Robust Neuro-Control for A Micro Quadrotor

Jack F. Shepherd III
Oregon State University
Corvallis, OR, USA
shepheja@engr.orst.edu

Kagan Tumer
Oregon State University
Corvallis, OR, USA
kagan.tumer@oregonstate.edu

ABSTRACT

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, and micro quadrotors are particularly difficult to control. In this paper, we evolve a hierarchical neuro-controller for micro (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, and then combined and 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 results show that stable quadrotor control is achieved through this architecture. In addition, the results show that neuro-evolutionary control recovers from disturbances over an order of magnitude faster than a basic PID controller. Finally, the neuro-evolutionary 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.

Categories and Subject Descriptors

I.2.9 [**Artificial Intelligence**]: Robotics

General Terms

Algorithms; Experimentation

Keywords

Learning::Evolution, adaptation; Learning::Learning

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

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. The recent growth of Micro Aerial Vehicle (MAV) platforms is a testament to their strategic importance. Small, light, and versatile crafts will dominate the field of reconnaissance, be it military intelligence, or search and rescue [**?**]. 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 [**?**]. However, as controller hardware and software become more capable, rotorcraft are gaining strength [**?**]. Their ability to hover and perform vertical take-offs and landings, make them ideal for close-quarter information gathering missions. A quadrotor craft consists of 4 rotors as shown in Figure **??**. Instead of a variable pitch main rotor as in a helicopter, the craft is maneuvered by adjusting the relative speed between the individual motors. 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. Such control makes the craft inherently unstable, where any motor unbalance requires constant controller input to keep the craft aloft. However, it significantly reduces mechanical complexity by using fixed-pitch rotors and smaller motors. It also provides for greater safety with smaller blades and the ability to enclose them within the airframe.

Related Work:
Previous work on quadrotor controllers ranges from traditional model-based Proportional Integral Derivative (PID) controllers [**?**, **?**], to modern and optimal model-based techniques [**?**, **?**, **?**, **?**, **?**]. However, 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-linearities in the dynamics and unknown parameters, has also proven successful [**?**, **?**, **?**]. Similarly, using a neural network to act as an observer for a modern control solution may be desirable [**?**].

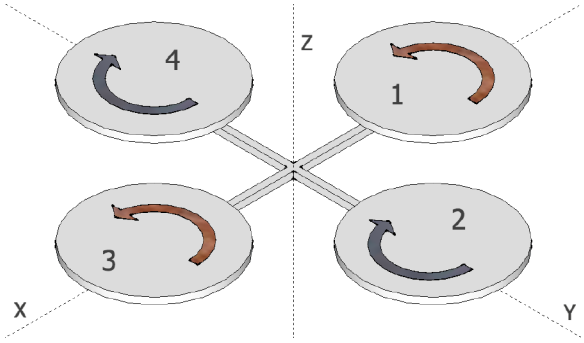

Figure 1: Quadrotor layout: *Roll* (rotation around the x axis) controlled by adjusting speeds of rotors 2 and 4 by opposite amounts. *Pitch* (rotation around the y axis) similarly adjusted using rotors 1 and 3. *Yaw* (rotation around the z axis) controlled by adjusting 2 and 4 together by an equal and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

Using both neural networks and reinforcement learning to train an entire controller from data points taken during human controlled flight has also been explored [?, ?]. 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. However, this sort of supervised reinforcement learning control has shown to be very effective, in the case of autonomous helicopter control [?, ?].

Model-free adaptive control has a proven success rate in many domains, from robotic manipulators [?], to single robotic navigation [?], to coordination of multiple autonomous vehicles [?, ?, ?]. Such control allows for design of a controller without the need for a fully developed model or pilot training and the associated stability routines. In this work, instead of a model of the world, a neural network is used to functionally approximate the necessary information for quadrotor flight. Neural networks create a non-linear mapping from inputs to outputs that we hypothesize can fully capture the quadrotor dynamics and create a robust controller.

Instead of using supervised learning and a recorded set of data points to determine the *correct* actions, as in the case of several previously noted works, we use a neuro-evolutionary algorithm to arrive at a trained controller. A neuro-evolutionary algorithm ranks each controller based on some cost function. Higher performing controllers are then mutated with some probability, in order to search for an optimal solution. This work uses the NeuroEvolution of Augmenting Topology (NEAT) algorithm [?]. This algorithm allows modifications in networks to include both changes to the weights as well as the topology. Our work uses the real-time NEAT algorithm to enable later implementation on real hardware for further training and tuning of controllers [?].

One shortcoming of many 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 [?]. A poor solution in an aerial vehicle means a crash and provides little useful information for learning.

Contributions:
The aim of this work has been to 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: unmodelled 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).

This paper explains the development and testing of a hierarchy of neuro-controllers for MAV quadrotor flight. Section 2 explains the model used to develop the simulator. Section 3 discusses the formulation of the controllers, as well as their training. Section 4 summarizes our key experimental discoveries with the final controllers. Section 5 provides concluding remarks and a reiteration of the contributions of this work.

2. MODELING & SIMULATION

The mathematical quadrotor model is based upon the previously developed model [?]. 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:

$$\begin{aligned} U_1 &= b\left(\Omega_1^2+\Omega_2^2+\Omega_3^2+\Omega_4^2\right) \\ U_2 &= b\left(\Omega_4^2-\Omega_2^2\right) \\ U_3 &= b\left(\Omega_3^2-\Omega_1^2\right) \\ U_4 &= d\left(\Omega_2^2+\Omega_4^2-\Omega_1^2-\Omega_3^2\right) \end{aligned} \tag{1}$$

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:

$$\Omega = \Omega_2 + \Omega_4 - \Omega_1 - \Omega_3 \tag{2}$$

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

$$ma = R\sum F_b \tag{3}$$

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,

$$I\alpha = -\omega \times I\omega - J_r(\omega \times \hat{z})\Omega + \tau_b \tag{4}$$

Table 1: 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

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:

$$\begin{aligned}
\ddot{x} &= (\cos\phi \sin\theta \cos\psi + sin\phi \sin\psi)\frac{1}{m}U_1 \\
\ddot{y} &= (\cos\phi \sin\theta \sin\psi - sin\phi \cos\psi)\frac{1}{m}U_1 \\
\ddot{z} &= -g + (\cos\phi \cos\theta)\frac{1}{m}U_1 \\
\ddot{\phi} &= \dot{\theta}\dot{\psi}\left(\frac{I_y - I_z}{I_x}\right) - \frac{J_r}{I_x}\dot{\theta}\Omega + \frac{l}{I_x}U_2 \\
\ddot{\theta} &= \dot{\phi}\dot{\psi}\left(\frac{I_z - I_x}{I_y}\right) + \frac{J_r}{I_y}\dot{\phi}\Omega + \frac{l}{I_y}U_3 \\
\ddot{\psi} &= \dot{\phi}\dot{\theta}\left(\frac{I_x - I_y}{I_z}\right) + \frac{1}{I_z}U_4
\end{aligned} \tag{5}$$

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 **??** [**?**].

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 selected quadrotor.

3. CONTROLLER FORMULATION

Flying a quadrotor, even in simulation, is a difficult problem because of the inherent sensitivity to differences in rotor speeds. 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 controllers that each make

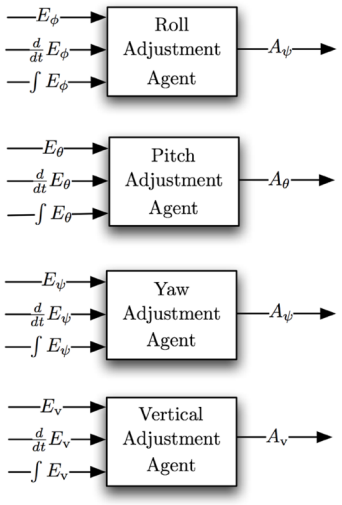

Figure 2: The structure of the attitude controllers independently generate adjustments to the rotor speeds to control the roll, pitch, and yaw angles as well as the vertical velocity. Each controller receives as input a difference from the desired set point as well as the difference's integral and derivative.

adjustments to the rotor speeds to independently control the four attitude variables.

3.1 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 trajectory, described by roll, pitch, and yaw angles, and a vertical velocity.

3.1.1 Attitude Controller Formulation

This controller receives information about the desired attitude as well as sensory information in the form of an error between the desired and actual, 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 outputs the desired rotor speeds for all four rotors.

Due to the quadrotor's attitude being very sensitive to even small variations in rotor speed, training the attitude controller directly proved challenging. Instead, it was broken down into the four controllers shown in Figure **??**, adjusting the roll, pitch, and yaw angles and vertical velocity independently using the appropriate three inputs. These four attitude parameters are combined as follows:

$$\begin{aligned}
\Omega_1 &= A_\mathrm{v} - A_\theta - A_\psi \\
\Omega_2 &= A_\mathrm{v} - A_\phi + A_\psi \\
\Omega_3 &= A_\mathrm{v} + A_\theta - A_\psi \\
\Omega_4 &= A_\mathrm{v} + A_\phi + A_\psi
\end{aligned} \tag{6}$$

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. Breaking down the attitude controller allowed each of the four neuro-controllers to be trained independently resulting in very quick learning of their respective effects on the craft.

When all four controllers were trained, they were com-

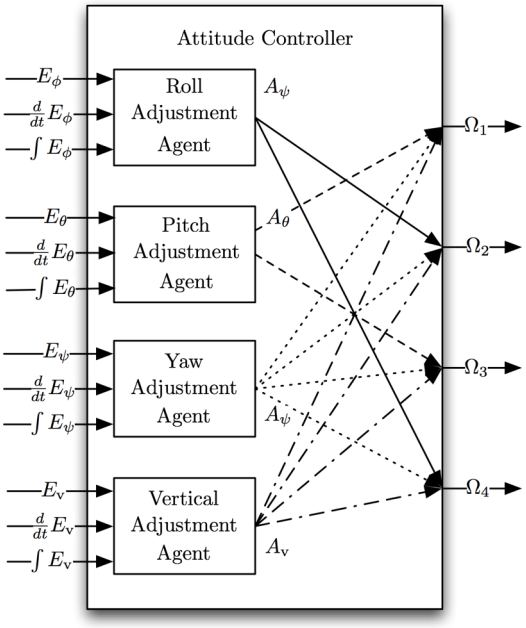

Figure 3: The inputs and outputs for each of the neuro-controllers are combined to form a single, higher level, attitude controller that outputs the rotor speeds based on the adjustments calculated by the four sub-controllers.

bined as shown in Figure **??** and stated in Equation **??**. This resulted in a single attitude controller that produces the desired rotor speeds from the original twelve inputs.

3.1.2 Attitude Controller Training

Each of the individual controllers described above and illustrated in Figure **??** (for roll, pitch, yaw, and vertical velocity) were trained separately. In order to start from a stable hover, A_v was first set to a level that just overcame gravity. A PID controller was developed for each of the three angles. This PID controller was tuned to provide a reasonably fast rise time without significant overshoot. The PID controller was used as a starting point for the training.

Instead of training against some stability objective function, the individual angle controllers were trained to match their respective PID controllers. Fitness during the neuro-evolutionary training was provided as:

$$F = e^{-|A_{nn} - A_{pid}|} \tag{7}$$

where A_{nn} is the adjustment estimated by 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.

During training, each of the attitude controllers was given a set of seven data points that covered the range of expected motion: $[-\frac{\pi}{2}, \frac{\pi}{2}]$ for the roll and pitch controllers and $[-\pi, \pi]$ for the yaw controller. The neuro-controller's ability to interpolate would allow it to successfully hit any angle in the given range and to hold the craft at that attitude. These data points were the same for all potential controller fitness evaluations. The fitness measured the controllers reaction compared to the PID action for 10 seconds. During the runtime, the neuro-controller, and not the PID controller was in control of the craft's next commanded rotor speed. This allowed the controller to learn to track the approximate shape of the PID curve in achieving a given set-point. At the end of 10 seconds, the simulation was reset and the next desired angle was presented.

There were 100 potential controllers in the population. Each started with zero hidden nodes and an otherwise fully connected network with random weights between -3 and 3. The NEAT algorithm allowed hidden nodes (3%) and additional links (5%) to be added with some probability. 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. Training was complete after 3000 episodes.

Learning the PID controllers used for fitness evaluation provided a stable starting point for further training. A new population was developed from the best controllers with up to 20% mutation of the weights. This new population was evaluated against the desired target 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.

Since previous work had difficulty in using a PID controller to adjust the altitude, the fourth controller responsible for vertical velocity 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.

Once all four controllers 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.

3.2 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.

3.2.1 Position Controller Formulation

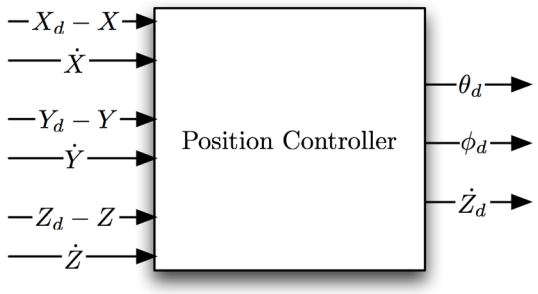

Figure 4: The inputs are the difference between the desired and current positions as well as the current speeds, and the outputs are the desired roll and pitch angles as well as the desired vertical velocity. The desired yaw angle is always set to zero, since full motion is achievable by setting roll, pitch and vertical velocity.

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. 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.

3.2.2 *Position Controller Training*

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 controllers. Training time however, took upwards of 15000 episodes to provide a controller with direct travel towards the destination.

4. EXPERIMENTAL RESULTS

Once the development of the entire hierarchy of controllers was complete, we performed several experiments to test their robustness. Our testing bore 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.

4.1 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 **??** shows the craft moving through a selection of waypoints as well as showing the optimal path. It is evident in Figure **??**a that on a very small scale this controller is not finding the optimal path. However, it is finding a path that arrives at the goal and maintains stability. When this same pattern is followed on a much greater distance scale, the path is much smoother, as seen in Figure **??**b.

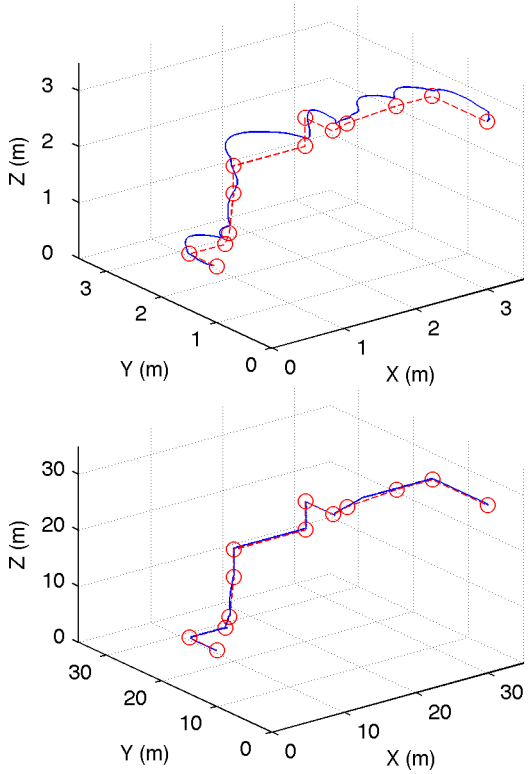

Figure 5: Final controller piloting through a series of waypoints. Straight line path is the dashed line, actual path is the solid line. a) Shows waypoints a few crafts lengths away. b) Shows the same pattern of waypoints where the distances have been scaled upwards an order of magnitude. The larger distances resulted in much smoother quadrotor flight paths, but both cases successfully reach the given waypoints.

4.2 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. The weight and small size requires slower controller hardware unable to process that a disturbance is happening, only to indicate that it has. Disturbance 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. The results were similar in all cases, but are shown in Figures **??** and **??**

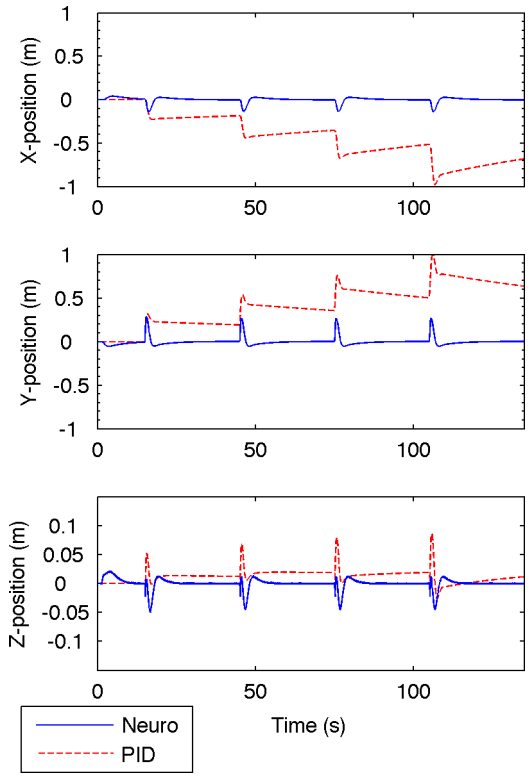

Figure 6: Position recovery for repeated pitch angle disturbances of 60° occurring every 30 seconds. The PID controller stabilizes the craft but would take several minutes to restore it to the starting location. The neuro-controller is able to stabilize and restore the position in less than 3 seconds.

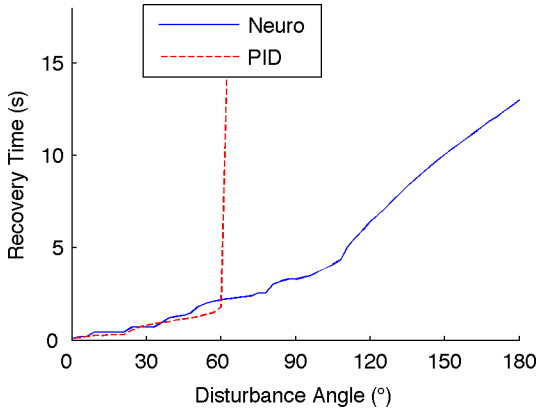

Figure 7: Time for controller recovery after pitch angle disturbances of increasing amounts. The neuro-controller can even handle being flipped upside down, whereas the PID controller does not recover at all from disturbances greater than 60°.

for the pitch axis case only. Figure **??** shows the 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. 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, but is not able to quickly restore the craft's position.

Figure **??** shows that a 60° disturbance is right at the extreme edge of the envelope the PID controller can handle. 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. As mentioned above, disturbances along the other axes perform similarly to a pure pitch disturbance.

4.3 Noise Tolerance

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 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 simulations, 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 time step. 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.

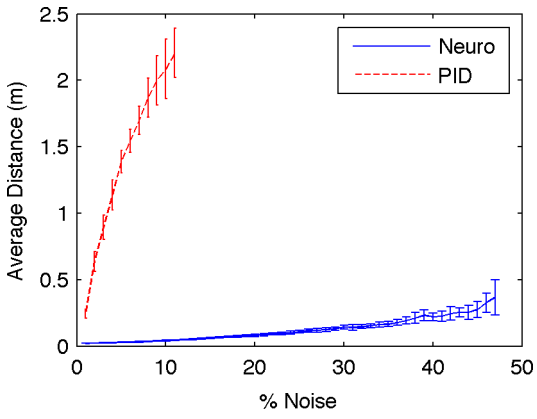

Figure 8: Plot of random sensor noise versus the average range of stable motion, showing the neuro-controller's ability to handle five time more noise. Noise values outside those plotted resulted in unstable flight.

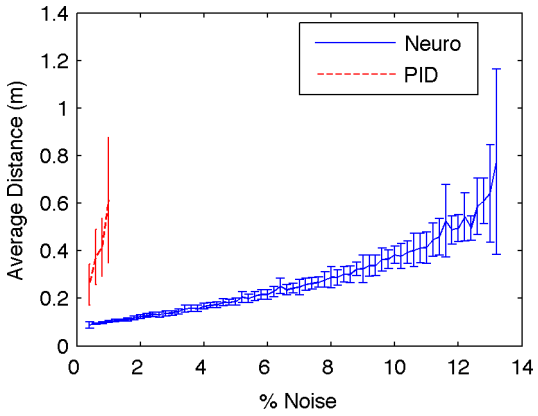

Figure 9: Plot of random actuator noise versus the average range of stable motion, showing the neuro-controller's ability to handle five time more noise. Noise values outside those plotted resulted in unstable flight.

4.3.1 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 **??** 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.

4.3.2 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 **??**.

4.4 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.

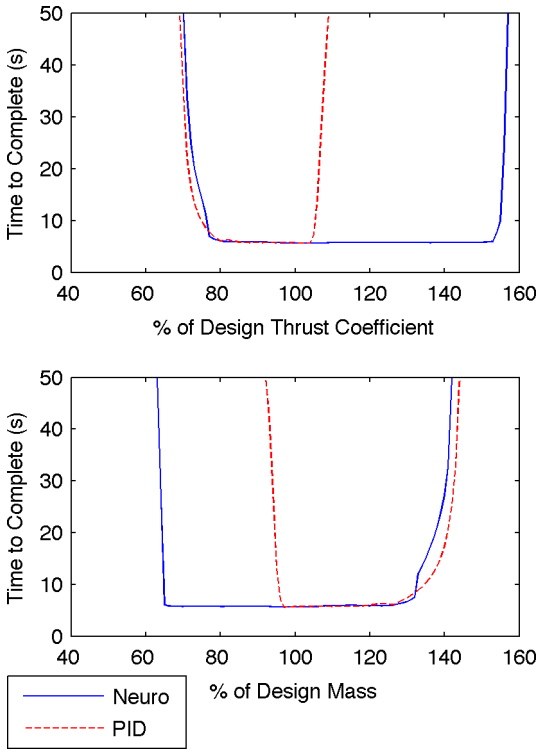

Figure 10: The time to complete a simple move illustrates the ability to handle changes in the design parameters. The neuro-controller is shown being able to handle roughly twice the variation in both mass and thrust coefficient.

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 **??**. The neuro-controller is able to handle roughly twice the parameter change as the PID controller.

5. 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 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 with-

stand several shortcomings of model based controllers when the model does not match reality.

In this paper, 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.

6. ACKNOWLEDGMENTS

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

7. REFERENCES

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

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

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

[7] 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.

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

[9] 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.

[10] 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.

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

[12] 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.

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

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] M. Salichon and K. Tumer. A neuro-evolutionary approach to micro aerial vehicle control. In *The Genetic and Evolutionary Computation Conference*, Portland, OR, July 2010.

[19] 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.

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

[21] 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.

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

[23] 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.

[24] 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.