# EN530.678 Nonlinear Control and Planning in Robotics

Project presentation

Planning and control of a quadrotor in obstacle-dense terrain

Soowon Kim
Hyungmu Lee

# Objective

- Catastrophes : earthquake, flood, volcano etc..

⇒ Difficult to reach out to find and save people

⇒ Quadrotor is an alternative way to find and save them!!

⇒ How to get to the destination safely and effectively arriving and precisely landing?

# Trajectory planning

- Dijkstra*

  Shortest path using uniform grid

- Differential Flatness**

  Flat output: $[x_d, y_d, z_d, \psi_d]$

- Polynomial trajectory optimization**

  Minimum snap

- Piecewise polynomial joint optimization***

  Unconstrained quadratic programming

*https://github.com/stormmax/quadrotor
**Bry, A., et al. IJRR (2015)
***Mellinger, D., et al. IEEE (2011)

# Polynomial trajectory optimization

$$P(t) = p_n t^n + P_{n-1} t^{n-1} + \cdots + p_1 t + p_0$$

$$J = \int_0^\tau P^{(4)}(t)^2 dt = \mathbf{p}^{\mathrm{T}} \mathbf{Q} \mathbf{p}$$ → **Minimum snap**

$$\mathbf{A}_0 \mathbf{p} = \mathbf{b}_0$$
$$\mathbf{A}_\tau \mathbf{p} = \mathbf{b}_\tau$$ → **Equality constraint**

$$\min_{\mathbf{p}} \mathbf{p}^T \mathbf{Q} \mathbf{p}$$

$$\mathbf{A}\mathbf{p} = \mathbf{b}, \mathbf{b} = \left[ x_0 \; \dot{x}_0 \; \ddot{x}_0 \; \dddot{x}_0 \; x_0^{(4)} \mid x_\tau \; \dot{x}_\tau \; \ddot{x}_\tau \; \dddot{x}_\tau \; x_\tau^{(4)} \right]^T$$ → **Quadratic programming (QP)**

# Piecewise polynomial joint optimization

$$\begin{bmatrix}\mathbf{p}_1\\ \vdots\\ \mathbf{p}_K\end{bmatrix}^T \begin{bmatrix}\mathbf{Q}_1 & & \\ & \ddots & \\ & & \mathbf{Q}_K\end{bmatrix} \begin{bmatrix}\mathbf{p}_1\\ \vdots\\ \mathbf{p}_K\end{bmatrix}$$

$$\begin{bmatrix}\mathbf{A}_1 & & \\ & \ddots & \\ & & \mathbf{A}_K\end{bmatrix} \begin{bmatrix}\mathbf{p}_1\\ \vdots\\ \mathbf{p}_K\end{bmatrix} = \begin{bmatrix}\mathbf{b}_1\\ \vdots\\ \mathbf{b}_K\end{bmatrix}$$

Works well with small number of segments

$$J = \begin{bmatrix}\mathbf{b}_1\\ \vdots\\ \mathbf{b}_K\end{bmatrix}^T \begin{bmatrix}\mathbf{A}_1 & & \\ & \ddots & \\ & & \mathbf{A}_K\end{bmatrix}^{-T} \begin{bmatrix}\mathbf{Q}_1 & & \\ & \ddots & \\ & & \mathbf{Q}_K\end{bmatrix} \begin{bmatrix}\mathbf{A}_1 & & \\ & \ddots & \\ & & \mathbf{A}_K\end{bmatrix}^{-1} \begin{bmatrix}\mathbf{b}_1\\ \vdots\\ \mathbf{b}_K\end{bmatrix}$$

Converted into unconstrained QP

$$J = \begin{bmatrix}b_{fix}\\ b_{free}\end{bmatrix}^T CA^{-T}QA^{-1}C^T \begin{bmatrix}b_{fix}\\ b_{free}\end{bmatrix} = \begin{bmatrix}b_{fix}\\ b_{free}\end{bmatrix}^T \begin{bmatrix}R_{11} & R_{12}\\ R_{21} & R_{22}\end{bmatrix} \begin{bmatrix}b_{fix}\\ b_{free}\end{bmatrix}$$

Rearranged cost function

$$b_{free}^* = -R_{22}^{-1}R_{12}^T b_{fix}$$

**Optimized free derivatives**

# Time allocation

$$J_\tau = \mathbf{p}^{\mathrm{T}}\mathbf{Q}\mathbf{p} + c_\tau \sum_{i=1}^{K} \tau_i$$

- Cost is minimized using gradient descent method.

$$\nabla_{\boldsymbol{\tau}} J \approx \left[\frac{J(\tau_1 + \epsilon) - J(\tau_1 - \epsilon)}{2\epsilon} \quad \cdots \quad \frac{J(\tau_K + \epsilon) - J(\tau_K - \epsilon)}{2\epsilon}\right]$$

$$\boldsymbol{\tau}_{new} = \boldsymbol{\tau}_{old} - \alpha \nabla_{\boldsymbol{\tau}} J$$

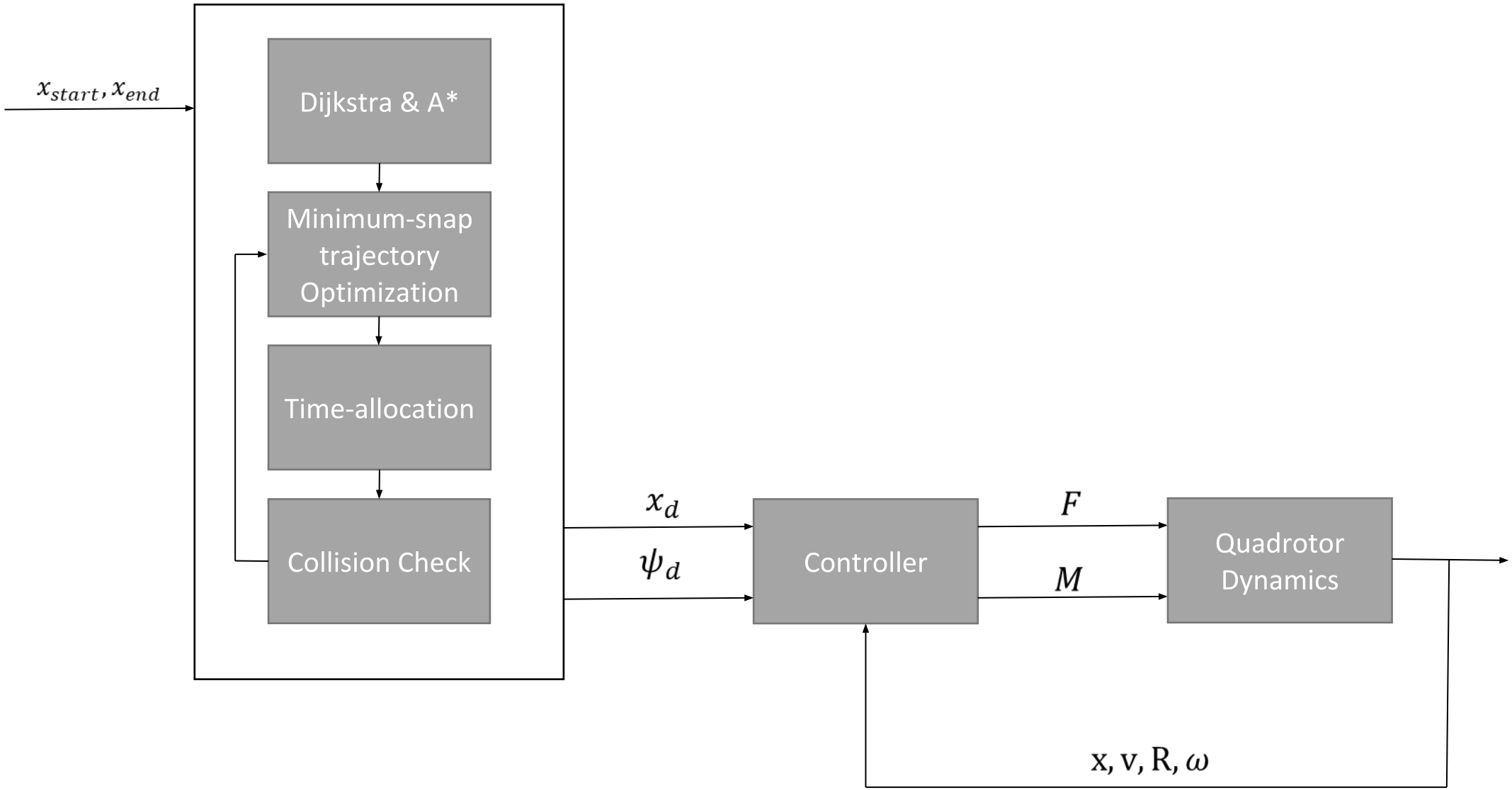
- The shape of optimized trajectory does not depend on the time penalty.

- Larger penalty gives faster movement

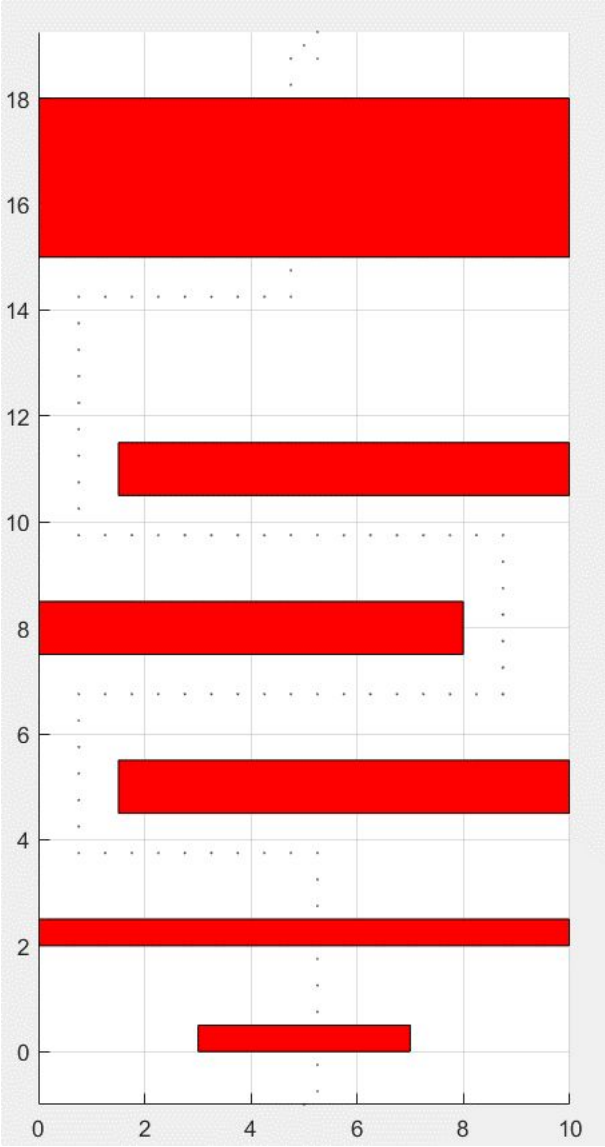
# Collision free trajectory

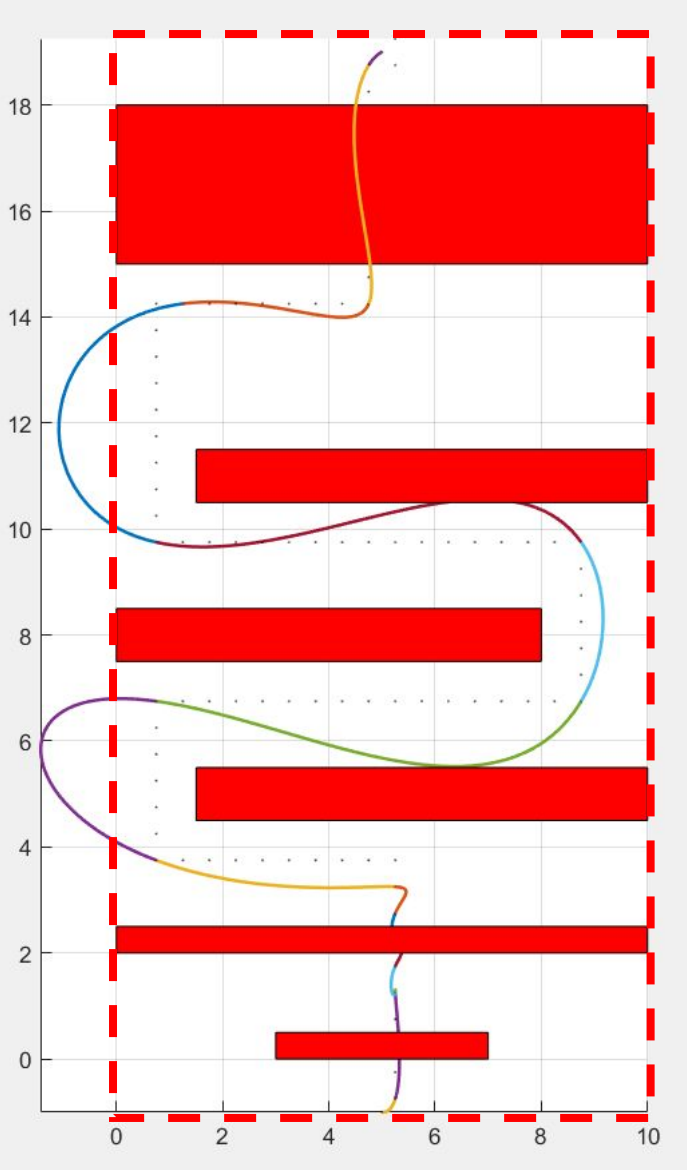
If a certain segment intersects with an obstacle, a mid-way point is added, then recalculate the trajectory.

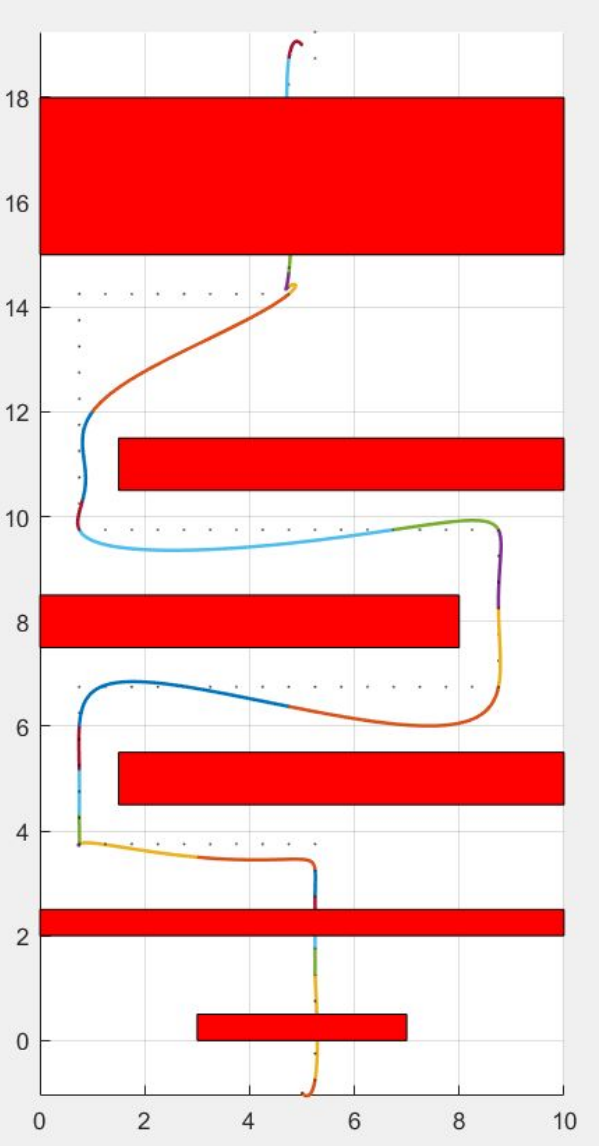
# Algorithm structure



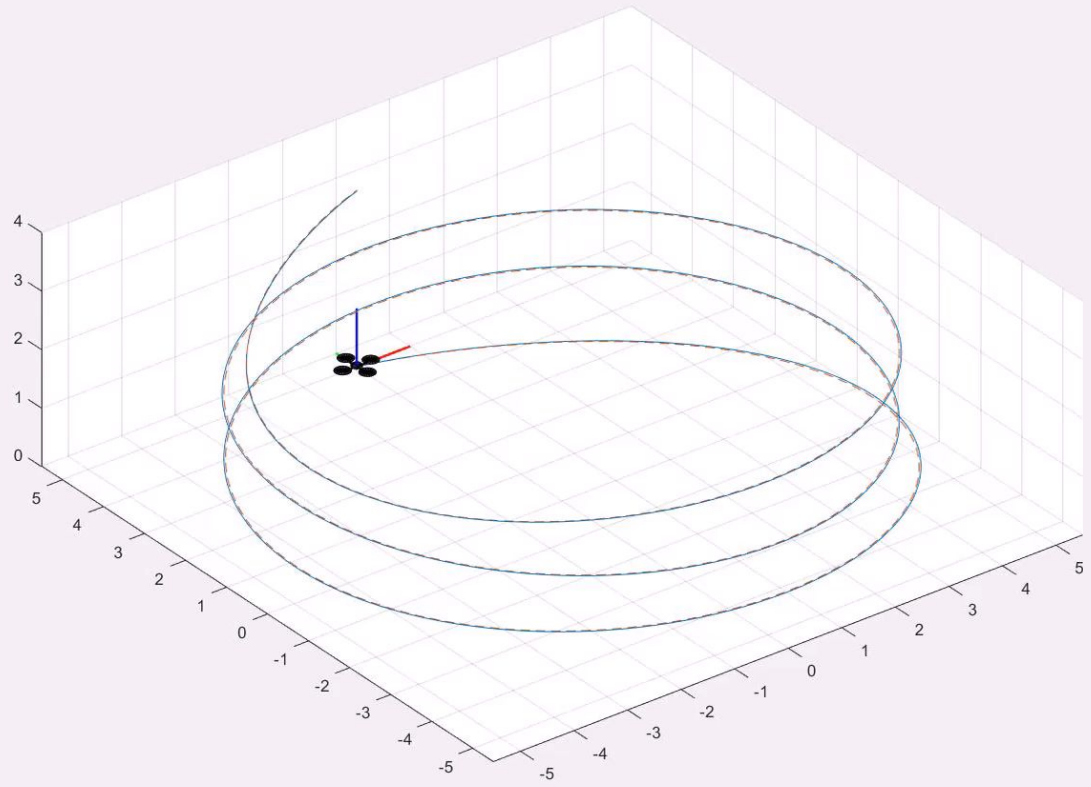
# Implementation (Dijkstra -> trajectory -> collision free)

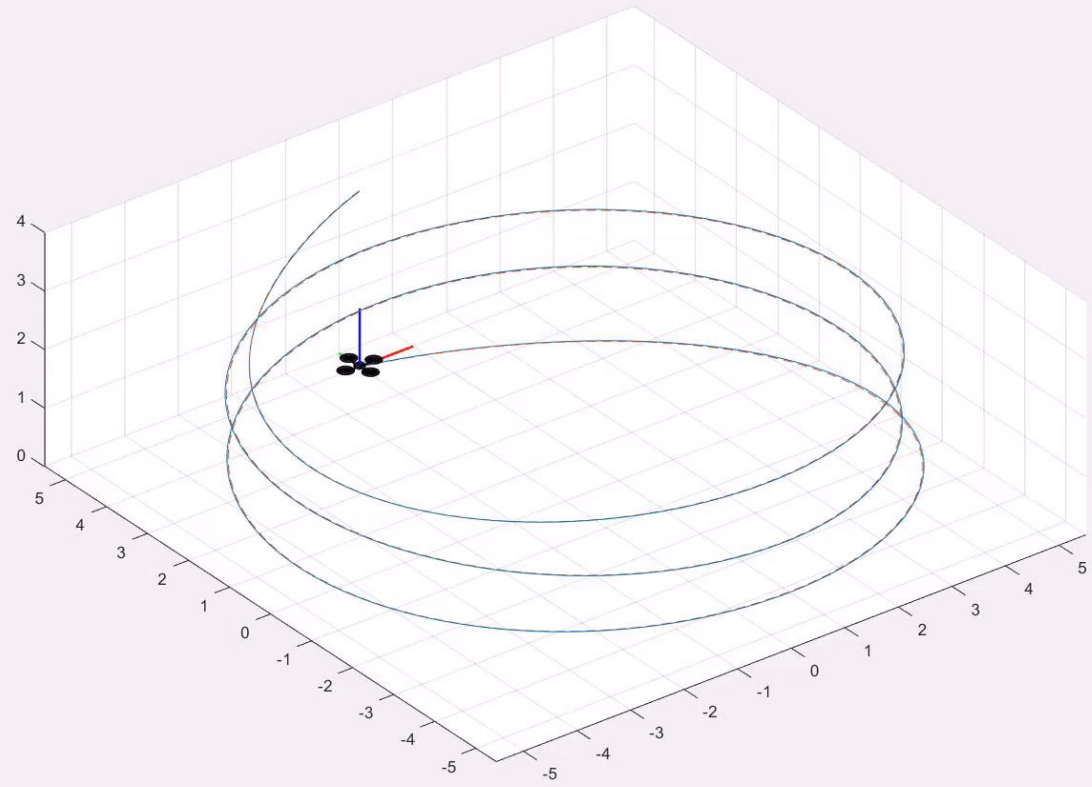


# Implementation – time allocation



# Dynamics

$$m\ddot{r} = -mgz_w + Fz_b$$

$$J\dot{\omega}^b + \omega^b \times J\omega^b = M$$

Control law*:

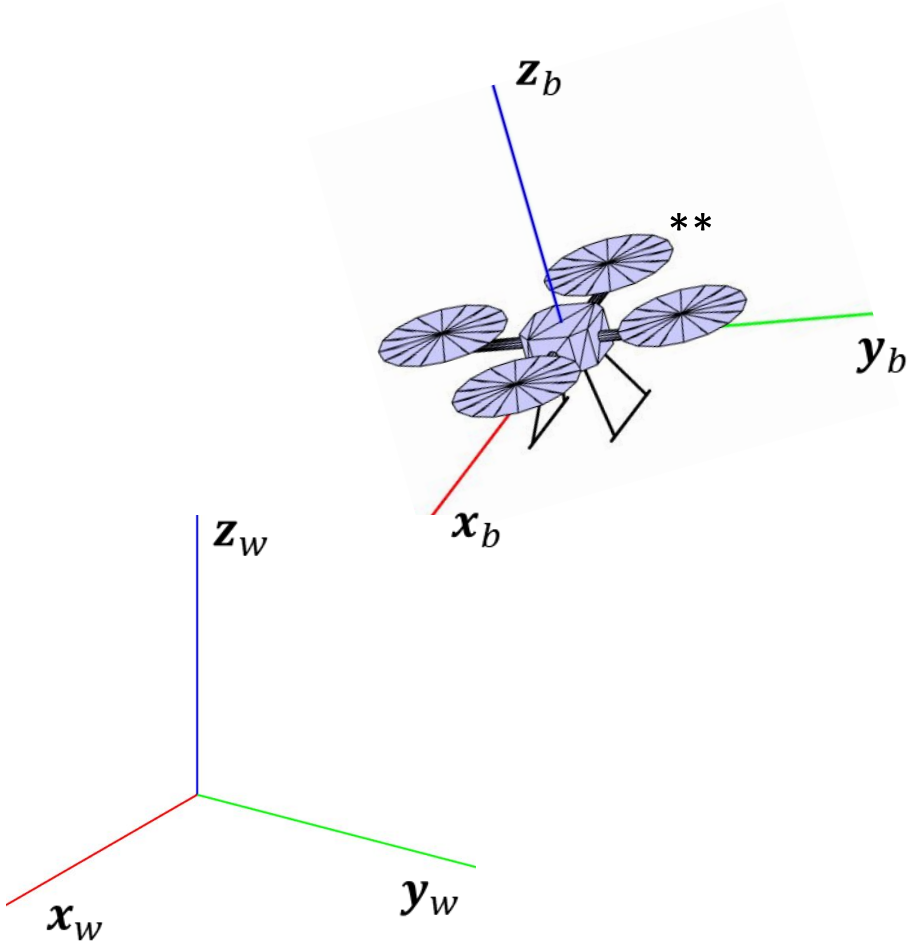
$$F = (-k_x e_x - k_v e_v + mg\mathbf{z}_w + m\ddot{x}_d) \cdot \mathbf{z}_b$$

$$M = -k_R e_R - k_\omega e_\omega + \omega \times J\omega - J(\hat{\omega} R^T R_d \omega_d - R^T R_d \dot{\omega}_d)$$

Model parameters**:

$$J = [0.0115, 0.0115, 0.0218]\ kgm^2$$

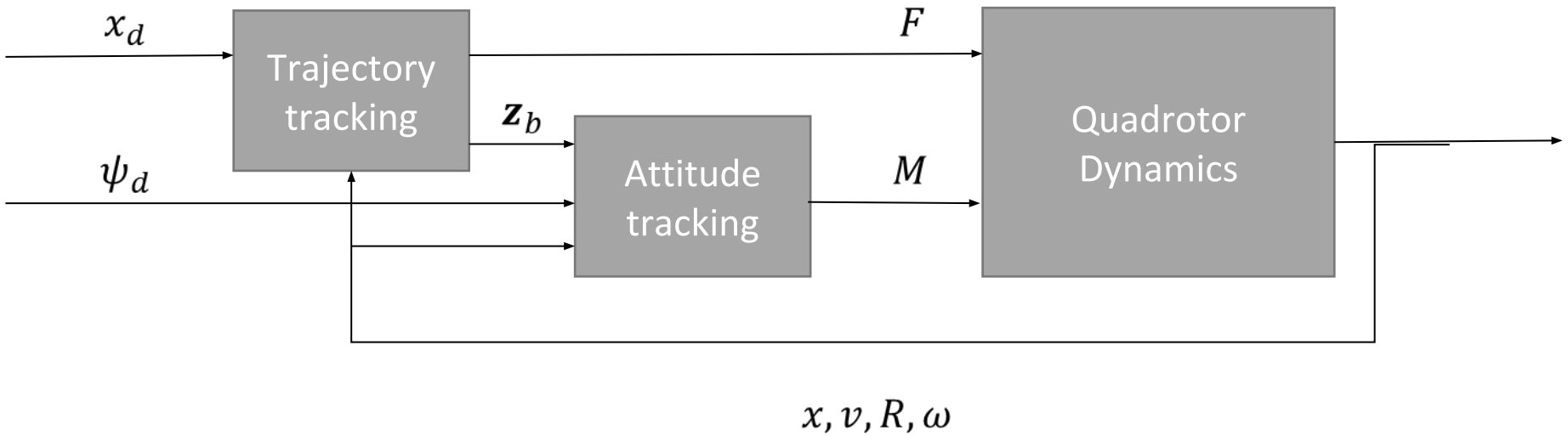
$$m = 1.477\ kg,\ d = 0.263\ m,\ c_{\tau f} = 8 \times 10^{-4} m$$



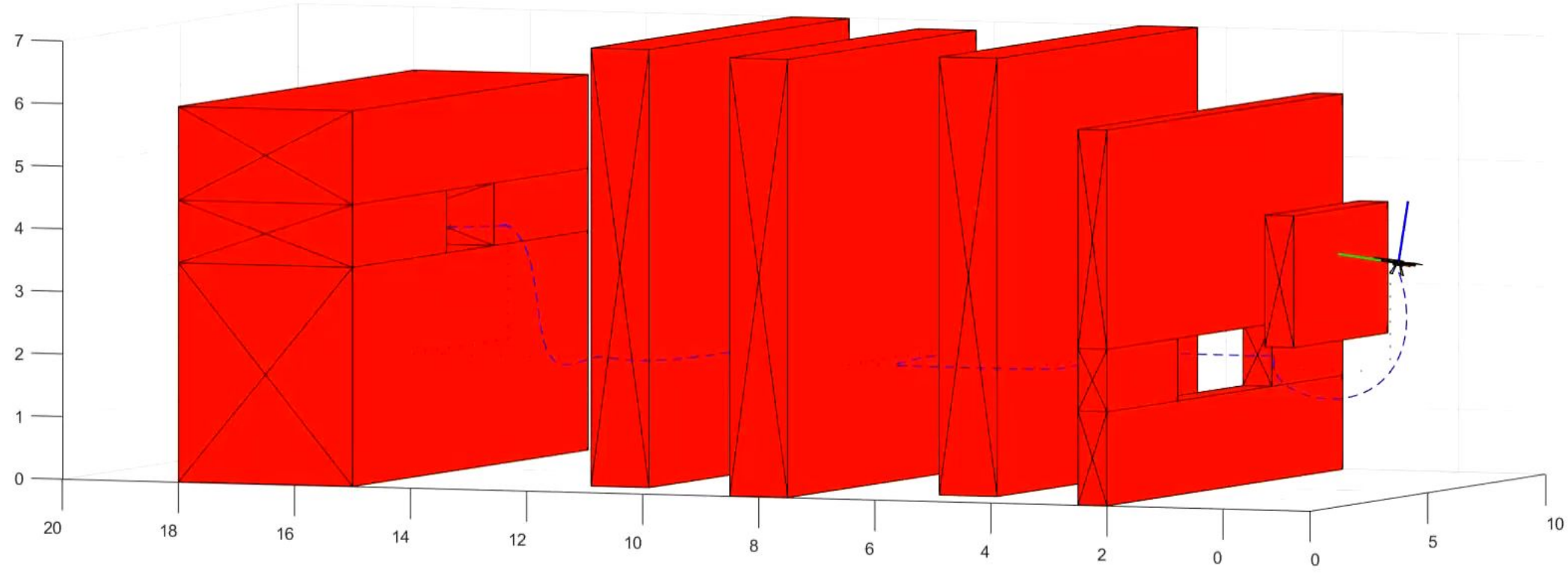

*Lee, T., et al. IEEE (2010)

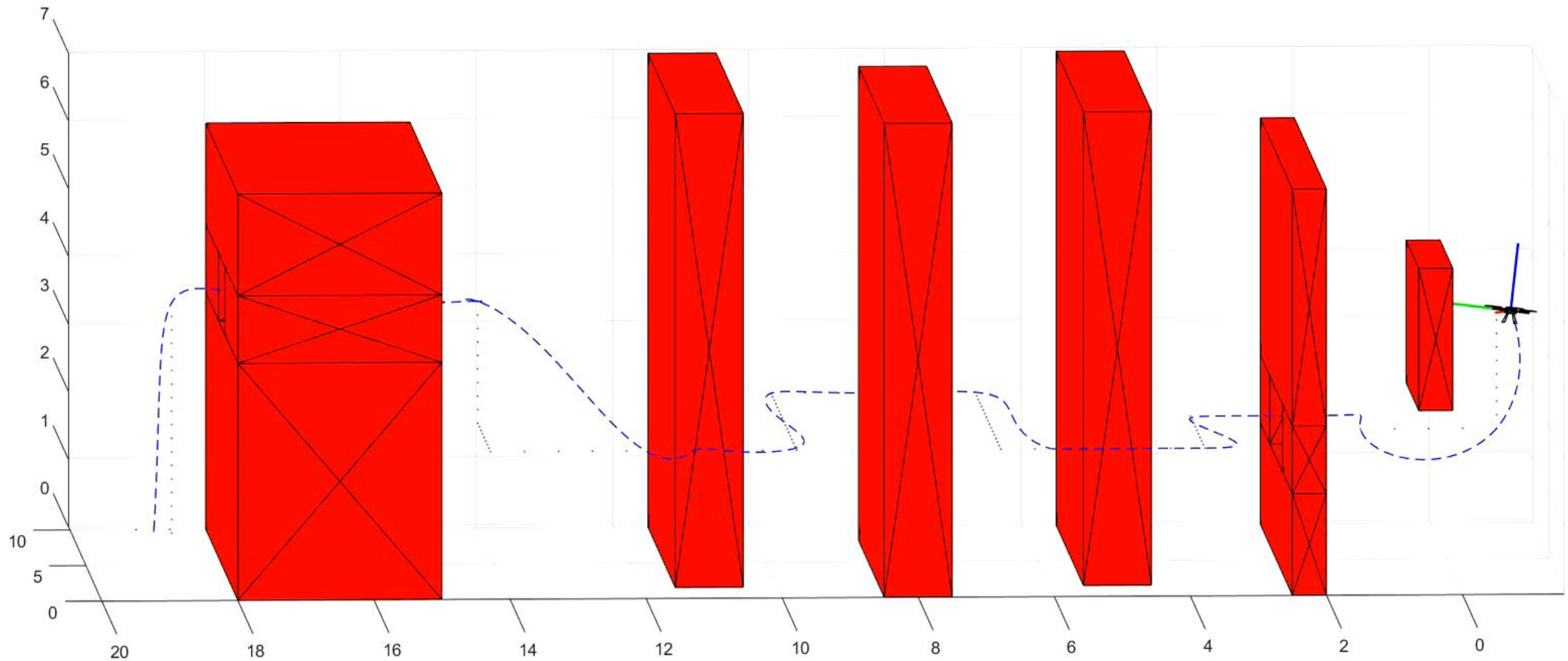
**https://github.com/tu-darmstadt-ros-pkg/hector_quadrotor.git

# Controller structure

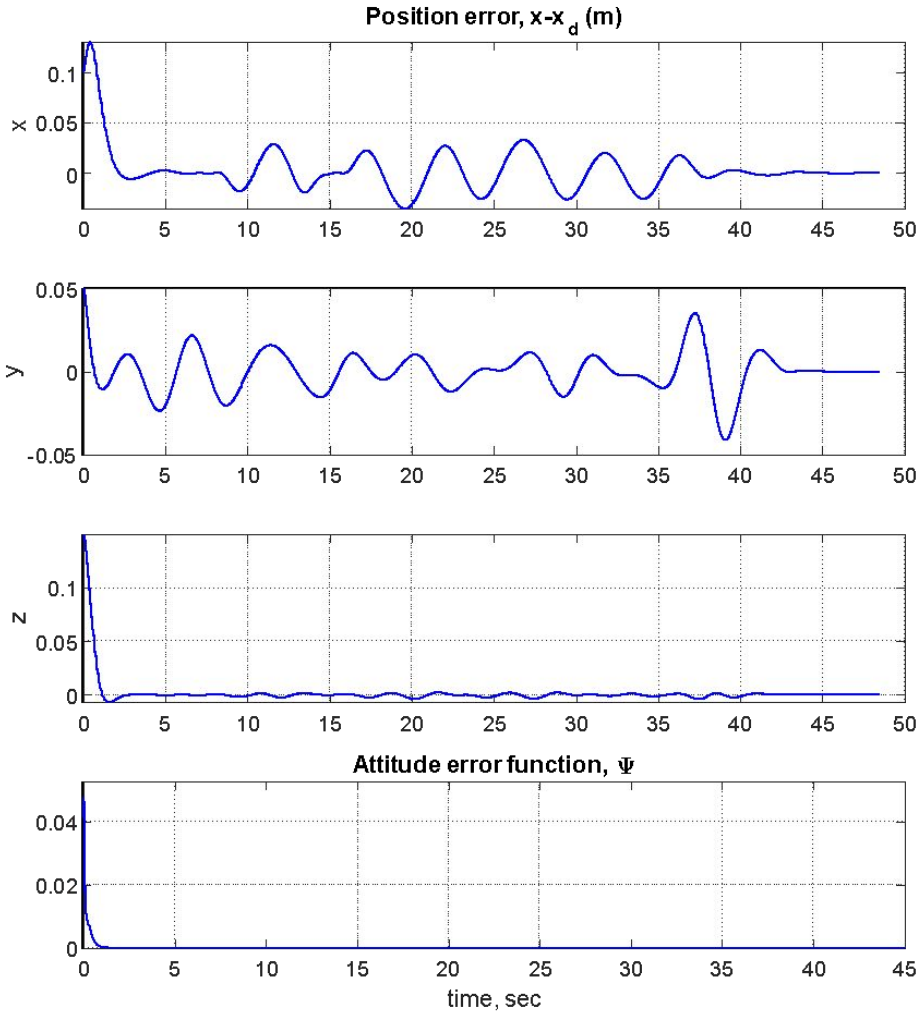


# **Implementation** (No disturbances)

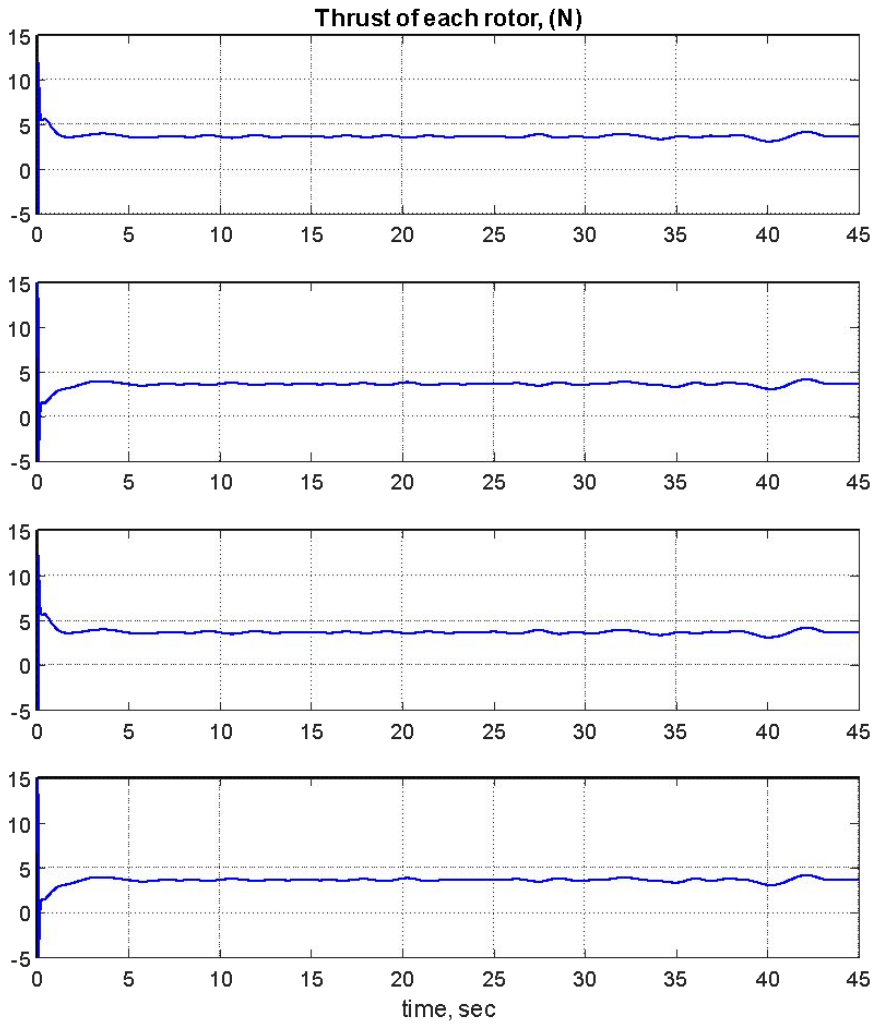



7
6
5
4
3
2
1
0
10
5
0
20
18
16
14
12
10
8
6
4
2
0

# **Results** (No disturbances)

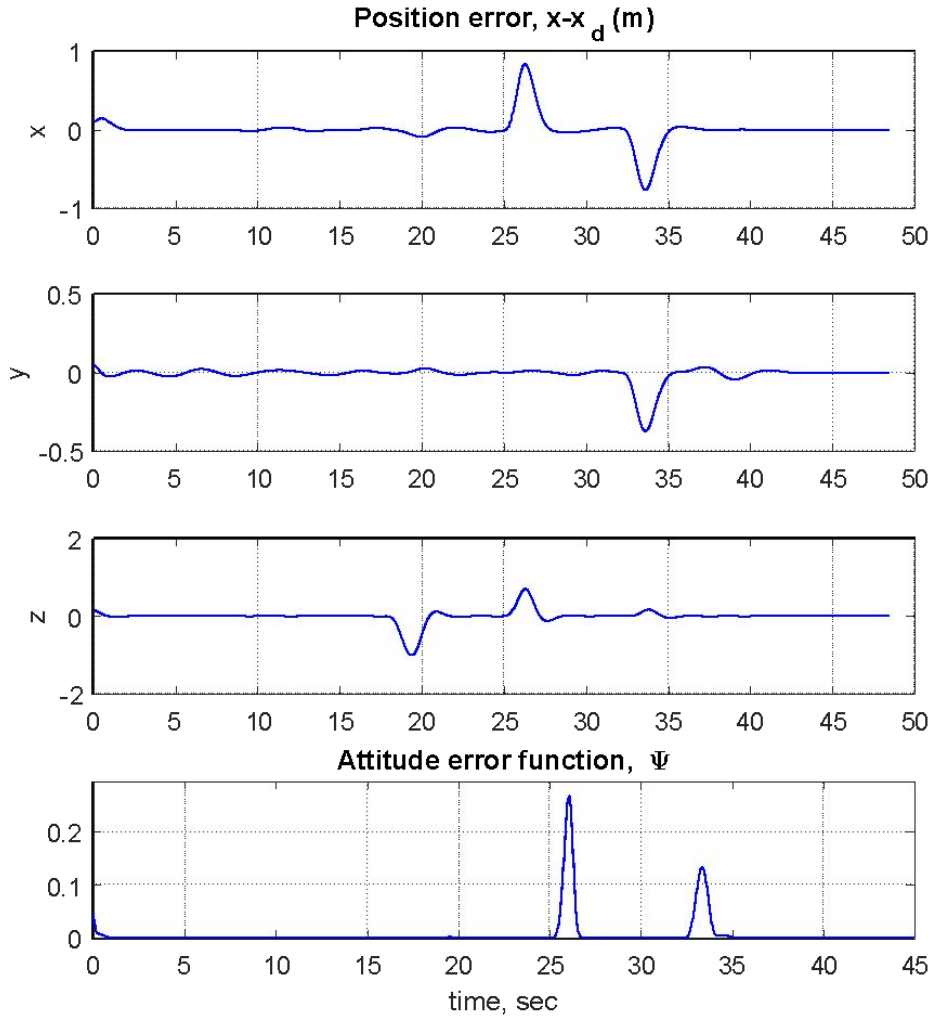



Position error $x - x_d$,
attitude error function $\Psi$

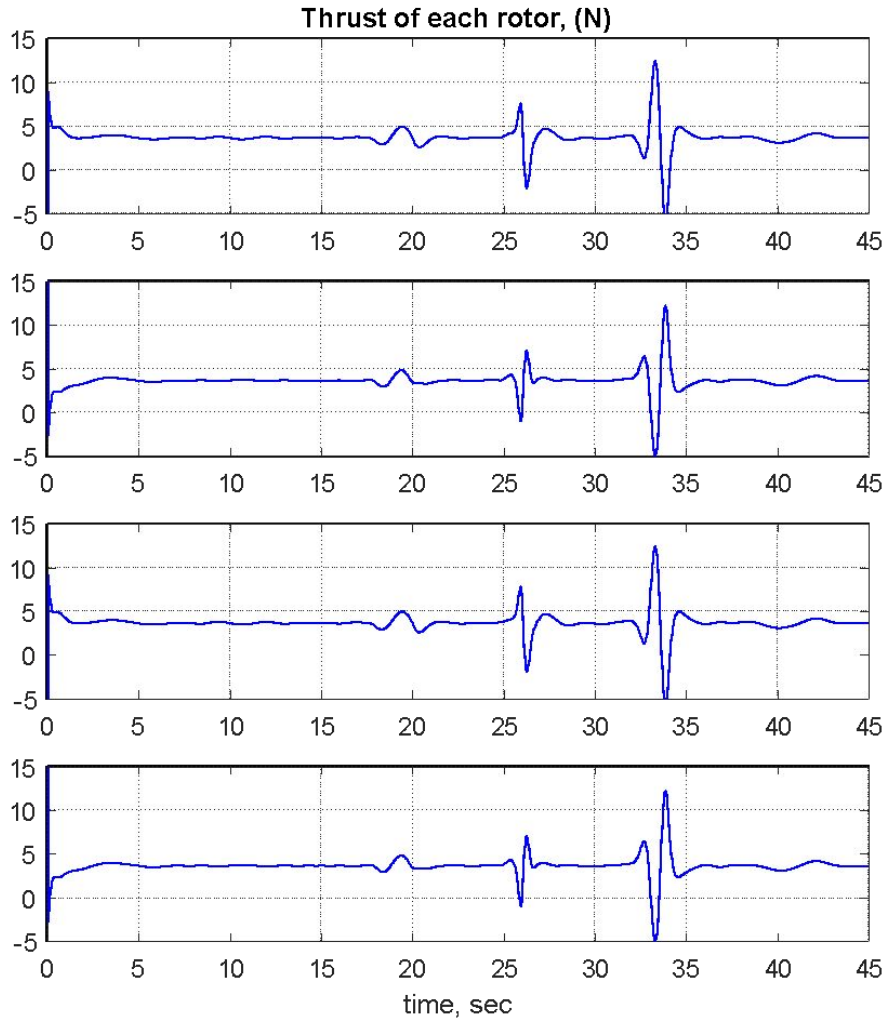



Thrust of each rotor (N)

# **Results** (With disturbances)




Position error $x - x_d$,
attitude error function $\Psi$




Thrust of each rotor (N)

# Reference


- Bry, A., Richter, C., Bachrach, A., & Roy, N. (2015). Aggressive flight of fixed-wing and quadrotor aircraft in dense indoor environments. *The International Journal of Robotics Research,34*(7), 969-1002.
- Mellinger, D., & Kumar, V. (2011). Minimum snap trajectory generation and control for quadrotors. *2011 IEEE International Conference on Robotics and Automation*.
- Lee, T., Leok, M., & Mcclamroch, N. H. (2010). Geometric tracking control of a quadrotor UAV on SE(3). *49th IEEE Conference on Decision and Control (CDC)*.
- Sreenath, K., Lee, T., & Kumar, V. (2013). Geometric control and differential flatness of a quadrotor UAV with a cable-suspended load. *52nd IEEE Conference on Decision and Control*.
- Murray, R. M. (2017). *Mathematical introduction to robotic manipulation*. Taylor & Francis.
- S. (n.d.). Stormmax/quadrotor. Retrieved from https://github.com/stormmax/quadrotor
- T. (2017, March 05). Tu-darmstadt-ros-pkg/hector_quadrotor. Retrieved from https://github.com/tu-darmstadt-ros-pkg/hector_quadrotor

# Appendix

$$\Psi(R, R_d) = \frac{1}{2}\mathrm{tr}\left[I - R_d^T R\right]$$

$$\|m\ddot{x}_d - mg\mathbf{z}_w\| < B, \Psi\big(R(0), R_d(0)\big) \leq \Psi_1 < 1$$

$$\alpha = \sqrt{\psi_1(2-\psi_1)}, e_{v_{max}} = \max\{\|e_v(0)\|, \frac{B}{k_v(1-\alpha)}\}$$

$$W_1 = \begin{bmatrix} \frac{c_1 k_x}{m} & \frac{-c_1 k_v}{2m}(1+\alpha) \\ \frac{-c_1 k_v}{2m}(1+\alpha) & k_v(1-\alpha) - c_1 \end{bmatrix}$$

$$W_{12} = \begin{bmatrix} k_x e_{v_{max}} + \frac{c_1}{m}B & 0 \\ B & 0 \end{bmatrix}$$

$$W_2 = \begin{bmatrix} \frac{c_2 k_R}{\lambda_{max}(J)} & -\frac{c_2 k_\omega}{2\lambda_{min}(J)} \\ -\frac{c_2 k_\omega}{2\lambda_{min}(J)} & k_\omega - c_2 \end{bmatrix}$$

For given constants $k_x, k_v$ if we can choose positive constants $c_1, c_2, k_R, k_\omega$ such that

$$c_1 < \min\left\{k_v(1-\alpha), \frac{4mk_x k_v(1-\alpha)}{k_v^2(1+\alpha)^2 + 4mk_x}, \sqrt{k_x m}\right\}$$

$$c_2 < \min\left\{k_\omega, \frac{4k_\omega k_R \lambda_{min}(J)^2}{k_\omega^2 \lambda_{max}(J) + 4k_R \lambda_{min}(J)^2}, \sqrt{k_R \lambda_{min}(J)}, \sqrt{\frac{2}{2-\alpha} k_R \lambda_{max}(J)}\right\}$$

$$\lambda_{min}(W_2) > \frac{4\|W_{12}\|^2}{\lambda_{min}(W_1)}$$

Then the system is exponentially stable.

Mapping from input (F,M) to rotor thrust

$$\mathbf{u} = \begin{bmatrix} F \\ M_1 \\ M_2 \\ M_3 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & d & 0 & -d \\ -d & 0 & d & 0 \\ c_{\tau f} & -c_{\tau f} & c_{\tau f} & -c_{\tau f} \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{bmatrix}$$