Evolved Navigation Control for Unmanned Aerial Vehicles

Gregory J. Barlow[1] and Choong K. Oh[2]
[1]*Robotics Institute, Carnegie Mellon University*
[2]*United States Naval Research Laboratory*
United States

1. Introduction

Whether evolutionary robotics (ER) controllers evolve in simulation or on real robots, real-world performance is the true test of an evolved controller. Controllers must overcome the noise inherent in real environments to operate robots efficiently and safely. To prevent a poorly performing controller from damaging a vehicle—susceptible vehicles include statically unstable walking robots, flying vehicles, and underwater vehicles—it is necessary to test evolved controllers extensively in simulation before transferring them to real robots. In this paper, we present our approach to evolving behavioral navigation controllers for fixed wing unmanned aerial vehicles (UAVs) using multi-objective genetic programming (GP), choosing the most robust evolved controller, and assuring controller performance prior to real flight tests.

2. Background

ER (Nolfi & Floreano, 2000) combines 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 (Keymeulen et al., 1998). ER uses a population-based evolutionary algorithm to evolve autonomous robot controllers for a target task. Most of the controllers evolved in ER research have been developed for simple behaviors, such as obstacle avoidance (Nolfi et al., 1994), light seeking (Lund & Hallam, 1997), object movement (Lee & Hallam, 1999), simple navigation (Ebner, 1998), and game playing (Nelson, 2003; Nelson et al., 2003). 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 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, especially the Khepera robot. Research on walking robots (Filliat et al., 1999) and other specialized robots (Harvey et al., 1994) has also been pursued. An application of ER that has received very little attention is UAVs. The UAV has become popular for many applications, particularly where high risk or accessibility is concerns. Although some ER research has been done on UAVs, this work has largely ignored the fixed wing UAV—by far the most common type—until recently. An autopilot for a rotary wing helicopter was evolved using evolutionary strategies (Hoffman et al., 1998) and compared to linear robust multi-variable control and nonlinear tracking control in simulation (Shim et al., 1998). In other work, higher level controllers were evolved with UAVs as the target platform (Marin et al., 1999), but experiments were done only in simulation, movement was grid-based, and the UAV could move in any direction at every time step. Because of the unrealistic nature of the simulation, it would have been difficult to control real UAVs with the evolved controllers. Related work was done to evolve a distributed control scheme for multiple micro air vehicles (Wu et al., 1999). Only simulation was used, the simulation environment was unrealistic, and no testing on real UAVs was attempted. A neural network control system for a simulated blimp has also been evolved (Meyer et al., 2003) with the goal of developing controllers capable of countering wind to maintain a constant flying speed. The evolved control system was only tested in simulation. Only recently has there been work on evolving GP controllers for fixed wing UAVs (Oh et al., 2004; Barlow, 2004; Oh & Barlow, 2004; Barlow et al., 2004; Barlow et al., 2005; Richards et al., 2005; Barlow & Oh, 2006).

In evolutionary computation, incremental evolution (Harvey et al., 1994) 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 in ER have been evolved using incremental evolution. There are two types of incremental evolution. Functional incremental evolution (Lee & Hallam, 1999; Gomez & Miikkulainen, 1997; Winkeler & Manjunath, 1998) changes the difficulty of the fitness function in order to increase the difficulty of the problem. Environmental incremental evolution (Harvey et al., 1994; Nelson, 2003; Nelson et al., 2003) changes the environment to increase difficulty without changing the fitness function.

Transference of controllers evolved in simulation to real vehicles is an important issue in ER. Some controllers have been evolved in situ on physical robots (Walker et al., 2003), but long evaluation time, the need for many evaluations to achieve good results, and the need for human monitoring during the evolutionary process all limit this approach. Alternatively, controllers evolved in simulation do not always transfer well to real vehicles, since the simulation is never a perfect model of the real environment. Adding noise to the simulation (in the form of both sensor error and state error) may help controllers transfer well from simulation to real robots (Jakobi et al., 1995; Gomez and Miikkulainen, 2004; Barlow et al., 2005). This approach is usually evaluated by evolving a controller in a noisy simulation environment and then testing the controller on a real vehicle. This works well for systems where tests can be performed easily, cheaply, and with little danger of damaging the vehicle, but what of systems where tests are expensive or dangerous? Controllers may be evolved with high levels of noise, but this does not guarantee good performance when that noise is not consistent with the real system. Experiments by Jakobi et al. (Jakobi et al., 1995) show that if the noise levels used in simulation are significantly different from those in the real world, there are no assurances that the evolved controller will perform as desired. If, however, a controller performs well when subjected to a wide range of sensor and state noise conditions in simulation, and the real environmental noise falls within the testing range, prior works suggest that the controller should also perform well on a real vehicle.

UAVs are one type of robot that requires assurance of the off-design performance (the performance under additional sensor and state noise) of an evolved controller before testing a controller on the robot. Even when subject to additional sources of noise, controllers should still be able to efficiently accomplish the desired task. Assurance of off-design performance is also necessary because poorly performing controllers could cause crashes, possibly destroying the UAV.

3. UAV Navigation Control

The focus of this research was the development of a navigation controller for a fixed wing UAV able to autonomously locate, track, and then circle around a radar site. There are three main goals for an evolved controller. First, the UAV should move to the target as quickly as possible. The sooner the UAV arrives in the vicinity of the target, the sooner it can begin its primary mission: surveillance, radar jamming, or one of the many other 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 necessary jamming power is directly proportional to the square of the distance to the radar. Third, the flight path should be efficient and stable. 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 or reduce the flying time and range of the UAV.

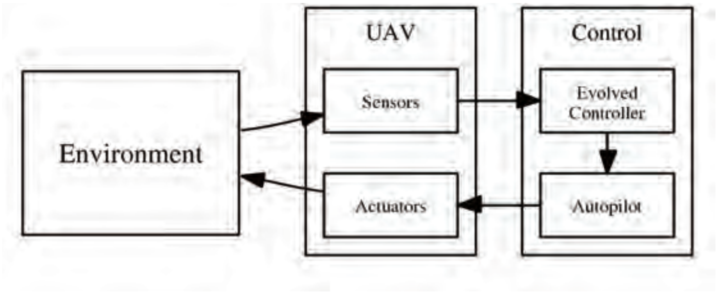

Figure 1. General UAV control diagram

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 updates the desired roll angle of the UAV control surface. The autopilot then uses this desired roll angle to change the heading of the UAV. A diagram of the control process is shown in Figure 1. 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.

3.1 Simulation

While there has been success in evolving controllers directly on real robots, 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 (Jakobi et al., 1995).

The simulation environment is a square, 100 nautical miles (nmi) on each side. Every time a simulation is run, 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. The radar site is also given a random position within the environment. In our current research, the UAV has a constant altitude of 3000 feet and speed of 80 knots. We can realistically assume constant speed and altitude because these variables are controlled by the autopilot, not the evolved navigation controller.

Our simulation can model a wide variety of radar types. The site type, emitter function, frequency, gain, noise, power, pulse compression gain, bandwidth, minimum emitting period, mean emitting period, minimum emitting duration, and mean emitting duration of the radar are all configurable in the simulation. For the purposes of this research, most of these parameters were held constant. Radars used in experiments are described based on two characteristics: emitting pattern and mobility. We modeled five types of radars: continuously emitting, stationary radars; continuously emitting, mobile radars; intermittently emitting, stationary radars with regular emitting periods; intermittently emitting, stationary radars with irregular emitting periods; and intermittently emitting, mobile radars with regular emitting periods.

Radars can emit continuously, intermittently with a regular period, or intermittently with an irregular period. The emitting characteristics of the radar are configured by setting the minimum emitting period, mean emitting period, minimum emitting duration, and mean emitting duration. If all four parameters are set to infinity, the radar is continuous. If the minimum and mean are the same for both period and duration, then the radar is considered to be emitting with a regular period. If the minimum and mean are different, the radar emits with an irregular period: at the start of each period, the lengths of the period and duration of emission are set randomly.

Radars can be either stationary or mobile. A stationary site has a fixed position for the entire simulation period. A mobile site is modeled by a finite state machine with the following states: *move*, *setup*, *deployed*, and *tear down*. When the radar moves, the new location is random, and can be anywhere in the simulation area. The finite state machine is executed for the duration of simulation. The radar site only emits when it is in the *deployed* state; while the radar is in the *move* state it does not emit, so the UAV receives no sensory information. The time in each state is probabilistic, and once the radar enters the deployed state, it must remain in that state for at least an hour.

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 ±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, a realistic value for a real UAV autopilot. The interval between these requests to the autopilot can be adjusted in the simulation.

3.2 Transference

Transference of evolved controllers to a real UAV is an important issue, so we designed several aspects of the simulation to aid in this process. First, we abstracted the navigation control 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 was designed so parameters could be tuned for equivalence to real aircraft and radars. 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 to radar emissions and to sensor accuracy. A noisy simulation environment encourages the evolution of robust controllers that are more applicable to real UAVs.

3.3 Problem Difficulty

The major difficulty of this problem is noise. Under ideal conditions, where the exact angle and amplitude of the incoming signals are known, a human could easily design a fit controller. Real-world conditions, however, are far from ideal. Even the best radar sensors have some error in determining the angle and amplitude of a radar. Environmental conditions, multipath, system noise, clutter, and many other factors increase the sensor noise. As this noise increases, the difficulty of maintaining a stable and efficient flight path increases.

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.

Another difficulty of designing controllers by hand is accounting for the more complex radar types. 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 as the best control strategies become less apparent. In this research, we are interested in evolving controllers for these difficult, real-world problems using many radar types where sensors are very noisy.

3.4 Fitness Functions

We designed four fitness functions to measure the success of individual UAV navigation controllers. The fitness of a controller was measured over 30 simulation runs, where the initial positions of the UAV and the radar were different for every run. We designed the four fitness measures to satisfy the three goals of the evolved controller: rapid movement toward the emitter, circling the emitter, and flying in a stable and efficient way.

3.4.1 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

$$fitness_1 = \frac{1}{T}\sum_{i=1}^{T}\left[\frac{d_i}{d_0}\right]^2 \tag{1}$$

where T is the total number of time steps, d_0 is the initial distance, and d_i is the distance at time i. We are trying to minimize $fitness_1$.

3.4.2 Circling distance

The secondary goal of the UAV is to circle closely around the source, since most applications of this type of controller require proximity to the target; when the UAV is within range of the target, it should circle 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. The distance is squared to apply pressure to GP to evolve very small circling distances. 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

$$fitness_2 = \frac{1}{N}\sum_{i=1}^{T} inrange \cdot d_i^2 \tag{2}$$

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 $fitness_2$.

3.4.3 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 the flight time necessary 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 level time, the amount of time the UAV spends with a roll angle of zero degrees, which is the most stable flight position for a UAV. This fitness metric only applies when the UAV is outside the in-range distance; once the UAV is in range, we want it to circle around the radar, requiring a non-zero roll angle. The level time is given as

$$fitness_3 = \sum_{i=1}^{T}(1 - inrange) \cdot level \tag{3}$$

where *level* is 1 when the UAV has been level for two consecutive time steps and 0 otherwise. We are trying to maximize *fitness*$_3$.

3.4.4 Turn cost

The second fitness measure intended to produce an efficient flight path is a measure of turn cost. While UAVs are capable of quick, sharp turns, it is preferable to avoid them in favor of more gradual turns. The turn cost fitness measure is intended to penalize controllers that navigate using a large number of sharp, sudden turns because this behavior may cause 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

$$fitness_4 = \frac{1}{T}\sum_{i=1}^{T} hardturn \cdot \left|\varphi_i - \varphi_{i-1}\right| \qquad (4)$$

where φ is the roll angle of the UAV and *hardturn* 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*$_4$.

3.5 Genetic Programming

We designed the four fitness functions to evolve particular behaviors, but the optimization of any one function could conflict heavily with the performance of the others. 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 slight 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 methods. Though this process isn't concerned with generating the most optimized controllers possible, it can obtain near-optimal solutions. In this research, we evolved UAV controllers using an implementation of NSGA-II (Deb et al., 2002) for GP. The multi-objective genetic algorithm employs non-dominated sorting, crowding distance assignment to each solution, and elitism.

The function and terminal sets used in this work combine a set of very common functions used in GP experiments with a set of 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>Arg, AoA<Arg }*

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 this 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. 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 ten degree change in the roll angle and shallow turns a two degree 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 forty-five degrees. Each of the six terminals returns the current roll angle.

GP was generational, with crossover and mutation similar to those outlined by Koza (Koza, 1992). The parameters used by GP are shown in Table 1. Tournament selection was used. Initial trees were randomly generated using ramped half and half initialization.

Population size	500
Tournament size	2
Simulation runs per evaluation	30
Maximum initial GP tree depth	5
Maximum GP tree depth	21
Crossover rate	0.9
Mutation rate	0.05

Table 1. GP parameters

In GP, evaluating the fitness of the individuals within a population takes significant computational time. The evaluation of each individual requires multiple trials, 30 trials per evaluation in this research. During each trial, the UAV and the radar are placed randomly and four hours of flight time are simulated. Evaluating an entire population of 500 individuals for a single generation requires 15,000 trials. Therefore, using massively parallel computational processors to parallelize these evaluations is advantageous. In this research, the master-slave model of parallel processing was used. The data communication between master and slave processors was done using the Message Passing Interface (MPI) standard under the Linux operating system. The master node ran the GP algorithm and did all computations related to selection, crossover, and mutation. Evaluations of individuals in the population were sent to slave nodes. The parallel computer used for the experiments was a Beowulf cluster made up of 46 computers running Linux. Each computer had two 2.4 GHz Pentium 4 processors with hyper-threading, for a total of 92 processors in the cluster. Hyper-threading provides a small performance gain for multiple simultaneous processes, so two slave nodes were run on each processor, for a total of 184 slave nodes spread over the 92 processors in the cluster.

4. Evolution Experiments

We used multi-objective GP to evolve autonomous navigation controllers for UAVs. Controllers were evolved on radar types of varying difficulties. We evolved controllers using subsets of the four fitness functions in order to evaluate the effect of each fitness measure on controller behavior. In order to gauge the performance of evolution for multiple objectives, we devised test functions to measure the performance of a controller on the task. We then evolved controllers using both direct evolution and incremental evolution for five radar types: continuously emitting, stationary radars; continuously emitting, mobile radars; intermittently emitting, stationary radars with regular emitting periods; intermittently emitting, stationary radars with irregular emitting periods; and intermittently emitting, mobile radars with regular emitting periods. In order to statistically measure the performance of GP on this problem, we did 50 evolutionary runs for each type of radar, where each run produced 500 controllers.

4.1 Effectiveness of Fitness Functions

To test the effectiveness of each of the four fitness measures, we evolved controllers using various subsets of the fitness metrics. These tests were done using the stationary, continuously emitting radar: this was the simplest of the radar types used for this research. The first fitness measure, the normalized distance, was included in every subset. The primary goal of the UAV is to fly from its initial position to the radar site as quickly as possible; $\textit{fitness}_1$ is the only one of the four fitness functions that measures this behavior. When only $\textit{fitness}_1$ 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 type control, changing the roll angle at every time step using sharp right and left turns..

Using only two fitness measures was not sufficient to achieve the desired behaviors. If $\textit{fitness}_1$ and $\textit{fitness}_2$ (circling distance) were used, the circling behavior improved, but the efficiency of the flight path was unchanged. If $\textit{fitness}_1$ and $\textit{fitness}_4$ (turn cost) 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 $\textit{fitness}_1$ and $\textit{fitness}_3$ (level time) 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 $\textit{fitness}_2$, 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 $\textit{fitness}_1$, $\textit{fitness}_2$, and $\textit{fitness}_4$ 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 $\textit{fitness}_4$ 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 able to overcome a noisy environment and inaccurate sensor data in tracking and orbiting a radar site. A variety of navigation strategies were evolved to satisfy these fitness functions. All four fitness measures had an impact on the behavior of the evolved controllers, and all four were necessary to achieve the desired flight characteristics.

4.2 Evolution

In this research, we used both direct and incremental evolution. In direct evolution, controllers were evolved from random initial populations for continuously emitting, stationary radars; continuously emitting, mobile radars; intermittently emitting, stationary radars with regular periods; intermittently emitting, stationary radars with irregular periods; and intermittently emitting, mobile radars with regular periods. For each experiment, we performed 50 evolutionary runs and then selected successful controllers.

While all four objectives are important, moving the UAV to the goal is the highest priority. To emphasize this objective, controllers evolved directly from random initial populations used functional incremental evolution, which incrementally changes the fitness function to increase the difficulty of the problem. Only the normalized distance fitness measure was used for the first 200 generations; the last 400 generations used all four of the fitness functions.

To improve the chances of successfully evolving acceptable controllers for the more complex radar types, we used environmental incremental evolution. Unlike the direct evolution experiments, which always started with a random initial population, these experiments used evolved populations from simpler radar types as seed populations. 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 using these seed populations. Populations were incrementally evolved on progressively more difficult radar types: continuously emitting, mobile radars; intermittently emitting, stationary radars; and intermittently emitting, mobile radars. Figure 2 graphically summarizes the process of incremental evolution used in this work.

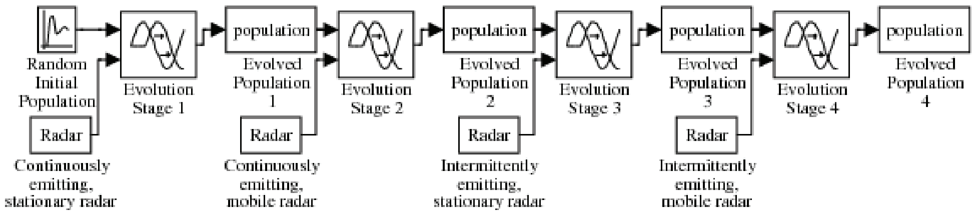

Figure 2. Environmental incremental evolution process

4.3 Test Metrics for Controller Evaluation

During controller evolution, four fitness functions determined 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 trial. We designed the four fitness functions to measure how well a controller satisfied the goals of moving toward the radar, circling the radar closely, and flying in an efficient and stable manner.

These four fitness functions worked well to evolve good controllers, but because the functions were designed to exert evolutionary pressure throughout each run, not all the values each function produces are immediately meaningful. For the purposes of testing evolved controllers, we designed four test functions which measure the same qualities as the four fitness functions. The values these test functions produce are more meaningful to an observer.

4.3.1 Flying to the radar

The primary goal of the UAV is to fly from its initial position to the radar site as quickly as possible. The first fitness function, $fitness_1$, measured how well controllers accomplish this task by averaging the squared distance between the UAV and the goal over all time steps. We normalized 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. However, this measure does include a slight bias against longer initial distances, and produces a value without much meaning for an observer. We eliminated this bias in the first test function, $test_1$, by measuring percent error in flight time to the target. The total simulated time of four hours, or 14400 seconds, is divided into T_{in}, the number of seconds the distance between the UAV and radar is less than 10 nmi, and T_{out}, when this distance is greater than or equal to 10 nmi.

$$T_{total} = T_{in} + T_{out} = 14400\ seconds \tag{5}$$

The error in the time it takes to fly to the radar is just the actual time, T_{out}, minus the shortest possible time, T_{expect}, which is computed from D, the shortest possible distance in nautical miles a UAV could travel to fly from its starting position to each radar location, and the UAV speed of 80 knots.

$$T_{expect} = \frac{D}{80\dfrac{nmi}{hour} \cdot \dfrac{1\ hour}{3600\ seconds}} = 45 \cdot D \tag{6}$$

The test function is given as

$$test_1 = \left[\frac{T_{\mathrm{out}} - T_{\mathrm{expect}}}{T_{\mathrm{expect}}} \right] \tag{7}$$

For our tests, a value for $test_1$ near zero indicates a good flight.

4.3.2 Circling the radar

In early tests, we found that finding the mean squared circling distance exerted more pressure to evolve good circling behavior than if we simply used the mean circling distance. The circling distance fitness function used to evolve the controllers used the mean squared distance between the UAV and the radar when this distance was less than 10 nmi. For our tests, we were more concerned with the actual mean circling distance, so the test function, $test_2$, is the mean circling distance between the UAV and the radar when this distance is less than 10 nmi. The circling distance test function is

$$test_2 = \frac{1}{T_{in}} \sum_{i=1}^{T} inrange \cdot d_i \tag{8}$$

where *inrange* equals 1 if the distance between the UAV and the radar is less than 10 nmi and 0 otherwise.

4.3.3 Efficient flight

The first fitness function used to measure the efficiency of flight, *fitness*$_3$, is the number of time steps the UAV spends with a roll angle of 0° while traveling to the target. When the mean value of this fitness function is taken over many simulated flights, it provides a good measure of the amount of time a UAV spends flying in the most efficient posture. For the ability to look at single flights as well as a large number of simulations, we created *test*$_3$, which measures the percentage of the expected time the UAV spends flying level.

$$test_3 = \frac{1}{T_{\text{expect}}} \left| \left(\sum_{i=1}^{T} (1 - inrange) \cdot level \right) - T_{\text{expect}} \right| \tag{9}$$

where *level* is 1 when the UAV has been level for two consecutive time steps and 0 otherwise. For our tests we would like *test*$_3$ to be as small as possible.

4.3.4 Stable flight

The second test function to evaluate the efficiency of flight is a measure of turn cost. While UAVs are capable of quick, sharp turns, it is preferable to avoid these in favor of more gradual turns. The original turn cost fitness function *fitness*$_4$, also used as *test*$_4$, was intended to penalize controllers that navigate using a large number of sharp, sudden turns because this behavior may cause unstable flight or stalling. The UAV can achieve a small turning radius without penalty by changing the roll angle gradually; the 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

$$test_4 = \frac{1}{T} \sum_{i=1}^{T} hardturn \cdot \left| \varphi_i - \varphi_{i-1} \right| \tag{10}$$

where φ is the roll angle of the UAV and *hardturn* is 1 if the roll angle has changed by more than 10° since the last time step and 0 otherwise. We would like to minimize *test*$_4$.

4.4 Controller Evaluation

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 acceptable 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 a roll angle of 0° for approximately half the distance to the radar, and turn sharply less than 0.5% of the total flight time. If a controller had *test*$_1$ less than 0.2, *test*$_2$ less than 2, *test*$_3$ less than 0.5, and *test*$_4$ less than 0.05, the evolution was considered successful. These baseline values were used only for our analysis, not for the evolutionary process.

4.5 Direct Evolution

Table 2 shows the number of successful runs and the success rate for each of the five radar types and the total number of successful controllers, average number of successful controllers for an evolutionary run, and maximum number of controllers evolved in an evolutionary run for each of the radar types using direct evolution.

	Evolutionary runs			Successful controllers		
Radar type	Total	Successful	Percent	Total	Average	Maximum
Continuous, stationary	50	45	90%	3,149	62.98	170
Continuous, mobile	50	36	72%	2,266	45.32	206
Inter. (regular), stationary	50	25	50%	1,891	37.82	156
Inter. (irregular), stationary	50	29	58%	2,374	47.48	172
Intermittent, mobile	50	16	32%	569	11.38	93

Table 2. Number of successful runs and controllers for direct evolution experiments

Unlike continuously emitting radars, intermittently emitting radars were quite difficult for evolution. This should come as no surprise; the sensors on-board the UAV receive only half as much information from this type of radar as from a continuously emitting radar. Since the controllers evolved in this research have no *a priori* knowledge of the radar location and no internal model of the world, evolution must devise a strategy for times when the emitter is turned off. Despite the increased difficulty of this experiment, evolution was able to produce a large number of successful controllers.

4.6 Incremental Evolution

The results of the incremental evolution experiments are shown in Table 3.

	Evolutionary runs			Successful controllers		
Radar type	Total	Successful	Percent	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

Table 3. Number of successful runs and controllers for incremental evolution experiments

To begin the incremental evolution process, we evolved controllers for continuously emitting, stationary radars. This new set of evolutionary runs was used as a seed for the incremental evolution experiments. Like the experiments described in Section 4.5, 45 of the 50 evolutionary runs were successful, for a success rate of 90%

In the second stage of incremental evolution, each of the seed populations was used as the initial population for an evolutionary run, which evolved for 400 generations on continuously emitting, mobile radars. The use of incremental evolution improved the success rate of evolution on this type of radar. The 90% success rate using incremental evolution was an increase over the 72% success rate using direct evolution.

In the third stage of incremental evolution, each of the populations evolved in the second stage was used as a seed population for 400 generations of evolution on intermittently emitting, stationary radars with regular periods. The use of multiple increments, or stages of evolution, dramatically increased the ability of evolution to produce adept controllers for this type of radar. The success rate for evolution on intermittently emitting, stationary radars increased from 50% for direct evolution to 84% in this experiment. This increase in success rate suggests that incremental evolution is a very effective technique for this problem.

In the fourth and final stage of incremental evolution, each of the populations evolved the third stage was used as a seed population to evolve controllers for intermittently emitting, mobile radars with regular periods over 400 generations. Using multiple stages of incremental evolution increased the ability of evolution to successfully produce good results for this radar type. The success rate for intermittently emitting, mobile radars was 32% for direct evolution, but the success rate jumped all the way to 74% with incremental evolution.

4.7 Transference to a Wheeled Mobile Robot

To evaluate the ability of evolved controllers to control real vehicles, we transferred evolved UAV navigation controllers to a wheeled mobile robot. We used a small autonomous mobile robot called the EvBot II (Mattos, 2003), shown in Figure 3. The robot is equipped with an on-board computer 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. The EvBot is equipped with an on-board passive sonar system that 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 passive sonar system is susceptible to environmental noise, and the direction of a source found by the acoustic array is only accurate within approximately ±45°.

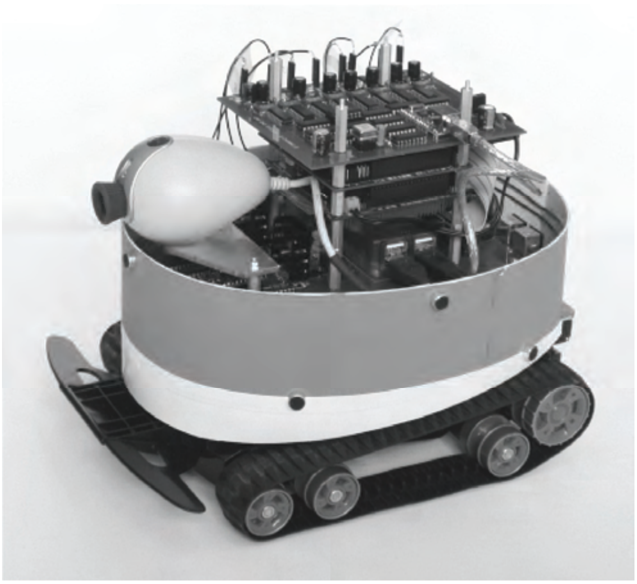

Figure 3. The EvBot II mobile robot equipped with an acoustic array

Testing the evolved controllers on the EvBot was attractive because the passive sonar system 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. Since the sensor accuracy of the acoustic array was so much worse than that of the AoA sensor in our simulation, we evolved new controllers to transfer to the EvBot. The only change in the simulation was changing the AoA accuracy from ±10° to ±45°.

Transference experiments were done in a 153 inch by 122 inch arena. A video camera with a fisheye lens was mounted above the maze environment to document 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. The EvBot was only calibrated to turn at multiples 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 0.02 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.

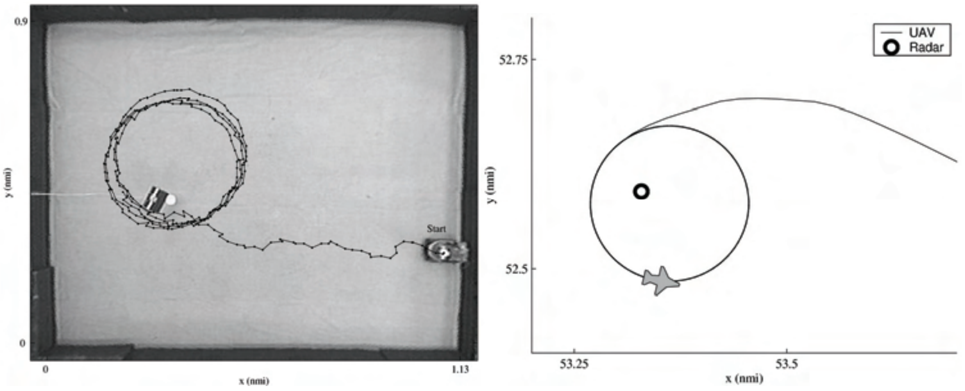

Figure 4. Circling behavior comparison for the EvBot (scaled maze size shown in nmi) and a simulated UAV, both running evolved controllers

Controllers evolved with a less accurate sensor were not as well adapted as those from previous work using more accurate sensors; flight paths were much less smooth and required more turns (Barlow et al., 2005). 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.

Figure 4 shows a path from one of the experiments compared to the circling behavior in simulation of controllers evolved with ±10° sensor accuracy. Running this evolved controller on the EvBot produces a tight circling behavior with a regular orbit around the target.

5. Robustness Analysis of GP Navigation Controllers

Over the 50 evolutionary runs with populations of 500 for each run, we produced 25,000 GP trees. In each evolutionary run, all 500 members of the final population fell along the Pareto front. Many of these controllers, however, only performed well according to one of the test metrics described in Section 4.3 while doing poorly at the others. The evolved controller best suited for transference to a real UAV would perform well on all of the test metrics. Rather than thoroughly testing all of the evolved controllers, including the most poorly performing, we chose to test only a small number of the best controllers. We established several performance metrics to evaluate controllers. Through successive performance metric evaluations, we selected the 10 best controllers for testing and subjected these evolved controllers to a series of robustness tests.

5.1 Performance Metrics

Multi-objective optimization produces a Pareto front of solutions, rather than a single best solution. In order to rank the controllers, each performance metric should combine the four test functions into a single value. The basis for all the performance metrics are a set of baseline values, values for each test function that describe a minimally successful UAV controller, defined in Section 4.4. Controllers are compared using four performance metrics: 1) *failures*, 2) *normalized maximum*, 3) *normalized mean*, and 4) *average rank*.

The first performance metric, *failures*, measures the percentage of flights with test function values which fail to meet at least one of the baseline values. A simulation run is a failure if there exists an m such that $test_m(r,n)$ is greater than $baseline_m$, where $test_{1 \ldots 4}(r,n)$ are the values of the four test functions for simulation run n for radar r. The failure percentage for a controller f and a test t is given as

$$metric_1(f,t) = \frac{F}{N} \tag{11}$$

where N is the total number of simulations and F is the number of simulation runs that fail.

The second performance metric, *normalized maximum*, measures how poorly a controller does when it fails. While the *failures* performance metric measures how often a controller fails, it does not measure how badly it might fail. Some controllers might perform well most of the time, but do not fail gracefully. The *normalized maximum* performance metric measures the worst failure for a particular controller. For each test function, the largest of the N values for each R radar is normalized by the baseline value for that function. Each value $test_m(r,n)$ is described by the test function (m), the radar type (r), and the simulation number (n). The maximum value over the M test functions is the *normalized maximum*, given as

$$metric_2(f,t) = \max_M \left(\frac{\max_{R,N}\left(test_m(r,n) - baseline_m\right)}{baseline_m} \right) \tag{12}$$

The third performance metric, *normalized mean*, measures how well a controller performs in relation to the baseline values. While the two metrics above measure the consistency of the controller and how wildly it can fail, this metric shows the typical performance of a controller. The *normalized mean* is given as

$$metric_3(f,t) = \frac{1}{M}\sum_{m=1}^{M}\left(\frac{baseline_m - \frac{1}{R}\sum_{r=1}^{R}\frac{1}{N}\sum_{n=1}^{N} test_m(r,n)}{baseline_m}\right) \tag{13}$$

The test function values for each objective are first averaged over the number of samples *N*, then over the number of radars *R*. For each objective, this average is normalized by the corresponding baseline value. We compute the normalized mean by taking the average over the *M* objectives.

The fourth performance metric, *average rank*, combines the values from the first three metrics into a single metric. To measure the relative performance of the controllers and give each metric equal weight, the values for all *g* controllers are normalized to be between 0 and 1.

$$norm_k(f,t) = \frac{metric_k(f,t) - \min_G\left(metric_k(g,t)\right)}{\max_G\left(metric_k(g,t)\right) - \min_G\left(metric_k(g,t)\right)} \tag{14}$$

The value of $metric_4(f,t)$ is the mean of these normalized metrics.

$$metric_4(f,t) = \frac{1}{3}\sum_{k=1}^{3} norm_k(f,t) \tag{15}$$

If we wish to find $metric_4(f,T)$ where *T* is a set of tests, we find $metric_k(f,T)$ values for each of the first three metrics

$$metric_k(f,T) = \frac{1}{T}\sum_{t=1}^{|T|} metric_k(f,t) \tag{16}$$

then normalize using Equation 14 and compute the metric using Equation 15.

5.2 Controller Selection

The combination of simulated sensor noise and random positioning of UAVs and radars created an uncertain fitness landscape for this problem. During evolution, we averaged the values from 30 simulation trials to help mitigate this uncertainty, but for these robustness tests, orders of magnitude more tests would make test function values for individual controllers statistically meaningful. Rather than running thousands of simulations for each of the 25,000 controllers created by evolution—an approach that would have been too computationally expensive—we chose to perform a series of robustness tests on 10 of the best controllers. We selected these controllers over several stages, reducing the number of controllers by an order of magnitude during each step.

First, we selected all controllers whose mean was lower than the baseline values for the four test functions described in Section 4. Of the 25,000 evolved controllers, 1,602 controllers had average fitness values better than the baseline values. This first method of selection was chosen because these 1,602 controllers had already been shown to perform well on the five radar types of interest: continuously emitting, stationary radars; continuously emitting, mobile radars; intermittently emitting, stationary radars with regular emitting periods; intermittently emitting, stationary radars with irregular emitting periods; and intermittently emitting, mobile radars with regular emitting periods.

Second, we ran 100 simulation trials on each of the five radar types for each of the 1,602 controllers and measured each flight using the test functions outlined in Section 4.3. For each radar type, we selected the best 35% of controllers using the *failures* performance metric described above. Then, we took the intersection of these five sets of controllers, leaving 298 controllers out of the 1,602. This selection method was chosen to eliminate controllers that did not perform consistently well on all five of the radar types. Since some radar types were more difficult than others, using a single cutoff number of failures to select controllers would not have caught controllers that did not perform as well on the easier radar types. The choice of 35% was made in order to select approximately 300 controllers for further tests.

Finally, we cut these 298 controllers down to 10 using the *normalized maximum* performance metric. This performance metric was applied to all 298 controllers and the 10 controllers with the lowest *normalized maximum* were selected as the best controllers for further testing. While counting the number of times a controller fails to meet the baseline values is a good way to compare controllers, this method gives no indication of the magnitude of failure. Using the *normalized maximum* metric helped eliminate controllers that usually performed well, but occasionally performed extremely poorly. Since a consistently sub-optimal controller is preferable to an unpredictable one, this metric was a good way to cut the set of controllers to a small number for final testing.

We compared these 10 evolved controllers to two designed controllers, a hand-written controller specific to this domain and a proportional-derivative (PD) controller, a common feedback controller. We did not use a proportional-integral-derivative (PID) controller because intermittent and mobile radars made the integral term detrimental to performance in preliminary tests. After examining many successful evolved controllers, we designed the hand-written controller using only the function set available to GP. We used strategies seen in evolved controllers, but tried to reduce the complexity to an easy to understand controller that performed well under the same conditions used in evolution. Given the current AoA, amplitude, and roll angle as inputs, if the amplitude is greater than zero, the hand-written controller will make a turn of fixed magnitude if necessary. If the AoA is greater than 10°, the roll angle will be increased. If the AoA is less than -10°, the roll angle will be decreased. If the AoA is between 10° and -10°, and the magnitude of the roll angle is greater than zero, the roll angle will be increased or decreased to move it closer to zero. Otherwise, the roll angle remains at 0°. The PD controller takes as input the current AoA and the AoA at the previous time step. The derivative of AoA is approximated by using the difference between the AoA at the previous time step and the current AoA. We adjusted the proportional and derivative gains to give good performance under the same conditions used for evolution.

5.3 Robustness Testing

During evolution, controller evaluation simulated an aircraft with constant speed, noise on the two sensor measurements (angle of arrival (AoA) and amplitude), and no state noise. The airspeed was 80 knots, AoA noise during evolution was ±10°, and the amplitude noise was $\pm 6dB$. To test the robustness of the evolved controllers, we increased the sensor noise and introduced sources of state noise. In addition to a control case with conditions identical to those during evolution, robustness tests fell into five categories by the type of noise: 1) AoA error, 2) amplitude error, 3) UAV airspeeds different from the speed used in evolution, 4) heading error, 5) and wind effects (position error). For each robustness test, we tested ten evolved controllers, a hand-written controller, and a PD controller against all five radar types. For every combination of controller and radar, we performed 10,000 simulation runs, for a total of 50,000 simulations for each controller per robustness test.

The sensor used most by evolved controllers was the AoA sensor. To test the robustness of evolved controllers, we varied the accuracy of the AoA sensor. The apparent AoA is given as the true angle to the target plus a normally distributed random number times the AoA accuracy. This accuracy was set to ±10° during evolution. For these robustness tests, we used AoA accuracies of ±{15°, 20°, 25°, 30°}.

While the amplitude sensor was not often used by evolved controllers, we did one test where we increased the amplitude error. While the controllers were being evolved, the amplitude error was set to 6 dB; the robustness test used an error of 12 dB. In both cases, this error was multiplied by a normally distributed random number and added to the true amplitude to compute the apparent amplitude.

During evolution, the speed of the UAV was held constant at 80 knots, but a UAV can obviously be flown at a variety of speeds. For instance, the UAV being considered for flight tests has a stall speed of 40 knots and a top speed of 110 knots. To test the performance of the evolved controllers at different speeds, we chose to test at 50 and 100 knots, values near the low and high end of the speed range.

The evolved UAV controllers have a single control variable, roll angle, which is used by the simulation to change the heading of the UAV at each time step. In the version of the simulation used to evolve controllers, there was no noise in this process. It is possible that on a real UAV, the autopilot might not be able to respond perfectly to turn requests or wind might push the UAV off-course. To test the robustness of evolved controllers this possibility, we added noise to the heading state variable. At each time step, a normally distributed random variable multiplied by a heading error was added to the heading computed using the desired roll angle. For our tests, we used heading error values of ±{0.5°, 1.0°, 1.5°, 2.0°}.

A significant source of state error for UAVs in the real world is wind. During evolution, the effects of wind on a UAV were not taken into account. While wind may function as a source of heading noise, the largest effect examined here was on position error. For our robustness tests, we make the simplifying assumption that the wind acts as an external force vector that can be summed with the propulsion force vector to obtain the new position of the UAV. In reality, this assumption is pessimistic, but for the purposes of these robustness tests would help to gauge the effects of wind on evolved controllers. The wind direction was set randomly for each simulation, but once set, the wind direction was held constant for the duration of the simulation. The wind speed was calculated at each time step as the mean wind speed plus some variance. For our tests, we used wind speeds of {5, 10, 20, 30} knots with a variances of {1, 1, 5, 5} knots.

5.4 Robustness Test Results

For each test, we ranked the 12 controllers (10 evolved controllers, labeled A to J; the hand-written controller, hd; and the PD controller, pd) based on each of the four performance metrics. Most of the results presented are rankings over several tests. When ranking over several tests, we used the averages of the performance metric values for each test except for the *average rank* metric; the technique used to compute this metric is described in Section 5.1.

Overall Ranking	best	2	3	4	5	6	7	8	9	10	11	12
failures	G	**D**	E	F	J	H	A	C	B	**pd**	I	**hd**
normalized maximum	**pd**	**D**	I	F	G	**hd**	J	E	B	A	H	**C**
normalized mean	**D**	E	**hd**	G	F	J	H	**pd**	A	C	B	I
average rank	**D**	G	E	**pd**	F	J	H	**hd**	A	B	C	I
Initial Ranking	best	2	3	4	5	6	7	8	9	10	11	12
Failures	**pd**	A	B	C	**D**	E	F	G	H	I	J	**hd**
normalized maximum	**pd**	G	E	J	F	I	B	A	**D**	C	H	**hd**
normalized mean	**pd**	I	J	A	B	C	G	E	**D**	F	H	**hd**
average rank	**pd**	J	G	I	E	B	A	F	C	**D**	H	**hd**
AoA Ranking	best	2	3	4	5	6	7	8	9	10	11	12
Failures	G	**D**	E	F	I	H	J	B	A	C	**pd**	**hd**
normalized maximum	**pd**	**D**	I	E	G	**hd**	B	J	H	F	A	C
normalized mean	**D**	G	E	**hd**	F	J	H	I	C	B	A	**pd**
average rank	G	**D**	E	F	**pd**	J	**hd**	H	I	B	C	A
Heading Ranking	best	2	3	4	5	6	7	8	9	10	11	12
Failures	**D**	E	F	G	J	H	**pd**	A	C	B	I	**hd**
normalized maximum	**pd**	I	**D**	J	H	F	B	**hd**	A	G	E	C
normalized mean	**pd**	**D**	H	J	F	**hd**	E	I	C	G	B	A
average rank	**pd**	**D**	J	F	H	E	G	I	B	C	A	**hd**
Wind Ranking	best	2	3	4	5	6	7	8	9	10	11	12
Failures	**D**	E	G	**pd**	F	J	H	C	B	A	**hd**	I
normalized maximum	**pd**	G	F	E	**D**	**hd**	I	J	A	B	C	H
normalized mean	**pd**	**hd**	**D**	G	E	F	J	H	A	C	B	I
average rank	**pd**	G	**D**	E	F	J	H	**hd**	A	C	B	I

Table 4. Controller rankings for robustness tests

Controller rankings, as shown in Table 4, are divided into five separate sections. The first section shows the overall rankings, averaged over all 16 robustness tests described in Section 5.3. The second section shows the initial rankings, using the same conditions under which the controllers were evolved (the control test). The third section shows the AoA rankings, averaged over the control test and the four tests with decreased AoA accuracy. The results from the next three tests, increasing the amplitude noise and changing the UAV speed to 50 and 100 knots, were very similar to those for the control test, so these results are not shown in the interests of space. The fourth section shows the heading rankings, averaged over the control test and the four tests with increased heading error. The fifth section shows the wind rankings, averaged over the control test and the four tests with increased wind speed. Based on these rankings, evolved controller D was the best candidate to transfer to a real UAV. In all rankings, controller D, the hand-written controller, and the PD controller are highlighted. The failure percentages for controller D and the PD controller for each of the robustness tests for each radar type are shown in Table 5.

In the initial ranking, the PD controller performs better than all the evolved controllers and the hand-written controller on all four metrics. This was not surprising, as we had tuned the controller parameters for the control test, giving an average failure rate of 0.04%. The hand-written controller, which was not optimized, performed the worst with an average failure rate of 69.82%. Controller D performed well, with an average failure rate of 7.13%, but was ranked tenth on the *average rank* metric, ahead of only one other evolved controller and the hand-written controller.

As we increased AoA noise, the performance of the hand-designed controllers declined compared to the evolved controllers. When the AoA error was increased from ±10° to ±15°, the PD controller failed in 100% of tests and the hand-written controller failed in 92.72% of tests. On the other hand, the failure rate for controller D was only 9.88%. The average failure rate for controller D only rose above 25% (to 90.24%) once the AoA error was ±30°. For the five different settings of AoA accuracy, controller D was the most robust to AoA sensor noise of the controllers.

An increase in amplitude error did not significantly change the performance of any controller. The performances of the evolved controllers and the hand-written controller on the two tests varying the speed were similar to performances on the control case. The only controller that had trouble with different speeds was the PD controller, which had average failure rates of 100% for a speed of 50 knots and 92.69% for a speed of 100 knots. In addition to being tuned for a particular AoA accuracy, these tests suggest that the PD controller is also tuned for a particular airspeed.

For the robustness tests with heading noise, the PD controller was the best, followed by controller D, which was the most consistent of the evolved controllers. The hand-written controller was the worst of the 12 controllers over these four tests and the control. Controller D actually failed significantly less than the PD controller; the average failure rate for the PD controller was over 50% when the heading error was ±1.5° and was 98.6% when the error was ±2°, while the average failure rate for controller D never got above 20%. The PD controller typically did not fail by large margins, and was ranked first on the *normalized maximum* and *normalized mean* performance metrics.

In the last series of tests, we added a different source of state error, wind. This series of tests clearly separated the evolved controllers; some of these controllers were simply not robust to the effects of wind. Controller D failed the fewest times, but was third in the *average rank* metric. The PD controller was best in this category. Despite the large number of evaluations, there is still some uncertainty in the performance metric values—for example, when AoA error is increased from the control test, the failure rate for controller D on continuous, mobile radars actually decreases slightly. For most of the robustness tests, this uncertainty was small—in the previous example this drop was on the order of 0.5%—but for wind this uncertainty was more apparent, especially for the higher wind speeds. One artifact of this uncertainty was the change in the failure rate for controller D on continuous, stationary radars from wind speeds of 10 to 20 to 30 knots. At 10 knots, the failure percentage was 16.33%. When the wind speed was increased to 20 knots, the failure percentage increased to 78.96%. However, when the wind speed was increased again to 30 knots, the failure percentage dropped to 61.05%. Other evolved controllers showed similar trends for increased wind speeds.

Test type	Cs		cm		irs		iis		irm		average	
	D	Pd	D	pd	D	pd	D	pd	D	pd	D	pd
control	0.00	0.04	10.19	0.03	1.18	0.07	10.39	0.04	13.87	0.03	7.13	0.04
AoA=15	0.00	100.0	9.67	100.0	6.55	100.0	16.43	100.0	16.74	100.0	9.88	100.0
AoA=20	0.00	100.0	9.80	100.0	20.10	100.0	26.06	100.0	24.81	100.0	16.15	100.0
AoA=25	0.01	100.0	9.40	100.0	35.39	100.0	37.76	100.0	34.90	100.0	23.49	100.0
AoA=30	99.65	100.0	99.05	100.0	77.15	100.0	90.12	100.0	85.25	100.0	90.24	100.0
Amp=12	0.00	0.04	10.83	0.02	1.41	0.04	10.73	0.04	13.21	0.07	7.24	0.04
Speed=50	0.00	100.0	12.38	100.0	0.56	100.0	3.70	100.0	16.53	100.0	6.63	100.0
Speed=100	0.00	92.92	10.28	92.42	2.63	92.23	17.04	92.79	14.83	93.09	8.96	92.69
Head=0.5	0.00	0.15	9.76	0.20	1.70	0.19	11.70	0.15	14.38	0.18	7.51	0.17
Head=1.0	0.00	2.57	10.84	2.46	4.48	2.50	16.88	2.66	16.08	2.60	9.66	2.56
Head=1.5	0.00	54.97	11.16	55.64	9.29	54.30	25.86	55.36	20.31	55.03	13.32	55.06
Head=2.0	0.00	98.56	11.04	98.57	19.85	98.55	37.63	98.67	29.38	98.64	19.58	98.60
Wind=5	0.02	0.39	11.43	0.00	2.40	0.01	12.52	0.05	15.70	0.14	8.41	0.12
Wind=10	16.33	0.05	25.66	0.00	21.13	1.39	30.97	0.02	30.82	1.11	24.98	0.51
Wind=20	78.96	97.54	72.10	94.76	49.96	95.12	72.05	100.0	60.77	94.83	66.77	96.45
Wind=30	61.05	98.36	71.20	100.0	63.43	96.64	95.42	97.28	86.10	99.67	75.44	98.39

Table 5. Failure percentages for the best evolved controller (D) and the PD controller (pd) listed by radar type (cs: continuous, stationary; cm: continuous, mobile; irs: intermittent, regular period, stationary; iis: intermittent, irregular period, stationary; irm: intermittent, regular period, mobile; avg: average)

The PD controller, as one might expect, performed extremely well under design conditions. When measuring how badly it failed using the *normalized maximum* performance metric, it was robust to all sources of noise, outperforming the other controllers in all tests. This controller was also robust to state noise, performing well under heading error and wind. However, the PD controller was susceptible to sensor noise. As the AoA error increased a small amount, the PD controller quickly failed. It was also dependent on the speed of the aircraft for good performance.

On most tests, the hand-written controller performed well, but was consistently worse than evolved controllers. This controller typically produced good results on the *normalized mean* performance metric, but was not robust to most forms of noise, though it did perform well on the speed and wind tests. The hand-written controller was included in these tests to show the difficulty of using the function and test sets to design an optimal controller by hand.

Overall, the best and most consistent controller was the best of our evolved controllers, controller D. In the overall rankings, this controller had the best *average rank* and finished in the top two on every performance metric. Unlike the hand-designed controllers, there was no category of tests where controller D performed poorly. This controller ranked highly in all of our tests, and its rank tended to increase as noise was increased.

6. Conclusions

Using multi-objective GP, 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 five radar types using both direct evolution and incremental evolution: continuously emitting, stationary radars; continuously emitting, mobile radars; intermittently emitting, stationary radars with regular periods; intermittently emitting, stationary radars with irregular periods; and intermittently emitting, mobile radars with regular periods. The use of incremental evolution dramatically increased the chances of producing successful controllers compared to direct evolution. Incremental evolution also produced controllers able to handle all five radar types.

Controllers were evolved to use inaccurate sensors in a noisy environment. 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.

We developed a series of robustness tests for evolved navigation controllers for UAV controllers developed in simulation. Before testing evolved controllers on a real UAV, we needed some assurance that the off-design performance of these controllers would be sufficient to accomplish the desired task and that controllers would be able to avoid behaviors that could potentially damage the aircraft. Also, since the controllers were evolved using multi-objective optimization, we needed to select a single best controller. When evolving controllers for systems where tests may be dangerous to the vehicle, robustness tests might be useful in selecting a controller and seeing how well it performs. The robustness tests described here apply several sources of sensor and state noise. If the real-world noise falls within the range where tests in simulation performed well, we can expect that transference will be successful.

The best evolved controller performed well in all robustness tests, consistently out-performing a hand-written controller and a proportional-derivative controller. When subjected to reasonable noise, this controller continued to perform well, even when many other controllers began to fail. Despite out-performing the hand-designed controllers and the other evolved controllers, our best controller has limits. When the AoA sensor was very inaccurate or the wind speed was high, this controller did not perform very well.

We have described the systematic development of a UAV navigation controller. We designed fitness functions and evolved controllers with multi-objective GP, using incremental evolution to increase the number of good controllers. To evaluate the ability of evolved controllers to control real vehicles, we transferred evolved UAV navigation controllers to a wheeled mobile robot. Finally, we used a series of robustness tests to screen the controllers and to select the single best controller. Based on this research, we feel confident that GP can evolve autonomous navigation controllers to effectively and safely control real UAVs in real flight environments.

7. References

Barlow, G.J. (2004). Design of autonomous navigation controllers for unmanned aerial vehicles using multi-objective genetic programming. *Master's thesis*, North Carolina State University, Raleigh, North Carolina.

Barlow, G.J.; Mattos, L.S.; Oh, C.K. & Grant, E. (2005). Transference of evolved unmanned aerial vehicle controllers to a wheeled mobile robot, *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 2087-2092, Barcelona, Spain, April 2005.

Barlow, G.J. & Oh, C.K. (2006). Robustness Analysis of Genetic Programming Controllers for Unmanned Aerial Vehicles, *Proceedings of the Genetic And Evolutionary Computation Conference,* pp. 135-142, Seattle, Washington, July 2006.

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

Deb, K.; Agrawal, S.; Pratap, A. & Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation,* Vol. 6, No. 2, 182-197.

Ebner, M. (1998). Evolution of a control architecture for a mobile robot, *Proceedings of the Second International Conference on Evolvable Systems,* pp. 303-310, Lausanne, Switzerland, September 1998.

Filliat, D.; Kodjabachian, J. & Meyer, J.-A. (1999). Incremental evolution of neural controllers for navigation in a 6-legged robot, *Proceedings of the International Symposium on Artificial Life and Robots,* Oita, Japan, January 1999.

Gomez, F. & Miikkulainen, R. (1997). Incremental evolution of complex general behavior. *Adaptive Behavior*, Vol. 5, No. 3-4, 317-342.

Gomez, F. & Miikkulainen, R. (2004). Transfer of neuroevolved controllers in unstable domains, *Proceedings of the Genetic and Evolutionary Computation Conference,* pp. 957-968, Seattle, Washington, June 2004.

Harvey, I.; Husbands, P. & Cliff, D. (1994). Seeing the light: Artificial evolution, real vision, *Proceedings of the International Conference on Simulation of Adaptive Behavior*, pp. 704-720, Brighton, UK, August 1994.

Hoffmann, F.; Koo, T.J. & Shakernia, O. (1998). Evolutionary design of a helicopter autopilot, In: *Advances in Soft Computing - Engineering Design and Manufacturing*, pp. 201-214, Springer-Verlag.

Jakobi, N.; Husbands, P. & Harvey, I. (1995). Noise and the reality gap: The use of simulation in evolutionary robotics, *Proceedings of the European Conference on Artificial Life*, pp. 704–720, Granada, Spain, June 1995.

Keymeulen, D.; Iwata, M.; Konaka, K.; Suzuki, R.; Kuniyoshi, Y. & Higuchi, T. (1998). Off-life model-free and on-line model-based evolution for tracking navigation using evolvable hardware, *Proceedings of the European Workshop on Evolutionary Robotics*, pp. 211-226, Paris, France, April 1998.

Koza, J. (1992). *Genetic Programming*. MIT Press, Cambridge, Massachusetts.

Lee, W.-P. & Hallam, J. (1999). Evolving reliable and robust controllers for real robots by genetic programming. *Soft Computing*, Vol. 3, No. 2, 63-75.

Lund, H.H. & Hallam, J. (1997). Evolving sufficient robot controllers, *Proceedings of the IEEE International Conference on Evolutionary Computation*, pp. 495-499, Indianapolis, Indiana, April 1997.

Marin, J.A.; Radtke, R.; Innis, D.; Barr, D.R. & Schultz, A.C. (1999). Using a genetic algorithm to develop rules to guide unmanned aerial vehicles, *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, pp. 1055-1060, Tokyo, Japan, October 1999.

Mattos, L.S. (2003). *The EvBot II*. Master's thesis, North Carolina State University, Raleigh, North Carolina.

Meyer, J.-A.; Doncieux, S.; Filliat, D. & Guillot, A. (2003). Evolutionary approaches to neural control of rolling, walking, swimming and flying animats or robots, In: *Biologically Inspired Robot Behavior Engineering*, pp. 1-43, Physica-Verlag.

Nelson, A.L. (2003). *Competitive relative performance and fitness selection for evolutionary robotics*. Ph.D. thesis, North Carolina State University, Raleigh, North Carolina.

Nelson, A.L.; Grant, E.; Barlow, G. & White, M. (2003). Evolution of complex autonomous robot behaviors using competitive fitness, *Proceedings of the IEEE International Conference on Integration of Knowledge Intensive Multi-Agent Systems*, pp. 145-150, Boston, Massachusetts, October 2003.

Nolfi, S. & Floreano, D. (2000). *Evolutionary robotics*. MIT Press, Cambridge, Massachusetts.

Nolfi, S.; Floreano, D.; Miglino, O. & Mondada, F. (1994). How to evolve autonomous robots: Different approaches in evolutionary robotics, *Proceedings of the International Workshop on the Synthesis and Simulation of Living Systems*, pp. 190-197, Boston, Massachusetts, July 1994.

Oh, C.K. & Barlow, G.J. (2004). Autonomous controller design for unmanned aerial vehicles using multi-objective genetic programming, *Proceedings of the Congress on Evolutionary Computation*, pp. 1538–1545, Portland, Oregon, June 2004.

Oh, C.K.; Cowart, G. & Ridder, J. (2004). Autonomous navigation control of UAVs using genetic programming, In: *NRL Review 2004*, pp. 197–199.

Richards, M.D.; Whitley, D.; Beveridge, J.R.; Mytkowicz, T.; Nguyen, D. & Rome, D. (2005). Evolving cooperative strategies for UAV teams, *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 1721-1728, Washington, DC, June 2005.

Shim, H.; Koo, T.J.; Hoffmann, F. & Sastry, S. (1998). A comprehensive study of control design for an autonomous helicopter, *Proceedings of the IEEE Conference on Decision and Control*, pp. 3653-3658, Tampa, Florida, December 1998.

Walker, J.; Garrett, S. & Wilson, M. (2003). Evolving controllers for real robots: A survey of the literature. *Adaptive Behavior*, Vol. 11, No. 3, 179-203.

Winkeler, J.F. & Manjunath, B.S. (1998). Incremental evolution in genetic programming, *Genetic Programming 1998: Proceedings of the Third Annual Conference*, pp. 403-411, Madison, Wisconsin, July 1998.

Wu, A.S.; Schultz, A.C. & Agah, A. (1999). Evolving control for distributed micro air vehicles, *Proceedings of the IEEE Conference on Computational Intelligence in Robotics and Automation*, pp. 174-179, Monterey, California, December 1999.