PID based Autonomous Aerial Vehicle

Jerrin Bright[1]
[1]School of Mechanical Engineering
Vellore Institute of Technology, Chennai, India

Abstract: Developed a ROSpy based control system for a quadcopter to transverse to a set of GPSs setpoint autonomously. The Control System has two modules namely the Altitude controller and the position controller, Altitude controller stabilizes the drone at the zero error Roll, Yaw, Pitch angles using a PID based controller, the position controller takes in the target GPS coordinate has setpoint values and calculates the roll yaw pitch angles to successfully move to the setpoint coordinates. these controllers work in synchronization to autonomously fly the drone from one coordinate to another.

Keyword: PID, Autonomous Drones, ROS, Python

1. Introduction

Systems can in general be split into open and closed systems. Open system has no feedback thus no control, whereas closed is controlled and will receive signal/ feedback. PID is a closed loop control system which stands for Proportional Integral Derivative controller. It is widely used in industries with wide range of efficient applications. We have used a Micro Aerial Vehicle (MAV) here for experimenting our controller codes which was converted to URDF so that we could using in Gazebo Simulation Environment using Robotic Operating System (ROS). The MAV is equipped with Inertial Measurement Unit (IMU), Barometer, Ultrasonic Sensor, Time of Flight Sensor and GPS Receiver.

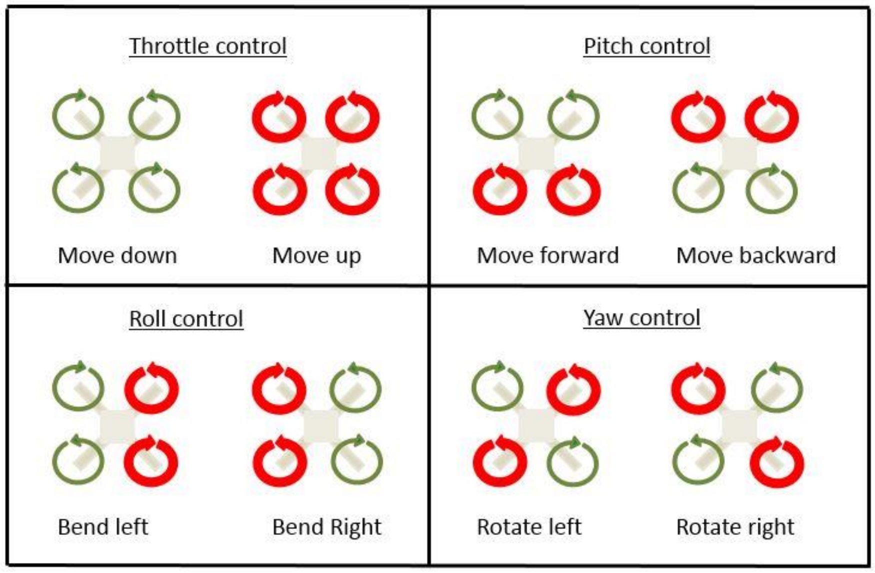

Figure 1. MAV Motion

The four major motions of a quadcopter include:

- Thrust/ throttle
- Pitch
- Roll
- Yaw

We are using Pixhawk controller here. PID Control system is used for the position and altitude control which will be explained in the coming section.

2. Methodology

In this section we will discuss all the types of controller that can be equipped.

Proportional Controller (P only)- Stabilizes unstable process. It helps in reducing the steady state error in the operation. But this controller can't always eliminate the steady state error. Thus, we will check along with Derivative controller next.

$$output = K_P * error \tag{1}$$

Proportional Derivative Controller (PD only)- Increases the net stability of the operation. Derivative part of the control system helps in predicting the future errors of the systems based on its response. Thus, it helps in controlling the sudden shift of the operation.

$$output = K_P * error + K_d * (error - previous\ error) \tag{2}$$

Proportional-Integral-Derivative Controller (PID)- Thus, this is a very dynamic system equipped with zero state error, fast response, no oscillations and high stability. Here in equation 3, Iterm is incremented for every estimated error value in the system.

$$output = K_P*error + (Iterm + error) * K_i + K_d*(error - previous\ error) \tag{3}$$

3. Our Work

We have used PID based control systems for the proposed system in fig. 1. The basic idea of PID is error of the system will be estimated based on setpoint (SP) values and the process variable (PV) value estimated. Thus, weights will be assigned and subsequent PWM values will be generated based on mathematical models and resulting speed of the rotors will be passed to the motors. This is the crux part of PID control system in general. We are using two PID modules in this work namely Altitude Controller (AC) and Position Controller (PC).

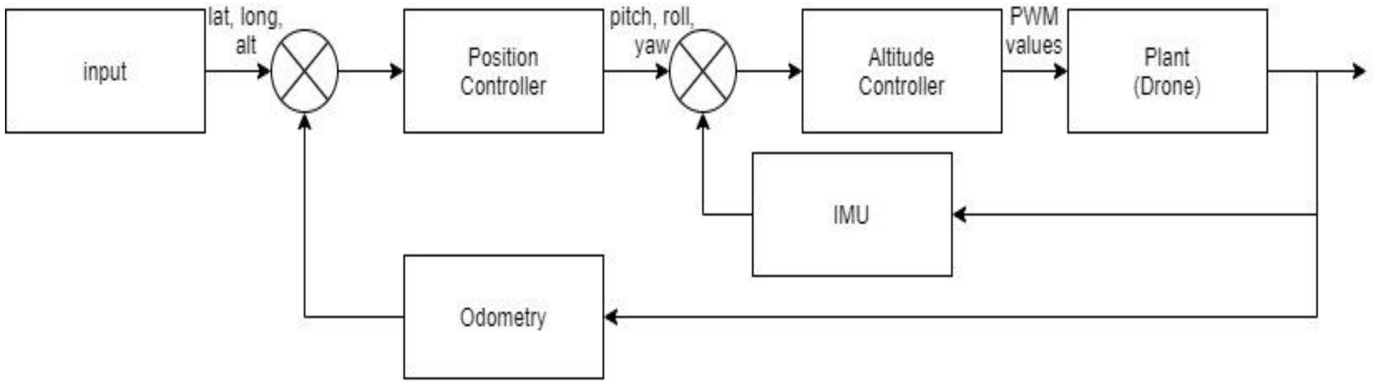

Figure 1. Control System for the Drone

The **Altitude Controller** stabilizes the drone at the zero-error roll, yaw and pitch (R-P-Y) angles using a PID based control system. That's, stabilizing the orientation of the drone (R-P-Y) using PID is the aim of altitude controller. Then tuning of the proportional, integral and derivative function has been done to get the best suitable controller configuration.

The **Position Controller** takes in the target coordinates as setpoint values and calculates the R-P-Y angles to successfully move to the setpoint coordinates. That's position of the drone with respect to the environment is estimated using this controller. This controller will be in sequence with the altitude controller designed as mentioned in the previous paragraph.

This altitude and position controllers work in synchronization to autonomously navigate in the environment in a robust manner. From fig. 1, we can clearly understand the workflow, the position that's the latitude, longitude and altitude is passed through the position controller as input and then the output of the position controller being the R-P-Y values (orientation) which will act as input of the altitude controller. Thus, a dynamic position controller was designed so that instantaneous change in the latitude, longitude and altitude (position) can be altered and navigation can be established accordingly.

4. Conclusion

Thus, in this work we have made a drone capable of autonomously navigating from one point to another with predefined setpoints declared using GPS receiver. Two controllers were used namely, position controller and altitude controller. The architecture is clearly visualized and explained for the ease of readers. This can later be advanced in such a way it could be made to navigate with obstacle avoidance capacity so that delivery like medical or food can be done efficiently and faster than traditional methods. It will be added to my future work. The source code for my work could be taken from https://github.com/jerriebright/PID-Drone

References

[1]. https://www.youtube.com/watch?v=wkfEZmsQqiA&list=PLn8PRpmsu08pQBgjxYFXSsODEF3Jqmm-y

[2]. http://brettbeauregard.com/blog/2011/04/improving-the-beginners-pid-introduction/