AN ABSTRACT OF THE DISSERTATION OF

Max Salichon for the degree of Doctor of Philosophy in Mechanical Engineering presented on December 4, 2009.

Title: Learning Based Methods Applied to the MAV Control Problem

Abstract approved: _____

Kagan Tumer

This thesis addresses Micro Aerial Vehicle (MAV) control by leveraging learning based techniques to improve robustness of the control system. Applying classical control methods to MAVs is a difficult process due to the complexity of the control laws with fast and highly non-linear dynamics. These methods are mostly based on models that are difficult to obtain for dynamic and stochastic environments. Due to their size, MAVs are affected by wind gusts and perturbations that push the limits of model based controllers where the linear approximation no longer holds. Instead, we focus on a control strategy that learns to map MAV states (e.g., heading, altitude, velocity) to MAV actions (e.g., actuator positions) to achieve good performance (e.g., flight time, minimal altitude and heading error) by maximizing an objective function. The main difficulty with this approach is defining the objective function and tuning the learning parameters to achieve the desired results. These learning based

techniques have been used with great success in many domains with similar dynamics and are shown to improve MAV robustness with respect to wind gusts, perturbations, and actuator failure. 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. Finally, we show significant improvements in the case of control surface actuator failure when using a multiagent system. The multiagent control system performs up to 8 times better than the PID controller when tracking a target heading.

Learning Based Methods Applied to the MAV Control Problem

by

Max Salichon

A DISSERTATION

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Doctor of Philosophy

Presented December 4, 2009
Commencement June 2010

Doctor of Philosophy dissertation of Max Salichon presented on December 4, 2009.

APPROVED:

Major Professor, representing Mechanical Engineering

Head of the School of Mechanical, Industrial, and Manufacturing Engineering

---

Dean of the Graduate School

I understand that my dissertation will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my dissertation to any reader upon request.

Max Salichon, Author

ACKNOWLEDGEMENTS

I would like to thank my advisor, Dr Kagan Tumer for his very valuable insight on multiagent techniques, for his support, and for always pushing me to improve the quality of my work.

I would like to thank my committee, Dr Belinda Batten, Dr Robert Paasch, Dr Mike Bailey, and Dr Merrick Haller for their time and patience throughout this project.

I would like to thank the AADI research group (Adaptive Agents and Distributed Intelligence) for their help and very useful comments.

I would like to thank Cressey Merrill and the OSU Scuba staff, for being awesome friends and for letting me be part of the group.

I would like to thank Olivia Pinon for her friendship and encouragements.

I would like to thank Dr Jim Funck for all his help and encouragements throughout graduate school.

I would like to thank Dr Reeb and Dr Brunner for their encouragements.

I would like to thank all my friends for the great and fun times.

I would like to thank my parents for all their support throughout the years.

I would like to thank my beautiful girlfriend Trista Baxter for all her support, for her encouragements, for being there for me, and for being an awesome girlfriend.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

LIST OF FIGURES

LIST OF FIGURES (Continued)

LIST OF FIGURES (Continued)

LIST OF FIGURES (Continued)

LIST OF FIGURES (Continued)

LIST OF FIGURES (Continued)

LIST OF FIGURES (Continued)

Chapter 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 [64, 63], reconnaissance, sensing [46], search and rescue, and enemy targeting [74]. MAVs can accomplish such demanding missions without endangering human lives, giving a very important edge to the organization using them. NOAA (National Oceanic and Atmospheric Administration) for example uses them increasingly to assess arctic ice change and affects on ecosystems and coasts, increase safety and success in fighting wildfires that threaten people and property, and monitor coasts, oceans, environments important for fish, and marine sanctuaries. The many benefits resulting from using MAVs pushed researchers into improving this platform to provide better and more stable flight characteristics. A wide variety of MAV platforms and control strategies have been studied and show promising results in this area [49, 92, 93, 48, 29, 56, 68, 36].

The target size for MAVs is typically from insect to bird size and most MAVs are between 15 and 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)[4, 55] with a range of Reynolds number similar to that of an insect or bird (10^3 - 10^5). 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, wind gusts and turbulences

are present [10, 78].

As a consequence MAVs present a number of challenges where integrating all the necessary components within the volume and weight requirements is not a trivial task. MAV restrictions include: limited processing power, limited control surfaces and actuators, limited number and quality of the sensors, and limited power available. They are also typically unstable and difficult to control due to fast and highly non-linear dynamics [93]. Large forces and moments that are difficult to predict as well as unsteady flow conditions are typically encountered in the flight environment that MAVs have to operate in. The MAV control system has to therefore be fast, flexible and robust in order to achieve proper platform stabilization and guidance.

MAV platforms can be categorized into three main types: flexible-wing/fixed-wing MAVs, rotorcraft MAVs, and UAVs. Those categories are dependent on a specific range of applications that the platform is used for. Indoor/urban terrain use, low speed, and hovering capabilities characterize the rotorcrafts. They are also highly unstable and difficult to control. Longer range, higher speeds, and higher altitude distinguish the fixed-wing/flexible wing MAV platform. They are also unstable, hard to fly, and susceptible to wind gusts. Lastly, long range of operation at high altitude and high speed designates the UAV platform. They are much more stable, bigger, and controllers can be designed more easily. They can however be detected more easily and are not as flexible in their utilization and configuration.

Several control techniques have been implemented on MAVs and can be divided

into two general groups: model-based, and learning-based control methods. Model-based control techniques are a more classical control methods where the system is studied and modeled in a first phase, and then a controller is designed based on the model. They are well suited for systems where models can be constructed without too much difficulty and the controller designed on a linearized version of the model can still perform well on the real system without excessive tuning or optimization. Model-based techniques such as PID and state space control can be applied to UAVs with success but more difficulties are encountered when those techniques are applied to the highly non-linear and unstable MAV platform. Tuning and optimization are required to achieve more stable flight characteristics and better performance. Learning-based control techniques as opposed to model-based control techniques do not require a model of the system to be designed. The idea behind learning methods is to learn an optimal mapping between inputs and outputs provided to the control system. The selection of the correct types of inputs provided by the sensors is critical to achieve optimal results. Those learning methods are well suited for non-linear systems such as the MAV platform.

Multiagent control techniques have been used for a wide range of problems and many different types of techniques have recently been introduced. The main multi-agent control methods can be classified into reinforcement learning techniques, evolutionary computation techniques, auction-based methods, and collectives which are a higher order type of methods that can be applied to previously cited methods. Reinforcement learning and evolutionary computation techniques have been modified and improved depending on the needs of specific applications. They are

well suited for MAV control problems that are highly non-linear and complex problems. These methods provide a high flexibility to the control system that adapts and learns when a particular configuration is modified such as adding an agent to the system or increasing the number of actions that an agent can take. These control systems also prove very useful when noise and/or failures occur in the system. Also, the selection of the inputs to the control system plays an important part in the resulting speed and efficiency of the multiagent control system.

The three contributions of this work are as follows. We first show (Chapter 4) that neuro-evolutionary techniques can be used to control multiple surfaces to improve the flight characteristics of an MAV by designing appropriate objective functions (e.g. roll moment value). We then show (Chapter 5) that learning based methods can improved wind gust robustness of MAVs when compared to a PID controller and that learning based methods coupled with segmented control surfaces increases robustness to actuator failure. Finally, we show (Chapter 6) that multiagent techniques further improve the robustness of the MAV platform and provide better recovery solutions in the case of actuator failures.

Chapter 2 – Literature Review

2.1 Micro Air Vehicles (MAVs)

2.1.1 Fixed-Wing/flexible-wing MAVs

A classic approach to MAV design is the fixed-wing MAV design. The fixed-wing configuration allows for a simpler design that is usually quicker to design and implement than a flexible-wing approach where the dynamics evolve with the position and shape of the wing. Those types of MAVs are also more suited for a classical model based approach of control where a model of the system is needed in order to design the controller. Successful implementation of such a platform was demonstrated in [68, 56] where an MAV platform as well as a controller were developed and showed good performance in fully autonomous flights including take-off and landing maneuvers. Fixed-wing MAVs are also easier to test in a simulator due to their better know flight characteristics and dynamics. Experiments of this nature were presented in [66] where several specific features are implemented in an MAV simulator. Fixed-wing MAVs are constantly being optimized and the design of better more efficient configurations is an active area of research. Some design experiments are shown in [55, 92, 50]. Fixed-wing MAVs are well suited for a large number of applications and can be modeled and managed with classical control methods. They are however susceptible to wind gust due to their low maximum

speed, small size, and small weight. Even though classical model-based control methods were successfully applied on this type of vehicle, the high non-linearity and very unstable flight characteristic result in less than optimal classical control solutions that usually require important and lengthy tuning and optimization procedures. In addition, their lack of flexibility does not allow other more intelligent methods of control to be easily applied to improve the performance of the system.

The flexible-wing MAV design has recently seen more interest and was used to improve stability of the vehicle as well as improve its wind gust resistance. In most cases, the payload capacity of the MAV was also improved. The wing in this case deforms continuously throughout the flight and absorbs the energy created by the instabilities of the air flow. Intelligent methods of control could more easily be applied and tested due to the higher flexibility of the platform.

Carbon fiber prototypes have been under development for several years at the University of Florida and the Air Force Research Laboratory Munitions Directorate. Abdulrahim et al [4] presents an overview of the development of different flexible-wing MAVs with sizes between 6 and 24 inches. The design of the shape and form of the flexible-wing is discussed for different size MAVs, and flight test comparison were conducted between their first 2D wing design, a later 3D wing design, and a standard fixed-wing MAV. The 3D version showed to be more stable, maneuverable, and with an overall better efficiency than the 2D and standard versions. The fuselage, and propulsion system were also optimized to improve the performances of the MAVs. This study showed some important advantages of the flexible-wing MAV concept. Using carbon fiber as a building material allows an

important design flexibility. This in returns allows for improvements of the flight characteristics of the MAVs by modifying the wing design and configuration. The wing can be made more or less flexible, and have different shapes and lengths depending on the size and flight characteristics needed for a particular application. MAV improvements based on design modifications include a higher maximum airspeed, higher climb rate, improved turning capability, and a higher lift to drag ratio. The higher lift to drag ratio is particularly important for MAVs as it improves their gliding capabilities. This feature would prove very useful if the control strategy uses a reinforcement learning or evolutionary algorithm [86] that could control the MAV in case of motor failure. In this situation, the controller would adapt and control the MAV as a glider, that could be landed safely.

Fundamentally, flexible-wings of MAVs deform and absorb energy from wind irregularities leading to better performance and flight characteristics as passive form of control. It is also possible to use this characteristic as an active form of control by using actuator to change the shape of the wing during flight. A study based on a similar platform was conducted by Garcia et al [36] to shows that roll control of such a MAV can be achieved by actively morphing the wing. The morphing of the wing mainly creates a roll motion that allows for more stable maneuvers compared to using the standard control surfaces. Wing morphing is obtained by attaching a string to the trailing edge of each wing. Those strings are attached to a single servo and therefore only one of the wings can be morphed at a time. Preliminary tests and simulations show that wing-morphing cannot be used directly to control the MAV. A controller was built to integrate wing-morphing with the other

actuators (elevator and rudder) in order to provide regular control actions to the pilot or flight controller. This study shows the potential of using wing-morphing as an active mode of control for MAVs. Only one actuator was used to modify the shape of the wing but such a concept could be extended to having multiple actuator morphing the wing in different places. This would provide a better control over the shape of the wing and therefore a more flexible way of controlling the MAV. Another extension from this study would be to use a different approach as far as the control strategy used to control the MAV. It was shown that wing-morphing cannot be used directly to control the MAV and a specific controller for integrating wing-morphing with the other modes of control was needed in addition to the regular flight controller. It seems like this configuration could be better approached with a control strategy that would not require a model of the dynamics and could use wing-morphing with multiple actuators directly without the need of an intermediate controller that would be difficult to design for optimal performance and stability. Such a control strategy could be a multiagent reinforcement learning control strategy [8] where each actuator would be an agent. Each agent would base its actions on information received about the system such as position, orientation, speed, and acceleration. Each action could correspond to an actuator position. An added advantage would also be the ability of the control system to adapt and perform well in conditions where noise and/or failures are present in the system. These techniques will be discuss in Section 2.2 of this chapter.

Waszak et al [93] also conducted a study on an elastic membrane wing to improve flight characteristics of MAVs. As seen previously, the flexible wing increases

stability of the MAV and results from this study show that the vehicle was capable of higher angles of attack than its fixed wing counterpart. This study shows the potential of the flexible-wing concept and its advantages over fixed-wing MAVs.

Flexible-wing MAV show significant improvements over more classical fixed-wing MAVs. Wing morphing allows for more stable flight characteristics, higher resistance to wind gusts, and an improved payload capacity. Furthermore, learning-based control techniques that do not require a model of the system could be implemented more easily than on fixed-wing vehicle due to the higher platform flexibility of flexible-wing MAVs. Additional actuators could be implemented to morph the wing and provide more control options. This approach would be better suited for this type of platform where classical model-based methods would require important tuning procedure to achieve better results and more stable flight characteristics. Even though, the design process is more complicated than more conventional approaches, the benefits of the flexible-wing approach are worse investing into.

2.1.2 Rotorcraft

Another approach to MAV design is the rotorcraft approach. This platform is highly unstable but has the benefits of being able to stop and hover as well as being better suited for an indoor/urban terrain types of missions. Helicopters due to their high instability are a very challenging control problem. Many control techniques have been applied to keep the system under control with some very successful implementations. Helicopter dynamics can be modeled without excessive difficulty

and the system can therefore be modeled to design a classical type of controller. Learning-based control methods can also be applied and have also shown impressive results such as the successful implementation of a learning technique by Ng [1, 62, 61]. The control system is not based on a purely reinforcement learning method since the controller learned from a model of the system that was obtained from real flight data. Difficult maneuvers where performed by the controller and demonstrate the efficiency and robustness of learning-based control strategies. Other helicopter-based implementations are presented in [26, 24, 25, 27].

A similar platform was also used in several studies to achieve same kinds of objectives as the helicopter platform. Quad-rotor vehicles advantages include a higher payload, and a simple configuration. They are however very unstable with extremely fast open-loop dynamics leading to an even more challenging problem. An example of quad-rotor configuration can be seen in Figure 2.1. Modeling, simulation and platform implementation of the quad-rotor concept in shown in [44, 17, 59].

Quad-rotor dynamics models can be obtained and used for designing model-based controller but results have been limited due to the extreme instability of the platform. This platform can however be easily modified and an intelligent type of controller would be well suited for controlling the system. Providing adequate inputs to the system and establishing a learning procedure would be key to the success of such a technique.

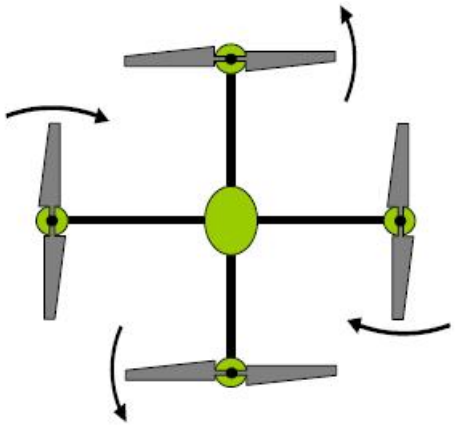

Figure 2.1: Quad-Rotor Configuration [44].

2.1.3 Model-based Control

Model-based control technique are a classical type of control methods that are based on obtaining the detailed dynamics of the system being studied, constructing a model based on those dynamics, and design a controller based on the model. Those techniques can be applied to non-linear systems but require linearization of the model and tuning and optimization of the controller to achieve good performance. PID (Proportional Integral Derivative) controllers are based on a control loop with feedback mechanism and have been extensively used in the industry. PID controller are also used in MAV applications [68, 44, 56, 94, 97, 65]. PID controller perform well on simpler MAVs such as fixed-wing MAV where the dynamics are better understood and where models of the system are better representation of the real MAV. For more complex systems such as flexible-wing MAV, PID controller require a more in-depth tuning and optimization to obtain the desired results. PID

gain tuning as also been improved using neural networks [65].

State space control techniques are also based on a model of the system and have been applied with success to MAV control [46, 45]. State space control provide robust controllers that perform well on non-linear systems. Similarly to PID controller, state space controllers require optimizations when applied on highly non-linear systems such as flexible-wing MAVs. The state space model of the system under investigation as the form shown in Equation 2.1.

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx + Du \end{aligned} \tag{2.1}$$

Where x is the vector of state variables, u is the vector of inputs, and y is the vector of output variables or sensed quantities.

2.1.4 Neural Networks

Neural network (NN) controllers are based on mapping a set of inputs to set of outputs and do not require a model of the system being controlled. This mapping is learned by the controller using different methods. Neural networks are well suited for systems where a straight-forward model of the system cannot easily be obtained. Learning is achieved based on a cost function that provide information on how far the NN is from the optimal solution to the problem considered. The NN learns by minimizing the cost function. Learning can be done using different strategies: supervised learning, unsupervised, and reinforcement learning. Supervised learning

is done by having the NN learn form the actions of a pilot that is familiar with the vehicle. In unsupervised learning, the NN learn from a set of data. And in the case of reinforcement learning, the NN learns form its interaction with the environment based on an action-reward system. Several examples of neural network implementations are shown in [41, 72, 26, 24, 25, 27].

2.1.5 Vision Control

Vision-based control can also be applied to an MAV platform. The high processing requirements of vision algorithms require in general processing of the information in a ground station with the information transmitted between the two units. Vision can be for multiple purposes that include obstacle avoidance, path planning, and state estimation. Many studies were conducted to include vision in a form or the other in the control loop of MAVs/UAV [34, 47, 75, 98, 77].

2.1.6 Unmanned Air Vehicle (UAV)

Unmanned Air Vehicles (UAVs) are a broader class of vehicles that include Micro Air Vehicles (MAVs) as well as bigger aircrafts that have a more conventional design. Larger UAVs are over 24 inches and are usually either fixed-wing airplanes or helicopters. They have a greater stability and a higher resistance to wind gusts. The payload capacity is also increased due to their size. It is therefore much easier to add more and better sensors and navigational equipment than on MAVs where

space and weight is very limited.

Larger UAV can be controlled with a conventional control method such as PID control or state space control without the need of extensive tuning. Dynamics models can be developed and represent the real system with accuracy than MAV models. Learning-based control methods could also be implemented and the controller is very likely to show expected performances. Several UAV control techniques are presented in [72, 69, 15, 14, 3, 5].

2.2 Multiagent Coordination for Control of Complex Systems

Classical control methods require the system to be modeled accurately enough to design the system's controller. For an important number of systems, the information for building the model is either unavailable or not accurate enough. Some systems can also be too complex and the potential models would either be too big to build or would represent a too simplistic representation of the real system. For a large number of systems, classical control methods can't be applied very efficiently and require important modifications and tuning to achieve reasonable performances. Non-linear systems are particularly difficult in that respect and the design of a robust and efficient controller present some challenges. An alternate approach to solving the control problem of those systems is the learning approach that do not require a system model. Learning techniques have been used and are getting more attention to solve multiagent coordination problems for control of complex systems. Solutions for this kind of problems can be found using a

wide range of methods and techniques that can be grouped in multiple areas of research. Those research areas include reinforcement learning, evolutionary computation, auction-based algorithms, and collectives. Detailed surveys of the field can be found in [67, 80].

2.2.1 Reinforcement Learning

Reinforcement learning is a learning method based on an action/reward system where there is no plant model and the algorithm learns from its environment. At each time step, agents receive information about the environment through their sensors and decide on the next action they need to perform. Each action modifies the environment and each agent receives a reward based on the overall goal that the system needs to achieve. The agents keep learning by trial and error until the solution to the problem is optimal or near optimal. Assigning rewards to agents is done using a reinforcement function that maps system state to agent actions. Agents will try to maximize that function in their learning process. The design of the reinforcement function varies depending on the environment and the goal that needs to be achieved. The performance of the algorithm is measured based on the convergence to the optimal solution and the speed of convergence.

Designing the reinforcement function (also called value function) is a difficult problem and many techniques have been developed to design a reinforcement function that provides optimal results for the problem being handled. Several algorithms for multiagent reinforcement learning are presented by Guestrin et al [43].

A variation of Q-learning and a variation of Least Squares Policy Iteration (LSPI) are used as a starting point. Those methods are then combined with policy search methods to obtain better results.

An example of experiments in simulation for a group of robots is presented in [28]. Two communication strategies are also presented and results show that those strategies significantly improve the learning of the robots. This chapter demonstrates the possibility of improved group performance through group organization based on the environment.

An application of reinforcement learning can be found in the RoboCup simulated soccer competition [51, 79]. This competition presents many challenges and provides a good test ground for experimenting new learning algorithms, techniques, and strategies. Some of the difficulties of the competition include multiple agents playing against multiple adversaries, real-time computing, large number of possible actions, noisy inter-agent communication, and limited sensing capabilities. In [51], a SARSA (State-Action-Reward-State-Action) algorithm used for defense strategies is applied to an offense strategy. Several tests showed that this method did not provide optimal results for this particular configuration and an optimization method based on inter-agent communication was then tested and provided better results. Those tests illustrate the difficulty of designing reinforcement learning algorithms. An algorithm performing well in a particular scenario might not provide optimal results in another more complex scenario. Optimizations are however possible and can be applied to improve performance of algorithms when they do not provide optimal results.

As seen previously, reinforcement learning algorithms can always be used to solve a particular problem but implementing an efficient algorithm can be difficult depending on the problem under investigation. Large multiagent problems are particularly challenging and many reinforcement learning methods become slow and inefficient when applied to those kinds of problems. The QUICR-learning approach [8, 7] was proposed to solve multiagent coordination problems in a faster and more efficient way. The QUICR-learning approach is described below.

In a multiagent system, agents use a reward to determine their next action. This reward is a function of the states of all the agents which are a function of all the previous actions (Equation 2.2). In large multiagent systems the reward is affected by a high number of actions which slows down the learning process.

$$R_t(s_t(a)) = \sum_{k=0}^{T-t} r_{t+k}(s_{t+k}(a)) \tag{2.2}$$

A standard method for this kind of problem is the standard Q-learning approach. Q-learning algorithms works by learning the value of the state-action pairs stored in Q-value tables. Those values correspond to the sum of the future rewards that needs to be maximized. Each Q-value pair is updated with the rule in Equation 2.3.

$$Q(s_t, a_t) = r_t + max_a Q(s_{t+1}, a) \tag{2.3}$$

The main drawback of this approach is that for large multiagent systems, the algorithm is slow. This is due the global reward system where the reward of each

agent is highly dependent on the actions of all the other agents. As the number of agents, actions, and time-steps increase in the system, the global reward depends on an increasingly large number factors making the learning process very slow.

An improvement over the standard Q-learning approach is the Local Q-learning method. It is based on the assumption that the agent's actions are independent. The reward can then be linearly separated so that each agent will be rewarded based on its actions only. Equation 2.4 shows the reward for the Local Q-learning method.

$$r_t(s_t) = \sum_i w_i r_{t,i}(s_{t,i}(a_i)) \tag{2.4}$$

This approach speeds up the learning process and works well as long as the agent and their actions are relatively independent. Otherwise, the agents maximizing their reward might maximize the system's overall objective.

QUICR-Learning was introduced to provide a faster algorithm than the standard Q-learning algorithm while at the same time maximizing the overall system's objective. The reward is not assumed to be linearly separable but it still provides feedback to each agent based on its own actions.

Equation 2.5 shows the difference reward that is used in the QUICR-Learning methods to maximize the system's goal and provide specific feedback to each agent.

$$D_t^i(s_t(a)) = R_t(s_t(a)) - R_t(s_t(a - a_{t,i})) \tag{2.5}$$

Where $a - a_{t,i}$ is a counterfactual state where agent i has not taken the action

it took in time step t. Equation 2.5 can be rewritten as shown in Equations 2.6 and 2.7.

$$D_t^i(s_t(a)) = \sum_{k=0}^{T-t} r_{t+k}(s_{t+k}(a)) - r_{t+k}(s_{t+k}(a - a_{t,i})) \tag{2.6}$$

$$D_t^i(s_t(a)) = \sum_{k=0}^{T-t} d_{t+k}(s_{t+k}(a)), s_{t+k}(a - a_{t,i}) \tag{2.7}$$

Some assumptions need to be made to overcome the problem created by $d_t(s_1, s_2)$ being usually non-Markovian. Those assumptions are as follows:

1. The counterfactual action $a - a_{t,i}$ moves agent i to an absorbing state, s_b
2. s_b is independent of the agent's current (or previous) state(s)

The new reward can then be written as a Markovian function. Equation 2.8 shows the reward where $s_t - s_{t,i} + s_b$ indicates that agent i's state was replaced with state s_b.

$$d_t^i(s_t) = r_t(s_t) - r_t(s_t - s_{t,i} + s_b) \tag{2.8}$$

The QUICR-Learning rule can than be written and is shown in Equation 2.9 and 2.10.

$$Q_{UICR}(s_t, a_t) = r_t(s_t) - r_t(s_t - s_{t,i} + s_b) + max_a Q(s_{t+1}, a) \tag{2.9}$$

$$Q_{UICR}(s_t, a_t) = d_t^i(s_t) + max_a Q(s_{t+1}, a) \tag{2.10}$$

Showing that QUICR-Learning maximizes the overall system's objective is done by showing that agent i maximizes $d_t^i(s_t)$. Equations 2.11, 2.12, 2.13, and 2.14 demonstrate that agents maximizing their customized reward will also maximize the overall system's objective.

$$\frac{\partial}{\partial s_i} d_t^i(s_t) = \frac{\partial}{\partial s_i}(r_t(s_t) - s_{t,i} + s_b)) \tag{2.11}$$

$$\frac{\partial}{\partial s_i} d_t^i(s_t) = \frac{\partial}{\partial s_i}(r_t(s_t)) - \frac{\partial}{\partial s_i}(r_t(s_t - s_{t,i} + s_b)) \tag{2.12}$$

$$\frac{\partial}{\partial s_i} d_t^i(s_t) = \frac{\partial}{\partial s_i}(r_t(s_t)) - 0 \tag{2.13}$$

$$\frac{\partial}{\partial s_i} d_t^i(s_t) = \frac{\partial}{\partial s_i}(r_t(s_t)) \tag{2.14}$$

A similar algorithm, DU (Difference Utility) used in other types of problems was converted and is show in Equation 2.15. In this equation, c_t is independent of state s_i.

$$DU_t^i(s_t) = r(s_t) - r(s_t - s_{t,i} + c_t) \tag{2.15}$$

A comparison between standard Q-Learning, Local Q-Learning, QUICR-Learning,

and the DU (Difference Utility) algorithms is then obtained when testing those algorithms on two specific problems. The first problem is a traffic congestion experiment where drivers can take different road to get to their destination, with each road having its own capacity. The other experiment is a multiagent grid problem where agents can move on a grid square and where they have to observe tokens placed on some of the squares. Results for both experiments showed that QUICR-Learning was faster and more efficient than the other algorithms.

QUICR-Learning is an elegant solution to the multiagent coordination problem. It was shown to be faster and more efficient than other reinforcement learning methods. QUICR-Learning provides a customized reward to each agent that depends primarily on its actions. The idea behind this reward system is to subtract actions that are not the agent's to the standard Q-Learning reward. This reward is more agent specific and allows agents to learn rapidly. QUICR-Learning does not assume that the agent's actions are independent and therefore provides rewards that maximize the overall system's objective. QUICR-Learning can be applied effectively to any multiagent coordination and can also be extended to other reinforcement learning methods. A potential application of this technique could also be for the control of flexible-wing MAVs as was mentioned in section one of this chapter [36]. Each actuator could be an agent that would obtain information about the system through the MAV sensors and the possible actions of the agent would be the actuator's position. Coordination between agents would be an important factor in keeping the MAV stable and performing flight maneuvers.

A method for improving the speed of reinforcement learning algorithms was

proposed by Ahmadi [9]. This method can be used with any value function reinforcement learning algorithm and is based on having the RL algorithm learn on an incremental number of subsets ordered by importance instead of leaning on the full set of features at once. The Incremental Feature-Set Augmentation for Reinforcement Learning Tasks (IFSA) is presented below.

IFSA is a general purpose algorithm that uses some information about the domain of the problem in order to increase the speed of the RL algorithm used to solve the problem. IFSA needs to be customized for the problem under consideration and the domain expert has to decide on how to organize the agent's state spaces into different feature sets.

As stated previously, IFSA can be used with any value function reinforcement learning algorithm but this chapter demonstrate the efficiency of the method using a Sarsa algorithm (state, action, reward, state, action). Equation 2.16 is used to update the $Q(s, a)$ function.

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)] \tag{2.16}$$

The key to the successful implementation of the IFSA method is to first separate the agent's state spaces into feature sets and then reorganize those sets by their importance. If the organization of those feature sets is done incorrectly, then IFSA may not lead to the expected speed increase of the RL algorithm being used. Once the feature sets are organized, the next step is to start the learning with the most important feature until the algorithm converges to a policy very close to the

optimal policy. The second most important feature is then used to augment the feature set used by the algorithm. The value obtained when learning on the initial feature set is used as the initial value for the new augmented feature set. This allows the algorithm to use use the learning from a reduced feature set and refine the learning with the new feature set. The algorithm learns rapidly on a basic set of features and the build-up with additional details provided by the additional features. Pseudo-code for the IFSA algorithm is shown is Algorithm 1.

Algorithm 1: Pseudocode for IFSA algorithm [9]
Let $\Phi = (\phi_0, \phi_1, ..., \phi_{m-1} \subseteq F)$ can be an ordered augmentation of the feature-set of the problem.
for $i = 0$ to $m - 1$ do
    Learn $V^*(\phi_0...\phi_i)$ until reasonably converged
    for all ϕ_0 ... ϕ_{i+1} do
        $V^*(\phi_0...\phi_{i+1}) = V^*(\phi_0...\phi_i)$
    end for
end for

IFSA has the advantage of being usable with any value function reinforcement learning algorithms on a wide range of problems. The main drawback is that its application is highly dependent on the domain of the problem. Someone needs to have a good knowledge of that domain in order to be able to build those feature set from the agent's state spaces and most importantly organize them based on their importance. If this not done correctly, the improvement on the speed of the

algorithm may not occur. Also, some problems might not lend themselves very well to a representation by subsets of features and in those cases, IFSA cannot be applied.

Another approach that improves upon standard reinforcement learning algorithms in presented in [21] and is based on the WoLF Policy Hill-Climbing (WoLF-PHC) which corresponds to the principle of "learn quickly while losing, slowly while winning". This technique is applied in multiagent environments based on a stochastic game framework. The WoLF-PHC as the name implies is an improved version of Policy Hill-Climbing. PHC is an extension to Q-Learning that was discussed earlier where the algorithm still updates the Q-values but in addition the agent selects its next action based on a mixed strategy. This mixed strategy is maintained and improved based on a learning rate δ which increase the probability that the action with the highest Q-value will be selected. WoLF-PHC builds on this technique by adding the "Win or Learn Fast Principle". The idea is that agents learn faster when selecting a wrong action, and learn slower when selecting a good action. The standard PHC is modified to incorporate the new principle which implies using two different learning rates, a winning and a loosing learning rate. The agent determines if he is winning or loosing by comparing the results of the current policy versus the average policy. If the results from the current policy are lower, then the agent is loosing and the loosing learning rate is used by the algorithm. The WoLF-PHC algorithm provided very good results when tested on several stochastic games, and it also out-performed other multiagent reinforcement learning algorithms. WoLF-PHC shows interesting characteristics that could prove

useful in an MAV application where the reinforcement learning algorithm needs to correct its actions very quickly if those could lead to a very unstable situation and potentially a crash of the MAV.

Further application of the WoLF algorithm can be found in [19, 22, 20]. An in-depth description of the algorithm is provided in [18].

Similar applications of reinforcement learning techniques can be found in the following chapters: [87, 2, 37, 82, 85].

2.2.2 Evolutionary Computation

Evolutionary computation (EC) is a technique based on a guided random search among a population of solutions. The algorithm starts with randomly generated solutions that are then graded and selected for breeding and mutating individual solutions in order to produce new solutions that will replace older ones. It is similar to biological evolution: reproduction, mutation, recombination, and selection. At each breeding cycle or generation, solutions are improved until an optimal or near optimal solution is found or until the time constraint has been reached.

Evolutionary computation could prove highly beneficial in systems where failure can occur. Agents in the system would evolve to the next optimal or near optimal solution in the event of an agent failing in the system. The system would then be able to recover and use the newer solution to optimally control the system. Similarly, when modifications and improvement are performed on the system (adding new agents or new actions), the system can evolve and use previously

learned information to adapt to a new system configuration without changes to the control system.

Evolutionary computation is well suited for multi-rover exploration tasks. Rover configuration changes when new information about the environment is discover and the control algorithm needs to be able to evolve to a new optimal solution. Furthermore, failure of one or multiple rovers does not cause failure of overall system, EC algorithms are therefore robust and can overcome many challenges.

[86] presents an application of evolutionary computation to a multi-rover problem as well as a method for shaping evaluation functions used by the rovers. The evaluation function play an important role on the performance of system and at the same time needs to provide specific information to a rover about its control policy, and to optimize the overall system's objective. A evaluation function meeting those criterions is shown in [86]. Also introduced in [86] is a study of the impact that different types of failures have on the system. Mobility, communication, and controller failures are tested on the multi-rover system and shows high performance of the control system when those failure occur. Similar studies [6, 70] also show analogous results.

Evolutionary computation algorithms are very robust techniques that can adapt to changing configurations as well as failures in the system. They would therefore represent a stable and dependable multiagent control technique that could be implemented on an MAV platform where configuration modification of the platform can be used to achieve better performance of the system and where the system needs to be able to recover and adapt from possible failures of elements of the

system.

2.2.3 Auction-based Robot Coordination

Auction-based algorithm for the control of multiagent systems have been used for the control and coordination of groups of robots exploring an unknown environment. Those techniques demonstrate strong robustness and efficiency and many methods have been proposed to solve specific exploration tasks [52, 54, 33, 53, 32, 58, 81, 31, 57, 12, 39, 38].

The concept of auction-based robot coordination relies on having each robot place a bid on a target, and then visiting the target that they won. Advantages of the method include high robustness, if a particular robot fails, the system continues exploring the terrain and adapt to the new robot configuration. Additionally, communications are efficient since only the bids on the targets are communicated. Another benefit is computation efficiency through parallel bid computation.

Most auction-based control methods proposed define their own set of bidding rules depending on the robot configuration, the environment, the targets, and that tasks that need to be accomplished. Since no standard method on generating bidding rules existed, Koenig [81] proposed a method for systematically producing those rules.

[81] investigates three possible team objectives that could be accomplished: MiniSum corresponding to minimizing the sum of the path costs over all robots, MimiMax corresponding to minimizing the maximum path cost over all robots,

and MiniAve corresponding to minimizing the average per target cost over all targets. Optimizing those objectives is a complex problem that is not directly computationally feasible.

Bidding rules to achieve the previously mentioned objectives are presented in [81], and need to be approximated using a greedy method to obtain the desired results. Those rules provided good performance for the selected objective and were reasonably close to the optimal solution.

Auction-based robot coordination methods provide robust decentralized method for coordinating large teams of robots for exploration of unknown environments. The system can quickly adapt to changing robot configuration has terrain is progressively discovered and target reassignments can be achieved to satisfy the system's objective. Additionally, the system can adapt to particular robot failure and reorganize the search in the most effective manner.

2.2.4 Collectives

Collectives are a set of entities that aim at conjointly maximizing an overall system's performance function and at the same time maximize their own performance function. They represent a higher level of multiagent control techniques that can be applied to lower level methods such as reinforcement learning and evolutionary computation techniques. Distributed intelligence performs better and is more robust for a wide range of multiagent systems than a single central intelligence and is therefore used extensively in multiagent control problems. [91] presents a

survey of collectives where the different characteristics of collectives are analyzed. The wide range of areas of application of collectives is also introduced and includes multiagent systems, and reinforcement learning.

[86, 83] show an application of collectives to a multi-rover control problem where evolutionary computation is implemented with the collectives concept. The algorithm showed high performance compared to other methods but was also proven very efficient when different types of failures occurred in the system such as mobility, communications, and controller failures.

[84] presents another example of implementing the collectives idea to a multi-rover control problem. The studies shows high performance of the collectives principle in a noisy and evolving environment where communications are limited between agents. Also addressed is the investigation of efficient reward method in order to achieve optimal system's performance.

Additional investigations on collectives are completed in [15, 14, 95, 96]. [89] shows how concepts of collectives can be applied to reinforcement learning methods with great success. The reinforcement learning rewards are designed to give the agent specific feedback based on its actions but also to maximize the overall system's reward.

Collectives are an important higher level tool that can be used in many different artificial intelligence (AI) and machine learning problems. The main concept of collectives is provide reward or evaluation functions to agents that are both specific to the agent and that also lead to maximization of the overall system's reward/evaluation function. This method was proven very reliable, and robust

and works well in noisy environments as well as when failures occur in the system. Applying this concept is therefore well-suited for MAV systems where configurations are flexible and where the control system needs to be able to operate in noisy environment and recover of potential failure of elements in the system.

2.3 Multiagent Control Techniques Applied to MAVs

The first section of this chapter gives an overview of the the different MAV configurations (fixed-wing, flexible wing, rotorcraft) and of the different MAV control strategies (PID, Neural Network, Vision Control). Many different solutions exists for designing and controlling an MAV. The second section of this chapter surveys the advances achieved in the field of multiagent coordination for control of complex systems. Many standards methods were improved upon to obtain better results in more complex problems. The potential use of multiagent coordination for control of MAV was briefly discussed when looking at a particular platform or technique but before exploring the subject in more details, it is important to know for kind of application the vehicle will be used for. If it is for indoor flight and requires hovering capabilities, then a rotorcraft would probably be a good platform, if it is for low altitude, medium speed (10-50 mph) and requires drawing minimal attention, then a flexible-wing MAV should be considered. Finally, if the application requires covering large distances at higher altitude and faster speed, the UAV platform could fit those needs. Each application has specific platform needs, and each platform has different requirements, capabilities and challenges. It is for those

reason that three different kinds of platforms will be presented with different ideas for applying multiagent techniques. The following sections will cover rotorcraft, flexible-wing MAV and UAV platforms and present possible new implementations of control strategies.

2.3.1 Rotorcraft

As stated previously, rotorcrafts are better suited for conditions where flying inside buildings and hovering to gather information are required for a particular missions or applications. Rotorcraft can fly at very low speed, stop, and hover to gather information. They are however typically harder to fly and stabilize and present important control challenges. An interesting platform for this type of environment would be a quad-rotor MAV. Advantages of this platform would be a higher payload, and a simple configuration. Quad-rotor vehicles have however unstable and extremely fast open-loop dynamics which are very difficult to control. Current research on control of quad-rotor MAV has mainly been a model-based approach which resulted in more or less success.

A good starting point for implementing new techniques on a quad-rotor vehicle would be to use one of the commercially available quad-rotor MAV such as VeraTech X-PRO Flyer (RC with auto-leveling and auto landing) or the RCToys Draganflyer (RC only). Those would provide a base that is ready to use which would eliminate the designing time of the vehicle and sensor package for a preliminary study. Different multiagent control techniques could then be implemented

such as QUICR-Learning [8] or IFSA (Incremental Feature-Set Augmentation) [9] which have been proven fast and successful for solving complex problems. Each rotor would be an agent that would receive information about the system through the onboard sensor package. The actions that agents would take could be represented by the direction and speed of rotation of a rotor. Additionally the WoLF Policy Hill-Climbing [21] could be a value as it would allow the control algorithm to learn quicker when a bad action is taken by an agent. Furthermore, other possible algorithms could be evolutionary computation algorithms that would also perform well in case of failures in the system as demonstrated in [86].

Future works on this type of platform could include modification of the platform itself but also a study of the sensor package onboard the vehicle. Modifications on the platform could include longer or shorter links between the rotors and the base to find an optimal length, improved stiffness of the material linking the rotors to the base to potentially reduce vibrations and improve stability, and different shapes, materials, and configurations for the rotor blades. Pushing the quad-rotor concept further, additional rotors could be added to the platform. Those rotors could either be of the same size, or could be smaller depending of the function those additional rotors would perform. Extra rotors would be well suited in the case of a multiagent type of controller since adding rotors would translate to adding agents to the system and could easily be implemented on an existing controller. Additional rotors would provide an added payload capacity as well as redundancy in case of a failure of one or more of the rotors that multiagent control systems could handle without fully loosing control of the system. An important question

would then be where to add the rotors in order to have an optimal configuration. They could be placed in between the existing rotors keeping a square shape or maybe they could be placed closer to the center of the vehicle. Simulation test could be performed to find an optimal position. Going along with the same logic, some or all rotors could have an extra degree of freedom where the could slide closer or further from the center of the vehicle. The added degree of freedom could be incorporated in the control strategy where some agents could have an extra action that they could perform which would be getting closer or farther from the center. The optimal configuration would then be learned by the multiagent control system which in turn could use this extra degree of freedom in case of a failure from one or more rotors. The control system would then modify the placement of the different rotors depending a particular flight maneuver or failure of one of the components. A potential way to increase the flight speed and make some maneuvers go smother would be to add the possibility of tilting the rotors in one or two directions. As before simulation tests could be run to find good orientation but that would be better approached by having the control system learn the optimal orientation of the rotors depending on the maneuver and/or failures.

Several features or configurations could therefore be implemented on this platform either by adding them individually and testing the system or by adding several at a time and conducting tests. Those potential features are four or more rotors (an even number of rotors would probably lead to better results) with either the same size or a different size blade, the possibility to modify the position of the rotors in the 2D plane, and the possibility to tilt the rotors in two different direc-

tions. The multiagent control system could then learn how to use those feature in an optimal fashion. Agents would then choose between the following actions: rotor speed and direction, position of the rotor in 2D space, and orientation of the rotor along two axis.

Another important aspect to consider for this problem is the sensor package. The outputs of the control system have been defined previously and correspond to the actions that the agents can take. The inputs of the control system play critical role in whether or not the controller will be able to stabilize the system and perform the different flight maneuvers. It is therefore very important to choose what kind of inputs to feed to the control system and also what kind of sensors to use. As for the configuration of the platform, it would be best to start with a ready to use configuration and then modify or add sensors depending on the configuration needs. On such a platform, size and weight of sensors will be some of the limitations in choosing the sensors. Some of the base sensors include Micro-Electro-Mechanical Systems (MEMS [42]) based inertial sensors that include accelerometers and gyroscopes, as well as sonar-based sensors. A small onboard computer also needs to be included in the electronics package. An example of MEMS inertial measurement unit would be the Crista IMU that was used in [44] for the study on quad-rotor MAVs. Those sensors could be used in a preliminary study for providing inputs to the control system. If results are good, those sensors could be kept as the base of the sensor package, and if not, further investigation would be needed to find more efficient and better quality sensors that still meet the size and weight requirements of the quad-rotor platform. Later on, a good addition to the sensor package would

be one or more small, light-weight, black and white cameras. Those could be used as an input to the control system to improve stabilization of the vehicle, avoid obstacles, and provide information about the environment to the ground station. A GPS system could potentially be used but if the use of the vehicle in mainly indoors, the GPS system will provide limited or no information to the system. If the application requires the identification of heat sources (vehicles, persons) then a heat sensor could be added if size and weight requirements are met.

Challenges for such a problem would mainly be the learning process of the control algorithm. The platform is commercially available, and the modifications discussed could be implemented without too much difficulty and would therefore pose only minor problems. The platform is however very unstable with very fast dynamics that would present important difficulties to the reinforcement learning techniques. Several simulations, tests, and tuning would be necessary before achieving any results on this kind of platform. A dynamic model and simulation environment is presented in [44] and could be used to start the learning process on the basic quad-rotor configuration. Modifications on the platform could be implemented with modifications of the quad-rotor simulator. Reinforcement learning applied directly would probably take a very long time to learn and speeding up the learning process would be necessary to obtain meaningful results. This could be improved by having the algorithm first learn the existing stabilization controller from commercially available platforms and then extending the learning from there. Setting guidelines for the algorithm and restricting the range of actions of each agent could also possibly speed-up learning. Another way to speed up the learning

would be to use a similar learning technique as Ng [1] where the algorithm learned to fly a helicopter from a model obtained from real flight data.

Overall, the quad-rotor platform presents a very high degree of flexibility and the configuration can be modified extensively. This platform also lends itself very well to the application of diverse multiagent control techniques. Standard control methods only saw limited results on this kind of platform and no study has been conducted on using multiagent techniques on this type of vehicle. Also existing quad-rotor vehicles could be used as a first approach for implementing multiagent techniques and the platform could be highly modified in future studies to improve performance and efficiency with a control system that would adapt very well to those kinds of modifications.

2.3.2 Flexible-wing MAV

Other types of missions such as surveillance, reconnaissance, search and rescue, enemy targeting could require a vehicle still small in size (under 24 inches) to avoid detection but that would have a longer range, higher maximum speed, and higher maximum altitude. MAVs would be well suited for those types of missions and could provide very valuable information.

Flexible-wing MAVs have several advantages over fixed-wing MAVs including more resistance to wind gusts, improved payload, and a much more flexibility of the platform that could be modified based on application requirements. Flexible-wing MAVs would therefore be a good platform for implementing multiagent control

techniques. As discussed in the previous section, speeding up the implementation and testing process could be done by using an already existing platform. A platform well suited for this purpose could be the GENMAV platform discussed in section 1. This platform has been studied, and tested and provided good preliminary results. Also, several research teams will use this platform in the future and more test results and possible modifications and improvements will therefore be available.

Multiagent control methods could be applied to the GENMAV platform with minor modifications and further improvements could be investigated once the project is under way. GENMAV would be a challenging platform due to unstable flight characteristics but the high flexibility would be a great leverage for improving stability and performances. A preliminary research could focus on applying multiagent control techniques with agents on the tail section of the vehicle. The rudder could be cut in 4 segments with an actuator on each section. Similarly, the elevator could have four sections with actuators on each. Each actuator could then be used as an agent and his possible actions would be the orientation of the elevator or rudder section. The onboard sensors would provide information to the agent that would then use it to stabilize the system and perform flight maneuvers. Reinforcement learning and evolutionary computation are a natural approach for a problem where an explicit model of the system and its environment cannot be obtained which results in the optimal solution being unknown. Several standard reinforcement learning method could be used as a first approach but in case those methods weren't able to provide expected results, improved method

could be applied to achieve better performance such as QUICR-Learning [8], IFSA (Incremental Feature-Set Augmentation) [9], and WoLF Policy Hill-Climbing [21].

The first configuration could then be compared with a modified configuration that would include adding two more actuators to morph the shape of the wing with a string attached to the end of the wing as was done in [36]. The two additional actuators would add two more agents to the system and their actions would be how much to pull on the string which would result in more or less morphing of the wing. Morphing the wing was shown to improve and stabilize maneuvers and combined with the previously defined control system could provide very interesting results and improvements. More flexibility could be provided by attaching more strings in different parts of the wing to provide different modes of deformation providing more agents to the control system and possibly a better configuration for some particular flight maneuvers.

Another similar approach could involve morphing the wing with a different method. Carbon fiber bands are used on the wing to reinforce it and provide rigidity. Those could be replaced by materials that could be extended or contracted by the control system. Morphing of the wing would then be done without additional strings which would provide a more natural morph of the wing with potentially better results during flight maneuvers. A possible material could be a piezoelectric material (crystals and some ceramics) that stretch an contract when an electric potential is applied. Another idea would be to install either a micro-hydraulic arm or a set of very thin strings in order to morph the wing.

A potentially useful platform flexibility could be the possibility to change the

dihedral angle and the orientation of the wings during flight. Both wings could attach to the fuselage through a pivot point and two additional agents would have the task of orienting the wings for maximal stability and efficiency during flight and maneuvers.

Similarly to the quad-rotor problem, the sensor package will play a important role in how fast and efficiently the algorithm will learn to control the system, the efficiency of the controller after the learning process, and in how well the controller will perform flight maneuvers. The basic sensor package should include small and light-weight accelerometers, gyroscopes, and GPS system. Additionally, a small black and white or color camera could be added for providing information to the user as well as to be potentially used by the control system to navigate, stabilize the vehicle and avoid obstacles. The potential usefulness of sonar-based sensors could also be investigated and potentially used by the control system.

Challenges of the flexible-wing platform include a platform that is not very stable. The algorithm would have to learn on an unstable platform with a fair amount of information available. Selecting what information could be more important could also help with the learning process. Similarly to the quad-rotor problem, simulations and tests would need to be performed before achieving relatively stable flights. Most platform modifications should not be too difficult except for the bands used for wing rigidity that could expand and contract. Several different materials would need to be investigated for a potential application to the GENMAV platform. This type of materials would need to be light-weight with the property of expanding and contracting when necessary. A modified F16 simulator

is used to simulate the GENMAV platform and could also be used with minor modification to start implementing multiagent control techniques. Some platform modifications could be implemented in the simulator while some others could be tested with a simpler configuration (wing in a wind-tunnel for example).

GENMAV would be a good flexible platform form implementing a multiagent control strategy. Minor modifications would allow an almost immediate implementation of a reinforcement learning algorithm that should provide interesting results and potentially really good flight performance. Further research could include deeper modifications of the platform with a control strategy that could adapt and keep learning to improve the overall performance of the system.

2.3.3 UAV

Some missions or application may require a long range and high altitude of operation. For those types of conditions, large UAVs should be considered. Large UAVs are the size of a small aircrafts and are much bigger than MAVs. Large UAVs have a more conventional style and are less flexible than MAVs. The idea of using a multiagent form of control on large UAVs is not new but hasn't been fully developed and there is still room for improvements.

A UAV with segmented control surfaces was tested by Abdulrahim [5]. The UAV used had a similar shape to a Russian MiG-27 with a wingspan of 5.5ft, and was only remote controlled. The wing ailerons were divided into 16 independent control surfaces that each had their own actuator. A reconfigurable controller was

developed to actuate all 16 servos depending on the configuration used. Flight tests shown promising results and improved performance over the unmodified aircraft. An actuation mode was chosen for each flight test and programmed into the controller in between the flights. Those tests demonstrate the concept of segmented control surfaces and provided good preliminary results but provided no method for finding an optimal actuation modes and an optimal controller for the system. Testing one configuration at a time is a very tedious process that would take a very long time and the optimal solution may never be achieved with this approach. The potential of segmented control surfaces was however demonstrated and would provide very good results with a better approach.

A better approach for controlling segmented control surfaces was presented in [15, 14] where the control technique was based on the theory of collectives. 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 probability collectives (PC) were implemented and are based on agents consisting of a actuator, sensor, and logic package taking actions and receiving rewards based on those actions. Up to 8 actuators were implemented on the platform. Two controllers were developed using PC and aimed at augmenting the vehicle stability, and lightening the longitudinal response to random vertical wind gusts. Promising results were obtained and are an encouraging step toward multiagent based control of UAVs.

Even though advances have been made in the area of multiagent control of UAV platforms, much more can be accomplished and investigated. Results are encouraging and proved that this approach works and can produce very good results. Further work in this domain could include a more conventional platform such as the UAV presented in [3] with many control surfaces (16 could be a starting point). Form there, different reinforcement learning algorithms could be applied using each actuator as an agent that would work toward achieving the global system goal which could be improving stability in a first step and then improving flight maneuvers and recovering from potential individual actuator failures in the system. Once again, a simulator would be needed such as the one presented in [71] to run multiple simulations and have the control algorithm learn from it. Simpler tasks would allow the algorithm to learn faster and then more complex tasks could be introduced to improve the controller. The UAV platform is a good starting point to test and show the value and potential of a new technique that could potentially be scaled down later on for applications on a smaller MAV platform.

2.4 Discussion

This chapter reviewed the different MAV/UAV platforms and the different control strategies implemented to control such vehicles. The rotor-based vehicle is mainly be used indoor or in urban environment at low speed for short range missions. Advantages of the vehicle are a high configuration flexibility (quad-rotor vehicle), and its ability to hover and maneuver at very low speeds. Disadvantages would

include high instability and difficulty to control as well as relatively low speed of operation. The fixed/flexible-wing MAV is used for a longer range of operation at higher speeds and on longer distances. Benefits of the vehicle include platform flexibility (higher flexibility for the flexible wing MAVs) and more stable flight characteristics than rotorcrafts. Its drawback include higher flight speeds that require faster processing of information and its size makes it susceptible to wind gusts. Finally, the UAV platform is used for long range reconnaissance at high altitude and faster speeds. Its advantages include stability of the vehicle, easier to obtain a mathematical model, and implementation of distributed effectors easier than on MAVs. Drawbacks of UAVs include its size making it easier to detect and harder to launch from any location.

Control strategies implemented on MAV/UAV platforms include model based controllers such as PID and state space control methods as well as techniques based on learning algorithms. Model-based methods work well when the system is well known and can be modeled. For non-linear systems, approximations are needed and do not always provide optimal results. Model-based control techniques provide good results on platforms such as UAV where the dynamics are know and understood and accurate models can be obtained. For MAV control, where the system is highly non-linear and very unstable, those methods require approximations and tuning which could be a laborious process and may not provide an optimal controller. In these situations, a learning method such as reinforcement learning that do not require a model of the system provides better results. The goal of the technique is to map a set of inputs provided by the sensors to a set of outputs

which are actuator positions in order to achieve an optimal solution. Learning methods are shown to provide promising results in the MAV control domain.

Several multiagent control methods were also reviewed. A wide range of methods have been explored but the primary techniques can be classified as reinforcement learning, evolutionary computation, and auction-based algorithms. Collectives are also part of the main techniques and are a higher level of multiagent methods that can be applied to lower level methods such as evolutionary techniques. Those learning methods have proven robust and efficient for controlling multiagent systems and have the added benefit of being applicable to non-linear systems without requiring a model of the system. As mentioned before those techniques are based on agents learning the correct actions based on a reward system. This reward system depends on the technique used and has a high influence on the speed and efficiency of the algorithm. Many improvements have been proposed on classical learning methods leading to faster and more robust algorithms that can also work in noisy environments and when failures occur in the system.

These multiagent control techniques 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 multiagent control system to stabilize the system and perform flight maneuvers. In this dissertation, multiagent control techniques are applied to a fixed-wing MAV platform with agent represented by actuators where control surfaces are segmented into multiple elements. Learning is done with a simulator (JSBSim) on a basic architecture to

provide a working control system. Chapter 3 describes the MAV platform (GEN-MAV), the system dynamics, and the basic controllers used in the experiments. Chapter 4, 5, and 6 present the results for the different experiments conducted.

Chapter 3 – MAV Model, System dynamics, and Basic Control

3.1 MAV Platform: GENMAV

The platform selected for these experiment is GENMAV [78], an MAV developed by the Air Force Research Laboratory Munition Directorate (AFRL/RW). GENMAV was developed to provide a base configuration that researchers could use and modify when implementing design and/or control techniques. GENMAV is a good MAV reference designed to be used for future research where a wide range of techniques can be applied and tested while some of the basic characteristics are known and available. GENMAV is also a flexible platform that could be modified depending on a particular application or technology.

Characteristics of GENMAV (Figure 3.1) include a 24 inch wingspan with a 5 inch chord, circular fuselage 17 inches long, and a dihedral angle of 7 degrees. The wing design was modified from previous versions in order to improve low speed performance. The tail section, originally a V-tail, then conventional tail, is now an elevon tail. An electric motor is used combined with a lithium polymer battery. Aerodynamic characteristics were obtained using the vortex-lattice method aero-prediction code AVL (Athena Vortex Lattice) and detailed data can be found in [78]. 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.

In Chapter 4, we modified GENMAV to include a greater number of control surfaces. This version of GENMAV includes a conventional tail with a rudder and elevator. As a first step, only the tail section was modified with the elevator broken down into multiple control sections. Test configurations include 4 control sections on each side of the elevator, for a total of 8 control surfaces. A configuration with 12 control surfaces was also tested (6 control sections on each side of the elevator).

In Chapter 5, the elevon tail version of GENMAV is used. No modifications are done and the different controllers use both elevons for altitude and roll control.

In Chapter 6, the GENMAV model uses ailerons and an elevator. GENMAV was modified so that simulation experiments with segmented ailerons could be conducted. 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.

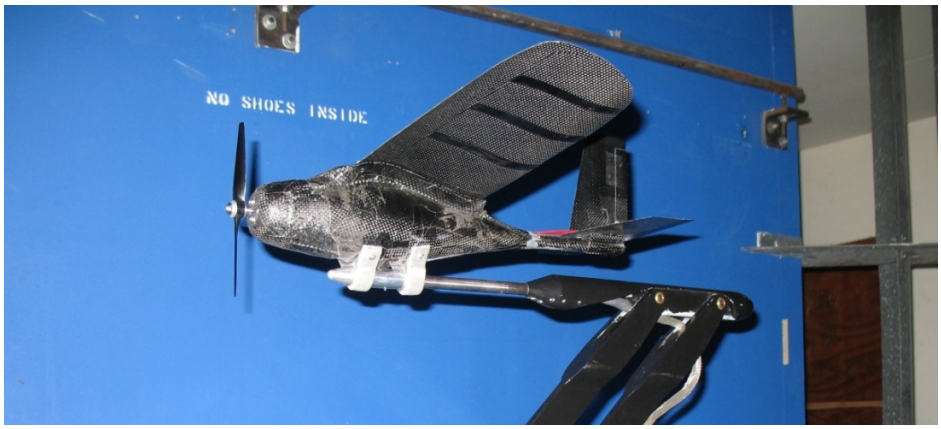

Figure 3.1: GENMAV Prototype [78]. 24in wingspan with 5in chord, 17in fuselage, and 7 degree dihedral angle.

3.2 Neuro-control for MAVs

The control of GENMAV is achieved through a feed-forward neural network using a neuro-evolutionary algorithm [6, 40, 60, 83]. The neuro-controller 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 as well as keep the control inputs within acceptable values using the parameter's derivative. The near optimal neuro-controllers are then saved and used for flight control where different desired altitudes and heading are achieved.

A simple neuroevolutionary algorithm was developed using the techniques outlined in [86]. 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 begin discarded.

In Chapter 4, the single hidden-layer, feed forward neural network [16] used has 6 inputs which correspond to the total forces and moments applied to GENMAV. The 8 outputs of the neural network are the angles of the elevator control surfaces. The experiment was conducted with 8 and 12 control surfaces on the elevator, each of which could move independently between -30 and +30 degrees.

For experiments in Chapter 5 and 6, the single hidden-layer, feed forward neural networks [16] 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 neuro-controllers for the basic configuration (no segmentation): altitude control, roll control, and heading control. Each neuro-controller 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 10 outputs corresponding to the 10 aileron segments. The multiagent controller setup includes additional neuro-controllers so that each control surface is adjusted by its own independent controller. A total of 14 neuro-controllers are used for the control surfaces and an additional neuro-controller is used for heading control.

3.3 System Dynamics: AVL / JSBSim

In Chapter 4, the neuro-controllers are coupled to the Athena Vortex Lattice (AVL) software package which is an aerodynamic prediction code based on a vortex-lattice method. AVL is used to estimate the aerodynamic characteristics of GENMAV under different conditions and configurations. The output of AVL includes the forces and moments for the entire configuration as well as the lift and drag coefficients.

The simulation runs consist of providing forces and moments as inputs to the

neuro-controllers, obtaining angles for elevator control surfaces from its outputs, running AVL to provide the resulting aerodynamic parameter values, computing the objective function, and having the neuro-controllers learn from the objective function.

The JSBSim experiments are conducted in Chapters 5 and 6. To conduct the experiments, the neuro-controller and PID controllers were coupled to JSBSim [13], 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. Some of the classes encapsulate concepts or mathematical constructs such as the equations of motion, a matrix, or a vector. Some classes manage collections of other objects. 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 neuro-controller, 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 neuro-controller learn from the

objective function. Figure 3.4 and 3.5 show the overall control system setup and the neuro-controller training diagram respectively.

The JSBSim model of GENMAV was obtained by computing the aerodynamic coefficients using the Athena Vortex Lattice software package (AVL)[30, 11, 23, 73, 35]. Vortex lattice methods are based on Prandtl's classical lifting line theory and assume an incompressible, irrotational, inviscid flow. AVL uses a file containing the geometry of Genmav where lifting 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 model is described in an XML (Extensible Markup Language) file so that model information and specifications can be organized in a structured document. The XML file is divided into several sections including: general section, metric section, mass balance section, ground reaction section, propulsion section. The engine and thruster information is located in 2 separate xml files which makes it easy to switch between different engines and thrusters. The JSBSim GENMAV model [76] is then loaded at the beginning of the simulation and used for the different experiments.

3.4 MAV Control

The controllers presented is this section are used in Chapters 5 and 6. 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 used the error between

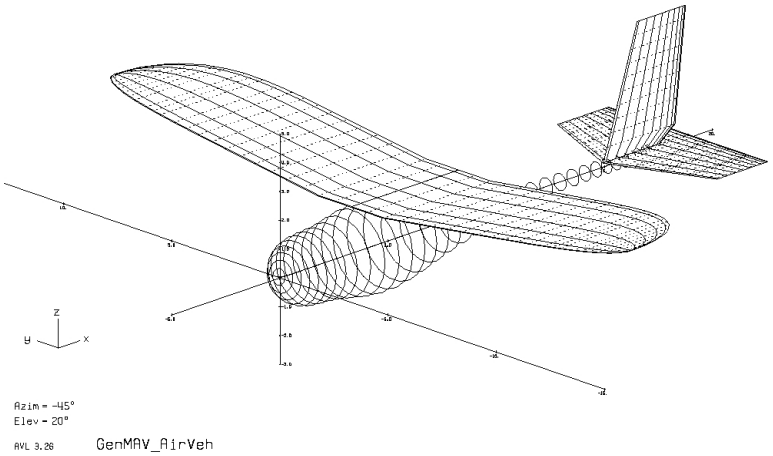

Figure 3.2: GENMAV in AVL, the aerodynamic prediction code. AVL calculates forces and moments applied to the MAV as well as the lift and drag coefficients.

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.

Figure 3.3 shows a block diagram of a PID controller where the control command is calculated with three separate parameters: the proportional, the integral and derivative values. The proportional term (Equation 3.1) is directly proportional to the error, the integral term (Equation 3.2) is based on the sum of previous errors and is used to correct small drift over time, and the derivative term (Equation 3.3) 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.

$$P = K_P.e(t) \tag{3.1}$$

$$I = K_I.\int_0^t e(\tau)d\tau \tag{3.2}$$

$$D = K_D.\frac{de(t)}{dt} \tag{3.3}$$

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

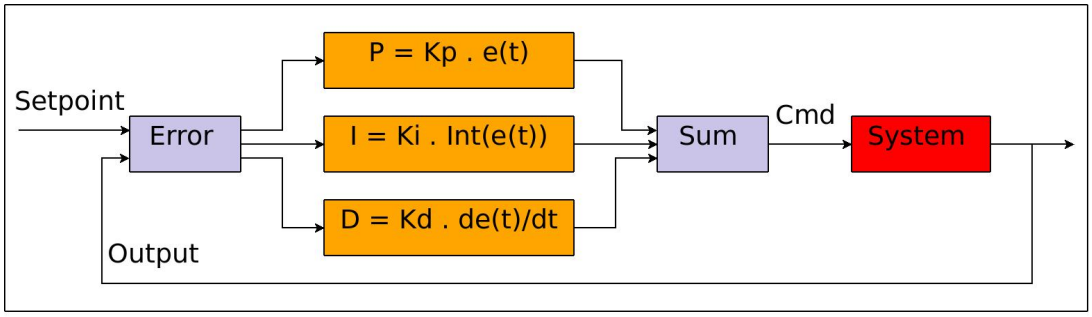

Figure 3.3: PID Controller. The weighted sum of the proportional, integral and derivative terms is used as the control command.

Figure 3.4 shows the control system block diagram that includes interactions with the simulator and the user or navigation algorithm. The user or naviagtion 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.

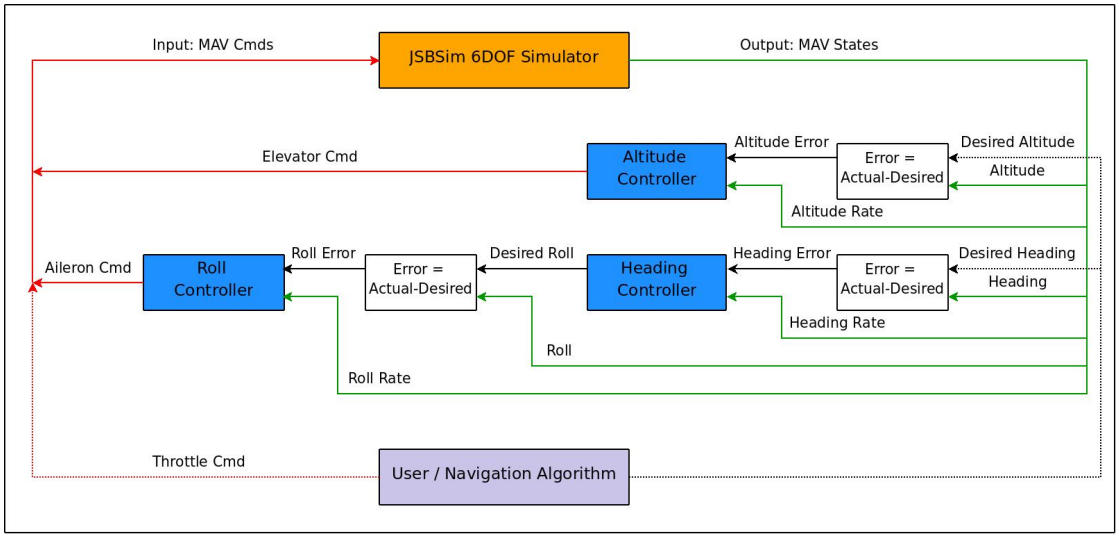

Figure 3.4: GENMAV Controller. Desired altitude and heading are provided by the user or navigation algorithm. The controller calculates the elevator and aileron commands that are fed to JSBSim which in return provides the MAV states.

The neuro-controllers configuration is done similarly with the same inputs and outputs as the PID controllers. Training the neuro-controllers is achieved using the different objective function from Section 6.2. Figure 3.5 shows the training cycle for one controller where the neuroevolutionary algorithm selects and mutates a neural network from the pool, the neural network controls the system for a short period of time using state information from JSBSim as input and outputing a control command to the simulator. The objective function is then used to calculate the reward that the neural network receives based on its flight performance. The algorithm then repeats this cycle until a near optimal solution is found. The neuro-controllers are then saved and used directly in the control loop with JSBSim.

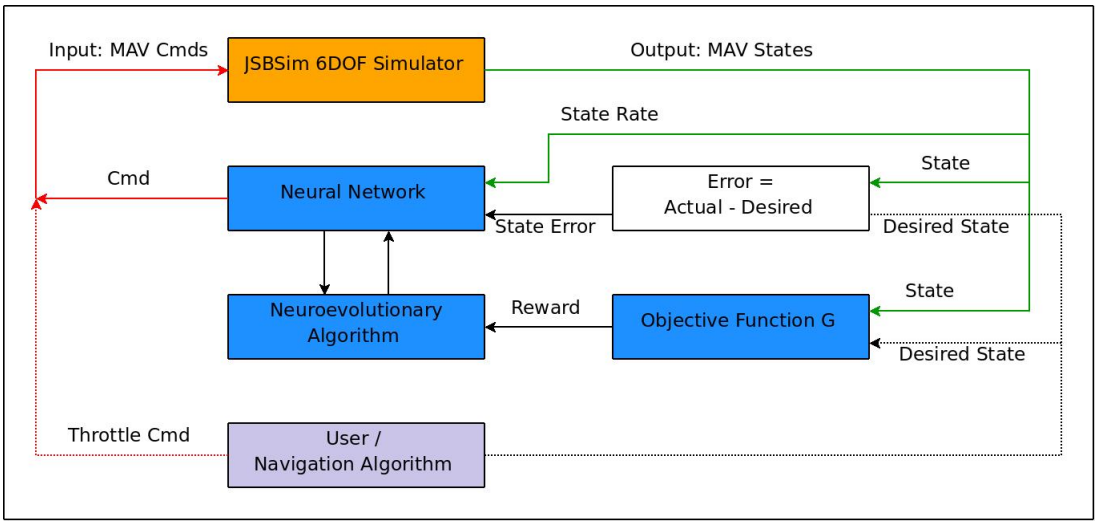

Figure 3.5: Neuro-controller training. A neural network is selected to control the system using the MAV state information provided by JSBSim. The neural network then receives a reward calculated with the objective function. The loop continues until an optimal solution is found.

Control of the segmented aileron version of GENMAV is similar to the unsegmented version except that the roll neuro-controller includes 6 outputs corresponding to the control commands sent to each aileron segment pairs instead of the single output. Altitude and heading control do not change with a two inputs, one output neuro-controller for each controlled parameter.

The multiagent neuro-controllers are setup with each neuro-controller commanding an aileron segment. Inputs are the same as before but the output directly control the position of a control surface. Twelve neuro-controllers control the aileron segments while two control the elevons for a total of fourteen neuro-controllers giving maximum flexibility to the control system.

Chapter 4 – A Neuro-evolutionary Approach to Control Surface Segmentation for MAVs

In this chapter with use the GENMAV platform described in Section 3.1, the neuro-controller described in Section 3.2, and the aeroprediction code AVL decribed in Section 3.3 to implement segmented control surfaces on an MAV.

4.1 Contribution of this Chapter

In this chapter, we show that neuro-evolutionary techniques can be used to control multiple surfaces to improve the flight characteristics of an MAV by designing appropriate objective functions (e.g., roll moment value). Section 4.3 shows the experimental results where drag on the MAV was reduced by up to 4%, and Section 4.4 discusses the relevance of the results and highlights directions Chapter 5.

4.2 Objective Functions

An important part of using neuro-controllers consists of designing an objective function that allows the neuro-controller to learn at a satisfactory rate and at the same time achieves the system's objective. The objective function used for these experiments is divided between meeting the target value of the desired

forces/moments and minimizing the actuator angles of the different control surfaces of the elevator (8 and 12 controls surfaces total). For all results presented, the lift does not vary significantly throughout the different experiments.

4.2.1 Minimizing the Drag

The first objective function G_{F_d} (See Results in Section 4.3.1) is calculated using the drag and roll moment desired value.

$$G_{F_d} = \alpha C_1 \left[C - \left(\frac{L_a - L_d}{L_d} \right)^2 \right] + (1 - \alpha) C_2 [F_{d_M} - F_d]^2 \tag{4.1}$$

Where C, C_1, and C_2 are normalization constants with values of 9, 200, and 484000 respectively, F_d is the force of drag, F_{d_M} is constant and equal to 0.142, L_d and L_a are the desired and actual roll moment values. In this case, the optimal value of α is also 0.998. The drag calculation is done internally in AVL.

4.2.2 Minimizing Actuator Deflection Angles

The second objective function G_{ω_1} (See results in Section 4.3.2) used is calculated using the actuator angles and roll moment target value.

$$G_{\omega_1} = \alpha C_3 \left[C - \left(\frac{L_a - L_d}{L_d} \right)^2 \right] + (1 - \alpha) C_4 \sum_{i=1}^{N} [\omega_M - \mid \omega_i \mid]^2 \tag{4.2}$$

Where L_d and L_a are the desired and actual roll moment values, ω_i is the

deflection of the control surface (with a maximum deflection of $\omega_M = \pm$ 30 degrees for each actuator), and C, C_3 and C_4 are normalization constants with values of 9, 200, and $2/N$ respectively, and N is the number of control surfaces. For these experiments, α needs to be 0.998 or above, otherwise the roll moment target value cannot be reached.

4.2.3 Minimizing Relative Angle Between Control Surfaces

Another similar objective function G_{ω_2} that was used was designed to minimize the actuator deflection angles relative to each other.

$$G_{\omega_2} = \alpha C_5 \left[C - \left(\frac{L_a - L_d}{L_d} \right)^2 \right] + (1 - \alpha) C_6 \sum_{i=1}^{N-1} \left[\omega_M - \frac{\mid \omega_i - \omega_{i+1} \mid}{2} \right]^2 \qquad (4.3)$$

Where C, C_5, and C_6 are normalization constants with values of 9, 200, and $2/N$ respectively, L_d and L_a are the desired and actual roll moment values, ω_i is the deflection of the control surface (with a maximum deflection of $\omega_M = \pm$ 30 degrees for each actuator), and N is the number of control surfaces.

4.3 Experimental Results

In order to evaluate the impact of using multiple control surfaces and training a neuro-controller to optimize the control surface angles, we performed the following experiments:

- The basic configuration consisted of a GENMAV with one control surface each for the elevator (2 control surfaces). This was used to obtain the aerodynamic parameter values that are used as a reference.
- A neuro-controller was used to control multiple control surfaces to explicitly minimize the:
  - drag (Section 4.3.1)
  - actuator angles (Section 4.3.2)
  - relative actuator angles (Section 4.3.3)
- Finally, a neuro-controller was used to control the system if the event of an actuator failure.

These results are based on 12 runs. A sweep of the neuro-controller parameters was conducted as a preliminary study in order to find values of the neuro-controller parameters that provided satisfactory results in terms of learning and optimization of the objective function. The neuro-controller is configured with 12 hidden units, a pool size of 4, 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 4.4.

4.3.1 Neuro-Controller Evolved to Explicitly Minimize Drag: G_{F_d}

We explored the potential to explicitly reduce drag by incorporating a drag term in the objective function that the neuro-controllers aims to optimize. This objective function (described in Equation 4.1) directly accounts for drag and roll moment target values but does not include the actuator angles.

Figures 4.1, 4.2, 4.3, and 4.4 present the results for the drag data with G_{F_d}. Figures 4.1 and 4.2 show an example of the elevon positions for the configuration with 2 and 8 control surfaces. While those solutions are similar to the ones found with G_{ω_1} and G_{ω_2}, the solution provided by G_{F_d} shows more symmetry in the elevon configuration.

Figure 4.4 shows the results for the drag with the 3 different objective functions G_{F_d}, G_{ω_1}, and G_{ω_2}. G_{ω_2} produces results that are similar to G_{F_d}, which indicates that minimizing the relative angles in between control surfaces can be used to indirectly minimize the drag. This is a particularly important result since the drag was available through the use of AVL, the aero-prediction code each time the elevon configuration is modified. The drag calculation for those configurations takes a significant amount of time and is usually not available directly when using flight simulators, and would not be possible for real flights of an MAV platform. Minimizing the relative deflections of the elevons can therefore provide a very good alternative to using the drag directly in the objective function calculations.

This solution is the intuitive solution that we would expect when trying to induce roll on the MAV while trying to minimize drag.

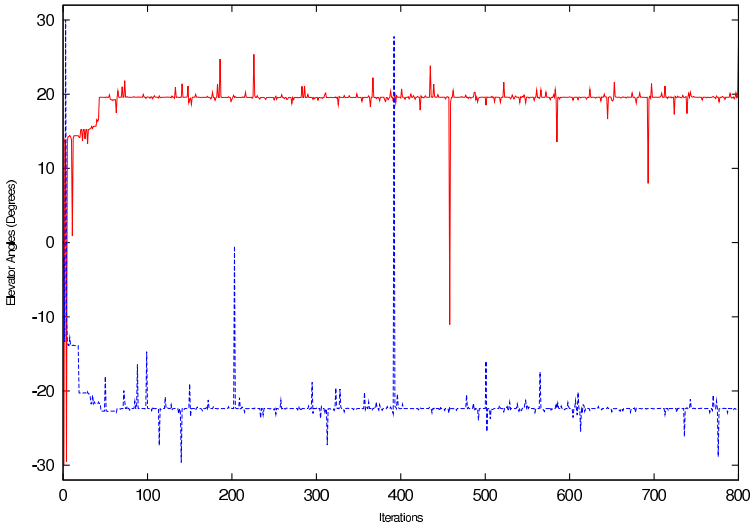

Figure 4.1: Elevon Angles (Min Drag, Roll Moment = 0.028). Example of elevon positions with 2 control surfaces. The learning is quicker than with 8 elevons due to lower level of complexity. Spikes represent different solutions since the learning is still active.

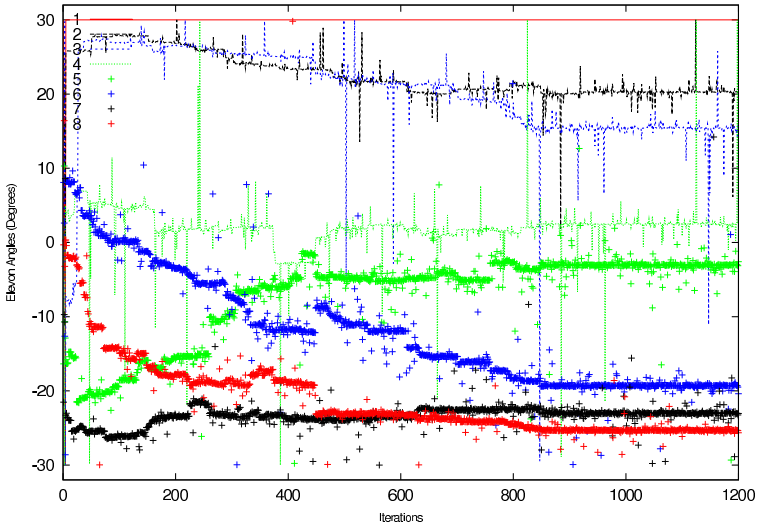

Figure 4.2: Elevon Angles (Min Drag, Roll Moment = 0.028). Example of elevon positions with 8 control surfaces. It takes longer to find the optimal elevon positions due to the higher complexity of the configuration.

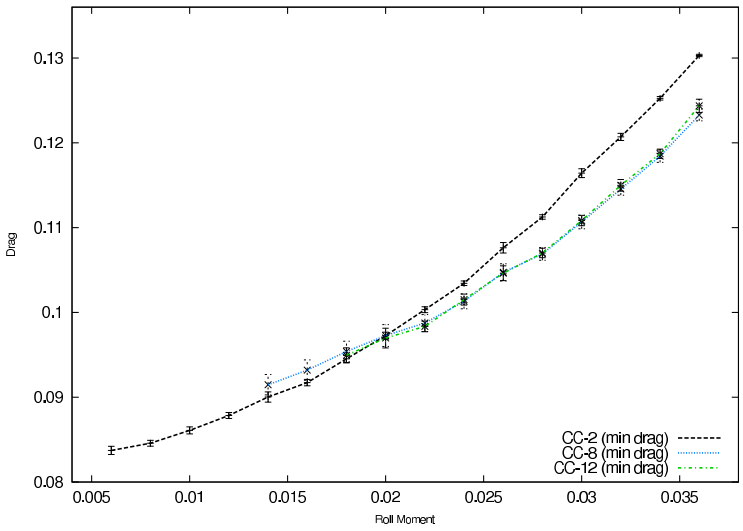

Figure 4.3: Drag vs Roll Moment (2, 8 and 12 Ctrl Surf). Configurations with 8 and 12 elevons perform significantly better than the base configuration with only 2 elevons. However, no significant difference is seen between 8 and 12 elevons

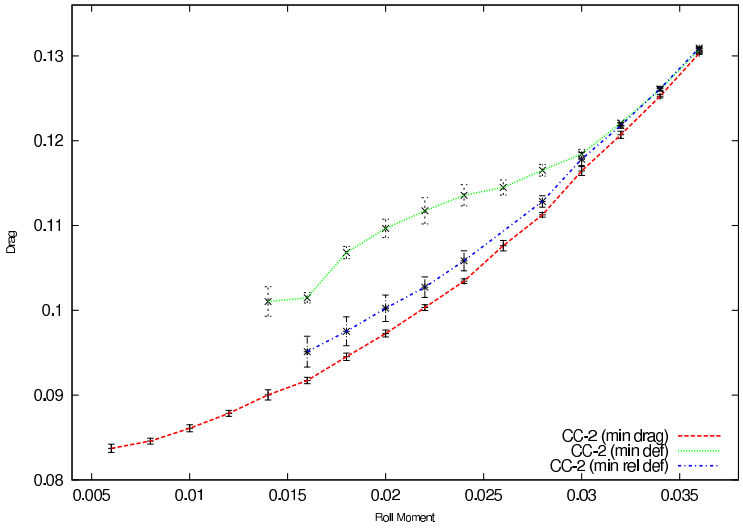

Figure 4.4: Drag between the three objective functions (2 elevons). The objective function minimizing the drag directly performs better than the other 2. However, the other 2 objective functions still perform well and can be used to indirectly minimize the drag.

4.3.2 Neuro-Controller Evolved to Minimize Actuator Angles: G_{ω_1}

Minimizing the control surfaces angles provides improved MAV flight characteristics such as smoother flight maneuvers which is an important benefit for MAVs. This section shows the results of experiments where several roll moment target values are achieved while at the same time the actuator angles are minimized.

Figures 4.5 and 4.6 show an example of the elevon angle values for a target roll moment value of 0.030. The elevon angles are progressively minimized as the neuro-controller learns the optimal solution for a particular desired roll moment. To achieve a roll moment target value of 0.030 with the standard configuration (2 control surfaces) requires the right and left elevons to move to 15 and -30 degrees respectively as shown in Figure 4.5. Figure 4.6 shows the elevon angles with the 8 control surfaces MAV configuration. This configuration reduces the elevon angles which allows for smoother maneuvers and does not require as much effort from the actuators.

A second and arguably more important benefit of segmented control surfaces is the potential for drag reduction. An example of drag results is shown in Figure 4.7 while Figure 4.8 shows the drag results for the configurations with 2 and 8 elevons. This particular MAV configuration coupled with the first objective function G_{ω_1} (Section 4.2.2) does not exhibit any significant drag reduction suggesting that another objective function might provide better results. Another intuitive solution would be to minimize the relative angle between a control surface and its 2 direct neighbors. This solution is presented in Section 4.3.3.

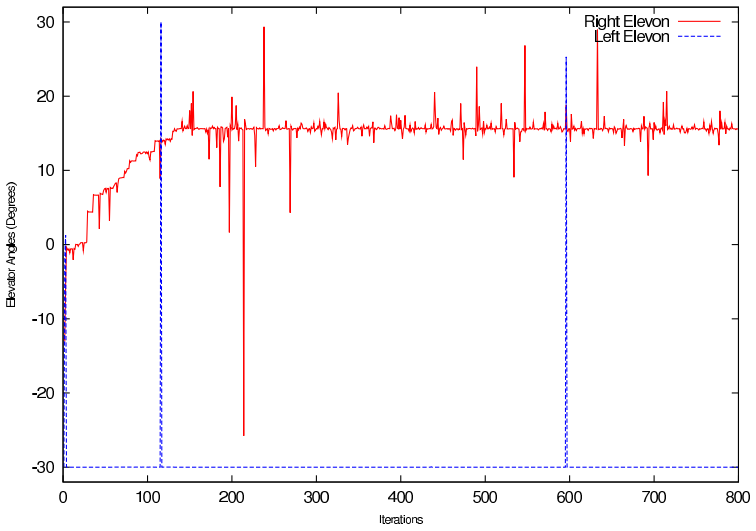

Figure 4.5: Elevon Angles (Min Angles, Roll Moment = 0.030). Example of elevon positions with 2 control surfaces. As was the case for minimizing the drag, the optimal positions are quickly found due to the lower level of complexity.

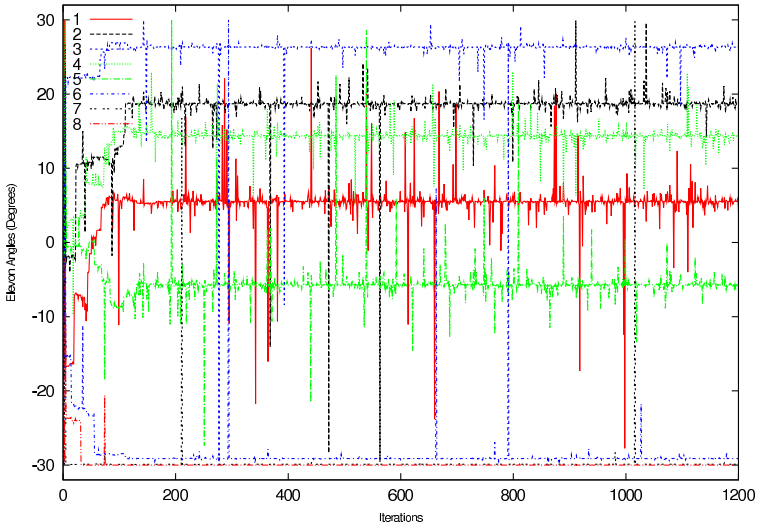

Figure 4.6: Elevon Angles (Min Angles, Roll Moment = 0.030). Example of elevon positions with 8 control surfaces. The learning with this objective function is quicker than with the drag objective function. The elevon positions are also nicely spread out.

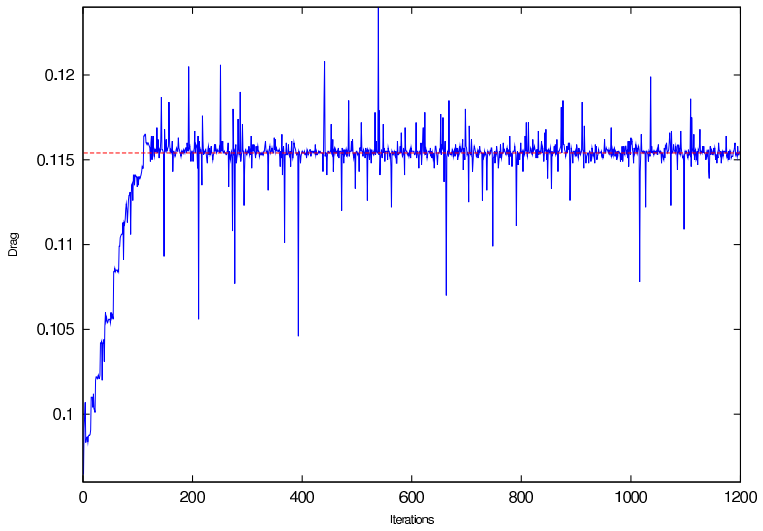

Figure 4.7: Drag with 8 elevons (Min Angles, Roll Moment = 0.030). Example of drag results when using the 8 elevon configuration. The drag slowly increases throughout the learning phase until the optimal solution is found.

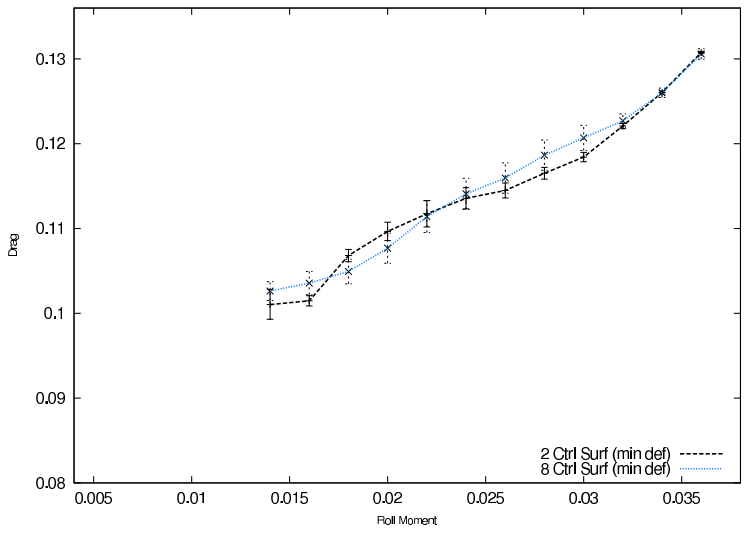

Figure 4.8: Drag vs Roll Moment (Min Angles, 2 and 8 elevons). Unfortunatly, no significant difference is seen for the drag results between the 2 and 8 elevons configuration. Another solution is presented in Section 4.3.3.

4.3.3 Neuro-Controller Evolved to Minimize Relative Actuator Angles:G_{ω_2}

Figures 4.9 and 4.10 show similar results as Section 4.3.2 with the second objective function G_{ω_2} presented in Section 4.2.3. The actuator angles are minized for smoother flight maneuvers. As with the first objective function G_{ω_1}, no significant drag reduction can be observed between the configuration with 2 and 8 control surfaces (Figure 4.11). However, G_{ω_2} induces significantly less drag than G_{ω_1} for some of the lower values of the roll moment as can be seen in Figure 4.12. G_{ω_2} would therefore be a better objective function than G_{ω_1} because it effectively minimizes the actuator while at the same time inducing less drag.

The results presented in this chapter are promising and show the potential for improving MAV flight characteristics by using a larger number of control surfaces. The control of such a modified MAV is possible with the use of a neuro-controller that if properly tuned and trained can provide optimal solutions to the MAV control problem.

4.3.4 Neuro-controller for Actuator Failure.

Results in this section show that it is possible for a neuro-controller to learn and adapt to changes in the environment, in this case failure of an actuator which changes the system's dynamics in order to remain in control of the MAV. First Figures 4.13 and 4.14 show that the target roll moment can still be achieved when an actuator fails by finding a new solution that compensates for the failure. This is the case for both objective functions, G_{ω_1} and G_{ω_2}.

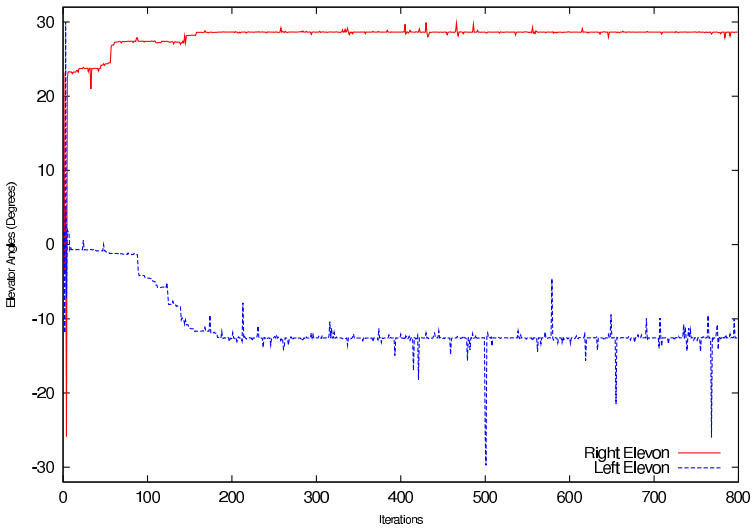

Figure 4.9: Elevon Angles (Min Rel Angles, Roll Moment = 0.028). Example of elevon positions with 2 elevons. The learning speed is slightly slower that when using the previous objective function but the solution is more symetric.

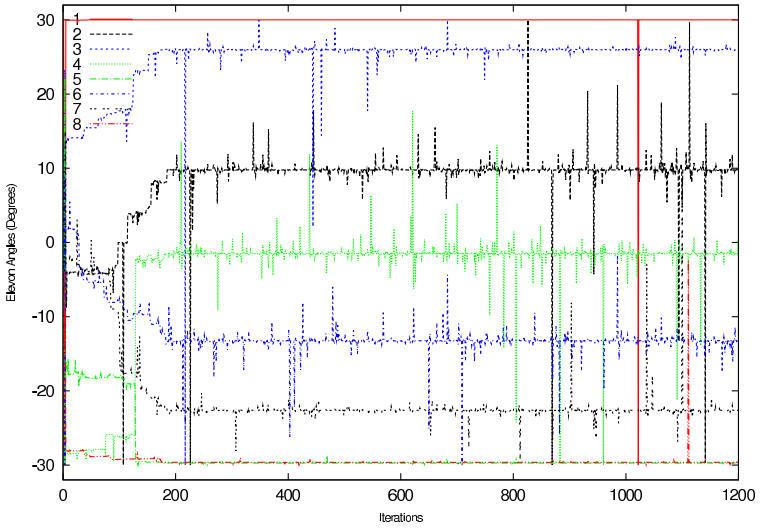

Figure 4.10: Elevon Angles (Min Rel Angles, Roll Moment = 0.028). Example of elevon positions with 8 elevons. The learning phase is fairly quick and the solution found after about 200 iterations.

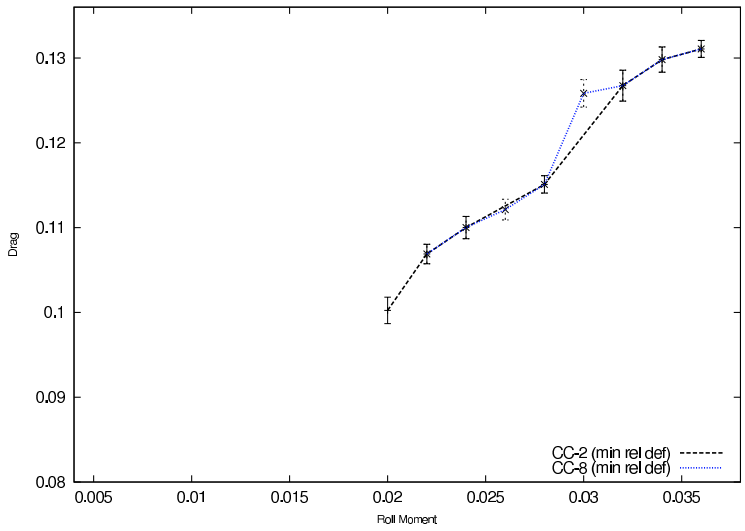

Figure 4.11: Drag vs Roll Moment (Minimize Rel Angles: G_{ω_2}). No significan difference can be seen in the drag results between the 2 and 8 elevon configurations. These results are similar to what was obtained with G_{ω_1}

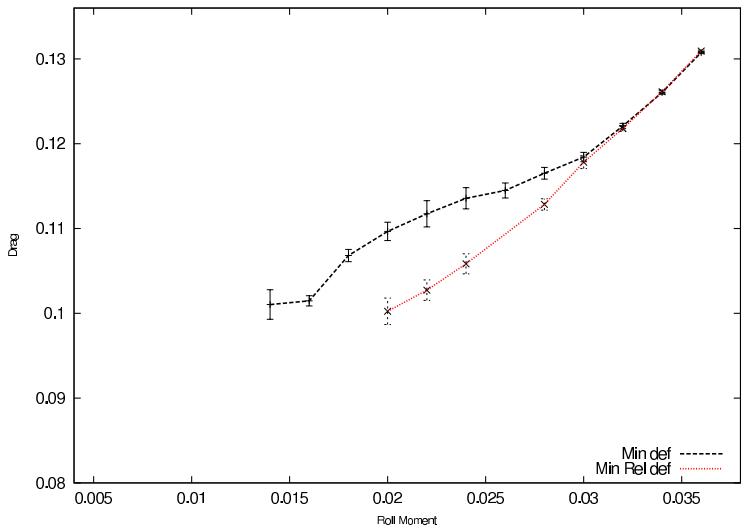

Figure 4.12: Drag: 1st and 2nd objective functions (G_{ω_1} and G_{ω_2}). A significant drag reduction can be seen when using G_{ω_2} instead of G_{ω_1}. G_{ω_2} is therefore a more efficient objective function for minimizing the drag.

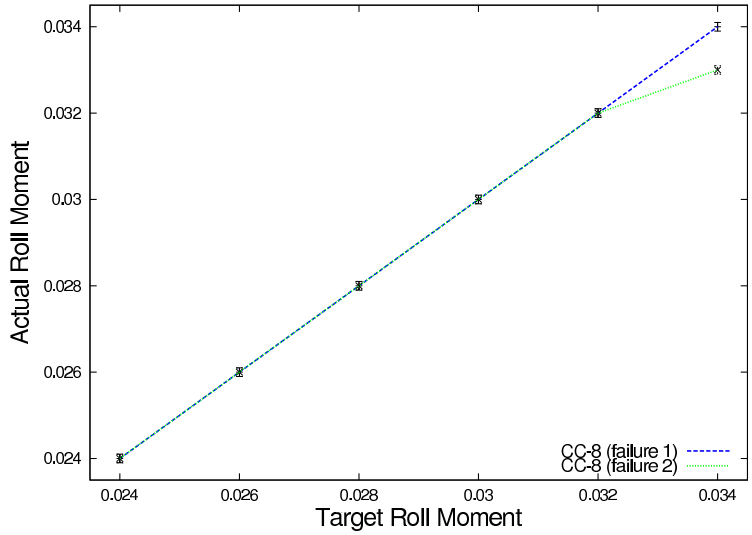

Figure 4.13: Target vs Actual Roll Moment with failures (G_{ω_1}). Even with failures in the system the neuro-controller using G_{ω_1} was able to adapt, compensate for the failure and still achieve the desired roll moment values.

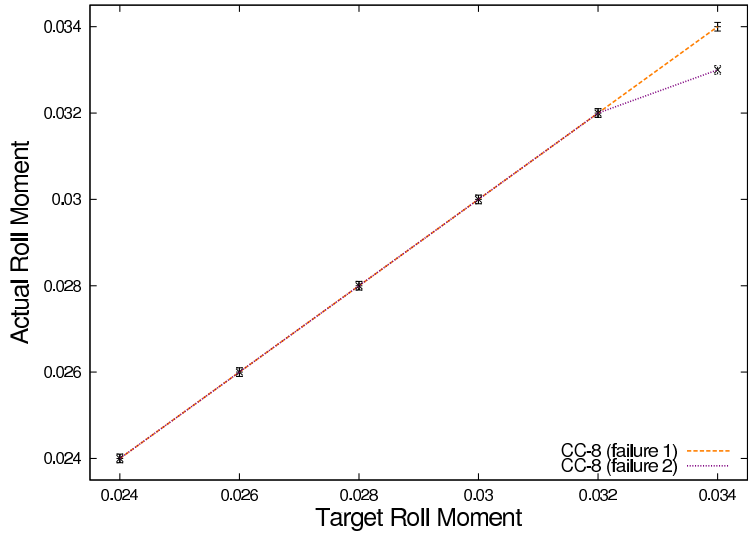

Figure 4.14: Target vs Actual Roll Moment with failures (G_{ω_2}). In the same as with G_{ω_1}, the neuro-controller using G_{ω_2} achieved the desired roll moment values.

Figures 4.15 and 4.16 show the results for the drag that result when comparing the system with and without failures. Results show that the drag is not negatively impacted when a failure occurs in the system. Drag is even slightly lower when a failure occur in some cases depending on how and where the failure occurs. This is primarily due to the objective function that in these cases minimize the actuator angles or relative angles between actuator which indirectly reduces the drag but does not necessarily always find the optimal solution for minimizing the drag.

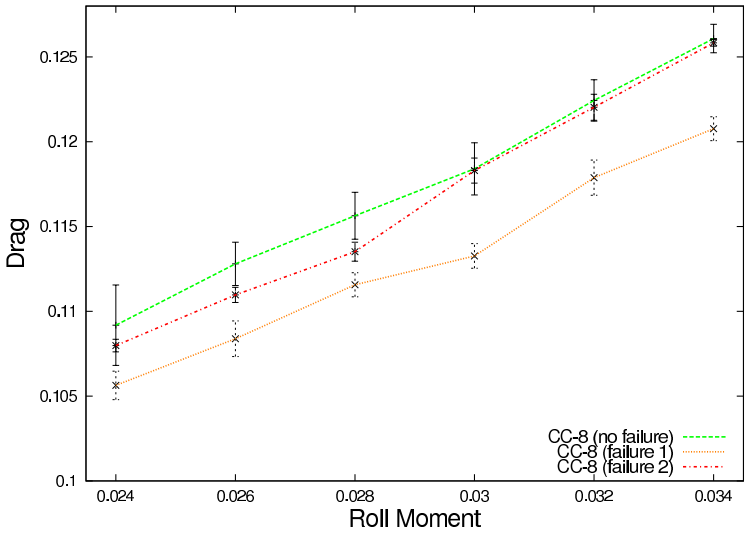

Figure 4.15: Drag Results: Failures (G_{ω_1}). The drag is not negatively impacted by failures in the system when the neuro-controller uses G_{ω_1}. In some instances, drag results are slightly better but it is only an small indirect benefit.

These results demonstrate that the control surfaces segmentation can be controlled by a neuron-evolutionary based controller in the event of an actuator failure therefore increasing the robustness of the platform. To be applicable on an actual platform, it is important to note that actuator failures need to be detectable by

the system through some type of sensing mechanism.

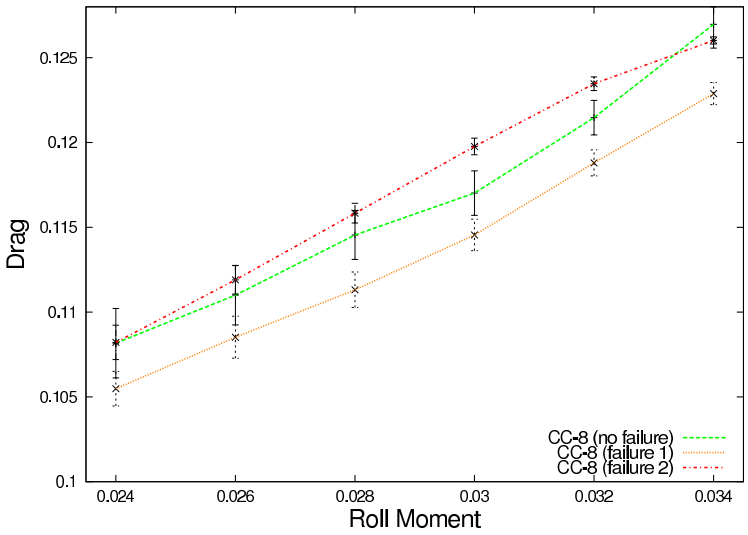

Figure 4.16: Drag Results: Failures (G_{ω_2}). Drag results here are very similar to what was obtained with G_{ω_1}. No negative impact of the failure on the drag and small improvements in some instances as an indirect benefit.

4.4 Discussion

MAVs present a new and promising 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 chapter presents a novel approach to control MAVs and provides improvement of the flight characteristics of such a platform by introducing a larger number of control surfaces on the elevon control sections.

Sections 4.3.1, 4.3.2,and 4.3.3 showed that controlling an MAV with a neuro-

controller was possible through segmented control surfaces. Using segmented control surfaces allows for smoother flight characteristics and flight maneuvers through minimization of actuator angles. Additionally, drag reduction of up to 5% can be seen for the larger values of the roll moment. If drag reduction is the objective, and if the drag is directly available the direct use of the drag in the objective function calculations provides the best results. However, if the drag is not available, minimizing the deflection between control surfaces still provides similar results and could be used instead. Results showed a drag improvement of up to 5% which was obtained by a gradual deflection of each actuator which lead to a smoother control effort. Also, simulations conducted with 8 and 12 control surfaces showed no significant differences between the 2 configurations which indicates that for this particular problem and configuration, 8 control surfaces is already the optimal configuration and increasing the number of control surfaces will not improve the drag or efficiency of the MAV. The solutions provided by the neuro-controller matches the intuition that a gradual actuator deflection would provide close to optimal solutions. Results presented in this chapter show the potential of such configurations to improve flight characteristics of MAVs that are inherently difficult to control. Neuro-controllers can effectively learn from the system and provide an optimal system's configuration therefore allowing such modifications on an MAV platform. Furthermore, such a configuration would provide a higher level of robustness to the system that could recover and adapt from potential failures of some elements in the system which is critical for completing the assigned missions as seen in section 4.3.4. The neuro-controller was able to learn and adapt to the new MAV configu-

ration that had a failure in the system and was able to provide a new solution for the control strategy in order to stay in control of the vehicle.

The results presented in this chapter are a first step that shows the potential of leveraging learning methods to accommodate a larger number of control surfaces on an MAV. Using these methods allow improvements in the flight characteristics of MAVs as well as provide more robust control strategies where recovering from potential failures is critical. A significant improvement is obtained with the use of multiagent techniques applied to the MAV control problem [6, 88, 90] and results are presented in Chapter 6. The system consists of independent agents (control surface actuators) that learn to maximize a reward that is specific to each agent but that also benefit the overall system. Another important goal is to provide a controller that increases the robustness of the MAV to wind gust and various perturbations. This is achieved in Chapter 5 with a dynamic simulation that provides important changes in the environment so that different control strategies can be established to maintain control of the vehicle in situations where the learning based controller outperforms the PID based controller when high instabilities are encountered.

Chapter 5 – A Learning Based Approach to Micro Aerial Vehicle Control

In this chapter we use the GENMAV platform described in Section 3.1, the neuro-controller described in Section 3.2, and the flight simulator JSBSim decribed in Section 3.3 to improve the control response when compairing to a PID controller for altitude and heading hold when conditions are optimal as well as when wind gusts and perturbation are present.

5.1 Contribution of this Chapter

In this chapter, we show that neuro-evolutionary techniques can be used to ease the implemention of a controller on different MAV platforms by adjusting itself to the platform characteristics. We also show that the neuro-controller can increase the robustness of the platform to wind gusts and turbulence by adapting to unknown environments. Section 5.3 shows the experimental results where the neuro-controller's altitude and heading control are compared with a PID controller [56] as well as the improved robustness of the neuro-controller that can maintain a better tracking of a target altitude when wind gust's magnitude increases compared to a PID controller. Better heading tracking was also achieved when the level of turbulence is increased. Section 5.4 discusses the relevance of the results

and highlights directions for Chapter 6.

5.2 Objective Functions

An important part of using neuro-controllers consists of designing an objective function that allows the neuro-controller 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.

The objective functions can be fairly simple as is the case for G_Z, G_{Φ_1}, and G_Ψ in Equations 5.1, 5.2, and 5.4 where simple information about the error is used, to more complex for G_{Φ_2} in Equation 5.3 depending on the response that we want to achieve. The more complicated objective functions were used to improve the system response when wind gusts and perturbations were present.

Three different controllers were created for altitude, roll, and heading control. Each controller uses his own objective functions that are specifically designed for that particular controller. Their respective objective 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.

Objective Function for Altitude Control: G_Z

G_Z was designed to minimize the error between the desired altitude and the actual altitude. $(G_Z)^2$ was also tested and gave very similar results without significant improvements so G_Z was used in the experiments section.

$$G_Z = \frac{\alpha_z}{\sum_{t=0}^{T} [Z_d - Z_a]} \tag{5.1}$$

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

Objective 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. As before $(G_{\Phi_1})^2$ and $(G_{\Phi_2})^2$ were also tested and provided no significant improvements over G_{Φ_1} and G_{Φ_2}.

$$G_{\Phi_1} = \frac{\alpha_{\Phi_1}}{\sum_{t=0}^{T} [\Phi_d - \Phi_a]} \tag{5.2}$$

G_{Φ_2} was designed to improve the system response in presence of wind gusts and perturbations by including the roll derivative to the objective function. G_{Φ_2} provided better results than G_{Φ_1} for these conditions and was kept for this part of the experiments.

$$G_{\Phi_2} = \frac{\alpha_{\Phi_2}}{\sum_{t=0}^{T} \left[(\Phi_d - \Phi_a) + \frac{d\Phi}{dt} \right]} \tag{5.3}$$

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.

Objective Function for Heading Control: G_Ψ

G_Ψ was designed to minimize the error between the desired heading and the actual heading. $(G_\Psi)^2$ was tested but provided no significant improvements over G_Φ, therefore G_Φ was used for the experiments.

$$G_\Psi = \frac{\alpha_\Psi}{\sum_{t=0}^{T} [\Psi_d - \Psi_a]} \tag{5.4}$$

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

5.3 Experimental Results

A sweep of the neuro-controller parameters was conducted as a preliminary study in order to find values of the neuro-controller parameters that provided satisfactory results in terms of learning and optimization of the objective functions. The neuro-controllers 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 5.3. Altitude, roll, and heading neuro-controllers were trained using the objective functions from Section 5.2. Training time for the altitude and roll neuro-controllers was 5 seconds while training time for the heading neuro-controller was 12 seconds.

5.3.1 Altitude Control

Figures 5.1, 5.2, 5.3, 5.4, 5.5, and 5.6 show the results for several random desired altitudes with each time a comparison between neuro-controller results and PID results. The altitude is shown in Figure 5.1 and 5.2, the altitude rate in Figures 5.3 and 5.4, and the elevon positions in Figures 5.5 and 5.5.

The altitude control PID was fairly straightforward to implement but required some tuning to achieve the desired results. Figure 5.2 shows the MAV altitude with the dashed 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 5.1 shows MAV altitude when the neuro-controllers are in control using the same desired altitude than for the PID altitude control. Objective function G_Z from Equation 5.1 was used in the training of the neuro-controller but $(G_Z)^2$ was also tested and showed very similar results. The training consisted several thousand 5 seconds flights where the altitude neuro-controller was flying the plane. The objective function was used to grade the performance of the neuro-controllers and the best neuro-controller was kept for the testing part of the experiment. Looking at the altitude from Figure 5.1, the altitude neuro-controller performs very well. The target altitude is reached very quickly and efficiently with only a

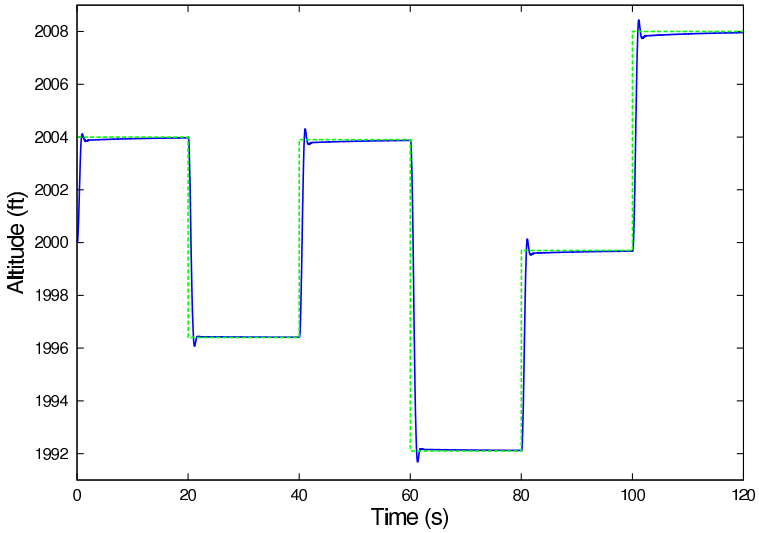

Figure 5.1: Desired and actual altitude: Neuro-controller. The control response is fast with the desired altitude value (dashed-line) followed very closely. The overshoot is minimal.

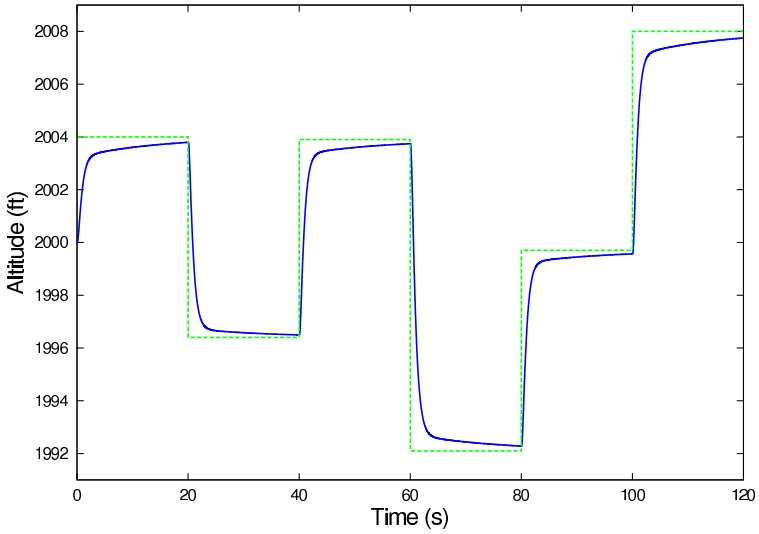

Figure 5.2: Desired and actual altitude: PID controller. The control response is a little slower than with the neuro-controller and the desired altitude value isn't tracked as closely. There is however no overshoot.

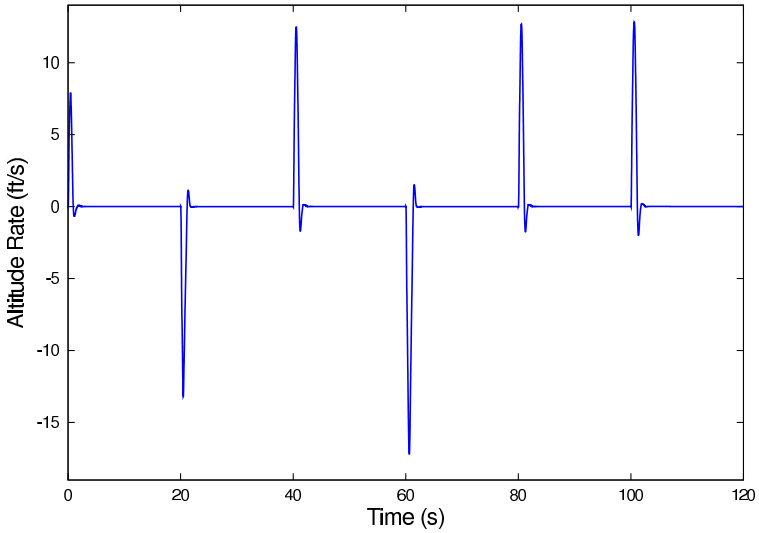

Figure 5.3: Altitude rate: Neuro-controller. The altitude rate is a little higher than with the PID controller due to the faster response. It is however still well within acceptable limits.

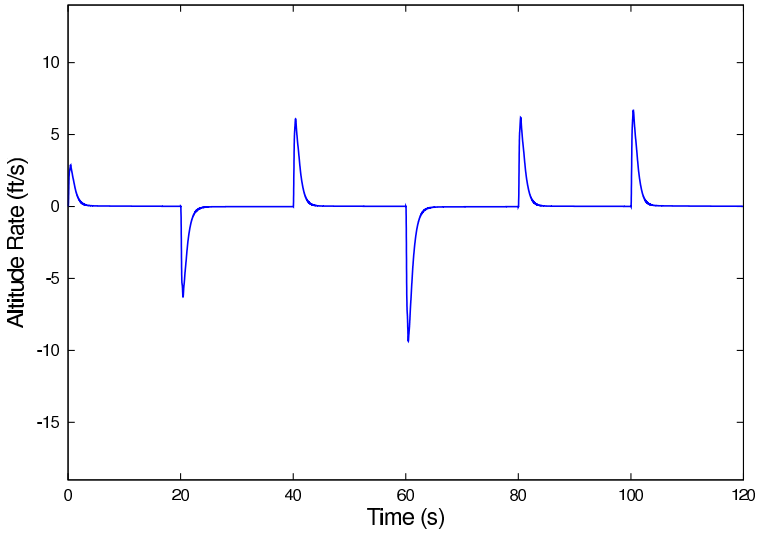

Figure 5.4: Altitude rate: PID controller. The altitude rate is minimal for this controller. The slower response of the PID controller compared to the neuro-controller minimizes altitude rate.

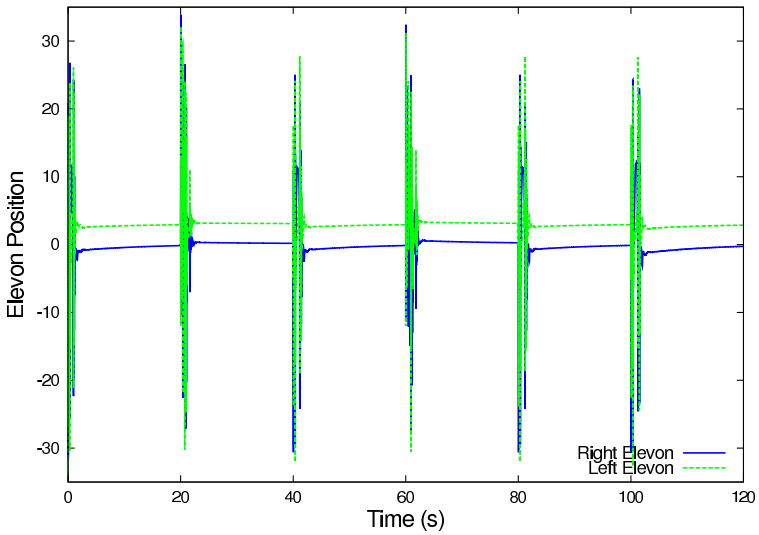

Figure 5.5: Elevon Position (altitude control): Neuro-controller. The faster response of the neuro-controller translates into slightly higher range of actuation of the elevons compared to the PID controller.

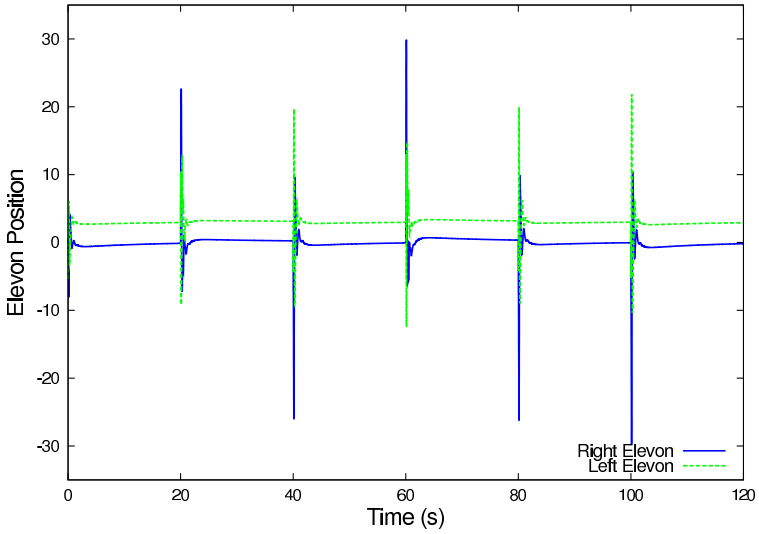

Figure 5.6: Elevon Position (altitude control): PID controller. The slower response of the PID controller is explained by the slightly shorter range of motion of the elevons compared to the neuro-controller.

very minimal overshoot. The neuro-controller's behavior is a little 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 neuro-controllers and PID controllers is shown in Figures 5.3 and 5.4. This provides additional information regarding the behavior of the system and is very helpfull 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 5.4) while the neuro-controller keeps it within 16ft/sec (Figure 5.3) which is still within the range of values providing a good behavior of the system. The difference between the two controllers is explained by the fact that the neuro-controller is a little more aggressive controller due to the form of the objective function.

The elevon positions corresponding to the altitude changes for both neuro-controller and PID controllers are shown in Figures 5.5 and 5.6. These graphs are also an important tool to assess the behavior of the system as they give information on the amplitude and frequency of the controller's response. Both controllers keep the elevon positions within ± 30 degrees while keeping oscillations to a minimum.

The altitude neuro-controller and PID controllers provide good authority over GENMAV and produce a good flight behavior. One interesting difference in the implementation of the controllers is the constant trim value. 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 experimenting and trial and error until the correct behavior was achieved. This necessitates additional tuning time before the MAV can fly correctly. In the neuro-controller case, however, no trim constant is needed. Once the neuro-controller is properly trained, no additional tuning or training is necessary to achieve good flight behavior. This is an advantage of the neuro-controller implementation where the neuro-controller adapts to the exacts 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.

5.3.2 Heading Control

As mentioned in section 3.4, 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. Figures 5.7 and 5.8 show the results for several random desired headings with each time a comparison between neuro-controller and PID results with the dashed line representing the desired heading and the continuous line representing the actual heading.

The heading control implementation was done in a very similar way as the

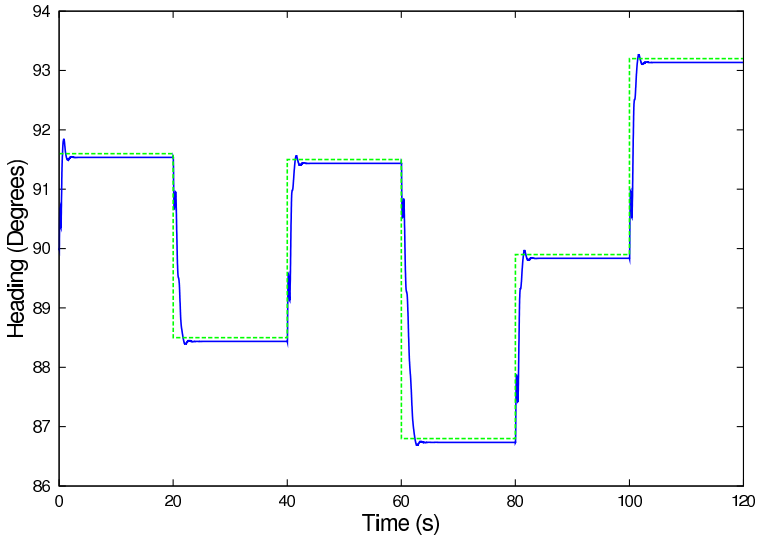

Figure 5.7: Desired and actual heading: Neuro-controller. The response is similar to what was obtained with altitude control: fast response with very close tracking of the desired value. The overshoot is very small and negligeable.

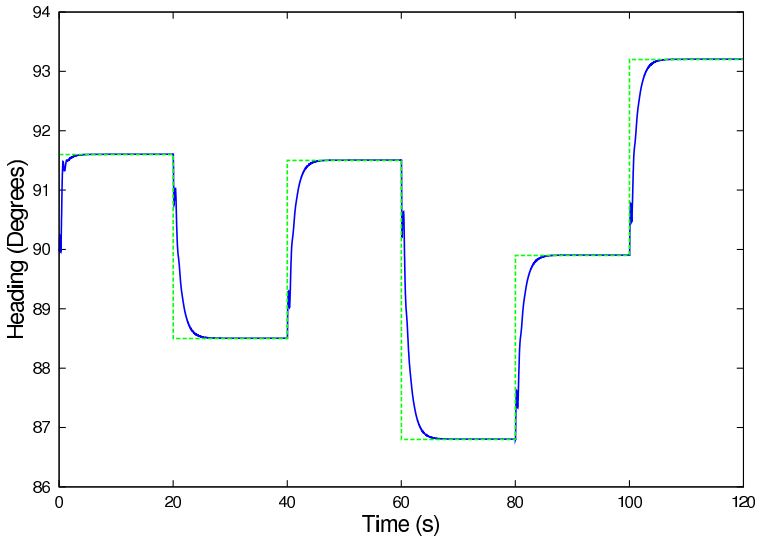

Figure 5.8: Desired and actual heading: PID controller. The response is a little slower than with the neuro-controller but the tracking of the desired value is very good. There is also no overshoot.

altitude control and likewise, this operation was fairly simple with an important amount of tuning necessary 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 neuro-controller was once again slightly more agressive and reached its a little quicker than its PID counterpart but with a very minimal amount overshoot while the PID controller reached its objective without any overshoot.

The heading control PID was tuned in an analogous fashion as the altitude control PID from Section 5.3.1 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.

Heading and rollneuro-controllers were trained using G_{Φ_1} and G_{Ψ} from Equations 5.2 and 5.4. $(G_{\Phi_1})^2$ and $(G_{\Psi})^2$ were also tested and produced very comparable results. Both heading and roll neuro-controllers adapted well to the MAV platform and provided good system behavior.

The elevon positions corresponding to the heading changes for both neuro-controllers and PID controllers are shown in Figures 5.9 and 5.10. Results here are very much alike except for the elevon angle range that the controllers use. Even though the heading/roll neuro-controllers are a little more aggressive in trying to achieve their objective, the neuro-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 ac-

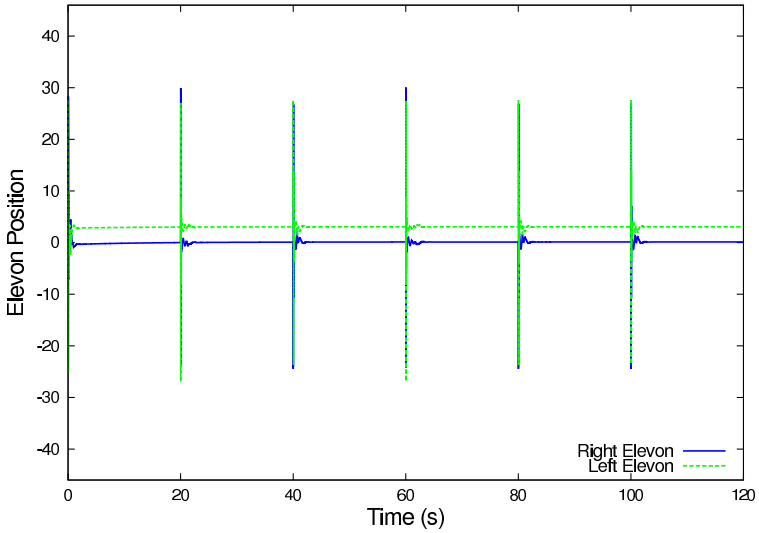

Figure 5.9: Elevon Positions (heading control): Neuro-controller. The elevon range of motion is minimized for the heading neuro-controller which produces a more efficient heading control while at the same time providing a quicker response.

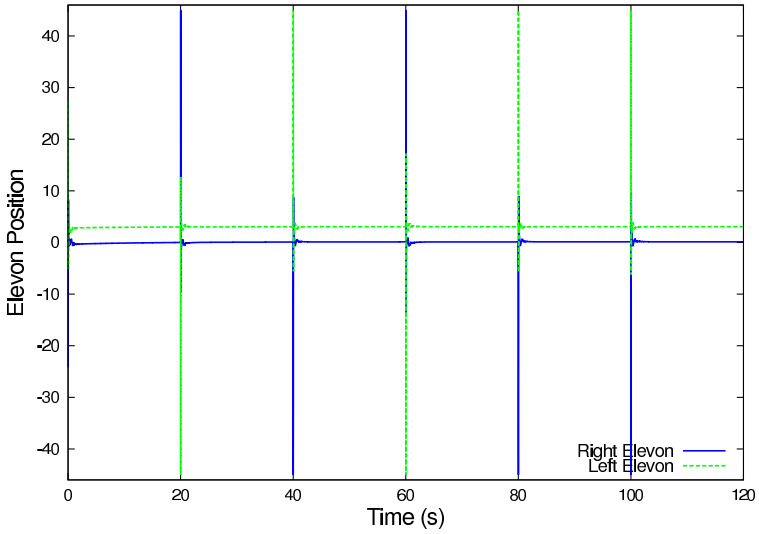

Figure 5.10: Elevon Positions (heading control): PID controller. The PID controller is less efficient than the neuro-controller for heading control: it is a little slower and uses a wider range of elevon actuation.

tual heading. The neuro-controllers being a little more 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.

As we can see in both Figures 5.9, and 5.10, 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 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 experimenting and trial and error while, the heading and roll neuro-controllers automatically adjust and do not need further tuning beyond the basic neuro-controller training. Once again, a significant advantage is gained when implementing controllers for similar platforms that have different specificities.

5.3.3 Wind Gusts and Turbulence

Wind gusts and turbulence are still experimental in JSBSim but provide nonetheless interesting changes in the enviroment. Wind gusts can be represented by a

force acting on the MAV with in this case a constant upward direction. Wind gusts were created at 4 seconds interval with the intensity of the wind gust increasing at every step. Turbulences are similar except that the direction of the force changes randomly. 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. Results for maintaining the altitude constant can be found in Figures 5.11 and 5.12 where the dashed line represents the desired altitude.

The neuro-controller and PID controller altitude curves have a very similar shape but the main difference is in the altitude value itself. The neuro-controllers are able to maintain the MAV altitude within 1 foot from the desired altitude while the PID controllers can only maintain it within about 6 feet.

To achieve these results, the neuro-controllers were trained in a similar way as in Sections 5.3.1 and 5.3.2 to start with but in this case objective function G_{Φ_2} from Equations 5.3 was used. These new objective functions provide more control over the roll response since they include the roll rate in addition to the error between desired and actual roll. Having a tighter control was important for the next part of the training as it incorporated changes in the environment. These environmental changes were small wind gusts at various time intervals. The neuro-controllers adapted to these changes and provided the resulting control behavior seen in Figures 5.11 and 5.12.

Similar results could not be obtained by tuning the PID controllers differently

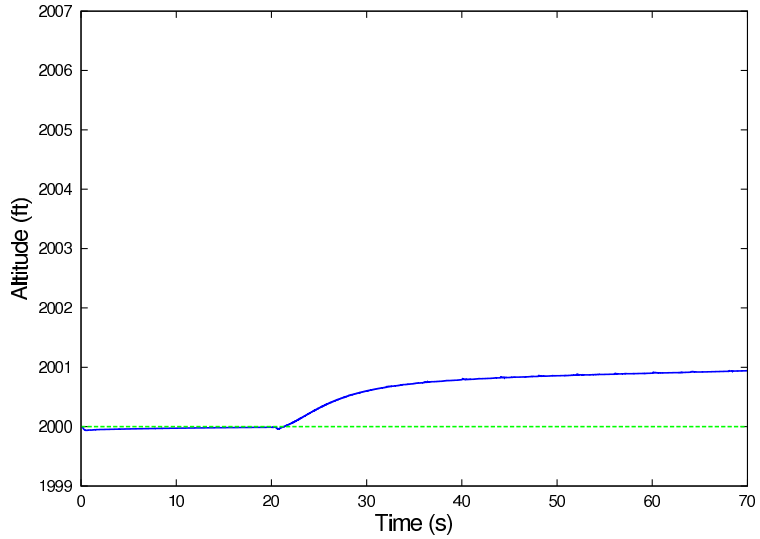

Figure 5.11: Desired and actual altitude (Wind Gusts): Neuro-controller. The neuro-controller achieves greater performance with wind gusts present by maintaining the altitude within a foot of the desired value versus 6 for the PID controller (Figure 5.12)

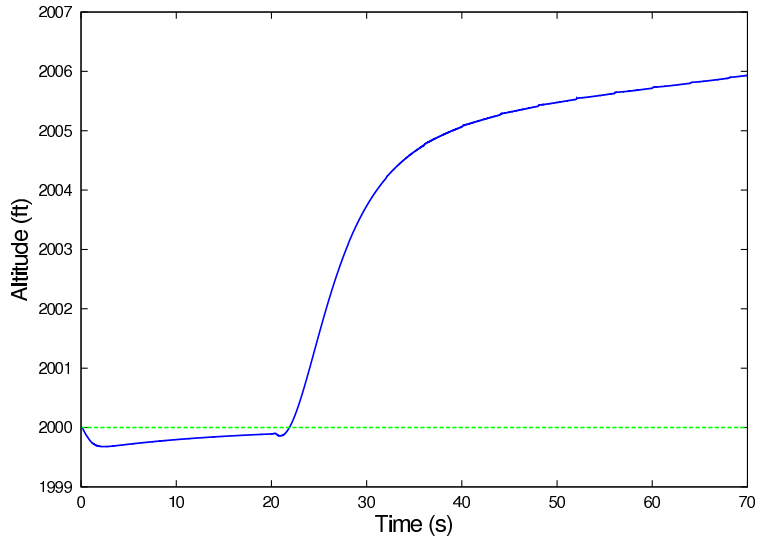

Figure 5.12: Desired and actual altitude (Wind Gusts): PID controller. The PID controller does not perform as well as the neuro-controller when wind gust are present: it stays within 6 feet of the desired value versus only 1 for the neuro-controller.

and in similar conditions as the what as been used for training the neuro-controllers. The PID controller is not a flexible controller that can adapt to new conditions, instead is specific to a particular platform and tuning conditions. The neuro-controllers required several thousand simulation runs to achieve these results but the changes to the controller were done automatically using the neuro-evolutionary algorithm. Running several thousand simulations and stopping in between each set to tweak the control parameters would be impossible within reasonable time constraints. Also, the training of the neuro-controllers was conducted within a particular set of environmental conditions. One can easily imagine, that these conditions are far from representing the wide variety of conditions that MAV encounter in the real world. Many other experimental and training conditions could be implemented to further improve the flexibility and robustness of the MAV platform. The neuro-controllers 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.

Although, the difference in maintaining the altitude constant is not huge between the PID and neuro-controllers, some application could require maintaining a fairly narrow altitude range in various conditions to achieve acceptable results. It is therefore important to keep as tight of a control as possible on the MAV so that good results can be obtained even if environmental conditions change or are unknown.

Experiments including turbulence were also conducted. The neuro-controllers were trained and tested in the same way as when performing the wind gust exper-

iments with increasing levels of turbulence instead of wind gusts. Results showing the MAV altitude and heading with increasing turbulence for both the neuro-controllers and PID controllers are presented in Figures 5.13, 5.14, 5.15, and 5.16.

Altitude remains fairly constant throughout the experiment and no significant difference is seen between the neuro-controllers and PID controllers. Figure 5.13 and 5.14 shows the altitude within half a foot of its desired value for both neuro-controllers and PID controllers. The small altitude variations are not relevant and altitude can be assumed near constant throughout the experiment for both controllers.

A difference is however visible for heading control where neuro-controllers were able to remain closer to the desired heading. Figure 5.15 shows that the heading stays within about a degree from the desired heading using the neuro-controllers while Figure 5.16 shows a difference of about three degrees. Once again, the difference between the two controllers is not very big but could make a difference depending on the the required accuracy of a particular application. This difference could also become much more important as the flight time of the vehicle increases while flying conditions are not optimal and present some level of turbulence as is the case in the majority of real world missions. Keeping a tighter control over the MAV heading could possibly slightly increase the range if deviations from the desired flight plan are kept to a minimum.

The adaptation capability of neuro-controllers can provide improvements on a variety of parameters such as flight time as well as increase the robustness of the platform in known or unknown environments. The training process in fairly

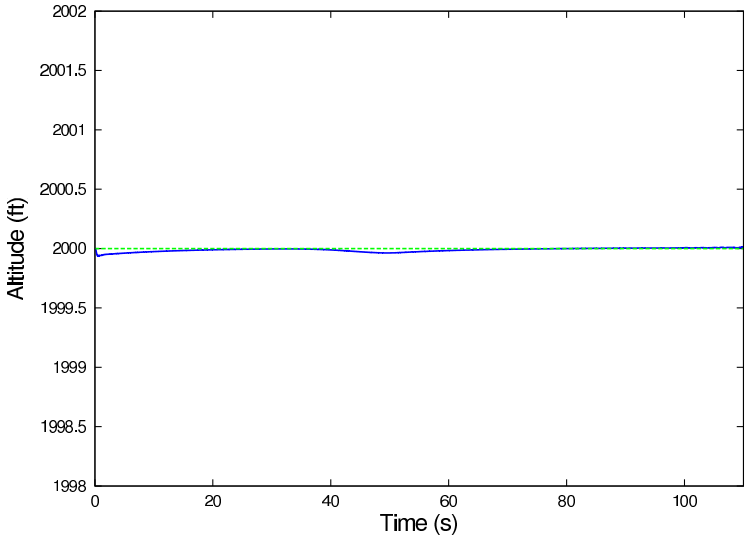

Figure 5.13: Desired and actual altitude (Turbulences): Neuro-controller. The altitude control is not affected by turbulence. The altitude remains very close to the desired value for the duration of the experiment.

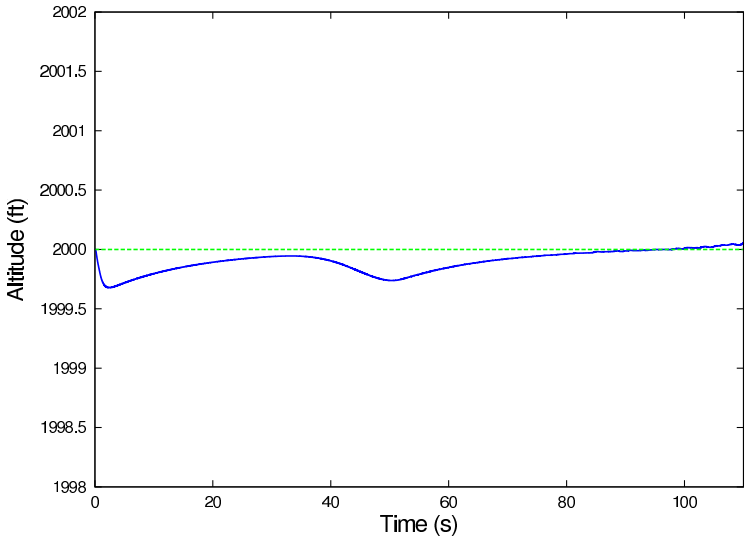

Figure 5.14: Desired and actual altitude (Turbulences): PID controller. The altitude control is not affected by turbulence. The small altitude variations are more important than for the neuro-controller but they are still negligeable.

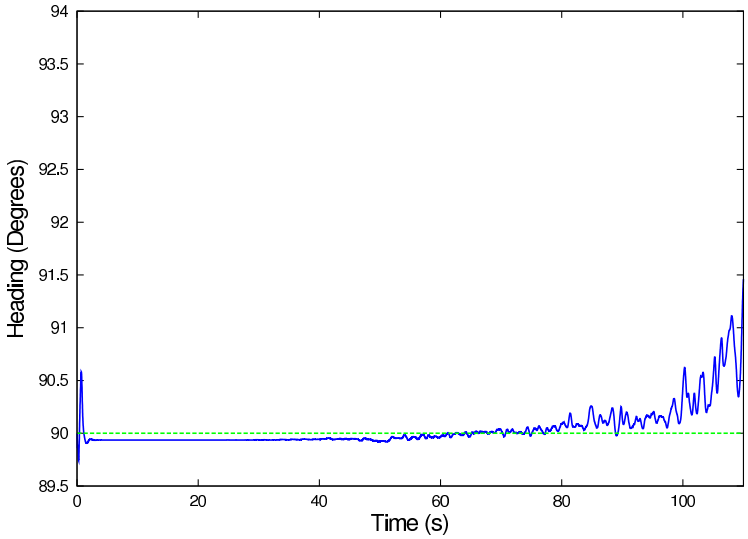

Figure 5.15: Desired and actual heading (Turbulence): Neuro-controller. The neuro-controller performs better than the PID when turbulence is present: the heading is maintained within a degree of the desired value versus 3 for the PID (Figure 5.16)

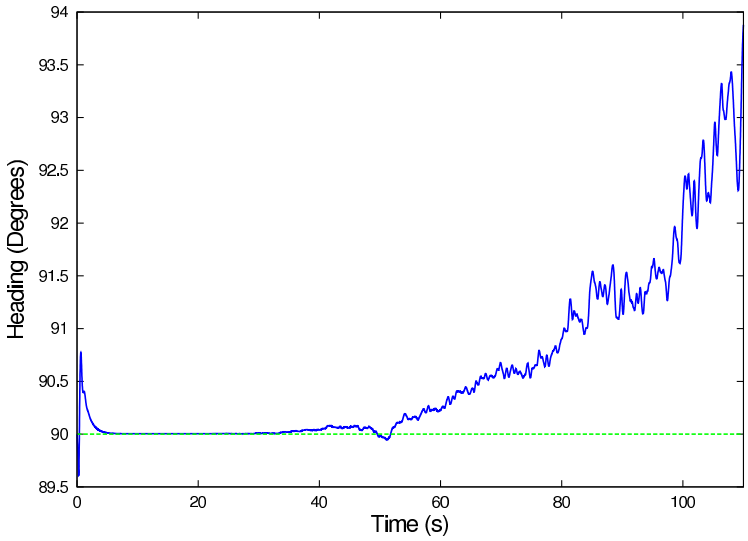

Figure 5.16: Desired and actual heading (Turbulence): PID controller. The PID controller does not perform as well as the neuro-controller in the presence of turbulence: the heading is kept within 3 degrees of the desired value versus 1 for the neuro-controller.

straightforward and acceptable results can be obtained quickly. The main difficulty remains the design of the objective function that can lead to a significant difference in the results. The great advantage is that the objective functions can be tailored to a specific objective that needs to be achieved such as minimizing heading variations or maximizing flight time and the resulting training should produce controllers that are optimal with respect to that objective.

5.4 Discussion

Micro Air 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 chapter 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.

Sections 5.3.1 and 5.3.2 showed that implementing a controller on an MAV with neuro-controllers was possible and required less tuning that 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 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 5.3.3 presented a first set of experiments where the neuro-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 neuro-controllers that were able to stay closer to the desired heading. Neuro-controllers can effectively learn and adapt from the system and its environment and provide an optimal system's configuration that improves performances. 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.

The results presented in this chapter are a first step that shows the potential of leveraging learning based methods to improve MAV control implementation and MAV performance. This methodology is applied to MAVs with segmented control surfaces in Chapter 6.

Chapter 6 focuses on leveraging learning based and multiagent based methods to control a modified version of GENMAV that includes segmented control surfaces. Experiments are conducted using GENMAV with six aileron segments on each wing for a total of twelve aileron segments. A central learning based controller

is then coupled with the modified GENMAV to start the learning process in a similar way as what was conducted in this chapter. Multiagent techniques are also implemented and tested and further improve the controller by increasing its robustness to actuator failure.

Chapter 6 – A Multiagent Based Approach to Micro Aerial Vehicle Control

In this chapter with use the GENMAV platform described in Section 3.1, the neuro-controller described in Section 3.2, and the flight simulator JSBSim decribed in Section 3.3 to control an MAV with segmented ailerons and improve the robustness of the controller when failures occur in the system.

6.1 Contribution of this Chapter

In this chapter, we show that neuro-evolutionary and multiagent techniques can be used to control multiple surfaces to improve MAV performance in terms of robustness and MAV response to control inputs by designing appropriate objective functions (e.g. altitude error value). Section 6.3 shows the experimental results, and Section 6.4 discusses the relevance of the results and highlights directions for future work.

6.2 Objective Functions

An important part of using neuro-controllers consists of designing an objective function that allows the neuro-controller 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.

The three controllers were designed for altitude, roll, and heading control. Each controller uses his own objective functions that are specifically designed for that particular controller. 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 neuro-controller commands each individual aileron segment and receive its own specific reward. This reward is presented in Equation 6.4.

Objective Function for Altitude Control: G_Z

G_Z was designed to minimize the error between the desired altitude and the actual altitude. $(G_Z)^2$ was also tested and gave very similar results without significant improvements so G_Z was used in the experiments section.

$$G_Z = \frac{\alpha_Z}{\sum_{t=0}^{T}\left[\beta_Z|Z_d - Z_a| + \gamma_Z|\frac{dZ}{dt}| + \delta_Z|\omega_E|\right]} \tag{6.1}$$

Where α_Z is an arbitrary 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 ω_E is the elevator angle.

Objective Functions for Roll Control: G_Φ

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

the actual roll. As before $(G_\Phi)^2$ was also tested and provided no significant improvements over G_Φ.

$$G_\Phi = \frac{\alpha_\Phi}{\sum_{t=0}^{T}\left[\beta_\Phi|\Phi_d - \Phi_a| + \gamma_\Phi|\frac{d\Phi}{dt}| + \delta_\Phi|\omega_A|\right]} \tag{6.2}$$

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 ω_A is the aileron angle.

Objective Function for Heading Control: G_Ψ

G_Ψ was designed to minimize the error between the desired heading and the actual heading. $(G_\Psi)^2$ was tested but provided no significant improvements over G_Φ, therefore G_Φ was used for the experiments.

$$G_\Psi = \frac{\alpha_\Psi}{\sum_{t=0}^{T}\left[\beta_\Psi|\Psi_d - \Psi_a| + \gamma_\Psi|\frac{d\Psi}{dt}|\right]} \tag{6.3}$$

Where α_Ψ is an arbitrary 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

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

$$G_{\Phi_i} = \frac{\alpha_\Phi}{\sum_{t=0}^{T} \left[\beta_\Phi |\Phi_d - \Phi_a| + \gamma_\Phi |\frac{d\Phi}{t}| + \delta_\Phi |\omega_{A_i}| \right]} \tag{6.4}$$

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 ω_{A_i} is the aileron segment position.

6.3 Experimental Results

A sweep of the neuro-controller parameters was conducted as a preliminary study in order to find values of the neuro-controller parameters that provided satisfactory results in terms of learning and optimization of the objective functions. The neuro-controllers 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 6.3. Altitude, roll, and heading neuro-controllers were trained using the objective functions from Section 6.2. Training time for the altitude and roll neuro-controllers was 5 seconds while training time for the heading neuro-controller was 12 seconds.

6.3.1 Altitude Control

Figures 6.1 and 6.2 show the results for several random desired altitudes with each time a comparison between neuro-controller results and PID results. The altitude control PID was fairly straightforward to implement but required some tuning to achieve the desired results. Figure 6.2 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 6.1 shows MAV altitude when the neuro-controllers are in control with the same target altitudes as when the system is under PID control. Objective function G_Z from Equation 6.1 was used in the training of the neuro-controller. Looking at the altitude from Figure 6.1, the altitude neuro-controller performs very well. The target altitude is reached very quickly and efficiently with only a very minimal overshoot. The neuro-controller'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 neuro-controller and PID controller provide good authority over GENMAV and produce good flight behavior. One interesting difference in the

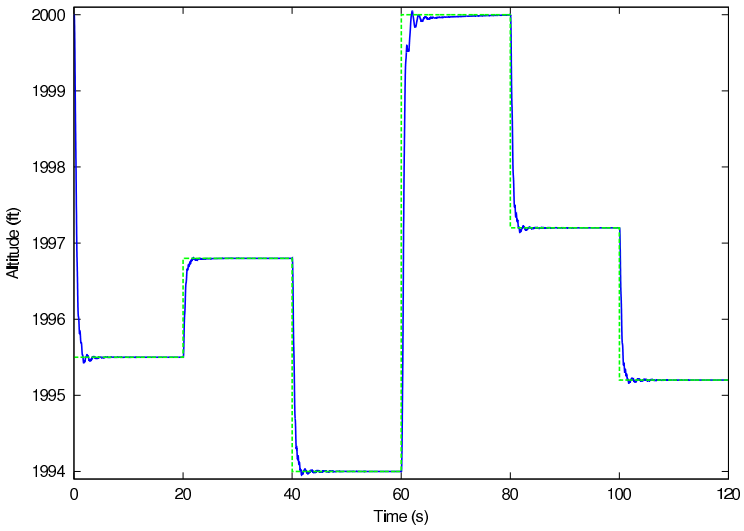

Figure 6.1: Desired and actual altitude: Neuro-controller. In this simple case of tracking a desired altitude (dashed line), the neuro-controlle is able to track the desired value very closely. The response is fast with very minimal overshoot.

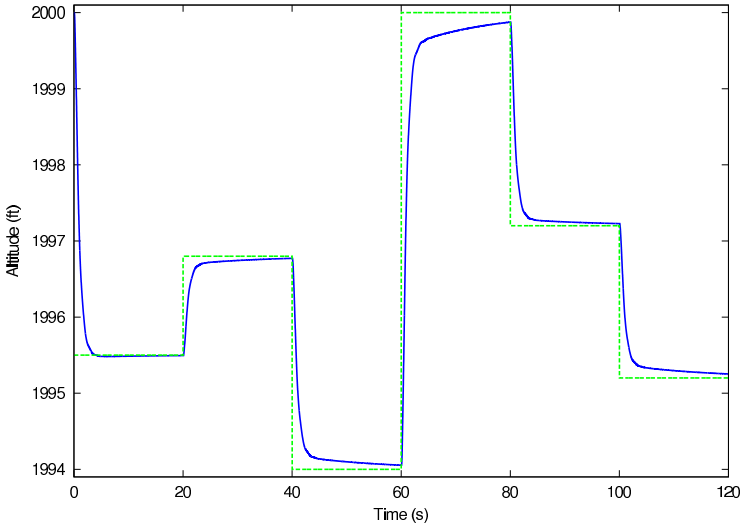

Figure 6.2: Desired and actual altitude: PID controller. After tuning, the PID controller was able to achieve acceptable performance with a response a little slower than the neuro-controller.

implementation of the controllers is the constant trim value. 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 experimenting and trial and error until the correct behavior was achieved. This necessitates additional tuning time before the MAV can fly correctly. In the neuro-controller case, however, no trim constant is needed. Once the neuro-controller is properly trained, no additional tuning or training is necessary to achieve good flight behavior. This is an advantage of the neuro-controller implementation where the neuro-controller 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.

6.3.2 Heading Control

As mentioned in section 3.4, 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. Figures 6.3 and 6.3 show the results for several random desired headings with each time a comparison between neuro-controller and PID results with the dashed line representing the desired heading and the continuous line representing

the actual heading. Results using the single neuro-controller and the multiagent neuro-controllers coupled with the segmented aileron model are shown in Figures 6.5 and 6.6.

The heading control implementation was done in a very similar way as the altitude control. Once again, the process was fairly simple but required tuning for the PID controller. Controllers for the segmented and unsegmented aileron models were all able to track the desired heading while providing good system behavior (Figures 6.3, 6.4, 6.5, and 6.6). Only negligeable differences can be observed for the simple task of heading tracking when conditions are optimal. The heading control PID was tuned in an analogous fashion as the altitude control PID from Section 6.3.1 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 neuro-controller and PID controller on the non-segmented aileron model are very much alike except for the elevon angle range that the controllers use (Figures 6.7 and 6.8). The range of aileron angles remains within 6 degrees for both controllers and the small difference is somewhat minimal and does not impact the overall behavior of the system.

Similar results can be seen on Figures 6.9 and 6.10 for segmented aileron model. Each control surface moves independently in the case of the multiagent neuro-controllers while control surfaces move in a symetric fashion for the single neuro-controller.

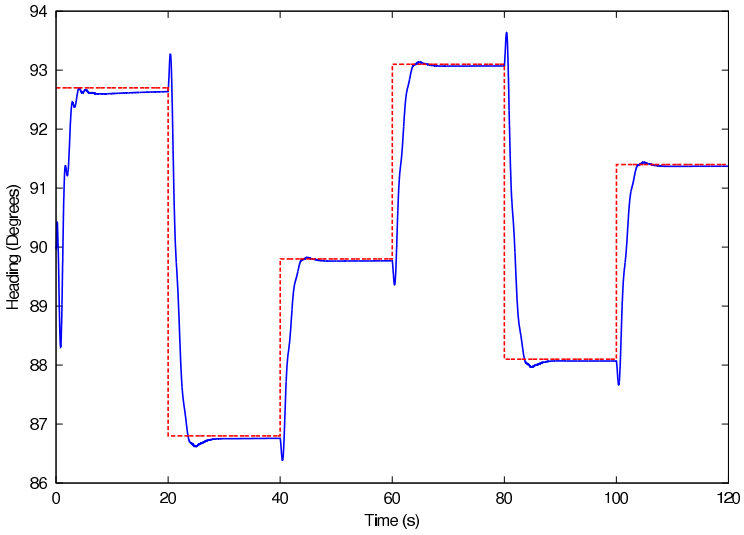

Figure 6.3: Desired and actual heading: Neuro-controller. In this other simple case of tracking a desired heading (dashed line), the neuro-controller is able to track the desired value closely. The response is fast with very minimal overshoot.

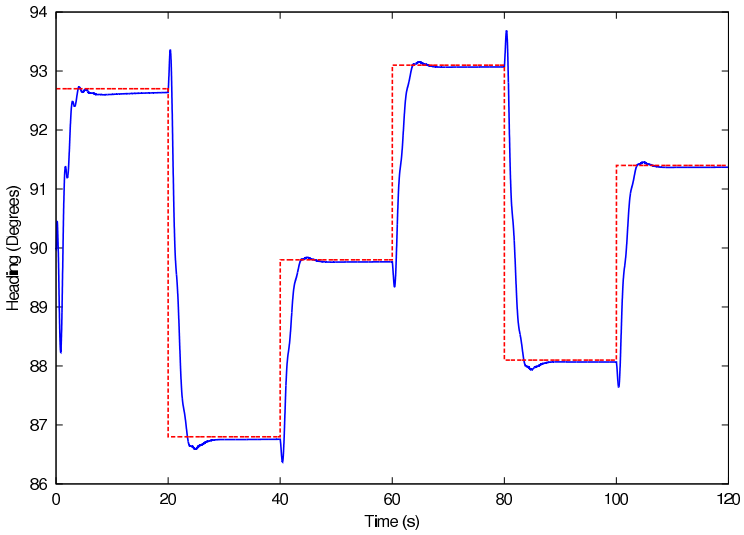

Figure 6.4: Desired and actual heading: PID controller. The response of the PID controller for this simple task is nearly identical to the neuro-controller's response. The response is fast with very minimal overshoot.

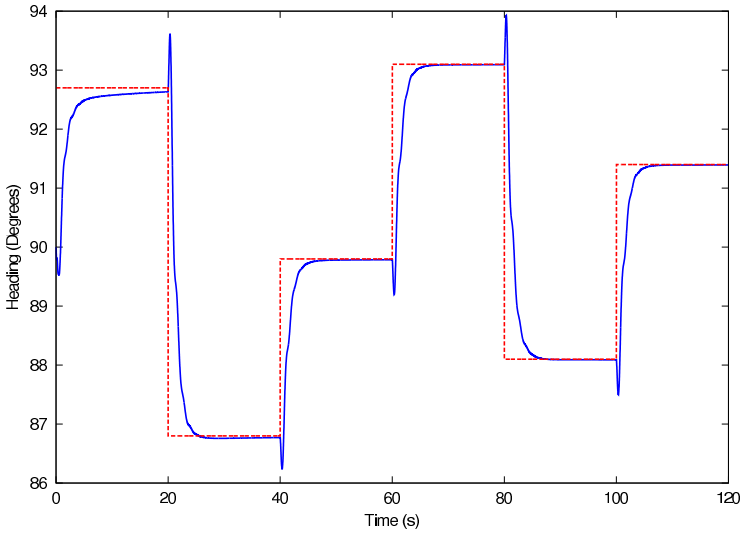

Figure 6.5: Desired and actual heading: Multiagent neuro-controllers. The response of the multiagent neuro-controllers is the best. The response is fast and smooth and the tracking of the desired value is near perfect. There is also no overshoot.

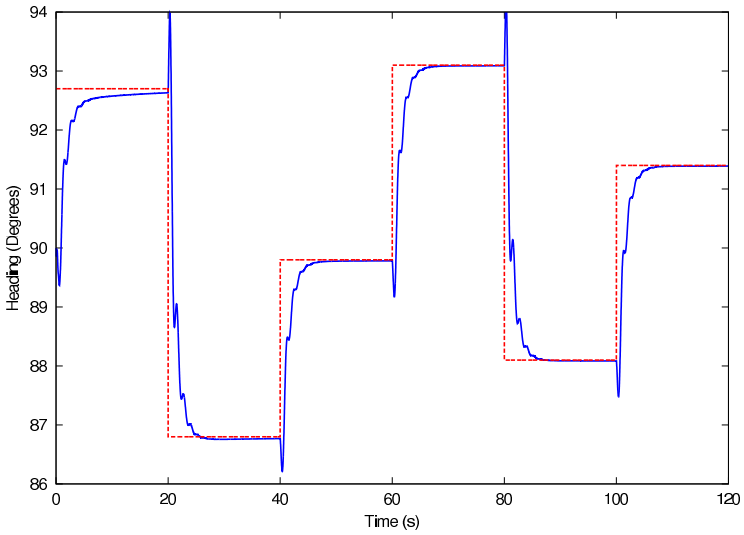

Figure 6.6: Desired and actual heading: Neuro-controller with segmented ailerons. This reponse is the 2^{nd} best. It is fast with a near perfect tracking of the desired value but it is not quite as smooth as the multiagent neuro-controllers'.

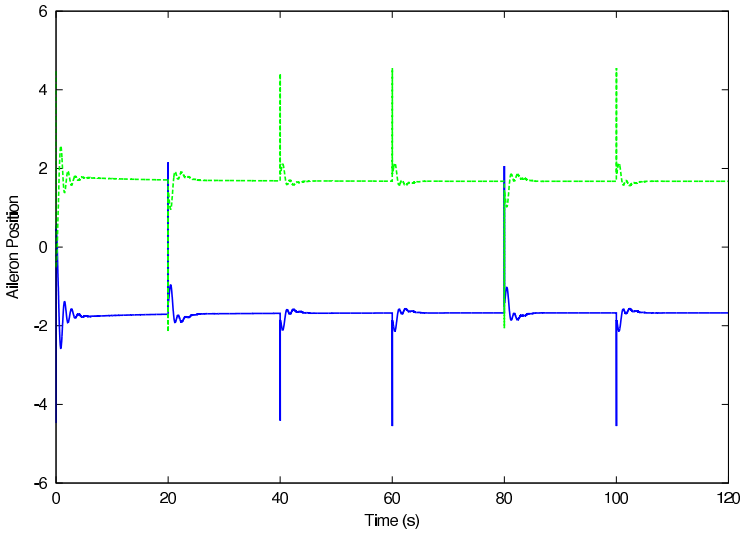

Figure 6.7: Aileron positions: Neuro-controller. These positions correspond to the heading tracking. Very similar results between the neuro-controller and the PID except for the range of actuation which is better for the neuro-controller (4 degrees instead of 6)

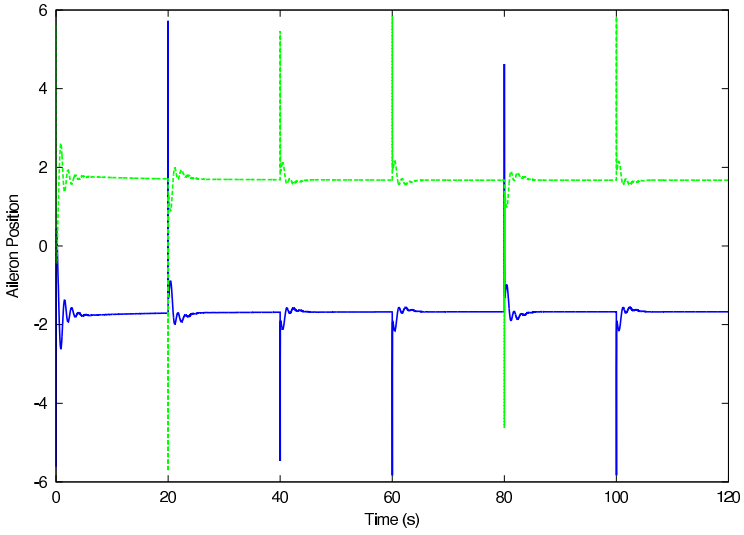

Figure 6.8: Aileron positions: PID controller (heading tracking). Very similar results between the PID and the neuro-controller except that the neuro-controller optimizes the range of actuation better with a range of 4 degrees instead of 6

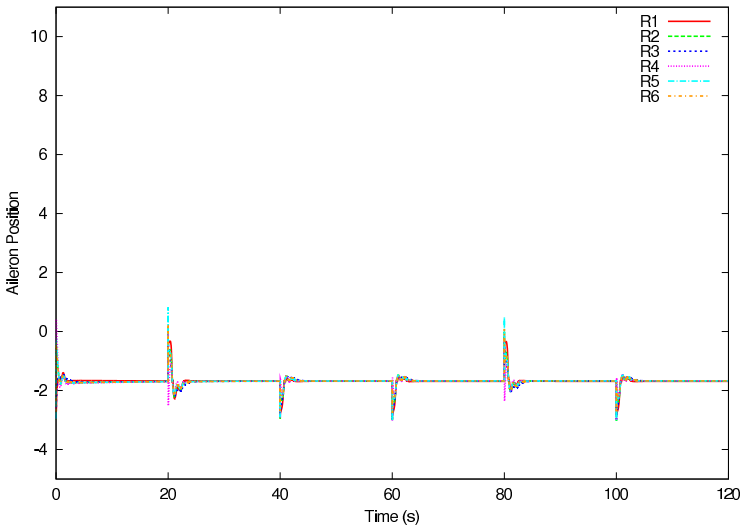

Figure 6.9: Aileron positions: Multiagent neuro-controllers. The range of motion is minimal for the multiagent neuro-controllers while producing the best control response which demonstrate the higher efficiency of this controller.

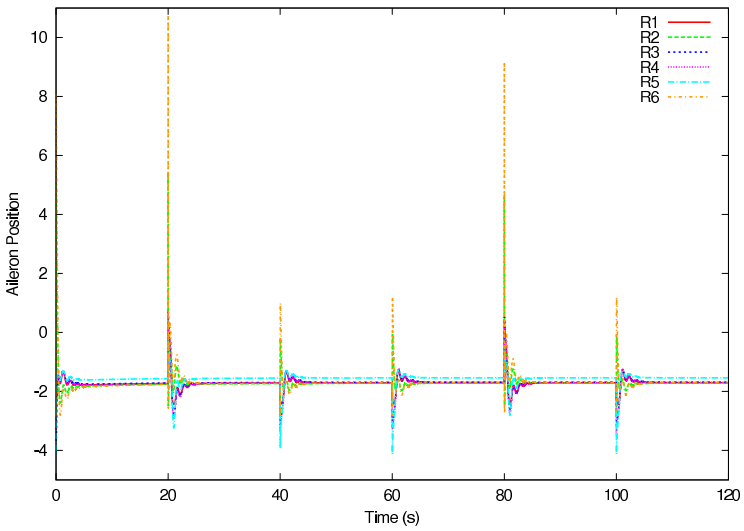

Figure 6.10: Aileron positions: Neuro-controller with segmented ailerons. The range of actuation is the highest for this controller which was not able to fully optimize its efficiency. However, the control response is still better than the PID controller.

Benefits of control surface segmentation were not apparent in the last 2 sections (Sections 6.3.1 and 6.3.2) due to the simplicity of the task that needed to be performed. These steps are however essential to provide the necessary background before moving on to more complicated tasks that require to train neuro-controllers beyond basic condiditons. These steps were completed with satisfactory results allowing for more advanced training in suboptimal conditions such as when a failure occurs in the system (Section 6.3.3).

6.3.3 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 therefore 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 neuro-controller as well as multiagent neuro-controllers.

Figures 6.11, 6.12, 6.13, and 6.14 show the altitude, heading, and aileron positions when failure 1 occurs for the multiagent neuro-controller. Failure 1 corresponds to actuator 4 on the left side of the MAV failing and stuck at around 5 degrees. The flexibility of the multiagent neuro-controllers allows them to adapt

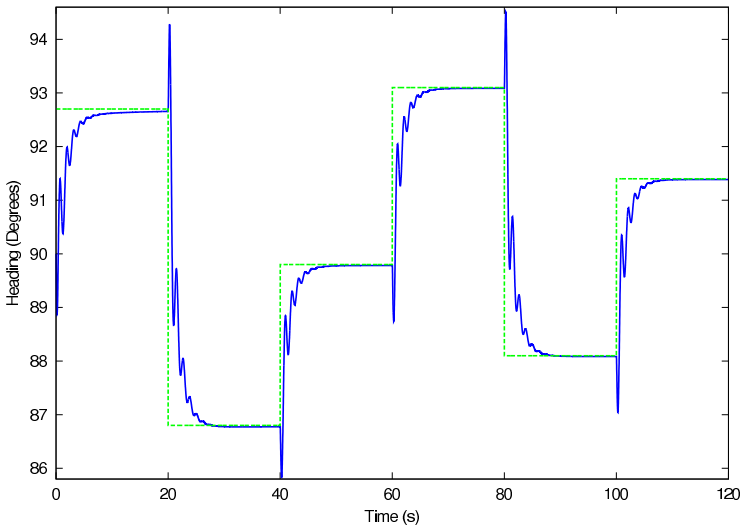

Figure 6.11: Failure 1, heading: Multiagent neuro-controllers. The left aileron actuator 4 failed at around 5 degrees. The multiagent neuro-controllers still maintain the desired heading very closely which is not the case for the other controllers.

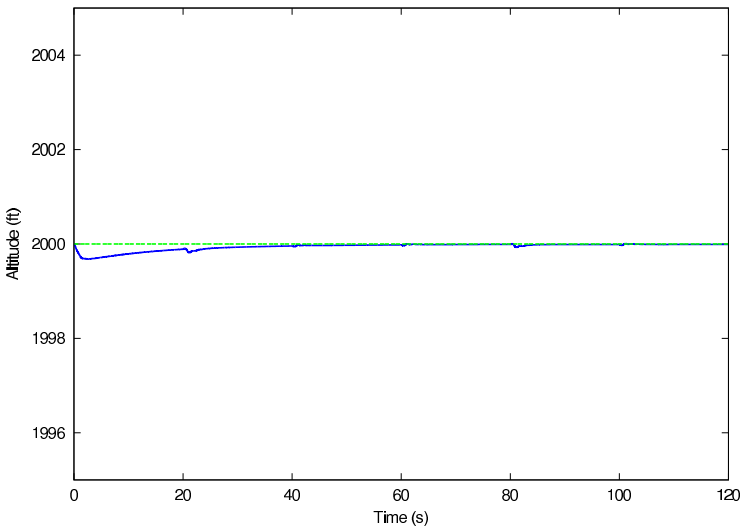

Figure 6.12: Failure 1, altitude: Multiagent neuro-controllers. The altitude is not affected by the failure since the ailerons and elevator are controlled by independent controllers.

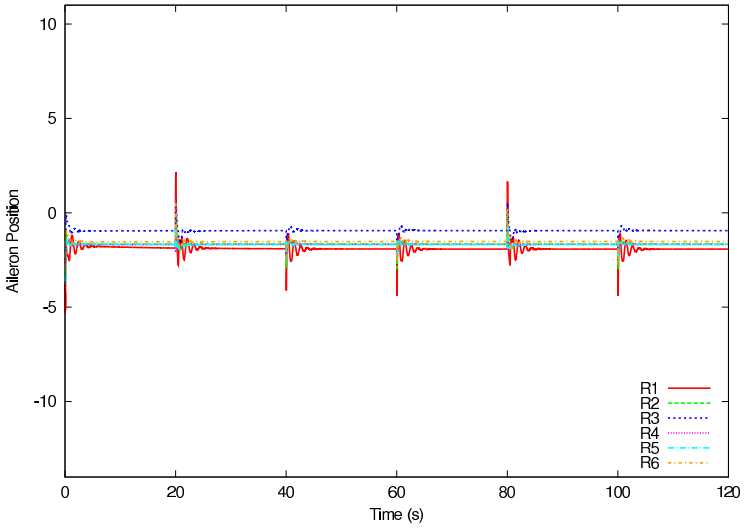

Figure 6.13: Failure 1, right ailerons positions: Multiagent neuro-controllers. Aileron segments on the right side are slightly adjusted to compensate for the failure that occured on left actuator number 4.

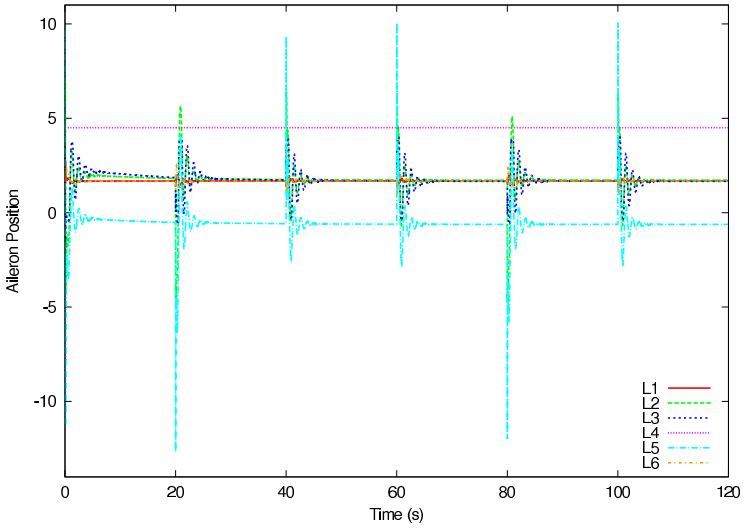

Figure 6.14: Failure 1, left ailerons positions: Multiagent neuro-controllers. The left aileron actuator 4 failed at around 5 degrees. The other aileron segments are adjusted to compensate for the failure.

and reconfiure themselves so that the control objective is achieved. In this case the multiagent neuro-controllers was still able to track the desired heading that was generated randomly every 20 seconds (figure 6.11). 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 6.12).

Figures 6.13 and 6.14 show the left and right aileron positions during failure 1. Aileron segment L4 remains at the failed position of about 5 degrees while the other aileron segments on each side of the MAV shifted from their usual position in order to compensate for the new dynamics of the system which allows a good behavior of the system in the event of an actuator failure.

The control response is not as smooth as before when 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 6.15 and 6.16 shows the desired and actual heading for the single neuro-controller paired with the segmented aileron model and the PID controller. A benefit of the control surface segmentation is clearly visible since the heading error remains within half a degree for the neuro-controller and goes above 2 degrees for the PID controller.

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 neuro-controllers combined with the aileron segmentation becomes even more apparent.

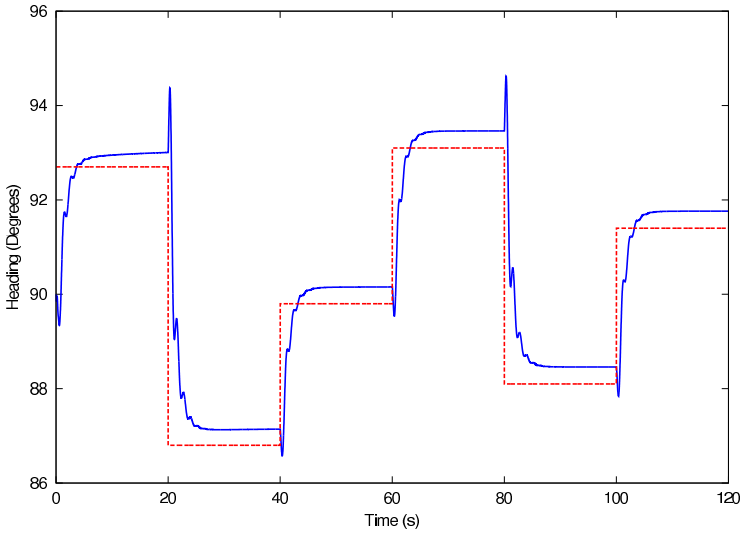

Figure 6.15: Failure 1, heading: Neuro-controller with segmented ailerons. Aileron segmentation benefits: The neuro-controller is able to remain within 1/2 degree of the desired heading versus over 2 degrees for the non-segmented model (Figure 6.16).

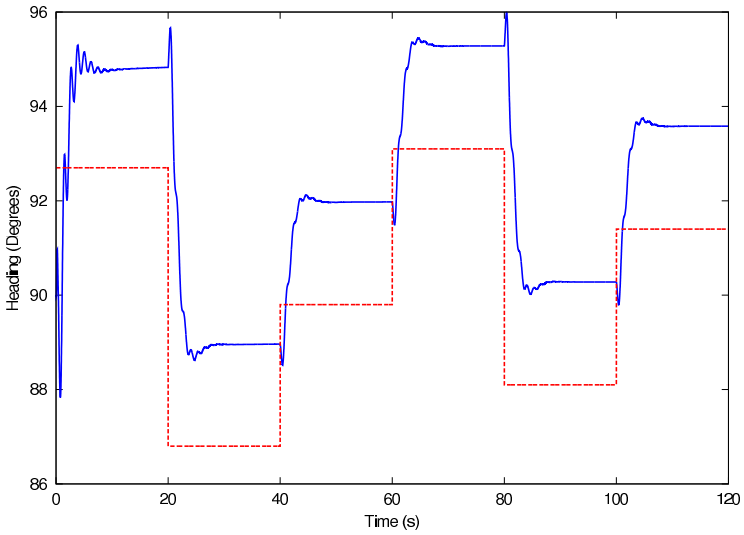

Figure 6.16: Failure 1, heading: PID controller. The PID/non-segmented model produces the highest heading error (over 2 degrees) compaired to the other controllers coupled to the segmented aileron version of GENMAV when failure 1 occurs.

Figures 6.17 and 6.20 show the desired and actual heading of the MAV as well as the corresponding aileron positions. As discussed previously, the control response is not as smooth when a failure is present but once again, the target heading value is achieved while the system demonstrate a good behavior.

The heading error is however much more prononced this time for the single neuro-controller and PID controller. The error is over a degree for the neuro-controller while it is now close to 5 degrees for the PID controller. In this case the multiagent neuro-controllers perform up to 5 times better than the PID controller. Figures 6.18 and 6.19 show the control response for both controllers.

Finally, a comparison between the PID and multiagent neuro-controllers was done with a wide range of failure positions. Results are presented in Figure 6.21 where the heading error was ploted as a function of the failure angle for both, the PID and multiagent neuro-controllers.

Unless the failed actuator angle remains around 2 degrees which 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 neuro-controllers perform 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 neuro-controllers perform up to

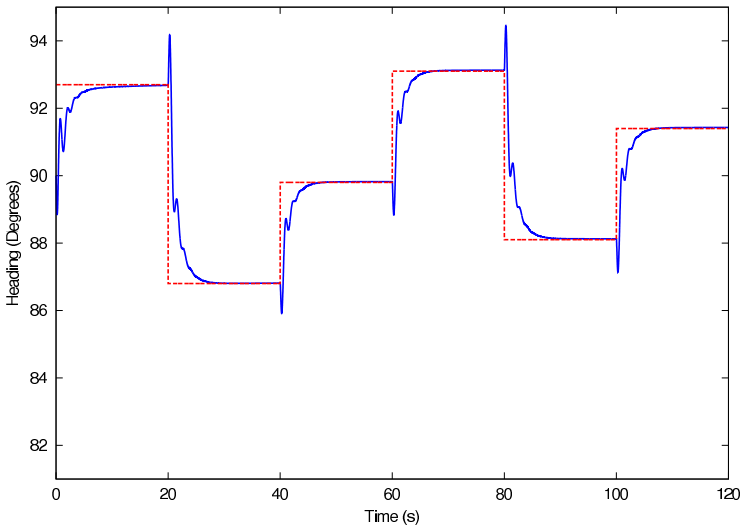

Figure 6.17: Failure 2, heading: Multiagent neuro-controllers. The left aileron actuator 4 failed at around -5 degrees. As was the case for failure 1, the multiagent neuro-controllers are still able to maintain the desired heading unlike the other controllers.

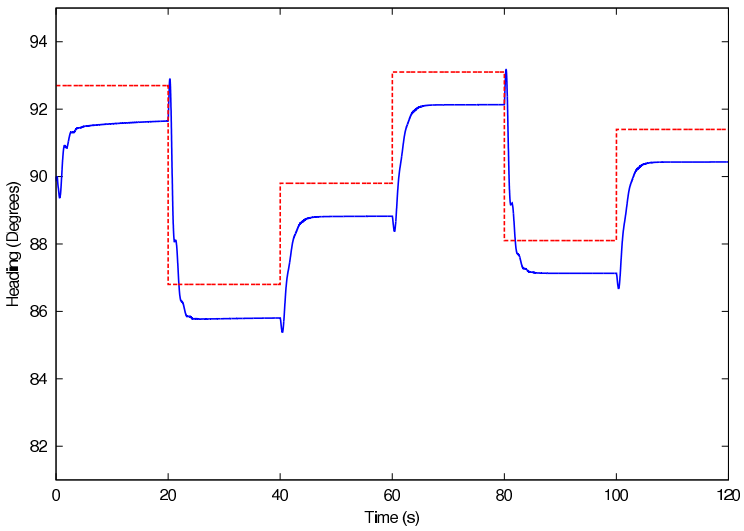

Figure 6.18: Failure 2, heading: Neuro-controller with segmented ailerons. Greater heading error than for failure 1 (over 1 degree) but still much better performance than the PID/non-segmented model (almost 5 degrees: Figure 6.19).

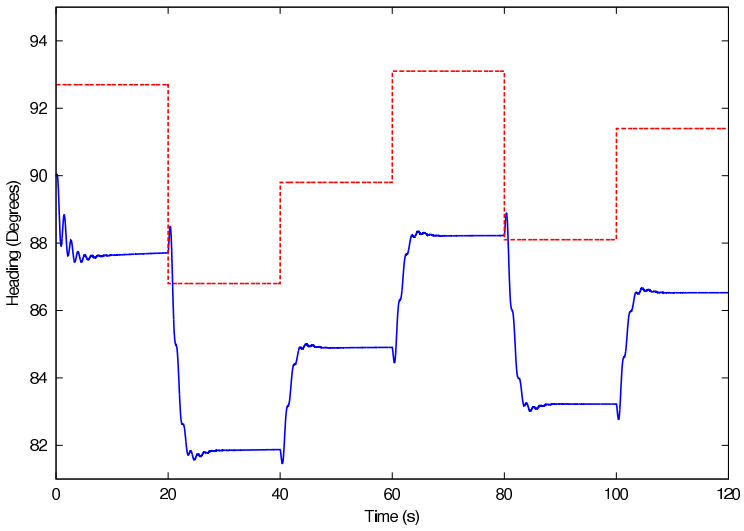

Figure 6.19: Failure 2 heading: PID controller. Once again, the PID/non-segmented model produces the highest heading error (almost 5 degrees) compaired to the other controllers coupled to the segmented aileron version of GENMAV when failure 2 occurs.

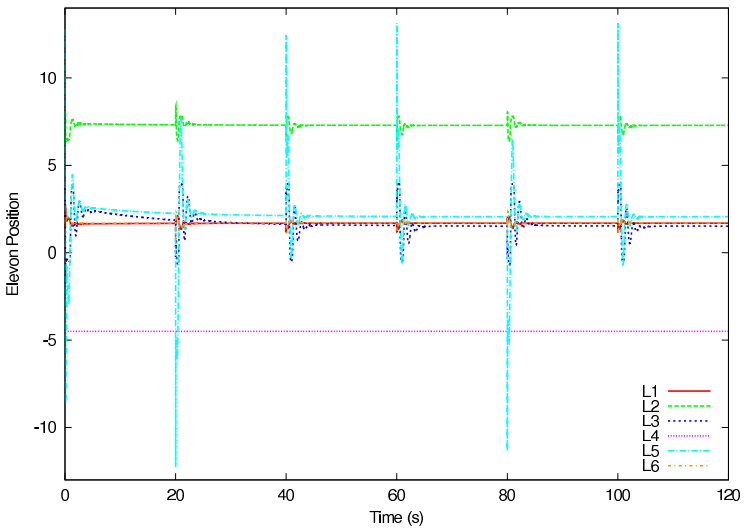

Figure 6.20: Failure 2 left ailerons position: Multiagent neuro-controllers. Aileron segment L4 failed with an angle of around -5 degrees. The other segments positions are adjusted to compensate for the failure.

8 times better than the PID controller. Beyond this limit, the heading error is still kept within the resonable values of 0 to 4 degrees which is still a minimum of 4 times better than the PID controller.

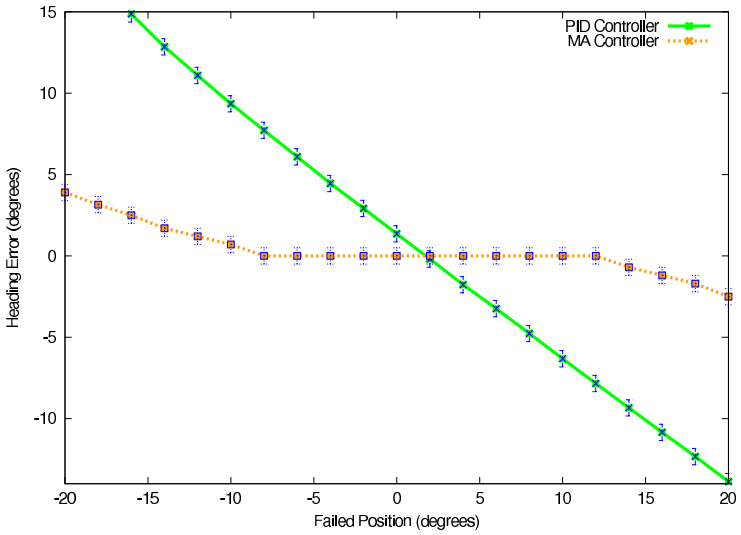

Figure 6.21: Heading error vs failure angle. The multiagent neuro-controllers reduce the heading error by up to a factor of 8 times better than the PID controller when a failure occurs while tracking a desired heading

These experiments demonstrate some of the benefits of multiagent neuro-controllers coupled with an MAV with segmented ailerons. The full potential of aileron segmentation can only be harnessed with a very flexible and adaptable controller such as the multiagent neuro-controllers used in these experiments. The multiagent neuro-controllers were able to successfully adapt to the new system dynamics created by the failure of an actuator and were able to reconfigure themselves so that minimal heading error was accomplished.

6.4 Discussion

Micro Air 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 chapter presents a novel approach to the MAV control problem and provides improvements 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 manoeuvers with up to two actuator failures.

Sections 6.3.1 and 6.3.2 showed that training a neuro-controller on an MAV with a neuro-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 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.

Aileron control surface segmentation was also implemented. Control through a single neuro-controller using symetric aileron segment position was shown to be possible with good behavior. Fully independent aileron control surfaces coupled to

a multiagent neuro-controllers were also attained with similar performance between the controllers for the most basic control tasks.

Section 6.3.3 showed significant improvements in the case of control surface actuator failure when using a multiagent neuro-controllers. The multiagent neuro-controllers were shown to performs up to 8 times better than the PID controller when tracking a desired heading.

The results presented in this chapter show the potential of leveraging multiagent based methods to improve MAV control implementation, performance, and robustness. The neuro-controllers presented in this chapter only use the aileron and elevator with the throttle set at a constant value for the duration of the experiments. An interesting study could add this functionality to the neuro-controllers so that different objectives could be achieved such as maintaining close to constant speed while performing heading or altitude change, or this could also improve the robustness to perturbations. 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 neuro-controllers are a great tool for finding optimal shapes, sizes, and locations for these openings as well as for controlling when and how much to open then based on specific sensor information. Future work in this area will also include improving the robustness of the neuro-controllers by training it in various environmental conditions so they can adapt to a broader spectrum of unknown real flight conditions. Experimental training and testing will also be done for specific environments so that slightly different controllers can be used depending on the area of operation. Furthermore, it will

also be possible to let the multiagent neuro-controllers 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.

Chapter 7 – Conclusion

Micro Air 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 dissertation presents a novel approach to the MAV control problem and provides improvements of the flight characteristics of such platform by introducing a larger number of control surfaces on the control sections. 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 4.3.1, 4.3.2, and 4.3.3 showed that controlling an MAV with a neurocontrollers was possible through segmented control surfaces. Using segmented control surfaces allows for smoother flight characteristics and flight maneuvers through minimization of actuator angles. Additionally, drag reduction of up to 5% can be seen for the larger values of the roll moment. If drag reduction is the objective, and if the drag is directly available the direct use of the drag in the objective function calculations provides the best results. However, if the drag is not available, minimizing the deflection between control surfaces still provides similar results and could be used instead. Results showed a drag improvement of up to 5% which was obtained by a gradual deflection of each actuator which lead to a smoother

control effort. Also, simulations conducted with 8 and 12 control surfaces showed no significant differences between the 2 configurations which indicates that for this particular problem and configuration, 8 control surfaces is already the optimal configuration and increasing the number of control surfaces will not improve the drag or efficiency of the MAV. The solutions provided by the neuro-controller matches the intuition that a gradual actuator deflection would provide close to optimal solutions. Results presented in this chapter show the potential of such configurations to improve flight characteristics of MAVs that are inherently difficult to control. Neuro-controllers can effectively learn from the system and provide an optimal system's configuration therefore allowing such modifications on an MAV platform. Furthermore, such a configuration would provide a higher level of robustness to the system that could recover and adapt from potential failures of some elements in the system which is critical for completing the assigned missions as seen in section 4.3.4. The Neuro-controller was able to learn and adapt to the new MAV configuration that had a failure in the system and was able to provide a new solution for the control strategy in order to stay in control of the vehicle.

Sections 5.3.1 and 5.3.2 showed that implementing a controller on an MAV with a neuro-controller was possible and required less tuning that 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 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 5.3.3 presented a first set of experiments where the neuro-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 an turbulence 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 neuro-controllers that were able to stay closer to the desired heading. Neuro-controllers 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 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.

Sections 6.3.1 and 6.3.2 showed similar results as Chapter 5 but aileron control surface segmentation was also implemented. Control through a single neuro-controller using symmetric aileron segment position was shown to be possible with good behavior. Fully independent aileron control surfaces coupled to a multiagent neuro-controllers were also attained with similar performance between the controllers for the most basic control tasks. Section 6.3.3 showed significant improvements in the case of control surface actuator failure when using multiagent neuro-controllers. The multiagent neuro-controllers were shown to performs up

to 8 times better than the PID controller when tracking a target heading. The multiagent neuro-controllers showed superior response due to their high degree flexibility and adaptability.

The three contributions of this work are as follows. We first show in Chapter 4 that neuro-evolutionary techniques can be used to control multiple surfaces to improve the flight characteristics of an MAV by designing appropriate objective functions (e.g., roll moment value). We then show in Chapter 5 that learning based methods can improved wind gust and turbulence robustness of MAVs when compared to a PID controller. Finally, we show in Chapter 6 that multiagent techniques coupled with segmented control surfaces further improve the robustness of the MAV platform and provide better recovery solutions in the case of actuator failures.

Continuation of this work could be done at different levels. First, controllers could be further improve to increase their performance and robustness through improved objective functions and with various training conditions. Secondly, multiagent techniques could be used for MAV navigation as a higher level of control for the platform to navigate to points of interest and avoid obstacles. Thirdly, multiagent techniques could be used for the coordination between multiple MAVs so that different tasks can be accomplished such as observation of points of interest or coordinated searches to for example find victims of natural disasters. Finally, implementation of all these techniques in hardware would provide very interesting feedback on how they perform in the real world.

Bibliography

[1] P Abbeel, A Coates, M Quigley, and A Y. Ng. An application of reinforcement learning to aerobatic helicopter flight. In *Neural Information Processing Systems (NIPS)*, 2006.

[2] P Abbeel, M Quigley, and A Y. Ng. Using inaccurate models in reinforcement learning. In *Proceedings of the Twenty-third International Conference on Machine Learning*, 2006.

[3] M. Abdulrahim. Flight dynamics and control of an aircraft with segmented control surfaces. In *42nd AIAA Aerospace Sciences Meeting and Exhibit*, 2004.

[4] M. Abdulrahim and J. Cocquyt. Development of mission capable flexible-wing micro air vehicle. In *53rd Southeastern Regional Student Conference*, 2002.

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

[6] A. Agogino and K. Tumer. Efficient evaluation functions for multi-rover systems. In *The Genetic and Evolutionary Computation Conference*, pages 1–12, Seattle, WA, June 2004.

[7] A. Agogino and K. Tumer. Reinforcement learning in large multi-agent systems. In *AAMAS-05 Workshop on Coordination of Large Scale Multi-Agent Systems*. Utrecht, Netherlands, July 2005.

[8] A. Agogino and K. Tumer. QUICR-learning for multi-agent coordination. In *Proceedings of the 21st National Conference on Artificial Intelligence*, Boston, MA, July 2006.

[9] M Ahmadi, M E Taylor, and P Stone. IFSA: Incremental feature-set augmentation for reinforcement learning tasks. In *The Sixth International Joint Conference on Autonomous Agents and Multiagent Systems*, May 2007.

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

[11] 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.

[12] M. Berhault, H. Huang, P. Keskinocak, S. Koenig, W. Elmaghraby, P. Griffin, and A. Kleywegt. Robot exploration with combinatorial auctions. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2003.

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

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

[15] 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.

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

[17] S Bouabdallah, A Noth, and R Siegwart. PID vs LQ control techniques applied to an indoor micro quadrotor. In *IEEE/RSJ International Conference on intelligent Robots and Systems IROS*, 2004.

[18] M Bowling. *Multiagent Learning in the Presence of Agents with Limitations.* PhD thesis, School of Computer Science Carnegie Mellon University, 2003.

[19] M Bowling. Convergence and No-Regret in Multiagent Learning. In *Neural Information Processing Systems (NIPS)*, 2004.

[20] M Bowling and M Veloso. Convergence of gradient dynamics with a variable learning rate. In *Eighteenth International Conference on Machine Learning (ICML)*, pages 27–34, June 2001.

[21] M Bowling and M Veloso. Rational and convergent learning in stochastic games. In *Seventeenth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1021–1026, 2001.

[22] M Bowling and M Veloso. Multiagent learning using a variable learning rate. *Artificial Intelligence*, 136:215–250, 2002.

[23] A Brzezinski, J Thrower, T C Garza, and B Young. Boeing blended wing body project. Technical report, 2003.

[24] G. Buskey, J. Roberts, P. Corke, M. Dunbabin, and G.F. Wyeth. The csiro autonomous helicopter project. In *International Symposium on Experimental Robotics (ISER)*, 2002.

[25] G. Buskey, J. Roberts, and G.F Wyeth. Online learning of autonomous helicopter control. In *Australasian Conference on Robotics and Automation*, 2002.

[26] G. Buskey, J. Roberts, and G.F. Wyeth. A helicopter named dolly behavioural cloning for autonomous helicopter control. In *Proceedings of the Australian Conference on Robotics and Automation (ACRA)*, 2003.

[27] G. Buskey, G.F Wyeth, and J. Roberts. Autonomous helicopter hover using an artificial neural network. In *International Conference on Robotics & Automation (ICRA)*, 2001.

[28] T S Dahl, M J Mataric, and G S Sukhatme. Adaptive spatio-temporal organization in groups of robots. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2002.

[29] 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.

[30] M Drela and H Youngren. Athena vortex lattice (AVL), 2004.

[31] K Dresner and P Stone. Multiagent traffic management: A reservation-based intersection control mechanism. In *The Third International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 530–537, July 2004.

[32] K Dresner and P Stone. Multiagent traffic management: An improved intersection control mechanism. In Frank Dignum, Virginia Dignum, Sven Koenig, Sarit Kraus, Munindar P. Singh, and Michael Wooldridge, editors, *The Fourth International Joint Conference on Autonomous Agents and Multiagent Systems*, New York, NY, July 2005. ACM Press.

[33] K Dresner and P Stone. Multiagent traffic management: Opportunities for multiagent learning. In K. Tuyls et al., editor, *LAMAS 2005*, volume 3898 of *Lecture Notes in Artificial Intelligence*, pages 129–138. Springer Verlag, Berlin, 2006.

[34] E W Frew, J Langelaan, and S Joo. Adaptive receding horizon control for vision-based navigation of small unmanned aircraft. In *American Control Conference*, 2006.

[35] 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.

[36] 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.

[37] A Geramifard, M Bowling, and R S Sutton. Incremental least-squares temporal difference learning. In *Twenty-First National Conference on Artificial Intelligence (AAAI)*, 2006.

[38] B Gerkey and M J Mataric. Sold!: Market methods for multi-robot control. Technical report, USC Institute for Robotics and Intelligent Systems IRIS, 2001.

[39] B P Gerkey and M J Mataric. Sold!: Auction methods for multi-robot coordination. *IEEE Transactions on Robotics and Automation, special issue on Advances in Multi-Robot Systems*, 18(5):758–786, 2002.

[40] 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.

[41] U Grasemann, D Stronger, and P Stone. A neural network-based approach to robot motion control. In Ubbo Visser, Fernando Ribeiro, Takeshi Ohashi, and Frank Dellaert, editors, *RoboCup-2007: Robot Soccer World Cup XI*. Springer Verlag, Berlin, 2008.

[42] M Greene. *http://spider.eng.auburn.edu/amstc/organization/Memsinertial.htm.*

[43] C Guestrin, M G Lagoudakis, and R Parr. Coordinated reinforcement learning. In *ICML '02: Proceedings of the Nineteenth International Conference on Machine Learning*, pages 227–234, San Francisco, CA, USA, 2002. Morgan Kaufmann Publishers Inc.

[44] 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.

[45] 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.

[46] 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.

[47] Z He, R V Iyer, and P R Chandler. Vision-based UAV flight control and obstacle avoidance. In *IEEE American Control Conference*, 2006.

[48] P.G. 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.

[49] D A. Jenkins, P G. Ifju, M Abdulrahim, and S Olipra. Assessement of controllability of micro air vehicles. Technical report, University of Florida, 2000.

[50] I Kajiwara and R T. Haftka. Simultaneous optimum design of shape and control system for micro air vehicles. In *AIAA Structures, Structural Dynamics and Material Conference*, pages 1612–1621, April 1999.

[51] Shivaram Kalyanakrishnan, Yaxin Liu, and Peter Stone. Half field offense in RoboCup soccer: A multiagent reinforcement learning case study. In Gerhard Lakemeyer, Elizabeth Sklar, Domenico Sorenti, and Tomoichi Takahashi, editors, *RoboCup-2006: Robot Soccer World Cup X*, pages 72–85. Springer Verlag, Berlin, 2007.

[52] S Koenig, J Melvin, P Keskinocak, C Tovey, and B Y Ozkaya. Multi-robot routing with rewards and disjoint time windows. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2007.

[53] S. Koenig, C. Tovey, M. Lagoudakis, V. Markakis, D. Kempe, P. Keskinocak, A. Kleywegt, A. Meyerson, and S. Jain. The power of sequential single-item auctions for agent coordination. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2006.

[54] S Koenig, C Tovey, X Zheng, and I Sungur. Sequential bundle-bid single-sale auction algorithms for decentralized control. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2007.

[55] 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.

[56] 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.

[57] M. Lagoudakis, P. Keskinocak, A. Kleywegt, and S. Koenig. Auctions with performance guarantees for multi-robot task allocation. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2004.

[58] M. Lagoudakis, V. Markakis, D. Kempe, P. Keskinocak, S. Koenig, A. Kleywegt, C. Tovey, A. Meyerson, and S. Jain. Auction-based multi-robot routing. In *Proceedings of the International Conference on Robotics: Science and Systems (ROBOTICS)*, 2005.

[59] P McKerrow. Modelling the Draganflyer four-rotor helicopter. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2004.

[60] David Moriarty and Risto Miikkulainen. Forming neural networks through efficient and adaptive coevolution. *Evolutionary Computation*, 5:373–399, 2002.

[61] A Y. Ng, A Coates, M Diel, V Ganapathi, J Schulte, B Tse, E Berger, and E Liang. Inverted autonomous helicopter flight via reinforcement learning. In *International Symposium on Experimental Robotics*, 2004.

[62] A Y. Ng and H J Kim. Stable adaptive control with online learning. In *Neural Information Processing Systems*, 2005.

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

[64] N Nigam and I Kroo. Persistent surveillance using multiple unmanned air vehicles. *IEEE*, 2008.

[65] 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.

[66] M W. Orr, S J. Rasmussen, E D. Karni, and W B. Blake. Framework for developing and evaluating mav control algorithms in a realistic urban setting. In *American Control Conference*, 2005.

[67] L Panait and S Luke. Cooperative multi-agent learning: The state of the art. *Autonomous Agents and Multi-Agent Systems*, 11(3):387–434, 2005.

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

[69] G Platanitis and S Shkarayev. Integration of an autopilot for a micro air vehicle. In *AIAA Infotech@Aerospace 2005 Conference and Exhibit*, 2005.

[70] J Polvichai, M Lewis, P Scerri, and K Sycara. Using a dynamic neural network to model team performance for coordination algorithm configuration and reconfiguration of large multi-agent teams. *Intelligent Engineering Systems Through Artificial Neural Networks, Smart Engineering System Design, ASME Press*, 16:565–574, 2006.

[71] B Pralio, G Guglieri, and F Quagliotti. Design of a flight simulation software tool for educational applications. In *AIAA Modeling and Simulation Technologies Conference and Exhibit*, 2004.

[72] V R. Puttige and S G. Anavatti. Real-time neural network based online identification technique for a UAV platform. In *CIMCA '06: Proceedings of the International Conference on Computational Inteligence for Modelling Control and Automation and International Conference on Intelligent Agents Web Technologies and International Commerce*, page 92, Washington, DC, USA, 2006. IEEE Computer Society.

[73] D M. Richwine and J H. Del Frate. Development of a low-aspect ratio fin for flight research experiments. Technical report, NASA, 1994.

[74] 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.

[75] H.-W. Schulz, M. Buschmann, L. Krger, S. Winkler, and P. Vrsmann. Vision-based autonomous landing for small UAVs first experimental results. In *AIAA*, 2005.

[76] A Simonis. Six-degree-of-freedom model of the air force research laboratory's generic micro air vehicle (GENMAV v2.2). Technical report, Oregon State University, 2009.

[77] B Sinopoli, M Micheli, G Donatoy, and T J Koo. Vision based navigation for an unmanned aerial vehicle. In *IEEE International Conference on Robotics an Automation (ICRA)*, 2001.

[78] 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.

[79] P Stone, R S. Sutton, and G Kuhlmann. Reinforcement learning for RoboCup-soccer keepaway. *Adaptive Behavior*, 13(3):165–188, 2005.

[80] P Stone and M Veloso. Multiagent systems: A survey from a machine learning perspective. *Autonomous Robots*, 8(3):345–383, July 2000.

[81] C. Tovey, M. Lagoudakis, S. Jain, and S. Koenig. The generation of bidding rules for auction-based robot coordination. In *Multi-Robot Systems: From Swarms to Intelligent Automata*. L. Parker, F. Schneider and A. Schultz, 2005.

[82] K. Tumer. Designing agent utilities for coordinated, scalable and robust multi-agent systems. In P. Scerri, R. Mailler, and R. Vincent, editors, *Challenges in the Coordination of Large Scale Multiagent Systems*, pages 173–188. Springer, 2005.

[83] 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.

[84] K. Tumer and A. Agogino. Efficient reward functions for adaptive multi-rover systems. In *AAMAS-05 Workshop on Learning and Adaptation in Multi-Agent Systems*. Utrecht, Netherlands, July 2005.

[85] K Tumer and A Agogino. Multiagent reward analysis for learning in noisy domains. In *Autonomous Agents and Multiagent Systems*, 2005.

[86] K Tumer and A Agogino. Distributed evaluation functions for fault tolerant multirover systems. In *Genetic and Evolutionary Computation Conference*, 2006.

[87] K Tumer and A Agogino. Distributed agent-based air traffic flow management. In *Autonomous Agents and Multiagent Systems*, 2007.

[88] 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.

[89] K. Tumer, A. Agogino, and D. Wolpert. Learning sequences of actions in collectives of autonomous agents. In *Proceedings of the First International Joint Conference on Autonomous Agents and Multi-Agent Systems*, pages 378–385, Bologna, Italy, July 2002.

[90] K. Tumer and D. Wolpert, editors. *Collectives and the Design of Complex Systems*. Springer, New York, 2004.

[91] K. Tumer and D. Wolpert. A survey of collectives. In K. Tumer and D. Wolpert, editors, *Collectives and the Design of Complex Systems*, pages 1–42. Springer, 2004.

[92] 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.

[93] Martin R. Waszak, Luther N. Jenkins, and Peter Ifju. Stability and control properties of an aeroelastic fixed wing Micro Aerial Vehicle. In *AIAA Atmospheric Flight Mechanics Conference and Exhibit*, 2001.

[94] 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.

[95] D. H. Wolpert and K. Tumer. Optimal payoff functions for members of collectives. *Advances in Complex Systems*, 4:265–279, 2001.

[96] D. H. Wolpert, K. Wheeler, and K. Tumer. General principles of learning-based multi-agent systems. In *Proceedings of the Third International Conference of Autonomous Agents*, pages 77–83, May 1999.

[97] J Young and A Ross Price. FPGA based UAV flight controller. In *11 Eleventh Australian International Aerospace Congress (AIAC)*, 2005.

[98] JC Zufferey and D Floreano. Toward 30-gram autonomous indoor aircraft: Vision-based obstacle avoidance and altitude control. In *IEEE International Conference on Robotics and Automation*, 2005.