Using Reinforcement Learning to Achieve Two Wheeled Self Balancing Control

Shih-Yu Chang[1], Ching-Lung Chang[2]

Department of Computer Science and Information Engineering
National Yunlin University of Science & Technology
[1]*M10317023@yuntech.edu.tw*
[2] *Chang@yuntech.edu.tw*

***Abstract*—The non-linear, unstable system of the two wheeled self-balancing robot has made it a popular research subject within the past decade. This paper outlines the design of a two wheeled robot with self balancing control systems using Reinforcement Learning. The BeagleBone Black platform was used to design the two wheeled robot. Along with the motor, the robot was also equipped with an accelerometer and gyroscope. Using the Q-Learning method, adjustments to the motor were made according to the dip angle and the angular velocity at that given time to return the robot to balance. The experimental results show that using this reinforcement learning method, the robot has the ability to quickly return to a balanced state under any dip angle.**

***Keywords*—Markov Decision Process; Reinforcement Learning; Q-Learning; Self-Balancing; Two Wheeled Robot**

I. INTRODUCTION

With the increase of environmental awareness, research into transportation methods that use alternative energy have been on the rise over the past few years. The Segway is one example: it greatly reduces noise and air pollution, while its high mobility allows it to access most public spaces. However, a two wheeled vehicle does not possess the same mechanical stability as a four wheeled vehicle. The special nature of the two wheeled design places it in a non-linear and unstable environment, where a self balancing control system is required to maintain a balanced state [9,10].

Current research into balance control methods can be divided into three categories: PID controller, Fuzzy Inference and Machine Learning. Juang et al.[2] proposed an improvement to the traditional PID controller, using the complementary filter approach with a PI-PD control. Adik S. Wardoyo et al. [3] combined fuzzy logic with PID control to reduce error and improve accuracy. Tsai et al. [4] decomposed the overall system into two subsystems and used radial-basis-function neural networks (RBFNNs) to achieve self-balancing, and Cui et al.[5] used the Support Vector Regression Method to map the correspondence between the robot's state and actions.

This paper uses the BeagleBone Black platform in the design of the two wheeled robot, accompanied with an accelerometer and gyroscope to detect the state of the robot. The self balancing control system was designed using the Q-learning reinforcement learning method, allowing the robot to return to a balanced state within the shortest timeframe when subjected to an external force.

II. TWO WHEELED SELF BALANCING ROBOT DESIGN

A. Robot Structure

The design of the two wheeled self balancing robot structure contains sensors for both the dip angle, and angular velocity. The wheels of the robot are attached by two DC motors. The computer controls all the parts mentioned above. Figure 1. , figure 2 show the complete robot structure.

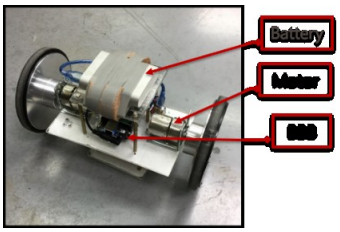

Figure 1. Robot Structure.

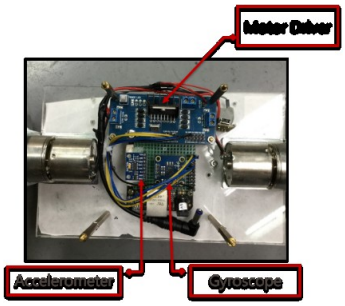

Figure 2. Robot Structure (2).

B. Signal Processing

The equipped accelerometer records the angle of the two wheeled robot, whereas the gyroscope records the angular velocity. However output signal often contains noise which leads to errors in calculation. Thus, a filter is used to eliminate the noise in order to obtain accurate data.

A complementary filter proposed by Colton [8] was used in this paper. The complementary filter uses a combination of a low-pass filter, numeric integration, high-pass filter to help

978-1-5090-3438-3/16 $31.00 © 2016 IEEE
DOI 10.1109/ICS.2016.28

eliminate the noise created by the accelerometer under motion and the accumulated discrepancy of the gyroscope. The complementary filter is calculated using the following formula:

$$angle_{t+1} = \varphi \times (angle_t + gyro \cdot dt) + (1 - \varphi) \times acc\_angle_{t+1} \quad (1)$$

Where $angle_{t+1}$ is the current angle; $angle_t$ is the previous angle; gyro is the angular velocity detected by the gyroscope, where the angle can be calculated using numeric integration. Since the gyroscope is more accurate in calculating the angular velocity of objects in motion, it is given a higher parameter value φ; acc_angle_{t+1} represents the angle obtained by the accelerometer.

III. SELF BALANCING CONTROL SYSTEM

A. Q-Learning

Reinforcement learning is a type of machine learning that determines the action within a specific environment in order to maximize a reward. One of the characteristics of reinforcement learning is that the agent can only receive a reward after performing the action, and thus must continue to interact with the environment in order to determine the optimal policy through trial and error.

Figure 3. shows the agent-environment interaction in reinforcement learning. The "agent" and the "environment" are the two main elements within reinforcement learning. The learner that is responsible for decision making is defined as the "agent", whereas everything it interacts with is referred to as the "environment". The learning process involves the environment presenting the current situation or state to the agent, to which the agent will select an suitable action. The environment will then respond with the new state and a reward value based on the action taken.

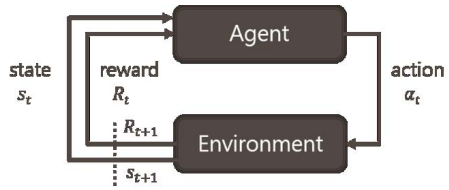

Figure 3. Reinforcement Learning Structure

The goal of the agent is to find a best policy that receives the maximum reward in the long run. This policy is denoted as π, meaning the probablity of taking action a when in a state s. The Q-Learning method groups the state with a corresponding action into a pair, where each pair is given an action value. This action value is then used to determine the long term effectiveness of the policy. The updating method uses the maximum of the following action value to update the previous value function using the following formula:

$$Q(s_t, a_t) = (1 - \alpha)Q(s_t, a_t) + \alpha[R_t + \gamma\, maxQ(s_{t+1}, a_{t+1}')] \quad (2)$$

Where $Q(s_t, a_t)$ represents the state and action pair at time t. After entering state s_{t+1} from s_t, a reward R_t is given and the corresponding action a_{t+1} is performed according to policy π, γ denotes the discount factor which takes into account the foresight of the long term goal.

Since the Q-Learning updating method uses the maximum action value, this method significantly reduces the time to achieve convergence. Therefore the Q-Learning method is selected for this experiment.

B. Learning System Definition

The two main elements of Reinforcement Learning are the "Agent" and the "Environment". In this paper, the "Environment" is defined as the space the two wheeled robot is in, whereas the "Agent" is defined as the robot itself. The Agent reads the current environment "State" and chooses an "Action" according to its policy.

The state is expressed as state: $\{\theta_{dip}, \dot{\theta_{dip}}\}$, where θ_{dip} is the dip angle and $\dot{\theta_{dip}}$ is the angular velocity. This experiment also uses the values D_{adc}, D_{gyro} as custom units of measurement for the angle and angular velocity, in order to decrease state complexity and memory usage. The balanced state is defined as target state: $\{\theta_{Tdip}, \dot{\theta_{Tdip}}\}$. This is the original state before the learning process, where the optimal θ_{Tdip} is 0° and the optimal $\dot{\theta_{Tdip}}$ is 0(degree/sec). Reward is given based on the change in angle difference $e(t)$ compared to the old angle difference $old\_e(t)$. Actions are defined as the clockwise or counterclockwise motor rotation under different PWM duty cycles.

The action value functions are updated using a combination of n-step TD prediction and Backward view of TD [1]. This increases the connection between different states and thus shortens the time required for convergence. Where state S_2 is entered at time t+2 and used to update the value function defined as $Q(t+1)$. Once the update is complete, the updated version of $Q(t+1)$ is then used to update $Q(t)$to speed up convergence.

IV. EXPERIMENTATION

In this paper, one complete training is defined as the transition from a balanced state into an unbalanced state and returning back to the balanced state. The learning appears to stabilize as the amount of training given increases, however, overtraining issues begin to arise after a certain amount. The sampling frequency was fine-tuned to correspond with the motor reaction time. A high sampling frequency will increase the state accuracy, but might not be enough time for the motor to react. A frequency of 20Hz was set, however considering system calculation delays, the actual sampling frequency was closer to 16Hz. The Q-Learning learn rate α and $\varepsilon\text{-}greedy$ rate was originally set at 100% and 5%, but gradually decreases throughout, The learning process under a discount factor γ of 0.95. The units of measurement D_{adc} and D_{gyro} were set at 3° and 100dps to decrease state complexity and

memory usage, the memory usage calculation using the following formula:

$$\left[\frac{180}{D_{adc}} \times 2 + 1\right] \times \left[\frac{500}{D_{gyro}} \times 2 + 1\right] \qquad (3)$$

When setting the units of measurement D_{adc} and D_{gyro} as 1, there is a total of 361,361 different states. Paired with 21 action controls, data saving as float type, uses a total memory of 30,354,324 byte. The amount of dip angles after partitioning under D_{adc} reduces to 121, and the amount of angular velocities under D_{gyro} to 11, giving a total of 1,331 different states. Paired with the same 21 action controls, using only 111,804 Byte when data saved as float type. This saves roughly 270 times more memory space. A higher value would have resulted in too few available states, thus decreasing the accuracy and transition smoothness. A total of 21 action controls were defined using a combination of clockwise and counterclockwise motor rotations under different PWM duty cycles in 5% intervals.

The rewards given are listed below in table 1. When the current state matches the target state, the maximum reward is given. If $e(t) \geq old\_e(t)$, a negative reward is given, a further punishment is given if a zone change is involved.

Table 1. Reward Numerical

State Evaluate	Zone Vary	Reward
$state = Targetstate$	none	100
$e(t) \geq old\_e(t)$	Same	-5
	Different	-10
$e(t) < old\_e(t)$	Same	+1
	Different	+1

The experimental results show the balancing abilities of the robot within five different zones based on the angular velocity unit D_{gyro}. After comparing the effects of different amount of training, three different stages of training are used to illustrate the effect of amount of training on the balancing capability. The three stages are 5000 times, 15000 times, 20000 times, representing the three stages of under-trained, well-trained, and over-trained.

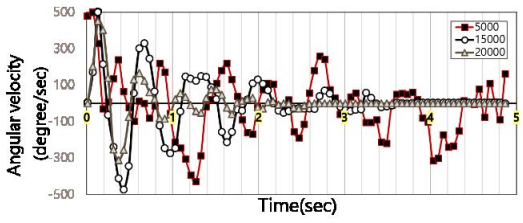

Figure 4. Zone V Result of Balancing Required Time Compare.

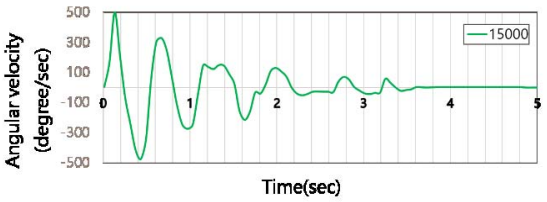

Figure 5. Zone V Result of Angular Velocity.

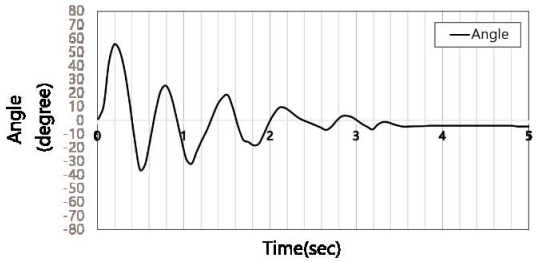

Figure 6. Zone V Result of Angle.

Among the largest of all regions: Zone V, represents an angular velocity of 400~499(degree/sec). The comparison between different amount of training is shown in figure 4. Similarly, 15000 times appears to be the optimal amount. The robot was unable to return to a balanced state with only 5000 times of training, required 2.6s for 15000 times and 2.8s for 20000 times. The changes in angular velocity and angle during the balancing process are shown in figure 5. and figure 6. The angle fluctuation is within 60°.

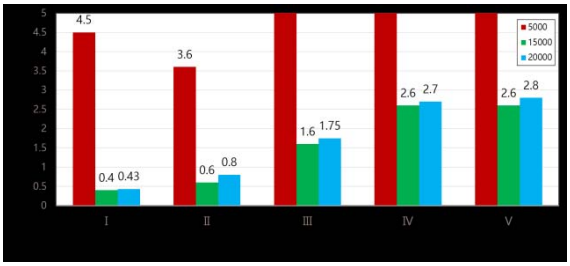

Figure 7. Result of Angular Velocity Required Time Compare.

The time taken to achieve a balanced state within the five different zones are shown in figure 7. , this shows 15000 times to be the optimal training amount. When increased to 20000 times, the time taken to achieve a balanced state did not decrease, proving signs of over-training.

Figure 8 and figure 9. show the state-action reciprocal diagram after 15000 times of training. The diagram is not completely smooth near the origin point due to a lack of simulation points.

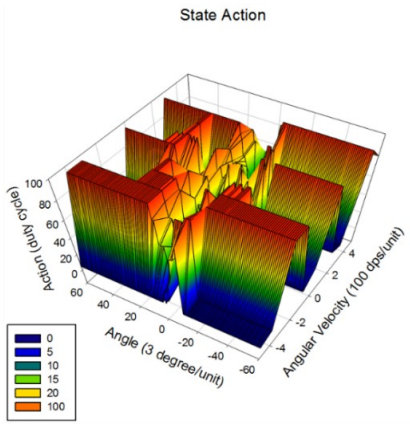

Figure 8. State-Action Reciprocal Diagram.

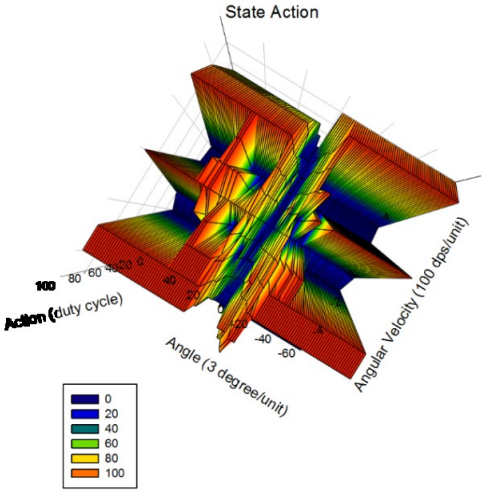

Figure 9. State-Action Reciprocal Diagram (2).

V. Conclusions

This paper uses reinforcement learning to achieve two wheeled self balancing control. The Q-Learning method was selected, using dip angle and angular velocity as state definitions and the clockwise/counterclockwise rotations combined with different PWM duty cycles as action controls. Custom units of angle and angular velocity were used to reduce state complexity and memory usage. N-step TD prediction and Backward view of TD was used to update the value functions in order to increase speed of convergence. Three different training amount was used to represent the three stages of training. Results show the effectiveness of the Q-Learning method and also the effect of training amount on the convergence speed.

References

[1] Richard S.Sutton, Andrew G.Barto., "Reinforcement Learning: An Introduction," MIT Press, Cambridge, MA, Sep 1998, 340 pp.

[2] Hau-Shiue Juang, and Kai-Yew Lum, "Design and Control of a Two-Wheel Self-Balancing Robot using the Arduino Microcontroller Board," *10th IEEE International Conference on Control and Automation (ICCA)*, June 2013, pp. 634-639.

[3] Adik S. Wardoyo, Hendi S, Darwin Sebayang, Imam Hidayat, and Andi Adriansyah, "An Investigation on the Application of Fuzzy and PID Algorithm in the Two Wheeled Robot with Self Balancing System Using Microcontroller," *Control, Automation and Robotics (ICCAR)*, 2015 International Conference on, May 2015, pp. 64-68.

[4] Ching-Chih Tsai, Hsu-Chih Huang, Shui-Chun Lin, "Adaptive Neural Network Control of a Self-Balancing Two-Wheeled Scooter," *IEEE Transactions on Industrial Electronics*, April 2010, Volume 57, Issue 4, pp. 1420-1428.

[5] Liangliang Cui, Yongsheng Ou, Junbo Xin, Dawei Dai, Xiang Gao, "Control of a Two-Wheeled Self-Balancing Robot with Support Vector Regression Method," *4th IEEE International Conference on Information Science and Technology*, April 2014, pp. 368-372.

[6] Mehryar Mohri, Afshin Rostamizadeh, Ameet Talwalkar, "Foundations of Machine Learning," The MIT Press ISBN 9780262018258, 2012

[7] Geoffrey Hinton, Terrence Joseph Sejnowski, "Unsupervised Learning and Map Formation: Foundations of Neural Computation," MIT Press, ISBN 0-262-58168-X, 1999.

[8] Shane Colton, "The Balance Filter," Massachusetts Institute of Technology, Tech. Rep., 2007

[9] Xiaogang Ruan, Jianxian Cai, Jing Chen, "Learning to Control Two-Wheeled Self-Balancing Robot Using Reinforcement Learning Rules and Fuzzy Neural Networks," *Fourth International Conference on Natural Computation*, 2008, Volume 4, pp. 395-398.

[10] Bin-Bin Li, Jin-Hui Zhu, Hua-Qing Min, "BMP: A self-balancing mobile platform," *International Conference on Machine Learning and Cybernetics*, 2012, Volume 3, pp. 868-874.