Evolving a Multiagent Controller for Micro Aerial Vehicles

Max Salichon, and Kagan Tumer, *Senior Member, IEEE*

***Abstract*—Micro Aerial Vehicles (MAVs) are notoriously difficult to control as they are light, susceptible to minor fluctuations in the environment, and obey highly non-linear dynamics. Indeed, traditional control methods, particularly those relying on difficult to obtain models of the interaction between an MAV and its environment have been unable to provide adequate control beyond simple maneuvers. In this paper, we address the problem of controlling an MAV (which has segmented control surfaces) by evolving a neuro-controller and fine-tuning it using multiagent coordination techniques. This approach is based on a control strategy that learns to map MAV states (position, velocity) to MAV actions (e.g., actuator position) to achieve good performance (e.g., flight time) by maximizing an objective function. The main difficulty with this approach is defining the objective functions at the MAV level that allow good performance. In addition, to provide added robustness, we investigate a multiagent approach to control where each control surface aims to optimize a local objective. Our results show that this approach not only provides good MAV control, but provides robustness to (i) wind gusts by a factor of six; (ii) turbulence by a factor of four; and (iii) hardware failures by a factor of eight over a traditional control method.**

***Index Terms*—Multiagent Control, Evolutionary Control, Neuro-Evolution, Micro Aerial Vehicles**

I. Introduction

Micro Aerial Vehicles (MAVs) offer a great alternative to piloted vehicles in many tasks such as surveillance [1], reconnaissance, sensing [2], search and rescue [3], and automatic target recognition [4], [5]. MAVs can accomplish such missions without endangering human lives, giving an important edge to the organization using them. Many MAV platforms and control systems have been studied [6]–[12], and this remains an active area of research. In this paper, we address the problem of controlling an MAV with segmented control surfaces using a neuro-controller which is fine-tuned using multiagent coordination techniques.

As opposed to larger Unmanned Aerial Vehicles (UAVs), MAVs are small, and typically range from insect to bird size (15 to 60 cm). The flight speed of these MAVs is typically between 10 to 50 mph [13], [14]. They are typically unstable and difficult to control due to highly non-linear dynamics [7]. Additionally, MAVs are much more sensitive to wind gusts and turbulences due to their size [15], [16]. As such, an MAV control system has to be fast, flexible and robust in order to achieve stable and reliable flight characteristics.

Max Salichon and Kagan Tumer are with the School of Mechanical, Industrial, and Manufacturing Engineering, Oregon State University, Corvallis, OR, 97331 USA. e-mail: max.salichon@gmail.com , kagan.tumer@oregonstate.edu

Manuscript received xxx

A. Contribution of this Work

In this paper, we show that evolving direct policies in the form of neural networks allows the robust control of micro aerial vehicles. In particular, the key contributions of this work are to:
(i) show that neuro-evolutionary techniques can control an MAV effectively;
(ii) show that the neuro-evolutionary controller is significantly more robust to wind gusts and turbulence than a traditional PID controller;
(iii) show that using segmented control surfaces and multiagent control not only improves MAV performance, but these improvements are obtained with smaller adjustments to aileron positions, leading to both more efficient and more responsive controllers.

The paper is organized as follows: Section II outlines the key work related to this research. Section III describes the MAV platform used in this work. Section IV presents the control algorithms and simulator. Section V presents the experimental results, where the performance and robustness of the neuro-evolutionary controllers are compared with a PID controller. Section V also discusses the results for a multiagent controller for MAVs with segmented control surfaces. Finally, Section VI presents a discussion on the relevance of the results and highlights directions for future work.

II. Related Work

Flexible-wing MAV designs have recently been used to improve vehicle stability, wind gust resistance, and payload capacity. In Flex-wing designs, the wing deforms to absorb energy from wind irregularities leading to better performance and flight characteristics as a form of passive control. It is also possible to use wing deformations as a form of active control by using an actuator to change the shape of the wing during flight, where roll control of an MAV was achieved by actively morphing the wing [12], [17]. Carbon fiber prototypes have been under development for several years [13] and show important advantages of the flexible-wing MAV concept that allows important design and control advantages. For example, flexible wings can lead to a higher maximum airspeed, higher climb rate, improved

maneuverability, and a higher lift to drag ratio which is particularly important for MAVs as it improves their gliding capabilities. Although flexible-wing designs have many benefits, they are often difficult to implement.

Segmented control surfaces offer a more practical approach than wing morphing for increasing aircraft controllability. A segmented control surfaces approach was implemented on a 5.5ft wingspan remote controlled UAV in [18], where wing ailerons were divided up into 16 independent actuated control surfaces. A reconfigurable controller was developed to actuate all 16 control surfaces and flight tests showed promising results and improved performance over an unmodified aircraft. Those tests provided good preliminary results for the use of segmented control surfaces, but provided no method for finding an optimal actuation mode for the system.

Traditional control techniques such as PID control have been used successfully for many control problems including aircraft control [2], [10], [11], [19]–[24]. These model based techniques perform well for large aircraft, where linear approximations of the system's dynamics produce good results. Unfortunately, model-based techniques are typically inadequate for MAV control due to inherent model inaccuracies and a failure of linear approximation techniques due to highly nonlinear dynamics and susceptibility to environmental disturbances. MAVs require more flexible controllers that can automatically adapt to model error and environmental changes.

Learning based control techniques (e.g., neuro-controllers where control policies are neural networks) are flexible, do not require a model of the system, can adapt to different platforms and dynamic environments, and are robust to noise and/or failures. In this work, we utilize a combination of neural networks and evolutionary computation techniques, which have been used successfully to solve a number of benchmark control problems including the inverted pendulum [25], the ball and beam problem [26], as well as different types of autonomous robot problems [27]–[29]. These techniques have also proven to be robust and efficient for complex control problems such as multi-robot control problems [30]–[33]. Using learning based techniques for controlling segmented control surfaces was presented in [29], [34], [35], where the control technique was based on the theory of collectives [36], [37]. Here, basic collectives were implemented, where agents consisted of a actuator, sensor, and logic package. The collectives based approach produced encouraging results for multiagent-based UAV control [34], [35].

III. MAV Platform: GENMAV

The platform selected for these experiment is GENMAV [16], an MAV developed by the Air Force Research Laboratory Munition Directorate (AFRL/RW). GENMAV is a flexible platform that could be modified depending on a particular application or technology. Characteristics of GENMAV (Figure 1) include a 24 inch wingspan with a 5 inch chord, circular fuselage 17 inches long, and a dihedral angle of 7 degrees. The weight of the platform is approximately 500 grams. The wing design was modified from previous versions in order to improve low speed performance. Additionally, the model of GENMAV was modified in order to test the impact of segmented ailerons. Six ailerons were added to each wing. In span, ailerons extend from the 50% to the 90% span points on the wing with each aileron 1" wide and 20% chord. The tail section was not modified. Aerodynamic characteristics were obtained using the vortex-lattice method aeroprediction code AVL (Athena Vortex Lattice [38]–[41]) and detailed data can be found in [16]. Similarly to other MAV platforms, GENMAV was designed for a flight speed of between 10 and 50 mph with an average flight speed around 30mph.

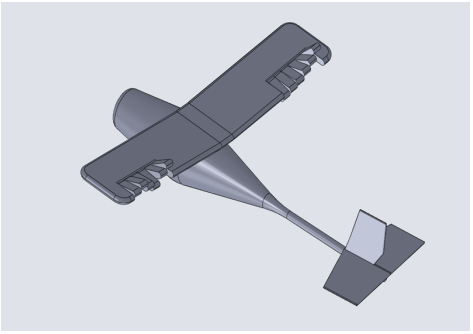


Fig. 1. GENMAV illustration reflecting the segmented control surfaces.

IV. Neuro-control for MAVs

The control of GENMAV is achieved through a feed-forward neural network using a neuro-evolutionary algorithm [32], [33], [42], [43]. The neural network learns the optimal control commands through the system objective function that is designed to minimize the error between the desired parameter value and the actual parameter value. The near optimal neural network control system is then saved and used for flight control where different desired altitudes and headings are achieved.

A. Neuro-Controller

A simple neuro-evolutionary algorithm was developed using the techniques outlined in [32], [33]. The algorithm maintains an initially empty pool of neural networks that are paired with some measure of their utility. While the pool is not full, the algorithm generates new random networks as seeds for future mutation, using values sampled from a Cauchy distribution. After this initial seeding period, the algorithm uses ϵ-greedy selection from the pool of networks and selectively mutates the chosen network using a different Cauchy distribution. In both cases, the new network is stored in the pool only after an agent has used it and sampled their resulting performance, with the poorest performing network being discarded.

The single hidden-layer, feed forward neural networks [44] used in these experiments have 2 inputs which correspond to the parameter value and parameter derivative (e.g altitude and altitude rate) retrieved from JSBSim. The single output of the neural networks is the control command or desired roll angle (e.g elevator down 20%). The experiments were conducted with three neural networks for the basic configuration (no segmentation): altitude control, roll control, and heading control. Each neural network output provides the elevator command, aileron command, and desired roll angle. For the segmented version of GENMAV, two different setups were used. The central controller setup is similar to the basic configuration except that the roll controller has 12 outputs corresponding to the 12 aileron segments. The multiagent controller setup includes additional neural networks so that each control surface is adjusted by its own independent controller. A total of 14 neural networks are used for the control surfaces and an additional neural network is used for heading control.

A sweep of the neural network parameters was conducted as a preliminary study in order to find values of the neural network parameters that provided satisfactory results in terms of learning and optimization of the objective functions. The neural networks are configured with 8 hidden units, a pool size of 20, an epsilon-greedy selection probability of $\epsilon = 0.05$, a level of initial weights of $\gamma = 0.1$, a level of mutations of $mutate\ \gamma = 0.05$, and a probability that a weight will be mutated of 0.02. Those parameters were then kept constant for the experiments described in section V. Altitude, roll, and heading neural network controllers were trained using the objective functions from Section IV-B. Training time for the altitude and roll neural network controllers was 5 seconds while training time for the heading neural network controller was 12 seconds (doubling or halving these times had little impact).

B. Objective Functions

An important part of using neural networks consists of designing an objective function that allows the neural network to learn at a satisfactory rate and at the same time achieves the system's objective. The objective functions used for these experiments are designed to minimize the error between the control parameter (altitude, roll, and heading) and its desired value. Since it was not possible to achieve proper control with a single neural network using the objective functions and conditions of our training environment, three different controllers were created. Each controller uses its own objective function that is specifically designed for that particular controller. The three controllers are for altitude, roll, and heading control. Their respective objective functions are G_Z for the altitude controller, G_Φ for the roll controller and G_Ψ for the heading controller. For the multiagent system, a neural network controls each individual control surface and receives its own specific fitness. This fitness is presented in Equation 4.

Objective Function for Altitude Control: G_Z

G_Z was designed to minimize the error between the desired altitude and the actual altitude:

$$G_Z = \frac{\alpha_Z}{\sum_{t=0}^{T}\left[\beta_Z|Z_d - Z_a| + \gamma_Z|\frac{dZ}{t}| + \delta_Z|\Theta_E|\right]} \quad (1)$$

where α_Z is a scaling constant, β_Z, γ_Z, and δ_Z are tuning constants, Z_d and Z_a are the desired and actual altitude, T is the simulation time, and Θ_E is the elevator position.

Objective Functions for Roll Control: G_Φ

Similarly, G_Φ is designed to minimize the error between the desired roll and the actual roll:

$$G_\Phi = \frac{\alpha_\Phi}{\sum_{t=0}^{T}\left[\beta_\Phi|\Phi_d - \Phi_a| + \gamma_\Phi|\frac{d\Phi}{t}| + \delta_\Phi|\Theta_A|\right]} \quad (2)$$

where α_Φ is a scaling constant, β_Φ, γ_Φ, and δ_Φ are tuning constants, Φ_d and Φ_a are the desired and actual roll, and T is the simulation time, and Θ_A is the aileron position.

Objective Function for Heading Control: G_Ψ

G_Ψ was designed to minimize the error between the desired heading and the actual heading:

$$G_\Psi = \frac{\alpha_\Psi}{\sum_{t=0}^{T}\left[\beta_\Psi|\Psi_d - \Psi_a| + \gamma_\Psi|\frac{d\Psi}{t}|\right]} \quad (3)$$

where α_Ψ is a scaling constant, β_Ψ and γ_Ψ are tuning constants, Ψ_d and Ψ_a are the desired and actual heading, and T is the simulation time.

Multiagent Objective Functions

Objective functions G_Z and G_Ψ for altitude and heading control remain the same but G_Φ changes to reflect the aileron segmentation. Aileron controllers receive a custom fitness G_{Φ_i} that includes the angle of the aileron segment that they control.

$$G_{\Phi_i} = \frac{\alpha_\Phi}{\sum_{t=0}^{T}\left[\beta_\Phi|\Phi_d - \Phi_a| + \gamma_\Phi|\frac{d\Phi}{t}| + \delta_\Phi|\Theta_{A_i}|\right]} \quad (4)$$

Where α_Φ is an arbitrary constant, β_Φ, γ_Φ, and δ_Φ are tuning constants, Φ_d and Φ_a are the desired and actual roll, and T is the simulation time, and Θ_{A_i} is the aileron segment position.

C. System Dynamics: JSBSim

To conduct the experiments, the neural network and PID controllers were coupled to JSBSim [45], a 6 DOF (Degrees Of Freedom) flight dynamics model (FDM) software library. JSBSim is a lightweight, data-driven, non-linear, six-degree-of-freedom (6DoF), batch simulation application aimed at modeling flight dynamics and

control for aircraft. JSBSim is a simulator that models physical entities such as the atmosphere, a flight control system, or an engine. In addition, it encapsulates mathematical constructs such as the equations of motion. Put together, JSBSim takes control inputs, calculates and sums the forces and moments that result from those control inputs and the environment, and advances the state of the vehicle (velocity, orientation, position, etc.) in discrete time steps.

The simulation runs consist of providing the error between desired heading and actual heading as well as the error between desired altitude and actual altitude as inputs to the neural network, obtaining angles for the control surfaces from its outputs, running JSBSim to provide the MAV state for the next time step, computing the objective function, and having the neural network learn from the objective function. Figures 2 and 3 show the overall control system setup and the neural network training diagram respectively.

D. MAV Controller

The MAV PID control is achieved through three different controllers, one for altitude, one for roll, and one for heading control. The altitude control PID uses the error between desired and actual altitude as well as altitude rate for its inputs, and it outputs the elevator command. Similarly, the roll and heading PID controllers take the error between desired and actual roll, and the roll rate as inputs for the roll control PID and the error between the desired and actual heading as inputs for the heading control PID. The outputs of the roll and heading PID controller are aileron command and desired roll respectively.

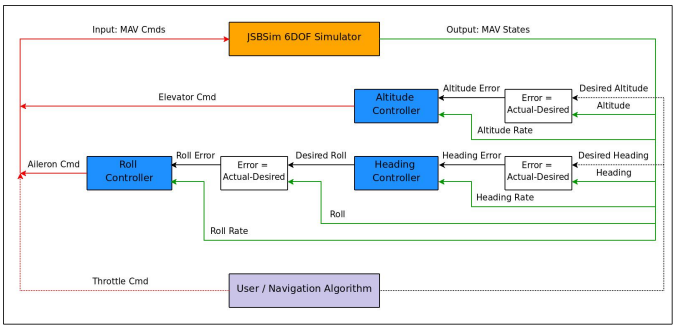


Fig. 2. Navigation Controller

1) Model and PID control: the control command for the PID controller is calculated with three separate parameters: the proportional, integral, and derivative values. The proportional term (Equation 5) is directly proportional to the error, the integral term (Equation 6) is based on the sum of previous errors and is used to correct small drift over time, and the derivative term (Equation 7) is based on the error rate of change. The weighted sum of these terms is the control command. Tuning of the PID controller is achieved by adjusting the three constants, K_P, K_I, and K_D which are PID gains, and $e(t)$ is the error:

$$P = K_P.e(t) \tag{5}$$

$$I = K_I.\int_0^t e(\tau)d\tau \tag{6}$$

$$D = K_D.\frac{de(t)}{dt} \tag{7}$$

Figure 2 shows the control system block diagram that includes interactions with the simulator and the user or navigation algorithm. The user or navigation algorithm provides the desired altitude and heading to the controllers as well as the throttle command to JSBSim. The elevator and aileron commands are provided by the controllers. The elevator command deflects the elevator up or down, while the aileron command deflects the ailerons in opposite directions. *In this work, the moving surfaces are "elevons" which combine the function of the elevators and ailerons.* The elevator command deflects the elevons up or down, while the aileron command deflects the elevons in opposite directions. The final elevon position is achieved by summing and scaling the elevator and aileron commands.

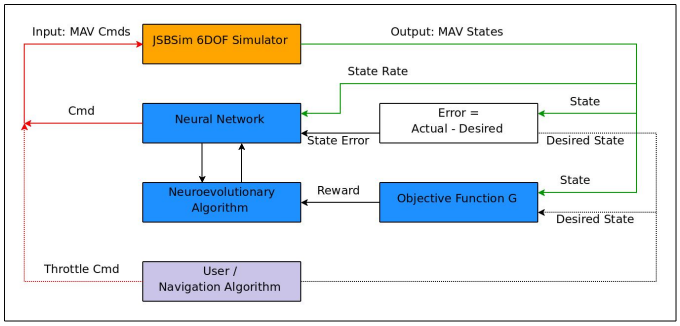


Fig. 3. Neural network training

2) Neuro-Evolutionary Control: Neural network control is done similarly with the same inputs and outputs as the PID controllers. Training the neural networks is achieved using the different objective function from Section IV-B. Figure 3 shows the training cycle for one controller. The evolutionary algorithm selects and mutates a neural network from its pool, and that neural network controls the system for a short period of time. The objective function is then used to determine "fitness" of that neural network's flight performance. The neural network performing the worst is then deleted form the pool. This cycle is repeated until a good solution is found. That neural network controller is then saved and used directly in the control loop with JSBSim.

3) Segmented Control Surfaces and Multiagent Control: Finally, we explore the impact of segmented control surfaces, where the control of the segmented aileron is similar to the unsegmented version except that the roll control neural network includes six outputs corresponding to the control commands sent to each aileron

segment. Altitude and heading control do not change and are achieved with a two inputs, one output neural network for each controlled parameter.

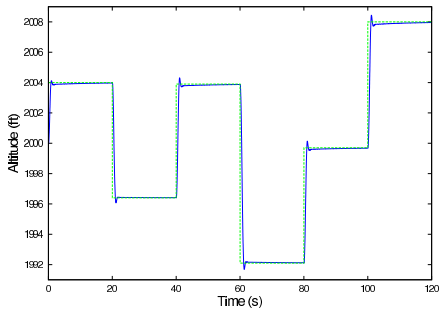
For this scenario, we also investigate the use of a multiagent system where each control surface is controlled by an independent neural network controller. The inputs to the controllers are the same as those discussed above, but the output directly control the position of a control surface. Twelve neural networks control the aileron segments while two control the elevons for a total of fourteen neural networks giving maximum flexibility to the control system.

V. Experimental Results

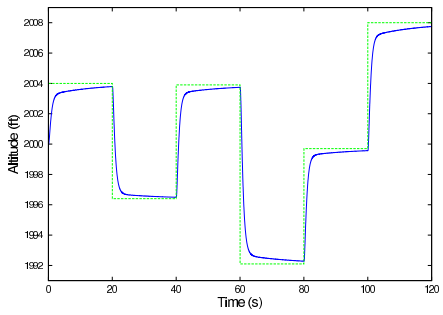
In this section we present simulation results showing the effectiveness of the proposed approach robustly control an MAV. In particular, we systematically investigate:

(A) Altitude control by neuro-evolutionary algorithm
(B) Heading control by neuro-evolutionary algorithm
(C) Wind gust resistance
(D) Turbulence handling
(E) Multiagent control of segmented surfaces
(F) Response to two types of failures

In each case, we compare the neuro-evolutionary algorithm to a PID controller and present results in system performance and the positions of the control surfaces.



(a) Neuro-evolutionary controller



(b) PID controller

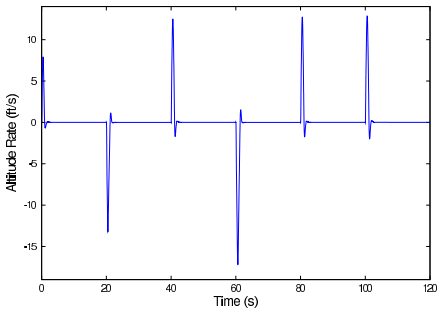
Fig. 4. Desired and actual altitude: the neuro-evolutionary controller achieves desired altitude faster than the PID controller with minimal overshoot

A. Altitude Control

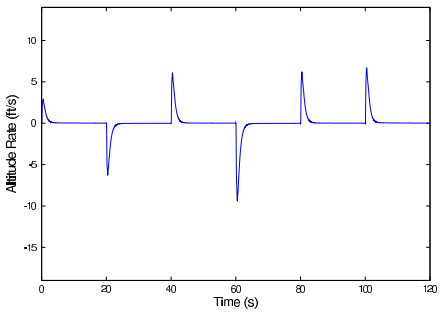
The altitude control PID was fairly straightforward to implement but required a significant amount of manual tuning to achieve the desired results. Figure 4(b) shows the MAV altitude with the dashed line representing the desired altitude and the solid line representing the actual altitude. The PID controller is able to track the desired altitude well with no overshoot. Here, the MAV gets within a foot of the target altitude in a few seconds. Further tuning decreased performance, therefore the PID altitude gains producing these results were maintained throughout these experiments.

Figure 4(a) shows the MAV altitude when using neural network control. Objective function G_Z from Equation 1 was used in the training of the neural network. The training consisted several thousand 5 second flights where the altitude control neural network was flying the plane. Looking at the altitude from Figure 4(a), the altitude control neural network performs well. The target altitude is reached quickly and efficiently with minimal overshoot. The neural network's behavior is more aggressive than the PID's, which allows for better and faster tracking of the desired altitude without compromising the behavior of the system and without creating instabilities.

The altitude rate for both neural network and PID controllers is shown in Figure 5. This provides additional information regarding the behavior of the system and is helpful when designing the controllers. The altitude rate should be kept within acceptable limits to avoid destabilizing the system and having too sharp of a response. The PID controller keeps the altitude rate with 10ft/sec (Figure 5(b)) while the neural network keeps it within 16ft/sec (Figure 5(a)). The difference between the two controllers is explained by the fact that the neural network has learned a more aggressive policy.

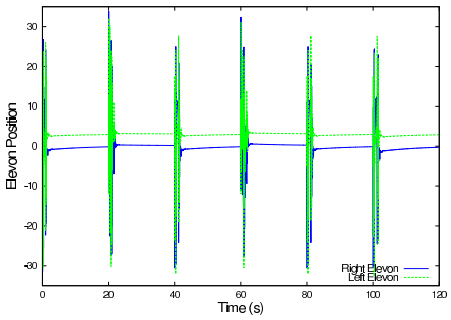


(a) Neuro-evolutionary controller

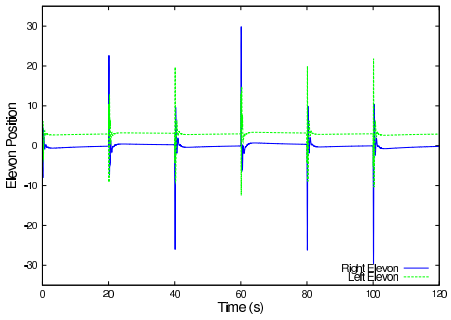


(b) PID controller

Fig. 5. Altitude rate: The change in altitude is higher for the neuro-controller, confirming the observed performance.



(a) Neuro-evolutionary controller



(b) PID controller

Fig. 6. Elevon Positions for altitude control: The neuro-evolutionary controller is slightly more aggressive in changing elevon positions but still performs with little to no oscillations.

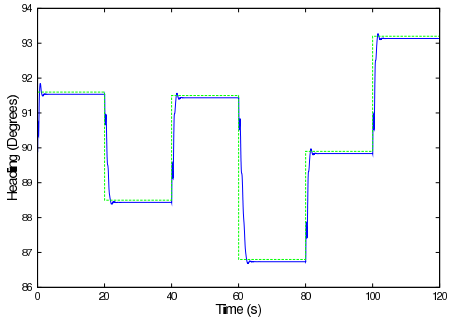
The elevon positions corresponding to the altitude changes for both neural network and PID controllers are shown in Figure 6. These graphs provide information on the amplitude and frequency of the controller's response, which is critical for understanding performance. Both controllers keep the elevon positions within ± 30 degrees while keeping oscillations to a minimum.

GENMAV's characteristics tend to pitch it up when the electric motor is on which makes it gain altitude. In the PID controller case, a constant trim value needs to be added to the elevator control input to keep the MAV flying at the desired altitude. This trim value was found by experimental trial and error. This necessitates additional tuning time before the MAV can fly correctly. In the neural network case, however, no trim constant is needed. Once the neural network is properly trained, no additional tuning or training is necessary to achieve good flight behavior. This is an advantage of the neural network implementation where the neural network adapts to the exact specifics of a platform and where tuning and adjustments are made automatically during training. This reduces the amount time required to achieve MAV flight capability which becomes invaluable when several different variations of a platform are considered.

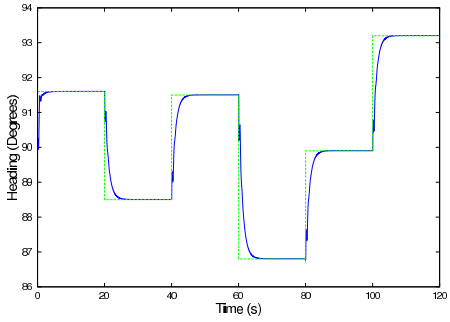
B. Heading Control

As mentioned in section IV-D, heading control is achieved with two cascaded controllers. The first one uses heading information to produce the desired roll angle while the second one uses the roll information to produce the aileron control input. Figure 7 shows the results for both neural network and PID control, with several random desired headings.

The heading control PID was tuned in an analogous fashion as the altitude control PID from Section V-A. Heading and roll control neural networks were trained using G_{Φ_1} and G_{Ψ} from Equations 2 and 3. Both controllers were able to track the desired heading closely while providing good system behavior. Both responses are fast with the desired heading reached within seconds. The neural network was once again more aggressive and reached its target a quicker than its PID counterpart, but with a minimal amount overshoot while the PID controller did not have any overshoot.



(a) Neuro-evolutionary Controller



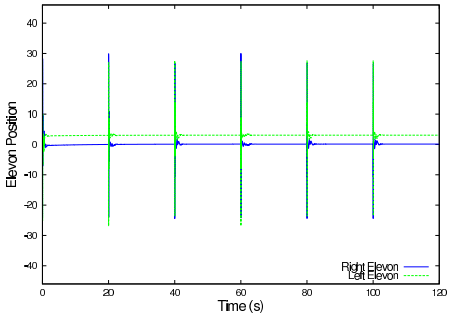
(b) PID controller

Fig. 7. Desired (dashed) and actual heading (solid): The performance is similar, but the neuro-controller has slightly faster response.

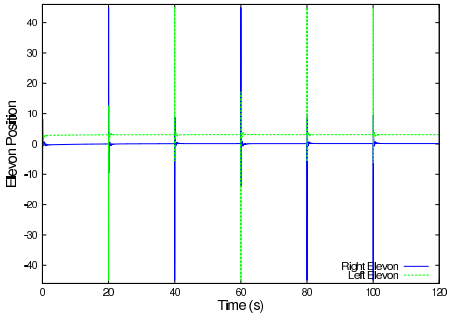
The elevon positions corresponding to the heading changes for both neural network and PID controllers are shown in Figure 8. Results here are alike except for the elevon angle range that the controllers use. Even though the heading/roll control neural networks are a more aggressive in trying to achieve their objective, the neural network controllers kept the elevon angle range within ± 30 degrees while the PID controllers used a wider range of elevon motion that is a little over ± 40 degrees. The difference is likely due to the speed and magnitude of the controller's response to a change in the error between the desired and actual heading.

Figures 8(a) and 8(b) show that the position of the left and right elevon is not at exactly zero in between the changes in altitude. This offset is necessary to compensate for the torque created by the electric motor so

that straight and level flight can be achieved. This is similar to the altitude control case where an altitude trim constant had to be added to the elevator control input from the PID controller. Since the heading and roll controllers are cascaded to achieve heading control, two different trim constants need to be added to the desired roll angle obtained with the heading control PID and to the aileron control input obtained from the roll control PID. The heading and roll neural network controllers automatically adjust and do not need tuning beyond the basic neural network training.



(a) Neuro-evolutionary controller



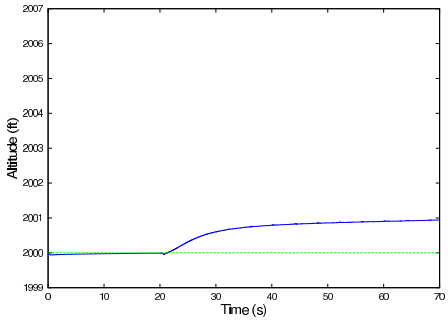
(b) PID controller

Fig. 8. Elevon Positions for heading control: The PID controller here needs large corrections (±45 degrees) to compensate its slow response, which is both wasteful and potentially dangerous.

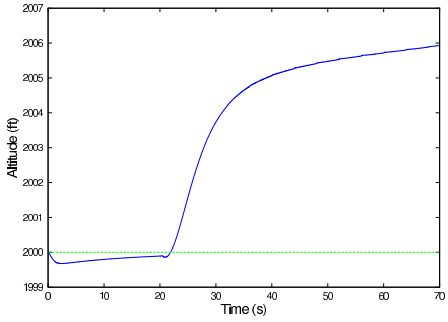
C. Response to Wind Gusts

Wind gusts were created at 4 second intervals with the intensity of the wind gust increasing by 10% at every step. The objective of the controllers was to keep the MAV altitude and heading as close as possible to the constant desired values which were in this case 2000 feet for the altitude and 90 degrees for the heading. Wind gusts start after 20 seconds of normal flight at an intensity of 2 m/s. Results for maintaining the altitude constant are shown in Figure 9 where the dashed line represents the desired altitude. Here, the neural network control system was trained as described in Sections V-A and V-B using Equations 2.

The neural network and PID controller altitude curves have a similar shape, the main difference is in the altitude value itself. The neural network control system is able to maintain the MAV altitude within 1 foot from the desired altitude, while the PID control system



(a) Neuro-evolutionary controller



(b) PID controller

Fig. 9. Desired and actual altitude: Wind gusts start at $t = 20s$ and repeat with increasing frequency every 4 seconds. The neuro-controller achieves 6 times better performance than the PID controller.

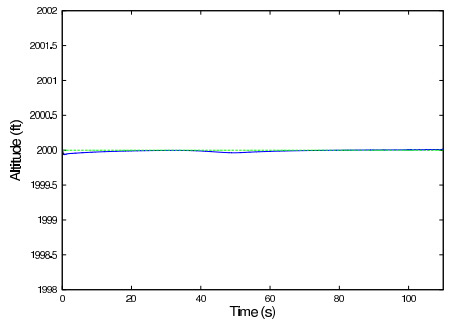
can only maintain it within about 6 feet. The neural network control system was more robust to wind gusts, demonstrating its effectiveness at stabilizing the MAV.

One can argue that similar results could be obtained by tuning the PID control system differently and in similar conditions as what has been used for training the neural network control system. It is in theory possible to do this, but it would require not only an accurate model of the wind gusts, but significant amounts of experimenting and tuning. The neural network control system required several thousand simulation runs to achieve these results but the changes to the controller were done automatically using the neuro-evolutionary algorithm. No additional modeling or knowledge of the wind gusts were required.

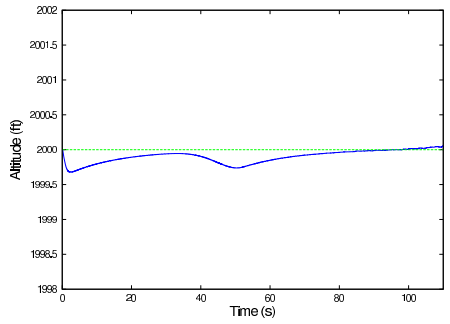
D. Response to Turbulence

Experiments including turbulences were also conducted. Turbulence is simulated on JSBSim as a set of external forces and moments on the vehicle, based on randomly parametrized sinusoidal functions. In these experiments, the amplitude of the forces acting on the MAV varied randomly between 1 and 5 % of the craft's airspeed. Results showing the MAV altitude and heading with increasing turbulences for both the neural network and PID controllers are presented in Figure 10 and 11.

Altitude remains fairly constant throughout the experiment and no significant difference is seen between the neural network and PID controllers. Figures 10(a) and 10(b) show the altitude within half a foot of its desired

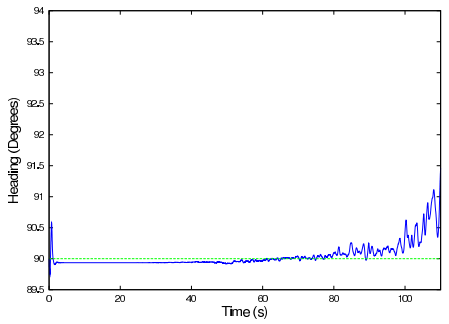


(a) Neuro-evolutionary controller

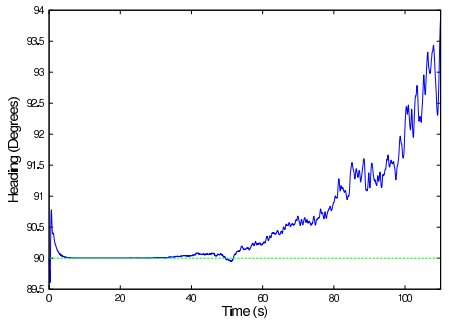


(b) PID controller

Fig. 10. Desired and actual altitude: Turbulence is added randomly and increases throughout the simulation. The neuro-controller is nearly unaffected whereas the PID controller suffers significantly.



(a) Neuro-evolutionary Controller



(b) PID controller

Fig. 11. Desired and actual heading: Turbulence is added randomly and increases throughout the simulation. The neuro-controller maintains course longer but is eventually knocked off-course. The PID controller goes off-course sooner and cannot recover.

value for both neural network and PID controllers. The small variations are not relevant and altitude can be assumed near constant for both controllers.

A difference is however visible for heading control where neural network controllers were able to remain closer to the desired heading. Figure 11(a) shows that the heading stays within about a degree from the desired heading for the neural network controllers while Figure 11(b) shows a difference of about three degrees. This difference between the two control systems is not large, but could make a difference depending on the application.

E. Multiagent Control of Segmented Surfaces

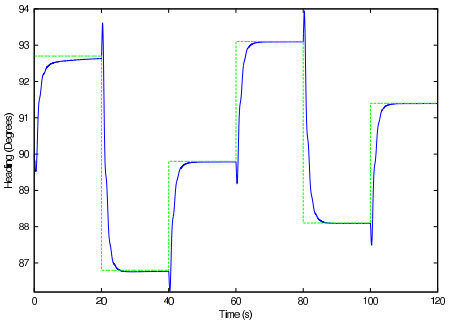
In this section, we present the results on segmented control surfaces. In these experiments, the multiagent controller uses one neural network for each control surface and the "single agent" controller uses a single neural network to control multiple control surfaces (as described in Section IV-D3. Figure 12 shows the performance of the multiagent control of segmented ailerons, and Figure 13 shows the corresponding aileron positions. Each control surface moves independently in the case of the multiagent controller while control surfaces move in a symmetric fashion for the single neural network controller. Benefits of control surface segmentation is not apparent when these results are compared to those in Sections V-A and V-B due to the simplicity of the task. In the next section, we discuss the impact of multiagent control in cases of actuator failures.

F. Actuator Failure

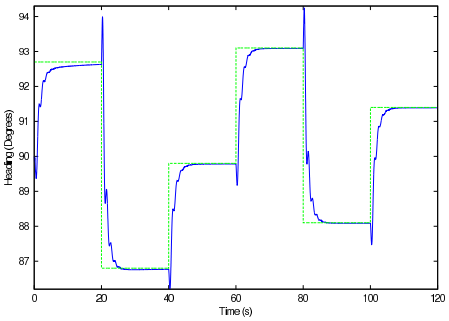
Actuator failures are not expected to happen on every flight but the risk is however there and failures caused by mechanical or electrical breakdown due to bad environmental conditions or factory defects can still occur. Some MAV missions can be of critical importance where failure is not an option and could mean the difference between life and death. For such missions, it is essential that the MAV platform is able to recover from potential failures. Results in this section show different failures of an actuator for different models: the standard system controlled by a PID controller and the segmented aileron model controlled by a single neural network as well as a multiagent controller.

1) Response to Actuator Stuck in Benign Position: Figure 14 shows the altitude and heading for the multiagent controller and the heading for the single agent and PID controllers, when failure 1 occurs. Failure 1 corresponds to actuator 4 on the left side of the MAV failing and remaining stuck at around 5 degrees. The flexibility of the multiagent system allows it to adapt and reconfigure itself so that the control objective is achieved. In this case the multiagent controller was still able to track the desired heading that was generated randomly every 20 seconds (Figure 14(a)). It is important to note that the altitude control is not affected by the aileron failure since the controllers and control surfaces are independent (Figure 14(d)).

The control response is not as smooth as before when

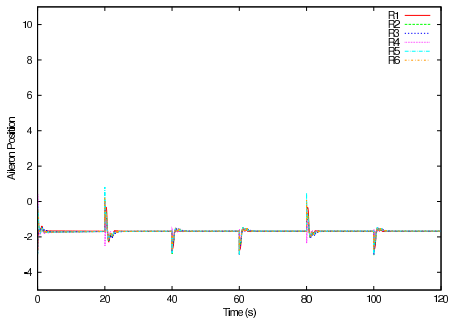


(a) Multiagent controller

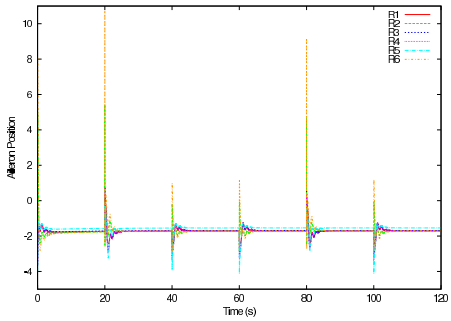


(b) Single agent controller

Fig. 12. Desired and actual heading for segmented surfaces. In the basic case, single vs. multiagent control give similar results.



(a) Multiagent controller



(b) Single agent controller

Fig. 13. Aileron position is segmented control surfaces. Even though performance was similar the multiagent controller achieves this with much smaller corrections than the single agent controller.

all actuators were working correctly but the target value is achieved, the behavior of the system is good and the performance is significantly better than what was obtained using the other controllers/configurations. Figures 14(b)-14(c) show the desired and actual heading for the single neural network controller paired with the segmented aileron model and the PID controller. The single agent still performs adequately (within half a degree) but the PID controller has difficulty remaining within 2 degrees of the desired heading.

2) Response to Actuator Stuck in Difficult Position: A different failure scenario was also tested where the failed actuator is the same as in failure 1 but the failure angle is around -5 degrees. This new failed position creates bigger differences between the controllers and the benefits of the multiagent system combined with the aileron segmentation becomes even more apparent.

Figure 15 shows the desired and actual heading of the MAV, the corresponding aileron positions, as well as the heading of the single agent and PID controllers. As discussed previously, the control response is not as smooth when a failure is present but once again, the multiagent controller achieves good target heading and acceptable system demonstrate a good behavior. The heading error is however much more pronounced this time for the single neural network and PID controllers. The error is over a degree for the neural network controller while it is now close to 5 degrees for the PID controller.

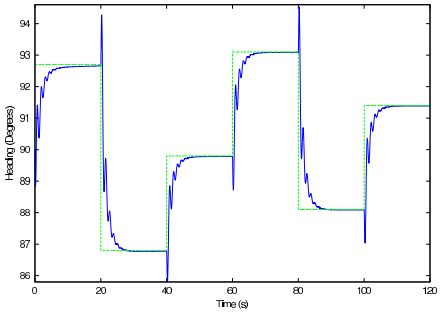
3) Recovery Range from stuck Actuator: Finally, a comparison between the PID and multiagent controllers was done with a wide range of failure positions. Figure 16 shows the heading error plotted as a function of the failure angle for both, the PID and multiagent controllers.

Unless the failed actuator angle remains around 2 degrees which is its normal position for straight and level flight, the failure affects the PID controller for all possible angles with the heading error increasing with the higher angles of actuator failure. The last 2 angles of failure of -18 and -20 degrees are not shown on the graph for the PID controller because the system becomes unstable in these cases, which is partly due to significant drag and reduced velocity created by the relatively large aileron deflections.

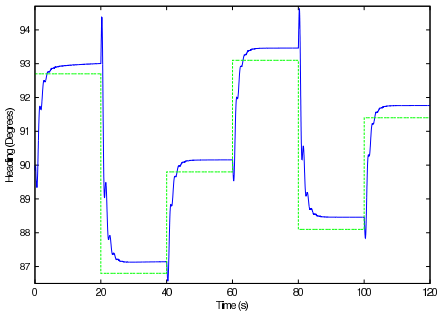
The multiagent controller performs much better and remains unaffected by actuator failures that remain within ± 10 degrees of position for straight and level flight. Within this boundary, the multiagent control system performs up to 8 times better than the PID controller. Beyond this limit, the heading error is still kept within the reasonable values of 0 to 4 degrees which is still a minimum of 4 times better than the PID controller.

VI. Discussion

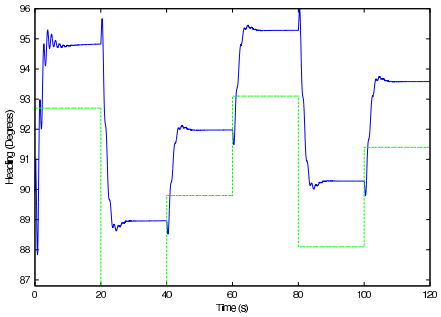
Micro Aerial Vehicles present a new and encouraging platform for collecting information in new and in some cases previously inaccessible environments. Yet, they typically present a challenging control problem which limits their applicability to the domains in which they are the most needed (e.g., dangerous search and rescue or reconnaissance). This paper presents a novel approach to the MAV control problem and provides improvements



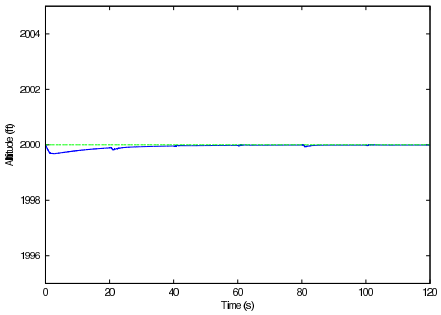
(a) Heading multiagent controller



(b) Heading single agent controller

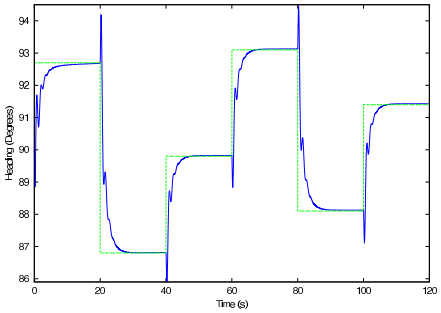


(c) Heading PID controller

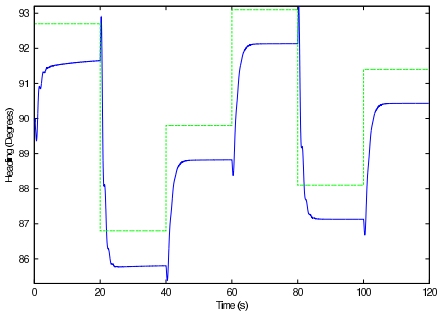


(d) Altitude multiagent controller

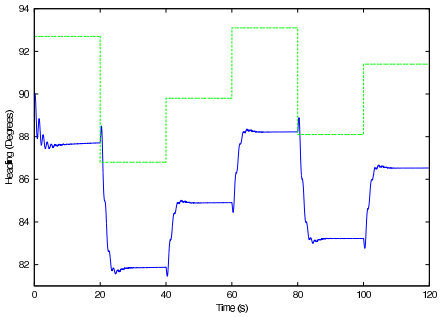
Fig. 14. Desired and actual heading (Failure 1). The multiagent controller performs significantly better than the single agent controller in this difficult task. Both learning algorithms outperform the PID controller that is unable to control the MAV in this failure mode. Altitude control is also shown for the multiagent controller.



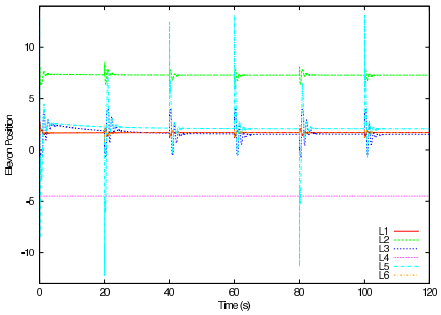
(a) Heading multiagent controller



(b) heading single agent controller (Failure 2)



(c) Heading PID controller (Failure 2)



(d) Left Aileron positions multiagent controller

Fig. 15. Desired and actual heading (Failure 2). The multiagent controller significantly outperforms both the single agent controller and the PID controller in this difficult task, and both learning algorithms outperform the PID controller in this failure mode. Aileron positions corresponding to multiagent control (Fig 15(a)) are also shown

of the flight characteristics of such platform by introducing a larger number of control surfaces on the aileron section. Robustness to actuator failure is also added to the platform and allows the MAV to stay in flight and perform maneuvers with up to two actuator failures.

Sections V-A and V-B showed that training a neural network controller on an MAV with a neuroevolutionary algorithm was possible and required less tuning than a PID controller. Additional trim constants were also not necessary for the neural network controllers as these controllers automatically adapted to the specifics of the platform. The neural network controllers

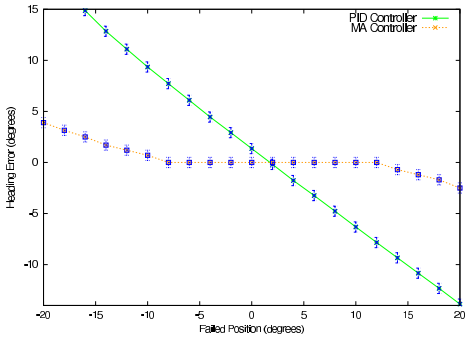


Fig. 16. Heading error vs failure angle: The PID heading error linearly correlates to the failed position of the elevon. The multiagent controller on the other hand compensates for a wide variety of failed positions and gracefully degrades after the failed position exceeds $\pm$ 10 degrees.

were also ready to use right after training without any further adjustments. Unlike model based control methods, no model analysis was needed to create these controllers which gives great flexibility in the implementation as they are not platform specific and could automatically reconfigure themselves to the particularities of a different platform.

Section V-C presented a first set of experiments where the neural network controllers were trained beyond the basic flight objectives to see if they could adapt to harder flight conditions and improve MAV robustness when wind gusts are present. Results were encouraging and showed improvements in maintaining the target altitude when using a slightly more complex objective function during training. Experiments with increasing levels of turbulence were also presented and showed better performance of the neural network controllers that were able to stay closer to the desired heading. This flexibility provides an important advantage as MAV control can be improved and custom made for a particular platform or known and unknown environmental conditions without requiring any significant amount of tuning as long as the objective function is designed correctly.

This work can be extended in two broad directions. First, the results shown in this study can be implemented in hardware. This transfer of simulation results to hardware is a difficult process in general, as the differences between simulation and hardware cause many unexpected failures. The selected approach, though, is naturally suited to handle such a transfer because it does rely on models. As such, it is less sensitive to differences betweens simulated and real environmental conditions. Indeed, our earlier work on the transfer of navigation algorithm from simulation to hardware for wheeled robots showed that minimal tuning was necessary to the algorithms [46]. The key issues however are in ensuring sensor and actuator functionality within limited power and weight restrictions.

Second, we can extend this work by conceptually moving towards a "morphing" wing design by significantly increasing the number of control surfaces. The results presented in this paper are a first step that shows the potential of leveraging multiagent based methods to improve MAV control implementation, performance, and robustness. The controllers presented in this paper only use the aileron and elevator controls. Adding openings of various sizes, shapes, and locations on the wings of MAVs have also been shown to improve the robustness of the vehicle to wind gusts and perturbation. A multiagent approach to selecting the location, size and shape of those openings seems to be a natural match to yield truly "morphing" micro aerial vehicles.

Acknowledgements: The authors would like to thank Chris Holmes-Parker for his help editing , Megan Colbath for the CAD model of the segmented control surface MAV, and Kelly Stewart for her help with the GENMAV configuration. This work was partially supported by NSF grant 0910358 and AFOSR grant FA9550-08-1-0187 .

REFERENCES

[1] N. Nigam and I. Kroo, "Persistent surveillance using multiple unmanned air vehicles," *IEEE*, 2008.
[2] J. Hall, D. Lawrence, and K. Mohseni, "Lateral control and observation of a micro aerial vehicle," in *45th AIAA Aerospace Sciences Meeting and Exhibit*, 2007.
[3] R. Kulkarni and G. Venayagamoorthy, "Bio-inspired algorithms for autonomous deployment and localization of sensor nodes," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 40, no. 6, pp. 663–675, 2010.
[4] P. Dasgupta, "A multiagent swarming system for distributed automatic target recognition using unmanned aerial vehicles," *IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans*, vol. 38, no. 3, pp. 549–563, 2008.
[5] P. Scerri, R. Glinton, S. Owens, D. Scerri, and K. Sycara, "Geolocation of RF emitters by many uavs," in *AIAA Infotech@Aerospace 2007 Conference and Exhibit*, 2007.
[6] D. A. Jenkins, P. G. Ifju, M. Abdulrahim, and S. Olipra, "Assessement of controllability of micro air vehicles," Proc. Sixteenth Int. Conf. On Unmanned Air Vehicle Systems, 2001.
[7] M. R. Waszak, J. B. Davidson, and P. G. Ifju, "Simulation and flight control of an aeroelastic fixed wing micro aerial vehicle," in *AIAA Atmospheric Flight Mechanics Conference and Exhibit*, 2002.
[8] P. Ifju, D. A. Jenkins, S. Ettinger, Y. Lian, and W. Shyy, "Flexible-wing-based micro air vehicles," in *40th AIAA Aerospace Sciences Meeting & Exhibit*, 2002.
[9] A. M. DeLuca, M. F. Reeder, M. V. OL, J. Freeman, I. Bautista, and M. Simonich, "Experimental investigation into the aerodynamic properties of a flexible and rigid wing micro air vehicle," in *24th AIAA Aerodynamic Measurement Technology and Ground Testing Conference*, 2004.
[10] R. Krashanitsa, G. Platanitis, B. Silin, and S. Shkarayev, "Aerodynamics and controls design for autonomous micro air vehicles," in *AIAA Atmospheric Flight Mechanics Conference and Exhibit*, 2006.
[11] W. J. Pisano, D. A. Lawrence, and P. C. Gray, "Autonomous uav control using a 3-sensor autopilot," in *AIAA Conference and Exhibit*, 2007.
[12] H. Garcia, M. Abdulrahim, and R. Lind, "Roll control for a micro air vehicle using active wing morphing," in *AIAA Guidance, Navigation and Control Conference*, 2003.
[13] M. Abdulrahim and J. Cocquyt, "Development of mission capable flexible-wing micro air vehicle," in *53rd Southeastern Regional Student Conference*, 2002.
[14] T. Kordes, M. Buschmann, S. Winkler, H.-W. Schulz, and P. Vorsmann, "Progresses in the development of the fully autonomous MAV CAROLO," in *2nd AIAA Unmanned Unlimited Systems, Technologies, and Operations Aerospace*, 2003.
[15] R. K. Arning and S. Sassen, "Flight control of micro aerial vehicles," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, 2004.

[16] K. Stewart, J. Wagener, G. Abate, and M. Salichon, "Design of the air force research laboratory micro aerial vehicle research configuration," in *45th AIAA Aerospace Sciences Meeting and Exhibit*, 2007.
[17] J. Valasek, J. Doebbler, M. Tandale, and A. Meade, "Improved adaptivereinforcement learning control for morphing unmanned air vehicles," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 38, no. 4, pp. 1014–1020, 2008.
[18] M. Abdulrahim and R. Lind, "Investigating segmented trailing-edge surfaces for full authority control of a UAV," in *AIAA Atmospheric Flight Mechanics Conference*, 2003.
[19] J. Hall, D. Lawrence, and K. Mohseni, "Lateral control of a tailless micro aerial vehicle," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, 2006.
[20] G. Platanitis and S. Shkarayev, "Integration of an autopilot for a micro air vehicle," in *AIAA*, 2005.
[21] W. Guo and J. F. Horn, "Modeling and simulation for the development of a quadrotor uav capable of indoor flight," in *AIAA Modeling and Simulation Technologies Conference and Exhibit*, 2006.
[22] S. Winkler, M. Buschmann, L. Kruger, H.-W. Schulz, , and P. Vorsmann, "State estimation by multi-sensor fusion for autonomous mini and micro aerial vehicles," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, 2005.
[23] J. Young and A. R. Price, "FPGA based uav flight controller," in *11 Eleventh Australian International Aerospace Congress (AIAC)*, 2005.
[24] P. Y. Oh, W. E. Green, and G. Barrows, "Neural nets and optic flow for autonomous micro-air-vehicle navigation," in *ASME International Mechanical Engineering Congress and Exposition*, 2004.
[25] F. Pasemann, "Evolving neurocontrollers for balancing an inverted pendulum," in *Computation in Neural Systems*, 1998, pp. 495–511.
[26] M. A. Marra, B. E. Boling, and B. L. Walcott, "Genetic control of a ball beam system," in *IEEE international Conference on Control Applications*, 1996.
[27] N. Bredeche, Z. Shi, and J.-D. Zucker, "Perceptual learning and abstraction in machine learning: an application to autonomous robotics," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 36, no. 2, pp. 172–181, 2006.
[28] J. Shepherd III and K. Tumer, "Robust neuro-control for a micro quadrotor," in *Proceedings of the Genetic and Evolutionary Computation Conference*, Portland, OR, July 2010, pp. 1131–1138.
[29] M. Salichon and K. Tumer, "A neuro-evolutionary approach to micro aerial vehicle control," in *Proceedings of the Genetic and Evolutionary Computation Conference*, Portland, OR, July 2010, pp. 1123–1130.
[30] J. Chen and M. Barnes, M.J.and Harper-Sciarini, "Supervisory control of multiple robots: Human-performance issues and user-interface design," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 41, no. 4, pp. 435–454, 2011.
[31] Y. Jin, Y. Liao, A. Minai, and M. Polycarpou, "Balancing search and target response in cooperative unmanned aerial vehicle (UAV) teams," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 36, no. 3, pp. 571–587, 2005.
[32] A. Agogino and K. Tumer, "Efficient evaluation functions for multi-rover systems," in *The Genetic and Evolutionary Computation Conference*, Seatle, WA, June 2004, pp. 1–12.
[33] K. Tumer and A. Agogino, "Coordinating multi-rover systems: Evaluation functions for dynamic and noisy environments," in *The Genetic and Evolutionary Computation Conference*, Washington, DC, June 2005.
[34] S. R. Bieniawski, "Distributed optimization and flight control using collectives," Ph.D. dissertation, Stanford University, 2005.
[35] S. R. Bieniawski, I. Kroo, and D. Wolpert., "Flight control with distributed effectors," in *AIAA Guidance, Navigation, and Control Conference, San Francisco, CA, August 15-18*, 2005.
[36] K. Tumer and D. Wolpert, Eds., *Collectives and the Design of Complex Systems*. New York: Springer, 2004.
[37] D. H. Wolpert and K. Tumer, "Optimal payoff functions for members of collectives," *Advances in Complex Systems*, vol. 4, no. 2/3, pp. 265–279, 2001.
[38] M. Drela and H. Youngren, "Athena vortex lattice (AVL)," 2008. [Online]. Available: http://web.mit.edu/drela/Public/web/avl/
[39] J. Becker, *Creating Vortex Lattice Aircraft Models for the Piccolo Simulator with AVL*, Cloud Cap Technology, 2621 Wasco Street, Hood River, OR 97031, March 2008.
[40] D. M. Richwine and J. H. D. Frate, "Development of a low-aspect ratio fin for flight research experiments," NASA, Tech. Rep., 1994.
[41] E. G. Garcia and J. Becker, "Uav stability derivatives estimation for hardware-in-the-loop simulation of piccolo autopilot by qualitative flight testing," in *1st Latin American UAV Conference*, 2007.
[42] F. Gomez and R. Miikkulainen, "Active guidance for a finless rocket through neuroevolution," in *Proceedings of the Genetic and Evolutionary Computation Conference*, Chicago, Illinois, 2003.
[43] D. Moriarty and R. Miikkulainen, "Forming neural networks through efficient and adaptive coevolution," *Evolutionary Computation*, vol. 5, pp. 373–399, 2002.
[44] C. M. Bishop, *Neural Networks for Pattern Recognition*. Oxford University Press, November 1995.
[45] J. S. Berndt, "Jsbsim: An open source flight dynamics model in C++," in *AIAA Modeling and Simulation Technologies Conference and Exhibit*, 2004.
[46] M. Knudson and K. Tumer, "Adaptive navigation for autonomous robots," *Robotics and Autonomous Systems*, vol. 59, pp. 410–420, June 2011.