Investigating Machine-Learning Oriented State Predictors for A Novel Inverted Pendulum System In ROS

Ziyu Du

Master of Science

School of Informatics
University of Edinburgh

2021

Abstract

The constraints of the traditional inverted pendulum system limit its application to balance problems in many scenarios such as service robot water transportation, but the use of some exact systems to model these scenarios will make the problem more complicated and difficult to calculate. To balance the two of them, a novel system for better modelling these scenarios as well as some explorations of balance problem based on this novel system are required practically.

In our project, we proposed and established a novel bowl and ball structure extended from classic inverted pendulum system to model these scenarios above. Based on the bowl-ball system, we create several state predictors extension to the sequence model RNN, LSTM and GRU. To find the optimal state predictor, we have compared the three of them and its comparison shows that LSTM-based predictor outperform for predicting the state of the ball when robot is moving in linear motion, while the performance of GRU-based predictor is more superior to others on turning motion.

Acknowledgements

First of all, I would like to extend my sincere gratitude to my supervisor, David Symons, who has provided me with valuable guidance in every stage of the project. Without his enlightening instruction, impressive kindness and patience, I could not have completed my thesis.

Second, i would like to express my gratitude to all the authors of the open source tools used in my projects. Without their remarkable works (Keras,Scikit-learn,ROS,etc.), the pletion of this thesis would not have been possible.

Finally, i am indebted to my parents for their accompanies, supports and encouragements, i love you.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Ziyu Du*)

Table of Contents

Chapter 1

Introduction

1.1 Motivation for the project

The inverted pendulum(IP) system is a classic high order, nonlinear, strong coupling and naturally unstable system which are often used as a experimental platform for verifying the methods and theories of various balancing control algorithm[4]. In the majority of cases, inverted pendulum system is composed of a pole and a mobile cart with bidirectional movement directions and the many successful controls of specific scenarios such as robotic arm[32] and aerial rocket launch[17] are based on the rod balance problem[4] of this system.

However, the constraints on the bidirectional movement of this cart and the fragile balance of the pole are too restrictive to model some other scenarios, especially for those scenes that require the robot to move with high degrees of freedom and have a high tolerance for balancing objects[18]. For example, warehouse robots transports items to target location or service robots carry drinks without spilling them. On the other hands, the exact models for these scenarios are hard to establish theoretically due to the physical properties of objects, for example, evaluating the fluid dynamics equations of liquid to predict whether it will spill is extremely complex in real time[19].

To solve this problem, we propose a novel bowl-ball system extended from classic inverted pendulum system to model the balancing problems mentioned above. Using the proposed novel system, we assumed that this balancing task could be reduced to that of keeping a ball within a hemispherical bowl mounted on the robot, while this robot could move freely.

Clearly, different velocity and direction of the robot could drive different forces on the ball, which leads to the possibility that the ball may slide out of the rim of the bowl.

To prevent the ball from moving out of the edge of the bowl and keep it in the bowl, a predictor to forecast the future state of the ball based on the current and previous information will be required.

1.2 Project objectives and Results

In our project, the objectives could be divided into two main parts. Firstly, in order to realize the expectation that the model can be used in wider scenarios we establish this proposed bowl-ball model on GAZEBO(a simulator on ROS) and my work has been open source in my github account[1] for the convenience of citing by others, anyone in need can fork it directly.

Secondly, optimal state predictors are required to forecasting the future state of the ball to avoid sliding out and keep it in the bowl. In our project, we develop a RNN[13] sequence model as our baseline and its optimized results are compared with LSTM[12] and GRU[9] which shows that LSTM predictor is suitable for forecasting the state of the ball when the robot is moving linearly, while GRU predictor outperform than others when only turning.

1.3 Document structure

The remainder of this thesis is structured as follows: Chapter 2 gives some background of our project including both modelling and time series prediction. Chapter3 reports the steps of implementing the simulation bowl-ball system in details as well as the configuration of environment and basic mechanism of ROS for reproduction. Chapter 4 specifically illustrates the manual control strategy for the mobile robot of the simulated model, including the development steps and theory of our control algorithm. Chapter 5 describes the data collection of our project, including how could we select features in concept ,how to perceive and record feature-based data and the related techniques. In Chapter 6, we developed and optimized the baseline of our project. Baseides, dataset information and preprocessing of data are illustrated. In Chapter 7, we compared and analyzed the results from RNN, LSTM and GRU. In Chapter 8, we summarises our conclusions and reported some potential limitations of our project, and finally, we also discussed the future work.

[1] $https://github.com/Dzy-HW-XD/Waiter-Balance\_project$

Chapter 2

Background

2.1 Modelling

Modelling is the first step of the whole project, this requires that the quality and reliability of modeling is significantly important. However, according to the literature we have reviewed, there are no ready-made models that can be applied into our projects directly, which means the understanding of the existing related model and its variants should be noted and significant.

Classic inverted pendulum system, consisting of a balancing pole and linear moving cart, also called the pole-cart system is a variable, high order, nonlinear, strongly coupled, naturally unstable system, which often regarded as the classic benchmark model for the study of control algorithms, widely used in the field of industrial fields. For example, the research results by Boubaker[4] in 2012 has been applied to the verticality control in rocket launch. Besides, this classic inverted pendulum system has good expansibility, and these novel variants could be used as a reliable research benchmark in a completely different areas. For example, Hehn and D'Andrea[11] in 2011 proposed a novel flying inverted pendulum system, replacing the moving cart from classic one into quadrotor aerial vehicle, which has been applied for exploring the optimal aircraft control strategy in the field of aerospace. And Chen et al.[7] proposed a extended spring load inverted pendulum for investigating the control strategy of biped robot in the field of humanoid robot. Inspired by these variants, a novel inverted pendulum model will be expanded from the classic one in this project based on some hypothesis which has been discussed specifically in following section.

2.2 Time Series Prediction

Time series is a cluster of data recorded by the same unified indicator in chronological order[5]. The idea of the time series prediction(TSP) is to use the past time series data for statistical analysis to infer the development trend [33] or the specific values in the future period of time. However, implementing a reliable time series prediction usually are considered difficult, because it is almost impossible to fully understand the complex relationship between features in both explicit and implicit[2], especially in the high dimension. However, with the rapid improvement of computer computing speed and the wide popularity of graphics processing unit, some special structure neural networks seem to be superior solutions to predict high dimensions and nonlinear time series data.

Neural Networks as a branch of machine learning could be considered as a simplified model that simulates the way human brain processes information[1]. Normally, artificial neural networks are composed of input,output and multiple hidden layers, and each layer is composed of several interconnect processing units(neurons) with weights. The training process of NNs is simple and intuitive which is implemented by back-propagation algorithm proposed Werbos[34] in 1976. In Weigend's introduction[33], combining with the gradient descent method, backpropagation(chain rule) algorithm is used to optimize the neural networks by updating the weights of neurons repeatedly and iteratively until the response of the network to the input reaches the predetermined target range, which is determined by the calculated gradient of objective function to weight vector. However, artificial neural networks are not suitable for time series prediction, because there is no sequence relationship between each training sample.

2.2.1 Recurrent Neural Network

Recurrent neural network proposed by [13] is a special neural networks with sequence data as input, recursion in the evolution direction of the sequence, and all nodes are connected in a chain. Compared with the artificial neural network, Sola and Sevilla[3] has summarized that the biggest feature of RNN is that it has dynamic memory. This characteristics of dynamic memory could be simply explained in Fig 2.1 that RNN takes the output of each time point as the input of the next time point, so when inputting the nth of the last time point, RNN may remember the information of the previous n inputs. And this dynamic memory fits the characteristics of memorizing time series information for time series prediction task.

From now, many time series prediction tasks in different fields RNN has been

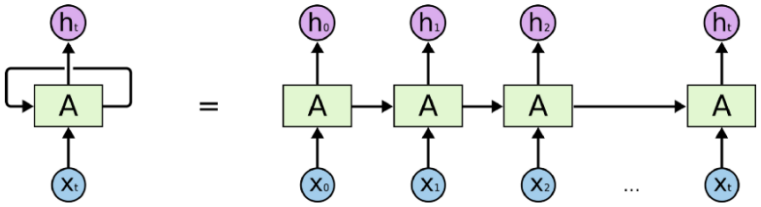

Figure 2.1: The Structure of Simple RNN Referenced from [22]

applied to time series prediction in many fields. For example, in 2018, Alper and Gözde[28] has applied RNN to forecasting the electricity load in Turkish market and they conclused that RNN outperformed the traditional time series algorithm AutoRegressive Integrated Moving Average(ARIMA) with 5% improvement. And in 2017, Selvin et al.[25] has applied RNN into forecasting the stock price and the performance of RNN is superior.

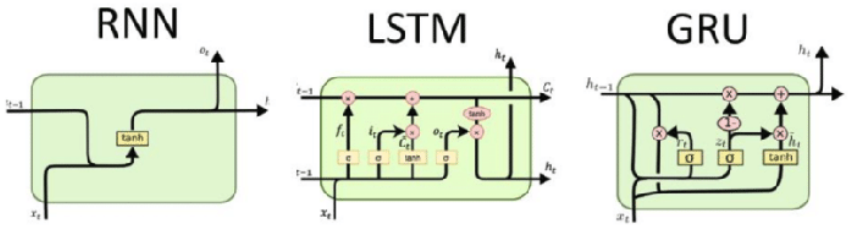

Figure 2.2: The Unit Structure of RNN(Left), LSTM(Mid) and GRU(Right) Referenced from [27]

2.2.2 Long short-term memory

Long short-term memory(LSTM) proposed by [12] could be considered a special RNN with additional gate structures shown in Fig 2.2. Hochreiter and Schmidhuber[12] has illustrated that unlike ordinary RNN, which only has one memory superposition mode, LSTM controls the transmission state through these three different gate states, keeps the information that needs to be remembered for a long time and forgets the unimportant information.

$$i_t = sigmoid(W_i X_t + U_i h_{t-1} + b_i) \tag{2.1}$$

$$f_t = sigmoid(W_f X_t + U_f h_{t-1} + b_f) \tag{2.2}$$

$$o_t = sigmoid(W_o X_t + U_o h_{t-1} + b_o) \tag{2.3}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot tanh(W_c X_t + U_c h_{t-1} + b_c) \tag{2.4}$$

$$h_t = o_t \odot tanh(c_t) \tag{2.5}$$

2.1 and 2.2 are the input and forget gate respectively which are used for determining the cell state2.4. Also, combining the output gate2.3 with the cell states2.4, the hidden states will be produced from 2.5, which contain the information of long term memory.

2.2.3 Gated recurrent units

Gated recurrent units proposed by [9] could be considered as the simplified LSTM fewer parameters. Junyoung et al.[9] in 2014 has introduced that there are only two different gates: reset gate and update gate, while the output gate is absent in GRU.

$$z_t = sigmoid(W_z X_t + U_z h_{t-1} + b_z) \tag{2.6}$$

$$r_t = sigmoid(W_r X_t + U_r h_{t-1} + b_r) \tag{2.7}$$

$$h_t = z_t \odot c_{t-1} + (1 - z_t) \odot tanh(W_h X_t + U_h(r_t \odot h_{t-1}) + b_h) \tag{2.8}$$

GRU could use the same gate(update gate)2.6 to forget and select memory at the same time, compared with LSTM using two different gates. And these principle has been illustrated intuitively by [22] that the reset gate2.7 determines how to combine the new input information with the previous memory, and the update gate2.6 defines the amount of previous memory saved to the current time step. Junyoung et al.[9] also noted that LSTM and GRU are difficult to explain which is better under the under specific scenarios and problems, some extra evaluations are required.

Chapter 3

Simulation Model Implementation

3.1 Set Up Environment

Before modelling it is necessary to list some configurations used in this project,such as programming language, library, Application Programming Interface. Some of them are independent but most of them are dependent on a specific compilation version. In this section, the detailed software development environment will be listed below to facilitate the compatibility between them and reproduction for other explorer.

Environment Configuration	Version
Linux	Ubuntu 18.04LTS
Robot Operating System(ROS)	Melodic Morenia
Gazebo	9.0
Python	2.7/3.6
Tensorflow	1.11.0
Keras	2.2.0
Scikit-Learn	0.24.0

Table 3.1: Project Environment Version List

Moreover, a certain amount of computing resources are needed, in this project, we will not develop calculation to cloud servers, but do it in local machine. The equipment used for calculation are intel(R) i7-10875H with 4 cores 16 threads and Navida Geforce RTX2060.

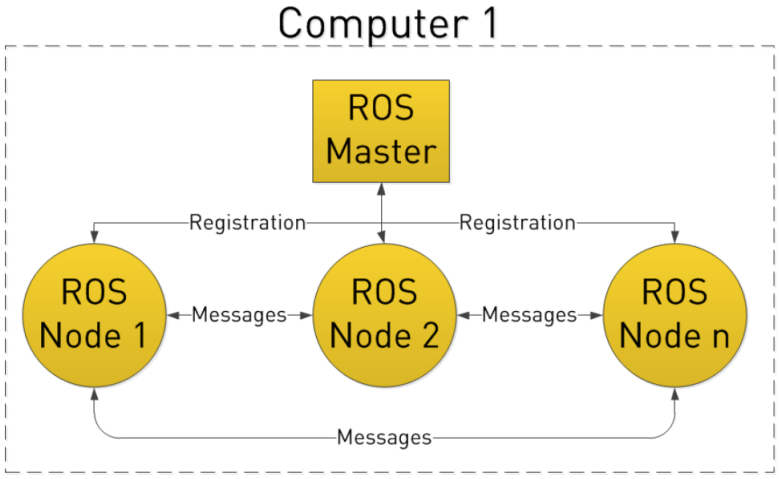

Figure 3.1: The Relationship Between Nodes and Master in ROS[24]

3.2 Basic Mechanisms of ROS

This section aims at illustrating several basic mechanisms of ROS including basic components of ROS and its communication mechanism which are used frequently in our experiments, for those who are newcomer to ROS.

3.2.1 ROS Nodes and Master

Node is the smallest unit in ROS. On the one hand, a node is an executable file that is usually coded in C++ or python; on the other hand, a node is used to achieve the cluster of functions for robots. However, in a practical project, due to the large number of components and functions of robots, We usually deploy a function distributed to a single node, which leads to an explosion in the number of nodes. Therefore, how to reasonably allocate and manage these nodes will determine whether the project can run smoothly and efficiently.

ROS provides an efficient node management method called ”master” to avoid the hazards above. In ROS, the master requires the node to register point-to-point, and then only successfully registered nodes are allowed to join the ROS environment. Figure3.1 has well illustrated the relationship between the nodes and the master. It should be noted that the uniqueness of the master leads to the collapse of the whole project once the master crashes, while the crash of a node will only affect the function of this node and will not affect others.

3.2.2 ROS Communication Mechanism

Communication mechanism enables nodes to transfer information, and there are three commonly used communication modes of ROS: topic communication, service communication, parameter service.

- Topic communication, also named publish-subscribe or talker-listener communication is the most commonly used communication method in ROS. This method follows one-way asynchronous mechanism, transmitting messages from the publisher node(talker) to the subscriber node(listener) without requiring any feedback. In the field of robotics, the broadcast characteristics of topic communication is very suitable for real-time periodic message transmission, such as sensor data reception and perception. Apart from that, in our project, the implementation of sensor sensing and data collection are based on this method.

- Service communication also called request-reply or client-server communication is often used for temporary and aperiodic scenarios, such as switch sensor, photographing, inverse solution calculation. Compared with topic communication, service communication is bidirectional and resource intensive. The requester node (client) will send a request through server channel to the respondent(server) and start blocking until receiving a reply. Besides, unlike the topic communication method with many to many mapping relationship, server communication only allow multiple clients to send requests to one server. In our project, there is no suitable scenario for server communication due to its request-reply mechanism, and the communication between nodes is achieved by topic communication method.

- Different from the first two communication modes, the parameter server can also be said to be a special communication mode. The special point is that the parameter server is the place where nodes store parameters, configure parameters and share parameters globally. Compared to topic and service communication, parameters server uses internet transmission and runs in the node manager to realize the whole communication. Besides, each configuration is stored on this parameter server as a key value pair and programmers can view each set of configurations through tool ”Rosparam” provided by ROS. Apart from that, in our project, this parameter server is often used without feeling it. For example, in the process of building simulation model, some significant parameters in the robot

description file will be stored in the parameter server and called when gazebo starts.

3.3 Building Simulation Model

Simulation is the virtual representation of real things or processes. After the development environment has been configured, building a reliable simulation model is a significant start of our whole project.More specifically, our simulation model is required to represent the key features of the selected physical or abstract features, including accurate dynamic characteristics, appearance, structural mechanics and material. In this section, our project will follow several steps to build a reliable simulation model: expected model description, model component creation, component stitching,

3.3.1 Model Hypothesis and Description

As we discussed in introduction and background sections, the scenario of this project is similar to an inverted pendulum system, but the differences are that we require the simulated model to be used in a wider research context or application scenario, such as moving items in a warehouse or having a robot waiter carry drinks without spilling them, while inverted pendulum system can only do linear motion and its balance is obviously fragile, Pole-Cart system is not applicable to these scenario above, therefore, in order to satisfy the corresponding requirements, the model established should meet the following conditional hypothesis or constraints:

1) This simulated model will be an extension of the inverted pendulum system, which means it has the same composition as an inverted pendulum system, a cart and balance components which is used to achieve the balancing task.

2) Mobile robot(cart) can move with higher degree of freedom compared to inverted pendulum system with only forward and backward freedom.

3) Like the inverted pendulum, the balance components are highly coupled to the entire simulation system, also has a much looser balancing condition than the inverted pendulum system.

To meet these three constraints, We assume a novel model shown in figure 3.3.3.1. This figure illustrates that our ultimate system is a balance system consists of two

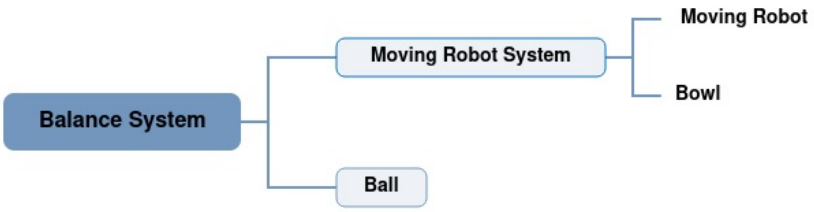

Figure 3.2: The Structure of Balance System

components, moving robot system and ball, where mobile robot system is a combination of mobile robot and bowl. Specifically, the mobile robot ensures high degree of freedom motion mounting the bowl. And many external factors such as acceleration, speed,friction, etc,guarantees the nonlinear motion of the ball in the bowl and the high coupling between the ball and the robot. It should be noted that in order to easily perceive the attributes of the ball, the ball should be combined with the mobile robot system at the robot level, which means the simulation environment could recognizes two robots without any joint connection. While the bowl and moving robot will be stitched together at link level to guarantee that they are connected in some joint way.

3.3.2 Components Creation

There is no open-source model on the online community that can be used directly. To solve this problem, in this subsection, we are aiming at creating these three components, moving robot, ball and bowl respectively and preparing for model integration.

3.3.2.1 Mobile Robot

As for moving robot part, the first thing we think of is the differential mechanism to satisfy these assumptions above. This mechanism allows the left and right wheels to rotate at different speeds which is necessary when the vehicle turns, making outer drive wheel to rotate faster than the inner drive[31]. Compared with only two movement of freedom(forward and backward) for cart of the inverted pendulum system, the differential cart can easily achieve higher degrees of freedom of movement(forward and backward,turning left and turning right).

There are several open source differential robots available on online community such as Turtlebot,Pioneer. In this project, we use Turtlebot3-waffle[1] as the mobile

[1]https://emanual.robotis.com/docs/en/platform/turtlebot3/overview/notices

robot of our whole system, which is shown below.

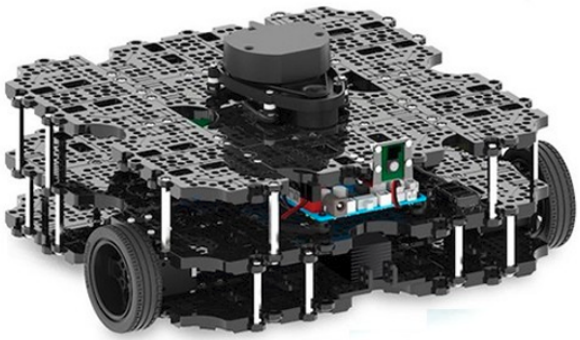

Figure 3.3: Turtlebot3-Waffle Mobile Robot(courtesy of [30])

Turtlebot3-Waffle is a ROS-based mobile and programmable robot which is easy to achieve differential control by calling application program interface that will be discussed in following sections. In this project, we fork the open source library of turtlebot3 [29] and make some modifications based on it in the following relevant steps.

3.3.2.2 Ball

Ball is an indispensable component of the balance system which is used for collecting data and verification. As we already discussed from previous section, the ball as a robot should be modelled in robot level.

Gazebo provides an applicable interface to create basic geometry objects,such as cube, cuboid,centrum etc. However,we can not directly use gazebo to build the ball model, because the model built on gazebo is saved in the .world file which makes it impossible for the environment to treat it as a separate robot. Also, this special file format regards ball as an object on external environment, making the model unable to be compiled directly by ROS so that we can not get some useful feedback information when the ball is moving in simulation world.

To solve this problem, we are trying to describe the ball model in Unified Robot Description Format(urdf.xacro) and gazebo format(gazebo.xacro) file which can be directly complied by ROS. It is well worth emphasizing that urdf file is a robot model file described in XML format which is used describe the robot model as a combina-

Properties of the Ball	Value
Mass(kg)	0.00001
Radius(m)	0.015
Coulomb Friction Coefficient for First Direction	0.4
Coulomb Friction Coefficient for Second Direction	0.4
Stiffness coefficient(N/m)	5000000
Damping coefficient(N*s/m)	10.0
Material	Wood
Minimum plunge depth	0.001

Table 3.2: The Properties of the Ball

tion of links and joints parsed by c++[16]. While gazebo file is used to define the simulate physical properties of robot and introduce plugin used to extend the function. urdf.xacro and gazebo.xacro are two important file formats in ROS and Gazebo, and operations based on these two file formats are extremely frequent in the following experiments.

Stiffness coefficient is the basic physical quantity used to describe the elastic deformation of materials under external force and damping efficiency represents the degree of energy consumption of vibrating structures(the ball).Besides, minimum plunge depth is a threshold that determines whether there is contact force. Moreover, ”Stiffness coefficient”, ”Damping coefficient” and ”Minimum plunge depth” are not adjustable, because these three determine the stiffness and strength of the ball, while friction coefficient for first and section direction will be changed with the turning of “Mass” and ”Radius” to ensure efficient interaction behavior of the ball. As we can see in table 3.2, in this project, values of ”Mass”, ”Radius” and ”Friction” are applicable but not optimal. During the reproduction process, you can adjust the parameters yourself through visual feedback. Also you can change the material of the ball to whatever you like.

3.3.2.3 Bowl

Another balancing component of our model is the ”bowl”. Similar to the joint between the pole and cart of the inverted pendulum system, the bowl as the carrier plate is used to contact with the ball and generate the interaction force to make the ball state non-

linearly change. Besides, the bowl also plays a role in calculating specific attributes such as relative position and velocity between bowl and ball.

In order to perceive the data of the bowl model in the simulation environment, we do not directly insert the bowl model from gazebo model base in this step, but use the interface provided by ROS. It should be noted that this powerful interface can also insert custom models with various 3D file format such as dae, stl, obj, which guarantees the controllability of the shape of the bowl. The bowl constructed by Solidwork shown in figure 3.4. It is well worth emphasizing that we smooth the surface of the bowl in order to simplify redundant and complex motion to make model more universal. For example, we ignore the effect of the force of the bulge in the bowl on the ball, making the state of the ball more predictable in the following experiment.

Another potential problem in this step is to calculate the inertia matrix. Unlike the inertia matrix calculation of the ball in previous step, the calculation of inertia matrix of bowl is extremely complex and abstract due to its irregular structure.

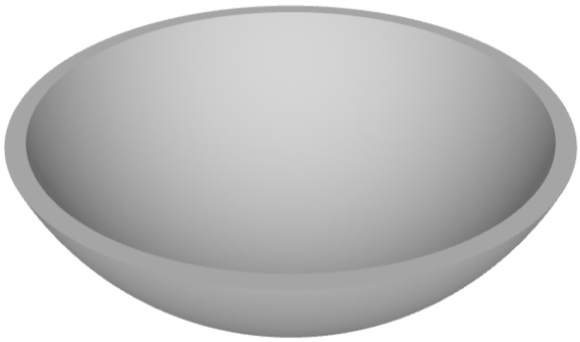

Figure 3.4: View of Bowl-like Structure

In this step, we are trying to use external tools to solve the problem. Fortunately, an open source tool called "meshlab"[2] can be used to process stereolithography file and compute the inertial parameters by easily following and clicking buttons on the pipeline: Filters → Quality Measure and Computations→Compute Geometric Measures. The calculated inertia tensor matrix shown in figure 3.3. However, these values can not directly used for gazebo plugin. Inertia values in xx,yy,zz direction are extremely large because they are calculated based on the (raw)original size of the bowl which is inappropriate for components combination. To solve this obstacle, we will process this set of data and resize the bowl in following step before we apply them to

[2]https://www.meshlab.net/

our simulation system.

	x	y	z
x	212267040.000	153.866318	-0.960848
y	153.866318	212284960.000	27.004116
z	-0.960848	27.004116	373092192.000

Table 3.3: The Inertia Matrix of Raw Bowl

3.3.3 Components Stitching

We have got a series of components, moving robot, the ball and the bowl from last section. However, the presence of these components in the environment alone without any connection is meaningless for our experiments. In this step, following the guidance shown in figure 3.8 we are aiming at combining these three components separately to establish and initialize a new balance system.

3.3.3.1 Stitching Cart and Bowl

Cart and bowl Components should be stitched together as the mobile robot system shown in figure 3.2. This requires not only the appearance of the bowl and the mobile robot as a whole, but also the sharing of the same and correct dynamics properties. In this sub-step, the moving robot model can be expanded and connected to other links through ”joint” technique which provides an interface for external components. However, we can not directly apply bowl into our moving robot due to its inconsistent sizes which has been discussed in previous section. So, some modifications on the size and inertia of ”bowl” should be done before building ”joint” connection.

	x	y	z
x	5.78	4.154e-6	-2.6e-8
y	4.154e-6	5.7	7.29e-7
z	-2.6e-8	7.29e-7	10.074

Table 3.4: The Inertia Matrix of Bowl After Scaling

The process of adjusting the size is intuitive and unconstrained because the result of the adjustment is directly fed back from Gazebo by modifying super-parameter ”scale”.

```
robot name is: turtlebot3_waffle
---------- Successfully Parsed XML ---------------
root Link: base_footprint has 1 child(ren)
    child(1):  base_link
        child(1):  bowl_link
        child(2):  caster_back_left_link
        child(3):  caster_back_right_link
        child(4):  imu_link
        child(5):  wheel_left_link
        child(6):  wheel_right_link
```

Figure 3.5: The Structure of Moving Robot System

```
robot name is: ball
---------- Successfully Parsed XML ---------------
root Link: base_ball_link has 1 child(ren)
    child(1):  ball_link
        child(1):  ball_imu_link
```

Figure 3.6: The Structure of the Ball

After several attempts, it can be seen that after the xyz directions of the bowl are scaled to 0.003, the size of the bowl is optimal, bowl mouth is almost equal to the area of base robot on the xy plane. And the optimized inertia matrix will become the 0.003^3 of raw inertia matrix(table 3.3), shown in table 3.4.

If the bowl falls off during exercise or structural collapse, the experimental results become meaningless for following experiment steps, because the measurement data has been completely contaminated by noise and cannot be used for algorithm training. To solve this problem, we plugin a fixed joints between bowl link and base link to ensure that the relative position of the bowl and the mobile robot will not change using a compatible interface provided by ROS.

Figure 3.5 has explained the result of this novel structure of mobile robot, consisting of a series of links in some connection. In more detail, base_footprint as the root link of whole mobile robot system has only child link called base_link which is recognized as the parent of all functional links shown in figure 3.5, such as bowl_link,imu_link.

3.3.3.2 Stitching Mobile Robot System and Ball

In previous section, mobile robot and bowl are connected at the link level. However, when stitching moving robot system and bowl, the ”joint” technique will not be able to be applied. This is because mobile robot system and ball should be placed in the

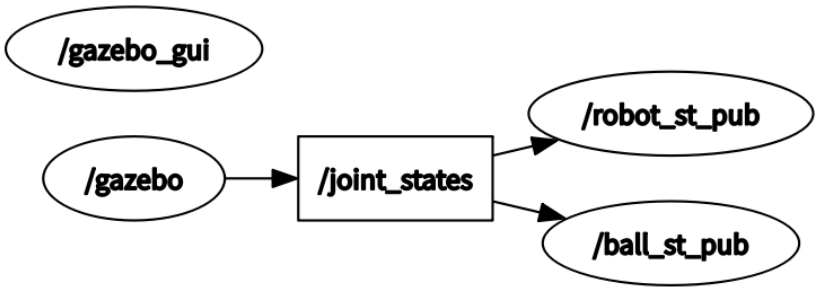

Figure 3.7: ROS Computation Graph for Two established nodes on ROS

simulation world as two different robots has been discussed in previous section.

A method called ”roslaunch” provided by ROS could solve this problem. ”roslaunch” is one of the ways to start the nodes of ROS. In this step, two ROS nodes, one for moving robot system, one for ball, will be established through the roslaunch mechanism. Further, activating these two nodes ensures the visualization of the robot in the simulation environment and the release of the status which will be used in following section.

Figure 3.7 has illustrated the relationship between node and topic when the simulation model is established and initialized. In ROS computation graph, these relationships are displayed visually and intuitively, ellipses represents nodes and rectangle represent published topic, also the arrow points from the publisher to the subscriber. More specifically, a node called ”gazebo” publishes a topic called ”joint_states” which contains link and joint information to the topic server, while other two nodes are subscribing to the information of the corresponding robot in topic respectively.

In the initial stage of the model, robots system can be visualized in simulation world shown in figure 3.8, the green part in this picture is the bowl of the moving robot, and the object at the bottom of a bowl is the ball we established in previous steps. Besides, the black part under the bowl is the turtlebot3 moving robot used for moving with high degree of freedom. To sum up, we have obtained a raw and novel balance system in both simulation and perception from last few steps. Base on that, in the following several sections, we will focus on improving the perception capability of the system and collect significant data for uses in machine learning algorithms.

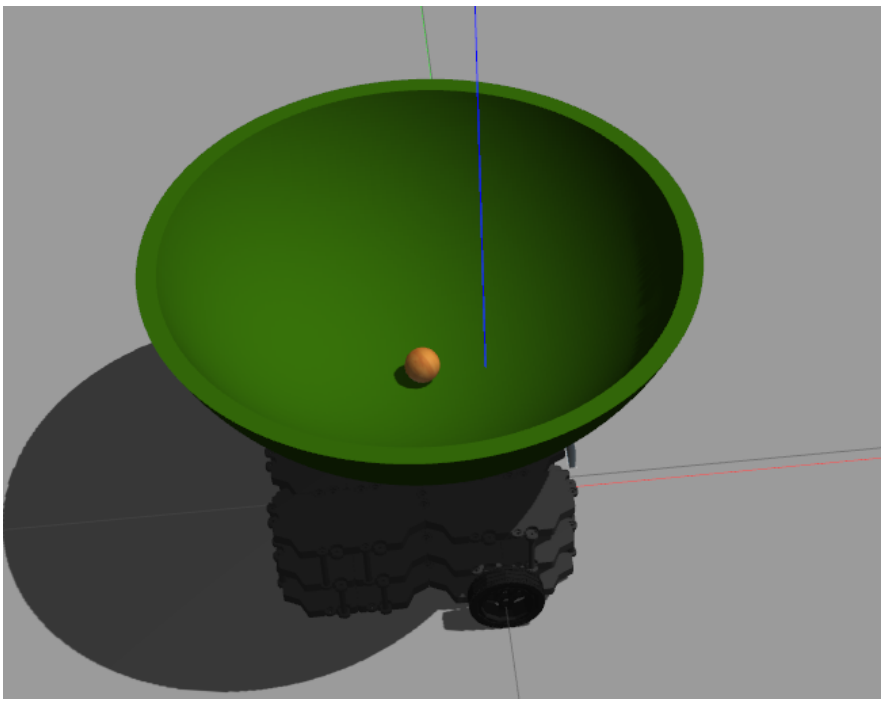

Figure 3.8: The display of our novel balance robot system on Gazebo

3.4 TF Modification

Transform also called ”TF” plays an important role in the field of robotics. In general, robot systems have many 3D coordinate axes that change with time, such as ground and base coordinates. TF is used to track these coordinates and find the association and mapping between coordinates to calculate significant geographic information. In our project, TF has provided great help for information acquisition and deep data perception, for example, we are able to obtain the geographic coordinates information of significant published link from the TF service, also some hidden information can also be calculated by tf, such as relative position information.

3.4.1 Raw TF Tree

TF is intuitively displayed in the form of TF tree which is organized in the form of parent-child coordinate system. The parent coordinate is at the top,while the child coordinate is at the bottom. Each coordinate system has a parent coordinate(except only root coordinate) and can have multiple child coordinate systems. In this step, we will put insights on analyzing the raw TF tree generated by this novel robot model and make full preparations for the next step of manual control and data collection.

The figure 3.9 illustrates the initialized TF tree of the robots we have obtained. As

we can see in this figure, there are two trees of different sizes, the small one describes the TF structure of the ball, and the large one describes the mobile robot system. However, this means that the two robots are independent of each other in terms of coordinate system, which leads to some implicit data that cannot be obtained through calculation, such as relative position between the ball and the bowl, relative velocity between the ball and the moving robot system.

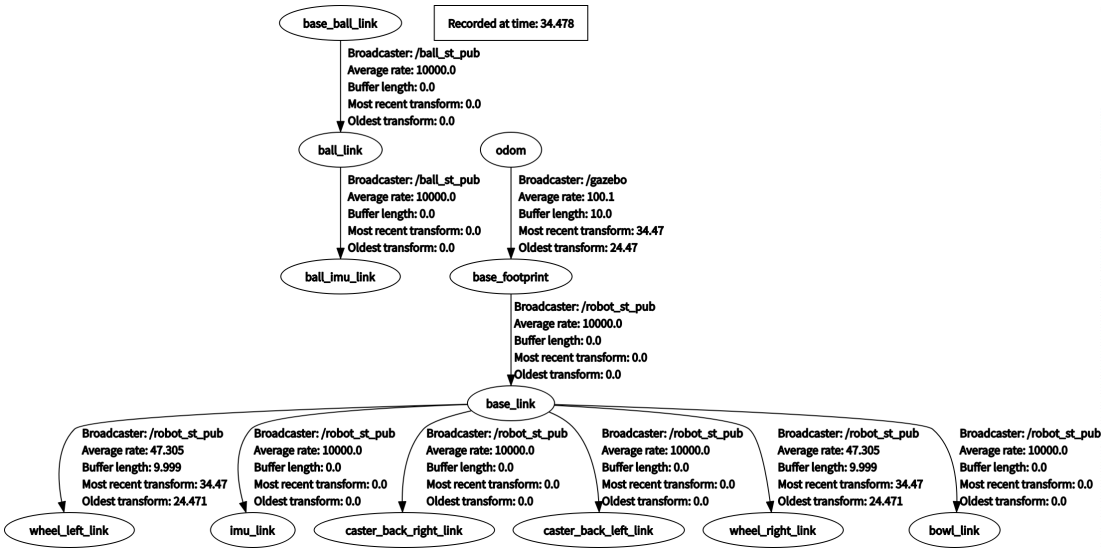

Figure 3.9: The Initialized TF Tree of Two Robots

One possible solution is to connect these two independent trees by sharing same coordinate(in ROS it can also be called frame). This means the relative position and rotation information of all links in the two robot systems can calculate by coordinate transform through this shared coordinate. For example, if base_link_ball of the ball is shared same coordinate with base_footprint, the position of ball_link of the ball in the bowl_link coordinate system can also be calculated.

3.4.2 Unifying coordinate system

Following the hypothesis discussed in previous step, we are aiming at exploring a suitable shared coordinate and combine and unify these coordinates of two robots through this shared frame. Exploration starts with the simulation world, a topic called ”/gazebo/model_states” is published by gazebo node. This topic contains the position and speed status of each robot in the simulation environment. More importantly, these position and speed states are published on a world coordinate which is coincidentally consistent with the coordinate of the root link of the moving robot. This discovery

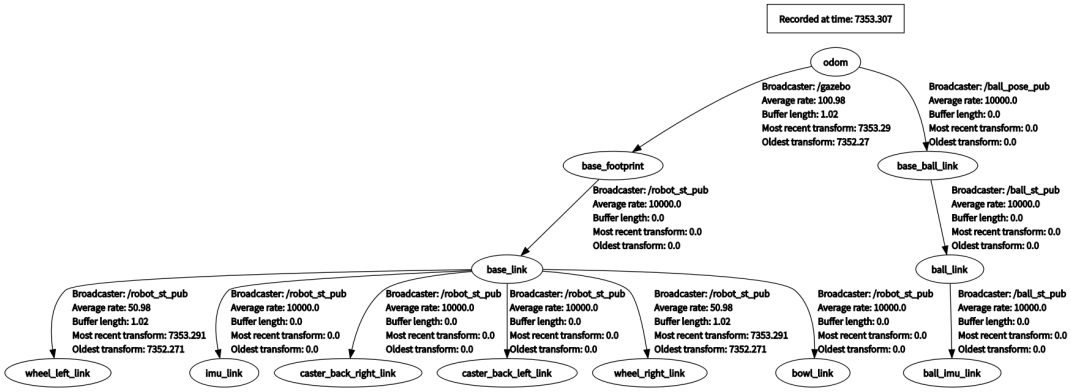

Figure 3.10: The Modified TF Tree of Two Robots

has made a significant contribution to the establishment of a unified coordinate system. More specifically, we are determined to map the geographic information of the ball in this topic to the same mobile robot coordinate system as the world coordinate mentioned before.

To implement this mapping, firstly, a new node called "ball_pose_pub" in figure 3.11 will be established for subscribing target "/gazebo/model_states" topic, every time the established node has listened to the information published in topic, the callback mechanism will be called and the monitored information will be passed into the callback function. Then, we extract the information of the ball from the passed data, and from the analysis in previous section, these extracted data can be considered as the translation and rotation information of the ball in the world coordinates which is same as the odometry coordinates("odom" in figure 3.9).

The results of unified coordinate system are shown in figure 3.10. As we can see

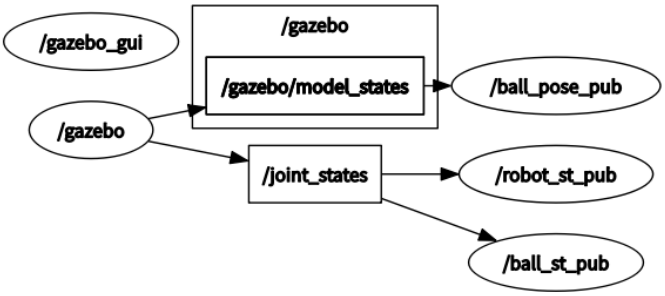

Figure 3.11: The Structure of Nodes and Topic after unifying coordinate system

in this figure, these two independent tf trees are combined together by sharing same root ”odom” coordinates and each sub-tree represents a single robot. The benefits of this integrated robot systems are that some changes of dynamic coordinates can be calculated directly for obtaining hidden attributes between the ball and the mobile robot in the following section.

Chapter 4

Manual Control Strategy

Manual control is a significant methodology for guaranteeing high-quality training and testing data cluster. If the mobile robot keeps the state unchanged, the state of the ball will not be changed either, because there is no trend of relative motion between this two objects, and these collected data based on this situation is completely meaningless for the training of machine learning algorithm. The purpose of manual control of moving robot is to continuously change the position of the mobile robot and record the high-quality attributes of the ball in different states for predicting the future states of the ball. Apart from that, another purpose of manual control of moving robot is to ensure the integrity of test data which will be discussed specifically in following section.

4.1 Differential Driver Plugin

Manual control can not be achieved without a differential driver on moving robot. Fortunately, this differential driver is one of the functional library provided by ROS which can be easily inserted into the moving robot system as a plugin. More code details can be found by yourself on turtlebot3_waffle.gazebo.xacro under the description file of turtlebot3[1].

One thing should be noted that this plugin requires a command topic called "cmd_vel" which will be used for delivering command to control this moving robot system. This topic consists of a geometry ROS message data type "Twist" which can be divided into two parts. As we can see in figure 4.1, one is linear velocity with x,y,z attributes, another is angular velocity of moving robot system with same attributes. The attributes x,y and z are the direction based on the coordinates of the base of robot which has

[1] $https://github.com/Dzy-HW-XD/Waiter-Balanc_project/tree/master/src/turtlebot3\_description$

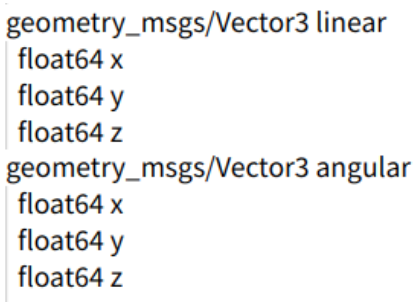

Figure 4.1: The Structure of ROS ”cmd_vel” Topic

been configured in differential driver plugin. However, one problem is that these six attributes are not fully used to control our moving robot system. This is because for robots placed on the ground, we ignore the need for upward movement in this experiment, which is generally needed in flying robots. This means the ”z” values in both linear and angular are always 0. Therefore, our robot only needs ”x” and ”y” values of angular velocity and ”x” value of linear velocity, so that it can move freely on the ground.

4.2 Control Algorithm

The manual control algorithm can be roughly described in the picture 4.2. Before demonstrating the details, several symbols in the figure need to be explained. In this figure, pink ellipses represents the states of our moving robot, yellow parallelograms and blue rectangle symbolize variables and calculation process respectively, diamond represents the judgment process of this algorithm. Besides, there are five states waiting to be called in control algorithm: Stop,move at constant speed, speed up,slow down, turn left and turn right. And the last four states are integrated into a gradient pink ellipse. Furthermore, the input of the keyboard determines the status of the moving robot system, as we can see in table 4.1, 5 input characters from keyboard are used to realize these five states of our moving robot system.

In the initial state, the variable of current velocity is keeping 0 until there is external keyboard input. Firstly, the target velocity can be calculated according to equation from table 4.1. Then, if the calculated target velocity is 0, our mobile robot system will tend to stop. If current velocity is equal to the calculated target velocity or this target velocity is greater than or equal to the upper limit of speed, our mobile robot system will moving without changing linear and angular speed. Moreover, in the next

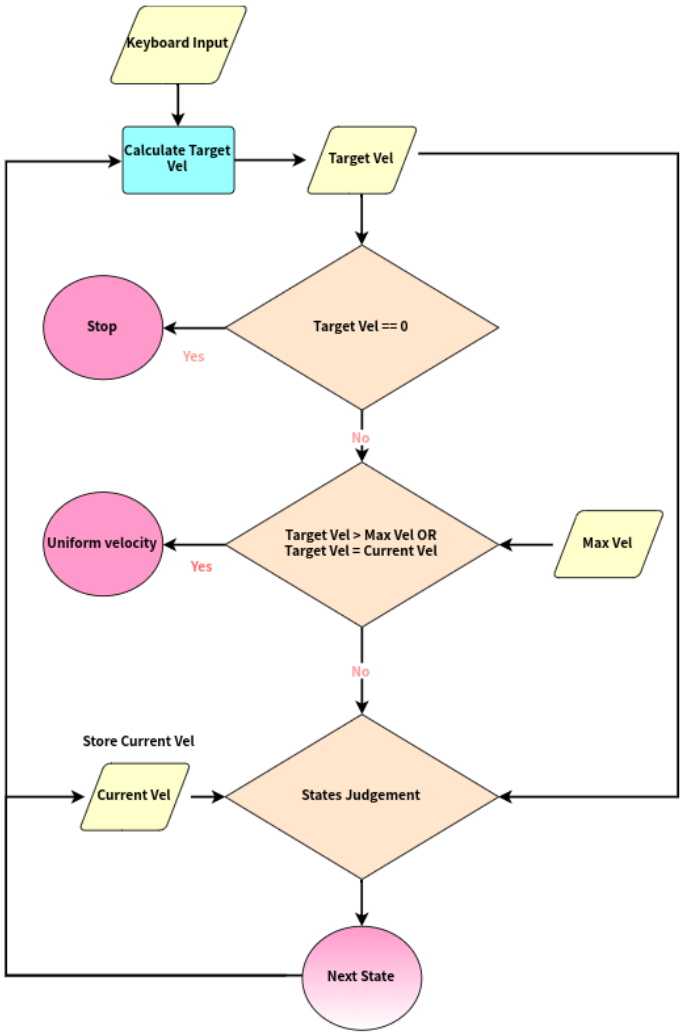

Figure 4.2: The Flow Chart of Keyboard Manual Control Algorithm

step, the status of the moving robot is determined according to the input characters from table 4.1. And the current status is stored in a buffer for the next comparison and target velocity calculation. So far, the control process of one signal input has ended, and the waiting for the next signal input has entered. This entire control process can be regarded as a feedback process, the result of one control process becomes the input of the next control. Apart from that, the setting of some hyper-parameters is also significant important and indispensable. The step size of linear and angular velocity as constants are used to adjust the speed has been set 0.1 and 0.4 meters per second respectively and the maximum linear and angular velocity of our moving robot has been set 5 and 4 meters per second respectively at the initial stage of simulation environ-

character	State	Target Vel
w	Speed Up	Target Vel = Current Vel + LIN_VEL_STEP_SIZE/2
x	Slow Down	Target Vel = Current Vel - LIN_VEL_STEP_SIZE/2
a	Turn Left	Target Vel = Current Vel + ANG_VEL_STEP_SIZE/2
d	Turn Right	Target Vel = Current Vel - ANG_VEL_STEP_SIZE/2
s	Stop	0

Table 4.1: The Computational Table of Keyboard Input

```
process[turtlebot3_teleop_keyboard-1]: started with pid [27524]

Control Your TurtleBot3!
---------------------------
Moving around:
        w
   a    s    d
        x

w/x : increase/decrease linear velocity (Burger : ~ 0.22, Waffle and Waffle Pi :
 ~5)
a/d : increase/decrease angular velocity (Burger : ~ 2.84, Waffle and Waffle Pi
: ~ 4)

space key, s : force stop

CTRL-C to quit
```

Figure 4.3: The Console for Controlling Moving Robot System

ment. Setting these hyper-parameters is not harsh which means there are no optimal values for these hyper-parameters. In our project, we have set these values to be small, in order to be able to better monitor the acceleration and deceleration process of the moving robot system. However, for those who want to reproduce the experiment, these hyper-parameters can be set more freely.

To achieve this control algorithm, we has quoted the open-source keyboard controller provided by willow garage, Inc. as a reference(available from turtlebot github libaray[2]) for controlling our moving robot system. And the console on terminal can be shown in figure 4.3, using this keyboard console, we can realize the free movement and speed control of mobile robot. A new node "turtlebot3 teleop keyboard" publishes a control signal topic, at the same time, Gazebo subscribed to this specific topic and our mobile robot system will move if a meaningful signal has been received.

[2] $https://github.com/ROBOTIS-GIT/turtlebot3/tree/master/turtlebot3\_teleop$

Chapter 5

Data Collection

Data is the core driver of training-based algorithm and the quality of data determines the precision and accuracy of this kind of algorithm. The main purpose of the data collection step is to collect significant training data as high-quality as we can for training process of our algorithm in following section. To achieve this goal, two potential problems are inevitable and worth discussing: What kinds of data could be collected as our training data and how to collect them efficiently and precisely. For example, considering the traditional inverted pendulum system, we will do not think that the temperature of the pole is strongly related to the predicted position of the pole. Similar to our project, some redundant attributes need to be selected and filtered before the training process. To answer these two questions, we are determined to divide data collection into four sub-steps, features selection, perception, timestamp synchronization and description of data. It should be noted that the perception step will illustrate the techniques of sensing attributes and introduce the impact of asynchronous problems which will be solved in the timestamp synchronization to ensure the high-quality and fidelity of the data.

5.1 Features Selection

What features are collected is determined by whether they will affect the predicted results. Firstly, considering that main purpose of the project is to predict the state of the ball using machine learning algorithm, the state of the ball must be trained into the algorithm for the supervised learning algorithm. From the Jain's explanation[14], kinetic energy expresses the ability of an object to do work at a certain time which contains the information of motion of object, while potential energy could be considered as a

State of the Ball
The Relative Position Between the Ball and the Bowl
The Relative Velocity Between the Ball and the Moving Robot
The Acceleration of the Ball
The Angular Velocity of the Ball
The Total Energy of the Ball
State of the Moving Robot
The Linear Velocity of the Movint Robot
The Angular Velocity of the Movint Robot
Control Signal
Linear Control Signal
Angular Control Signal

Table 5.1: The Attributes of the Collected Data

physical quantity that describes the position of an object and mechanical energy(sum of kinetic and potential energy) could be considered as the state of an object in the macro sense, which will be denoted as the output of our collected data.Then, for the whole system, there still are some dependent and independent attributes that have some hidden relationship with the prediction have not been proposed yet.

Table 5.1 has illustrated the remaining attributes of data to be collected. In this table, features can be divided into three parts, the states of the ball, the states of the moving robot and the control signal. More specifically, position, velocity and acceleration of the ball as the basic measurement properties of the ball, they are able to implicitly affects the trend of the future energy of the ball. Besides, in order to obtain the hidden relationship between mobile robot and predicted label, it is significant necessary to collect linear and angular velocity of the moving robot system as the attributes of the training set. Finally, the control signal is also indispensable as the only external input signal.

5.2 Perception

Perception provides a possible solution for sensing feature-based data we mentioned in previous section. Machine perception is a process in which robots obtain cognitive signals through the combination of sensors. Since the experiment is based on a sim-

Features	**Techniques**
The Relative Position Between the Ball and the Bowl	Dynamic TF conversion
The Relative Velocity Between the Ball and the Cart	Odometer Sensor
The Acceleration of the Ball	Inertial Measurement Unit Sensor
The Angular Velocity of the Ball	Inertial Measurement Unit Sensor
The Total Energy of the Ball	Odometer Sensor
The Linear Velocity of the Movint Robot	Odometer Sensor
The Angular Velocity of the Movint Robot	Odometer Sensor
Linear Control Signal	Subscribe /cmd_vel Topic
Angular Control Signal	Subscribe /cmd_vel Topic

Table 5.2: The Perception Techniques corresponding to features

ulation environment, our perception process can be greatly simplified, because some significant data has been published by the simulation environment, and we can directly obtain them from ROS servers directly without using sensors. So some hybrid perception techniques will be applied to perceive these target feature-based information.

These hybrid perception techniques has been shown in table 5.2. In this table, some difficulties have been solved well in the previous sections, for example, section 3.4.2 has established a complete TF tree and we can easily calculate the relative position between the ball and the bowl using dynamic TF conversion and section 4 has built a node that publishes control topic and the target control signal can easily be obtained by subscribing it. Besides, inertial measurement unit and odometer are two different sensors used for measuring angular rate, acceleration of an object and odometry information(position and velocity) respectively which can be easily applied to the corresponding robot as plugins as we discussed in section 3.3. It is well worth mentioning that the relative velocity and the total energy of the ball could not be acquired directly, but through some calculation process after perception from odometer and IMU sensor.

However, in the communication mode of ROS, if the data want to be collected, it should be published to the ROS server in the form of topic, which leads to an asynchronous problem caused usually caused by different release frequency and code structure. The asynchronous problem can be simply explained as the inability to obtain information at the same point of time(timestamp) and this potential problem will destroy the accuracy and quality of data.In our project, each feature to be collected in table 5.2

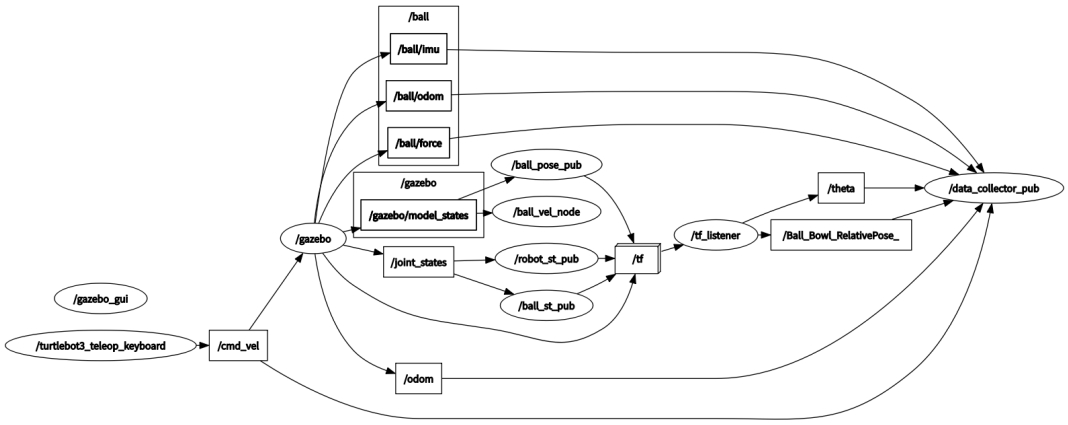

Figure 5.1: The Structure of Nodes and Topics After Timestamp Synchronization

is published by a node and this node continuously release information of target feature to a topic channel. If we directly read the information in two topics without subscribing them, when we pause the simulation world, it is possible to find that the latest data of the two topics will have different timestamps. To solve this potential problem, some additional technique need to be implemented in following section.

5.3 Timestamp Synchronization

Synchronization timestamp is a good solution for solving the asynchronous problem mentioned before. Specifically, a extra node will be established to subscribe and monitor all topic containing target features and the callback function is called only when the timestamps of all topics are the same.This technique does not require changing the rest of the code structure to eliminate computer runtime, however, one thing should be noted that the publishing frequency of each topic should comply with the multiple relationship with each other, because if it is not, the same timestamp will never appear. For example, a topic with 10hz publishing frequency will never contain the same timestamp as a topic published at 3 Hz.

For rapid implementation of synchronization timestamp, ROS has provided a convenient synchronization method called ”message filter” introduced in its official website[1], one thing should be noted that when the callback function is called, it means that the information of all topics has been synchronized, and the cluster of these syn-

[1] $http://wiki.ros.org/message\_filters$

geometry_msgs/Pose	RelativePosition
sensor_msgs/Imu	ball_imu
nav_msgs/Odometry	ball_odom
nav_msgs/Odometry	car_odom
geometry_msgs/Twist	cmd

Figure 5.2: The Structure of the Established ROS Message

chronized information is our target collection which will be published to a new topic defined by a new ROS message structure shown in figure 5.2. The implementation of establishing new ROS message is guided by the official tutorial[2]. Corresponding to table 5.2, ”cmd” contains the control signal; ”RelativePosition” contains the relative position information between the ball and the bowl calculated by TF conversion guided by ROS tutorial[3];”ball_imu” contains the acceleration and angular velocity measurement of the ball; ”car_odom” contains the velocity information of the moving robot and ”ball_odom” contains the ball’s, while both of them will be used for calculating the total energy and the relative velocity respectively at the step before the training process.

Up to now, the theoretical difficulties of collecting data have all been solved and the simulation model is ready to collect data. As we can see in figure5.1, all topics with target features are subscribed to by node called ”data collector publisher” which will be used as an interface to transform discrete attributes into trainable tuple during the process of moving.

5.4 Data Description

Data collection is the process of continuously controlling the mobile robot to record the topic published by ”data collector” which is established in previous section. In this process, the state of the ball is affected by many complex factors, such as acceleration, friction, rigid body collision and its own potential energy, which leads to the motion of the ball keeps nonlinear and complex and high coupling with the whole system. One thing should be noted that path of controlled mobile robot should include all situations as much as possible. Because if the training data only contains the linear motion of the moving robot, the state of the ball is unpredictable when the robot is turning.

[2] $http://wiki.ros.org/ROS/Tutorials/CreatingMsgAndSrv$
[3] $http://wiki.ros.org/tf/Tutorials/Writing\%20a\%20tf\%20listener\%20\%28Python\%29$

To facilitate data recording, ROS provides a tool for saving and restoring the running state of the simulation world, called "Rosbag". Using Rosbag guided by tutorial[4], we have successfully collected raw data from the published node. However, these raw data can not be put into the algorithm directly for training because there are some missing values and meaningless configuration data, also some follow-up data processing needs to be performed to calculate the total energy and relative velocity of the ball mentioned in section5.1.

For the data processing process, you can view my open source jupyter notebook format file in my Github repository[5]. It well worth mentioning that the total energy of the ball in Tab5.1 as the predicted label of our experiment is calculated following the equation5.1 and 5.2, where m represents the mass of the ball, v represents the relative speed between the ball and the bowl and h represents the relative height between the centre of the ball and the bottom of the bowl, KE and PE are kinetic and potential energy of the ball respectively.

$$KE = \frac{1}{2}mv^2 \tag{5.1}$$

$$PE = mgh \tag{5.2}$$

Finally, we processed training data we collected can be checked in my Github repository[6]. In addition to training data, verification data, test data will also be collected and more details will be described specifically in the following algorithm section.

[4] $http://wiki.ros.org/rosbag/Commandline$
[5] $https://github.com/Dzy-HW-XD/Waiter-Balance\_project/blob/master/dataProcess.ipynb$
[6] $https://github.com/Dzy-HW-XD/Waiter-Balance\_project/blob/master/processed\_data\_new.csv$

Chapter 6

State Predictor Development

This experimental section is inspired by the idea of time series prediction introduced by [23] and [33]. Considering the objectives of our project, we are aim to predict the total energy of the ball when the mobile robot moves for further balancing task, this requires forward-looking forecasts, which means we are concerned about the trend of the total energy of the ball in the future, not now. For example, the total energy of the ball could be predicted at the fourth time point according to the attributes of the first three time points. Following this thought, in this section we focus on building a recurrent neural network(RNN) baseline algorithm for this time series energy prediction task and making some reliable optimization on this baseline. Besides, in order to fully explore, some popular time series prediction algorithms will also be compared and evaluated in the evaluation chapter. One additional thing should be noted that in this section we develop target machine learning algorithm using open source deep learning framework proposed by [8], and also data processing in our project, such as data split, normalization is completed with the assistance of open source machine learning library scikit-learn[21].

6.1 Data Split and Normalization

Before developing machine learning algorithms, it is necessary to divide the data set into training set, validation set and test set. Training set helps us train the model and determines the parameters of the fitting curve, validation set is usually used for model optimization, while testing set is used to test the accuracy of the trained model and modify this trained model based on the test results. In our project, the training data set and validation data are obtained from the processed data in section5 following the

Data Type	Size
Tranining DataSet	10401
Validation DataSet	6934
Linear Testing DataSet	1168
Turning Testing DataSet	1136

Table 6.1: The Size of Training, Validation, Linear Testing and Turning Teseing Data Set

ratio of 6:4. However, due to the complexity of the movement of the mobile robot, we decide to collect two different test sets, one for only linear motion, another for turning motion, to test the performance of the model in two motion modes respectively. The process of collecting these two different testing guided by section5. The size of these data set is shown in table6.1. One thing should be noted that in order to make time series data trainable, we apply sliding window strategy for splitting these data set. More specifically, sliding window strategy determines the size of the dependence of the predicted total energy of the ball in future and the past. For example, if we set window size is 6, then the energy at a certain time point is related to the six time points before this.

$$\hat{x} = \frac{x - mean(x)}{sd(x)} \tag{6.1}$$

In our project, we initialize the window size to 4 in our baseline model and before the training process, for independent identically distributing and whitening data[26], we normalize data to following the equation 6.1, implemented by StandardScaler function in Sklearn[21], where $mean(x)$ represents the mean of the data and $sd(x)$ represents the standard deviation of the training samples.

6.2 Baseline

The baseline of our project is recurrent neural networks consisting of a hidden layer of 8 units followed by a dense layer with one unit output. Adam optimizer proposed by [15] is applied to minimized the mean square root loss function with the initial learning rate 0.1. Besides, some other super-parameters are as follows: in the initial stage, the input size of this neural networks is set to 18.The batch size is set to 25 and epoch is set to 200. The hyper-parameters listed above are not optimal, we just set reasonable values for follow-up optimization. Besides, dropout proposed by [20] as a solution of network over-fitting problem[6], we do not use it for the time being. Figure6.1 has il-

```
Model: "sequential_13"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
simple_rnn_15 (SimpleRNN)    (None, 8)                 208
_________________________________________________________________
dense_12 (Dense)             (None, 1)                 9
=================================================================
Total params: 217
Trainable params: 217
Non-trainable params: 0
```

Figure 6.1: The Structure of Baseline for Our Model

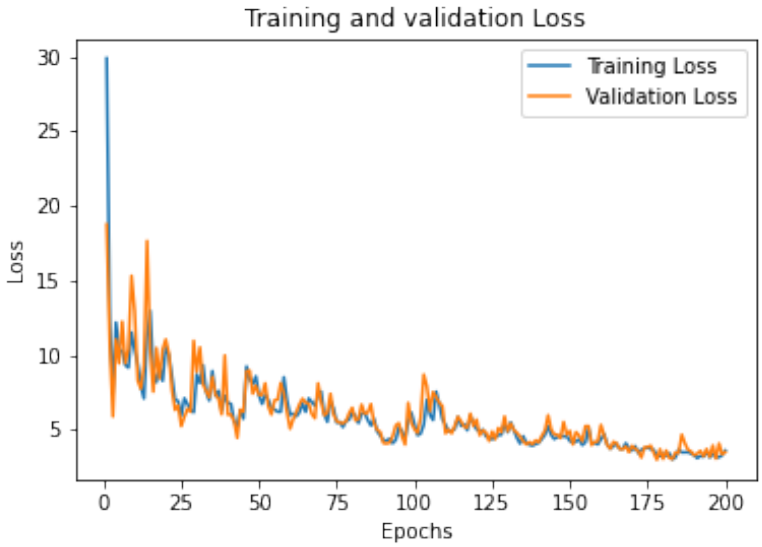

Figure 6.2: The Training and Validation Loss of the Baseline

lustrated the details of the structure of our baseline with 217 total trainable parameters.

6.3 Optimization

The resulted training and validation loss of the baseline is shown in figure6.2. As we can see in this picture, when the epoch reaches about 25, both training and validation loss decreases to a certain value and stops convergence. The poor performance of both training and validation loss illustrate that our baseline model is under fitted, which means the adaptability of our algorithm to fresh samples is significantly weak. In order to improve the generalization ability of the model, a deeper neural network needs to be constructed. Besides, the decrease of both of the training and validation loss curve is accompanied by oscillation, which means excessive learning rate always makes the

```
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
simple_rnn_7 (SimpleRNN)     (None, 4, 32)             1600
_________________________________________________________________
simple_rnn_8 (SimpleRNN)     (None, 8)                 328
_________________________________________________________________
dense_5 (Dense)              (None, 1)                 9
=================================================================
Total params: 1,937
Trainable params: 1,937
Non-trainable params: 0
_________________________________________________________________
```

Figure 6.3: The Optimized RNN Baseline Model

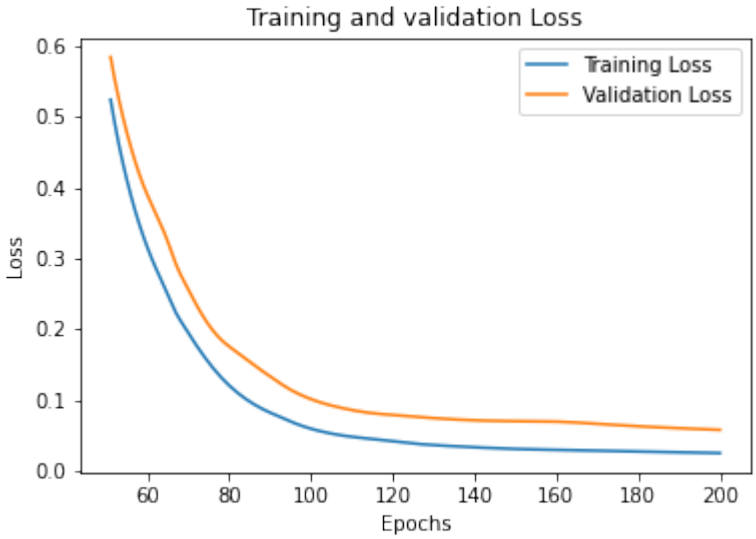

Figure 6.4: The Training and Validation Loss Curve After Optimization from Epoch 50 to 200

network skim over the optimum.

The structure of the optimized RNN model is shown in figure 6.3, as we can see in this figure, we modify the unit size of the input layer from 8 to 32 and add a extra hidden RNN hidden layer. As network complexity increases, the number of total trainable parameters also climbs to 1937 from 217. Besides, compared to the initial learning rate 0.1, we re-explored the learning rate and found that 0.001 is the possible optimum. The loss curve of optimized model is shown in figure 6.4, from this, the loss of both training and validation has been reduced to extremely small values, 0.0258 and 0.0585 respectively. In our project, we considered this improved model as the optimum and use it to test the training set.

Chapter 7

Evaluation

7.1 Baseline Results

There are two main parts in this chapter. Firstly, the predicted results will be evaluated and analyzed through the comparison with the true values of the testing dataset. Secondly, evaluation will be carried out between different models, following the same steps in section6, we have built two additional models(Simple RNN, Long Short-Term Memory(LSTM)[12] and Gated recurrent units(GRU)[10] in our project) which also can be considered recurrent neural networks with advanced structures. Comparing the performances of the among them, we possibly get the potential optimal model.

The Left Figure of 7.1 has illustrated the results of the optimized baseline based on the linear motion testing dataset, while the right one represents the results on turning motion. These results we obtained are generally in line with our expectations we discussed in previous section.

However, intuitively comparing the performance of the turning motion with the straight motion in figure7.1, our optimized model has better fitting ability for predicting the states of the ball on turning motion, because our optimized algorithm tends to assumes that the predicted energy should fluctuate although the energy is stationary in practice. The picture 7.1 on the left illustrates the problem that the total energy of the ball is predicted to about 23, while the true value is only 1. This error may be attributed to physics engine, because when we collected the training set we found that the state of the ball is more insensitive to the linear motion of the car compared to the turning motionwhich possibly makes our model difficult to fit the straight-line relationship features during the process of training.

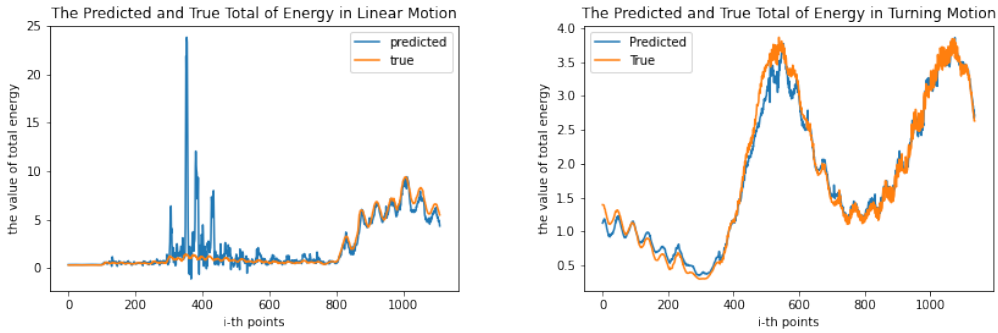

Figure 7.1: The Testing Results Predicted by Optimized RNN Baseline

7.2 Evaluation Criteria

There are five different evaluation criterias will be used in our project, mean square error, root mean square error, mean absolute error, mean absolute percentage error and R square. As for MSE, RMSE, MAPEE, the smaller the value, the better the performance of the algorithm. For R square, the closer the value is to 1, the better the performance, on the contrary, the closer to 0, the worse the performance. However, The value of R2 may be negative, which means the prediction accuracy of the algorithm is not even as good as directly calculating the average value. Besides, for equation 7.4, SS_{tot} represents the total sum of squares and SS_{res} represents the sum of squares of residuals.

$$MAE = (\frac{1}{n})\sum_{i=1}^{n}|y_i - x_i| \tag{7.1}$$

$$RMSE = \sqrt{(\frac{1}{n})\sum_{i=1}^{n}(y_i - x_i)^2} \tag{7.2}$$

$$MAPE = (\frac{100\%}{n})\sum_{i=1}^{n}|\frac{\hat{y}_i - y_i}{y_i}| \tag{7.3}$$

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}} \tag{7.4}$$

7.3 Comparison Evaluation

In our project, we are determined to choose LSTM and GRU as our extra algorithms for further evaluation because both of them can be used for sequential forecasting and describing the output of continuous state in time. Besides, as advanced structures of RNN, this section can be seen as an extension of the experiment to find a optimal

algorithm with by comparing the results among through the comparison of the three of them. The process of establishing the algorithm will not be repeated, but in order to make the comparison more convincing, several points should be noted and mentioned. The number of units in each layer should be guaranteed to be the same.

LSTM and GRU NNs are both trained and tested with the same training,testing set with RNN. The evaluation results are plotted in Tab 7.1. The performances of LSTM and GRU on predicting the energy of the ball in cart turning motion are all superior from the intuitive observation on Fig 7.1,7.2 and 7.3. The evaluation calculation of turning motion for all three do not differ much. The difference of MSE,RMSE and MAPE between LSTM and GRU are only 0.017, 0.0254 and 0.0245 respectively. Besides, the calculated R2 parameters of all three of them are greater than 0.98 which also shows the strong interpretability for dependent variables of our data set. However, when it comes to predict linear motion data set, although the evaluation results in linear motion of LSTM and GRU is outperform RNN one, there are still large errors(4 and 3 respectively) between the predicted results and the real values in the energy stationary stage shown in both Fig 7.2 and 7.3, which are Unacceptable for further exploration.

Apart from that, Tab 7.1 also shows that LSTM and GRU have stronger generalization ability to linear motion data than RNN. The MSE, RMSE and MAPE of RNN on linear motion data set is 0.3542, 0.2734, 0.3176 larger than LSTM's respectively which is the optimal one among three. And the R square of LSTM is twice as big as RNN's, which shows its superior of fitting ability on linear motion set. Furthermore, Compared these two different types of data, the generalization ability of the all three neural networks to turning motion data set is stronger than that of linear motion data set, because these evaluation metrics calculated on straight line motion are lower than those on curve motion without exception. LSTM has the strongest interpretability for linear motion data and GRU for turning motion data, which has been marked red in Tab 7.1.

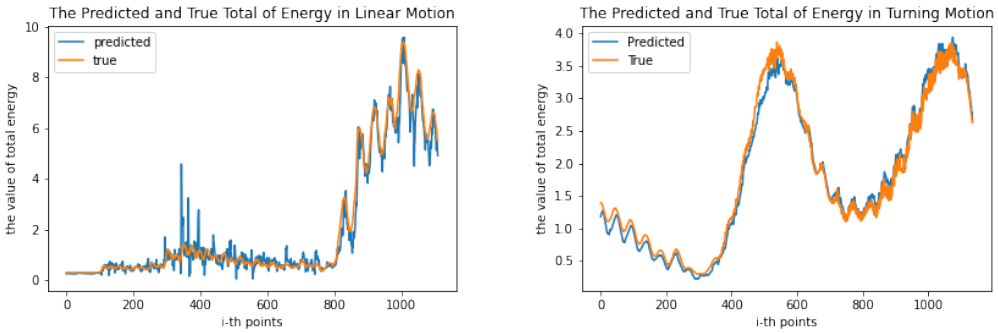

Figure 7.2: The Testing Results Predicted by LSTM

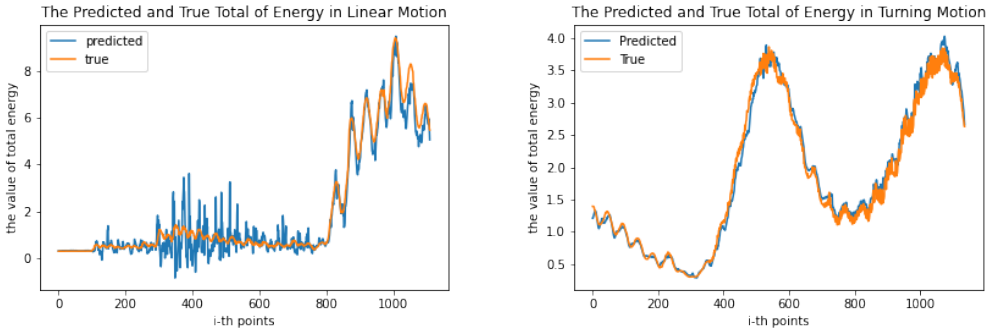

Figure 7.3: The Testing Results Predicted by GRU

Testing Results On Various Model	**MSE**	**RMSE**	**MAE**	**MAPE**	**R2**
RNN(Turning Motion)	0.1122	0.3350	0.1122	0.0824	0.9814
RNN(Linear Motion)	0.6157	0.7847	0.6157	0.5223	0.4855
LSTM(Turning Motion)	0.1206	0.3473	0.1206	0.0845	0.9805
LSTM(Linear Motion)	0.2615	0.5113	0.2615	0.2047	0.9697
GRU(Turning Motion)	0.1036	0.3219	0.1036	0.0600	0.9841
GRU(Linear Motion)	0.3282	0.5729	0.3282	0.2911	0.9589

Table 7.1: The Turning and Linear Testing Results on Various Model

Chapter 8

Conclusions

In this project, we have explored an optimal recurrent neural network(RNN) state predictor based on a novel bowl-ball model which has been established on gazebo simulator for the ball balancing task, and we also compared its performance with other advanced Long Short-Term Memory(LSTM) and Gated recurrent units(GRU) neural network predictors. Now we can conclude that LSTM and GRU NNs state predictor outperform the established RNN model in our experiment, and LSTM is more suitable for predicting the state of the ball when the mobile robot is moving linearly, while GRU is more reliable for predicting in curve motion.

However, there are still some remaining unsolved problems in our project. When predicting stationary energy for a period of time, our predictors tend to think that there will be large fluctuations, which could cause unacceptable errors in this period of time for further exploration. Besides, the models we build and compare may fall into local optima because we optimized our model based on the same neural structures.

To avoid the problems mentioned above, in the future, more data could be collected as training set to fit into our state predictor for better generalization of non-linear relationships during the stationary energy period of time. And to find the global optimum, some modifications on networks or data structures need to be implemented, such as increasing the depth of baseline and optimizing based on that or modifying the window size of the fitting data. In future work, some explorations on implementing a state predictor robot controller will be performed for further balancing task.

Bibliography

[1] Deep learning: Methods and applications, author=Deng, L. and Yu, D, year=Foundations and Trends in Signal Processing. 2014, 7: 3–4 [2015-10-23],.

[2] The impact of empirical accuracy st1udies on time sertes analysis and forecasting, author=Robert FILDES and Spyros MAKREDAKIS,.

[3] Oludare Abiodun, Aman Jantan, Oludare Omolara, Kemi Dada, Abubakar Umar, Okafor Linus, Humaira Arshad, Abdullahi Aminu Kazaure, Usman Gana, and Muhammad Kiru. Comprehensive review of artificial neural network applications to pattern recognition. *IEEE Access*, PP:1–1, 10 2019.

[4] O Boubaker. The inverted pendulum: A fundamental benchmark in control theory and robotics, 2012. 1–6. Web.

[5] George Box and Gwilym Jenkins. Time series analysis: forecasting and control, Oakland, California: Holden-Day.

[6] Rich Caruana, Steve Lawrence, and Lee Giles. Overfitting in neural nets: Backpropagation,conjugate gradient, and early stopping, Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium. Vol. 1. IEEE, 2000. 114–119 vol.1. Web.

[7] Chen, Hua, Patrick M Wensing, and Wei Zhang. Optimal control of a differentially flat two-dimensional spring-loaded inverted pendulum model, IEEE robotics and automation letters 5.2 (2020): 307–314. Web.

[8] Francois Chollet et al. Keras, 2015.

[9] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling, 2014.

[10] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling, 2014.

[11] M Hehn and R D'Andrea. A flying inverted pendulum, 2011 IEEE International Conference on Robotics and Automation. 763–770. Web.

[12] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[13] John Hopfield. Neural networks and physical systems with emergent collective computational abilities. *PNAS April 1, 1982 79 (8) 2554-2558*, 1982.

[14] Mahesh C. Jain. Textbook of engineering physics, 2009.

[15] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[16] Chris Lalancette, hane Loretz, Ioan Sucan, and Jackie Kay. Urdf-ros wiki. http://wiki.ros.org/urdf.

[17] Jos´e L Lima, Jos´e C Goncalves, Paulo G. Costa, and A. Paulo Moreira. Inverted pendulum virtual control laboratory.

[18] Adam Mills, Adrian Wills, and Brett Ninness. Nonlinear model predictive control of an inverted pendulum. In *2009 American Control Conference*, pages 2335–2340, 2009.

[19] Kyle E. Niemeyer and Chih-Jen Sung. Recent progress and challenges in exploiting graphics processors in computational fluid dynamics, The Journal of Supercomputing volume 67, pages528–564 (2014).

[20] Alex Krizhevsky Ilya Sutskever Ruslan Salakhutdinov Nitish Srivastava, Geoffrey Hinton. Dropout: a simple way to prevent neural networks from overfitting, 2017.

[21] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[22] Gábor Petneházi. Recurrent neural networks for time series forecasting. 1 Jan 2019.

[23] P. C. B. PHILLIPS. Time series regression with a unit root. Econometrica 55.2 (1987): 277–301. Web.

[24] Morgan Quigley, Brian Gerkey, Ken Conley, Josh Faust, Tully Foote, Jeremy Leibs, Eric Berger, Rob Wheeler, and Andrew Ng. Ros: an open-source robot operating system. In *ICRA Workshop on Open Source Software*.

[25] Sreelekshmy Selvin, R Vinayakumar, E. A Gopalakrishnan, Vijay Krishna Menon, and K. P. Soman. Stock price prediction using lstm, rnn and cnn-sliding window model. In *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pages 1643–1647, 2017.

[26] J. Sola and J. Sevilla. Importance of input data normalization for the application of neural networks to complex industrial problems. *IEEE Transactions on Nuclear Science*, 44(3):1464–1468, 1997.

[27] Toni Toharudin, Resa Pontoh, Rezzy Caraka, Solichatus Zahroh, Youngjo Lee, and Rung Chen. Employing long short-term memory and facebook prophet model in air temperature forecasting. *Communication in Statistics- Simulation and Computation*, 01 2021.

[28] Alper Tokgöz and Gözde Ünal. A rnn based time series approach for forecasting turkish electricity load. In *2018 26th Signal Processing and Communications Applications Conference (SIU)*, pages 1–4, 2018.

[29] Melonee Tully. Rurtlebot3- git. https://github.com/ROBOTIS-GIT/turtlebot3.git.

[30] Melonee Tully. Turtlebot e-manual. https://www.roscomponents.com/en/mobile-robots/215-turtlebot-3-waffle.html/courses-no.

[31] Ramiro Velázquez and Aimé Lay-Ekuakille. A review of models and structures for wheeled mobile robots: Four case studies. In *2011 15th International Conference on Advanced Robotics (ICAR)*, pages 524–529, 2011.

[32] Jia-Jun Wang. Simulation studies of inverted pendulum based on pid controllers. *Simulation Modelling Practice and Theory*, 19(1):440–449, 2011. Modeling and Performance Analysis of Networking and Collaborative Systems.

[33] A.S. Weigend. *Time Series Prediction: Forecasting The Future And Understanding The Past*. Taylor & Francis, 2018.

[34] P. Werbos. Beyond regression: new fools for prediction and analysis in the behavioral sciences. *PhD thesis Harvard University*, 1976.