Alstrom. It presented a way of how bicycle would drive itself with learning from the SARSA algorithm, an on-policy reinforcement learning technique. This enabled the bicycle to balance and drive itself towards a goal using shaping. This work used a basic reward function and most important used an algorithm which uses discrete action space (Randlov & Alstrøm, 1998).

Stanford university paper uses a Deep Q learning technique for controlling a bicycle which worked on extending Randlov's work and formation of new reward function for

further improving the control process (Cam, Dembia, & Israeli, 2013). But the algorithm does not work for continuous action spaces and it claims that a robust reward functions needs to be designed for improving the bicycle controllability. Also, in regard to the punishment function, it is defined at a threshold roll angle value of 30˚ which seems to unfavorable and hence a better threshold value less than 30˚ should be used.

Some researchers from the research lab of Kyung Hee University, Seoul, South Korea worked on designing controller for self-driving bicycle using the deep reinforcement learning techniques which included using three methods and compare their results by implementing algorithms in various scenarios (Choi, 2019). But there were no signs of hardware implementation done which could truly test the performance of the algorithm on a physical bicycle.

MIT Autonomous Bicycle Project came up with a new innovative means of balancing the bicycle which offers improved user experience by bringing the convenience of mobility-on-demand-systems to bicycle sharing. The bicycle's innovative design made use of two rear tires which provides two different configurations as shown in the Figure 1.1. The first configuration is the same as regular bike in which both the rear tires remain intact and act as single tire for normal bicycle operation by user. While the second configuration provides autonomous driving operation which transforms the bike into tricycle with the rear tires separated with the help of linear actuators (Sanchez, 2020). However, it limits the bicycle for autonomous navigation only while reaching to the user and after the user has left the bicycle to drive itself to the next.

Figure 1.1: MIT Autonomous Bicycle Configurations (Sanchez, 2020)

Another research done in University of Tsukuba, Tsukuba, Japan on an experimental autonomous bicycle as shown in Figure 1.2 was developed and stabilized using a simple feedback controller . The feedback controller was designed to apply steering torque under the feedback of other states of bike. The author claims that using cubed steering angle feedback term is effective to achieve good stabilization for the experimental model. However, it does not explain much about the feasibility of the feedback term introduced in the paper (Tsubouchi, Suzuki, Koyanagi, & Yuta, 2001).

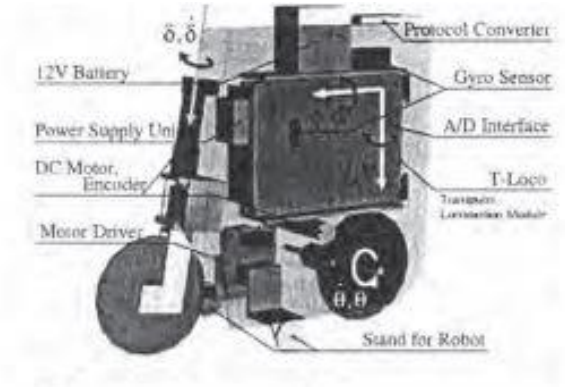


Figure 1.2: Autonomous Bicycle (Tsubouchi, Suzuki, Koyanagi, & Yuta, 2001)

Many researchers working on the bicycle stabilization also used gyroscopic stabilization in order to balance the bike. Gyroscopic stabilization is a method of balancing by using a heavy rotating flywheel which provides the reaction force in order to balance the bicycle. Though this is a really good method, but it requires robust hardware installation and a faster feedback from the sensors especially from the IMU in order to decide the orientation of flywheel. Moreover, its application is mostly found in cases of stationary balancing. As seen in Yetkin *et al.* (2014), it presents two controllers sliding mode controller and PID controller designed for the inverted pendulum and bicycle setup in order to validate the robust capabilities of Control Moment Gyroscope (CMG) used for static stabilization. Similar work can also be found in Lam *et al.* (2011).

Figure 1.3: Bicycle Setup (Yetkin, et al., 2014)

Arizona State University's RISE Lab also achieved bicycle balancing using steering control when the bicycle speed is more than 4.6 m/s or more. However, the weight to torque ratio was not sufficient to balance the bicycle analyzing from their preliminary plans. (Deng, Moore, Bush, Mabey, & Zhang, 2018)

As a step towards understanding human/bicycle interactions, Zhang and Yi (2010) presented a dynamic model and balance control of human/bicycle systems. The dynamic model in this paper considered number of assumptions, the main assumption being human modeled as a point mass on the bicycle system. Balance control design by using a coupled two sliding mode surface approach and integrating the controller with nonlinear disturbance observer was also presented followed by comparison of simulation results obtained by considering sliding mode control with and without nonlinear disturbance observer. Experimental testing and validation are still undergoing at Rutgers university (Zhang & Yi, 2010).

Arizona State University developed a self-balancing bicycle system as shown in Figure 1.4. The system includes the bicycle, a sensor coupled to bicycle, steering control assembly for adjusting the steering angle and a controller coupled to sensor and steering control assembly configured to receive the value from sensor and adjust the steering angle based on value received respectively. (Zhang , Moore, Jonathan, Mabey, & Wenhao, 2020)

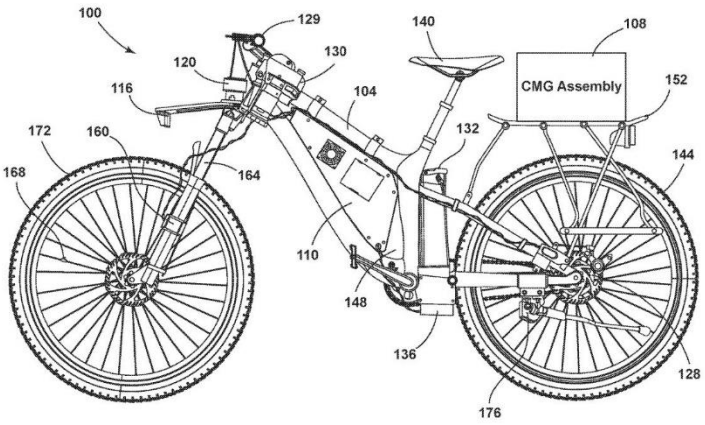


Figure 1.4: Self-balancing Bicycle (Zhang , Moore, Jonathan, Mabey, & Wenhao, 2020)

1.4 Thesis Outline

This thesis presents robotic steering stabilization of bicycle using a reinforcement learning technique. Talking about the contribution that was made in this thesis, the first part is the complete design of the algorithm for the software testing process and also improving the performance by tweaking parameters, addition of noises and actor-critic networks in order to get the trained model to be utilized for further implementations. The other main part is the hardware implementation of the deep deterministic algorithm by using the trained model and also how it could be integrated with the existing hardware with the required manipulations. The contribution to the hardware included integration of the Jetson Tx2 with the central PSoC 5 microcontroller which acts as the main component for controlling the bicycle moving components and its regularization and also with installation of safety mechanisms which included servo braking and steering angle control using radio control in case of extreme bicycle behavior conditions. Furthermore, this thesis includes the results obtained from simulations as well as hardware testing on a physical bike model. The organization of the remaining part of this thesis is as follows:

Chapter 2 presents the complete dynamic model of bicycle. This model is made by modifying the Randlov's dynamic equations and each of the terms defined are explained in detail in this chapter.

Chapter 3 covers the complete explanation of the Deep Deterministic Policy Gradient algorithm (DDPG). This algorithm is deployed on the bicycle model to achieve its stabilization. It includes the explanation for using this algorithm and why it was selected.

The complete neural structure with the complete learning process for the system is also presented.

Chapter 4 presents the complete software testing process. It includes the testing on the OpenAI pendulum environment and custom bicycle environment designed along with the results presented.

Chapter 5 presents the hardware testing on the Arizona State university RISE Lab's bicycle model. It presents the components used, algorithm workflow process and the results obtained from the testing on the bike model.

Chapter 6 summarizes the work and points out some ideas for future directions.

CHAPTER 2

DYNAMIC MODEL

2.1 Dynamic Model of Bicycle

The bicycle system has multiple components all of which contribute to the final dynamic model in an interesting manner. It is necessary to get a proper dynamic model as it plays a crucial role in getting the appropriate state values feeded to the algorithm and output the response or action directed to the actuator. In this paper, a modified Randlov dynamic model was used which was derived using the Euler-Lagrange equations.

Before deep diving into the dynamic equations, the understanding of the parameters used in modeling of bicycle is important. The Figure 2.1 shows the side view and back view of the bicycle labeled with the various parameters used in the dynamic equation.

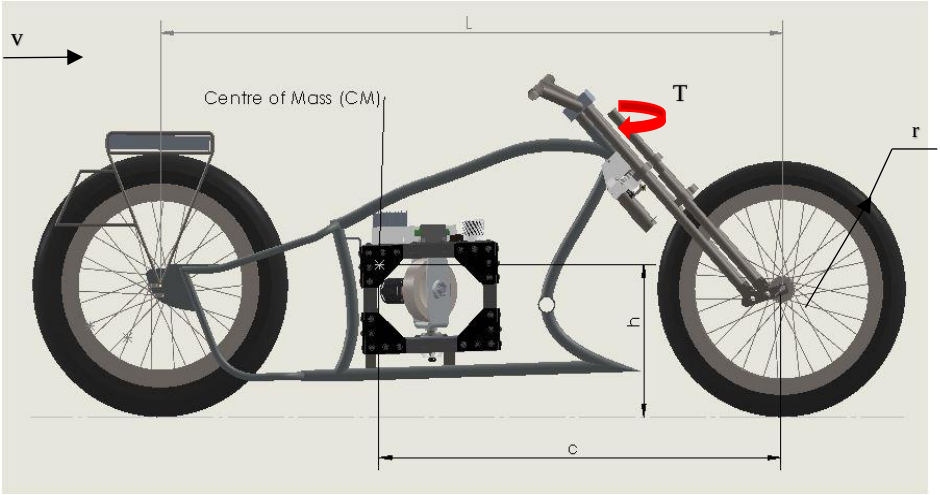


Figure 2.1: Bicycle Side View

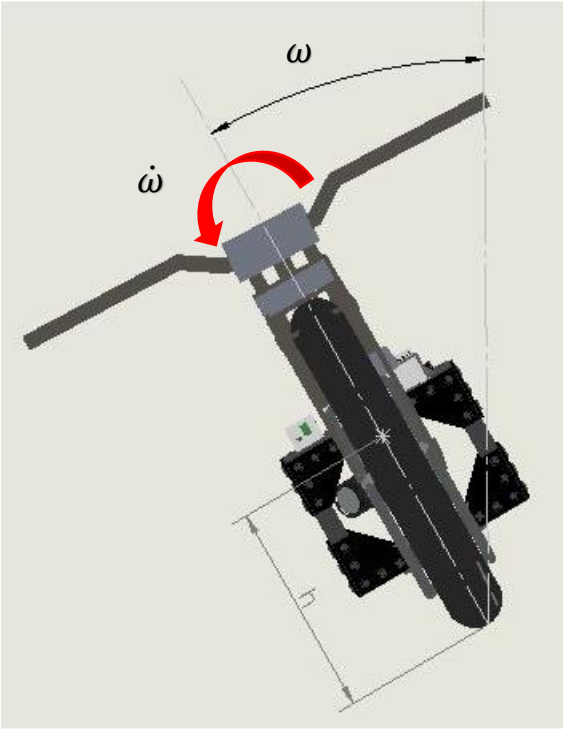


Figure 2.2: Bicycle Rear View

As seen in the Figure, **θ** represents the steering angle of the bicycle handle and ***ω*** represents the roll angle of bicycle measured from the vertical. These two parameters serve as vital components in the dynamic and govern the complete system.

The following equations describe the mechanics of the bicycle system completely. The roll acceleration can be given as:

$$\ddot{\omega} = \frac{1}{I_{bicycle+CMG}}\left(Mhg\sin\omega - \cos\omega\left(I_{dc}\dot{\sigma}\dot{\theta} + \text{sign}(\theta)\cdot v^2\left(\frac{M_d r}{r_f} + \frac{M_d r}{r_b} + \frac{Mh}{r_{\text{CM}}}\right)\right)\right) \quad (2.1)$$

$$\ddot{\omega} = \frac{1}{Mh^2}\left(Mhg\sin\omega - \cos\omega\left(M_d r^2\dot{\sigma}\dot{\theta} + \text{sign}(\theta)\cdot v^2\left(\frac{M_d r}{r_f} + \frac{M_d r}{r_b} + \frac{Mh}{r_{\text{CM}}}\right)\right)\right) \quad (2.2)$$

The equation for steering acceleration is given by:

$$\ddot{\theta} = \frac{T - I_{dv}\dot{\sigma}\dot{\omega}}{I_{dl}} \quad (2.3)$$

$$\ddot{\theta} = \frac{T - \frac{3}{2}M_d r^2 \dot{\sigma}\dot{\omega}}{\frac{1}{2}M_d r^2} \quad (2.4)$$

The above equations contain the moment of inertia of the tires which are substituted with the respective mass radius product, the inertias are considered about the axis passing through the center of tire, along the steering column and along the bicycle velocity as shown in the Figure 2.3. The equations are as follows:

$$I_{dc} = M_d r^2 \quad (2.5)$$

$$I_{dv} = \frac{3}{2}M_d r^2 \quad (2.6)$$

$$I_{dl} = \frac{1}{2}M_d r^2 \quad (2.7)$$

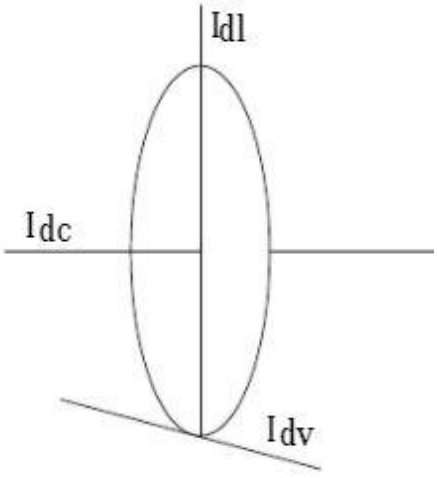


Figure 2.3: Axis for Moments of Inertia for Tire (Randlov & Alstrøm, 1998)

The front and back tires of the bicycle follow different paths and hence have different radius, same goes for the radius traced by the center of mass of the bike. The radii of the tires and center of mass (CM) is shown in the Figure 2.4 below:

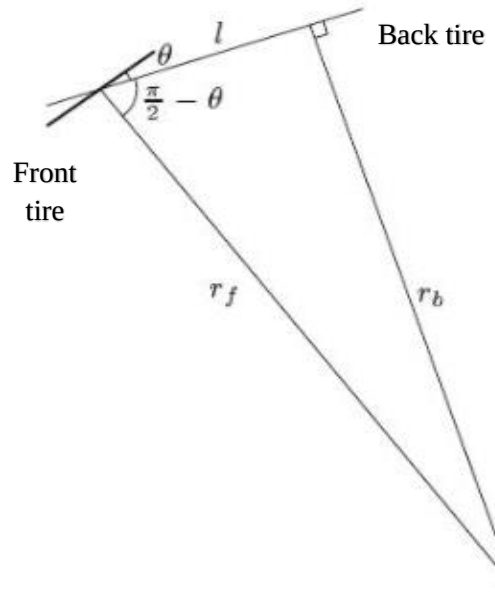


Figure 2.4: Radius of Front Tire and Back Tire (Randlov & Alstrøm, 1998)

Hence the equations for the radii of front tire, back tire and center of mass of bike are:

$$rf = \frac{l}{\left|\cos\left(\frac{\pi}{2}-\theta\right)\right|} = \frac{l}{|\sin\ \theta|} \tag{2.8}$$

$$r_b = l\left|\tan\left(\frac{\pi}{2}-\theta\right)\right| = \frac{l}{|\tan\ \theta|} \tag{2.9}$$

$$r_{\mathrm{CM}} = \sqrt{(l-c)^2 + \frac{l^2}{(\tan\theta)^2}} \tag{2.10}$$

Equations 2.2 and 2.4 are the final dynamic equations for the bicycle. The main reason for modifying the Randlov's dynamic model (Randlov & Alstrøm, 1998) as it considers the cyclist and its inertial term but for the current case, Control Moment Gyroscope (CMG) is present on the actual bike and therefore the inertia of the complete bike including the CMG is considered as shown in equation 2.2.

The equation considers the effects from the gravitational forces, angular momentum of the tires, inertias of the complete bike system (including the gyroscope) and the centrifugal forces considered for front tire, back tire and center of mass. It also considers an important term that aids the stabilization of system which is the cross effects originating from conservation of angular momentum of tires.

The current system of equations does not consider the moment of inertias arising from the CMG as it is not considered functional. These effects would be required to be added to the equation 2.2 if the system is integrated and made to work simultaneously with the robotic steering mechanism.

CHAPTER 3

ALGORITHM

3.1 Selection of Reinforcement Learning Technique

There are a number of reinforcement learning algorithms that exist and selection of one algorithm was a crucial task as there is no comprehensive distinction between the methods. But there were some aspects which helped in the decision making and are researched in detail.

Randlov's paper (Randlov & Alstrøm, 1998) highlights using SARSA algorithm for learning a bicycle. SARSA which stands for State-Action-Reward-State-Action is an on-policy algorithm which means it learns the Q value based on the action obtained from the current policy. The major drawback is that it can only handle discrete action spaces.

Other algorithm which was explored was Deep Q Learning technique (DQN) that is capable of achieving good performance for Atari video games (Mnih, et al., 2013). This algorithm overcomes the limitation of the Q learning which is the lack of generality and achieved success for high dimensional observation spaces, but the action space that it can handle is still discrete and low dimensional. It relies on finding action that maximizes the value function but in case of continuous domains an iterative optimization process is required. One method of adapting DQN to continuous domain is to discretize the action space but the action space generated is too large and also can lead to throw away important information.

After exploring Deep Q algorithm in depth, Rainbow algorithm (Hessel, et al., 2017) was researched. This paper examines the six extensions to the DQN algorithm and presents an integrated agent Rainbow made using combination of all these baselines. Furthermore, it provides performance analysis of integrated agent with these baselines and details about how the agent outperforms others. However, the agent focuses on value-based methods and does not consider policy based RL algorithms nor the actor critic methods. The integrate agent is based on methods which involve DQN and thus have discrete action space which is unlikely to work on the bicycle system as it requires continuous control.

Finally, Deep Deterministic Policy Gradient (DDPG) algorithm (Lillicrap, et al., 2019) was explored. DDPG is an off-policy, model free algorithm which uses the actor-critic method and tackles estimating values by learning policies in high dimensional continuous action spaces. In case of bicycle stabilization task, continuous observation and action spaces is a requirement which is fulfilled using the DDPG algorithm. Hence it was best suited algorithm selected.

3.2 Deep Dive into Deep Deterministic Policy Gradient Algorithm

3.2.1 Basic Introduction

DDPG is an off-policy algorithm that uses the deep neural networks to represent the policy. Some of the features of the algorithm include. Firstly, it uses the actor-critic approach, Actor and Critic are the two components around which the algorithm is centered. The actor network takes responsibility of the policy which takes in states as input and outputs an action. The critic network is responsible for generating the action value function

Q(s, a) by taking combination of states and action as input. It assesses how well is the actor network performing. Second, it adopts the feature of replay buffer and using separate target networks from DQN, replay buffer is used as a storage buffer to store the states, rewards and actions and use it to train the networks while target networks help in updation of network weights. Finally, for performing the action exploration the noise is added to the action term which follows the Ornstein-Uhlenbeck process (Uhlenbeck & Ornstein, 1930).

3.2.2 Structure of Actor-critic Networks

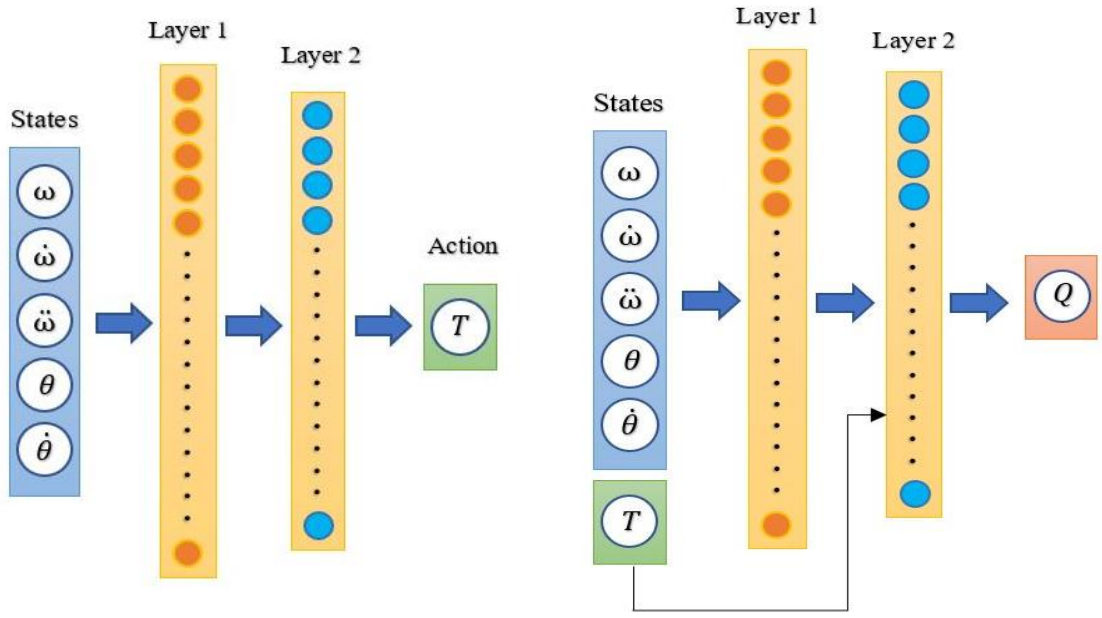


Figure 3.1: Structure of Actor Network (Left) and Critic Network (Right)

As the algorithm contains two networks actor and critic, both of them would be discussed in detail. The complete internal structure is shown in the Figure 3.1. The actor network takes in a 5-dimensional state vector and outputs the 1-dimensional action. The

network has following layers, 2 fully connected layers of 400 and 300 units respectively with glorot-normal initializers and a final tanh activated layer. Considering the critic network, it takes in the combination of the states and actions and generates the action value function Q(s, a). The layers for the critic network include the first 2 layers are similar to the actor network and the final layer outputs the q value. Also, the actions were included directly in the second hidden layer of the critic network. The final layer weights for both actor and critic are taken from uniform distribution [-3 x 10^{-3}, 3 x 10^{-3}] and biases are zero. Adam optimizer is used in both networks with learning rate of 0.0001 and 0.001 for actor and critic respectively. Replay buffer size is set to 10^6. The values for the parameters used are different from the experiment section of DDPG paper (Lillicrap, et al., 2019) as it allows user defined flexibility in order to explore different settings.

3.2.3 Bicycle Learning

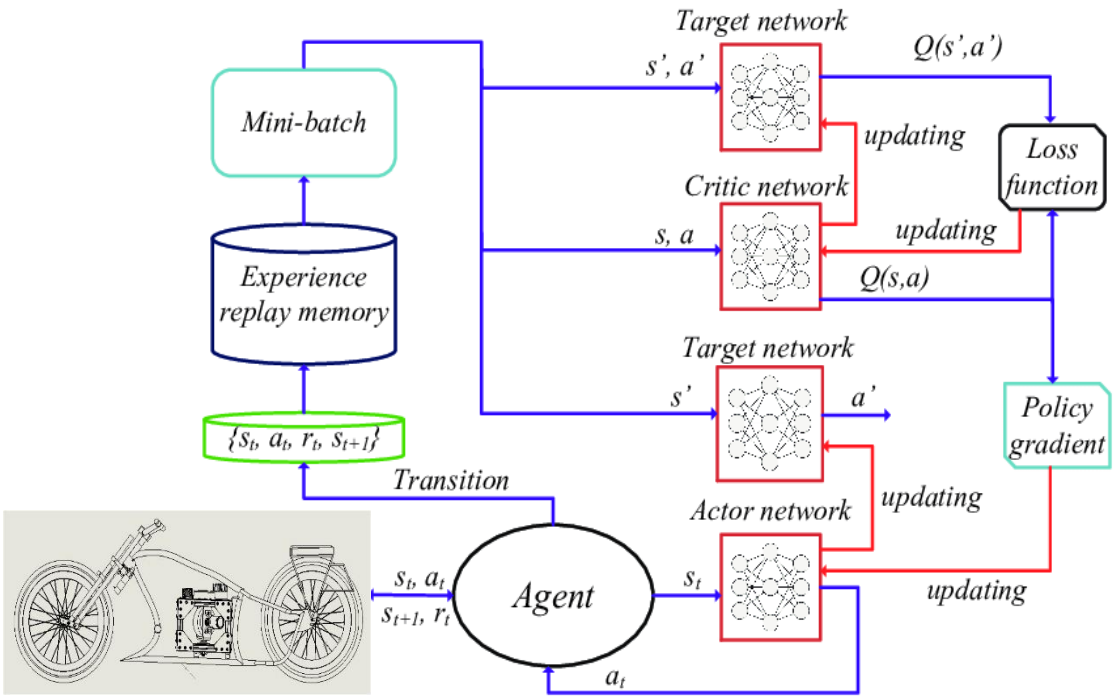


Figure 3.2: Complete Ddpg Algorithm Workflow

This section outlines the complete workflow of algorithm implemented on the bicycle as shown in the Figure 3.2. The various states of the bicycle include $(\omega, \dot{\omega}, \ddot{\omega}, \theta, \dot{\theta})$ where $\omega, \dot{\omega}, \ddot{\omega}$ are the roll angle, roll velocity and roll acceleration of the bike measured from the vertical axis and $\theta, \dot{\theta}$ are the steering angle and steering velocity of the handlebar of the bicycle respectively. These states are sent from the sensors on the bicycle to the controller and returns the action torque T. The complete process is described as follows:

The bicycle observes the state s_t which is sent to the actor network, the network takes in the state and outputs the action a_t as per the following equation:

$$a_t = \mu(s_t|\theta\mu) + N_t \qquad (3.1)$$

This part of the process refers to the action exploration stage where for given state, actions are calculated and the term N_t refers to the noise term which comes from the Ornstein-Uhlenbeck process.

Next, the bicycle sends the other state at time t+1, s_{t+1} and returns reward r_t to the agent. The parameters obtained $[s_t, a_t, s_{t+1}, r_t]$ are then sent to the replay buffer R and stored in order to be used later for training of the networks.

The above process continues in a loop and soon the replay buffer has N minibatch samples available to be sent to the networks. As R fills up, a minibatch of data is sent to the networks where actor network takes in the states s_{t+1} , critic network takes in the states s_t , s_{t+1} and the actions a_t while the loss function takes in the reward r_t for its calculation.

Next stage is the training of the critic network which includes minimizing of the loss function. $\mathrm{Q}(\mathrm{s},\mathrm{a}|\theta^Q)$ and $Q'(\mathrm{s},\mathrm{a}|\theta^{Q'})$ are the main network and target network of the critic where θ^Q and $\theta^{Q'}$ refer to the weight of the main critic network and target network respectively. θ^Q is optimized by minimizing the loss function given by:

$$L = \frac{1}{N}\sum_i \left(y_i - Q(s_i, a_i \mid \theta^Q)\right)^2 \qquad (3.2)$$

$$\text{where } y_i = r_i + \gamma Q'\left(s_{t+1}, \mu'\left(s_{t+1} \mid \theta^{\mu'}\right) \mid \theta^{Q'}\right) \qquad (3.3)$$

Following that is the training of the actor network. Considering $\mu(\mathrm{s},\mathrm{a}\,|\theta\mu)$ and $\mu'\left(\mathrm{s},\mathrm{a}\middle|\theta^{\mu'}\right)$ be the main network and the target network of the actor, the main actor is updated using the deterministic policy gradient theorem which is:

$$\nabla_{\theta^\mu}\mu|_{s_i} \approx \frac{1}{N}\sum_i \nabla_a Q(s, a \mid \theta^Q)\Big|_{s=s_i, a=\mu(s_i)} \times \nabla_{\theta^\mu}\mu(s \mid \theta^\mu)|_{s_i} \qquad (3.4)$$

Finally, the parameters of target networks are updated using soft update technique with learning rate τ by using the following equations:

$$\theta^{Q'} \leftarrow \tau\theta^{Q} + (1-\tau)\theta^{Q'} \quad (3.5)$$

$$\theta^{\mu'} \leftarrow \tau\theta^{\mu} + (1-\tau)\theta^{\mu'} \quad (3.6)$$

The bicycle learning process involved training algorithm with three conditions no noise, with Ornstein-Uhlenbeck (OU) noise and Gaussian noise. In case of no noise condition, the training was done without considering any noise in the algorithm. For algorithm with OU noise, the noise was added to the actions coming from the actor network which followed the Ornstein-Uhlenbeck process. Finally, the last condition used for learning process was training using addition of Gaussian noise to the neural network.

CHAPTER 4

SOFTWARE TESTING

4.1 Device Configuration

The DDPG algorithm and the bicycle environment is developed and implemented on the Windows 64-bit OS, i5-8300H CPU @ 2.3GHz processor. PyCharm is used as an integrated development environment (IDE) with Python 3.7 as the programming language. There were number of packages used, out of which the most important ones include Pandas3D game engine (Goslin & Mine, 2004), TensorFlow 2.1 framework was used for the algorithm implementation and OpenAI gym (Brockman, et al., 2016) for creation of the custom bicycle environment.

4.2 Agent Information

The class Agent_DDPG developed using the PyCharm IDE is used to initialize the actor and critic networks, replay memory and the noise functions. It specifies 3 functions: bike learning, evaluate and hardware implementation. Bike learning function exhibits the exact procedure explained in the algorithm section starting from action exploration, training of the networks, getting the actions and rewards for every step executed and saving the learned model as checkpoints file. Evaluate function restores the saved learned model and is implemented on the custom bicycle environment made using Pandas3D game engine. It

basically using the trained actor network and inputs the states received from the custom environment step and creates a simulation video of the executed actions calculated using the received states. It acts as a verification of how the bike performs and its stabilization ability. Whereas the hardware function is for communication with the hardware devices.

4.3 OpenAI Gym Pendulum Environment Testing

4.3.1 Setup

Before testing the algorithm implementation on the custom bicycle environment, it was implemented on OpenAI pendulum environment (Brockman, et al., 2016) in order to verify algorithm's capability. The setup was easy which required using the gym 0.17.2 package and loading the pendulum environment. It takes in two dimensional states and outputs only one action torque which is applied to the pendulum so that it remains vertically balanced. The environment was chosen as it is similar to the bicycle stabilization problem having the same goal. The environment is predefined containing the dynamic model of pendulum which calculates the new states at definite timestep and rewards.

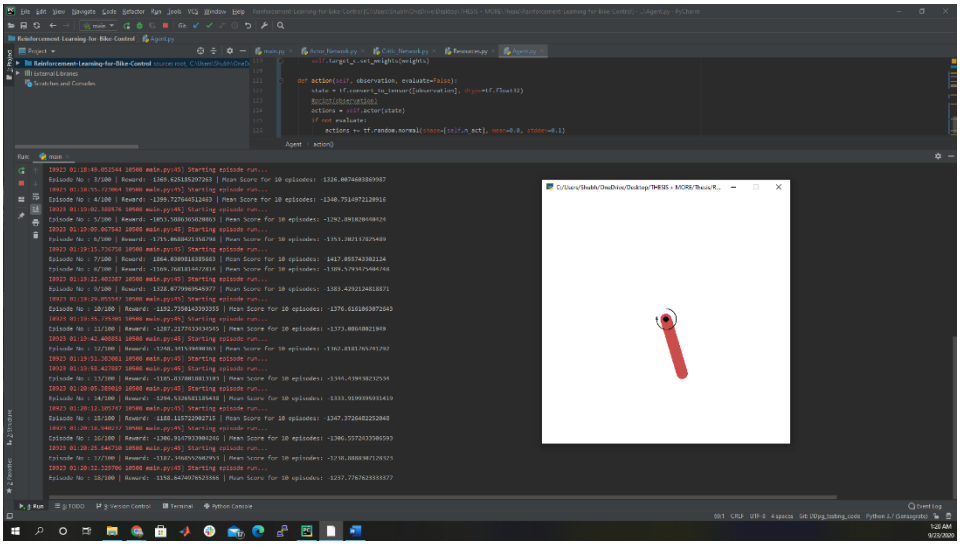
4.3.2 Network Structure Design for Pendulum Environment

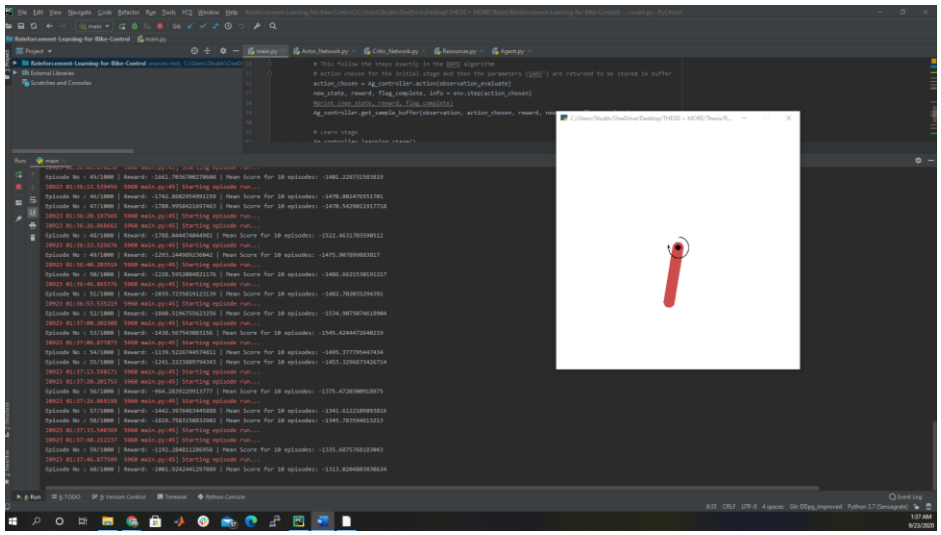
The actor network takes in a 3-dimensional state vector and outputs the 1-dimensional action. The network has three layers, 2 fully connected layers of 400 and 300 units respectively with kernel initializers and bias initializers equal to weights and bias calculated as a uniform distribution function of fan-in layer and a final tanh activated layer.

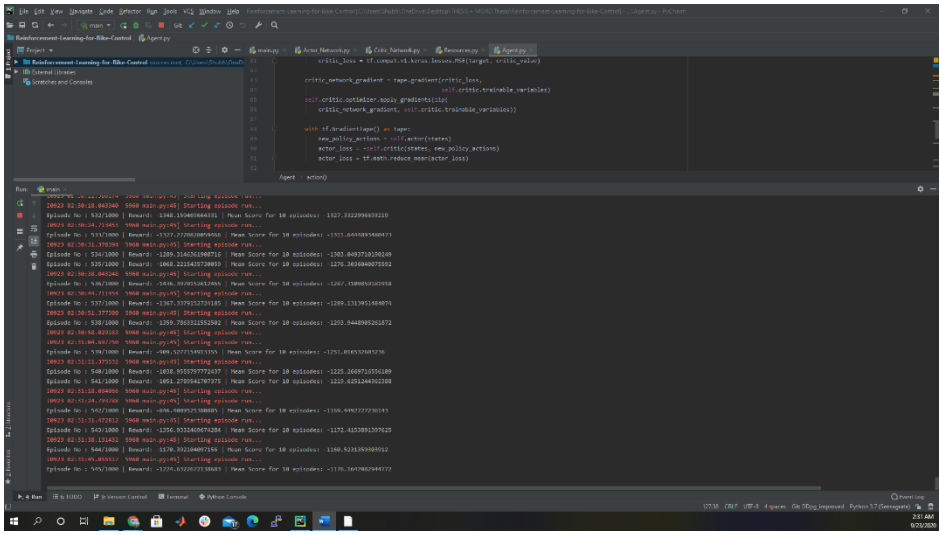
Considering the critic network, it takes in the combination of the states and actions and generates the action value function Q(s, a). It also has three layers; the first 2 layers are similar to the actor network and the final layer outputs the q value. Also, the actions were included directly in the second hidden layer of the critic network. The final layer weights and biases for both actor and critic are taken from uniform distribution [-3 x 10^{-3}, 3 x 10^{-3}] and [-3 x 10^{-4}, 3 x 10^{-4}] respectively. Adam optimizer is used in both networks with learning rate of 0.001 and 0.01 for actor and critic respectively denoted as alpha and beta. Gamma value is taken as 0.99. Replay buffer size is set to 10^6.

4.3.3 Results

The following results shown are snaps of the PyCharm IDE on which the test was conducted with the total reward function plotted. The training was conducted for 1000 episodes.







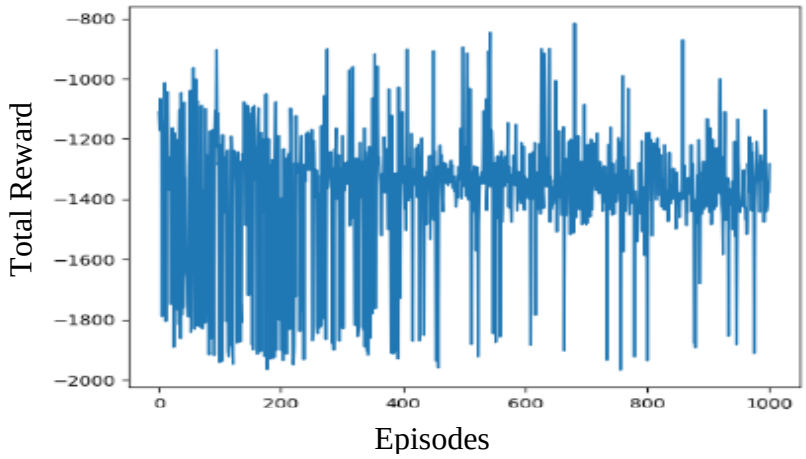


Figure 4.1: PyCharm Ide Snapshots for Pendulum Environment and Total Reward Plot

Plot shows that during the training process, the reward value starts from around -1900 and it decreases and reaches around -1200 at the end of 1000 episodes. The simulation results do show network learning as the mean score decreases and thus this validated that the algorithm can be used further to be implemented on the custom bicycle environment.

4.4 Custom Bicycle Environment Testing

4.4.1 Setup

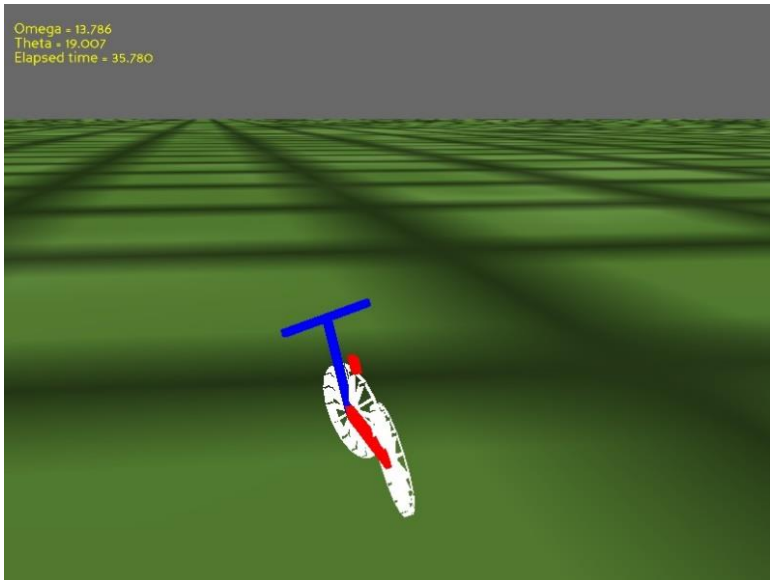


Figure 4.2: Bicycle Custom Environment

Bicycle environment setup followed the same steps of the pendulum environment and had to be created as custom environment with gym registration in order to use its features. The custom environment BicycleBalanceEnv developed have the features as follows:

The Initialization function defines the parameter values related to the bicycle. The values used are taken from the physical bicycle model which was used in hardware testing. It also initializes the values of masses of tires, masses of bicycle system, position of center of mass, velocity of bike and many other components. The main initialization parameters considered are listed.

Angles at which the episode is considered failed and to start a new episode is dependent on the angles ω (Roll angle) and θ(steering angle) set as 20˚ and 90˚ respectively. For defining the observation space , low and high limits were set as ω = [-20, 20] and θ = [-80, 80]. While for the action space, output torque T was set as [-2 , 2] N-m.

The step function in the environment is highly accessed function as it defines the complete dynamic equations of the bicycle. It is responsible to return the observed states and rewards to the agent.

4.4.2 Reward Function

The reward function was designed considering the most important parameters that affect the bicycle environment which were found to be roll angle ω, roll velocity $\dot{\omega}$ and roll acceleration $\ddot{\omega}$. The coefficients of the reward function were assigned depending on the importance and desired domination required. For the bicycle problem, two approaches were considered given in detail as follows:

Approach 1: Importance given to roll angle ω;

$$\text{Reward function} = \boldsymbol{r}(s,a) = \begin{cases} \text{last reward before falling down} & |\omega| > \frac{\pi}{9} \\ -(\omega^2 + 0.1\dot{\omega}^2 + 0.01\ddot{\omega}^2) & |\omega| < \frac{\pi}{9} \end{cases}$$

where the term takes responsibility for balancing the bicycle. In this reward function, the bicycle is considered as falling down if the angle between the bicycle and the vertical plane is greater than π/9 rad (or 20 degree). When the bicycle falls down, the reward at this time is used until the end of the episode. The coefficients for each term are selected based on their contributions to the reward. Particularly, we use a coefficient of 1.0 for ω^2, which is the most important in the balancing term while the other terms are given less importance. The effects of the components on the reward value are discussed in the results section. The model training was done for 10000 episodes and individual contribution of each term and total reward were plotted.

Approach 2: Importance given to roll acceleration $\ddot{\omega}$;

$$\text{Reward function} = \boldsymbol{r}(s,a) = \begin{cases} \text{last reward before falling down} & |\omega| > \frac{\pi}{9} \\ -(0.01\omega^2 + 0.1\dot{\omega}^2 + \ddot{\omega}^2) & |\omega| < \frac{\pi}{9} \end{cases}$$

Experimentation of reward function was done in order to explore different possibilities and to see if better and faster learning of the model can be achieved. In this reward function, we use a coefficient of 0.01 for ω^2, which means giving least importance to roll angle and instead giving highest priority to the acceleration term by setting its coefficient as 1.0. The

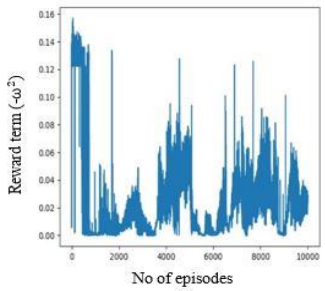
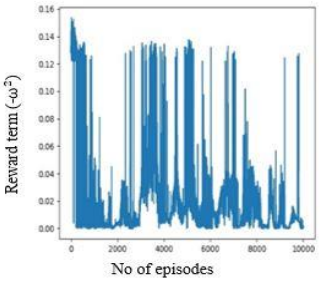
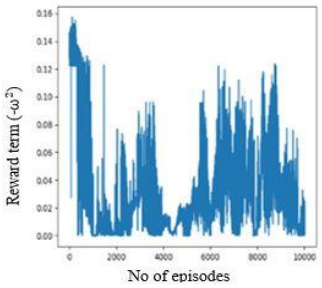
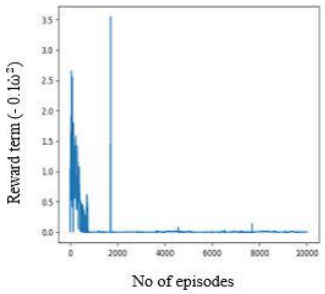
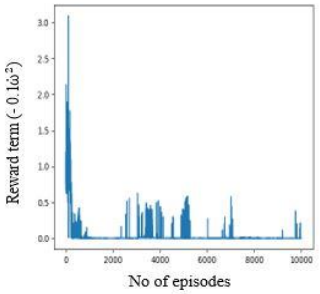
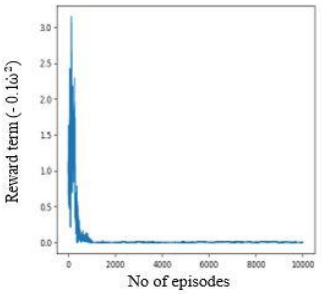
figures in results section shows the effects of the components on the reward value individually as well as total contribution shown by total reward plot.

Both the reward functions were able to achieve satisfactory stabilization for the bicycle model but to select the best performing model was important in order to be used for evaluation on the physical bike model. The detailed analysis is performed in the next section.

4.4.3 Results

After setup of the custom environment the execution of the algorithm was divided into three main steps, bike learning, evaluation and simulation. Considering the bike learning stage, this is the basic core code which defined the efficiency of other steps. This stage encompassed the complete neural networks training using the parameters obtained from the environment. The rewards and reward terms plots are obtained which are calculated at each timestep and saved as NumPy array file (.npy) format which gives an idea of the training level achieved. Whereas the evaluation and simulation stages use the saved model to evaluate the performance of trained actor network by passing in random states.

The training was conducted for 10000 episodes in order to ensure the proper training of model is achieved. As discussed in the bicycle learning process, the training of networks considered the variations of having no noise, addition of action noise according to the Ornstein-Uhlenbeck process and addition of gaussian noise. The individual reward terms and total reward obtained during the training for these trained models are compared for both the approaches and the inferences made are discussed in detail as follows.

Plot	Approach 1		
	DDPG with no noise	DDPG with OU noise	DDPG with Gaussian Noise
Reward term 1			
Reward term 2			

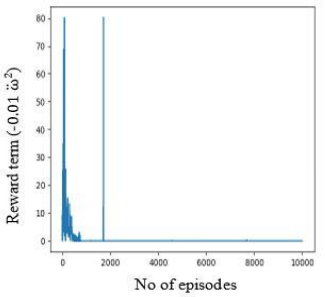
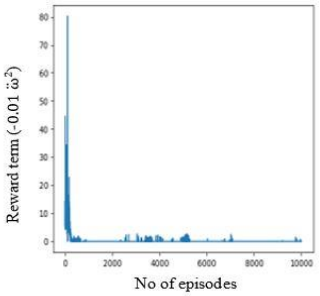
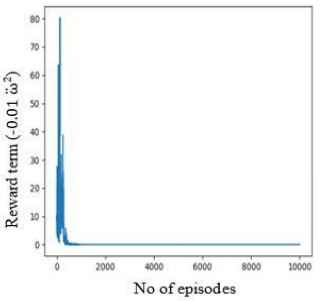
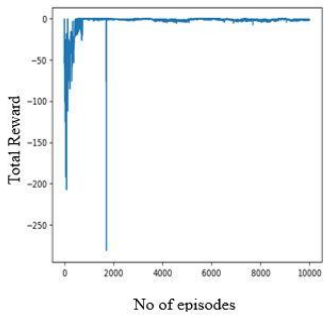
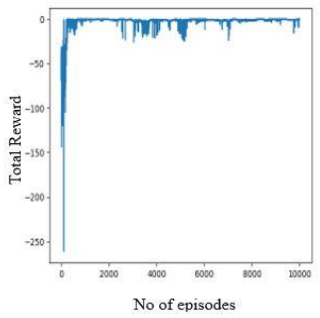
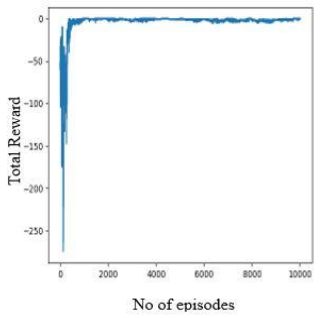
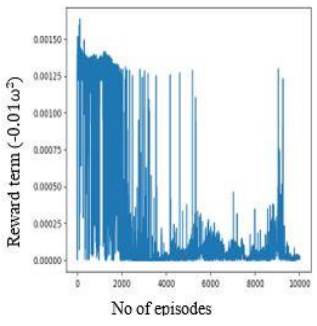
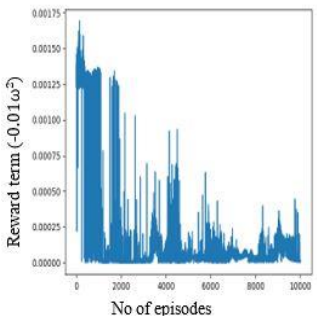
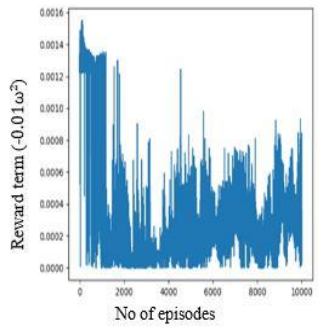
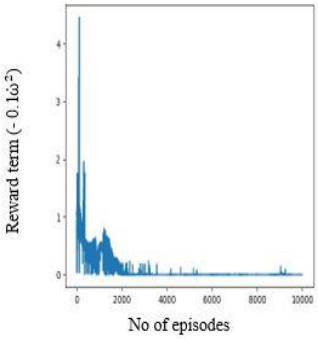
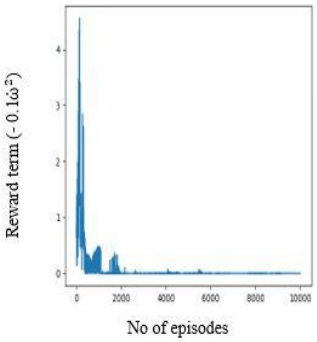
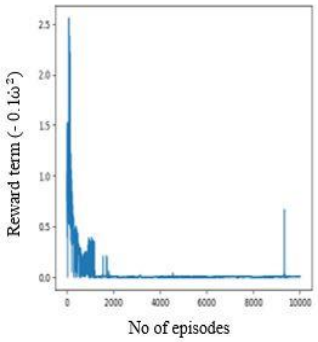
Plot	Approach 1		
	DDPG with no noise	DDPG with OU noise	DDPG with Gaussian noise
Reward term 3			
Total Reward			

Figure 4.3: Comparison of Models Considering No Noise and with Noise Addition with Reward Function Based on Approach 1

Considering Approach 1, in case of the results obtained for reward term 1, the maximum value reached at the end of training for model considering no noise, OU noise and Gaussian noise are 0.0295 rad^2 , 0.001 rad^2 and 0.00013 rad^2 respectively. Thus, the roll angle ω calculated would be $\pm$ 9.84˚, $\pm$ 1.812˚ and $\pm$ 0.6532˚ . As seen from the calculated values, all the values are below the threshold hence all of them attained satisfactory results. From the plots, it can be observed that model that considered noise performed better than others and boosted the performance. But out of the two, introduction of gaussian noise gave more boost and better performance.

Considering the total reward plot, training that included gaussian noise was quite stable than the other two plots throughout. Though all of the plots showed faster learning but the convergence results for model considering gaussian noise was better.

Plot	Approach 2		
	DDPG with no noise	DDPG with OU noise	DDPG with Gaussian Noise
Reward term 1			
Reward term 2			

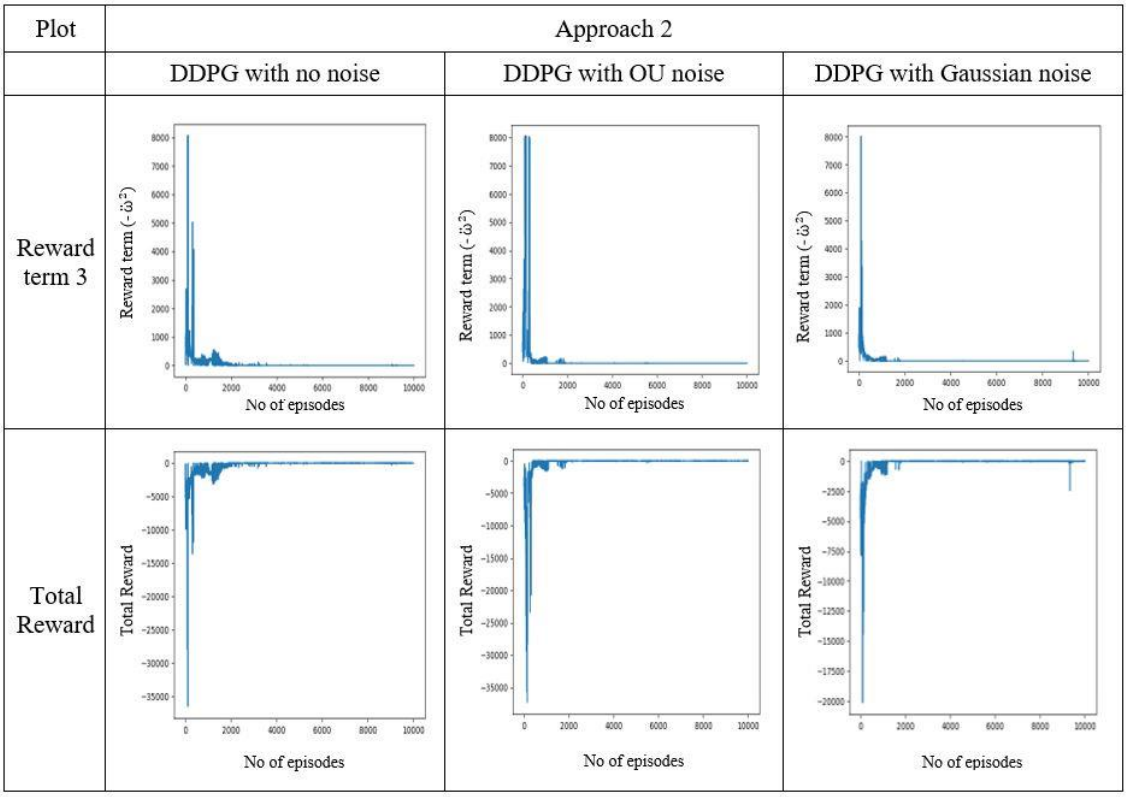


Figure 4.4: Comparison of Models Considering No Noise and with Noise Addition with Reward Function Based on Approach 2

Considering Approach 2, in case of the results obtained for reward term 1, the maximum value reached at the end of training for model considering no noise, OU noise and Gaussian noise are 0.00003 rad^2 , 0.00002 rad^2 and 0.000085 rad^2 respectively. Thus, the roll angle ω calculated would be $\pm$ 3.14˚, $\pm$ 2.56˚ and $\pm$ 5.28˚ . As seen from the calculated values, all the values are below the threshold hence all of them attained satisfactory results, but the roll angle values were greater than the ones in approach 1 except for no noise condition. The results considering the condition of noises, the model with OU noise performed better than the gaussian noise.

While in case of total reward plot, the training in approach 1 was better than the ones in approach 2. Considering the total training process, the values of reward in case of approach 1 were significantly less than the ones in approach 2.

After assessing the results of both the approaches, approach 1 gave promising results. Hence, after careful consideration Approach 1 was selected for further deployment on the hardware testing process.

During the preliminary stage, comparison was done between considering the no noise and OU noise condition. Hence, the hardware testing performed before was done by implementing the model which considered OU noise and reward function based on approach 1 which proved to be better than the no noise condition. Then considering the improvement required, the addition of gaussian noise to the network was performed for training process which gave better results than the former and this model would be implemented on the hardware system as a future work.

CHAPTER 5

HARDWARE TESTING

5.1 Overview of the Self Balancing System

5.1.1 System Overview

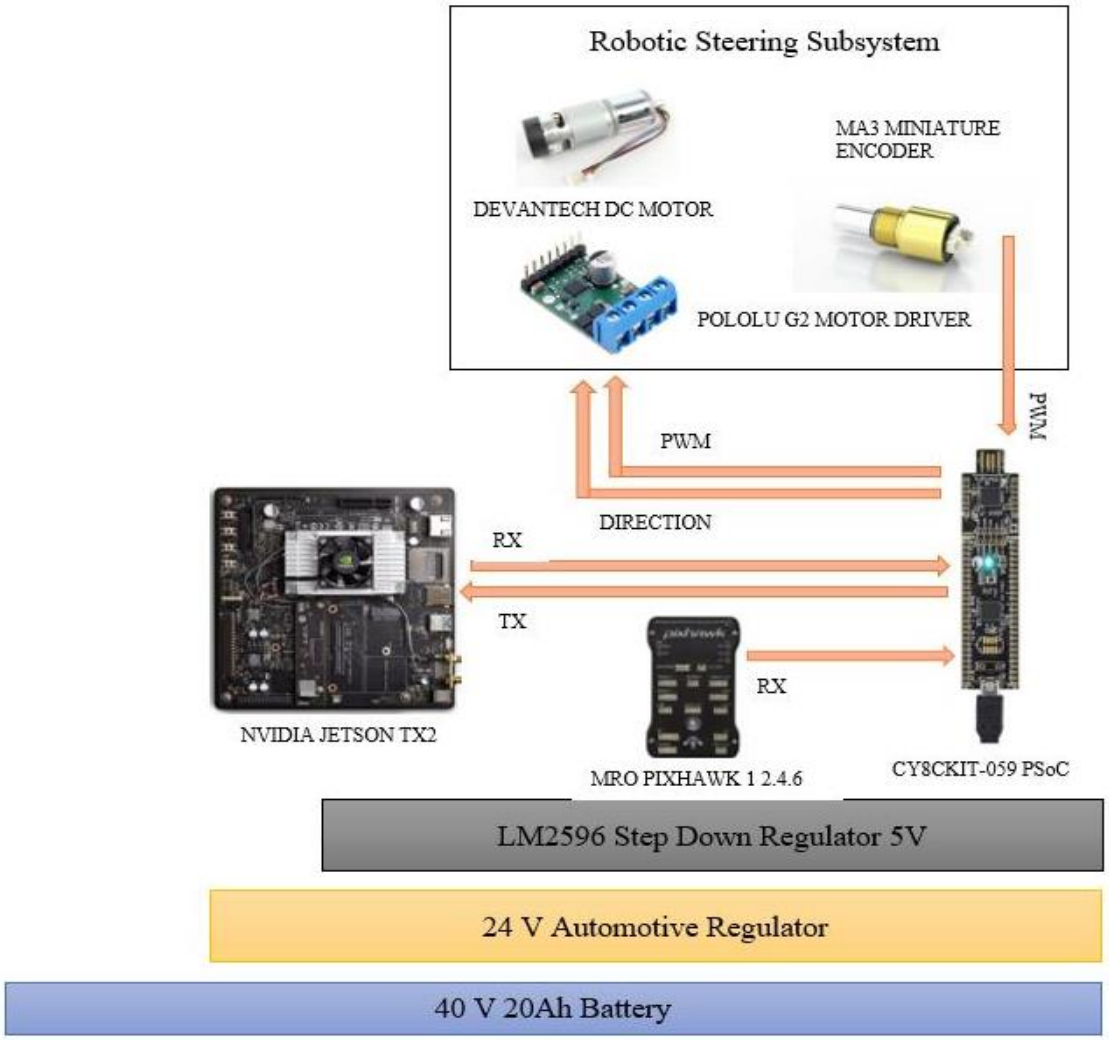


Figure 5.1: System Block Diagram

The complete hardware system with all the components used on the bicycle are shown in the block diagram. The main subsystems on the bicycle include robotic steering and the control moment gyroscope but for implementation of algorithm only robotic steering is used. The steering control system allows the angular position of the bicycle front fork to be controlled so that the bike can be steered while in motion.

The DDPG algorithm is implemented on the supercomputing edge device Nvidia Jetson Tx2 and it is communicated with the PSoC 5 microcontroller through UART port which acts as a central member for the complete process. More components include mRo Pixhawk which is actually designed for flight control application and contain embedded processors better that the PSoC 5. But while considering integration of multiple systems into one, it was found that flexibility of PSoC seems to be better than the former. However, Pixhawk is used for application of Inertial Measurement Unit (IMU) and uses UART communication with the RX line connected to the PSoC 5. Pixhawk flight controller is able to transmit the motorcycle's attitude data at real-time frequency to the PSoC and these values are then transmitted to the Jetson Tx2 as state values. It also has editable filter and level-horizon offset calibration once it is installed on a vehicle.

Considering the robotic steering control, it uses 1:49 planetary gear brushed DC motor which is responsible for the torque generation on the handlebars. The selection of the motor was depending on steering system is a low frequency and static torque system. The associated motor driver used is Pololu G2 which provides current sensing required for accurate torque control of the steering actuator. For sending the instantaneous state values for the steering to the PSoC 5, hall effect absolute encoders were used.

Considering the frame of the bicycle, two adult sized training wheels were added in order to serve two purposes; (1) Safety of the bicycle and components when it tends to collapse and (2) The training wheels were set in a manner that when one side of the wheel touches the ground it makes an angle equal to the threshold value set at which bicycle is said to fail balancing task with respect to the vertical.

The hardware system of the bicycle had the components installed with the required electronics which was done for previous research on the bicycle for stationary balancing using control moment gyroscope and robotic steering control. The complete overview of the system used and its role is discussed in detail below. The deployment of the DDPG algorithm discussed in the software testing required more components to be added and also modifications to the current robotic steering mechanism which used torque control replacing the position control for actuating the steering motor. It also involved addition of safety mechanisms to the bicycle. The system discussed here is only related to the robotic steering mechanism which is required to be functional.

5.1.2 Power System

The power system is responsible for distribution of electrical energy to various components for their working. It includes 4 voltage rails, the 48V rail is for high power applications such as the CMG powered directly from the battery with 30A fuse for protection and emergency stop button which triggers a relay to disconnect all the powered applications. A regulated 24V DC rail for intermediate power applications, such as robotic steering and high-level computing, is derived from the 48V rail with a step-down voltage

regulator that can supply up to 20A. A regulated 12V supply in order to power the Nvidia Jetson TX2 for its functioning stepped down for 48V rail. Finally, voltage is reduced to 5V DC rail for microcontroller and other peripherals.

5.1.3 Robotic Steering Actuator

The balancing of bicycle is achieved with the help of steering the handlebars. Therefore, some mechanism should be present in order to transmit the torque and finally cause the steering which was achieved using the robotic steering actuator. The robotic steering system initially developed in Deng *et al* (2018) was implemented for the bicycle. The robotic steering system shown in Figure 5.3 enables automated balance and trajectory control while the bicycle is moving forward. This system consists of the steering control DC motor, shown in Figure 5.2, motor driver and an encoder. This system can operate as a closed-loop torque actuator.

In order to accommodate the application of high static torque, low speed and low frequency of the robotic steering a high planetary gear ratio 24 V brushed DC motor was selected.

- Nominal Voltage: 24V
- No Load RPM: 143
- Stall Current: 13A
- Stall Torque: 10.39 Nm

Figure 5.2: Devantech 24v, 49:1 Geared Brushed Dc Motor

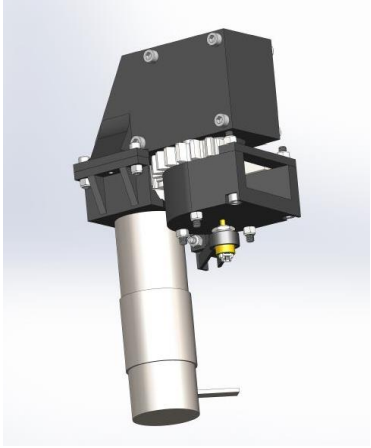


Figure 5.3: Robotic Steering System

5.1.4 Sensors Information

The selection of Inertial Measurement Unit (IMU) was based on the robustness and modularity of the device selected. mRo Pixhawk 1 2.4.6 was primarily selected for the IMU despite its default use which is for flight controller application. Pixhawk systems have opensource ground control software which allows the user to calibrate the IMU for specific vehicles. Figure 5.4 shows the Pixhawk controller.

Figure 5.4: Pixhawk 1 2.4.6 Flight Controller

Pixhawk is made to communicate with the PSoC 5 microcontroller with the help of UART communication where it receives the attitude values. The sending and receiving of the messages is done using Mavlink which is a very lightweight messaging protocol for communication.

Another sensor, hall effect encoder sensor was used in conjunction to the steering actuator to get the steering responses and communicated values to PSoC 5. This encoder comes prefabricated from US Digital with ball bearing sleeve and a shaft. This allows the mechanical system of the robotic steering to be less susceptible to mechanical alignments. The main encoder processor and the magnets are shielded from minor magnetic effect with a metallic outer hull. Figure 5.5 shows the MA3 miniature absolute encoder.

Figure 5.5: MA3 Miniature Absolute Magnetic Shaft Encoder

5.1.5 Embedded AI Computing Device

Embedded AI computing devices make it possible to deploy algorithms on the system at ease. The Jetson TK1, TX1 and TX2 models all carry a Tegra processor (or SoC)

from Nvidia that integrates an ARM architecture central processing unit (CPU). Jetson is a low-power system and is designed for accelerating machine learning applications.

For the bicycle application, Nvidia Jetson TX2 kit is used which is flashed with the Jetpack 4.4, a Software Development Kit (SDK) for the jetson boards which includes Linux for Tegra (L4T) OS. It provides GPU which helps in deploying algorithms at a faster rate.

- Dual-core NVIDIA Denver2 + quad-core ARM Cortex-A57
- 256-core Pascal GPU
- 8GB LPDDR4, 128-bit interface
- 32GB eMMC
- 4kp60 H.264/H.265 encoder & decoder
- Dual ISPs (Image Signal Processors)
- 1.4 gigapixel/sec MIPI CSI camera ingest

Figure 5.6: Nvidia Jetson Tegra X2

5.1.6 Pololu G2 Motor Driver

Pololu G2 shown in Figure 5.7 is a high-power motor driver which is used to drive the Devantech DC motor. The selection of this motor driver was based on that it included current sensing which is required for closed loop torque control for the steering actuator.

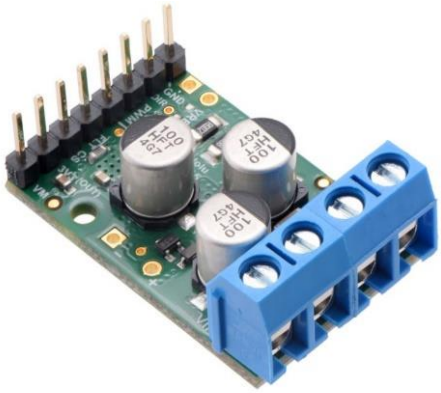
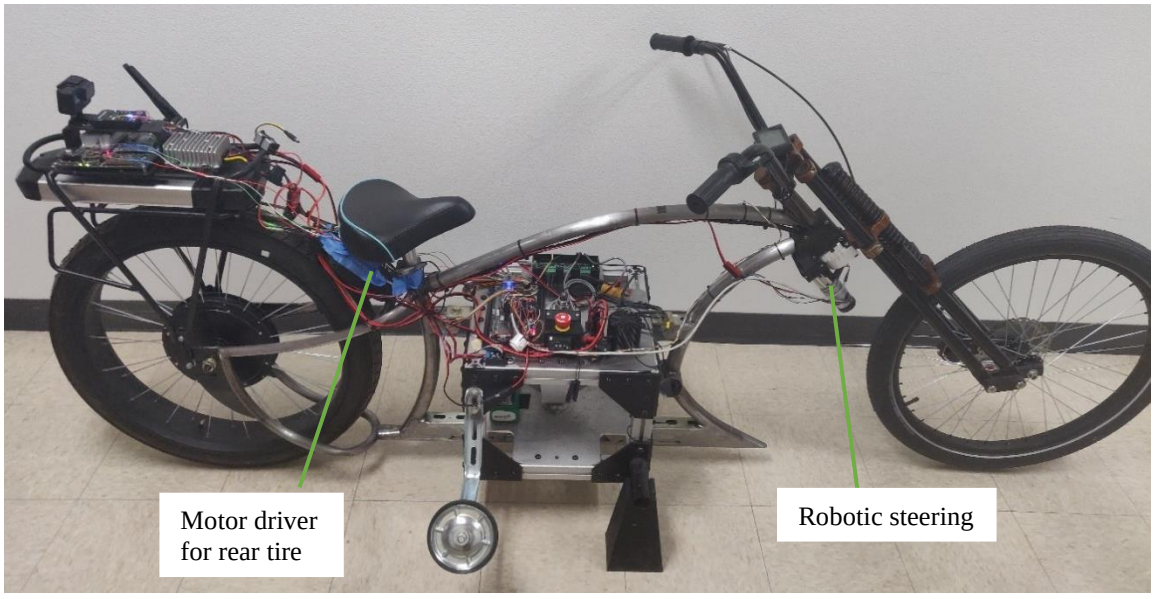


Figure 5.7: Pololu G2 High-power Motor Driver 24v21

The Arizona State University RISE Lab's bicycle model used for the hardware implementation is shown in the Figure 5.8. The bicycle is installed with the required sensors and actuators as discussed above in this chapter.



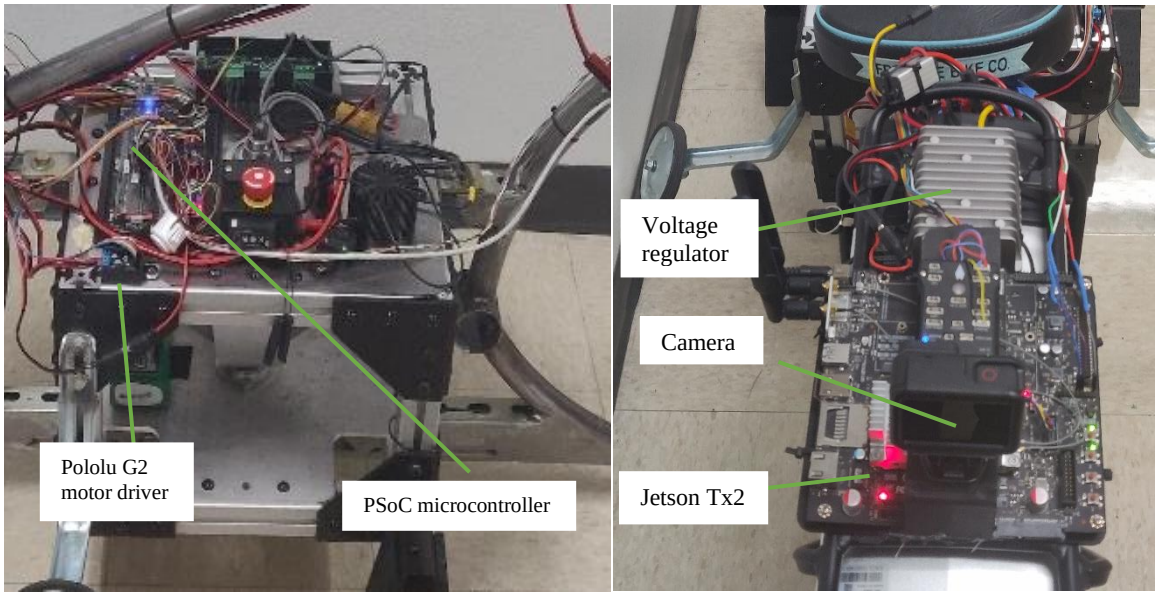


Figure 5.8: Complete Hardware Setup on Bicycle

5.2 Algorithm Deployment on Hardware and Its Workflow

Hardware implementation was the most crucial process after conducting the software testing. The trained model during the software testing is saved so that it could be used during the hardware implementation.

The DDPG algorithm was implemented on the Jetson Tegra X2 and some modifications were made in the code in order to setup serial communication with the microcontroller. The additions made to the code are explained in detail below with the complete workflow:

Firstly, the python libraries required for the code to be working were installed and verified. These packages included installation of TensorFlow 2.1 and its dependencies, OpenAI gym v0.17.3, pySerial 3.4 and also required environment wrappers.

The next step was defining the hardware function which required the opening of serial port of Jetson TX2. The port /dev/ttyTHS2 was used for UART communications with a baud rate of 9600 and is able to read and write the data sent and received from the PSoC.

After opening the port, the code is able to read the data received which is in binary form and needs to be unpacked in the form readable by the algorithm. This states data received is then sent to the trained actor network with output the action.

The action calculated is again packed into the binary form and sent to the PSoC 5. Finally, action is converted to an analog signal and fed to the steering actuator which shifts the handlebar appropriately.

The above steps are made to run continuously for each timestep till the process is functional. Detailed code information can be found in appendix B github repositories.

5.3 Results

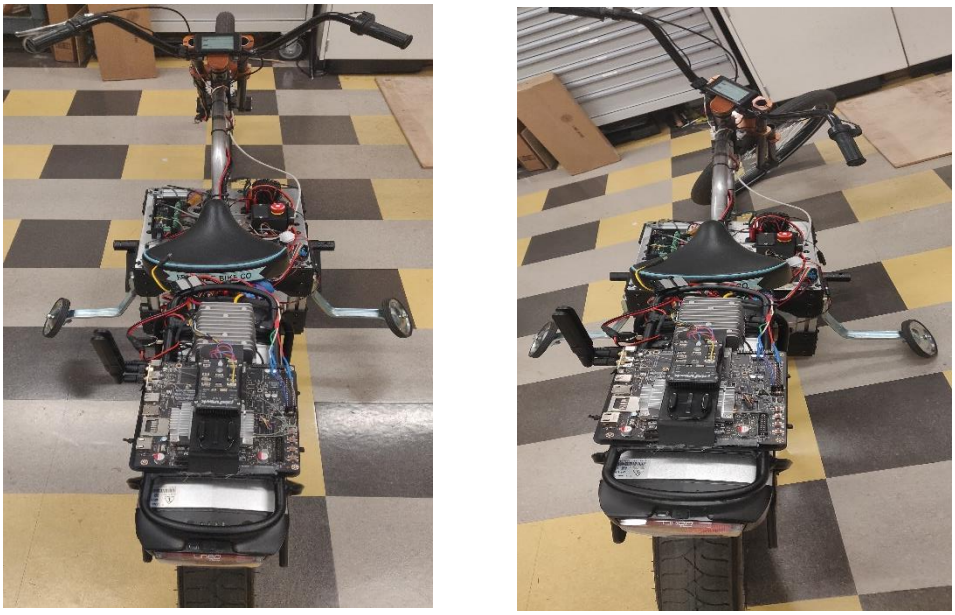
5.3.1 Plots and Trial Runs

After the complete hardware setup, a careful verification test was carried in order to check the controllability of sensors and actuators using the central PSoC 5LP microcontroller. These tests were made in order to ensure the bicycle does not pose a danger to any living being and also to understand about the responses or actions produced by the algorithm through a preliminary test.

After successful verification of working components and installation of safety mechanisms, an outdoor test was conducted. The bicycle was made to undergo four trial runs and roll angle of the bicycle for every time step was monitored using two methods.

First method was using roll angle plot in which plots for roll angle v/s number of episodes ran by the bicycle were plotted in order to keep a track of roll angle of the bike is within the threshold value. Second method was visual inspection in which the bicycle was installed with training wheels at an angle equal to threshold roll angle value (angle used for failing the episode) which was set at $\pi / 9$ or 20˚.

The following images show the bicycle position with zero roll angle, completely upright and initial start condition of the trial runs as shown in Figure 5.9 (a) and Figure 5.9 (b) shows the failure of trial when the training wheel touch the ground i.e. bicycle roll angle reaches the threshold.



(a) Bicycle upright condition (b) Bicycle failure condition

Figure 5.9: Bicycle Positions (a) Bicycle Upright Condition and (b) Bicycle Failure Condition

Considering the outdoor test that was conducted, the behavior of bicycle was closely monitored. The outdoor test conducted consisted of 4 trial runs and Figure 5.10, 5.11, 5.12, 5.13 depicts snapshots of the experiments with each trial showing the condition of initial start, self-balancing and the final failure followed by the plots of the roll angle of bicycle versus episodes run which were plotted during the experiment in order to study its characteristics.

(a) Initial start condition (b) Self-balancing condition (with rolling) (c) Bicycle failing condition

Figure 5.10: Trial Run 1

(a) Initial start condition (b) Self-balancing condition (with rolling) (c) Bicycle failing condition

Figure 5.11: Trial Run 2

(a) Initial start condition (b) Self-balancing condition (with rolling) (c) Bicycle failing condition

Figure 5.12: Trial Run 3

(a) Initial start condition (b) Self-balancing condition (with rolling) (c) Bicycle failing condition

Figure 5.13: Trial Run 4

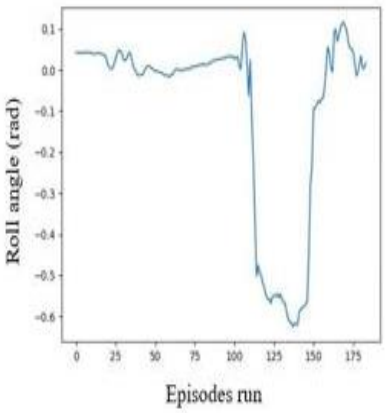
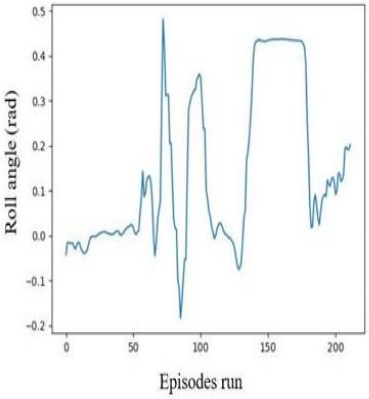
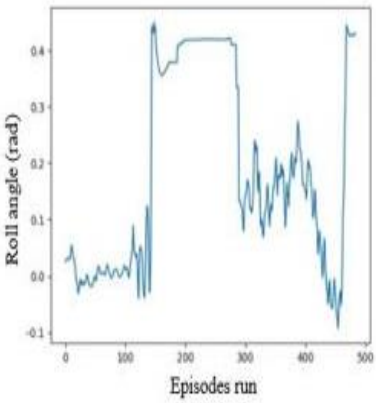
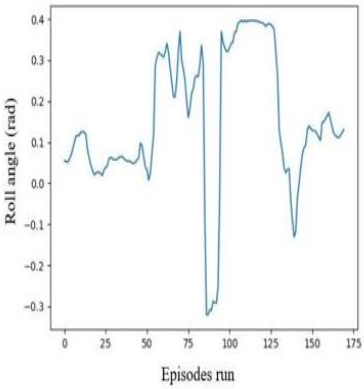
Trial Run	Roll angle	Trial Run	Roll angle
Trial Run 1		Trial Run 3	
Trial Run 2		Trial Run 4	

Figure 5.14: Plots of Roll Angle V/S Number of Episodes Run for Four Trial Runs During Hardware Testing

5.3.2 Trial Run Analysis

The results of trial run 1 show that the threshold roll angle was reached at 115th episode and the bicycle was able to maintain its balance for 17 seconds. Similarly, trial run 2 failed at 150th episode but it required some support during the start of the run for some seconds. The stabilization time with and without support was found to be 20.2 seconds and 7 seconds respectively. For trial run 3 and 4, the action torque was increased to 4 N-m limit (positive and negative domain) which was originally set to be between -2 N-m and 2 N-m. For trial run 3, the failing episode was 75 with balancing time of 11 seconds and for trial 4, the failing episode was 105 with balancing time of 15.5 seconds.

During the testing phase, it was observed that the bicycle was able to successfully stabilize itself with active steering which was fed with the required torque values for the times during trials mentioned above. Just before the failing of the stabilized state it undergoes oscillations which gradually increase as seen in the plots and leads to final failure or collapsing. It was noted that during this time the state-action response time on the Tx2 is delayed and slows down to some extent.

Depending upon the system response observed, there are some possible ways in which this could be improved. The improvement in the area of model training was considered which included addition of gaussian noise during the training process and compared with other models as shown in the results section of software testing. It showed better performance and hence it would be considered for implementation as a future work. Another area of improvement which can be done is improving communication between the Jetson and PSoC devices, the current hardware uses UART communication for data

transfer and hence the communication rate can be made faster using Serial Peripheral Interface (SPI) or other communication peripheral. Considering the robotic steering operation, it was noted that the torque value was not sufficient in order to move the steering handle, hence torque value could be modeled as a function of states which should increase as the roll angle also increases for improved stabilization. One of the methods used in this process was transferring the learned model from the software testing process to the bicycle hardware and testing it. The possible improvement here could be either training the model directly using the physical bicycle model or making the custom bicycle virtual environment to be modelled near to physical bicycle conditions.

CHAPTER 6

DISCUSSION

6.1 Conclusion

Bicycle or two-wheeler stabilization is a complex task that requires deep understanding of methods by which control of bicycle can be achieved in conditions of riding with or without human rider. The other aspect which needs to be taken care of is bicycle control when the bike is stationary as well as in motion.

Previous work on the bike achieved stationary balancing using a Control Moment Gyroscope (CMG) with assisted robotic steering mechanism. This system extends on the previous work and is capable of working individually without any assistance from the CMG related hardware. This work has deep deterministic policy gradient algorithm implemented with Tensor Flow 2 which gives the capability of robust model deployment and ease of use which decreased the required code lines significantly and boosting the code execution on the Jetson hardware. An outdoor test was conducted to verify the system's capability and its performance in conjunction with deployed algorithm. The results verification was successfully performed by plotting them as shown in results section of hardware testing and also by visual inspection while performing the test. The tests were performed in a controlled and safer environment without posing any danger to any living thing.

This thesis represents a significant improvement from the results presented in the Cam *et al.* (2013) which used different reinforcement learning algorithm for balancing task but have some disadvantages highlighted before and most importantly there are no hardware implementations done for this prior to this thesis research.

Ultimately, bicycle research is a good platform for future investigation of controls and human-robot interaction. The system designed is highly flexible for future additions because of modular design and can be upgraded to make it more optimized.

6.2 Future Work

Future work will include implementation of the new model with the gaussian noise that was trained and gave better performance than the previous model on the hardware system, adding capability to code in order to perform navigation task and driving the bicycle to a particular goal, tested with and without obstructions on the path. This refers to trajectory planning and control essential for autonomous driving application. This would require designing of complex and robust reward function which includes the goal term added with a certain variable coefficient according the level of importance given to it while conduction of the task.

It would also include integration of the bicycle subsystems to provide balance assistance across the full range of bicycle speeds, with and without a human rider. Implementation of robust robotic steering system will help the bicycle to balance on uneven road conditions and counter other disturbance effects.

Addition of autonomicity to bikes would enable it to extend its horizon for work in the areas of delivery systems and bike-sharing which would help in increasing the sustainability. Due to the expandability of the Pixhawk flight controller, GPS and autonomous driving can be easily implemented.

Another area of interests lies in the communication of the bicycle systems with the vehicles and infrastructure to establish safer transportation system on street and highly crowded regions with inbuilt tracking.

REFERENCES

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., & Zaremba, W. (2016). *OpenAI Gym.*

Cam, B., Dembia, C., & Israeli, J. (2013). *Reinforcement learning for bicycle control.* Retrieved from http://cs229.stanford.edu/proj2013/CamDembiaIsraeli-RLBicycle.pdf

Choi, S. L. (2019). *Toward Self-Driving Bicycles Using State-of-the-Art Deep Reinforcement Learning Algorithms.* Symmetry, 11(2), 290.

Deng, W., Moore, S., Bush, J., Mabey, M., & Zhang, W. (2018). *Towards Automated Bicycles: Achieving Self-Balance Using Steering Control.* ASME 2018 Dynamic Systems and Control Conference.

Fawaz, Z., Smith, R., Muench, P., Lakshmanan, S., & Mohammadi, A. (2019). *Design and benchtop validation of an autonomous bicycle with linear electric actuators.* In Unmanned Systems Technology XXI (Vol. 11021, p. 110210B). International Society for Optics and Photonics.

Getz, N., & Marsden, J. (1995). *Control for an autonomous bicycle.* In Proceedings of 1995 IEEE international conference on robotics and automation (Vol. 2, pp. 1397-1402). IEEE.

Goslin, M., & Mine, M. (2004). *The Panda3D graphics engine.* Computer, 37(10), 112-114.

Hessel, M., Modayil, J., Van Hasselt, H., Schaul, T., Ostrovski, G., Dabney, W., & Silver, D. (2017). *Rainbow: Combining Improvements in Deep Reinforcement Learning.* arXiv preprint arXiv:1710.02298.

Introducing the self-driving bicycle in the Netherlands. (2016). Retrieved from YouTube: https://www.youtube.com/watch?v=LSZPNwZex9s

Lam, P. Y. (2011). *Gyroscopic stabilization of a kid-size bicycle.* In 2011 IEEE 5th International Conference on Cybernetics and Intelligent Systems (CIS) (pp. 247-252). IEEE.

Lillicrap, T., Hunt, J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., . . . Wierstra, D. (2019). *Continuous control with deep reinforcement learning.* arXiv.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). *Playing Atari with Deep Reinforcement Learning.* arXiv preprint arXiv:1312.5602.

Randlov , J., & Alstrøm, P. (1998). *Learning to Drive a Bicycle Using Reinforcement Learning and Shaping.* Proceedings of the 15th International Conference on Machine Learning. 463-471. .

Sanchez, N. (2020). *The MIT Autonomous Bicycle Project*. Retrieved from https://www.media.mit.edu/projects/AutonomousBicycleProject/overview/

Tsubouchi, T., Suzuki, H., Koyanagi, E., & Yuta, S. (2001). *An Experimental Autonomous Bicycle and Its Stabilized Moving.* IFAC Proceedings Volumes, 34(4), 69-74.

Uhlenbeck, G., & Ornstein, L. (1930). *On the theory of the Brownian Motion.* Physical review, 36(5):823.

Yetkin, H., Kalouche, S., Vernier, M., Colvin, G., Redmill, K., & Ozguner , U. (2014, June). *Gyroscopic stabilization of an unmanned bicycle.* In 2014 American Control Conference (pp. 4549-4554). IEEE.

Zhang , W., Moore, S., Jonathan, B., Mabey, M., & Wenhao, D. (2020). *Robotic steering mechanism for autonomous bicycle.* U.S. Patent Application No. 16/586,468.

Zhang, Y., & Yi, J. (2010). *Dynamic modeling and balance control of human/bicycle systems.* In 2010 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (pp. 1385-1390). IEEE.

APPENDIX A

BICYCLE PARAMETER VALUES

Parameter	Parameter Explanation	Numerical values
c	Horizontal distance between front tire and CM	1.0653 m
h	Height of CM over the ground	0.45782 m
l	Distance between front and back tire	1.11 m
M	Mass of complete bicycle system	63.797 kg
r	Radius of tire	0.34 m
T	Torque	[-2 , 2] N-m
v	Velocity of bicycle	10 km/hr
g	Acceleration due to gravity	9.82 m/s^2
M_d	Mass of tire	1.7 kg
r_f	Radius for front tire	10^8 m
r_{CM}	Radius for center of mass	10^8 m
r_b	Radius for back tire	10^8 m

APPENDIX B

PROJECT GITHUB REPOSITORY

The complete code for OpenAI gym pendulum testing can be found in this GitHub repository:

https://github.com/Shubhamturakhia/OpenAI-Pendulum-testing-using-DDPG

The complete code for bicycle testing using the deep deterministic policy gradient algorithm can be found in this GitHub repository:

https://github.com/Shubhamturakhia/RLbike