A Case Study of Knowledge Acquisition: from Connectionist Learning to an Optimised Fuzzy Knowledge Base

M. Mohammadian, X. Yu and J.D. Smith
Department of Mathematics and Computing
University of Central Queensland
Rockhampton, QLD 4702, Australia

Abstract — This paper presents an automated knowledge acquisition architecture for docking a truck problem. The architecture consists of a neural network controller, a fuzzy rule maker, and a fuzzy controller. The neural network controller is used to learn from trials the driving knowledge. The driving knowledge is then extracted by the fuzzy rule maker to form a driving knowledge rule base. The driving knowledge rule base is further optimized using a genetic algorithm. Computer simulations are presented to show the effectivenss of the architecture.

1. Introduction

Connectionist learning, that is learning by neural nets, has a well known or as some might say uncanny ability to produce reliable results in situations which are mathematically intractable. However connectionist models are not accessible to software engineering, nor in their usual manifestation as a sequential computer model are they particularly fast. In this paper we attempt to get the best of the connectionist and symbolic AI worlds by using connectionist learning to generate a symbolic knowledge base. We have used the truck backing problem to inform the discussion because as an example it is easy to follow and the problem is so well researched as to make comparative results readily available.

Backing a truck to dock is difficult for all but the most skilled truck drivers, as those among us who have tried backing a caravan or boat trailer know very well. Normal driving instincts lead to erroneous movements, much practice is needed to get it right. The truck docking problem is a typical control problem about the determination of a global strategy for guiding the truck that will minimise the length of the truck's pathway from a starting location and orientation to being neatly backed against the dock. In the problem the only available information is the knowledge of local conditions. The problem is difficult to study with the use of any existing analytic approaches.

Nguyen and Widrow [1] first investigated the application of neural networks to the truck docking problem. They used two neural networks to build a self-learning controller to emulate and control the truck: one for identifying the truck's dynamic characteristics, and the other for controlling the emulator. After training, the self-learning controller was able to guide the truck to the dock from almost any initial starting positions. The study demonstrated that the use of neural networks substantially reduced the amounts of human effort and design time which would have required to devise such controller. However the controller required a significant amount of time to train; learning was time-consuming. In some cases the learning algorithm did not converge.

Neural networks generally represent their knowledge in terms of a weight matrix which is not accessible to human understanding at present. Fuzzy logic, on the other hand, is easy to understand and well proven in control applications. Kosko and Kong [2] studied the truck docking problem using fuzzy rules. They first trained the fuzzy controller by encoding a common sense FAM (Fuzzy Associate Memory) bank. They then developed an adaptive fuzzy controller which generates FAM rules directly from training data using a product-space clustering algorithm. Compared with the neural controller, the fuzzy controller was computationally much lighter than the neural controller. However the extraction of driving knowledge was obtained by an off-line statistical approach which seems to be too complicated. The fuzzy controller was further tuned by Wiggins [3] using a genetic algorithm in order to get the optimum truck paths. Wang and Mendel [8] discussed the same problem using methodology similar to Kosko and Kong[2]. They proposed a numerical-fuzzy controller for the docking a truck problem. However the extraction of knowledge was again in an off-line manner. A lot of human experience was involved in the construction of the fuzzy rule base.

In this paper we consider the same problem but develop a knowledge acquisition architecture which enables the extraction of driving knowledge to be automated. The architecture consists of a neural network controller, a fuzzy rule maker plus genetic algorithm, and a fuzzy controller. The neural network controller is a modification of the controller by Nguyen and Widrow [1] that produces satisfactory training data very quickly. Unlike the way of extraction of knowledge by Kosko and Kong[2], the acquisition of driving knowledge is done in an on-line manner which coincides with human being's learning pattern. This driving knowledge is then put into a truck driving expert system. The rules which form the knowledge base of the expert system can be updated whenever new training knowledge is generated. After knowledge base generation the rules are then optimised by a genetic algorithm which produces a shorter path and less docking error with fewer rules.

The paper is organized as follows. The knowledge acquisition architecture is presented in Section 2. Computer simulations are given in Section 3. The findings are discussed in Section 4. The neural controller developed by Nguyen and Widrow [1] is reviewed in Appendix A. Appendix B outlines the fuzzy controller by Kosko and Kong [2] and the genetic optimisation by Wiggins [3].

2. Knowledge Acquisition Architecture

2.1 Overview

The docking a truck problem has been studied by Nguyen and Widrow [1] using neural networks and Kosko and Kong [2] using fuzzy logic. The fuzzy controller developed by Kosko and Kong was optimised by Wiggins [3]. For brief description of their results, readers are referred to Appendices A and B.

There are some problems with the neural controller of Nguyen and Widrow and the fuzzy controller of Kosko and Kong, though one hastens to add both are excellent at actually docking trucks. The neural controller needs a substantial amount of time to train. By contrast, the fuzzy controller is computationally much lighter than neural controller. However it does need a real human expert to extract the driving knowledge. Kosko and Kong used a product-space clustering algorithm to extract the driving knowledge in an off-line manner which seems quite complicated.

Humans learn in an on-line manner; knowledge is updated as new experience is obtained. The knowledge a human has is the result of incremental updating rather a single computation following the collection of all training data. Neural networks do mimic the human style of knowledge acquisition. However the knowledge representation as a weight

0-7803-0985-5/93/$3.00 © 1993 IEEE

matrix is not yet understood.

In this section we aim to develop a knowledge acquisition architecture which enables knowledge acquisition to be automated in an on-line manner. Note that knowledge acquisition [4] is one of the major bottle-necks in developing modern knowledge intensive artificial intelligence systems.

The architecture consists of three parts: the neural network block, the fuzzy rule maker, the genetic optimisation block.

2.2 Neural Network block

The neural network is the same as the one developed by Nguyen and Widrow[1] and trained by the backpropagation algorithm. However the neural network weights are updated at each iteration step, rather after each training run, to minimise the objective function. This modification reduces the training time significantly. Normally two or three training runs will give a satisfactory training trajectory from a given starting position. The selection of the parameters in the objective function affects significantly the performance of the backing-up process, such as the speed of convergence, the smoothness of the trajectories etc.

2.3 Fuzzy rule maker

The trained neural net then feeds the Fuzzy Rule Maker (FRM) which generates a set of fuzzy rules to simulate the neural network's behaviour. The definition of fuzzy sets and membership have to be pre-set prior to the training data coming in. Firstly the linguistic variables corresponding to the input/output data are defined. Then the associated fuzzy sets are set to be intervals without overlapping. The membership functions can be either triangular or trapezoidal. We used triangular shaped membership functions in this study. The workspace should be partitioned in a logical manner - that is for fuzzy memberships where finer control is needed the fuzzy sets should be smaller. The FRM should be able to estimate the fuzzy rules underlying expert behaviour in various cases, even when the expert or fuzzy engineers can not articulate these rules.

2.4 Genetic optimisation block

The fuzzy rules (we rather call them raw fuzzy rules) are then sent to the genetic optimisation block and assessed. The parameters for optimisation may be the overlaps between the fuzzy sets, or shapes of membership functions, or the definition of the linguistic fuzzy variables. We can tune the fuzzy controller by either changing the membership functions without changing the linguistic variable definitions, or changing the definitions of each linguistic variables. The genetic algorithm chooses randomly overlaps between fuzzy sets and finds their fitness. The fitter overlaps then combined together to generate new overlaps with higher fitness. The objective function can be associated with another criterion - the shortest path from starting location to the desired dock.

2.5 Knowledge acquisition procedure

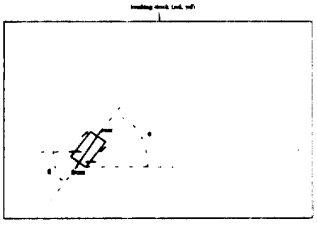

Fig.1 Diagram of simulated truck and loading zone

Before we detail the knowledge acquisition procedure, let us first specify the docking workspace and the truck. Fig.1 shows a simulated truck and loading zone [2]. The three variables, x, y and ϕ determine the truck position. ϕ is the angle of the truck with the horizontal. The coordinate pair (x, y) specifies the position of the rear centre of the truck in the plane. The goal is to make the truck arrive at the loading dock at a right angle ($\phi_d = 90^o$) and to align the position (x, y) of the truck with the desired loading dock (x_d, y_d). In this study only backing up is considered, the truck moves backwards by some fixed distance at every stage.

The knowledge acquisition procedure goes as follows:

1. Train the modified neural network controller to get a training trajectory for an initial location.
2. Get the input/output data of the training trajectory and classify the data into groups according to the definitions of the linguistic fuzzy sets with respect to the input variables x and ϕ.
3. For each group calculate the mean (average) of the output variable data $\hat{\theta}$ and the standard deviation of the mean $\sigma_{\hat{\theta}}$ in the corresponding input fuzzy sets with respect to x and ϕ. Locate the corresponding output linguistic fuzzy variable for the mean of θ.
4. IF $\hat{\theta} \pm \sqrt{\sigma_{\hat{\theta}}}$ belong to one and the only one output linguistic fuzzy set, then record it as a fuzzy *if-then* rule into the FAM bank. If $\hat{\theta} \pm \sqrt{\sigma_{\hat{\theta}}}$ belong to more than two output linguistic fuzzy sets, which means we have more than two rules which have the same antecedent/s but different consequent/s, then simply choose the one with highest membership value. The input fuzzy sets construct the antecedent of the fuzzy rule while the output fuzzy set/s construct the consequent of the fuzzy rule. If the system has multiple input/output fuzzy sets then the input fuzzy sets are joined in through conjunctive "and" in antecedent part and so does the output fuzzy sets in consequent part of the fuzzy *if-then* rule.
5. Repeat Step 3 until all groups for the training trajectory are treated.
6. Go back to Step 1 to get another training trajectory with the neural controller until all the expected starting locations are tried. If a training trajectory gives a rule in some region in conflit with a rule produced with previous training trajectory, choose the one with less standard deviation.
7. The FAM bank that has been obtained is sent to genetic optimisation block. The membership functions for linguistic fuzzy sets of the FAM bank are tuned to minimise the docking error and shorten the path.

The architecture is depicted in Fig. 2.

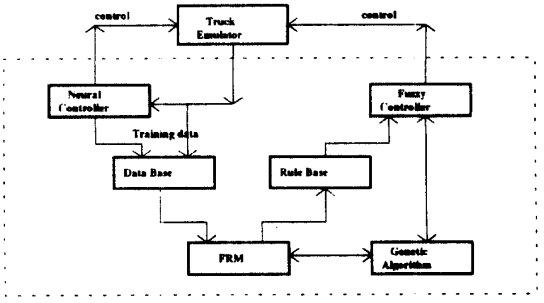

Fig. 2 The knowledge acquisition architecture

3. COMPUTER SIMULATIONS

The following computer simulations are performed on an IBM compatible

486/33MHz Personal Computer.

We first present the simulation results to show the effectiveness of the modified neural controller. The initial location is at (70, 20) with azimuth 144°. The truck with size 10x5 moves 0.2 each time. The desired loading dock is located at (50, 100) with azimuth 90°. The neural network contains 20 neurons with its weights pre-set to be random numbers between -0.5 and 0.5. This setting seems a good choice as it gives quite good docking trajectory. The parameters of the objective function are chosen as $a_1 = 0.5$, $a_2 = 0.5$, $a_3 = 0.05$. The learning rates are $\eta_\alpha = 0.1$ and $\eta_\gamma = 0.1$. Rather update the weights of the neural network after each training run, we update the weights after each iteration. This means we no longer expect that the weights of the neural network eventually converge to certain values. What we are concerned is the training data - truck trajectories which is crucial for driving knowledge acquisition. Experiments show that this treatment substantially reduces the training time. A satisfactory trajectory will normally be found after just a few runs. Fig. 3 depicts the docking performance of the truck starting from (70, 20) with azimuth 144°.

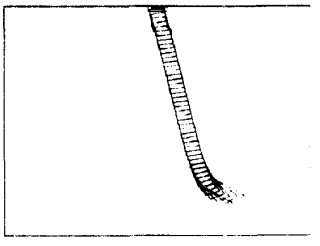

Fig.3 Truck trajectory of the modified neural controller for initial position (70, 20) with azimuth 144°

Now we present the simulation results for knowledge acquisition. Unlike the neural controller, the fuzzy controller has only two inputs, the x position and azimuth ϕ [2]. The output is the steering angle θ. This is because of the assumption that there exists enough clearance between truck and the loading dock for the y coordinate to be ignored.

The linguistic fuzzy subsets for x, ϕ and θ, and the membership functions associated with the subsets for x, ϕ, are the same as used in Kosko and Kong's simulations. But the membership functions associated with subsets for θ are set to have no overlaps. They are defined as follows:

NB $\leftrightarrow$ [-30, -20] NM $\leftrightarrow$ [-20, -7.5] NS $\leftrightarrow$ [-7.5, -2.5]
ZE $\leftrightarrow$ [-2.5, 2.5] PS $\leftrightarrow$ [2.5, 7.5] PM $\leftrightarrow$ [7.5, 20]
PB $\leftrightarrow$ [20, 30]

The reason of setting overlap to zero is that we do not know at the beginning what overlap would give better results. The overlaps as a parameter for optimisation will be tuned later using a genetic algorithm.

Before starting training, each entry of the entire FAM bank is initialised as ZE. To acquire the driving knowledge in terms of human linguistic language using the developed knowledge acquisition architecture, we start the training from the initial positions: (10, 10), (30, 10), (50, 10) with azimuths -50, 30, 150 and get the FAM bank as in Table 1.

The entries with ZE are because the truck trajectories never pass through those regions. Although the rules in Table 1 give quite good truck trajectories, it does not mean that the rules obtained are the best. With the knowledge acquisition architecture proposed, the more training data one gets, the finer the control rules.

Figure 4 depicts the truck trajectory using this FAM bank to control the

ϕ \ x	LE	LC	CE	RC	RI
RB	PB	PB	PB	PB	ZE
RU	PB	PB	PB	PB	ZE
RV	NB	PM	PS	PB	PB
VE	NM	ZE	ZE	PS	PS
LV	ZE	NB	ZE	NM	PB
LU	ZE	ZE	NB	NM	NM
LB	ZE	ZE	ZE	ZE	ZE

Table 1. FAM bank for fuzzy controller with training data for initial positions: (10, 10), (30, 10), (50, 10) with azimuths -50, 30, 150.

truck from the initial position (70, 20) with azimuth 144°. The docking error is 4.05. Even some rules are missing from the bank, the fuzzy controller still performs well.

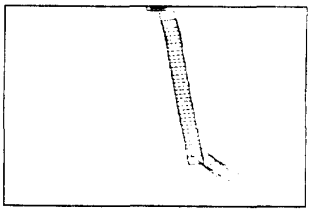

Fig.4 Truck trajectory of the fuzzy controller for initial position (70, 20) with azimuth 144°. Overlap is 0%.

The overlaps between the fuzzy subsets for θ can be used as parameters for optimisation. In our study we consider a simple case one overlap parameter case. The objective is to minimise the docking error. We start the genetic algorithm with 10 randomly generated overlap parameters. The value of each parameter is converted to a string of 0's and 1's with length 7. The genetic algorithm produces 10 generations and the overlap parameter which gives minimum docking error among those docking errors produced by other overlap parameters is chosen as the "best" value. The genetic algorithm produces the better overlap parameter 15%, which gives the docking error 1.75.

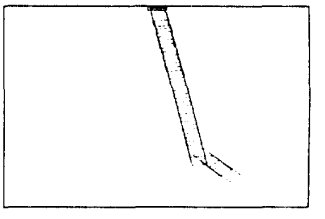

Fig.5 Truck trajectory of the fuzzy controller for initial position (70, 20) with azimuth 144°. Overlap is 15%.

More generations may give value which is better than the "best" overlap parameter. However it very much depends on the values of initial overlap parameters randomly chosen and random crossover between the overlap parameters with high fitness.

A more complete FAM bank is given in Table 2 which is done with zero overlap between output fuzzy subsets after 50 starting positions are tested. The table shows a finer control than Table 1.

Even after up to 50 more training sets, we still do not have rules for all regions. To obtain those missing rules, one has to test more starting positions so that the truck will pass through those regions and more rules will be generated. However it seems not necessary to test more starting position as for example all the neighbouring blocks around (LB, CE) have rules except (LB, CE). It should be able to be guessed like humans doing.

φ \ x	LE	LC	CE	RC	RI
RB	PB	PB	PB	PB	PB
RU	PM	PB	PB	PB	PB
RV	NS	PS	PS	PM	PM
VE	NM	ZE	ZE	ZE	PS
LV	NM	NM	NS	PS	PB
LU	NM	NM	NB	NM	NM
LB	NB	NB	ZE	NB	NB

Table 2. FAM bank for fuzzy controller with training data for 50 initial positions

It is should be possible to infer the rule for (LB, CE). This will be investigated in the future.

4. Discussions and Conclusions

We have studied the truck driving knowledge acquisition in the context of a knowledge acquisition architecture for the automated acquisition of knowledge.

The modification to the neural controller is proved to be effective as it reduced substantially the training time required. It breaks the traditional customs of neural network learning which try to train the network weights to converge to certain values. The approach is acceptable in this case because the purpose is the extraction of training data. We have shown that the quality of a truck trajectory is affected by the choice of the parameters of the objective function. This area will be studied further.

The fuzzy rule maker basically uses a statistical approach to manipulate the training data and find fuzzy rules. The approach used does not work occasionally as the mean and variance do not always reflect some extreme cases like sudden turning of the truck etc. A more sophisticated approach will be developed.

The genetic algorithm has demonstrated its effectiveness in optimising the fuzzy controller. In this paper a simple case that only one overlap parameter is used to optimise the fuzzy rules has been discussed. It is intended to use more than two parameters to tune the fuzzy rules to give better performance.

The case study is an illustration of how connectionist learning and knowledge acquisition, a bottle-neck of AI, can be well combined and support each other. It is our intention to use the knowledge acquisition architecture to study more complicated practical control systems.

Acknowledgement

The second and third authors are indebted to the University of Central Queensland for Research Grant URG 93/31.

References

[1] D. H. Nguyen and B. Widrow, "Neural networks for self-learning control systems", *IEEE Control Systems Magazine*, April, 1990.

[2] B. Kosko, *Neural Networks and Fuzzy Systems*, Prentice-Hall, Englewood Cliff, NJ07632, 1992.

[3] R. Wiggins, "Docking a truck: a genetic approach", AI Expert, May, 1992

[4] S. Sestito and T. Dillon, *Automated Knowledge Acquisition*, Prentice Hall, 1992.

[5] T. Takagi and M. Sugeno, "Fuzzy identification of systems and its applications to modelling and control", *IEEE Trans. on Syst., Man, and Cybernetics*, vol.15, pp.116-132, 1985.

[6] B. Kosko, *Neural Networks for Signal Processing*, Printice Hall, Englewood Cliffs, NJ07632, 1992.

[7] D. E. Goldberg, *Genetic Algorithms in Search, Optimisation, and Machine Learning*, Addison-Wesley Inc., 1989.

[8] L-X.Wang and J. M. Mendel, "Generating fuzzy rules by learning from examples", *IEEE Trans. Syst., Man, and Cybernetics*, vol.22, pp.1414-1427, 1992.

Appendix A : Neural Controller

In this section the neural controller developed by Nguyen and Widrow [1] for the docking a truck problem is briefly reviewed.

A.1. Neural networks and back-propagation algorithm

A neural network is composed of many simple and similar processing elements. Adaline is used as the processing element. Normally two layers of Adalines with two sets of weights suffice for implementing any nonlinear function. An Adaline is defined to have an input vector $X=\{x_i\}$, an output y, and a weight vector $W=\{w_i\}$, i=1,2,...,n, which satisfy that $y(X)=f(\sum_{i=1}^{n} w_i x_i)$, where $f()$ is sigmoid function. Assume there are m output, then y can be considered as a vector whose entries $y_j(X)=f(\sum_{i=1}^{n} w_{ji} x_i)$, $j=1,...,m$. The desired output is denoted as $d(X)$.

The algorithm used to train the layered neural networks is known as *back-propagation* [1] which guides the weights of neural networks to converge to a set of weights that minimizes the mean-square error $J=E(\|d(X)-y(X)\|)$. Updating weights is undertaken by updating rules

$$\gamma_{ki,new}=\gamma_{ki,old}-\eta_\gamma\frac{\partial J}{\partial \gamma_{ki}},\ \alpha_{ij,new}=\alpha_{ij,old}-\eta_\alpha\frac{\partial J}{\partial \alpha_{ij}} \qquad (A1)$$

where γ_{ki} and α_{ij} are the weights for the first layer and the second layer respectively with $k=1,...,m$, $i=1,...,n_{neu}$, $j=1,...,n$. Here m is the number of outputs, n_{neu} the number of hidden neurons and n the number of inputs. η_γ and η_α are the learning rate parameters.

A.2. Neural network docking truck system

The neural network docking truck system developed by Nguyen and Widrow [1] consists of two neural networks: one for identifying the truck dynamic characteristics, and the other for controlling the emulator. Before training the neural controller, the neural network emulator is first trained which behaves like the truck to be controlled. The inputs for the neural controller are two coordinates x, y, azimuth ϕ, the output is the steering angle θ. The truck is assumed only to move backwards with constant speed. The objective function is

$$J=E(\alpha_1(x_d-x)^2+\alpha_2(y_d-y)^2+\alpha_3(\phi_d-\phi)^2) \qquad (A3)$$

the mean-square error over all training runs. The constants $\alpha_1,\alpha_2,\alpha_3$ are chosen to weigh the importance of each error component.

APPENDIX B : Fuzzy Controller

In this section, the fuzzy controller proposed by Kosko and Kong [2] is reviewed. The optimisation scheme of fuzzy controller using genetic algorithm [3] is outlined.

B.1. Fuzzy docking truck system

In the fuzzy docking truck system [2] , the truck emulator was replaced by

$$x' = x + r\cos(\phi'),\quad y' = y + r\sin(\phi'),\quad \phi' = \phi + \theta \qquad \text{(B1)}$$

which show that the truck moves backwards from position (x,y) with azimuth ϕ to (x', y') with azimuth ϕ'. This replacement as discussed in [2] does not affect the post-raining performance of the docking truck system, since the truck emulator network backpropagates only errors. The variable ranges are $0 \le x \le 100$, $0 \le y \le 100$, $-90 \le \phi \le 270$, $-30 \le \theta \le 30$.

Unlike the neural controller, the fuzzy controller has only two inputs, x position and azimuth ϕ. The output is still the steering angle θ. This is because of the assumption that there exists enough clearance between truck and the loading dock so the y coordinate could be ignored.

The common-sense linguistic fuzzy-set values of the fuzzy variables which an expert may use to describe the driving knowledge are as follows :

azimuth ϕ	position x	steering-angle θ
RB : Right Below	LE : Left	NB : Negative Big
RU : Right Upper	LC : Left Centre	NM : Negative Medium
RV : Right Vertical	CE : Centre	NS : Negative Small
VE : Vertical	RC : Right Centre	ZE : Zero
LV : Left Vertical	RI : Right	PS ; Positive Small
LU : Left Upper		PM : Positive Medium
LB : Left Below		PB : Positive Big

Each linguistic variable (or fuzzy subset) is associated with a fuzzy membership function. A fuzzy membership function is defined as a mapping function $m_A(a) : a \rightarrow [0,1],\ a \in A$. This function $m_A(a)$ indicates the degree to which a belongs to the fuzzy set A [2]. The triangular shaped fuzzy membership functions are used in this study.

Fuzzy rules are normally written in terms of antecedent-consequent pairs of IF-THEN statements . An example of this kind of IF-THEN rules is **If** x = **CE and** ϕ = **VE then** θ = **ZE** . This fuzzy rule indicates that if the truck is in the centre of the working lot and the azimuth is vertical, then the controller should not produce a positive or negative steering-angle signal. Since the truck is assumed only to move backwards with constant speed, the truck will eventually reach the dock with the desired azimuth.

A common-sense fuzzy associated memory (FAM) bank can obtained as

ϕ \ x	LE	LC	CE	RC	RI
RB	PS	PM	PM	PB	PB
RU	NS	PS	PM	PB	PB
RV	NM	NS	PS	PM	PB
VE	NM	NM	ZE	PM	PM
LV	NB	NM	NS	PS	PM
LU	NB	NB	NM	NS	PS
LB	NB	NB	NM	NM	NS

Table B.1 FAM-bank for fuzzy controller

associated with the fuzzy membership functions

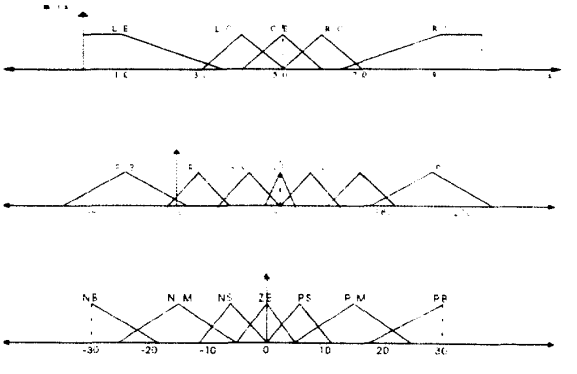

Fig.B.1 Fuzzy membership functions for each linguistic fuzzy set value

The FAM bank in Table B.1 says that given input conditions of x and ϕ, the output θ is then determined.

The output fuzzy set Θ can be determined by the correlation-minimum inference

$$\Theta = \sum_{i=1}^{N} \Theta_i = \sum_{i=1}^{N} \min(f_i, s_i) \qquad \text{(B2)}$$

where Θ_i is the invoked output FAM rule, N is the number of Θ_i, and f_i is the antecedent fit value and s_i the consequent fit value in the ith FAM rule. To defuzzify the output fuzzy set Θ, the Centroid defuzzification scheme is used which gives centroid $\bar{\theta}$ as:

$$\bar{\theta} = \frac{\sum_{i=1}^{N} \theta_i m_\Theta(\theta_i)}{\sum_{i=1}^{N} m_\Theta(\theta_i)} \qquad \text{(B3)}$$

where Θ defines a fuzzy subset of the steering-angle universe of discourse $\theta = \{\theta_1, \ldots, \theta_N\}$.

The most interesting result is the adaptive FAM system which generates FAM rules that represent driving knowledge acquisition. The following FAM bank is extracted from the training data from the neural controller using product-space clustering algorithm associated with Differential Competitive Learning (DLC) [2]. The FAM bank is slightly different from Table.B.1. However the control effects are the same. They both drive the truck to the dock

ϕ \ x	LE	LC	CE	RC	RI
RB	PS	PM	PM	PB	PB
RU	ZE	PM	PM	PB	PB
RV	NS	ZE	PS	PM	PB
VE	NM	NM	ZE	PM	PM
LV	NB	NB	NS	PS	PS
LU	NB	NB	NB	NS	PS
LB	NB	NB	NM	NM	NS

Table B.2 Adaptive FAM bank for fuzzy controller

B.2 Optimisation using genetic algorithm

Genetic algorithms are a search procedure which optimises the goodness of fit of a function to some observations [7]. The algorithm is called genetic because it is based on mechanics of natural selection and natural genetics.

Typically, Genetic algorithms use a bit string to encode a solution. Each genetic code (bit string) represents a member or individual to the function(s) being optimized albeit it is not necessarily an optimal solution. A number of members which can be generated on a random basis forms a genetic algorithm population. The genetic algorithms are known to exponentially increase the occurrence frequency of those members (bit strings) that poses higher than average "fitness". A genetic algorithm normally proceeds as follows: 1. Evolution - Fitness is determined for members of a population of bit strings; 2. Selection - Population members are assigned a number of copies in a mating pool that is used to construct a new population .the higher a population member's fitness, the more copies in the mating pool it receives; 3. Recombination (crossover) - Members from the mating pool are recombined (crossed over) randomly to form new individuals. They are cut into two pieces at a randomly selected crossover (cut) point. The pieces are swapped between members to form new members; 4. Mutation - Each bit (locus) of a member's binary string (gene) is flipped with some small probability << 1.0. Mutation is a mechanism for maintaining diversity in the genetic algorithm population. The new members that result form these steps from a new population, and the process is iterated.

Genetic algorithms differ from other optimisation and search procedures by considering a population of points from the search space and therefore reduce the chance of trapping into local optima.

Anything that a fuzzy function can emulate, the genetic algorithms can optimise. The fuzzy controller can be tuned [3] by choosing steering-angle θ as the parameter for optimisation. As initially the overlaps between linguistic variables for θ are set to zero, the numbers of first generation can be chosen as different positive numbers. By crossovering the numbers to generate new generations, an overlap which gives better docking performance may be obtained. The performance indexes may be the docking error, the optimum driving path, and the combination of both, depending upon if minimum docking error, shortest driving path, and minimum docking error as well as shortest driving path, are required respectively.