

Polynomial-Neural-Networks-Based Mobile Robot Path Planning

C. L. Philip Chen and Farid Ahmed

Wright State University

Computer Science & Engineering Department

Dayton, OH 45435.

1. ABSTRACT

A Polynomial-Neural-Network-based (PNN-based) path planning with an obstacle avoidance scheme is proposed for mobile robot navigation. The PNN is a feature-based mapping neural network which can be successfully trained to interpolate an unknown function by observing few samples. In this work, a very useful method of data analysis technique called the Group Method of Data Handling (GMDH) [1] is used to build the PNN. The built PNNs are used for the path planning of a sonar sensor guided mobile robot. The major advantage of using the PNNs is to efficiently use the environment data and to reduce the computational complexity. Also, in this approach, no preprocessing of range data is required.

2. INTRODUCTION

Path planning in mobile robot navigation is associated with the problem of finding a collision free and minimal cost path between a start point and a goal point in a given environment. The layout of the environment may be known a-priori or it may be unknown. A significant amount of work has been done in the field of path planning for mobile robot in the known and unknown dynamic environment. Two major approaches of them are the configuration space approach pioneered by Lozano-Perez [2] and the potential field approach first proposed by Khatib [3]. In the first method an environment is typically modeled by a set of polygons and the robot is considered as a point object. The obstacles are then enlarged accordingly. Now a minimum distance path is calculated that avoids the enlarged obstacles. In the potential field approach a globally computable potential

field-like metric is associated with each of the obstacles. A path is chosen which corresponds to a valley in the global graphical representation of that metric.

The motivation to the present work is the excellent interpolation property of the GMDH 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 [4, 5]. In the present work, these two paradigms, namely, PNN and GMDH are used in a different way. We addressed two problems here. First one is to develop the PNN that will allow the robot to navigate to the goal location from any initial location. In the second problem, collision avoidance using the developed PNNs is investigated. 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 a polynomial. It is to be noted that the whole approach is equally applicable to both known environment and an environment with some dynamic changes. Section 2 illustrates the model in details. Simulation results are furnished in section 3 which is then followed by discussion and conclusion in section 4.

2. THE MODEL

2.1 The PNN

The polynomial neural network can be treated as a feedforward network as depicted in figure 1.

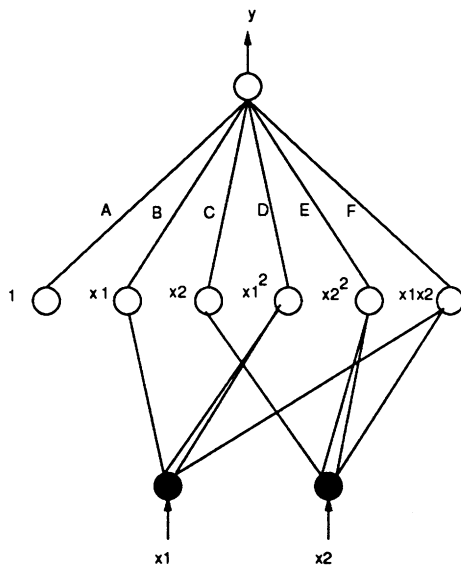


Figure 1: The Feedforward Equivalent Network of PNN

The input variables comprise the input layer. The hidden layer consists of some combination of these input variables. The hidden units are thus not restricted to only linear summing elements.

The weight matrix from the hidden layer to the output layer is the one which is learned by the GMDH technique. The output unit works as a conventional processing element. Figure 1 depicts the case, when there is only one output and two input variables. Basically there is no thresholding involved here.

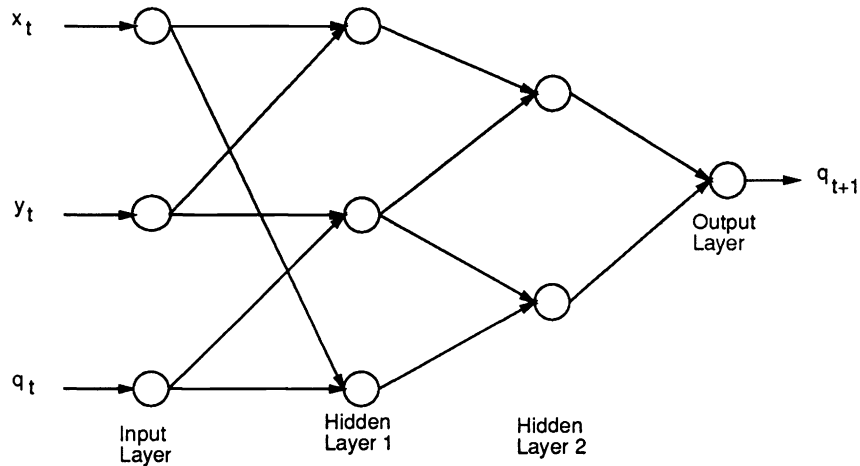


Figure 2: Construction of a PNN using GMDH technique

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 units 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.

2.2 Building the PNN

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 [4]. Because of its supervised nature, inputs and outputs should be available a priori. The input data to the present system is a sequence of the current locations of the mobile robot represented in the cartesian coordinate. The output is the orientation of the robot in the next step. That means the robot will make one step toward this direction. The step size is a fixed quantity. In order that the model does not become overspecialized, the input data set is subdivided into a training set and a checking set.

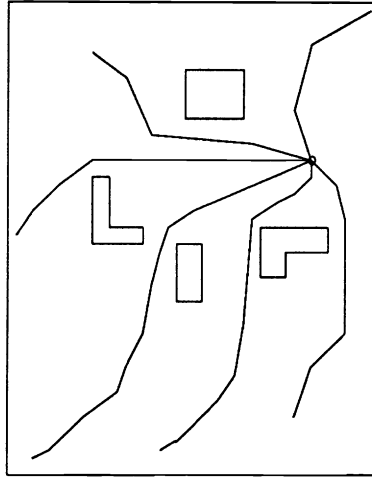


Figure 3: Polynomial Paths in the environment

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 obviously 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. Figure 3 shows the learned polynomial paths.

In our network the input variables are the present location of the robot, x_t, y_t and the orientation, θ_t . The output is the heading direction at the next step. Now the robot will advance in this direction a distance of s unit. The following equations give the relationships.

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

$$x_{t+1} = x_t + s \cos\theta_{t+1} \quad (2)$$

$$y_{t+1} = y_t + s \sin\theta_{t+1} \quad (3)$$

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.

2.3 Finding the closest Polynomial

In effect these polynomials act like attractive path for the mobile robot. If the robot can move

into any point on any of these polynomial paths then it can go to the goal point, following the polynomial that has been learned. Now, our problem is 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 the problem may be treated as an associative retrieval problem, where the coefficients of the learned polynomials form the memory matrix.

In one approach, the Euclidean distance measure in terms of position and orientation from the initial location to all the polynomials are calculated. These values are then used to find the closest learned polynomial.

2.4 Getting into the closest polynomial

Two approaches are considered to move the robot to this closest polynomial. The first approach is the iterative procedure. The estimated next location can be obtained using the built PNNs. The new location is used to predict another next location until the newly predicted locations are within some predefined tolerant distance. Then the robot follows the selected polynomial to the goal location.

The second approach is a fast and non-iterative scheme. A heuristic polynomial is 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. The goal location can then be reached.

3. EXPERIMENTAL RESULTS

The GMDH technique is very sensitive to the splitting of data into training and testing set. 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, as shown in figure 3. The following table shows how closely the built polynomials represent the data set of the environment.

Table 1 : Statistics of 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,

$$A = 0.8286E+00, B = -0.5280E+01, C = 0.1653E+01,$$

$$D = 0.2800E+01, E = -0.6430E+00, F = 0.2392E+01.$$

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

The network is as follows:

Table 2 : Learned Weights for the example polynomial

Layer	Learned Weights					
Output	0.828621	-5.28023	1.65346	2.79959	-0.642972	2.39169
	12.5399	-47.5846	31.5669	40.8442	-0.319126	-35.9942
	-41.0933	43.7552	30.8135	-1.95502	-2.59405	-26.2381
Hidden	3.75715	-9.19382	6.27136	-4.55211	-8.74043	14.3049
	5.75713	-11.0026	5.97662	-5.25081	-10.39851	17.2059
	-0.822634	-16.5021	18.4413	-4.14268	-14.2169	18.1113

4. DISCUSSION & CONCLUSION

As mentioned earlier that no preprocessing of the range data is required in this approach.

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. The word 'known' here means we have the knowledge of the obstacles at the time of training the network.

In future this scheme will be compared with other related planning methodology and also real time obstacle avoidance using this simulator is to be done with our sonar sensor guided mobile robot.

5. REFERENCES

1. A. G. Ivakhnenko, "The Group Method of Data Handling in Prediction Problems," *Soviet Automatic Control*, vol. 9, No. 6, 1976, pp 21-30.
2. 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.
3. O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *Int. J. of Robotics Res.* 5(1), 1986, pp 90-98.
4. 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.
5. A. D. McAulay et al, "Polynomial neural networks for Airborne Applications," *Proc. of the IEEE NAECON Conf.*, 1989, pp 682-687.