ARTIFICIAL NEURAL NETWORK BASED LOCAL MOTION PLANNING OF A WHEELED MOBILE ROBOT

István ENGEDY
Advisor: Gábor HORVÁTH

I. Introduction

The navigation system of a mobile robot must handle numerous well separable subtasks for proper operation. These tasks are the localization, motion planning and mapping.

In the motion planning problem the tasks are to determine the path from one point to another one (called target), and to control the movement, to make the robot follow the previously computed path. Thus motion planning is often discussed as two separate subtasks, the path planning and the movement.

A Path Planning

Path planning is the procedure, which plans the path from the current location to the end point, of which the robot will have to go through. Obviously, the map of the intermediate area is needed to be known, so the obstacles and unreachable areas could be avoided. There are two basic approaches found in the literature, topological landmark based maps, and metric maps [1].

In case of metric maps usually the discretization of the area is needed, to be able to use graph-based algorithms in path planning [1]. Algorithms, that are working on finite graphs, are for example the well-known Dijkstra, Bellman-Ford, or A* algorithms [2].

There are also soft-computing methods for path planning on metric maps [3]. They usually utilize fuzzy logic controllers, artificial neural networks, and reinforcement learning methods. They are used for a couple of reasons, like they are computationally efficient, they do not require an exact mathematical description of the problem, they better tolerate noisy input data, and they can adapt to changing environments.

B Movement

The movement is the procedure of keeping the robot on the path. The motion could be complex [4], because the movement of the robot must meet various physical constraints. The most common solution is to use various controllers, such as P, PD or PID controllers.

The movement control could be carried out using soft computing methods, including ANN-s. Some of these methods are able to simultaneously perform the path planning and the movement problems too [2].

II. Former Results

In our former work [5], we have shown a solution of the mobile robot motion planning problem. The robot was a nonholonomic car-like robot. The target of the robot was a predefined position and orientation in the working area. There were also static and moving obstacles that the robot had to have to avoid.

We have presented an ANN based mobile robot controller for obstacle avoidance. The ANN was trained with the classical backpropagation through time (BPTT) training approach. We have shown that it can be extended using regularization, where through the regularization term additional constraint can be taken into consideration, and how this is used for avoiding static or moving obstacles in a navigation problem. It was also shown that real time online training is also possible.

Although the online training is a good solution to adapt to changing environment, it has also a couple of drawbacks. The most serious one is the increased need of computational power. Another problem with it is not capable of remembering the already learned movements, if a new movement in a new situation is trained.

There was another problem in our former work: the description of the configuration space (coordinates of the robot and target) were relative to an absolute coordinate system. This way the ANN had to learn the same movements in different positions.

III. Modified Navigation Method

In the modified navigation method the controller ANN is still trained with the BPTT method, the output is the same as it was before, but the input is slightly different. Now the state of the robot is described with the relative position and orientation of the target to the robot. The description of the state is also changed to a polar coordinate system description. The input of the ANN is extended with a polar occupancy grid map of the obstacles in front of the robot.

A The BPTT method and obstacle avoidance

The BPTT could be used for robot navigation, as D. Nguyen and B. Widrow showed it [6], "Fig. 1.a" where the controller of the robot is an artificial neural network and the whole control loop is opened during the training. In this situation not only ANNs take place in the chain, but numerous models of the system, which are not needed to be trained, but the error must be propagated back through them.

To make the BPTT method able to take account of the obstacles, a potential function can be defined based on the location and the size of the obstacles. To actually repel the robot away from the obstacles, this function must be used to extend the cost function of the Delta-rule. The goal of the weight modification is not only to minimize the error at the end of the simulation chain, but also to minimize the potential of the path, to get the robot the farthest from the obstacles.

The regularization term could be added to the error during the backpropagation at the appropriate places through the simulation chain, and the weights of the ANN can be modified using the usual way as it can be seen in "Fig. 1.b".

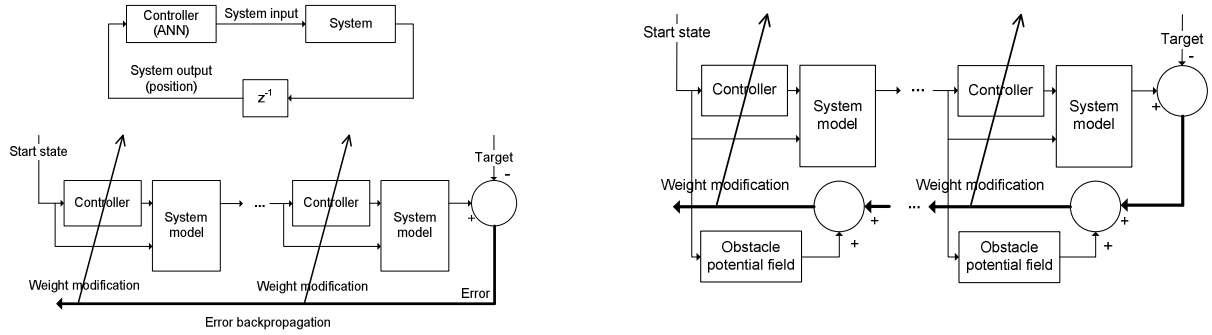

Figure 1. a) (left) BPTT in use of training in a control loop, b) (right) Adding the obstacle potential to the error

B Robot state and obstacle description

The state of the robot is described with its relative position to the target in a polar coordinate system. As it is shown in "Fig. 2.a" this state description contains the following values: α, the angle between the direction the robot is facing and the direction of the line pointing to the target from the robot, d, the distance of the robot and the target, and β, the angle between the direction the robot is facing and the target orientation.

Describing the robot state this way, the target of the training, where we wish the neural network controller to guide the robot, is the origo of the coordinate system. So we want to train the neural

network that at the end of the unfolded chain, the state of the robot will be zero in all three values. This way numerically the state is the error we want to minimize.

The obstacles, similar to the target, are described in the polar coordinate system of the robot. Each obstacle has two values, its distance to the robot (*b*), and the angle between the direction of the robot and the line pointing from the robot to the obstacle (β). In "Fig. 2.b" it is shown how these coordinates and the distance and angle between the target and the obstacle can be calculated into each other, which is needed in the backpropagation part of the BPTT training.

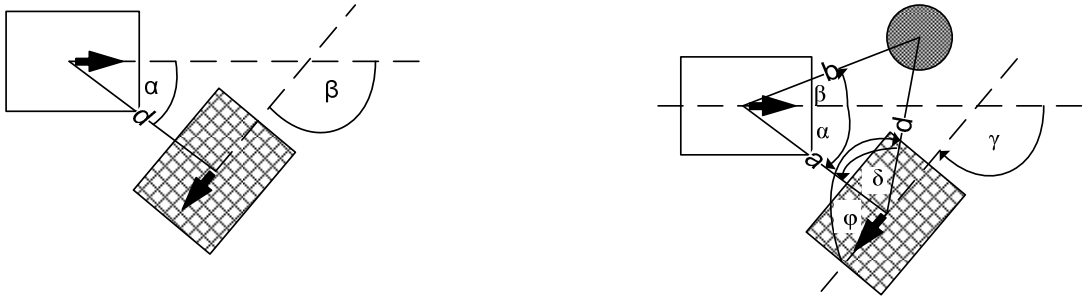

Figure 2. a) (left) State of the robot (white rectangle) in polar coordinate system. b) (right) The obstacle (dotted circle) and its relative position to the robot and the target (checked rectangle)

C The extension of the neural network input

To solve the problems of the online training we have proposed the following solution. The different movement patterns in the different situations could be stored in the controller ANN. To do that, the different situations must be differentiated from each other at the input of the ANN. This way an offline trained ANN could be used as a controller.

To make the ANN aware of the obstacles, we have extended its input with a polar grid "Fig. 3.", where each cell in the grid is an input of the ANN. If there is an obstacle in the grid cell, its value will be 1, if the cell is empty, the value will be 0. The grid is only in front of the robot, since it is moving only forward, we do not need to know what is behind the robot. The movement only depends on the direction and distance of the target, and the obstacles in front of the robot.

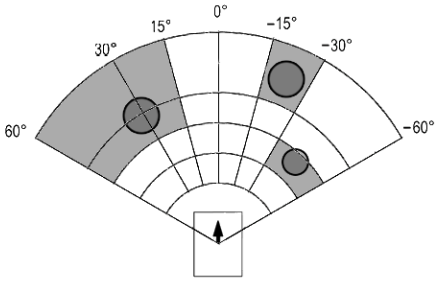

Figure 3. The polar grid in front of the robot. The gray cells are the occupied cells, in which there is at least 20% of an obstacle, the dark circles are the obstacles, and the rectangle is the robot.

With the use of this polar grid, the ANN is able to learn and remember the movement patterns for each situation in which it was trained with the regularized BPTT algorithm. This way it can be used in offline training mode too.

The offline training needs a set of problems that is going to be solved. We used randomly generated situations for the robot. Since the robot state is described with the relative distance and angle of the target, these values can be changed randomly. The angles could take any value from [-π; π), and we restricted the robot–target distance to the [200 mm; 3000 mm] interval. For comparison, the shaft length of the robot was 120 mm. We used 1–3 obstacles, which were placed randomly between the robot and the target. For each situation, we trained the ANN only for one BPTT step, to

prevent overfitting to any situation. With these randomly selected situations and one BPTT training step per situation, we achieved, that the network was trained to solve these kinds of situations in general.

IV. Simulation Results

The method was tested in simulation only. The former simulator application was modified with the previously mentioned changes. The tests showed that the polar coordinate description and robot centric approach is more suitable for this problem. The convergence of the training was considerably faster than it was with the Descartes-coordinate system.

We have also tested its capabilities with offline training in a random generated situation. The robot was able to get to the target without collision. Since it was not aware of all of the obstacles and their exact location, its path was not optimal by any means, but we haven't expected that.

V. Conclusion

In this paper we have shown that the classical BPTT training approach can be extended using regularization, to solve the motion planning problem among obstacles, and this could be done effectively, if the state of the robot and its environment is described using a polar coordinate system. It was also shown that this is also possible using offline training, by extending the inputs of the neural network with a polar grid. This offline trained ANN is able to navigate the robot on a collision free course towards the target, however there are some limitations on its capabilities.

It is not guaranteed, that the path of the robot will be optimal by any means, or that the robot will reach the target, and it won't get stuck in a local minima. With this offline training only a little, local part of the environment is taken into account, on which it is not possible to compute a globally optimal path. This method is useful when the navigation system doesn't have any information on the rest of the environment, only the local surroundings of the robot.

If a bigger part or the whole map of the environment is known, other path planning algorithms can be used to compute a globally optimal path. Using our method with waypoints on the path, the robot is able to follow it, without colliding with any static or moving obstacle. This way path planning algorithms that are incapable of taking the constraints of the robot or dynamic changes in the environment into account, could be also used for mobile robot motion planning.

Acknowledgment

The authors gratefully acknowledge the support of the Hungarian Fund for Scientific Research (OTKA), Grant #73496

This work is connected to the scientific program of the "Development of quality-oriented and cooperative R+D+I strategy and functional model at BME" project. This project is supported by the New Hungary Development Plan (Project ID: TÁMOP-4.2.1/B-09/1/KMR-2010-0002).

References

[1] Sebastian Thrun, Wolfram Burgard, Dieter Fox: Probabilistic Robotics, MIT Press 2005

[2] Stuart Russel, Peter Norwig: Artificial Intelligence A Modern Approach, Prentice Hall 2003

[3] Pratihar, D.K.: Algorithmic and soft computing approaches to robot motion planning. Mach. Intell. Robot Control 5,1-16 (2003)

[4] J.-C. Latombe, Robot Motion Planning. Boston , MA : Kluwer, 1991.

[5] I. Engedy and G. Horváth, "Artificial Neural Network based Mobile Robot Navigation," in Proc. of the IEEE International Symposium on Intelligent Signal Processing, pp. 241-246, Budapest, Hungary, Aug. 2009

[6] D. Nguyen and B. Widrow, ``The Truck Backer-Upper: An Example of Self-Learning in Neural Networks," Proceedings of the International Joint Conference on Neural Networks, 2:357-362, 1989.