Practical Training Using Simulation and Robotic Kits for Better Future Engineers

Ali Younes[1]*and *Zain* Alabedeen Ali[1]

[1]Bauman Moscow State Technical University, 2nd Baumanskaya str., 5/1, 105005, Moscow, Russia

Abstract. Every engineering course has two main parts; theoretical and practical. In the last decades, the gap between various engineering institutions has been reduced by using online resources, where all the students and lecturers have access to technological advances in their field. The difficulty arises in the practical part; to gain valuable engineering skills, students need hands-on experience, which means implementing the theoretical part and infer the results of their work. The top-tier institution provides its students with needed equipment and facilities, while many others struggle to fund one plat-form (e.g. universities in small cities and developed regions). To address this problem, we propose using simulation and robotic kits (Lego Mindstorms) in the practical part of courses related to mobile robots, and fuzzy logic. To show that, it is possible to improve the engineering curriculum in the cases of low funds, and improve the quality of future engineering graduates.

1 Introduction

The methodology of preparing engineers [1-7] differs from institution to the other, with one shared goal to produce high-qualified engineers. A high-qualified engineer is an engineer who has the essential engineering knowledge in his field, alongside the ability to apply his knowledge in real-world problems. During the undergraduate study, every engineering student can gain – to some extent – the basics of engineering in classes and can extend his knowledge about his areas of interest by using available resources. While gaining real-world experience, a student has to implement what he has learned. Implementation could be in a type of homework, laboratory work, or project.

The training of robotics engineers seems to be challenging. Many universities can't afford the cost of complex robotics systems needed for the practical part of robotic courses. This problem is a real handicap that faces most of the program coordinators especially in the developing countries, and in most cases, the lecturer chooses to ignore the practical part and makes the course heavily direct to the theory of that discipline.

The results obviously, students have to study vague algorithms just to pass the exam and forget them after that.

Motion planning for mobile robots is a great example of our problem. Localization [8], global and local motion planning [9, 10, 11], considered fertile directions for improving the

* Corresponding author: ay20-5-1994@hotmail.com

engineering mentality of the students. The huge number of available algorithms, with the improvements from one to the other, could help students to form ideas about how to develop systems to solve problems. Implementing what has been taught in lectures, would deepen the understanding, and improve thinking skills. The problem is that most of the work in that direction relies on using expensive robotic platforms equipped with a set of sensors (LIDAR, RealSense cameras ...) which costs thousands of dollars. The result is most of the courses worldwide are taught in classes, without any implementation.

In this paper, we are concerning this problem, where we have two solutions in mind, the first is to use open-source robotic simulators (e.g. V-REP [12], Gazebo [13], and Webot [14]). The second is to use available robot kits (e.g. Lego Mindstorms [15]). We will introduce a comprehensive study about using the second solution (Lego Mindstorms platform) to achieve a motion planning algorithm. Also, we will provide additional usage on the level of the low-level controller; Where we use a fuzzy logic controller [16], supporting the main idea of the paper.

The experiments will be on a Lego EV3 robot. The motivation behind choosing Lego, is its availability by large numbers in a lot of technical institutions, even in high-schools. Also, the ease of use and extendibility. The challenges in our choice are the low resources of the EV3 brick, where its specifications: ARM9 microcontroller, 16 MB flash memory, and 64 MB RAM. An additional challenge was to use only parts from one kit, adding just a wifi dongle.

The results of this paper was a training package for mobile robots and fuzzy logic courses, with modular blocks, allows students to make experiments with a wide range of algorithms. The package was developed in MATLAB [17]/ SIMULINK[18], using the Lego support package [19] and the fuzzy logic toolbox[20].

2 Motion planning systems for mobile robots

The essential task for a mobile robot is to go to a target position and do a specific task there. It is very simple in words, but in reality, it consists of a couple of important subproblems. The correct way to address this problem, and any robotic problem, is to divide it into the following steps:

2.1 Perception

The robot has to determine its position relative to a predefined coordinate system, depending on its motion and data from sensors. Here we have two cases; (1) the surrounding environment is known (given) to the robot, and its initial position also is known, here the perception goal is to solve a pure localization problem. (2) unknown environment; hence the robot has to build a map a localize in it simultaneously, i.e. a SLAM (Simultaneous Localization And Mapping) problem.

In this paper, our demonstration is from the first case, the map will be given to the robot, and the localization system uses data from wheels' encoders and a gyroscope sensor.

2.2 Planning

After knowing its position, and get the map of the environment, the robot has to plan a trajectory, by which he can reach the goal. The trajectory itself is a set of points (sub-goals), the goal is to find the optimal (according to some criteria) trajectory, and adapt to the changes of the environment. The planning problem is seen as a combination of (1) global planner, which tries to find the best possible trajectory from the start to the end, considering a static environment and known (or built) map. (2) Local planner, which avoids obstacles and plan to reach the goal also as optimal as possible.

In this paper, we have a dedicated block for the global planner, the trajectory will be computed on the host PC, before running it on the robot. As a test, we have implemented the A* algorithm. The local planner will a separate block as part of the program which runs on the robot.

2.3 Control

The control aims to assure that the robot is following the planned trajectory, i.e. transforming the commands to motor torques. Mostly, PID controllers are used as low-level controller, in recent years, a lot of non-conventional controllers proposed, as an adaptive control method.

In this paper, we are using a fuzzy logic controller, it uses information about the position and the orientation of the robot, to produce velocity commands to motors of the robot, the results were better than using PID controller, also without the need for tuning controller constants.

3 Perception part

The perception part is computed by using the odometry block in our system; The inputs are the past position of the robot (x, y, and theta (course angle)), motors’ encoders, and the gyroscope reading.

The position is computed by the following equations (variables match Fig. 1):

$$\delta s = \frac{\delta s_1 + \delta s_2}{2} * r_{wheel}$$

$$\theta_{new} = \theta_{past} + \delta\theta$$

$$x_{new} = x_{past} + \delta s * \cos(\theta)$$

$$y_{new} = y_{past} + \delta s * \sin(\theta)$$

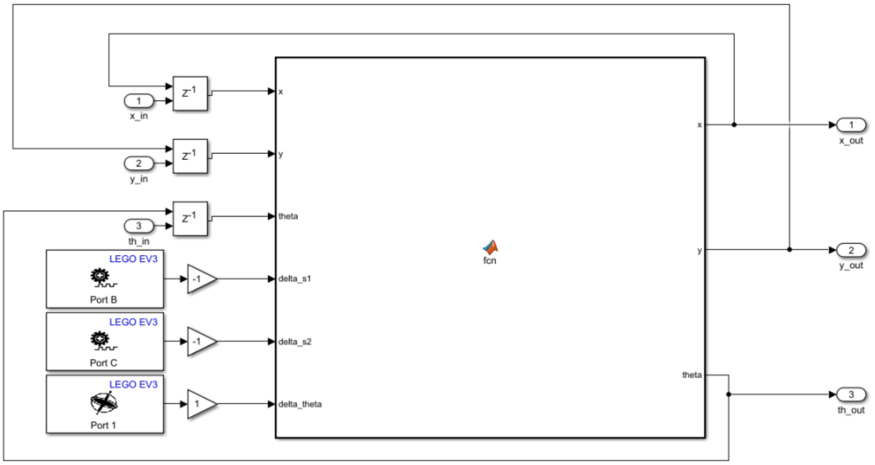


Fig. 1. Odometry system

Testing this block was done by running it on the real robot, drawing the computed path in real-time, and compare with the real robot movement. It is better to test it alongside the low-level controller.

4 Planning part

The global planning step was achieved off-line on the host PC, after generating the optimal trajectory, we have run an additional step to find the critical points (critical points are the vertices of the trajectory on which the robot has to rotate). The implemented algorithm A*, take as input the initial position of the robot (x_init, y_init), the goal position (x_goal, y_goal), and the discrete map of the environment as a png pho-to (will be transferred to an occupancy map with just ones and zeros representing free space or obstacles). The output of the algorithm is a vector of critical points (x_critic, y_critic).

The student could test any other planning algorithm to use it as part of our system, and compare the results, as it is a modular Matlab function and it runs offline. A test case for our implementation is given in Fig. 2.

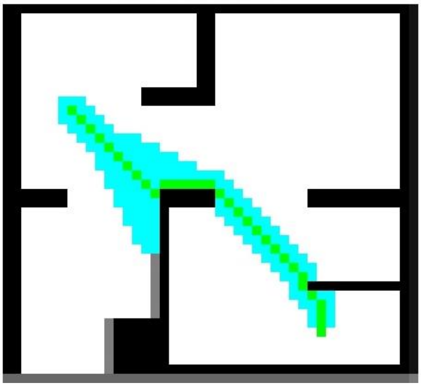


Fig. 2. A test case for our A* implementation

For the local planner, we have used an ultrasonic sensor to detect obstacles along the trajectory, when detecting an obstacle, the robot will avoid it, the student has the following choices for it (1) turn the robot for a predefined angle, and check if it can avoid the obstacle and go back to its planned trajectory, (2) add two additional critical points to the path, and continue running the updated trajectory, trying to avoid the obstacle automatically, or (3) update the map, and rerun the global planner from its current position to the goal, to find the optimal solution (local planner block is shown in Fig. 3).

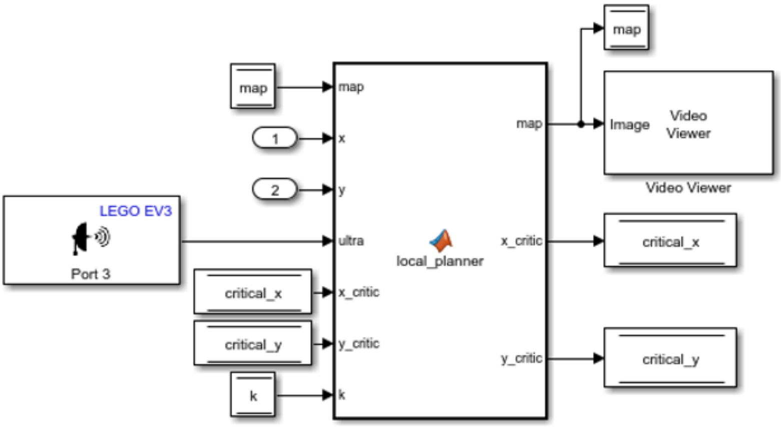


Fig. 3. Local planner block

5 Control part

We have used a fuzzy logic controller [21], the input for it is the angle of deviation (Q) and the distance to the next critical point (dist), the outputs are the velocity of the motors.

In the fuzzy logic controller, we have to define member functions for all inputs and outputs, assigning words describing each situation, after defining words we have to define rules in a wordy paradigm. Member function and rules of our controller are shown in figures 4, 5, 6, and table 1.

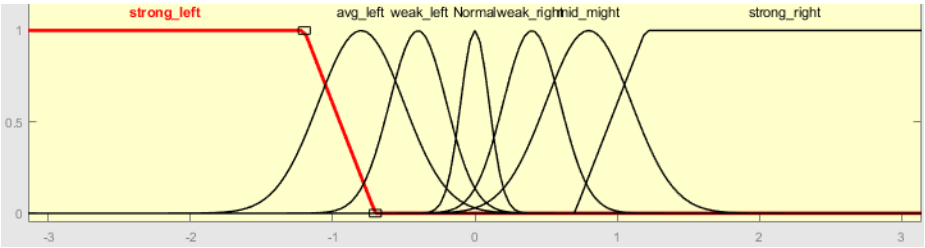


Fig. 4. Member function for the angle of deviation Q (input)

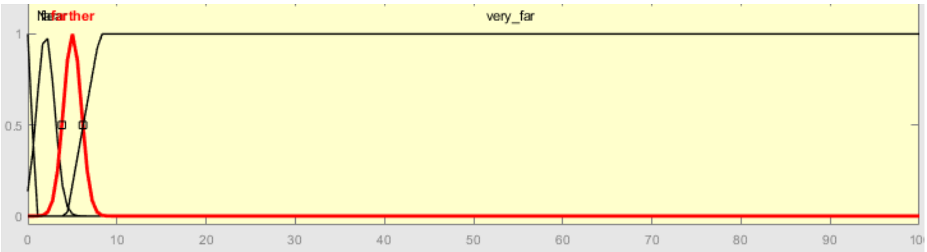


Fig. 5. Member functions for the distance to the goal (input)

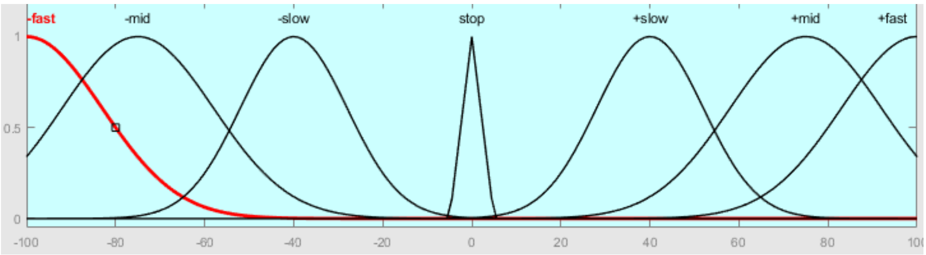


Fig. 6. Member function for the velocity of one of the motors (output)

Table 1. Rules of the fuzzy logic controller (- corresponds to all the cases)

Q	distance	Left motor	Right motor
-	Near	stop	stop
Strong left	-	+fast	-fast
Avg left	-	+mid	-mid
Weak left	-	+slow	-slow
Normal	Far	+mid	+mid
Normal	Farther	+fast	+fast
Normal	Very far	+fast	+fast
Weak left	-	-slow	+slow
Avg left	-	-mid	+mid
Strong left	-	-fast	+fast

6 The whole system and results

The whole motion planning system consists of the Matlab function of the global planner, which will give two vectors (x_critic, y_critic) to the Simulink model shown in Fig.7. The model consists of an odometry block producing coordinates that will be used in the local planner and the controller, we have a Matlab function that computes the angle of deviation and distance to the next critic point, and those will be fed to the fuzzy logic controller, which outputs the velocity for each of the motors.

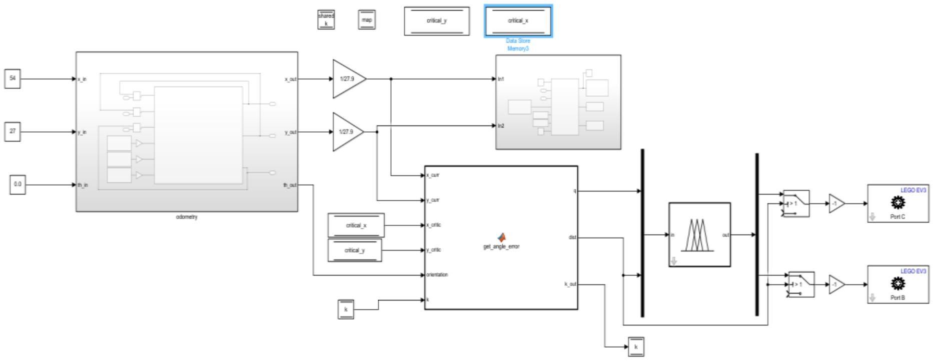


Fig. 7. The whole motion planning system

The system was tested on a real Lego robot, connected to the host PC using WIFI, and running the simulation in an external mode (to receive data from the robot), by feeding a map of the workspace, the start, and the goal position, the robot can successfully achieve its mission.

The code, the Simulink model, and the robot design are available upon request.

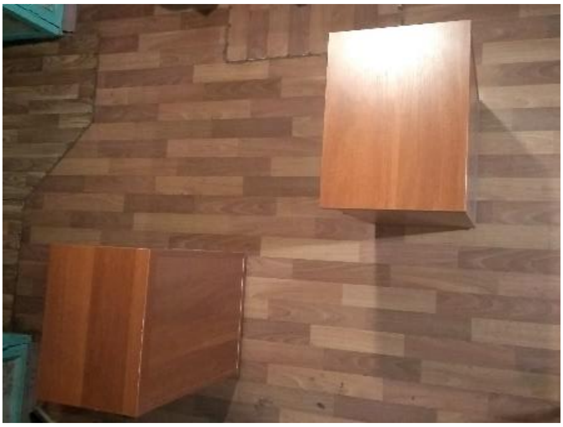
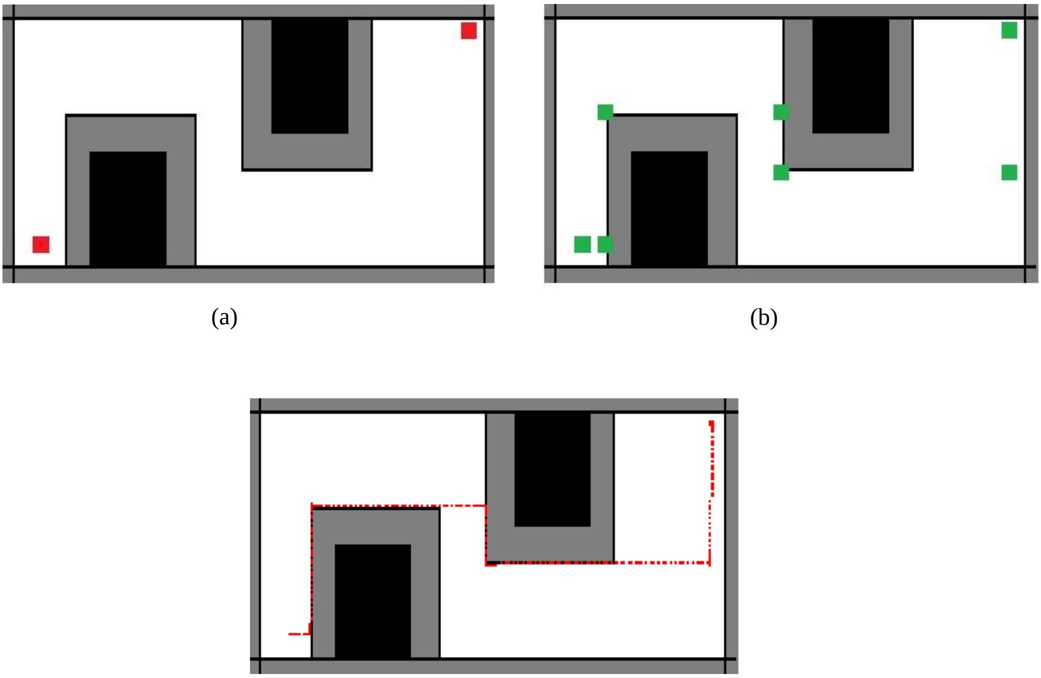


Fig. 8. A test case for our system (a real-world photo)



(a)

(b)

(c)

Fig. 9. Visualization of the process of motion planning (from left to right)
(a) The interpreted map with obstacles in black, safety constraints in gray, and start and goal position in red
(b) The critical points in green as the output of the global planner (theoretical path)
(c) The real path drawn according to the received data from the odometry system

7 Conclusion

In this paper, we have discussed the problem of training the engineering student in cases of low resources, like the case in most developing countries. One of the solutions is to use robotic kits that are available, to test, implement, and compare various algorithms.

We have concentrated on the motion planning for mobile robots, in addition to using a fuzzy logic controller as a low-level controller. We proposed a system consists of a state estimator (odometry), a global planner, a local planner, and a fuzzy logic controller. Analyzing the requirements of the motion planning system, we have built a package that able to implement the whole system in MATLAB/SIMULINK and implemented it on a mobile robot built using the Lego EV3 kit.

The proposed system gives good practice for students studying topics related to robotics, can be used as part of the practical work for courses related to mobile robots and fuzzy logic, and it is modular in a way allows the student to test many global and local planner algorithms, also to test various combination for the member functions and rules for the fuzzy logic controller.

For future work, we are concerning study and suggest other solutions for the problem, like using simulation software for machine learning experiments, use sensors from mobile phones in robotics projects, and develop novel usage of cheap equipment.

References

1. K.A. Gary, *The Software Enterprise: preparing industry-ready software* engineers, in Software Engineering: Effective Teaching and Learning Approaches and Practices, pp. 115-135, IGI Global (2009)
2. T. Nataliya, E. Olga, B. Vladimir, & S. Daria, *The Design methodology of Bauman Moscow State Technical University*, In IFToMM World Congress on Mechanism and Machine Science, pp. 843-851, June 2019, Springer, Cham (2019)
3. T.A. Ward, *Common elements of capstone projects in the world's top-ranked engineering universities*, in European Journal of Engineering Education, **38(2)**, pp. 211-218 (2013)
4. A. Yadav, G.M. Shaver & P. Meckl, *Lessons learned: Implementing the case teaching method in a mechanical engineering course*, in Journal of Engineering Education, **99(1)**, 55-69 (2010)
5. A.A. Larionova, N.A. Zaitseva, A.S. Fadeev, V.V. Filatov, V.N. Zhenzhebir & T. S. Pshava, *The use of organizational and technological innovations in the process of managerial and engineering personnel's training*, in Eurasian journal of analytical chemistry, **12(7b)**, 1573-1580 (2017)
6. V. Mayorova, *Integration of educational and scientific–technological areas during the process of education of aerospace engineers*, in Acta Astronautica, **69(7-8)**, 737-743 (2011)
7. V. Shakhnov, A. Vlasov, L. Zinchenko & E. Rezchikova, *Visual learning environment in electronic engineering education, in 2013 International Conference on Interactive Collaborative Learning (ICL)*, September 2013, pp. 379-388, IEEE (2011)
8. F. Dellaert, D. Fox, W. Burgard & S. Thrun, *Monte carlo localization for mobile robots, in ICRA* (Vol. 2, pp. 1322-1328), May 1999, (1999)
9. P. Moghadam, W.S. Wijesoma & D.J. Feng, *Improving path planning and mapping based on stereo vision and lidar, in 2008 10th International Conference on Control, Automation, Robotics and Vision*, (pp. 384-389), December 2008, IEEE, (2008)
10. R. Fedorenko & B. Gurenko, *Local and global motion planning for unmanned surface vehicle, in MATEC Web of Conferences* (Vol. 42, p. 01005), EDP Sciences (2016)

11. S. Thrun, *When robots meet people: Research directions in mobile robotics*, IEEE Intelligent Systems (1998)
12. E. Rohmer, S.P. Singh & M. Freese, *V-REP: A versatile and scalable robot simulation framework, in 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1321-1326, IEEE, November, (2013)
13. N. Koenig & A. Howard, *Design and use paradigms for gazebo, an open-source multi-robot simulator*, in 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No. 04CH37566) (Vol. 3, pp. 2149-2154), IEEE, September, (2004)
14. O. Michel, *Cyberbotics Ltd. Webots™: professional mobile robot simulation*, in International Journal of Advanced Robotic Systems, 1(1), 5, (2004)
15. D. Benedetelli, *LEGO MINDSTORMS EV3 Laboratory: Build, Program, and Experiment with Five Wicked Cool Robots. No Starch Press*, (2013)
16. L.A. Zadeh, *Fuzzy logic= computing with words*, in Computing with Words in Information/Intelligent Systems 1 (pp. 3-23), Physica, Heidelberg, (1999)
17. MATLAB 2019a, The MathWorks, Inc., Natick, Massachusetts, United States.
18. SIMULINK 2019a, The MathWorks, Inc., Natick, Massachusetts, United States.
19. Fuzzy Logic Toolbox for MATLAB 2019a, The MathWorks, Inc., Natick, Massachusetts, United States.
20. Simulink Support Package for LEGO MINDSTORMS EV3 Hardware MATLAB 2019a, The MathWorks, Inc., Natick, Massachusetts, United States.
21. A.M. Kostygov, D.A. Dadenkov & A.A. Kaverin, *Nechetkay Sistema upravleniya dvizheniya mobilnogo robota*, in Modern problems of science and education, **(5)**, 136-136, (2013)