AN ABSTRACT OF THE THESIS OF

Jack F. Shepherd III for the degree of Master of Science in Mechanical Engineering
presented on December 1, 2009.

Title: A Hierarchical Neuro-Evolutionary Approach to Small Quadrotor Control

Abstract approved: _____

Kagan Tumer

Quadrotors are unique among Micro Aerial Vehicles in providing excellent maneuverability (as opposed to winged flight), while maintaing a simple mechanical construction (as opposed to helicopters). This mechanical simplicity comes at a cost of increased controller complexity. Quadrotors are inherently unstable, in the sense that they are essentially unflyable by a human without stability assistance. Prior work has developed model-based controllers that successfully control quadrotors operating near hover conditions, but small quadrotor dynamics make such control more difficult.

In this thesis, we present a hierarchical neuro-controller for small (0.5 kg) quadrotor control. The first stage of control aims to stabilize the craft and outputs rotor speeds based on a requested attitude (pitch, roll, yaw, and vertical velocity). This controller is developed in four parts around each of the variables, initially training them to achieve results similar to a PID controller. The four parts are then combined and the controller is trained further

to increase robustness. The second stage of control aims to achieve a requested (x, y, z) position by providing the first stage with the appropriate attitude.

The simulation results show that stable quadrotor control is achieved through this control architecture. In addition, the results show that the hierarchical control approach recovers from discrete angle disturbances over an order of magnitude faster than a basic PID controller, and can even recover from a disturbance that knocks the craft upside down. Finally, using the average volume used to maintain a stable hover, we show that the hierarchical controller provides stable flight in the presence of 5 times more sensor noise and 8 times more actuator noise as compared to the PID controller.

A Hierarchical Neuro-Evolutionary Approach to Small Quadrotor Control

by

Jack F. Shepherd III

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Master of Science

Presented December 1, 2009
Commencement June 2010

Master of Science thesis of Jack F. Shepherd III presented on December 1, 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 thesis will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my thesis to any reader upon request.

Jack F. Shepherd III, Author

ACKNOWLEDGEMENTS

This work was partially supported by AFOSR grant FA9550-08-1-0187 and NSF grant IIS-0910358.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF FIGURES (Continued)

LIST OF TABLES

Chapter 1 – Introduction

The ability to safely and accurately gather information about an environment is critical to the rapid and safe deployment of personnel in response to many military or civilian needs. Successful incident response, from search and rescue to armed combat, often requires the need for situational awareness about an unknown environment. The need to acquire this information, is what drives the field of mobile robotics.

Traditionally, this information may have come from local experts. This first hand information is often incomplete especially in extenuating circumstances such as a natural disaster, when the terrain may have changed, or when the information needed is about the immediate location of trapped or injured persons. With the advent of satellite technology, we can acquire pictures of an area that are both recent, and can be overlaid with information like heat sensitive maps showing potential locations of people. Although these pictures present more accurate and immediate information than relying on local guides, they are limited by the overhead viewing angle and the difficulty of getting a satellite into place at just the right time. In the end, these alternate sources cannot compete with real-time, low-altitude images. This is why search and rescue missions use spotter planes and large numbers of people to scan territory. Similarly military missions employ surveillance aircraft and ground scouts.

Although immediate information from the field is often what is required, acquiring it may

involve putting humans at risk. As the field of mobile robots advances, we are learning to send them where humans cannot safely go. Since their first major field test at the site of the collapsed World Trade Center, ground based search and rescue robots have garnered a lot of interest [18, 19]. Ground based robots have the advantage of being able to roll inside buildings, or into small enclosed spaces. However, for many needs a slightly bigger aerial picture is important. In addition, sometimes access to a building can best be done through an open window, another situation more suited to an aerial vehicle.

The recent growth in interest of Micro Aerial Vehicle (MAV) platforms is a testament to their increasing strategic importance. Small, light, and versatile crafts will dominate the field of reconnaissance, be it military intelligence, or search and rescue [10]. MAVs biggest strengths and weaknesses lie in their size. Being small, MAVs are easier to transport, harder to detect, and cheaper to operate. However, it also makes them more unstable, harder to control, and requires greater miniaturization of payload sensors and controller hardware.

Traditional MAVs have been miniaturized airplanes due to their greater inherent stability [14]. However, advances in controller technology has allowed the development of rotorcraft MAVs [31]. With full size aircraft, rotorcraft offer a clear advantage for surveillance due to their ability to hover. At the slow speeds of MAVs, this advantage is not as pronounced. However, many missions do still benefit from this hovering ability, especially as it results in increased maneuverability. In cluttered environments, whether these be urban streets or forested mountains, the ability to move backwards or laterally remains a significant advantages. Another advantage that rotorcraft have over other MAV platforms, is the ability to perform vertical take-offs and landings. To deploy, space is only needed to pull the

craft out of a backpack and place it on the ground. Even in the micro size scale, the runway demands of airplane based MAVs can be tricky to meet in crowded environments.

The advantages of MAV rotorcraft are often lost when the disadvantages of certain designs are examined. Helicopters, for example, are the most common style of rotorcraft but suffer on the micro-scale due to their mechanical complexity. Helicopters generate lift through their single overhead propellor. An additional propellor is located in the tail to counteract the torque from the main propellor. Instead of simple mechanical linkages linked to control surfaces as on an airplane, motion other than lift is controlled by changing the pitch of the propellor blades. The need to vary blade pitch on a rotating structure requires complex mechanical linkages. When these fail, the forces on even a small helicopter are enough to slice through skin and bone, creating a very dangerous situation.

Quadrotor craft share the same advantages in maneuverability with other rotorcraft, but without the mechanical complexity of a helicopter. Instead of a single main rotor and a secondary tail rotor, a quadrotor has four equal sized rotors as shown in Figure 1.1. These rotors are not variable-pitch rotors as on a helicopter, instead, the craft is maneuvered by adjusting the relative speed between the individual rotors. This makes the quadrotor mechanically much simpler, which aids in scaling down to a micro-scale, but makes control much more complex. A quadrotor is inherently unstable, any motor unbalance requires constant controller input to keep the craft aloft. Quadrotors also provide for greater safety with smaller blades and the ability to enclose them within the airframe.

The essence of quadrotor control is simple. The speed difference between two pairs of

motors is adjusted to change the attitude of the craft. For example, increasing the speed of rotor 1 while at the same time decreasing the speed of rotor 3 by the same amount will cause the craft to pitch up. Adjusting in the opposite direction causes the opposite motion. Roll adjustments are similarly made using rotors 4 and 2. Finally, the difference in rotation directions between the pairs of motors can be used to cause the craft to rotate about the z-axis. Although control of each axis independently is simple, the change of speed in any one rotor will affect multiple axis, and this coupling introduces instability and complexity to the platform requiring advanced control methods.

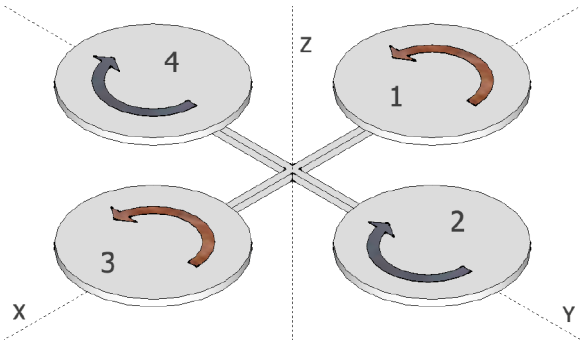

Figure 1.1: Quadrotor layout: *Roll* (rotation around the x axis) controlled by adjusting speeds of rotors 2 and 4 by opposite amounts. *Pitch* (rotation around the y axis) similarly adjusted using rotors 1 and 3. *Yaw* (rotation around the z axis) controlled by adjusting 2 and 4 together by an equal and opposite amount to adjustments done to 1 and 3.

This thesis explains the development and testing of a hierarchy of neuro-controllers for MAV quadrotor flight. The aim of this work has been to overcome this complex control problem and develop a controller capable of maintaining quadrotor stability during motion in three dimensions. We hypothesize that the adaptive controller developed will be robust and maintain this stability in several types of conditions: un-modeled disturbances (to represent wind gusts), sensor and actuator noise (as is present in all physical environments), and paramet-

ric differences (to show the range in design parameters describing a craft capable of being stabilized by this controller).

Chapter 2 provides background material covering previous work to control quadrotors. It also covers some of the literature on neuro-controllers and their successful application in other domains.

Chapter 3 explains the quadrotor physics model used to develop the simulator. This chapter also explains both PID controllers as well as neuro-controllers, both of which were used in this work.

Chapter 4 discusses the formulation of the specific controller architecture we developed. It goes through details of each of the steps taken to develop a controller for the quadrotor model we had. The reasoning behind some of the necessary trade-off decisions is also explained.

Chapter 5 is a summary of our key experimental discoveries with the final controllers. In this chapter, we provide compelling results showing the increased robustness in our controller compared to a more traditional model-based PID controller.

Chapter 6 provides concluding remarks and a reiteration of the contributions of this work. In this chapter, we also explore some of the potential future avenues for this work.

Chapter 2 – Background & Related Work

This chapter has two focuses, one is to present background on previous methods used to control quadrotors, most of which have not been adaptive in nature. The other focus is to present successful applications of adaptive control. The goal of this thesis has been to apply this type of control to quadrotors and develop a controller that takes advantages of the robustness of adaptive methods.

2.1 MAV Flight Control

Previous work on quadrotor controllers have used traditional model-based controllers. The simplest of these are Proportional Integral Derivative (PID) controllers [7, 13]. This type of controller is explained in more detail in Chapter 3, but it is a relatively easy to implement controller that operates on the error between a desired setpoint and the current value, as well as on how that error changes. The goal is always to try to minimize that error and the work mentioned was able to track the roll, pitch, and yaw angles using this method.

More commonly, quadrotors are controlled using modern and optimal model-based techniques [3–5, 8, 17, 30]. These groups have been able to develop linear quadratic regulators based around a linearized model of the quadrotor dynamics. Although their controllers

had some success, they were tied very closely to the model, and could not handle some of the non-linearities and disturbances experienced in outdoor flight conditions or in flight positions that did not approximate a hover.

These methods have been entirely model-based and MAVs are very susceptible to disturbances and unknown dynamics, making accurate models very difficult and leading to the exploration of adaptive methods for control. A primarily model-based controller, with the addition of neural networks to account for non-linearites in the dynamics and unknown parameters, has proven successful [9, 11, 21, 23]. All of these solutions attempt to capture the non-linearities and adjust the model-based controller to handle them.

Adaptive control methods are often more successful than model-based methods where the operating domains are known to be non-linear. Along these lines, using both neural networks and reinforcement learning to train an entire controller from data points taken during human controlled flight has been explored [12, 30]. However, because of the inherent instabilities of quadrotors, they are not flyable by a human without some stability assistance. Since the previous work uses data taken under human pilot control, it creates an artificial constraint of needing to use these smoothing and stability routines even after the adaptive controller is trained.

2.2 Successful Learning Control Applications

Model-free adaptive control techniques are often used in such challenging and non-linear domains. For more than 40 years, the inverted pendulum, or pole balancing problem has been a standard benchmark for control techniques [1]. In this problem, a cart must move back and forth to keep a pole balanced on end. This is a task doable by almost any human, and yet is a challenging control problem. Traditional control can solve this problem, because the physics of the system can be fully described. The advent of neural networks and reinforcement learning soon made balancing a single pole an entirely trivial problem, solvable without any model of the dynamics of the pole. Adaptive control techniques have moved on to the double-pole problem, where the second pole must remain balanced on top of the first, and even a 2-D pole where a cart may move anywhere on a plane [16].

One subset of adaptive control that has shown strong promise in unknown mobile robot domains, is neuro-evolution. Using neuro-evolutionary techniques very similar to those used in this work, fully autonomous controllers have been implemented and shown to work well with physical land-based robots [15, 20]. Adaptive control was also a critical aspect to Stanley, the successful DARPA grand challenge autonomous vehicle [28]. It has also been used and has shown great promise on board NASA's Deep Space 1 probe [24]. This type of model-free control has been taken into many domains and has a proven success rate, from robotic manipulators [29], to single robot navigation [20], to coordination between multiple autonomous vehicles [25].

Working without complete a priori knowledge of the system is especially advantageous when

the system has complex non-linearaties that will either be a challenge to control, or are difficult to model accurately. In quadrotor flight, both are true. The dynamics of the craft are inherently non-linear. Beyond this, several effects, including ground effects, blade flapping, and dynamics within the motors were not included in the simulator. A controller developed and tested on this simulator would thus need to be shown to be robust in order to maintain control when implemented on actual hardware. This has shown to be the case with supervised reinforcement learning control in the case of autonomous helicopter control [6, 22].

The last few controllers mentioned in the previous section did indeed used model-free techniques for controlling quadrotors. However, they either used them as an add-on to a model-based controller and were targeted at accounting for one set of un-modeled disturbances, or they were used with a pilot providing supervision to the controller to teach it the correct actions. Both of these approaches have drawbacks and require significant knowledge of the system to maintain control.

In this work, instead of using supervised learning and a recorded set of datapoints to determine the *correct* actions, we use a neuro-evolutionary algorithm to arrive at a trained controller. This work uses the real-time NeuroEvolution of Augmenting Topology (NEAT) algorithm [26,27]. This particular algorithm, explained further in Chapter 3, allows modifications to controllers in complex domains that make the design process more efficient.

Chapter 3 – Modeling & Controllers

This chapter develops the model used to simulate the quadrotor flight for both training and testing purposes. The physical parameters of the craft used are also presented. This chapter explains the controller used to solve the problem of stable quadrotor flight, as well as the PID model used in training. Finally, Neuro controllers in general, as well as the particular NEAT algorithm used, are explained in detail.

3.1 Quadrotor Physics Model

The mathematical quadrotor model is based upon the previously developed model [4]. We briefly summarize that development here. As a starting point, there are two coordinate systems to be aware of, that of the earth, and that of the craft. These are related through three successive rotations:

Roll: Rotation of ϕ around the x-axis

Pitch: Rotation of θ around the y-axis

Yaw: Rotation of ψ around the z-axis

The main aerodynamic effects, U_i, applied by the rotors in the body frame are proportional

to the square of the rotor speeds Ω_i:

$$\begin{aligned} U_1 &= b\,(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2) \\ U_2 &= b\,(\Omega_4^2 - \Omega_2^2) \\ U_3 &= b\,(\Omega_3^2 - \Omega_1^2) \\ U_4 &= d\,(\Omega_2^2 + \Omega_4^2 - \Omega_1^2 - \Omega_3^2) \end{aligned} \tag{3.1}$$

where U_1 is the total thrust due to all four rotors, U_2 is the difference between the thrusts of rotors 2 and 4 (i.e. the roll moment), U_3 is similarly the difference between the thrusts of rotors 3 and 1 (i.e. the pitch moment), and U_4 is the difference between the thrusts of oppositely spinning motors (i.e. the yaw moment). The thrust and drag coefficients, b and d respectively, are craft dependent values. There are also gyroscopic effects proportional to the difference in rotor speeds Ω:

$$\Omega = \Omega_2 + \Omega_4 - \Omega_1 - \Omega_3 \tag{3.2}$$

In order to reach the equations of motion, we start with a force balance,

$$ma = R \sum F_b \tag{3.3}$$

where R is the rotation matrix and F_b are the body forces. This yields the x, y, and z acceleration of the craft in earth coordinates. To find angular accelerations we also require a torque balance,

$$I\alpha = -\omega \times I\omega - J_r(\omega \times \hat{z})\Omega + \tau_b \tag{3.4}$$

where I is the body inertia matrix, J_r is the rotor inertia, α is the angular acceleration, ω is the angular velocity, and τ_b are the airframe torques.

This leads to the equations of motion in terms of the above applied forces and torques:

$$\begin{aligned}
\ddot{x} &= (\cos\phi \sin\theta \cos\psi + sin\phi \sin\psi) \frac{1}{m} U_1 \\
\ddot{y} &= (\cos\phi \sin\theta \sin\psi - sin\phi \cos\psi) \frac{1}{m} U_1 \\
\ddot{z} &= -g + (\cos\phi \cos\theta) \frac{1}{m} U_1 \\
\ddot{\phi} &= \dot{\theta}\dot{\psi} \left(\frac{I_y - I_z}{I_x} \right) - \frac{J_r}{I_x} \dot{\theta}\Omega + \frac{l}{I_x} U_2 \\
\ddot{\theta} &= \dot{\phi}\dot{\psi} \left(\frac{I_z - I_x}{I_y} \right) + \frac{J_r}{I_y} \dot{\phi}\Omega + \frac{l}{I_y} U_3 \\
\ddot{\psi} &= \dot{\phi}\dot{\theta} \left(\frac{I_x - I_y}{I_z} \right) + \frac{1}{I_z} U_4
\end{aligned} \tag{3.5}$$

where x, y, and z represent the craft's position in the earth reference frame, and ϕ, θ, and ψ are the roll, pitch, and yaw angles as explained above. The other variables, not including U_i and Ω, are parameters of the craft. The values of these parameters, as used in this work, are in Table 3.1 [2].

Assuming a constant acceleration over some small time step, we can determine the changes in velocity and position during that timestep. An appropriate choice of timestep provides us with a relatively accurate simulation based on a few parameters of the quadrotor.

Table 3.1: Physical Quadrotor Parameters

Variable	**Value**	**Description**
m	$0.4794\,kg$	mass
l	$0.225\,m$	craft diameter
b	$3.13 \times 10^{-5}\,N\,s^2$	thrust factor
d	$9 \times 10^{-7}\,N\,m\,s^2$	drag factor
J_r	$3.74 \times 10^{-5}\,kg\,m^2$	rotor inertia
I_x	$0.0086\,kg\,m^2$	moment of inertia along x
I_y	$0.0086\,kg\,m^2$	moment of inertia along y
I_z	$0.0172\,kg\,m^2$	moment of inertia along z

3.2 Controllers

In this work, we use two type of controllers that regulate inputs to a system in order to affect a desired change in the system's output. We develop a neuro-controller, however in order to train this neuro-controller we use a PID controller. Both types are explained below.

3.2.1 PID Controllers

PID stands for Proportional Integral Derivative and is a type of closed loop feedback controller. The operation of a PID controller is shown in Figure 3.1. A separate controller is developed for each output variable that must be controlled. When the inputs to the system are coupled, this can cause trouble as two different controllers try to adjust the same inputs to affect different outputs. As will be explained later, the dynamics of a quadrotor can be separated and this coupling removed.

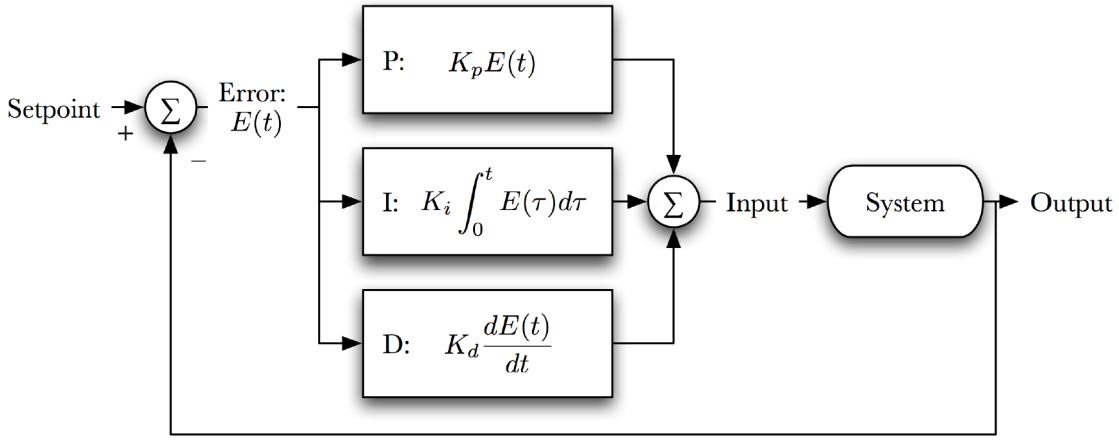

Figure 3.1: Showing the form of a PID Controller, K_p, K_i, and K_d are tuned so that the sum shown (the input to the system) drives the output towards the desired setpoint.

The basic calculation for nearly any feedback controller is to calculate an error value for every timestep. This is just the difference between the desired setpoint, and the actual output. Both the integral and derivative of this error term is calculated in order to provide information about how the error is changing. The error, its integral and derivative are then each multiplied by a different gain K_p, K_i, and K_d respectively. The results are summed together and this is used as the input into the actual system. The system is treated as a black box, and the PID makes continual adjustments as it attempts to drive this error term to zero and make the output equal the desired setpoint.

When this is broken down into a system with discrete timesteps it becomes:

$$I_0 = K_pE(t_0) + K_i\left(\sum_t E(t)\Delta t\right) + K_d\frac{E(t_0) - E(t_{-1})}{\Delta t} \tag{3.6}$$

where $E(t_0)$ is the error at the current timestep, $E(t_{-1})$ the error at the previous timestep, Δt is the size of the timestep, K_p, K_i, and K_d are the controller gains, and I_0 is the input to be set for the system for the current timestep.

The system gains, K_p, K_i, and K_d, are tuned to get the desired performance. This performance is often described in terms of the following four variables: rise time, overshoot, settling time and steady-state error. All of these are measured in response to a step change in the setpoint. Rise time is a measure of how long it takes to reach a new equilibrium. Overshoot is a measure of how much the output overshoots the desired setpoint before returning to the equilibrium. Settling time is a measure of how long the system oscillates before reaching equilibrium. Finally, the steady-state error is how far off the final equilibrium is from the desired setpoint.

There are several formalized methods for tuning the system gains, however for many systems, the most efficient method for an initial set of gains is an order of magnitude manual estimation. Once the parameters are in the ballpark, further refining is possible with the guidelines shown in Table 3.2.1.

Table 3.2: Effect of Increasing PID Gains

Parameter	**Rise time**	**Overshoot**	**Settling Time**	**Steady-State Error**
$\mathbf{K_p}$	Decrease	Increase	-	Decrease
$\mathbf{K_i}$	Decrease	Increase	Increase	Eliminate $\forall K_i \neq 0$
$\mathbf{K_d}$	-	Decrease	Decrease	-

3.2.2 Neuro Controllers

Neuro-controllers develop from a stochastic search of a population of neural networks. A neural network is a function approximator that maps inputs to outputs. A typical topology for a neural network is shown in Figure 3.2. It consists of three inputs, each connected to six hidden nodes, which are in turn connected to two outputs. This example is referred to as a fully connected network because each node on any layer has a connection to every node on the previous and next layer. Each of this connections, often called links, is associated with a weight. Each node then has a value equal to the weighted sum of the outputs of the nodes of the previous layer, all evaluated by some activation function. This activation function is generally a non-linear scaling, which is what gives the neural network the ability to approximate almost any function.

Neural networks are often trained by adjusting the values of the weights for each link. This affects how important each input, or combination of inputs is to each output. When the desired outputs are known for a given set of inputs, the value for these weights can be calculated. However, in many real-world domains, the desired output is not known. For this type of domain, quadrotor control being an example, the use of an evolutionary algorithm is a good way to search through the possible weight values in order to find a useful mapping from the inputs to the outputs.

An evolutionary algorithm is essentially a search method. It starts from a population of neural networks, evaluates each of these networks against some metric, and higher performing networks are then mutated with some probability, in order to search for an optimal

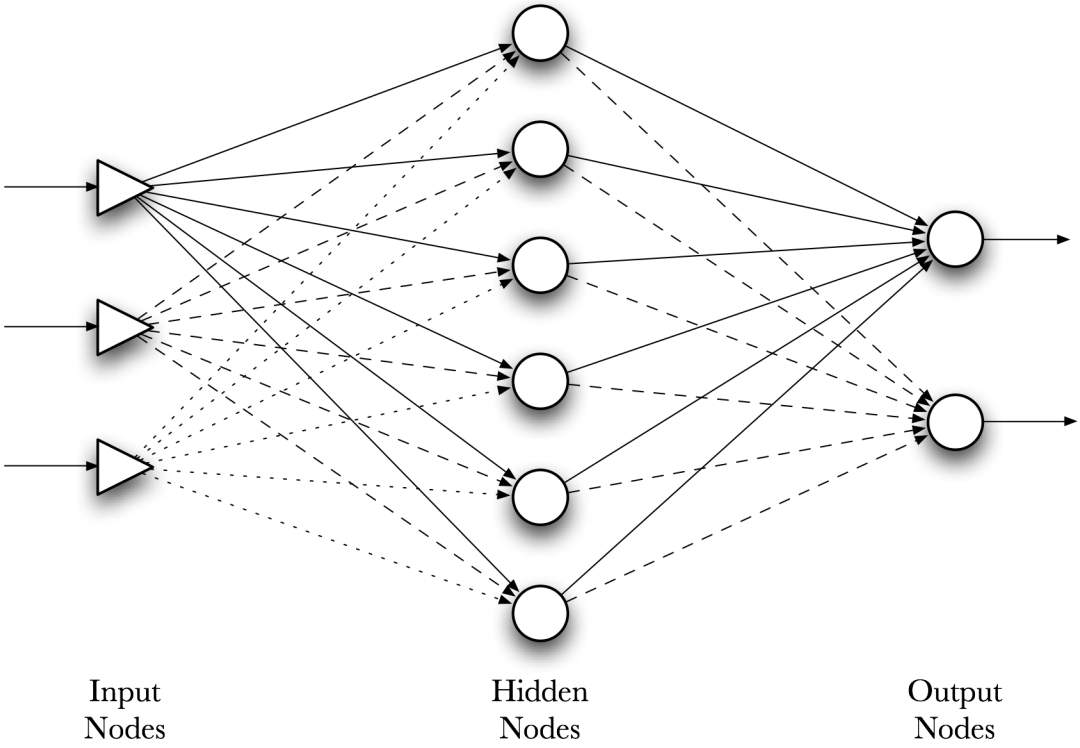

Figure 3.2: Shows a typical, single hidden layer, fully connected, neural network

solution. The following algorithm explains the steps involved in a generic evolutionary algorithm:

1. Randomly initialize the weights of a population of networks
2. For each network in the initial population
3. Control the system with the network for some number of timesteps
4. Evaluate the network
5. Repeat steps 6-11 until a good enough solution is found
6. Select the best network with some probability
7. Mutate this network
8. Control the system with this network for some number of timesteps
9. Evaluate the network
10. Insert network into the population
11. Remove the worst network

12. Use found solution for actual control

This remains very generic; different algorithms have very different methods to select the best network as well as different methods for mutating the network. The amount of time to test a controller and how to evaluate it are going to be domain specific choices. Also domain specific are: the size of the initial population, the topology selected for the network, and the diversity in the randomness of the initial population.

For most neuro-controllers, the topology is determined by the designer up front. However, in complex domains, this is often challenging and requires significant testing to determine the correct topology. There is a tradeoff between having too many nodes, and therefor too many links, which makes training very slow, and too few nodes, such that there is not enough flexibility to accurately capture the dynamics of the system.

The complexity of the quadrotor system means that determining the topology upfront would be challenging. To overcome this, we use the NeuroEvolution of Augmenting Topologies (NEAT) algorithm. This algorithm allows mutations to change the topology in addition to the weights. The number of nodes, and the existence of links between them can change with each mutation. This allows for complicated structures to evolve that capture the dynamics of the system.

Chapter 4 – Controller Formulation

Flying a quadrotor, even in simulation, is a difficult problem because of the sensitivity to differences in rotor speeds. One shortcoming of neuro-evolutionary algorithms is that in large search domains, they require many iterations before even a mediocre solution is found. Although neuro-evolution work done with land-based robots has been very successful, a poor solution in that domain means the robot does not move towards the goal but may still provide enough information for learning [20]. A poor solution in an aerial vehicle means a crash and provides little useful information for learning. Although, theoretically, the information needed for learnability could be encoded in a single objective function, the search space is very large, and the solution space of successfully flying controllers is very small. Since we have an understanding about the way the craft responds to speed differences in the four rotors, we used this knowledge in the development of our controller.

Instead of having a single controller with inputs of a desired position and outputs of the required rotor speeds to move towards that point, we developed a hierarchy of controllers. At the highest level, the position controller is responsible for moving the craft by suppling the attitude controller with the desired roll, pitch, and yaw angles as well as the vertical velocity. This attitude controller is further decomposed into several simpler agents that each make adjustments to the rotor speeds to independently control the four attitude variables.

4.1 Attitude Controller

The attitude controller is responsible for stability and movement of the craft. It adjusts the rotor speeds to place the craft into a desired attitude and corresponding trajectory, described by roll, pitch, and yaw angles, and a vertical velocity. Due to the quadrotor's attitude being sensitive to small variations in rotor speed, training the attitude controller directly proved challenging. Instead, it was broken down into the four agents, adjusting the roll, pitch, and yaw angles and vertical velocity independently using the appropriate three inputs. Figure 4.1 shows the incremental steps taken to develop, train, and select the final attitude controller. These steps are explained later in detail.

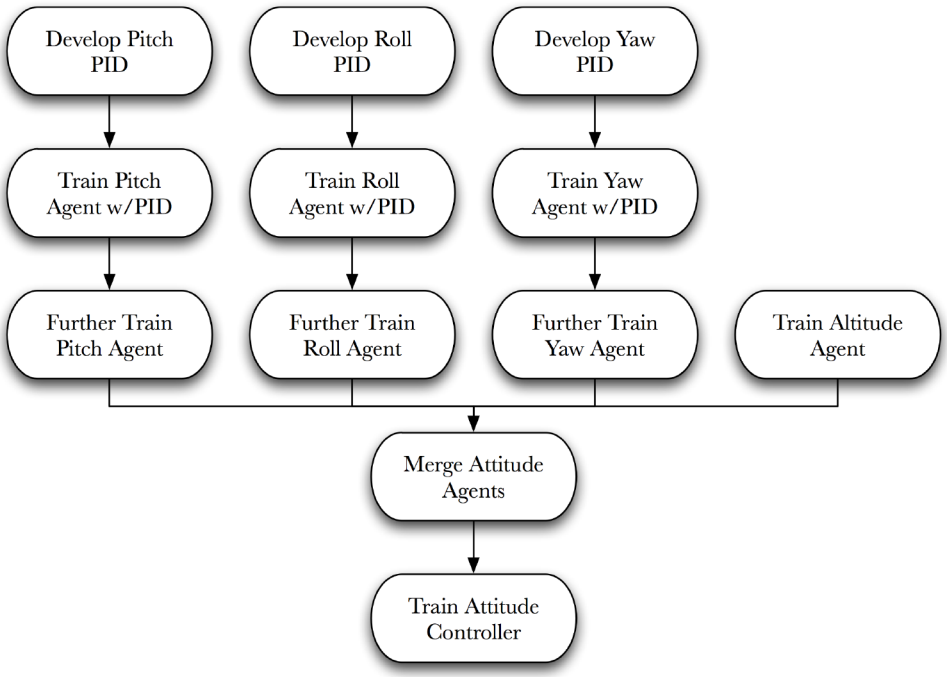

Figure 4.1: Showing each step taken in the process of developing the attitude controller.

As described previously, neuro-evolutionary controllers involve an undirected stochastic search. For an aerial domain, this shortcoming makes learning very challenging due to the large search space and narrow range of controllers capable of maintaing flight. To overcome this, instead of training neuro-controllers from scratch and ranking them based on how successfully they achieved the desired attitude, we developed PID controllers capable of maintaing flight and trained the neuro-controllers to match the output from the PID for the same inputs.

The four agents of the attitude controller receive the same inputs that a PID controller does, shown in Figure 4.2. These agents receive information about the desired attitude as well as sensory information in the form of an error between the desired and actual, as well as the derivative and integral of that error for all four attitude variables. This provides the controller with information about the absolute error, and how that error is changing. From this, the controller sets the desired rotor speeds for all four rotors.

This parallels the operation of a PID controller. PID controllers were selected to be trained against, over other model-based controllers, due to their mathematical simplicity, and their ability to be roughly tuned online in short order. Prior work has shown that PID controllers are successful at controlling the attitude of quadrotors, but had difficulty controlling altitude. For this reason, the altitude controller was trained differently, as will be explained further below.

With the goal of seeding the neuro-controller search with a working PID controller we can explore the steps shown in Figure 4.1 in more detail. The first step was to develop a baseline

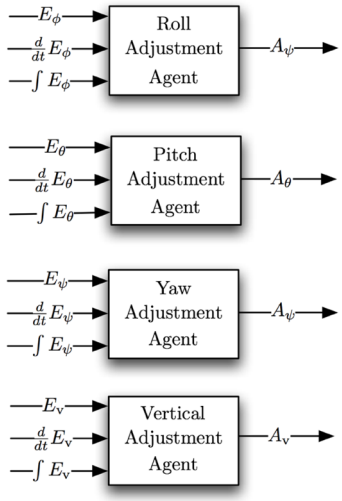

Figure 4.2: Showing the structure of the attitude agents which independently generate adjustments to the rotor speeds to control the roll, pitch, and yaw angles as well as the vertical velocity. Each agent receives as input a difference from the desired setpoint as well as the difference's integral and derivative.

PID controller that would provide stable flight. A separate set of weights K_p and K_d were developed for each of the three angles. Initial testing indicated no steady-state error so there was no need for an integral term, so $K_i = 0$ to prevent integral windup concerns. Tuning was done online, making increasingly finer adjustment to the parameters to achieve rise times on the order of 3 seconds. Overshoot upon reaching the desired angle was kept under 10 %. Final PID values used in later steps, are shown in Table 4.1. The resulting PID controllers were by no means fully tuned, but they kept the quadrotor in a stable flight pattern for pitch and roll angles in the range of $[-\frac{\pi}{3}, \frac{\pi}{3}]$. Achievable yaw angles, allowed pointing the craft in any direction $[-\pi, \pi]$.

The next step was to train the neuro-controllers for all three angles from these PID values. As mentioned, the realtime NEAT algorithm was used for this. A population of 100 po-

Table 4.1: PID Parameters

	$\mathbf{K_p}$	$\mathbf{K_d}$	$\mathbf{K_i}$
Roll Controller	0.0005	0.08	0
Pitch Controller	0.0005	0.08	0
Yaw Controller	0.005	0.18	0

tential controllers was created. Each started with zero hidden nodes and an otherwise fully connected network with random weights between -3 and 3. For each mutation, the NEAT algorithm allowed hidden nodes and additional links to be added with 3 % and 5 % probability respectively. The remaining time, mutations changed the weight values, but not the structure of the controller. Because the adjustments made to the rotor speeds would need to be both positive and negative depending on the sign of the desired attitude variable, a hyperbolic tangent was used as the activation function, giving activated nodes values between -1 and 1.

In order to start from a stable hover, the rotor speed was first set to a level that just overcame gravity. Each agent was trained separately, with the developed PID controllers making adjustments to eliminate rotation along the axes not currently being used in training.

Figure 4.3 shows, in detail, how training occurred for each of the three attitude angles. The paragraph numbers below refer to the lines in this figure.

Steps 1-3: Each training iteration, the worst controller was removed. The best controller was selected with some probability (i.e. some chance of selecting a random controller from the population rather than the best). The selected controller was then mutated as described above. This new controller then needed to be evaluated.

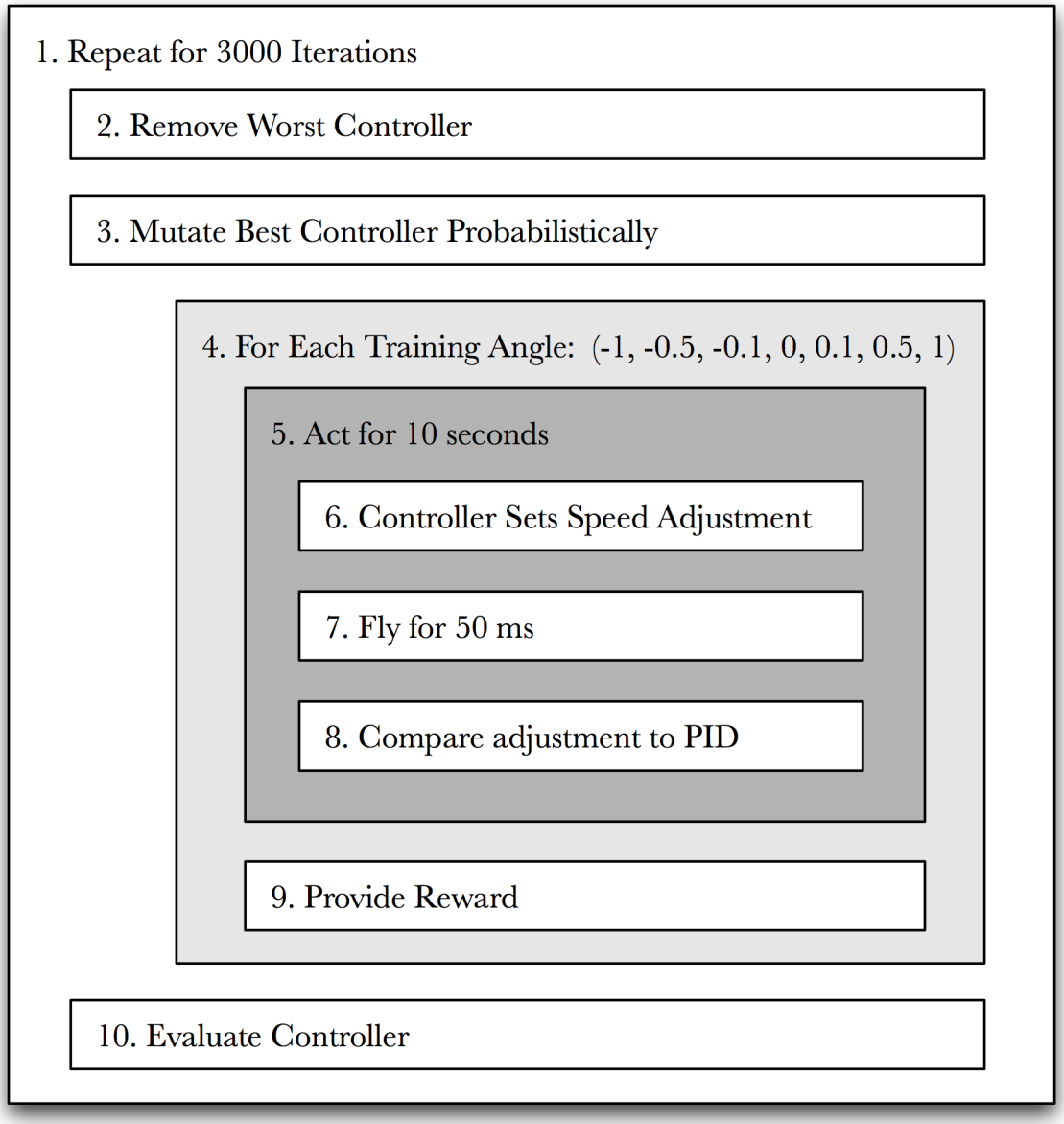

Figure 4.3: Showing the process of training each of the angular agents.

Step 4: This evaluation involved running the controller for ten seconds each, through a series of different angles. For the roll and pitch agents, these angles were $(-1, -0.5, -0.1, 0, 0.1, 0.5, 1)$ radians. For the yaw agent, the angles also included $(-2, 2)$.

Steps 5-9: During those ten seconds of simulation for each angle, the controller was asked to make adjustments every 50 ms. Each neuro-controller adjustment was compared to the adjustment the PID controller would have made, and this comparison was stored as a reward.

Step 10: The final fitness was based on these comparisons:

$$F = e^{-\sum |A_{nn} - A_{pid}|} \tag{4.1}$$

where A_{nn} is the adjustment from the neuro-controller and A_{pid} is the adjusted for the same inputs as calculated by the PID controller. This fitness is based on the distance between the PID and neuro-controller rotor speed changes. The fitness was the negative exponential of this distance so that small distances would equate to large fitnesses near one and large distances to a fitness near zero.

This process was run for 3000 iterations for each angular agent. 3000 iterations were used after testing revealed that this was a sufficient length of training to achieve good performance from the neural network. At this point, the neuro-controllers were performing as well as the PID controllers they were trained on. In most cases, the neuro-controllers were actually performing better at this point, due to their greater ability for non-linear interpolation through out the range of desired angles.

In order to increase the performance of the neuro-controllers, they underwent further training at this point. A new population was developed from the best controllers in the previous training with up to 20 % mutation of the weights. This new population was evaluated against the desired target angle based on the actual rise time and overshoot. This population was trained for another 1000 episodes at the end of which, the best controller was selected to be used in the combined attitude controller. This completed the training for the independent angular agents.

The final agent, controlling vertical velocity was trained differently. Since previous work had difficulty in using a PID controller to adjust the altitude, the fourth agent was evaluated based on the actual vertical velocity seen during the 10 seconds of simulation. Although the angles can be controlled independently of each other, the thrust adjustment will be different depending on the roll and pitch angles of the craft. An increase in the thrust adjustment of a level quadrotor will only increase the vertical velocity, however when the craft is tilted, an adjustment of the same amount will result in a smaller increase in the vertical velocity, as well as an increase in speed in the direction of the tilt.

To account for this, each evaluation of a vertical velocity controller included 70 different configurations, each run for ten seconds. This allowed for a wide range of roll and pitch values to be used in any evaluation. This was run for 1000 training episodes after which the best agent was selected for incorporation into the attitude controller.

When all four agents were trained, they were combined as shown in Figure 4.4. These four

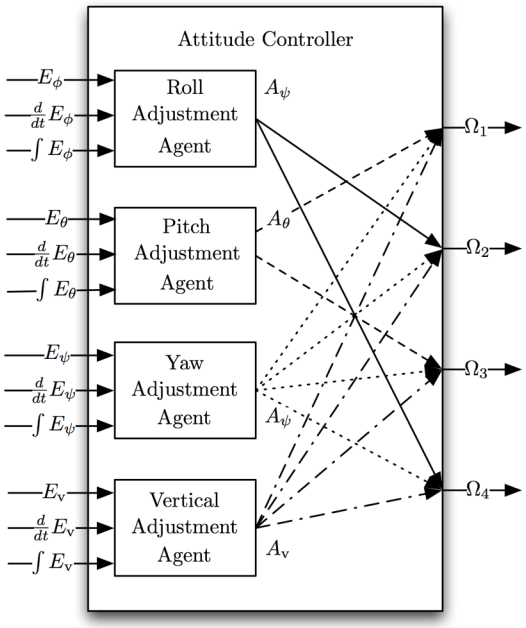

Figure 4.4: Showing how the inputs and outputs for each of the agents are combined to form a single, higher level, attitude controller that outputs the rotor speeds based on the adjustments calculated by the four sub-agents.

attitude parameters are combined mathematically as follows:

$$\begin{aligned} \Omega_1 &= A_\mathrm{v} - A_\theta - A_\psi \\ \Omega_2 &= A_\mathrm{v} - A_\phi + A_\psi \\ \Omega_3 &= A_\mathrm{v} + A_\theta - A_\psi \\ \Omega_4 &= A_\mathrm{v} + A_\phi + A_\psi \end{aligned} \tag{4.2}$$

where Ω_i is the speed of each rotor, A_ϕ, A_θ, and A_ψ are the adjustments to control the roll, pitch and yaw angles, and A_v is the adjustment to control the vertical velocity. This resulted in a single attitude controller that produces the desired rotor speeds from the original twelve inputs.

Once all four agents were trained, they were combined into a single attitude controller as mentioned previously. Depending on the final operation environment, this attitude controller could be further trained to handle specific types of disturbances. For this work, further training was not provided to the controller and testing showed that training separately was sufficient to yield a final attitude controller that can be used to stabilize the quadrotor into any requested attitude.

After training was complete for each of the agents, they had developed one or two hidden nodes partially connected to the inputs and outputs. Further training of the attitude controller did not alter the structure of the combined controller, but did tweak the weights a little more.

4.2 Position Controller

To actually move the craft, a higher level controller must provide attitude targets to the attitude controller. For this work, the higher level controller was only concerned with moving a craft to a new (x, y, z) position. However, depending on the mission, this may be better suited by a controller with greater perception of its surroundings and the ability to perform path-planning.

In this work, the position controller receives as inputs the difference between the desired and current positions in the x, y, and z directions, and the current velocity in all three directions. The controller outputs the attitude needed to move towards the desired position. These are

both shown in Figure 4.5.

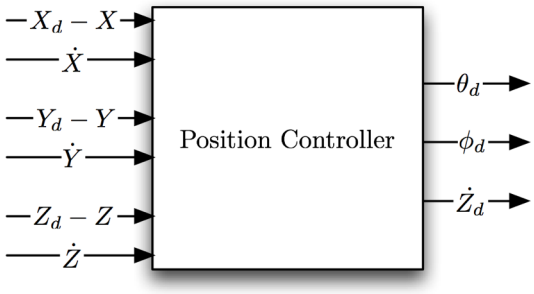

Figure 4.5: Showing the inputs of the difference between the desired and current positions as well as the current speeds, and the outputs of the desired roll and pitch angles as well as the desired vertical velocity. The desired yaw angle is always set to zero, since full motion is achievable by setting roll, pitch and vertical velocity.

Due to the quadrotor's flexibility, only two attitude angles are needed to reach any position. Although using all three angles would allow various pointing maneuvers (e.g. changing spatial position through roll and pitch, while the yaw angle aims a solid-mounted camera at a fixed point), for this work the desired yaw angle was set to 0, allowing the full motion of the craft to be controlled by only the roll and pitch angles in conjunction with the desired vertical velocity. This greatly reduces the search space for successful controllers.

Training consisted of evaluating each potential controller based on the ability to reach a desired position as well as the total distance travelled in doing so. The set of training destinations included a grid of points covering every direction of travel from the origin. Initial population and mutation properties were similar to those used for the attitude agents. Training time however, took upwards of 15 000 episodes to provide a controller with direct travel towards the destination.

This training significantly altered the structure of the position controller. Seven hidden nodes were added, as well as more complicated links between them, resulting in more than 60% of the final links being additional links connecting hidden nodes. Over 73 different nodes were added during the training process with only the seven mentioned remaining as part of the final controller. This added complexity shows the power of using NEAT to explore mutations to both the structure and the weights during the training process.

Chapter 5 – Experimental Comparison: Neuro and PID Controllers

Once the development of the entire hierarchy of controllers was complete, we performed multiple experiments to test their robustness. Our testing included basic verification of our controller to maintain stable flight through series of waypoints. It also tested the ability to reject disturbances, such as wind gusts, and to remain robust in the face of both sensor and actuator noise. Finally, we tested how our design would function on a quadrotor of different design parameters.

Our testing bore out our hypothesis that the developed adaptive controller is robust and maintains craft stability in several types of conditions: un-modeled disturbances (to represent wind-gusts), sensor and actuator noise (as is present in all physical environments), and parametric differences (to show the range in design parameters describing a craft capable of being stabilized by this controller). The success of the controller in each of these areas, as well as the controller's ability to move the craft through series of waypoints is explored in this section.

5.1 Waypoint Control

The most important test of the controller was to see if it is able to perform actual flight. For this test, a series of waypoints was selected. Each waypoint was given to the controller and when the craft had reached stability around that point, a new waypoint was provided. Figure 5.1 shows the craft moving through a selection of waypoints as well as showing the optimal path. It is evident in Figure 5.1a that on a very small scale this controller is not finding the optimal path, but is closer to achieving it than the PID controller. When this same pattern is followed on a much greater distance scale, the path is much smoother, as seen in Figure 5.1b. The waypoints in the first part of the figure are roughly a craft diameter apart. It is thus not surprising that the path is not the direct line, the quadrotor requires greater space to make a move and re-stabilize in a new position. When the craft is moving farther this is not as big a concern and it can track a direct route to the waypoint.

Table 5.1 shows a comparison between the total distance traveled when under PID control, as well as Neuro control. Both of these are then compared to the optimal, straight line path. These are for random sets of waypoints. Although the neuro-controller travels 50 % more distance then optimal while moving through these waypoints, the PID controller takes nearly twice the distance on average to move between waypoints. Were this on a physical system, not only does the quadrotor travel more distance under PID control, but there would be a direct cost in terms of energy and the speed to arrive on location as compared to the neuro-controller. The actual energy savings would depend on the speeds and maneuvers used to travel that extra distance, but a 35 % decrease in the amount of circuitous travel will

translate into significant energy savings.

Table 5.1: Waypoint Tracking Distances Travelled & Percent Over Optimal

Optimal Path	Neuro Controller Path		PID Controller Path	
9.33	12.76	37 %	16.21	74 %
10.60	17.06	61 %	19.47	84 %
10.29	13.97	36 %	16.80	63 %
9.89	15.45	56 %	18.97	92 %
10.32	14.17	37 %	16.82	63 %
8.69	14.85	71 %	18.25	110 %
9.80	15.39	57 %	18.40	88 %
8.49	13.47	59 %	15.25	80 %
10.21	14.15	39 %	18.13	78 %
9.53	15.64	64 %	21.26	123 %
9.71	14.69	51 %	17.96	85 %

5.2 Disturbance Rejection

One major obstacle for MAVs is disturbances. The effects of even very small wind gusts become quite noticeable due to the craft's light weight. This same weight and corresponding small size requires slower controller hardware unable to process that a disturbance is happening, only to indicate that it has. Disturbance rejection is thus modeled as discontinuous jumps in attitude and/or position from which the controller must recover.

We tested the controller by knocking the craft about the roll and pitch axes with isolated disturbances for each axis in addition to disturbances affecting both axes. Figures 5.2 and 5.3 show the effects of disturbances on the pitch and roll axes respectively. Figure 5.4 shows the effect of disturbing in both axis at the same time. In both cases the figure shows the

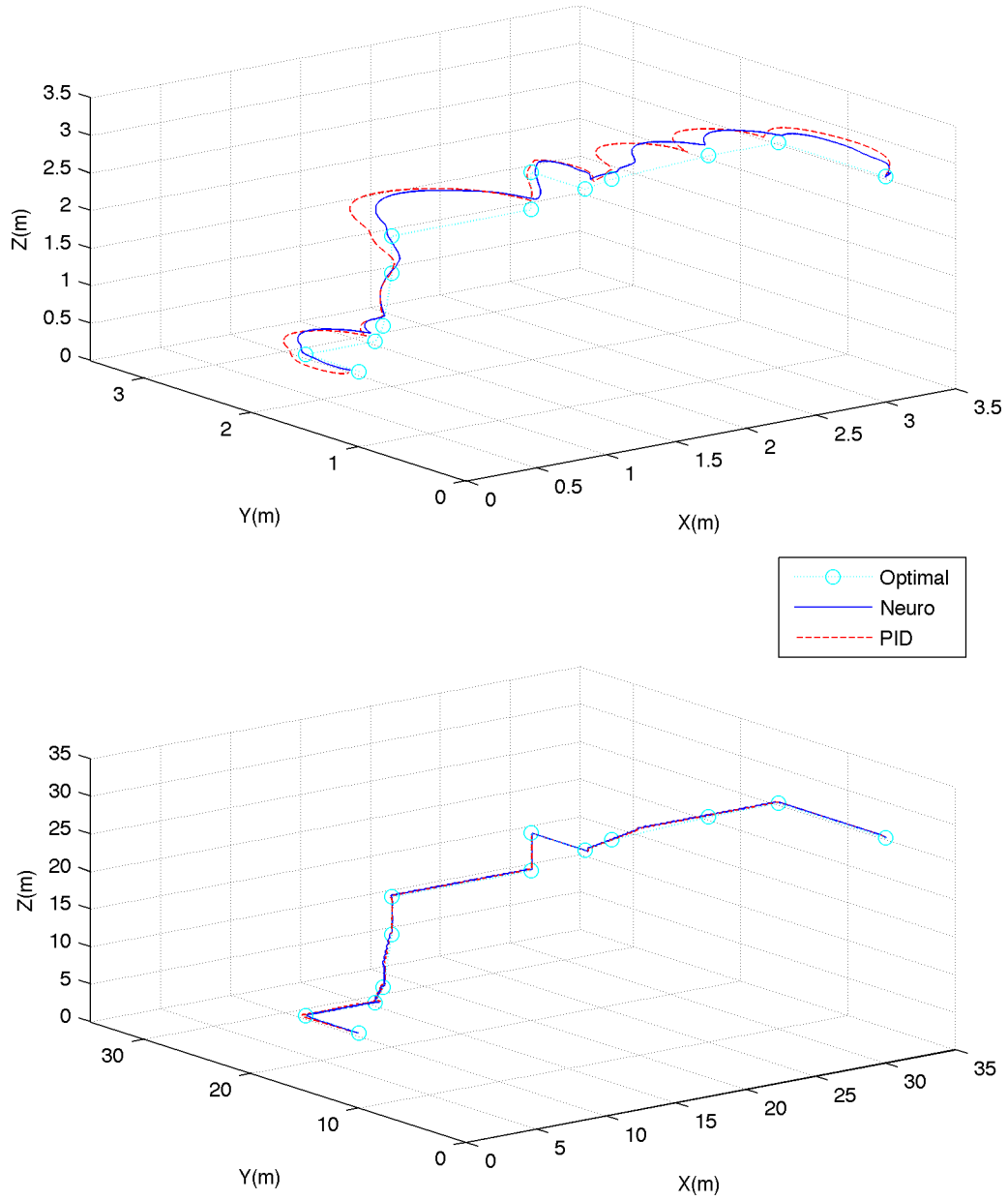

Figure 5.1: Final controller and PID controller piloting through a series of waypoints. The optimal straight line path is also shown, with the waypoints identified. The top figure shows waypoints a few crafts lengths away. The bottom figure shows the same pattern of waypoints where the distances have been scaled upwards an order of magnitude. The larger distances resulted in much smoother quadrotor flight paths, but both cases successfully reach the given waypoints more efficiently than the PID Controller.

effects of disturbances buffeting the craft every 30 seconds, knocking the pitch angle 60°. The neuro-controller is very successful at stabilizing the craft, and restoring it to its original hover position in less than three seconds regardless of the axis (or axes) disturbed. This is vastly superior to the PID controller which is unable to restore the craft's position before the next disturbance strikes. The PID controller does maintain stability for disturbances affecting the pitch or roll axes independently, but is not able to quickly restore the craft's position. When a wind gust is simulated to affect both pitch and roll axes simultaneously, the PID is only able to maintain stability for the first few simulated gusts. After that, the PID loses control, and flight stability is lost, as seen in Figure 5.4.

Figure 5.5 shows that the a 60° disturbance is right at the extreme edge of the envelope the PID controller can handle for pitch disturbances, and 50° is about the limit for roll disturbances. The neuro-controller however, can still recover after being flipped completely upside down. In this case, it does require several meters of altitude in which to recover, but it performs a complete recovery in under 15 seconds. When a disturbance occurs in both axes the PID can handle even less of a disturbance as was evident in the previous position recovery plots. Combined disturbance were tested for the same amount of disturbance along each axes at the same time (e.g. 30° of pitch disturbance and 30° of roll disturbance).

5.3 Robustness

Real world controllers, unlike in simulation, must deal with noisy data. This comes in both the form of noisy sensors describing the craft's position and attitude, and noisy actuators

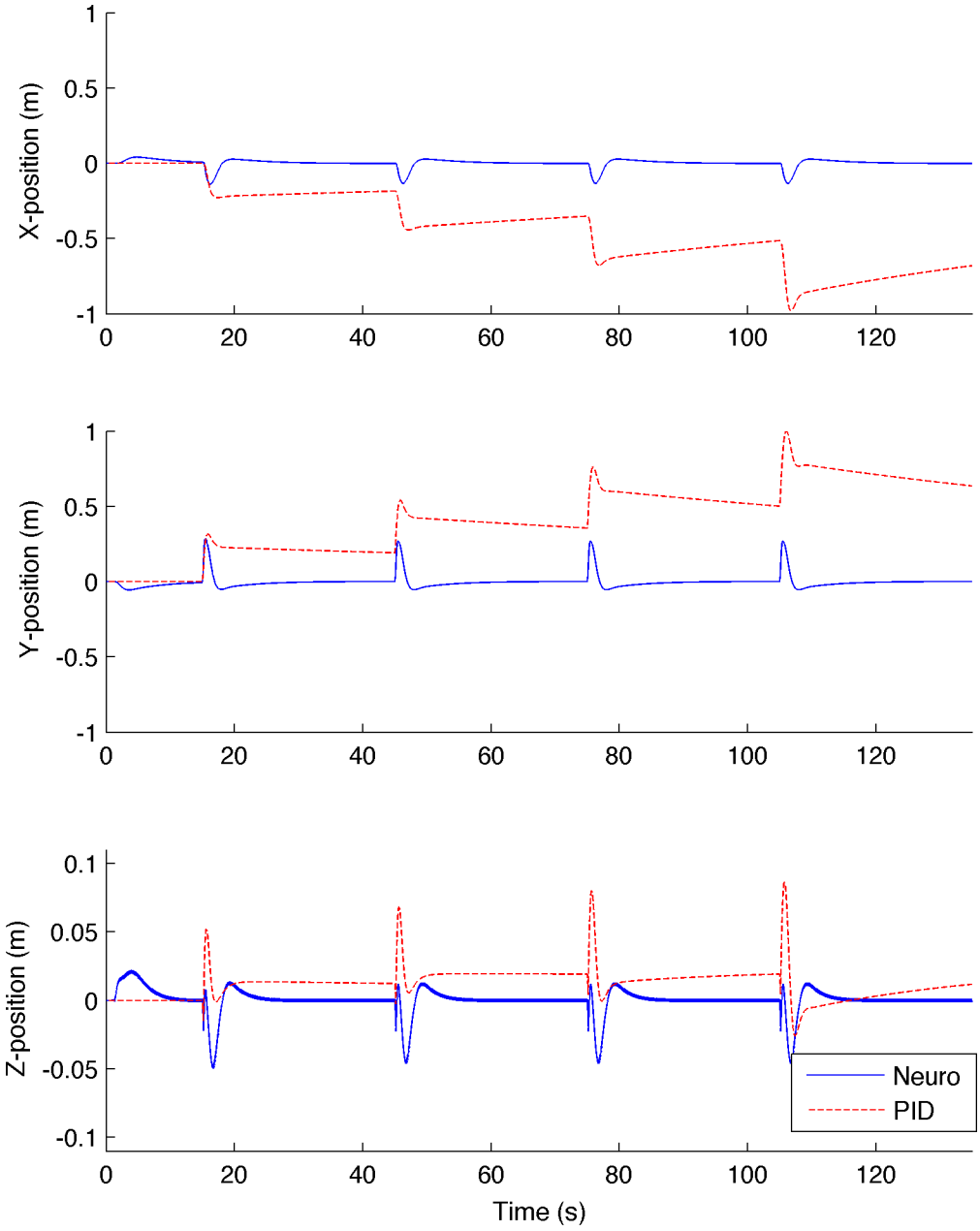

Figure 5.2: Showing position recovery for repeated pitch angle disturbances of 60° occurring every 30 seconds. The PID controller stabilizes the craft but would take several minutes to restore it to the starting location. The neuro-controller is able to stabilize and restore the position in less than 3 seconds.

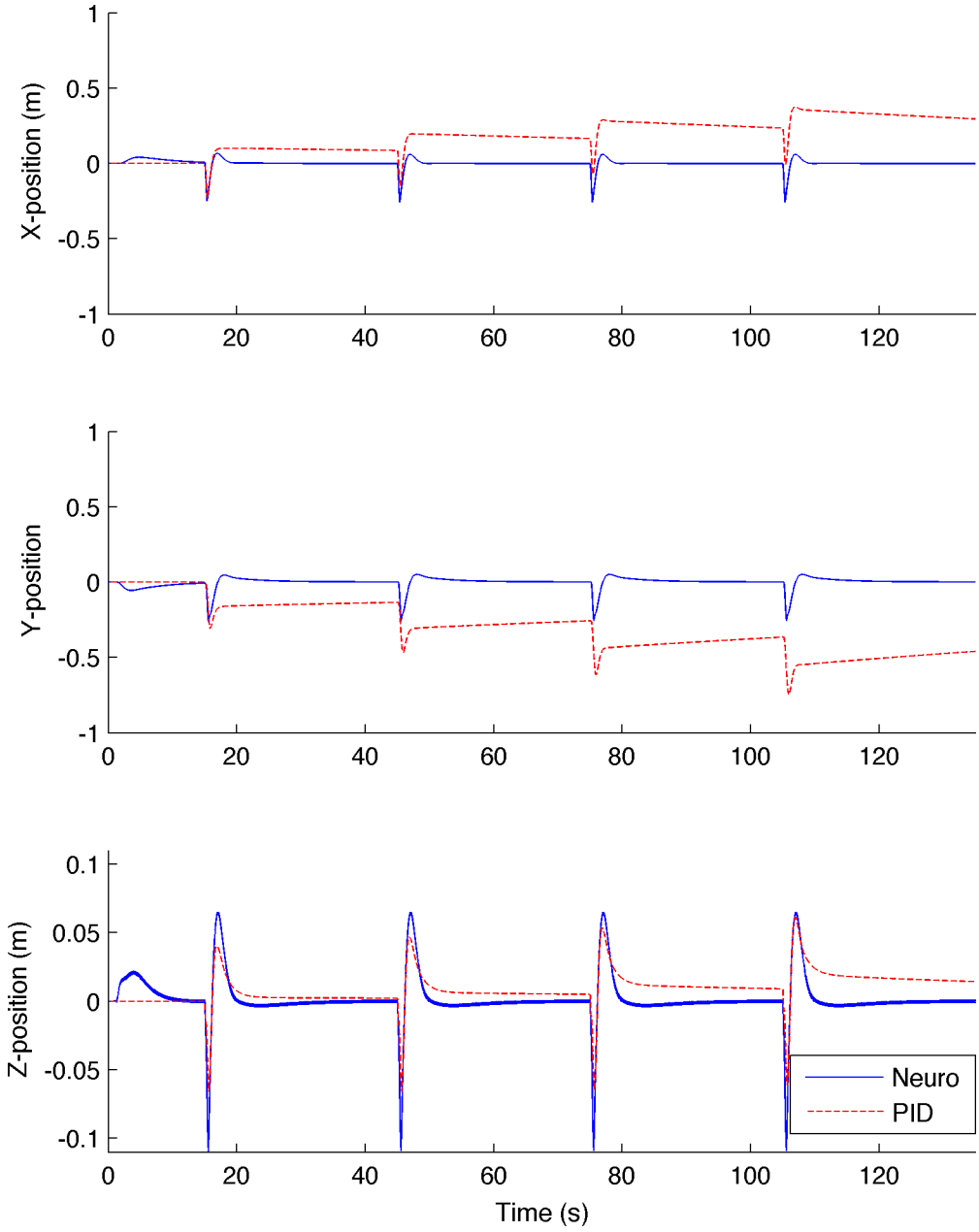

Figure 5.3: Showing position recovery for repeated roll angle disturbances of 50° occurring every 30 seconds. The PID controller stabilizes the craft but would take several minutes to restore it to the starting location. The neuro-controller is able to stabilize and restore the position in less than 3 seconds.

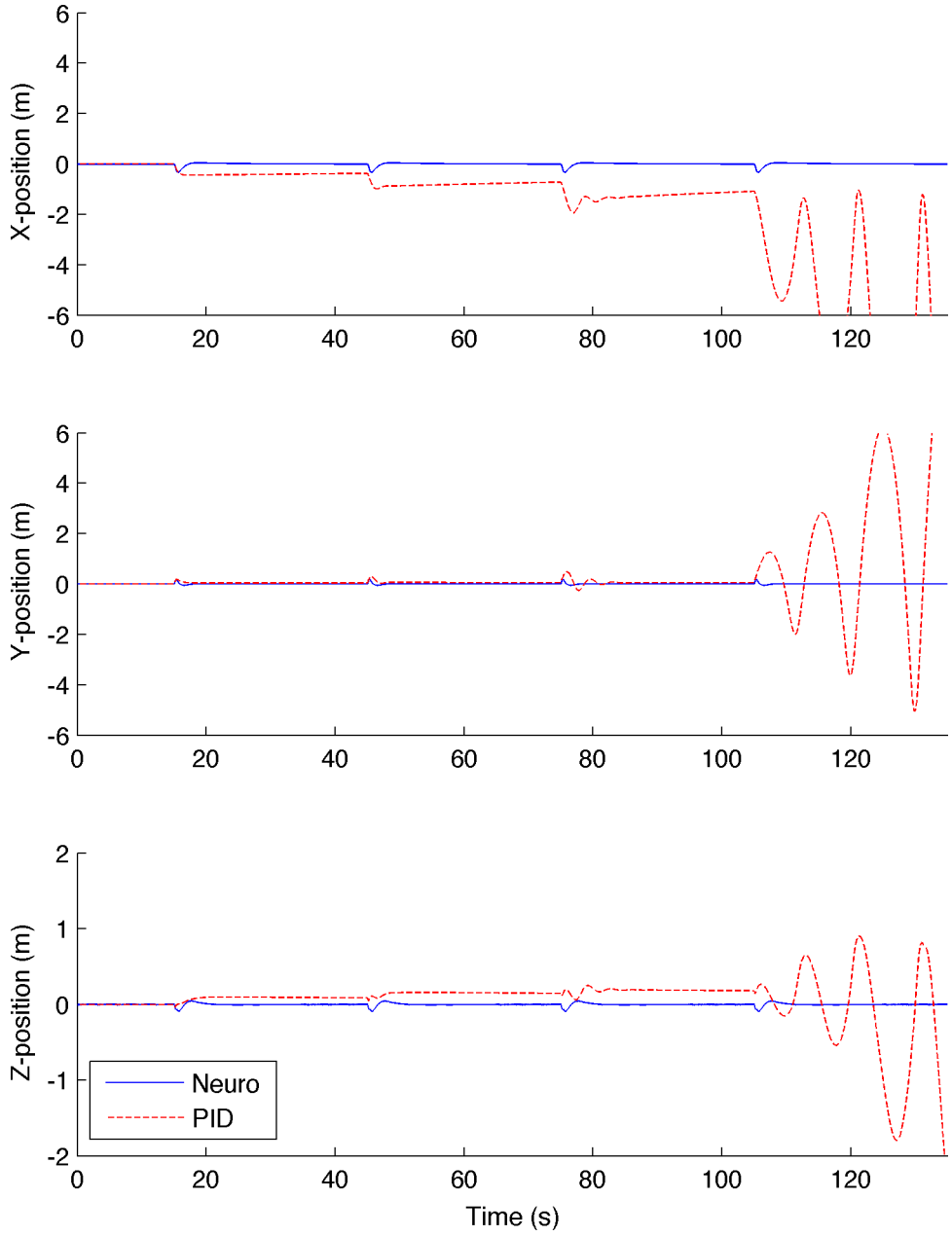

Figure 5.4: Showing position recovery for repeated angle disturbances of 60° in both pitch and roll angles occurring every 30 seconds. The PID controller stabilizes the craft but would take several minutes to restore it to the starting location. The neuro-controller is able to stabilize and restore the position in less than 3 seconds.

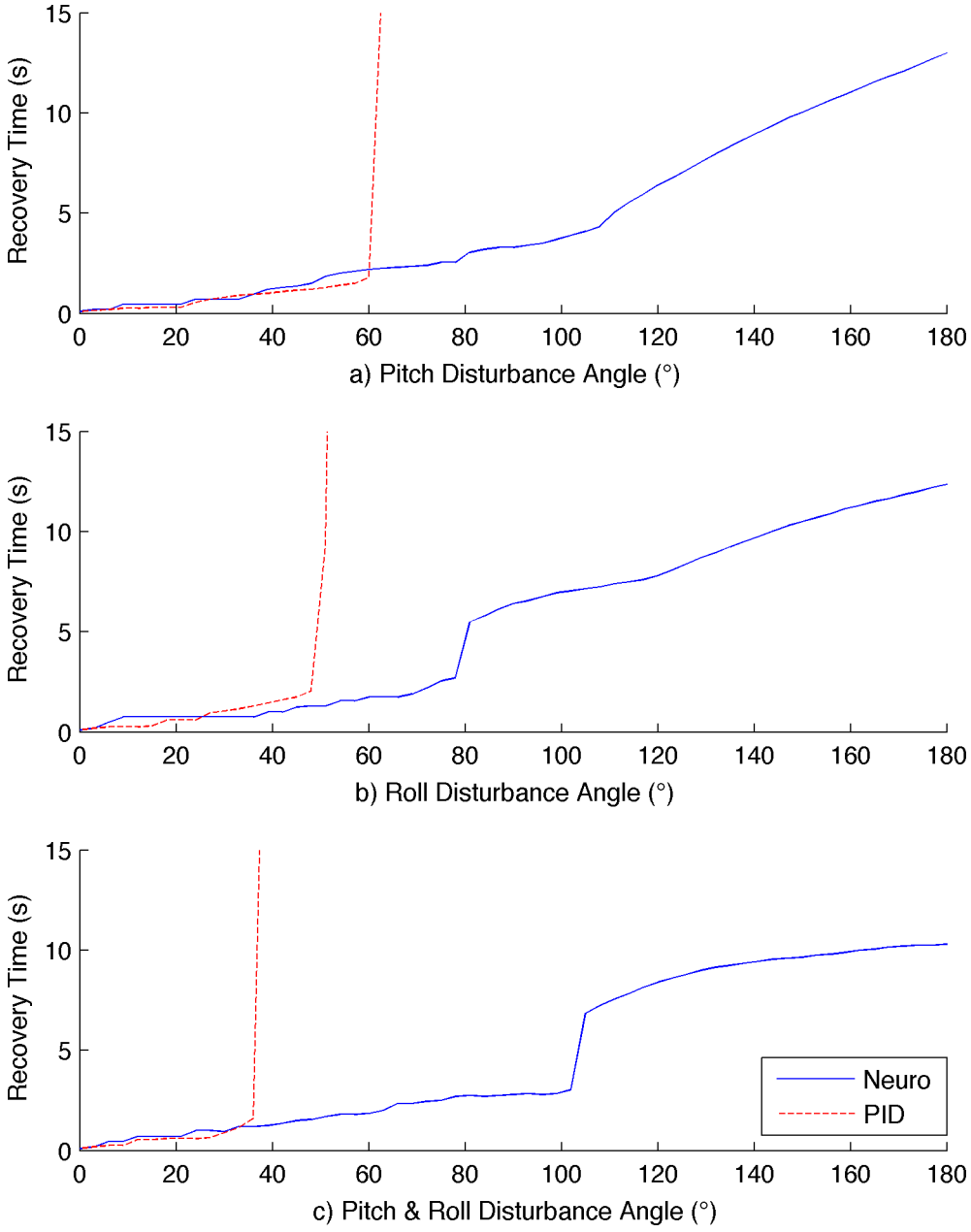

Figure 5.5: Time for controller recovery after a)pitch, b)roll, and c) pitch & roll angle disturbances of increasing amounts. The neuro-controller can even handle being flipped upside down, whereas the PID controller does not recover at all from disturbances greater than 60° or less in some cases.

controlling the rotor speeds. Our simulator was developed with this in mind, allowing for the addition of varying amounts of both types of noise. Beyond the two sources of noise, there are two types of noise, biased and random. Biased noise will affect the ability to know the actual position or attitude unless the controller goes through some additional training with the sensors that will actually be used in flight. Thus in our simulation testing, we focused on how well the designed controllers could handle random noise of increasing amounts. We expected that random variations in values would cancel out, and so as long as the noise does not swamp out the signal, stability should be maintained.

Our simulator does not model the dynamics of the motors. They were allowed to achieve great changes of speed in a single timestep. In noisy conditions this amplified the distance the quadrotor would travel in maintaing stability. The erratic response to noise may be filtered by the dynamics in actual motors when this controller is used on hardware.

5.3.1 Sensor Noise

The developed controller needs position, velocity, and attitude sensor data. Our testing showed that the final controller can withstand large random variations in all of these readings. The larger the noise level, the larger the sphere of space the quadrotor would move through in maintaining a stable hover. Figure 5.6 shows how the noise variation alters the stability of the craft. The neuro-controller was able to take about 45 % noise in the sensors while maintaining a stable hover within 0.5 m. The PID controller on the other hand was only able to handle about 10 % noise, and even this required a 2 m radius of space in which

to hover. The neuro-controller is able to handle nearly 5 times as much noise, and needs only 1/12 of the volume to do so.

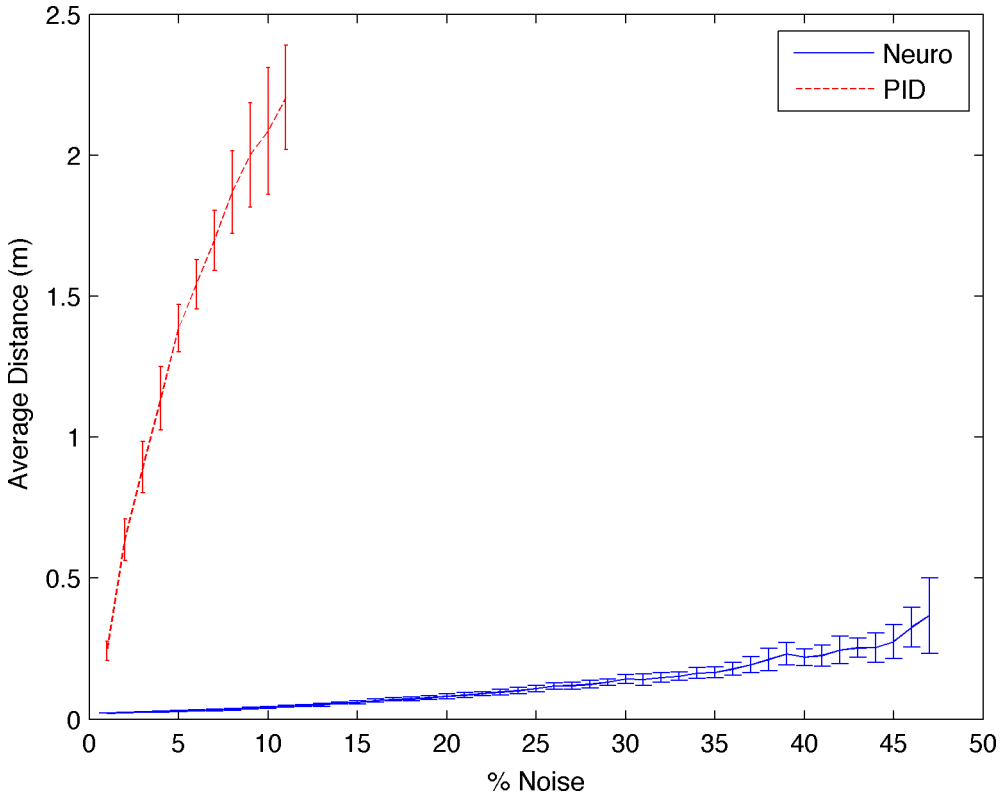

Figure 5.6: Plot of random sensor noise versus the average range of stable motion, showing the neuro-controller's ability to handle five time more noise. Noise values outside those plotted resulted in unstable flight.

5.3.2 Actuator Noise

Actuator noise can stem from various outside influences preventing the rotor from reaching the desired speed, or it could be the result of dynamics in the motor itself. In order to explore

the effects of this sort of noise, random noise was added to each of the actuators. Our testing showed that even 2 % actuator noise with a PID controller caused catastrophic failure. The trained neuro-controller was able to maintain stability with up to 15 % noise. With increasing noise, the quadrotor would move through an increasing sphere of space while attempting to maintain stability. The average distance from the origin used in maintaining a hover under increasing amounts of noise is shown in Figure 5.7.

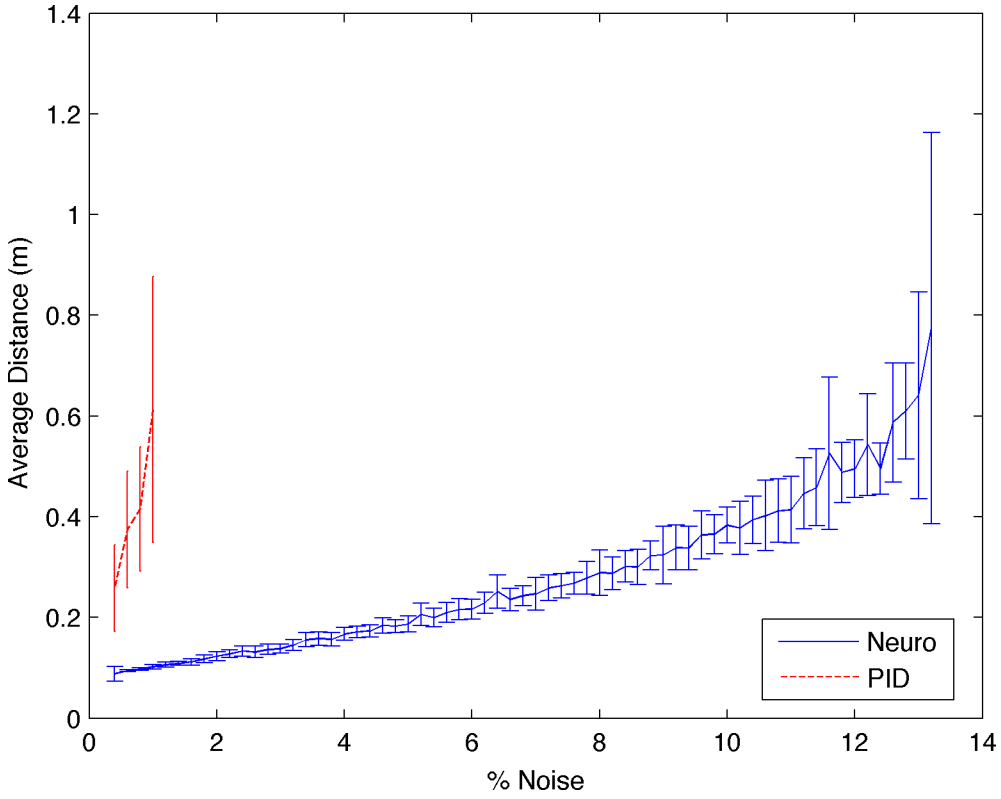

Figure 5.7: Plot of random actuator noise versus the average range of stable motion, showing the neuro-controller's ability to handle five time more noise. Noise values outside those plotted resulted in unstable flight.

5.4 Parameter Sensitivity

A significant challenge when moving between a simulated controller design and actual hardware is model inaccuracies. Often the controller is being tested before final hardware is assembled, meaning design changes can still occur affecting the parameters of the system and thus the simulated system no longer matches the hardware. Even if the hardware is available before controller design commences, difficulty in measuring some parameters (drag coefficients for example) can result in inaccuracies between the simulation and the hardware. A robust controller, is able to handle a range of parameter values allowing for stability even when the hardware and simulation do not match.

To test this aspect of robustness, the controller, after being trained with the design parameters was asked to perform a simple move on a simulated quadrotor with adjusted parameters. The time to complete the maneuver, while changing two key design parameters, the thrust coefficient and mass are shown in Figure 5.8. The neuro-controller is able to handle roughly twice the parameter change as the PID controller.

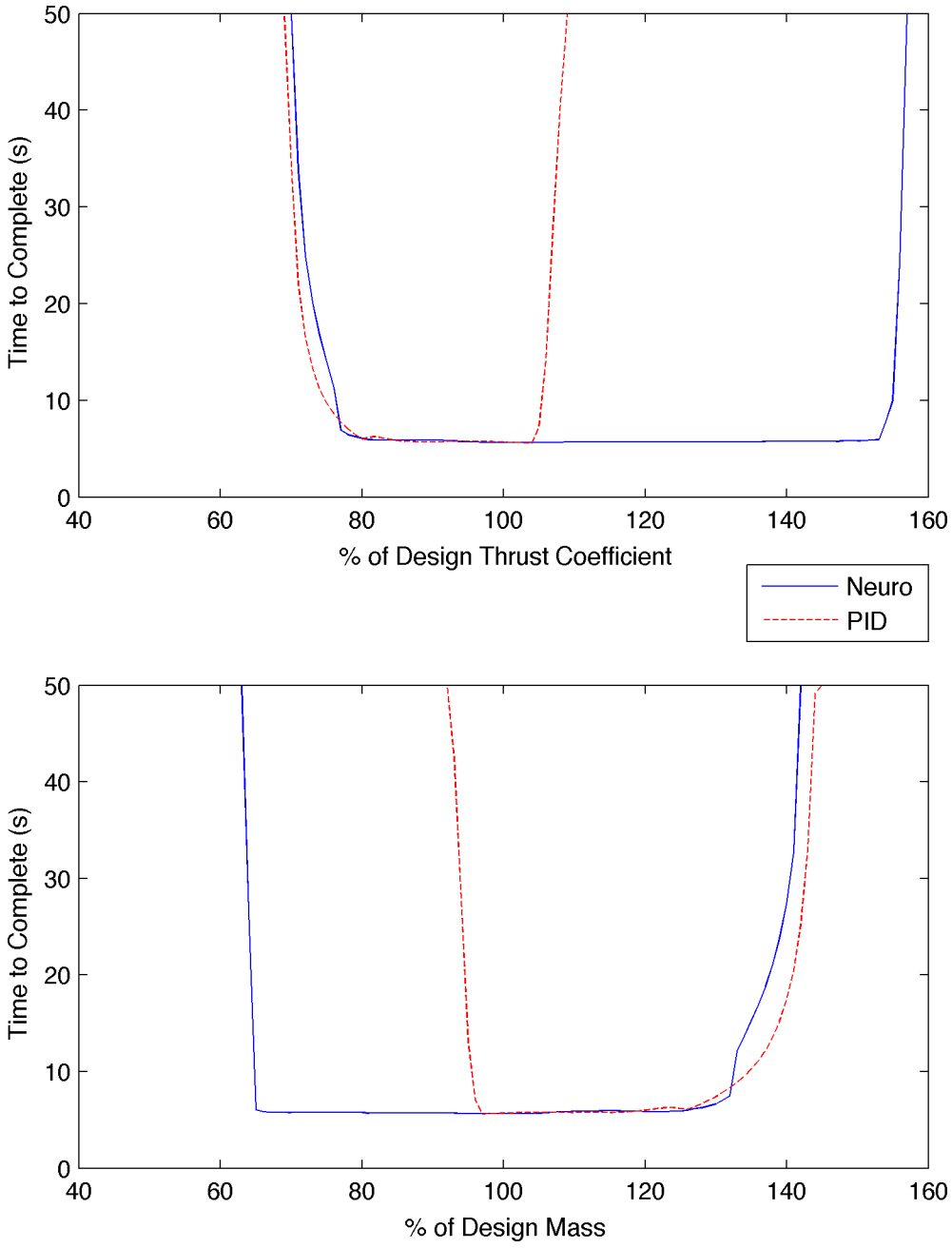

Figure 5.8: Plot showing the time to complete a simple move, illustrating the ability to handle changes in the design parameters. The neuro-controller is shown being able to handle roughly twice the variation in both mass and thrust coefficient.

Chapter 6 – Conclusions

Quadrotors are unique among MAVs in providing excellent maneuverability, allowing hover flight as opposed to the required forward motion for winged flight, while maintaing a simple mechanical construction without the need for control surfaces as on an airplane or variable-pitch propellers as on helicopters. This mechanical simplicity comes at a cost of increased controller complexity. Quadrotors are inherently unstable as they are highly sensitive to even the smallest differences in rotor speeds.

Prior work has developed model-based controllers that successfully control UAV size quadrotors operating near hover conditions, but such control becomes more difficult for small quadrotors. This work has demonstrated a consistent way to develop an adaptive controller for quadrotor craft. It has also highlighted the ability of this type of controller to withstand several shortcomings of model based controllers when the model does not match reality.

In this thesis, we explain our physical model and how this was derived. We also explain both the PID and neuro-controllers we used in the development of our final hierarchy of control. We present a hierarchical neuro-controller for small (0.5 kg) quadrotor control. The first stage of control aims to stabilize the craft and outputs rotor speeds based on a requested attitude (pitch, roll, yaw, and vertical velocity). This controller is developed in

four parts around each of the variables, initially training them to achieve results similar to a PID controller. The four parts are then combined such that the controller could be trained further to increase its robustness. The second stage of control is to achieve a requested (x, y, z) position by providing the first stage with the appropriate attitude.

The simulation results show that stable quadrotor control is achieved through this control architecture. In addition, the results show that the hierarchical control approach recovers from disturbances over an order of magnitude faster than a basic PID controller. It provides stable flight in the presence of 5 times more sensor noise and 8 times more actuator noise than the PID controller. Finally, although the controller was designed around a single set of design parameters, the robustness allows for significant variation in these parameters without retraining the controller.

Further work will be to validate this simulation work with implementation on actual hardware. Current work also includes expanding the high level position controller to not only track towards a desired position, but to accept sensory input to allow for obstacle avoidance. This could tie in with navigational algorithms already developed by Knudson et al. Another route of evaluation would be to compare these results to a LQR controller which has been shown by others to have some success at controlling quadrotor flight.

Bibliography

[1] C. Anderson. Learning to control an inverted pendulum using neural networks. *IEEE Control Systems Magazine*, 9(3):31–37, 1989.

[2] S. Bouabdallah. Autonomous systems lab. *http://www.asl.ethz.ch/education/master/aircraft/2008-L10-Homework.pdf*, 2008.

[3] S. Bouabdallah, P. Murrieri, and R. Siegwart. Design and control of an indoor micro quadrotor. *IEEE International Conference on Robotics and Automation*, 5:4393–4398, 2004.

[4] S. Bouabdallah, P. Murrieri, and R. Siegwart. Towards autonomous indoor micro VTOL. *Autonomous Robots*, 18(2):171–183, Jan 2005.

[5] S. Bouabdallah and R. Siegwart. Full control of a quadrotor. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 153–158, Jan 2007.

[6] A. Calise and R. Rysdyk. Nonlinear adaptive flight control using neural networks. *IEEE Control Systems Magazine*, 18(6):14–25, 1998.

[7] P. Castillo, R. Lozano, and A. E. Dzul. Advances in industrial control. *Modelling and Control of Mini-Flying Machines*, pages 39–59, Jan 2005.

[8] I. Cowling, O. Yakimenko, J. Whidborne, and A. Cooke. A prototype of an autonomous controller for a quadrotor UAV. *European Control Conference*, 2007.

[9] A. Das, F. Lewis, and K. Subbarao. Backstepping approach for controlling a quadrotor using lagrange form dynamics. *Journal of Intelligent and Robotic Systems*, 56:127–151, 2009.

[10] W. Davis, B. Kosicki, D. Boroson, and D. Kostishack. Micro air vehicles for optical surveillance. *Lincoln Laboratory Journal*, 9(2):197–214, 1996.

[11] T. Dierks and S. Jagannathan. Neural network output feedback control of a quadrotor UAV. *Proceedings of the 47th IEEE Conference on Decision and Control*, pages 3633–3639, 2008.

[12] J. Dunfied, M. Tarbouchi, and G. Labonte. Neural network based control of a four rotor helicopter. *2004 IEEE International Conference on Industrial Technology*, 3:1543–1548, Jan 2004.

[13] B. Erginer and E. Altug. Modeling and PD control of a quadrotor VTOL vehicle. *2007 IEEE Intelligent Vehicles Symposium*, pages 894–899, Jan 2007.

[14] J. Evans, G. Inalhan, J. Jang, R. Teo, and C. Tomlin. Dragonfly: A versatile UAV platform for the advancement of aircraft navigation and control. *20th Conference Digital Avionics Systems*, 1:14–18, 2001.

[15] D. Floreano and F. Mondada. Automatic creation of an autonomous agent: Genetic evolution of a neural-network driven robot. *Proceedings of the 3rd International Conference on Simulation of Adaptive Behavior*, 3:421–430, 1994.

[16] F. Gomez and R. Miikkulainen. 2-d pole balancing with recurrent evolutionary networks. *Proceeding of the International Conference on Artificial Neural Networks (ICANN)*, pages 758–763, 1998.

[17] G. Hoffmann, S. Waslander, and C. Tomlin. Quadrotor helicopter trajectory tracking control. *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, 2008.

[18] A. Jacoff, E. Messina, and J. Evans. A standard test course for urban search and rescue robots. *NIST SPECIAL PUBLICATION SP*, pages 253–259, 2001.

[19] M. Kadous, R. Sheh, and C. Sammut. Caster: a robot for urban search and rescue. *Proceedings of the 2005 Australasian Conference on Robotics and Automation*, 2005.

[20] M. Knudson and K. Tumer. Neuro-evolutionary navigation for resource-limited mobile robots. *Intelligent Engineering Systems through Artificial Neural Networks*, pages 27–34, 2008.

[21] T. Madani and A. Benallegue. Adaptive control via backstepping technique and neural networks of a quadrotor helicopter. *Proceedings of the 17th World Congress of The Internaional Federation of Automatic Control*, 2008.

[22] A. Ng, A. Coates, M. Diel, V. Ganapathi, J. Schulte, B. Tse, E. Berger, and E. Liang. Autonomous inverted helicopter flight via reinforcement learning. *Experimental Robotics IX*, 21:363–372, 2006.

[23] C. Nicol, C. Macnab, and A. Ramirez-Serrano. Robust neural network control of a quadrotor helicopter. *Candian Conference on Electrical and Computer Engineering*, Jan 2008.

[24] B. Pell, D. Bernard, S. Chien, E. Gat, N. Muscettola, P. Nayak, M. Wagner, and B. Williams. An autonomous spacecraft agent prototype. *Autonomous Robots*, 5(1):29–52, 1998.

[25] F. Ruini and A. Cangelosi. Distributed control in multi-agent systems: A preliminary model of autonomous MAV swarms. *Information Fusion, 2008 11th International Conference on*, pages 1–8, 2008.

[26] K. Stanley, B. Bryant, and R. Miikkulainen. Evolving neural network agents in the NERO video game. *Proceedings of the IEEE 2005 Symposium on Computational Intelligence and Games*, pages 182–189, 2005.

[27] K. Stanley and R. Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary Computation*, 10(2):99–127, 2002.

[28] M. Ventures. Stanley: The robot that won the darpa grand challenge. *Journal of Field Robotics*, 23(9):661–692, 2006.

[29] R. Wai. Tracking control based on neural network strategy for robot manipulator. *Neurocomputing*, 51(1):425–446, 2003.

[30] S. Waslander, G. Hoffmann, and J. Jang. Multi-agent quadrotor testbed control design: Integral sliding mode vs. reinforcement. *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3712–3717, Jan 2005.

[31] L. Young, E. Aiken, J. Johnson, R. Demblewski, J. Andrews, and J. Klem. New concepts and perspectives on micro-rotorcraft and small autonomous rotary-wing vehicles. *Proceedings of the 20th AIAA Applied Aerodynamics Conference*, 2002.