Autonomous Controller Design for Unmanned Aerial Vehicles using Multi-objective Genetic Programming

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

Gregory J. Barlow
Center for Robotics and Intelligent Machines
Dept. of Electrical and Computer Engineering
North Carolina State University
Raleigh, NC 27695-7911
Email: gjbarlow@ncsu.edu

***Abstract*— Autonomous navigation controllers were developed for fixed wing unmanned aerial vehicle (UAV) applications using 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. Controllers were evolved for three different kinds of radars: stationary, continuously emitting radars, stationary, intermittently emitting radars, and mobile, continuously emitting radars. We selected realistic flight parameters and sensor inputs to aid in the transference of evolved controllers to physical UAVs.**

I. Introduction

The field of evolutionary robotics (ER) [1] combines research on behavior-based robot controller design with evolutionary computation. A major focus of ER is the automatic design of behavioral controllers with no internal environmental model, in which effector outputs are a direct function of sensor inputs [2]. ER 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 behaviors, such as obstacle avoidance [3], light seeking [4], object movement [5], simple navigation [6], and game playing [7], [8]. In many of these cases, the problems to be solved were designed specifically for research purposes. While simple problems generally require a small number of behaviors, more complex real-world problems might require the coordination of multiple behaviors in order to achieve the goals of the problem. Very little of the ER work to date has been intended for use in real-life applications.

Early in ER research, Brooks noted that the evolution of robot controllers would probably need to occur in simulation [9]. While some controllers 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, because evaluations are usually 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.

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 [3], [4], [7], [8], [10], [11], GP has also been shown to produce functional behaviors for autonomous robot control [5], [6].

One of the main difficulties of ER is the formulation of fitness functions [12]. 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 proof of concept research, the problem to be solved has often been adapted in ways that made the formulation of these fitness metrics easier, such as the simplification of the environment [7]. While co-evolution and competitive fitness metrics have been used to generalize fitness function formulation, these methods usually require changing the problem to fit the competitive fitness model [8], [13]. For problems without a single, easily quantifiable objective, an alternative that has attracted a great deal of research in the last several years is multi-objective optimization, which allows the evolutionary algorithm to optimize on multiple fitness metrics [14]–[16].

A majority of the research in ER has focused on wheeled mobile robot platforms [3]–[8], [10], [17], especially the Khepera robot [3]–[5], [17]. Research on walking robots [10] and other specialized robots [11] has also been pursued. An application of ER that has received very little attention is unmanned aerial vehicles (UAVs). The UAV is becoming increasingly popular for many applications, particularly where high risk or accessibility are issues.

Many problems have multiple objectives, but conventional GP uses only a single scalar fitness function. For problems with multiple goals, the objectives must be combined into a single function using weighting [5]. An alternative is multi-objective GP, where evolution optimizes over multiple objectives [16]. Weighting of the different objectives is not necessary for multi-objective optimization because it simultaneously satisfies multiple functions without requiring scaling factors between the objectives. Since this technique produces multiple fitness values for each individual, a non-dominated sort is used to determine the relative rank of individuals in the population [14]. 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 [15]. It is up to the designer to choose a solution from this set.

In this paper, we present our approach to evolving behavioral navigation controllers for fixed wing UAVs using multi-objective GP. The goal is to produce a controller that can locate an electromagnetic energy source, navigate the UAV to the source efficiently using sensor measurements, and circle closely around the emitter, which is a radar in our simulation. Controllers were evolved for three different kinds of radars: stationary, continuously emitting radars, stationary, intermittently emitting radars, and mobile, continuously emitting radars. Multi-objective optimization and GP were used to satisfy the objectives. While there has been success in evolving controllers directly on real robots [3], 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 [17].

After describing the problem and the simulation environment, we outline the multi-objective GP algorithm, the GP parameters, and the four fitness measures. We present simulation results for evolved controllers and discuss transference to a real UAV.

II. 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 then orbit around a radar site. There are three 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, whether that is jamming the radar, surveillance, or another of the many applications of this type of controller. 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. 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 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 assumption 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 radar types. For the research presented in this paper, we modeled three types of radars: 1) stationary, continuously emitting radars, 2) stationary, intermittently emitting radars with a period of 10 minutes and duration of 5 minutes, and 3) mobile, continuously emitting radars. 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 the experiments described in this research, the AoA is accurate to within $\pm 10^\circ$ at each time step, a realistic value for this type of sensor. This means that the radar can be anywhere inside a 20° cone emanating from the UAV. 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 easily 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. In this research, we are interested in evolving controllers for these difficult, real-world problems.

III. 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 [14] multi-objective genetic algorithm. 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 are defined as

F = { Prog2, Prog3, IfThen, IfThenElse, And, Or, Not, ¡,≤, ¿, ≥, ¡ 0 , ¿ 0, =, +, -, *, ÷, X ¡ 0, Y ¡ 0, X ¿ max, Y ¿ max, Amplitude ¿ 0, AmplitudeSlope ¿ 0, AmplitudeSlope ¡ 0, AoA ¿ 0, AoA ¡ 0 }

T = { 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 UAV is free to move outside of the area during the simulation, but the radar is always placed within it. The two available sensor measurements are the amplitude of the incoming radar 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.

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. Fitness Functions

Four fitness functions determine the success of individual UAV navigation controllers. The fitness of a controller was measured over 30 simulation runs, 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

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

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 $fitness_2$.

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

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

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

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.

Applying the term multi-objective optimization to this evolutionary process is a misnomer, because this research was concerned with the generation of behaviors, not optimization. In the same way that a traditional genetic algorithm can be used for both optimization and generation, so can multi-objective optimization. Even though the controller doesn't generate the most optimized controllers possible, it can obtain near-optimal solutions.

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. Maintaining sufficient diversity in the population is often an issue when using incremental evolution [21], but did not appear to be a problem here.

V. Results

Multi-objective GP produced controllers that satisfied the three goals of this problem. In order to statistically measure the performance of GP, we did 50 evolutionary runs for each type of radar. Each 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. To select a single controller from these successful individuals, increasingly optimal fitness values were chosen until only a single controller met the criteria. Controllers were evolved for stationary, continuously emitting radars, stationary, intermittently emitting, radars, and mobile, continuously emitting radars. The results of our experiments are shown in Table II.

The first experiment evolved controllers on a stationary, continuously emitting radar. Of the 50 evolutionary runs, 45 runs were acceptable under our baseline values. The number of acceptable controllers evolved during an individual run ranged from 1 to 170. Overall, 3,149 acceptable controllers

TABLE II

EXPERIMENTAL RESULTS FOR THREE RADAR TYPES.

	Runs		Successful controllers		
Radar type	Total	Successful	Total	Average	Maximum
Continuous	50	45	3149	63	170
Intermittent	50	25	1891	37.8	156
Mobile	50	36	2266	45.3	206

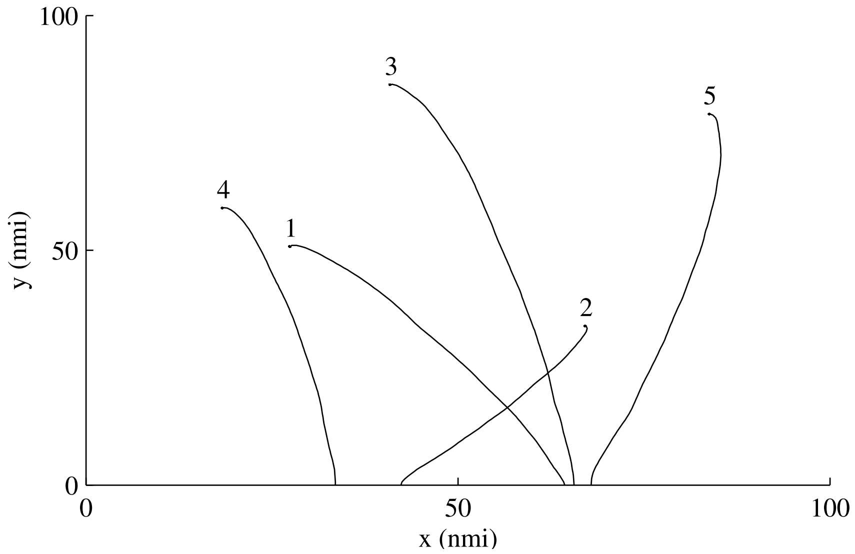

Fig. 1. Five UAV flight paths to continuously emitting, stationary radars.

TABLE III

FITNESS VALUES FOR FIVE UAV FLIGHT PATHS TO CONTINUOUSLY EMITTING, STATIONARY RADARS.

Flight	Normalized Distance	Circling Distance	Level Time	Turn Cost
1	0.067	1.299	2,346	0.014
2	0.044	1.189	1,384	0.007
3	0.094	1.440	3,531	0.023
4	0.064	1.291	2,245	0.014
5	0.085	1.383	3,122	0.008
Baseline	0.15	4	1,000	0.05

were evolved, for an average of 63 successful controllers per evolutionary run. Figure 1 shows five sample flight paths to five different emitter locations for an evolved controller. This controller has a complexity of 162 nodes, too large a tree to show here. The fitness values for each simulated flight are shown in Table III. This evolved controller flies to the target very efficiently, staying level a majority of the time. Almost all turns are shallow. Once in range of the target, the roll angle is gradually increased. Once the roll angle reaches its maximum value to minimize the circling radius, no change to the roll angle is made for the remainder of the simulation. Populations tended to evolve to favor turning left or right.

The second experiment evolved controllers for a stationary, intermittently emitting radar. The radar was set to emit for 5 minutes and then turned off for 5 minutes, giving a period of 10 minutes and a 50% duty cycle. The only change from the first experiment was the radar configuration. However, this experiment was far more difficult for evolution than the first experiment, because the radar only emits half of the time in this experiment. A new set of 50 evolutionary runs were done, and 25 of the runs produced at least one acceptable solution. The number of controllers in an evolutionary run that met the baseline values ranged from 1 to 156, a total of 1,891 successful controllers were evolved, and the average number of acceptable controllers evolved during each run was 37.8. Figure 2 shows five sample flight paths to five different emitter locations for an evolved controller. The fitness values for each simulated flight in Figure 2 are shown in Table IV. The flight paths for the controllers evolved on intermittently emitting radars were similar to those evolved on continuously emitting radars. In some cases, controllers evolved a waiting behavior, where near the beginning of flight,

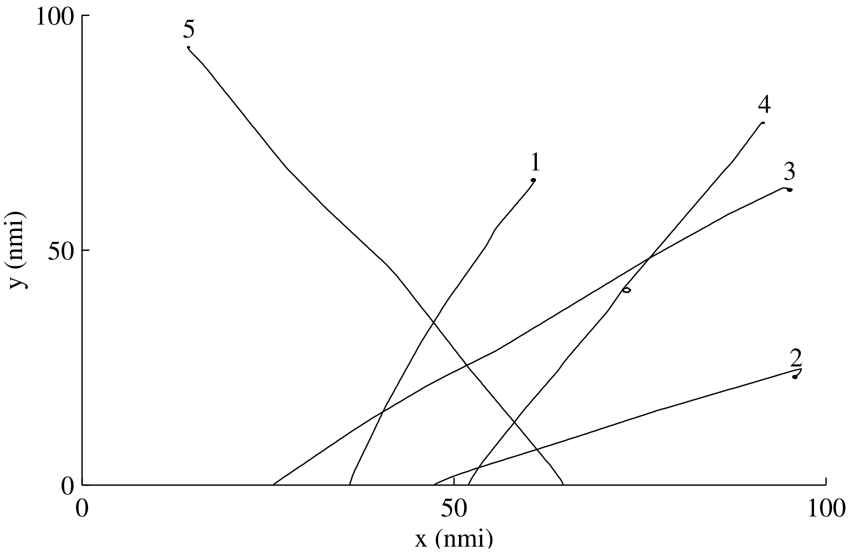

Fig. 2. Five UAV flight paths to intermittently emitting, stationary radars. Radars were set to emit for 5 minutes and then turned off for 5 minutes, giving a period of 10 minutes and a 50% duty cycle.

TABLE IV
FITNESS VALUES FOR FIVE UAV FLIGHT PATHS TO INTERMITTENTLY EMITTING, STATIONARY RADARS.

Flight	Normalized Distance	Circling Distance	Level Time	Turn Cost
1	0.072	1.363	2,657	0.023
2	0.056	1.332	1,957	0.031
3	0.099	1.505	3,748	0.043
4	0.095	1.422	3,426	0.014
5	0.111	1.505	4,286	0.028
Baseline	0.15	4	1,000	0.05

the UAV would circle during the period when the radar was not emitting. This happened infrequently for the best controllers. Also, sometimes the UAV would overshoot its target if the radar was not emitting when the UAV arrived. Once the UAV began circling, controllers were able to circle regardless of whether the radar was emitting or not. Despite the increased complexity from the first experiment, GP was able to evolve many successful controllers.

The third experiment evolved controllers for a mobile, continuously emitting radar. The mobility was modeled as a finite state machine with the following states: *move*, *setup*, *deployed*, and *tear down*. When the radar moves, the new location is random anywhere in the simulation area. The finite state machine is repeated for the duration of simulation. The radar site only emits when it is in the *deployed* state; while the radar is in the other states, the UAV receives no sensory information. The time in each state is probabilistic, and the minimum amount of time spent in the deployed state is an hour or 25% of the simulation time. The simulation is identical to the first two experiments other than the configuration of the radar site. Of the 50 evolutionary runs, 36 were acceptable under our baseline values. The number of acceptable controllers evolved in each run ranged from 1 to 206. A total of 2266 successful controllers were evolved for an average of 45.3 acceptable controllers per evolutionary run. While not as difficult for evolution as the second experiment, the mobile radar was more challenging than the stationary radar. Figure 3 shows two sample flight paths to two different mobile radars for an evolved controller. The fitness values for each simulated flight are shown in Table V.

To test the effectiveness of each of the four fitness measures, we ran evolutions with various subsets of the fitness metrics. These tests were done using the stationary, continuously emitting radar, the simplest of the three radar types presented above. The first fitness measure, the normalized distance, was included in every subset. When only $fitness_1$ was used to measure controller fitness, flight paths were very direct. The UAV flew to the target in what appeared to be a straight line. To achieve this direct route to the target, the controller would use sharp and alternating turns. The UAV would almost never fly level to the ground, and all turns were over $10°$. Circling was also not consistent; the controllers frequently changed direction while within the in-range boundary of the radar, rather than orbiting in a circle around the target. For this simplest of fitness measures, evolution tended to select very simple bang-bang control, changing the roll angle at every time step using sharp right and left turns, with the single goal of minimizing the AoA. In a comparison, evolved controllers

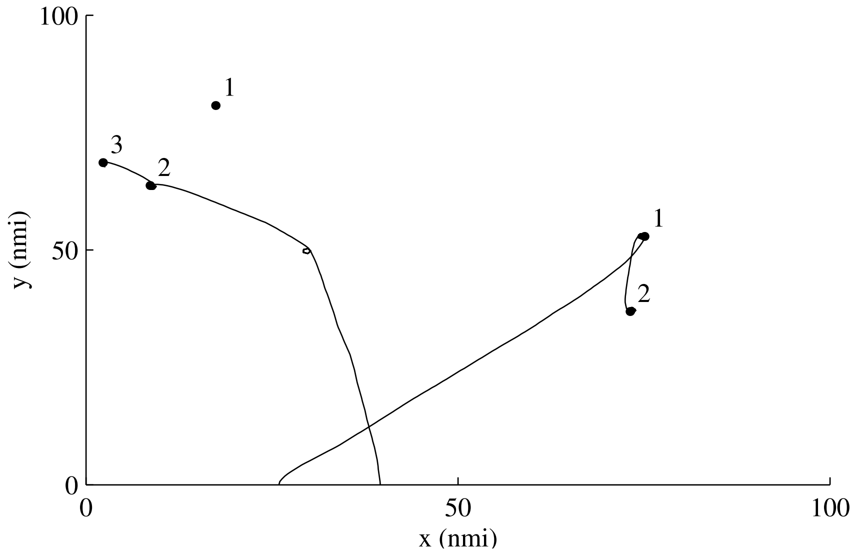

Fig. 3. Two UAV flight paths for continuously emitting, mobile radars. Numbers indicate deployed radar positions.

TABLE V
FITNESS VALUES FOR TWO UAV FLIGHT PATHS TO CONTINUOUSLY EMITTING, MOBILE RADARS.

Flight	Normalized Distance	Circling Distance	Level Time	Turn Cost
1	0.091	4.279	2,929	0
2	0.077	3.079	2,997	0
Baseline	0.15	4	1,000	0.05

exhibited slightly better performance than a human-designed, rule-based controller. Further comparisons were not made, because the human-designed controller performance degraded rapidly as additional fitness measures and radar types were considered.

Using only two fitness measures was not sufficient to achieve the desired behaviors. If $fitness_1$ and $fitness_2$ were used, the circling behavior improved, but the efficiency of the flight path was unchanged. If $fitness_1$ and $fitness_4$ were used, turns were shallower, but the UAV still failed to fly with its wings level to the ground for long periods. Circling around the target also became more erratic and the size of the orbits increased. If $fitness_1$ and $fitness_3$ were used, the UAV would fly level a large amount of the time, but circling was very poor, with larger radius orbits or erratic behavior close to the target. Sharp turns were also very common.

If three of the fitness measures were used, evolved behavior was improved, but not enough to satisfy the mission goals. If all fitness measures were used except $fitness_2$, the UAV would fly efficiently to the target, staying level and using only shallow turns. Once in range of the radar, circling was generally poor. Evolved controllers either displayed large, circular orbits or very erratic behavior that was unable to keep the UAV close to the radar. If $fitness_1$, $fitness_2$, and $fitness_4$ were used, the UAV would circle well once it flew in range of the radar. While flying toward the radar, the UAV failed to fly level, though turns tended to be shallow. The best combination of three fitness measures was when only $fitness_4$ was removed. In this case, circling was good and the UAV tended to fly straight to the target. The level time fitness measure also tended to keep the turns shallow and to eliminate alternating between right and left turns. However, turn cost was still high, as many turns were sharp.

When we used all four of the fitness functions, the evolved controllers were sufficiently robust. A variety of strategies were evolved to accomplish the mission goals for each of the three experiments, and many controllers were sufficiently fit to be considered successful. The evolved controllers were able to overcome a noisy environment and inaccurate sensor data in tracking and orbiting a radar site. In short, the use of four objectives with GP was successful. The four fitness measures selected all had an impact on the behavior of the evolved controllers, and all four were necessary to achieve the desired flight characteristics.

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, 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.

VI. 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. Controllers were evolved for three different radar types. First, navigation controllers were evolved for stationary, continuously emitting radars, and then two other experiments added difficulties to this simplest radar case. Intermittently emitting and mobile radars were used in the second and third experiments. The four fitness functions used for this research were sufficient to produce the desired behaviors, and all four measures were necessary for all three cases. We used methods to aid in the transference of the evolved controllers to real UAVs. In the next stage of this research, we will test the controllers evolved in this research on physical UAVs.

In the near term, future research will focus on evolving UAV navigation controllers capable of responding to targets even more elusive than the radar types described here, including intermittently emitting mobile targets and multiple targets. Long term goals are the development and demonstration of general agent architectures that will support autonomous, adaptive, and cooperative unmanned vehicle activities.

References

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

[2] D. Keymeulen, M. Iwata, K. Konaka, R. Suzuki, Y. Kuniyoshi, and T. Higuchi, "Off-life model-free and on-line model-based evolution for tracking navigation using evolvable hardware," in *Proceedings of the First European Workshop on Evolutionary Robotics*, (Paris), April 1998.

[3] S. Nolfi, D. Floreano, O. Miglino, and F. Mondada, "How to evolve autonomous robots: Different approaches in evolutionary robotics," in *Proceedings of the IV International Workshop on Artificial Life* (R. A. Brooks and P. Maes, eds.), (Cambridge, MA), MIT Press, July 1994.

[4] H. H. Lund and J. Hallam, "Evolving sufficient robot controllers," in *Proceedings of the IEEE International Conference on Evolutionary Computation*, pp. 495–499, 1997.

[5] 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.

[6] 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.

[7] A. L. Nelson, *Competitive Relative Performance and Fitness Selection for Evolutionary Robotics*. PhD thesis, North Carolina State University, 2003.

[8] A. L. Nelson, E. Grant, G. Barlow, and M. White, "Evolution of complex autonomous robot behaviors using competitive fitness," in *Proceedings of the IEEE International Conference on Integration of Knowledge Intensive Multi-Agent Systems*, (Boston, MA), September 2003.

[9] R. A. Brooks, "Artificial life and real robots," in *Toward a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life*, (Cambridge, MA), pp. 3–10, MIT Press, 1992.

[10] 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.), Oita University Press, 1999.

[11] I. Harvey, P. Husbands, and D. Cliff, "Seeing the light: Artificial evolution, real vision," in *Proceedings of the Third International Conference on Simulation of Adaptive Behavior*, pp. 704–720, MIT Press, 1994.

[12] T. Back, U. Hammel, and H.-P. Schwefel, "Evolutionary computation: Comments on the history and current state," *IEEE Transactions on Evolutionary Computation*, vol. 1, April 1997.

[13] L. Panait and S. Luke, "Methods for evolving robust programs," in *GECCO* (E. C.-P. et al., ed.), (Chicago), pp. 1715–1728, July 2003.

[14] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan, "A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: Nsga-II," in *Proceedings of the Parallel Problem Solving from Nature VI Conference*, (Paris, France), pp. 849–858, 2000.

[15] C. A. C. Coello, "An updated survey of evolutionary multiobjective optimization techniques: State of the art and future trends," in *Proceedings of the Congress on Evolutionary Computation*, pp. 3–13, 1999.

[16] K. Rodriguez-Vazquez, C. M. Fonseca, and P. J. Fleming, "Multiobjective genetic programming: A nonlinear system identification application," in *Late Breaking Papers at the 1997 Genetic Programming Conference*, pp. 207–212, 1997.

[17] 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.

[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.

[21] R. I. Eriksson, "An initial analysis of the ability of learning to maintain diversity during incremental evolution," in *Data Mining with Evolutionary Algorithms* (A. A. Freitas, ed.), pp. 120–124, 2000.