Prototyping, Design and Learning in Outdoors Mobile Robots Operating In Unstructured Environments

Hani Hagras, Victor Callaghan, Martin Colley

Department of Computer Science, University of Essex, Wivenhoe Park, Colchester, CO4 3SQ, England, UK

Email: hani@essex.ac.uk

Outdoor environments are particularly challenging to mobile robots as they offer dynamic, unstructured and highly variable situations where the inconsistency of the terrain, the irregularity of the product and the open nature of the working environment result in complex problems of identification, modelling, sensing and control. Problems can range from the effects of varying weather and ground conditions on vehicle sensors and traction performance, through to the need to deal with the presence of unauthorised people and animals. Another major challenge to autonomous robotics is the uncertainty that characterises real-world environments. This is why it is difficult, or effectively impossible, to produce simulations of outdoor robots which are sufficiently true to real-dynamics to learn good enough controllers.

In this paper we are going to present our methodology for using small lab based indoor robots to prototype large outdoor vehicles in agriculture environment. The indoor prototype robot uses our patented techniques (British Patent number “99 10539.7”) which enables rapid online and autonomous learning of controllers.

Introduction

Broadly speaking, our work situates itself in the recent line of research that concentrates on the realisation of artificial agents strongly coupled with the physical world. A first fundamental requirement is that such agents must be grounded in that they must be able to carry on their activities in the real world, in real time [1]. Another important point is that adaptive behaviour cannot be considered as a product of an agent considered in isolation from the world, but can only emerge from strong coupling of the agent and its environment [2]. Despite this, many robotics researchers regularly use simulations to test their models. However, the validity of such computer simulations to build autonomous robots is frequently criticised and is the subject of much debate. Even so computer simulations may still be very helpful in the training and testing of robotics models. However as Brooks [3] pointed out “it is very hard to simulate the actual dynamics of the real world”. This may imply that effort will go into solving problems that simply do not arise in the real world with authentic robots and that programs which work well on simulated robots will completely fail on actual robots.

There are several reasons why those using computer models (simulations) to develop control systems for real robots operating in outdoor environments may encounter problems [4]:

a) Numerical simulations do not usually consider all the physical laws of the interaction of a real agent with its own environment, such as mass, weight, friction, inertia, etc.
b) Physical sensors deliver uncertain values, and commands to actuators have uncertain effects, whereas simulative models often use grid-worlds and sensors that return perfect information.
c) Physical sensors and actuators, even if apparently identical, may perform differently because of slight variations in the electronics and mechanics or because of their different positions on the robot or because of the changing weather or environmental conditions.

Even where researchers are using real robots to learn behaviors, these behaviors if learnt successfully are usually frozen in the robot. Thus if some of the robot dynamics or the environmental circumstances are changed, the robot must repeat a time-consuming learning cycle [4].

From the above discussion it is clear that using computer simulations for developing prototypes for robot controllers that operate in outdoor environment has significant disadvantages which is best illustrated by the fact that when transferring the learnt controllers from the simulated world to real outdoor changing world these controllers will usually fail [4]. However, developing a prototype based on small-scale, but real robots provides much of the reality missing form simulation (e.g. noise and imprecision associated with sensors and actuators) thereby developing controllers that takes such defects into account, producing a more realistic controller. After learning the preliminary controllers on small-scale robot prototypes (tested under hard conditions simulating the outdoor conditions) the controllers can then transferred to the outdoor target vehicles, requiring less modification than a simulated counterpart, and normally well within the intrinsic adaptation capability of the controller. Thus the outdoor target vehicles then run the adaptation technique to adjust the learnt indoor controllers to their specific kinematics and dynamics rather than repeating the learning cycle from the beginning. This prototyping technique allows us to learn controllers that do dangerous manoeuvres (like helicopters and underwater vehicles) in a controlled situation rather than trying the whole learning cycle in outdoor expensive robots which might cause harms to the outdoor robots during learning. In addition, the same the adaptation techniques can adapt the robot to any environmental or kinematics changes that can occur over the life of the vehicle thus producing a continuos learning system.

Fuzzy logic Control (FLC) has become a popular approach to reactive robot control in recent years [5]. Given the uncertain and incomplete information an autonomous robot has about the environment, fuzzy rules provide an attractive means for mapping sensor data to appropriate control actions in real time. The success of fuzzy control is owed in a large part to the technology's ability to convert qualitative linguistic descriptions into complex mathematical functions. The methodology of the FLC appears very useful when the processes are too complex for analysis by conventional quantitative techniques or when the available sources of information are interpreted qualitatively, inexactly or uncertainly, which is the case of mobile robots. However fuzzy controllers have the problem that their parameters are usually determined by human designers using a trial and error approach. Also as the number of input variables increases (which is the case of mobile robots) the number of rules increases exponentially which creates much difficulty in determining large numbers of rules.

Evolutionary algorithms constitute a class of search and optimisation methods guided by the principles of natural evolution and genetics. It is the case that Genetic Algorithms (GA) have been successfully applied to solve a variety of difficult theoretical and practical problems by imitating the underlying processes of evolution such as selection, recombination and mutation. The GA based approach is a well accepted technique for enabling systems to adapt to different control tasks [6] and there is much work reported in the literature on designing fuzzy controllers using GA [7, 8, 4, 9, 10, 12, 13, 14, 2]. However virtually all this work was undertaken using simulation as, in conventional GA, it takes a large number of iterations to develop a good controller. Thus it is not feasible for a simple GA to learn online and adapt in real-time. The situation is worsened by the fact that most evolutionary computation methods developed so far assume that the solution space is fixed (i.e. the evolution takes place within a pre-defined problem space and not in a dynamically changing and open one), thus preventing them from being used in real-time applications [15]. Hence prior to our work it was *not considered feasible for a simple GA to online learn and adapt a robotic controller* [9] in outdoor unstructured environments.

In the next section we will introduce the target environment (outdoor farm tasks) and the experimental target prototype vehicles. We will describe our patented Fuzzy-Genetic techniques for learning and control, explaining how it is used to learn fuzzy rules and membership functions for different behaviours and co-ordination mechanisms so as to fulfil specified missions. We then present the online adaptation technique and the experiments and results.

The Targeted Environment

Providing vehicle operators with a degree of automation to relieve them from some of the more repetitive tasks is seen to be of benefit by many companies. Indeed, prototype driverless cars have already navigated public roads and several of the major vehicle manufacturers are suggesting that self-guiding cars will be commercially available for motorway operation by 2010 [16]. Road environments are relatively well structured and well defined and have regular geometry. In this work we move to the less structured environment of a farm; addressing the control of autonomous or semi-autonomous agricultural vehicles. In particular we are interested in the problem how to adapt the vehicle control algorithms as the environmental conditions change.

Our prototyping strategy will lead to the implementation of an outdoor vehicle that is able to 'follow' a path or maintain a heading through a dynamic, unstructured environment, using navigational data from imprecise sensors. A dynamic unstructured environment can be categorised as unpredictable and constantly changing, requiring the vehicle control system to be adaptive so as to maintain correct operation. Within the agricultural environment the inconsistency of the terrain, the irregularity of the crop and the dynamic, open nature of the working environment give rise to complex problems of identification, sensing and control. The main advantage of the fuzzy logic approach is that it enables the controller to accommodate the approximate, imperfect and noisy information presented in real-world environments, whilst still producing smooth control outputs. While fuzzy systems are well suited to deal with such control, their parameters tend to be derived in advance from prior knowledge about the environment. Any large change in the environment, outside the range of the pre-determined rules, can cause the incorrect operation of a fuzzy logic controller (e.g. changes lead to the use of the wrong rule). We have chosen the agricultural environment for our initial work, as this offers a most severe test of our methods, providing a useful benchmark for our achievements.

In the experimental work we will show the generality and robustness of our techniques by demonstrating that they can be ported, with minimal changes, between the prototyping robot (operating indoors) and various sizes target robots (mostly operating outdoors). As mentioned earlier, the proposed algorithms are adequate for outdoor robots where the dynamics of the robot as well as the environmental conditions are rapidly changing and require the robot to quickly modify itself to these changes. This work is particularly useful for producing intelligent machines for inaccessible environments where reprogramming robots would be difficult or expensive. We will show how such an approach both saves money and increases reliability by allowing the robot to automatically adapt without further programming to the changing user and environmental needs it will experience throughout its lifetime.

The Experimental vehicles

Both the prototype and target robots are based around a distributed field bus control system. This offers many advantages, for example, distribution enables the processing to be sized cost-effectively across a range of vehicle applications (e.g. processing power is added in proportion to sensor quantity and type). In particular, we use the CANbus (Controller Area Network) developed for automotive industry, Motorola processors and the Tornado/VxWorks Real Time Operating System (RTOS). The current design is influenced largely by the real-time nature of the problem (e.g. use of both parallel and distributed processing).

The robot controller is first learnt and tested in an indoor prototype robot and then moved to an outdoor target robot (each of which has different physical shapes and different sensor/effector characteristics). The designed system parameters are robot-independent and the architecture is portable between the different sizes of robot.

The Indoor Prototype Robots

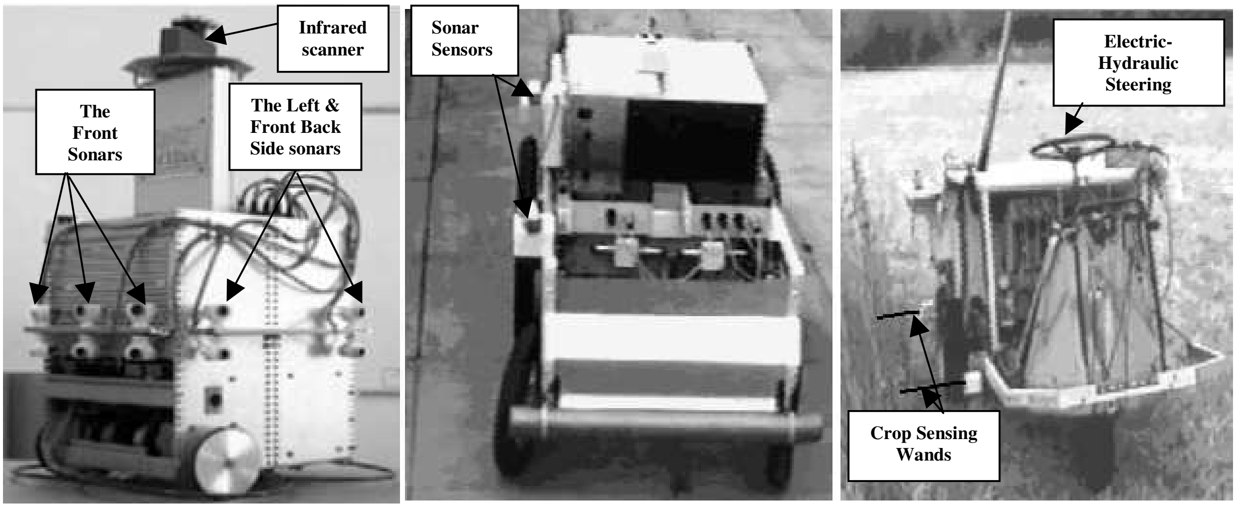
The indoor robot has a ring of 7 ultrasonic proximity detectors, an 8-axis vectored bump switch, an IR beacon locator (to aid navigation) plus two independent stepper motors for traction and differential steering. The hardware is based on embedded Motorola 68040 processors running the VxWorks RTOS. In the initial experiments, infrared beacons were used to simulate the goals. Control programs were developed using the Tornado VxWorks cross-development environment which provides seamless workstation-robot development via ethernet interconnection. The network can be disconnected for autonomous operation. The robot and its sensor configuration are shown in Figure (1-a).

The Outdoor Target Robots

The outdoor target robots are of two types; an electric and a diesel powered vehicle. Both robots have a mechanism for determining distance to nearby objects. Ultrasound has proved convenient, being well able to cope with farm needs such as detection of plants and crops. However, the system is essentially independent of sensor type and has given good results with a mechanical wand (arms connected to digitised potentiometers to sense the edge of a crop). The approach we use allows proximity sensors to be exchanged with little or no change to the control system. In the future, we plan to equip the robots with GPS, compass and vision for goal determination. All our robots use a similar computational architecture, to simplify development work. The hardware is based on embedded Motorola processors (68040) running VxWorks RTOS. The diesel powered robot has one actuator for controlling the speed of the front wheels and another for controlling the steering angle of the front wheel. As with other farm vehicles, the diesel robot uses its engine to provide both traction and electrical power (including battery charging). The electrical and diesel robot and their sensor configurations are shown in Figure (1-b) and (1-c) respectively.

Hierarchical Fuzzy Logic Controllers (HFLC)

Most commercial Fuzzy Logic Control (FLC) implementations feature a single layer of inferencing between two or three inputs and one or two outputs. For autonomous vehicles, however the number of inputs and outputs are usually large and the desired control behaviours are more complex. Our indoor prototyping vehicle experiments use eight inputs (7 sonar inputs and an infrared bearing sensor) and two outputs (left and right wheel speed). If we assume that each input will be represented by three fuzzy sets and each output by four fuzzy sets, using a single layer of inferencing will lead to determining $3^8 = 6561$ rules which would be difficult, if not impossible, to determine.



(a) (b) (c)

Figure (1): Prototype robot (indoor) . b) Target robot(outdoor-electrical) . c) Target robot (outdoor-diesel)

However, by using a *hierarchical* assembly of fuzzy controllers, the number of rules required can be significantly reduced. For example, the experimental system can be divided into four co-operating behaviours, obstacle avoidance, left and right wall following and goal seeking. If, as before, these behaviours represent each input by three fuzzy sets then obstacle avoidance with three inputs produces $3^3 = 27$ rules. The left wall following has two inputs producing $3^2 = 9$ rules, right wall following is the same and goal seeking has one input (more accurately represented by seven fuzzy sets) producing 7 rules. Thus the total number of rules now required is 27 + 9 + 9 + 7 = 52 rules which is much easier to determine. To use such a hierarchical mechanism, a co-ordination scheme is required to combine these behaviours into a single action. Saffiotti [5] has suggested a fuzzy context rule combination method to perform the high level co-ordination between such behaviours. The context dependent rules are characterised by each behaviour generating *preferences* from the perspective of its goal. Each behaviour has a context of activation, representing the situations where it should be used. The preferences of all behaviours, weighted by a true value of their contexts, are fused to form a collective preference. One command is then chosen from the collective preference.

We use a variant of the method suggested by Saffiotti [5] and Tunstel [17]. In this we apply fuzzy logic to both implement the individual behaviour elements and the related arbitration (allowing both fixed and dynamic arbitration policies to be implemented) [18]. To achieve this we implement each behaviour as an independent FLC aimed at a simple task, (e.g. edge following or obstacle avoidance) with a resultant small set of inputs and outputs to manage. We chose fuzzy logic to implement the basic behaviours, as it excels in dealing with the kind of imprecise and uncertain knowledge associated with a robot's sensors and actuators.

The outputs of each fuzzy behaviour are fused according to directions supplied by a high-level planner, which may be a person. This fusion itself is a fuzzy process in which different behaviours are co-ordinated to give a coherent output. The high-level planner can also define situations and sequences in which individual behaviours are active, commonly referred to as "willed" operation. The combination of these methods produce a system that is capable of completing complicated tasks from a set of simple tasks which can be more easily designed than more monolithic alternatives. Fuzzy co-ordination facilitates expression of partial and concurrent activation of behaviours, thereby allowing behaviours to be active concurrently to differing degrees which gives a smoother control characteristics than switched counterparts [5]. As mentioned earlier, using a hierarchical strategy results in much fewer rules (i.e. much simplified design problem) [5]. In addition, it allows flexible modularised design where new behaviours can be added easily and different tasks achieved by changing the co-ordination parameters (either in willed or automatic mode). Our use of a fuzzy arbitration mechanism, and the flexibility arising, is significant addition to earlier work such as the subsumption architecture [3].

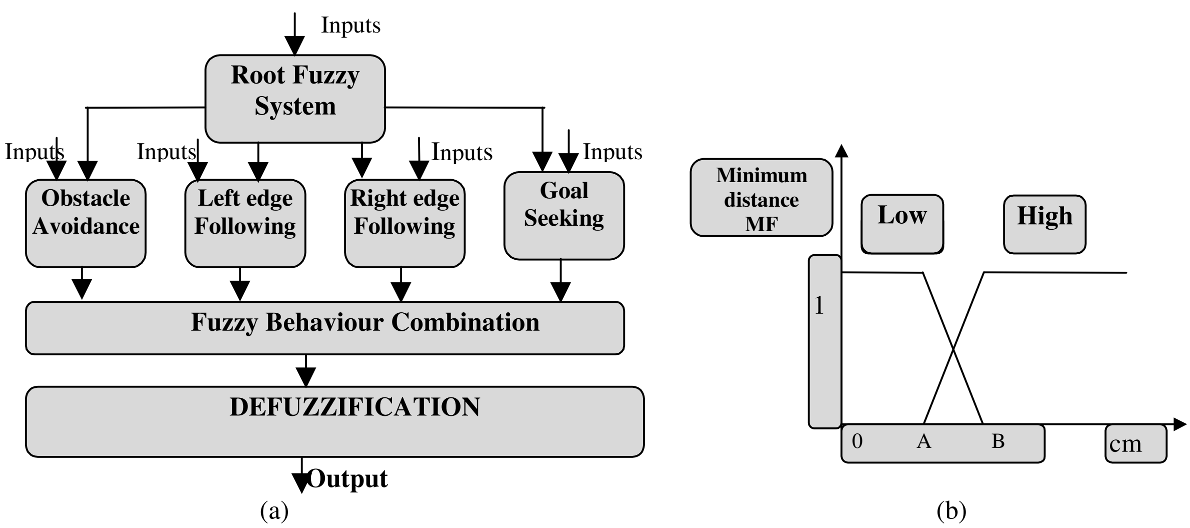


Figure (2): a) Behaviour co-ordination system. b) Membership function for co-ordination parameters.

In our design each behaviour uses a FLC using singleton fuzzifier, triangular membership functions, product inference, max-product composition and height defuzzification. The selected techniques were chosen due to their computational simplicity and real-time considerations, more information about fuzzy logic can be found in [11].

$$Y_t = \frac{\sum_{p=1}^{M} y_p \prod_{i=1}^{G} \alpha_{Aip}}{\sum_{p=1}^{M} \prod_{i=1}^{G} \alpha_{Aip}} \tag{1}$$

Where M is the total number of rules, y_p is the crisp output for each rule, $\Pi\alpha_{Aip}$ is the product of the membership functions for each rule's inputs and G is the number of inputs.

Ruspini [19] defines fuzzy command fusion as the interpretation of each behaviour-producing unit, acting as an agent, expressing preferences as to which command to apply. Degrees of preferences are represented by a probability distribution (fuzzy in our case) over the command space. In our HFLC architecture a fuzzy operator is used to combine the preferences of different behaviours into a collective preference. Accordingly, command fusion is decomposed into two steps: preference combination and decision-making. In Figure (2-a) each behaviour is treated as an independent fuzzy controller. Using fuzzy behaviour combination, we obtain a collective fuzzy output which we then defuzzify to obtain a final crisp output. Fuzzy meta-rules or context rules enable *more flexible arbitration* policies to be achieved. These rules have the form IF context THEN behaviour [5] which means that a behaviour is activated with a strength determined by the context (i.e. a fuzzy-logic formula). When more than one behaviour is activated, their outputs are fused and each behaviour output scaled by the strength of its context.

In case of using fuzzy numbers for preferences, product-sum combination and height defuzzifcation are used. The final output equation, provided by Saffiotti [5], is given below:

$$Y_{ht} = \frac{\sum_i (mm_y * y_t)}{\sum_i mm_y} \tag{2}$$

Where i represents the behaviours activated by context rules which can be; right/left edge-following behaviour, obstacle-avoidance or goal seeking. Y_t is the behaviour command output (left and right velocity in case of prototyping indoor robots, steering and wheel speeds in case of the target outdoor robots). These vectors are fused in order to produce a single vector Y_{ht} to be applied to the mobile robot. mm_y is the behaviour weight.

The behaviours implemented in this system constitute the minimum set of behaviours needed to demonstrate the architecture working in an outdoor environment. From other work [3], we know that four behaviours are sufficient to complete the tasks we have in mind; namely, goal-seeking, obstacle-avoidance, right edge-following and left edge-following.

In behaviour co-ordination there are a few parameters that must be calculated in the root fuzzy system. These are the minimum distance of the front sensors (represented by d1), the minimum distance of the left side sensors (represented by d2), the minimum distance of the right side sensors (represented by d3). The minimum of the fuzzy MF of d1, d2, d3 represented by d4, which reflects how obstacle free the robot path is. After calculating these values, each is matched to its membership function as shown in Figure (2-b). These fuzzy values are used as inputs to the context rules which are:

IF d1 IS LOW THEN OBSTACLE AVOIDANCE,
IF d2 IS LOW THEN LEFT WALL FOLLOWING,
IF d3 IS LOW THEN RIGHT WALL FOLLOWING,
IF d4 IS HIGH THEN GOAL SEEKING.

The context rules determine which behaviour is fired, and to what degree. The final output is calculated using Equation (2). The behaviour weights are calculated dynamically taking into account the situation of the mobile robot. For example, the obstacle avoidance behaviour weight needs to increase as the obstacle comes closer. This can be done by calculating the minimum distance of the front sensors d1 and then calculating the weight of the obstacle avoidance behaviour using the membership functions in Figure (2-b). Then, by using the context rules we can determine which behaviours are active and apply Equation (2) to obtain the final output.

Hierarchical Fuzzy Genetic Systems for Online Learning and Adaptation

According to Kasabov [15] an Intelligent System (IS) should be able to learn quickly from large amounts of data. He also sates that an intelligent system should also adapt in a real time and in an on-line mode as new data is encountered. Also the system should be able to accommodate in an incremental way any new problem solving rules as they become known. It should be memory-based, plus possess data and exemplar storage and retrieval capacities. In addition, he says that an IS should be able to learn and improve through active interaction with the user and the environment. It should have parameters to represent short and long term memory, age, forgetting, etc. Finally he states it should be able to analyse itself in terms of behaviour, error and success. To our knowledge, no system in the field of outdoor robots had satisfied these criteria.

To achieve a high-level behaviour via manipulating rule-bases with a GA, it is necessary first to develop low level competence, which can then be modified and enhanced to produce an emergent high-level behaviour. This can be done at a low level by manipulating the individual behaviours, and at a higher level by manipulating combinations of the lower-level behaviours (the latter occurring once a reasonable degree of proficiency has been attained, to form a controller capable of performing a higher-level task). For this, a simple objective function is sufficient, as the controllers would already have a degree of competence for the behaviour desired. Such a solution can be achieved by implementing a hierarchical learning procedure [7].

In our hierarchical learning procedure we start using the indoor prototype based on the use of working (but not necessarily optimum) set of fixed membership functions We then commence learning general rules in each individual behaviour by relating the input sensors to the actuator outputs. In this phase the membership values are not important as the robot learns general rules such as, *if the obstacle is close then turn left.* However in order to achieve a sub-optimal solution for the individual behaviour (a subset of the large search space) we need to find next the most suitable membership functions for the learnt rules. After finding a sub-optimal solution for each behaviour we combine these behaviours and learn the best co-ordination parameters that will give a "good enough" solution for the large search space to satisfy a given mission or plan. After learning the system parameters, the controller is then transferred to the outdoor vehicles where the online adaptation technique is triggered to adjust the learnt controller on the prototype robots to the outdoor robots needs. Also if the controller fails to maintain the desired states, the adaptation technique modifies the poor rules in the relevant behaviours to adjust to the outdoor target vehicle's differing environmental and kinematics conditions without the need to restart the learning cycle.

This hierarchical procedure results in a fast learning time for finding a solution for learning and adaptation in outdoor environments as will be demonstrated in the experiments section. In particular, it learns controllers for the outdoor robots which might involve dangerous manoeuvres (like helicopters and underwater vehicles) in a controlled indoor robots, thus avoiding the risks associated with online learning using heavy and expensive outdoor robots. Also our learning techniques learn general controllers that can be applied to different outdoor robots performing the same mission and applying a short adaptation cycle thus saving the need to learn a new controller for each different outdoor robot.

Figure (3) provides an architectural overview of our solution, which we term an *Associative Experience Engine.* This forms the learning engine within the control architecture and is the subject of British patent application 99-10539.7. The behaviours are represented as parallel Fuzzy Logic Controllers (FLC) and form the hierarchical fuzzy control architecture presented in the previous section. Each FLC has two modifiable parameters, the *Rule Base* (RB) for each behaviour and the *Membership Functions* (MF). The behaviours receive their inputs from sensors. The output of each FLC is then fed to the actuators via the *Co-ordinator,* which weights their effect. When a behaviour, or collection of behaviours, fail to respond correctly to a situation, a learning cycle is initiated.

For the indoor prototype robot the learning cycle performed is dependent upon the *Learning Focus,* which is supplied by the coordinator according to a higher-level plan. For example, if the *Learning Focus* is to learn the MF for individual behaviors, then the input membership functions of each behavior are learnt alone.

When learning or modifying the rule bases, the learning cycle is sub-divided into local situations. This reduces the size of the model to be learnt. The accent on local models implies the possibility to learn by focusing at each step on small parts of the search space only. The interaction among local models, due to the intersection of neighboring fuzzy sets causes the local learning to reflect on global performance [20]. To further reduce the search space, the system determines if it had encountered similar situations before by checking the stored experiences in the *Experience Bank*. The robot tests different solutions from the experience bank by transferring the "remembered" experiences, which are stored in a queue to the corresponding behaviors. If any of these experiences show success, then they are stored in the FLC and we avoid generating a new solution for our system. An *Experience Assessor* assigns each experience solution a fitness value to indicate the importance of this solution. When the Experience bank becomes full it is the role of the *Experience Survival valuer* to determine which parameters are retained and which are discarded, according to the parameter importance. If the use of the past experiences did not solve the situation, we use the highest fitness experience as a starting point for the new learning cycle. We then fire an *Adaptive Genetic Algorithm (AGA*) mechanism using adaptive crossover and mutation parameters which help to speed the search for new solutions. The AGA is constrained to produce new solutions in a certain range defined by the *Contextual Constraints* supplied by sensors and defined by the coordinator according to the *learning focus.* This avoids the AGA searching places where solutions are not likely to be found. By doing this we narrow the AGA search space to where we are likely to find solutions. The AGA search converges faster as it started from a good point in the search space supplied by the experience recall mechanism and it used adaptive learning parameters and avoided searching regions where solutions are not likely to be found. After generating new solutions wither rules or MF or coordination parameters the system tests the new solution and gives it fitness through the *Solution Evaluator* to be discussed next section. The AGA generates new solution until reaching a satisfactory solution.

The online learning mechanism, in addition to the fuzzy behaviours, is also organised as a hierarchy thus leading to one description of this architecture as being a "double-hierarchy". The online learning mechanisms can be regarded as a hierarchy because there is a tiered set of actions. At the highest level a population of solutions are stored in the *Experience Bank* and tested in a queue. If one of these stored experiences leads to a solution then the search ends, if none of these stored experiences leads to a solution then each of these experiences acquires fitness by the *Experience Assessor* depending how well each solution performed in the situation. The highest fitness experience is used as a starting position to the lower level GA that is used to generate new solutions to the current situation. This hierarchy preserves the system experience, and speeds up the genetic search by starting the genetic algorithm from the best found point in the space.

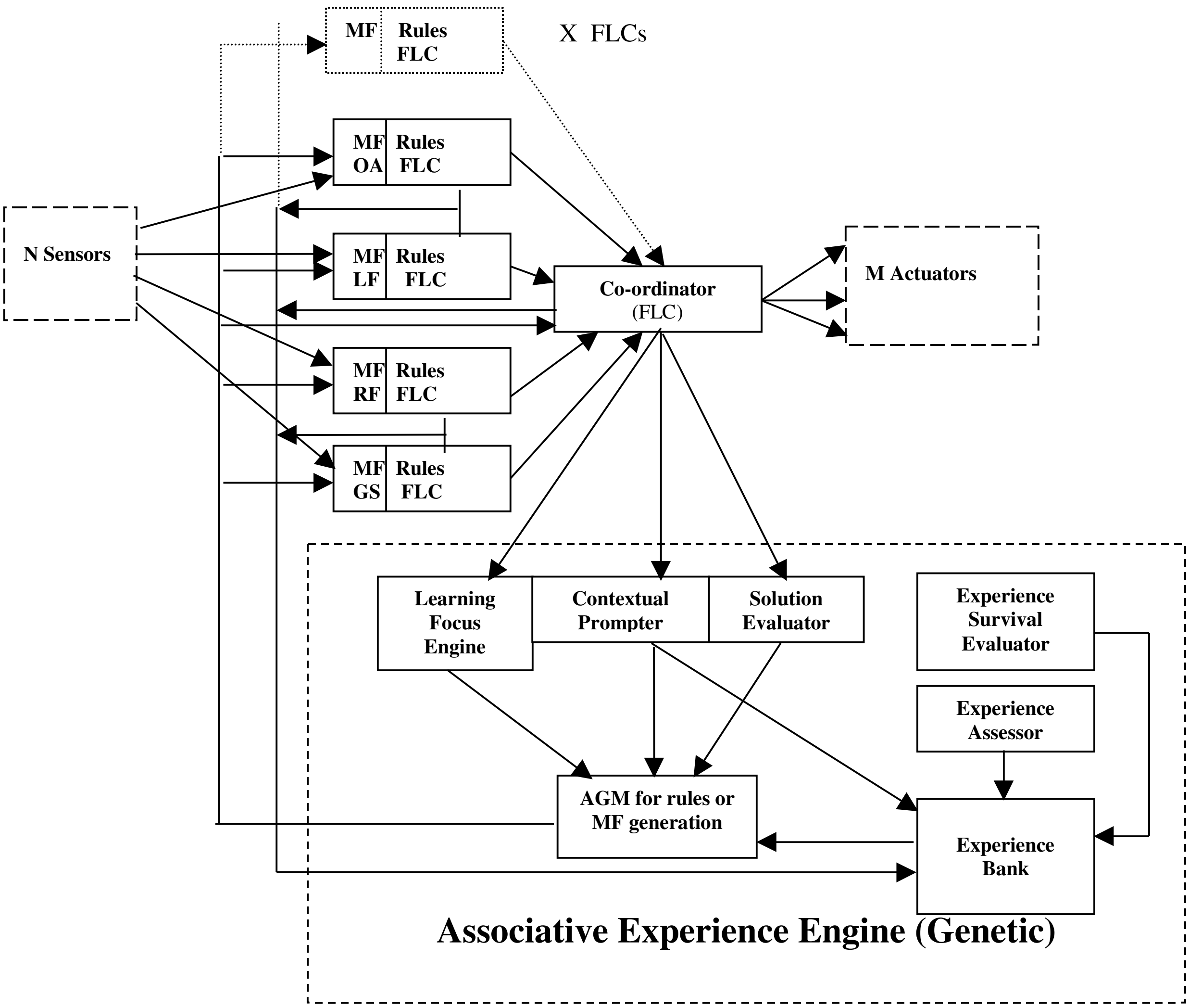


Figure (3): Architectural Overview of Associative Experience Learning Engine (UK patent No 99-10539.7)

Learning General Behaviours Rules

The rule-base of the behaviour to be learnt is initialised randomly. The designer supplies a preliminary input MF for each behaviour. As was explained earlier the values of the membership functions are not important as we are seeking general rules. In the following sections we will introduce the various steps of the algorithm to learn the rule base of behaviours that receive immediate reinforcement such as edge following and goal seeking. For learning the rules of behaviours that receive delayed reinforcement please refer to [21].

After the rule-base initialization, the indoor prototype robot starts moving. If the rule-base contains poor rules then it will begin deviating from its objective. In this case our algorithm is fired to generate new set of rules to correct this deviation. The GA population consists of all the rules contributing to an action (which is usually a small number as the rules base of each behavior consists only of 9 rules). As the case with classifier systems, in order to preserve the system performance, the GA is allowed to replace a subset of the classifiers (the rules in our case). The worst m classifiers are replaced by the m new classifiers created by the application of the GA on the population [14]. The new rules are tested by the combined action of the performance and apportionment of credit mechanisms. In our case, only two rule actions will be replaced (those already identified with being predominantly responsible for the error).

The system fitness is determined by the *Solution Evaluator* and is evaluated by how much it reduces the normalised absolute deviation (d) from the normal value as well as maintaining both a high speed and a straight heading. This is given by:

$$d = \frac{| normal.value - deviated.value |}{\max .deviation} \qquad (3)$$

Where the normal value will correspond to the value that gives the maximum normal membership function. For example, 40 cm in case of edge following or zero degrees in case of goal seeking (as was specified by the human designer of the membership functions). The deviated value is any value deviating from the nominal value. The maximum deviation corresponds to the maximum deviation that can occur as was specified by the human designer membership functions using indoor robots.

The fitness of the solution is given by d1 - d2, where d2 is the normalised absolute deviation before introducing a new solution, and d1 is the normalised absolute deviation following the new solution. The deviation is measured using the robot's sensors, which gives the robot the ability to adapt to the imprecision and noise found in the real sensors rather than relying on estimates from previous simulations.

The fitness of each rule at a given situation is calculated as follows. We can write the crisp output Y_t as in (1). If the robot has two output variables, then we have Y_{t1} and Y_{t2}. The normalised contribution of each rule p output (Y_{p1}, Y_{p2}) to the total output Y_{t1} and Y_{t2} can be denoted by S_{r1}, S_{r2} where S_{r1} and S_{r2} is given by:

$$\mathbf{S_{r1}} = \frac{\dfrac{Y_{p1}\prod_{i=1}^{G}\alpha_{Aip}}{\sum_{p=1}^{M}\prod_{i=1}^{G}\alpha_{Aip}}}{Y_{t1}}, \qquad \mathbf{S_{r2}} = \frac{\dfrac{Y_{p2}\prod_{i=1}^{G}\alpha_{Aip}}{\sum_{p=1}^{M}\prod_{i=1}^{G}\alpha_{Aip}}}{Y_{t2}} \qquad (4)$$

We then calculate each rule's contribution to the final action $S_c = \frac{S_{r1} + S_{r2}}{2}$. The most two effective rules are those that have the two greatest values of S_c, we use only mutation to generate new solutions because of the small population formed by the fired rules.

Note that the two output variables can be left and right wheel speeds, in the case of robots with differential wheel velocity steering.

Memory Application

After determining the rule actions to be replaced, the robot then matches the current rules to sets of rules stored in an *Experience Bank* containing each rule and its best fitness value up to date. The fitness of the rule in a given solution is supplied by the *Solution Evaluator* and is given by**:**

$$\mathbf{S_{rt}} = \text{Constant} + (d_1 - d_2) . S_c . (1-F). V \qquad (5)$$

$d_1 - d_2$ is the deviation improvement or degradation caused by the adjusted rule-base produced by the algorithm. If there is improvement in the deviation, then the rules that have contributed most will be given more fitness to boost their actions. If there is degradation then the rules that contributed more must be punished by reducing their fitness w.r.t to other rules, repeating the process for the next most effective rule. F is the normalised steering. V is the average speed. This makes S_{rt} maximised by maximising the deviation improvement compared to the previous action and higher speed with minimum adjustments to steering. The variable V was introduced to favour rules with faster speeds and F was introduced to penalise instant differences (turns) between the speeds applied to the wheels. The *Experience Survival Valuer* keeps the best fitness solution for each rule in the *Experience Bank.*

After determining the rule actions to be replaced, the robot then matches the current rules to sets of rules stored in a memory containing each rule and its best fitness value up to date. For each rule action to be replaced, we will replace the current action in the behaviour rule-base by the best fitness action stored in the experience bank. If the deviation decreases, then the robot will keep the best rules in the behaviour rule-base. If the deviation remains the same or increases, the robot uses the GA to produce a new set of solutions. These are obtained by mutating the best rules found until the deviation begins decreasing or the rule is proved ineffective. This action is supposed to speed up the GA search as it starts the GA from the "best-found" point in the solution space instead of starting from a random point. This is then considered a solution for the current situation and the rule fitness is calculated and is compared with the best fitness rule stored in the memory. If its fitness is greater than the best kept one in the *Experience Bank* then it replaces the best one, otherwise the best one still is kept in the *Experience Bank*.

Using GA to Produce New Solutions

The GA begins its search for a new rule action to replace those identified with poor performance by mutating the two most effective rules to generate new solutions. The chromosome consists of the two rule consequents (right and left wheel velocities in case of indoor robots and steering and speed in case of outdoor robots), representing each consequent by 2 bits leads to a 4 bit chromosome. We adjust two rule actions and hence the population consists of two chromosomes.

A mutation rate of 0.5 was chosen after gathering empirical evidence gathered from experimenting with different mutation rates from 0 to 1.0. This was achieved by monitoring the time the indoor prototype robot needed to achieve its goal (e.g. reaching its goal or following a wall within the desired behavior), as shown in Figure (4-a). It was noticed that at mutation values less than 0.3 there was almost no convergence (which means not finding a good solution) because the population size and the chromosome size are small, and the low mutation rates does not introduce sufficient new genetic material to produce different solutions. The same occurs for high mutation rates (higher than 0.7) as the mutation rate reaches 1.0 the only genetic material available are the primary chromosomes (e.g. 0101) and its inversion (1010) which is not enough for producing any new solutions. Thus 0.5 gave the optimum value for finding a solution after an average of 96 seconds of the indoor prototype robot time. The robot also uses the *Contextual Constraints* supplied by the sensory information to narrow the search space of the GA and thus reduce the learning time. For example, if the robot is implementing left wall following and it is moving towards the wall, then any action that suggests going to the left will not lead to a solution. Hence, if we use a left side sensor and it senses that we are going towards the wall, then the GA solutions will be constrained not to go left, the same applies for any behavior. This is similar to using a constrained-GA, where the sensor information constrains the GA search space and forces it to look at areas where solutions are likely to be found and avoids doing blind searches in the whole search space, also avoiding abnormal outputs which results from mutation. We have also conducted learning experiments without the

Contextual Constraints. It was noticed that this increased the learning time compared to those using *Contextual Constraints* by up to four times (average). This is because the GA now has to search in the entire search space, instead of just searching the limited regions where solution are likely to be found.

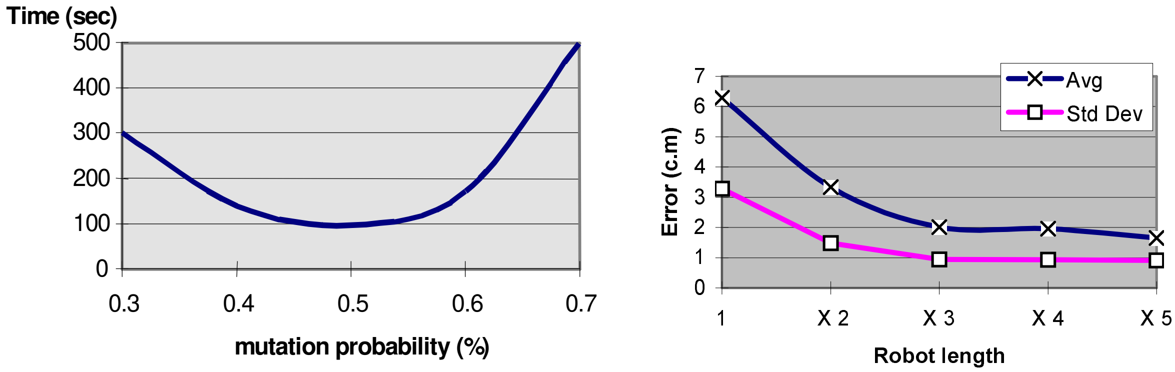


Figure (4): a) Mutation rate versus time for an edge following process. b) The relative robot length plotted against the error after learning

Stopping Criteria

The robot assumes it has learnt the rule-base for the behavior if it succeeds in maintaining the nominal value for the behavior for an optimum learning distance. The optimal learning distance has been related to units of length of the robot, so that the algorithm can be applied in an invariant manner to different size robots. In order to determine the optimal learning distance we have conducted numerous experiments using different learning distances corresponding to the robot's length (e.g. 1x robot's length, 2x robot's length, etc.). We then follow the same track that was used during the learning phase to determine the absolute deviation at each control cycle from the optimum value which is maintaining a constant distance from a wall in case of edge following and going towards a goal in case of goal seeking. The average and standard deviation of this error are calculated over 10 experiments over different tracks. From Figure (4-b) it is obvious that the average and standard error for the wall following stabilizes at three times the robot's length at average value of 2 cm and standard deviation of 1 and become almost the same for a larger number of experiments. Thus we use three times the robot length as our learning length criteria.

Learning the Fuzzy Membership Functions of the Individual Behaviours

The fuzzy Membership Functions (MF) quantify the raw vehicle sensor input data into value ranges, such as normal, near or very near. The value ranges are then used by the fuzzy inference engine, in conjunction with the control rules, to determine an appropriate output response. In most fuzzy systems, the shape of MF of the antecedent, the consequent and fuzzy rules are determined and tuned through trial and error by human operators.

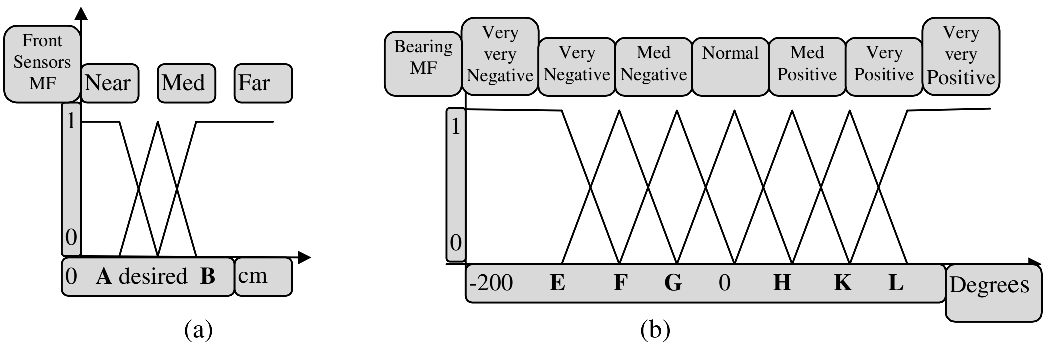
In this section, we will use our Fuzzy-Genetic techniques for determination of the MF online and through interaction with the environment, we will use the same sensors used in the prototype robot in the outdoor robots. We will use the best learnt rule-bases from the previous section as these rules are general rules relating input to output using general linguistic terms not specific for certain sensors values. Thus we want to find the best MF that when combined with a previously learnt rule-base will give the optimum action taking into account the variation of the sensor characteristics. We will try to adapt the MF to an existing learnt rule-base. Thus the robot only requires a short learning cycle to adapt the MF to the appropriate rule-base.

The robot is equipped with a short time memory that enables it to store the previous 2000 actions. The short time memory is important when the proposed algorithm produces an interim solution, as it allows the robot to return to same starting position in order to evaluate the new solution. Returning to the same position involves replaying the values in the short time memory and using the wheel encoders to bring the robot to the approximate starting position. This position can then be fine-tuned so that it matches the starting position. The fine-tuning is achieved using a pilot fuzzy logic controller that tries to bring the ultrasound and infrared reading to the values of the starting position.

Population Initialisation and fitness assignment

The parameters to be modified in the MF are shown in Figure (5). Those labelled A, B in Figure (5-a) in case of edge following; those labelled E, F, G, K, L, in Figure (5-b) in case of goal seeking (the goal seeking is represented by 7 MFs as this was found to be the least number to give satisfactory performance). Each one of these parameters is represented by 5 bits, a value which was found empirically to be the minimum number of bits required to achieve reasonable results and low computation time. The values for the MED fuzzy set in case of edge following correspond to the desired distance from the edge in case of edge following. The Z fuzzy set in case of goal seeking is zero, which means that the deviation from the target is desired to be zero. These parameters are aligned together to form a chromosome which represents a possible solution to the problem. In case of the edge following behaviours, we have 2 sensors. Each is represented by 3 fuzzy sets (one is the desired distance and two to be identified) and each fuzzy set is represented by 5 bits which leads a chromosome of 2 * 2 * 5 = 20 bits. In case of goal seeking we have only 1 sensor represented by 7 fuzzy sets (one is the zero and six to be identified), again each is represented by 5 bits, which leads to a chromosome of 1 * 5 * 6 = 30 bits.

All the chromosomes are initialised to be within the sensor range. This is a fair assumption as it is very easy to know the maximum and minimum sensors ranges from the manufacturer data sheet. This technique is known as GA guided constrained optimisation [8]. It incorporates human heuristic knowledge into the optimisation algorithm, thus during optimisation the GA is forced to produce ordered parameters (for example B is always greater than the desired value and the desired value is greater than A, etc) within the sensor range. The algorithm is also capable of determining the sensor ranges, but this will take a longer time to converge to a solution as will be shown later.



(a) (b)

Figure (5): a) Membership function of the side sensors. b) Membership function of infrared scanner used for goal seeking.

In case of the edge following (left and right) and goal seeking behaviours, the fitness function looks at minimising the deviation from a desired value. For example, in edge following, we place the indoor prototype robot at the desired distance from the edge and ask the robot to maintain this value from the edge irrespective of the irregularities found in this edge. In case of goal seeking the problem is easy, as it is required to keep the deviation from the goal to zero degrees.

The deviation is measured by the robots physical sensors (sonar in the case of edge following and infrared scanner in the case of goal seeking). This gives the robot the opportunity to adapt to the real sensor imprecision. The chromosome fitness is evaluated by how much it reduces the average of the absolute deviation (d) from the normal value, according to Equation (3). The total chromosome fitness is supplied by the *Solution Evaluator* and is given by:

$$\text{Fitness} = \text{constant1} + \left(1 - \frac{\sum_{i=1}^{N} d_i}{N}\right).(1-F).\ V \qquad (6)$$

Where N is the total number of control steps performed by testing the solution. F is the average normalised (with respect to the maximum steering) steering of the robot over the N Control steps. V is the average speed of the robot over N steps. This makes the Fitness maximised by minimising the deviation from the desired value and higher speed with minimum adjustments to the steering. At the end of testing the solution the robot checks if the average absolute deviation exceeds 10% of the normal value. If so the robot then examines the "*normal value – deviated value*". If this value is positive this indicates that the robot ended at values below the normal value. In this case, the Near fuzzy set (Medium Negative in case of goal seeking) is blamed for this deviation. The same occurs if the value is negative as this indicates that the robot ended at values greater than the normal value. In this case, the Far fuzzy set (Medium Positive in case of goal seeking) is blamed for this deviation. So in the chromosome generation the GA is constrained to generate higher values for the Near fuzzy set if the deviation was positive and smaller values for the Far fuzzy set if the deviation was negative. The values for Near and Far fuzzy sets are stored in the *Experience Bank*. By doing this step the robot uses its learning expertise to narrow the GA search space, so the GA search time can be minimised.

Generation of New Solutions

We use a population of 4 chromosomes which was found to be the smallest population to give convergence whilst maintaining computational simplicity and real time performance.

After all the solutions have determined their fitness values, the Adaptive Genetic Algorithm (AGA) is now ready to generate the new solutions. We used the Srinivas method [22] to adapt the control parameters (mutation and crossover probabilities).

The strategy used for adapting the control parameters depends on the definition of the performance of the GA. In a non-stationary environment (which is the case for the outdoor environment), where the optimal solution changes with time, the GA should possess the capacity to track optimal solutions. The adaptation strategy needs to vary the control parameters appropriately whenever the GA is not able to track the located optimum. It is essential to have two characteristics in the GA for optimisation. The first characteristic is the capacity to converge to an optimum (local or global) after locating the region containing the optimum. The second characteristic is the capacity to explore new regions of the solution space in search of the global optimum. In order to vary P_c (crossover probability) and P_m (mutation probability) adaptively, to prevent premature convergence of the GA, it is essential to be able to identify whether the GA is converging to an optimum. One possible way of detecting convergence is to observe the average fitness value f' of the population in relation to the maximum fitness value f_{max} of the population. f_{max} - f' is likely to be less for a population that has converged to an optimum solution than that for a population scattered in the solution space. P_c and P_m are defined as follows:

$$\begin{aligned} P_c &= (f_{max}-f'')/(f_{max}-f') \qquad f'' \geq f' \\ P_c &= 1 \qquad f'' < f' \end{aligned} \qquad (7)$$

$$\begin{aligned} P_m &= (f_{max}-f)/(2*(f_{max}-f')) \qquad f \geq f' \\ Pm &= 0.5 \end{aligned} \qquad (8)$$

Where f" is the larger of the fitness values of the solutions to be crossed and f is the fitness of the individual solutions. The method means that we have P_c and P_m for each chromosome. This single point crossover was chosen for computational simplicity and real time performance.

In [22] this method was superior to the simple GA giving a faster convergence (e.g. 8:1) and adapting the GA for non-stationary environments. It also relieves the designer from determining these values heuristically.

We use an elite strategy, meaning that the best individual is automatically promoted to the next generation and used to generate subsequent populations. We also use constrained optimisation as explained above so that we can minimise the search space of the GA and achieve a faster conversion. In addition the GA values are constrained to lie within the sensor range supplied from the data sheet.

In order to justify the use of these techniques we have conducted various experiments using the (AGA) approach with both an open range and a constrained range and the Simple GA (SGA) approach with a constrained range. We then calculated the average of the learning iterations over 5 learning sessions, each learning session corresponds to learning the 3 behaviours MF (right/left edge following, goal seeking,), from random positions. Five learning sessions were chosen as it was found that the average over more experiments (10, 15 or 20) yields approximately the same average values.

The SGA was tried with different parameters in the range [0.5 1.0] for P_c and [0.001 0.1] for P_m. The best performing value was found to be $P_c = 0.6$ and $P_m = 0.1$. It was found that the constrained AGA approach converges to a solution on average after *only 5 iterations and 8 minutes of the real indoor robot time*. Most of the time is consumed in moving forward to test the solution and then moving backward to the same position. The AGA approach with an open range also converges, but after a greater number of iterations (12 iterations on average). This is because it needs more time to explore the search space and determine its limits. This takes about 20 minutes of the robot time to converge to a solution. The SGA approach with defined limits and $P_c = 0.6$ and $P_m = 0.1$ converges to a solution after an average 14 iterations and 28 minutes of the robot time. These experiments justify that the proposed algorithm converges in a very short time interval due to the AGA and the constrained limits.

Stopping Criteria

The robot stops learning and assumes it had learnt the required behaviour if the robot succeeds in maintaining the nominal value for the behaviour over a distance sufficient enough to prove that the learnt MF is adequate. From empirical evidence we found this value to be equivalent to 3 times the robot length, following the same procedure explained above in the case of learning the rule bases.

Behaviour co-ordination learning

In the previous sections we have shown how sub-optimal solutions for the independent behaviours can be learnt. In this section we will show how to co-ordinate these sub-optimal behaviours to obtain a planned goal as it is a difficult task for a human to predict the interaction between the different behaviours and their best combination. It is difficult, if not impossible, to find a good solution by trying to optimise the system as a whole because it would take a long time thus not enabling online learning and adaptation which is the aim of this work. We have solved this problem by using a fuzzy based behaviour co-ordination mechanism. Here the fuzzy Membership Function (MF) shown in Figure (2-b) for behaviour co-ordination will be learnt online in a manner that satisfies some high level mission or plan (usually supplied by a high level planner which could be a person). The values of A, B are the base values to be learnt by the GA for d1, d2, d3 and d4.

Population Initialisation

Each parameter to be identified in the MF is represented by 5 bits. The use of 5 bits was found empirically to be the minimum number of bits to achieve reasonable results and fast computation time. Note that if a behaviour is deleted by the planner, then its parameters will not be encoded. These parameters are aligned together to form a chromosome that represents a possible solution for the problem. This means that when all the behaviours are present we have maximum 4 * 2 * 5 = 40 bit chromosome.

Fitness Evaluation

After the chromosomes initialisation the robot starts moving to test the proposed solution. The determination of the fitness function depends on the high level mission. The high level missions can be regarded as a deviation minimisation problem. In the case of edge-following while avoiding obstacles and goal seeking, the mission can be viewed as follows: minimising the deviation from a *normal* desired distance in following the edge, *AND* minimising the deviation from a *normal* safe distance in avoiding obstacles, *AND* minimising the deviation from the goal to a *normal value* of zero with minimum steering deviation and a high speed. The *AND* function is represented by a mathematical sum. The deviation is measured by the robot's physical sensors (sonar in case of edge following and infrared scanner in case of goal seeking). This gives the robot the opportunity to adapt to the sensor imprecision.

The chromosome fitness is evaluated by how much it reduces the average of the total absolute deviation d_i (which is calculated as the sum of the individual deviations over the existing behaviours *m, in a given mission* at each control step), while using minimum steering deviation and high speed. Where k is a given behaviour and d_i is given by:

$$d_i = \sum_{k=1}^{m} \left(\frac{|\,normal.value - deviated.value\,|}{\max .deviation}\right)_k \qquad (9)$$

The *normal value* will correspond to the value that reflects the mission goals; the *deviated value* is any value deviating from the normal value. The *maximum deviation* corresponds to the maximum deviation that can occur. The solution Fitness is given by:

$$\text{Fitness} = \text{constant} + \left(1 - \frac{\sum_{i=1}^{N} d_i}{N}\right).(1-F).V \qquad (10)$$

Where N is the total number of control steps done until the end of testing the solution. F is the average normalised (with respect to the maximum steering) steering of the robot over N Control steps. V is the average speed of the robot over N steps. This maximises the fitness by minimising the total absolute deviation and the steering deviation and maximising the speed.

Transferring the Learnt Controller from Indoor Prototype Robots to the Outdoor Robots

In the previous sections we explained how to learn the robotic controller online and through interaction with the environment using the indoor prototype robots. In this section we discuss the issues arising in transferring the controller from the prototype to the target.

In our approach, the prototype robots have the same computational power as the target robots together with the same physical sensors. We also took great care to ensure that all the algorithm parameters were robot independent, so that the algorithm can be moved between different robots with only minimal changes. There are clearly big differences between the indoor and outdoor environments. Unstructured and dynamic environments, such as the agricultural domain, represent a difficult challenge to mobile robot navigation and control techniques. In this setting the inconsistency of the terrain, the irregularity of the product and the open nature of the working environment generate complex problems of sensing and control. To assist smooth transfer of solutions, we did what we could to make the indoor experiments more realistic. For example, in the prototype we utilised sensors with more noise and imprecision together with materials and geometry found in the target environments (e.g. objects constructed from hay). After transferring the indoor prototype controller to the target (outdoor vehicle), the target then begins running an online adaptation module that allows it to adapt to the differences between the prototype and target vehicles and environments. This procedure is useful in learning controllers for the outdoor robots which might involve dangerous manoeuvres using small and cheap indoor prototype robots in a controlled indoor environment, thus avoiding the risks associated with online learning using heavy and expensive outdoor robots. Also our learning techniques learn general controllers that can be applied to different outdoor robots performing the same mission and applying a short adaptation cycle thus saving the need to learn a new controller for each different outdoor robot. Perhaps more importantly, this adaptation session is important in the case that environmental or robot kinematics changes occur and the robot needs to rapidly adapt to these changed circumstances.

Although fuzzy logic allows a degree of imprecision to exist within the environment, significant environmental or robot differences will prevent the rule sets from operating correctly. One solution to this problem is to provide a new rule set for each of the different environments, although prior knowledge of the different environments would be needed (which is difficult if not impossible for dynamic outdoor environments). An alternative solution is to allow the existing co-ordinated rule set to be adapted to compensate for the environmental differences. This latter approach only requires a basic rule set to be present from the prototype robot controller and does not require prior knowledge of all possible environments. We start the adaptation from the best learnt HFLC by the indoor prototype robots, instead of starting from a random point. If the system was started with random rule bases, the system could still modify its actions but would take approximately six times the time needed when started from a good rule base containing some inappropriate rules for the new environment. This figure was found by practical experimentation.

The algorithm featured in this work discovers the rules (in different behaviours) that, if modified, can lead the robot to adapt to its environment. Whilst it might seem a good idea to change the co-ordination parameters, or the membership functions of the individual behaviours, to adjust the robot behaviour when it fails, this is not the best approach. This is can readily be illustrated by considering a case where the rules in the individual rule bases become inappropriate, then clearly changing the co-ordination parameters will never correct the robot behaviour (as was proved by experimentation). Also changing the MF is a difficult task, as it needs to be done to each individual behaviour necessitating the robot to be taken away from the field for MF calibration. However, our method allows the poor rules to be found and corrected for each behaviour, online and in-situ without the need to repeat the whole learning cycle, modifying only the actions of a small number of rules that performed poorly.

Adaptation of Co-ordinated Behaviours in the Target Robots

Imagine a situation where a robot is inside a geometrical structure and it is very close to (or has collided with) an obstacle. Alternatively, consider when a robot is following a wall and is very close to (or has collided with) an obstacle. In these cases we will modify all the rule bases whose behaviours were active. The robot then uses its short term memory to return back to its pre-failure position, identifying the two most dominant rules (which can belong to two different behaviours). The robot then replaces the actions of these rules to solve this situation. The way the robot determines the dominant rules is done by calculating the contribution of each rule to the final output in a given situation as follows: Apply Equation (2) and substitute Y_t from Equation (1). Then we can write the crisp output Y_{ht} as:

$$Y_{ht} = \frac{\sum_{y=1}^{4} mm_y \dfrac{\sum_{p=1}^{M} y_p \prod_{i=1}^{G} \alpha_{Aip}}{\sum_{p=1}^{M} \prod_{i=1}^{G} \alpha_{Aip}}}{\sum_{y=1}^{4} mm_y} \tag{11}$$

Where M is the total number of rules, Y_p is the crisp output for each rule, $\Pi\alpha_{Aip}$ is the product of the membership functions of each rule inputs. G is the number of the input variables, mm_y is firing strength of each of the four behaviours.

Because we have two output variables, which can be steering and wheel speeds in case of the outdoor robots, then we have Y_{ht1} and Y_{ht2}. The contribution of each rule p in the behaviour y to the total output Y_{ht1}, Y_{ht2} is denoted by S_{ra11}, S_{ra22} where S_{ra11}, S_{ra22} are given by:

$$\mathbf{S_{ra11}} = \frac{\dfrac{mm_y}{\sum_{y=1}^{4} mm_y} \cdot \dfrac{Y_{p1} \prod_{i=1}^{G} \alpha_{Aip}}{\sum_{p=1}^{M} \prod_{i=1}^{G} \alpha_{Aip}}}{Y_{ht1}}, \quad \mathbf{S_{ra22}} = \frac{\dfrac{mm_y}{\sum_{y=1}^{4} mm_y} \cdot \dfrac{Y_{p2} \prod_{i=1}^{G} \alpha_{Aip}}{\sum_{p=1}^{M} \prod_{i=1}^{G} \alpha_{Aip}}}{Y_{ht2}} \tag{12}$$

We then calculate each rule's contribution to the final action Sa_{c1} by:

$$Sa_{c1} = \frac{S_{ra11} + S_{ra22}}{2}.$$

The most effective rules, are those rules with the greatest values of Sa_{c1}. If there is an improvement in the distance (moved before collision) produced by the modified rule base, then the rules that contributed most must be given increased fitness to boost their actions. If there is no improvement then the rules that contributed most must be punished by reducing their fitness w.r.t to other rules and the solutions that had small contributing actions are examined. The fitness of each rule is supplied by the *Solution Evaluator* and is given by:

$$\mathbf{S}_{\mathbf{rat}\ 1} = \text{Constant} + (d_{new}\text{-}d_{old})\ .S_{ac1}.\ (1\text{-}F).\ V \qquad (13)$$

Where d_{new} is the distance moved by the robot using the modified rule base, d_{old} is the distance moved by the robot before modifying the rule base, d_{new}-d_{old} is the distance improvement or degradation caused by the adjusted rule-base produced by the algorithm. F is the normalised steering. V is normalised average speed. Thus S_{rat1} is maximised by increasing the speed and distance moved by the robot before behaviour failure with minimum adjustments to steering. In the first population of the GA, because there is no distance moved yet, we blame the rules that have contributed most to the behaviour failure. The fitness of each rule is given by:

$$\mathbf{S}_{\mathbf{rat}} = \quad \text{Constant} - S_{ac1} \qquad (14)$$

The most dominant rules are determined according to Equation (14), the more they contributed to the bad action, the less will be their fitness. The actions of these rules can be modified to solve this situation. After determination of the rules whose consequences need to be replaced, the robot then matches the current rules to sets of rules stored in the *Experience Bank*. If, for example; rules 1,2 from the obstacle-avoidance rule base, rule 3 of the left edge-following behaviour and rule 4 of the right edge following need to be replaced, AND the first rule cluster in the Bank contains the consequences of rules 1 of obstacle avoidance; rule 3 of left edge following and rule 6,7 of right edge following, then the consequences of rules 1 for obstacle avoidance and 3 for left edge following will be changed and rule 2 for obstacle avoidance and 4 for right edge following will remain the same. Then the robot begins moving with this modified rule-base. If it survives and gets out of this situation with no behaviour failures, then these rules are kept in the rule-base of the controller. In this way we have saved the process of learning a solution to this problem from the beginning by using the memorised experience that had worked in different environmental conditions. If the robot fails again, it measures the distance it had moved to determine the fitness of the solution proposed by this memory. After all the memory clusters have been examined and the robot still fails to produce a desired response. The best solution proposed by *Experience Bank,* according to its fitness, is kept in the rule-base of the controller in order to serve as a starting position of the Adaptive Genetic Algorithms (AGA) search instead of starting from a random point. This memory effect will serve to speed up the search.

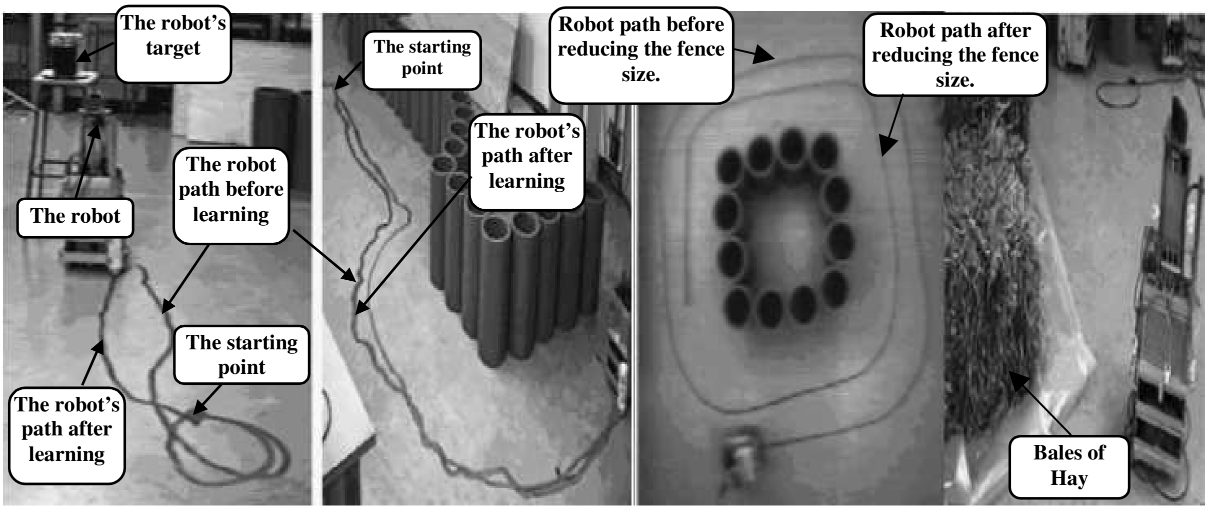
However, a problem occurs as the system begins accumulating experience that exceeds the physical memory limits. This implies that we must delete some of the stored information. To deal with this, for every rule cluster we attach a *difficulty counter* to count the number of iterations taken by the robot to find a solution to a given situation, we also attach a *frequency counter* to count how frequently this rule have been retrieved. The *degree of* *importance* of each rule cluster is calculated by the *Experience Survival Valuer* based on the product of the *frequency counter* and the *difficulty counter*. This attempts to keep the rules that required a lot of effort to learn (due to the difficulty of the situation) and also the rules that are used frequently. When there is no more room in the *Experience Bank*, the rule cluster that had the least *degree of importance* is selected for replacement. If two rule clusters share the same importance degree, tie-breaking is resolved by a least-recently-used strategy. The rule that has not been used for the longest period of time is replaced. Thus an *age parameter* is also needed for each rule cluster. The value of the age parameter increases over time, but is initialised whenever the associated cluster is accessed. The limit for the memory clusters is set to 2000 rule clusters, so if we exceed this limit the *Experience Survival Valuer* begins using the *degree of importance* and the age operator to optimise the memory.

The AGA is using Srinivas method [22] explained above. If, for example, rule number 5 of the obstacle avoidance and rule 7 of the left wall following are chosen for reproduction by roulette wheel selection due their high fitness. This implies they have either contributed more with their actions to the final action that caused improvement, or contributed less with their actions to final action that caused degradation. Then we apply adaptive crossover and mutation operators to both chromosomes. The resultant offspring will be used to replace the consequent of rules 1 and 2 for obstacle avoidance that were largely blamed for the behaviour failure. The AGA is constrained to produce offspring according to the *Contextual Constraints* supplied by the input sensors in accordance with Section (5.1.2). The robot ends the adaptation when the robot achieves the desired response.

Experimental Results

This section describes the experiments and results. Figure (6-a) shows the indoor prototype robot learning the rule base of the goal seeking behaviour, using an imprecise infrared beacon system to emulate a goal. The robot learnt the rule-base in an average of 96 seconds over 5 trials starting from different positions with different initialised rule bases (the average error over more experiments was found to converge after 5 trials).

Figure (6-b) shows the indoor prototype robot learning to follow an irregular edge offering poor ultrasound reflections and hence introducing imprecision and noise into the system. As before, the prototype robot succeeded in learning to follow this irregular edge in an average time of 96 seconds (over 5 trials with different starting points and rule-bases). The robot was tested on a variety of different edges (including hay bales) as the edge shown in Figure (6-c), emulating crop harvesting in outdoor environments by following an irregular edge of decreasing radius. The robot followed these different edges with an average deviation of 5.2 cm and a standard deviation of 2.8.



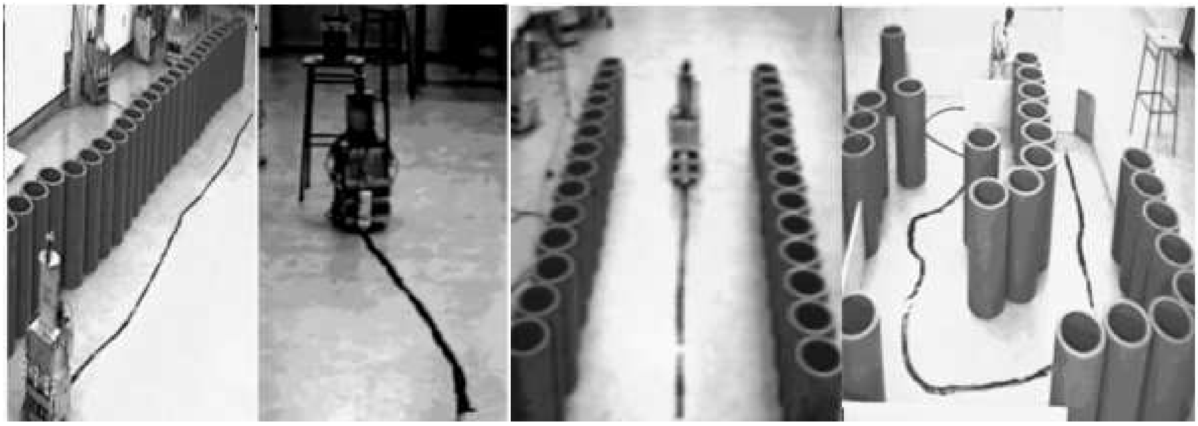
(a) (b) (c) (d)

Figure (6): Indoor robot learning goal seeking behaviour. b) Learning left wall following behaviour. c) Emulating the harvesting. d) Following fences of bales.

After learning the rule bases of different behaviors. The prototype robot then learnt the MF for the different behaviors to form an optimum behaviour for a local solution (which will later coordinated to produce a solution for the whole mission). Figure (7-a) shows the prototype robot learning the MF for right edge following. The robot learnt the MF in 8 minutes over 5 iterations,. After learning the MF the robot was tested on edges different from those used for training resulting in an average deviation of 1.2 cm with a standard deviation of 0.8. Figure (7-b) shows the robot after learning the MF for goal seeking which took an average time of 5 minutes with an average deviation of 1.1 cm and standard deviation of 0.9. Note that the robot response in terms of the average and standard deviation improves which reflects that the robots had learnt a sub optimal behaviour after learning the best MF that suits the previously learnt rule base.

Figure (7-c) shows the robot after learning to coordinate the left, right edge following behaviours, obstacle avoidance and goal seeking behaviours to follow a corridor. The robot learnt this mission in average time of 12 minutes. The robot was then tested in different geometrical structures producing an average deviation of 1 cm and a standard deviation of 0.7. Note that the average and standard deviation improve as the robot learns a semi-optimal behaviour. Our techniques gave behaviours that produce better average and standard deviation than their manual counterparts by the ratio of 4:1.

After developing a robotic controller on the prototype robot (indoors), the same controller is then transferred to the target (outdoor) vehicles which then adapted the controller to its specific needs (using the same algorithm as in the prototype). Thus the method is robot independent and can be ported between different robots with minimum changes.



(a) (b) (c) (d)

Figure (7): Path after learning right edge following MF. b) Path after learning the goal seeking MF. c) After learning behaviour co-ordination following a corridor. d) After learning behaviour co-ordination in a more complicated environment.

(a) (b) (c) (d)

Figure (8): a) Electrical outdoor robot path following an irregular edge before and after adaptation in outdoor environment. b) Electrical outdoor robot path before adaptation in an outdoor corridor. c) Electrical outdoor robot path after adaptation in an outdoor corridor. d) Electrical outdoor robot after adaptation navigating in a complex corridor.

Figure (8-a) shows the outdoor electrical robot modifying its controller to suit the outdoor environment and the robot kinematics. The robot learns the necessary rules in 79 seconds following an irregular metallic fence in outdoor environment with an average deviation of 3 cm and a standard deviation of 1. Figure (8-b) shows the robot before adapting to the outdoor environments following the center line of an outdoor metallic corridor displaying large deviations of 30 cm with a standard deviation of 9. Figure (8-c) shows the robot after adaptation following a center line with an average deviation of 2.1 cm and a standard deviation of 0.8 under different environmental conditions such as rain, wind, sunshine and starting from different positions. Figure (8-d) shows the outdoor electrical robot coordinating all its behaviours which are the left/right edge-following and the obstacle avoidance behaviour (learnt in [21]) to follow the center line of the corridor while avoiding closely spaced objects.

Figure (9-a) shows the electrical outdoor robot after adaptation following an irregular crop edge in outdoor changing and dynamic environment with an average deviation of 3 cm and a standard deviation of 1. Figure (9-b) shows the same robot after adaptation avoiding obstacles while following an irregular crop edge.

Figure (9-c) shows the diesel powered robot with different sensors (mechanical wands to sense the distance from the crop) and different physical and geometrical appearance. The robot applied the same algorithm and used the indoor controller, adapting to its own needs and following an irregular hay crop (full of gaps) with an average deviation of 2.4 cm and a standard deviation of 1.

(a) (b) (c)

Figure (9): a) Electrical outdoor robot path after adaptation following an irregular crop edge. b) Electrical outdoor robot after adaptation following an irregular crop edge whilst avoiding obstacles. c) Diesel outdoor robot following an irregular crop edge after adaptation.

An extreme circumstanced environment that was tried is a dynamic environment with other moving robots and in an outdoor unstructured environment. The robot had done well in these environments keeping itself safe and performing the specified jobs.

Conclusions and Future Work

In this paper we have presented a novel technique for prototyping robot design. The prototyping technique is based on developing and testing the outdoor robots on cheap indoor prototype robots. The indoor robot controllers are learnt online and through interaction with an environment that resembles the outdoor environment (e.g. including irregular shaped objects and noisy and imprecise sensors). The principal advantages in doing this is that it avoids the problems associated with software simulation and it learns controllers for the outdoor robots which might involve dangerous manoeuvres (like helicopters and underwater vehicles) using small and cheap indoor prototype robots in a controlled indoor environment, thus avoiding the risks associated with online learning using heavy and expensive outdoor robots. Also our learning techniques learn general controllers that can be applied to different outdoor robots performing the same mission and applying a short adaptation cycle thus saving the need to learn a new controller for each different outdoor robot.

The indoor prototype robots used a novel patented Fuzzy-Genetic system for online learning, which uses a hierarchical learning procedure that learns the general fuzzy rule bases of the individual behaviours and then learns the best MF that suits these general rule bases thus forming sub-optimal behaviours. The robot then learns the best co-ordination between the behaviours to have a good response. Using our hierarchical learning procedure has the following advantages: the system proceeds by learning general behaviours, which can then be combined in different ways to achieve different goals, also we have a simplified design procedure as a complex problem is decomposed into a set simpler ones. These results in a fast online autonomous learning system for real-world based physical robots which is, as far as we can determine, a unique achievement [23, 24]. Also the fuzzy controller is based on a hierarchical fuzzy controller, which have the following advantages:

- It simplifies the design of the robotic controller and reduces the number of rules to be determined.
- It uses the benefits of fuzzy logic to deal with imprecision and uncertainty.
- It uses fuzzy logic for the co-ordination which provides a smooth transition between behaviours with a consequent smooth output response.
- It offers a flexible structure where new behaviours can be added or modified easily.

Also the learning system is based on a double hierarchical fuzzy-genetic system which has the following advantages:

- The hierarchy preserves the system experience, and speeds up the genetic search as the GA is started from the best known point in the search space.
- It rapidly produces good (but sub-optimal) solutions for the individual behaviours and the co-ordination parameters faster than any other reported method utilising real robots see [18], [25] for comparisons with other methods.

Our architecture has been designed as a generic solution and not specific to any particular type of robot. After developing a good controller in the indoor robot, it was then moved to the outdoor vehicles that runs a fast adaptation procedure to adapt the indoor controller to the outdoor kinematics and environment. By doing this we had saved repeating the learning cycle from the beginning by testing the controller in the indoor robots and then quickly adapting it to suit the robot requirements. Our techniques have been tested in learning and adapting controllers for two different outdoor vehicles, one is diesel powered and the other is electrically powered. Our system applies a continuos learning system in which the robot learns and gains experience through its life through interaction with the real environment. Also we have satisfied the definition of intelligent system stated in [15]

Advancing the state of knowledge in the field of online learning has potential benefits for a wide set of embedded control systems such as vehicles, factory machinery, telecommunication medical instrumentation and emerging areas such as intelligent-buildings and flying robots. It is also particularly appropriate to situations where modelling or reprogramming are difficult or costly (e.g. inaccessible environments such as underwater, outer space or environments where one robot is required to accomplish a variety of tasks). In such environments it is necessary to perform rapid online learning through interaction with the real physical world. Such approach both saves money and increases reliability by allowing the robot to automatically adapt without further programming to the changing user and environment needs it will experience throughout its lifetime.

References

[1] L. Steels, "When are Robots Intelligent Autonomous Agents", Journal of Robotics and Autonomous Systems, Vol. 15, pp.3-9, 1995.

[2] M. Dorigo, M. Colombetti, "Robot Shaping: Developing Autonomous agents through learning", Artificial Intelligence Journal, Vol.71, pp. 321-370, 1995.

[3] R. Brooks, "Artificial Life and Real Robots", MIT press 1992.

[4] O. Miglino, H. Lund, S. Nolfi, "Evolving Mobile Robots in Simulated and Real Environments", Technical report NSAL-95007, Reparto di sistemi Neurali e vita Artificale, Instituto di Psicologia, Consiglio Nazionale delle Ricerche, Roma, 1995.

[5] A. Saffiotti, "Fuzzy Logic in Autonomous Robotics: Behaviour Co-ordination", Proceedings of the 6th IEEE International Conference on Fuzzy Systems, Vol.1, pp. 573-578, Barcelona, Spain, 1997.

[6] R. Filho, "Genetic Algorithms Programming environments", IEEE transactions on Computers, pp. 28-43, June 1994.

[7] T. Fukuda, N. Kubota , "An Intelligent Robotic system based on Fuzzy approach", Proceedings of the IEEE, Vol. 87, No. 9, pp.1448-1470, September 1999.

[8] G. Tan, X. Hu, "More on Designing Fuzzy Controllers Using Genetic Algorithm: Guided Constrained optimisation", Proceedings of the 6th IEEE International Conference on Fuzzy systems, Barcelona, Spain, pp. 497-502, 1997.

[9] G. Linkens, O. Nyongeso, "Genetic Algorithms for Fuzzy Control, Part II: Online System Development and Application", IEE proceedings Control theory applications, Vol.142, pp.177-185, 1995.

[10] D. Leitch, "A New Genetic Algorithm for the Evolution of Fuzzy System", Ph.D. thesis, University of Oxford 1995.

[11] C. Lee, “Fuzzy logic in control systems: Fuzzy logic controller, Part I“, IEEE transactions on Systems, Man, Cybernetics, Vol.20, pp. 404-432, March 1990.

[12] A. Bonarini, F. Basso, “Learning Behaviours implemented as Fuzzy logic and Reinforcement Learning”, Second Online Workshop on Evolutionary Computation, 1996.

[13] F. Hoffmann, "Incremental Tuning of Fuzzy Controllers by means of Evolution Strategy”, Proceedings of the GP-98 Conference, pp. 550-556, Madison, Wisconsin, 1998.

[14] M. Dorigo, "Genetics-based Machine Learning and Behaviour Based Robotics: A new synthesis", IEEE transactions on Systems, Man, Cybernetics, pp. 141-154, 1993.

[15] N. Kasabov, “Introduction: Hybrid intelligent adaptive systems. International Journal of Intelligent Systems” Vol.6, pp.453-454, 1998.

[16] R. Arkin "Behaviour-based Robotics", MIT Press, 1998.

[17] E. Tunstel, T. Lippincott, M. Jamshidi, "Behaviour Hierarchy for Autonomous Mobile Robots: Fuzzy Behaviour Modulation and Evolution", International Journal of Intelligent Automation and soft computing, Vol.3, No.1, pp. 37-49, 1997.

[18] H.Hagras, V. Callaghan, M. Colley, M. Carr-West, “A Behaviour based Hierarchical Fuzzy Control Architecture for Agricultural autonomous Mobile Robots”, Proceedings of the International Conference on Computational Intelligence For Modelling, Control and Automation, pp. 166-171, Vienna- Austria, February 1999.

[19] E. Ruspini, "Truth as Utility a Conceptual Synthesis", In Proceedings of the 7th Conference on uncertainty in Artificial Intelligence, pp. 458-464, Los Angeles, CA 1991.

[20] A. Bonarini, "Comparing Reinforcement Learning Algorithms Applied to Crisp and Fuzzy Learning Classifier systems", Proceedings of the Genetic and Evolutionary Computation Conference, pp. 52-60, Orlando, Florida, July, 1999.

[21] H.Hagras, “An Evolutionary Computing Based Technique for Embedded-Agent System Design for Mobile Robots”, PhD thesis, University of Essex, March 2000.

[22] M. Srinivas, L. Patnaik, “Adaptation in Genetic Algorithms”, Genetic Algorithms for pattern recognition, Eds, S.Pal, P.Wang, CRC press, pp. 45-64, 1996.

[23] H. Hagras, V. Callaghan, M. Colley, “On Line Calibration of the sensors Fuzzy Membership Functions in Autonomous Mobile Robots” to appear in the Proceedings of the 2000 IEEE International Conference on Robotics and Automation, San Francisco-USA, April 2000.

[24] H. Hagras, V. Callaghan, M. Colley, “Learning Fuzzy Behaviour Co-ordination for Autonomous Multi-Agents Online using Genetic Algorithms & Real-Time Interaction with the Environment ” to be appear in the Proceedings of the 2000 IEEE International Conference on Fuzzy Systems, San Antonio-USA, May 2000.

[25] H. Hagras, V. Callaghan, M. Colley, “A Fuzzy-Genetic Based Embedded-Agent Approach to Learning and Control in Agricultural Autonomous Vehicles”, Proceedings of the 1999 IEEE International Conference on Robotics and Automation, pp. 1005-1010, Detroit- U.S.A, May 1999.