GOAL-SEEKING NAVIGATION BASED ON MULTI-AGENT REINFORCEMENT LEARNING APPROACH

ABDUL MUIZZ BIN ABDUL JALIL

UNIVERSITI TEKNOLOGI MALAYSIA

GOAL-SEEKING NAVIGATION BASED ON MULTI-AGENT REINFORCEMENT LEARNING APPROACH

ABDUL MUIZZ BIN ABDUL JALIL

A project report submitted in fulfilment of the
requirements for the award of the degree of
Master of Engineering (Mechatronic and Automatic Control)

School of Electrical Engineering
Faculty of Engineering
Universiti Teknologi Malaysia

JULY 2022

DEDICATION

This thesis is dedicated to my parent, who has been support me financially and morally throughout of this thesis. Despite all the obstacle encounter, this thesis was able to finished and without them my studies would not be finished.

ACKNOWLEDGEMENT

During the project is carry out, I was able to can acquaintances with many people such as researchers and academicians. They have contributed to my understanding and thoughts. In particular, I wish to express my sincere appreciation to my thesis supervisor, Professor Dr Mohd Ridzuan Bin Ahmad, for his encouragement, guidance, and critics. I am also indebted to my other supervisor, Carlos Mateo from the ImVIA laboratory at the University Institute of Technology, Le Creusot, France, who has taught me many things about deep reinforcement learning and has shaped my deep understanding of this topic.

I was able to finish the projects due to the resources used at the ImVIA laboratory. The project was able to complete by using the graphic card NVIDIA P400 2GB as well as a computer workspace.

I also want to thank my fellow postgraduate student who has been support me during my presence in the laboratory. I also extend my appreciation to my colleagues for extending their help and other occasions' support. I also want to appreciate all my family members as they also play their support.

ABSTRACT

Mobile robotics has been applied in many fields of industry and has been an impact on many industries. Most modern industries depend on mobile robots ranging from indoor to outdoor applications such as robot vacuum to robot delivery. The most important aspect of mobile robots is the navigation algorithms that allow the robot to move through certain terrain to reach the desired state. Many of the algorithms that contributed the most are the SLAM (simultaneous localisation and mapping) and the path planning algorithm. SLAM is mostly used to estimate the feedback states such as localization and perspective map, whereas path planning is mainly a planner from state estimation. The RL (reinforcement learning) researcher has been studying the RL in robotics navigation focusing on areas such as motion planning, and perception estimation. There have been breakthroughs in the decades and these have been closing the gap between RL and control systems since they are the same but initially develop from different areas and directions but recently, much of it is converged to develop in the same field. Although the approached problems are what makes it different, it is still the same problems. This study explores implementing the multi-agent in DRL (deep RL), specifically to train a single policy in a multi-agent environment since robotics simulation can run many models and therefore, it is unnecessary to run many simulators to train the policy in a batch. Since RL is semi-supervised learning through reward signal, similar to the cost function in a control system where the policy will try to maximize the return of the expected reward of trajectory. The scope of this study mainly covers indoor navigation and motion planning. The toolkits to perform the study are stable-baselines3, Gazebo simulator, and OpenAI Gym. The robot used in the simulation for this study is the Turtlebot3 burger since it does not require a stability controller and it has the least number of velocity commands. The Turtlebot3 burger sensors are odometry, IMU (inertial measurement unit), and a laser that serves as feedback observation states. The UKF (unscented Kalman filter) is used as state estimation and to utilize any feedback states. Although the desired states are not a part of the observation states, this is not the case for DRL. Given the nature of the robotics simulation, is possible to run a single simulator but still be able to train the policy in batches. Since there is no RL environment specifically to conduct this study, the multiagent environment was implemented to meet the study objectives. The policy network was constructed using LSTM (long short-term memory) and MLP (multilayer perceptron) served as feature extraction and decoder. The integration between the DRL algorithms with ROS (Robot Operating System) is able to train and communicate between these various connections but was not able to achieve similar results as position and obstacle avoidance because of numerous reasons. Due ROS being peer-to-peer system is possible to use other DRL library such as RLlib (Industry-Grade RL) with ROS as future work as RLlib is support for distribution training for DRL.

ABSTRAK

Robotik boleh gerak telah digunakan dalam banyak bidang industri dan telah memberi kesan kepada banyak industri. Kebanyakan industri moden bergantung kepada robot boleh gerak yang terdiri daripada aplikasi dalam bangunan seperti robot vakum kepada robot penghantaran. Aspek terpenting robot boleh gerak ialah algoritma navigasi yang membolehkan robot bergerak melalui kawasan tertentu untuk sampai ke keadaan impian. Kebanyakan algoritma yang banyak menyumbang ialah SLAM (penyetempatan and peta serentak) dan algoritma perancangan laluan. SLAM kegunaannya lebih kepada untuk menganggar keadaan suap balik seperti lokalisasi dan peta perspektif, di mana perancangan laluan adalah terutamanya sebagai perancang daripada anggaran negeri. Penyelidik RL (pembelajaran pengukuhan) telah mengkaji RL dalam navigasi robotik yang memfokuskan dalam bidang seperti perancangan gerak, anggaran persepsi dan terdapat kejayaan dalam beberapa dekad dan ini telah merapat jurang antara RL dan sistem kawalan kerana ia adalah sama tetapi pada mulanya berkembang dari kawasan dan hala tuju berbeza tetapi baru-baru ini, kebanyakannya disatukan untuk membangun dalam bidang yang sama. Kajian ini meneroka pelaksanaan pelbagai ejen dalam DRL (pembelajaran pengukuhan mendalam), khususnya untuk melatih satu dasar dalam persekitaran berbilang ejen. Memandangkan simulasi robotik boleh menjalankan banyak model dan oleh itu, adalah tidak perlu menjalankan banyak simulator untuk melatih dasar dalam satu kelompok oleh kerana RL ialah pembelajaran berpenyelia separa melalui isyarat ganjaran, serupa dengan fungsi kos dalam sistem kawalan di mana polisi akan cuba memaksimumkan pulangan ganjaran trajektori yang dijangkakan. Skop kajian ini terutamanya meliputi navigasi dalaman dan perancangan gerakan. Set alat digunakan melaksanakan kajian adalah stable-baselines3, simulator Gazebo dan OpenAI Gym. Robot yang digunakan dalam simulasi untuk kajian ini ialah Turtlebot3 burger kerana tidak memerlukan pengawal kestabilan dan mempunyai bilangan arahan halaju yang paling sedikit. Penderia burger Turtlebot3 ialah odometri, IMU (unit ukuran inersia) dan laser yang berfungsi sebagai keadaan pemerhatian suap balik. UKF (turas Kalman tanpa wangian) digunakan sebagai anggaran keadaan dan untuk menggunakan mana-mana keadaan maklum balas. Walaupun keadaan yang dikehendaki bukan sebahagian daripada keadaan pemerhatian, ini tidak berlaku untuk DRL. Memandangkan sifat simulasi robotik, adalah mungkin untuk menjalankan satu simulator tetapi masih dapat melatih dasar dalam kelompok. Memandangkan tiada persekitaran RL khusus untuk menjalankan kajian ini, persekitaran multiagen telah dilaksanakan untuk memenuhi objektif kajian. Rangkaian polisi telah dibina menggunakan LSTM (ingatan jangka pendek dan panjang) dan MLP (perseptron berbilang lapisan) berfungsi sebagai penyarian sifat dan penyahkod. Penyepaduan antara algoritma DRL dengan ROS (sistem pengendalian robot) dapat melatih dan berkomunikasi antara pelbagai sambungan ini tetapi tidak dapat mencapai keputusan yang sama seperti pengelakan kedudukan dan halangan kerana pelbagai sebab. Oleh sebab ROS ini sistem rangkaian rakan ke rakan adalah kemungkinan untuk menggunakan rujukan DRL lain seperti RLlib (gred-industri RL) dengan ROS sebagai kerja masa hadapan kerana RLlib ada sokongan untuk latihan pengagihan untuk DRL.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

3D		Three-Dimensional
A2C	-	Advantage Actor Critic
ANN	-	Artificial Neural Network
API	-	Application Programming Interface
AUV	-	Autonomous Underwater Vehicle
DDPG	-	Deep Deterministic Policy Gradient
DRL	-	Deep Reinforcement Learning
IMU	-	Inertial measurement unit
LMR	-	Legged Mobile Robot
LSTM	-	Long Short-Term Memory
MDP	-	Markov Decision Process
MLP	-	Multi-layer Perceptron
PPO	-	Proximal Policy Optimization
RL	-	Reinforcement Learning
ROS	-	Robot Operating System
SAC	-	Soft Actor-Critic
SLAM	-	Simultaneous Localization and Mapping
TD3	-	Twin Delayed DDPG
UAV	-	Unmanned Aerial Vehicle
UB	-	Université de Bourgogne
UKF	-	Unscented Kalman Filter
URDF	-	Unified Robotics Description Format
UTM	-	Universiti Teknologi Malaysia
WMR	-	Wheeled Mobile Robot
ZOH	-	Zero-order Hold

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

1.1 Background

The study of robotics has been around since 1961 after the concept of robotics was introduced in 1940 by Isaac Asimov in his works. Since then, many problems related to robotics have been studied. Mobile robots have numerous applications in industries from agriculture, automotive, robot vacuum, all the way to courier shipping management, and some other examples are shown in Figure 1.1, like a vacuum aspirator robot from Roomba (a), a warehouse robot used in Amazon warehouse (b) and robot delivery by Amazon Prime (c) to name a few of them. Mobile robots have many forms and can be classified as wheeled mobile robot, legged mobile robot, and drone [1]. What makes mobile robot's different from other robots it's the mobility to move through the terrain on specific applications, as mentioned before. The navigation model of perception can be divided into geometric, topological, semantical, and perception models navigation. Other relevant aspects are the types of environments where robots are used, such as outdoor, semi-outdoor, indoor, and semi-indoor [2]. The robots have actuators such as motors attached to their joints, and they are what give the ability to control through electricity. Typically, the control system can be an open-loop or closed-loop control system where the measuring from the sensor is feedback to a controller and the controller will compute the cost function to predict the motion into motors in this case. The controller for motions is usually a variation of PID controller, Full-State-Feedback, or the others.

(a) (b) (c)

Figure 1.1 Robotics application. (a) Robot vacuum. (b) Warehouse robot. (c) Delivery robot

Many navigation problems in mobile robotics have been researched. Currently, the navigation problems involved mapping, path planning, localization, and motion control. To produce perception mapping by scanning the surroundings to produce an occupancy map represented by binary or probabilistic values for robot understanding of obstacles in environment space [3]. But mapping requires localization of the robot which is to determine the position of the robot in a known map which obtains by feature extraction using vision or time of flight sensors for scanning the environment. By determining robot localization, mapping the unknown environment is possible then a path to the goal position can be realized. Currently, the most common approach for solving this problem is the simultaneous localization and mapping (SLAM) techniques which intend to do all related problems in multi-tasking such as mapping the environment while measuring the robot's location in the environment. Because of drawbacks in measuring the states, often SLAM is deployed with filter methods such as the Kalman filter to estimate the probabilistic localization as well as estimate the obstacles states [4]. Map on the other hands, can represent something like occupancy grid in Figure 1.2.

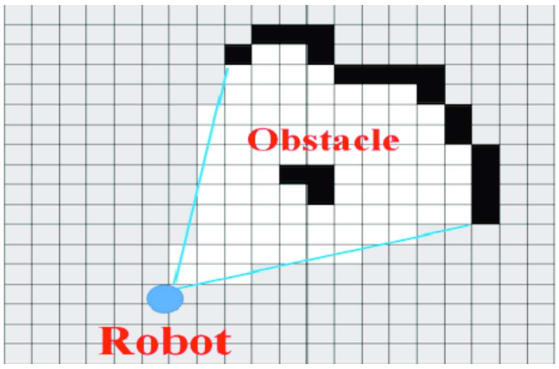

Figure 1.2 Occupancy grid mapping and localization

The DRL, incorporates a deep learning approach in the RL framework (Figure 1.3) [5][6], and it differs from DL in how the data is collected. While in supervised learning, the data consists of a set of predicted and ground truth collection of samples, in DRL there is no ground truth instead the data is collected dynamically from the environment such as the surroundings and learned from the reward signal compute from the environment itself during training. Thus, DRL can be regarded as semi-supervised where humans design the shape of the reward to be learned to maximize or minimize the behaviour, depending on whether it is a positive or negative reward. Based on the reward, the gradient descent can be achieved, to update the weight parameters that lead to the optimal policy and if the policy is a controller thus the terms control policy is called [7]. The notations differ between RL and control system theory, and they usually refer to the same thing because it is developed in different fields such as computer science and control engineering field. Because the way the policy itself is deep neural networks means that it cannot be analysed in the same aspect as in control systems such as stability, time transient analysis, etc.

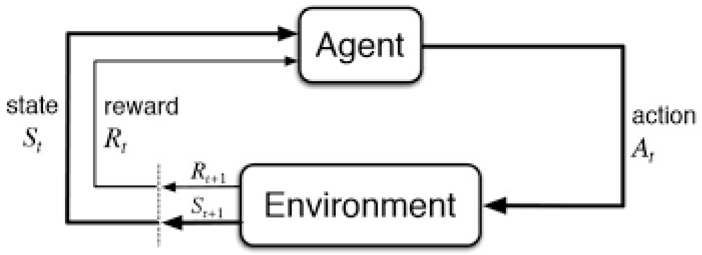

Figure 1.3 The basic component of RL

1.2 Problem Statement

Often, most of the controllers develop using a control system approach usually required to analyse the dynamics of the robot mechanism made of the chassis and the actuator used as a transducer (conversion electrical to mechanical energy). The mathematical of this dynamic system can sometimes is hard to estimate and can consume time to obtain near accurate modelling of the dynamic system as illustrate in Figure 1.4. Although most of this process can be simplify using system identification. But this approach needs very good data collection that described all the necessary behaviour. The DRL model-free is a good solution that does not need the robot dynamic system and can control by knowing the pattern obtain from observation states. What kinds of data need it for the neural network to learn needs to be considered.

The robots, specifically the mobile robots, have many forms. For grounded applications, wheeled robots are the most common. The only needed problem to be considered is the kinematics of the robot since four-driven-wheeled robots are completely different from two-driven-wheeled robots. Another thing that needs to be aware of is the types of wheels used, such as the omnidirectional wheel (mecanum wheel). The kinematics of two driven wheels either came with a castor wheel or not. Although two driven wheels with a castor are most suitable since are kinematically stable two driven wheels without a castor can be stable by applying the control systems.

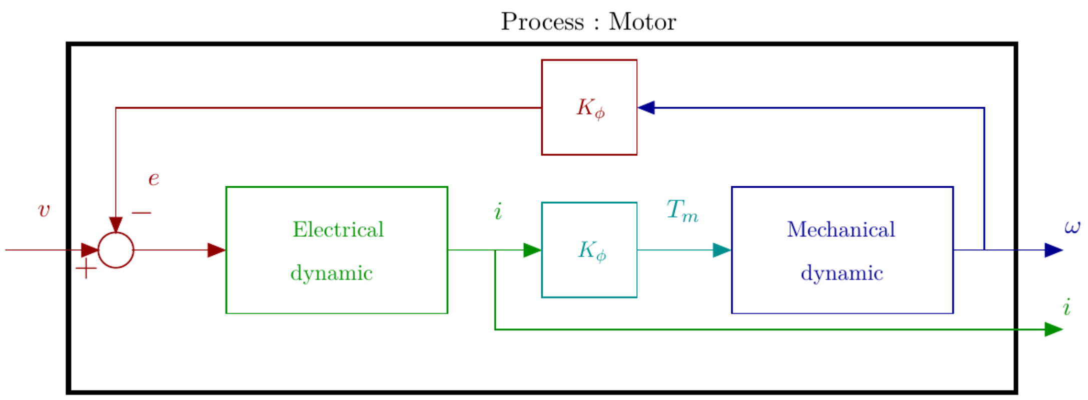

Figure 1.4 Electrical motor modelling

Controlling the mobile robot by knowing the kinematics relationship between actuator state vector and state vector that can be easily understood such as translational or rotational motion, or both. Although simulation of motor behaviour in robotics simulation often is unaccountable because there is an issue of dynamical modelling of actuation parts, for example, is a misrepresentation of them, which are necessary when we want to transfer learned control policies from simulated to real worlds. Some robots have embedded with low-level movement and one of them is the Turtlebot3 [8]. This robot also comes with simulation as well which is needed for training in the simulation since it is much safer and easy to configure. This robot used ROS to communicate all different components such as the sensor and actuator. Changing between simulation and the real robot can easily change and this option will leave as an option depending on the time constraints to carry out this project. The Turtlebot3 has 2 types, which are burger and waffle. The burger version is lightweight and easy to control since it required only linear and angular velocity for moving forward and backwards, as well as rotating left or right at the centre of the wheelbase, *b*. The equation (1.1) below shows the relationship between robot velocity with the velocity of the wheel. Transforming from actuator vector to robot local frame vector. The velocity can be obtained by reading the wheel velocity but this is taken care of by Turtlebot3 embedded controller OpenCR board. The equation (1.2) is to convert from local coordinates to global coordinates. If the direction of the robot heading is not important, the orientation between the global frame and local frame can be discarded. But for Turtlebot3, there is a heading, and therefore is better to have this information for neural network policy to learn such as the goal heading. All the information in equations (1.1), (1.2) is sent to the ROS odometry message such as pose (linear and quaternion), and twist (linear and angular)

$$\begin{bmatrix} v_t \\ \omega_t \end{bmatrix} = \begin{bmatrix} 1/2 & 1/2 \\ 1/b & -1/b \end{bmatrix} \begin{bmatrix} v_R \\ v_L \end{bmatrix} \tag{1.1}$$

$$\begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos(\theta) & 0 \\ \sin(\theta) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v_t \\ \omega_t \end{bmatrix} \tag{1.2}$$

Considering the reality and practical problems when it comes to measuring the states may lead to overestimating. The reason for this is because the noise will always present that may influence by the environment such as electromagnetic noise, from the electric motor from the robot joint and etcetera. This presence of noise can be removed by filtering by using a lowpass filter for removing high-frequency noise and a highpass filter for removing low-frequency noise. The IMU is known for its high sensitivity (prone to high-frequency noise such as large oscillation) and the gyroscope for low sensitivity (prone to low frequency such as drifting). Both IMU and gyroscope can estimate the states of orientation and this is where sensor fusion came in.

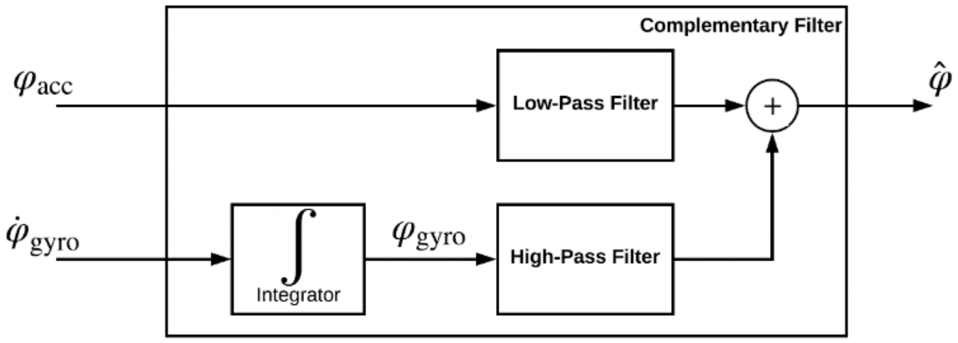

Figure 1.5 Complementary filter

The sensor fusion like a complementary filter in Figure 1.5 can compensate for the weakness of both of these sensors but the filter based on signal processing has some drawbacks due to fixed frequency filtering as well as the presence of noise is not completely removed. The states can be estimates such as the Kalman filter [9]. The Kalman filter takes the consideration of noise by estimating the noise using probability such as the gaussian distribution function. Both of these sensors are commonly used in robots and this includes the Turtlebot3.

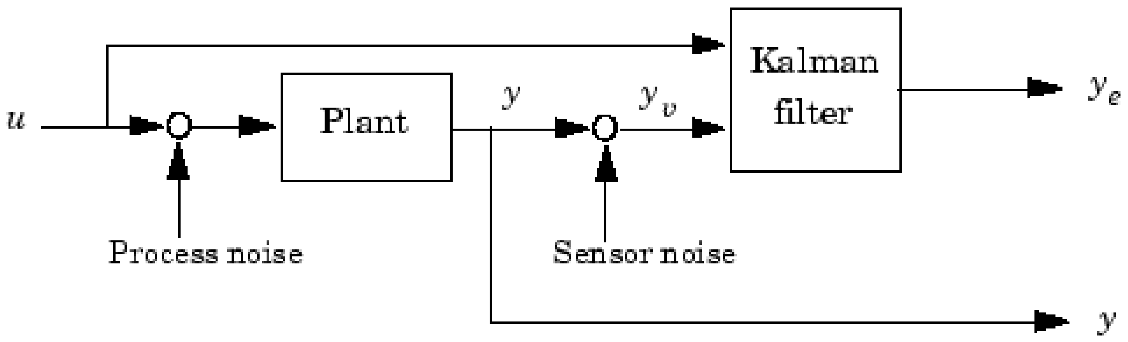

Figure 1.6 Kalman filter

The Figure 1.6 shows that the Kalman filter used the output system and the input system to find the optimal states. The Kalman filter is required to know the internal states of the system for better estimation. There are also variations of Kalman filters like UKF and EKF for example. They work very similarly in a way it can be discretized-time and recursive. The Figure 1.7 generalizes the states that are trying to estimate.

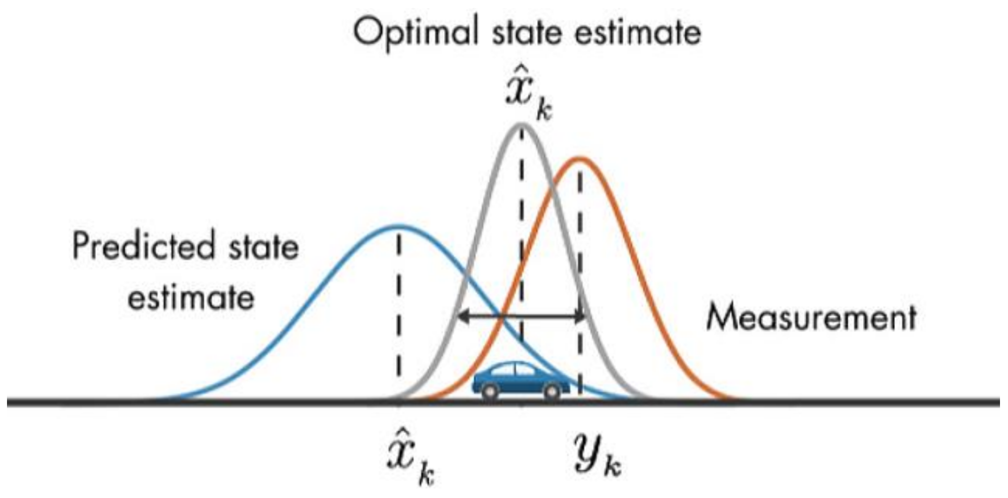

Figure 1.7 State estimation

Since the neural network is used as a controller, how it is structured is very important since it is mapping from observation states to action states. For complex problems, a large model such as multimodal normally would be considered but if the states from the environment such as Lidar for detecting obstacle as odometry and IMU for localization usually a simple that does not require a large model. The model usually processes such as feature extraction known as encoder and from feature space to actions is the decoder. The decoder is usually MLP since it can be solved in linear regression, hence why it is effective as the decoder. The Figure 1.8 is the autoencoder with main used as to remove noise such as salt and pepper noise but generally most neural network will have similar block structure. The encoder is usually depending on the input size and shape such as if the input is a signal such as a discrete-time signal, image, point-cloud, etcetera.

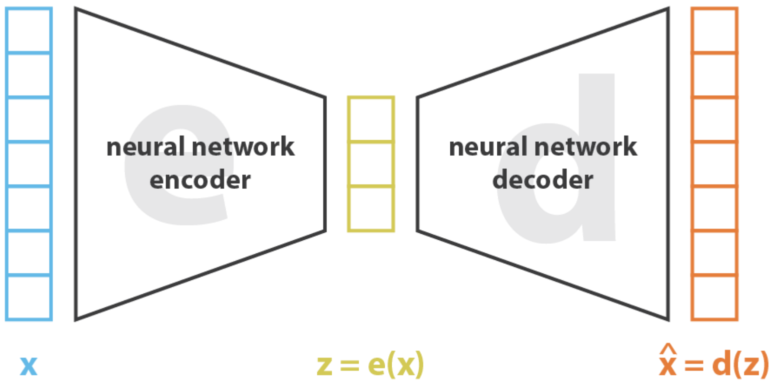

Figure 1.8 Basic structure of the neural network

Considering the inputs such as camera, lidar, or both can determine the complexity of neural network such as image coming from the camera has large feature needs to be reduced but of signal processing such as Lidar can be simple model although it may need to be filtered since the presence of noise will contain in the measurement. The map of the environment such as the occupancy grid sometimes may not be included since is normally not required for a controller such as a reactive controller. Using LIDAR and others means of obtaining localization is sufficient to serve as a vision for the robot. The equations (1.3) to (1.5) shows the relationship between the states and controller output.

$$v_{k+1} = f\left(s_k\right) \tag{1.3}$$

$$x_{feature} = f\left(s_k, v_k\right) \tag{1.4}$$

$$v_{k+1} = f\left(x_{feature}\right) \tag{1.5}$$

The last that needs to be considered is DRL algorithms since is normally depending on the shape of observations and actions. The observation states here can be from sensor raw measurement or after applying to filter for noise reduction. For robotics control, the actions in DRL are continuous actions because it is a discrete-time signal with ZOH as shown in equations (1.3) or (1.5). Currently, many studies have simplified the problem by combining the states and previous actions such as velocity command to compute the Q-value as well as actions command. The algorithm that used action as continuous is called policy optimization. The policy or the neural network can then be updated or backpropagation. But many algorithms based on policy optimization are by using the Actor-Critic architecture and the minimum neural network typically used is 2. The Actor is mapping the observation state space to action state space and the Critic network is used as a function approximator to compute a scalar from action, states, or both.

1.3 Project Objectives

The project objectives as stated below are:

(a) To develop a neural network controller based on the DRL algorithm.

(b) To implement multi-agents during the training phase.

(c) To evaluate the robot behaviour’s performance.

1.4 Project Scopes

The scope of this study is limited to as stated below:

(d) To explore RL in the context of the control system.

(e) To train a policy using multiple of the same robots.

(f) To test the policy after training.

1.5 Report Outlines

This chapter shows brief of history about robotics, especially control system as well as RL. It’s also discussed the problem is facing and what sort of approach or method is used. Given that a lot of these approach is develop from different fields and will have some different in terms of terminology, analysis method, and as well as what sort of tools is used. Noted that many of the system can be analysed same way in controller system where RL is in computer field. But the similarity is very significant despite in different field but different approach which will be discussed in next chapter.

REFERENCES

[1] S. G. Tzafestas, *Introduction to Mobile Robot Control*. .

[2] J. Yan, A. A. Diakité, and S. Zlatanova, “A generic space definition framework to support seamless indoor/outdoor navigation systems,” *Trans. GIS*, vol. 23, no. 6, pp. 1273–1295, 2019.

[3] F. Gul, W. Rahiman, and S. S. Nazli Alhady, “A comprehensive study for robot navigation techniques,” *Cogent Engineering*, vol. 6, no. 1. 2019.

[4] J. Aulinas, Y. Petillot, J. Salvi, and X. Lladó, “The SLAM problem: A survey,” *Front. Artif. Intell. Appl.*, vol. 184, no. 1, pp. 363–371, 2008.

[5] Y. Li, “Deep Reinforcement Learning: An Overview,” Jan. 2017.

[6] L. Buşoniu, T. de Bruin, D. Tolić, J. Kober, and I. Palunko, “Reinforcement learning for control: Performance, stability, and deep approximators,” *Annu. Rev. Control*, vol. 46, pp. 8–28, 2018.

[7] A. Kuhnle, J.-P. Kaiser, · Felix Theiß, N. Stricker, and G. Lanza, “Designing an adaptive production control system using reinforcement learning,” *J. Intell. Manuf.*, vol. 32, pp. 855–876, 2021.

[8] “TurtleBot3.” [Online]. Available: https://emanual.robotis.com/docs/en/platform/turtlebot3/appendix_opencr1_0/. [Accessed: 08-Jun-2022].

[9] B. Alsadik, “Kalman Filter,” *Adjust. Model. 3D Geomatics Comput. Geophys.*, pp. 299–326, Jan. 2019.

[10] X. Wang, W. Xiong, H. Wang, and W. Y. Wang, “Look Before You Leap: Bridging Model-Free and Model-Based Reinforcement Learning for Planned-Ahead Vision-and-Language Navigation.”

[11] D. Silver *et al.*, “Mastering the game of Go with deep neural networks and tree search,” *Nat. 2016 5297587*, vol. 529, no. 7587, pp. 484–489, Jan. 2016.

[12] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, “Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor,” *35th Int. Conf. Mach. Learn. ICML 2018*, vol. 5, pp. 2976–2989, Jan. 2018.

[13] T. P. Lillicrap *et al.*, “Continuous control with deep reinforcement learning,”

in *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings*, 2016.

[14] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, “Stable-Baselines3: Reliable Reinforcement Learning Implementations,” 2021.

[15] E. Liang *et al.*, “RLlib: Abstractions for Distributed Reinforcement Learning,” 2018.

[16] O. I. Abiodun, A. Jantan, A. E. Omolara, K. V. Dada, N. A. E. Mohamed, and H. Arshad, “State-of-the-art in artificial neural network applications: A survey,” *Heliyon*, vol. 4, no. 11. Elsevier, p. e00938, 01-Nov-2018.

[17] Abdel-Nasser Sharkawy, “Principle of Neural Network and Its Main Types: Review,” *J. Adv. Appl. Comput. Math.*, vol. 7, no. 1, pp. 8–19, Aug. 2020.

[18] L. Tai, G. Paolo, and M. Liu, “Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation,” *IEEE Int. Conf. Intell. Robot. Syst.*, vol. 2017-Septe, pp. 31–36, 2017.

[19] X. Chen, J. Hu, C. Jin, L. Li, and L. Wang, “UNDERSTANDING DOMAIN RANDOMIZATION FOR SIM-TO-REAL TRANSFER,” 2022.

[20] J. Liang, U. Patel, A. J. Sathyamoorthy, and D. Manocha, “Realtime Collision Avoidance for Mobile Robots in Dense Crowds using Implicit Multi-sensor Fusion and Deep Reinforcement Learning,” no. May, 2020.

[21] G. Georgakis *et al.*, “Cross-modal Map Learning for Vision and Language Navigation,” Mar. 2022.

[22] K. Smagulova and A. P. James, “A survey on LSTM memristive neural network architectures and applications,” *Eur. Phys. J. Spec. Top. 2019 22810*, vol. 228, no. 10, pp. 2313–2324, Oct. 2019.

[23] I. Rasheed, F. Hu, and L. Zhang, “Deep reinforcement learning approach for autonomous vehicle systems for maintaining security and safety using LSTM-GAN,” *Veh. Commun.*, vol. 26, p. 100266, Dec. 2020.

[24] X. Yu, G. Li, C. Chai, and N. Tang, “Reinforcement learning with tree-LSTM for join order selection,” *Proc. - Int. Conf. Data Eng.*, vol. 2020-April, pp. 1297–1308, Apr. 2020.

[25] M. X. Jiang, C. Deng, Z. G. Pan, L. F. Wang, and X. Sun, “Multiobject tracking in videos based on LSTM and deep reinforcement learning,” *Complexity*, vol. 2018, 2018.

[26] J. D. Williams and G. Zweig, “End-to-end LSTM-based dialog control optimized with supervised and reinforcement learning,” Jun. 2016.

[27] L. Chen *et al.*, “Decision Transformer: Reinforcement Learning via Sequence Modeling.”

[28] V. Mnih *et al.*, “Asynchronous Methods for Deep Reinforcement Learning,” 2016.

[29] L. Tai and M. Liu, “Towards Cognitive Exploration through Deep Reinforcement Learning for Mobile Robots,” Oct. 2016.

[30] L. Butyrev, T. Edelhäußer, and C. Mutschler, “Deep Reinforcement Learning for Motion Planning of Mobile Robots,” Dec. 2019.

[31] J. C. Jesus, J. A. Bottega, M. A. S. L. Cuadros, and D. F. T. Gamarra, “Deep deterministic policy gradient for navigation of mobile robots in simulated environments,” *2019 19th Int. Conf. Adv. Robot. ICAR 2019*, pp. 362–367, Dec. 2019.

[32] X. Zhang, J. Ma, Z. Cheng, M. Tomizuka, and T. H. Lee, “Velocity Obstacle Based Risk-Bounded Motion Planning for Stochastic Multi-Agent Systems,” 2022.

[33] G. Brockman *et al.*, “OpenAI Gym,” Jun. 2016.

[34] K. Takaya, T. Asai, V. Kroumov, and F. Smarandache, “Simulation environment for mobile robots testing using ROS and Gazebo,” *2016 20th Int. Conf. Syst. Theory, Control Comput. ICSTCC 2016 - Jt. Conf. SINTES 20, SACCS 16, SIMSIS 20 - Proc.*, pp. 96–101, Dec. 2016.

[35] T. Okudo and S. Yamada, “Reward Shaping with Subgoals for Social Navigation,” Apr. 2021.

[36] N. Botteghi, B. Sirmacek, J. Ai, K. A. A. Mustafa, M. Poel, and S. Stramigioli, “On Reward Shaping for Mobile Robot Navigation: A Reinforcement Learning and SLAM Based Approach,” Feb. 2020.

[37] W. Zhang, N. Liu, and Y. Zhang, “Learn to Navigate Maplessly with Varied LiDAR Configurations: A Support Point-Based Approach,” *IEEE Robot. Autom. Lett.*, vol. 6, no. 2, pp. 1918–1925, 2021.

[38] D. Ferigo, S. Traversaro, G. Metta, and D. Pucci, “Gym-Ignition: Reproducible Robotic Simulations for Reinforcement Learning,” *Proc. 2020 IEEE/SICE Int. Symp. Syst. Integr. SII 2020*, pp. 885–890, Jan. 2020.

[39] M. G. Bellemare, W. Dabney, and R. Munos, “A Distributional Perspective on

Reinforcement Learning," 2017.

[40] F. P. Audonnet, A. Hamilton, and G. Aragon-Camarasa, "A Systematic Comparison of Simulation Software for Robotic Arm Manipulation using ROS2," 2022.