State Estimation ...

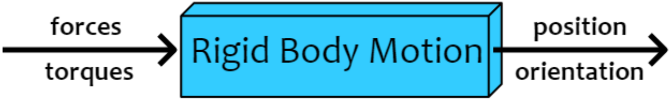
Lecture 19

... of Quadrotor Flight:

- 3D Kinematic Model
  - using Rotation Matrix
  - using Unit Quaternions
  - using Euler Angles
- Exercise

Summary of Rigid Body Motion Eqs

Lecture 19



Eqs. of Motion with the Rotation Matrix

Lecture 19

$$\dot{\boldsymbol{p}}^e = \boldsymbol{v}^e \tag{1a}$$

$$\dot{\boldsymbol{v}}^e = \frac{1}{m}\boldsymbol{f}^e_{total,ext} = \frac{1}{m}\mathbf{R}^e_b\boldsymbol{f}^b_{total,ext} \tag{1b}$$

$$\dot{\mathbf{R}}^e_b = \mathbf{R}^e_b\left[\omega^b\right]_\times \tag{1c}$$

$$\dot{\omega}^b = \left(\mathbf{J}^b\right)^{-1}\left(-\left[\omega^b\right]_\times \mathbf{J}^b\omega^b + \tau^b_c\right) \tag{1d}$$

Eqs. of Motion with Quaternion

Lecture 19

$$\dot{\boldsymbol{p}}^e = \boldsymbol{v}^e \tag{2a}$$

$$\dot{\boldsymbol{v}}^e = \frac{1}{m}\boldsymbol{f}^e_{total,ext} = \frac{1}{m}\mathbf{R}^e_b(\boldsymbol{q})\boldsymbol{f}^b_{total,ext} \tag{2b}$$

$$\dot{\boldsymbol{q}} = \begin{bmatrix} \dot{s} \\ \dot{\mathbf{v}} \end{bmatrix} = \frac{1}{2}\begin{bmatrix} -\boldsymbol{v}^T \\ s\mathbf{I}_3 + [\mathbf{v}]_\times \end{bmatrix}\omega^b = \frac{1}{2}\begin{bmatrix} -v_1\omega^b_x - v_2\omega^b_y - v_3\omega^b_z \\ s\omega^b_x - v_3\omega^b_y + v_2\omega^b_z \\ v_3\omega^b_x + s\omega^b_y - v_1\omega^b_z \\ -v_2\omega^b_x + v_1\omega^b_y + s\omega^b_z \end{bmatrix} \tag{2c}$$

$$\dot{\omega}^b = \left(\mathbf{J}^b\right)^{-1}\left(-\left[\omega^b\right]_\times \mathbf{J}^b\omega^b + \tau^b_c\right) \tag{2d}$$

Eqs. of Motion with Quaternion (cont)
Lecture 19

where:

$$
\begin{aligned}
\mathbf{R}_b^e(\boldsymbol{q}) &= \begin{bmatrix} -\boldsymbol{v} & s\mathbf{I}_3 + [\mathbf{v}]_\times \end{bmatrix} \begin{bmatrix} -\boldsymbol{v}^T \\ s\mathbf{I}_3 + [\mathbf{v}]_\times \end{bmatrix} \\
&= 2 \begin{bmatrix} s^2 + v_1^2 - 0.5 & v_1 v_2 - s v_3 & v_1 v_3 + s v_2 \\ v_1 v_2 + v_3 s & s^2 + v_2^2 - 0.5 & -s v_1 + v_2 v_3 \\ v_1 v_3 - s v_2 & v_2 v_3 + v_1 s & s^2 + v_3^2 - 0.5 \end{bmatrix}
\end{aligned}
$$

Eqs. of Motion with Euler Angles

Lecture 19

$$\dot{\boldsymbol{p}}^e = \boldsymbol{v}^e \tag{3a}$$

$$\dot{\boldsymbol{v}}^e = \frac{1}{m}\boldsymbol{f}^e_{total,ext} = \frac{1}{m}\mathbf{R}^e_b(\boldsymbol{e})\boldsymbol{f}^b_{total,ext} \tag{3b}$$

$$\dot{\boldsymbol{e}} = \begin{bmatrix} r \\ p \\ y \end{bmatrix} = \frac{1}{\cos p}\begin{bmatrix} \cos p & \sin r \cdot \sin p & \cos r \cdot \sin p \\ 0 & \cos r \cdot \cos p & -\sin r \cdot \cos p \\ 0 & \sin r & \cos r \end{bmatrix} \omega^b \tag{3c}$$

$$\dot{\omega}^b = \left(\mathbf{J}^b\right)^{-1}\left(-\left[\omega^b\right]_\times \mathbf{J}^b\omega^b + \tau^b_c\right) \tag{3d}$$

Eqs. of Motion with Euler Angles (cont)

Lecture 19

where:

$$\mathbf{R}_b^e(\boldsymbol{e}) = \mathbf{R}_z(y)\mathbf{R}_y(p)\mathbf{R}_x(r)$$

$$\mathbf{R}_z(y) = \begin{bmatrix} \cos y & -\sin y & 0 \\ \sin y & \cos y & 0 \\ 0 & 0 & 1 \end{bmatrix} \mathbf{R}_y(p) = \begin{bmatrix} \cos p & 0 & \sin p \\ 0 & 1 & 0 \\ -\sin p & 0 & \cos p \end{bmatrix} \mathbf{R}_x(r) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos r & -\sin r \\ 0 & \sin r & \cos r \end{bmatrix}$$

Observations

Lecture 19

In a KF algorithm, special care must be taken for the:

- The rotation matrix to be kept orthonormal;
- The quaternion to be kept unitary;

The Euler Angles do not suffer from this problem, although it is a good idea to keep them to within a constant 2π interval.

- See [Diebel,2006] for a compendium of attitude representations and equivalences

Simple Integration Technique

Lecture 19

- Given a differential equation $\dot{y}(t) = f(y, u, t)$, where $y(t)$ is unknown, $u(t)$ is a known input function, and the initial condition $y(0) = y_0$ is known also, we can obtain an approximation of $y(k\Delta t)$, $\tilde{y}(k\Delta t)$ as

$$\tilde{y}(k\Delta t) = \tilde{y}((k-1)\Delta t) + \dot{y}(t)\Delta t$$

$$\boxed{\tilde{y}(k\Delta t) = \tilde{y}((k-1)\Delta t) + f(y(k\Delta t), u(k\Delta t), k\Delta t)\Delta t}$$

$$\Rightarrow \tilde{y}(\Delta t) = y_0 + f(y_0, u(0), 0)\Delta t, \;\; \tilde{y}(2\Delta t) = \tilde{y}(\Delta t) + f(\tilde{y}(\Delta t), u(\Delta t), \Delta t)\Delta t$$

$$\tilde{y}(3\Delta t) = \tilde{y}(2\Delta t) + f(\tilde{y}(2\Delta t), u(2\Delta t)2\Delta t)\Delta t, \ldots$$

- The global error $|y(k\Delta t) - \tilde{y}(k\Delta t)|$ is proportional to Δt
- The smaller Δt, the better the approximation

Exercise: Quadrotor Flight Estimation

Lecture 19

`git clone git@github.com:lkdo/ex_quad_flight_ukf.git`

- `main.py`
- `pandaapp.py` and `res` folder
  - `CF21_plus.egg`
  - `CF21_cross.egg`
  - `tex` folder (texture images)
- `rigidybody.py`
- `ftaucf.py`
- `logger.py` and `plotter.py`

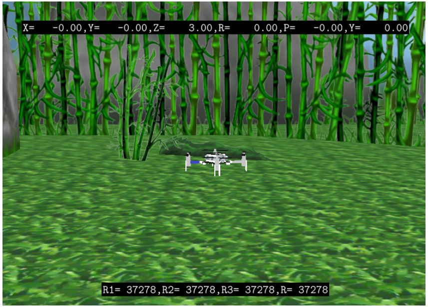
- `python3 (v3.9.2)`

Dependency packages:

- `numpy (v1.20.1)`
- `matplotlib (v3.3.4)`
- `panda3d (v1.10.3)`

Exercise: Quadrotor Flight Estimation

Lecture 9



- `py main.py [ "step" | "ramp" | "sin" | "manual" ]`

Exercise: Quadrotor Flight Estimation
Lecture 19

- Explore and inspect the simulation template
- Implement a UKF based on the Euler Angles motion model (files `ukf.py` and `rigidbody.py`)
- Discussion about the averaging of rotations, relevant because of the sigma-points weighted averaging mechanism.

ROS: C++ EKF & UKF implementation for 3D Motion
Lecture 19

- GitHub https://github.com/cra-ros-pkg/robot_localization, branches “noeticdevel” (ROS1) and “ros2” (ROS2);
- Check-out src/ekf.cpp and src/ukf.cpp;
- Project: Consider a Cubature KF implementation in this framework.

Well Done!