Robustness Analysis of Genetic Programming Controllers for Unmanned Aerial Vehicles

Gregory J. Barlow[*]
The Robotics Institute
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213
gjb@cmu.edu

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

ABSTRACT

While evolving evolutionary robotics controllers for real vehicles is an active area of research, most research robots do not require any assurance prior to operation that an evolved controller will not damage the vehicle. For controllers evolved in simulation where testing a poorly performing controller might damage the vehicle, thorough testing in simulation—subject to multiple sources of sensor and state noise—is required. Evolved controllers must be robust to noise in the environment in order to operate the vehicle safely. We have evolved navigation controllers for unmanned aerial vehicles in simulation using multi-objective genetic programming, and in order to choose the best evolved controller and to assure that this controller will perform well under a variety of environmental conditions, we have performed a series of robustness tests. The results show that our best evolved controller outperforms two hand-designed controllers and is robust to many sources of noise.

Categories and Subject Descriptors

I.2.9 [**Artificial Intelligence**]: Robotics—*Autonomous vehicles*; I.2.2 [**Artificial Intelligence**]: Automatic Programming—*Program synthesis*; I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search

General Terms

Performance, reliability

Keywords

evolutionary robotics, genetic programming, robustness, transference, unmanned aerial vehicles

[*]Gregory J. Barlow is also affiliated with the U.S. Naval Research Laboratory.

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 introduce a series of tests for choosing the best evolved controller for an unmanned aerial vehicle (UAV) from a population of controllers generated by multi-objective genetic programming and assuring the controller performance prior to real flight tests.

Transference of controllers evolved in simulation to real vehicles is an important issue in evolutionary robotics. Some controllers have been evolved *in situ* on physical robots [14], 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 [2, 5, 7]. 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. As the experiments in [7] show, 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 [2, 5, 7].

Unmanned aerial vehicles (UAV) are one type of robot that require 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.

The UAV has become popular for many applications, particularly where high risk or accessibility are 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 in [6] and compared to linear robust multi-variable control and nonlinear tracking control in simulation in [13]. Higher level controllers were evolved with UAVs as the target platform in [9], 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 in [15]. Like the experiments in [9], only simulation was used, the simulation environment was unrealistic, and no testing on real UAVs was attempted. Recently, Barlow et al. [1, 2, 3, 10] and Richards et al. [11] have evolved genetic programming (GP) controllers for fixed wing UAVs.

2. CONTROLLER EVOLUTION

In previous work, we have developed autonomous navigation controllers for fixed wing UAVs using multi-objective genetic programming [1, 2, 3, 10]. The goal is for a UAV to autonomously locate, track, and then orbit around a radar site. There are three main goals for an evolved controller. First, the UAV should move to the vicinity of the radar as quickly as possible. The sooner the UAV arrives in the vicinity of the radar, the sooner it can begin its primary mission, such as surveillance. Second, once in the vicinity of the electromagnetic source, the UAV should circle as closely as possible around the radar. This goal is especially important when proximity to the source affects the success of the application. Third, the flight path should be stable and efficient. The roll angle should change as infrequently as possible, and any change in roll angle should be small. Making frequent changes to the roll angle of the UAV could create dangerous flight dynamics and could reduce the endurance and range of the UAV.

Only the navigation portion of the controller is evolved; the low level flight control is done by an autopilot. The navigation controller receives radar emissions as input, and based on this sensory data and past information, the navigation controller determines a desired roll angle for the UAV. The autopilot 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 vehicles, since the evolved controllers are not designed for a specific UAV airframe or autopilot.

The controller is evolved in simulation. The simulation environment is a square 100 nautical miles (nmi) on each side. 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. 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.

Our simulation can model a wide variety of radars. Stationary radars were modeled as early warning radars, mobile radars as target acquisition radars [12]. Only the sidelobes of the radar emissions are modeled. The sidelobes are the parts of the emitted signal that are not part of the main beam, so they have a much lower power than the effective portion of the radar signal. If a controller can track a radar based only on the sidelobes, the radar can be tracked no matter the direction in which it is pointed, increasing the robustness of the system. Additionally, Gaussian noise is added to the amplitude of the radar signal. The receiving sensor can perceive only two pieces of information: the amplitude and the angle of arrival (AoA) of incoming radar signals. The AoA measures the angle between the heading of the UAV and the source of incoming electromagnetic energy. Real AoA sensors do not have perfect accuracy in detecting radar signals, so the simulation models an inaccurate sensor. The accuracy of the AoA sensor can be set in the simulation; in this research, controllers were evolved with an AoA accuracy of $\pm 10^\circ$.

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, 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, a realistic value for an autopilot. 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 [2].

As in previous work, navigation controllers were evolved using multi-objective GP with non-dominated sorting, crowding distance assignment to each solution, and elitism. We evolved UAV controllers using an implementation of NSGA-II [4] for GP. The function and terminal sets are defined as

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

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

The two available sensor measurements are the amplitude of the incoming signal and the AoA, or angle between the heading and the source of incoming electromagnetic energy. Additionally, the slope of the amplitude with respect to time is available to GP. When turning, there are six available actions. Turns are either 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.

Evolution was generational, with crossover and mutation similar to those outlined by Koza in [8]. The parameters used by GP and the four fitness functions used to evolve controllers are shown in [1, 3, 10]. Tournament selection was used. Initial trees were randomly generated using ramped half and half initialization. We performed 50 evolutionary runs at 500 individuals each, producing 25,000 GP trees. All computation was done on a Beowulf cluster parallel computer with ninety-two 2.4 GHz Pentium 4 processors.

Controllers were evolved using environmental incremental evolution [1, 3] to track four radar types: continuously emitting, stationary radars; continuously emitting, mobile radars; intermittently emitting, stationary radars; and intermittently emitting, mobile radars. The intermittently emitting radars had periods of 10 minutes and emitting durations of 5 minutes. The mobile radars were modeled as finite state machines with *setup*, *deployed*, *tear down*, and *move* states. A radar only emits in the *deployed* state. When a radar moves, the new location can be anywhere in the simulation area. These four radar types were used to evolve controllers, but for testing we also used a fifth radar type, intermittently emitting, stationary radars with irregular emitting periods. These radars are identical to intermittently emitting, stationary radars with regular emitting periods except that the period and duration of emission are set randomly at the beginning of each period. The mean periods and durations of the radars with irregular emitting periods are equal to those with regular emitting periods.

Environmental incremental evolution increases the difficulty of the environment or task faced by evolution in steps, while leaving the fitness function unchanged. In the first stage of evolution, randomly initialized populations were evolved on continuously emitting, stationary radars for 600 generations. In each of the next three stages, the evolved population from the prior stage was used as a seed population and evolved for 400 generations. Controllers evolved over continuously emitting, mobile radars in the second stage, intermittently emitting, stationary radars in the third stage, and intermittently emitting, mobile radars in the fourth stage. The use of incremental evolution increased the likelihood of evolving successful controllers and produced controllers able to handle all radar types [1, 3].

3. METHODS

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 fitness functions while doing poorly at the others. The evolved controller best suited for transference to a real UAV would perform well on all of the fitness and test functions. Rather than test 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.

3.1 Test Functions

During controller evolution, four fitness functions determined the success of individual UAV navigation controllers [1, 3, 10]. 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 the 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 the observer.

3.1.1 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 $test_1$ 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.

$$T_{total} = T_{in} + T_{out} = 14400\, seconds \tag{1}$$

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 a UAV could travel to fly from its starting position to each radar location.

$$T_{expect} = \frac{D}{80\frac{nmi}{hr} * \frac{1\,hr}{3600\,s}} = 45 * D \tag{2}$$

This test function is given as

$$test_1 = \left[\frac{T_{out} - T_{expect}}{T_{expect}}\right] \tag{3}$$

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

3.1.2 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

$$test_2 = \frac{1}{T_{in}} \sum_{i=1}^{T} in\_range * distance_i \tag{4}$$

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

3.1.3 Efficient flight

The first fitness function used to measure the efficiency of flight, $fitness_3$, is equal to 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.

$$test_3 = \left| \frac{\left(\sum_{i=1}^{T}(1 - in\_range) * level\right) - T_{expect}}{T_{expect}} \right| \tag{5}$$

where *level* is 1 when the UAV has been level for two consecutive time steps and 0 otherwise. For our tests we'd like $test_3$ to be as small as possible.

3.1.4 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, which is 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, 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

$$test_4 = \frac{1}{T} \sum_{i=1}^{T} hard\_turn * |roll\_angle_i - roll\_angle_{i-1}| \tag{6}$$

where *roll_angle* is the roll angle of the UAV and *hard_turn* 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 $test_4$.

3.2 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. An acceptable controller would have fitness or test function values at least as good as the baseline values. These baseline values were used only for our analysis, not for the evolutionary process. We defined a minimally successful UAV controller as able to move quickly to the target radar site with less than 20% error, 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 5% of the total flight time, giving baseline values of $baseline = \{0.2, 2.0, 0.5, 0.05\}$. 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 $\exists m$ such that $test_m(r, n) > baseline_m$where $test_{1...4}(r, n)$ 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

$$metric_1(f, t) = F/N \tag{7}$$

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 radars 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

$$metric_2(f, t) = \max_M \left(\frac{\max_{R,N}(test_m(r,n)) - baseline_m}{baseline_m} \right) \tag{8}$$

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

$$metric_3(f, t) = \frac{1}{M} \sum_{m=1}^{M} \left(\frac{baseline_m - \frac{1}{R}\sum_{r=1}^{R} \frac{1}{N} \sum_{n=1}^{N} test_m(r,n)}{baseline_m} \right) \tag{9}$$

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 so each metric has equal weight, the values for all g controllers are normalized to be between 0 and 1.

$$norm_k(f, t) = \frac{metric_k(f, t) - \min_G (metric_k(g, t))}{\max_G (metric_k(g, t)) - \min_G (metric_k(g, t))} \tag{10}$$

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

$$metric_4(f,t) = \frac{1}{3}\sum_{k=1}^{3} norm_k(f,t) \qquad (11)$$

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

$$metric_k(f,T) = \frac{1}{T}\sum_{t}^{|T|} metric_k(f,t) \qquad (12)$$

then normalize using Equation 10 and compute the metric using Equation 11.

3.3 Selecting controllers for testing

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 fitness functions described in [1, 3]. 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 3.1. 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 wouldn't 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 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.

3.4 Robustness tests

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 $\pm 10^\circ$, 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 $\pm 10^\circ$ during evolution. For these robustness tests, we used AoA accuracies of $\pm\{15, 20, 25, 30\}^\circ$.

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

Table 1: Controller rankings for robustness tests

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

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 $\pm\{0.5, 1.0, 1.5, 2.0\}^{\circ}$.

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. Wind was modeled as having a constant direction throughout simulation, set randomly for each 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.

4. 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. In the interest of space, 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 to compute this metric is described in Section 3.2.

Controller rankings, as shown in Table 1, are divided into five separate sections. The first section shows the overall rankings, averaged over all 16 robustness tests. 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

Table 2: 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)

	cs		cm		irs		iis		irm		avg	
Test type	D	pd	D	pd	D	pd	D	pd	D	pd	D	pd
control case	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

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 2.

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 $\pm10^\circ$ to $\pm15^\circ$, 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 $\pm30^\circ$. For the five different settings of AoA accuracy, controller D was the most robust to AoA sensor noise of the controllers.

As mentioned, we do not show the rankings for the amplitude noise and speed tests in the interest of space. 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 test 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 $\pm1.5^\circ$ and was 98.6% when the error was $\pm2^\circ$, 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.

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.

5. CONCLUSIONS

In this paper, we developed a series of robustness tests for evolved navigation controllers for unmanned aerial vehicle 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. Based on these robustness tests, we feel confident in the ability of these evolved controllers to safely control a real UAV. In the next stage of this work, a physical UAV will be controlled by our best controller.

6. ACKNOWLEDGMENTS

The Swampworks project office of the Office of Naval Research provided financial support for this work. The U.S. Naval Research Laboratory (Code 5730) provided computation time on their Beowulf cluster. Gregory J. Barlow is supported by a National Defense Science and Engineering Graduate Fellowship.

7. REFERENCES

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

[2] G. J. Barlow, L. S. Mattos, C. K. Oh, and E. Grant. Transference of evolved unmanned aerial vehicle controllers to a wheeled mobile robot. In *Proceedings of the IEEE International Conference on Robotics and Automation*, Barcelona, Spain, April 2005.

[3] G. J. Barlow, C. K. Oh, and E. Grant. Incremental evolution of autonomous controllers for unmanned aerial vehicles using multi-objective genetic programming. In *Proceedings of the 2004 IEEE Conference on Cybernetics and Intelligent Systems*, Singapore, December 2004.

[4] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, April 2002.

[5] F. J. Gomez and R. Miikkulainen. Transfer of neuroevolved controllers in unstable domains. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2004)*, June 2004.

[6] F. Hoffmann, T. J. Koo, and O. Shakernia. Evolutionary design of a helicopter autopilot. *3rd On-line World Conference on Soft Computing*, 1998.

[7] 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*, pages 704–720, 1995.

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

[9] J. A. Marin, R. Radtke, D. Innis, D. R. Barr, and A. C. Schultz. Using a genetic algorithm to develop rules to guide unmanned aerial vehicles. In *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, Tokyo, Japan, 1999.

[10] C. K. Oh and G. J. Barlow. Autonomous controller design for unmanned aerial vehicles using multi-objective genetic programming. In *Proceedings of the Congress on Evolutionary Computation*, pages 1538–1545, Portland, OR, June 2004.

[11] M. D. Richards, D. Whitley, J. R. Beveridge, T. Mytkowicz, D. Nguyen, and D. Rome. Evolving cooperative strategies for UAV teams. In *Proceedings of the 2005 Genetic and Evolutionary Computation Conference*, Washington, DC, June 2005.

[12] D. C. Schleher. *Introduction to Electronic Warfare.* Artech House, 1986.

[13] H. Shim, T. J. Koo, F. Hoffmann, and S. Sastry. A comprehensive study of control design for an autonomous helicopter. In *IEEE Conference on Decision and Control*, December 1998.

[14] J. Walker, S. Garrett, and M. Wilson. Evolving controllers for real robots: A survey of the literature. *Adaptive Behavior*, 11(3):179–203, 2003.

[15] A. S. Wu, A. C. Schultz, and A. Agah. Evolving control for distributed micro air vehicles. In *IEEE Conference on Computational Intelligence in Robotics and Automation*, November 1999.