The Development of a Testbed for Evolutionary Learning Algorithms for Mobile Robotic Colonies

[1]Damion Gastelum, [1]Thomas Jones, [2]Amit Agarwal, [2]Jay Kothari, [2]Supriya Bhat, [3]Hong Kyu Lee, [4]Edward Grant, [4]Andrew Nelson, [5]Stuart Rubin, and [2]Gordon K. Lee

[1]Department of Mechanical Engineering
[2]Dept. of Electrical & Computer Engineering
San Diego State University
San Diego, CA 92182-1326
glee@kahuna.sdsu.edu

[3]Department of Electrical Engineering
Korea University of Technology and Education
Chonan P.O. Box 55, Chonan
Chungnam, 330-600, Korea

[4] Department of Electrical & Computer Engineering
Center for Robotics & Intelligent Machines
North Carolina State University
Raleigh, NC 27695-7911

[5]SPAWAR Systems Center
53560 Hull Street
San Diego, CA USA
srubin@spawar.navy.mil

Abstract

There exist many applications in military, commercial and civilian scenarios where multiple robots are required to perform complex tasks in a coordinated manner. For example, autonomous intelligent robot colonies may be used in reconnaissance missions or seek-and-capture scenarios involving a complex set of interactions between machines as well as between machines and humans and may cover long distances to remote sites. Because of the nature of the tasks, new classes of robotic systems will be required that have a high level of specification for efficiency and reliability. This, we believe, can only be accomplished through sophisticated intelligent control and efficient sensor integration as an integral part of the design of the robot and the robot's supporting systems. This paper presents an overview of a mobile robotic system testbed that is being developed at San Diego State University in partnership with North Carolina State University and SPAWAR Systems Center. Further some initial results of our work are presented.

Key words: *coordinated robots, evolutionary algorithms, swarm technology*

I. Introduction

Cooperative mobile robotic colonies, which can be considered a subset of Unmanned Vehicles (UV), shows great promise for the application of swarm technology. Swarming is the result of many small, relatively cheap, machines interacting and reacting to local sensing to yield a global behavior. For military applications, for example, this emergent behavior, "a behavior that isn't explicitly programmed, but results as the natural interaction of multiple entities", will be used in the future to control complex, unmanned systems to limit exposure of military personnel to hostile action [1]. The proliferation of unmanned systems not only will benefit from the application of this biologically-inspired control strategy, but will more than likely *require* its implementation to achieve the desired cost savings goals, and warfighting efficiencies envisioned, including reducing fratricide (friendly-fire) incidents, improved reduction in bandwidth requirements, and reduction in the number of operators and support personnel.

Of the myriad UV systems in use by the Department of Defense (DOD) today, none currently swarm, as defined above. Many routinely launch in an operator-controlled mode, then continue their mission autonomously, switching back to operator controlled mode for recovery. This is usually not a problem when the UV is operating by itself. The challenge is when the UV must operate with manned vehicles or other UVs. Currently there are eleven different DOD UAV programs in operation, and an additional ten programs on the drawing boards [2]. They range in size from the AeroVironment Wasp (less than one lb. and 8 in.) to the X-45C Unmanned Combat Air Vehicle (UCAV) (35,000 lb and 36 ft.). Over the past decade the Department of Defense has invested over $3 billion dollars on UAVs and has programmed over $16 billion in this decade for their development, procurement and operations [2]. By 2010 the number of UAVs fielded by the Department of Defense will exceed 300, not including micro and mini UAVs. Autonomous Operations (AO) technology, including swarming, is a capability requirement to meet the ever-increasing numbers and complexity of UAVs fielded [2].

The biggest gap filled in mission requirements will be by small, mini and micro UAVs. Those missions not practical for manned aircraft, such as continuous perimeter defense of a ship, in port or at sea, could be fulfilled by a swarm of small, cheap UAVs. Such a system, equipped with explosives detection sensors, could have prevented the USS COLE terrorist attack [3].

To achieve the utmost benefit to the Theater Commander, UAVs will be required to communicate and coordinate with other UAVs and manned aircraft. The Defense Advanced Research Projects Agency (DARPA) and the Armed Services have formed an Intelligent Autonomy Working Group to address this challenge [2]. It is easily foreseeable that one solution would be the application of Swarm Technology, treating each vehicle, unmanned or manned as a node in a distributed network within the battlespace architecture.

Future applications of Unmanned Undersea Vehicles (UUV) is governed by the Department of the Navy's Unmanned Undersea Vehicles Master Plan. The four main areas envisioned include: Maritime Reconnaissance, Undersea Search and Survey, Communications and Navigation Aids, Sub Track and Trail [4]. Each, in varying degrees, are candidates for application of Swarm Technology. This would be manifest in autonomous swarms of UUVs doing intelligence collection, surveillance, area reconnaissance, performing as mobile communications or navigation links to manned submarines, or ultimately as unmanned sub hunter/killers.

This paper presents some preliminary results of one of the many research programs that focus on swarm technologies and discusses some of the approaches that may be used to address on-going and future needs.

II. Learning

Training such multiple mobile robotic systems as UVs in swarms is a complex task. Fundamental to coordination of a robot colony or colonies that must perform specific tasks is the notion of learning. Indeed, training multiple mobile robotic systems in swarms may appear to be as complex a problem as herding cats. But learning implies communication, i.e., the passing of information between machines (intra-colony learning) as well as between machines and humans (extra-colony learning).

Learning in this environment may be divided into three levels of complexity: 1) local learning, 2) team/colony learning, and 3) multi-colony learning. In local learning, that is, individual robot learning, the robot must gain proficiency in performing local tasks either through off-line training from a human instructor or through learning from semi-independent exploration and experimentation. By construction, local learning usually satisfies local specific and therefore simple tasks or problems that need to be solved (gather temperature information, locate nearest neighbor, for example).

In team or colony learning, robot groups must gain proficiency in performing group tasks. The learning can again be supervised, unsupervised or some combination. Learning is done through information sharing.

The most complex scenario is multi-colony learning (more than one team or colony). In this case, a large group of robots must communicate and share learned strategies and the colonies may be communicating remotely; hence there may be time delays or other long distance communication issues that need to be addressed.

Each robot learns about a specific task assigned using a given learning strategy. Through its communication agent, knowledge can be passed to a team level learning agent. The team level learning agent provides communication between individual robots in a team and can serve as a coordinator, depending upon the specific task or set or tasks to be performed.

In this learning structure, a team or colony of robots can use any of several learning techniques suggested in the literature such as genetic algorithms, neural networks, heuristics or model-based control. Learning at the colony level can be reflected in and mapped to an intermediate representation using a mapping communication agent. The other teams or colonies can translate this intermediate representation into the appropriate form for their learning technique.

The goal of machine learning is to tractably and reliably acquire specific situational and commonsense knowledge for use in problem-solving systems. Neat systems are symbolic; scruffy systems (e.g., neural networks) are sub-symbolic [5]. *Intelligent* systems that learn have become such a prominent tool that it has increasingly taken a more important role in almost every industry. As such, they have to operate with great efficiency and accuracy. This may not sound very difficult if the environment in which they operate remains unchanged, since their behaviors could be pre-programmed. However, if the environment is ever changing, it becomes extremely difficult, if not impossible, for system developers to enumerate every possible behavior of the robot – thus the need for learning.

There are various techniques by which the system can learn. Evolutionary methods provide an attractive approach as they mimic, to some extent, biological evolution. For each generation, a population goes through some training by performing one or more complex tasks. A fitness function is generated to measure the success of each individual in the population. Part of the evolution process is that genetic operators (crossover, mutation, reproduction) affect the next generation. The premise is that evolution improves a subset of the population (survival of the fittest).

The field of evolutionary robotics has been reviewed in recent publications [6]-[9]. Important issues raised in this literature include 1) the application of ER methods to more sophisticated problems; 2) methods of performance and fitness evaluation; 3) embodied evolution in real robots vs. evolution in simulation; and 4) the coupling of simulation to reality. In this paper we focus mainly on the last issue.

III. Components of the Learning Process

The development of evolutionary robot (ER) controllers requires formulation of three components: (a) selection of the evolutionary parameters (e.g., population size, chromosome length, encoding/decoding representations of the chromosomes, cross-over operator, selection operator, mutation operator); (b) fitness function selection or generation; and (c) control structure that uses the evolution results (e.g., neural network, fuzzy logic, adaptive neural fuzzy inference).

The first component has generated much research; we plan to continue with the efforts on investigating the effects of these parameters on the learning convergence rate. Regarding the fitness function selection, most commonly, a task specific fitness selection function is formulated by hand and by trial and error. For complex behaviors, this can require in-depth knowledge of the dynamics of the behavior to be evolved. One method used to address the problem of evolution of more complex behaviors is incremental evolution [10]-[11]. Direct evaluation by humans has also been used in some ER work ([12], for example).

However, all of these methods limit the automation aspect that is central to ER. The algorithm should be robust, self-learning and self sustainable. To make it robust, robots are denied any global information about the environment and are limited to the information acquired by the sensors (possibly later communication from a fellow robot). We are developing a genetic algorithm to make it self-learning, which means that they are bound to make mistakes but learn from these mistakes, as humans do. To make it self sustainable we plan to provide it with cognitive characteristics which can help it decide what it right and what to do under the present circumstances.

To do this, we are using a fuzzy formulation for mapping the correlation coefficients to a fuzzy set through a membership function. The results of this effort are presented in a separate paper [13].

Finally, we are developing an adaptive neural fuzzy control structure such that the parameters of this ANFIS are tuned by the evolutionary process. This is the subject of [14].

IV. The Testbed

The evolution of robot controllers requires simulation and actual testing in the field. To address this issue, we have developed a testbed that is based upon the work of the Center for Robotics and Intelligent Machines at North Carolina State University [15]. In particular, we chose to use the PC/104 architecture as the central computing hardware. The baseline systems design criterion for the CPU board and required interfaces is met by the MZ104 MachZ. This is an embedded PC-on-a-Chip.

The mobile robotic system has two main hardware components: the PC/104 stack and the custom PCB. The PC/104 stack acts as the brain of the system, running Matlab, issuing commands, and interfacing to the camera and wireless network. The PCB receives simple commands from the PC/104 stack and controls the locomotion mechanism accordingly.

The PC/104 Stack consists of two connected PC/104 boards: the MZ104 and the PCMCIA adapter. The MZ104 is a complete computer (minus video display), including an 8 MB bootable Disk-On-Chip flash-chip. A USB port interfaces directly to the USB camera for image acquisition. An RS232 serial ports interfaces to the Basic-X on the custom PCB to control locomotion. Attached to the MZ104 via the PC/104 bus is the PCMCIA PC-Card adapter that holds two PC-cards. The wireless Ethernet PC-card is used to retrieve data from the self-navigation system for monitoring purposes and for Matlab to acquire a software-license for operation. The 96 MB ATA-flash-card acts as primary program storage for the system, containing a local, trimmed-down installation of Matlab. The EvBot Matlab installation includes the image processing toolbox and an entire custom Linux distribution (except for the kernel, which is booted from the 8 MB Disk-On-Chip).

The custom PCB is specific to the mobile robots. It is generic for any locomotion control mechanism accepting RS232 input. In this application, it is used for locomotion and self-navigation. This is briefly documented here to explain system interfacing and to clarify the overall system functionality. The PCB contains the Basic-X connected to the two motor drivers (dual high-current CMOS-inverters connected to each pole of the motors). The important concept is that the Basic-X can actuate the motors so as to move the platform forward and backward and to turn the platform in-place when so instructed from the MZ104 via the RS232 link. The mobile robot platform is shown in Figure 1.

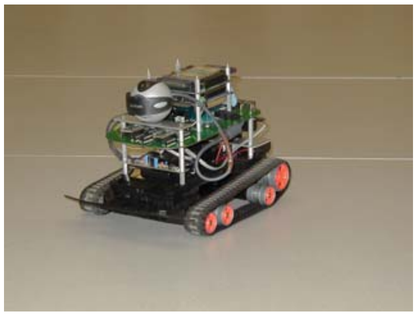
Figure 1: The Mobile Robot Platform

V. Testing and Evaluation Procedures

Testing learning algorithms for robot colonies can be accomplished by selecting a complex task that the colony must perform. Many games requiring high levels of skill can be scored in a tournament using relatively simple and deterministic metrics (measures), e.g., Checkers-Playing neural networks [16], Go-Playing neural networks [17] or Capture-the Flag [15]. In cases where at least one team or player of an evolving population achieves a win in a tournament, metric complexity can be reduced further to best number of games won in a tournament.

If one employs a neural network controller, which is typical in many evolutionary robot studies, the acquired weights in a neural network can be replaced with a symbolic model using a more or less domain-specific modeling language. The model captures features, which can be seen to provide for accelerated learning, improved predictive accuracy, and the potential for explanative content, which weights cannot offer. Given a set of sensors, which gather data over time, it is *NP-hard* to find meaningful associations (i.e., features) in the data using numeric weights. Rather than precondition the sensory inputs, we plan to use a set of sensory models that will evolve using a domain-specific language. These models will evolve through competition. New models will evolve by chance or functional transformation. Initially, models will evolve by pure chance, which of course is intractable in the limit. Models will subsequently evolve through the concomitant evolution of *recursive functionals*, which map models to candidate isomorphs. This is a heuristic process. It differs from formal logics in that (1) it is self-organizing, (2) it is far less sensitive to noise and missing information, (3) partial knowledge is transferable across domains, and (4) computation is not serially bottlenecked.

The implications for autonomous robots that learn by doing or by being told are profound (e.g., fusing acquired knowledge across an autonomous robotic swarm).

A maze has been designed, based upon the NCSU Maze (Figure 2), which can be modified to provide more complex and unknown obstacles for the robots to traverse. Initially, we will use the Go-Playing game in order to study the learning behavior of the robot colonies. Other tasks and goal-playing scenarios will also be employed in order to study the learning behavior of robot colonies.

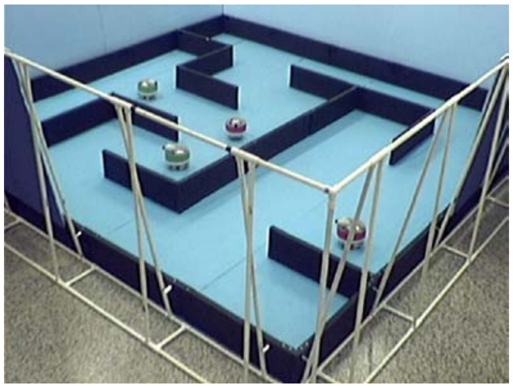
Figure 2: The NCSU Maze

VI. Preliminary Results

In order to evaluate some of the algorithms that we are developing in support of the mobile robotics testbed program, we developed an evolutionary-based controller using an artificial neural network (ANN) controller with a variable population (chromosome) size. An initial population of two random binary sets representing the chromosomes were generated and used to tune the ANN controller for learning. The main goal of this initial study is to see how the changes in the length of chromosomes affect the rate of learning. Once the initial population is used to get the initial error, we used the genetic operators to modify the characteristics of the chromosome either by crossover or mutation. Depending upon the crossover or mutation probabilities, the initial population (the child) is changed, which is then used for the next learning process. This process continues till the stopping criteria (in this case, a minimum error is achieved).

Though the results depend to some extent upon the initial randomly-generated population along with the probability for crossover and mutation, we have observed that different length of chromosome do affect the learning.

For a fixed set of generations, we have simulated several different tests. Results are shown in Table 1 for a maximum of 20 generations. We noticed that as the population size of the chromosomes is increased, the error criteria had a higher probability of being met in less than the maximum number of generations;

however, we also found population size lengths in which they did not meet the required criteria in the specified number of generations.

One observation that requires further study is that when the population size is large enough and the mutation probability is small enough, the genetic algorithm converges to a global optimum (satisfies the error criterion) with a high probability resulting in good algorithm performance. This is subject to further study.

Table 1: Results of Population Size Variation

Length Of Chromosomes	Criteria Met/not Met	Comments
2	Met	Met at generation near 20
5	Not Met	
8	Not Met	
10	Met	Met at generation near 20
12	Met	A bit faster met near 18
15	Not Met	
18	Met	Fastest in our run, met near 15
20	Not Met	

we noticed was that as the number of generation were increased, they usually tended towards meeting the criteria in less number of generations, but at the same time there are lengths in which they did not meet the required criteria in the specified number of generations.

Figure 3 shows two examples of the simulation results using a population size of 12 and a population size of 15. One of our tasks is to adaptively change the population size as well as crossover and mutation operators as part of the generation evolution.

VII.Conclusions and Future Work

In this paper, a testbed for developing evolutionary learning algorithms for robot colonies is presented. Mobile robots will learn to navigate through several types of mazes. Further, software simulation modules will also be developed to test and evaluate competition, cooperation, and subsequent learning by an autonomously synthesized set of software agents. In particular, a swarm of *softbots* will be assigned starting conditions and reconnaissance goals. Their movement in the virtual world will be governed by an externally supplied rule base.

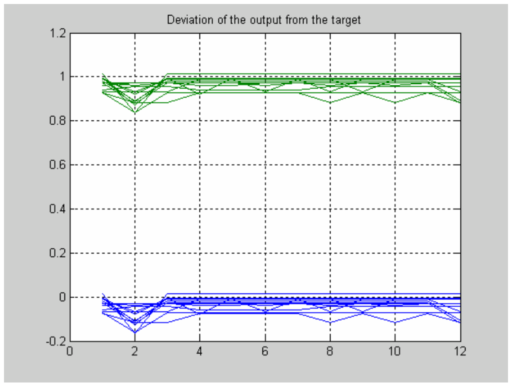

Figure 3a: Population Size is 12. Top plots - learning metric; bottom plots - errors

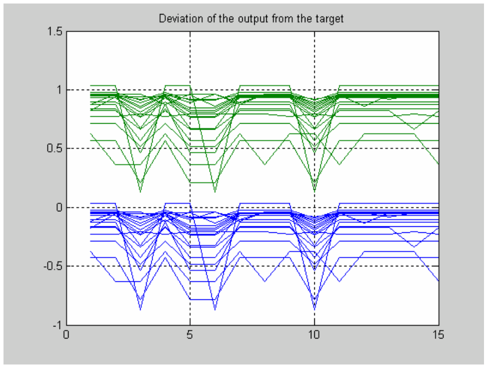

Figure 3b: Population Size is 15. Top plots - learning metric; bottom plots - errors

In this paper, a testbed for developing evolutionary learning algorithms for robot colonies is presented. Mobile robots will learn to navigate through several types of mazes. Further, software simulation modules will also be developed to test and evaluate competition, cooperation, and subsequent learning by an autonomously synthesized set of software agents. In particular, a swarm of *softbots* will be assigned starting conditions and reconnaissance goals. Their movement in the virtual world will be governed by an externally supplied rule set. Performance metrics will be set to encourage cooperative behavior (i.e., transference and fusion of acquired knowledge).

The fundamental capability for domain transference or, “learning how to learn” can be expected to enable the creation of truly intelligent autopilots that take their mission orders from down-loadable knowledge bases. The results of this on-going effort will enable the development of robotic reconnaissance vehicles and sensors, tactical mobile robots, mobile command, control and communications platforms, and targeting systems operating on land, sea, or in the air. It is hoped that this on-going effort will provide some insight into the learning requirements for mobile robotic colonies.

VIII. References

1. B. Clough, "UAV Swarming? So What Are Those Swarms, What Are The Implications, and How Do We Handle Them?", Air Vehicles Directorate; Air Force Research Laboratory; Air Force Materiel Command; Wright-Patterson Air Force Base, OH 45433-7542, AFRL-VA-WP-TP-2002-308, April 2002.
2. Unmanned Aerial Vehicles (UAV) Roadmap 2002; Office of The Secretary of Defense, Washington D.C. Internet: http://www.acq.osd.mil/usd/uav_roadmap.pdf, 2003
3. Homegrown, Affordable UAV Concept to be Produced for Navy Use; Zoccola, Mary; Internet: http://www.dt.navy.mil/pao/excerpts%20pages/2001/UAV3_01.html, 2001.
4. The Navy Unmanned Undersea Vehicle (UUV) Master Plan, Office of Naval Research, Washington, D.C.; Internet: http://www.onr.navy.mil/02/baa/expired/baa01%5F012/pip/uuvmp.pdf, 2002.
5. S.H. Rubin, "Computing with Words," *IEEE Trans. Syst. Man, Cybern.*, vol. 29, no. 4, pp. 518-524, 1999.
6. S.H. Rubin, "On Knowledge Amplification by Structured Expert Randomization (KASER)," *SSC-San Diego Biennial Review*, San Diego, CA, TD 3117, pp. 70-91, 2001.
7. S. Amarel, "On Representations of Problems of Reasoning about Actions," *Machine Intelligence*, vol. 3, pp. 131-171, 1968.
8. S. Nolfi and D. Floreano, *Evolutionary Robotics: the Biology, Intelligence, and Technology of Self-organizing Machines,* MIT Press, 2000.
9. Nelson, A.L., Grant, E. and Lee, G., "Using Genetic Algorithms to Capture Behavioral Traits Exhibited by Knowledge Based Robot Agents", Proceedings of the ISCA Int'l Conference on Computers in Industry and Engineering, San Diego, 2002.
10. J-H. Lin and J.S. Vitter, "Complexity Results on Learning by Neural Nets," *Mach. Learn.*, vol. 6, no. 3, pp. 211-230, 1991.
11. S.H. Rubin, "A Heuristic Logic for Randomization in Fuzzy Mining," *J. Control and Intelligent Systems*, vol. 27, no. 1, pp. 26-39, 1999.
12. A.W. Biermann, "Automatic Programming: A Tutorial on Formal Methodologies," *J. Symbolic Computation*, vol. 1, no. 2, pp. 119-142, 1985.
13. B. Supriya and G.K. Lee, " Performance Metrics for Evolutionary Controllers", to appear, 2003.
14. J. Kothari, A. Agarwal and G.K. Lee, "Evolutionary Learning Using an Adaptive Population Size for an ANFIS Controller", to appear, 2003.
15. Galeotti, J., Rhody, S., Nelson, A., Grant, E., and Lee, G., "EvBots-The Design and Construction of a Mobile Robot Colony for Conducting Evolutionary Robotic Experiments, ", Proceedings of the ISCA Int'l Conference on Computers in Industry and Engineering, San Diego, 2002.
16. J. Hindin, "Intelligent Tools Automate High-Level Language Programming," *Computer Design*, vol. 25, pp. 45-56, 1986.
17. L.A. Zadeh, "Fuzzy Sets," *Information and Control*, vol. 8, pp. 338-353, 1965.