Relative Positioning for Team Robot Navigation

Patrick McDowell, Brian Bourgeois, Pamela J. McDowell
Naval Research Laboratory
Stennis Space Center, MS 39529
Patrick.McDowell@nrlssc.navy.mil

S.S. Iyengar, Jianhua Chen
Dept. of Computer Science
Louisiana State University
Baton Rouge, LA

Abstract—

The research presented in this paper approaches the issue of robot team navigation using relative positioning. With this approach each robot is equipped with sensors that allow it to independently estimate the relative direction of an assigned leader. Acoustic sensor systems are used and were seen to work very effectively in environments where datum relative positioning systems (such as GPS or acoustic transponders) are typically ineffective. While acoustic sensors provide distinct advantages, the variability of the acoustic environment presents significant control challenges. To address this challenge, directional control of the robot was accomplished with a feed forward neural network trained using a genetic algorithm, and a new approach to training using recent memories was successfully implemented. The design of this controller is presented and its performance is compared with more traditional classic logic and behavior controllers.

Keywords-component; formations, acoustic positioning, neural networks

1 Introduction

This paper focuses on the development of control algorithms and sensor schemes required for robots to accomplish formation maneuvering using vessel relative positioning. Vessel relative positioning, i.e. positioning relative to one's neighbor(s), is examined as a viable alternative for close-in maneuvering because of the complexity, physical limitations and external infrastructure typically required for each robot to accurately determine its position with respect to a common datum. Acoustic sensors are used in this work since they are the only viable alternative for underwater applications, the ultimate target for this research. While acoustic systems can typically be created with inexpensive hardware and fairly simple signal processing (the system used for this research uses a standard computer sound card), the inherent variability of acoustic signals requires the development of more sophisticated control algorithms that are able to adapt to changing conditions. The underlying goal of this research is the development of new control algorithms capable of in situ learning.

In this work classic logic, behavior based and neural network controllers have been developed to control multiple vehicle formations. They are fashioned after biologically inspired formations observed in nature, most notably lines of ducks and caterpillars. The classic logic control systems used were modeled after Braitenburg machines [1] and served as a baseline capability for comparison. The behavior base controller was

constructed to overcome a limitation observed in the strictly reactive classic logic controller. This work includes a baseline study of the feasibility of using a neural network as a navigation controller; this was undertaken since neural networks more easily lend themselves to machine production than competing symbolic methods such as automated code writing using C or Lisp, and hold the promise of being the foundation for a real-time adaptive system. This work includes a new approach to the training of neural networks using a genetic algorithm that finds both the network weights and node transfer functions. The initial feasibility study was done strictly in simulation, allowing exhaustive testing of the solution space to demonstrate the generation of a viable navigation controller. Since exhaustive testing is impractical for physical applications, two methods of training an actual robot were developed – the first used an operator supervised method to create a training set and the second introduces the concept of using recent memories as the basis for constructing the training set.

The next section discusses the four main categories of formation maneuvering described in the literature and expands upon the topic of relative navigation and the use of acoustic sensors. The following section presents the control methods and algorithms developed in this work, followed by section IV that describes the sensor hardware and signal processing algorithms which generate the data used by the controllers. Section V presents the results and observations from the computer simulations and testing with land robots.

2 Background

Coordinating multiple autonomous vehicles moving in formation has become an active area of investigation in robotics, multi-agent systems, and control. A formation is important from a robotic standpoint in that it can be a key part of getting a team of robots to work together. Using formations helps 1) team members track other team members, 2) ensure that no members get lost, 3) prevent collisions, 4) reduce problems with common sensor interference and can be used to increase effective sensor footprints for search and survey tasks. It also augments communication by reducing the distance that a team member will have to transmit or relay a message, which is particularly important for underwater applications where communication bandwidths are severely limited.
In general, existing works on formation control fall into one of the following 4 categories:

1. Leader-follower approach: each vehicle (except a special leader vehicle) follows one or more designated leader vehicles with required relative position and orientation. The group in University of Pennsylvania has done extensive work in this leader-follower paradigm [2-5]. In [2], control laws have been developed for vehicles that follow one or two leaders. The stability of the control laws has been established.

2. The behavior-based approach [6] motor schemas (algorithms) are designed for each of the basic behaviors (such as keeping formation, avoiding obstacles, etc.), and the control command (heading and speed of the vehicle) is obtained by a weighted combination of the basic behaviors.

3. The potential field approach [7-9]: this approach associates an artificial potential field with a group of vehicles. The potential and the interaction force between a vehicle and its neighbor vehicle are dependent on the distance between these two vehicles. The artificial potentials should be designed in such a way that the desired group configurations are obtained at a global minimum of the potential function. The movement of the vehicles of the group should be toward minimizing the potential.

4. Virtual structure approach [10]: consider the group of vehicles as forming a rigid structure, and move the group along a trajectory with control laws that minimize the deviation of each vehicle from the desired position in the virtual structure.

An important point to consider for formation control is the method and systems that will be used for each member to determine its position relative to other members. Frequently this is approached with the assumption that each platform can ascertain and communicate its position to the other team members, allowing each member to independently determine its relative position to all others for use in a control algorithm. However, determining an individual's position relative to an external datum can be a challenge and presents disadvantages. Approaches include:

- External positioning infrastructures (GPS, acoustic transponders)
    - Small and inexpensive onboard systems but expensive infrastructure and vulnerable to interference and jamming
    - limited coverage, typically line-of-sight and ineffective in occluded areas
    - resolutions and update rates may not be sufficient for close-in maneuvering
- Self-contained inertial systems
    - Expensive
    - Position drift rates on the order of 1% without external aiding

Relative positioning on the other hand requires sensors onboard each robot that provide it with the capability to independently determine its position relative to its neighbors instead of with respect to a regional datum. Ideally, the inter-robot communication system will be designed so that relative positioning can be accomplished simultaneously with information passing. This is the form of navigation we see most frequently in nature when more than one entity is involved – flocks of birds, schools of fish, and people walking down a sidewalk. This is also the method of positioning most often used for human piloted vehicles, particularly for close maneuvering – bicycles, automobiles, ships and combat aircraft formations. We also see from these same examples that the leader-follower paradigm is most commonly used and that approach is adopted in this research.

Compared with existing works about formation control, this work has some distinctive features: it makes use of a machine learning technique to learn the control laws to move into (acquire) formation, and to keep (follow) formation. In a work [11] that uses many of techniques described in this paper, a genetic algorithm is used to evolve neural network controllers for simulated "prey" creatures to learn a herding behavior protecting

against predators. However, that work does not address the issue of forming a particular geometric shape (line, tree, etc.).
Another key difference is the inter-vehicle communication/positioning scheme. The group at the University of Reading [12] has concentrated on both non-adaptive and adaptive flocking behavior with mobile robots using logic based on that of Reynolds's boids. Their robots are using ultrasonic sonars for obstacle avoidance and frequency multiplexed infrared light for inter-robot communications. GPS, lasers and cameras are also commonly used for inter-robotic positioning. In our scheme we navigate relative to an assigned leader using a frequency multiplexed chirping scheme and acoustic sensors. The work in this paper utilizes acoustics as a lower power and computationally efficient (compared to vision systems) sensor approach for mobile robots. There are many examples in nature where sound is used very effectively to localize objects of interest. However, acoustics is the only viable option for underwater applications. That being said, sound propagation can be very complex in all but benign environments, requiring a controller that can adapt and learn with changes in the environment. The development of a controller suitable for this purpose is discussed in the next chapter.

3 Control Methods

The algorithms described in this paper are part of a progression of algorithms developed with the end goal of controlling a robot or agent in an unstructured environment without a-priori knowledge or requiring that the robot execute multitudes of physical trials. Feed forward neural networks were selected as the control method because they offer flexibility [13] and lend themselves to being machine generated. Initially, to verify that neural networks would suffice as controllers for the target application of formation maneuvering, a simulation using an innovative genetic algorithm (GA) based training technique was created [14]. Next, in transferring the work to a team of mobile robots, a technique based on Braitenburg vehicles was first developed in order to provide a baseline for testing of the various robot, sensor, and lab systems. While this system worked well, it was purely reactionary, so a more complex behavior based system using three behavior modules was developed. These systems were functional, but were hand coded and did not lend themselves to being produced by a machine, thus limiting their applicability in changing environments. In order to fulfill the stated design goals, the next algorithms in the progression produced a neural network, which in turn controlled the robot. In order to avoid requiring the robot to make a multitude of trial executions, a human-in-the-loop training algorithm was developed. The human acted as a supervisor that helped the robot associate the correct action with given sensor readings. The final algorithm discussed replaces the human with a reinforcement learning technique that uses a supervisory program to find sensor/action pairs that move the robot towards its programmed goal. The algorithm introduces the use of a recent memory created by an environmental exploration routine. This technique removes the human from the loop and uses the recent memory to limit the physical actions required by the robot in the learning process.

The control algorithm for the formation maneuvering task must have the ability to direct a follower robot towards its leader's sound source using the sound intensity values

collected by the follower robot's left and right microphones. It must rely on the amplitude difference between the two values in order to guide the robot towards the sound source on the lead robot. All control schemes presented use the same basic approach, maximization of sound intensity at the microphones, equalization of the intensity between the two microphones, and limiting the maximum intensity. This approach does not directly measure and control inter-robot distance; its consequence is for the robots to try and get directly behind and close on its leader until a defined intensity value is reached. Inter-robot distance is thus indirectly determined by the defined intensity.

This strategy assumes that the microphone sensing the most intense sound is closest to the sound source. This assumption is generally true, but there are several cases in which sound reflections can cause this assumption to break down, causing the robot to make directional errors. In later experiments, this problem was greatly alleviated by installing a baffle system, described in section YY, around microphone.

3.1 Neural Network Proof of Concept Simulation

Initially, there was a need to show that feed forward neural networks would suffice as controllers in formation maneuvering tasks. In order to realize this goal, a multi-robot simulation system was developed to test the controllers in formation maneuvering tasks. The simulator used a GA to find the weights and transfer functions for a feed forward neural network controller that guided a follower robot towards a lead robot. In the training sessions, robots in the population that followed a simulator controlled lead robot were promoted to the next generation. After the specified number of generations the controller of the best follower robot was used to control all robots in various formation maneuvering tasks. Line formations of up to 20 robots and binary tree formations of 8 robots were successfully maneuvered in the simulator. More detail on this aspect of the work is provided in [14]. The algorithms discussed in this section build upon this work and use the GA technique developed for this simulator. The next paragraphs provide more detail on this technique.

The GA was selected over more traditional methods such as Back Propagation so that the need for explicit desired values for each training example could be avoided. In the current discussion of formation maneuvering, this data could be acquired from the simulator, but for other applications the correct motions at each moment are often not known. In these instances a GA can use a fitness function that promotes overall system performance by rewarding controllers that best fulfill a programmed goal.

Because it is not always easy to determine which transfer function would work best for a given problem, in this research the genetic algorithm also selects the type of transfer function that will be applied at each node of the hidden layer. It can select from linear (no transfer), discrete (if the input is greater than 0 then output is 1, otherwise it is –1), and sigmoid ($y = 1/(1+\exp(-v))$). So in this implementation, transfer functions in the hidden layer can vary within a network. While including network transfer functions as part of the optimization algorithm is not widely covered in texts or the literature, there is some indication that it has been experimented with. For example Zhao [40] reported

using a GA to optimize the number and placement of radial basis nodes in their modular neural network. That work is different than this work in that the GA worked with node placement and topology, while the weights were found with more conventional back propagation techniques. In his overview of work on Particle Swarm Optimizations (PSO) Hu [15] indicates that evolutionary computation methodologies have been applied to finding transfer functions in neural networks, but does not list specific instances.

Using the GA to solve for transfer functions proved to be vital to the development of these algorithms. Initial tests using sigmoid functions and linear functions experienced convergence problems. With no other changes, application of Perceptron [16] type discrete transfer functions resulted in a marked, and surprising improvement. This realization that it would be easy to make an incorrect transfer function selection in an otherwise correct arrangement of a neural network motivated use of GA selected transfer functions. Full details on the algorithm used in this work can be found in [17].

3.2 Classic Logic Approach

The robots were first programmed using simple logic that closely modeled that of Braitenberg vehicles [10] as a baseline method to be used for robot, sensor, and system testing. Braitenberg vehicles typically use light sensors to directly control motors on wheeled vehicles. Various wire arrangements allow the vehicles to seek light or avoid light. For this work, sound sources and microphones were used instead of light sources and light sensors. The basics of the classic logic method are as follows:

```
While (running) {
        If (either of microphones reads a very loud intensity) then
                Velocity = Stop;
        Else
                Velocity = Go;
        If  (microphones read intensities that are close to equal) then
                Direction = straight;
        Else
                If (right microphone value is greater than left) then
                        Direction = right;
                Else
                        Direction = left.
} /* End while. */
```

The first if statement keeps a robot from colliding with its leader. The second statement creates an “on center zone” so that the robot will go straight. This statement is not as important in the simulator because reflections, bounces and noise are not modeled, but without it in the lab the robots will always turn left or right, creating a serpentine path. The final if statement selects the direction to turn, given that the microphone intensity was sufficiently different enough to not fall in the on center zone. Note that this is a strictly reactive control scheme in that it only reacts to the current sensor values. One implication of this control method is that the robot will not be able to discern if it is

heading directly towards or away from the source. The next section describes a method used to solve this problem.

3.3 Behavior Approach

The Behavior approach was intended to solve some of the problems seen with the strictly reactive approach by considering the current and past sensor values. Examples include allowing the robot to determine if it is getting closer to the source over time, or scanning to determine the most likely direction of the leader when it is at farther ranges.

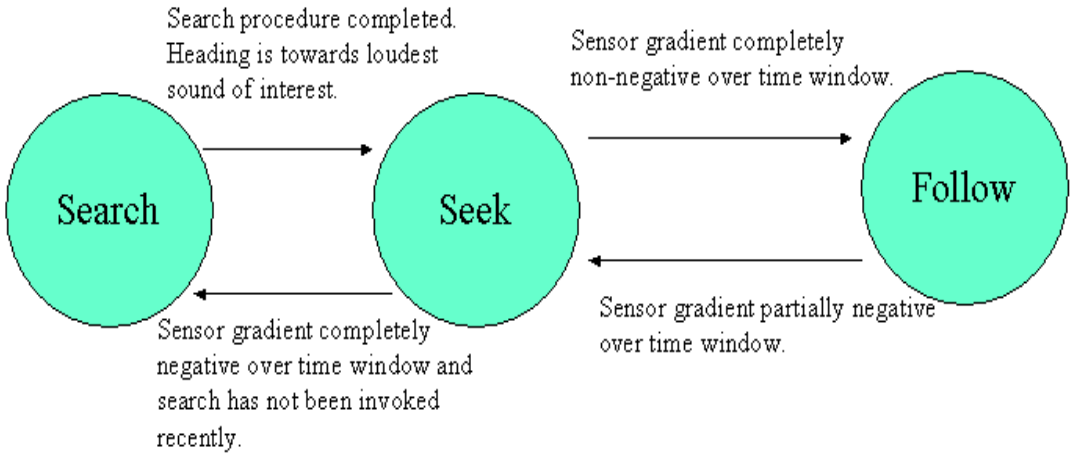

Figure 1. The 3 behaviors and how control is transferred between them. Note that the sensor gradient over time regulates all the transitions except that of Search to Seek. After the Search function is complete, control automatically transfers to Seek. A timer keeps the control from returning to Search for a tunable time period.

As shown in Fig. 1, three behaviors are used to find and follow the sound source. They are follow, seek, and search. The follow mode is for maintaining a desired distance behind a lead robot, the seek mode is used when the robot knows the direction to the lead robot but is too far away for the follow mode, and the search mode is used to determine the direction of the lead robot. The selection of the behavior is based on the gradient history of microphone intensities. Much like subsumption, only one behavior is active at a time, with the difference being that the search behavior requires a waiting period before it becomes active again. This helps alleviate oscillating between behaviors because of problems caused by fluctuating sound intensities due to reflections.

When directly behind its leader, a robot is normally in follow mode, which uses the same basic logic as the classic logic module. As the distance from the leader increases it is difficult for the robot to discern the direction to the leader because the difference in power between the two microphones lessens due to spreading losses [18]. In order to give the robot more freedom to maneuver the seek mode is entered. It uses essentially the same algorithm as the follow mode, but facilitates maneuvering by lessening the *"close"* parameter. That is, in follow mode the left and right microphone values can be different by up about 30 percent and the robot will still go straight, but in seek mode, they can typically only be different by about 10 percent.

If the robot realizes that it has lost track of the source by detecting a negative sensor gradient, search acts to reorient it towards the peak level of the sound source. This is accomplished by rotating the robot and taking advantage of the directivity of the microphones to localize the source. This procedure is similar to that detailed by other researchers [19] when determining if the sound source is in front of the robot or behind, except that the angle to the source is never directly calculated. Here the robot rotates until the source is in front of it, as opposed to making a rotation, solving for the angle to the source, and then continuing in that direction. Once oriented in the general direction of the source, the robot returns to seek mode, and is prevented from entering search for a specified period of time.

3.4 Onboard Neural Network

While the previously discussed methods worked well, they are very application specific and do not have the ability to adapt themselves to changing sensor or environmental conditions. In order to remedy this problem a robot control architecture that generates neural networks on the fly for specific behaviors and environmental conditions was envisioned [14]. A key part of the system is the module that generates the neural networks. Two of the first methods for programmatically generating the networks and their training sets are discussed in this section.

The neural network controller, shown in Fig. 2, is based on a feed forward neural network with one hidden layer trained by a version of GA described earlier. In the simulation the network's output was a speed and heading adjustment, but since the robot is a physical system, it was decided that for the initial tests it would be best to let the operator retain control of the speed adjustments. Along the same lines, the networks used on the robots do not control the amount that it turns. They only indicate the direction of the turn, i.e. left, right, or no turn (straight). The degrees per turn is an operator adjustable parameter. For larger turns the robot repeats the turn command until it is oriented in the manner it desires.

3.4.1 Human in the Loop Training

In transferring the neural network control system to the mobile robots several issues were addressed. In the previously mentioned work, the neural network was trained by repeatedly letting a simulated robot controlled by the network follow a computer controlled simulated robot that made random course changes. The fitness function in the GA optimized the distance between the leader and follower. After several generations, a controller was "grown" that could keep a follower robot consistently close to a leader robot. While this approach is ideal for a simulator, physically executing the generations of runs to develop a good controller on the lab robots was not an option due to time and wear and tear on the systems.

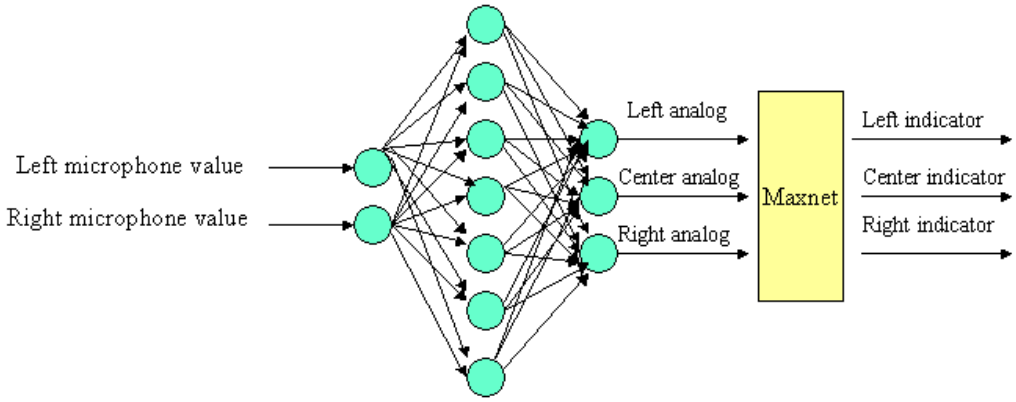

Fig. 2. This figure shows a typical neural net used to control the robots. The inputs consist of the conditioned microphone values coming from the listening program. There is one hidden layer and the output layer has 3 nodes, one for each decision that can be made. Since the robot commands, left, right and forward (center) can only be done one at a time, a winner take all scheme (maxnet) is used to condition the outputs.

To avoid running physical tests, the robot was trained to estimate where a sound source was in relation to the direction the robot was currently facing. The robot was allowed only three possible choices: the sound is either to the left of the robot, in front of it, or to the right. The training set for the neural network was developed by collecting microphone data for each of these three sectors. This was accomplished by a human in the loop telling the training program what sector the source is in and then by moving the source to various locations in that sector during the data collection process; this process was repeated for each sector. The GA fitness function was designed to maximize the correct matches between the output of the network and the training data. Once the robot could guess the direction of the sound source, that information is then used to guide the robot towards the source.

The FFNN used to control the robots is essentially a neural network version of the classic logic system, with the exception that the training process determines the size of the on-center zone instead of the operator (the *"close"* parameter). This approach allows straightforward compensation for variations in robot characteristics such as voltage levels, leader frequency, microphone response, and amplifier characteristics, etc. This approach of growing a custom network for each robot helps alleviate the need to calibrate each system. However, if the conditions for which the training occurred change markedly, the resulting network is not suitable. In this case a new network would need to be grown. This technique was effective but required a human in the loop. The next section describes a memory based reinforcement learning technique designed to remove the human from the training loop.

3.4.2 Reactive Learning using Recent Memory

In this section we introduce the concept of learning using recent memories. A memory is defined here as a collection of sensor measurements and the corresponding control action taken based upon those measurements. If the action taken brings the robot closer to its goal (also defined in terms of sensor measurements) then this memory is defined as a

positive example. For this initial work a recent memory consists of single sensor/action pairs, i.e. left and right microphone intensity values and an action of left, right or no turn. To create a training set from these memories, the system determines which actions have a direct affect on the pursuit of goals. Direct means that an action taken at time step T (0) influences sensor readings in time step T (1) in a measurable way. The recent memory is searched for these occurrences. If the action had a positive effect it is put into a positive example set, and vice versa. Using these sets, a GA is used to train a neural network to estimate the correct actions based on the sensor values. Fig. 3 below outlines the reactive learning process.

In the diagram shown in Fig. 3, the recent memory is sent to the intensity filter. The intensity filter finds all occurrences in which the sensor readings at time T (1) strongly indicate that a correct action was taken at time T (0). For the case of robots learning to follow, the intensity of the left and right sensor readings at T (1) would be higher than those at T (0). The amount of improvement in the sensor strengths is currently an operator settable parameter, usually set to 20 to 30 percent improvement between T(0) and T(1). All occurrences of the just described situation are put into the positive example set, and examples in which the opposite occurred, i.e. sensor readings at T (1) that are substantially weaker than those at T (0) are put into the negative example set. Neutral examples are not used because the sensor readings at time T (1) do not indicate whether correct actions or incorrect actions were taken at time T (0).

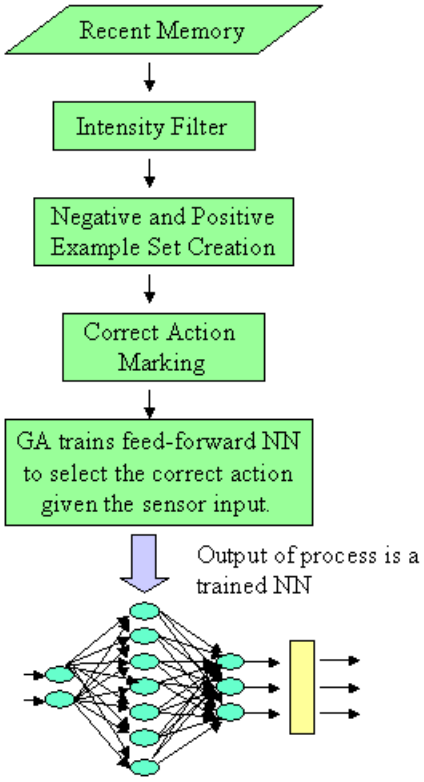

Figure 3. The reactive learning process. Key to the process is the intensity filter. It makes sure that the memory is composed of examples in which actions at time T (0) had a direct influence on sensor values at T (1). As a part of the robot architecture, this process would be called when an observer function notices

goals are not being met. This condition could occur for a variety of reasons, including changing environmental conditions, or fouled sensors.

Once the two sets are created, the actions are marked. For each example in the positive example set, it is assumed that the action taken at T (0) caused the improvement noticed in T (1), so the action at T(0) is marked as correct. In the negative example set, it is assumed that the action taken at T (0) caused the deficit in the goal parameter noticed at T (1), so the action taken at T (0) is marked as incorrect. Since the correct action is not known for the negative examples, the training file is set up to request the GA to reward the selection of an alternate action from the remaining actions available.

Once all the actions have been marked, the training file is written. The GA uses the training file to train a FFNN in a similar manner as to that discussed in the previous chapter. The inputs to the network are the sensor readings. The output is the action to be taken. The fitness function rewards networks that choose the action associated with those sensor values. The GA creates each successive generation of networks from the successful networks of the current generation. The stopping criteria for the GA has been implemented in two ways. In the first, the GA executes a fixed number of generations. Since the robot's onboard computers do not have as much processing power as the development computers, a provision to stop the GA when there has been no improvement for a specified number of generations was implemented. The best scoring neural network of the last generation is used as the reactive controller.

The steps below outline the learning from a recent memory process for a robot whose goal is to follow its leader in a formation maneuvering task. Note that only the positive example set is used in this outline.

1. Let a follower robot record a memory consisting sensor/action pairs while seeking to either a fixed or moving sound source. The robot can be guided by an operator, or by some other algorithm, such as the classic logic method, another neural network, or the random but purposeful controller.
    a. A sensor action pair SA consists of a left microphone value, L, a right microphone value, R, and an action A.
    SA = {L, R, A}
    The L value of a sensor action pair SA is indicated by SA→L
    The R value of a sensor action pair SA is indicated by SA→R
    The A value of a sensor action pair SA is indicated by SA→A
    b. A recent memory M is defined as a sequence starting at time 0 and completing at time n-1 of n sensor/action pairs.
    M = SA [0] …..SA [n-1]
2. Create training sets of positive examples, P, and negative examples, N, from M using the following logic:
    /* Initialize P and N to the empty set. */
    P = N = ∅ ;
    For j = 0 until j = n -2
        /* total sound energy = left mic value + right mic value */
        /* E0 = total sound energy at time 0 */

```
E0 = SA[j]→R + SA[j]→L;

/* E1 = total sound energy at time 1 */
E1 = SA[j+1]→R + SA[j+1]→L;

If (E(1) >E(0)) then
        /* Sensor/action pair at time 0 is put in positive example set. */
        P = SA[j] ∪ P;
Else
        /* Sensor/action pair at time 0 is put in negative example set. */
        N = SA[j]  ∪ N;
End for
```

3. Using the set of positive examples as the new training set, let the GA based technique generate a new neural network. In order to increase efficiency of the GA, the current best neural network can be used to seed the initial population.

Step 2 implements the local optimization that is key to reactive learning. As stated earlier, for this step to be valid, the nature of the signal that the sensors are sampling has to be such that the results of an action taken at a particular time step can be sensed during the next time step. The intensity filter (Fig. 3) helps to eliminate cases where the effects of low amplitude and/or noise hinder action/sensor coupling.

As previously stated, in some cases the negative example set can also be used. To do this, the fitness function in the GA is programmed to encourage selection of an alternate action. The reasoning is that if a given action does not yield positive results for the next time period, one of the other choices would have been better. A training set that is sufficiently representative of the conditions that will be encountered will have enough examples to resolve any ambiguities.

Table 1. Data collected from a typical memory. Note that the action taken at T(1) seems correct, but would be put into the set of negative examples.

Time T(n)	Left Sensor Value	Right Sensor Value	Total energy	Action Taken (left, straight, right)	Example Set
0	10	0	10	Left	positive
1	5	6	11	Straight	negative
2	5	5	10	Straight	positive
3	7	7	14	Straight	

That being said however, for these techniques to work, choices made at time T (0) need to be accurately reflected in the sensor values by time T (1) more often than not. If this is not true, the training set will be too ambiguous to encourage correct decisions. Looking

at Table 1 above, it can be seen that the total energy at T (0) is 10 (left sensor value + right sensor value). At T (1) the total energy is 1. According to rule 2 from the learning from a memory process, T (0) would be put into the positive example set. Using this same rule T (1) would be labeled a as a negative example, although it is clear from the data at T (2) and T (3) that straight is the correct choice at time T(1). Using negative examples, the GA would erroneously encourage an alternate choice, left or right. With a limited number of occurrences of these erroneous examples, the GA can form a working neural net, but as their numbers increase, the data set becomes more ambiguous, causing the resulting network to be less effective.

This algorithm was used to create controllers that were tested both in simulation and using the mobile robots. Memories collected in the simulator worked well when using both the positive and negative example sets, but memories collected using the mobile robots worked only when using the positive example set. The unpredictable nature of sound in the lab populated the training set with negative examples similar to the one illustrated in Table 1, rendering the learning process ineffective when they were used.

Ah-Hwee Tan et al. [20] compared a reactive plan execution system to a reactive learning algorithm in a simulated mine avoidance application. In this system the autonomous agent used a small array of sonars to detect the mines as it navigated towards its goal location. The plan execution system was a rule based system relying on a-priori knowledge while the learning system used a reinforcement based algorithm. Both systems made heavy use of data quantization in order to reduce the state space. To train the learning system, 1000 trial runs consisting of 30 sense-act-learn cycles were used. Using this method the learning system returned performance similar to that of the plan execution system. The reactive learning algorithm discussed in this section differs in that it avoids the large number of trial runs during the learning process. While this is not as important when running in a simulation, it has implications with physical robots because it saves time, battery power, etc.

The next section provides details of the robot's hardware and sound system. It also discusses the software used to generate the communication signals and process them.

4 Robots and Sensors

Formation following was demonstrated using land robots equipped with audio transmission and '2 ear' listening systems. The systems operate in a semi-passive manner meaning that the robots do not exchange position or bearing and range information. Instead, each robot listens for a chirp emitted by its leader and steers itself towards it by turning in the direction of the strongest signal (left or right). A frequency multiplexed communication scheme is used where each leader transmits in its own pre-specified frequency band, and followers are 'assigned' to a leader by listening in the specified band. For this work the signaling is used by followers only to determine a relative direction to steer in order to follow the leader, but the same signaling scheme could readily be expanded to include communication of specific information such as the leaders speed.

4.1 Hardware

ActiveMedia Pioneer 2DX robots were equipped with acoustic transmit and receive systems shown on the robot in Fig. 4; a line diagram of this system is shown in Fig. 5. The sound systems were constructed from commonly available and inexpensive components. The receive system consisted of two omni-directional microphones mounted in a stand that allows various microphone angles and separation distances, battery powered amplifiers and a standard 12bit PC sound card. Typically the microphones are placed at 30 - 45 degrees angles away from the center of the robot, which yields about 18 to 24 inches of separation. The smaller computer electret type microphones were tested and offered higher frequency response up to 20 kHz. In testing we found these were not suitable for outdoor use due to their sensitivity to wind and vibration, so we chose the larger dynamic microphones, commonly used for stage performances, with a frequency response up to 12 kHz.

For this application a sound card with a line input option was required for two input channels since the standard microphone input is mono. The line input requires a larger voltage than the microphone input so additional amplification is required between the microphones and the sound card. The initial design used two battery powered mono guitar distortion amplifiers which were later replaced with a single smaller battery powered stereo amplifier providing up to 50dB of gain.

The transmit system is composed of the sound card (one channel line out) and a battery powered omni-directional speaker. The battery powered speaker provides greater signal amplitude and thus greater range for outdoors; this could also be achieved by using a soundcard with higher power, but these typically require an additional 12volt power source. Sound studies showed that the majority of the ambient noise was 3 kHz and below and determined to be due primarily to wind, motors, fans, etc.

Hardware filtering before sampling by the sound card could be used to reduce the impact of this noise on dynamic range, but would not help with impulsive sources of noise such as doors closing, footsteps, bumpy paths, etc. Outdoor acoustic testing showed effective ranges of 30+ feet without filtering. Custom designing narrowband hardware filters could increase the effective range by rejecting out of band energy to use the available dynamic range more effectively, but this was not deemed necessary for these experiments.

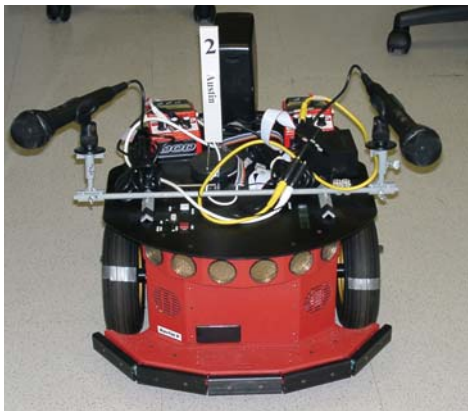

Fig. 4. Pioneer 2DX robot with microphones and amplifiers mounted. Notice how the microphones are mounted on each side of the robot, like ears.

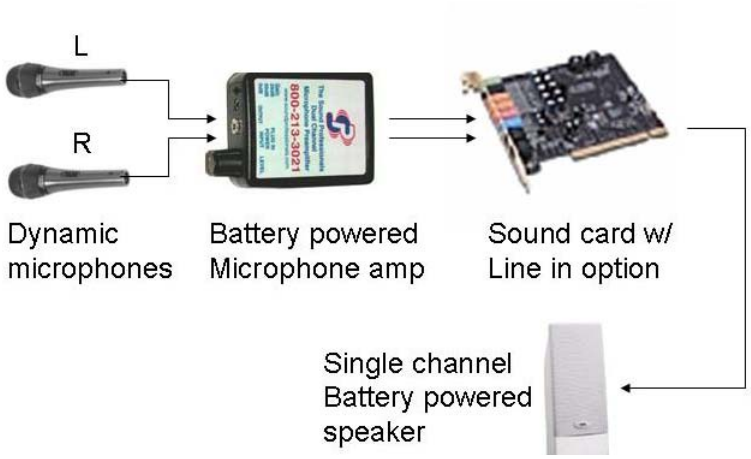

Fig. 5. Robot transmit and listening hardware

4.2 Acoustic environment considerations

As mentioned previously, the basic operational premise for the acoustic sensor system is that the direction of the leader is indicated by the microphone with the largest signal. In theory this is straightforward as the amplitude of an acoustic signal falls off with the square of the distance [21] from the source as shown in Fig. 6. In practice this was found to be considerably more complicated as was also seen in related works using acoustic guidance [19, 22-24]. At very close ranges (less than 10ft) there is sufficient spacing between the microphones at the frequencies used to detect a change in amplitude due to attenuation alone. However, and as seen in figure 6, at larger ranges the difference in amplitude due only to the distance between the microphones quickly becomes too small to be detectable with the available 12-bit dynamic range. Coupled with the fact that the microphones are omni-directional at the frequencies being used, initial testing yielded effective results for ranges of only a few feet. The later addition of baffles to the microphones allowed the system to take advantage of directivity between the left and right microphone, an inherent design in most biological listening systems, and dramatically improved the effective range of the system.

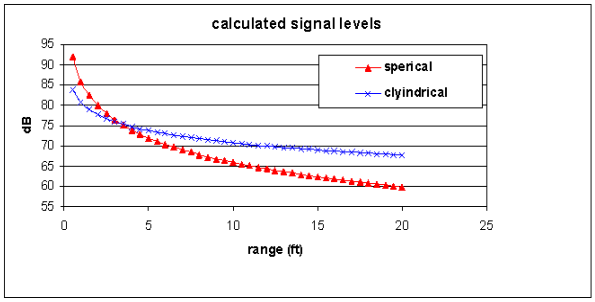

Figure 6. This figure shows the calculated [18] signal loss due to the spreading of acoustic energy. Note that these calculations are medium independent, and are applicable to both water and air. In general, spherical spreading is used in air, where there is not a top and bottom boundary constraining the propagation of the sound. Cylindrical spreading is communally used in the water, where the seafloor and the sea surface provide a bounding influence on the sound propagation.

Working indoors in a quiet lab environment presented interesting challenges as well. Initial tests used constant frequency pulses and were almost completely ineffective. Testing with a sound pressure level meter revealed that this type of signal created standing waves in the lab, creating areas of high and low intensity completely uncorrelated with the range from the source. This was overcome by using a 'chirp' signal, i.e. a signal that sweeps from one frequency to another. By sweeping rapidly between frequencies standing waves are not created and chirps proved much more effective in the lab.

Another complicating factor observed was the time variability of the signal strength received. Even with stationary equipment significant variations in sound intensity were often observed both indoors and out. Coupled with contamination by intermittent outside noise sources, determination of relative direction using this system is not dependable when looking only at individual samples. Time averaging of the signal in the desired frequency band proved an effective measure to combat these issues.

4.3 Signal Processing Algorithms

Since the PC on the robot uses a relatively low powered 400 MHz processor that is tasked with many simultaneous processes, the processing of the sound needs to be accomplished efficiently. Significant improvements in processing speed can be achieved with a DSP board, but with additional development cost, time and higher power consumption.

The chirp signals used for robot testing were 1 second in duration with a 40% duty cycle and an upsweep range of 200 Hz. Signals used by the leaders ranged from 500 to 5000 Hz and a receiver sampling rate of 11025 Hz (standard on most sound cards) ensured aliasing was not an issue for these signals. A very simple yet remarkably robust approach to signal detection using fast Fourier transforms (fft) was used for the majority of the experiments. The receiver processing consisted of capturing ¼ second of data, performing the fft and computing the signal power using only the values in the frequency range of interest. Note that the last operation effectively bandpass filters the captured data, thus rejecting all out of band energy and improving signal detection. A running average is then computed for 4 consecutive data captures, giving a 1 second long sliding

window that ensures a transmitted pulse will be contained in the average. Initial system implementations sent the left and right averaged signal strength separately to the neural network controllers, and improved performance was later achieved by using the ratio of the two channels to provide relative signal strength.

Recent experimentation has yielded more effective and efficient processing. Since the fft algorithm requires NlogN multiplies, performing more and shorter fft's requires less computations; testing with sample sizes as short as 256 (23 msec) and time averaging these has been shown to be very effective and extremely fast. An improved approach to signal detection has also been tested successfully. This approach examines the ratio of the in-band power to the power in an adjacent unused frequency band to determine if a signal exists. Unlike the original method that passes a result to the neural network without detection, this approach only passes data when the in-band signal is significantly larger than the out of band signal. This allows the neural network to 'ignore' periods where no signal is being received or where broadband noise is so loud that it is overpowering the transmitted signal. Work is underway to develop an algorithm that is even more computationally efficient and will provide superior detection by matching to the specified frequency rate of the transmitted chirp.

5 Experimental Results

Tests in both the simulator and in the lab were carried out with each of the algorithms. Attempts to create a portable robot tracking system accurate to a few cm's that would work indoors or outdoors reinforced our assertion that relative positioning is a sound option for obstructed environments; consequently robot tests were evaluated using video. Lab tests consisted of a formation maneuvering task in which the follower robots follow a leader in either a series of laps around the lab or a follower follows a leader making random course changes. The simulator tests consisted of a follower robot tracking its leader for two laps during which inter robot distance and heading differences were recorded. This section begins by discussing issues addressed during initial testing and presents snapshots taken from video of the robots executing a line formation in the lab. Next a series of graphs using data collected in a simulator tracking inter-robot distance and robot heading difference are presented along with discussion.

5.1 Lab Tests using Mobile Robots

Initial testing was done using input data from the listening system that was scaled so that its range was between –1 and 1. Using this method line formations that were able to run laps around the lab were successfully created. The FFNN controllers worked well as long they were within close range (about 1 to 3 feet) of their lead robots, but once out of that range they failed. Changing the inputs from scaled microphone values to relative microphone values largely alleviated this problem and allowed ranges of approximately 5 feet in the lab. The logic below outlines the process.

The variables are defined as follows:

left : the value the sound system returns for the left microphone.
right : the value the sound system returns for the right microphone.

Big : the greater of the left and right microphone values.

The algorithm is given below:

```
/* Get the larger of the two microphone values. */
If (left > right)
        Big = left
Else
        Big = right

/* Normalize the left and right values. */
left = left/Big
right = right/Big
```

This procedure removed the intensity information from the inputs and left only directionality information. When using the non-relative values, training examples from the same orientation, but a different range are very different from each other numerically. Using the relative values, they are closer to the same at the ranges used in these tests. The result was that the limited number of training points provided a better description of what the robot would experience during following maneuvers. This in turn helped to lower the unpredictability of the networks responses. With this change it is possible to create networks that operate smoother than either the classic logic or behavior methods.

Fig. 7 shows a sequence of frames from the start of the formation using neural network controllers to when the formation rounds the first turn of the rectangular grid laid out on the lab floor. Fig. 8 below shows the robots rounding the curve and heading back towards the operator's consoles.

Fig. 7. The lab robots in a line formation maneuvering in a square pattern on the lab floor. The lead robot does not have microphones, it is being controlled by one of the lab's computers. The two following robots are equipped with microphones.

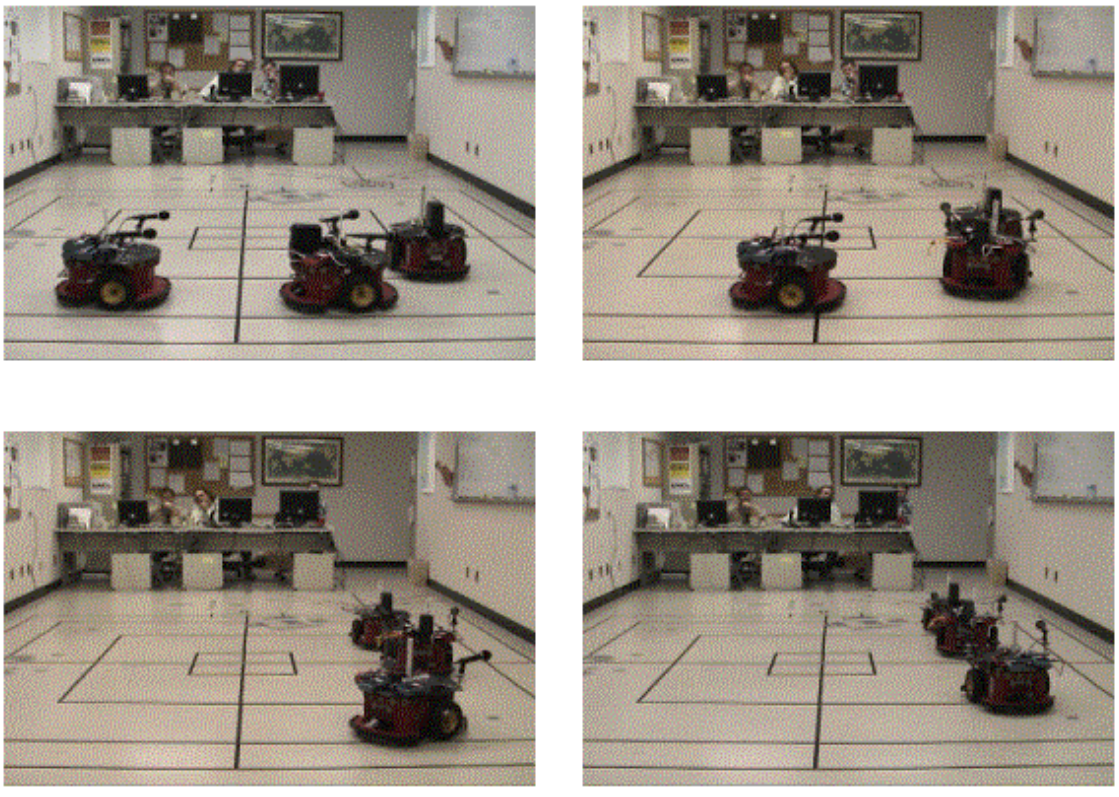

Fig. 8. The lab robots continuing their line formation from Fig. 7.

Up to eight laps were run in formation using both the classic logic approach and neural network controllers. Testing of the line formation with two robots outdoors, a much better acoustic environment, yielded considerably superior results allowing leader-follower separations on the order of 20+ feet.

By incorporating a baffle that dampens the effects of sound reflections the performance of all the algorithms was improved, most markedly that of the behavior based method. The behavior based routine was shown to work well in simulation, but initial results in the lab were less than successful. The sound intensity fluctuations created by the reflective surfaces caused the behavior arbitration method to rapidly oscillate between behaviors and slowed the robot to a crawl. Once the effects of these reflections were dampened with the baffles the algorithm's performance improved dramatically.

Reeve and Webb[23] reported good results in seeking to a stationary sound source in their work investigating cricket phonotaxis. Instead of using FFTs and a neural network to isolate the chirp and control the robot, they used a neural oscillator that acted both as a band pass filter and a directional controller. Using this approach their robots sought to a sound source emitting 20ms chirps in the 4.7kHz sound range. The weights were found by using a closed form solution for their neural model. Since their goal was to emulate the physiological structures and functioning of the cricket, their system is not designed with learning and changing environment in mind. That being said, it does show that neural networks are effective controllers, and does suggest that with a system of tuning the neural oscillators, it may possible to replace the FFT process, resulting in a savings of computational resources.

Kushleyev and Vohra [19] developed a system to localize sound based on a two-sensor system modeled after a human head. They show good results for a simulated robot tracking a moving sound source. As expected, results for experiments without noise added are better than those with noise. Their algorithm is rule based and uses an FFT based process to determine range and distance to the target. Their testing indicates that they too experienced many problems with noise and reflections in their lab. Since their system is rule based it does not lend itself to changing environments. They do not show results with multiple robots in simulation or in the lab.

5.2 Simulator Results

Simulator tests were also conducted. The simulator tests measured the leader to follower robot's distance and the heading difference between the leader and follower robots' courses. In the test the lead robot traversed two laps around a preprogrammed course in the simulator. The tests are started in formation, by maneuvering the leader follower pair to the starting position and then engaging the automatic navigation control for the leader robot. The robot's turning rates and speeds were set the same for all tests. Fig. 9 illustrates the test track in the simulator.

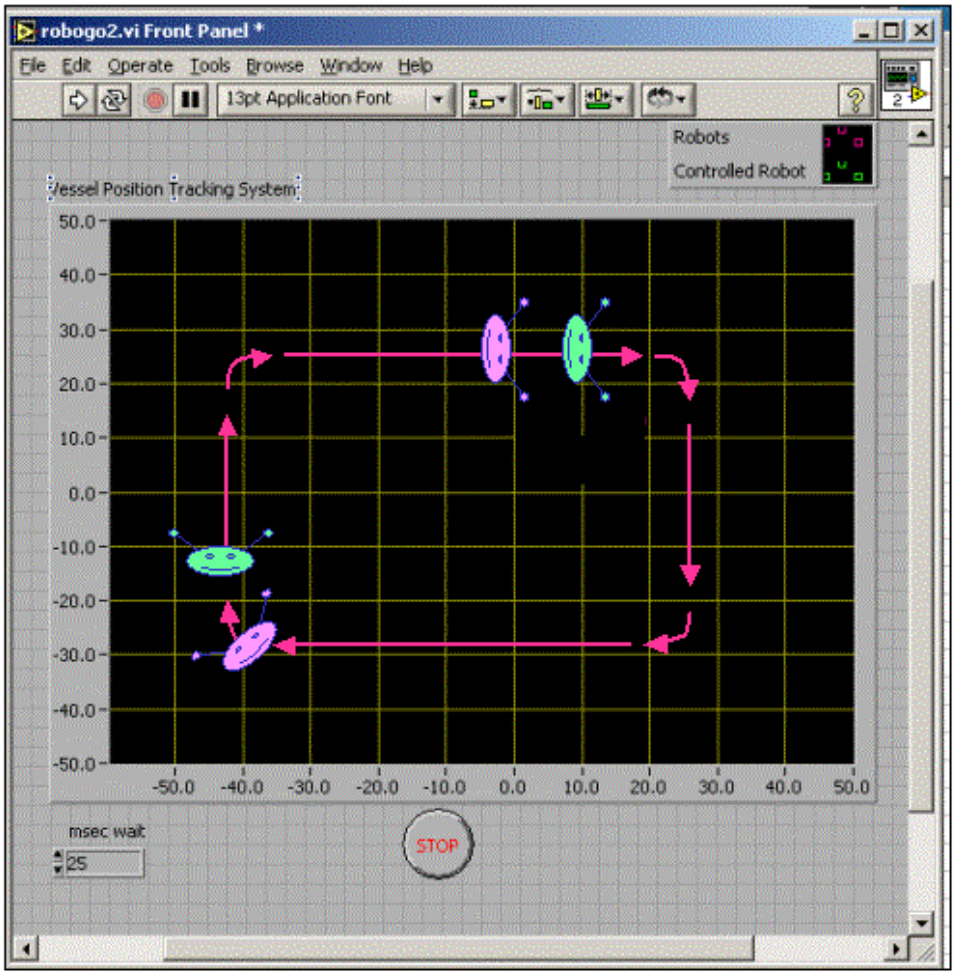

Fig. 9. The test course for the following robots. The lead robot (green smiley face) is controlled by the simulator. The follower robot (pink smiley face) is controlled by the current controller being tested. Over two laps the inter robot distance and course differences are recorded.

The tests are started in the upper left hand corner of the track, when the leader/follower combination is in formation and tracking (see Fig. 9). The test is complete when the combination rounds the upper left corner at the end of the second lap and is heading towards the corner to the upper right.

Figure 10 shows the average distance between the leader and a follower controlled by a typical reactive neural network. The distance between the two remains stabile except during the turns. The turns are the 8 places in the curve where the distance fluctuates. Figure 11 shows the heading differences measured during the same test. The graph has 8 peaks in it, each one corresponding to the leader making a course change. For comparison Figures 12 and 13 show a similar test run using the classic logic controller.

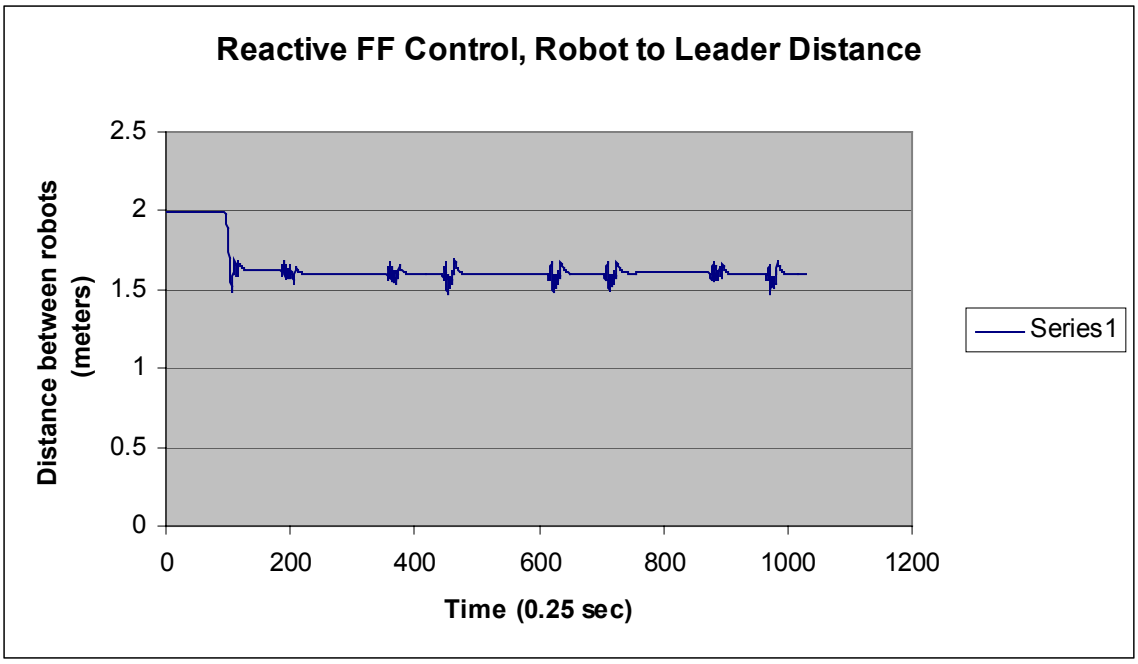

Fig. 10. The leader/follower distance over two laps. The follower robot is controlled by a reactive FFNN. As can be seen in the figure, the robot closed the distance between it and its leader and maintained it throughout the run. The fluctuations in the line are when the leader is making a turn on its preprogrammed course. As can be seen there are 8 turns over two laps.

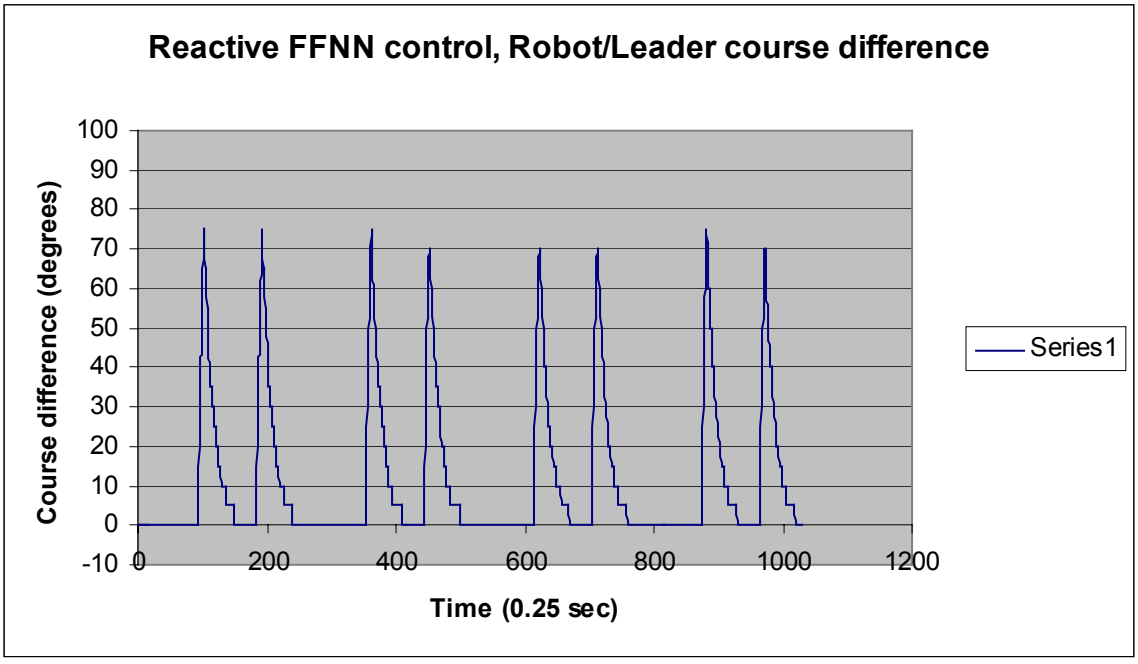

Fig. 11. The heading difference between the leader and the follower robot controlled by a reactive FFNN during a two lap test in the simulator. Notice that each of the 8 peaks (the peak denotes a course change by the leader) occur at about the same time as the fluctuations occur in the curve in figure 10.

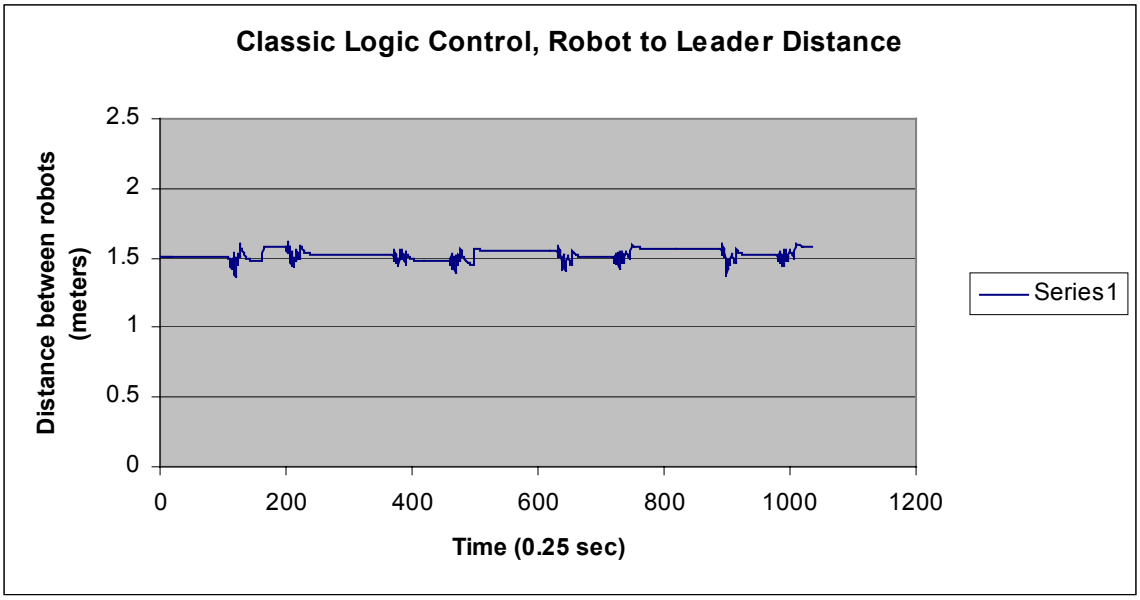

Fig. 12 The leader/follower distance for the two lap test when the follower is using the classic logic controller.

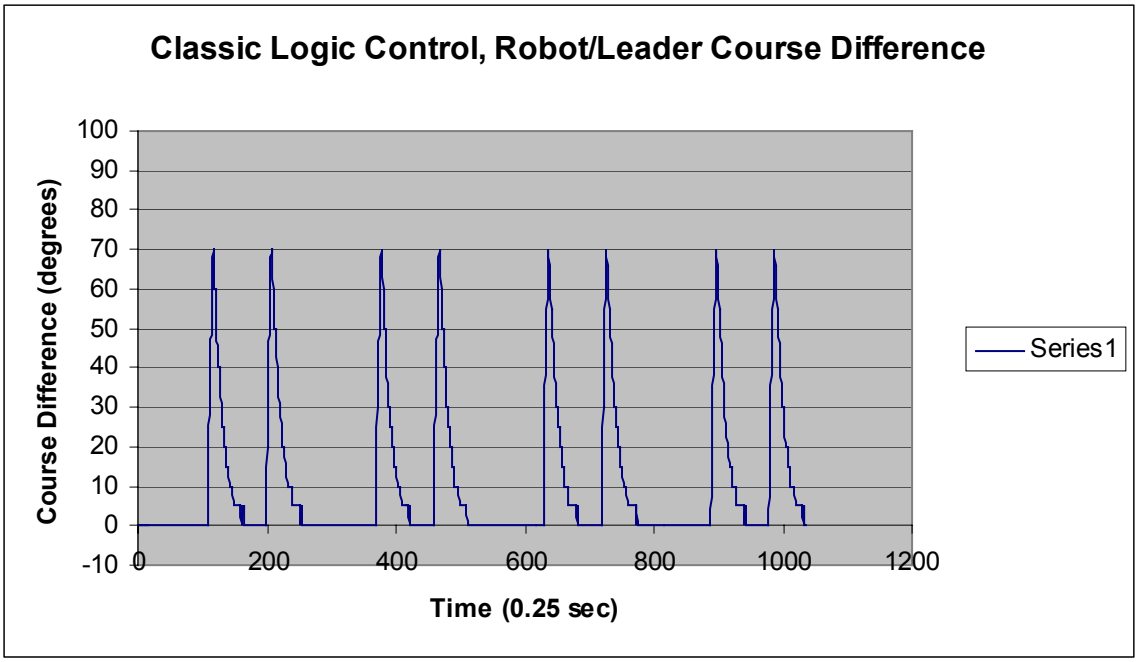

Fig. 13. This figure shows the course difference between the leader and follower when the follower is using the classic logic controller. Each peak corresponds to when the leader is making a turn.

Comparing Figures 10 and 12, it can be seen that the inter-robot distances are very similar, but the curve shapes are quite different. The curve in Fig. 10 is smoother overall, indicating that the distance between the robots varies less with this controller. The reason for this is that the classic logic controller tends to oscillate when it is nearly centered. The straight percentage parameter damps this phenomenon but has the adverse effect that it causes the robot to take more time to match its course with the leaders. These

oscillations can be seen in Fig. 13. These are the places in the curve at the bottom of the peak where the blue line gets thick. Looking at the peaks, the oscillations can be seen at the tail end of the peaks, right when the course difference is approaching zero. A detail of the tail end of the peak is shown in Fig. 14 below.

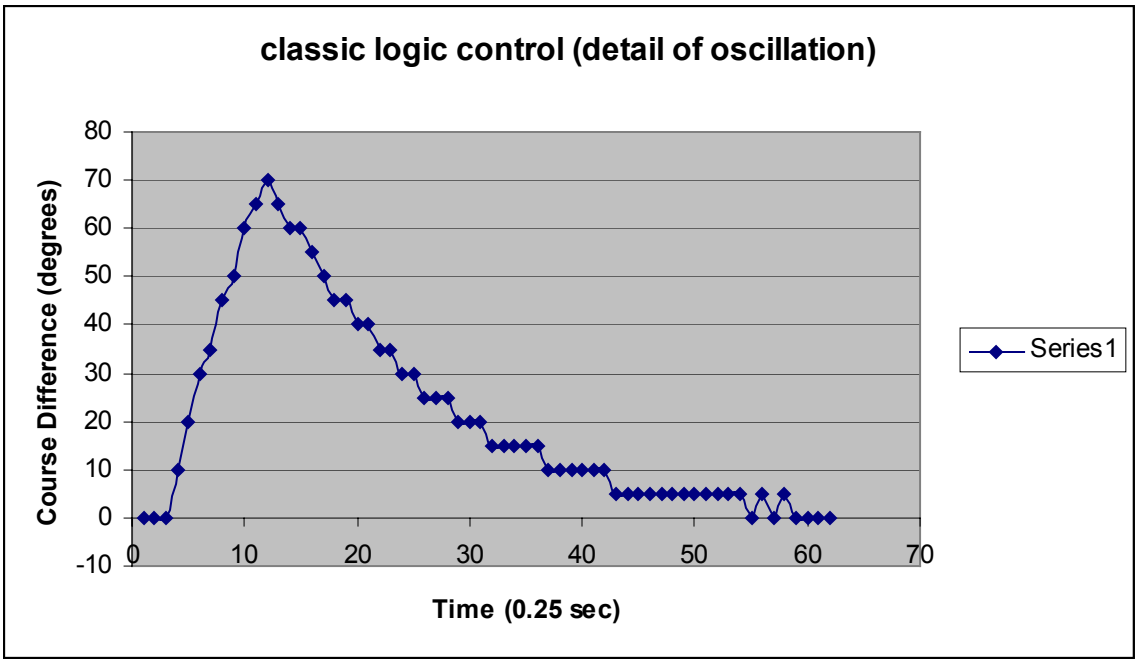

Figure 14. A detail of the first peak from figure X10. Looking at the curve at about time step 55, the oscillations can be seen as the curve moves from a course difference of 0 to 5 degrees and back twice.

Looking at Fig. 11, these oscillations are not present. Not only have these oscillations been observed in these tests, the same phenomenon occurs in the lab using the mobile robots.

Figures 15 and 16 show similar tests run with the behavior controller. Recall that it considers past values in order to switch between seek, search, and follow behaviors. For this test it remained mostly in follow behavior, which is essentially the same as the classic logic controller, but because it did move into seek behavior many oscillations in the plots are seen. The oscillations are caused because in seek behavior the on-center zone is narrower, making the robot more sensitive to amplitude differences.

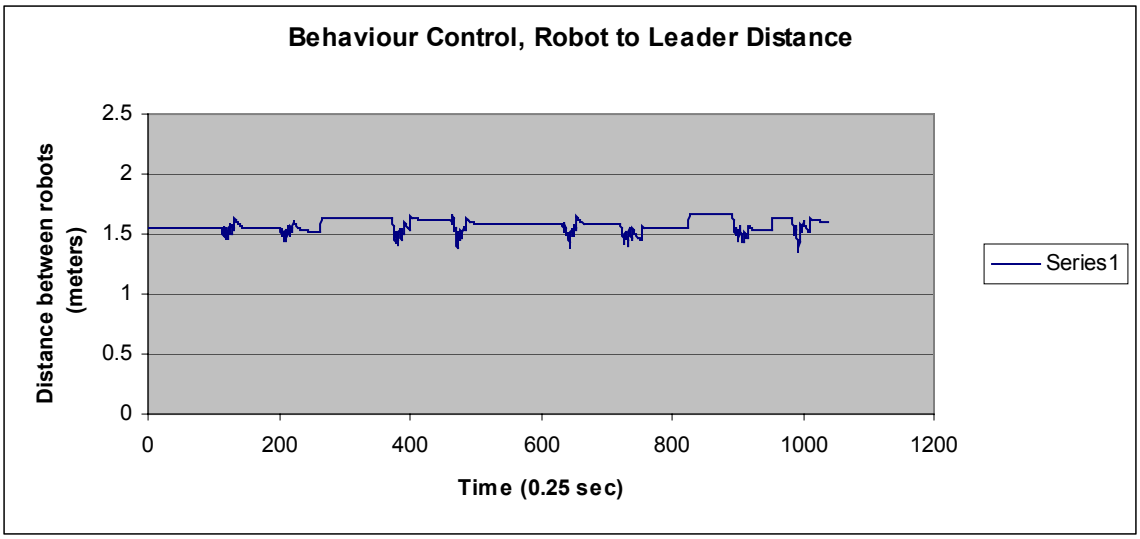

Fig. 15. The leader/follower distance for the two lap test when the follower is using the behavior controller.

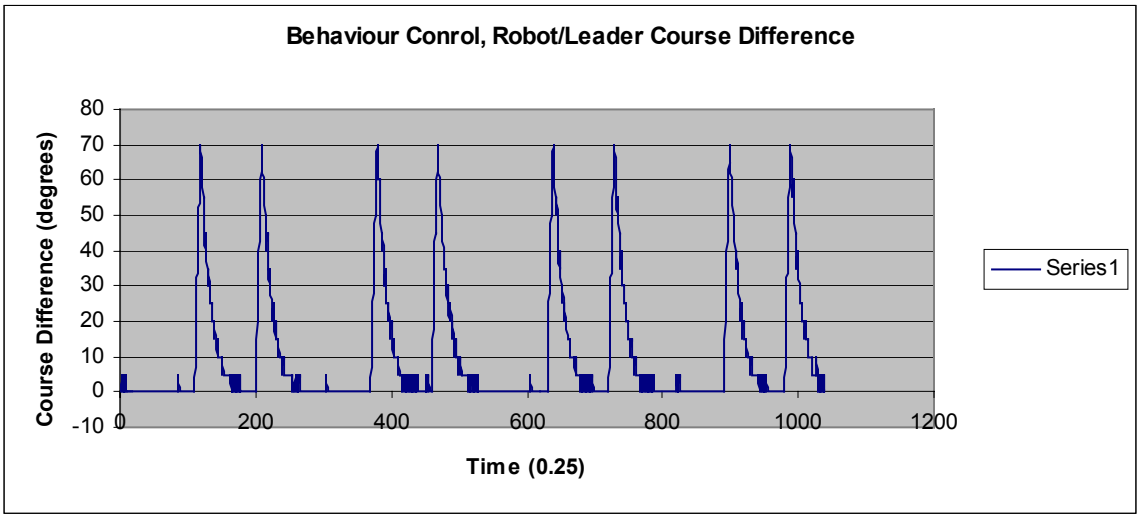

Fig. 16. The course difference between the leader and follower when the follower is using the behavior controller. Each peak corresponds to when the leader is making a turn. Notice that this controller takes more time to complete turns than the reactive controllers. The thick blue lines at the end of each peak denote severe oscillation at the tail end of each turn. This is caused by the robot being in seek mode. In seek mode, the straight percentage parameter is set low, enabling the robot to quickly turn to its source, but at this level, it no longer damps oscillations.

Figures 10 through 16 clearly show that the performance of the FFNN controllers can match or exceed that of the hand programmed controllers. During initial testing, subjective observations indicated that the FFNN were smoother than the behavior or classic logic controllers in both the simulator and onboard the robots. This series of simulator tests backs up those early impressions. The FFNNs avoid oscillations, and can maintain stabile distances from the follower to the leader.

6 Summary

This work has shown, in concept, that formation maneuvering using acoustic sensors is possible in both a simulated environment and in the physical world of the lab. It has also shown that the structure of the formation can remain viable without the use of a centralized controller, or external infrastructure based communication and positioning system. Since the communications between the robots uses very low bandwidth acoustic methods, this work is relevant to underwater applications.

This research has many facets, so the opportunities for improvement are abundant. Aside from equipment related improvements such as better microphones, speakers, and amplifiers efforts need to focus on improved signaling schemes, behavior arbitration, on the fly learning, and inter-robot communications.

The memory-learning scheme currently being used is the first step in the way to a system in which on the fly learning is integrated into the control system, allowing continuous real-time adaptation in an unstructured environment. To date, the goal has been to show that if a network were tuned correctly and could learn, it could be used to control the vehicles in a formation-maneuvering situation using acoustic systems. An appropriate next step will be to seamlessly automate the learning process.

Because acoustic reverberation and multi-path causes problems with amplitude based following systems, one alternative method that has been considered is supplementing the control system using low bandwidth acoustic communications between robots to provide a following robot with the intentions of the leader. This additional information can then be used to assist the following robot in determining the proper actions in the presence of misleading sensor data.

References

1. Ronald C. Arkin, *Behavior Based Robotics*, MIT Press, 1998, pp. 10-14
2. J. Desai, J.P. Ostrowski, V. Kumar, "Controlling Formations of Multiple Mobile Robots," *Proceedings of 1998 IEEE Int. Conf. on Robotics and Automation*, 1998, pp. 2864-2869
3. J. Desai, V. Kumar, J.P. Ostrowski, "Control of Changes in Formation For a Team of Mobile Robots," *Proceedings of 1999 IEEE Int. Conf. on Robotics and Automation*, 1999, pp. 1556-1561
4. J. Desai, J.P. Ostrowski, V. Kumar, "Modeling and Control of Formations of Nonholonomic Mobile Robots," *IEEE Trans. on Robotics and Automation*, vol. 17, no. 6, December 2001
5. R. Fierro, A. Das, V. Kumar, and J. P. Ostrowski, "Hybrid Control of Formations of Robots," *IEEE Int. Conf. on Robotics and Automation*, Seoul, Korea, May 2001, pp. 157-162
6. T. Balch, R. Arkin , "Behavior-Based Formation Control for Multirobot Teams," *IEEE Trans. on Robotics and Automation*, vol. 14, no. 6, December 1998
7. N.E. Leonard, E. Fiorelli, "Virtual Leaders, Artificial Potentials and Coordinated Control of Groups," Proceedings of the 40th IEEE Conf. on Decision and Control, December 2001, pp. 2968-2973
8. T.R. Smith, H. Hanbmann, N.E. Leonard, "Orientation Control of Multiple Underwater Vehicles With Symmetry-Breaking Potentials," *Proceedings of the 40th IEEE Conf. on Decision and Control*, December. 2001, pp. 4598-4603
9. H. Yamaguchi, T. Arai, "Distributed and Autonomous Control Method for Generating Shape of Multiple Robot Group," *Proceedings of IEEE Int. Conf. on Intelligent Robots and Systems*, 1994, pp. 800-807

10. M.A. Lewis, K-H. Tan (1997). "High Precision Formation Control of Mobile Robots Using Virtual Structures," *Autonomous Robots*, volume 4, Springer 1997, pp. 387-403
11. G. Werner, M. Dyer (1992). "Evolution of Herding Behavior in Artificial Animals", Proceedings of Int. Conf. on Simulation of Adaptive Behaviors, 1992.
12. University of Reading, "Flocking Robots", http://www.cyber.rdg.ac.uk/CIRG/robots/flocking.htm
13. Tom Mitchell, *Machine Learning*, McGraw-Hill, 1997, page 105
14. P. McDowell, J. Chen, B. Bourgeois, "UUV Teams, Control From A Biological Perspective", *Proceedings of the Oceans 2002 MTS/IEEE Conference*, Biloxi MS, pp 331-337
15. Xiaohui Hu, "PSO Tutorial", http://www.swarmintelligence.org/tutorials.php
16. Tom Mitchell, *Machine Learning*, McGraw-Hill, 1997, page 86
17. P. McDowell, "Biologically Inspired Learning System", Louisiana State University, Baton Rouge, La, December 2005, page 41
18. Marshall Bradley, *Environmental Acoustics Handbook 2nd Edition*, 1996, page 23
19. Aleksandr Kushleyev, Vaibhav Vohra, "Sound Localization," University of Maryland, 2004
20. Ah-Hwee Tan, Samin Karim, Liz Sonenberg, "Reactive Plan Execution versus Reactive Learning: A Case Study on Minefield Navigaiton", *IAT*, France, 2005
21. Robert J. Urick, *Principles of Underwater Sound, 3rd Edition*, McGraw-Hill, 1983, page 100
22. Vinay Shah, "Practical Implementation of the Sound Localization Algorithm on a Robot," Rensselaer Polytechnic Institute, Research Experience for Undergraduates, August, 2003
23. Reeve, R. and Webb, B., "New neural circuits for robot phonotaxis," *Philosophical Transcations of the Royal Society*, 2003, pp 2245-2266
24. M. Rucci, J. Wray, G.M. Edelman, "Robust localization of auditory and visual targets in a robotic barn owl," *Robots and Autonomous System*, vol 30, 2000, pp 181-193

Author Biographies

Patrick McDowell received his bachelor's degree in Computer Science in 1984 from the University of Idaho. He spent the next 15 years working as a computer scientist for a small defense contractor where he specialized in real time data acquisition, application development, and image processing. In 1999 he received his master's degree in computer science from the University of Southern Mississippi. In 2000 he began work at the Naval Research Lab where he has focused on application of machine learning techniques to autonomous underwater navigation. In 2005 he received his Ph.D. in Computer Science from Louisiana State University. His research interests include legged robotics, machine learning, and artificial intelligence.

Brian S. Bourgeois received his Ph.D. in Electrical Engineering from Tulane University located in New Orleans, LA in 1991. Since then he has worked at the Stennis Space Center, MS detachment of the Naval Research Laboratory. He has worked on research projects spanning an array of technologies including airborne survey sytems, acoustic backscattering, bathymetry and imaging sonar systems, the ORCA unmanned underwater vehicle and the development of an autonomous survey system for hydrographic survey ships. He is presently the head of the Position, Navigation and Timing team at NRL with research interests including underwater positioning and communications and autonomous navigation.

Ms. McDowell received her M.S. in Applied Physics in 2002 from the University or New Orleans. She is presently a candidate for a Ph. D. in Engineering and Applied Science. She joined the Naval Research Laboratory in 1991 as a research engineer and has spent

most of that time working in experimental and theoretical acoustic modeling. Ms. McDowell's specific research interest lie in the areas of sonar performance analysis.

Dr. S. S. Iyengar is the Chairman and Roy Paul Daniels Chaired Professor of Computer Science at Louisiana State University and is also Satish Dhawan Chaired Professor at Indian Institute of Science. He has been involved with research in high-performance algorithms, data structures, sensor fusion, data mining, and intelligent systems since receiving his Ph.D. degree (1974) and his M.S. from the Indian Institute of Science (1970). He has been a consultant to several industrial and government organizations (JPL, NASA etc.). In 1999, Professor Iyengar won the most prestigious research award titled Distinguished Research Award and a university medal for his research contributions in optimal algorithms for sensor fusion/image processing.

Dr. Jianhua Chen received her Ph.D. in computer science in 1988 from Jilin University, Chang Chun, China. In August 1988, She joined the Computer Science Department of Louisiana State University, Baton Rouge, USA, where she is currently an associate professor. Dr. Chen's research interests include Machine Learning and Data Mining, Fuzzy Sets and Systems, Knowledge Representation and Reasoning.