Path Tracking Control for Quadrotor Helicopters

Katie Miller
Graduate Student Mentor: Maryam Kamgarpour
Faculty Advisor: Professor Claire Tomlin

July 31, 2008

Abstract

This work sought to design control laws for a quadrotor helicopter to track a desired path. The control laws were developed on a model of the quadrotor dynamics, which was obtained by linearizing its nonlinear equations of motion. After, the effectiveness of the control laws were studied by including modeling uncertainties and disturbances such as wind. The results enabled us to understand whether there is a need for more complex control laws to address the unmodeled dynamics.

1 Introduction

Quadrotor helicopters are a popular device used in the research of unmanned aerial vehicles (UAV) because of the ease with which they can be designed and maintained. The increased interest is also due to the commercially available remote control quadrotors, which are easy to modify for autonomous flight and have recently been developed. A quadrotor consists of four rotors, one on each of its corners; Figure 1 shows a quadrotor in flight. There are numerous predicted uses for these autonomous vehicles either individually or working as a team, including: surveillance, search and rescue, and mobile sensor networks [2].

Figure 1: A quadrotor helicopter in flight, picture taken from [3].

This paper continues with a brief description of the quadrotor’s equations of motion in the following section. Sections III and IV describe the controllers we developed to control the three positions and two angles of the quadrotor respectively. Section V discusses the integration of these two controllers. A detailed analysis of how disturbances affect the performance of our controllers and the resulting control laws is including in Section VI.
2 Quadrotor Dynamics

For this research we used the linearized equations of motion developed in [3]. This model is shown below in its matrix form.

\[
\begin{bmatrix}
\ddot{r}_x \\
\ddot{r}_y \\
\ddot{r}_z
\end{bmatrix} =
\begin{bmatrix}
0 & \bar{T} & 0 \\
0 & 0 & \dot{T} \\
mg & 0 & 1
\end{bmatrix}
\begin{bmatrix}
\phi \\
\theta \\
T
\end{bmatrix}
\]

Let \( r_x, r_y, \) and \( r_z \) be the distance in each respective axis from the quadrotor’s center of mass to a fixed reference frame. Also, \( T = \sum_{i=1}^{4} T_i \) where \( T_i \) is the thrust for rotor \( i \) and \( \bar{T} = mg \). The angular equations of motion are shown in the next set of matrices. The angles \( \phi, \theta \), and \( \psi \) are the Eulerian angles measured from a body-fixed reference frame. This model is dependent on the small angle approximation, so the angles need to be approximately between \( \pm 30 \) degrees for it to be valid.

\[
\begin{bmatrix}
I_x \ddot{\phi} \\
I_y \ddot{\theta} \\
I_z \ddot{\psi}
\end{bmatrix} =
\begin{bmatrix}
0 & l & 0 & -l \\
l & 0 & -l & 0 \\
K_r & -K_r & K_r & -K_r
\end{bmatrix}
\begin{bmatrix}
T_1 \\
T_2 \\
T_3 \\
T_4
\end{bmatrix}
\]

These models have numerous variables, many of which have specific values that were adhered to throughout the simulations. Table 1 lists the variables, gives a description of each, and includes the value we used throughout our experiments.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>( g )</td>
<td>Gravity</td>
<td>9.81</td>
</tr>
<tr>
<td>( I_x )</td>
<td>X axis moment of inertia</td>
<td>.02</td>
</tr>
<tr>
<td>( I_y )</td>
<td>Y axis moment of inertia</td>
<td>.02</td>
</tr>
<tr>
<td>( I_z )</td>
<td>Z axis moment of inertia</td>
<td>.047</td>
</tr>
<tr>
<td>( K_r )</td>
<td>Proportional term between thrust and torque</td>
<td>1</td>
</tr>
<tr>
<td>( l )</td>
<td>Moment arm length</td>
<td>1</td>
</tr>
<tr>
<td>( m )</td>
<td>Mass</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 1: Quadrotor model variables

Equation 1 consists of three independent second order differential equations, one for each axis. Likewise, Equation 2 is composed of three independent second order differential equations, one for each angle. This gives us a 12-state system characterizing the quadrotor’s equations of motion, which we modeled in MATLAB to understand the dynamics. After, we manipulated the control factors to allow us to control the path of the quadrotor helicopter. To do this, we implemented Proportional Derivative (PD) controllers for the positions \( x, y, \) and \( z \) and Proportional Integral Derivative (PID) controllers for the angles \( \phi \) and \( \theta \) independently of each other. A detailed description of the process follows along with the results in the following sections.

3 Position Control Design

To control the position of the quadrotor, we developed a PD controller for each axis independently. We then assumed that the angles in Equation 1 were the control inputs and could be directly influenced. This enabled us to decouple the position with its respective angle.

3.1 Z Controller

To start, we wanted the quadrotor to hover at a certain height, in this case, two meters. Other design characteristics included a fast response time and a small overshoot. Response time is how fast the system reaches its steady state, and overshoot is the maximum distance above the steady state the system reaches.
As can be seen, $z$ is not directly influenced by an angle. Instead, it relies on gravity and the sum of the thrust forces. From Equation 1, the equation of motion for $\ddot{r}_z$ is represented as:

$$\ddot{r}_z = g + \frac{T}{m}$$ (3)

If $z_d$ is the desired height, then error can be defined as:

$$e = z_d - r_z$$ (4)

$$\dot{e} = \dot{z}_d - \dot{r}_z$$ (5)

We then developed a PD controller based on the error and set the control input as:

$$\frac{T}{m} = -K_D\dot{e} - K_P e - g$$ (6)

with $-g$ counteracting the gravitational force. This causes the system to be represented by the following equation:

$$\ddot{r}_z = -K_D\dot{e} - K_P e$$ (7)

Below is the Simulink model describing this system.

![Simulink model of the Z PD controller model.](image)

After designing our controller, we realized that it was not realistic because the numerator was to a higher order than the denominator. To fix this, we added a fast pole, $(\tau s + 1)$, to the denominator. This ensures that the order of the denominator is at least as high as the numerator. To prevent adverse effects to our controller, we choose a small value for $\tau$, .01, because this prevents it from being close to the other eigenvalues and disrupting the dynamics of the system. The simulation of the hover trajectory is shown in Figure 3, which met our design criteria with no overshoot and a response time of approximately .7 seconds.
3.2 X and Y Controllers

The quadrotor dynamics in the x and y directions are similar as seen in Equation 1; and, therefore, the same controller can be used for both. We will refer only to the y controller. Our goal when designing the PD controller was to move from an initial condition of zero to one. For this controller response time was not a priority, instead the design parameters included an overshoot of less than 10 percent and keeping the angle, $\phi$, within its linear operating range of $\pm 30$ degrees. Motion in the y direction is characterized by the equation:

$$\ddot{r}_y = g\phi$$

(8)

We set $g\phi$ as the control input; similarly, the control input for x is $g\theta$. If the desired y position is $y_d$, then the error dynamics can be characterized by the following two equations:

$$e = y_d - r_y$$

(9)

$$\dot{e} = \dot{y}_d - \dot{r}_y$$

(10)

We can then create the PID controller

$$u = K_D\dot{e} + K_pe$$

(11)

Figure 4 is a Simulink model of the controlled system with the PD controller.

![Figure 4: Simulink model of independent PD controller for y position.](image)
The parameters of the controller were found by experimenting with the values for the two variables, $K_D$ and $K_P$, which are the derivative and proportional gains respectively. To ensure stability of the system, all the eigenvalues need to be in the left half of the plane. At first one of the design principles was to have a system with all of the closed loop eigenvalues less than negative one. However, from experiments, this seemed to conflict with the other design criteria. As the derivative gain is increased, it causes the value of the eigenvalue closest to the imaginary axis to increase, but decreases the value of the overshoot. When the proportional gain increases in value, it decreases the value of the largest eigenvalue, but amplifies the overshoot. In order to have little or no overshoot, the proportional term needs to be very small and the derivative term significantly larger. This causes the largest eigenvalue to be closer to -.2. Figure 5 shows the position of $y$ using the PD controller with values of 2.5 and .5 for the terms $K_D$ and $K_P$ respectively.

![Figure 5: Position of y using PD controller.](image)

### 3.3 Waypoint tracking

We developed the controllers for each axis independently, making it easy to put them together to track a desired path. For example, we used the PD controllers for the x and y positions to produce a square waypoint tracking system shown in Figure 6. The quadrotor moves in each direction separately, so a different controller is used on each side of the square. We assumed the height would stay the same in the z axis keeping this model two dimensional.
Figure 6: Square waypoint tracking trajectory.

4 Angular Control Design

As can be seen from Equation 1, the angle directly influences the position. In order for the position to follow a path, the angle needs to be able to quickly and accurately track the desired value. Therefore, when designing the controller for the angle, the priority was a fast response with an overshoot of less than 10 percent. To ensure a better response, we chose a PID controller for this term. Figure 7 shows the structure of the system we represented.

![Simulink model of the independent PID controller for phi.](image)

Each term in a PID controller, $K_P$, $K_I$, and $K_D$, has a specific influence on the response of the system. The table below describes the effects we saw when experimenting with these values. The column “Effect on eigenvalues” refers to the way the term changes the closed loop eigenvalue closest to the imaginary axis. The placement of this eigenvalue is important because the farther away it is from the imaginary axis, the more stable the system is.
<table>
<thead>
<tr>
<th>Term</th>
<th>Increases</th>
<th>Decreases</th>
<th>Effect on eigenvalues</th>
</tr>
</thead>
<tbody>
<tr>
<td>$K_P$</td>
<td>Overshoot</td>
<td>Response Time</td>
<td>Decreases</td>
</tr>
<tr>
<td>$K_I$</td>
<td>Overshoot</td>
<td>Steady State Error</td>
<td>Increases</td>
</tr>
<tr>
<td>$K_D$</td>
<td>Response Time</td>
<td>Overshoot</td>
<td>Increases</td>
</tr>
</tbody>
</table>

Table 2: PID term functions

Using our knowledge of the parameters’ functions, we experimented with values until the system met our design criteria. The eigenvalue closest to the imaginary axis was approximately -.4, so the system is more stable than the PD controller described in the previous section. Below is the response of $\phi$ using the PID controller we developed.

![Figure 8: Simulation of phi using a PID controller.](image)

5 Integrating Angular Controller with Position Controller

After developing the controllers separately for the position and angle, we put them together in one model according to Equations 1 and 2, which is shown in Figure 9. As can be seen, it creates a nested loop with the angle controller in the middle. The position PD controller outputs a desired angle, and then the angle's PID controller tries to match this desired number using the controller developed in Section IV. The actual value outputted from the $\phi$ PID controller is then used in the position plant.

![Figure 9: Model of the controller used for the integrated system.](image)
Once again, it was necessary to add a fast pole to make our control system more realistic because the order of the numerator was higher than the denominator. We then input the parameters we had found using the independent controllers. This system is shown in the Simulink representation in Figure 10.

![Simulink representation of control system](image)

Figure 10: Two independent controllers integrated for complete model.

The response of the original system when the two controllers were put together was not the best possible response result; however, after further tuning, we were able to get a better response shown in Figure 11. When comparing this response to Figure 5, it is easy to see that there is a bigger overshoot for this model. This is expected because the angle for Figure 5 was part of the control input and, therefore, an exact value. When the two controllers are put together, the system is more realistic.

![Position of y using two controllers together with further tuning](image)

Figure 11: Position of y using two controllers together with further tuning.

6 Uncertainty in Linear Model

Our model is linear and does not take into account any disturbances or uncertainties. To test the robustness of the model and to discover if it would produce good results in more complex situations that involve aerodynamic effects, such as air speed variation described in [2], we added uncertainty and disturbances.

6.1 Wind Disturbance

We started with wind disturbance. To do this, we added a random Gaussian distributed number to the velocity in the y-direction to see how it affected our simulation. With small disturbances of mean ≤ 1 and a small variance of ≤ .5, the response of the system was very similar to the original. The steady state value
remained the same, but oscillation due to wind was introduced. The figure below shows a plot of the position of y with an added wind disturbance of mean = 1 and variance = .25.

![Position of y with Wind Disturbance](image)

**Figure 12**: Simulink simulation of y position with small wind disturbance.

When adding more disturbance with a larger variance or mean, the system still goes to its steady state; however, it introduces a larger overshoot and more oscillation. The larger the mean, the larger the overshoot is. Below is a simulation with a mean of 5 and variance of 1. As can be seen, the overshoot is much larger than in the previous figure.

![Position of y with Wind Disturbance](image)

**Figure 13**: Simulink simulation of large wind disturbance for y position.

### 6.2 Saturation

The thrust of each of the four motors can be up to ±8N with a total possible thrust of ±32N. The dynamics in the x and y directions, as shown in Equation 1, are dependent on the sum of two rotors each, therefore,
the total possible thrust in each direction is ±16N. This is not taken into account in the earlier plots. We added this constraint by inserting a saturation block into our Simulink model, as shown in Figure 14.

After adding this constraint, we simulated the model again in Figure 15, using Simulink. As you can see, the plot oscillates much more when the saturation is taken into account because the values used in the earlier plot were not all possible with the thrust constraints. Also, including saturation prevents the angle from staying within its linear operating range.

If both saturation and wind disturbance are included in the same model, the system does not perform well. This is because the wind causes the desired angle to oscillate. Our linear system is not able to track an oscillatory angle, and the thrust becomes very large since the input is constantly changing. This causes our system to track the desired position poorly as shown in the simulation, Figure 16.

### 6.3 Additive Uncertainty

We further tested the robustness of our system by including additive uncertainty. This problem could occur, for example, when there is a slight error in the machinery causing a rotor’s thrust to not fully counteract another. This would create a small disturbance that needs to be added to the equation. We started out by testing this on each of the independent controllers separately. This uncertainty could be represented in equation form as:

\[ I_y \phi = l(T_1 - T_3) + \Delta T \text{ where } |\Delta T| \leq .1. \]  \hspace{1cm} (12)
The additive uncertainty introduced a steady state error of approximately .2 in the position PD controller. This simulation is shown in Figure 17.

Next, we included it in the PID controller for $\phi$. The equation for this controller could be represented as:

$$\ddot{r}_y = g\dot{\phi} + \Delta d \quad (13)$$

The system is shown in the Figure 18 with $\Delta d = .1$. The uncertainty introduced oscillation into the step response.
Figure 18: Phi PID controller with additive uncertainty of .1.

When we included additive uncertainty on the entire system by adding a constant .1 to both channels, it combined the effects of the independent controllers as shown in Figure 19. The uncertainty introduced oscillation and a steady state error of .2.

Figure 19: Position of y using the integrated controllers with additive uncertainty of .1.

7 Conclusion

In this paper we have used the quadrotor’s linearized equations of motion to develop control laws for waypoint tracking paths. We have analyzed how these laws perform with uncertainties such as wind and saturation. These results show that while the linear control laws are adequate under perfect conditions, when uncertainty is introduced, they do not perform well. In future work, it would be interesting to use the developed linear
control laws on the more accurate non-linear representation of the system developed in [3]. This would provide further information as to whether these control laws are effective for the system.

References

