Technical Paper

Journal of Aerospace System Engineering
Vol.12, No.3, pp.45-52 (2018)

ISSN 1976-6300
EISSN 2508-7150
https://doi.org/10.20910/JASE.2018.12.3.45

오픈소스 기반 Pixhawk 미션모드 비행제어법칙 구조 분석

이영호[1,] · 신승찬[1] · 목지현[1] · 고상호[1,†]

[1]한국항공대학교 항공우주 및 기계공학부

Pixhawk mission mode flight control-law structure analysis based on Open-Source

Yeongho Lee[1], Seungchan Shin[1], Jihyun Mok[1] and Sangho Ko[1]

[1]School of Aerospace and Mechanical Engineering, Korea Aerospace University

Abstract

This paper deals with the analysis of the inner-loop algorithm of the Pixhawk which is a representative multi-copter open source. The algorithm is based on flight control-law structure. The inner-loop algorithm of the Pixhawk can be divided into a position controller and an attitude controller. The position controller generates the attitude of the multi-copter to move to the destination The position controller also generates the demand force and moment acting on each actuator. We confirm that the position controller saturates the desired acceleration and speed by using a proper relational expression. The expression can be used in order to prevent the sudden change in the attitude of a multi-copter.

초 록

본 논문에서는 현재 대표적으로 사용되고 있는 멀티콥터 오픈소스 기반 비행제어 컨트롤러인 Pixhawk의 내부루프 알고리즘 분석에 대해 연구한 내용을 다룬다. Pixhawk의 내부루프 알고리즘은 위치 제어기, 자세제어기로 나눌 수 있는데, 위치 제어기는 목표로 이동하기 위한 멀티콥터의 자세를 제시하며 자세제어기는 원하는 자세로 변할 수 있도록 구동기에 작용하는 요구 추력과 모멘트 변화를 발생시킨다. 여기서 위치제어기는 멀티콥터의 급격한 자세변화를 방지하기 위해 적절한 관계식을 이용하여 요구 가속도 및 속도에 대해 제한을 하고 있는 것을 확인 할 수 있다.

Key Words : Pixhawk(픽스호크), Mission-Mode(미션모드), Position Controller(위치 제어기), Attitude Controller(자세제어기)

1. 서 론

최근 멀티콥터는 단순한 구조와 제어 원리 그리고 저가형 비행 제어시스템 보급으로 인해 민간분야에서 많은 관심이 증가하고 있다[1,2]. 그 이유는 오픈소스 기반 비행제어 컨트롤러가 무료로 배포되고 있어 누구나 쉽게 펌웨어를 다운로드 받아 하드웨어에 올리는 것이 가능하기 때문이다. 특히 Table 1과 같이 현재 대표되고 있는 멀터콥터 오픈프로젝트 중 Pixhawk는 다른 오픈 소스와 달리 상업적으로 사용하여도 공개할 의무가 없으며 또한 누구나 코드를 받아 자유롭게 수정할 수 있는 장점이 있다[3]. 이에 국·내외 많은 실험실에서는 Fig. 1에서와 같이 Pixhawk의 외부루프 알고리즘(회피 기동, 카메라 영상 장치, 항법 등)을 검증하기 위한 하드웨어로써 많이 사용하고 있다.

Received: Nov. 10, 2017 Revised: May. 08, 2018 Accepted: June. 11, 2018
† Corresponding Author
Tel: +82-2-300-0119, E-mail: sanghoko@kau.ac.kr

Table 1 Open-Source Project

멀티 위 (Multiwii)	- 프로 미니 및 마이크로 컨트롤러 (아두이노 기반) - 닌텐도 사와 협업
에어로 쿼드 (Aero Quad)	- 아두이노 기반 - 쿼드콥터를 대상 기종
크레이지 플라이 (Crazyflie)	- 소형기체를 타깃 - 안전한 실내에서의 비행을 착안 - SKY-Rover와 협업
아두파일럿 (Ardupilot/APM)	- DIY Drones의 오픈 소스 드론 프로젝트 - 아두이노 기반의 하드웨어 - GCS 운용(APM Mission Planner)
PX4 오토파일럿 (Pixhawk)	- 자동항법 시스템 가능 - 상업적으로 사용하여도 공개 할 의무가 없음 - GCS 운용 (Qground Control) - Flight plot, HILS 제공 (ROS 와의 연계가능)

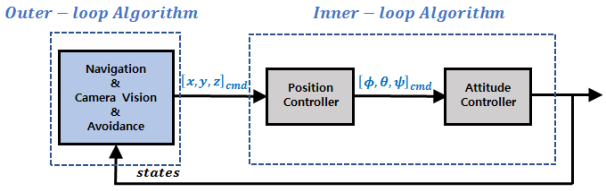


Fig. 1 Pixhawk Algorithm

하지만 이러한 대다수의 실험실에서의 외부루프 알고리즘 적용방식은 개발한 코드를 단순히 내부루프 알고리즘(위치제어기, 자세제어기)의 입력 성분에 맞추어 코드를 개발하는 것에만 초점이 맞추어져 있을 뿐 기존 내부루프 알고리즘의 검증 에 대한 연구는 미비하다. 또한 Pixhawk의 오픈소스가 일반 다수의 사용자들로 하여 매일 수정되고 있어 코드에서 제시한 내부루프 알고리즘과는 현재 많은 차이를 보이며 코드 량 또한 방대하여 내부루프 알고리즘을 이해하는 데에 많은 어려움이 있다. 따라서 일반 사용자들은 내부루프 알고리즘 이해를 위해 단순히 오픈포럼에서의 의견 교류를 통해서만 내부루프의 구조의 분석결과와 알고리즘의 안정성을 믿고 사용하고 있다.

이에 본 논문은 오픈소스 기반 비행제어 컨트롤러인 Pixhawk의 미션모드 내부제어기 구조를 분석하고 이해하려 한다.

2. Pixhawk 구조

2.1 좌표계 및 회전변환

Pixhawk 제어기 구조는 Fig. 2에와 같이 2-1-3 $(\Theta-\Phi-\Psi)$ 회전 변환을 이용하고 있다. 좌표계의 기호표기 방식은 지면좌표계의 기호는 I, 2-1-3 회전 변환에서 2-1$(\Theta-\Phi)$ 회전 변환 시 나타나는 좌표계의 기호는 C , 이후 3(Ψ) 회전으로 나타나는 기체 좌표계의 기호는 B 로 정의한다. 회전 변환 행렬의 기호 표기방식은 지면좌표계에서 2-1 회전변환으로 생성되는 좌표계의 회전변환 행렬을 R_C, 지면 좌표계에서 기체좌표계로의 회전 변환 행렬을 R_B로 정의한다.

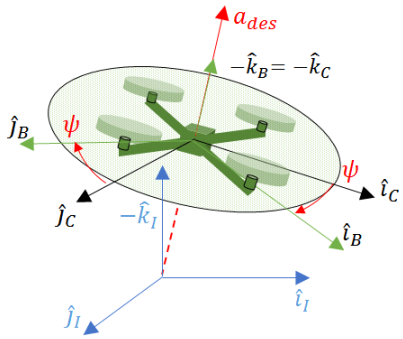


Fig. 2 Coordinate system

2.2 구성

Pixhawk의 제어기 구성은 Fig. 3에서 보이는 바와 같이 자세제어기(Position Controller), 위치제어기(Attiude Controller), 구동기 관계식(Motor Dynamic)으로 구성되어 있다[4]. 위치 제어기는 지면 좌표계를 기준으로 하여 목표점$[x_{I,cmd}, y_{I,cmd}, z_{I,cmd}]$을 입력 시 현재 위치, 현재 속도와의 상관관계를 이용하여 요구 가속도를 생성한다. 특히 요구 가속도가 크면 멀티콥터의 급격한 자세 변화를 발생 시키므로 위치 제어기는 요구 속도 및 가속도에 대해 제한(Saturation)을 두고 있다. 자세제어기는 기체 좌표계를 기준으로 하여 요구 오일러 각 변화 값$[\phi_{B,error}, \theta_{B,error}, \psi_{B,error}]$

을 입력 시 현재 각속도 성분과의 상관관계를 이용하여 요구 각 가속도를 생성한다. 구동기 관계식은 요구 각 가속도와 요구 가속도를 이용하여 구동기 4개의 요구 회전수를 구하고 이를 이용하여 멀티콥터의 요구 모멘트 및 추력을 생성한다.

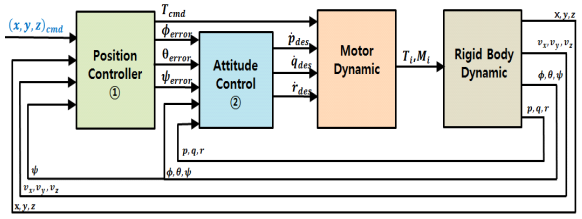


Fig. 3 Pixhawk Inner-loop Algorithm

2.3 운동 모델

멀티콥터는 구동기의 추력과 반동 모멘트를 이용하여 전체 추력과 모멘트를 발생시킨다. 여기서 구동기 각각의 추력과 반동모멘트는 Eq. 1과 같이 구동기의 회전수(ω_i)의 제곱의 비례관계로 표현할 수 있으며, 비례관계 상수는 각각 추력계수(k_f), 모멘트 계수(k_M)로 정의한다[5].

$$f_i = k_f \omega_i^2, \quad m_{mot,i} = (-1)^i k_m \omega_{i,}^2 \quad where,\ i = 1,2,3,4 \tag{1}$$

Pixhawk에서 제공하고 있는 비행체 모델 중 (+)형태의 쿼드콥터를 대상으로 하여 전체추력을 F, 무게중심에서 구동기 위치까지의 거리를 l, 롤, 피치, 요 모멘트를 각각 L, M, N이라 하면 Eq. 1을 확장하여 Eq. 2와 같이 표현할 수 있다. 여기서 Fig. 2와 같이 전진방향을 구동기 1번으로 한다면 Eq. 2에서 전체 추력은 구동기 각각의 추력의 합으로 롤 모멘트는 구동기 2번과 4번, 피치 모멘트는 구동기 1번과 3번의 추력을 이용하여 표현할 수 있다. 요 모멘트는 구동기의 각각의 반동모멘트 합으로써 나타낼 수 있다[5].

$$\begin{bmatrix} F \\ L \\ M \\ N \end{bmatrix} = \begin{bmatrix} k_f & k_f & k_f & k_f \\ 0 & k_f \ell & 0 & -k_f \ell \\ -k_f \ell & 0 & k_f \ell & 0 \\ -k_m & k_m & -k_m & k_m \end{bmatrix} \begin{bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{bmatrix} \tag{2}$$

3. Pixhawk 제어기 구성

3.1 위치 제어기 구조

위치 제어기 구조는 Fig. 4과 같이 요구 속도(V_{des}) 생성과 요구 가속도(a_{des}) 생성 부분으로 구성된다[3].

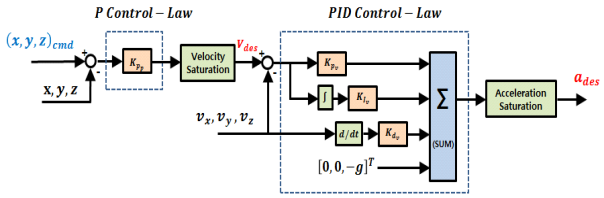


Fig. 4 Position controller model

3.1.1 요구 속도(V_{des}) 생성

Equation 3과 같이 목표점을 $[x_{cmd}, y_{cmd}, z_{cmd}]^T$, 현재 위치를 $[x, y, z]^T$라고 하면 요구속도는 목표점과 현재 위치와의 오차에 대해 이득(K_{P_p})을 곱하여 구할 수 있다. 요구속도는 목표점이 멀수록 선형으로 비례하여 커지므로 무한정 커지는 것을 제한해야 한다.

$$V_{des} = K_{P_p} \begin{bmatrix} x_{cmd} - x \\ y_{cmd} - y \\ z_{cmd} - z \end{bmatrix} = v_{x,des}\,\hat{i}_I + v_{y,des}\,\hat{j}_I + v_{z,des}\,\hat{k}_I \tag{3}$$

3.1.2 요구 속도 제한

요구 속도를 제한방식은 Fig. 5와 같이 요구 속도를 x-y 평면으로 투영한 성분 과 z방향 성분 $V_{xy,des}$로 분리하여 계산한다.

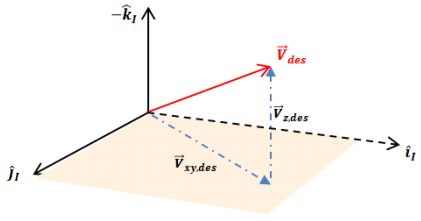


Fig. 5 Schematic of desired velocity

$$V_{z,des} = v_{z,des}\,\hat{k}_I$$
$$V_{xy,des} = v_{x,des}\,\hat{i}_I + v_{y,des}\,\hat{j}_I \tag{4}$$

먼저 $V_{z,des}$의 제한은 사용자가 지정한 최댓값을 기준으

로 제한한다. 다음으로 $V_{xy,des}$의 제한은 Fig. 6와 같이 현재 위치가 목표점을 중심으로 한 목표점 반경(Threshold)에 들어오는지에 따라 다르게 적용한다.

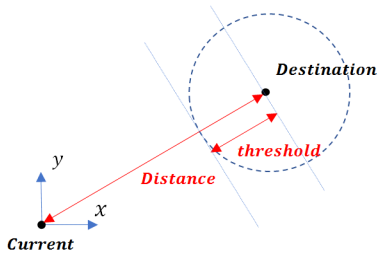


Fig. 6 Distance and Threshold

목표점 반경을 기준으로 속도를 제한하는 방식은 *step Ⅰ*, *step Ⅱ,Ⅲ* 순으로 진행된다(Fig. 7). 여기서 *step Ⅱ*와 *step Ⅲ* 는 동일한 제한 방식을 사용하고 있다.

step Ⅰ : 멀티콥터의 현재위치가 목표점 반경에 들어오지 못한다면 수평 속도 최댓값($V_{xy,\max}$)을 이용하여 제한을 둔다. 여기서 속도 최댓값은 사용자가 변수로 지정한 일정한 값을 이용한다.

step Ⅱ,Ⅲ : 멀티콥터의 현재위치가 목표점 반경에 들어 온다면 수평 속도 제한 값($V_{xy,\lim}$)과 요구 속도 중 작은 값을 사용한다. 여기서 속도 제한 값()은 목표점 반경에서 수평 속도 최댓값을 가지면서 일정하게 감소하여 목표점에서는 0.01[m/s]의 속도를 가진다.

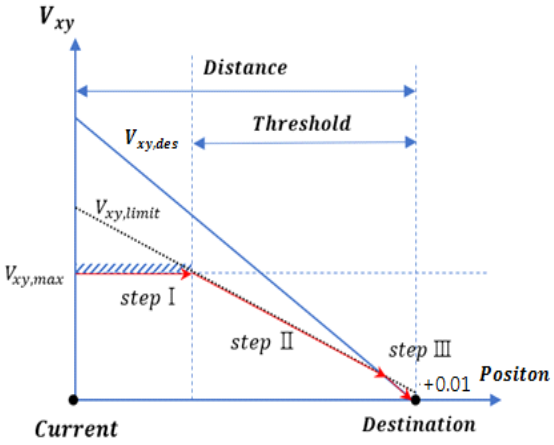


Fig. 7 Limit the horizontal velocity of the desired velocity

3.1.3 요구 가속도 (a_{des}) 생성

Equation 5와 같이 제한된 요구 속도($V_{des,sat}$)와 현재 속도 의 오차에 대해 PID 제어법칙을 적용한다. 여기서 생성된 요구 가속도(a_{des})에는 제자리 비행을 위해 필요한 중력 가속도 값 $[0,0,-g]$이 더해진다.

$$a_{des} = K_{P_v}[V_{des,sat} - V] + K_{I_v}\int [V_{des,sat} - V] - K_{D_v}\dot{V} + \begin{bmatrix} 0 \\ 0 \\ -g \end{bmatrix} \quad (5)$$

3.1.4 요구 가속도 (a_{des}) 제한

요구 가속도의 크기와 기울기는 사용자가 지정한 허용 기울기($tilt_{\max}$)와 요구 가속도 최댓값으로 제한이 된다. 여기서 허용 기울기는 Fig. 8과 같이 z축 방향 $\hat{z}_I$에서 최대로 기울어진 정도를 의미하며, 요구 가속도 최댓값은 사용자가 변수로 지정한 일정한 값을 이용한다. 사용되는 좌표평면은 x-y평면에 수직이고 요구 가속도 방향벡터가 포함된 평면을 기준으로 한다.

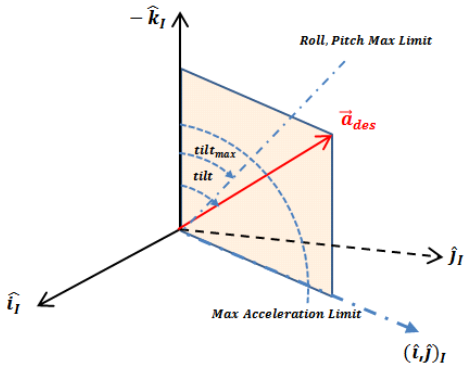


Fig. 8 Schematic of desired acceleration

해당 평면을 기준으로 원점에서 각각의 점(A~C)을 이은 선을 요구 가속도 방향벡터라 할 때 각각의 상황에 대해 가속도 제한은 다음과 같다(Fig. 9).

case Ⅰ : 최대 요구 가속도 크기와 허용 기울기에 벗어나지 않은 구간이므로 가속도 제한이 적용되지 않는다.

case Ⅱ : 점 A는 최대 요구 가속도 크기와 허용 기울기가 크게 벗어난 유형으로 우선 수직방향으로만 가

속도를 발생시켜 고도를 제어한다.

case Ⅲ : 점 B의 요구 가속도 방향벡터는 최대 가속도 크기를 벗어난 유형으로 수평방향의 가속도 성분을 줄임으로써 요구 가속도 방향벡터의 기울기 및 요구 가속도 최댓값을 조절한다.

case Ⅳ : 점 C로의 요구 가속도 방향벡터는 최대 허용각을 크게 벗어난 유형으로 기울기를 제한하기 위해 수평방향의 요구 가속도의 크기를 줄인다.

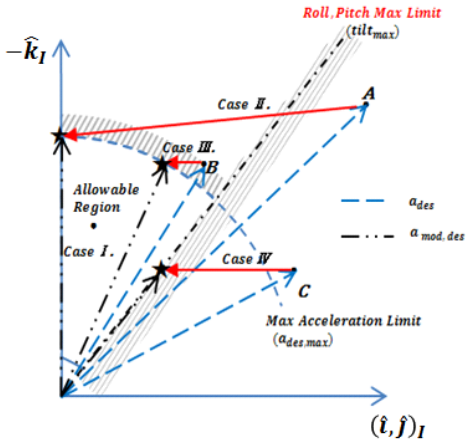


Fig. 9 Limit the required acceleration

3.2 오일러 각 변화 값 생성

위치제어기의 입력으로 사용되는 오일러 각 변화 값 $[\phi_{B,error}, \theta_{B,error}, \psi^*_{B,error}]$을 구하기 위해 Fig. 10과 같이 현재위치, 목표점, 요구가속도, 현재 가속도를 이용한다.

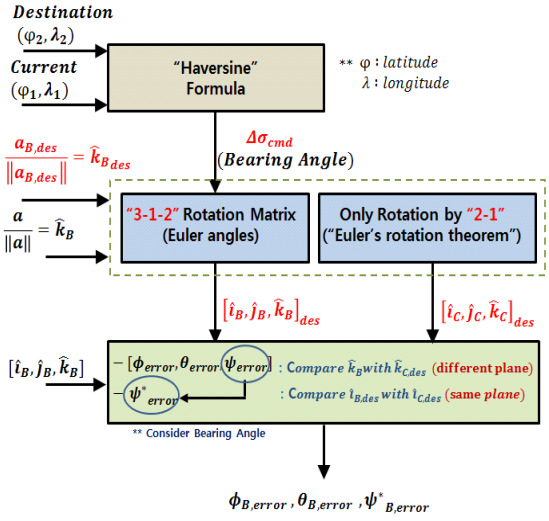


Fig. 10 Generation of required angle errors

오일러 각 변화 값 $[\phi_{error}, \theta_{error}, \psi_{error}]$은 요구 가속도의 단위벡터인 $[i_I, j_I, k_I]_{des}$와 현재 가속도의 단위벡터인 $[i_I, j_I, k_I]$ 간의 외적을 이용하여 계산한다. 여기서 ψ_{error}는 2-1-3 회전변환 특성 상 기체 좌표계의 z축 방향과 2-1 회전 변환된 좌표계의 z축 방향이 동일하므로 비행체의 시선 각이 고려되지 않는다. 따라서 시선 각이 고려된 요 각(ψ^*_{error})를 유도하기 위해 2-1 회전변환($\theta_{error} - \phi_{error}$)된 좌표평면 C에서의 x축 방향 단위벡터 $\hat{i}_{C,des}$와 기체좌표계의 x축 방향 단위벡터 $\hat{i}_{B,des}$를 이용한다.

3.2.1 좌표계 C 의 방향벡터 $[\hat{i}_{C,des}, \hat{j}_{C,des}]$

먼저 x축 방향벡터 $\hat{i}_{C,des}$을 구하기 위해 오일러 변화 값 $\phi_{B,error}, \theta_{B,error}$을 이용한다. Fig 11과 같이 $\phi_{B,error}, \theta_{B,error}$ 변화를 하나의 각도 $\Omega(\phi,\theta)$ 변화로 생각하면 회전축(Rotation Axis)을 법선벡터로 하는 평면위에 단위벡터 $\hat{i}_I$와 $\hat{i}_{C,des}$가 놓이게 된다. $\hat{i}_{C,des}$는 Eq. 7과 같이 지면좌표계의 x축방향벡터 $\hat{i}_I$ 와 회전평면의 단위 벡터 basis1, basis2, 회전변환 각 $\Omega(\phi,\theta)$으로 표현 할 수 있다. 여기서 회전축은 요구 가속도와 현재 가속도의 외적으로 유도된 결과이며, $\Omega(\phi,\theta)$은 두 벡터 사이의 각으로 정의한다. 다음으로 y축 방향벡터 $\hat{j}_{C,des}$은 요구 가속도 의 단위벡터와 좌표계의 C의 x축 방향벡터 $\hat{i}_{C,des}$의 외적을 이용하여 유도한다.

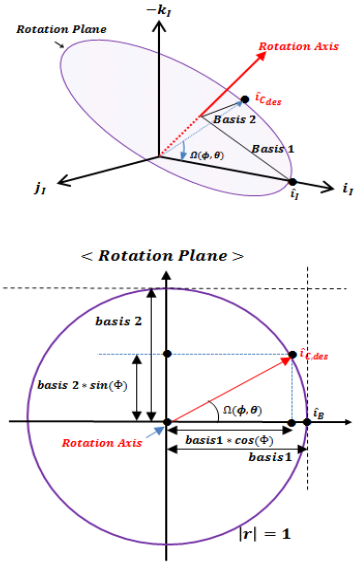
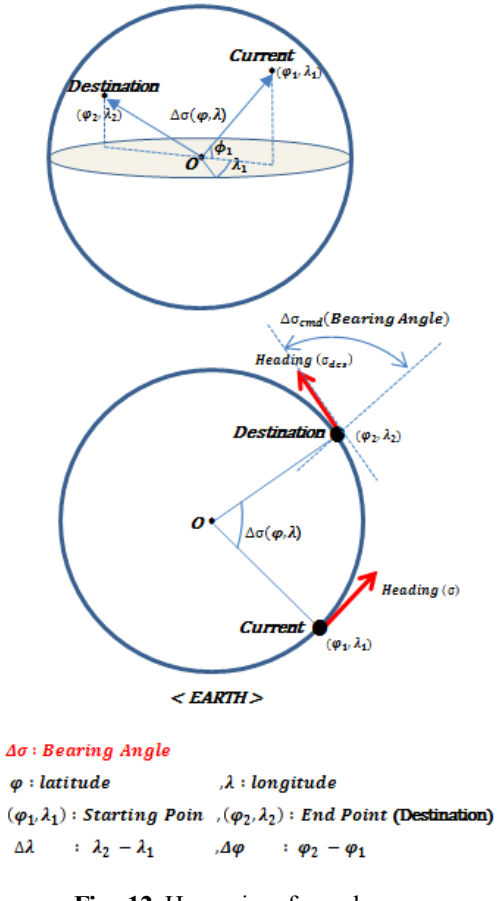


Fig. 11 Euler's rotation theorem

$$\hat{i}_{C,des} - \hat{i}_B =$$
$$basis2 \times \sin(\Omega(\phi,\theta)) + basis1 \times (1 - \cos(\Omega(\phi,\theta)))$$
$$j_{C,des} = \frac{a_{des}}{\|a_{des}\|} \times i_{C,des} = k_{B,des} \times i_{C,des}$$
$$where, \quad (6)$$
$$\Omega(\phi,\theta) = \tan^{-1}\left(\frac{\|k_I \times k_{B,des}\|}{\|k_I \cdot k_{B,des}\|}\right)$$
$$basis2 = (axis \times \hat{i}_I), \quad basis1 = basis2 \times axis$$
$$axis = k_{B,des} \times k_I$$

3.2.2 기체 좌표계 방향벡터 $[\hat{i}_{B,des}, \hat{j}_{B,des}, \hat{k}_{B,des}]$

기체 좌표계 방향벡터를 유도하기 위해 3-1-2 회전 변환을 이용한다. 우선 3(Ψ)의 회전 각도를 구하기 위해 Fig. 12에서 보이는 바와 같이 "Haver Sine formula"를 적용하여 현재의 위치와 목표점 위치 사이의 시선 각 변화($\Delta\sigma$)를 구한다[7].



$\Delta\sigma$: Bearing Angle
φ : latitude ,λ : longitude
(φ_1, λ_1) : Starting Poin ,(φ_2, λ_2) : End Point (Destination)
$\Delta\lambda$: $\lambda_2 - \lambda_1$,$\Delta\varphi$: $\varphi_2 - \varphi_1$

Fig. 12 Haversine formula

멀티콥터가 지구와 같이 큰 구 위에 놓일 경우 목표점까지의 요구 시선 각 변화는 지구 중심을 기준으로 현재 위치 점과 목표점 사이의 각과 동일하다. 이에 위도와 경도표기를 각각 φ, λ로 경도 차이를 $\Delta\lambda$로 표기하면 시선 각 변화는 Eq. 7과 같이 표현 할 수 있다.

$$\Delta\sigma_{cmd}(\varphi,\lambda) =$$
$$\tan^{-1}\left(\frac{\sin(\Delta\lambda)\cos(\varphi_2)}{\cos(\varphi_1)\sin(\varphi_2) - \sin(\varphi_1)\cos(\varphi_2)\cos(\Delta\lambda)}\right) \quad (7)$$

다음으로 시선 각 변화를 이용하여 3(Ψ)회전 변환 시, 나타나는 좌표계의 단위벡터를 $[\hat{i}_{\sigma,des}, \hat{j}_{\sigma,des}, \hat{k}_{\sigma,des}]$라고 할 때, 이는 Eq. 8과 같이 유도할 수 있다.

$$\hat{i}_{\sigma,des} = [\cos(\Delta\sigma_{cmd}), \sin(\Delta\sigma_{cmd}), 0]^T \quad (8)$$

Equation 8의 결과와 요구 가속도의 단위벡터와의 외적을 이용해 Eq. 9와 같이 기체 좌표계의 단위벡터를 유도할 수 있다[7].

$$\hat{j}_{B,des} = \hat{k}_{B,des} \times \hat{i}_{\sigma,des}$$
$$\hat{i}_{B,des} = \hat{j}_{B,des} \times \hat{k}_{B,des}, \quad \hat{k}_{B,des} = \frac{a_{des}}{\|a_{des}\|} \quad (9)$$

3.2.3 오일러 각 변화 값 $[\phi_{B,error}, \theta_{B,error}, \psi_{B,error}]$

먼저 롤 각과 피치 각 변화는 Eq. 10와 같이 요구 가속도의 단위벡터와 현재 가속도의 단위벡터간의 외적을 이용하여 계산한다.

$$e_B = [\phi_{B,error}, \theta_{B,error}, \psi_{B,error}] = R_B \cdot \left(\frac{z_{B,des} \times z_B}{\|z_{B,des} \times z_B\|}\right) \quad (10)$$

다음으로 시선 각이 고려된 요 각($\psi^*_{B,error}$)는 Eq. 11과 같이 2-1 회전 변환된 좌표계성분 $\hat{i}_{C,des}$ 와 기체좌표계의 $\hat{i}_{B,des}$의 사이의 각으로 유도한다[8].

$$\psi^*_{error} = \tan^{-1}\left(\frac{\|\hat{i}_{B,des} \times \hat{i}_B\|}{\|\hat{i}_{B,des} \cdot \hat{i}_B\|}\right) \quad (11)$$

3.3 자세 제어기 구조

자세 제어기 구조는 Fig. 14과 같이 요구 각속도(ω_{des}) 생성과 요구 각 가속도 생성($\dot{\omega}_{des}$)으로 구성된다.

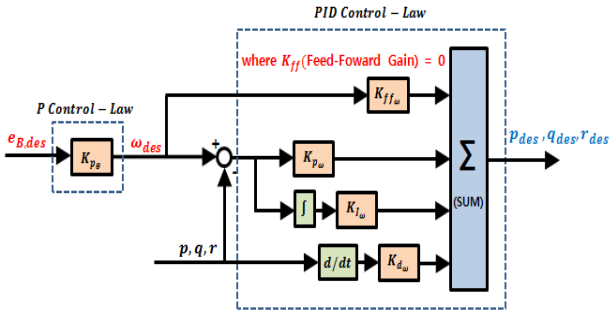


Fig. 13 Attitude controller model

3.3.1 요구 각속도(ω_{des}) 생성

Equation 12와 같이 오일러 각 변화 값 에 이득(K_{P_θ})을 곱하여 기체좌표를 기준으로 한 요구 각속도를 생성한다.

$$\omega_{des} = K_{P_\theta} e_B = K_{P_\theta} \begin{bmatrix} \phi_{error} \\ \theta_{error} \\ \psi^*_{error} \end{bmatrix} \tag{12}$$

3.3.2 요구 각 가속도 생성($\dot{\omega}_{des}$)

Equation 13와 같이 기체좌표계 각속도(ω)와 요구 각속도(ω_{des})를 비교하여 오차에 대해 PID 제어법칙을 하여 요구 각가속도를 생성한다.

$$\dot{\omega}_{des} = K_{P_\omega}[\omega_{des} - \omega] + K_{I_\omega}\int [\omega_{des} - \omega] - K_{D_\omega}\dot{\omega} + K_{ff_\omega}\omega_{des} \tag{13}$$

4. 시뮬레이션

4.1 시나리오 구성

시뮬레이션 시나리오는 다음과 같이 구성하였다. 초기 위치 [0,0,4]m에서 호버링 중이던 기체가 [1,1,5]m로 직선비행 하도록 구성하였으며 이때 시선 각은 진행방향을 추종하도록 하였다. 시뮬레이션 결과는 Equation 14-15에 나타내었다.

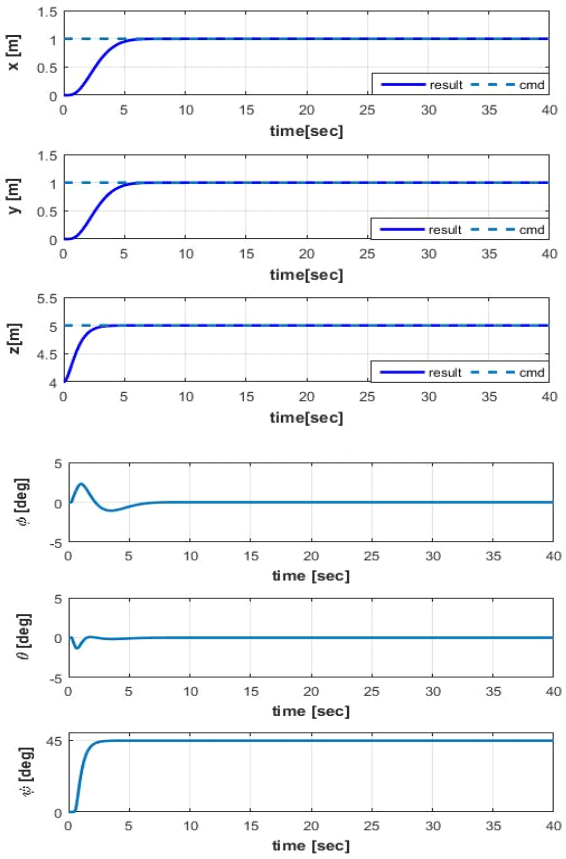


Fig. 14 Simulation Result(Position & Attitude) using MATLAB

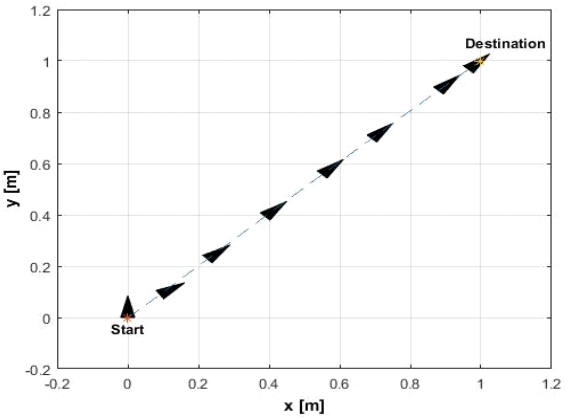


Fig. 15 Line of sight change Result (X-Y plane)

4.2 제어기 성능 비교

기존 멀티콥터에서 적용되고 있는 요구 오일러 각 유도방법은 Equation 14와 같이 비행체의 운동방정식에서 호버링을 기준으로 선형화 하여 요구 가속도와 요구 오일러 각 관계식을 유도한다[6].

$$\phi_{error} = \frac{1}{g}(a_{x,des}\sin(\psi^*_{error}) - a_{y,des}\cos(\psi^*_{error}))$$
$$\theta_{error} = \frac{1}{g}(a_{x,des}\cos(\psi^*_{error}) + a_{y,des}\sin(\psi^*_{error})) \quad (14)$$

이와 달리 Pixhawk의 제어기 구조는 요구 각도로 변환하기 위해 좌표계간의 관계를 이용한다. 이는 Fig. 16에서와 같이 4.1에서 제시한 시나리오를 기존제어기와 동일한 제어이득을 이용하여 비행 시 더욱 안정적인 비행을 할 수 있는 장점이 있다.

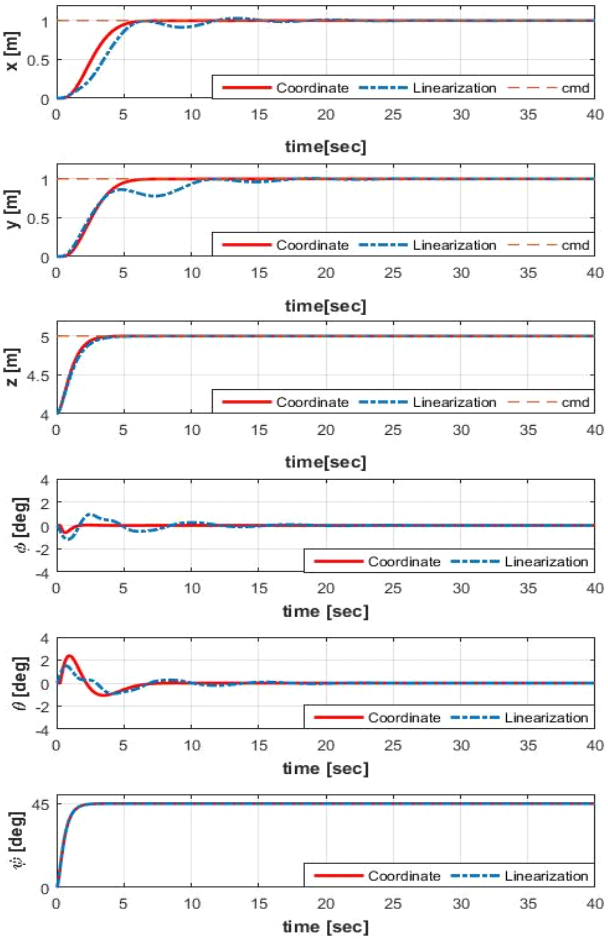


Fig. 16 Simulation Comparison Result (Position & Attitude) using MATLAB

5. 결 론

비행 컨트롤러인 Pixhawk의 자세 및 위치제어기가 PID 제어 법칙이 적용되고 있으며, 여기서 위치제어기는 기체의 급격한 거동을 방지하기 위해 요구속도와 요구 가속도에 대해 적절한 관계식을 이용하여 제한이 되는 것을 확인하였다. 특히 Pixhawk는 위치제어기로부터 생성된 요구 가속도를 오일러 각 변화 값으로 변환하는 것에 대해 기존에 사용되는 방식과 큰 차이를 보인다. 기존 변환 방식은 호버링을 기준으로 선형화 하여 요구 가속도와 오일러 각 관계식을 유도한다. 이와 달리 Pixhawk는 좌표계간의 관계를 이용해 변화하고 있으며, 이는 기존의 변환방식과 비교하여 큰 각 변화에 대한 요구 자세 각을 제시할 수 있는 장점이 있다.

후 기

본 논문은 2016년도 정부(미래창조과학부)의 재원으로 한국연구재단의 지원을 받아 수행된 사업임 (No.2016M1B3A1A03937725)

본 논문은 국방과학연구소 생존성 기술 특화연구센터의 사업으로 지원받아 연구되었음. (계약번호 UD150013ID)

References

[1] Taejin Chang, "Regulatory environment and structural change of UAV industry." *Journal of Aerospace System Engineering 9.3,* pp. 17-22, 2015.

[2] Seung Jae Hwang, Tae Hwan Cho, Yang Won Kim, and Jin Deog Chung, "Multi-copter Wind-tunnel Test." *Journal of Aerospace System Engineering 11.6,* pp. 10-16, 2017.

[3] https://github.com, "Px4 Pro Autopliot Software," [Internet]. Github , [cited 2017 Nom. 3.] (https:// github.com/PX4/Firmware/)

[4] Irobotnews.com, "Be open to the sky with open source," [Internet]. Robot news, [cited 2017 Nov. 3.] (http:// www.irobotnews.com/news/articleView.html?idxno=7168)

[5] Mellinger, Daniel, "Minimum snap trajectory generation and control for quadrotors." Robotics and Automation (ICRA), 2011 IEEE International Conference on. IEEE, 2011.

[6] Robusto, C. Carl. "The cosine-haversine formula." *The American Mathematical Monthly 64.1,* pp. 38-40, 1957.

[7] Baker, Martin J. "Maths-Axis Angle to Matrix." Abgerufen am 21 (2016).

[8] Weiss, Stephan "Monocular-SLAM-based navigation for autonomous micro helicopters in GPS-denied environments." Journal of Field Robotics 28.6 (2011): 854-874.