ROS를 이용한 드론의 상태 추정과 제어기 설계

김관수*, 강현호*, 이상수*, 유성현*, 이동훈*, 이동규*, 김영은**, 안춘기*
*고려대학교 전기전자공학과
**고려대학교 메카트로닉스
e-mail : ks9138@korea.ac.kr

State Estimator and Controller Design of an AR Drone with ROS

Kwan-Soo Kim*, Hyun-Ho Kang*, Sang-Su Lee*, Sung-Hyun You*,
Dhong-Hun Lee*, Dong-Kyu Lee*, Young-Eun Kim**, Choon-Ki Ahn*
*Dept. of Electrical Engineering, Korea University
**Dept. of Mechatronics, Korea University

요 약

본 논문에서는 ROS (Robot Operating System)에 대해서 소개하고 ROS를 이용해 드론의 제어기와 필터를 구현해본다. 드론이 강인한 성능을 보이기 위해서는 기체의 상태에 대한 더 정확한 추정이 필요하다. 드론이 기체좌표계로 출력하는 각 축(x축, y축, z축)에 대한 선속도, 선가속도를 더 정확히 추정하기 위해 칼만 필터를 설계하며 칼만 필터를 통과한 상태 변수를 제어 입력으로 하는 PID (Proportional Integral Derivative) 제어기를 설계한다. 실험적인 부분에서는 제어기와 자율 주행 알고리즘을 접목시켜 드론이 자신의 상태를 추정하고 알고리즘을 순차적으로 진행하는 과정을 살펴본다. 마지막으로 알고리즘을 통해 드론의 임무 수행 여부를 살펴보고 정밀한 제어를 위한 추가적인 제어기 설계 방법과 연구 방향을 제시하고자 한다.

1. 서론

ROS는 Robot Operating System의 약자로 로봇 운영 체제를 말한다. 로봇 응용 프로그램을 개발하는 플랫폼으로 개발되었기 때문에 개발을 위한 다양한 라이브러리와 개발 도구, 장치 드라이버, 하드웨어 추상화, 오픈 소스 패키지 등을 제공한다.[1]

ROS는 오픈 소스 패키지를 제공하기 때문에 많은 사람들이 다양한 패키지를 사용할 수 있으며 버그가 발견되면 수정해서 다시 배포할 수 있다. 이런 과정은 전세계 단위로 로봇 응용프로그램 개발에 빠른 성장을 촉진한다. 또 다양한 라이브러리와 개발 도구, 오픈 소스 패키지의 제공은 협업을 가능하게 한다. 예를 들어 자율 청소 로봇을 만들기 위해서 컴퓨터 비전을 연구하는 연구실, 제어를 연구하는 연구실, 실내 환경 맵핑을 연구하는 연구실이 협업할 수 있다. 이러한 장점들로 인해 많은 기업과 연구실에서 ROS를 사용하고 있으며 앞으로도 여러 로봇 소프트웨어 플랫폼 중 가장 많이 사용될 것으로 보인다.

드론은 무선전파로 조종할 수 있는 무인 항공기를 말한다. 처음에는 군사용 무인항공기로 개발됐지만 지형의 영향을 받지 않고 자유롭게 움직일 수 있다는 드론의 장점이 이후에 발생한 드론 가격의 하락, 크기의 다양성과 결합하여 수요의 급증을 불러일으켰다. 현재는 다양한 분야에서 드론을 활용하기 위한 연구가 진행되고 있다. [2]

실제로 수송 분야에서는 드론 택배 서비스를 위한 군집 비행 연구가 진행되고 있다. 기상 분야에서는 드론을 이용해 대기 오염 정도를 실시간으로 파악하려는 연구가 진행되고 있으며 농업분야에서는 농약과 비료를 살포하는데 드론을 이용하려는 연구가 진행되고 있다. 뿐만 아니라 재난 안전 분야에서는 화재 발생 시 진압하기 위한 소방 드론에 관한 연구가 진행되고 있다. [3, 4]

드론을 더욱 강인하게 제어하기 위해서는 우선 드론의 상태에 대한 정확한 파악이 필요하다. 드론에 부착되어 있는 센서는 센서 자체가 가지고 있는 부정확성으로 인해 드론의 상태를 정확하게 측정할 수 없다. 때문에 센서로부터 얻어낸 상태정보 하나만 가지고 드론의 상태를 정확하게 파악했다고 말할 수 없다.

본 논문에서는 필터를 이용해 드론의 상태를 더욱 정확히 추정하고 원하는 목표지점까지 정확히 움직이는 제어기를 설계해 ROS로 구현한다. 드론이 출력하는 상태 관련 데이터를 칼만 필터에 통과시켜 상태 추정의 정확도를 높였다. 마지막으로 사용한 플랫폼 ROS에 대해 고찰해본다.

본 논문은 본론, 실험, 결론으로 구성되어 있다. 본론에서는 드론을 구동시키기 위한 ROS와 임무 수행을 위해 필요한 칼만 필터, PID 제어에 관하여 다룬다. 실험에서는 AR Drone을 사용한 구현 및 결과 고찰을 하였다.

2. 본론

2.1 드론 좌표계

드론 좌표계에는 관성 좌표계와 기체 좌표계가 있다. 관

성 좌표계는 사람의 관점을 기준으로 하는 좌표계이고 기체 좌표계는 드론의 관점을 기준으로 하는 좌표계이다. 드론은 기체 좌표계를 기준으로 움직이기 때문에 드론으로부터 받는 데이터는 기체 좌표계 기반 데이터이며 드론에게 보내주는 입력 값 또한 기체 좌표계 기반 데이터여야 한다. 따라서 제어를 쉽게 하기 위해서는 드론으로부터 받은 데이터를 관성 좌표계로 변환한 후에 사용하고 다시 기체 좌표계로 변환하여 드론에게 보내주어야 한다. 좌표계 변환을 위해 오일러 각도를 사용한다. 오일러 각도는 두 좌표계의 각 축사이의 각도를 나타낸다.

(그림 1) 드론 관성 좌표계와 기체 좌표계

(그림 1)에서 보이는 $roll\,\phi$, $pitch\,\theta$, $yaw\,\psi$ 각도는 관성 좌표계와 기체 좌표계 사이의 x축, y축, z축 사이의 각도를 나타낸다.

$$R_x(\phi)=\begin{bmatrix}1 & 0 & 0\\ 0 & \cos(\phi) & \sin(\phi)\\ 0 & -\sin(\phi) & \cos(\phi)\end{bmatrix}, \tag{1}$$

$$R_y(\theta)=\begin{bmatrix}\cos(\theta) & 0 & -\sin(\theta)\\ 0 & 1 & 0\\ \sin(\theta) & 0 & \cos(\theta)\end{bmatrix}, \tag{2}$$

$$R_z(\psi)=\begin{bmatrix}\cos(\psi) & \sin(\psi) & 0\\ -\sin(\psi) & \cos(\psi) & 0\\ 0 & 0 & 1\end{bmatrix}. \tag{3}$$

식 (1), (2), (3)은 각각 x축, y축, z축으로 관성 좌표계를 기체 좌표계로 변환하는 행렬이다. 위 3개의 식을 모두 적용해 한 번에 변환하는 좌표 변환 행렬은 아래 식 (4)와 같다. c는 cos을 s는 sin을 나타낸다.

$$R = R_x(\phi)\,R_y(\theta)\,R_z(\psi), \tag{4}$$

$$=\begin{bmatrix}c(\theta)c(\psi) & c(\theta)s(\phi) & -s(\theta)\\ s(\psi)s(\theta)c(\psi)-c(\phi)s(\psi) & c(\phi)c(\psi)+\,s(\phi)s(\theta)s(\psi) & s(\phi)c(\theta)\\ c(\phi)s(\theta)c(\psi)+s(\phi)s(\psi) & s(\theta)c(\phi)s(\psi)-\,s(\phi)c(\psi) & c(\theta)\,c(\phi)\end{bmatrix}.$$

기체 좌표계를 관성 좌표계로 회전 변환하는 행렬은 R로 나타낸다. 회전 변환 행렬 R에 대해 아래와 같이 나타낼 수 있다.

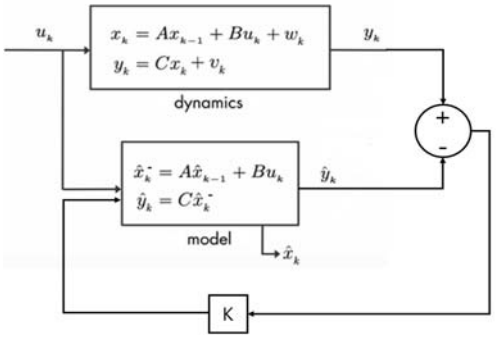
$$R^T = R_z(\psi)^T R_y(\theta)^T R_x(\phi)^T, \tag{5}$$

$$=\begin{bmatrix}c(\theta)c(\psi) & s(\psi)s(\theta)c(\psi)-c(\phi)s(\psi) & c(\phi)s(\theta)c(\psi)+s(\phi)s(\psi)\\ c(\theta)s(\phi) & c(\phi)c(\psi)+\,s(\phi)s(\theta)s(\psi) & s(\theta)c(\phi)s(\psi)-\,s(\phi)c(\psi)\\ -s(\theta) & s(\phi)c(\theta) & c(\theta)\,c(\phi)\end{bmatrix}.$$

기체 좌표계를 관성 좌표계로 회전 변환하는 행렬은 식 (5)와 같이 나타낼 수 있다.

2.2 Kalman filter 알고리즘

칼만 필터는 물체의 상태를 나타내는 수학적 모델과 실제로 물체에 부착되어 있는 센서로부터 받은 상태값을 이용해 물체의 상태를 더 정확히 추정하는 필터이다.



(그림 2) Kalman Filter block diagram

칼만 필터는 앞의 식들을 이용해 예측과 업데이트를 진행하며 이를 통해 물체에 대한 더 정확한 상태 추정을 가능하게 한다. 본 논문에서는 드론으로부터 얻어낸 드론의 기체 좌표계 기반 선속도 데이터를 칼만 필터에 보내주어 드론의 선속도 데이터를 보정했다. 보정된 선속도 데이터를 이용해 드론이 움직인 거리를 계산하고 이를 출력하도록 모델링해 거리 기반 레퍼런스를 사용할 수 있도록 했다.

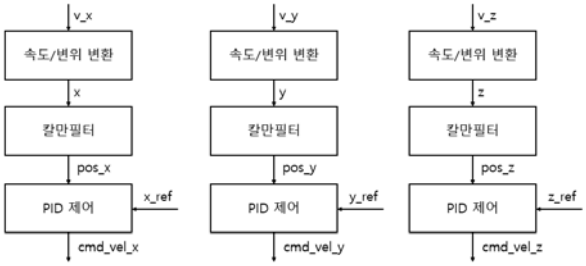
3. 실험

3.1 실험 환경 및 방법

앞에서 어떤 방법을 사용하여 드론의 ROS를 활용한 상태 추정과 자율주행을 구현할지를 결정하였고, 제안된 방법들을 활용하여 실험을 진행했다. 먼저 데이터의 흐름 측면에서 접근해보도록 하겠다. 데이터가 가공되는 과정과 노드마다 입력과 출력을 정하고, 알고리즘에 대해 설명하겠다. 실험의 목적은 드론이 특정 좌표에 위치할 수 있도록 하는 것이며, 이를 ROS로 구현하였고 그 과정에서 칼만 필터와 PID 제어를 활용하였다. 그리고 실제 드론에 적용 시켰을 때의 결과에 대해서 살펴보고자 한다.

(그림 3)은 드론에서 출력되는 각 좌표의 속도 값을 받아 속도/위치 변환 및 칼만 필터를 활용하여 드론의 좌표를 추정하고 그 좌표를 활용해 드론을 제어하는 과정을 나타낸 것이다.

가장 먼저 드론에서 출력되는 기체 좌표계 속도 값을 위치 값으로 변환한다. 속도 값을 10Hz로 받아오기 때문



(그림 3) 데이터 처리 알고리즘

에 시간 간격인 0.1초와 속도 값을 곱하고 누적하는 방식으로 변위를 구해낸다.

칼만 필터에서 출력된 위치와 레퍼런스 위치의 차이를 error라고 했을 때, error 값이 일정 범위 안에 들기 전까지 PID 제어를 통해 드론이 주행할 방향 및 빠르기를 결정한다. 이 때 PID 제어에 사용되는 시간은 싸이클이 10Hz이므로 0.1초이고, 제어에 사용되는 계수들은 실험을 통하여 찾도록 하였다.

이러한 과정을 x, y, z에 대하여 각각 병렬적으로 진행될 수 있게끔 하였으며, ROS를 사용하면 각 노드 사이에서 발행하고 구독할 메시지들을 미리 선언하여 일 방향 데이터의 흐름을 혼동 없이 구현해 내기 편리하다. 따라서 각 좌표축 별로 노드와 메시지를 선언한 후, ROS의 런치 기능을 사용하면 좌표별 차이 없이 동시에 실행이 가능하다. 즉, 동시적인 x, y, z 축 별 병렬적 제어를 할 수 있다.

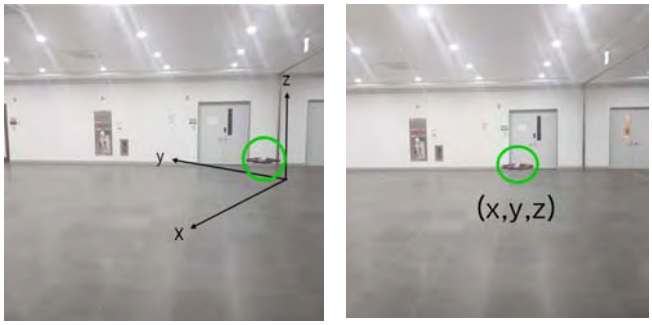
군집 드론 제어를 하기 위해서는 다수의 드론에서 정보를 받아와 동시에 처리할 수 있어야 한다. ROS에서 각 드론을 모듈화시켜 하나의 PC에서 다수의 드론을 제어할 수 있다.

PUTTY 프로그램을 사용하여 드론에 접속하여 vi 편집기를 통하여 드론의 IP 주소를 변경할 수 있다. 이러한 방식으로 두 개 이상의 드론 IP 충돌 문제를 해결할 수 있다. 각각의 드론을 서로 다른 랜카드에 연결시켜 드론의 정보를 받고, 동시에 두 개의 드론에 접속하여 제어할 수 있다. (그림 6)는 두 개의 드론을 동시에 ROS를 활용하여 제어한 모습을 나타낸다.

3.2 실험 결과

본 논문에서 설계한 ROS를 활용한 드론 제어는 드론에 내장된 IMU에서 나오는 선속도 데이터만으로 진행되었다. 하나의 센서에서 나오는 값만을 활용했기 때문에 두 개 이상의 센서로부터 데이터를 받아 제어를 했을 때에 비해 연산 시간은 단축되었으나 정확성이 떨어지는 단점이 있다. 이를 극복하기 위해서 optical flow등을 활용하여 다른 방법으로도 위치 정보를 얻어 낼 수 있도록 한 후 두 개 이상의 센서에서 나온 데이터를 가공하여 위치 정보를 얻는다면 더 정확한 상태 추정이 가능할 것이다.

선속도에 대한 제어기만을 설계했지만 각속도, 각도, 선가속도, 위치에 대한 제어기 등 다양한 제어기를 설계할 수 있다. 본 논문에서 위치에 대한 제어기도 설계해 두 제

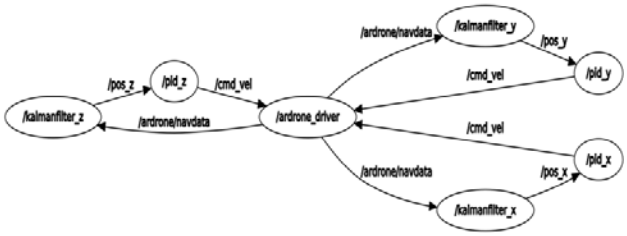


(그림 4)목표지점으로의 드론 이동

앞선 본론에서 roslaunch 명령어를 사용하면 여러 노드를 동시에 실행시킬 수 있다고 하였다. 이 roslaunch를 활용하면 각 축마다 순차적인 각 reference 좌표로의 이동이 아닌, 각 좌표 별 처리한 속도 값을 동시에 갖도록 하여 지정된 공간 상의 위치로 도달하도록 할 수 있다.

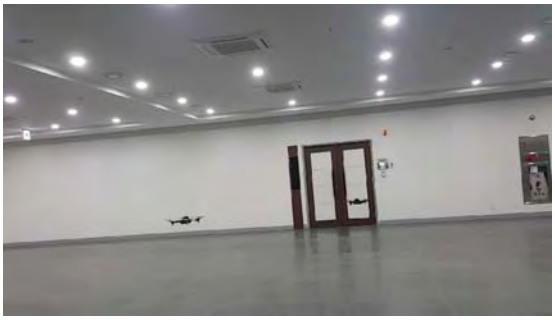
런치파일(launch_control.launch)을 실행하면 pid_x, pid_y, pid_z 노드를 모두 실행할 수 있도록 한다. 드론을 이륙시키고 각 축 별로 칼만 필터 노드를 실행시켜 Hovering 되어 있는 드론에 각 축 별 속도 값을 넣어준다.

실험결과 지정해 준 (x,y,z) 레퍼런스 위치로 드론이 (그림 4)와 같이 목표로 이동하였고, 그 각 축 별 error가 일정 범위 이내에 들어왔을 때 드론이 다시 그 위치를 유지한 채 Hovering을 하였다. 동시적 명령 수행이 성공적으로 진행됐음을 알 수 있었다.



(그림 5) node의 연결도

ROS에는 rqt graph라는 실행되고 있는 모든 노드와 노드 사이에서 전달되는 메시지의 흐름을 볼 수 있도록 하는 그래프가 지원된다. rqt graph 명령어를 실행하여 관찰해본 3-axis 동시적 실험의 rqt 그래프에서 세 개의 병렬적인 축 별 데이터 처리가 이루어지는 것을 확인할 수 있었다.



(그림 6) 멀티 드론 제어

어기를 함께 사용했다면 위치와 선속도 둘 다에 대한 목표값이 설정되기 때문에 훨씬 더 좋은 제어를 할 수 있었을 것이다.

4. 결론

이 실험에서는 ROS를 드론의 위치 추정과 속도 제어에 사용하였다. 이 과정에서 ROS의 프로그래밍 언어 비의존성, 독립적인 노드 또는 패키지의 동시적, 병렬적 처리 그리고 bag파일을 활용한 데이터 재활용의 장점에 대하여 알 수 있었다. ROS를 활용하면 두 개 이상의 드론을 제어하는 데에도 매우 편리할 것이다. 드론의 IP를 각각 다르게 하여 각 드론의 제어를 할 수 있는 것 뿐만 아니라, 각 드론을 노드화 하여, 드론 사이에서의 메시지 교환으로 다수의 드론 사이의 상호 정보 교환 및 군집적 제어가 가능하다. 드론뿐만 아니라 로봇, 카메라, 다양한 센서도 ROS를 활용하면 노드화하여 각 기기들 간의 정보 교환이 가능하기 때문에 인공적 기기들의 생태계를 구성하는 것도 가능할 것이다. 추후에 두 개 이상의 드론의 동시 제어 및 상호 간의 정보 교환을 통한 포메이션 드론 연구를 진행할 계획이다.

참고문헌

[1] Michael Cashmore, Maria Fox, Derek Long, Daniele Magazzeni, and Bram Ridder. ROSPlan: Planning in the Robot Operating System. AAAI Publications. April. 2015.

[2] R clarke. Understanding the drone epdimic. computer law & security review. June. 2014.

[3] A Sanjab, W Saad, T Basar. Prospect theory for enhanced cyber-physical security of drone delivery systems: A network interdiction game. IEEE ICC. Feb. 2017.

[4] B Laszlo, R Agoston, Q Xu. Conceptual Approach of Measuring the Professional and Economic Effectiveness of Drone Applications Supporting Forest fire Management. Elsevier. 2018.

[5] Morgan Quigley, Brian Gerkey, William D. Smart. Programming Robots with ROS A PRACTICAL INTRODUCTION TO THE ROBOT OPERATING SYSTEM. O'REILLY. Dec. 2015.

[6] 표윤석. ROS 로봇 프로그래밍. Ruby paper. March. 2015.

[7] Mahtani. Effective Robotics Programming with ROS, 3/E. Packt Publishing. Dec. 2016.

[8] Artur Banach. Visual control of the Parrot drone with OpenCV, Ros and Gazebo Simulator. June. 2016.

[9] N.L.M. Jeurgens. Identification and Control Implementation of an AR.Drone 2.0. Feb. 2017.

[10] Daniel P. Bovet, Marco Cesati. Understanding the Linux Kernel: From I/O Ports to Process Management 3rd Edition. O'REILLY. 2006.

[11] Karl J. Aström, Tore Hägglund. PID Controllers: Theory, Design, and Tuning 2nd Edition. ISA. 1995.

[12] Dan Simon. Optimal State Estimation: Kalman, H Infinity, and Nonlinear Approaches. WILEY. 2006.

[13] Greg Welch, Gary Bishop. An Introduction to the Kalman Filter. ACM, Inc. 2006.

[14] Katsuhiko Ogata. Modern Control Engineering 5th Edition. Pearson. Sep. 2009.