Incremental Evolution of Autonomous Controllers for Unmanned Aerial Vehicles using Multi-objective Genetic Programming

Gregory J. Barlow[1,2], Choong K. Oh[2], and Edward Grant[1]

[1] Center for Robotics and Intelligent Machines, Department of Electrical and Computer Engineering, North Carolina State University, Raleigh, NC 27695-7911
{gjbarlow, egrant}@ncsu.edu
[2] U.S. Naval Research Laboratory, 4555 Overlook Ave. S.W., Washington, DC 20375
choong.oh@nrl.navy.mil

Abstract. Autonomous navigation controllers were developed for fixed wing unmanned aerial vehicle (UAV) applications using incremental evolution with multi-objective genetic programming (GP). We designed four fitness functions derived from flight simulations and used multi-objective GP to evolve controllers able to locate a radar source, navigate the UAV to the source efficiently using on-board sensor measurements, and circle closely around the emitter. We selected realistic flight parameters and sensor inputs to aid in the transference of evolved controllers to physical UAVs. We used both direct and environmental incremental evolution to evolve controllers for four types of radars: 1) continuously emitting, stationary radars, 2) continuously emitting, mobile radars, 3) intermittently emitting, stationary radars, and 4) intermittently emitting, mobile radars. The use of incremental evolution drastically increased evolution's chances of evolving a successful controller compared to direct evolution. This technique can also be used to develop a single controller capable of handling all four radar types. In the next stage of research, the best evolved controllers will be tested by using them to fly real UAVs.

1 Introduction

Incremental evolution [1] is the process of evolving a population on a simple problem and then using the resulting evolved population as a seed to evolve a solution to a related problem of greater complexity. Solutions to a variety of complicated problems have been evolved using incremental evolution. There are two types of incremental evolution. Functional incremental evolution [2–4] changes the difficulty of the fitness function in order to increase the difficulty of the problem. Environmental incremental evolution [5] changes the environment to increase difficulty without changing the fitness function.

Evolutionary robotics (ER) [6] uses a population-based evolutionary algorithm to evolve autonomous robot controllers for a target task. Most of the controllers evolved in ER research to date have been developed for simple problems requiring a small number of behaviors; very little of the ER work to date has been intended for use in real-life applications. A majority of the research in ER has focused on wheeled mobile robot platforms [3,5,6]. An application of ER that has received very little attention is the unmanned aerial vehicle (UAV). UAVs are becoming increasingly popular for many applications, particularly where high risk or accessibility are issues.

Genetic programming (GP) has been increasingly successful in the evolution of robot controllers capable of complex tasks. While artificial neural networks have traditionally been the most popular controller structure used in ER [1, 5], GP has also been shown to produce functional behaviors for autonomous robot control [3].

One of the main difficulties of ER is the formulation of fitness functions [7]. For many problems explored to date in ER, fitness functions that combined multiple objectives were synthesized using extensive human knowledge of the domain or trial and error. For problems without a single, easily quantifiable objective, an alternative is multi-objective optimization, which allows the evolutionary algorithm to optimize on multiple fitness metrics [8]. Rather than combining multiple objectives into a single function [3], multi-objective GP optimizes over multiple functions [9]. A non-dominated sort is used to determine the relative rank of individuals in the population [10], since this technique produces multiple fitness values for each individual. Very rarely does multi-objective optimization produce a single best solution. Instead, a Pareto front of solutions is produced, where all solutions on that front are non-dominated [8].

In this paper, we present our approach to incrementally evolving autonomous behavioral navigation controllers for fixed wing UAVs using multi-objective GP. Both types of incremental evolution, functional and environmental, were used to evolve controllers. The goal is to produce a controller that can locate a radar, navigate the UAV to the source stably and efficiently using on-board sensor measurements, and then circle around the emitter. Controllers were evolved for a variety of radar types. While there has been success in evolving controllers directly on real robots [6], simulation is the only feasible way to evolve controllers for UAVs. A UAV cannot be operated continuously for long enough to evolve a sufficiently competent controller, the use of an unfit controller could result in damage to the aircraft, and flight tests are very expensive. For these reasons, the simulation must be capable of evolving controllers which transfer well to real UAVs. A method that has proved successful in this process is the addition of noise to the simulation [11].

2 Unmanned Aerial Vehicle Simulation

The focus of this research was the development of a navigation controller for a fixed wing UAV. The UAV's mission is to autonomously locate, track, and circle around a radar site. There are several main goals for an evolved controller. First, it 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. Second, once in the vicinity of the source, the UAV should circle as closely as possible around the radar. This goal is especially important for radar jamming, where the distance from the source has a major effect on the necessary jamming power. Third, the flight path should be efficient and stable. The roll angle of UAVs 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 signals as input, and based on this sensory data and past information, the navigation controller changes 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 UAV platforms; the evolved controllers are not designed for any specific UAV airframe or autopilot.

The controller is evolved in simulation. The simulation environment is a square 100 nautical miles (nmi) on each side. The simulator gives the UAV a random initial position in the middle half of the southern edge of the environment with an initial heading of due north and the radar site a random position within the environment every time a simulation is run. In our current research, the UAV has a constant altitude and a constant speed of 80 knots. This is realistic because the speed and altitude are controlled by the autopilot, not the evolved navigation controller.

Our simulation can model a wide variety of radars. Stationary radars were modeled as early warning radars, mobile radars as target acquisition radars [12]. Only the sidelobes of the radar emissions are modeled. The sidelobes are the parts of the emitted signal that are not part of the main beam, so they have a much lower power than the effective portion of the radar signal. If a controller can track a radar based only on the sidelobes, the radar can be tracked no matter the direction in which it is pointed, increasing the robustness of the system. 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 the experiments described in this research, the AoA is accurate to within $\pm 10°$ at each time step, a realistic value for this type of sensor. 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.

While a human could design a controller that could home in on a radar under perfectly ideal conditions, the real-world application for these controllers is far from ideal. While sensors to detect the amplitude and angle of arriving electromagnetic signals can be very accurate, the more accurate the sensor, the larger and more expensive it tends to be. One of the great advantages of UAVs is their low cost, and the feasibility of using UAVs for many applications may also depend on keeping the cost of sensors low. By using evolution to design controllers, cheaper sensors with much lower accuracy can be used without a significant drop in performance. As the accuracy of the sensors decreases and the complexity of the radar signals increases – as the radars emit periodically or move – the problem becomes far more difficult for human designers. Flying a physical UAV using an evolved controller is a future goal of this research, so transference from simulation to a real UAV was taken into account from the beginning. Navigation control was abstracted from UAV flight, simulation parameters were tuned for equivalence to real aircraft, and noise was added to the simulation.

3 Multi-objective Genetic Programming

UAV controllers were designed using multi-objective genetic programming which employs non-dominated sorting, crowding distance assignment to each solution, and elitism. The multi-objective genetic programming algorithm used in this research is very similar to the NSGA-II [10] multi-objective genetic algorithm. The function and terminal sets used in this research were

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

$$T = \{ \textit{HardLeft, HardRight, ShallowLeft, ShallowRight, WingsLevel, NoChange, rand}, 0, 1 \}$$

The UAV has a GPS on-board, and the position of the UAV 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 radar is always placed within the simulation area, but the UAV is free to move outside of it. The two available sensor measurements are the amplitude and the AoA of the incoming radar signal. The slope of the amplitude with respect to time is also available. 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 maintains the current roll angle. 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 [13]. The parameters used by GP are shown in Table 1. Tournament selection was used. Initial trees were randomly generated

Table 1. 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

using ramped half and half initialization. No parsimony pressure methods were used in this work, as code bloat was not a major problem. All computations were done on a Beowulf cluster parallel computer with ninety-two 2.4 GHz Pentium 4 processors. The data communication between master and slave processors was possible using the Message Passing Interface (MPI) standard [14] under the Linux operating system.

3.1 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 random for every run. We designed the four fitness measures to satisfy the goals of the evolved controller: moving toward the emitter, circling the emitter closely, and flying in an efficient and stable manner.

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 $fitness_1$.

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

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

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

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

$$fitness_3 = \frac{1}{T - N} \sum_{i=1}^{T} (1 - in\_range) * level$$

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

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 $fitness_4$.

3.2 Incremental Evolution

Functional incremental evolution incrementally changes the fitness function to increase the difficulty of the problem. Controllers evolved from random initial populations used a form of functional incremental evolution. Controllers evolved for 600 generations, but for the first 200 generations, we used only one of the four fitness functions. Flying to the goal was the most basic behavior for a navigation controller. To place more importance on this behavior, the first 200 generations used only the normalized distance fitness function. The last 400 generations used all four of the fitness functions.

Environmental incremental evolution incrementally increases the difficulty of the environment or task faced by evolution, while leaving the fitness function unchanged. In this research, random populations are initialized and then evolved for 600 generations on continuously emitting, stationary radars to create seed populations. Controllers for more difficult radars are then evolved for 400 generations with all four fitness functions using these seed populations. The controllers in the seed population are not immediately able to solve this new problem well, but since many aspects of the problem are similar, the seed population provides an excellent basis for evolving fit controllers for the new task. The more complex radar types may be evolved over multiple stages of evolution, using progressively more difficult radar types.

Maintaining sufficient diversity in the population is often an issue when using incremental evolution [15]. If the diversity of a population decreases too much during an early stage of evolution, the final evolution might still have a very difficult time producing a good solution. While this was a concern in this research, one of the features of the multi-objective optimization algorithm had potential to counter loss of diversity. Like NSGA-II [10], the algorithm used for this research attempts to spread solutions across the Pareto front by incorporating a crowding distance into fitness evaluation, encouraging diversity in the population.

4 Results

Multi-objective GP produced controllers that satisfied the goals of this problem. In order to statistically measure the performance of GP, we did 50 evolutionary runs for each experiment. Each evolutionary run lasted for 600 generations and produced 500 solutions. Since multi-objective optimization produces a Pareto front of solutions, rather than a single best solution, we needed a method to gauge the performance of evolution. To do this, we selected values we considered minimally successful for the four fitness metrics. We defined a minimally successful UAV controller as able to move quickly to the target radar site, circle at an average distance under 2 nmi, fly with the wings level to the ground for at least 1,000 seconds, and turn sharply less than 0.5% of the total flight time. If a controller had a normalized distance fitness value ($fitness_1$) of less than 0.15, a circling distance ($fitness_2$) of less than 4 (the circling distance fitness metric squares the distance), a level time ($fitness_3$) of greater than 1,000, and a turn cost ($fitness_4$) of less than 0.05, the evolution was considered successful. These baseline values were used only for our analysis, not for the evolutionary process.

Experiments were done for four radar types using direct evolution: 1) continuously emitting, stationary radars, 2) continuously emitting, mobile radars, 3)

Table 2. Direct evolution experimental results

	Runs			Successful controllers		
Radar type	Total	Successful	Percentage	Total	Average	Maximum
Continuous, Stationary	50	45	90%	3,149	63	170
Continuous, Mobile	50	36	72%	2,266	45.3	206
Intermittent, Stationary	50	25	50%	1,891	37.8	156
Intermittent, Mobile	50	16	32%	569	11.38	93

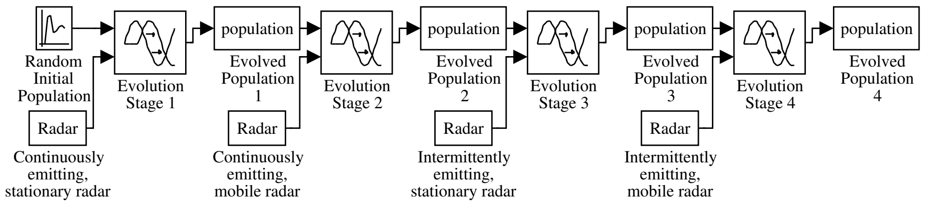

Fig. 1. Incremental evolutionary process

intermittently emitting, stationary radars, and 4) intermittently emitting, mobile radars. The intermittently emitting radars had periods of 10 minutes and emitting durations of 5 minutes. The mobile radars were modeled as finite state machines with *setup*, *deployed*, *tear down*, and *move* states. A radar only emits in the *deployed* state. When a radars moves, the new location can be anywhere in the simulation area. For each of these experiments, a population of 500 individuals was randomly initialized and then evolved for 600 generations. Results from the four experiments are shown in Table 2. Multi-objective GP was able to successfully evolve controllers for all four of these radar types. For both mobile and intermittently emitting radars, the UAV receives sensor information from the radar less than 100% of the time, which increases the difficulty of the problem for evolution. The results show that the continuously emitting, stationary radar proved the easiest for evolution and the intermittently emitting, mobile radar the most difficult. For more detailed results on the direct evolution experiments, refer to [16, 17].

Another set of experiments was performed using environmental incremental evolution to improve the chances of evolving controllers for the more complex radar types. The same four radar types were used, but instead of evolving controllers in four separate experiments, evolved controllers from simpler radar types were used to seed evolutions for the more complex radar types. Figure 1 shows the evolutionary process. In the first stage of evolution, randomly initialized populations were evolved on continuously emitting, stationary radars for 600 generations in the same manner as the direct evolution experiments. In the second stage, these evolved populations were used as seed populations and evolved for 400 generations on continuously emitting, mobile radars. In the third stage, the

Table 3. Incremental evolution experimental results

	Runs			Successful controllers		
Radar type	Total	Successful	Percentage	Total	Average	Maximum
Continuous, Stationary	50	45	90%	2,815	56.30	166
Continuous, Mobile	50	45	90%	2,774	55.48	179
Intermittent, Stationary	50	42	84%	2,083	41.66	143
Intermittent, Mobile	50	37	74%	1,602	32.04	143

evolved populations were evolved on intermittently emitting, stationary radars. Finally, in the fourth stage of evolution, evolution took place on intermittently emitting, mobile radars. Results from each stage of evolution are shown in Table 3. For continuously emitting, stationary radars, the results verify those from the direct evolution experiments. Figure 2 compares the number of successful evolutionary runs for direct and incremental evolution. Figure 3 compares the total number of successful controllers for direct and incremental evolution. For the three more complex radar types, the use of environmental incremental evolution dramatically increased both the total number of successful controllers and the number of evolutionary runs that produced successful controllers. As mentioned in Section 3.2, diversity is often an issue when using incremental evolution. In these experiments, populations tended to remain diverse, possibly because of the use of crowding distance in the multi-objective GP algorithm.

When autonomous navigation controllers are used to fly real UAVs, it is essential to have a single controller that can handle multiple radar types. Based on the information available to the UAV, it is difficult to know what kind of radar the UAV is approaching, and it is far easier to have one robust controller that is used all the time rather than switching between several simpler controllers. The final population for intermittently emitting, mobile radars evolved using incremental evolution produced 1,602 successful controllers. The controllers were evaluated separately on all four radar types, and every controller was successful for each type.

5 Conclusions

Using genetic programming with multi-objective optimization, we were able to evolve navigation controllers for UAVs capable of flying to a target radar, circling the radar site, and maintaining an efficient flight path, all while using inaccurate sensors in a noisy environment. We used methods to aid in the future transference of evolved controllers to real UAVs. Controllers were evolved for four radar types using both direct evolution and incremental evolution. Using incremental evolution dramatically increased the chances of producing successful controllers. Incremental evolution also produced controllers able to handle all four radar types. In the future, we will test the controllers evolved in this research on physical UAVs. Our research will also focus on evolving distributed,

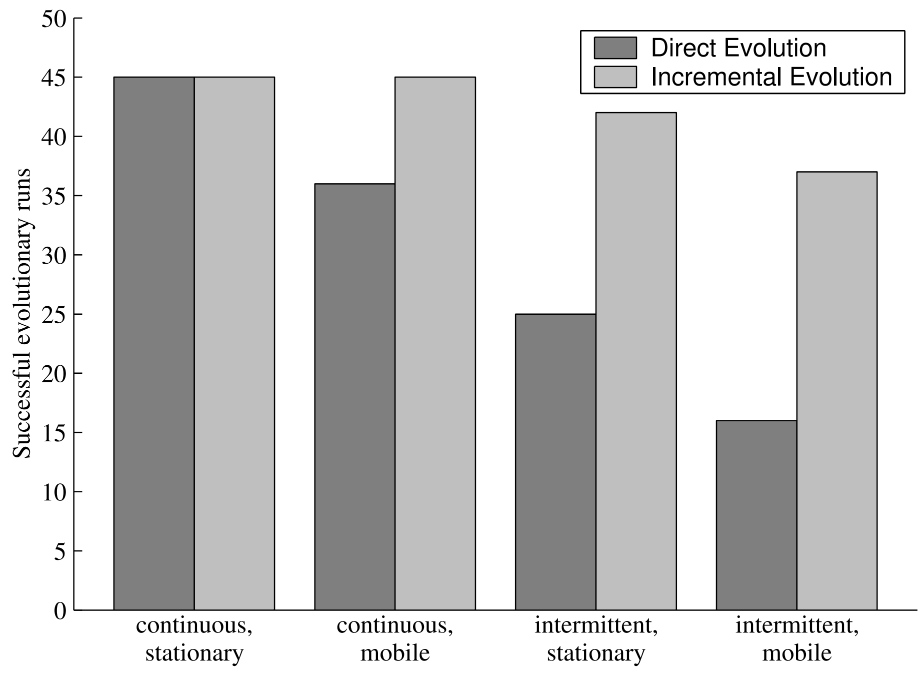

Fig. 2. Number of successful evolutionary runs for each radar type using direct evolution and environmental incremental evolution

multi-agent UAV navigation controllers for responding to multiple radar sites with multiple UAVs.

6 Acknowledgments

The authors would like to acknowledge the financial support provided by the Office of Naval Research with Dr. M. Soto as program manager and the computer resources provided by the Naval Research Laboratory.

References

1. Harvey, I., Husbands, P., Cliff, D.: Seeing the light: Artificial evolution, real vision. In: Proceedings of the Third International Conference on Simulation of Adaptive Behavior. (1994) 704–720
2. Gomez, F., Miikkulainen, R.: Incremental evolution of complex general behavior. Adaptive Behavior **5** (1997) 317–342
3. Lee, W.P., Hallam, J., Lund, H.H.: Applying genetic programming to evolve behavior primitives and arbitrators for mobile robots. In: Proceedings of the IEEE International Conference on Evolutionary Computation. (1997) 495–499

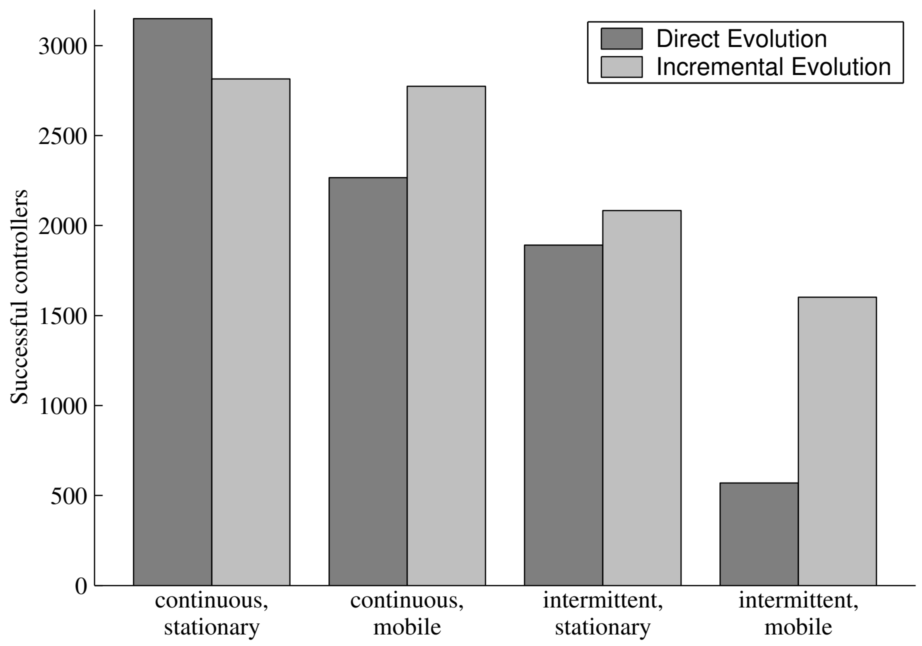

Fig. 3. Number of successful controllers evolved for each radar type using direct evolution and environmental incremental evolution

4. Winkeler, J.F., Manjunath, B.S.: Incremental evolution in genetic programming. In: Genetic Programming 1998: Proceedings of the Third Annual Conference. (1998) 403–411
5. Nelson, A.L., Grant, E., Barlow, G., White, M.: Evolution of complex autonomous robot behaviors using competitive fitness. In: Proceedings of the IEEE International Conf. on Integration of Knowledge Intensive Multi-Agent Systems. (2003)
6. Nolfi, S., Floreano, D.: Evolutionary Robotics. MIT Press (2000)
7. Back, T., Hammel, U., Schwefel, H.P.: Evolutionary computation: Comments on the history and current state. IEEE Transactions on Evolutionary Computation **1** (1997)
8. Coello, C.A.C.: An updated survey of evolutionary multiobjective optimization techniques: State of the art and future trends. In: Proceedings of the Congress on Evolutionary Computation. (1999) 3–13
9. Rodriguez-Vazquez, K., Fonseca, C.M., Fleming, P.J.: Multiobjective genetic programming: A nonlinear system identification application. In: Late Breaking Papers at the 1997 Genetic Programming Conference. (1997) 207–212
10. Deb, K., Agrawal, S., Pratap, A., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation **6** (2002) 182–197
11. Jakobi, N., Husbands, P., Harvey, I.: Noise and the reality gap: The use of simulation in evolutionary robotics. In: Proceedings of the 3rd European Conference on Artificial Life. (1995) 704–720

12. Schleher, D.C.: Introduction to Electronic Warfare. Artech House (1986)
13. Koza, J.: Genetic Programming. MIT Press (1992)
14. Pacheco, P.: Parallel Programming with MPI. Morgan Kaufmann Publishers, Inc. (1996)
15. Eriksson, R.I.: An initial analysis of the ability of learning to maintain diversity during incremental evolution. In: Data Mining with Evolutionary Algorithms. (2000) 120–124
16. Oh, C.K., Barlow, G.J.: Autonomous controller design for unmanned aerial vehicles using multi-objective genetic programming. In: Proceedings of the Congress on Evolutionary Computation. (2004)
17. Barlow, G.J.: Design of autonomous navigation controllers for unmanned aerial vehicles using multi-objective genetic programming. Master's thesis, North Carolina State University, Raleigh, NC (2004)