Self Ball Bouncing Quadrotor

최호광

서울대학교 공과대학 기계항공공학부

2018. 5.

초 록

일명 '드론'으로 잘 알려진 쿼드로터는 소형 비행체로 네 개의 로터를 통해 작동하며 그 기동성과 제어의 용이함으로 인해 제어시스템을 활용한 무인비행체로서 각광 받고 있다.

본 논문은 쿼드로터를 활용하여 스스로 공을 튀기는 시스템을 구축하는 방법론에 관한 논문이다. 가장 먼저 쿼드로터의 동역학적 요소를 파악하여 주어진 로터 입력에 대한 쿼드로터의 운동을 살펴본다. 또한 사용될 공의 운동방정식을 세우고 그에 따른 공의 궤적을 예측한다. 공은 실제로 하드웨어 실험 시 실시간으로 모션감지 센서를 통해 위치와 속도성분이 계산될 것이다. 이로써 미래 시점의 공의 위치를 파악하고 정해진 높이에서 계산된 다음 충돌위치와 시점을 계산한다. 공을 다시 충분한 높이로 쳐 올리기 위해 필요한 쿼드로터의 방향과 속도를 계산하고 앞서 계산된 충돌 위치와 시점을 활용하여 쿼드로터의 궤적을 생성한다. 여기서 계산된 궤적을 쿼드로터가 잘 따르도록 제어하는 제어기를 설계한다. 본 논문에서는 LQT제어기를 설계하여 계산된 궤적을 따르도록 제어해보았다.

스스로 공을 튀기는 쿼드로터를 구성하는 알고리즘은 MATLAB 2018을 사용하여 작성하였으며 Simulink를 이용해 시뮬레이션을 수행하였다.

목 차

1. 연구목표

쿼드로터를 활용하여 자동 제어를 통해 공을 반복하여 튀긴다는 것은 기본적으로 쿼드로터의 동역학적 특성에 대한 이해와 이를 제어하는 제어기에 대한 개념을 바탕으로 한다. 또한 공 튀기기를 성공적으로 수행한다는 것은 쿼드로터를 활용한 민첩한 움직임과 외부 사물에 대한 상대적인 운동을 더 높은 차원에서 가능하도록 할 것이다. 이 연구를 통해 쿼드로터를 제어하기위한 배경지식을 익히고 실제로 주어진 미션을 수행하는 제어 시스템을 구성하는 것을 연구 목표로 한다.

연구를 수행함에 있어 기본적으로 학습해야 할 요소는 다음과 같다.

① Quadrotor Dynamics
쿼드로터의 동역학적 특성을 이해하기 위하여 관성 좌표계와 쿼드로터의 동체에 대한 좌표계를 설정하고 쿼드로터에 작용하는 힘과 모멘트에 대한 운동방정식을 정리하고 각 변수에 대한 식을 유도한다. 이는 비선형 미분방정식으로 표현 될 것이다.

② Controller Design
제어기는 다양한 종류가 존재한다. 이때 학부 수준에 적합한 LQT(Linear Quadratic Tracker) 제어기를 통하여 쿼드로터를 제어한다. 기본적인 LQT 제어기에 대한 이론을 학습하고 이를 통한 제어 성능을 살펴 본다. LQT 제어기는 선형화된 식을 바탕으로 구성되므로 Quadrotor Dynamics에서 구한 운동방정식을 선형화하는 과정을 거친다. 또한 매트랩을 통하여 주어진 reference 값에 대하여 tracking을 하는 쿼드로터의 성능을 알아보고 각 변수의 시간에 따른 변화를 그래프로 그려본다.

③ Trajectory generation
공을 튀기기 위하여 쿼드로터는 반복적으로 공이 낙하하는 지점으로 이동하여야 하며 이는 짧은 시간(2초 이내)에 달성되어야 하므로 적절한 Trajectory를 선택하는 것이 필요하다. 또한 낙하하는 공을 다시 공중으로 튀기기 위하여 공과 만나는 지점의 쿼드로터의 속도성분의 방향이 매우 중요하다. 이를 위해서는 공이 낙하하는 지점에서 쿼드로터의 z축 성분의 속도가 커야한다. 따라서 단순히 공의 낙하지점으로 이동하는 것 뿐 만이 아니라 공과 만나는 지점의 쿼드로터의 운동성에 대한 조건을 만족하는 Trajectory를 생성해야 한다. 이를 위해 Trajectory generation에 대한 학습이 필요하다.

④ Matlab & Simulink program
실제 하드웨어 실험 이전에 시뮬레이션을 수행함에 있어 Matlab과 Simulink에 대한 기본적인 이해가 필요하다. Quadrotor Dynamics에서 구한 운동방정식을 코딩하고 주어진 입력에 대하여 실제 쿼드로터의 반응과 일치하는 반응을 보이는지 확인하는 과정을 거친다. 또한 LQT 제어기를 설계하여 Matlab에 입력하고 쿼드로터의 움직임을 제어해본다. 전체적인 시뮬레이션을 위하여 Simulink를 통하여 피드백 제어 루프를 설계하고 주어진 입력 값에 대한 시스템의 반응을 살펴본다. 또한 Matlab을 통하여 그래프를 그리고 제어 성능을 분석해 본다.

2. 연구수행 내용 및 결과

연구목표에서 설정하였던 Quadrotor Dynamics와 LQT controller design을 수행한 이후 Trajectory generation의 방법론을 익히고 이를 Matlab과 Simulink를 통하여 구현하였다.

가. Quadrotor Dynamics[1)]

쿼드로터는 기본적으로 4개의 로터의 추진력에 의하여 작동한다. 또한 4개의 로터는 Angular momentum을 상쇄시키기 위해 2개의 로터는 시계 방향으로, 또 다른 2개의 로터는 반시계 방향으로 회전한다. 이를 수식을 통해 표현하면 다음과 같다.

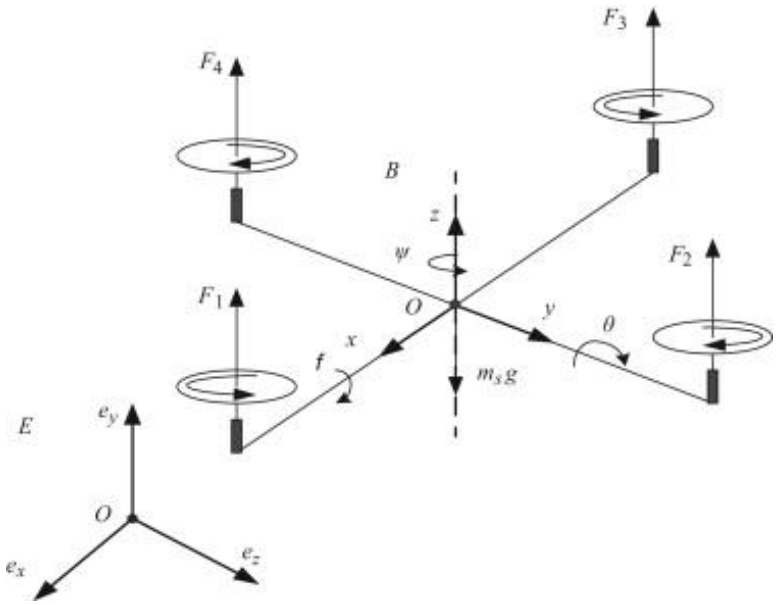

Fig. 1 - Quadrotor configuration

P, A를 각각 inertial frame에서의 쿼드로터의 위치, 각도 성분이라고 하고 V, Ω를 body frame에서의 쿼드로터 속도와 각속도라고 할 때, 외력에 대하여

$\Sigma F_{ext} = m\dot{V} + \Omega \times mV$, $\Sigma M_{ext} = J\dot{\Omega} + \Omega \times (J\Omega)$이고 $\dot{P} = R_t^{B\to I} V$, $\Omega = R_r^{I\to B}\dot{A}$이다.

여기서 $R_t^{B\to I}$는 Body Frame에서 Inertial Frame의 속도 변환(Translational) 행렬이며 $R_r^{I\to B}$는 Inertial Frame에서 Body Frame의 각속도(Rotational) 변환 행렬이다. 이는 다음과 같다.

$$R_t^{B\to I} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c\phi & -s\phi \\ 0 & s\phi & c\phi \end{bmatrix} \begin{bmatrix} c\theta & 0 & s\theta \\ 0 & 1 & 0 \\ -s\theta & 0 & c\theta \end{bmatrix} \begin{bmatrix} c\psi & -s\psi & 0 \\ s\psi & c\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

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

1) 참고문헌 [2].

$$\Omega = \begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix} + C_{\psi}\left(\begin{bmatrix} 0 \\ \dot{\theta} \\ 0 \end{bmatrix} + C_{\theta} \begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix} \right) = \begin{bmatrix} c\psi c\theta & s\psi & 0 \\ -s\psi c\theta & c\psi & 0 \\ s\theta & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix}$$ (1 -> 2 -> 3 변환)

힘과 모멘트는 각각 다음과 같이 표현된다.

$\Sigma F_{ext} = F_{thrust} - F_{aero} - F_{gravity}$, $\Sigma M_{ext} = M_{thrust} - M_{aero} - M_{gyro}$이고 각 항에 해당하는 변수를 구하면 다음과 같다.

힘 : $F_{thrust} = \begin{bmatrix} 0 \\ 0 \\ b(\omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_4^2) \end{bmatrix}$ (b는 추력에 관한 계수), $F_{aero} = K(R_t^{B \to I})^{-1}\dot{P}$ (K는 공력에 관한 계수), $F_{gravity} = m(R_t^{B \to I})^{-1}G$ ($G = [0\ 0\ g]^T$)

모멘트 : 1번, 3번 로터가 시계방향으로 회전하고 2번, 4번 로터가 반시계방향으로 회전하는 것을 고려하면 다음과 같다. $M_{thrust} = \begin{bmatrix} db(\omega_2^2 - \omega_4^2) \\ db(\omega_3^2 - \omega_1^2) \\ K_d(\omega_1^2 - \omega_2^2 + \omega_3^2 - \omega_4^2) \end{bmatrix}$ (d: 무게중심으로부터 로터의 거리, K_d: 로터의 각운동량에 관한 계수), $M_{aero} = K_r R_r^{I \to B}\dot{A}$ (K_r은 공력에 관한 계수), $M_{gyro} = \sum_{i=1}^{4} J_r(\Omega \times \hat{e_3})(-1)^{i+1}\omega_i$ (J_r: 로터의 회전관성)이다. 여기서 주목할 점은 자이로효과가 위상차 90도를 두고 발생하여 body frame의 x, y축에 대하여 서로 de-coupled되어 있음을 알 수 있다.

운동방정식의 state변수로 사용할 $X = [x_1\ x_2\ x_3\ x_4\ x_5\ x_6\ x_7\ x_8\ x_9\ x_{10}\ x_{11}\ x_{12}]^T = [x\ y\ z\ \phi\ \theta\ \psi\ \dot{x}\ \dot{y}\ \dot{z}\ p\ q\ r]^T$의 derivatives를 구해보면 다음과 같다.

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \\ \dot{x}_5 \\ \dot{x}_6 \\ \dot{x}_7 \\ \dot{x}_8 \\ \dot{x}_9 \\ \dot{x}_{10} \\ \dot{x}_{11} \\ \dot{x}_{12} \end{bmatrix} = \begin{bmatrix} x_7 \\ x_8 \\ x_9 \\ (c(x_6)/c(x_5))x_{10} - (s(x_6)/c(x_5))x_{11} \\ s(x_6)x_{10} + c(x_6)x_{11} \\ -t(x_5)c(x_6)x_{10} + t(x_5)s(x_6)x_{11} + x_{12} \\ \frac{b}{m}(\omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_4^2)s(x_5) \\ \frac{b}{m}(\omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_4^2)(-s(x_4)c(x_5)) \\ \frac{b}{m}(\omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_4^2)(c(x_4)c(x_5)) \\ \dot{x}_{10} \\ \dot{x}_{11} \\ \dot{x}_{12} \end{bmatrix}$$, 이때, ω_i는 I번째 로터의 각속도이다.

여기서 $\dot{\Omega} = \begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} \dot{x}_{10} \\ \dot{x}_{11} \\ \dot{x}_{12} \end{bmatrix}$를 따로 구하면 다음과 같다. $\dot{\Omega} = J^{-1}(\Sigma M_{ext} - \Omega \times (J\Omega))$이므로,

$$\begin{bmatrix}\dot{x}_{10}\\ \dot{x}_{11}\\ \dot{x}_{12}\end{bmatrix}=\begin{bmatrix}\frac{1}{J_{11}}[db(\omega_2^2-\omega_4^2)-J_r x_{11}(\omega_1-\omega_2+\omega_3-\omega_4)+(J_{22}-J_{33})x_{11}x_{12}]\\ \frac{1}{J_{22}}[db(\omega_3^2-\omega_1^2)+J_r x_{11}(\omega_1-\omega_2+\omega_3-\omega_4)-(J_{11}-J_{33})x_{10}x_{12}]\\ \frac{1}{J_{33}}K_d(\omega_1^2-\omega_2^2+\omega_3^2-\omega_4^2)+(J_{11}-J_{22})x_{10}x_{11}\end{bmatrix}$$

이와 같이 Quadrotor Dynamics를 분석해 보았다. 이를 Matlab을 통해 코딩하여 주어진 입력값 $u=[\omega_1\ \omega_2\ \omega_3\ \omega_4]^T$에 대한 쿼드로터의 운동을 분석할 수 있다. 입력값에 쿼드로터가 hovering하는 값인 $\omega=\omega_1=\omega_2=\omega_3=\omega_4=\sqrt{\frac{mg}{4b}}$ 를 입력하면 초기값에 대하여 State변수가 일정함을 알 수 있다.

나. LQT controller design[2)]

선형화된 시스템 $\dot{\bar{x}}=A\bar{x}+B\bar{u}$에 대하여 $\mathfrak{I}=\int_0^\infty(\bar{x}^TQ\bar{x}+\bar{u}^TR\bar{u})dt$를 최소화 하는 제어를 수행하는 LQT controller를 설계하기 위해서는 위에서 구한 쿼드로터의 운동방정식을 선형화하는 과정이 필요하다. $\dot{\bar{x}}=A\bar{x}+B\bar{u}$의 형태로 표현하기 위하여 평형점(hovering 상태)에서 선형화하여 구한 A , B matrix는 다음과 같다.

$$A=\begin{bmatrix}0&0&0&0&0&0&1&0&0&0&0&0\\0&0&0&0&0&0&0&1&0&0&0&0\\0&0&0&0&0&0&0&0&1&0&0&0\\0&0&0&0&0&0&0&0&0&1&0&0\\0&0&0&0&0&0&0&0&0&0&1&0\\0&0&0&0&0&0&0&0&0&0&0&1\\0&0&0&0&\frac{4bc^2}{m}&0&0&0&0&0&0&0\\0&0&0&-\frac{4bc^2}{m}&0&0&0&0&0&0&0&0\\ &&&&O&&&&&&&\\ &&&&O&&&&&&&\\ &&&&O&&&&&&&\\ &&&&O&&&&&&&\end{bmatrix}\qquad B=\begin{bmatrix} &&O'&&\\ &&O'&&\\ &&O'&&\\ &&O'&&\\ &&O'&&\\ &&O'&&\\ &&O'&&\\ &&O'&&\\ \frac{2bc}{m}&\frac{2bc}{m}&\frac{2bc}{m}&\frac{2bc}{m}\\ 0&\frac{2dbc}{J_{11}}&0&-\frac{2dbc}{J_{11}}\\ -\frac{2dbc}{J_{22}}&0&\frac{2dbc}{J_{22}}&0\\ 2K_dcJ_{33}-2K_dcJ_{33}&2K_dcJ_{33}&-2K_dcJ_{33}\end{bmatrix}$$

($c=\sqrt{\frac{mg}{4b}}$, $O=[0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0]$, $O'=[0\ 0\ 0\ 0]$)

여기서, Riccati equation $0=PA+A^TP-PBR^{-1}B^TP+Q$를 만족하는 P, Q, R에 대하여 $\bar{u}=-R^{-1}B^TP\bar{x}=-K\bar{x}$이다. 매트랩에서 명령어 lqr을 통해 이를 계산한다. 이는 다음과 같이 표현된다. K = lqr(A,B,Q,R). 이와 같이 선형화된 시스템에 대하여 LQT controller를 설계 하였다. 이를 적용하여 쿼드로터의 제어 성능을 알아볼 수 있다. Hovering 상태의 초기 위치에서 주어진 위치로 이동하는 쿼드로터의 제어성능을 살펴보면 다음과 같다. 이에 대하여 관성좌표계에서 표현되는 x, y, z, ϕ, θ, ψ의 변화를 그래프로 그려보았다.

2) 참고문헌 [5].

① $(x_0\,,\,y_0\,,\,z_0)=(0\,,0\,,10)\;\rightarrow(x_f\,,\,y_f\,,\,z_f)=(5\,,5\,,10)$

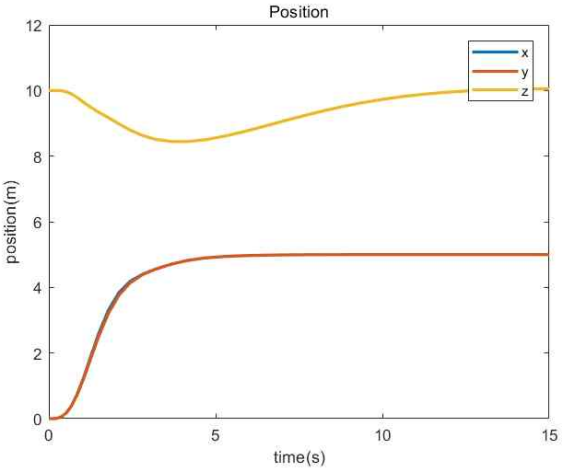

Fig. 2 - Position 1

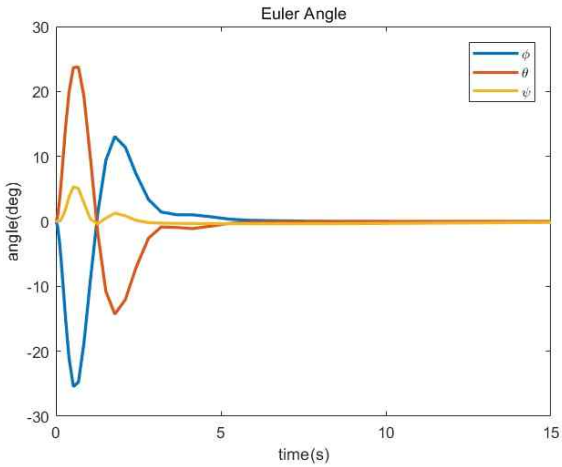

Fig. 3 - Euler Angle 1

각 로터의 Saturation을 고려하지 않았으므로 5m이상 이동하는데 매우 빠른 반응을 보인다. 만일 로터가 충분한 각속도를 형성할 수 있으면 이를 제어하는 것이 가능함을 알 수 있다. 오일러각의 변화는 25도 이내에서 이루어 졌음을 확인해 볼 수 있다. 또한 z position이 일시적으로 낮아짐을 확인할 수 있다.

② $(x_0\,,\,y_0\,,\,z_0)=(0\,,0\,,10)\;\rightarrow(x_f\,,\,y_f\,,\,z_f)=(1\,,1\,,10)$

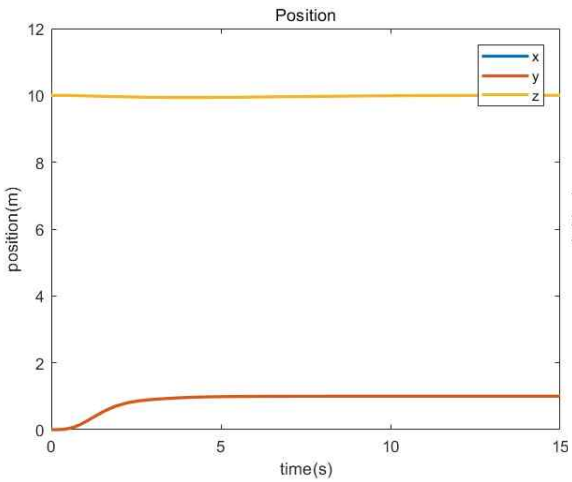

Fig. 4 - Position 2

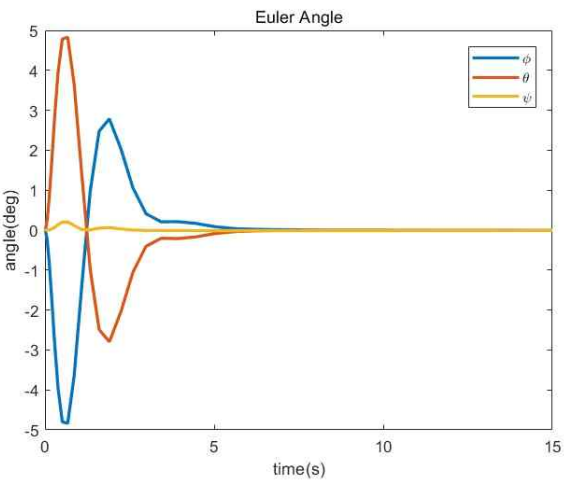

Fig. 5 - Euler Angle 2

작은 거리를 기동함에 있어서는 빠른 반응과 안정적인 자세를 가짐을 확인해 볼 수 있다. 실제로 쿼드로터가 공을 튀기기위해서는 공이 상승 하강을 거치는 과정에서 생기는 변위를 쿼드로터가 따라가야한다. 공이 정확히 수직상승하지 않는다는 점을 고려할 때, 발생하는 xy 평면상의 거리는 약 1~2m 내외 일 것이다. 이를 고려하면 LQT 제어기는 좋은 성능을 보임을 확인 할 수 있다.

다. Trajectory generation[3)]

(1) Trajectory

앞서 가.에서 살펴 본 Quadrotor Dynamics와 함께 실제 쿼드로터의 기동이 Small angle assumption을 만족한다고 할 때, 쿼드로터의 위치에 대한 방정식은 다음과 같다.

$$\begin{bmatrix}\ddot{X}\\ \ddot{Y}\\ \ddot{Z}\end{bmatrix}=\begin{bmatrix}\theta T_z\\ -\phi T_z\\ T_z-g\end{bmatrix}$$ (where, T_z : 쿼드로터 수직방향 추진력)

여기서 우리는 쿼드로터의 궤적이 Affine input에 따라 움직인다고 가정하고 이를 만족하는 Affine input을 $T_z=A_T t+B_T$, $\ddot{\theta}=A_\theta t+B_\theta$, $\ddot{\phi}=A_\phi t+B_\phi$와 같이 시간에 대한 1차 방정식으로 놓는다.

이때 우리는 공의 위치와 속도로부터 주어진 Affine input이 만족해야하는 Boundary condition을 얻을 수 있다.

① 현시점으로부터 impact 시점까지의 시간 T_f

② 공과 쿼드로터가 만나는 지점 $(X_f,\ Y_f,\ Z_f)$

③ 공과 쿼드로터가 만나는 시점에 쿼드로터가 가져야하는 Orientation (n_{f1}, n_{f2}, n_{f3})

④ Impact 시점에서 쿼드로터가 가져야하는 수직성분 속도 $V_\perp$

쿼드로터의 위치, 속도, θ, ϕ에 대한 미분 방정식을 풀어 boundary condition을 만족시키는 식을 찾으면 총 6개의 식을 얻을 수 있다. 이로부터 $A_T, A_\theta, A_\phi, B_T, B_\theta, B_\phi$의 6개 미지수를 찾을 수 있다. 최종적으로 6개의 식은 1개 미지수에 대한 4차방정식으로 정리되며 이는 Matlab을 이용하여 해를 찾을 수 있다. 자세한 과정은 코드에 첨부되어있다. 따라서 이를 통해 쿼드로터가 공을 다시 튀겨 올릴 조건을 만족하는 Trajectory를 얻을 수 있다. 뒤이어지는 과정으로는 이러한 Trajectory를 따르도록 4개의 로터에 필요한 각속도를 계산해야 한다.

(2) Rotor Input Calculation

위에서 구한 Affine input과 Quadrotor Dynamics를 이용해 계산된 Trajectory를 따르는 로터의 각속도를 구해낸다. 이때 쿼드로터의 회전 기동이 크게 작용하지 않으므로 자이로 효과는 무시하며 쿼드로터의 Z축 방향 회전운동 ψ, $\dot{\psi}$는 크지 않으므로 0으로 간주한다. 필요한 4개의 로터 각속도 계산 과정을 간략히 나타내면 다음과 같다.

$$\begin{bmatrix}\omega_1^2\\ \omega_2^2\\ \omega_3^2\\ \omega_4^2\end{bmatrix}=I_b^{-1}\begin{bmatrix}T_z\\ J\begin{bmatrix}\ddot{\phi}\\ \ddot{\theta}\\ 0\end{bmatrix}+\left(\begin{bmatrix}\dot{\phi}\\ \dot{\theta}\\ 0\end{bmatrix}\times J\begin{bmatrix}\dot{\phi}\\ \dot{\theta}\\ 0\end{bmatrix}\right)\end{bmatrix}$$ where I_b is physical coefficient of Quadrotor

3) 참고문헌 [1]

Trajectory generation을 몇 가지 경우에 적용하여 다음과 같은 Trajectory를 얻을 수 있다.

① $T_f = 1.5s$, $X_0 = [0\ 0\ 10]$, $X_f = [0.5\ 0.8\ 11]$, $n_f = [0.05\ 0.075\ 2]$, $V_\perp = 6m/s$

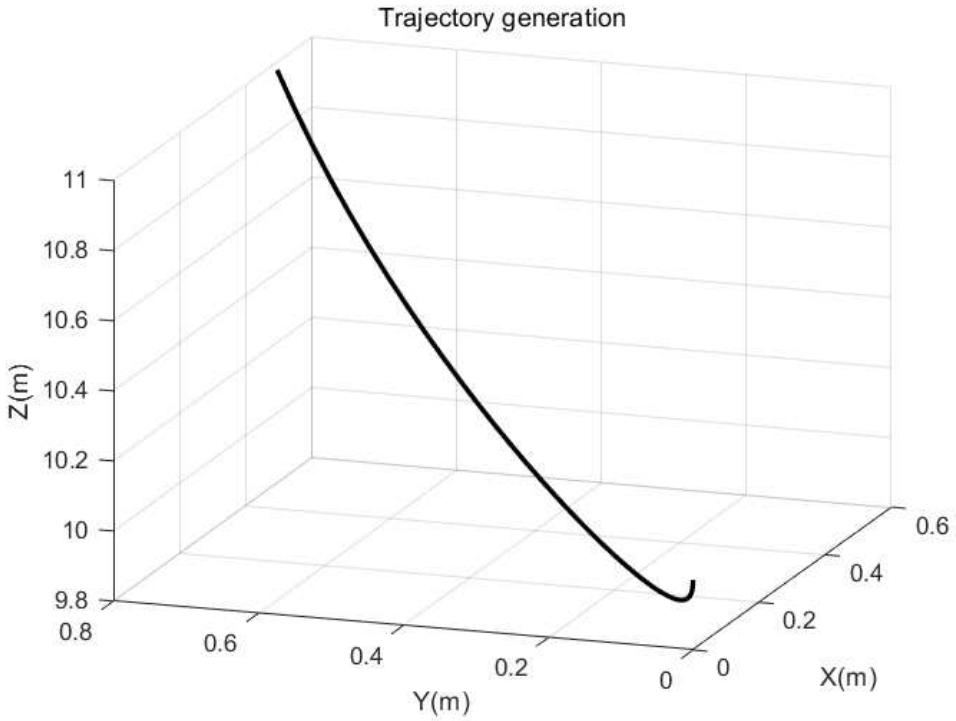

Fig. 6 - Trajectory generation 1

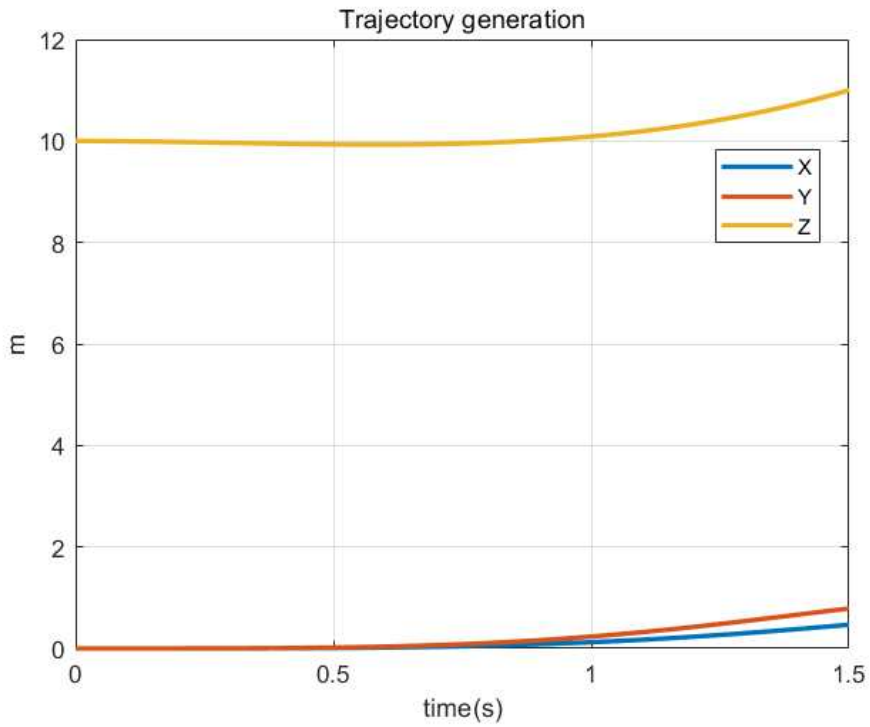

Fig. 7 - Trajectory generation 1

② $T_f = 1.5s$, $X_0 = [0\ 0\ 10]$, $X_f = [0.8\ 0\ 12.5]$, $n_f = [0.05\ 0.075\ 2]$, $V_\perp = 6m/s$

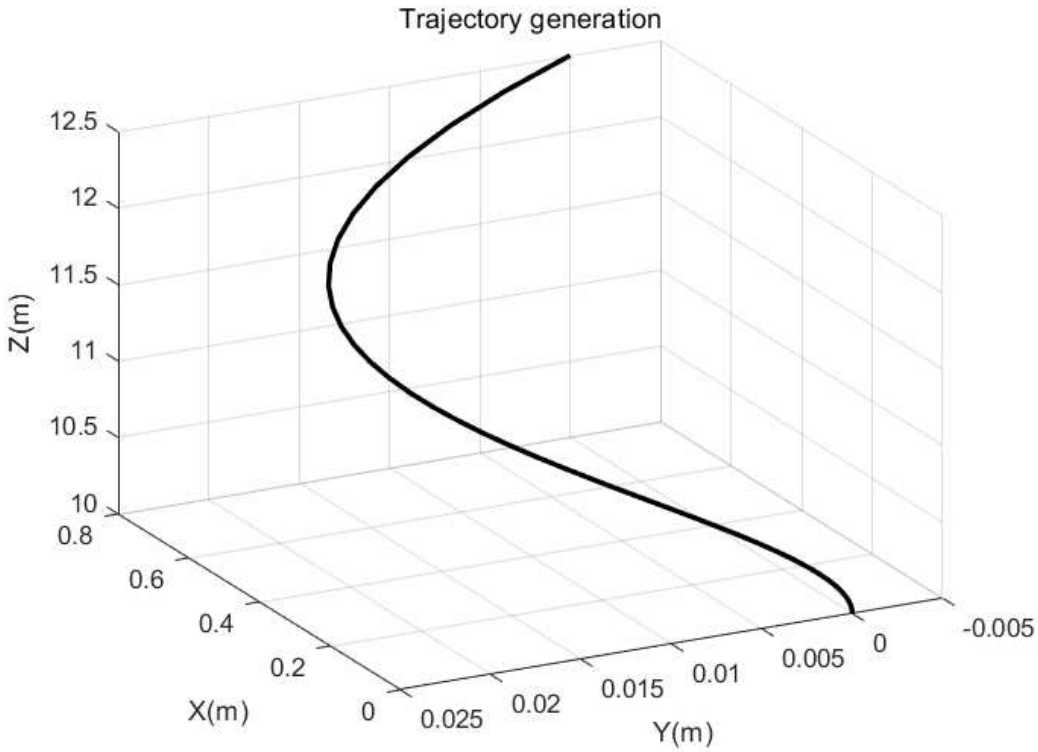

Fig. 8 - Trajectory generation 2

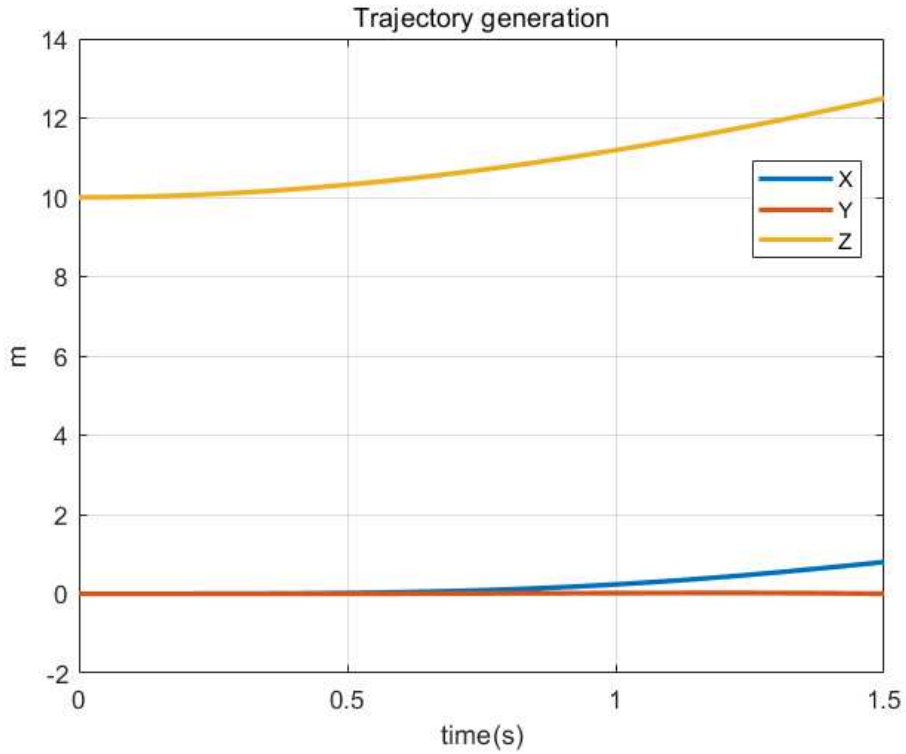

Fig. 9 - Trajectory generation 2

위 두 경우에 대하여 쿼드로터가 Trajectory를 따르도록 하는 Rotor input은 다음과 같다.

① $T_f = 1.5s$, $X_0 = [0\ 0\ 10]$, $X_f = [0.5\ 0.8\ 11]$, $n_f = [0.05\ 0.075\ 2]$, $V_\perp = 6m/s$

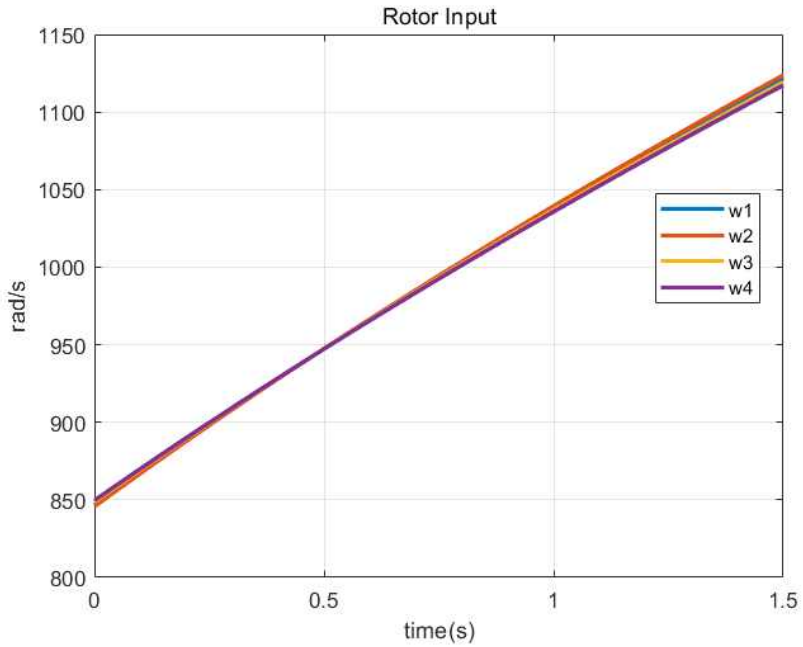

Fig. 10 - Rotor Input 1

② $T_f = 1.5s$, $X_0 = [0\ 0\ 10]$, $X_f = [0.8\ 0\ 12.5]$, $n_f = [0.05\ 0.075\ 2]$, $V_\perp = 6m/s$

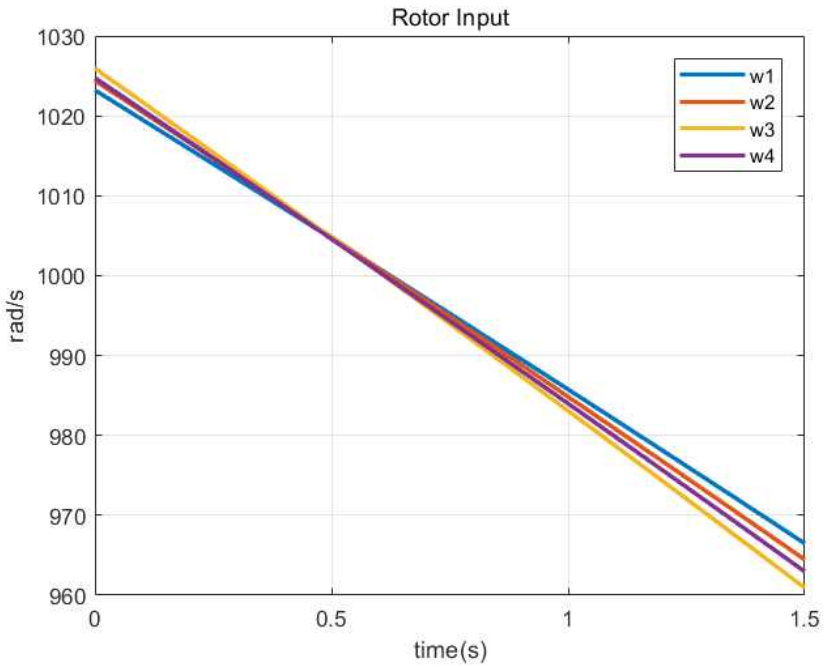

Fig. 11 - Rotor Input 2

위 두 경우 모두 Rotor Input이 feasible한 Rotor Input을 내고 있음을 확인해 볼 수 있다. 따라서 설계된 Trajectory generation이 적절함을 확인할 수 있다.

현재까지 설계된 알고리즘의 Simulink 구조도를 보면 다음과 같다.

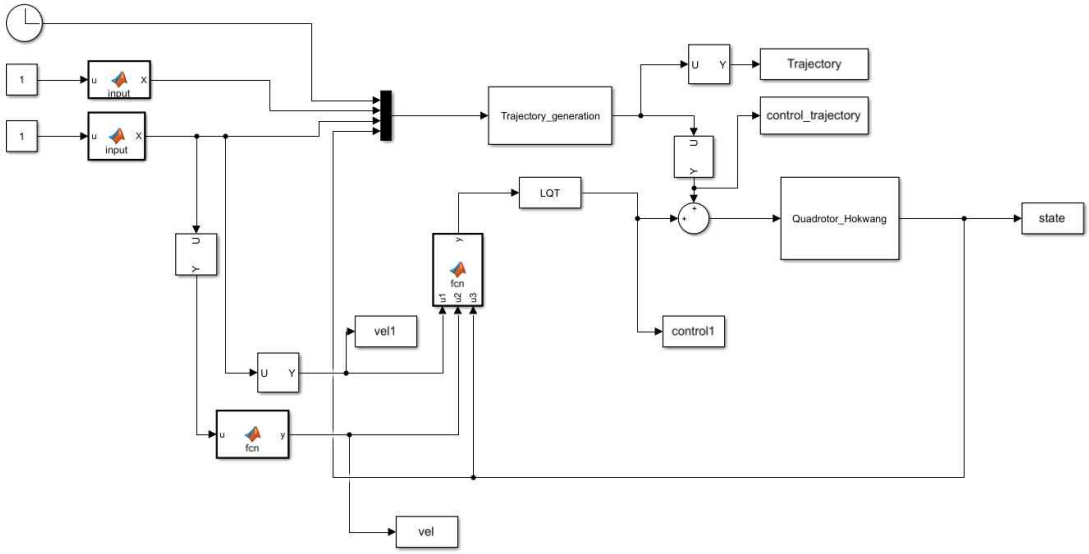

Fig. 12 - Structure

구조도는 다음과 같은 과정으로 설계되어있다.

① 공으로부터 계산된 boundary condition이 Trajectory generation block으로 들어간다.

② 계산된 Trajectory로부터 필요한 로터 각속도를 계산하여 System으로 보낸다.

③ System으로부터 현재 State를 받아 LQT block과 Trajectory generation block으로 보낸다.

④ 계산된 Trajectory를 따르도록 쿼드로터를 제어한다.

LQT 제어기를 통해 계산된 Trajectory를 수행한 그래프를 살펴보면 다음과 같다.

① Z축 변위가 큰 경우 Trajectory를 충분히 잘 따르는 제어가 가능함을 알 수 있다.

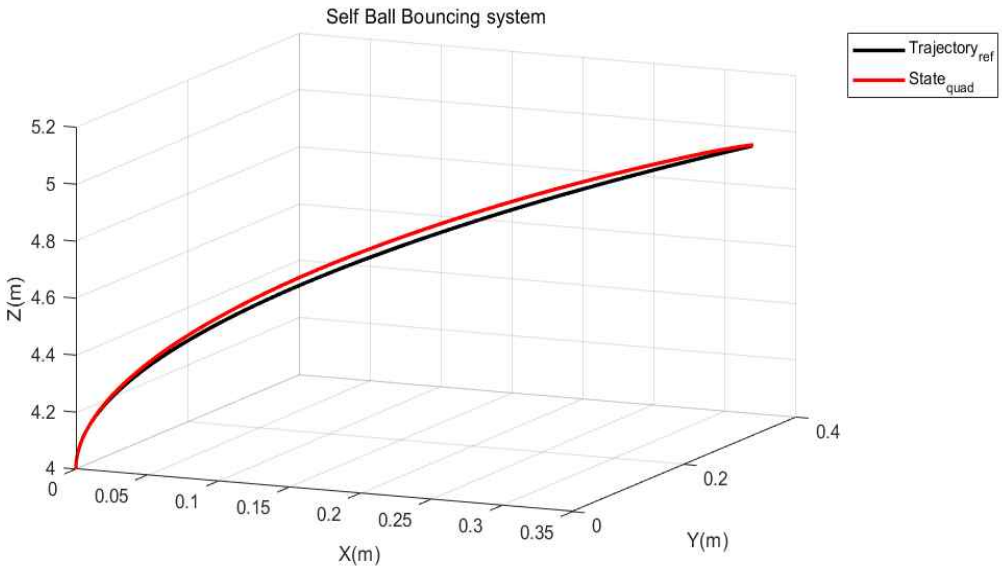

Fig. 13 - Trajectory following

② Z축 변위가 음수인 동시에 X, Y축 변위가 존재하는 경우 Trajectory를 따르는 제어가 어려운 것을 확인할 수 있다. 그 이유는 주어진 시간 내에 쿼드로터가 X, Y축으로 이동하기 위해서는 충분한 로터 각속도가 필요한데 이는 동시에 Z축이 감소하는 것과 상반되기 때문이다.

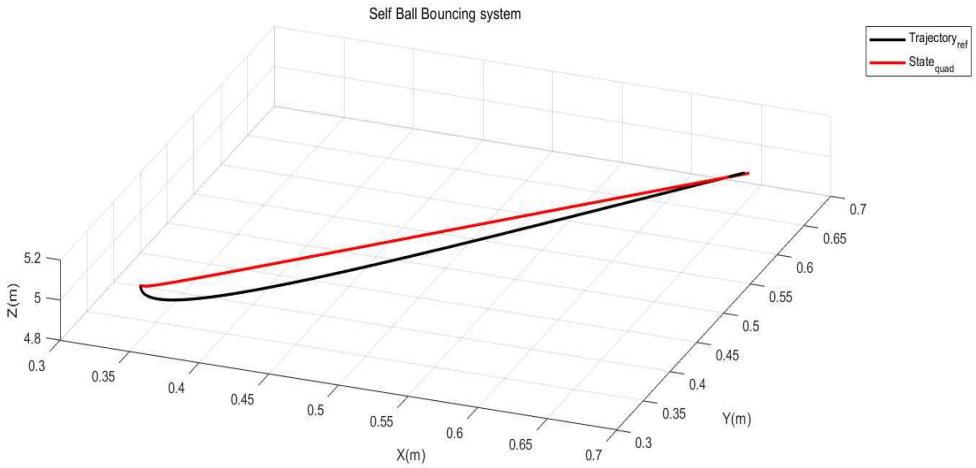

Fig. 14 - Trajectory following 2

라. Ball dynamics & Prediction

(1) Ball dynamics

공은 탁구공으로 가정하여 질량, 탄성계수, 공기 저항계수 등을 이용한다. 실제로 공은 공기 중에서 중력과 공기저항의 영향을 받는다. 이를 토대로 공의 운동방정식을 세우고 주어진 공의 초기 위치와 속도에 따라 공의 궤적을 그릴 수 있다. 제일 처음 사람이 공을 던져준다고 할 때 공은 관성 좌표계에서 수직 방향의 속도성분 뿐만 아니라 수평방향의 속도성분 또한 가진다. 이를 고려하여 공의 궤적을 그린다. 원하는 다음 충돌 위치는 공의 궤적이 초기 높이와 같아지는 시점을 따른다. 이렇게 계산된 다음 충돌 위치와 충돌까지 주어진 시간을 계산하면 이를 Trajectory generation에 활용할 수 있다. 공의 회전은 고려하지 않기로 한다. 또한 공이 쿼드로터와 충돌 할 때 쿼드로터의 질량이 공보다 훨씬 크므로 쿼드로터의 운동량 변화를 무시하고 공의 속도 변화를 고려한다. 따라서 Ball dynamics는 다음과 같다.

① 체공 시

$$\begin{bmatrix} \ddot{x}_{ball} \\ \ddot{y}_{ball} \\ \ddot{z}_{ball} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} - K_D[V_{ball}]^T[V_{ball}] \quad (\text{where, } K_D = \frac{\rho A_{ball} C_D}{2m_{ball}})$$

② 충돌 시 순간 변화

$V_{ball+} = V_{ball-} - (1+\beta)((V_{ball-} - V_{quad})^T n)n$ (where, β:coefficient of restitution, n: orientation of the quadrotor)

마. Simulation

앞서 라.에서 구한 Ball dynamics를 포함하여 스스로 공을 튀기는 쿼드로터 시스템의 시뮬레이션을 구성하면 다음과 같다.

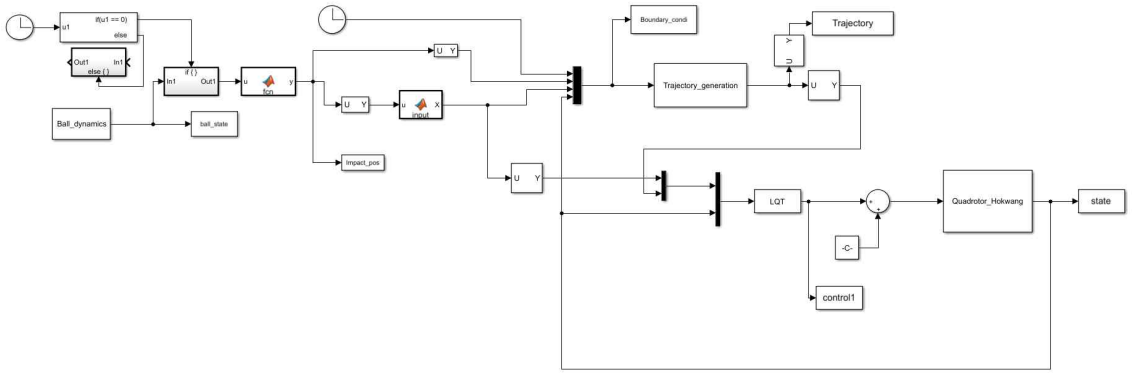

Fig. 15 - Simulation structure

이를 통해 수행한 시뮬레이션 모습은 다음과 같다.

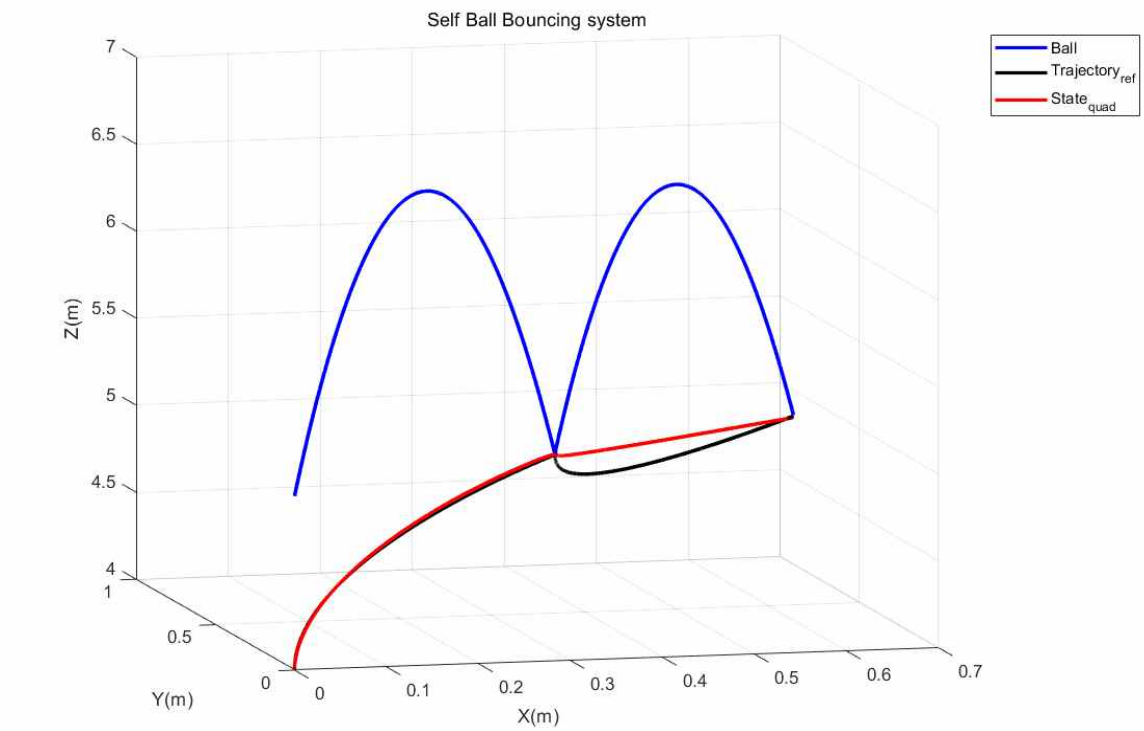

Fig. 16 - Self Ball Bouncing system

위 그래프는 초기에 사람이 공을 5m높이에서 던져 주었을 때 공의 궤적을 보여주고 다시 공이 5m 높이에 도달하는 시점을 충돌 시점으로 하여 궤적을 형성하여 이를 따르도록 제어한 모습을 볼 수 있다. 첫 충돌 까지는 쿼드로터가 계산된 궤적을 잘 따라가지만 두 번째 궤적부터는 쿼드로터의 실제 이동과 계산된 궤적 사이에 어느 정도 차이가 발생함을 확인할 수 있다.

3. 결론 및 추후 연구 방향

결론

쿼드로터를 활용하여 스스로 공을 튀기는 제어시스템을 구성한 결과 공의 위치 예측을 통해 쿼드로터가 현재 시점으로부터 최종 충돌시점 까지 이동해야하는 궤적을 생성할 수 있었고 이를 따르는 제어기를 설계하여 실제로 공을 다시 튀겨 올릴 수 있는 알고리즘을 성공적으로 구현할 수 있었다. 하지만 LQT 제어기가 제자리 비행(hovering) 상태의 선형화된 값을 활용하여 쿼드로터를 제어 한다는 점에서 생성된 궤적을 정확히 따르는 제어기를 설계할 수 없었다. 그럼에도 불구하고 공의 위치와 속도에 따른 다음 충돌 지점과 다시 공을 충분한 높이로 튀기기 위해 쿼드로터가 움직여야 하는 궤적을 정확히 생성하였다는 점에서 큰 의미가 있다고 할 수 있다. 또한 MATLAB과 Simulink를 활용한 제어시스템 구성의 실용성을 확인할 수 있었다.

추후 연구 방향

계산된 궤적을 더욱 잘 따르기 위해서는 비선형 제어기를 도입해야 될 것으로 예상된다. 그 이유는 궤적 생성에서 주어지는 값은 위치뿐만 아니라 매 시점에서 요구되는 속도, 가속도, 각속도, 각가속도를 포함하기 때문이다. 비선형 제어기는 안정한 자세에서 선형화된 것이 아니라 급격한 기동을 포함한 다양한 자세와 조건의 기동을 가능하게 하므로 이번 논문에서 구해낸 궤적을 따르는데 활용될 수 있을 것으로 예상된다.

본 논문에서는 궤적 생성을 쿼드로터의 초기 위치에서 단 한번 궤적생성을 한 후 이를 따라가도록 제어하였다. 하지만 이를 한 단계 더 발전시킨다면 쿼드로터의 기동에 따라 매순간 최적의 궤적을 생성해낸다면 예상치 못한 변수에 반응할 수 있을 것으로 예상된다. 즉, 공이 외력에 의해 매순간 최종지점이 다르게 예측된다면 쿼드로터의 궤적 또한 매순간 새롭게 계산되어야 할 것이다.

이번 연구는 시간과 비용의 문제로 인해 하드웨어 테스트 이전 단계인 SILS (Software-in-the-loop simulation)를 완료하는 것으로 연구를 마무리 하였다. 하지만 여기서 한 걸음 더 나아가 하드웨어 테스트인 HILS (Hardware-in-the-loop simulation)까지 수행한다면 알고리즘을 실증할 수 있을 것으로 예상된다.

이 논문에서 사용된 알고리즘은 쿼드로터를 활용한 공튀기기를 넘어서 다른 임무를 수행하는 쿼드로터에 사용될 수 있을 것이다. 그 이유는 이 논문에서 사용된 궤적 생성 알고리즘은 주어진 경계 값이 있을 때 이를 만족하는 궤적을 생성하는 알고리즘으로 포괄적인 개념을 가지고 있기 때문이다.

4. 참고문헌

[1] Mark Muller, Sergei Lupashin, and Raffaello D'Andrea, "Quadrocopter Ball Juggling", IEEE/RSJ International Conference on Intelligent Robots and Systems, pp.5113–5120, 2011.

[2] J. Park, Y. Kim, S. Kim, "Landing site searching and selection algorithm development using vision system and its application to quadrotor", IEEE Transactions on Control Systems Technology, vol. 23, no. 2, pp. 488-503, 2015.

[3] Mellinger, D., Michael, N., and Kumar, V., "Trajectory generation and control for precise aggressive maneuvers with quadrotors," The International Journal of Robotics Research, Vol. 31, No. 5, 2012, pp. 664-674.

[4] Brian L. Stevens, Frank L. Lewis, Eric N. Johnson, *Aircraft Control and Simulation: Dynamics, Controls Design, and Autonomous Systems, 3rd Edition,* Wiley-Blackwell, 2015.

[5] R. M. Murray, *Lecture 2 - LQR Control*, Control and Dynamical Systems, California Institute of Technology, 2006.

[6] S. Bouabdallah, A. Noth, and R. Siegwart, ""PID vs LQ control techniques applied to an indoor micro quadrotor,"" in Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on, vol. 3, Sept.-2 Oct. 2004, pp. 2451–-2456 vol.3.

[7] D. Mellinger, N. Michael, and V. Kumar, ""Trajectory generation and control for precise aggressive maneuvers with quadrotors,"" in Int. Symposium on Experimental Robotics, 2010.

[8] B. W. McCormick, *Aerodynamics Aeronautics and Flight Mechanics,* John Wiley & Sons, Inc, 1995.