A Neuro-Evolutionary Approach to Micro Aerial Vehicle Control

Max Salichon
Oregon State University
204 Rogers Hall
Corvallis, Oregon 97331-6001
max.salichom@gmail.com

Kagan Tumer
Oregon State University
204 Rogers Hall
Corvallis, Oregon 97331-6001
kagan.tumer@oregonstate.edu

ABSTRACT

Applying classical control methods to Micro Aerial Vehicles (MAVs) is a difficult process due to the complexity of the control laws with fast and highly non-linear dynamics. Such methods rely heavily on difficult to obtain models and are particularly ill-suited to the stochastic and dynamic environments in which MAVs operate. Instead, in this paper, we focus on a neuro-evolutionary method that learns to map MAV states (position, velocity) to MAV actions (e.g., actuator position). Our results show significant improvements in response times to minor altitude and heading corrections over a traditional PID controller. In addition, we show that the MAV response to maintaining altitude in the presence of wind gusts improves by a factor of five. Similarly, we show that the MAV response to maintaining heading in the presence of turbulence improves by factors of three.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning

General Terms

Algorithms, Experimentation

Keywords

Learning, Evolution, Adaptation, Control

1. INTRODUCTION

Micro Air Vehicles (MAVs) have recently seen more attention due to the large number of missions and tasks that they can accomplish such as surveillance [16], reconnaissance, sensing [11], and search and rescue. MAVs can accomplish such demanding missions without endangering human lives, making them highly desirable. Indeed, a wide variety of MAV platforms and control strategies have been studied and show promising results in this area [12, 13, 19].

GECCO'10, July 7–11, 2010, Portland, Oregon, USA.

Unlike the more frequently used Unmanned Aerial Vehicle (UAV) platforms, MAVs typically range from insect to bird size. Most accepted definitions of MAVs place their wing size to within 15 to 60 cm (6 to 24 inches). The flight speed of those MAVs is on average between 5 and 20 m/s (10 to 50 mph). MAVs must have a high maneuverability and an accurate control system to be able to operate at low altitude, around buildings and obstacles, and where wind and gusts are present [4, 21]. As a consequence MAVs present a number of challenges: limited processing power, limited control surfaces and actuators, limited number and quality of the sensors, and limited power available. Furthermore, because of their small size, they are particularly difficult to control and highly susceptible to external disturbances such as wind gusts and turbulence.

Using adaptive control methods, rather than more traditional methods (e.g., model-based) offer an appealing alternative to MAV control. In particular, neuro-evolutionary approaches where a neural network is evolved to approximate a mapping from sensor readings to actuator outputs offer great promise [2, 10, 15, 24]. Evolutionary computation techniques which can be leveraged to learn such mappings have been used successfully to solve benchmark control problems including the inverted pendulum [18] and the ball and beam [14] problems. These techniques have also proven to be robust and efficient for complex control problems such as multi-rover control problems [3] where a large number of agents have to achieve desired behavior at the system level, while also reaching individual behavior targets.

In this paper, we show that an evolutionary algorithm can be used to implement a neuro-controller on different MAV platforms. We also show that this neuro-controller can increase the robustness of the platform to wind gusts and turbulence by adapting to unknown environments. Section 3 describes the platform and simulation of the system dynamics. Section 4 describes the algorithm design, the evaluation functions used for training the controllers and the experimental setup. Section 5 presents the experimental results where the neuro-evolutionary algorithm's altitude and heading control are compared with a Proportional, Integral, Derivative (PID) controller [13]. In addition it presents results showing the improved robustness of the neuro-controller which can maintain a target altitude in the presence of significantly higher wind gusts as compared to a PID controller. Better heading tracking is also achieved with significantly higher levels of turbulence. Finally, Section 6 discusses the relevance of the results and highlights directions for future work.

2. RELATED WORK

Traditional control techniques such as model-based PID control have been used successfully for many control problems including aircraft control [11, 20, 19, 13, 17]. These model based techniques perform very well for full size aircrafts where linear approximations of the system's dynamics produce good results. Micro Aerial Vehicle size renders them particularly susceptible to wind gusts and disturbances. These conditions make them notoriously difficult to control since many of the assumptions used in the controllers break down in such environments. Accurately modeling the MAV and its environment is also critical to design model based controllers which are typically not very robust to modeling inaccuracies. Moreover, PIDs typically involve tuning and optimization of the gains to achieve optimal results. This step can however be improved using neural networks [17]. Improvements can therefore be achieved by providing more flexible controllers that can adapt automatically to model error and changes in the environment.

Learning based techniques are flexible, do not require a model of the system and can adapt to different platforms and dynamic environments. These techniques have been proven to be robust and efficient for complex control problems such as multi-rover control problems [3] where a large number of agents have to maximize the overall system level evaluation function that rates the performance of the full systems as well as their own evaluation functions. They are therefore well suited for MAV control where the system is highly non-linear, unstable and where obtaining an accurate model of the system is not a trivial task. The configuration of these platforms is flexible and provide many parameters that can be used by the learning based control system to stabilize the system and perform flight maneuvers. Another benefit of such a system would be its ability to control the system when noise and/or failures occur and adapt to the new structure of the system. A difficulty in applying a learning technique is the design of the evaluation function so that the correct mapping between MAV states (position, velocity) and MAV actions (actuator positions) can be achieved. Tuning of the learning parameters is also necessary in order to achieve optimal results. Sections 5.1, 5.2, and 5.3 show the results of learning based methods applied to the MAV control problem.

3. MAV PLATFORM

The platform selected for these experiments is GENMAV [21], an MAV developed by the Air Force Research Laboratory Munition Directorate (AFRL/RW). The system dynamics are simulated using JSBSim as described below.

3.1 MAV Platform: GENMAV

GENMAV was developed to provide a base configuration that researchers could use and modify when implementing design and/or control techniques. GENMAV is also a flexible platform that could be modified depending on a particular application or technology.

Characteristics of GENMAV 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 tail section uses elevons for control. Elevons are the two independent control surfaces of the horizontal tail stabilizer and are used in place of the standard ailerons and elevator control surfaces for both roll and pitch control. Aerodynamic characteristics were obtained using the vortex-lattice method aeroprediction code AVL (Athena Vortex Lattice) and detailed data can be found in [21]. Similar to other MAV platforms, GENMAV was designed for a flight speed of between 10 and 50 mph with an average flight speed around 30mph.

3.2 System Dynamics: JSBSim

To conduct the experiments, the system dynamics were simulated using JSBSim [6], 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 collection of program code mostly written in the C++ programming language, but some C language routines are included. Some of the C++ classes that comprise JSBSim model physical entities such as the atmosphere, a flight control system, or an engine. Taken 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 system evaluation function, and having the neural network learn the control laws from the evaluation function. Figure 2 shows the overall control system setup.

The JSBSim model of GENMAV was obtained by computing the aerodynamic coefficients using the Athena Vortex Lattice software package (AVL)[9, 5]. AVL uses a file containing the geometry of Genmav where wing profile and control surfaces are defined. Flow around the GENMAV model made of flat vortex panels is then simulated using 3D vortex lattice methods. The JSBSim GENMAV model is then loaded at the beginning of the simulation and used for the different experiments.

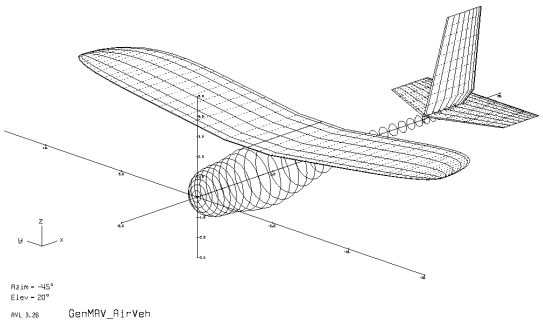


Figure 1: GENMAV in AVL

4. NEURO-CONTROL FOR MAVS

The control of GENMAV is achieved through a feed-forward neural network trained using an evolutionary algorithm [3, 15, 22]. The neural network learns the correct control laws through the system evaluation function that is designed to

minimize the error between the desired parameter value and the actual parameter value as well as keep the control inputs within acceptable values using the parameter's derivative. The best control law found in this process is then saved and used for flight control where different desired altitudes and headings are achieved.

4.1 Neuro-Evolutionary Control

A simple neuro-evolutionary algorithm was developed using the techniques outlined in [22]. The algorithm maintains an initially empty pool of controllers that are paired with some measure of their utility. While the pool is not full, the algorithm generates new random controllers 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 controllers and selectively modifies the chosen controller using a different Cauchy distribution. In both cases, the new controller replaces the poorest performing controller in the pool only after it has been used and its performance has been evaluated.

The single hidden-layer, feed forward neural networks [8] used as controllers 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: altitude control, roll control, and heading control. Each neural network output provides the elevator command, aileron command, and desired roll angle. The elevator and aileron commands were then summed to get the final elevon positions that range from -45 to +45 degrees.

4.2 Evaluation Functions

An important part of using neural networks consists of designing an evaluation function that allows the controllers to learn at a satisfactory rate and at the same time achieves the desired system characteristics. The evaluation functions used for these experiments are designed to minimize the error between the control parameter (altitude, roll, and heading) and its desired value.

Three different controllers were created and they each use their own evaluation functions that are specifically designed for that particular controller. The three controllers are for altitude, roll, and heading control. Their respective evaluation functions are G_Z for the altitude controller, G_{Φ_1} and G_{Φ_2} for the roll controller and G_Ψ for the heading controller. The more complicated evaluation function G_{Φ_2} was used to improve the system response when wind gusts and perturbations were present.

Evaluation Function for Altitude Control: G_Z

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

$$G_Z = \frac{\alpha_z}{\sum_{t=0}^{T} [Z_d - Z_a]} \tag{1}$$

Where α_z is an arbitrary constant, Z_d and Z_a are the desired and actual altitude, and T is the simulation time.

Evaluation Functions for Roll Control: G_{Φ_1} and G_{Φ_2}

Similarly, G_{Φ_1} and G_{Φ_2} were designed to minimize the error between the desired roll and the actual roll.

$$G_{\Phi_1} = \frac{\alpha_{\Phi_1}}{\sum_{t=0}^{T} [\Phi_d - \Phi_a]} \tag{2}$$

G_{Φ_2} was designed to improve the system response in presence of wind gusts and perturbations by including the roll derivative to the evaluation function.

$$G_{\Phi_2} = \frac{\alpha_{\Phi_2}}{\sum_{t=0}^{T} \left[(\Phi_d - \Phi_a) + \frac{d\Phi}{t} \right]} \tag{3}$$

Where α_{Φ_1} and α_{Φ_2} are arbitrary constants, Φ_d and Φ_a are the desired and actual roll, and T is the simulation time.

Evaluation Function for Heading Control: G_Ψ

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

$$G_\Psi = \frac{\alpha_\Psi}{\sum_{t=0}^{T} [\Psi_d - \Psi_a]} \tag{4}$$

Where α_Ψ is an arbitrary constant, Ψ_d and Ψ_a are the desired and actual heading, and T is the simulation time.

4.3 MAV Controller

The MAV Proportional Integral Derivative (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 altitude and actual altitude as well as altitude rate for its inputs, and it outputs the elevator command. Similarly, the roll and heading PID controller 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.

The PID control command is calculated with three separate parameters: the proportional, the 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 called PID gains.

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

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

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

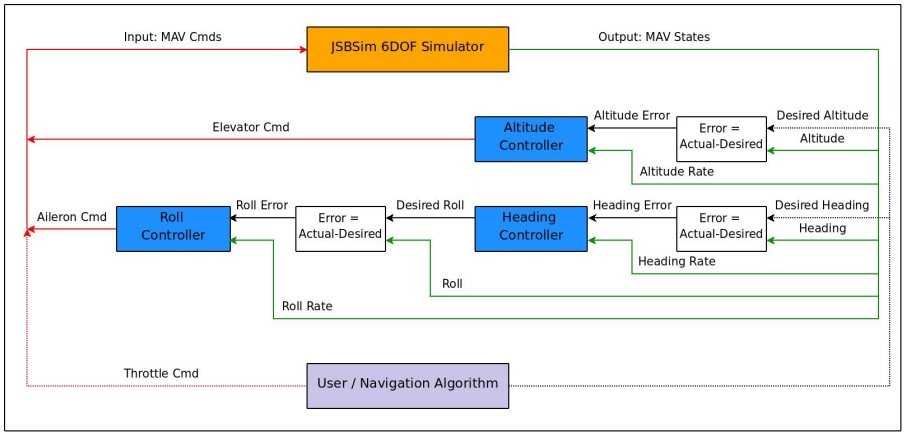


Figure 2: Controller

Where K_P, K_I, and K_D are constants (PID gains), and $e(t)$ is the error.

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 both 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.

The neuro-controllers are developed using the same inputs and outputs as the PID controllers. Modifying the controllers is achieved using the different evaluation function from Section 4.2. The training cycle for one controller is as follows: the evolutionary algorithm selects and mutates a controller from the pool, this controller is then used in the system for a short period of time, using state information from JSBSim as input and outputting a control command to the simulator. The evaluation function is then used to calculate the fitness that the controller receives based on its flight performance. The algorithm then repeats this cycle until a near optimal solution is found. That controller is then saved and used directly in the control loop with JSBSim.

5. EXPERIMENTAL RESULTS

In these experiments, 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. These parameters were then kept constant for the experiments described in section 5. Altitude, roll, and heading neural network controllers were trained using the evaluation functions from Section 4.2. Training time for the altitude and roll controllers was 5 seconds while training time for the heading controller was 12 seconds.

5.1 Altitude Control

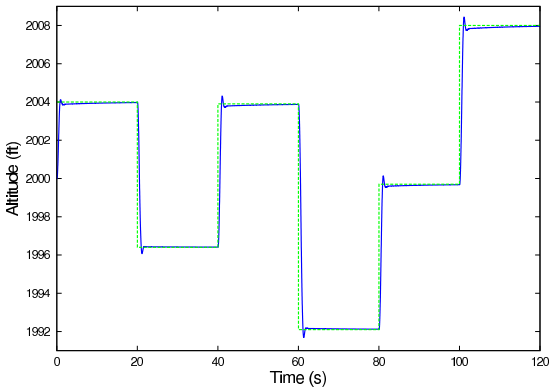
Figures 3 and 4 show the results for arbitrary desired altitude profiles, and compare the performance of the neural network and PID controllers. The altitudes are shown in Figure 3 and the altitude rates in Figure 4.

The altitude control PID was fairly straightforward to implement but required some tuning to achieve the desired results which is common for PID controllers. Figure 3(b) shows the MAV altitude with the dashed line representing the desired altitude and the solid line representing the actual altitude. The controller is able to track the desired altitude well with no overshoot. The MAV gets within a foot of the target altitude in a few seconds and trying to improve it further does not yield a very good behavior. Therefore, the PID altitude gains producing these results were kept for the duration of these experiments.

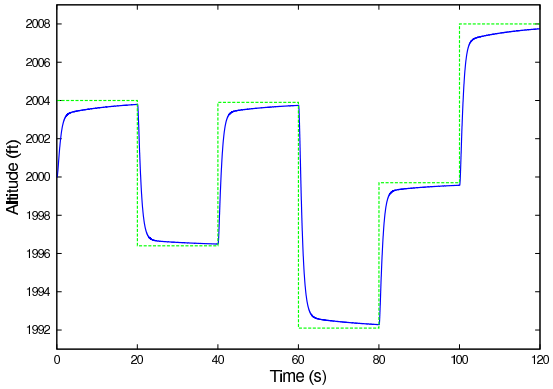
Figure 3(a) shows MAV altitude when the neural networks are in control with the same target altitudes as when the system is under PID control. Evaluation function G_Z from Equation 1 was used in the training of the neural network. Looking at the altitude from Figure 3(a), the altitude control neural network performs very well. The target altitude is reached very quickly and efficiently with only a very minimal overshoot. The neural network's behavior is a more aggressive than the PID's behavior which allows for better and faster tracking of the desired altitude without compromising the behavior of the system and without creating instabilities in the system.

The altitude rate for both neural network and PID controllers is shown in Figure 4. This provides additional information regarding the behavior of the system and is very 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 4(b)) while the neural network keeps it within 16ft/sec (Figure 4(a)) which is still within the range of values providing a good behavior of the system. Both controllers also keep the elevon positions within ± 30 degrees while keeping oscillations to a minimum. The difference between the two controllers is explained by the fact that the neural network is a more aggressive controller.

The altitude neural network and PID controllers provide good authority over GENMAV and produce good flight behavior. One interesting difference in the implementation of the controllers is the constant trim value. GENMAV's char-

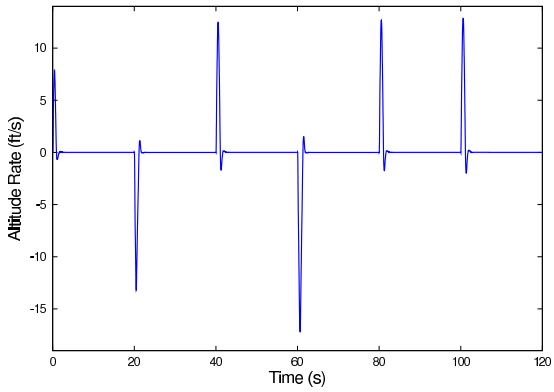


(a) NN controller

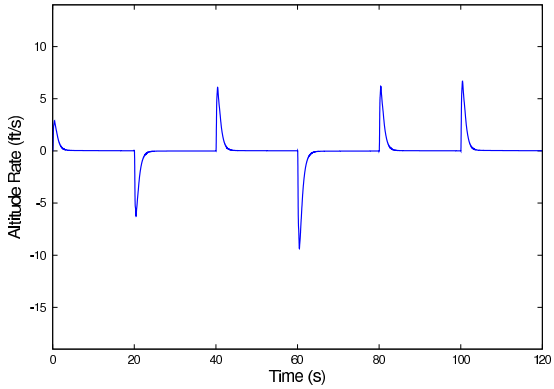


(b) PID controller

Figure 3: Desired and actual altitude



(a) NN controller



(b) PID controller

Figure 4: Altitude rate

acteristics 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 experimenting and trial and error until the correct behavior was achieved. 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, therefore, greatly reduces the amount time required to achieve MAV flight capability which becomes invaluable when several different variations of a platform are considered.

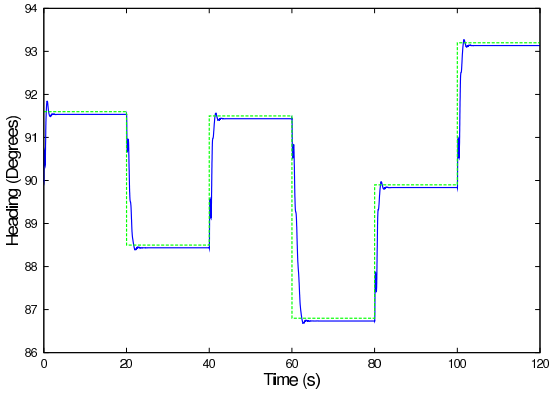
5.2 Heading Control

As mentioned in section 4.3, heading control is achieved with two cascaded controllers. The first one using heading information to produce the desired roll angle while the second one uses the roll information to produce the aileron control input. Figure 5 shows the results for several random desired headings with each time a comparison between neural network and PID results with the dashed line representing the desired heading and the solid line representing the actual heading.

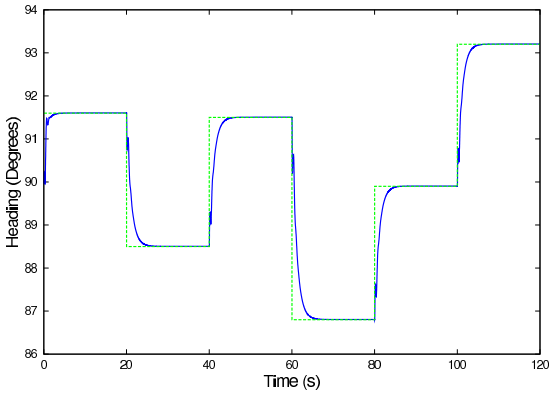
The heading control implementation was done in a very similar way as the altitude control. Once again, the process was fairly simple but required significant tuning for the PID controller. Both controllers were able to track the desired heading closely while providing good system behavior. Both responses are fast with the desired heading reached within seconds. The neural network was once again slightly more aggressive and reached its desired value more quickly than its PID counterpart but with a very minimal amount overshoot while the PID controller reached its desired values without any overshoot.

The heading control PID was tuned in an analogous fashion as the altitude control PID from Section 5.1 with the PID gains pushed so that the response is as fast as possible without compromising the system. Once the desired behavior was obtained the gains were kept for the rest of the experiments.

The elevon positions corresponding to the heading changes for both neural network and PID controllers are very much alike except for the elevon angle range that the controllers use. Even though the heading/roll control neural networks are a little more aggressive in their control strategies, 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. The neural networks being a little more



(a) NN Controller



(b) PID controller

Figure 5: Desired and actual heading

aggressive, tend to react a little quicker with more elevon deflection than the PID controllers which leads to achieving the target heading slightly faster while requiring smaller elevon deflection. This difference is however somewhat minimal and does not greatly impact the overall behavior of the system.

The position of the left and right elevons is not exactly zero degrees 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 two controllers, 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. As before, each trim constant is obtained by trial and error while the heading and roll neural network controllers automatically adjust and do not need further tuning beyond the basic neural network training. Once again, a significant advantage is gained when implementing controllers for similar platforms that have different specifications.

5.3 Wind Gusts and Turbulence

Though the neuro-evolutionary algorithm provided slightly better control of the MAV than a PID, the differences were minimal when the conditions were benign. In this section, we explore the robustness of both algorithms to sudden changes in their environment, by introducing wind gusts and turbulence. Wind gusts are represented by a force acting on the MAV in a constant upward direction (In the direction to maximally disturb the MAV's attempt to maintain a stable altitude). Wind gusts were created at 4 second intervals with the intensity of the wind gust increasing at every step. Turbulence also provides a disturbance, but in this case the direction of the force changes randomly. The controllers aimed 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 and turbulence start after 20 seconds of normal flight.

Response to Wind Gusts:

The neural network and PID controllers can both keep the heading within a small margin of error (not shown) in the presence of wind gusts, mainly because the direction of the gusts are orthogonal to the heading. However, significant differences appears in the performance of the controllers in maintaining the altitude. Figure 6 shows the performance of the neuro-evolutionary algorithm and the PID controller where the dashed line represents the desired altitude. The neuro-evolutionary algorithm keeps the altitude within 1 foot from the desired altitude while the PID can only maintain it within 6 feet.

To achieve these results, the neuro-controller was trained in a similar way as in Sections 5.1 and 5.2 to start with but in this case evaluation function G_{Φ_2} from Equations 3 was used. This new evaluation function provide more control over the roll response since they include the roll rate in addition to the error between desired and actual roll. Superior performance was obtained from the neuro-controller in the presence of wind gusts which could prove very beneficial for some applications that require holding a fairly narrow altitude range to achieve acceptable results. Improvements on altitude hold were by a factor of five in this case.

One might 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. However, though it is theoretically possible to do this, it would require a degree of hand tuning that is impractical at best, impossible at worst. Also, note that the training of the neural network control system was conducted within a particular set of environmental conditions. Many other experimental and training conditions could be implemented to further improve the flexibility and robustness of the MAV platform. The neuro-controller could also be trained for particular environmental conditions if some of these are known in advance, for example flying in urban environment versus flying in coastal areas.

Response to Turbulence: We also explored the performance of both algorithms in the presence of turbulence. The neural network training was conducted with increasing levels of turbulence instead of wind gusts. Figures 7 and 8 show the MAV altitude and heading for both the neural network and PID controllers in the presence of turbulence.

Altitude remains fairly constant throughout the experiment without significant difference between neural network and PID controllers. The small altitude variations are within half a foot of the desired value (Figure 7) and are therefore not highly significant.

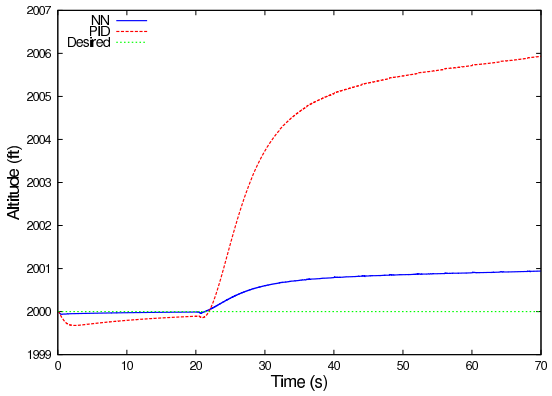


Figure 6: Desired and actual altitude (Increasing Wind Gusts Magnitude)

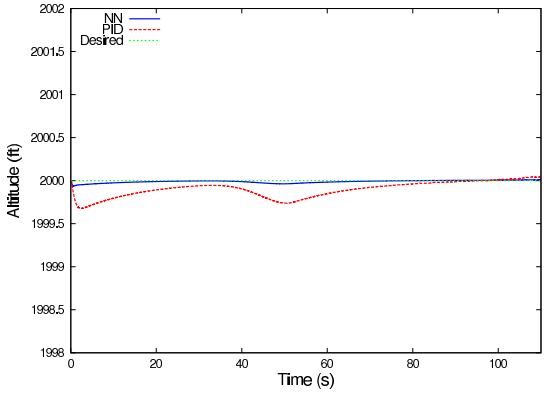


Figure 7: Desired and actual altitude (Increasing Turbulence)

A significant difference is, however, visible for heading control where neural network controllers were able to remain closer to the desired heading. The heading is kept within about a degree from the desired heading using the neural network controllers while a difference of about three degrees is seen using the PID controllers (Figure 8). Heading hold improves by a factor of three with turbulence present which could possibly slightly increase the range since deviations from the desired flight plan are kept to a minimum.

6. DISCUSSION

Micro Air Vehicles present a new and encouraging platform for collecting information in new and in some cases previously inaccessible environments. Yet, they are typically challenging to control 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 in the implementation of controllers as well as adds robustness to wind gusts and turbulence.

Sections 5.1 and 5.2 showed that training a neuro-controller on an MAV with an evolutionary algorithm was possible and required less tuning than a PID controller. Additional trim constants were also not necessary for the neuro-controllers as these controllers automatically adapted to the specifics of the platform. The neuro-controllers were also ready to use

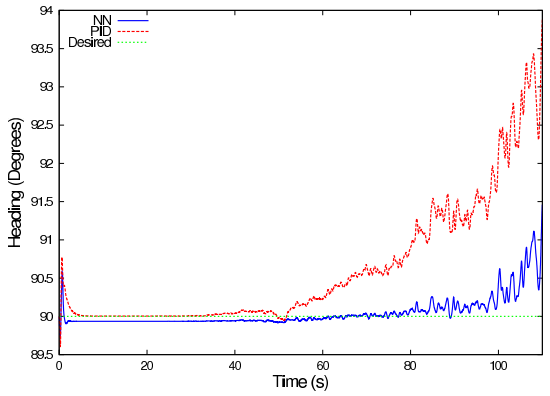


Figure 8: Desired and actual heading (Increasing Turbulences)

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 5.3 presented the first set of experiments where the neuro-controllers were trained beyond the basic flight requirements to determine whether they could adapt to harder flight conditions and improve MAV robustness to wind gusts and turbulence. Results were encouraging and showed that altitude hold in the presence of wind gusts improves by a factor of five and heading hold in the presence of turbulence improves by a factor of three. Neural networks can effectively learn and adapt from the system and its environment and provide an optimal system's configuration that improves performance. This flexibility provides an important advantage as MAV control can be improved and customized for a particular platform or known and unknown environmental conditions without requiring any significant amount of tuning as long as the evaluation function is designed correctly.

The results presented in this paper are a first step that shows the potential of leveraging learning based methods to improve MAV control implementation and MAV performance. Future work in this area will include improving the robustness of the controller by training it in various environmental conditions so it can adapt to a broader spectrum of unknown real flight conditions. Experimental training and testing could also be done for specific environments so that slightly different controllers could be used depending on the area of operation. Furthermore, it could also be possible to let the neural network based controller adapt automatically to any environment it might end up operating in by keeping the learning active with some type of limited range of possible control solutions so that the MAV remains under proper control to avoid crashes.

The controllers presented in this paper only modify the elevon angles while the throttle remains at a constant value for the duration of the experiments. An interesting extension is to allow the throttle values to change as well. This would add functionality such as maintaining constant speed while performing heading or altitude change, or potentially improve the robustness to wind gusts and perturbations.

Finally, this methodology could also be extended to seg-

menting the flight control surfaces. For example, for an 5.5ft wingspan Unmanned Aerial Vehicle (UAV) the wing ailerons were divided into 16 independent control surfaces that each had their own actuator [1]. Flight tests showed promising results and improved performance over the unmodified aircraft. Those tests demonstrated the concept of segmented control surfaces and provided good preliminary results but provided no method for finding an optimal actuation mode for the system. A better approach for controlling segmented control surfaces was presented in [7] based on deriving specific agent evaluation functions [3, 22]. The device under investigation is called MiTE which stands for Miniature Trailing Edge Effector. Those devices are actuated with a deflection angle of up to 90 degrees and are 1-5% of the chord in height. The UAV used for the experiment was a flying wing with 6ft wingspan and 30 degrees of leading edge sweep. Basic multiagent control was implemented and was based on agents consisting of an actuator, sensor, and logic package taking actions and receiving fitness values based on those actions. Promising results were obtained and are an encouraging step toward multiagent based control of UAVs [3, 7, 23]. Extending those results to the MAV domain requires more finely-tuned evaluation functions that are capable of providing the necessary fitness values in the presence of external disturbances.

Future research will therefore focus on leveraging evolutionary and multiagent methods to control a modified version of GENMAV that includes segmented control surfaces. Experiments will be conducted using GENMAV with six aileron segments on each wing for a total of twelve aileron segments. A single neuro-controller will then be coupled with the modified GENMAV to start the learning process in a similar way as what was conducted in this paper. Evolving a multiagent controller is expected to further improve both the controller's performance and its robustness.

Acknowledgments

The authors would like to thank Kelly Stewart for her invaluable help with the GENMAV configuration in AVL. This work was partially supported by the Air Force Office of Scientific Research under grant FA9550-07-1-0540 and the National Science Foundation under grant IIS-0910358.

7. REFERENCES

[1] M. Abdulrahim and R. Lind. Investigating segmented trailing-edge surfaces for full authority control of a uav. In *AIAA Atmospheric Flight Mechanics Conference*, 2003.

[2] A. Agogino, K. Stanley, and R. Miikkulainen. Online interactive neuro-evolution. *Neural Processing Letters*, 11:29–38, 2000.

[3] A. K. Agogino and K. Tumer. Efficient evaluation functions for evolving coordination. *Evolutionary Computation*, 16(2):257–288, 2008.

[4] R. K. Arning and S. Sassen. Flight control of micro aerial vehicles. In *AIAA Guidance, Navigation, and Control Conference and Exhibit*, 2004.

[5] 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.

[6] J. S. Berndt. Jsbsim: An open source flight dynamics model in C++. In *AIAA Modeling and Simulation Technologies Conference and Exhibit*, 2004.

[7] S. R. Bieniawski. *Distributed Optimization and Flight Control Using Collectives*. PhD thesis, Stanford University, 2005.

[8] C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, November 1995.

[9] M. Drela and H. Youngren. Athena vortex lattice (AVL), 2008.

[10] 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.

[11] 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.

[12] P. Ifju, D. A. Jenkins, S. Ettinger, Y. Lian, and W. Shyy. Flexible-wing-based micro air vehicles. In *40th AIAA Aerospace Sciences Meeting & Ex*, 2002.

[13] 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.

[14] 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.

[15] D. Moriarty and R. Miikkulainen. Forming neural networks through efficient and adaptive coevolution. *Evolutionary Computation*, 5:373–399, 2002.

[16] N. Nigam and I. Kroo. Control and design of multiple unmanned air vehicles for a persistent surveillance task. In *AIAA*, 2008.

[17] 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.

[18] F. Pasemann. Evolving neurocontrollers for balancing an inverted pendulum, 1998.

[19] W. J. Pisano, D. A. Lawrence, and P. C. Gray. Autonomous uav control using a 3-sensor autopilot. In *AIAA Conference and Exhibit*, 2007.

[20] G. Platanitis and S. Shkarayev. Integration of an autopilot for a micro air vehicle. In *AIAA*, 2005.

[21] 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.

[22] 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.

[23] K. Tumer and A. Agogino. Distributed agent-based air traffic flow management. In *Proceedings of the Sixth International Joint Conference on Autonomous Agents and Multi-Agent Systems*, pages 330–337, Honolulu,HI, May 2007.

[24] A. Wu, A. Schultz, and A. Agah. Evolving control for distributed micro air vehicles. In *IEEE International Symposium on Computational Intelligence in Robotics and Automation*, pages 174–179, 1999.