Learning the Goal Seeking Behaviour for Mobile Robots

Abhinav Khare, Ritika Motwani, S. Akash, Jayesh Patil, Rahul Kala
Department of Information Technology
Indian Institute of Information Technology
Allahabad, India

e-mails: abhinav.khare31@gmail.com, ritika.motwani0712@gmail.com**,** akashshivram9@gmail.com,

donny6397@gmail.com, rkala001@gmail.com

Citation: A. Khare, R. Motwani, S. Akash, J. Patil and R. Kala, "Learning the Goal Seeking Behaviour for Mobile Robots," 2018 3rd Asia-Pacific Conference on Intelligent Robot Systems (ACIRS), Singapore, 2018, pp. 56-60.
Full Text Available At: https://ieeexplore.ieee.org/abstract/document/8467230

Abstract—**Machine Learning techniques have been widely used for the navigation of mobile robots moving towards a region of interest while avoiding obstacles. Surprisingly, there is a sparingly little literature on the use of machine learning techniques for navigating the robot towards a precisely defined goal configuration amidst static and dynamic obstacles. The need to have the robot reach a precise configuration is needed for applications like placing the robot for charging, robots carrying mobile manipulation, etc. This paper takes the problem of planning motion of an autonomous robot to reach a specific goal configuration in presence of static and dynamic obstacles as a machine learning problem; and compares two approaches, namely supervised learning and reinforcement learning using neural networks. The comparisons are done on a simulation as well as on the physical robot Amigobot operating at the robotics laboratory of the host institute. Experimental results show that reinforcement learning is more suited to solve the problem as the technique does not require a human expert to generate data, which is hence expensive.**

Keywords-motion planning; machine learning; neural networks; socially aware robot navigation; supervised learning; reinforcement learning.

I. Introduction

With the growing industry of self-driving cars, advancements in astronomical field and introduction of robots, there is a need for robots to become autonomous and be able to navigate themselves in a dynamic environment. For this purpose, we need accurate motion planning and tracking algorithms, many of which are inspired by supervised and reinforcement learning techniques. This paper's aim is, to provide a solution for a scenario where, given an initial position, the robot has to reach the goal configuration while avoiding all the obstacles (static as well as dynamic) on the map [1, 2]. Many algorithms and approaches have been proposed for motion planning using different technologies. Deliberate algorithms are time-consuming. Hence, we use reactive algorithms with the help of machine learning for the aforementioned objective.

This paper uses two machine learning techniques, supervised learning and reinforcement learning for solving the problem of reaching a predefined goal configuration. In general, prior efforts have already been made for the navigation of the robots using machine learning techniques incorporating avoidance of obstacles, moving towards a general direction or towards a region of interest, displaying multi-robot behaviours like aggregation, etc. Contrarily, there is very little literature on the use of machine learning techniques for getting the robot to a specific goal configuration amidst obstacles. The problem is challenging from a machine learning perspective due to the maintenance of a balance between the obstacle avoidance behaviour and goal-seeking behaviour. Emphasis on obstacle avoidance can lead the robot reasonably far away from the goal, making the overall task difficult, while an emphasis on goal seeking can risk the robot being too close to the obstacles. Often the robot needs to select a proper homotopic class of getting to the goal and avoid being trapped amidst obstacles. Reactive approaches like Artificial Potential Field, Vector Field Histogram, Velocity Obstacle, etc. have a limited model complexity and simple navigation rules which may be restrictive to make the best possible navigation function. Machine Learning techniques have the advantage that the model complexity can be based on the data so as to potentially learn the best navigation function.

The paper uses both reinforcement learning and supervised learning for learning the navigation function. Reinforcement learning works by maximizing the rewards by taking suitable actions as per the situation and thus learning a policy. Since the machine automatically generates the data, human intervention is not needed. However, the design of a reward function scheme can be critical to the performance of an algorithm. Similarly, since the machine learns by a mere optimization of objectives, the machine can learn a trajectory that does not imitate humans. The humans while walking do not necessarily optimize any objective. On the contrary, supervised learning requires a labeled data set by a human, which can only be generated by asking people to volunteer and provide navigation data. This paper uses a simulation tool for the same. Since humans generate the data, the trajectories are socially acceptable and human-alike. However, the robots being socially different have a little advantage in imitating humans. The paper experimentally analyses the two machine learning approaches and the approach with better results is tested on a hardware robot as well.

The bot correctly moves to avoid all the obstacles that come across its path without any prior knowledge of the path and any learning procedure but with the help of the neural network. Hence, the learning algorithm used in software implementation was proven efficient.

Otte [3] did an extensive survey of various algorithms and techniques for path planning of robotic systems. General assumptions made during path planning include localization, goal knowledge, and accuracy. The objective was always a specified task like avoiding obstacles, taking the shortest path and reaching specified locations. Supervised learning and reinforcement learning techniques are being extensively used for the problem. Lamini et al. [4] used Q-learning with fuzzy logic in a software implementation of pathfinding problem for a specified goal. Reinforcement Learning with Q-learning, implemented using a Holonomic Multi-Agent System(H-MAS) and Q-Embedded Table (QET) improved the quality of action taken, leading to an optimal path towards the goal. Yun's [5] work shows predictable behaviour in driving and pathfinding. Most humans at any traffic signal stay towards their right if they do not want to go straight ahead. Such knowledge can make autonomous driving a bit easier as the robotic system doesn't have to worry much about a car hitting it from right in traffic crossings. Most of the related works deal with rather small subgoals of autonomous driving. Various research works [6-8] proposed supervised learning to be well efficient for the task. Supervision avoids all unnecessary thinking on the part of the robot but makes it less intelligent in never seen before scenarios. Thrun et al. [9] worked on a grid-based indoor model. Kala et al. hand tuned a fuzzy system for navigation of a single [10] and multiple robots [11], while the fuzzy system was also locally optimized.

Saha et al. [12] proposed an improved reward estimation that uses Inverse Reinforcement Learning. Modifying a simple reinforcement model to an inverse reinforcement model might increase the accuracy to quite an extent. Semi-Markov Decision Trees with Distance-Minimisation Inverse Reinforcement Learning have proven to be helpful in pathfinding as they use what in simple words would be hidden reward functions. Barto et al. [13] compared the two learning approaches used in this paper. Supervised learning makes a robot well capable of doing tasks but fails to provide it with social ethics and emotions. Reinforcement learning has proved to bring human-like behaviour to robots. The robots could communicate with humans and other robots effectively. Hence, social awareness is a requisite and Tung and Ngo [14] proposed ways to achieve it. They proposed a model called Proactive Social Motion Model that considers not only human states relative to the robot but also social interactive information about humans and human group interactions. They use unsupervised learning to reach their objective. Although unsupervised learning can be a good option, it is still not a well-explored approach. Their work involved only one dynamic obstacle(human). The key point to be noted is that most of the research done in this field is to move the bot in a physical environment such that it moves aimlessly avoiding all static as well as dynamic obstacles. However, this project is significant and unique as it adds a specific goal or the destination such that the bot moves in the environment and stops after reaching the goal or its destination avoiding static as well as dynamic obstacles it encounters in its path.

The novelty of the approach lies in the fact as most of the background and related work is focused on the given agent avoiding obstacles in its path and carrying out its routine task in a given bound area. Our approach is multifaceted as, along collision avoidance, the agent reaching its goal is also a primary requirement for a trial to be deemed successful. In addition to that, we explore two completely different approaches to achieve the objective, and finally implement the more feasible approach to hardware after comparing the results of the approaches.

II. Methodology

A. Supervised Learning

The methodology used is to first generate data for the model and then to use it to train the model. Generation of data on the real robot can be expensive in terms of availability of the robot and generation of diverse maps, and hence simulations are used for better training. A simulator in the form of a game is provided to the user. The user plays the game with given map several times with maximum success rate. Several maps are used to train the car. The data is generated from sensor readings. Normalized data is used as input to the neural network. The data is divided into training set and testing dataset. The training dataset helps train the model and later the test data validates the model's weights. Later, a new map is presented, on which the car predicts its next move (in terms of velocity and angular velocity) with the help of its sensor readings.

The inputs for the supervised learning are: (i) The displacement from goal, (ii) Angular deviation, angle difference of goal and the robot, (iii) Sensor readings from 6 proximity sensors in six directions. The output is the angular velocity and the linear velocity for the robot. The supervised learning approach works by taking a set of inputs along with the corresponding correct outputs and the robot then learns by comparing its actual output with the correct output as stored in the dataset and moves accordingly, while minimising errors.

B. Reinforcement Learning:

Again, simulations are used for training the initial Reinforcement Learning model to minimize the costs incurred in the implementations on the real robot. In usual Reinforcement Learning scenarios that involve game implementations, the entire screen matrix is taken into account, so that the state of the game is the entire pixel or screen matrix and the learning model is later trained on that. However, since this paper involves hardware implementation and real-world physical environment out of the software domain, it is not possible to maintain a pixel matrix of the real world's physical space with the given tools. Hence, the input data comprises of sensor readings built into the sprite of the game. The robot in the simulation has three sensors. These sensors fan out and sweep across the play area to detect walls, obstacles, and goals. Their readings collectively determine the reward. Fig. 1 shows the procedure followed by reinforcement learning. Fig. 2 shows the middle sensor detecting a nearby object.

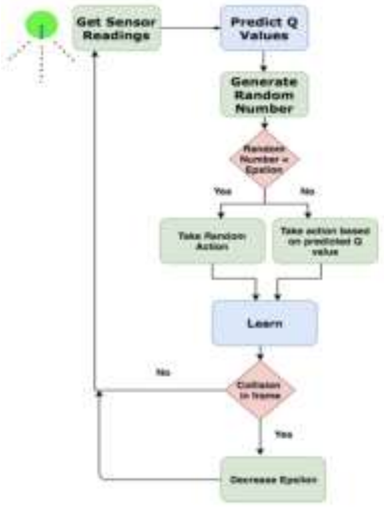

Figure 1. The procedure followed for reinforcement learning

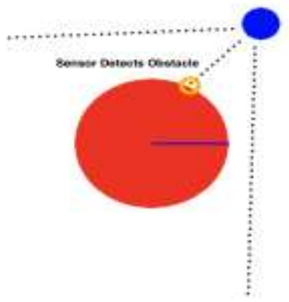

Figure 2. Obstacle detection by the sensor

The methodology for taking in the readings and converting them into a reward is as follows: At every frame of the game, each of the virtual sensors measures the proximity reading. The sensors have a maximum threshold. Higher reading implies that the obstacle is further away from the sprite and is better in terms of avoiding imminent collisions with obstacles. An exception is an obstacle beyond the goal in the direction of the goal. In such a case the obstacle should not affect the robot, and this is explicitly handled. The final reward at a given frame is a function of distance from an obstacle (D_{obs}), distance from the goal (D_{goal}) and the angle difference (A_{diff}) between the goal and the obstacle. Equation (1) is the governing equation for the experiment carried out.

$$\text{Reward} = -\alpha + D_{obs}/\beta + \psi * D_{goal} - A_{diff} \qquad (1)$$

α, β and ψ are constants, which may be fine-tuned to offer optimal rewards in frames. Q-learning is used to learn the policy to maximize the rewards.

III. Results

A. Supervised Learning

In case of supervised learning, the final result is that the robot reaches the required destination avoiding all obstacles in the map. In Fig. 3(a) the red curve shows the velocity during training of the robot. The blue curve in the graph shows variation in predicted velocity as the robot proceeds towards goal. As during training, the driver is human, and speed varies a lot due to careful driving. The predicted velocity curve is smooth depicting the model's ability to approximate it to human driving skills. The root mean squared error in velocity was found to be 6 pixels/time-stamp. In Fig. 3(b), the red curve shows the angular velocity of the car during training. The blue curve shows the variation of angular velocity as predicted during the cars self-driven journey towards the goal. The curve here is smooth again. The repressor well fits the training set values. The root mean squared error in angular velocity was found to be 0.37 units. The units are specific to the tool developed and relate to real values by constants.

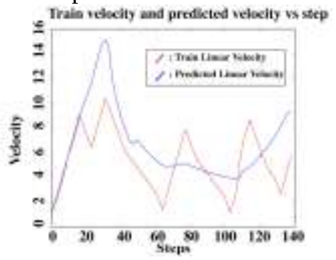

Figure 3(a). Train linear velocity and predicted linear velocity vs. time-steps

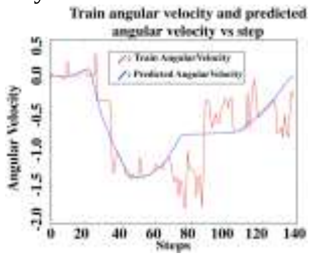

Figure 3(b). Train angular velocity and predicted angular velocity vs. time-steps

In Fig, 4(a), the motion of the robot for testing was as follows: The car moves forward, takes a sharp right turn, proceeds toward the goal, and when it is very close to the goal (car is to the left of goal), car takes a left turn and dashes into the goal. The predicted velocity varies smoothly with the propagation of time and is very close to the nature of test set's curve. In Fig. 4(b), the predicted angular velocity, again forms a smooth curve. The angular velocity during test phase was discrete, and a sudden change in angular velocity was recorded. The neural model, none the less shows gradual change, thus making the drive smooth.

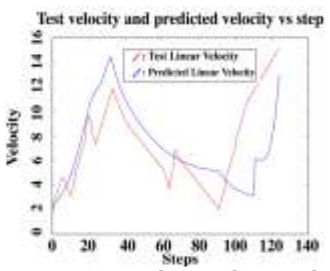

Figure 4(a). Test linear velocity and predicted linear velocity vs. time-steps

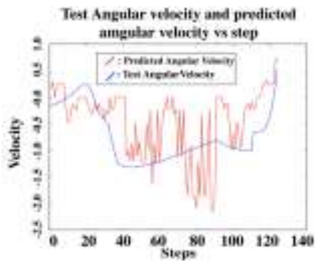

Figure 4(b). Test angular velocity and predicted angular velocity vs. time-steps

B. Reinforcement Learning

In the case of reinforcement learning, the robot reaches the goal while avoiding the static and dynamic obstacles. The graph in Fig, 5(a) shows how with increasing the training frames or iterations, the distance the robot takes to reach the goal decreases gradually. The unit distance along the axis is the average of 10 readings per iteration. The graph starts at 2500 iterations. As the number of iterations increases the distance the robot travels to reach the goal decreases gradually. The irregularities and sudden peaks represent a typical scenario due to the stochastic nature of the training. The goal may be in a corner surrounded by obstacles and become unreachable, in which case, even with sufficient training it would be difficult for the robot to reach the goal, and hence the distance may exceed the expected amount by a large margin. However, the gradual decrease indicates the decreasing distance required to reach the goal which is consistent with the expectations as the network learns more.

The graph in Fig. 5(b) is distinct from others and showcases the collision avoidance success, i.e., how far can the robot move without a single collision with any of the obstacles. As soon as the collision occurs, the iterator moves on to the next model with an experience of one more frame thereby generating a continuous curve. As evident from the graph, the distance keeps on increasing with more experience which implies, that as the neural network gets trained, the collision avoidance capability of the sprite increases gradually at first, and then drastically at later stages.

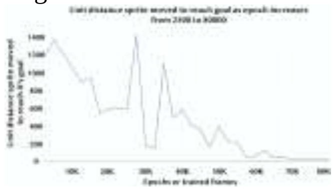
Figure 5(a). Unit distance robot moved to reach goal vs. training iterations

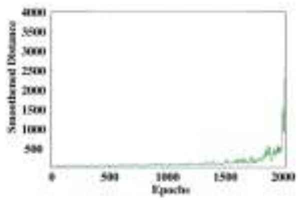

Figure 5(b). Smoothened distance vs. epochs

C. Comparison

Figure 6 shows a distance-based comparison between supervised learning and reinforcement learning approach. After the network is trained, the game runs several thousand times, and for each successful iteration, the total distance travelled to reach the goal is recorded. The means of those distances along with the error margins (given by the standard deviation) have been plotted in Fig 6.8. Evidently, the supervised learning approach made the sprite travel a larger distance to reach its goal. The comparison was carried out for same maps with only the location of dynamic obstacles serving as a variating factor.

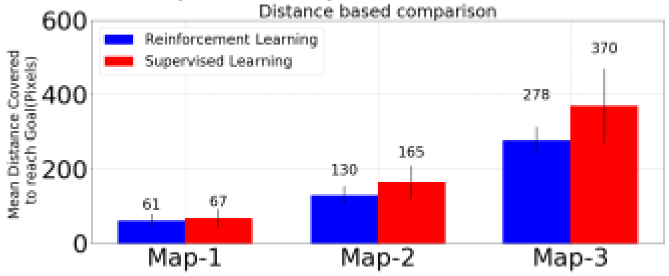

Figure 6. Distance-based comparison of supervised and reinforcement learning approaches

IV. Hardware Implementation:

Fig. 7 shows the hardware used for the implementation which is a P150 Amigo Bot. The bot has similar sensors as in the software implementation. However, the scale for the readings generated by the hardware's sensor is different from the software version. The readings of each of the sensors in the software implementation were in the range [1,20], however, in hardware sensors the maximum reading was 4820. Training on actual hardware was not feasible due to multiple collisions it would encounter, and the sheer time it would take to train. Another problem with transitioning from the software implementation to hardware was the sweeping nature of getting readings in hardware by sensors. The sensors returned a combined reading of the entire sweep in a given range, and not individualised readings mapped to those sensors when using the python wrapper of the Aria library. Thus, the readings of the hardware were normalised, and the average of left and right sweeps was taken to generate the third reading to make them fit in the same format as the software implementation.

Figure 7. P150 Amigo Bot used to test hardware implementation

These normalised readings constituted the state of the hardware and neural network successfully predicted the Q values. The first step of the implementation on the hardware involved running the game on the accompanying hardware simulator. Hence, the model was first tested on MobileSim simulator. RosAria node was used for communicating with the simulator. After successful simulations, the Amigo Bot was stimulated automatically using the neural network. Fig. 8 shows the simulator working on a map loaded on MobileSim. In the hardware, the coordinates of the goal were given relative to robot.

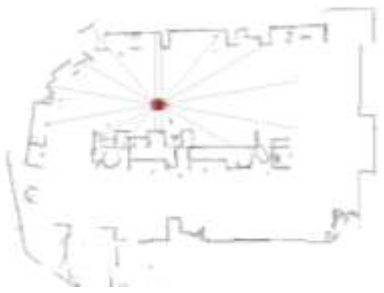
Figure 8. Screenshot of simulating reinforcement approach on MobileSim simulator

The reinforcement learning approach of the problem was implemented on the hardware due to its better support for dynamic environments and stochastic obstacles in terms of distance travelled to reach the goal. The neural network and models developed for software implementation of the problem were adapted to be used on hardware (Amigo Bot) which resulted in the bot actively avoiding obstacles and eventually reaching the goal.

V. Conclusion

The algorithm expertly navigated the robot to move in an environment, avoiding obstacles and reaching a specific goal using supervised and reinforcement learning. The results were also shown for the Amigobot robot in a physical environment. The Amigo bot moved in the environment avoiding all the obstacles and eventually reached the goal. Reinforcement learning was found to be better in terms of optimality, while the results were socially acceptable for humans operating in the environment and being a reactive technique, completeness is not guaranteed, for which fusion with a deliberative algorithm may be tried in the future. Other future work may involve testing the algorithm in an uncontrolled real-world environment. Individual cases, where the bot fails to recover post a collision when it may be surrounded by obstacles while at a corner, or in a complicated maze of obstacles, where the negative rewards from the obstacle overpower those of the goal and the robot consequently fails to move in the required direction of the goal can be tackled. While most of the existing work focuses on avoiding obstacles in a given region, using the proposed algorithm, the problem of avoiding the obstacles while simultaneously reaching the goal was tackled, which can have multiple applications in real-world scenarios like robots delivering goods or enriching the feature set of existing robots which do routine collision avoidance while operating, thereby allowing them to also move independently among various destinations.

References

[1] S. M. LaValle, *Planning Algorithms*, Cambridge University Press, 2006

[2] R. Tiwari, A. Shukla, R. Kala, *Intelligent Planning for Mobile Robotics: Algorithmic Approaches*, IGI Global Publishers, Hershey, PA, 2013.

[3] M. W. Otte, "A survey of machine learning approaches to robotic path-planning”, Technical Report, *Department of Computer Science, University of Colorado at Boulder,* 2009.

[4] C. Lamini, Y. Fathi, S. Benhlima, "H-MAS Architecture and Reinforcement Learning method for autonomous robot path planning," In *Proceedings of he 2017 Intelligent Systems and Computer Vision (ISCV)*. IEEE, Fez, 2017, pp. 1-7.

[5] S. Yun, J. Choi, Y. Yoo, K. Yun, J. Y. Choi, "Action-Decision Networks for Visual Tracking with Deep Reinforcement Learning," In *Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition*, Honolulu, HI, 2017, pp. 1349-1358.

[6] N. Motamedidehkordi, S. Amini, S. Hoffmann, F. Busch, M. R. Fitriyanti, "Modeling tactical lane-change behavior for automated vehicles: A supervised machine learning approach," In *Proceedings of the 2017 5th IEEE International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS)*, Naples, 2017, pp. 268-273.

[7] X. Da, R. Hartley and J. W. Grizzle, "Supervised learning for stabilizing underactuated bipedal robot locomotion, with outdoor experiments on the wave field," In *Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA)*, Singapore, 2017, pp. 3476-3483.

[8] D. Shukla, Ö. Erkent and J. Piater, "Supervised Learning of Gesture-Action Associations for Human-Robot Collaboration," In *Proceedings of the 2017 12th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2017)*, Washington, DC, 2017, pp. 778-783.

[9] S. Thrun, A. Bücken, *Learning Maps for Indoor Mobile Robot Navigation*, Technical Report No. CMU-CS-96-121, Department of Computer Science, Carnegie Mellon University, Pittsburgh PA, 1996.

[10] R. Kala, A. Shukla, R. Tiwari, “Fusion of probabilistic A* algorithm and fuzzy inference system for robotic path planning,” *Artificial Intelligence Review*, vol. 33, no. 4, pp. 275-306, 2010.

[11] R. Kala, “Navigating Multiple Mobile Robots without Direct Communication,” *International Journal of Intelligent Systems*, vol. 29, no. 8, pp. 767–786, 2014.

[12] O. Saha, P. Dasgupta, "Improved reward estimation for efficient robot navigation using inverse reinforcement learning," In *Proceedings of the 2017 NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*, pp. 245-252. IEEE, 2017.

[13] A. G. Barto, T. G. Dietterich, “Reinforcement learning and its relationship to supervised learning”. *In Handbook of learning and approximate dynamic programming, IEEE*, pp.47-60, 2004.

[14] X.T. Truong, T.D. Ngo, “Toward socially aware robot navigation in dynamic and crowded environments: A proactive social motion model.” *IEEE Transactions on Automation Science and Engineering*, vol. 14, no. 4, pp. 1743-1760, 2017.