The Integration of an Optimised Fuzzy Logic Navigation Algorithm into a Semi-Autonomous Robot Control System.

P. Webb, C. Fayad and C. Brettenbach
School Of Mechanical, Materials, Manufacturing, Engineering and Management
The University of Nottingham
University Park
Nottingham
NG7 2RD

Abstract: The requirements for greater complexity and robustness in real time software for autonomous and semi-autonomous mobile robot has resulted in the proposal of a number of different software architectures being proposed. This paper describes the implementation of a layer-based architecture and concentrates in particular on the realisation of the Navigation Layer. The Navigation layer described here is developed using fuzzy logic and examines the use of genetic algorithms to optimise the design of a fuzzy logic controller. The design and selection of linguistic variables and rules for rule-based fuzzy logic controllers requires expert knowledge of the system which is to be controlled. This knowledge is usually obtained by trial-and-error or by consulting and observing a human operator controlling a real system; neither leads to an optimal solution due to the limited space of search. The use of genetic algorithms offers a possible solution to this problem.

1. **Introduction:** Autonomous navigation systems are often classified depending on the characteristics of the environments within which they have to move. These environments may be classed as structured or known, unstructured or unknown, and partially known or semi-structured[1]. The robot control system described in this paper is designed to operate in a semi-structured environment. This means that the robot is equipped with an incomplete map of its environment and has to be able to account for the appearance and disappearance of objects within the workspace. The controller for such a robot must be able to run in real-time and run a number of different processes concurrently. The software must be robust and able to withstand the failure of one of the concurrent modules and maintain the functionality of the remainder of the system. This means that in the event of a sensory failure the robot could still be recovered using manual control.

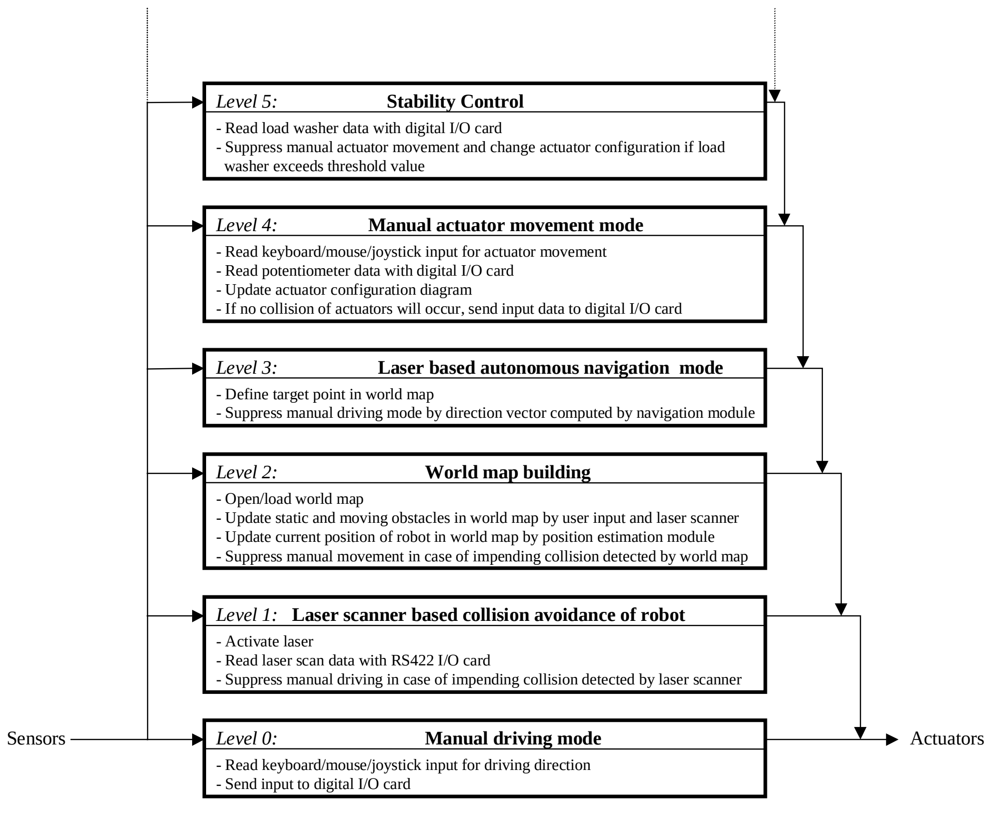
An architecture that satisfies the above criteria is that proposed by Brooks which consists of a series of layers, or levels of competence. This is the approach used in the system described in this paper. In particular the layer which provides Navigation is discussed in this paper. A number of different approaches have been proposed for the navigation of mobile robots in semi-structured and unstructured environments, these include map based, potential field[1] and fuzzy logic[2]. The algorithm described in this paper is based on fuzzy logic.

2. **The Demonstrator Vehicle.** The architecture described in this paper is implemented on a technology demonstration and development vehicle that has been constructed around a Wheelbarrow Mk7 Explosive Ordnance Disposal Vehicle. The vehicle is connected to the control station by an umbilical which caries control

information, sensory information, CCTV signals, Laser power and a high-speed serial data link (RS422). The vehicle control system and user interface uses a 300 MHz Pentium III computer and hand held control box. The system software is written using C++ operating under the Windows NT operating system.

The external sensory input to the control interface is obtained from an infrared scanning laser range finder mounted on the front of the vehicle. The Laser system is manufactured by Erwin Sick of Germany and provides a full 180° scan with a maximum range of 50m, an angular resolution of 0.5° and an accuracy of ±50mm.

3. **The Control Architecture.** The control architecture is designed according to Brooks's[3] principles of layers each with different behavioural competencies. This not only allows the system to operate robustly but also the control program of the robot can be extended at a later date by simply adding new levels on top of the existing ones. On adding new levels the communication structure of the previous levels is not changed but rather enhanced. Through suppression and inhibition of specific software elements in the layers below it is possible to modify previously implemented behaviour patterns without changing the existing structure. By using this approach the earlier mentioned point of robust program execution is fulfilled. If the control functions of a particular level fail then the behaviour patterns of the lower levels still work properly. A diagram of the structure of the individual layers used in the control architecture of the robot described in this paper is shown in figure 1.



Level 5: Stability Control
- Read load washer data with digital I/O card
- Suppress manual actuator movement and change actuator configuration if load washer exceeds threshold value
Level 4: Manual actuator movement mode
- Read keyboard/mouse/joystick input for actuator movement
- Read potentiometer data with digital I/O card
- Update actuator configuration diagram
- If no collision of actuators will occur, send input data to digital I/O card
Level 3: Laser based autonomous navigation mode
- Define target point in world map
- Suppress manual driving mode by direction vector computed by navigation module
Level 2: World map building
- Open/load world map
- Update static and moving obstacles in world map by user input and laser scanner
- Update current position of robot in world map by position estimation module
- Suppress manual movement in case of impending collision detected by world map
Level 1: Laser scanner based collision avoidance of robot
- Activate laser
- Read laser scan data with RS422 I/O card
- Suppress manual driving in case of impending collision detected by laser scanner
Level 0: Manual driving mode
- Read keyboard/mouse/joystick input for driving direction
- Send input to digital I/O card
Sensors
Actuators

Figure 1: Robot Control Architecture.

For the internal structure of each level of competence a different approach is used. Each level of competence is built up of several software elements, the finite state machines. A finite state machine holds a (finite) number of data structures and is able to perform basic operations (e.g. send, receive, if-then-decision, simple data manipulation). The communication between the separate finite state machines is achieved by sending and receiving messages.

Since the layers in the above structure resemble objects in an object orientated design (OOD) [2, 3] the approach is used for the internal structure of the levels of competence. The OOD also facilitates the integration of software modules developed by previous developers treating them as black box with unknown content. The only thing of interest is the objects' interface, which is used for embedding it into the control program.

This paper specifically covers the operation of levels 3 and 2. The laser based autonomous navigation mode and world map building layer. Level 0 is a low level control layer, which provides communication and control of actuators. Level 1the collision avoidance layer uses the Laser to provide real time collision avoidance to protect the robot from collisions whilst being driven manually. The system operates on a 'warn and stop' principle, initially warning an operator of an impending collision and finally stopping the robot before an actual collision occurs. Level 4 has been previously described[4] whilst level 5 is currently under construction.

4. **World map building layer.** The above illustrated control architecture is designed as an overall control of the system, as the mobile robot is to be used in completely known, partially known, and unknown environments. Both in known and unknown environments, the world map building layer generates three different maps of the robot's world. The initial map which is entered by the operator, the current map that is a map generated by the laser and a combined map made up from the two previous maps. The use of the combined map means that the initial map does not need to be complete. The position of the robot relative to the initial map is calculated by tracking stationary objects and calculating their displacement relative to the robot. To enable this to occur, the initial position of the robot within the entered world must be known.
In unknown environments, which is the case of navigation in this paper, the system is not provided with an initial map. Navigation becomes of a reactive nature because there is neither planning, nor memory or map building. In fact, the vehicle relies completely on information acquired from sensors set on board, which makes it rather difficult to build a precise and complete world map. Thus, each decision is taken using only input values generated instantly and the map building layer is resolved to providing on-line local maps only.

5. **The Autonomous Navigation Layer.** Behaviour-based control shows potentials for reactive robot navigation as it does not require exact world maps. Nevertheless, one key issue of behaviour-based control remains how to efficiently co-ordinate different behaviours together.
In Brooks[5], co-ordination of multiple reactive behaviours is done by according different levels of activation depending on behaviour priorities: one behaviour is fired and other behaviours are inhibited according to their suitability. Artificial potential field is another traditional approach for implementing reactive behaviours. This approach suffers from a

drawback as much effort must be made prior to simulation to test and adjust thresholds regarding potential fields for collision avoidance, target steering, edge following and etc…[6].
Fuzzy logic also has been used as one approach in behaviour-based control as it provides the opportunity to decompose each relevant behaviour and quantitatively formulate it in the shape of fuzzy sets and rules. It allows also co-ordinating conflicts between different types of behaviours. Unlike traditional approaches where appropriate behaviours are chosen by inhibiting other behaviours, the fuzzy logic based approach fuses different types of behaviour using fuzzy reasoning. Fuzzy logic gives the advantage of firing all types of behaviours simultaneously[7,8].

In this study, fuzzy logic is applied in the inference procedure for the mobile robot navigation to guide the robot in a collision free path from a staring location to a target location. The autonomous navigation layer allows the robot to automatically navigate from a starting position to a target position defined as an absolute point (x,y) by the user. The system operates in real time and is based around a fuzzy logic controller that calculates an escape vector whenever an obstacle is detected in the path of the robot. Each command generation is done using only sensory input values available from current readings.

The max-min inference method proposed by Mamdani is applied for fuzzy inference and the Centroid of Area is used to deffuzzify the inference results of fuzzy rules.

5.1. **Fuzzy Controller.** For collision avoidance, a reactive control system using fuzzy logic has been developed. It implements an intelligent reactive control strategy in that it senses the immediate environment and reacts to current sensory data. The software development for a fuzzy logic control system is facilitated by the use of available tools dedicated to fuzzy rule base generation. A fuzzy controller takes the form of a set of IF-THEN rules whose antecedents and consequents are membership functions.

The key steps for solving a problem using fuzzy logic are as follows:
1. Define the fuzzy problem in detail
2. Identify all important variables and their ranges
3. Determine scaling factors
4. Determine membership profiles for each variable range
5. Determine rules (propositional statements) including action needed
6. Select the deffuzification methodology
7. Test the system for correct answers; if needed go back to step 3

Behaviour control makes reactive control possible, decomposing the motion control system into a set-of-special purpose routines (behaviours) that achieve simple but distinct tasks when subject to a particular stimuli.
Fuzzy logic and reactive/behaviour control share common advantages for designing intelligent systems. Individual fuzzy rules (and behaviours) can be formulated independently, and additional rules (and behaviours) can easily be added to the control system if needed.

5.2. **Behaviours.** The approach is behaviour-based and Fuzzy Logic is used as a reliable technique for behaviour fusion. In this paper, the robot alternates between two behaviours: goal seeking and obstacle avoidance.
In traditional approaches, escaping from an eminent train heading forward becomes of a higher importance than painting the racks in white[5]. Suppressing one behaviour to activate the other is not anymore a necessity as Fuzzy Logic allows different behaviours to run simultaneously.
In order to do behaviour fusion using fuzzy logic, first all behaviours are classified in fuzzy sets and fuzzy rules, such as (1) obstacle avoidance and (2) goal steering. Second, rules are weighed by fuzzy reasoning. All rules fire simultaneously; however with different weights. Then clipped output sets are brought together in a Min-Max inference algorithm and the Centorid deffuzification method is used to determine one appropriate control action by efficiently weighting different behaviours.
However, it remains that one of the central factors in the design of efficient and robust fuzzy logic controllers is the selection of the membership functions and rules.

(1) Obstacle Avoidance Behaviour. In this control scheme, the fuzzy controller input signals are the angle of the nearest object relative to the robot and its range as measured by the Laser sensor. An obstacle located to the left-hand side of the mobile robot is assumed as an obstacle with a negative heading angle. Likewise, an obstacle located to the right-hand side of the robot is assumed to be of a positive heading angle. The output of the controller is an escape vector consisting of a distance and heading angle, leading the robot to an approximate target. Once this vector has been followed the robot then returns to a recalculated heading vector.
Obstacles within the path of the robot are detected using a laser with a scanning range of almost 50 meters. Each obstacle detected has a depth variable relative to the position of the robot.
The range universe is partitioned in 3 fuzzy subsets, where parameters and membership functions are chosen. The linguistic variables that describe the range information: Far (F), Medium (M), and Near (N) (Figure 1).

The laser scans from –90 to 90. The **direction universe** is divided into six subsets. The linguistic variables that describe the angle heading: Large Negative (LN), Medium

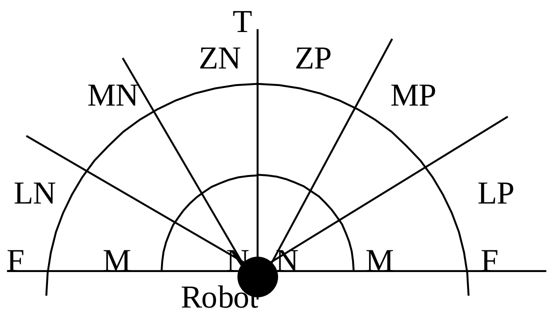


Figure 1 Division of the space in fuzzy subsets

Negative (MN), Zero Negative (ZN), Zero Positive (ZP), Medium Positive (MP), Large Positive (LP) (Figure 1).

Each of the range and direction fuzzy subsets is illustrated in a fuzzy membership (Figure2 and Figure 3).

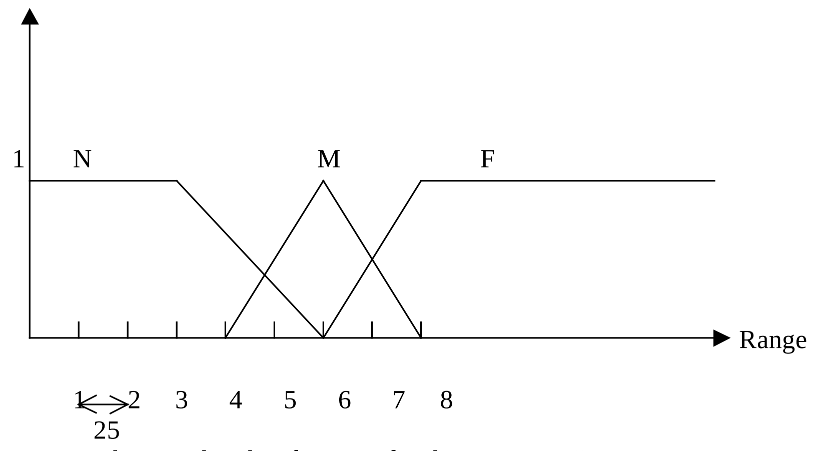


Figure 2 The Membership function for the set "range"

Scaling factor =25

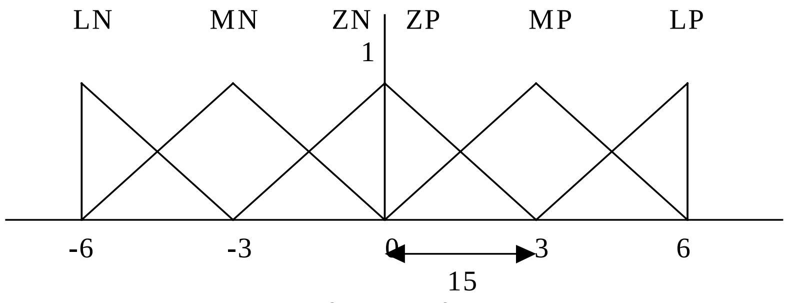


Figure 3 The membership function for the set "direction"

Scaling factor =15

The derivation of control rules is a step of major significance to the performance of the fuzzy logic controller. A number of methods for rules derivation have been considered that can be put into three classes:

(1) Based on human operator's experience
(2) Based on observing control action performance
(3) Based on learning[9]

In this approach, the set of rules used was generated as a combination of methods 1 and 2.

The fuzzy control rules examine all the possible combinations of the fuzzy linguistic input variables. An input parameter is allowed to belong to two fuzzy sets at most. With n input parameters, there are 2^n applicable rules.

In this navigation system, the rules are:

IF obstacle is NEAR AND ZERO POSITIVE THEN move MEDIUM AND MEDIUM NEGATIVE

IF obstacle is MEDIUM AND ZERO POSITIVE THEN move NEAR AND MEDIUM NEGATIVE

IF obstacle is NEAR AND MEDIUM POSITIVE THEN move MEDIUM AND NEAR NEGATIVE

IF obstacle is MEDIUM AND MEDIUM POSITIVE THEN move NEAR AND MEDIUM NEGATIVE

IF obstacle is NEAR AND ZERO NEGATIVE THEN move MEDIUM AND MEDIUM POSITIVE

IF obstacle is MEDIUM AND ZERO NEGATIVE THEN move NEAR AND MEDIUM POSITIVE

IF obstacle is NEAR AND MEDIUM NEGATIVE THEN move MEDIUM AND NEAR POSITIVE

IF obstacle is MEDIUM AND MEDIUM NEGATIVE THEN move NEAR AND MEDIUM POSITIVE

(2) Target Steering Behaviour. In Obstacle Avoidance Behaviour, the robot follows an escape vector leading to a temporary target. Target Steering Behaviour helps the robot not to lose sight of its main Target so when there are no obstacles around the robot, the main reactive behaviour is target steering.

IF obstacle is LARGE POSITIVE OR LARGE NEGATIVE OR FAR THEN move toward Main Target

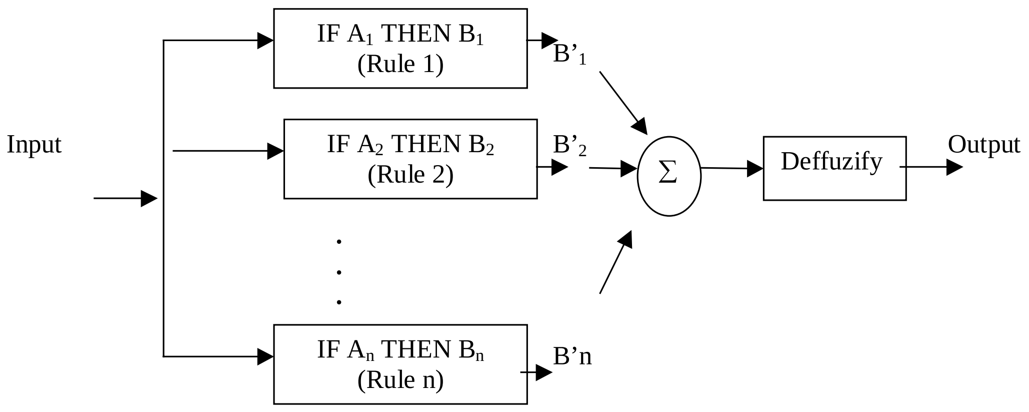


Figure 4 Fuzzy rule System Architecture

After fuzzification the resulting fuzzy values are subjected to a stage called inference where the fuzzy rules are applied (Figure 4). Finally, The defuzzification interface converts the fuzzy output into a crisp (nonfuzzy) value which is then used to generate a modified driving (or escape) vector.

In this approach, the Centre of Area (COA) method is used. It is a way of calculating the centre of gravity of the resultant fuzzy set after inference.
Arithmetically, for fuzzy solution region A, this procedure is formulated as:

$$R = \frac{\sum_{i=0}^{n} d_i . \mu_A(d_i)}{\sum_{i=0}^{n} \mu_A(d_i)} \qquad \text{(Equation 1)}$$

Where R represents the crisp output variable, d is the ith domain value, and $\mu_A(d)$ is the truth membership value for that domain point[10].

The above controller does function correctly but it does not necessarily have optimal performance. The main reason for this non-optimal behaviour is that the performance of the system depends to large extent on the shape and membership of the fuzzy sets. These sets are derived heuristically and may therefore themselves be non-optimal. A way of improving this is to use a genetic algorithm to modify the sets through an evolutionary process until an optimal set is produced.

5.3. ***Optimisation of the fuzzy controller.*** One of the central factors in the design of efficient and robust fuzzy logic controllers is the selection of the membership rules and parameters. The existing approaches for choosing the membership functions are based on trial-and-error process, and lack learning and autonomy. One method of removing the uncertainty associated with the selection of these variables is the use of genetic algorithms (GA)[11]. These have been widely used in a variety of problems especially in machine learning and function optimisation[12].
GAs were created by John Holland in the 1960's[13] and proved capable of solving complex problems in large scale of applications.
These are based on a survival of the fittest principle and have been applied to fuzzy logic controllers design for searching the membership function and rule space in order to improve performance[14],[15].

The scope of this work covers only optimisation of fuzzy set parameters. Triangular fuzzy sets have three parameters to find (a1,b1,c1) and trapezoidal sets have four (a2,b2,c2,d2) (Figure5).

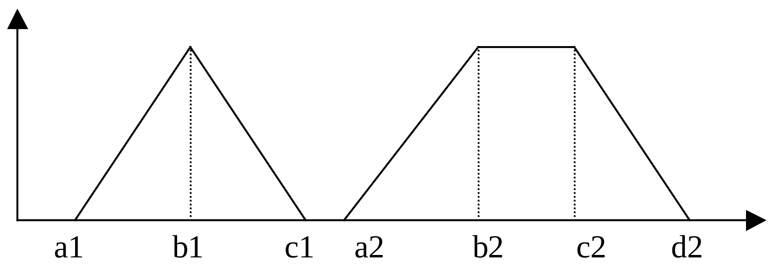


Figure 5 Fuzzy set parameters

Distance fuzzy parameters are scaled by a scaling factor equal to 25(Figure 2). Dimensions like 100 and 250 are represented by 4 and 10.
Direction fuzzy parameters are scaled by 15 (Figure 3), and values like –90, -75, -60, -45, -30, -15, 0, 15, 30, 45, 60, 75, and 90 are represented by –6, -5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5, and 6.

In the algorithm used here each chromosome solution is represented as a series of genes of 1s and 0s where a defined set of three binary genes is decoded representing one parameter of the direction fuzzy logic controller. Likewise, a defined set of four binary genes is decoded representing parameters of the distance fuzzy logic controller

a1	b1	c1	…	…	…	a2	b2	c2	d2	…	…	…	…	….

Figure 6 Chromosome of genes

Once the parameters have been encoded the fitness of the resulting chromosomes is judged by a fitness function applied to numerically rate the effectiveness of each

chromosome in the population. In this navigation application the fitness function evaluates the robot's path taking into account distance travelled and clearance from obstacles. Any path that leads a robot to collide with an obstacle is aborted and claimed infeasible.
After applying the fitness function the principles of crossover and mutation are applied. Crossover creates a new chromosome by combining two existing chromosomes with high fitness ratings and mutation occasionally randomly change the value of a gene in a chromosome; a 0 is changed to a 1 and vice versa.
The genetic algorithm used in this paper uses the concept of non-overlapping populations. In each generation, the entire population is replaced with new individuals.
After applying the fitness function, two principles of crossover and mutation are applied, where:
Crossover: create a new chromosome by combining two existing chromosomes with high fitness ratings and selected using the weighted roulette wheel principle. Probability of crossover is 0.7
Mutation: occasionally randomly change the value of a gene in a chromosome, a 0 is changed to a 1 and vice versa. Probability of mutation is 0.01.

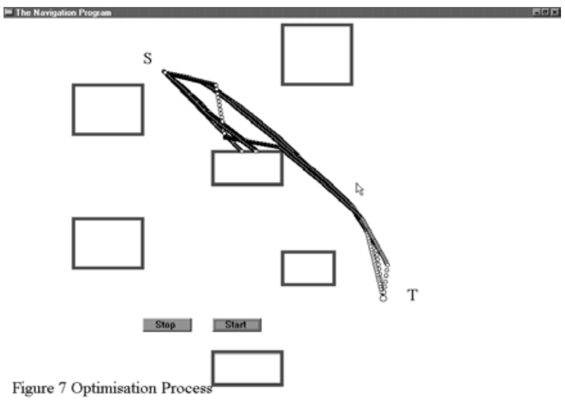


Figure 7 Optimisation Process

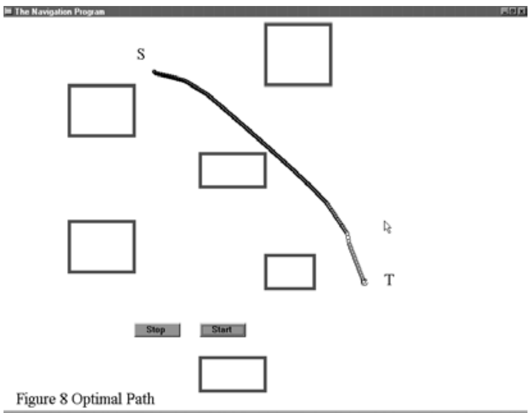


Figure 8 Optimal Path

50 generations are to be generated where each generation has a population of 20 chromosomes. The initial generation is generated randomly and tested. Then, best parents are chosen from the initial populations. Selected parents produce children with the number weighted by their individual fitness. Crossover and Mutation operations are applied. The fitness value is calculated for each population in all different generations. Process is repeated until a number of iterations is reached.
Figure 7 shows some instances of the optimisation process. Results of best fitness values of each generation is stored in a text file with the corresponding chromosome. The best chromosome is decoded and used it to design the final FLC (Figure 8).

6. **Conclusion.** This abstract describes a real time robot control system. The levels described have been tested using both simulation and a real robot. The use of a layered control structure provides an easily expandable structure to which pre-existing and newly developed functions can easily be added. An optimised Navigation layer has also been developed which uses a fuzzy logic controller which has been optimised using a genetic algorithm.

7. References

1 E. S. Tzafetas, J.D. Mathieu, and S. Tzafetas, “Overview of Experimental Evaluation of Potential Field-Based Robot Navigation,” Advanced Robotics beyond 2000: The 29th International Symposium on Robotics, N.E.C., Birmingham, UK, 27th April 1998.
2 C. Czarnecki, R. John, and S. Bennett, ‘The Application of Fuzzy Logic to Real Time Multiple Robot Collision Avoidance Internet, Proceedings ICSC Fuzzy Logic Symposium 1995, p. C116-C121.
3 R.A. Brooks, “A robust layered control system for a mobile robot” in IEEE Journal of Robotics, Vol. 2, No. 1, March 1986, pp. 14-23
4 P. Webb and J.I. Robson, “An Intuitive Control and Sensory System for the Enhancement of Remotely Controlled Vehicle Operation”, Proc. Mechatronics 1998, 1998.
5 R.A. Brooks, “A robust layered control system for a mobile robot,” IEEE J. of Robotics and Automation, RA-2, PP.14-23, April 1986.
6 T. Wang and B. Zhang, “Time varying potential field based ‘perception-action’ behaviours of mobile robot’. Proc. IEEE Int. Conf. On Robotics and Automation, pp.2549-2554, May 1992
7 Saffiotti A., “The Uses of Fuzzy Logic in Autonomous Robot Navigation: a catalogue raisonne”, Soft Computing 1(4) 180-197, Springer-Verlag.
8 Li W, Ma C and Wahl F, “A Neuro-fuzzy system architecture for behaviour-based control of a mobile robot in unknown environment”, Fuzzy Sets and Systems 87 (1997) 133-140.
9 Ben-Arieh D, Fuzzy Logic Controller for Part Routing, 5th Industrial Engineering Research Conference Proceedings
10 Cox, Earl. The fuzzy systems handbook : a practitioner's guide to building, using, and maintaining fuzzy systems, 2nd ed. c1999

11 Leitch D, Probert PJ, New techniques for genetic development of a class of fuzzy controllers, IEEE Transactions on systems man and cybernetics, Part C-Applications and reviews, 1998, Vol.28, No.1, pp.112-123.
12 D. E. Goldberg. *Genetic Algorithm in Search, Optimization, and Machine Learning*. Addison-Wesley, Reading, MA, 1989.

13 Holland J, Genetic Algorithms, Scientific American, Vol. 267, No. 1, July 1992, pp 44-50.

14 Frank Hoffmann, Gerd Pfister, A new Learning Method for the Design of Hierarchical Fuzzy Controllers Using Messy Genetic Algorithms, IFSA'95, Sao Paulo

15 Frank Hoffmann and Gerd Pfister, Automatic Design of Hierarchiacl Fuzzy Controllers Using Genetic Algorithms, EUFIT'94, 20-23, September 1994