Transference of Evolved Unmanned Aerial Vehicle Controllers to a Wheeled Mobile Robot

Gregory J. Barlow*, Leonardo S. Mattos, and Edward Grant
Center for Robotics and Intelligent Machines
Department of Electrical and Computer Engineering
North Carolina State University, Raleigh, NC 27695
gjb@cmu.edu, {lsmattos, egrant}@ncsu.edu

Choong K. Oh
U.S. Naval Research Laboratory
4555 Overlook Ave. S.W., Washington, DC 20375
choong.oh@nrl.navy.mil

Abstract— **Transference of controllers evolved in simulation to real vehicles is an important issue in evolutionary robotics (ER). We have previously evolved autonomous navigation controllers for fixed wing UAV applications using multi-objective genetic programming (GP). Controllers were evolved to locate a radar source, navigate the UAV to the source efficiently using on-board sensor measurements, and circle around the emitter. We successfully tested an evolved UAV controller on a wheeled mobile robot. A passive sonar system on the robot was used in place of the radar sensor, and a speaker emitting a tone was used as the target in place of a radar. Using the evolved navigation controller, the mobile robot moved to the speaker and circled around it. The results from this experiment demonstrate that our evolved controllers are capable of transference to real vehicles. Future research will include testing the best evolved controllers by using them to fly real UAVs.**

I. Introduction

While some controllers in evolutionary robotics (ER) [1] have been evolved *in situ* on physical robots, evolution requires many evaluations to produce good behaviors, which generally takes an excessive amount of time on real robots. Evolving controllers in simulation is less constraining for many problems, because evaluations are much faster and can be parallelized. Since simulation environments cannot be perfectly equivalent to the conditions a real robot would face, transference of controllers evolved in simulation to real robots has been an important issue.

In early ER work, controllers were often evolved directly on physical robots [2], [3]. Though the availability of computational power has made simulation increasingly attractive, some research using embodied evolution continues [4], [5]. While embodied evolution can directly test the performance of controllers in the real world, it can be slow, which constrains the complexity of problems that can be solved in a reasonable amount of time.

Simulation has been used since the beginning of ER research [6] for evolving robot controllers with varying degrees of success. Some simulated controllers are never tested on actual robots, and some fail to transfer well. However, many controllers evolved in simulation have been successfully transferred to real robots [7]–[10]. Adding noise to the simulation is one method that has proven successful in evolving controllers that transfer well to real robots [8].

In previous research, we evolved autonomous navigation controllers for flying unmanned aerial vehicles (UAVs) in simulation using multi-objective genetic programming [11]–[13]. Controllers were evolved to locate a radar, navigate the UAV to the source efficiently using sensor measurements, and circle around the radar. With the goal of making evolved navigation controllers robust and the intent of transferring controllers to physical UAVs, we abstracted navigation from low level flight, tuned the simulation parameters for equivalence to real UAVs, and added noise to the simulation. While flight tests are planned for the future, UAVs were not immediately available for testing. We wanted to ensure that evolved navigation controllers were transferable to real vehicles before beginning expensive UAV flight tests. To evaluate the ability of evolved controllers to control real vehicles, we transferred evolved UAV navigation controllers to a wheeled mobile robot.

II. Mobile Robot Platform

In this research, we used a small autonomous mobile robot called the EvBot II (shown in Fig. 1). A colony of nine EvBot II mobile robots was developed by the Center for Robotics and Intelligent Machines at North Carolina State University for evolutionary robotics experimentation [14], [15]. The EvBot is 9 inches wide by 12 inches long by 10 inches tall, and is constructed on a two track treaded wheel base. The robots features powerful motors, wheel encoders and an utility board that hosts two microcontrollers, which can drive up to four DC or servo motors in closed loop. The robot is equipped with a PC/104 onboard computer running Linux that is responsible for all computation, data acquisition and high-level control. The robot is connected to a wireless network and supports video data acquisition through a USB video camera.

*Present address: The Robotics Institute, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213

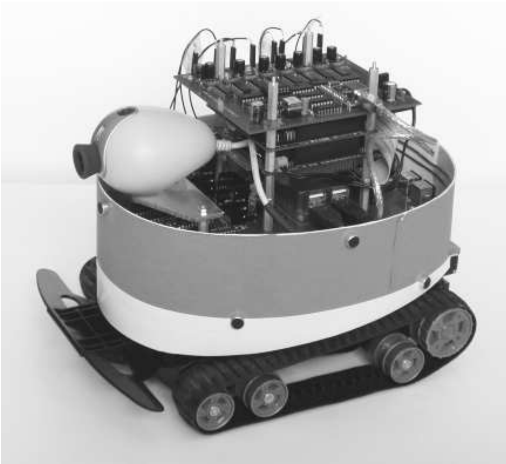


Fig. 1. The EvBot II, a small mobile robot equipped with an acoustic array

The EvBot is equipped with an on-board passive sonar system. This sensor system makes use of an acoustic array formed by eight microphones distributed in a fixed 3-D arrangement around the robot. It uses data collected from the array to perform beamforming and to find the direction and intensity of sound sources. The audio data is obtained by the passive sonar system through the use of a custom USB data acquisition board, which simultaneously samples all eight sensors. The sampled signals are then transferred to the PC/104 computer and processed by a beamforming algorithm. The beamforming algorithm used in this application performs shifting and linear combination of the sampled input signals to calculate the magnitude of the sounds coming from all 360° around the robot [16]. The passive sonar system is susceptible to environmental noise, and the direction of a source found by the acoustic array is only accurate within approximately $\pm 45°$.

III. Unmanned Aerial Vehicle Control

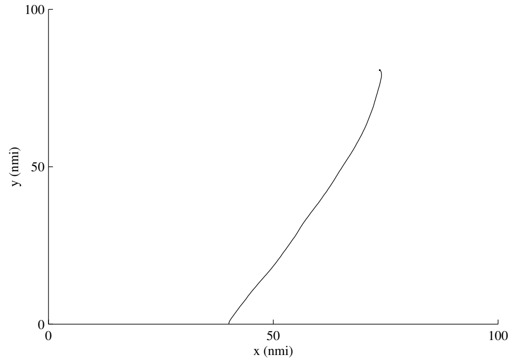
In previous work, we have developed autonomous navigation controllers for fixed wing UAVs using multi-objective genetic programming [11]–[13]. The goal is for a UAV to autonomously locate, track, and then orbit around a radar site. There are three main goals for an evolved controller. First, the UAV should move to the vicinity of the radar as quickly as possible. The sooner the UAV arrives in the vicinity of the radar, the sooner it can begin its primary mission, such as surveillance. Second, once in the vicinity of the source, the UAV should circle as closely as possible around the radar. This goal is especially important when proximity to the source affects the success of the application. Third, the flight path should be stable and efficient. The roll angle should change as infrequently as possible, and any change in roll angle should be small. Making frequent changes to the roll angle of the UAV could create dangerous flight dynamics and could reduce the flying time and range of the UAV.

Only the navigation portion of the flight controller is evolved; the low level flight control is done by an autopilot. The navigation controller receives radar electromagnetic emissions as input, and based on this sensory data and past information, the navigation controller defines the desired roll angle of the UAV control surface. The autopilot then uses this desired roll angle to change the heading of the UAV. This autonomous navigation technique results in a general controller model that can be applied to a wide variety of vehicle platforms; the evolved controllers are not designed for a specific UAV airframe or autopilot.

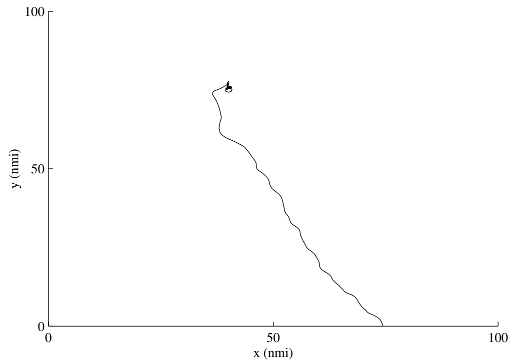
The controller is evolved in simulation. The simulation environment is a square 100 nautical miles (nmi) on each side. For each simulation, the initial positions of the UAV and the radar are set randomly. The initial UAV position is always on the southern edge of the simulation area with an initial heading of due north; the initial radar position may be anywhere in the environment. In our current research, the UAV has a constant altitude and a constant speed of 80 knots. Each experimental run simulates four hours of flight time, where the UAV is allowed to update its desired roll angle once a second. The interval between these requests to the autopilot can also be adjusted in the simulation.

Only the sidelobes of the radar emissions are modeled. The sidelobes of a radar signal have a much lower power than the main beam, making them harder to detect. However, the sidelobes exist in all directions, not just where the radar is pointed. This model is intended to increase the robustness of the system, so that the controller doesn't need to rely on a signal from the main beam. Additionally, Gaussian noise is added to the amplitude of the radar signal. The receiving sensor can perceive only two pieces of information: the amplitude and the angle of arrival (AoA) of incoming radar signals. The AoA measures the angle between the heading of the UAV and the source of incoming electromagnetic energy. Real AoA sensors do not have perfect accuracy in detecting radar signals, so the simulation models an inaccurate sensor. The accuracy of the AoA sensor can be set in the simulation. In previous experiments, the sensed AoA was set to be accurate within $\pm 10°$ at each time step, a realistic value for this type of sensor. For the transference experiments described here, the AoA was accurate within $\pm 45°$, the approximate accuracy of the acoustic array on the EvBot. As might be expected, controllers evolved on lower assumed levels of error were not particularly fit when error increased this much. Fig. 2(a) shows a simulated flight path for a controller evolved and tested on a continuously emitting, stationary radar with a sensor accurate within $\pm 10°$. Fig. 2(b) shows the same controller tested with a sensor accurate within $\pm 45°$. Rather than using one of the controllers evolved in previous research [11]–[13], we evolved new controllers to transfer to an EvBot. The only change in the simulation from the previous research was a change in the accuracy of the simulated sensor to $\pm 45°$.

Transference of these controllers to a real UAV is an important issue. Flying a physical UAV with an evolved controller is planned as a demonstration of the research,



(a)



(b)

Fig. 2. Flight paths for a UAV controller to a continuously emitting, stationary radar using a sensor accurate within (a) $\pm 10°$ and (b) $\pm 45°$.

so transference was taken into consideration from the beginning. Several aspects of the controller evolution were designed specifically to aid in this process. First, the navigation control was abstracted from the flight of the UAV. Rather than attempting to evolve direct control, only the navigation was evolved. This allows the same controller to be used for different airframes. Second, the simulation parameters were designed to be tuned for equivalence to real aircraft. For example, the simulated UAV is allowed to update the desired roll angle once per second reflecting the update rate of the real autopilot of a UAV being considered for flight demonstrations of the evolved controller. For autopilots with slower response times, this parameter could be increased. Third, noise was added to the simulation, both in the radar emissions and in sensor accuracy. A noisy simulation environment encourages the evolution of robust controllers that are more applicable to real UAVs.

While a human could easily design a controller using a very accurate sensor, the sonar sensor used in this experiment was very noisy. By using evolution to design controllers, cheaper sensors with lower accuracy can be used. As the accuracy of the sensors decreases, the problem becomes far more difficult for human designers.

TABLE I

GENETIC PROGRAMMING PARAMETERS.

Population Size	500	Maximum Initial Depth	5
Crossover Rate	0.9	Maximum Depth	21
Mutation Rate	0.05	Generations	600
Tournament Size	2	Trials per evaluation	30

IV. MULTI-OBJECTIVE GENETIC PROGRAMMING

As in our previous work [11]–[13], navigation controllers intended for UAVs were designed using multi-objective genetic programming which employs non-dominated sorting, crowding distance assignment to each solution, and elitism. We evolved UAV controllers using an implementation of NSGA-II [17] for genetic programming. The function and terminal sets combine a set of very common functions used in GP experiments and some functions specific to this problem. The function and terminal sets, unchanged from our previous work, are defined as

F = {*Prog2, Prog3, IfThen, IfThenElse, And, Or, Not,* $<$, $\leq$, $>$, $\geq$, < 0, > 0, =, +, -, *, $\div$, $X < 0$, $Y < 0$, $X > max$, $Y > max$, $Amplitude > 0$, $AmplitudeSlope > 0$, $AmplitudeSlope < 0$, $AoA > Arg$, $AoA < Arg$}

T = {*HardLeft, HardRight, ShallowLeft, ShallowRight, WingsLevel, NoChange, rand, 0, 1*}

The position is given by the x and y distances from the origin, located in the southwest corner of the simulation area. This position information is available using the functions that include *X* and *Y*, with *max* equal to 100 nmi, the length of one side of the simulation area. The vehicle is free to move outside of the area during the simulation, but the target is always placed within it. The two available sensor measurements are the amplitude of the incoming signal and the AoA, or angle between the heading and the source of incoming electromagnetic energy. Additionally, the slope of the amplitude with respect to time is available to GP. When turning, there are six available actions. Turns may be hard or shallow, with hard turns making a $10°$ change in the roll angle and shallow turns a $2°$ change. The *WingsLevel* terminal sets the roll angle to 0, and the *NoChange* terminal keeps the roll angle the same. Multiple turning actions may be executed during one time step, since the roll angle is changed as a side effect of each terminal. The final roll angle after the navigation controller is finished executing is passed to the autopilot. The maximum roll angle is $45°$. Each of the six terminals returns the current roll angle.

Genetic programming was generational, with crossover and mutation similar to those outlined by Koza in [18]. The parameters used by GP are shown in Table I. Tournament selection was used. Initial trees were randomly generated using ramped half and half initialization. No parsimony pressure methods were used in this work, as code bloat was not a major problem.

In GP, the evaluation process of individuals in a population takes significant computational time, since the simulation must be run multiple times to obtain fitness values for individuals. Therefore, using massively parallel computational processors to parallelize these evaluations is advantageous. Parallel computation was designed by employing the concept of master and slave nodes. Among multiple computer processors, one processor was designated as a master and the rest were set as slaves. The master processor distributes individual evaluations over the slave processors, and each slave processor reports its results back to the master after completing computation. After the master processor collects all individual fitness values from slave processors, GP moves to the selection process. The data communication between master and slave processors was possible using the Message Passing Interface (MPI) standard [19] under the Linux operating system. All computations were done on a Beowulf cluster parallel computer with ninety-two 2.4 GHz Pentium 4 processors.

V. Fitness Functions

Four fitness functions determine the success of individual UAV navigation controllers. The fitness of a controller was measured over 30 simulation trials, where the UAV and radar positions were different for every run. We designed the four fitness measures to satisfy the three goals of the evolved controller: moving toward the emitter, circling the emitter closely, and flying in an efficient way.

A. Normalized distance

The primary goal of the UAV is to fly from its initial position to the radar site as quickly as possible. We measure how well controllers accomplish this task by averaging the squared distance between the UAV and the goal over all time steps. We normalize this distance using the initial distance between the radar and the UAV in order to mitigate the effect of varying distances from the random placement of radar sites. The normalized distance fitness measure is given as

$$fitness_1 = \frac{1}{T}\sum_{i=1}^{T}\left[\frac{distance_i}{distance_0}\right]^2$$

where *T* is the total number of time steps, $distance_0$ is the initial distance, and $distance_i$ is the distance at time *i*. We are trying to minimize $fitness_1$.

B. Circling distance

Once the UAV has flown in range of the radar, the goal shifts from moving toward the source to circling around it. An arbitrary distance much larger than the desired circling radius is defined as the in-range distance. For this research, the in-range distance was set to be 10 nmi. The circling distance fitness metric measures the average distance between the UAV and the radar over the time the UAV is in-range. While the circling distance is also measured by $fitness_1$, that metric is dominated by distances far away from the goal and applies very little evolutionary pressure to circling behavior. The circling distance fitness measure is given as

$$fitness_2 = \frac{1}{N}\sum_{i=1}^{T} inrange * (distance_i)^2$$

where *N* is the amount of time the UAV spent within the in-range boundary of the radar and *inrange* is 1 when the UAV is in-range and 0 otherwise. We are trying to minimize $fitness_2$.

C. Level time

In addition to the primary goals of moving toward a radar site and circling it closely, it is also desirable for the UAV to fly efficiently in order to minimize flight time to get close to the goal and to prevent potentially dangerous flight dynamics, like frequent and drastic changes in the roll angle. The first fitness metric that measures the efficiency of the flight path is the amount of time the UAV spends with its wings level to the ground, which is the most stable flight position for a UAV. This fitness metric only applies when the UAV is outside the in-range distance, since once the UAV is within the in-range boundary, we want it to circle around the radar. The level time is given as

$$fitness_3 = \sum_{i=1}^{T}(1 - inrange) * level$$

where *level* is 1 when the UAV has been level for two consecutive time steps and 0 otherwise. We are trying to maximize $fitness_3$.

D. Turn cost

The second fitness measure intended to produce an efficient flight path is a measure of turn cost. While UAVs are capable of very quick, sharp turns, it is preferable to avoid them. The turn cost fitness measure is intended to penalize controllers that navigate using a large number of sharp, sudden turns because this may cause very unstable flight, even stalling. The UAV can achieve a small turning radius without penalty by changing the roll angle gradually; this fitness metric only accounts for cases where the roll angle has changed by more than $10°$ since the last time step. The turn cost is given as

$$fitness_4 = \frac{1}{T}\sum_{i=1}^{T} h\_turn * |roll\_angle_i - roll\_angle_{i-1}|$$

where *roll_angle* is the roll angle of the UAV and *h_turn* is 1 if the roll angle has changed by more than $10°$ since the last time step and 0 otherwise. We are trying to minimize $fitness_4$.

E. Combining the Fitness Measures

These four fitness functions were designed to evolve particular behaviors, but the optimization of any one function could conflict heavily with the performance of the others. Even though the controller doesn't generate the most optimized controllers possible, it can obtain near-optimal solutions. Combining the functions using multi-objective optimization is extremely attractive due to the use of non-dominated sorting. The population is sorted into ranks, where within a rank no individual is dominant in all four fitness metrics.

While all four objectives were important, moving the UAV to the goal was the highest priority. There are several techniques to encourage one objective over the rest; in this research, we used a simple form of incremental evolution [20]. For the first 200 generations, only the normalized distance fitness measure was used. Multi-objective optimization using all four objectives was used for the last 400 generations of evolution.

We have previously evolved UAV navigation controllers using both direct evolution and environmental incremental evolution for a variety of radar types, including: 1) continuously emitting, stationary radars, 2) intermittently emitting, stationary radars, 3) continuously emitting, mobile radars, and 4) intermittently emitting, mobile radars. Only the continuously emitting, stationary radar case was used in the experiments described in this paper since it could be directly tested using the EvBot.

VI. Transference to a Wheeled Mobile Robot

Testing the evolved controllers on the EvBot was attractive because the passive sonar system [15] is an acoustic analog to the radar sensor used in simulation. The two signal types propagate similarly, and both sensors detect signal strength and direction. The similarities between the two systems made it possible to transfer evolved UAV controllers to an EvBot. In evaluating the transference of the evolved controllers, we were not interested in showing optimal behavior on the robot platform. Instead, our concern was that the controllers should exhibit the same behaviors on the real robots as they did in simulation, particularly robustness to noise.

Transference experiments were done in an arena constructed for EvBot tests. This 153 inch by 122 inch area can be set up as a maze for certain experiments, but for this experiment it was completely open. A video camera with a fisheye lens is mounted above the maze environment to document experiments. To test the controllers, we performed a series of experiments. In each experiment, the robot was placed along one wall facing toward the middle of the environment. A speaker was suspended a foot above the ground and continuously emitted a 300 Hz tone. A circle was placed directly underneath the speaker as a visual reference point, since the fisheye lens tended to distort the location of the speaker in images captured by the overhead camera. Robot movement was discretized into steps, much like in the simulation. At each time step, the controller was executed to produce a roll angle. Since the EvBot is not controlled by a roll angle, the robot would turn to control direction. The differential drive system on the EvBot allowed all turns to happen in place, without changing the location of the robot. The EvBot was only calibrated to turn at multiples of $5°$ and the magnitude of

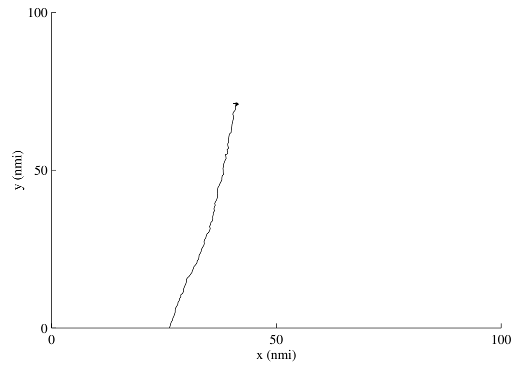


Fig. 3. Simulated path for a UAV flying to a continuously emitting, stationary radar using an evolved controller and a sensor accurate within $\pm 45°$.

the turn angle was always rounded down to the nearest multiple of 5. Calibrating the EvBot to turn at angles smaller than $5°$ would have been unreliable due to the size of the EvBot and the characteristics of its motors. After turning, the EvBot would always move forward the same amount, mimicking the constant speed of the UAV in simulation. The EvBot moved 3 inches per time step, and in simulation the UAV moved $0.0\overline{2}$ nautical miles per time step. If these values are used to scale the maze environment, then the maze would represent an area approximately 1.13 nmi by 0.90 nmi. Hence, these experiments were not testing the entire flight path, only the very end of flight when the vehicle nears the target.

An evolved controller was tested 10 times on an EvBot. We chose this controller from the evolved population based primarily on good fitness values for normalized distance and circling distance, though level time and turn cost were also used. This controller was able to successfully drive the EvBot from its starting position to the speaker and then circle around the speaker. This small number of tests was enough to confirm that the controllers were consistently able to perform the task as desired.

Fig. 3 shows a simulated flight path for the evolved controller. This controller had good fitness values, though comparing Fig. 2a and Fig. 3 shows that the controllers evolved with a more inaccurate sensor were not as well adapted as those from previous work using more accurate sensors [12]; the flight path is much less smooth and requires more turns. This controller was very successful when transferred to the EvBot. Fig. 4 shows a path from one of the experiments using this controller. Running this evolved controller on the EvBot produces a tight circling behavior with a regular orbit around the target. This behavior is very similar to the corresponding circling behavior in simulation, shown in Fig. 5. This controller produces less efficient flight paths than other previously evolved controllers because the sensor was less accurate [11]. However, the robots produced the four desired behaviors reliably. One sign of an extremely fit controller is a centering of the emitter when the controller begins to circle. A sign that the controllers evolved for use on the EvBot were less

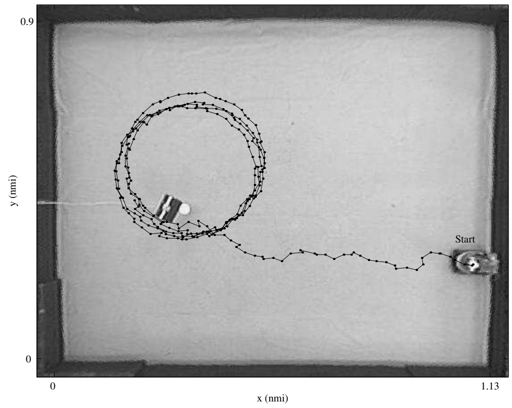


Fig. 4. Path for an EvBot running an evolved controller moving to a continuously emitting, stationary speaker using a real acoustic array sensor. The scaled maze size is shown in nautical miles.

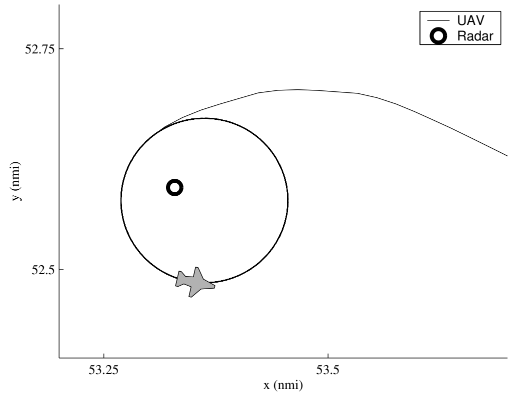


Fig. 5. Circling behavior in simulation of evolved navigation controllers (controllers were evolved and tested on $\pm 10^{\circ}$ accurate sensors).

well evolved than those from previous research was that the target was not as close to the center of the orbit when the robot began to circle.

VII. Conclusions

Using genetic programming with multi-objective optimization, we evolved autonomous UAV navigation controllers capable of flying to a target radar, circling the radar site, and maintaining a stable and efficient flight path. Four fitness functions were used to produce the desired behaviors. Controllers were evolved to use inaccurate sensors in a noisy environment. UAVs were not immediately available for flight testing, so we tested the transferability of the evolved controllers by using them to control a wheeled mobile robot. Evolved UAV controllers were successfully transferred to a wheeled mobile robot equipped with a passive sonar system which provided the angle and amplitude of sound signals from a stationary speaker. Using evolved navigation controllers, the mobile robot moved to the speaker and circled around it. The results from this experiment demonstrate that our evolved controllers are capable of transference to real physical vehicles. In the next stage of this research, we will test evolved controllers on physical UAVs.

VIII. Acknowledgments

Financial support for this work was provided by the U.S. Office of Naval Research with Dr. M. Soto as program manager. Computational resources were provided by the U.S. Naval Research Laboratory (Code 5730). The EvBot robot colony was provided by the Center for Robotics and Intelligent Machines at North Carolina State University.

References

[1] S. Nolfi and D. Floreano, *Evolutionary Robotics*. MIT Press, 2000.

[2] S. Nolfi, D. Floreano, O. Miglino, and F. Mondada, "How to evolve autonomous robots: Different approaches in evolutionary robotics," in *Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems*, 1994.

[3] D. Floreano and F. Mondada, "Evolution of homing navigation in a real mobile robot," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 26, no. 3, pp. 396–407, 1996.

[4] M. Ebner, "Evolution of a control architecture for a mobile robot," in *Proceedings of the Second International Conference on Evolvable Systems*, pp. 303–310, 1998.

[5] D. Marocco and D. Floreano, "Active vision and feature selection in evolutionary behavioral systems," in *From Animals to Animats 7*, MIT Press, 2002.

[6] J. R. Koza, "Evolution of subsumption using genetic programming," in *Proceedings of the First European Conference on Artificial Life*, pp. 110–119, MIT Press, 1992.

[7] D. Filliat, J. Kodjabachian, and J.-A. Meyer, "Incremental evolution of neural controllers for navigation in a 6-legged robot," in *Proceedings of the Fourth International Symposium on Artificial Life and Robots* (Sugisaka and Tanaka, eds.), 1999.

[8] N. Jakobi, P. Husbands, and I. Harvey, "Noise and the reality gap: The use of simulation in evolutionary robotics," in *Proceedings of the 3rd European Conference on Artificial Life*, pp. 704–720, 1995.

[9] W.-P. Lee, J. Hallam, and H. H. Lund, "Applying genetic programming to evolve behavior primitives and arbitrators for mobile robots," in *Proceedings of the IEEE International Conference on Evolutionary Computation*, pp. 495–499, 1997.

[10] A. L. Nelson, E. Grant, G. J. Barlow, and T. C. Henderson, "A colony of robots using vision sensing and evolved neural controllers," in *Proceedings of the IEEE Conference on Intelligent Robots and Systems*, (Las Vegas), October 2003.

[11] G. J. Barlow, "Design of autonomous navigation controllers for unmanned aerial vehicles using multi-objective genetic programming," Master's thesis, North Carolina State University, Raleigh, NC, March 2004.

[12] C. K. Oh and G. J. Barlow, "Autonomous controller design for unmanned aerial vehicles using multi-objective genetic programming," in *Proceedings of the Congress on Evolutionary Computation*, (Portland, OR), June 2004.

[13] G. J. Barlow, C. K. Oh, and E. Grant, "Incremental evolution of autonomous controllers for unmanned aerial vehicles using multi-objective genetic programming," in *Proceedings of the 2004 IEEE Conference on Cybernetics and Intelligent Systems (CIS)*, (Singapore), pp. 688–693, December 2004.

[14] J. M. Galeotti, "The EvBot: A small autonomous mobile robot for the study of evolutionary algorithms in distributed robotics," Master's thesis, North Carolina State University, Raleigh, NC, 2002.

[15] L. S. Mattos, "The EvBot II," Master's thesis, North Carolina State University, Raleigh, NC, 2003.

[16] L. S. Mattos and E. Grant, "Passive sonar applications: Target tracking and navigation of an autonomous robot," in *Proceedings of the IEEE International Conference on Robotics and Automation*, (New Orleans, LA), April 2004.

[17] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, pp. 182–197, April 2002.

[18] J. Koza, *Genetic Programming*. MIT Press, 1992.

[19] P. Pacheco, *Parallel Programming with MPI*. Morgan Kaufmann Publishers, Inc., 1996.

[20] J. F. Winkeler and B. S. Manjunath, "Incremental evolution in genetic programming," in *Genetic Programming 1998: Proceedings of the Third Annual Conference*, pp. 403–411, 1998.