Acquiring Mobile Robot Behaviors by Learning Trajectory Velocities

KOREN WARD
School of Information Technology and Computer Science, The University of Wollongong
Wollongong, NSW, Australia, 2522.
koren@uow.edu.au

ALEXANDER ZELINSKY
Research School of Information Sciences and Engineering, The Australian National University
Canberra, ACT, Australia, 0200.
Alex.Zelinsky@anu.edu.au

Abstract. The development of robots that learn from experience is a relentless challenge confronting artificial intelligence today. This paper describes a robot learning method which enables a mobile robot to simultaneously acquire the ability to avoid objects, follow walls, seek goals and control its velocity as a result of interacting with the environment without human assistance. The robot acquires these behaviors by learning how fast it should move along predefined trajectories with respect to the current state of the input vector. This enables the robot to perform object avoidance, wall following and goal seeking behaviors by choosing to follow fast trajectories near: the forward direction, the closest object or the goal location respectively. Learning trajectory velocities can be done relatively quickly because the required knowledge can be obtained from the robot's interactions with the environment without incurring the credit assignment problem. We provide experimental results to verify our robot learning method by using a mobile robot to simultaneously acquire all three behaviors.

Keywords: robot learning; fuzzy associative memory; trajectory velocity learning; TVL, unsupervised leaming; associative learning; multiple behavior learning.

1. Introduction

Providing mobile robots with reactive behaviors using conventional programming methods can be a difficult and time consuming task. This is because:

- It is hard to foresee and encode all situations the robot could encounter.
- It can be hard to decide precisely which action the robot should take and how fast it should move in all situations.
- It may not be possible to predict precisely what information will emerge from the sensors in many situations.
- It may not be possible to know if the final encoded behavior is robust enough for the intended task or whether it will work effectively in different environments.

1.1. Learning Robot Behaviors via Demonstrated Actions

To make the task of encoding robot behaviors simpler, some researchers have been able to show that certain, robot behaviors can be learnt by demonstrating actions with the robot via remote control. For example, Pomerleau (1993) Krose and Eecen (1994) and Tani and Fukumura (1994) showed how training data derived from demonstrated actions could be used to train neural network based controllers for mobile robot navigation. Also, Wyeth (1998) demonstrated how neural networks could be trained to enable a visually guided mobile robot to avoid objects and fetch a ball. Alternatively, Castellano *et al* (1996) showed how Fuzzy Associative Memory (FAM) matrices (see Kosko (1992)) could be trained to obtain wall following behavior by demonstrating the behavior with a sonar equipped mobile ro-

bot. However, the main disadvantage of using demonstrated actions to derive robot controllers is that it can be difficult to obtain non-conflicting training data that accurately and comprehensively describes any specific behaviors. Furthermore, it can be difficult to extract an efficient controller from the training data that performs adequately. Also, it may not be practical or possible to repeatedly teach a robot the same or different situations in order to improve its actions if the robot exhibits inadequate behavior in unknown environments. For these reasons it is highly desirable for control systems to be built that allow robots to automatically produce or improve their behaviors via some form of unassisted robot learning process.

1.2 Learning Robot Behaviors Without Supervision

To enable robots to learn behaviors automatically, a variety of techniques have been used. These techniques include the use of Reinforcement Learning (RL) (e.g. Materic (1997), Kaelbling, (1993), Connel and Mahadevan (1992)); Genetic Algorithms (GAs) (e.g. Floreano and Mondada (1994, 1995, 1996), Jakobi (1994), Jakobi *et al* (1992), Nolfi *et al*, (1994)); and some more unique approaches (like Sharkey (1998), Nemhzow *et al*, (1990), Michaud and Materic (1998)).

RL has successfully been applied to robots for learning a wide variety of behaviors and tasks (e.g. robot soccer Asada *et al* (1995), light seeking Kaelbling (1991), box pushing Connel and Mahadevan (1992), and hitting a stick with a ball Kalmar, *et al*, (1998)). Generally, these tasks are acquired by using RL to get the robot to learn either low-level behaviors as in Connel and Mahadevan (1992), Kaelbling (1995), Asada *et al* (1995) or behavior switching policies like Materic (1997) and Kalmar, *et al* (1998). In many cases, results are achieved by simplifying the problem by providing the robot with minimal input sensing and few output actions or by carefully hand coding appropriate control and/or reinforcement signals to facilitate learning. For example, Kaelbling (1991) devised a robot called "Spanky" to demonstrate how different RL algorithms compared in performance at learning light seeking behavior. This robot was equipped with 5 whiskers and 2 infra red (IR) sensors. By grouping the front sensors and using thresholds on the IR sensors Kaelbling was able to represent the entire input space with just 5 bits. By also providing the robot with only 3 possible actions for performing its behavior, the problem was successfully reduced to one of finding a solution that works well from the $2^5 \times 3 = 96$ possible outcomes.

In cases where the robot is equipped with more comprehensive sensing, for example Connel and Mahadevan's (1992) box pushing sonar robot, good results become much harder to achieve. This is because the input to output state space becomes to large to be learnt in real time due to the credit assignment problem (as explained in Watkins (1989)). This limiting constraint of RL results mainly from the time lag between time steps and the uncertainty associated with assigning credit to actions when delayed reinforcement signals are involved. Despite this limitation, Connel and Mahadevan were able to show that RL could achieve results, comparable to hand coded solutions, by using the learnt input-output associations to statistically classify the input space. Impressive RL results were also achieved with the vision equipped soccer playing robots of Asada *et al* (1995). However, these results were obtained by performing learning off-line in simulations and installing the learned associations in the real robot. Unfortunately, if this approach is used in unknown or typical unstructured environments, learning becomes a much harder task due to the difficulties associated with adequately modeling the environment and robot's sensors (particularly if noisy sensors such as ultrasonic sensors are used). For a more detailed survey of RL see Kaelbling (1996).

Besides RL, most other forms of unassisted robot learning involve using various search methods within the space of possible controllers in an attempt to find a control solution that performs well. The most common of these search methods involves the use of genetic algorithms Holland (1986), Meeden (1996), genetic programming Koza (1991) or some more novel search methods like Schmidhuber (1996). Generally, these approaches fail to produce results as good as those achieved with RL, particularly where robots with considerable sensing are involved (i.e. where the state space is greater than 10,000). This lack of performance is due mainly to the size of the required search space and the time it takes to evaluate the performance of possible control solutions on the physical robot. Despite this, Floreano and Mondada (1994, 1995, 1996) were able to demonstrate that object avoidance, homing behavior and grasping behavior could be evolved with GAs on their "Khepera" robot over considerable periods of time (days in the case of grasping behavior). Here object avoidance and homing behaviors required the robot's motion to be carefully monitored with external laser sensors. The grasping behavior was achieved incrementally by saturating the environment with graspable balls and gradually reducing the ball density as competence was acquired. In addition to these results, Jakobi (1994), Jakobi *et al* (1992), Nolfi *et al*

(1994), Grefenstette and Schultz (1994) and Schultz (1991) managed to evolve various robot behaviors on a simulator which were then installed in a real robot. However, like the soccer playing robots of Asada *et al* (1995), controllers obtained this way can only be expected to produce good results in known highly structured environments that are capable of being effectively modeled in the computer. For a comprehensive survey on evolved robot controllers, see Materic and Cliff (1996).

An alternative method to RL and evolutionary techiques for producing behaviors on robots without significant human assistance was demonstrated by Sharkey (1998). This method involved operating the robot with a hand coded innate controller based on artificial potential fields Khatib (1985), and by collecting sensor-input and command-output associations generated by the robot's motion. After considerable operation, the subsequent collected training data was preprocessed and used to train a neural network which was then used to control the robot. By using this approach Sharkey was able to demonstrate that improved performance and shortcomings inherent in the innate controller could be overcome with this method. This however, requires an innate controller to be devised and implemented first. Furthermore, there can be no guarantee that a poorly performing hand coded controller will produce an adequate neural network controller no matter how much training is performed.

To improve on existing unassisted robot learning methods we have devised a method for rapidly acquiring object avoidance, wall following and goal seeking behaviors on mobile robots equipped with range sensing devises. Unlike most existing robot learning methods, which are based on learning associations between sensors and actions, our approach is based on learning association between sensors and trajectory velocities. Previously, Singh *et al* (1993) demonstrated that faster learning times could be achieved with RL by using a simulated robot to learn associations between the environment state and so called "Dirichlet" and "Neuman" parameters. However this approach was environment specific and also suffered from the credit assignment problem. By using our approach learning is rapid, online and suitable for implementation on a real robot. Furthermore, there is no credit assignment problem and all three behaviors are learnt simultaneously on a single associative map. Also, the acquired behaviors can have their object clearance distance changed by adjusting a single threshold parameter and the robot also learns to appropriately control its velocity.

In Section 2, we describe our approach to unassisted robot learning by illustrating some of the similarities and differences between our trajectory velocity learning approach and the traditional RL approach to learning object avoidance behavior. In Section 3 and 4, we describe how we use Fuzzy Associative Memory (FAM) matrices (see Kosko (1992)) to store and incrementally learn associations between sensor range readings and trajectory velocities. Section 5 describes various experiments we conducted with our robot in a variety of environments.

2. Trajectory Velocity Learning

To overcome the slow learning times associated with existing unassisted robot learning methods and to enable robots with considerable sensing to learn in real time we have decided to alter the actual learning task. Instead of performing the difficult task of learning associations between sensor inputs and output responses, as for example in conventional RL, we use the robot to learn a much simpler task: i.e. learning associations between sensors and trajectory velocities as depicted in Figure 1. We refer to this form of learning as Trajectory Velocity Learning (TVL). Each input vector is comprised of immediate sensor range readings and trajectory velocities can be described as appropriate velocities for negotiating a predefined set of immediate straight or curved paths based on their collision distances with nearby objects. Hence, if one of the robot's immediate trajectories shown in Figure 1 collides with a close object, appropriately, it should have a slower velocity than other trajectories that lead into free space.

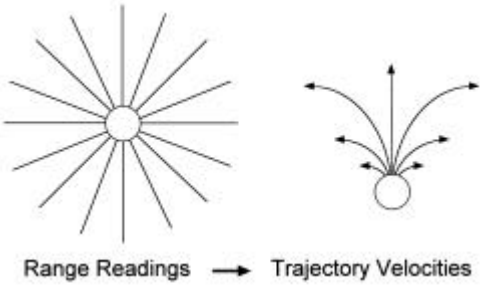


Figure 1. Learning associations between range readings and trajectory velocities.

Recently, Fox *et al* (1997) demonstrated that trajectory velocities can be used as an efficient means of making control decisions for avoiding obstacles with a mobile robot. However, with Fox *et al*'s *"dynamic window"* approach, trajectory velocities were not learnt. Instead, they were calculated by considering all objects in the

vicinity of the robot. This requires all objects around the robot to be accurately detected in order to calculate trajectory velocities. Usually, this is difficult to achieve even with the most sophisticated sensing devices. In particular, sonar sensors will only return a range reading if the sonar beam is almost normal to the object surface. Alternatively, if learnt environment maps or occupancy grids are used to assist in locating objects around the robot, the robot's ability to negotiate unknown regions of the environment is restricted. This also has the disadvantage that the robot has to accurately estimate its position which is difficult to achieve using only odometry.

By using the robot to learn associations between sensor range readings and trajectory velocities these deficiencies are overcome. This is because the robot learns to perceive its environment in terms of trajectory velocities eliminating the need for object locations to be known in advance when control decisions are made. Furthermore, the use of an associative map to look up trajectory velocities directly from sensor data enables trajectory velocities to be determined quickly. This results in fast response times and can allow more trajectories to be considered as candidates during each time step.

To limit the amount of learning required, the robot's available trajectories are limited to being either a line straight ahead or a preset number of arcs to the left or right of the robot with preset radii. Thus the learning task involves associating a number of instantaneous trajectory velocities (7 in the case of Figure 1) with each input vector state. Although this may require the discovery of more information than learning one to one associations between input vectors and simple command responses, the required knowledge can be obtained from the robot's interactions with the environment without incurring the credit assignment problem or requiring fitness evaluations. Thus, learning is achieved faster than other typical unassisted robot learning methods.

To explain this with an example, consider the task of learning object avoidance behavior using conventional RL. Figure 2(a) shows the path and commands performed by a robot prior to an unintentional collision with an object. When this occurs, there is no certain way of deciding which responses leading up to the collision should have been taken or even how many responses were at fault. So with RL, correct command responses can only be discovered through the repeated assignment of positive or negative credit to those actions suspected of being correct or incorrect respectively. Hence, over long periods of time and after many similar collisions, correct responses may accumulate larger credit and exhibit appropriate actions for this and other similar situations. Unfortunately, when the input space is of considerable size and many possible command responses exist, learning becomes unacceptably slow regardless of the RL algorithm deployed (see Kaebling (1996)). Hence, such learning tasks are viable only when applied to robots with limited sensing and few command responses or alternatively by performing learning in fast simulations.

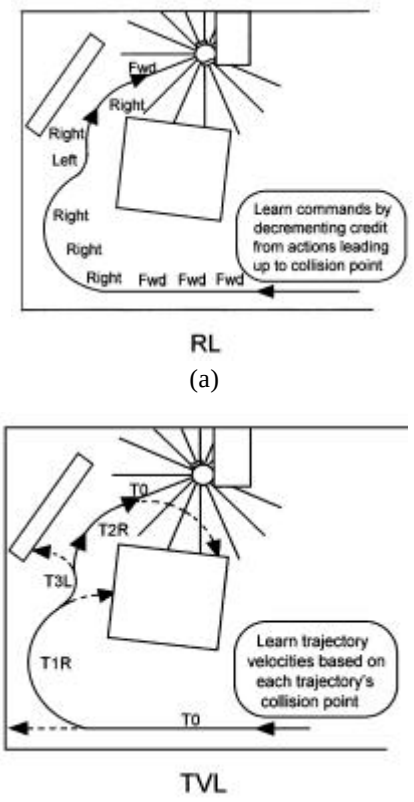
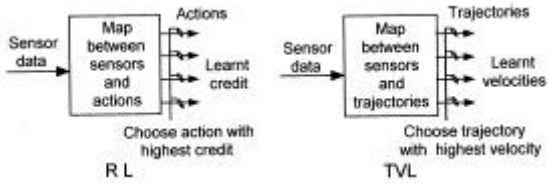


Figure 2. Two different ways a mobile robot can learn from environmental stimuli. (a) Learning appropriate command responses by assigning credit to actions via RL. (b) Learning appropriate trajectory velocities by considering collision points of traversed trajectories.

Alternatively, if the learning task is based on leaming associations between sensor range readings and appropriate instantaneous velocities for traversing trajectories, these can easily be calculated by considering the collision point of each traversed trajectory as shown in Figure 2(b). To obtain sensor-trajectory velocity associations, sensor data is recorded and held until the current trajectory's collision point is obtained. Trajectory collision points can be obtained by using accumulated sensor data and odometry to estimate their positions, or alternatively, by just following each selected trajectory slowly until a collision occurs. Once

the current trajectory's collision point is determined, a preset constant deceleration rate is used to calculate the appropriate velocity for each point leading up to each trajectory's collision point (if any). By associating the calculated velocities with the corresponding sensor data that occurred along the trajectory, training patterns are obtained and used to train an associative map. Thus, if the robot were to follow any similar colliding trajectories, while complying with learnt trajectory velocities, it would come to a safe halt just before coming into contact with the object. Trajectories which do not collide with objects (i.e. complete full circles) can be appropriately associated with a maximum safe velocity.

Therefore by following various trajectories and calculating appropriate velocities based on trajectory collision points, a TVL robot, through its sensors, acquires knowledge of appropriate velocities it should negotiate each of its pre-designated trajectories at any instant. The robot thereby becomes capable of controlling its velocity and making direction choices based on this information. By providing the robot with a single instruction to "*follow trajectories which are perceived to be fastest*", object avoidance behavior automatically becomes exhibited since trajectories which lead into free space will be perceived to have faster velocities than trajectories which collide with nearby objects. Hence, the basic differences between conventional RL and TVL when used to learn object avoidance behavior are: the type of information that is stored, the means by which it is learnt and the manner by which actions are decided as Figure 3 depicts.



(a) (b)

Figure 3. Basic differences between: (a) Reinforcement Learning (RL) and (b) Trajectory Velocity Learning (TVL) for learning object avoidance.

2.1. Learning Multiple Behaviors with TVL

TVL also makes it possible for the robot to produce other behaviors, besides object avoidance behavior (shown in Figure 4(a)), without the need for the robot to learn different associative maps. For example, if we instruct the robot to "*follow fast trajectories which are closest to the nearest detected object*", as in Figure 4(b), wall following behavior becomes exhibited in the direction closet to the robot's forward motion. By "*following fast trajectories closest to the right of nearest object*" produces left wall following behavior and conversely "*following fast trajectories closest to the left of the nearest object*" produces right wall following behavior.

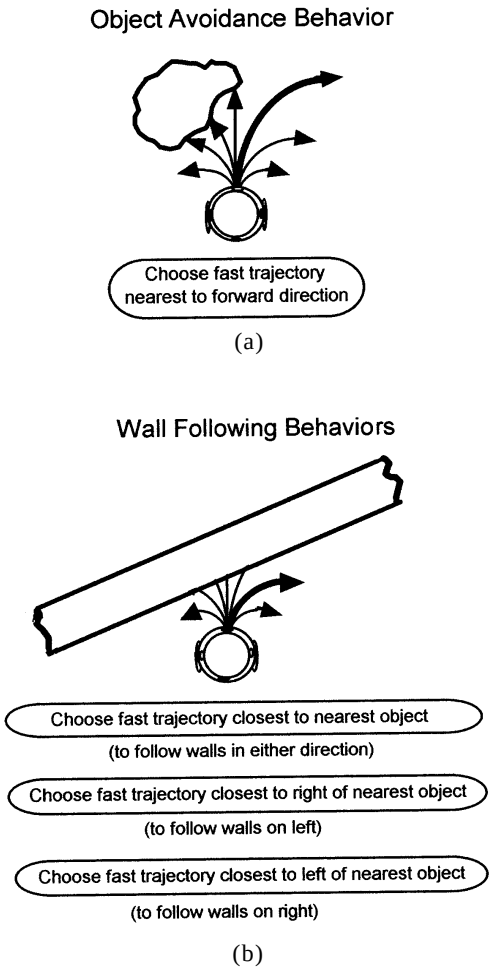


Figure 4. Using TVL to produce multiple behaviors. (a) Object avoidance. (b) Wall following.

Goal seeking behavior also becomes possible if the robot has perception of both its trajectory velocities and the goal location. This is achieved by providing the robot with an instruction to follow fast trajectories toward the perceived goal location. Fortunately, this also produces an implied obstacle avoidance capability without the need to switch behaviors since any convex object encountered between the robot and the goal will cause the direct trajectory to be perceived to be slower than those which lead around the obstacle. A TVL robot like that in Figure 5(a) consequently chooses faster

alternative trajectories resulting in a path around the obstacle being negotiated while still maintaining pursuit of the goal location. However, if the robot encounters a deep crevice, like the one depicted in Figure 5(b), simply following a fast trajectory toward the goal will not escape the crevice. To escape deep local minimums when seeking goals the robot could attempt to follow walls in both directions for increasing periods of time or alternatively could use purposive maps Zelinsky and Kuniyoshi (1996) to learn the shortest path to the goal.

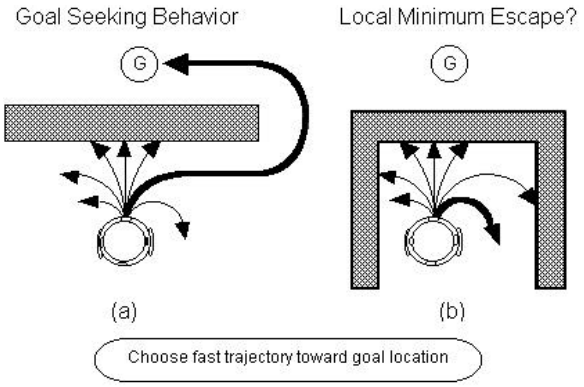


Figure 5. Goal seeking with a TVL robot. (a) Robot successfully avoids object. (b) Robot trapped by deep crevice.

2.2 Adjusting TVL Behaviors

Varied control of the robot's behaviors is also possible by providing a variable velocity threshold to determine if perceived trajectory velocities are considered to be fast or slow. This makes it possible to control the wall clearance distance in wall following behaviors and the object avoidance distance in object avoiding behaviors as shown by the TVL simulator screen dumps in Figure 6. For example, if the velocity threshold is lowered the robot follows trajectories closer to objects before itsvelocity falls below the threshold causing another faster trajectory to be selected. When avoiding objects, as in Figure 6(a), the robot moves cautiously closer to objects before avoiding them. Also, when performing wall following, as in Figure 6(c), a low velocity threshold results in walls being followed more closely and at lower speed. Conversely, raising the threshold causes the robot to maintain larger object clearances and results in the robot moving faster and more competently through the environment when performing its behaviors, as Figure 6(b) and 6(d) show.

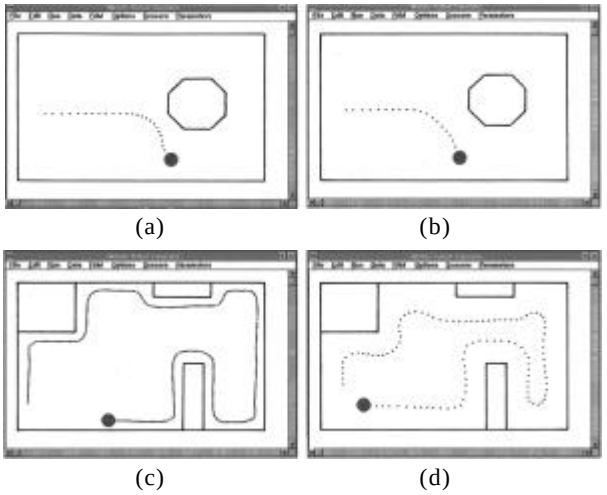


Figure 6. Controlling object clearance distances with a velocity threshold. (a) and (c) Low velocity threshold: small object clearance. (b) and (d) High velocity threshold: larger object clearances.

Figure 7 shows a schematic of TVL and summarizes how mobile robots can become capable of performing certain adjustable behaviors by making simple choices based on learnt trajectory velocities.

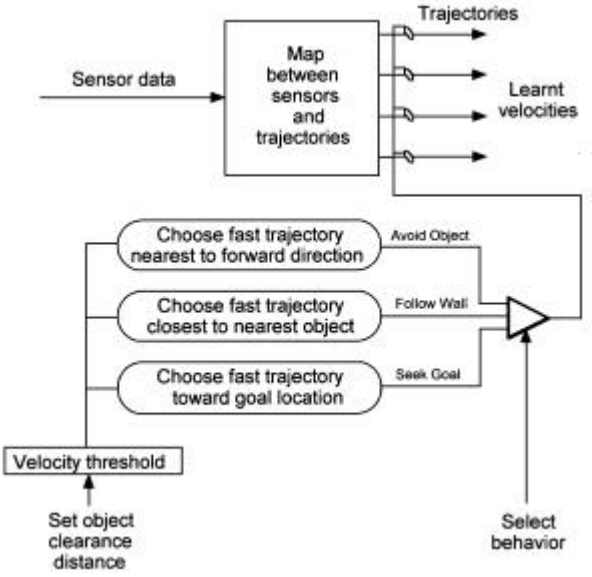


Figure 7. Using TVL to acquire multiple adjustable robot behaviors simultaneously.

If conventional RL or GA robot learning methods were used to acquire the same control and competency, these methods would require each behavior to be learnt in separate maps (or in neural nets). They would require the implementation of adequate reinforcement signals or fitness evaluation functions. It would be difficult for behaviors learnt with RL or GAs to also acquire an ade-

quate means of controlling the robot's velocity. It may not be possible to provide a facility for adjusting the robot's object clearance distance. Learning could also be expected to take a long time due to the credit assignment problem or the fitness evaluation problem. However, RL and GA methods have the advantage that they are suitable for learning a variety of behaviors with a diverse range of sensing devices. Although, TVL (on its own) cannot be expected to acquire complex tasks that have been successfully learnt with RL and GAs (e.g. robot soccer Asada *et al* (1995), box pushing Connel and Mahadevan (1992), hitting a stick with a ball Kalmar *et al* (1998)), TVL potentially could facilitate learning complex task with RL or GAs by providing an effective means of rapidly acquiring essential low level behaviors.

2.3 What Type of Robot Learning is TVL?

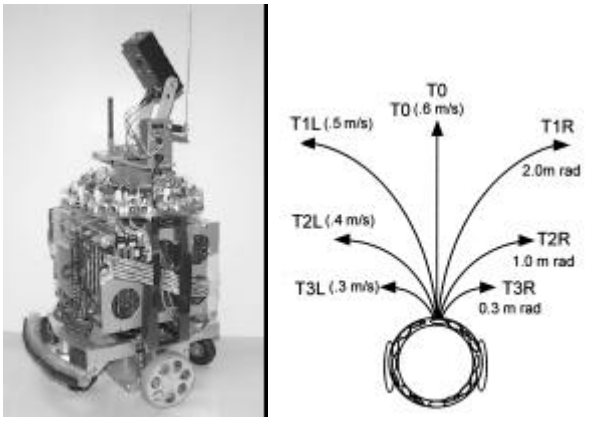
With TVL the robot does not have to perform the desired behaviors or experience collisions in order to learn from its environment. TVL therefore does not use collisions (or the detection of trajectory collision points) as a behavior based reinforcement signal. Instead, collisions are used as a means of obtaining reference points within the environment that enables associations between sensors and trajectory velocities to be directly obtained. Thus, TVL does not learn behaviors via trial and error or suffer from the credit assignment problem and therefore cannot be regarded as a reinforcement learning process. As explained in Section 3 and 4, each sensor-velocity association obtained from the robot's interactions with the environment is used to train Fuzzy Associative Memory (FAM) (or FAMs) via a supervised machine learning process called compositional rule inference (see Sudkamp and Hammell (1994)).

Although, supervised machine learning is used to learn sensor-velocity associations, TVL requires no teacher (either human or from an innate controller) as other typical robot learning methods involving supervised machine learning processes have, e.g. Sharkey (1998) and Tani and Fukumura (1994). This is because the robot does not directly learn actions and therefore has no need for any actions to be demonstrated. TVL instead learns only trajectory velocities which enables a variety of behaviors to be easily produced with simple instructions. For this reason we prefer not to refer to teacher and non-teacher robot learning methods as supervised or unsupervised robot learning respectively, as is often the case in other published works.

The closest form of supervised machine learning applied to robotics that resembles TVL is *Direct Inverse Control*, see Kraft and Campagna (1990) and Guez and Selinski (1988). An example of this is where a robot arm uses neural networks to directly learn the mapping from desired trajectories (of the arm) to control signals (e.g. joint velocities) which yield these trajectories. As with TVL no teacher is required. Associations (or training patterns) are generated by engaging the arm in random trajectory motion. Each training pattern's output is obtained shortly after the training pattern's input is given.

2.4 TVL Robot Setup

To conduct our TVL experiments we use the Yamabico mobile robot Yuta *et al* (1991) shown in Figure 8(a). This robot is equipped with 16 sonar sensors arranged in a ring equally spaced around the robot and a bump sensor for detecting collisions. We provide the robot with seven trajectory locomotion commands for negotiating its environment labeled T3L to T3R on Figure 8(b). Each trajectory has a maximum velocity in the forward and reverse direction appropriate for safely negotiating these trajectories. Stop and spin commands are also utilized to halt the robot when collisions occur or to face the robot in a desired direction.



(a) (b)

Figure 8. (a) Yamabico mobile robot equipped with a 16 sensor sonar ring. (b) Robot's trajectory commands with trajectory radii and maximum velocities shown.

In the following sections, we describe how associations between input vectors and trajectory velocities can be learnt by using Fuzzy Associative Memory (FAM) matrices. We discuss the limitations of using a single FAM to learn a mapping between sensors and trajectory velocities and explain how improved perception and performance can be achieved by using multiple FAMs to individually map sensors to each trajectory.

3. Using FAM Matrices to Map Sensors to Trajectory Velocities

Considerable work in fuzzy control uses a matrix to represent fuzzy rule consequents called a Fuzzy Associative Memory (FAM) matrix (see Kosko (1992) for a concise description). A FAM matrix can be described as an N dimensional table where each dimension represents a specific input. The size of each dimension equals to the number of fuzzy membership functions used to describe the representative input. For example, a FAM matrix with 2 inputs and 3 membership functions for describing each input (e.g. small, medium and high) would be require a table with $3^2 = 9$ entries to store all possible fuzzy rule consequents. Like lookup tables, FAM matrices have the advantage of allowing fuzzy rule consequents to be directly accessed from the input vector which enables their output to be produced quickly. Furthermore, the output is derived by what is effectively a parameterized smoothing (weighted averaging) over a small neighborhood of table entries which provides good generalization and immunity to isolated incorrect entries. This is explained in Section 4.2. FAM matrices also have the advantage of being trainable via a supervised machine learning process called compositional rule inference, see Sudkamp and Hammell (1994). Unlike conventional neural networks, FAM matrices have the advantage of being able to learn and recall associations quickly, are capable of incremental learning and can enable the designer to appropriately divide up the input space. Furthermore, the acquired knowledge within FAMs is capable of being interpreted by examining the fuzzy rules which comprise table entries.

3.1 Using a Single FAM Matrix to Map Sensors to Trajectory Velocities

The main disadvantage with using a FAM matrix to classify robot sensor data is that the size of the matrix increases exponentially with increasing numbers of inputs and fuzzy sets. For example, a FAM which has 16 inputs connected to sensors and 4 membership functions describing each input will require 4^{16} or 4,294,970,000 entries to store all possible rule consequents. This not only will require considerable memory but for learning purposes may also require a lot of data to be learnt in order to fill those entries. One way to effectively reduce the size of the FAM and consequently the amount of data required to fill all entries in the FAM is by grouping sensors together.

In our initial TVL experiments we used a single FAM matrix and arranged the robot's sensors into five groups of three (as shown in Figure 9(a)). We produced the input vector by taking the minimum sensor reading from each sensor group. We also described the input domain using a reduced number of membership functions for inputs at the sides and toward the rear of the robot, as shown in Figure 9(a), so that the robot's perception is more acute toward the front. Furthermore, the structure of the membership functions, shown in Figure 9(b), are concentrated toward the near vicinity of the robot so that range readings of closer objects can be interpreted more accurately. The resulting arrangement allows the total input search space to be described with just $5 * 4 * 3 * 3 * 4 = 720$ possible fuzzy rules.

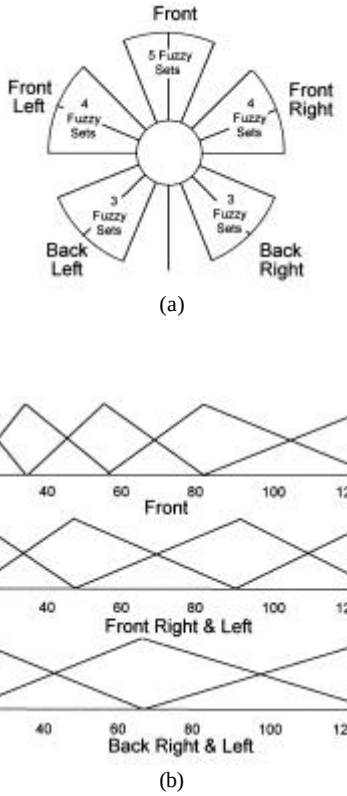


Figure 9. (a) Sonar sensors arranged into five groups with three sensors in each group. (b) Membership functions used to fuzzify the input vector.

Each of the 720 FAM entries is used to store the consequences of each possible rule. So to describe all the appropriate velocities associated with the robot's seven trajectories, a total of $720 * 7 = 5040$ entries are required within the FAM as shown in Figure 10.

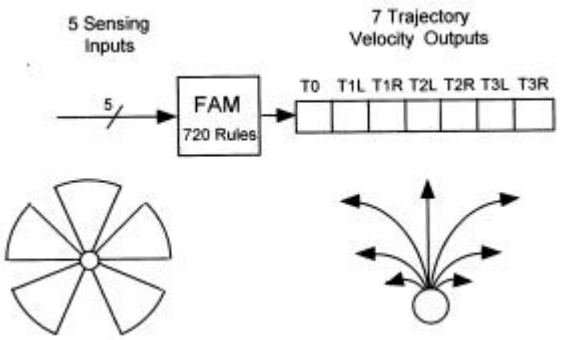


Figure 10. Using a FAM matrix to store velocities of 7 trajectories.

For TVL to work effectively, FAM entries should always be initialized to low velocities. Thus, as learning progresses, the robot becomes increasingly aware of faster trajectories that exist around objects (and into free space) and becomes capable of negotiating the environment via these faster trajectories. If the FAM is instead initialized with fast or random velocities, an inexperienced robot would perceive many trajectories that collide with nearby objects to be inappropriately fast. Consequently, when performing behaviors, the robot would choose to follow these inappropriately fast trajectories at speed and end up colliding heavily with objects. (The requirement for unfamiliar input vectors to always produce low velocities is one reason why conventional neural nets are unsuitable for TVL (see Section 4.3 for further details)).

Although the use of grouped sonar sensors results in the robot having considerably coarse perception, we found the single FAM arrangement (in Figure. 9) to be simple and effective for demonstrating the fast learning times possible with TVL within structured and uncluttered environments. However, for more difficult environments it is better to avoid coarsely grouped sensors by using multiple FAM matrices.

3.2 Using Multiple FAM Matrices to Map Sensors to Trajectory Velocities

To overcome the coarse perception that results from having grouped sonar sensors, we decided to replace the robot's single FAM by providing the robot with seven FAM matrices for storing velocities belonging to each of the robot's seven trajectories as shown in Figure 11. Each FAM matrix receives its own independent input vector which is derived from sensors that are considered the most relevant for detecting objects in the vicinity of the FAM's trajectory. For example the most appropriate sensors for resolving the forward trajectory (T0) would of course be the front sensor as well as some neighboring sensors to the left and right of the front sensor.

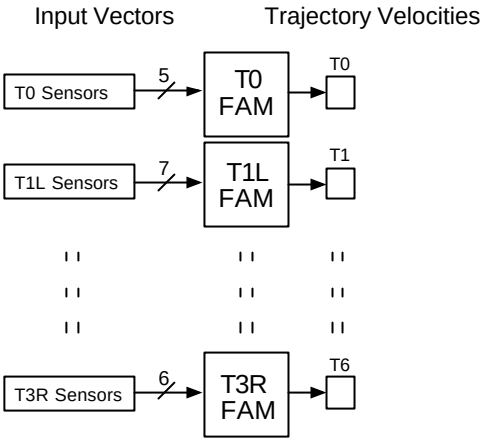


Figure 11. Storing associations between sensors and trajectory velocities in 7 FAM matrices

Although this multi-FAM configuration requires almost seven times as much processing to lookup the robot's seven trajectory velocities, this does not significantly reduce response times for the robot due to the high speed at which FAMs can produce their outputs directly from input vectors. Furthermore, having independent FAMs to store each trajectory's velocities provides increased immunity to sensor damage and can assist the robot in adapting to sensor malfunctions as explained in Ward and Zelinsky (1997).

3.3 Selecting Appropriate FAM Inputs from Available Sensors

To decide which sensors produce the most relevant information for resolving the pathways of individual trajectories, three factors need to be considered: (1) the position of each sensor, (2) the reflective nature of sonar signals and (3) the divergence (or beam angle) of transmitted sonar signals. Although sensors adjacent to a specific trajectory have obvious relevance to that trajectory, due to their position, they will not always return a signal from objects located on the trajectory's pathway. In particular, flat walls can be a problem. For example, Figure 12 shows typically how a flat wall could be detected by a sonar ring.

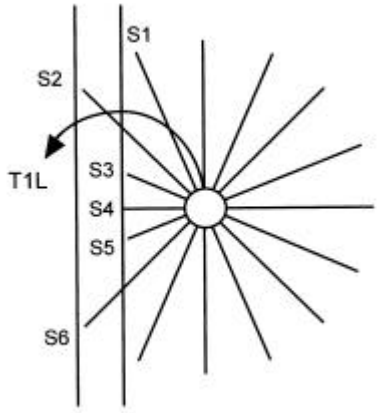


Figure 12. Detecting a flat wall with a sonar sensor ring.

Although the flat wall in Figure 12 lies directly in the path of trajectory T1L, adjacent sensors S1 and S2 fail to return an echo from the wall due to the acute angle of incidence of their respective sonar signals. However, sensors S3 to S5 do detect the presence of the wall due to their transmitted signals being almost normal to the wall. Hence, to be able to resolve appropriate velocities of trajectory T1L, sonars S3, S4 and perhaps S5 should be included (as well as other sensors) as inputs into the FAM matrix of trajectory T1L.

To determine which sensors to include as inputs into each FAM, we placed the robot in various locations and orientations near flat walls and used the above criteria to decide which sensors would be needed to resolve each FAM's trajectory.

Similarly, by considering different robot locations near flat walls, we divided each FAM input into membership functions such that the various collision points of each FAM's trajectory could be resolved with reasonable accuracy. Figure 13 shows how we allocated sensors and fuzzy membership functions to the FAM matrices of trajectories T0 through to T3L. FAM's belonging to trajectories T1R through to T3R are allocated to sensors and fuzzy membership functions in the same fashion as T1L to T3L except their inputs are connected to symmetrically opposite sensors on the right-hand side of the robot.

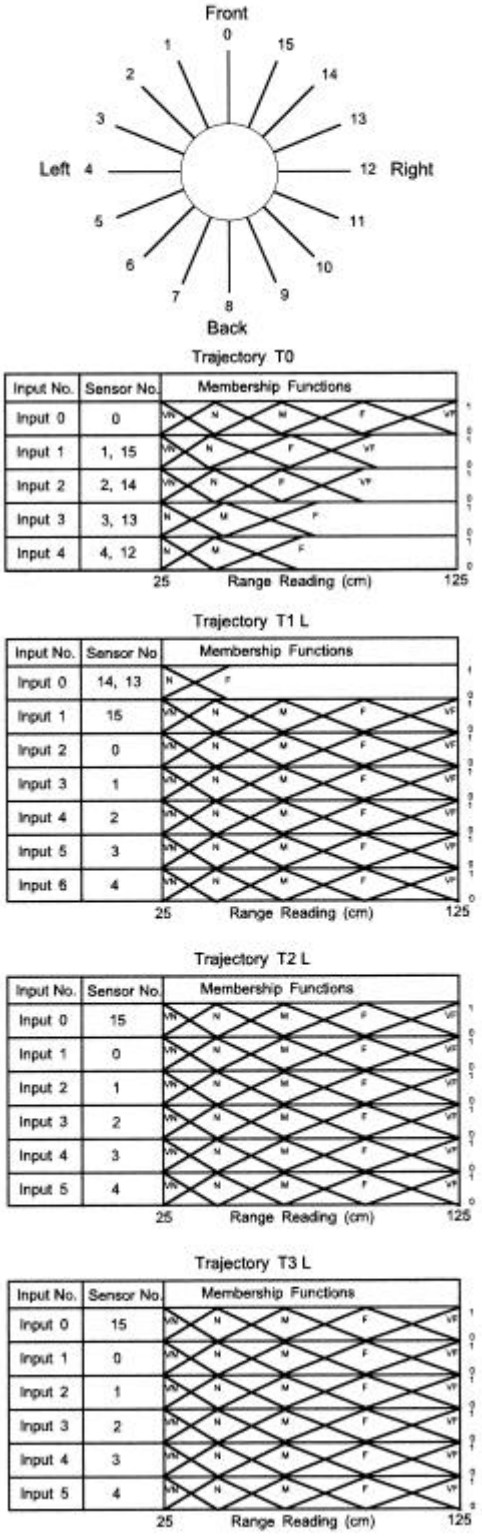


Figure 13. Sensors and fuzzy membership functions designated to the FAM matrices of trajectories T0, T1L, T2L and T3L. (Fuzzy Sets: VN Very Near, N Near, M Medium, F Far, VF Very Far)

Despite configuring the robot's perception to be capable of resolving trajectory velocities with respect to flat walls, extensive experimentation indicated that this arrangement also provides adequate perception of trajectories near most isolated objects and irregular environment features. The reason for this is that in most cases ultrasonic waves are more likely to be reflected back to the receiver from internal corners or irregular surfaces than from continuous flat surfaces that are not normal to most of the sensors on the sonar ring. Consequently, we were able to achieve adequate resolution of our unstructured laboratory environments for robust behaviors to be acquired by the robot in relatively short periods of time.

Increasing the input space of TVL FAMs is not a serious a matter as may be the case with other learning methods. This is because during learning, the robot generates large numbers of training patterns which are able to be immediately loaded into the FAMs via compositional rule inference (see Sudkamp and Hammell (1994)). Thus, the extent to which you divide up the input space of each FAM, is largely a matter of how much memory resources are available and how accurate you would like each FAM to be. The final multi-FAM configuration, described in Figure 13, maps 9 of the 16 sonars to 125,720 fuzzy regions. Although this is nearly 25 times the total number of fuzzy regions allocated to the single FAM configuration described in Figure 9 (i.e. 5,050 regions), acquiring all 3 behaviors in our laboratory environments took only slightly longer than with the single FAM configuration (see Section 5 for details).

Although our experiments demonstrated that FAMs worked well in this application, other supervised learning methods (like CMAC neural networks Kraft and Campagna (1990)) may also be suitable for learning associations between sensors and trajectory velocities on mobile robots. However, if non-FAM learning methods are used, precautions must be taken so that unfamiliar input vectors always produce low velocities as output. Otherwise, high speed collisions with unknown objects could occur (as explained in Section 3.1). To conduct our experiments we implemented this requirement by always initializing the TVL FAMs with low velocities (0.1m/s).

4. Learning Trajectory Velocities

Since TVL is based on learning perception rather than specific actions, there is no need for the robot to perform any of its desired behaviors to acquire the perception needed to perform those behaviors. In fact, the simplest way to learn trajectory velocities is to select trajectories randomly and follow each until a collision with an object or a full circle in free space occurs. Alternatively, trajectory velocities can be found by estimating the collision points of engaged trajectories by using accumulated sensor data and odometry. Although this may require considerable computation and may be less accurate, it has the advantage of enabling the robot to learn while performing any of its behaviors. However, if the robot's motion is constrained by the engagement of a particular behavior, the robot may never venture into some regions of the environment and thus may not learn trajectories associated with input vectors that occur in those regions. For example, a robot that spends most of its time following walls may never learn which trajectory velocities should be associated with input vectors that occur some distance from the walls. One effective learning strategy for overcoming this problem is to encourage the robot to explore by engaging different behaviors.

4.1 An Effective Learning Strategy

To reduce the overhead required to calculate trajectory collision points from accumulated sensor readings, we performed our TVL experiments by following trajectories until a collision with an object is detected or a full circle is completed. When this occurs, the robot is stopped momentarily to update the FAM entries associated with the traversed trajectory. This is done by associating each input vector experienced along the trajectory with appropriate velocities. These velocities are calculated by using a predefined deceleration rate to estimate the appropriate velocity at each point up to the collision point. Thus, if the robot were to repeat the path while complying with the learnt velocities, it would come to a safe halt just prior to the collision point. Velocities associated with trajectories that complete a full circle are set at the trajectory's maximum allowable velocity. Each training pattern is used to incrementally adjust the relevant FAM's trajectory velocities as explained in Section 4.2. Figure 14 describes the basic learning algorithm used to conduct our experiments. Although odometry is required to measure the distances between trajectory collision points and recorded points where sensor readings were taken, these distances are relatively short and therefore unaffected by odometry errors.

loop
scan all trajectories around robot
choose trajectory with least learning experience
rotate robot to face chosen trajectory
repeat
move one time step along chosen trajectory
record input vector
until *collision or full circle occurs*
associate appropriate velocities
with each input vector
update FAM with the resulting training exemplars
if collision occurred reverse robot a short
distance along previous path
end loop

Figure 14. Basic algorithm for learning trajectory velocities.

To speed up learning, the selection of trajectories to follow is done by applying the current sensor data to the FAM matrix and choosing the trajectory which has received the least amount of learning. The amount of learning each FAM entry has received is recorded by accumulating the sum of the respective fuzzy rule's stimulation levels resulting from all training exemplars which map to the entry, i.e.

$$E_i = \sum_{(\mathbf{x},v)\in T_i} mA^i(\mathbf{x}) \qquad (1)$$

where $mA^i(\mathbf{x})$ represents minimum input membership of training pattern T_i with input $\mathbf{X}$. Thus by using the result of Equation (1) each time the robot processes an input vector, the robot is not only able to perceive the learnt velocities associated with the input vector but also the amount of learning responsible for each entry's current value.

4.2 Updating FAM Entries

Various techniques for generating fuzzy rules directly from training data have been reported. Generally, these methods can be classified by the learning technique involved, the most common being neural networks Lin and Cunningham (1995), genetic algorithms Homaifar and McCormick (1995), iterative rule extraction methods Abe and Lang (1993) and direct fuzzy inference techniques Turksen (1992). The most significant differences in these methods lies in the time it takes to generate fuzzy rules and whether or not fuzzy input and output sets need to be predefined. Unfortunately, the methods which do not require the predefinition of fuzzy sets, require far too much training time for on line robot learning to be possible. With this in mind we have chosen to adopt the compositional rule inference approach proposed by Zadeh (1973) and more recently investigated by Sudkamp and Hammell (1994). This approach enables the robot to incrementally adjust its FAM entries while experiencing its environment.

Consequent adjustment of FAM matrices is achieved by accumulating the weighted average of training exemplars which stimulate each consequent entry, namely:

$$V_i = \frac{\sum_{(\mathbf{x},v)\in T_i} mA^i(\mathbf{x})v_i}{\sum_{(\mathbf{x},v)\in T_i} mA^i(\mathbf{x})} \qquad (2)$$

where $mA^i(\mathbf{x})$ represents minimum input membership of training pattern T_i with input $\mathbf{X}$ and $\boldsymbol{v}$ represents the velocity associated with training pattern T_i

Defuzification is performed by using the weighted averaging method with the difference that we use the crisp consequent values rather than representative values from fuzzy output sets to save computation time. Thus the anticipated velocity from each FAM matrix is given by:

$$V = \frac{\sum_{i=1}^{n} mA^i(\mathbf{x})v_i}{\sum_{i=1}^{n} mA^i(\mathbf{x})} \qquad (3)$$

Because FAM entries can be accessed directly from the input vector, both updating FAM entries and inferring trajectory velocities from sensor data can be done quickly regardless of the size of the FAM matrix. This enables the robot to both perceive the velocities of its trajectories and learn trajectory velocities within the real time constraints of normal robot operation.

4.3 Conflicting Training Patterns

Due to the reflective nature of ultrasonic waves and the presence of undetectable features in most environments, there will inevitably be situations where the robot experiences the same sensor data but discovers different velocities for the same traversed trajectory. Some typical examples of this can be seen in Figure 15.

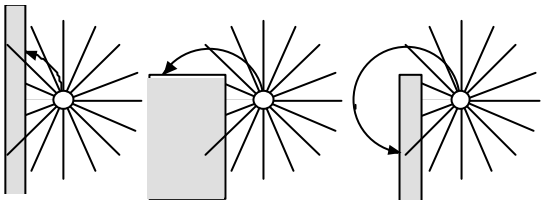


Figure 15. Conflicting training patterns. (i.e. same sensor data but different trajectory velocities).

Our experiments have shown that there was no need to devise sensor pre-processing or conflict resolution

strategies to deal with these problems because most were adequately dealt with by the averaging effect of the FAM updating procedure (explained in Section 4.2). This effectively cancels noise effects in the training data by taking the averages of all readings which map to the same FAM entries. It also tends to adjust FAM entries updated by conflicting training patterns (like those shown in Figure 15) safely downward. This occurs because in typical environments there are usually more internal corners and flat walls than external corners. Therefore, if any conflicting patterns are generated, there will usually be more conflicting patterns associated with lower velocities than with higher velocities. When performing behaviors, the only effect the learnt conflicting patterns have is they cause the robot to fail to perceive the faster alternative trajectories that exist near external corners. Therefore when the robot is near external corners, it may have to move further along its current trajectory before becoming aware of the faster alternative trajectories that exist.

4.4 Symmetrical Learning

Because the robot's sensors and trajectories are symmetrical, the knowledge required in the FAMs belonging to the left and right trajectories can also be considered symmetrical. Thus, any training patterns derived from trajectories on the left can not only be used to update the appropriate left FAM but can also be used to update the opposite right FAM, as shown in Figure 16.

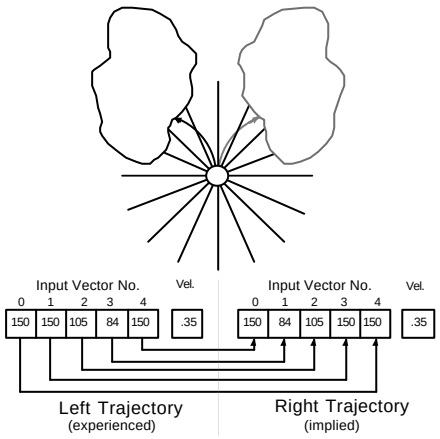


Figure 16. Updating 2 symmetrically opposite FAMs with one training pattern.

4.5 Rotating Foveal Perception

Similarly, training data derived from trajectories on the right can be used to update symmetrically opposite FAMs on the left. This can be used to both speed up learning and provide the robot with more comprehensive perception of its environment. However, special consideration (as explained in Ward and Zelinsky (1997)) has to made if the robot's sensors have significant differences in performance or if some sensors become damaged.

The ability to escape dead-ends when performing behaviors can easily be encoded by providing the robot with rotating foveal perception. This is achieved by enabling the robot to rotate its perception within the sonar ring so that the robot can perceive the environment as if it were facing other directions. For example, if the robot finds itself in a situation where all immediate trajectories are perceived to be slower than the velocity threshold, the robot is able to increases its view of perceived trajectories without physically rotating itself by rotating its perception within the sonar ring. Thus by rotating its perception all the way around the 16 sensors comprising the ring the robot becomes capable of perceiving the velocities of $16 * 7 = 112$ trajectories while maintaining its current direction. This enables the robot to make quick decisions as to what direction it should face when it finds itself in tight situations such as dead-ends. For example, Figure 17 shows what happens when a robot performing wall following enters a dead-end. In this situation all the immediate perceived trajectory velocities lie well below the velocity threshold. So the robot rotates its perception within the ring to find faster trajectories. Consequently, the robot discovers that the nearest trajectory to the closest object that is faster than the threshold is trajectory T3L in the direction of sensor 6. In response, the robot rotates counter-clockwise to face the direction of sensor 6 and then proceeds along trajectory T3L at the appropriate velocity.

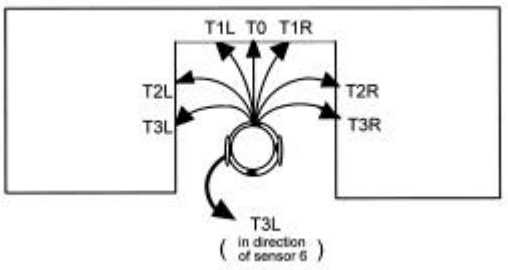


Figure 17. Rotating the robot's perception virtually within the sonar ring to locate faster trajectories.

5. Experimental Results.

We conducted a number of learning experiments in the four structured environments shown in Figure 18 (a to d), These environments were primarily used for testing the accuracy of behaviors, comparing the performance of different FAM configurations and for observing the effects of adjusting the velocity threshold parameter. The unstructured lab environment, Figure 18 (e & f), was used to test the robot's ability to acquire competences in indoor environments where a diverse range of input vectors are possible and features exist that are difficult for sonar sensors to detect.

(a) (b) (c)

(d) (e) (f)

Figure 18. Environments used to perform TVL robot experiments. (a) - (d) Structured environments. (e) - (f) The robot laboratory.

Prior to each learning trial we initialized all FAM velocity entries to 0.1 m/s and the velocity threshold to 85% of trajectory maximums. To learn behaviors within each of these environments we engaged the robot in exploratory learning (as explained in Section 4.1) and monitored the robot's competence by periodically switching the robot's behavior to object avoidance and wall following to see if the robot could perform these behaviors without colliding with walls or objects. To determine the amount of learning occurring within the robot at any moment we measured the average change of all the velocity entries within the FAM matrices per time step for each minute of learning time i.e.

$$\Delta V = \frac{\sum_{i=1}^{n} v_i - v_{i-1}}{n} \qquad (4)$$

where $v_i - v_{i-1}$ is the total change occurring in all FAM velocity entries during each time-step and n is the number of time steps occurring in each minute of learning.

Because we initially set all FAM entries to low velocities (i.e. 0.1m/s), the robot's increasing competence can be measured by monitoring the robot's average velocity in addition to its collision rate.

5.1 Single FAM TVL Experiments

We conducted a number of TVL experiments within the structured environments shown in Figure 18 with the single FAM configuration described in Section 3.1. Generally, the robot acquired competent behaviors in less than 15 minutes of learning time. However, some trials took up to 20% longer than others due to the random manner by which trajectories were selected during learning. Often, this resulted in similar trajectories being traversed many times before unfamiliar trajectories were experienced. The graphs in Figure 19 show the average learning rate and acquired competence of 5 trials conducted in the circular, square and corridor environments.

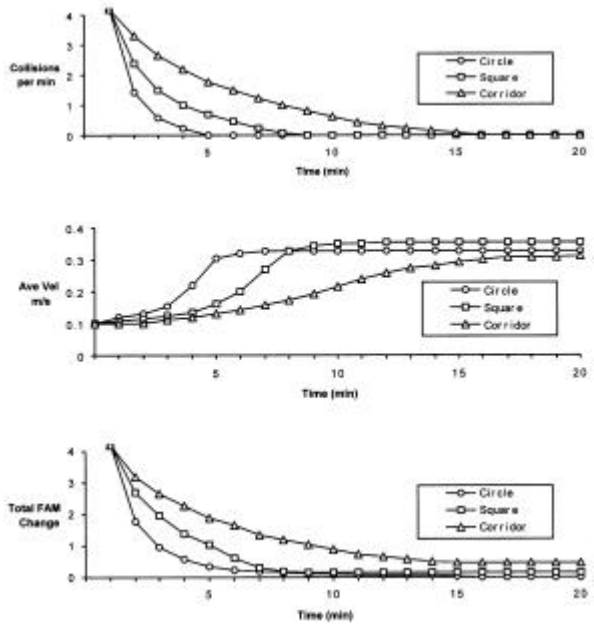
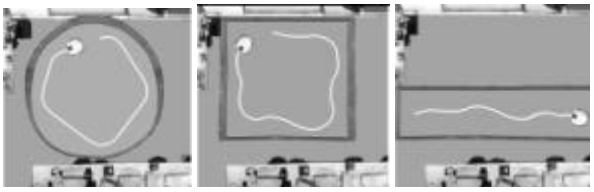


Figure 19. Learning curves and performance measures for the circular, square and corridor environments with the single FAM configuration.

Although the robot generally acquired the ability to avoid collisions quickly (e.g. less than 9 min in the circular and square environments), inappropriate trajectories were regularly selected (when performing behaviors) until almost all FAM entries ceased to change. This caused the robot to turn too early, too late or too much in order to avoid collisions or get nearer to walls. Also, for all trials the final competence acquired from the square environment also worked reasonably well in the circular environment and visa versa. Although this demonstrates fast learning in simple environments as well as the generalizations possible the single FAM configuration, the robot's coarse perception appeared to limit the robot's ability to maintain parallel paths to walls and consistent wall clearance distances while performing the wall following behavior. Figure 20 shows typical wall following paths exhibited by the robot in the circular, square and corridor environments.



(c) (b) (c)

Figure 20. Wall following behavior within (a) circular (b) square and (c) corridor environments.

Adjusting the robot's velocity threshold after learning, (as explained in Section 2.2) produced significant changes to the average velocity and wall clearance distances while performing object avoidance and wall following behaviors. The results are shown by the graphs in Figure 21.

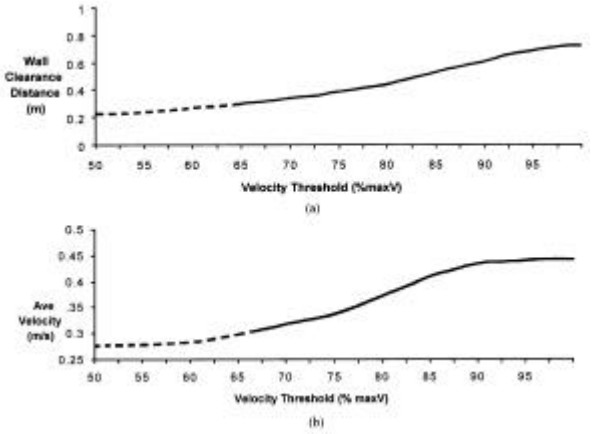


Figure 21. Relationship between trajectory velocity threshold and: (a) the average wall clearance distance, (b) the average velocity, when performing wall following in the circular and square environments (Dashed lines indicates collision prone behavior).

Varying the threshold from 65% to 95% of trajectory maximums resulted in a corresponding change in the wall clearance distance of between 0.32 meters to 0.67 meters at an average velocity of 0.32 m/s to 0.45 m/s. We found following walls at closer distances to be possible with lower thresholds, however, the inability of the single FAM robot to maintain consistent wall clearances resulted in increased collisions as the threshold was further reduced.

We conducted goal seeking trials with the environment shown in Figure 22. This was done by firstly placing objects at set positions in the environment and performing learning until competent object avoidance behavior was produced. The robot was engaged in goal seeking behaviour by providing it with an instruction to follow fast trajectories toward a set goal location (as described in Section 2.1). Relative positions of the robot and goal were maintained by using odometry. Although this provided only limited accuracy, we found it sufficient to demonstrate the acquired goal seeking capability of the robot within the environment due to the short distances involved and the brief duration of the trials. Because of the greater number of input vectors possible within environments containing obstacles

learning took approximately 50% longer than in the same environment without objects.

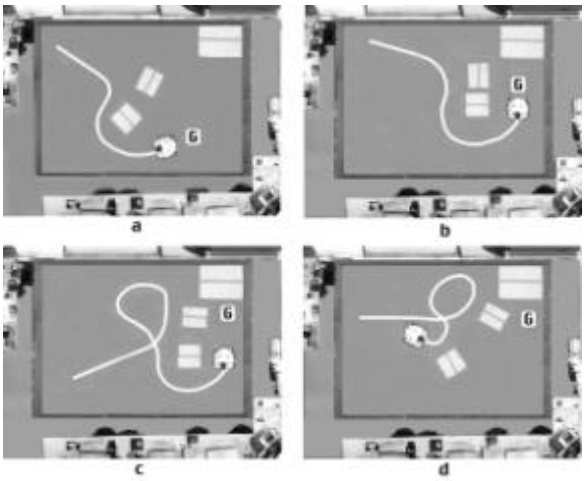


Figure 22. Goal seeking paths exhibited by robot. (a) - (c) goals successfully achieved. (d) unsuccessful goal seeking attempt.

All goals depicted in Figure 22 were successfully found without collisions except for the case shown in Figure 22(d). This situation produced a local minimum despite the gaps between the objects being wide enough for the robot to pass. Examination of the sonar data revealed this to be mainly due mainly to the robot's coarse perception which makes narrow gaps hard to resolve.

When learning trials were attempted in the laboratory environment (with the structured artifacts removed) the single FAM configuration had considerable difficulty learning how to avoid all collisions. Further examination of the sonar data and FAM contents indicated two reasons for this: (1) some objects such as the sharp edges of tables are difficult for a sonar ring to detect when approached from certain angles as shown in Figure 23, and (2) by having coarsely grouped sonars the robot had difficulty differentiating its orientation with respect to nearby objects. For example, in some positions the robot could be turned as much as 15 degrees with respect to nearby objects without any change occurring to the input vector. Hence, some input vectors become associated with fast velocities by experiencing near misses during learning. When the same input vectors occurred while performing behaviors the fast learnt trajectories sometimes lead to collisions with objects. (Note: we found this problem could be improved to some extent by using the sonar sensors to detect collision points and defining a collision to be further from objects when learning than when performing behaviors.)

Figure 23. Some environment features that are hard for a sonar ring to detect.

5.2 Multi-FAM TVL Experiments

To improve the robot's coarse perception caused by having coarsely grouped sonar sensors, we provided the robot with 7 FAM matrices as described in Section 4.2. Although all the FAMs in this configuration (except FAM T0) are considerably larger than the single FAM configuration (e.g. Trajectory T1L with the multi-FAM has 31,250 entries as opposed to 720 for the single FAM) the learning time needed to produce competent behaviors was only 10% – 20% longer within the same environments. For example, Figure 24 compares the average learning times from 5 trials in the square environment with both the single and multiple FAMs being trained simultaneously from the same

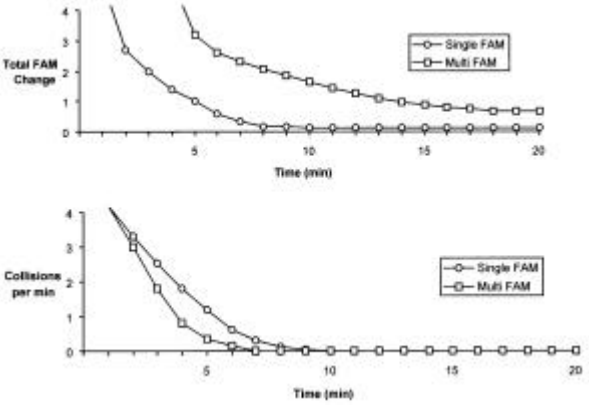
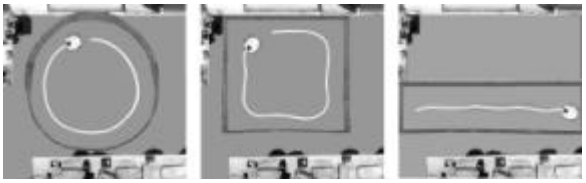


Figure 24. Average learning time and collision rate for single and multiple FAM configurations within the square environment.

We found the relatively small difference in learning times occurred because the robot generates large numbers of training patterns during learning. Many of these redundantly map to same FAM entries in the single

FAM whereas they tend to become distributed over more FAM entries in the multiple FAMs. Although significant fluctuations in the multi-FAM's velocity entries continued to occur with ongoing learning this appeared to have no effect on the robot's behaviors.

We found the robot's capabilities to be greatly improved with the multi-FAM configuration. Narrow spaces could be negotiated with relative ease, there were fewer collisions and the robot exhibited more precise motion after similar learning periods. Figure 25 shows typical wall following pathways exhibited by the multi-FAM robot in the circular, square and corridor environments.



(a) (b) (c)

Figure 25. Wall following behavior within (a) circular (b) square and (c) corridor environments with the multi-FAM configuration.

The ability of the robot to negotiate closely paced objects was also greatly improved. Goals that were out of reach to the single FAM robot due to the presence of narrow gaps (e.g. Figure 22(d)) could now be successfully reached by the multi-FAM robot as shown in Figure 26(a). However, where environments contained a goal in a narrow passageway, like that shown in Figure 26(b), the robot would also fail to enter the passageway and reach the goal if the robot did not happen to venture into the passageway during learning. To obtain consistent results (in Figure 26(b)) over 5 trials, we always commenced learning by starting the robot close to the passage entrance and by facing it toward the goal. Thus, the robot would first learn faster trajectories at the entrance and within the passageway before proceeding on to learn the rest of the environment.

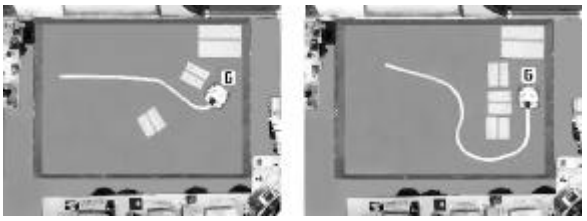


Figure 26. Goal seeking behavior with the multi-FAM robot. (a) Obtaining a goal that the single FAM robot could not reach. (b) Negotiating narrow passages to reach a goal.

Similarly, learning trials conducted within the laboratory (with the structured artifacts removed) also demonstrated improvement performance when compared with that of the single FAM configuration. The average learning time and acquired competence of 5 consecutive trials for single and multiple FAM configurations are shown in Figure 27.

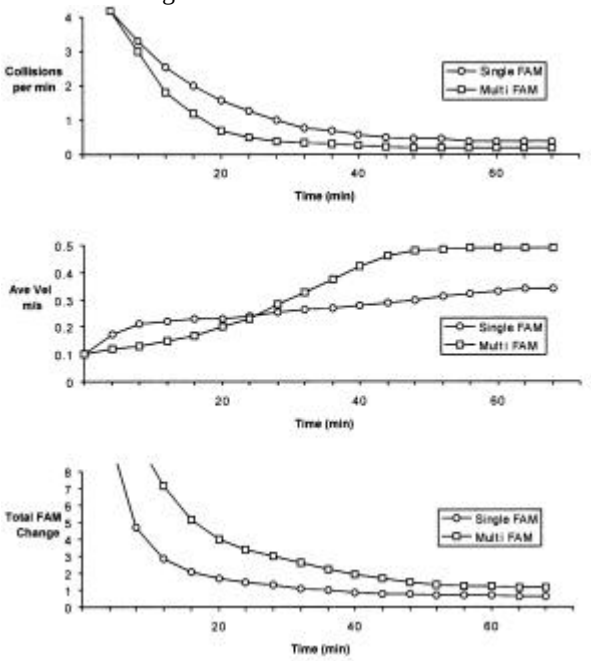
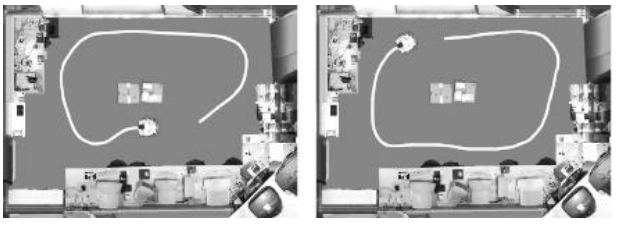


Figure 27. Comparison of learning times and competence measures performed in the laboratory environment with the single and multiple FAM configurations.

Although some change was still occurring to the FAMs after 60 minutes of learning, the robot's behaviors appeared to stop improving approximately 45 minutes after learning commenced. During this time the robot's average velocity at performing behaviors increased from 0.1 m/s to 0.5 m/s. Figure 28 shows typical wall following and object avoidance paths exhibited by the robot in the lab after 45 minutes of leaming.



(a) (b)

Figure 28. Typical paths exhibited by the multi-FAM robot after 45 minutes of learning in the laboratory environment (a) object avoidance, (b) wall following.

Despite the improved perception provided by the multi-FAM configuration, the robot still experienced occasional collisions even after 60 minutes of learning. This was due to the presence of object features which are difficult to detect using sonar sensors, (as explained in Section 5.1). We also found that when ongoing learning was done in environments that contained many features that are hard for sonar sensors to detect, the robot would actually reduce the speed of its free space motion when performing its behaviors. This occurs because an increased number of training patterns with larger range readings tended to become associated with slower velocities.

Although this may appear to corrupt the learning process by making the robot appear slower and less competent at performing its behaviors, this can in fact be considered an appropriate way for an adaptive agent to respond to a difficult environment. For example, if an adaptive life-form were to experience regular collisions with objects which are hard for its senses to detect, cautious motion (as well as fear of collision and reduced competence) would similarly become exhibited. Furthermore, when the undetectable features were physically tagged to make them visible to the sonar ring, further learning resulted in restoration of the robot's speed and competence.

We also conducted experiments in more cluttered environments using the same learning algorithm. However, learning times and final competences were both inconsistent and relatively poor for any worthwhile results to be reported. Even after hours of learning, the robot was often unable to negotiate much of its environment no matter what behavior was selected. The difficulty with learning cluttered environments was due mainly to our trajectory learning algorithm confining the robot to small sections of the environment during learning. Thus, the robot would often fail to learn many unique features and narrow passages because they were not experienced during learning. With further work, we hope to overcome this by implementing learning algorithms that base trajectory selection (while learning) on unfamiliar environment regions. Or alternatively, by engaging the robot in obstacle avoidance (or wandering) behavior and by using accumulated sensor data and odometry to estimate trajectory collision points for the purpose of adaptively learning trajectory velocities.

6. Summary and Conclusion

Although various unassisted robot learning methods have been around for some time now, they generally result in long learning times when used to learn behaviors on real robots. The main reasons for this are the credit assignment problem in reinforcement learning methods and the robot fitness evaluation problem in evolutionary or classifier techniques.

To avoid the credit assignment problem and the fitness evaluation problem in existing unassisted robot learning methods, we have developed a mobile robot learning technique based on learning associations between sensor range readings and appropriate trajectory velocities. Appropriate trajectory velocities are determined in relation to trajectory collision distances with nearby objects. This enables the robot to learn to perceive its environment in terms of trajectory velocities so that trajectories that collide with nearby objects are perceived to be slower than those which lead into free space.

To learn trajectory velocities the robot has to detect the collision points of engaged trajectories from which appropriate velocities are calculated. These are then associated with sensor data that occurred along the negotiated trajectory and used to train fuzzy associative maps. For the purpose of conducting our experiments collision points were detected by following engaged trajectories until a collision or full circle occurred. Once the robot has sufficiently learnt to perceive its environment in terms of trajectory velocities, object avoidance, wall following and goal seeking behaviors can be performed by giving the robot instructions like *"follow fast trajectories nearest to forward direction"*, *"follow fast trajectories nearest to closest object"* and *"follow fast trajectories nearest to goal location"* respectively.

We tested this learning technique on a Yamabico mobile robot equipped with 16 sonar sensors and a bump sensor. We provided the robot with 7 trajectory commands. Therefore, the robot's learning task was to learn a mapping between its sonar sensors and 7 trajectory velocities. To effectively learn sensor-velocity associations, FAM (Fuzzy Associative Memory) matrices Koza (1991) were used and trained via compositional rule inference Sudkamp and Hammell (1994).

Our experiments demonstrated that the robot was able to effectively learn uncluttered environments and could produce all behaviors in relatively short periods of time (10-15 min. in structured environments, 30-50 min. in the lab environment). Attempts to learn cluttered environments were less successful. Many, obstacles would tend to confine the robot to small sections of the environment during learning. This prevented the robot from learning many unique features in the environment.

Although more work is required before trajectory velocity learning can be applied to robots for performing useful tasks, the experiments described in this paper demonstrate that this approach to learning certain mobile robot behaviors has considerable benefits and cost savings. It can enable mobile robots equipped with range sensing devices to acquire object avoidance, wall following and goal seeking behaviors simultaneously and relatively quickly in unknown environments. At the same time, the robot also learns to appropriately control of its velocity. The object clearance distance of all behaviors can be adjusted by changing a single velocity threshold parameter. Dead end escape behavior can also be incorporated into the robot by rotating the robot's perception within the sonar ring when all immediate trajectories are perceived to be below the velocity threshold. TVL has the disadvantage that it requires range sensors or range readings derived from other types of sensors. Also, TVL is limited to learning behaviors that can be described in terms of fast trajectories near to some predefined criteria. For this reason, when it comes to learning complex tasks with robots, TVL can only be expected to facilitate other learning techniques or control architectures by providing a means of rapidly acquiring certain low level behaviors.

References

Abe, S. and Lang, M.S. 1993. A Classifier using fuzzy rules extracted directly from numerical data, *Proc. 2nd IEEE Int. Conf. on Fuzzy Systems*, pp. 1191-1198.

Asada, M. Noda, S. Tawaratsumida, S. and Hosoda, K. 1995. Vision-based reinforcement learning for purposive behavior acquisition, *IEEE Int. Conf. on Robotics and Automation*, pp. 146-153.

Castellano, G. Attolico, G. Stella, E. Distante, A. 1996. *Automatic generation of rules for a fuzzy robotic controller*, Proc. IEEE IROS '96, pp. 1179-1186.

Connell, J. and Mahadevan, S. 1992. Rapid task learning for real robots, in *Robot Learning* by J Connell, Kluwer Academic Publishers.

Floreano D. and Mondada, F. 1994. Automatic creation of an autonomous agent: genetic evolution of a neural-network driven robot, In *From Animals to Animats III*, Cambridge, MA: MIT Press.

Floreano D. and Mondada, F. 1996. Evolution of homing behavior in a real mobile robot, *IEEE Transactions on Systems, Man, and Cybernetics*, 26(3):396-407.

Floreano, D. and Mondada, F. 1995. Evolution of neural control structures: some experiments on mobile robots, *Robotics and Autonomous Systems*, 16:183-195.

Fox, D. Burgand, W. and Thrun, S. 1997. The dynamic window approach to collision avoidance. *IEEE Robotics & Automation Magazine*, 4(1):23-33.

Grenfenstette, J. and Schultz, A. 1994. An evolutionary approach to learning in robots. *In: Proc. of the Machine Learning Workshop on Robot Learning*, New Brunswick, NJ.

Guez A. and Selinski, J. 1988. A trainable neuromorphic controller, *Journal of Robotic Systems*, August.

Holland, J. H. 1986. Escaping brittleness: the possibilities of general purpose learning algorithms applied to parallel rule-based systems. In *Machine learning, an artificial intelligence approach*. (Vol. II), Los Altos: Morgan Kaufmann.

Homaifar A. and McCormick. E. 1995. Simultaneous design of membership functions and rule sets for fuzzy controllers using genetic algorithms. *IEEE Trans. on Fuzzy Systems*, .3(2):129-138.

Jakobi, N. 1994. Evolving sensorimotor control architectures in simulation for a real robot, *Masters thesis*, University of Sussex, School of Cognative and Computer Sciences.

Jakobi, N. Husbands, P. and Harvey, I. 1992. Noise and the reality gap: the use of simulation in evolutionary robotics, *Proc. of the 3rd Int. Conf. of Artifical Life*, Springer, Berlin, pp. 110-119.

Kaelbling. L.P. 1995. An adaptive mobile robot. *Proc. of the 1st European Conf. on Artificial Life*, pp. 41-47, Cambridge, MA, MIT Press.

Kaelbling, L. 1995. *Reinforcement learning in embedded systems*, MIT Press.

Kaelbling. L. P. 1996. Reinforcement learning: a survey. *Journal of Artificial Intelligence Research*, 4:237-285.

Kalmar, Z. Szepesvari, C. and Lorincz, 1998. A. Module based reinforcement learning: experiments with a real robot, *Autonomous Robots*, 5(3/4):273-295.

Khatib, O. 1985. Real-time obstacle avoidance for manipulatios and mobile robots. *Proc. of the IEEE Int. Conf. on Robots and Automation*, pp. 500-505, St. Louis.

Kosko. B. 1992. *Neural networks and fuzzy systems*, Englewood Cliffs, NJ, Prentice Hall, Inc.

Koza, J. 1991. Evolution and co-evolution of computer programs to control independent acting agents. *Proc. of the 1st Int. Conf. on the Simulation of Adaptive Behavior*. Cambridge, MA, MIT Press/Bradford Books.

Kraft, L.G. and Campagna, D. 1990. A summary comparison of CMAC neural networks and traditional adaptive control systems, *Neural Networks for Control*, MIT Press, Cambridge, MA.

Krose, B.J.A. and Evans, M. 1994. A self-organizing representation of sensor space for mobile robot navigation. *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pp. 9-14.

Lin, Y. and Cunningham, G.A. 1995. A new approach to fuzzy-neural system modeling, *IEEE Transactions on Fuzzy Systems*, 3(2):190-197.

Materic, M.J. 1997. Reinforcement learning in the multi-robot domain, *Autonomous Robots* 4(1).

Materic, M.J. and Cliff, D. 1996. Chalanges in evolving controllers for physical robots, *Robotics and Autonomous Systems*, 19:67-83.

Michaud, F. and Matrec, M.J. 1998. Learning from history for behavior-based mobile robots in non-stationary conditions, *Autonomous Robots*, 5(3/4):335-354.

Meeden. L. A. 1996. An incremental approach to developing intelligent neural network controllers for robots. *Trans. on Systems, Man and Cybernetics*, 26(3):474-485.

Nehmzow, U. Smithers, T. & Hallam, J. 1990. Steps toward intelligent robots, *DAI Research Paper No. 502*, Dept. of Artificial Intelligence, Edinburgh University.

Nolfi, S. Floriano, D. Miglino, O. and Mondada, F. 1994. How to evolve autonomous robots: different approaches in evolutionary robotics, *Proc. of Artificial Life IV*, pp. 190-197.

Pomerleau, D.A. 1993. *Neural network perception for mobile robot guidance*, Kluwer Academic Publishers.

Schmidhuber, J.H. 1996. A general method for multi-agent learning and incremental self-improvement in unrestricted environments. in *Evolutionary Computation: Theory and Applications*. Scientific Publishing Co., Singapore.

Schultz, A.C. 1991. Using a genetic algorithm to learn strategies for collision avoidance and local navigation, *Proc. of the 7th Int. Symposium on Unmanned, Untethered, Submersible Technology*, Durham, NH, pp. 213-225.

Sharkey, N.E. 1998. Learning from innate behaviors: a quantitative evalution of neural network controllers, *Autonomous Robots*, 5(3/4):317-334.

Sudkamp, T. and Hammell, R.J. 1994. Interpolation, completion and learning fuzzy rules, *IEEE Trans. on Systems, Man and Cybernetics*, 24(2):332-342.

Singh, S.P. Barto, A.G. Grupen, R.A. and Connolly, C.I. 1993. Robust reinforcement learning in motion planning with harmonic functions, *Proc. of NIPS '93*.

Tani J. and Fukumura, N. 1994. Learning goal directed sensory-based navigation of a mobile robot, *Neural Networks*, 7(3):553-563.

Turksen, I.B 1992. Fuzzy second generation expert system design for IE/OR/MS. *Proc. of IEEE Int. Conf. on Fuzzy Systems*, pp. 797-786.

Watkins. C. J. 1989. Learning from delayed rewards. *PhD Thesis*, King's College Cambridge UK.

Ward, K. and Zelinsky, A. 1997. An exploratory robot controller which adapts to unknown environments and damaged sensors. *Int. Conf. on Field and Service Robots*, pp. 477-484 Canberra, Australia.

Yuta, S. Suzuki, S. and Iida, S. 1991. Implementation of a small size experimental self-contained autonomous robot: sensors, vehicle control and description of sensor behavior, *Proc. of 2nd Int. Symposium on Experimental Robotics*, pp. 25-27, Toulouse, France.

Wyeth, G. 1998. Training a vision guided mobile robot, *Autonomous Robots*, 5(3/4):381-394.

Zadeh, L.A. 1992. Outline of a new approach to the analysis of complex systems and decision processes, *IEEE Trans. on Systems, Man and Cybernetics*, 3:779-786.

Zelinsky, A. and Kuniyoshi, Y. 1996. Learning to coordinate behaviors in mobile robots, *Journal of Advanced Robotics*, 10(2):143-159.

Koren Ward received her BCompSc degree in Computer Science from the University of Wollongong in 1995. Currently she is a PhD candidate at the University of Wollongong working on the development of rapid alternatives to existing robot behaviour learning methods. Her research interests include knowledge discovery from data, internet robotics and human-computer interfaces.

Alexander Zelinsky was born in Wollongong, Australia in 1960. He worked for BHP Information Technology as a Computer Systems Engineer for 6 years before joining the University of Wollongong, Department of Computer Science as a Lecturer in 1984. Since joining Wollongong University he has been an active researcher in the robotics field and obtained his PhD in robotics in 1991. He spent nearly 3 years (1992-1995) working in Japan as a research scientist with Prof. Shinichi Yuta at Tsukuba University, and Dr. Yasuo Kuniyoshi at the Electrotechnical Laboratory. In March 1995 he returned to the University of Wollongong, Department of Computer Science as a Senior Lecturer. In October 1996 he joined the Australian National University, Research School of Information Science and Engineering as Head of the Robotic Systems Laboratory, where he is continuing his research into mobile robotics, co-operative multiple robots and human-robot interaction. In January 2000 Dr. Zelinsky was promoted to Professor and Head of Systems Engineering at The Australian National University. Prof. Zelinsky is a member of the IEEE Robotics and Automation Society and is currently President of the Australian Robotics & Automation Association.