Applied learning of artificial intelligence techniques by using the Gazebo simulator and Turtlebot3 multi-robot system

Alexandru STAN, Mihaela OPREA

Petroleum-Gas University of Ploiesti
Department of Automatics, Computers and Electronics
39 Bucuresti Av., RO-100680, Romania
alex_stan2010[at]yahoo.com, mihaela[at]upg-ploiesti.ro

Abstract: *Robotics became an important educational testing resource for different methods and techniques, especially from the artificial intelligence domain. Thus, either classical artificial intelligence methods (such as planning and scheduling methods, informed search strategies, knowledge based systems, intelligent agents and multi-agent systems) or computational intelligent methods (such as artificial neural networks, genetic algorithms, swarm intelligence and nature-inspired algorithms) have been applied in different real world or simulated systems that were developed by using educational robots (e.g. Khepera, Turtlebot, Pioneer, Nomad, LegoMindStorm). Various educational robot's simulation frameworks or integrated software tools have been developed so far (e.g. Webots, Gazebo, MiMicS, RoboNetSim). The paper focuses on some artificial intelligence techniques learned by using the Gazebo robot simulator and a Turtlebot3 multi-robot system. Details related to, for example, how is configured a multi-robot system, how is made the simulation of the multi-robot system are provided step by step in order to improve student learning process efficiency.*

Keywords: Educational robotics, Artificial intelligence techniques, Multi-robot system configuration, Robot simulator.

1. Introduction

The educational area has undergone significant changes in the past few years being impacted by Covid pandemic and technological enhancement, many courses being switched from traditional classes to online. The teachers adapt very quickly to these changes using online platforms to deliver the information needed, but the laboratories were the biggest problem. How can we move all the instrumentation and equipment's needed in online? How can we deliver to students the hand-on experience of a laboratory using an online approach?

https://doi.org/10.58503/icvl-v17y202210

There are no simple answers to those questions. The only possibility is the development of virtual environments using simulator platforms and models for the equipment used in laboratory classes.

In this paper, we will present all the steps needed for creating a virtual learning environment for studying artificial intelligence using Gazebo simulator and Turtlebot 3 mobile robot. The simulating system will be developed using Robotic Operating System or ROS, an open-source platform dedicated to developing algorithms for robot's control. The advantage of using this platform is the fact that the methods developed in the simulator can be also used in a real multi-robot system.

Artificial intelligence is a fast-growing domain with vast applicability in many domains such as robotics, autonomous driving, economics and medicine. In the domain of mobile robots' artificial intelligence increases the capability of the robots to quickly adapt and explore unknown environments, make decisions and collaborate with other robots or humans.

The aim of this paper is to create an original method and to present the method of learning to students that want to start learning artificial intelligence, that can facilitate learning using a simulator in which the methods can be tested on a mobile robotic system.

Teaching artificial intelligence to students can be a challenging task but using a hand-on experience using a mobile robot such as Turtlebot 3 robot and Gazebo simulator could be a funny and easy to understand approach.

The paper is structured as follows: Section 2 presents a brief overview on multi-robot system modelling and simulation, the Section 3 presents the steps needed to install and configure the Gazebo simulator for simulating a Turtlebot 3 multi robot system and in Section 4 is presented a training scenario for artificial intelligence methods and some examples of methods to be used in the educational purpose. In the last section of the paper, we will present the conclusions and the future works.

2. Multi-robot system modelling: Case study Turtlebot 3 multi-robot system

In the past years, it was observed an increase in interest of using Gazebo and Petri Nets for simulating different environment for example in the paper (Kim et al., 2021) present a method for evaluating multi-robot formation control algorithms using a simulator developed in Gazebo or in the paper (Kloetzer et al., 2020) present a Petri Net simulation for path planning for robotic teams.

The modelling of a multi-robot system involves three main stages: design or choose a model for the multi-robot system, implement the model as a simulation and test it for various scenarios while doing model adaptation to the simulation results, and implement and test the final version of the model in a real-world multi-robot system.

Each stage tackles in detail the requirements related to robot coordination, robot communication and control and so on, according to the multi-robot system goal achievement (e.g. unknown environment exploration, autonomous driving or following a specific path in a very strict way inside a manufacturing plant). In this section, we will make a short overview on multi-robot system modelling and simulation.

From the variety of multi-robot systems models that were proposed in the literature, we mention three types: the multi-agent model, the Petri Nets based model (Stan & Oprea, 2020), and swarm intelligence-based model. A very short overview of some selected papers that use such models is made in this section.

Intelligent agents can easily model mobile robots due to their basic characteristics: autonomy, pro-activity, reactivity and social ability. Moreover, multi-agent systems which are systems composed of minimum two intelligent agents that have a common global goal and are embedded in a dynamic, usually, unknown environment, represent a straightforward model for multi-robot systems being distributed systems. Several research papers on modelling a multi-robot system as a multi-agent system were reported in the literature. For example (Calegari et al., 2020) in her systematic literature review point out that the agents and multi-agent systems (MAS) have been at the core of the design of intelligent systems since their very beginning, and their long-term connection with logic-based technologies, which characterised their early days, might open new ways to engineer explainable intelligent systems. We will take that point and we will extend further by giving new engineers the tools needed to develop the skills needed for creating intelligent systems.

For example, one of the challenging problems in mobile multi-robot systems is their control and coordination algorithm as is described in the paper (Ben-Gal et al., 2020) for large scale swarm systems or in (Stan, 2022) for small scale mobile robotic systems, the computational model of the system can be very hard to handle with a classical approach. The A.I. methods can be the solution for such systems, but almost all A.I. methods used for mobile robotics need advanced simulators and robot system models in order to do the training of the method in a medium close to the real environment.

For the simulator, we will use the Turtlebot3 Burger mobile robots. TurtleBot3 Burger is a small, affordable, programmable, ROS-based mobile robot for use in education, research, hobby, and product prototyping. The goal of TurtleBot3 developers is to dramatically reduce the size of the platform and lower the price without having to sacrifice its functionality and quality.

TurtleBot3's core technology is simultaneous localization and mapping (SLAM), making it the perfect choice for our application of developing and testing different A.I. methods into a simulated environment and to learn from results.

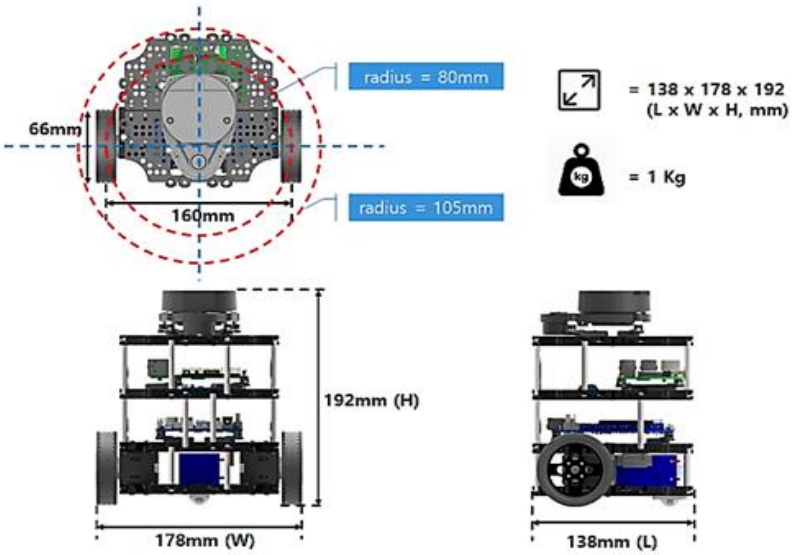

Figure 1. Turtlebot3 Burger robot specifications

In figure 1 is presented the Turtlebot3 Burger robot with the dimensional specifications. The system used in the simulator will be composed of three of these robots which can cooperate and communicate with each other in order to accomplish certain tasks.

For the modelling, we will consider the mathematical model of a 2 wheels mobile robot described in "Programming Robots with ROS" (Quigley et al., 2017) and the Turtlebot3 Burger model described in the paper the Turtlebot3 Burger model described in (Nica et al., 2021) as follows:

$$v_l = \frac{E_{lc} - E_{lp}}{T_e} * \frac{\pi}{180} \; (rad/sec) \quad (1)$$

$$v_r = \frac{E_{rc} - E_{rp}}{T_e} * \frac{\pi}{180} \; (rad/sec) \quad (2)$$

Where:

v_l- is the rotational speed of the left wheel

v_r- is the rotational speed of the right wheel

E_{lc}, E_{rc} - is the current value of the encoder of left/right wheel

E_{lp}, E_{rp}- is the previous value of the encoder of left/right wheel

Knowing that the radius of the Turtlebot3 Burger wheel is 65mm we can calculate the linear and angular speeds for our system:

$$V_l = v_l * 65 \text{ (m/sec)} \quad (3)$$

$$V_r = v_r * 65 \text{ (m/sec)} \quad (4)$$

$$\omega_k = \frac{V_r - V_l}{D} \text{ (rad/sec)} \quad (5)$$

Where: V_l, V_r- is linear speed of the left/right wheel and ω_k is the angular speed of the robot.

3. Install and configuration for Gazebo simulator

Gazebo is an open-source 3D simulator that provides realistic rendering of environments using high-performance physics engines such as ODE and Bullet. In that simulator we can model different types of mobile robots such as Turtlebot3, Khepera, etc and many types of sensors used alongside A.I. techniques such as laser range finders, cameras, Kinect, etc.

System requirements for Gazebo simulator are (source mathworks.com):

- Processor (CPU) - Quad core Intel® i5, or equivalent;
- Memory (RAM) – 4 GB or more;
- Graphics card(GPU) - Dedicated GPU with 1 GB or more graphics memory;
- Disk space – At least 20 GB free disk space.

For our example, it is also needed to install the Robotic Operating System (ROS) for the Turtlebot 3 Burger packages. Next in the paper, I'll present the installation procedure for booth ROS and Gazebo, bases on the documentation provided by the developer Robotis:

STEP 1: Install Ubuntu Ubuntu 16.04 LTS Desktop

STEP 2: Install ROS Kinetic

STEP 3: Install ROS Packages

STEP 4: Install Turtlebot packages and simulation packages from Robotis official documentation

STEP 5: Configure the env variables for Turtlebot 3 robot:

Modify the content of ~/.bashrc using nano or vi, the content is displayed in Figure 2.

```
export ROS_MASTER_URI=http://192.168.0.103:11311
export ROS_HOSTNAME=192.168.0.103
export TURTLEBOT3_MODEL=burger
```

Figure 2. Environment variables configuration for Ubuntu

For ROS_MASTER_URI and ROS_HOSTNAME use IP of the laptop/PC. For the variable TURTLEBOT_MODEL use "burger".

STEP 6: Modify the ROS_MASTER_URI and HOSTNAME in ~/.bashrc

STEP 7: Install Tensorflow and Keras

1.Install Anaconda
2.Install ROS required packages
$ pip install msgpack argparse
$ pip install -U rosinstall empy defusedxml netifaces
3.Install Tensorflow
4.Install Keras

More information regarding the Turtlebot 3 packages can be found in documentation of the Turtlebot 3 from Robotis (Robotis e-Manual for Turtlebot 3-Open Robotics. https://emanual.robotis.com/).

Now, the Robot Operating System (ROS) and simulator Gazebo should be installed and configured to run the simulations for Turtlebot 3 multi-robot system. ROS together with Tensorflow permits the development of various A.I. algorithms in different languages such as Python, C or C++. Machine learning methods for robotic systems can be implemented and studied using the simulator Gazebo.

4. Training scenario for Artificial Intelligence methods using Gazebo simulator

The training scenario for the multi-robot system will be defined in Gazebo. We will use 4 different scenarios, with different grades of difficulties to train a Deep Q-learning method for environment exploration. On each step the simulator will generate a new objective location and the robots will try to reach that location.

The Deep Q-learning algorithm defines the learning rules for the intelligent agents using the next steps:

1. Initialize Q with a random value
2. For the current observation S select an action A with probability $\in$
$$A = arg_A max Q(S, A; \phi) \quad (6)$$
3. Execute action A, add reward R and store the next observation S'
4. Add experience (S,A,R,S') to experience vector
5. Take a sample of M experiences (S_i, A_i, R_i, S'_i) from experiences vector
6. If S'_i is the final state, set the value for the objective function y_i with R_i
$$y_i = R_i + \gamma max_{A'} Q_t(S'_i, A'; \phi_t) \quad (7)$$
7. Update Q by minimizing the values from experience vector
$$L = \frac{1}{M} \sum_{i=1}^{M} (y_i - Q(S_i, A_i; \phi))^2 \quad (8)$$
8. Update the value of $\in$ using the decay rate chosen

This is an example for one of the many artificial intelligence algorithms that can be applied in mobile robotics. Using these steps, we can implement in ROS, using C++ or Python, the control method for the simulated robot system.

Next, we will present in Gazebo the training arena for the algorithm using different difficulties. The four arenas presented in the simulation package provided

by the Robotis and the complete documentation can be found in the official documentation on the Robotis website. The first one will be a 4x4 meters arena with walls and no objects for the first training scenario described in Figure 3.

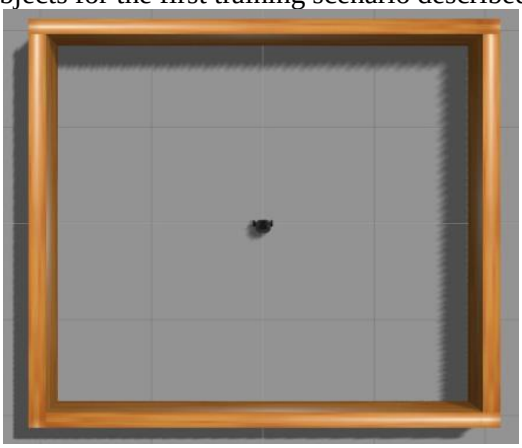

Figure 3. Training scenario 1 – no objects

The second scenario will have the same 4x4 meters map but this time with fixed round obstacles inside.

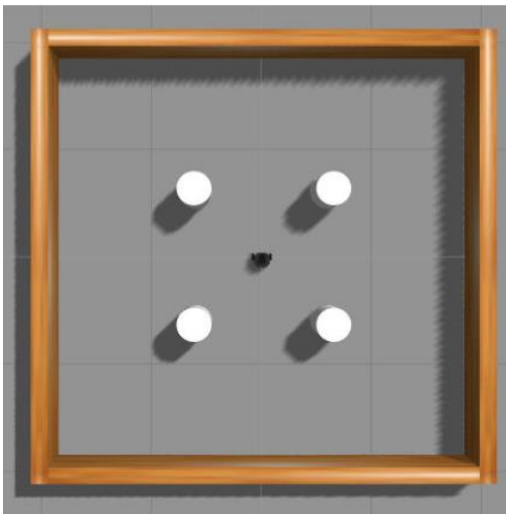

Figure 4. Training scenario 2 – fixed obstacles

As we can see in Figure 4, the robot start initial position is in the centre of the map and we have 4 fixed round objects inside. The objective location will be generated random on each step and the robot will try to reach that location without collide the obstacles.

Using the DNQ algorithm, in case of collision the robot will receive a big penalty and in case of reaching the objective the robot will receive a reward. By

minimizing the experience vector, we will try to obtain the shortest path from initial location to the objective.

The next scenario will introduce the dynamical obstacles, represented by the same 4 round object, but this time the objects moves in a circular path.

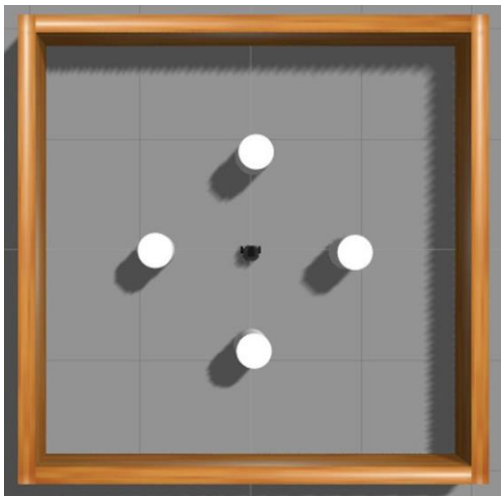

Figure 5. Training scenario 3 – moving obstacles

In Figure 5 we can see the obstacles and the robot's initial position. The rules are the same but now the training time for the DNQ algorithm will be longer due to the complexity introduced by the moving obstacles.

The last training scenario is the most difficult and simulates the real environment. The map this time is a maze type map with a wall inside and two random moving obstacles. The map is presented in Figure 6.

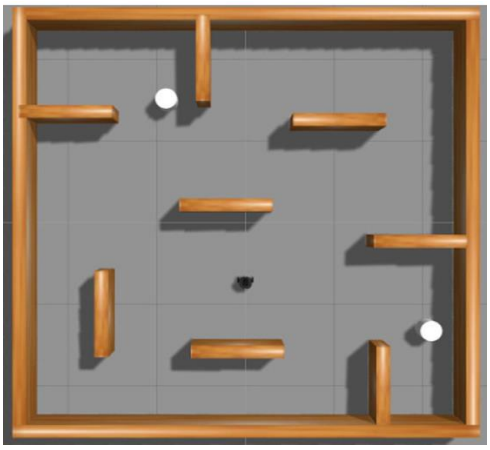

Figure 6. Training scenario 4 – maze with moving obstacles

5. Conclusion and Future Work

In conclusion, we can say that the virtual environment presented in this paper can be used for studying and developing artificial intelligence-based methods and can be tested using the Turtlebot 3 mobile robot system. ROS and Gazebo simulator offers good interoperability with new technologies used in AI such as Tensorflow and Keras and also offers the possibility to develop in many popular programming languages such as Python and C++.

As we have shown in the section 4, Gazebo offers the tools for creating a very detailed environment for training and testing machine learning methods with many pre-build libraries and objects to easily deploy objects or interactive agents such as the robot Turtlebot 3.

As further work, I intend to use this virtual environment to develop and study machine learning methods for environment exploration using a multi Turtlebot 3 robot system.

References

Aakash S., Huosheng H. (2019). A multi-robot simulator for the evaluation of formation control algorithms. *11th Computer Science and Electronic Engineering (CEEC) 2019.*

Ben-Gal, I., Ben Chaim, M., Ben-Moshe, B., Hacohen, S., Kagan, E., Khmelnitsky, E., Lineykin, S., Medina, O., Novoselky, A., Scvalb, N., Shovel, S., Yozevitch, R. (2020). *Autonomous Mobile Robots and Multi-robot systems Motion-Planning, Communication and Swarming*, Wiley.

Calegari, R., Ciatto, G., Mascardi, V., Omicini, A. (2020). *Logic-based technologies for multi-agent systems: a systematic literature review*, Springer.

Gazebo Official website: https://gazebosim.org/home

Kloetzer, M., Mahulea, C. (2020). Path planning for robotic teams based on LTL specifications and Petri net models. *Discrete Event Dyn Syst*, 30, 55–79.

Nica, C., Stan, A. C., Oprea, M. (2021). On the development of a mobile TurtleBot3 Burger multi-robot system for manufacturing environment monitorization, *International Conference on Emerging Trends and Technologies on Intelligent Systems ETTIS 2021*.

Quigley, M., Gerkey, B., Smart, W. D. (2015). *Programming Robots with ROS*, O'Reilly.

Robotis e-Manual for Turtlebot 3- Open Robotics. https://emanual.robotis.com/

ROS Official website: https://www.ros.org/

Stan, A. C. (2022). A decentralised control method for unknown environment exploration using Turtlebot 3 multi-robot system, *ECAI 2022*.

Stan, A. C., Oprea, M. (2020). Petri Nets Based Coordination Mechanism for Cooperative Multi-Robot System. *Journal of Electrical Engineering, Electronics, Control and Computer Science*.

Tensorflow Official website: https://www.tensorflow.org/

Ubuntu Official website: https://ubuntu.com/