American Journal of Applied Sciences

Original Research Paper

Feedback Control of Quad-Rotors with a Matlab-Based Simulator

[1]Stephen Armah, [1]Sun Yi, [2]Wonchang Choi and [2]Dongchul Shin

[1]Department of Mechanical Engineering, North Carolina A and T State University, Greensboro, USA
[2]Department of Architectural Engineering, Gachon University, Seongnamsi, South Korea

Article history
Received: 02-11-2015
Revised: 21-06-2016
Accepted: 21-06-2016

Corresponding Author:
Wonchang Choi
Department of Architectural Engineering, Gachon University, Seongnamsi, South Korea
Email: wchoi@gachon.ac.kr

Abstract: Precise regulation of position and attitude (roll, pitch and yaw) is one of the most critical tasks to be considered in designing controllers for quadrotor types of Unmanned Aerial Vehicles (UAVs). The MATLAB-based simulator for quadrotors using PD-feedback control to achieve stabilization was developed. MATLAB/Simulink models are used to design and validate the control algorithms and the simulation results are presented. The quadrotor non-linear dynamics and kinematics were implemented using MATLAB S-function to generate a continuous state output. The simulation results are presented graphically and quantitatively. The developed MATLAB GUI interface is robust and interactive to implement the different control algorithms under various operating conditions.

Keywords: Quadrotor, UAVs, PD-Feedback Control, GUI, Euler Angles, MATLAB S-Function

Introduction

Mobile robot control has been the focus of active research in the past decades. During the last decade, with the advance in relevant technology, the demand of flying mobile robots or Unmanned Aerial Vehicles (UAVs) has rapidly increased. UAVs emergence also has to do with the simplicity of their construction and maintenance, their ability to hover and their Vertical Take-Off and Landing (VTOL) capability Armah (2015).

Rotorcraft UAVs have a variety of configurations that include conventional helicopter with a main and tail rotor, a coax with counter-rotating coaxial rotors and quad-rotors Corke (2011). Quad-rotor types of UAVs has four rotors that are controlled independently. The movement of the quadrotor results from changes in the speed of each rotor. The structure of quadrotor in this research paper is assumed to be rigid and symmetrical, the center of gravity and the body fixed frame origin are coincided, the propellers are rigid and the thrust and drag forces are proportional to the square of propeller's speed. It is also assumed that the earth is flat and non-rotating, which is a valid assumption for quad-rotors.

This research presents application of PD-feedback control to achieve stabilization for roll, pitch, yaw, altitude and position motions. MATLAB/Simulink models are used to design and validate the control algorithms and the simulation results are presented. The PD-gains in this research were obtained by tweaking the various values to obtain satisfactory responses. The quadrotor's non-linear dynamics and kinematics were implemented using MATLAB *S-function* to generate a continuous state output. Robust and interactive MATLAB GUI interface developed to implement the different control algorithms simulation models is also presented.

Quadrotor's Dynamics and Kinematics

Figure 1 shows the notation for the quadrotor model Hassan *et al*. (2013) and http://scholarsarchive.byu.edu/cgi/viewcontent.cgi?article=2324&context=facpub (2015). Different coordinate frames are used in identifying quadrotor's location and attitude and translation and rotation matrices can then be applied to transform one coordinate reference frame into another. Quadrotors has three main coordinate systems attached to it; the body-fixed frame, $\{B\}$, the vehicle frame, $\{V\}$ and the global inertial frame, $\{I\}$. There are two other coordinate systems, not shown, that are of interest, the vehicle-1 frame, $\{V^1\}$ and vehicle-2 frame, $\{V^2\}$. Frame $\{V^1\}$ is obtained by rotating frame $\{V\}$ about the Z_V-axis by a positive yaw angle, φ_V, assuming no rolling or pitching, so that X_V and Y_V are aligned with X_B and Y_B respectively. Frame $\{V^2\}$ is also obtained by rotating frame $\{V^1\}$ in a right-handed rotation about the $Y_V{}^1$-axis by the pitch angle, $\theta_V{}^1$, assuming no rolling, so that $X_V{}^1$ and $Y_V{}^1$ are aligned with X_B and Y_B respectively.

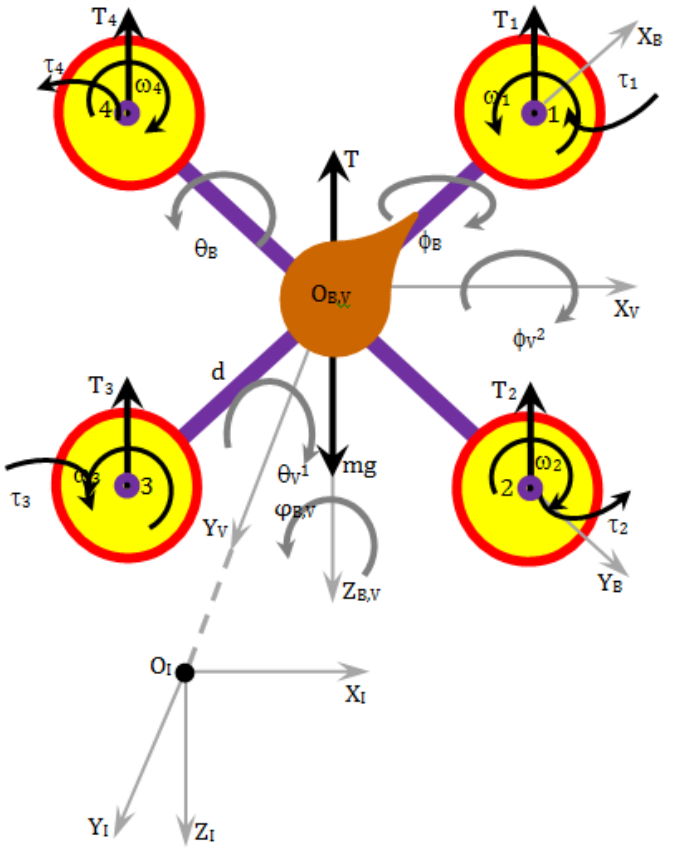

Fig. 1. Coordinates for the quadrotor

The configuration of the quadrotor is represented by its six degrees of freedom in terms of position, $(x_I, y_I, z_I)^T$ and the attitude defined using the Euler angles, $(\varnothing_{V^2}, \theta_{V^1}, \varphi_V)^T$. This gives a 12-state system characterizing the quadrotor's equations of motion, as $x = \left(x_I, y_I, z_I, \varnothing_{V^2}, \theta_{V^1}, \varphi_V, \dot{x}_B, \dot{y}_B, \dot{z}_B, \dot{\varnothing}_B, \dot{\theta}_B, \dot{\varphi}_B\right)^T$.
http://scholarsarchive.byu.edu/cgi/viewcontent.cgi?article=2324&context=facpub (2015).

The quadrotor has four rotors, labeled 1 to 4, mounted at the end of each cross arm. The rotors are driven by electric motors powered by electronic speed controllers. The vehicle's total mass is m and d is distance from the motor to the center of mass.

The forces and the moments on the quadrotor is primarily due to gravity and the four propellers and since there are no aerodynamic lifting surfaces, it will be assume that the aerodynamic forces and moments are negligible. The total upward thrust, T, on the vehicle is given by:

$$T = \sum_{i=4}^{i=1} T_i \tag{1}$$

Where:
ω_i = The relationship between the rotor speed

The upward thrust T_i, is defined as Corke (2011):

$$T_i = a\omega_i^2, i = 1,2,3,4 \tag{2}$$

where, $\alpha>0$ is the lift constant that depends on the air density, the cube of the blade radius, the number of blades and the chord length of the blade Corke (2011).

The translational dynamics of the vehicle in the world coordinates is given by Newton's second law:

$$m\frac{dv_B}{dt_I} = F_B \tag{3}$$

Where:
$v_B = (\dot{x}_B, \dot{y}_B, \dot{z}_B)^T$ = The quadrotor's linear velocity
$F_B = \left(f_{x_B}, f_{y_B}, f_{z_B}\right)^T$ = The total force applied to the quadrotor
d/dt = The time derivative in frame $\{I\}$

From equation of Coriolis, Equation 3 becomes:

$$m\frac{dv_B}{dt_I} = m\left(\frac{dv_B}{dt_B} + \omega_{B/I} \mathrm{x} v_B\right) = F_B \tag{4}$$

where, $\omega_{B/I} = \omega_B = \left(\dot{\varnothing}_B, \dot{\theta}_B, \dot{\varphi}_B\right)^T$ is the angular velocity of the vehicle in frame $\{B\}$.

Now, F_B is made up of the gravity force, F_{g_B} and the total thrust from the motors, F_{T_B}, given as:

$$F_B = F_{g_B} + F_{T_B} = {}^B R_V \begin{pmatrix} 0 \\ 0 \\ mg \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ -T \end{pmatrix} \tag{5}$$

Where:

g = The gravitational acceleration

${}^B R_V$ = The rotation matrix from frame $\{V\}$ to frame $\{B\}$ given by Equation 19

Substituting Equation 5 into Equation 4 and rearranging, to obtain:

$$\dot{v}_B = -\omega_B \mathrm{x} v_B + {}^B R_V \begin{pmatrix} 0 \\ 0 \\ g \end{pmatrix} - \begin{pmatrix} 0 \\ 0 \\ T/m \end{pmatrix} \tag{6}$$

The rotational dynamics of the airframe in frame $\{I\}$ is given by Euler's equation of motion http://scholarsarchive.byu.edu/cgi/viewcontent.cgi?article=2324&context=facpub (2015):

$$\frac{dh_B}{dt_I} = \Gamma_B \tag{7}$$

Where:

$h_B = J\omega_B$ = The quadrotor's angular momentum

$\Gamma_B = \left(\tau_{x_B}, \tau_{y_B}, \tau_{z_B}\right)^T$ = The applied torque to the airframe

Similarly, using the equation of Coriolis and rearranging, Equation 7 becomes:

$$\Gamma_B = J.\left(\dot{\omega}_B + \omega_B \mathrm{x}\, \omega_B\right) \tag{8}$$

where, J is the constant rotational inertia matrix of the vehicle given by Guenard *et al*. (2008):

$$J = \begin{pmatrix} J_{xx} & -J_{yx} & -J_{zx} \\ -J_{xy} & J_{yy} & -J_{zy} \\ -J_{xz} & -J_{yz} & J_{zz} \end{pmatrix} \tag{9}$$

And if the airframe's mass distribution is assumed to be symmetrical with respect to frame $\{B\}$, then $J_{xy} = J_{xz} = J_{yz} = 0$ and $J_{xx} = J_{yy}$. The moments of inertia are calculated as:

$$\begin{aligned} J_{xx} = J_{yy} &= \frac{2m_e r^2}{5} + 2d^2 m_c \\ J_{zz} &= \frac{2m_e r^2}{5} + 4d^2 m_c \end{aligned} \tag{10}$$

Where:

m_c = Mass of the quadrotor's center (assuming a spherical dense center, with radius r)

m_e = The mass at the end of each cross arm, where the propellers are located

The pair wise differences in rotor thrust cause the vehicle to rotate. The torque about frame $\{B\}$ x-axis, the rolling (positive) torque, is given by:

$$\tau_{x_B} = \tau_{\varnothing_B} = d\left(T_4 - T_2\right) \tag{11}$$

Substituting Equation 2 into Equation 11, to obtain:

$$\tau_{\varnothing_B} = ad\left(\omega_4^2 - \omega_2^2\right) \tag{12}$$

And similarly for the y-axis, the pitching (positive) torque is:

$$\tau_{y_B} = \tau_{\theta_B} = ad\left(\omega_1^2 - \omega_3^2\right) \tag{13}$$

Due to Newton's third law, the drag of the propellers produces a yawing torque on the body of the quadrotor. The aerodynamic torque is given by Corke (2011):

$$\tau_i = \pm b\omega_i^2, i = 1,2,3,4 \tag{14}$$

where, b is the torque constant, which depends on the same factors as a.

This torque exerts a reaction torque on the airframe which acts to rotate the airframe about the propeller shaft in the opposite direction to its rotation. Therefore the total yawing (positive) torque is given by Hassan *et al*. (2013):

$$\tau_{z_B} = \tau_{\varphi_B} = \sum_{i=1}^{i=4} \tau_i = b\left(\omega_2^2 + \omega_4^2 - \omega_1^2 - \omega_3^2\right) \tag{15}$$

where, the different signs are due to the different rotation directions of the rotors, thus a yaw torque can be created simply by appropriate coordinated control of all four rotor speeds.

The forces and torques on the quadrotor's airframe, obtained by combining Equation 1, 12, 13 and 15, can be written in matrix form as Corke (2011):

$$\begin{pmatrix} T \\ \Gamma_B \end{pmatrix} = C \begin{pmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{pmatrix} \Rightarrow \begin{pmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{pmatrix} = C^{-1} \begin{pmatrix} T \\ \tau_{x_B} \\ \tau_{y_B} \\ \tau_{z_B} \end{pmatrix} \tag{16}$$

Which gives the rotor speeds required to apply a specified thrust T and torque Γ_B to the airframe, where:

$$C = \begin{pmatrix} a & a & a & a \\ 0 & -ad & 0 & ad \\ ad & 0 & -ad & 0 \\ -b & b & -b & b \end{pmatrix} \tag{17}$$

The matrix C is of full rank if a, b, d>0, thus making the vehicle controllable.

The linear velocities in the different frames are related by http://scholarsarchive.byu.edu/cgi/viewcontent.cgi?article=2324&context=facpub (2015):

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{pmatrix}_I = \begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{pmatrix}_V = {}^V R_B \begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{pmatrix}_B \tag{18}$$

where, ${}^V R_B$ is the transformation matrix from frame $\{B\}$ to frame $\{V\}$ given by http://scholarsarchive.byu.edu/cgi/viewcontent.cgi?article=2324&context=facpub (2015):

$$\begin{aligned} {}^V R_B &= [{}^B R_V]^T = \left[{}^B R_{V^2}(\varnothing)^{V^2} R_{V^1}(\theta)^{V^1} R_V(\varphi) \right]^T \\ &= \begin{pmatrix} c\theta c\varphi & s\varnothing s\theta c\varphi - c\theta s\varphi & c\varnothing s\theta c\varphi + s\varnothing s\varphi \\ c\theta s\varphi & s\varnothing s\theta s\varphi + c\varnothing c\varphi & c\varnothing s\theta s\varphi - s\varnothing c\varphi \\ -s\theta & s\varnothing c\theta & c\varnothing c\theta \end{pmatrix} \end{aligned} \tag{19}$$

And the angular velocities are related as http://scholarsarchive.byu.edu/cgi/viewcontent.cgi?article=2324&context=facpub (2015):

$$\begin{pmatrix} \dot{\varnothing} \\ \dot{\theta} \\ \dot{\varphi} \end{pmatrix}_V = \begin{pmatrix} 1 & s\varnothing tan\theta & c\varnothing tan\theta \\ 0 & c\varnothing & -s\varnothing \\ 0 & -s\varnothing sec\theta & c\varnothing sec\theta \end{pmatrix} \begin{pmatrix} \dot{\varnothing} \\ \dot{\theta} \\ \dot{\varphi} \end{pmatrix}_B \tag{20}$$

where, $s \triangleq \sin, c \triangleq \cos, \varnothing \triangleq \varnothing_{V^2}, \theta \triangleq \theta_{V^1} and\ \varphi \triangleq \varphi_V$.

Control Algorithms

Control of the quadrotor input, $(\omega_1, \omega_2, \omega_3, \omega_4)^T$, is about applying the appropriate thrust, T and/or torque, Γ_B, to the airframe, which will be determined using the traditional PID-feedback controller, given by Katsuhiko (2002):

$$PID(e) = K_P e(t) + K_I \int_0^t e(\tau)\, d\tau + K_D \frac{de(t)}{dt} \tag{21}$$

Where:
e = Defined for each task below, is the error between the desired value and the output value
K_p = The proportional gain
K_I = The integrator gain
K_D = The derivative gain and t is time

In general the dynamics of rotational systems has a second order transfer function of the form Corke (2011):

$$\frac{\Theta(s)}{\tau(s)} = \frac{1}{Js^2 + Bs} \tag{22}$$

Where:
B = The aerodynamic damping
J = The rotational inertia
$\Theta(s)$ = The output signal (e.g., pitch)
$\tau(s)$ = The input signal (e.g., pitching torque)

B is generally quite small, thus the integrator controller is not necessarily in the regulation of the quadrotor and therefore PD-controller is applied in this research.

Attitude Controllers

This section presents control algorithms that make quadrotor pitch, roll and yaw. The PD-controller is used to determine the required torques based on the error between desired angles $(\theta_B^*, \varnothing_B^* and\ \varphi_B^*)$ and actual angles $(\theta_B, \varnothing_B and\ \varphi_B)$. For real-time application, the actual vehicle angles would be estimated by an inertia navigation system. The required rotor speeds are then calculated from the respective torques using Equation 16. Typically, the rate of the desired angles $(\dot{\theta}_B^*, \dot{\varnothing}_B^* and\ \dot{\varphi}_B^*)$ are small and can be ignored.

Controlling Pitch Motion

The pitch is controlled by increasing the speed, which in turn increase the thrust, of either rotor 1 or 3, whiles keeping the speed of rotor 2 or 4 the same or zero, as illustrated in Fig. 2. The required pitching torque on the airframe is given by:

$$\begin{aligned} &\tau_{y_B} = \tau_{\theta_B} = K_{p_\theta}\left(\theta_B^* - \theta_B\right) + K_{d_\theta}\left(\dot{\theta}_B^* - \dot{\theta}_B\right) \\ &\Rightarrow \tau_{\theta_B} = K_{p_\theta}\left[\left(\theta_B^* - \theta_B\right) - {K_{d_\theta}}/{K_{p_\theta}}\left(\dot{\theta}_B\right)\right] \end{aligned} \tag{23}$$

Controlling Roll Motion

Similarly, the roll is controlled by increasing the speed, which in turn increase the thrust, of either rotor 2 or 4, whiles keeping the speed of rotor 1 or 3 the same or zero, as illustrated in Fig. 3. The required rolling torque on the airframe is given by:

$$\tau_{x_B} = \tau_{\varnothing_B} = K_{p_\varnothing}\left(\varnothing_B^* - \varnothing_B\right) + K_{d_\varnothing}\left(\dot{\varnothing}_B^* - \dot{\varnothing}_B\right)$$
$$\Rightarrow \tau_{\varnothing_B} = K_{p_\varnothing}\left[\left(\varnothing_B^* - \varnothing_B\right) - {K_{d_\varnothing}}/{K_{p_\varnothing}}\left(\dot{\varnothing}_B\right)\right] \quad (24)$$

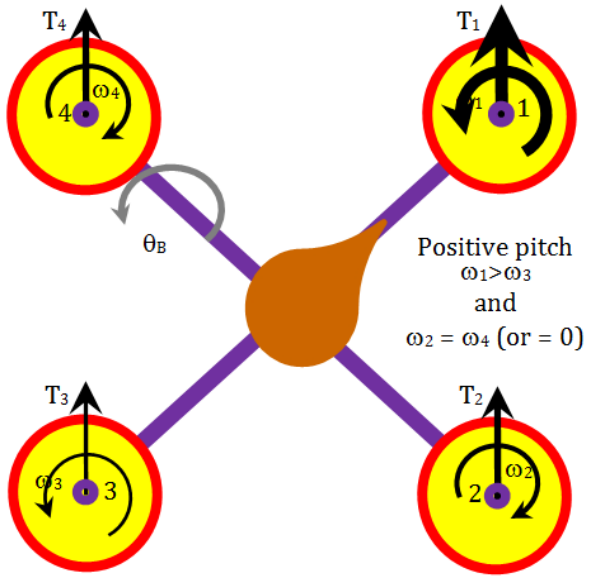

Fig. 2. Coordinates for pitch motion

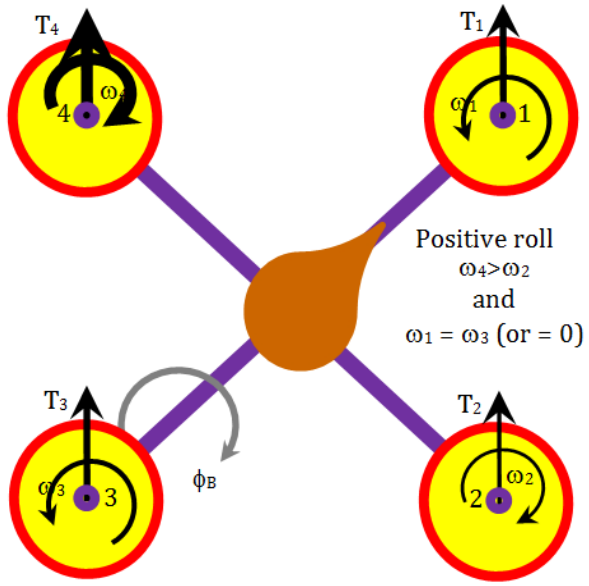

Fig. 3. Coordinates for roll motion

Controlling Yaw Motion

The yaw is controlled by simultaneously applying the same speed, which in turn changes the thrusts, of rotors 2 and 4 whiles keeping speeds of rotors 1 and 3 the same or zero or vice versa, as illustrated in Fig. 4. The required yawing torque on the airframe is given by:

$$\tau_{z_B} = \tau_{\varphi_B} = K_{p_\varphi}\left(\varphi_B^* - \varphi_B\right) + K_{d_\varphi}\left(\dot{\varphi}_B^* - \dot{\varphi}_B\right)$$
$$\Rightarrow \tau_{\varphi_B} = K_{p_\varphi}\left[\left(\varphi_B^* - \varphi_B\right) - {K_{d_\varphi}}/{K_{p_\varphi}}\left(\dot{\varphi}_B\right)\right] \quad (25)$$

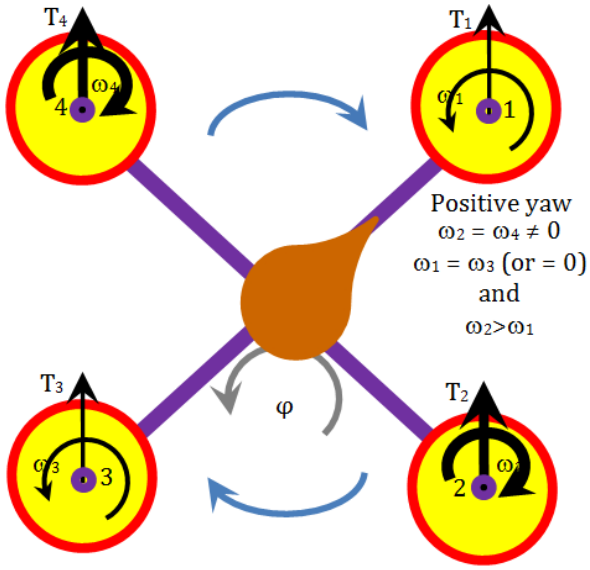

Fig. 4. Coordinates for yaw motion

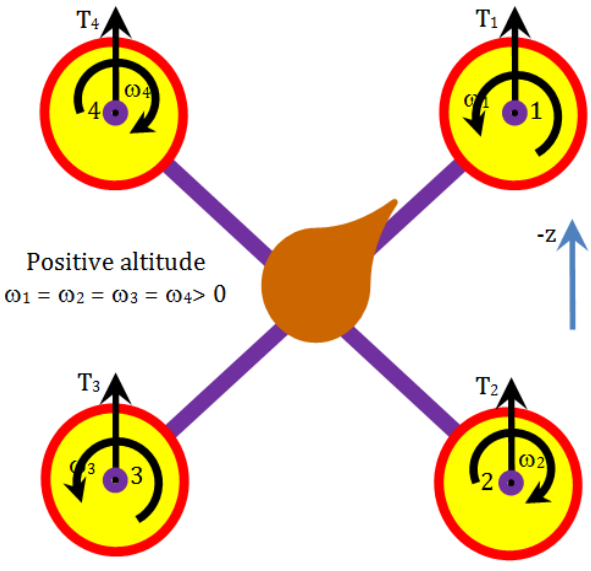

Fig. 5. Coordinates for altitude motion

Position Controllers

This section presents control algorithms that make quadrotor undergoes translational motions in the *x*, *y* and *z* directions. Again PD-controller is used to determine the required control inputs based on the error between desired positions $\left(x_I^*, y_I^* \text{ and } z_I^*\right)$ and actual positions (x_I, y_I and z_I). For real-time application, the actual vehicle positions and velocities would be estimated by an inertia navigation systems or GPS receivers. The required rotor speeds are then calculated from the respective torques using Equation 16. Once again, rate of the desired positions $\left(\dot{x}_I^*, \dot{y}_I^* \text{ and } \dot{z}_I^*\right)$ are small and can be ignored.

Controlling Altitude Motion

The altitude is controlled by simultaneously applying the same speed, which in turn changes the thrusts, of all the four rotors, as illustrated in Fig. 5. For upward motion the total thrust, *T*, must be bigger than the weight, *mg*, of the quadrotor, taken into account the drag on the vehicle. The total thrust on the airframe is given by:

$$\begin{aligned} &T = K_{p_z}\left(z_I^* - z_I\right) + K_{d_z}\left(\dot{z}_I^* - \dot{z}_I\right) + T_o \\ &\Rightarrow T = K_{p_z}\left[\left(z_I^* - z_I\right) - \frac{K_{d_z}}{K_{p_z}}\left(\dot{z}_I\right)\right] + T_o \end{aligned} \tag{26}$$

where, the additive term is given as:

$$T_o = mg = 4a\omega_o^2 \tag{27}$$

ω_o is the average rotor speed necessary to generate a thrust, T_o, equal to the weight of the vehicle.

A feed-forward control approach-used to counter the effect of gravity, which otherwise is a constant disturbance to the altitude motion. The alternative approach would be to have very high gains for the PD-controller. This second approach, not used in this research, often leads to actuator saturation and instability (Corke, 2011).

Controlling Motion in the x and y Directions

To move the vehicle in the *x*-direction (along X_V its nose pitch down, which generates a force Corke (2011):

$$f = R_{y_V}\left(\theta_V\right)\begin{pmatrix}0\\0\\T\end{pmatrix} = \begin{pmatrix}T\sin\theta_V\\0\\T\cos\theta_V\end{pmatrix} \tag{28}$$

which gives the force component that accelerates the vehicle in the X_V-direction as:

$$f_{x_V} = T\sin\theta_V \approx T\theta_V \tag{29}$$

where, θ_v is small. The velocity in this direction can be controlled using the P-controller (Corke, 2011):

$$f_{x_v}^* = mK_{f_x}\left(v_{x_V}^* - v_{x_V}\right) \tag{30}$$

Combining Equation 29 and 30, the desired pitch angle required to achieve the desired forward velocity is obtained as:

$$\theta_V^* = \frac{mK_{f_x}}{T}\left(v_{x_V}^* - v_{x_V}\right) \tag{31}$$

For a vehicle in vertical equilibrium the total thrust is equal to the weight of the airframe, thus $m/T = m/T_o$ in Equation 31 can be approximately substituted by g^{-1}.

Now, the desired velocity in frame $\{V\}$ relative to frame $\{I\}$ is then determined as (Corke, 2011):

$$v_{x_V}^* = {}^V R_I(\varphi) v_{x_I}^* = [{}^I R_V]^T(\varphi) v_{x_I}^* \tag{32}$$

where, the desired velocity in frame $\{I\}$ is given by the P-controller:

$$v_{x_I}^* = K_{P_{x_I}}\left(x_I^* - x_I\right) \tag{33}$$

Thus, to reach a desired *x*-position, the appropriate velocity is calculated and from that the appropriate pitch angle which will generate the force to move the vehicle is obtained. This indirection is consequence of the vehicle being under-actuated-the vehicle have just four rotor speeds to adjust but its configuration space is 6-dimensional. Substituting Equation 33 into Equation 32 and then the result into Equation 31, a compact form control algorithm for computing the desired pitch angle can be obtained as:

$$\theta_V^* = K_{P_x}\left([{}^I R_V]^T(\varphi)\left(x_I^* - x_I\right) - K_{D_x}\dot{x}_V\right) \tag{34}$$

Similar analysis can be carried out to obtain the control algorithm to move the vehicle in the *y*-direction (along Y_V). To reach the desired *y*-position, the appropriate velocity, $v_{y_V}^*$, is calculated and from that the appropriate roll angle, $\varnothing_V^*$, which will generate the force to move the vehicle is obtained. The compact form control algorithm for computing the desired roll angle is also given as:

$$\varnothing_V^* = K_{P_y}\left([{}^I R_V]^T(\varphi)\left(y_I^* - y_I\right) - K_{D_y}\dot{y}_V\right) \tag{35}$$

To move the vehicle forward or sideway its airframe must first pitch down or roll down respectively so that

the thrust vector has a horizontal component to accelerate it; the total thrust must be increased so that the vertical thrust component still balances gravity. As it approaches its goal the airframe must be rotated in the opposite direction, pitching up or rolling up, so that the backward component of the thrust decelerates the forward or the sideway motion. Finally, the airframe rotates to the horizontal with the thrust vector vertical. The cost of the under-actuation is once again maneuver. The pitch and the roll angles cannot be arbitrarily set, they are means to achieve the translation control. Equation 34 and 35 can be combined as:

$$\begin{pmatrix}\theta_V^* \\ \varnothing_V^*\end{pmatrix}=\begin{pmatrix}K_{P_x} \\ K_{P_y}\end{pmatrix}*\left([{}^I R_V]^T(\varphi)\left(p_I^*-p_I\right)-K_{D_{xy}}\begin{pmatrix}\dot{x}_V \\ \dot{y}_V\end{pmatrix}\right) \tag{36}$$

where, $p_I=(x_I,y_I)^T$, $K_{D_x}=K_{D_y}=K_{D_{xy}}$ and:

$$[{}^I R_V]^T(\varphi)=[{}^I R_V]^{-1}(\varphi)=\begin{pmatrix}c\varphi & -s\varphi \\ s\varphi & c\varphi\end{pmatrix} \tag{37}$$

Simulation of the Control Algorithms

The simulations were carried out using MATLAB/Simulink models to design and validate the control algorithms. Simulink block, Quadrotor Dynamics, as shown in Fig. 6, is used to implement the dynamics and kinematics of the quadrotor discussed in Section II. The block employs a MATLAB *S-function* to generate a continuous state output, *x*.

S-functions (system-functions) provide a powerful mechanism for extending the capabilities of the Simulink environment. *S-functions* follow a general form and can accommodate continuous, discrete and hybrid systems. An algorithm in an *S-function* is implemented by following a set of simple rules and using an *S-Function* block to add it to a Simulink model. The block consists of a set of inputs, a set of states and a set of outputs, where the outputs are a function of the simulation time, the inputs and the states.

The Quadrotor Dynamic block is connected to a Control Mixer block, whose inputs are the three torques and the total thrust acting on the airframe. The Control Mixer block is used to compute the four rotor speeds, which serves as inputs to the Quadrotor Dynamics block. These blocks were developed by using Simulink models in Corke (2011).

Simulink Models for the Controllers

This section discusses various Simulink models developed for the different control algorithms discussed in section III.

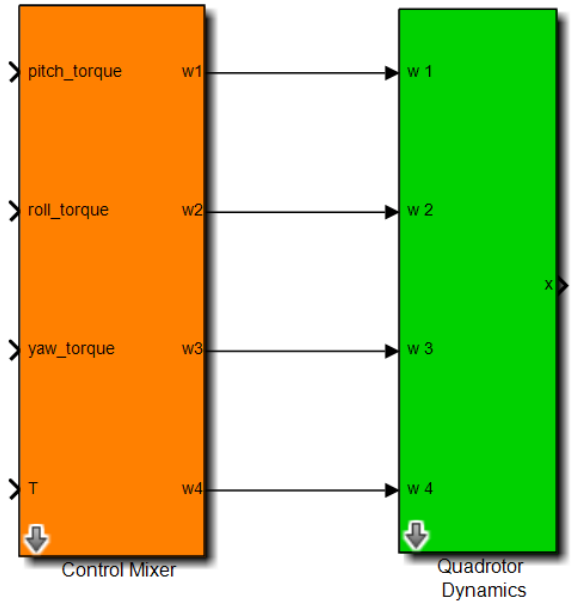

Fig. 6. Quadrotor dynamics and control mixer Simulink blocks Corke (2011)

Altitude Controller

The Simulink model shown in Fig. 7 is the altitude control algorithm. For a given desired height, z^*, the altitude control loop, based on Equation 26, is used to generate the required thrust, *T*, whiles the torque, Γ_B, to the airframe is set to zero.

Yaw Controller

The Simulink model shown in Fig. 8 is the yaw control algorithm. The vehicle must first moved to a desired height before yawing, hence the altitude controller is combined with the yaw controller, with an if-else control blocks to implement the conditional statement. Thus, for a desired yaw, *yaw**, the altitude control loop drives the quadrotor to a desired height, z^* and based on Equation 25, the yaw control loop is used to generate the required yaw torque, τ_{φ_B}, whiles keeping the other two torques on the airframe at zero.

Pitch Controller

The Simulink model shown in Fig. 9 is the pitch control algorithm. The vehicle must first moved to a desired height before pitching, hence the altitude controller is combined with the pitch controller, with an if-else control blocks to implement the conditional statement. Thus, for a desired pitch, *pitch**, the altitude control loop drives the quadrotor to a desired height, z^* and based on Equation 23, the pitch control loop is used to generate the required pitch torque, τ_{θ_B}, whiles keeping the other two torques on the airframe at zero.

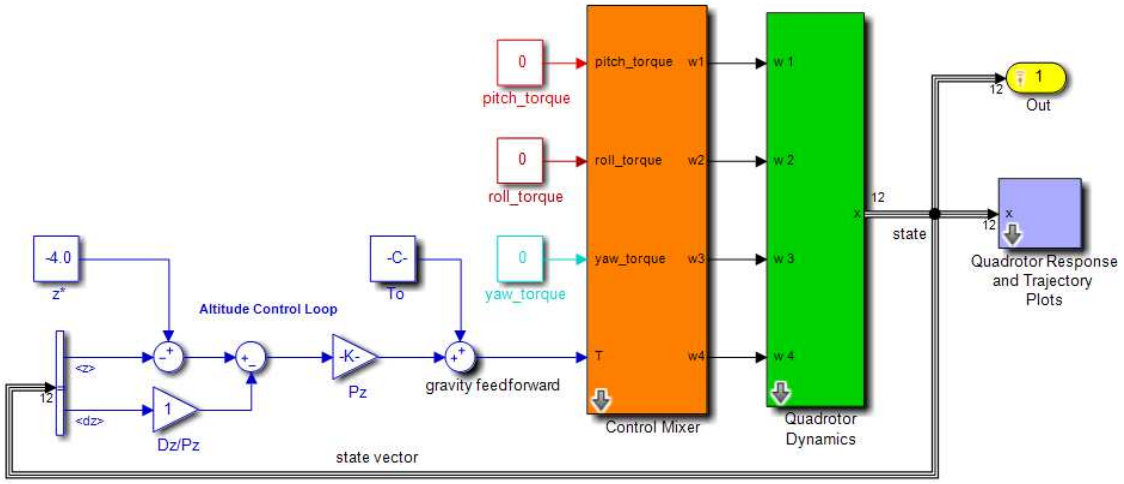

Fig. 7. Simulink model for altitude control Corke (2011)

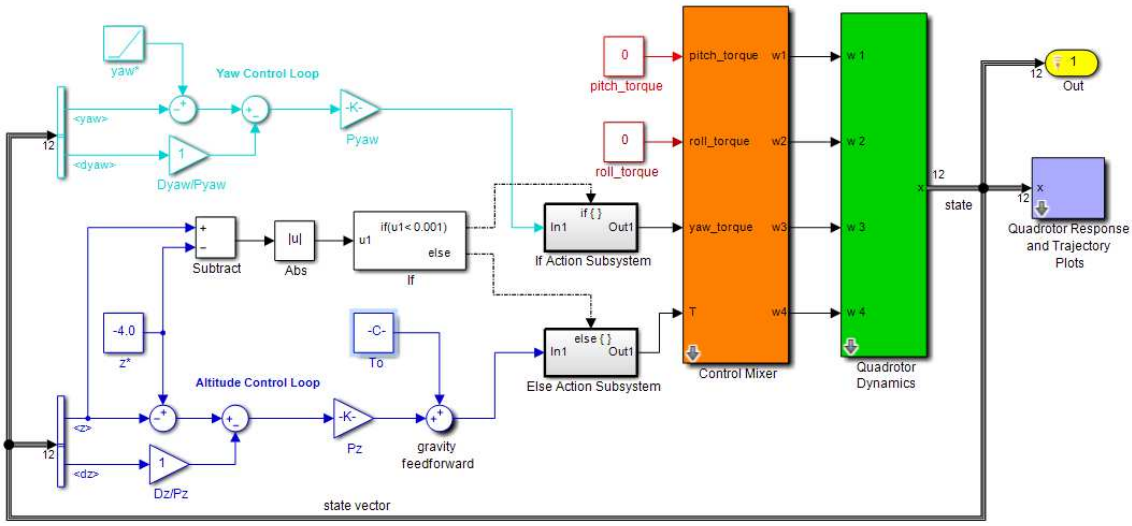

Fig. 8. Simulink model for yaw control Corke (2011)

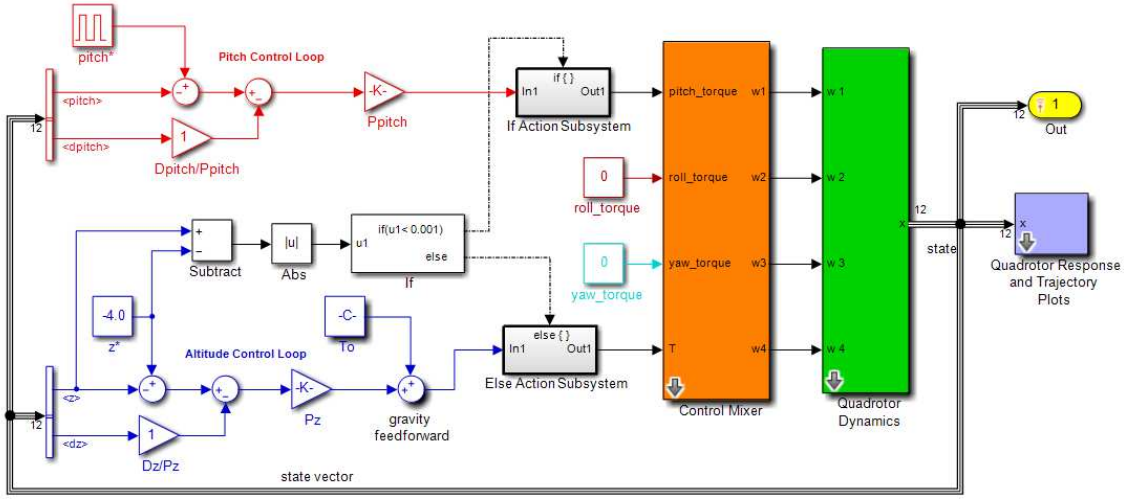

Fig. 9. Simulink model for pitch control Corke (2011)

Roll Controller

The Simulink model shown in Fig. 10 is the roll control algorithm. The vehicle must first moved to a desired height before rolling, hence the altitude controller is combined with the roll controller, with an if-else control blocks to implement the conditional statement. Thus, for a desired roll, $roll^*$, the altitude control loop drives the quadrotor to a desired height, z^* and based on Equation 24, the roll control loop is used to generate the required roll torque, $\tau_{\varnothing_B}$, whiles keeping the other two torques on the airframe at zero.

Position Controller

The Simulink model shown in Fig. 11 is the x and y directions control algorithms. The vehicle must first moved to a desired height before moving to the desired position in the xy-plane and the yaw angle, φ, is also needed, hence this model combines the altitude controller, yaw controller and the xy-plane motion controller.

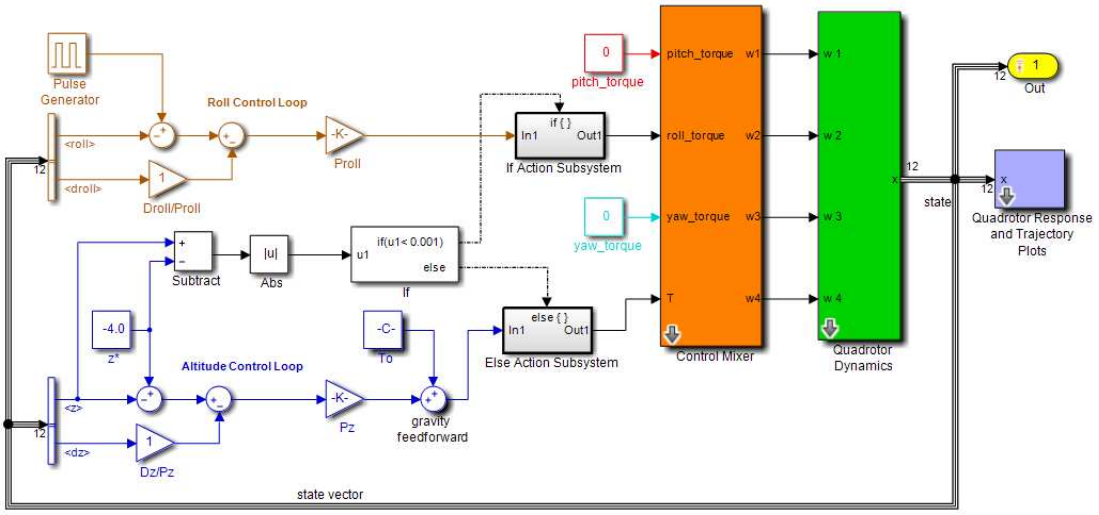

Fig. 10. Simulink model for roll control Corke (2011)

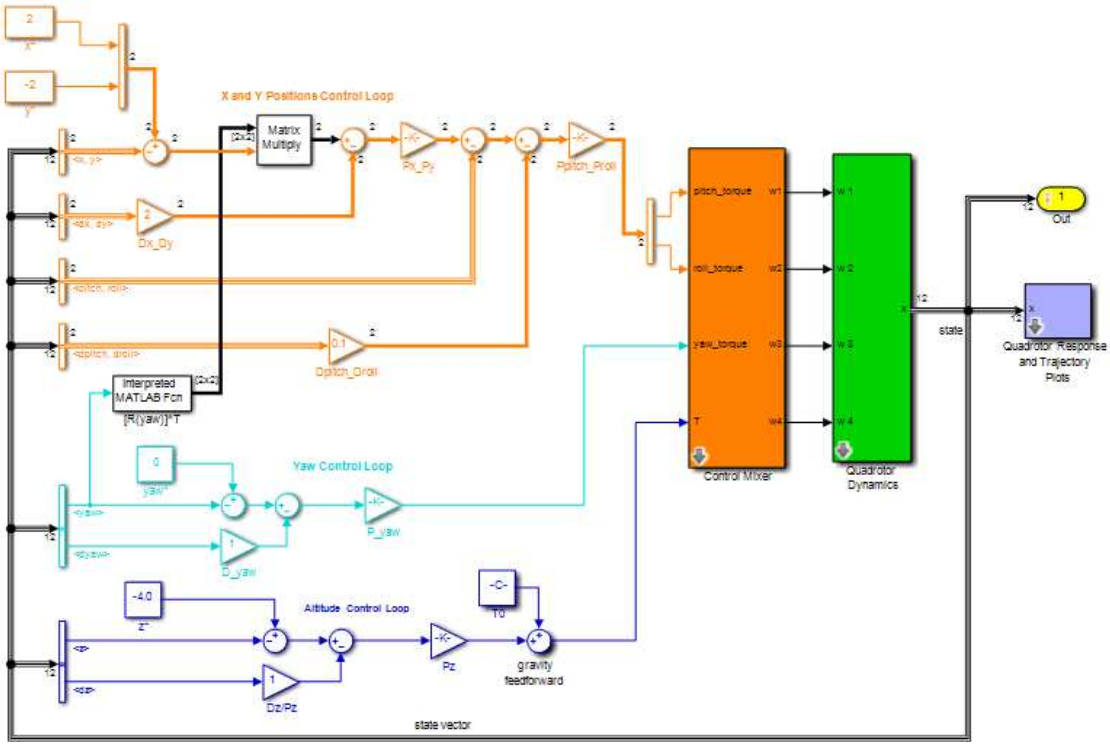

Fig. 11. Simulink model for position control Corke (2011)

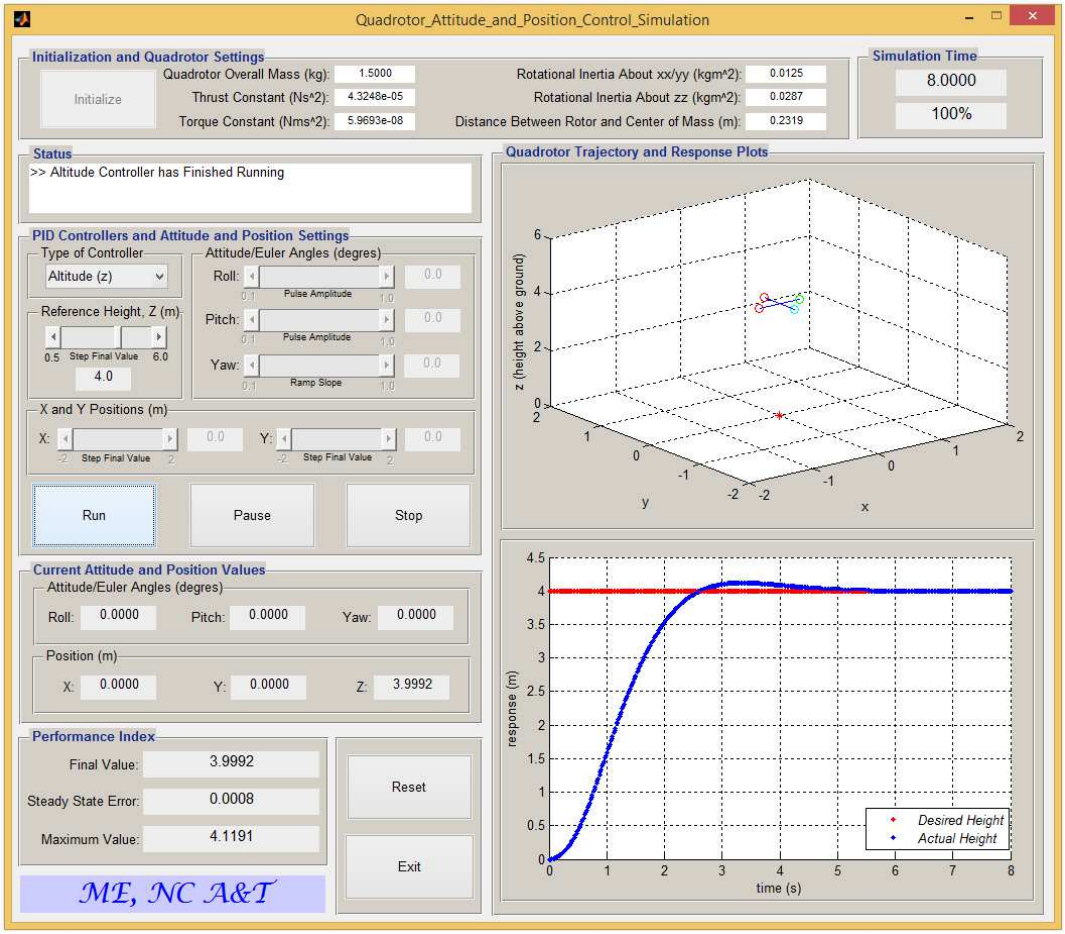

Fig. 12. MATLAB GUI used to implement the various control algorithm simulations

Table 1. Parameters used for the quadrotor

Parameter	Symbol	Value	Units
Mass	m	1.5000	kg
Thrust constant	a	4.3248e-05	Ns^2
Torque constant	b	5.9693e-08	Nms^2
Inertial matrix	I_{xx}, I_{yy} and I_{zz}	$\begin{pmatrix} 0.0125 & 0 & 0 \\ 0 & 0.0125 & 0 \\ 0 & 0 & 0.0287 \end{pmatrix}$	kmg^2
Distance between rotor and center of mass	d	0.2319	m

MATLAB GUI for the Simulations

A screenshot of a robust and interactive MATLAB GUI interface developed to implement the various control algorithms simulations discussed above is shown in Fig. 12.

The GUI interface has five sections: Initialization and information about the quadrotor settings, selecting the type of controller and its parameter settings, displaying the quadrotor current attitude and position values, displaying some performance index values of a run simulation and plotting the quadrotor's response and trajectory.

The GUI interface works by first initializing the quadrotor's parameters and then selecting the type of controller and its parameter settings required for the simulation. The selected controller is then simulated using its parameter settings selected. The quadrotor attitude and position real-time state values and responses can be displayed and the 3D trajectory of the quadrotor can also be visualized. An animated quadrotor is graphically designed, which receives the data from the simulation and execute the response in real time. After the simulation some performance index values are displayed.

The quadrotor real-time response and trajectory plots, shown in the GUI, were achieved by using S-function block in the Simulink models. The quadrotor real-time attitude and position state values, displayed in the GUI, were also achieved by using MATLAB Output block run-time object and an event-listener mechanism add_exec_event_listener command and then synchronizing the run-time object with the Simulink execution. The state output, x, is the input to the two blocks.

Table 1 gives a summary of the basic physical parameters used for the quadrotor Becker *et al*. (2012).

Results and Discussion

Simulation of Individual Controllers Results

All the time domain Simulink simulations were carried out under 10 sec duration for each model (refer to the simulation setups in the Fig. 7-11). The responses, using the various control algorithms, are shown in the Fig. 12-16. The PD-gains in this research were obtained by tweaking the various values to obtain the satisfactory responses. The various PD-gains, some performance indicators.

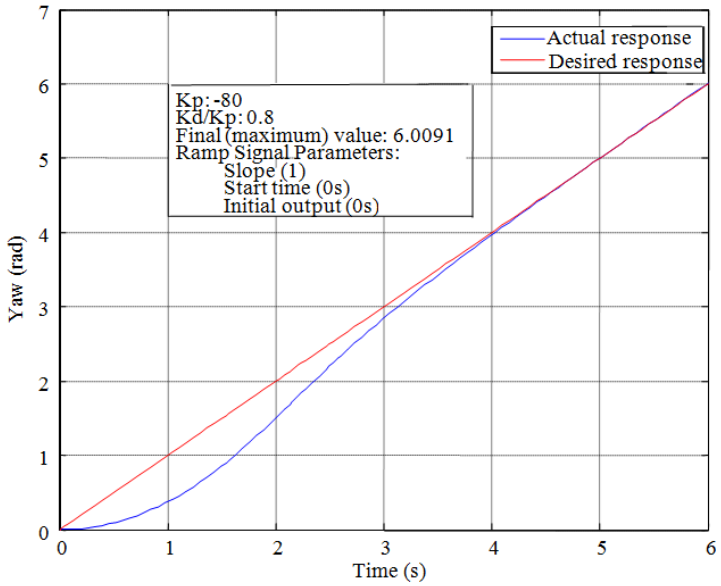

Fig. 13. Yaw control: Ramp response

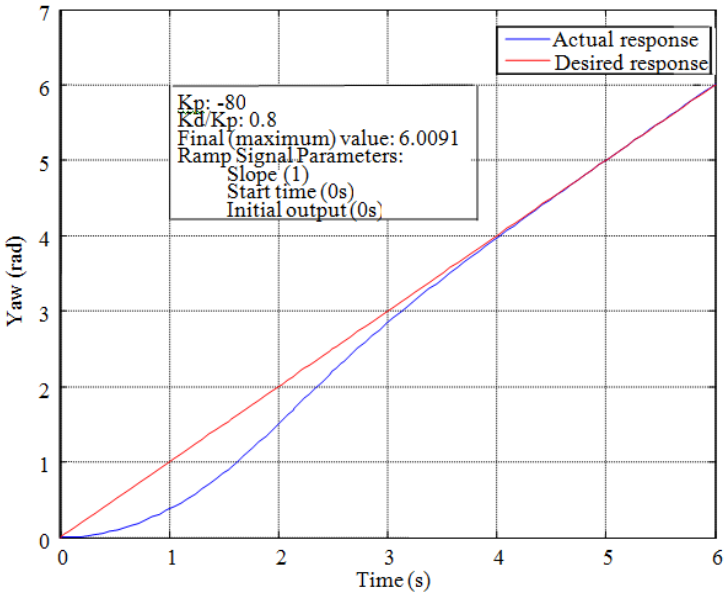

Fig. 14. Pitch and roll control: Pulse response

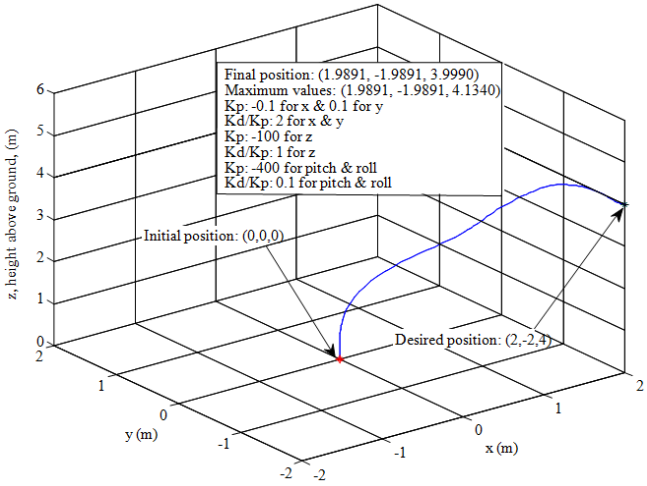

Fig. 15. Position control trajectory: Step response

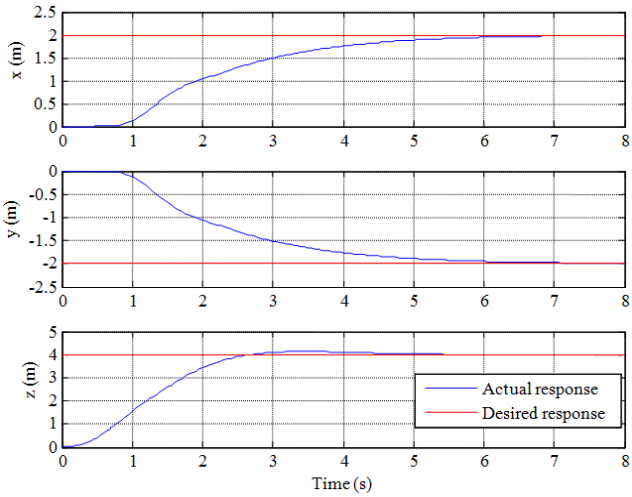

Fig. 16. Position control: Step response

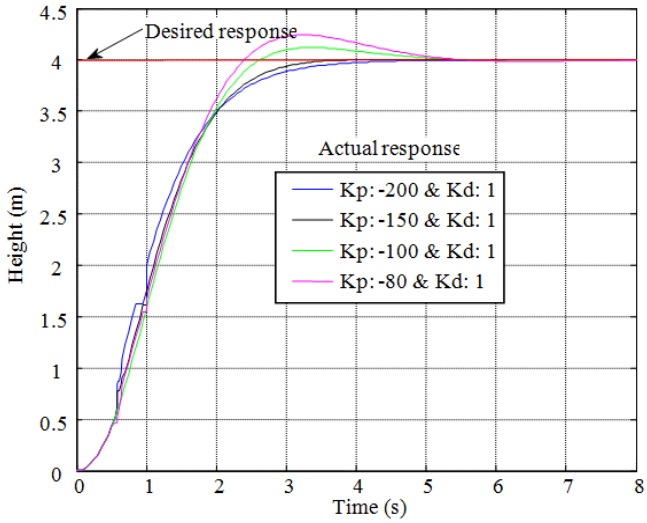

Fig. 17. Effect of p-gains on the altitude response

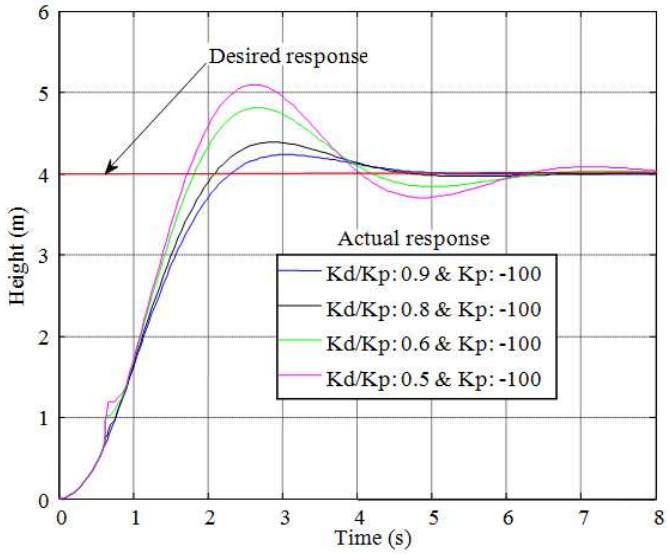

Fig. 18. Effect of d-gains on the altitude response

Table 2. Effects of PD gains on the altitude response with a desired height of 4m

K_p	K_d/K_p	Maximum overshoot	Settling time (s)	Rise time (s)	Peak value (m)
-200.0	1.0	0.0000	3.3914	1.8278	3.9999
-150.0	1.0	0.0724	2.9407	1.7048	4.0029
-100.0	1.0	2.9775	4.0636	1.5818	4.1191
-80.00	1.0	6.0702	4.6391	1.5520	4.2428
-100.0	0.9	5.8473	4.2467	1.4405	4.2339
-100.0	0.5	27.3765	7.2984	1.0993	5.0951

Even though there were steady-state errors in the various values obtained, the results were encouraging. The errors in the steady state can be due to using PD-gains which were not determined from classical control design approach but by tweaking. Another possible cause of error in the steady state is in the modeling of the dynamics and the kinematics of the quadrotor such as unmodeled aerodynamic effects and uncertainties (e.g., wind disturbance and saturation). Furthermore, according to Brockett's conditions Brockett (1982), feedback control law, which is a continuously differentiable, time-invariant, cannot be used to obtain error-free stabilization.

Effect of the PD Gains on the Altitude Response

Table 2 above and Fig. 17 and 18, shows how changes in the values of the PD-gains affect the altitude control response during 8 seconds of simulation. Theoretically, the values in the table conforms the literature http://www.cds.caltech.edu/~murray/courses/cds101/fa02/caltech/astrom-ch6.pdf 2015. For example, as the K_p increases at constant K_d, the overshoot increases and the rise time decreases. Also, as the K_d increases at constant K_p, the overshoot decreases and there are minor changes in the rise time.

Conclusion

This study demonstrated design of feedback controllers for position and attitude regulation of quadrotor UAVs. Also, MATLAB/Simulink models are used to design and validate the control algorithms. The simulation results are presented graphically and quantitatively. A robust and interactive MATLAB GUI interface for implementation of the various control algorithms simulation models is developed. Even though the final steady-state values obtained from the experiments are affected by unexpected disturbances, noise and modelling errors, the simulation results were encouraging.

In future, the authors will implement these PD-feedback controllers developed on the Parrot AR. Drone. In fact, an attempt has been made at the implementation, but challenges and difficulty were encountered. The main difficulty was in using the *S-function* to implement the dynamics and kinematics of quadrotors. The problem is in C-code generation for real-time application.

In order to address this problem, linearized models are being developed to represent the dynamics of the quadrotor for the different controllers. Another approach being study is to use system identification to determine the models based on data collected from the Parrot AR.

Drone. These approaches will allow the use of a classical control design approach to determine robust control laws instead of the tweaking approach used in this research to determine the PD-gains.

Performance, in terms of efficiency and accuracy, will then be compared of controllers such as pole-placement, Linear Quadratic Regulator (LQR), Linear Quadratic Gaussian (LQG), Model Predictive Controller (MPC) and Model Reference Adaptive Control (MRAC). In addition, robust control algorithm considering time delays will be studied.

Acknowledgement

This research was supported by NC Space Grant and Gachon University. The findings and opinions expressed in this study, however, are those of the authors alone and not necessarily the views of the sponsoring agency.

Author's Contributions

Stephen Armah: Participated in all experiments, coordinated the data-analysis and contributed to the wiring of the manuscript.

Sun Yi and Wonchang Choi: Designed the research plan and organized the study.

Dongchul Shin: Designed the research plan and organized the study and contributed to the writing of the manuscript.

Ethics

This paper is original and contains unpublished material. The authors declare that there is no conflict of interests regarding the publication of this paper.

References

Armah, S., 2015. Adaptive control for autonomous navigation of mobile robots considering time delay and uncertainty. PhD Thesis, North Carolina A&T State University.

Becker, M., R.C.B. Sampaio, S. Bouabdallah, V. Perrot and R. Siegwart, 2012. In-flight collision avoidance controller based only on OS4 embedded sensors. J. Brazilian Society Mech. Sci. Eng., 34: 295-307. DOI: 10.1590/S1678-58782012000300010

Brockett, R.W., 1982. New Directions in Applied Mathematics. 1st Edn., Springer-Verlag, New York, pp: 27.

Corke, P., 2011. Robotics, Vision and Control: Fundamental Algorithms in Matlab. 1st Edn., Springer Science and Business Media, Berlin Heidelberg Springer, ISBN-10: 3642201431, pp: 570.

Guenard, N., T. Hamel and R. Mahony, 2008. A practical visual servo control for a unmanned aerial vehicle. IEEE Trans. Robot., 24: 331-340. DOI: 10.1109/TRO.2008.916666

Hassan, M., S. Faiz, D. Hazry, A. Faizan and A. Warsi *et al*., 2013. Stabilized controller design for attitude and altitude controlling of quadrotor under disturbance and noise conditions. Am. J. Applied Sci., 10: 819-831. DOI: 10.3844/ajassp.2013.819.831

Katsuhiko, O., 2002. Modern Control Engineering. 4th Edn., Prentice Hall, Upper Saddle River, ISBN-10: 0130609072, pp: 964.

http://www.cds.caltech.edu/~murray/courses/cds101/fa02/caltech/astrom-ch6.pdf 2015

http://scholarsarchive.byu.edu/cgi/viewcontent.cgi?article=2324&context=facpub 2015

Nomenclatures

$\{V\}$	Vehicle Frame/Coordinate System
$\{B\}$	Body Frame/Coordinate System
$\{O\}$ or $\{I\}$	Global Inertia Frame/Coordinate System
A	System (State) Matrix
B	Input Matrix
C	Output Matrix
D	Feed forward/Feed through Matrix
s	Laplace Transform Operator
z	Z Trans form Operator
$\Re$	Real Plane
C	Complex Plane
$CONT$	Controllability Matrix
$OBSER$	Observability Matrix
t	Time
x	State Vector
y	Output Vector
u	Control Input Vector
e	Error
r	Reference Input
K_p	Proportional Gain
K_I	Integral Gain
K_D	Derivative Gain
ξ	Plant Damping Ratio
ξ_f	Filter Damping Ratio
ω_n	Plant Natural Frequency
ω_f	Filter Natural Frequency
K	Control Gains Matrix
K	Real Constant Tuning Parameter
I	Identity Matrix
I	Desired Closed-loop Poles (DCLP)
β_1	Real Part of DCLP
β_2	Imaginary Part of DCLP
J_c	Cost Function
J_r	Rotational Inertia
J	Rotational Inertia Matrix

B_r	Aerodynamic Damping
G_p	Plant Transfer Function
G_r	Plant Transfer Function without Time Delay
G_m	Reference Model Transfer Function
G_{mt}	DC Motor Dynamics Transfer Function
θ	MIT Rule MRAC Updating Parameter
γ	MRAC Tuning Parameter
T_d	Time Delay
Δ_m	Added/Perturbed Mass
m_q	Quadrotor's Total Mass
m_{ar}	AR.Drone 2.0 Overall Mass (Indoor)
m_e	Mass at End of Cross Arm of Quadrotor
m_c	Mass of Quadrotor Center
T_s	AR.Drone 2.0 System Sampling Time
d_q	Distance between Quadrotor's Rotor and Center of Mass
$\omega_B/\omega_B/I$	Quadrotor's Angular Velocity w.r.t. {B}
${}^{B}R_V$	Rotational Matrix from {V} to {B}
v_B	Quadrotor's Linear Velocity w.r.t. {B}
F_B	Total Force on Quadrotor w.r.t. {B}
Γ_B	Applied Torque on Quadrotor w.r.t. {B}
h_B	Angular Momentum on Quadrotor w.r.t. {B}
ω_i, i = 1, 2, 3, 4	Quadrotor Rotors Angular Velocity
T_i, i = 1, 2, 3, 4	Quadrotor Rotors Upward Thrust
τ_i, i = 1, 2, 3, 4	Aerodynamic Torque on Quadrotor Rotors
g	Gravitational Strength
(J_{xx}, J_{yy}, J_{zz})	Quadrotor Moments of Inertial About the x, y and z axes
a	Thrust Constant
b	Torque Constant
(x, y)	DDWMR Position
(ρ, α, β)	Polar Coordinates
x_g	Desired x-value
y_g	Desired y-value
x_g	Desired Goal State
x_o	Initial State
x_f	Final State
φ_g	Desired Heading or Yaw or Turn
φ_{obs}	Obstacle Heading/Yaw/Turn
$\varnothing$	Roll
θ	Pitch
φ	Heading/Yaw/Turn
ω	Heading/Yaw/Turn Rate
ω_r	Right Angular Velocity
ω_l	Left Angular Velocity
k_v, k_α, k_β	Control Gains
k_ρ, k_h, k_d	
R	Radius of the DDWMR Wheels
L	Distance the Between Wheels of the DDWMR
ν	Linear Velocity
ν_o	Constant Linear Velocity
τ_s	Time at Last Switch
$\tau(s)$	Torque Input Signal
d_{τ_s}	Distance Between Goal and Point at Last Switch
α_B	Blending Function
β_B	Tuning Parameter for the Blending Function
u_{GTG}	Go-to-goal Mode
u_{AO}	Obstacle Avoidance Mode
u_{FW}^{c}	Clockwise Follow-wall Mode
u_{FW}^{cc}	Counterclockwise Follow-wall Mode
Δ_o	Distance Between Robot and Obstacle
Δ_x/Δ_y	Change in x/y
e'	Corrected Error
K_g	Quadrotor DC Motor Dynamics Steady-State Gain
τ	Quadrotor DC Motor Dynamics Time Constant
r_{yR}	Real Stability Radius
Δ	Perturbation Matrix
E and F	Scaling Matrices for Δ
Δ_{k_g}	Perturbation in Quadrotor DC Motor Dynamics Steady-State Gain
Δ_{τ_g}	Perturbation in Quadrotor DC Motor Dynamics Time Constant