Drones Demystified!

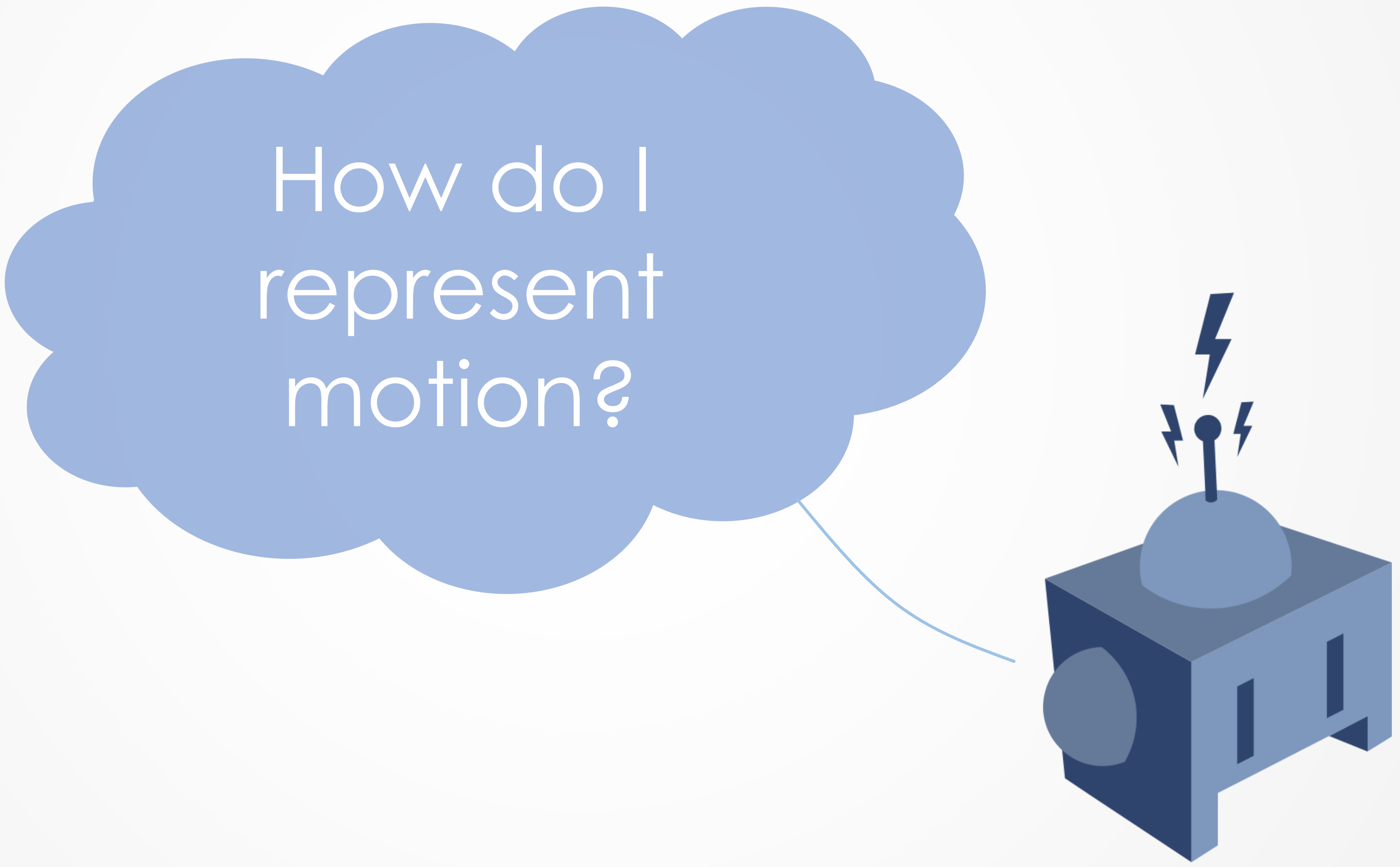
K. Alexis, C. Papachristos, Autonomous Robots Lab, University of Nevada, Reno

A. Tzes, Autonomous Robots & Intelligent Systems Lab, NYU Abu Dhabi

Drones Demystified!

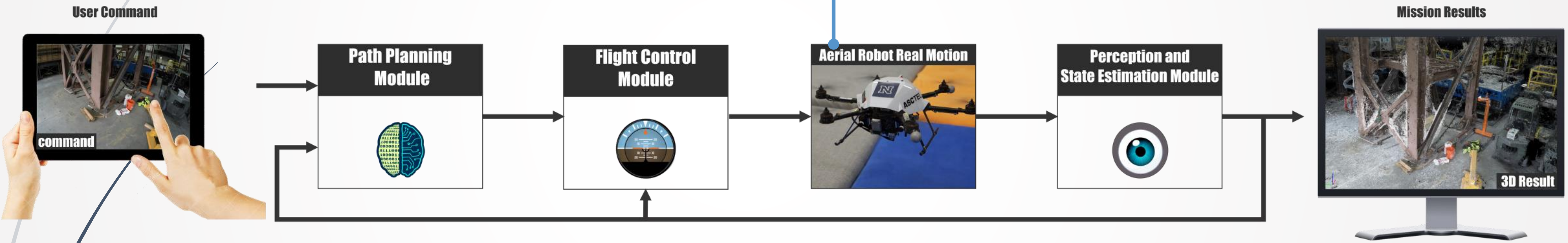
Topic: Coordinate Frames

The Aerial Robot Loop



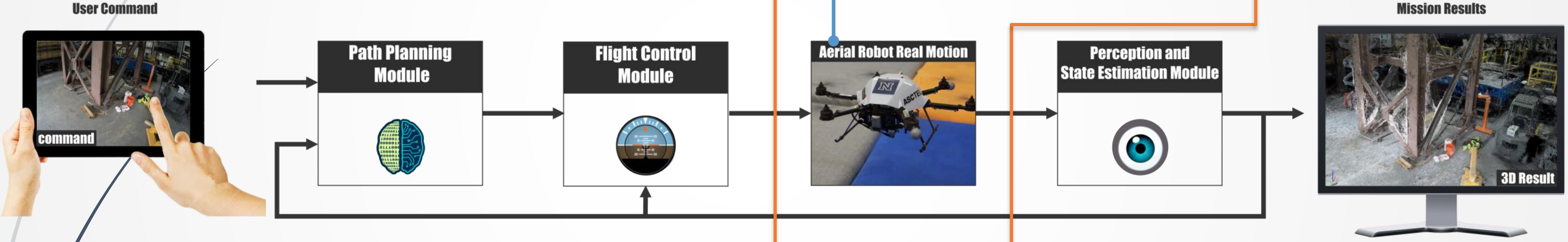
The Aerial Robot Loop

- Real-life Aerial Robot expressing its dynamic behavior in response to the control inputs and external disturbances



The Aerial Robot Loop

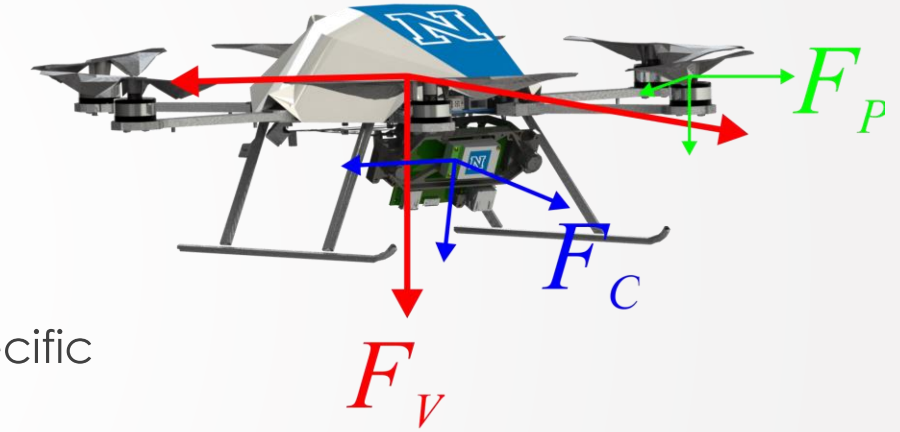
- Real-life Aerial Robot expressing its dynamic behavior in response to the control inputs and external disturbances

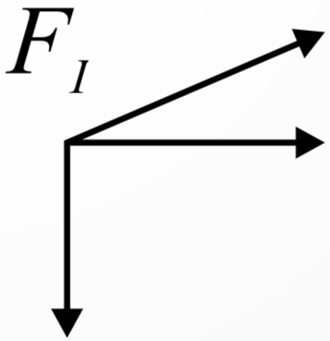


Section 1 of our course

Coordinate Frames

- In Guidance, Navigation and Control of aerial robots, reference coordinate frames are fundamental.
- Describe the relative position and orientation of:
  - Aerial Robot ***relative*** to the Inertial Frame
  - On-board Camera ***relative*** to the Aerial Robot body
  - Aerial Robot ***relative*** to Wind Direction
- Some expressions are easier to formulate in specific frames:
  - Newton's law
  - Aerial Robot Attitude
  - Aerodynamic forces/moments
  - Inertial Sensor data
  - GPS coordinates
  - Camera frames

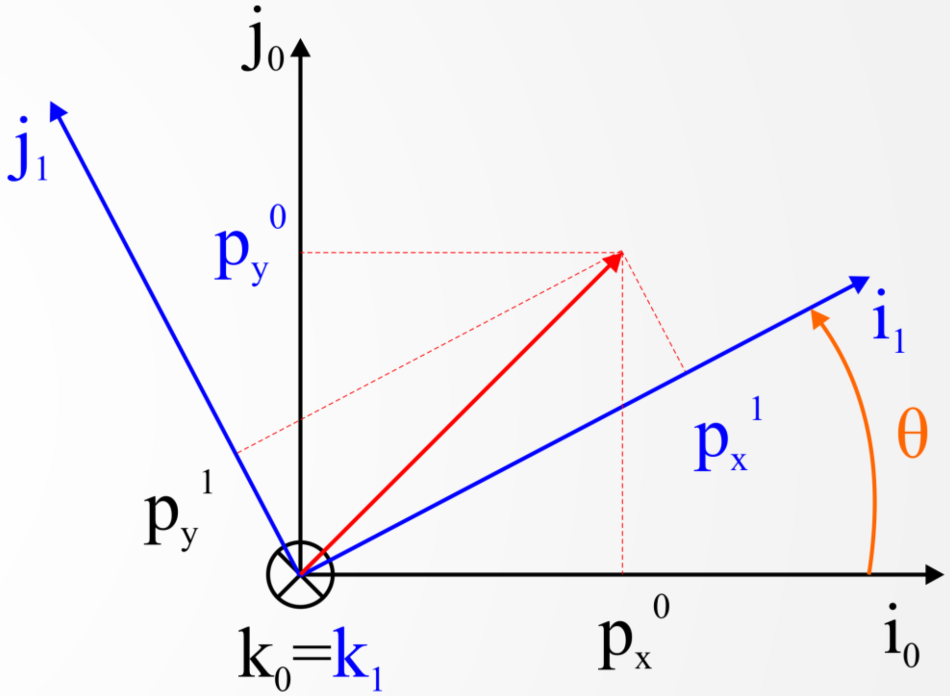




Rotation of Reference Frame

- Rotation around the k-axis

$$\mathbf{p} = p_x^0\mathbf{i}^0 + p_y^0\mathbf{j}^0 + p_z^0\mathbf{k}^0$$
$$\mathbf{p} = p_x^1\mathbf{i}^1 + p_y^1\mathbf{j}^1 + p_z^1\mathbf{k}^1$$
$$\mathbf{p}^1 = \begin{pmatrix} p_x^1 \\ p_y^1 \\ p_z^1 \end{pmatrix} = \begin{pmatrix} \mathbf{i}^1\mathbf{i}^0 & \mathbf{i}^1\mathbf{j}^0 & \mathbf{i}^1\mathbf{k}^0 \\ \mathbf{j}^1\mathbf{i}^0 & \mathbf{j}^1\mathbf{j}^0 & \mathbf{j}^1\mathbf{k}^0 \\ \mathbf{k}^1\mathbf{i}^0 & \mathbf{k}^1\mathbf{j}^0 & \mathbf{k}^1\mathbf{k}^0 \end{pmatrix} \begin{pmatrix} p_x^0 \\ p_y^0 \\ p_z^0 \end{pmatrix}$$



$$\mathbf{p}^1 = \mathcal{R}_0^1\mathbf{p}^0,\ \mathcal{R}_0^1 = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Rotation of Reference Frame

- Rotation around the i-axis

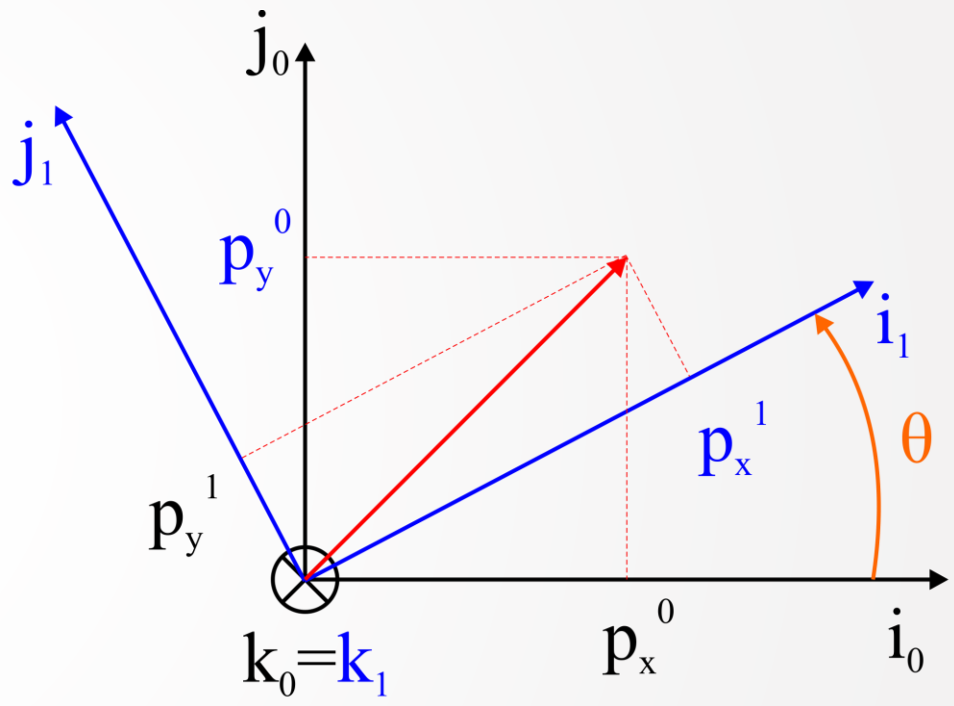
$$\mathcal{R}_0^1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & \sin\theta \\ 0 & -\sin\theta & \cos\theta \end{bmatrix}$$

- Rotation around the j-axis

$$\mathcal{R}_0^1 = \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix}$$

- Rotation around the k-axis

$$\mathcal{R}_0^1 = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



- Orthonormal matrix properties
  - $(\mathcal{R}_a^b)^{-}1 = (\mathcal{R}_a^b)^T = \mathcal{R}_b^a$
  - $\mathcal{R}_b^c \mathcal{R} a^b = \mathcal{R}_a^c$
  - $\det(\mathcal{R}_a^b) = 1$

Rotation of Reference Frame

- Rotation around the i-axis

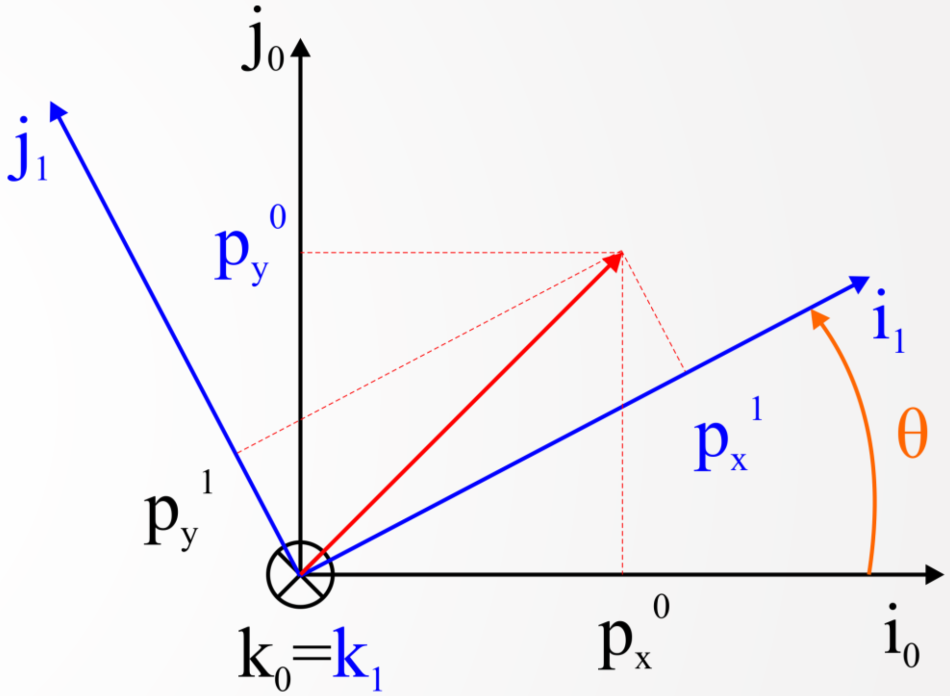
$$\mathcal{R}_0^1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & \sin\theta \\ 0 & -\sin\theta & \cos\theta \end{bmatrix}$$

- Rotation around the j-axis

$$\mathcal{R}_0^1 = \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix}$$

- Rotation around the k-axis

$$\mathcal{R}_0^1 = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



- Orthonormal matrix properties
  - $(\mathcal{R}_a^b)^-1 = (\mathcal{R}_a^b)^T = \mathcal{R}_b^a$
  - $\mathcal{R}_b^c \mathcal{R}a^b = \mathcal{R}_a^c$
  - $\det(\mathcal{R}_a^b) = 1$

Rotation of Reference Frame

- Let $q = |\mathbf{q}|,\ p = |\mathbf{p}|$

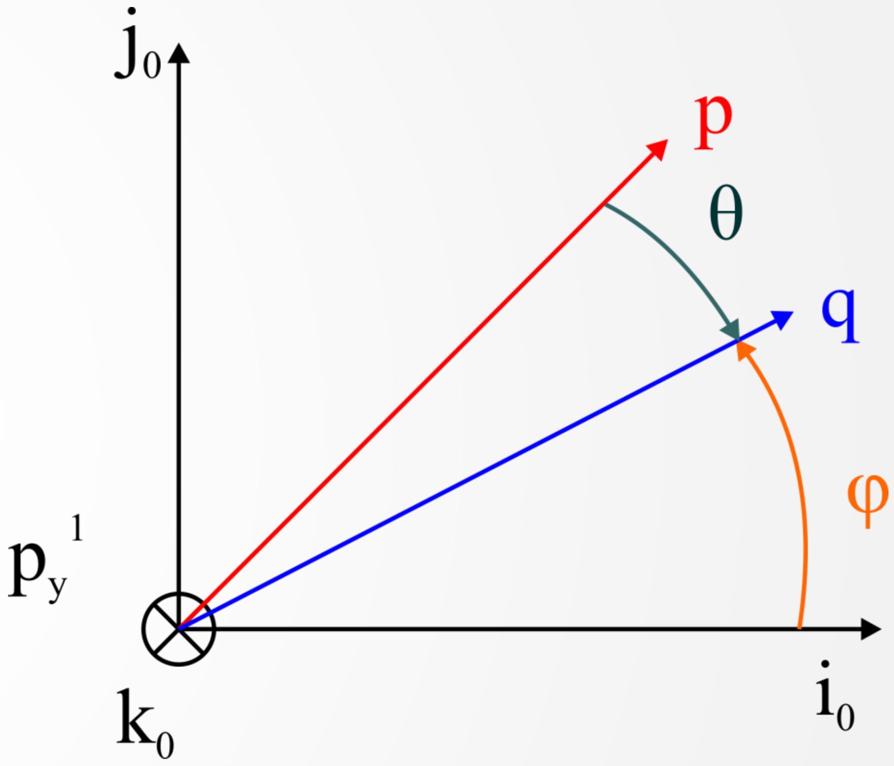
$$\mathbf{p} = \begin{bmatrix} p\cos(\theta+\phi) \\ p\sin(\theta+\phi) \\ 0 \end{bmatrix} = \begin{bmatrix} p\cos\theta\cos\phi - p\sin\theta\sin\phi \\ p\sin\theta\cos\phi + p\cos\theta\sin\phi \\ 0 \end{bmatrix} =$$
$$\begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p\cos\phi \\ p\sin\phi \\ 0 \end{bmatrix}$$

- And define:

$$\mathbf{q} = \begin{bmatrix} p\cos\phi \\ p\sin\phi \\ 0 \end{bmatrix}$$

- Then:

$$\mathbf{p} = (\mathcal{R}_0^1)^T \mathbf{q} \Rightarrow \mathbf{q} = \mathcal{R}_0^1 \mathbf{p}$$



Rotation of Reference Frame

- Let $q = |\mathbf{q}|,\ p = |\mathbf{p}|$

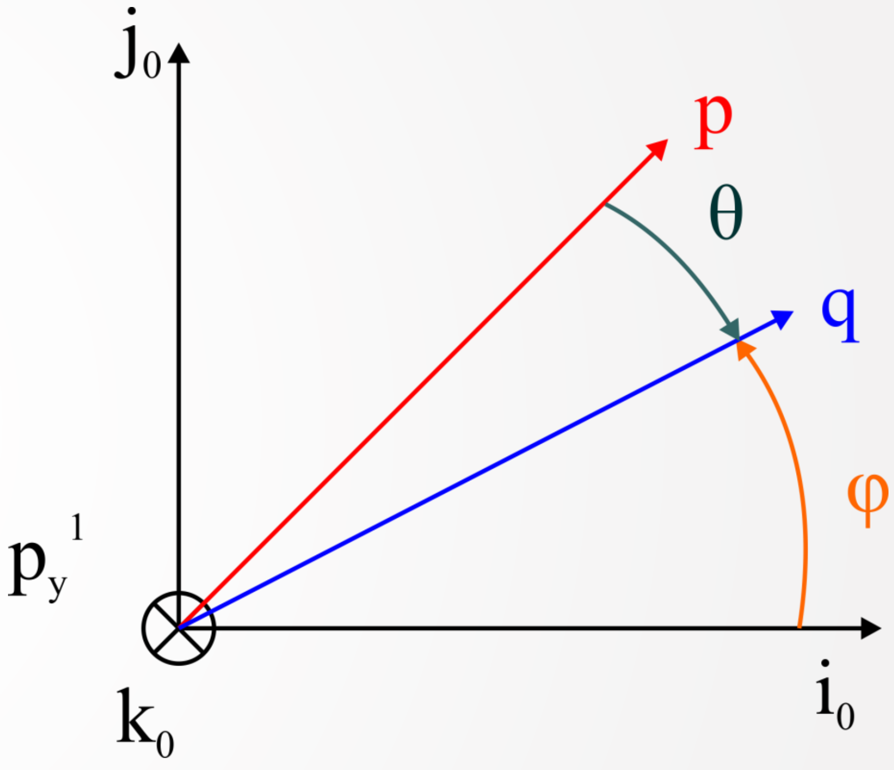
$$\mathbf{p} = \begin{bmatrix} p\cos(\theta+\phi) \\ p\sin(\theta+\phi) \\ 0 \end{bmatrix} = \begin{bmatrix} p\cos\theta\cos\phi - p\sin\theta\sin\phi \\ p\sin\theta\cos\phi + p\cos\theta\sin\phi \\ 0 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p\cos\phi \\ p\sin\phi \\ 0 \end{bmatrix}$$

- And define:

$$\mathbf{q} = \begin{bmatrix} p\cos\phi \\ p\sin\phi \\ 0 \end{bmatrix}$$

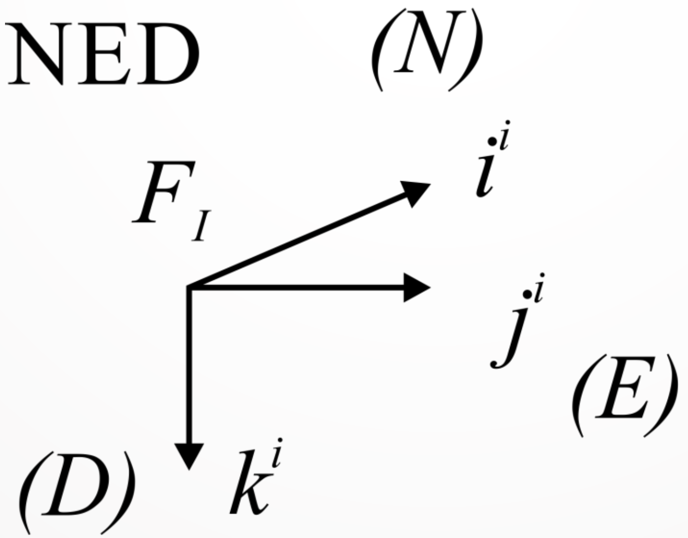
- Then:

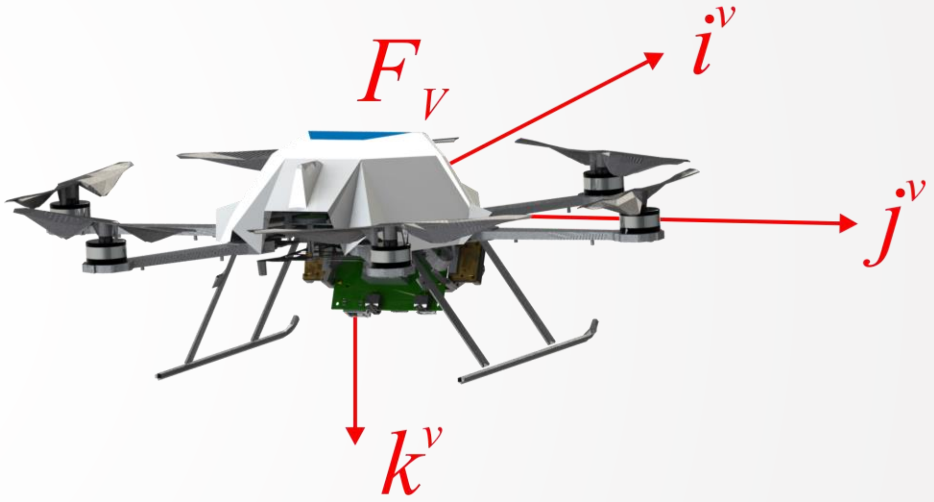
$$\boxed{\mathbf{p} = (\mathcal{R}_0^1)^T\mathbf{q} \Rightarrow \mathbf{q} = \mathcal{R}_0^1\mathbf{p}}$$



Inertial & Vehicle Frames

- Vehicle and Inertial frame have the same orientation.
- Vehicle frame is fixed at the Center of Mass (CoM).
- Both considered as “NED” frames (North-East-Down).





How to represent orientation?

Euler Angles

$$[\phi, \theta, \psi]$$

- Advantages:
  - Intuitive – directly related with the axis of the vehicle.
- Disadvantages:
  - Singularity – Gimbal Lock.

Quaternions

$$[q_1, q_2, q_3, q_4]$$

- Advantages:
  - Singularity-free.
  - Computationally efficient.
- Disadvantages:
  - Non-intuitive

How to represent orientation?

Euler Angles

$$[\phi, \theta, \psi]$$

- Advantages:
  - Intuitive – directly related with the axis of the vehicle.
- Disadvantages:
  - Singularity – Gimbal Lock.

We will start here…

Quaternions

$$[q_1, q_2, q_3, q_4]$$

- Advantages:
  - Singularity-free.
  - Computationally efficient.
- Disadvantages:
  - Non-intuitive

ϕ - roll

θ - pitch

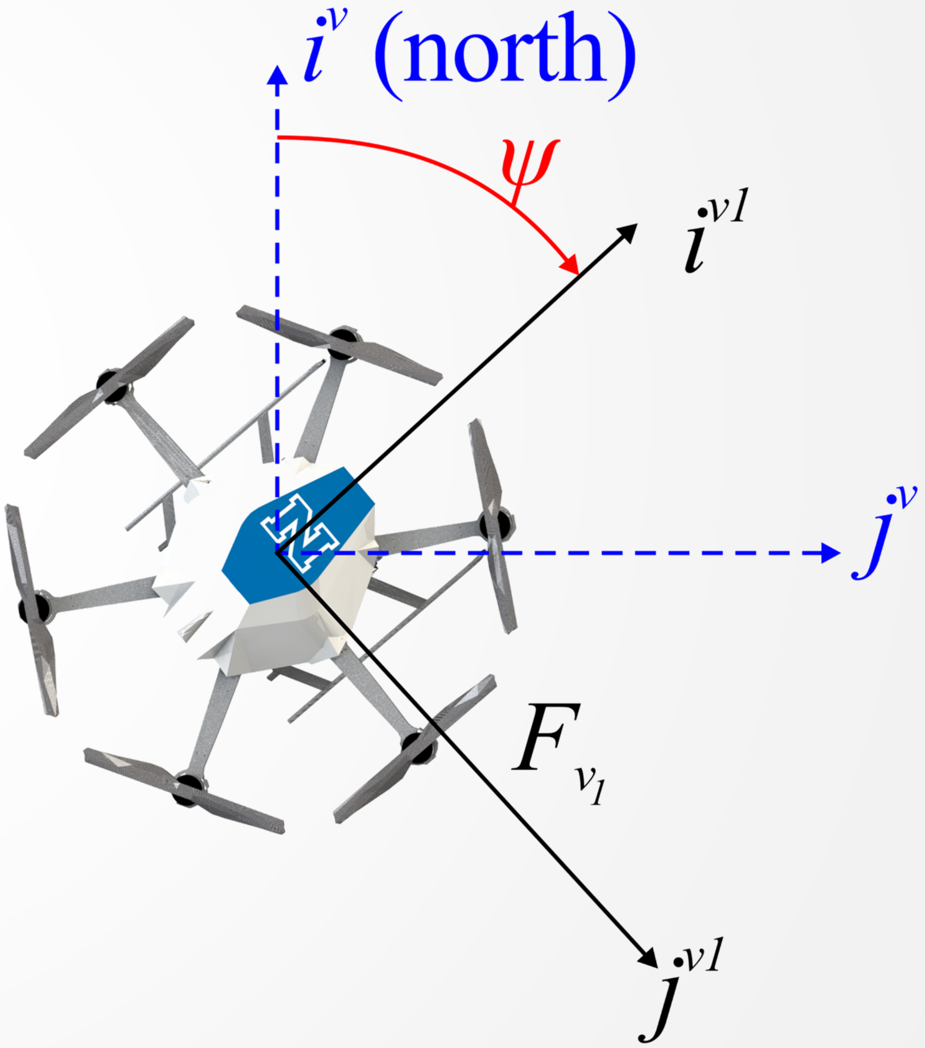
ψ - yaw

Vehicle-1 Frame

$$\mathbf{p}^{v_1} = \mathcal{R}_v^{v_1}\mathbf{p}^v,$$

$$\mathcal{R}_v^{v_1} = \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- ψ represents the yaw angle

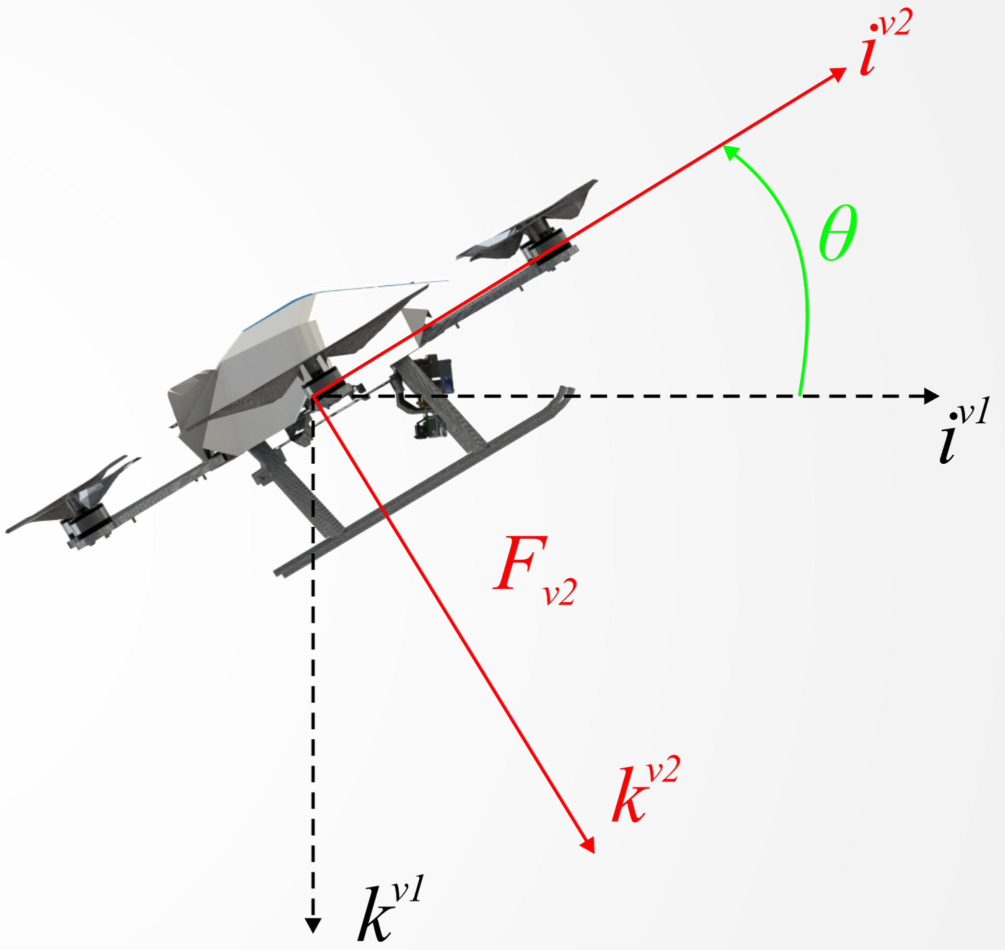


Vehicle-2 Frame

$$\mathbf{p}^{v_2} = \mathcal{R}_{v_1}^{v_2}\mathbf{p}^{v_1},$$

$$\mathcal{R}_{v_1}^{v_2} = \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix}$$

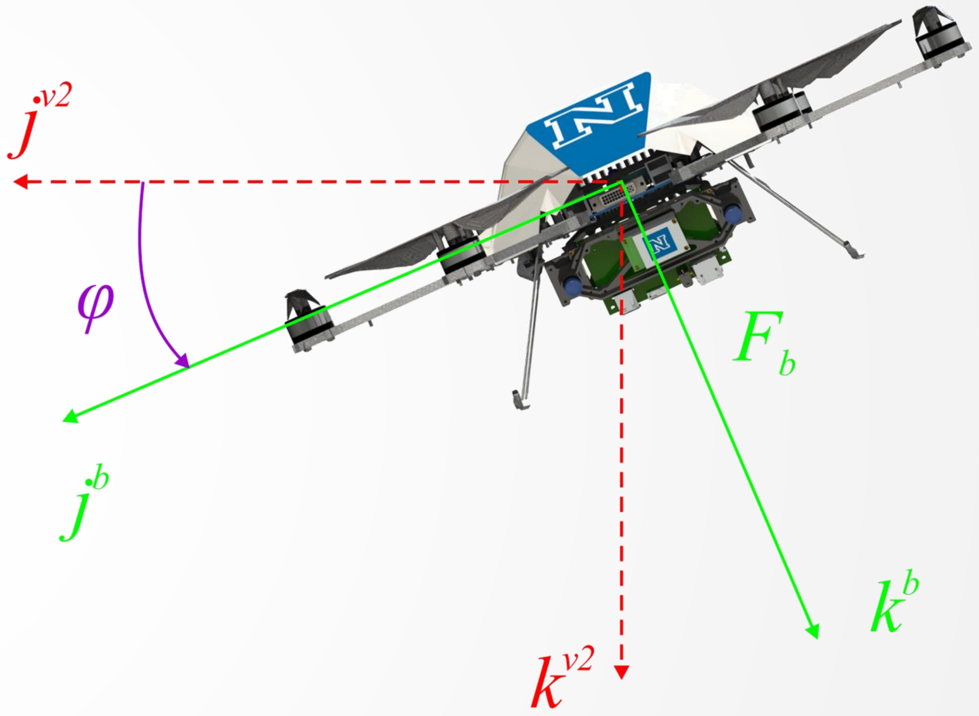
- θ represents the pitch angle



Body Frame

$$\mathbf{p}^b = \mathcal{R}_{v_2}^b \mathbf{p}^{v_2},$$

$$\mathcal{R}_{v_2}^b = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix}$$



- φ represents the roll angle

Inertial Frame to Body Frame

- Let:

$$\mathcal{R}_v^b(\phi,\theta,\psi) = \mathcal{R}_{v_2}(\phi)\mathcal{R}_{v_1}^{v_2}(\theta)\mathcal{R}_v^{v_1}(\psi)$$

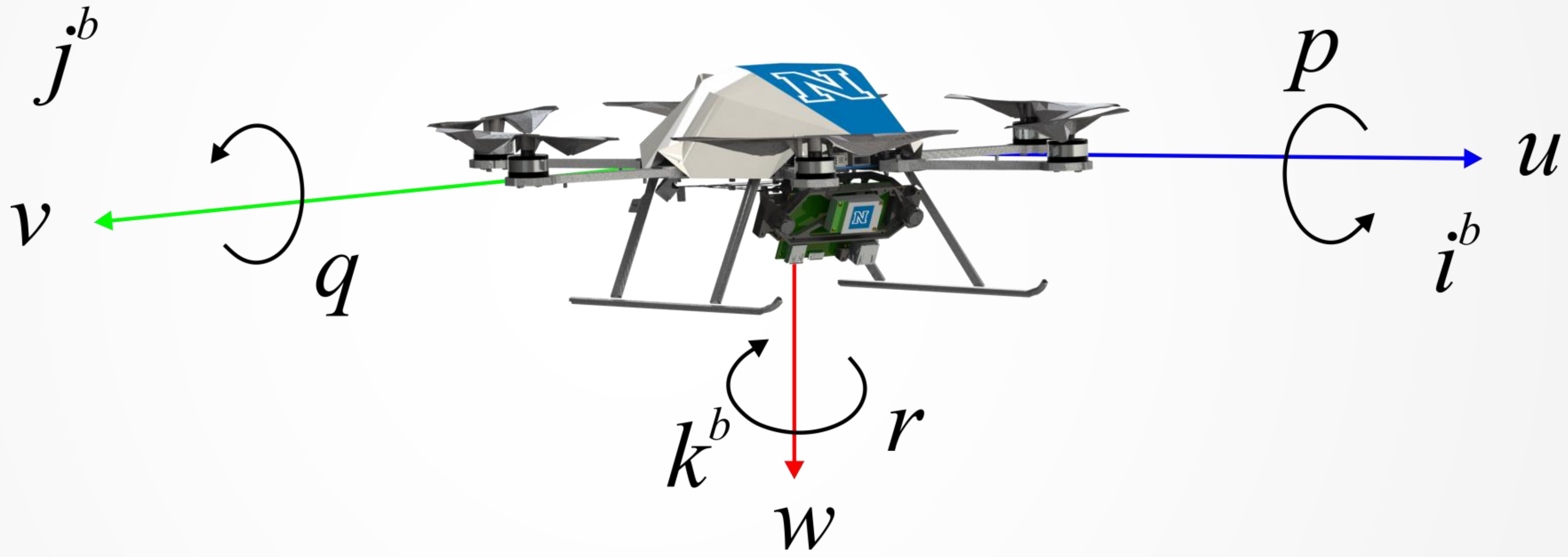
$$= \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix} \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} c_\theta c_\psi & c_\theta s_\psi & -s_\theta \\ s_\phi s_\theta c_\psi - c_\phi s_\psi & s_\phi s_\theta s_\psi + c_\phi c_\psi & s_\phi c_\theta \\ c_\phi s_\theta c_\psi + s_\phi s_\psi & c_\phi s_\theta s_\psi - s_\phi c_\psi & c_\phi c_\theta \end{bmatrix}$$

- Then:

$$\mathbf{p}^b = \mathcal{R}_v^b \mathbf{p}^v$$

Further Application to Robot Kinematics



- $[p,q,r]$: body angular rates
- $[u,v,w]$: body linear velocities

Relate Translational Velocity-Position

- Let *[u,v,w]* represent the body linear velocities

$$\frac{d}{dt}\begin{bmatrix} p_n \\ p_e \\ p_d \end{bmatrix} = \mathcal{R}_b^v \begin{bmatrix} u \\ v \\ w \end{bmatrix} = (\mathcal{R}_v^b)^T \begin{bmatrix} u \\ v \\ w \end{bmatrix}$$

- Which gives:

$$\begin{bmatrix} \dot{p}_n \\ \dot{p}_e \\ \dot{p}_d \end{bmatrix} = \begin{bmatrix} c_\theta c_\psi & s_\phi s_\theta c_\psi - c_\phi s_\phi & c_\phi s_\theta c_\psi + s_\phi s_\psi \\ c_\theta s_\psi & s_\phi s_\theta s_\psi + c_\phi c_\psi & c_\phi s_\theta s_\psi - s_{c\psi} \\ -s_\theta & s_\phi c_\theta & c_{c\theta} \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix}$$

Body Rates – Euler Rates

- Let *[p,q,r]* denote the body angular rates

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} \dot\phi \\ 0 \\ 0 \end{bmatrix} + \mathcal{R}_{v_2}^{b}(\phi) \begin{bmatrix} 0 \\ \dot\theta \\ 0 \end{bmatrix} + \mathcal{R}_{v_2}^{b}(\phi)\mathcal{R}_{v_1}^{v_2}(\theta) \begin{bmatrix} 0 \\ 0 \\ \dot\psi \end{bmatrix} =$$

$$\begin{bmatrix} \dot\phi \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} 0 \\ \dot\theta \\ 0 \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ \dot\psi \end{bmatrix} =$$

$$\begin{bmatrix} 1 & 0 & -\sin\theta \\ 0 & \cos\phi & \sin\phi\cos\theta \\ 0 & -\sin\phi & \cos\phi\cos\theta \end{bmatrix} \begin{bmatrix} \dot\phi \\ \dot\theta \\ \dot\psi \end{bmatrix}$$

- Inverting this expression:

$$\begin{bmatrix} \dot\phi \\ \dot\theta \\ \dot\psi \end{bmatrix} = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi\sec\theta & \cos\phi\sec\theta \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix}$$

How to represent orientation?

Euler Angles

$$[\phi, \theta, \psi]$$

- Advantages:
  - Intuitive – directly related with the axis of the vehicle.
- Disadvantages:
  - Singularity – Gimbal Lock.

Quaternions

$$[q_1, q_2, q_3, q_4]$$

- Advantages:
  - Singularity-free.
  - Computationally efficient.
- Disadvantages:
  - Non-intuitive

A glimpse…

Quaternions

- Complex numbers form a plane : their operations are highly related with 2-dimensional geometry.
- In particular, multiplication by a unit complex number:

$$|z^2| = 1$$

which can all be written:

$$z = e^{i\theta}$$

gives a rotation

$$\mathcal{R}_z(w) = zw$$

by angle θ

Quaternions

- Theorem by Euler states that any given sequence of rotations can be represented as a single rotation about a *fixed-axis*
- Quaternions provide a convenient parametrization of this effective axis and a rotation angle:

$$\bar{q} = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix} = \begin{bmatrix} \bar{E} \sin \frac{\zeta}{2} \\ \cos \frac{\zeta}{2} \end{bmatrix}$$

- Where $\bar{E}$ is a unit vector and ζ is a positive rotation about $\bar{E}$

Quaternions

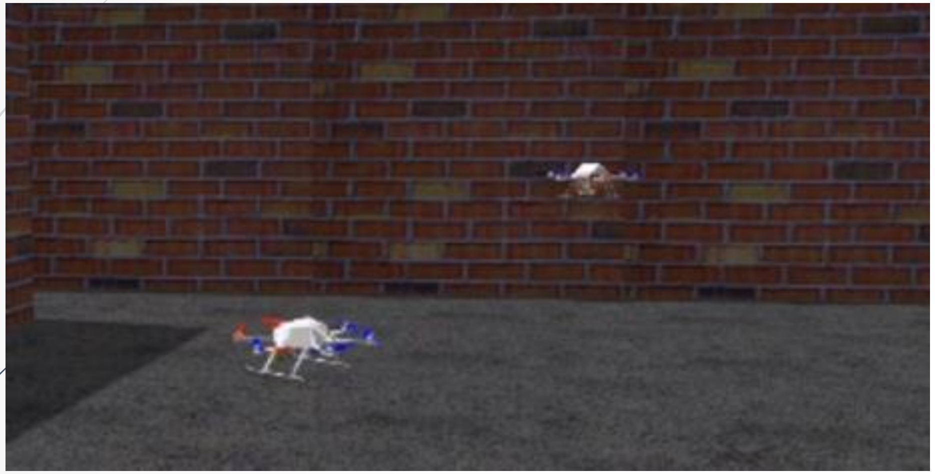
- Note that $|\bar{q}| = 1$ and therefore there are only 3 degrees of freedom in this formulation also.
- If $\bar{q}$ represents the rotational transformation from the reference frame A to the reference frame B, the frame A is aligned with B when frame A is rotated by ζ radians around $\bar{E}$
- This representation is connected with the Euler angles form, according to the following expression:

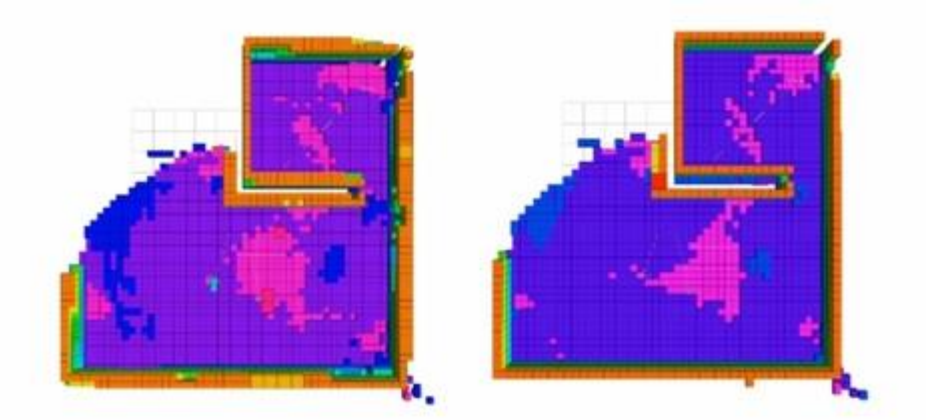
$$\begin{bmatrix} \sin\theta \\ \phi \\ \psi \end{bmatrix} = \begin{bmatrix} -2(q_2q_4 + q_1q_3) \\ \arctan 2[2(q_2q_3 - q_1q_4), 1 - 2(q_1^2 + q_2^2)] \\ \arctan 2[2(q_1q_2 - q_3q_4), 1 - 2(q_2^2 + q_3^2)] \end{bmatrix}$$

Quaternions

- This representation has the great **advantage** of being:
  - Singularity-free and
  - Computationally efficient to do state propagation (typically within an Extended Kalman Filter)
- On the other hand, it has one main **disadvantage**, namely being far less intuitive.

Development Framework: RotorS





- RotorS is a MAV gazebo simulator developed by the Autonomous Systems Lab at ETH Zurich. It provides several multirotor models based on real-flight system identification. There are simulated sensors coming with the simulator such as an IMU, a generic odometry sensor, and a Visual-Inertial sensor, which can be mounted on the multirotor. This package also contains some example controllers, basic worlds, a joystick interface, and example launch files. Below we provide the instructions necessary for getting started. See RotorS' wiki for more instructions and examples (https://github.com/ethz-asl/rotors_simulator/wiki).

Code Example

- Python Coordinate Transformations Example
  - https://github.com/unr-arl/drones_demystified/tree/master/python/coord-trans
  - `Functionality identical to default settings of MATLAB Aerospace Toolbox`
  - `Implements: Quaternion-to/from-RotationMatrix, Quaternion-to/from-RollPitchYaw`
  - `python QuatEulerMain.py`

Find out more

- http://page.math.tu-berlin.de/~plaue/plaue_intro_quats.pdf
- http://mathworld.wolfram.com/RotationMatrix.html
- http://mathworld.wolfram.com/EulerAngles.html
- http://blog.wolframalpha.com/2011/08/25/quaternion-properties-and-interactive-rotations-with-wolframalpha/
- http://www.mathworks.com/discovery/rotation-matrix.html
- http://www.mathworks.com/discovery/quaternion.html?refresh=true
- http://www.cprogramming.com/tutorial/3d/rotationMatrices.html
- http://www.cprogramming.com/tutorial/3d/quaternions.html

- **Help with Linear Algebra?** https://www.khanacademy.org/math/linear-algebra
- **Always check:** http://www.kostasalexis.com/literature-and-links1.html

Thank you!

Please ask your question!