

AN EFFICIENT OBSTACLE AVOIDANCE SCHEME IN MOBILE ROBOT PATH PLANNING USING POLYNOMIAL NEURAL NETWORKS

Farid Ahmed and C. L. Philip Chen
Wright State University
Computer Science & Engineering Department
Dayton, OH 45435.

Abstract

Application of Polynomial Neural Networks(PNN) in mobile robot path planning with an obstacle avoidance scheme is proposed. Given an environment and a desired goal location (position and orientation), PNN's are built from some selected starting locations to reach this goal. These PNNs comprise the memory of our model. An efficient associative retrieval technique is then applied to make the robot follow a minimal cost polynomial path. In the movement, when it faces an obstacle, the robot uses a contour finding algorithm to get away from the obstacle. The major advantage of using the PNNs is its interpolating capability with a moderate size of data space. Also no preprocessing of the range data is necessary.

1 Introduction

Finding a minimal cost and collision free path between a start point and a goal point in a given environment has long been an important research area in mobile robot navigation. Two of the pioneering works in the field of path planning for mobile robot in the known and unknown dynamic environment are configuration space approach [1] and the potential field approach [2].

This work is an extension of our previous work [3] using the excellent interpolation property of the Group Method of Data Handling (GMDH) [4, 5] technique and the feature-based mapping capability of the PNN, which proved to be an effective type of neural network in robotics computation and aircraft combat application [6, 7]. In the present work,

these two paradigms, namely, PNN and GMDH are used in a different way. Here we build a network of polynomial paths first from the environment data. Then we address the problem of collision avoidance as an associative retrieval problem. With the selected polynomial, the robot follows an estimated path until it hits any obstacle. A contour-finding algorithm is then utilized to move the robot away from the obstacle or direct the robot along the contour of the obstacle until it reaches another learned polynomial.

2 The PNN

We can treat the polynomial neural network as an equivalent feedforward network as depicted in figure 1.

The input variables comprise the input layer. The hidden layer consists of some combination of these input variables. These nonlinear elements are found by the GMDH technique. The hidden units are thus not restricted to only linear summing elements. The weight matrix from the hidden layer to the output layer is also learned by the GMDH. The output unit works as a conventional processing element. Figure 1 depicts the case of a very simple network having only one output, y and two input variables, x_1 and x_2 .

Figure 2 shows how GMDH technique is utilized to get a network of the "survival of the fittest" processing elements. Note that, each processing element(PE) in each layer has exactly two inputs from the previous layer. Thus, if m is the number of PEs in layer l , then the number of constructed PEs in layer $l+1$ is $n = \binom{m}{2}$. From these n PEs k best uni-

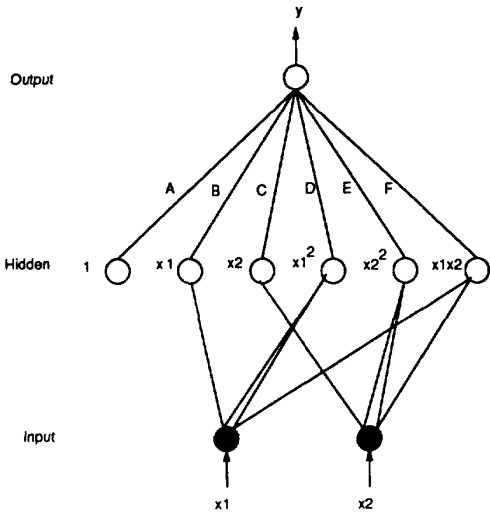


Figure 1: The Feedforward Equivalent Network of PNN

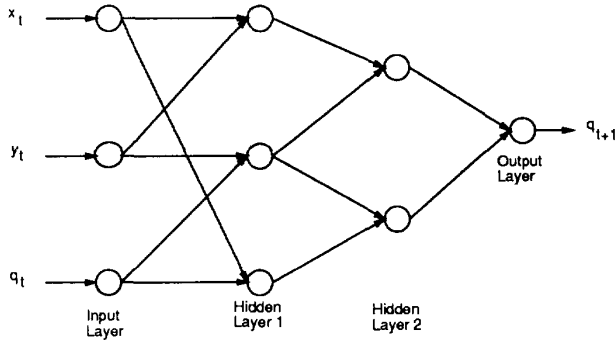


Figure 2: Use of GMDH technique to get the next direction from the present robot position

are chosen, according to a selection criterion to construct the next layer. This process is continued until we get a single PE in the final layer. Now, what each processing element does is that it produces the value of the dependent variable in a polynomial representation of its inputs. In our model, we use second degree of polynomials. Thus, each polynomial for each element has six coefficients associated with it.

The network constructed from the GMDH algorithm is an adaptively synthesized supervised learning model. At the final stage of the network we get a representation of the output as a polynomial function of the inputs. A detailed description of this algorithm is found in [5, 6].

3 The Working Model

At first, we designate a desired goal location in the environment. Then, several starting locations are chosen. Now using the data set and the GMDH technique, a number of networks of polynomials are built. These PNNs are built to avoid the obstacles of the known environment. At the output of each generated PNN, we get a polynomial in terms of the input variables. Each such polynomial represents the mapping of the trajectory between a starting location and the goal location.

Our input data set is a sequence of the current locations of the mobile robot (x_t, y_t, θ_t) represented in the cartesian coordinate. The output is the orientation of the robot in the next step (θ_{t+1}) . That means the robot will make one step toward this direction. The step size is a fixed quantity, s . The following equations give the relationships.

$$\theta_{t+1} = A + B_1x_t + B_2y_t + B_3\theta_t + C_1x_t^2 + C_2y_t^2 + C_3\theta_t^2 + D_1\theta_t x_t + D_2\theta_t y_t + D_3x_t y_t + E\theta_t x_t y_t$$

$$x_{t+1} = x_t + s \cos\theta_{t+1}$$

$$y_{t+1} = y_t + s \sin\theta_{t+1}$$

where, A, B_1, \dots, E are the coefficients of the polynomial, θ_{t+1}, x_{t+1} and y_{t+1} are the next step heading direction, x and y coordinate respectively.

Now, we pose the question, how to reach the goal point from any initial point in the environment. This point may lie in the training set or it may be a completely new location unknown to the trained network. At this stage, we approach the problem as an associative retrieval problem, where the coefficients of the learned polynomials form the memory matrix. The bottom line here is to find the minimal cost polynomial, where the cost function is defined as the Euclidean distance measure in terms of position and orientation from the initial location to all the polynomials.

In our earlier work [3] we proposed two approaches to move the robot to this closest polynomial. Here we use the faster scheme, where we make a heuristic polynomial estimated from the starting location using the knowledge of heading direction and the goal location. The robot moves along the estimated polynomial until it intersects with the the

closest polynomial. In its course when it hits an obstacle, then it repeats this process until the goal location can then be reached.

4 Simulation Results

In order that the model does not become overspecialized, the input data set is subdivided into a training set and a checking set. The GMDH technique is very sensitive to the splitting of data into training and testing set [5]. Random selection and the Duplex are some of the available methods used for the data splitting. Here we have used some heuristic in addition to the Duplex method. This procedure greatly reduced the overfitting problem. The heuristic is used to remove some of the outliers from the data set. In the simulation we built 6 polynomials for the six obstacle free paths. The following table shows how closely the built polynomials represent the data set of the environment.

Table 1 : Learned Polynomials

Polynomial	Variance	Percent Error
1	.017	.429%
2	.011	3.12%
3	.002	.029%
4	.011	.082%
5	.012	1.222%
6	.005	.09%

The percent error in the above table is with respect to the training data.

Let us explain the network with a very simple example polynomial. It was found that polynomial 1 was generated after only one GMDH layer. That means the next step heading direction depends only upon two input variables. For this case these are x_t and θ_t .

$$\theta_{t+1} = A+B*U+C*V+D*U*U+E*V*V+F*U*V$$

Where,

$$A = 0.8286E+00, B = -0.5280E+01, \\ C = 0.1653E+01, D = 0.2800E+01, \\ E = -0.6430E+00, F = 0.2392E+01.$$

$$U = y_t \quad V = \theta_t$$

5 Conclusion

Results show that the built polynomials are very accurate representative of the environment data. This resulted in a very precise obstacle avoidance scheme. The proposed scheme is equally applicable to both the known environment and the environment with some dynamic change. As mentioned earlier that no preprocessing of the range data is required in this approach.

In future, real time obstacle avoidance using this simulator is to be implemented with our sonar sensor guided mobile robot.

6 References

1. T. Lozano-Perez and M. A. Wesley, "An algorithm for planning collision free paths among polyhedral obstacles," *Commun. ACM*, 22(10), October 1979, pp 560-570.
2. O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *Int. J. of Robotics Res.* 5(1), 1986, pp 90-98.
3. C. L. Philip Chen, Farid Ahmed, "Polynomial Neural Networks-based mobile robot path planning," *Proc. SPIE*, Apr 1993, Orlando, FA.
4. A. G. Ivakhnenko, "The Group Method of Data Handling in Prediction Problems," *Soviet Automatic Control*, vol. 9, No. 6, 1976, pp 21-30.
5. Stanley J. Farlow, "Self-Organizing Methods in Modeling," Marcel Dekker inc.
6. C. L. Philip Chen and A. D. McAulay, "Polynomial neural network for robot forward and inverse kinematics learning computations," *Proc. of SPIE Applications of Artificial Intelligence IX*, 1991, pp. 394-405.
7. A. D. McAulay et al, "Polynomial neural networks for Airborne Applications," *Proc. of the IEEE NAECON Conf.*, 1989, pp 682-687.