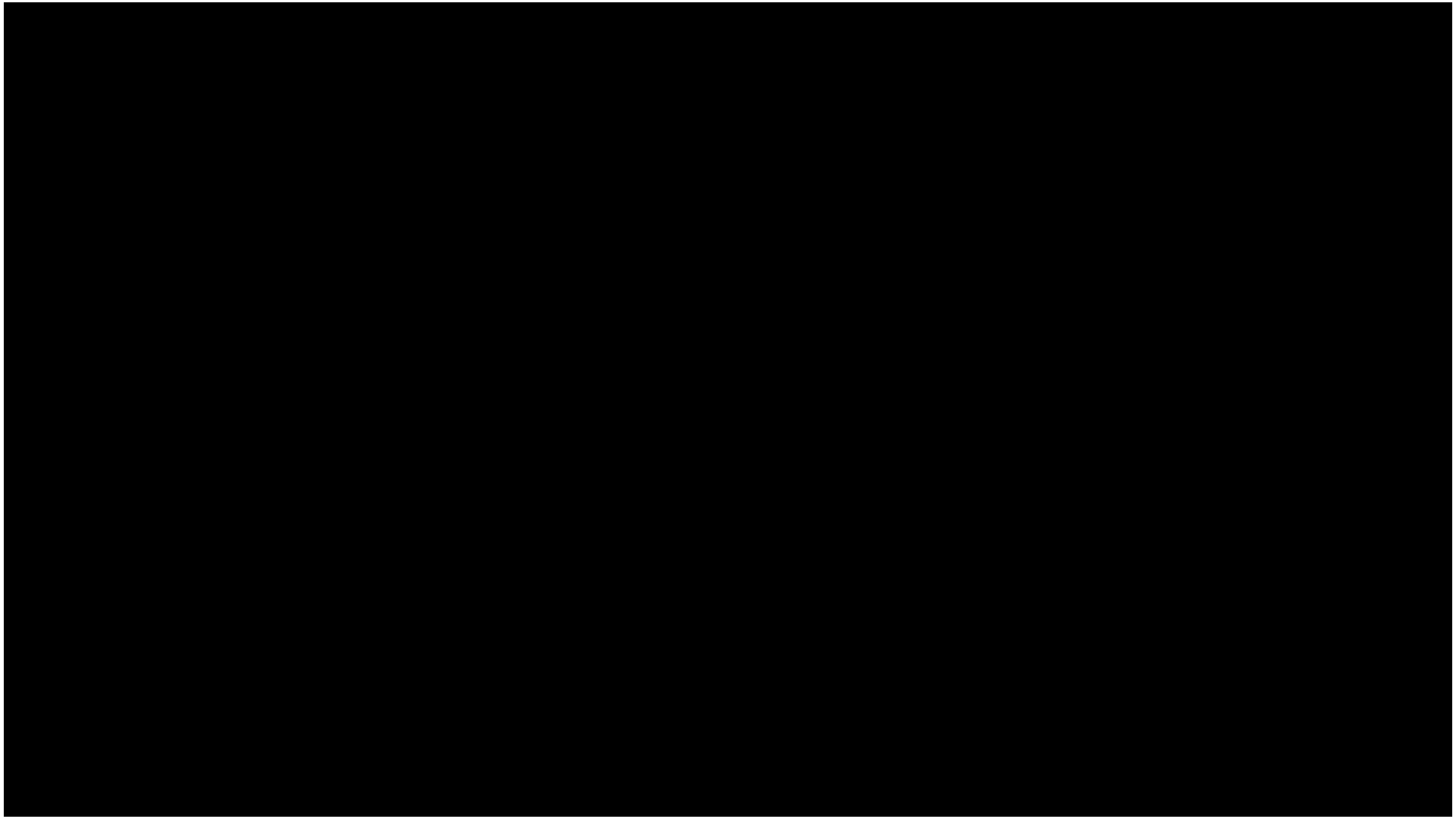
Lecture 24

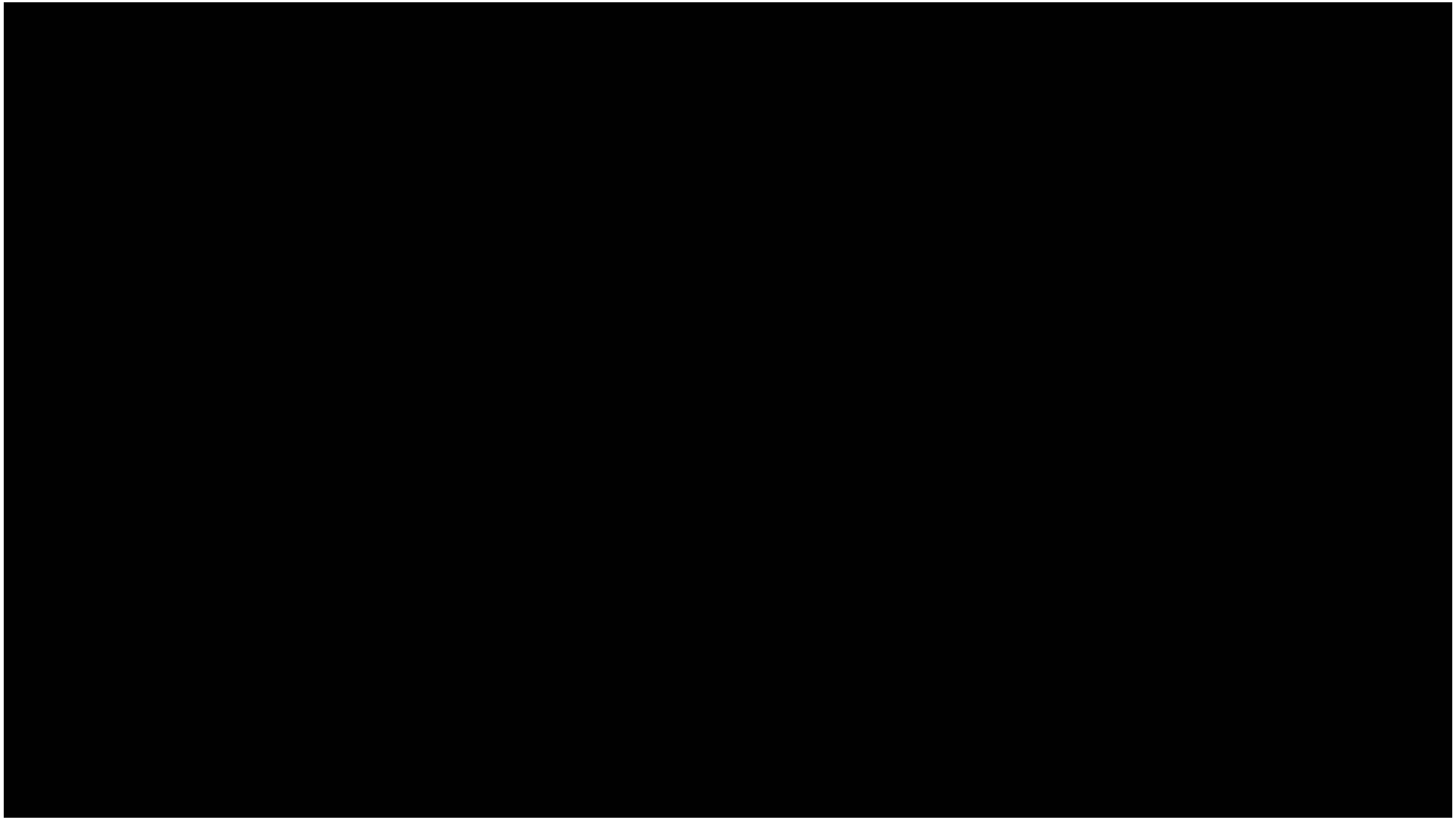
Autonomous Aerial Vehicles

CS 3630



Skydio 2 demo

Most capable drone of its time, no longer available for consumer purchase



Skydio X10

for commercial use

Gyroscopes

From wikipedia

- Used to be mechanical: spinning wheel
- Now MEMS
- Measures angular velocity
- Need to integrate for attitude

$$R_b^n(t) = R_b^n(0) \int_{\tau=0}^{t} \exp \hat{\omega}(\tau) d\tau$$

- Challenge: noise and bias

Accelerometers

From wikipedia

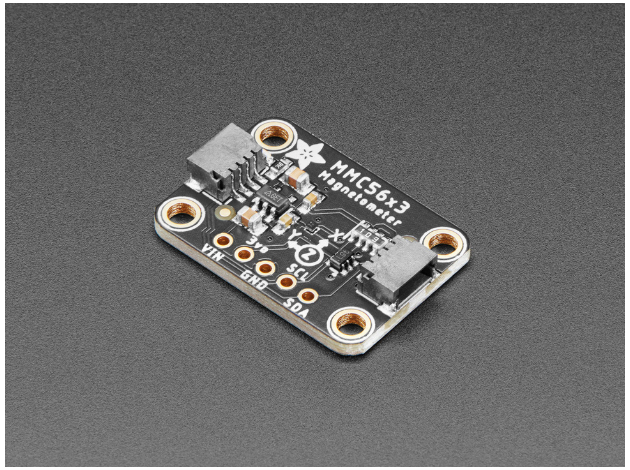
Measures force, translated to acceleration

Double integration: very challenging!

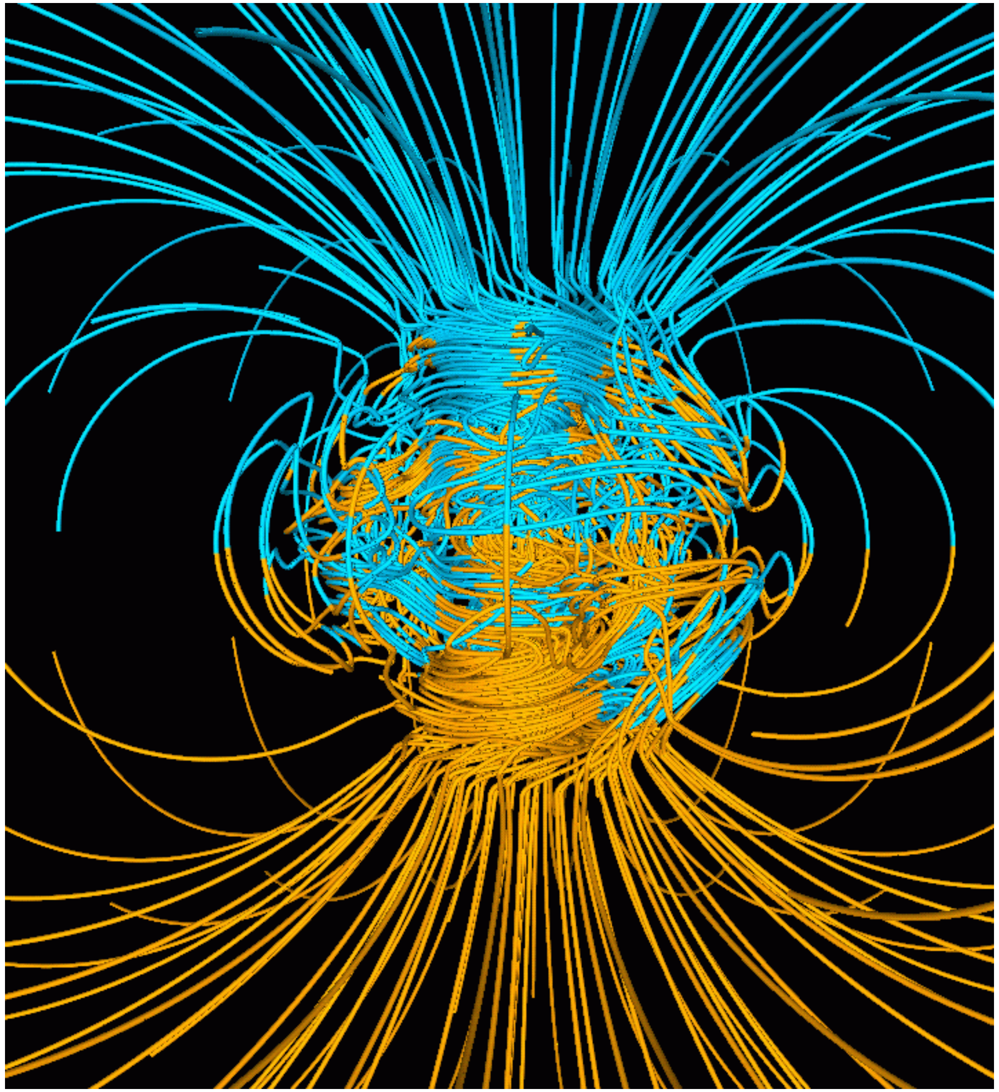
In phones: more useful to "aid" gyroscope.

Magnetometers

- Earth magnetic field is 3D and complex
- Unreliable near metal
- Still helpful



From [Adafruit]() ($5.95)



Public Domain,
https://commons.wikimedia.org/w/index.php?curid=1712490

AHRS

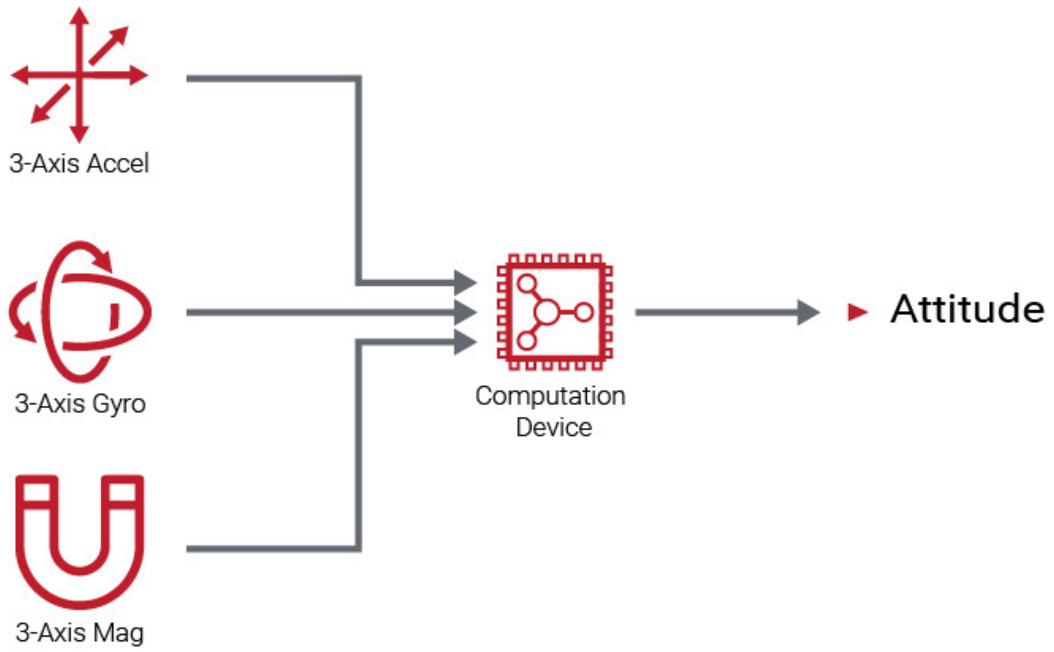


Image from [VectorNav]() course

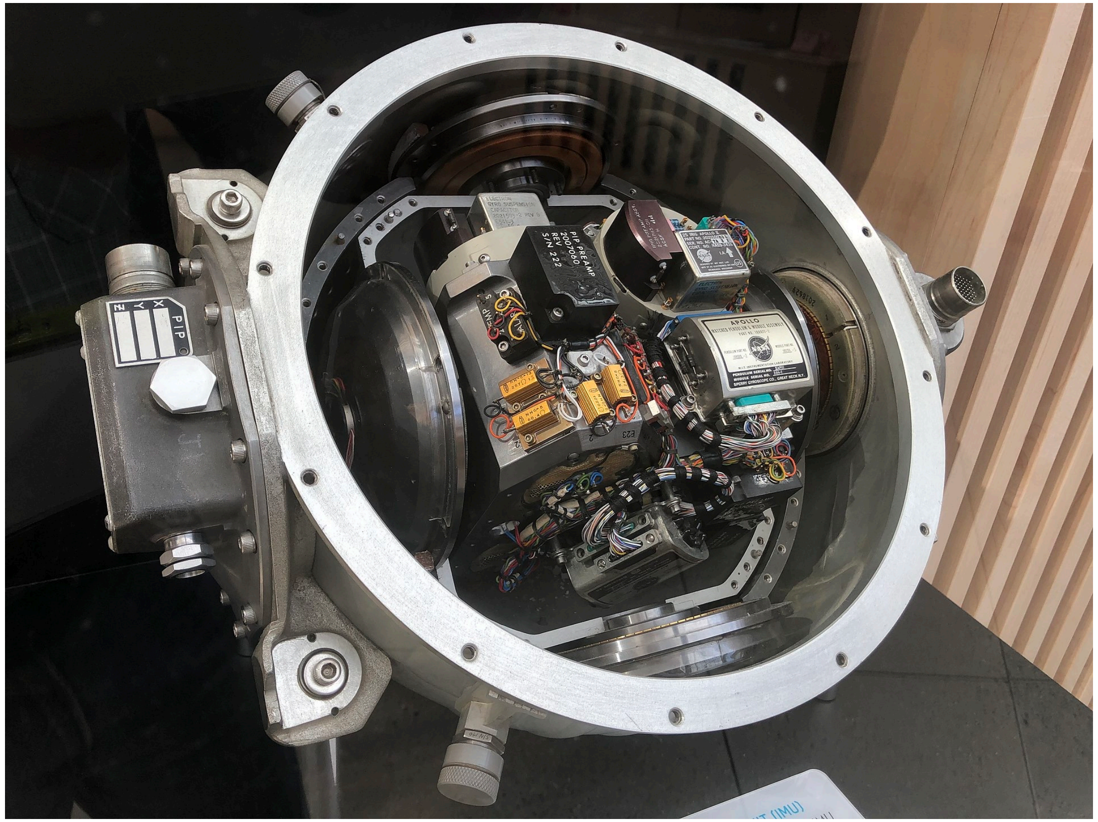
AHRS = Attitude and heading reference system

Gyro = accurate and fast, but attitude drifts!

Accelerometer points to gravity = 2 out of 3DOF

Magnetometer provides (complicated) signal on heading

INS



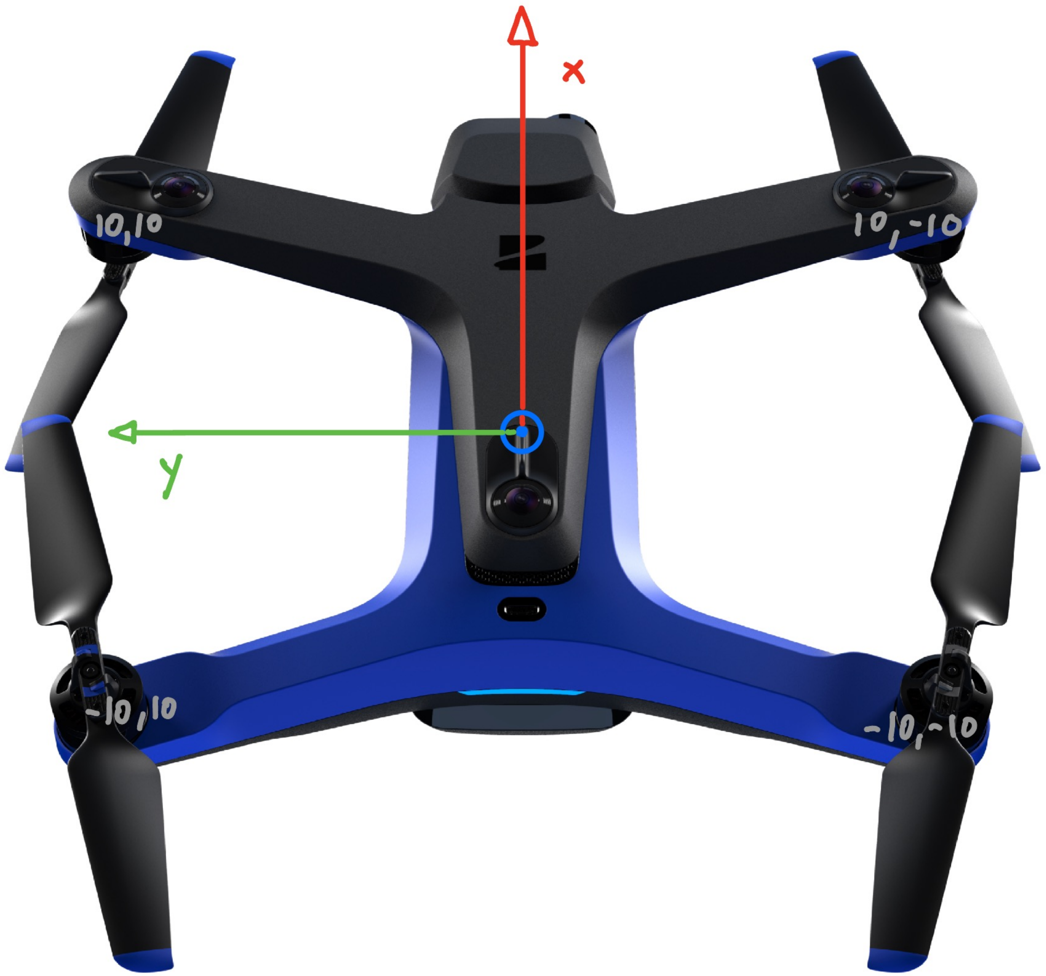
Apollo INS By ArnoldReinhold - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=82248569

- INS = Inertial Navigation System
- Also tries to
  - Integrate accelerometer
  - estimate accelerometer biases
- Needs either:
  - Very very good IMUs (military)
  - Aiding with GPS or other correction signal, e.g., a map!
- Now: strapdown-MEMS:

Cameras

- Light-weight & cheap!
- Passive: low power & stealth
- Supports:
  - Visual odometry
  - Localization
  - Visual SLAM
- For Skydio:
  - Tracking people
  - 3D reconstruction

Defining Position and Orientation

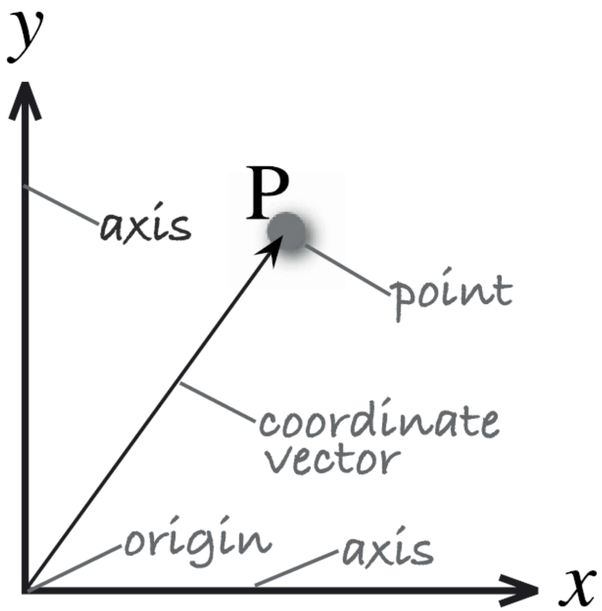


- **Body frame** *B*: FLU = Forward-Left-Up
- **Navigation Frame** *N: ENU = East-North-Up*

- the vehicle's position $r^n \doteq [x, y, z]^T$,
- its linear velocity $v^n = \dot{r}^n \doteq [u, v, w]^T$,
- the attitude $R_b^n \doteq [i^b, j^b, k^b] \in SO(3)$, a 3×3 rotation matrix the navigation frame $\mathcal{N}$,
- the body angular velocity $\omega^b \doteq [p, q, r]^T$.

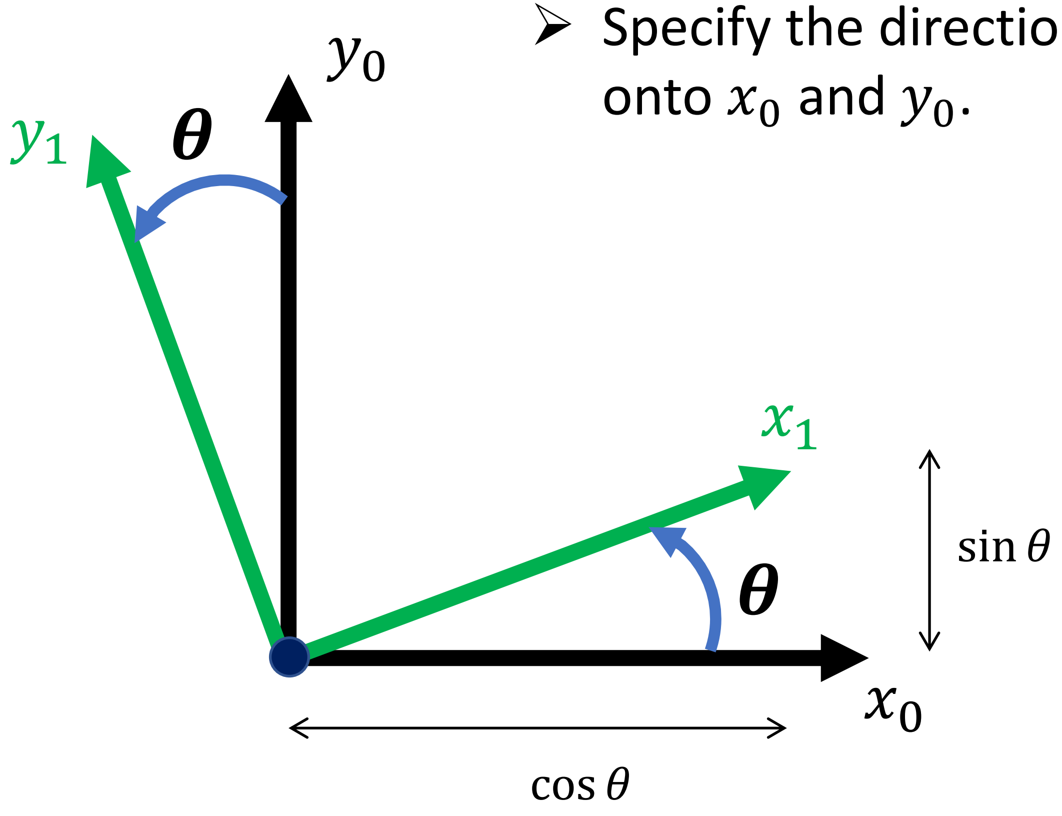
Reference Frames

- Robotics is all about management of reference frames
  - **Perception** is about estimation of reference frames
  - **Planning** is how to move reference frames
  - **Control** is the implementation of trajectories for reference frames
- The relation between references frames is essential to a successful system



Specifying Orientation in the Plane

Given two coordinate frames with a common origin, how should we describe the orientation of Frame 1 w.r.t. Frame 0?



- Specify the directions of x_1 and y_1 with respect to Frame 0 by projecting onto x_0 and y_0.

$$x_1^0 = \begin{bmatrix} x_1 \cdot x_0 \\ x_1 \cdot y_0 \end{bmatrix} = \begin{bmatrix} \cos\theta \\ \sin\theta \end{bmatrix}$$

Notation: x_1^0 denotes the x-axis of Frame 1, specified w.r.t Frame 0.

$$y_1^0 = \begin{bmatrix} y_1 \cdot x_0 \\ y_1 \cdot y_0 \end{bmatrix} = \begin{bmatrix} -\sin\theta \\ \cos\theta \end{bmatrix}$$

We obtain y_1^0 in the same way.

Rotation Matrices (3D)

All of the properties of SO(2) apply as well to SO(3)!

All rotation matrices have certain properties:

1. The two columns are each unit vectors.
2. The two columns are orthogonal, e.g., $c_1 \cdot c_2 = 0$.
3. $\det R = +1$

For such matrices $R^{-1} = R^T$

- The first two properties imply that the matrix R is ***orthogonal***.
- The third property implies that the matrix is ***special***! (After all, there are plenty of orthogonal matrices whose determinant is -1, not at all special.)

The collection of 3×3 rotation matrices is called the ***<u>Special Orthogonal Group of order 3</u>***, or, more commonly $\boldsymbol{SO(3)}$.

Rotation Matrices for 3D rotations

To build a rotation matrix, say R_1^0: project the axes of Frame 1 onto Frame 0. Each column of R_1^0 corresponds to the projection of one axis of Frame 1 onto Frame 0.

$$R_1^0 = [x_1 \cdot F_0 \mid y_1 \cdot F_0 \mid z_1 \cdot F_0] = \begin{bmatrix} x_1 \cdot x_0 & y_1 \cdot x_0 & z_1 \cdot x_0 \\ x_1 \cdot y_0 & y_1 \cdot y_0 & z_1 \cdot y_0 \\ x_1 \cdot z_0 & y_1 \cdot z_0 & z_1 \cdot z_0 \end{bmatrix}$$

Rotation Matrices for 3D rotations

To build a rotation matrix, say R_1^0: project the axes of Frame 1 onto Frame 0. Each column of R_1^0 corresponds to the projection of one axis of Frame 1 onto Frame 0.

$$R_1^0 = \begin{bmatrix} \boxed{x_1 \cdot F_0} & \boxed{y_1 \cdot F_0} & \boxed{z_1 \cdot F_0} \end{bmatrix} = \begin{bmatrix} x_1 \cdot x_0 & y_1 \cdot x_0 & z_1 \cdot x_0 \\ x_1 \cdot y_0 & y_1 \cdot y_0 & z_1 \cdot y_0 \\ x_1 \cdot z_0 & y_1 \cdot z_0 & z_1 \cdot z_0 \end{bmatrix}$$

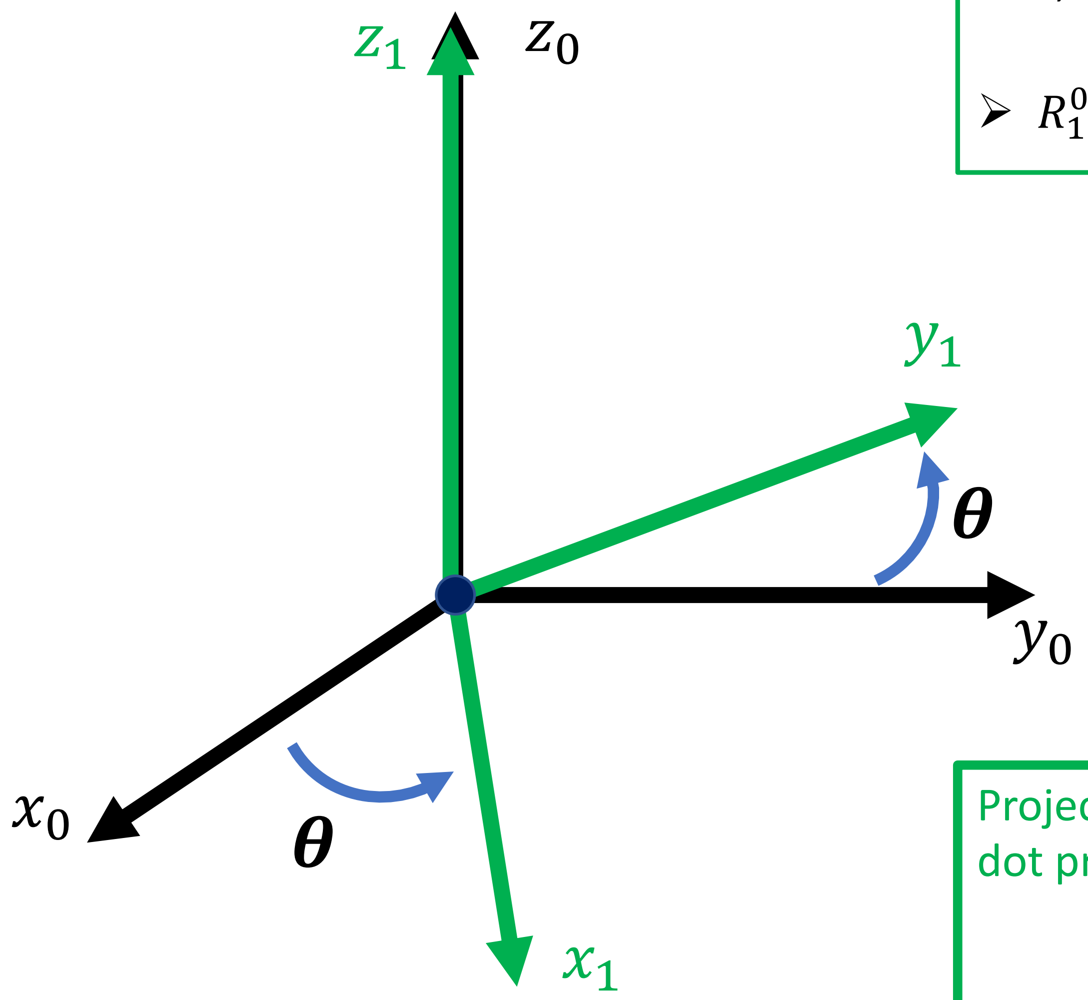
Project the x-axis of Frame 1 onto the axes of Frame 0

Project the y-axis of Frame 1 onto the axes of Frame 0

Project the z-axis of Frame 1 onto the axes of Frame 0

This process is exactly the same as the process for building rotation matrices in SO(2), even though it can be more difficult to visualize in 3D for rotation matrices in SO(3).

The simplest example: rotation about the z axis



Recall: for rotation in the plane, we built a rotation matrix as a function of θ, the angle between x_1 and x_0 (and also between y_1 and y_0):

- $R_1^0 = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$ FOR ROTATION IN THE PLANE

This is *easily* extended to the case of rotation in 3D about the z-axis, since all of the interesting action is in the x-y plane (the two z-axes are the same)!

In fact, you'll see that the 2D rotation matrix shows up in the 3D rotation matrix:

- $R_1^0 = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$ FOR ROTATION IN 3D

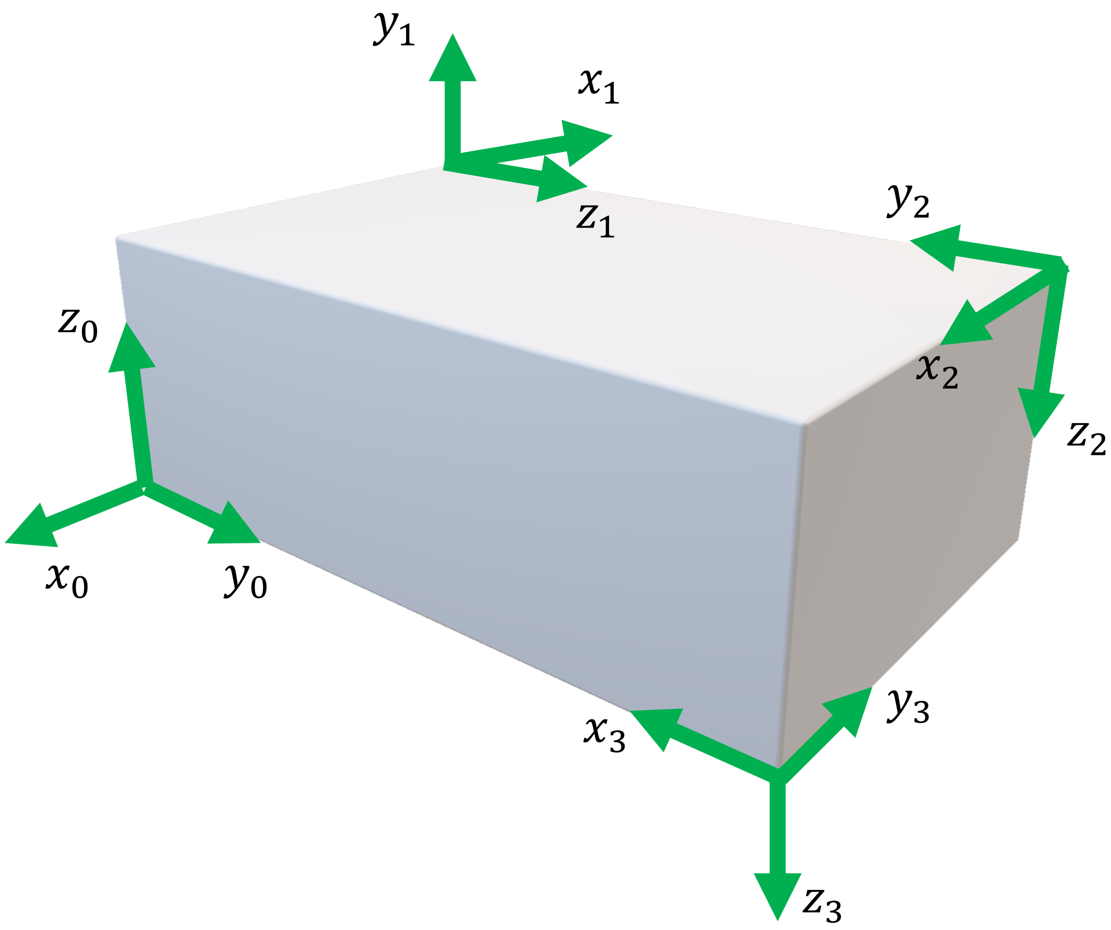
Projecting z_1 onto Frame 0 involves three dot products:

$$z_1 \cdot x_0 = 0$$
$$z_1 \cdot y_0 = 0$$
$$z_1 \cdot z_0 = 1$$

A bunch of examples:

A rectangular solid: all angles are multiples of $\pi/2$.

$$R_j^i = \begin{bmatrix} x_j \cdot x_i & y_j \cdot x_i & z_j \cdot x_i \\ x_j \cdot y_i & y_j \cdot y_i & z_j \cdot y_i \\ x_j \cdot z_i & y_j \cdot z_i & z_j \cdot z_i \end{bmatrix}$$



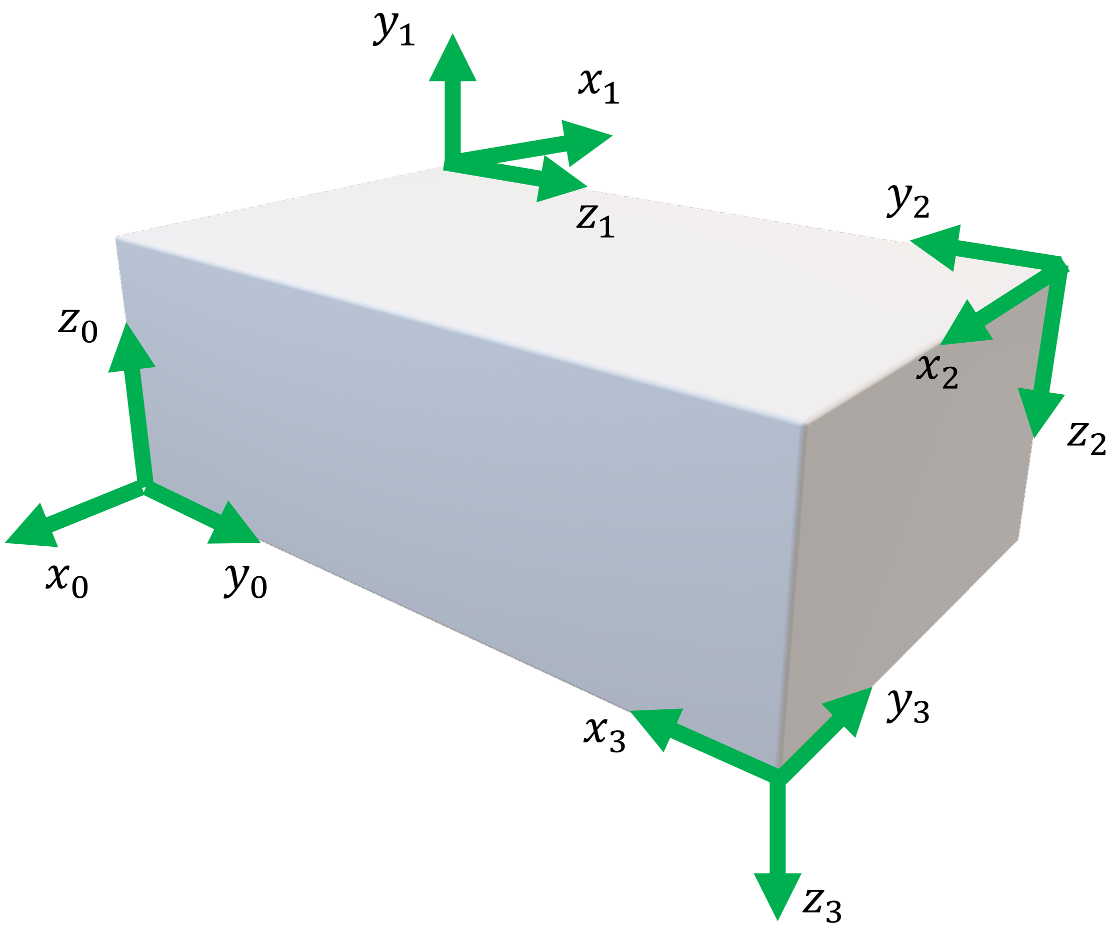
$$R_1^0 = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

$$R_0^1 = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

A bunch of examples:

A rectangular solid: all angles are multiples of $\pi/2$.

$$R_j^i = \begin{bmatrix} x_j \cdot x_i & y_j \cdot x_i & z_j \cdot x_i \\ x_j \cdot y_i & y_j \cdot y_i & z_j \cdot y_i \\ x_j \cdot z_i & y_j \cdot z_i & z_j \cdot z_i \end{bmatrix}$$



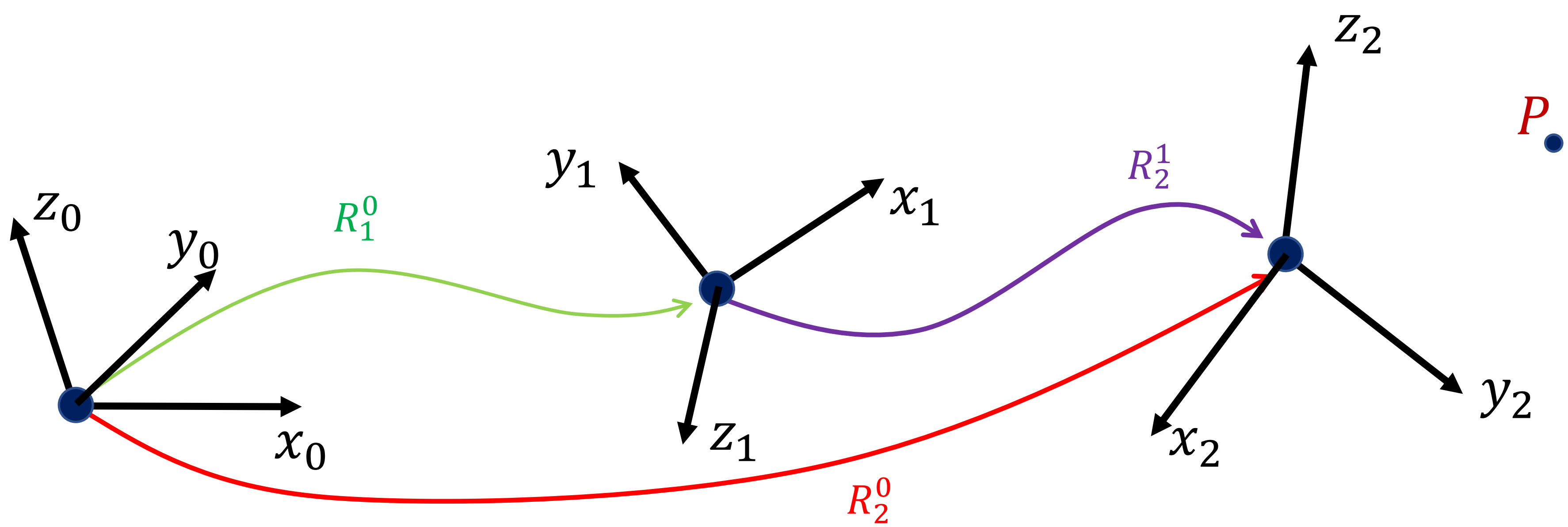
$$R_1^0 = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \qquad R_0^1 = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

$$R_1^0 R_0^1 = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} -1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\boxed{(R_1^0)^{-1} = R_0^1 = (R_1^0)^T}$$

Composition of Rotations

For now, only consider the rotation, *not the translation*!
This is an "exploded" view of three coordinate frames that share the same origin.



From our previous results, we know:

$$P^0 = R_1^0 P^1$$
$$P^1 = R_2^1 P^2$$

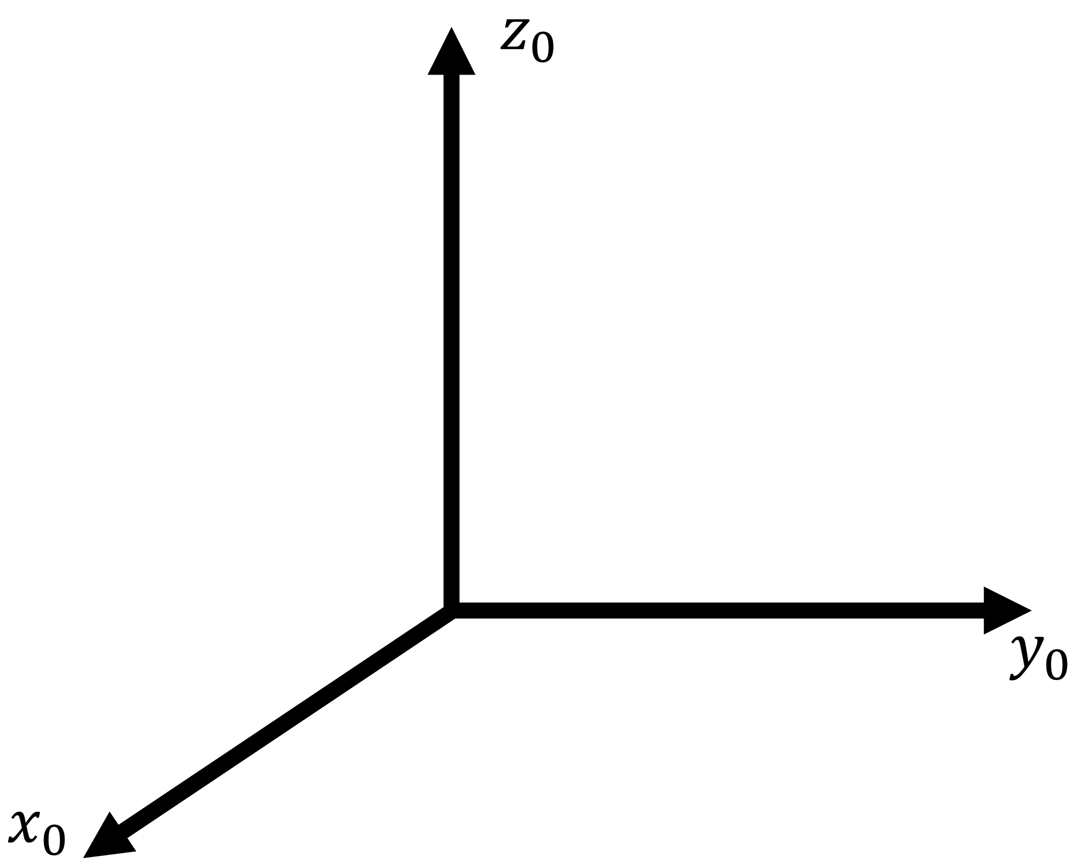
$$P^0 = R_1^0 R_2^1 P^2$$

But we also know: $P^0 = R_2^0 P^2$

$$R_2^0 = R_1^0 R_2^1$$

This is the composition law for rotation transformations.

Rotations about Coordinate Axes



$$R_{x,\psi} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\psi & -\sin\psi \\ 0 & \sin\psi & \cos\psi \end{bmatrix}$$

$$R_{y,\theta} = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}$$

$$R_{z,\phi} = \begin{bmatrix} \cos\phi & -\sin\phi & 0 \\ \sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Parameterization of 3D Rotations

- Consider the three successive rotations: $R = R_{z,\phi} R_{y,\theta} R_{x,\psi}$

$$R_{z,\phi} R_{y,\theta} R_{x,\psi} = \begin{bmatrix} \cos\phi & -\sin\phi & 0 \\ \sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\psi & -\sin\psi \\ 0 & \sin\psi & \cos\psi \end{bmatrix}$$

$$= \begin{bmatrix} C_\phi C_\theta & -S_\phi C_\psi + C_\phi S_\theta S_\psi & -S_\phi S_\psi + C_\phi S_\theta C_\psi \\ S_\phi C_\theta & C_\phi C_\psi + S_\phi S_\theta S_\psi & -C_\phi S_\psi + S_\phi S_\theta C_\psi \\ S_\theta & C_\theta S_\psi & C_\theta C_\psi \end{bmatrix}$$

Parameterization of 3D Rotations

Any rotation matrix can be expressed in this form!

$$\begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} = \begin{bmatrix} C_\phi C_\theta & -S_\phi C_\psi + C_\phi S_\theta S_\psi & S_\phi S_\psi + C_\phi S_\theta C_\psi \\ S_\phi C_\theta & C_\phi C_\psi + S_\phi S_\theta S_\psi & -C_\phi S_\psi + S_\phi S_\theta C_\psi \\ -S_\theta & C_\theta S_\psi & C_\theta C_\psi \end{bmatrix}$$

1. Solve for θ using $\mathrm{r}_{31} = S_\theta$
2. Solve for ψ using $\mathrm{r}_{32} = C_\theta S_\psi, r_{33} = C_\theta C_\psi$
3. Solve for ϕ using $\mathrm{r}_{11} = C_\phi C_\theta, \mathrm{r}_{21} = S_\phi C_\theta$

The function $ATAN2(y, x)$ returns the angle whose tangent is y/x, in the appropriate quadrant. Thus:

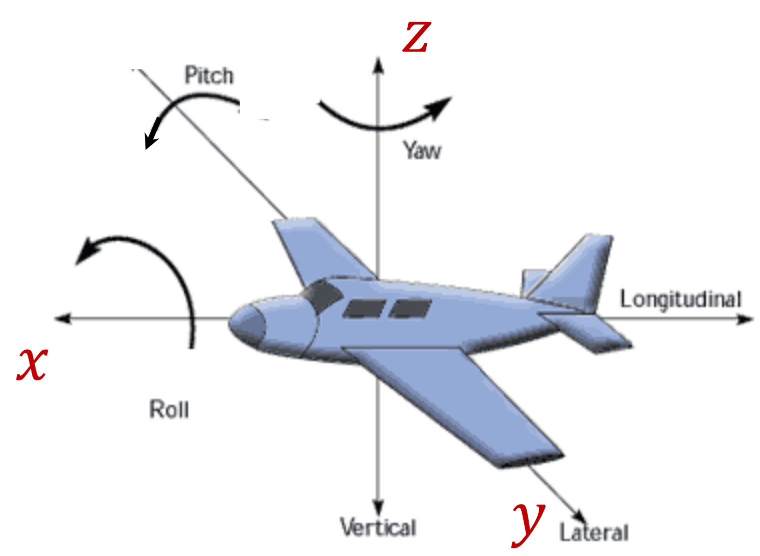
$$\psi = ATAN2(r_{32}, r_{33})$$
$$\phi = ATAN2(r_{21}, r_{11})$$

We can parameterize SO(3) using these three angles, ϕ, θ, ψ.

Roll, Pitch, and Yaw

When we parameterize the rotation matrices using $R = R_{z,\phi}R_{y,\theta}R_{x,\psi}$, the angles are called roll, pitch, and yaw:

- Yaw is a rotation about the world's z-axis.
- Pitch is a rotation about the plane's y-axis (note, this axis moves as a function of the yaw angle).
- Roll is a rotation about the plane's x-axis (note, this axis moves as a function of the yaw angle and the pitch angle).



➢ Remember, things break down when $\theta = \pm\frac{\pi}{2}$ --- but this makes sense!

If $\theta = \pm\frac{\pi}{2}$ then the planes x-axis is aligned with the world's z-axis.

In this case, roll and yaw are rotations about the same axis!

- Roll, Pitch, and Yaw are useful when the plane is roughly horizontal.
- If the plane tips completely up or completely down (i.e., $\theta = \pm\frac{\pi}{2}$), things have already gone very wrong, so it's not such a big deal that the parameterization breaks down for this case.

This coordinate frame assignment is known as *Forward-Left-Up (FLU).*

- The x-axis is the Forward direction.
- The y-axis points to the Left.
- The z-axis points Up.

Singularities for Roll,Pitch,Yaw

- For Roll, Pitch, and Yaw, when $S_\theta = 1$, we have:

$$R_{z,\phi}R_{y,\theta}R_{x,\psi} = \begin{bmatrix} C_\phi C_\theta & -S_\phi C_\psi + C_\phi S_\theta S_\psi & S_\phi S_\psi + C_\phi S_\theta C_\psi \\ S_\phi C_\theta & C_\phi C_\psi + S_\phi S_\theta S_\psi & -C_\phi S_\psi + S_\phi S_\theta C_\psi \\ -S_\theta & C_\theta S_\psi & C_\theta C_\psi \end{bmatrix}$$

$$= \begin{bmatrix} 0 & -S_\phi C_\psi + C_\phi S_\psi & -S_\phi S_\psi + C_\phi C_\psi \\ 0 & C_\phi C_\psi + S_\phi S_\psi & -C_\phi S_\psi + S_\phi C_\psi \\ 1 & 0 & 0 \end{bmatrix}$$

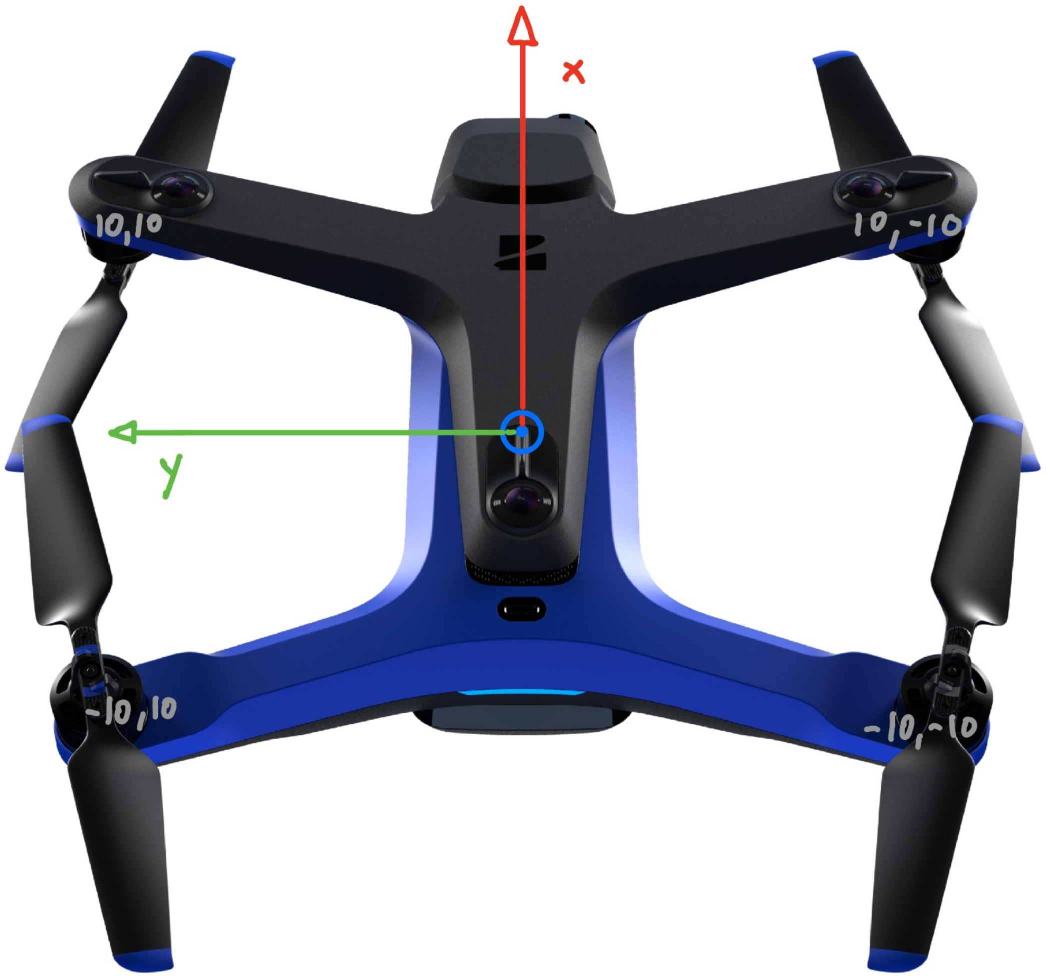
When $S_\theta = 1$, there are infinitely many solutions for ψ and ϕ.
In this case, we have, e.g.,

$$r_{22} = C_\phi C_\psi + S_\phi S_\psi = C_{\phi-\psi}$$

So, only the difference $\phi - \psi$ is uniquely determined.

So, this is only a local parameterization.
It works when $r_{31} \neq \pm 1$.

Defining Position and Orientation



- **Body frame** *B*: FLU = Forward-Left-Up
- **Navigation Frame** *N: ENU = East-North-Up*

- the vehicle's position $r^n \doteq [x, y, z]^T$,
- its linear velocity $v^n = \dot{r}^n \doteq [u, v, w]^T$,
- the attitude $R_b^n \doteq [i^b, j^b, k^b] \in SO(3)$, a 3×3 rotation matrix the navigation frame $\mathcal{N}$,
- the body angular velocity $\omega^b \doteq [p, q, r]^T$.

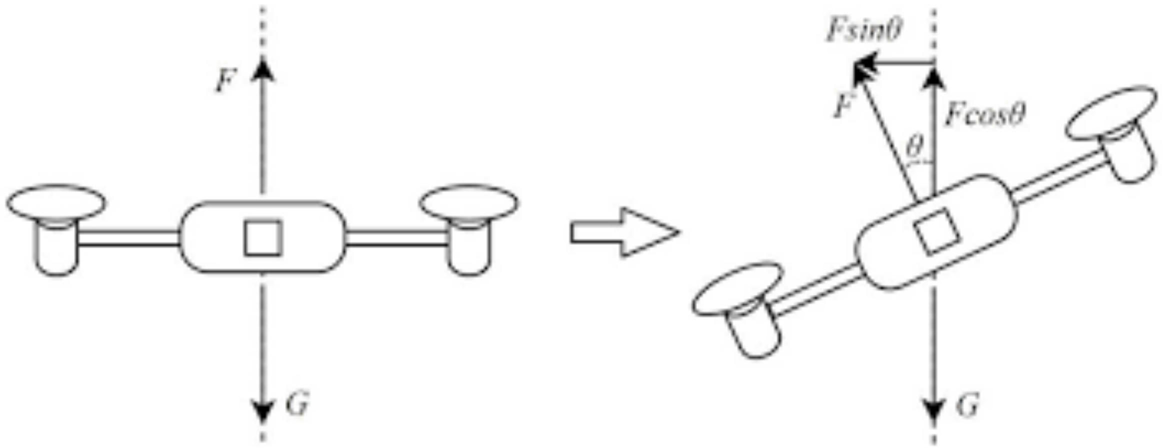
Hover

$$F_z^b = \sum_{i=1}^{4} f_i.$$

- Assume weight = 1kg
- g = 10 m/s^2
- Need to provide 10N of thrust!

- $f_i = 0N$ for $i \in 1..4$: downwards acceleration at $-10\frac{m}{s^2}$.
- $f_i = 2.5N$ for $i \in 1..4$: stable hover $0\frac{m}{s^2}$.
- $f_i = 5N$ for $i \in 1..4$: upwards acceleration at $10\frac{m}{s^2}$.

Forward Flight



$$F^n = R_b^n \begin{bmatrix} 0 \\ 0 \\ F_z^b \end{bmatrix} = \hat{z}_b^n F_z^b$$

$$F^n = \begin{bmatrix} 0 \\ \sin\theta \\ \cos\theta \end{bmatrix} F_z^b.$$

- Force is always up *in body frame*
- Need to rotate to navigation frame
- Thrust is always aligned with body z-axis expressed in navigation frame
- Maintain altitude:

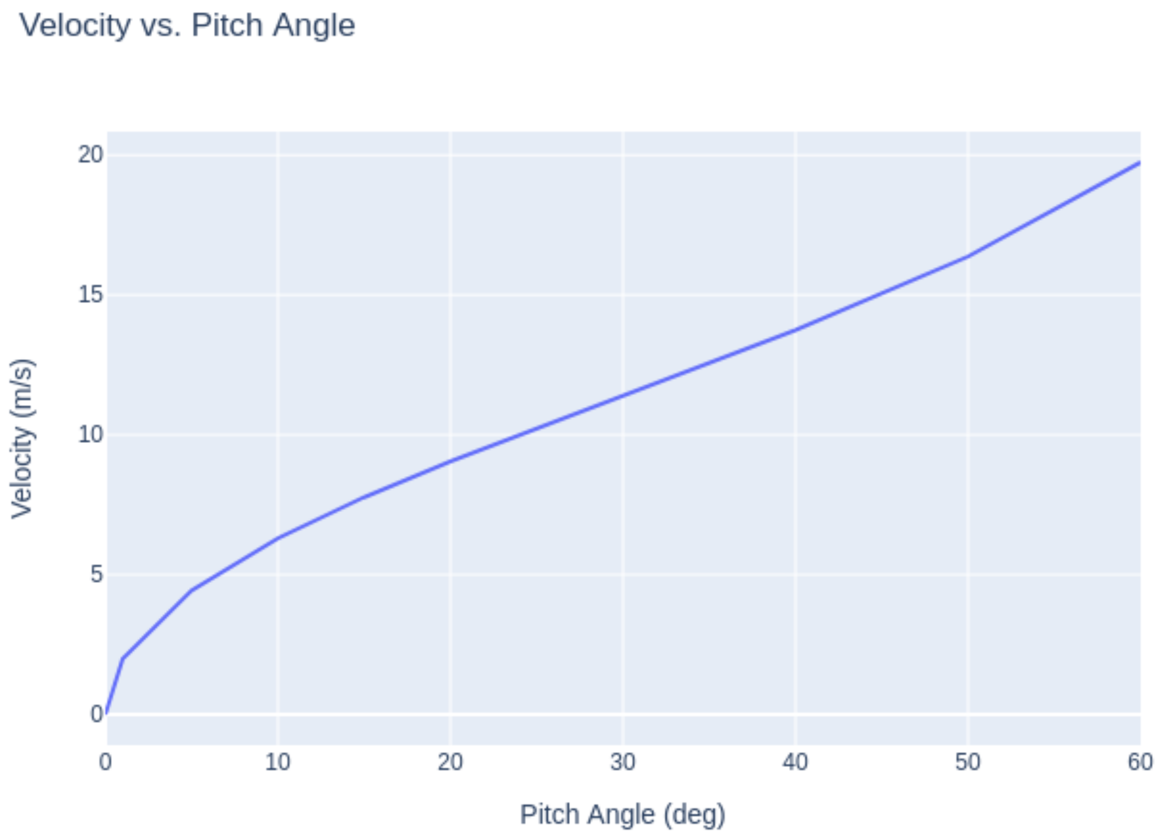
$$\cos\theta F_z^b == 10N.$$

- That means:

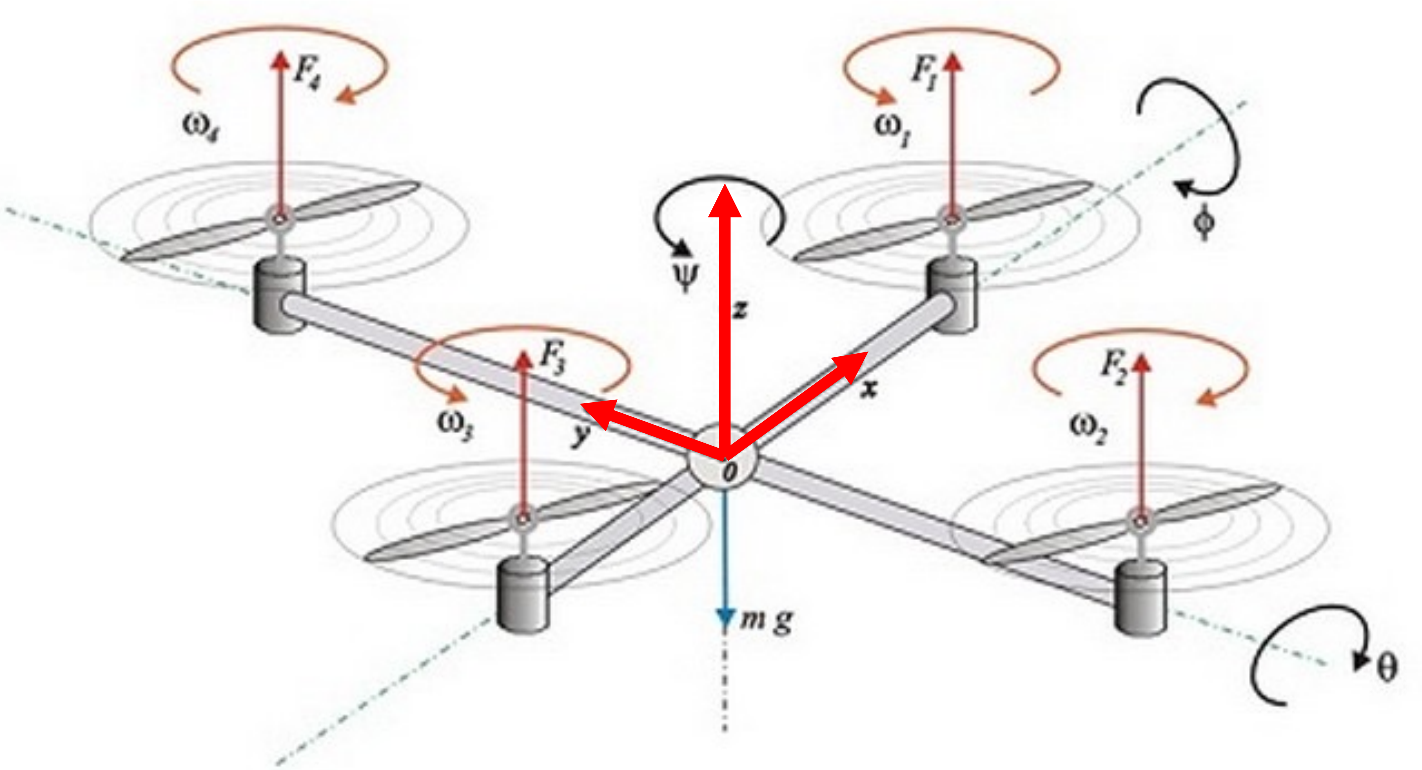
$$F_y^n = \sin\theta F_z^b = \sin\theta \frac{10N}{\cos\theta} = \tan\theta \cdot 10N$$

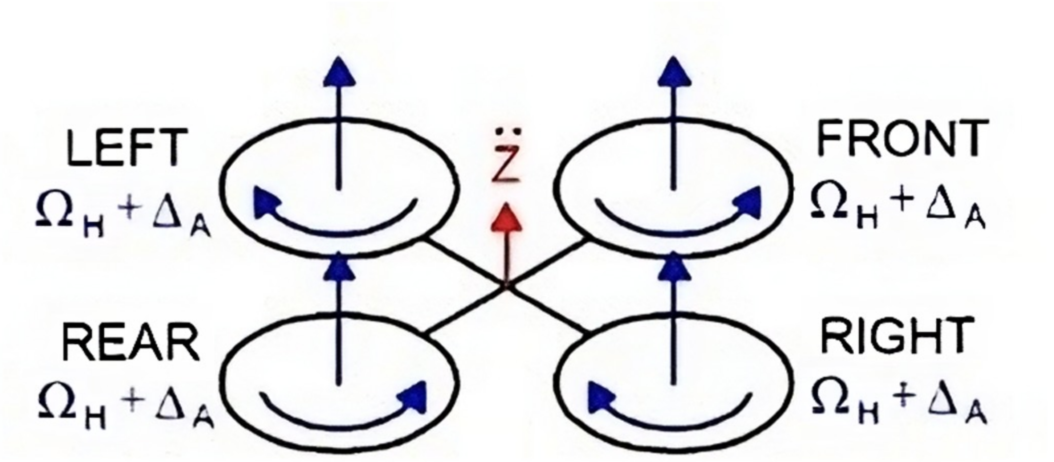
Drag

- Air resistance increases quadratically with velocity
- Max velocity 20 m/s

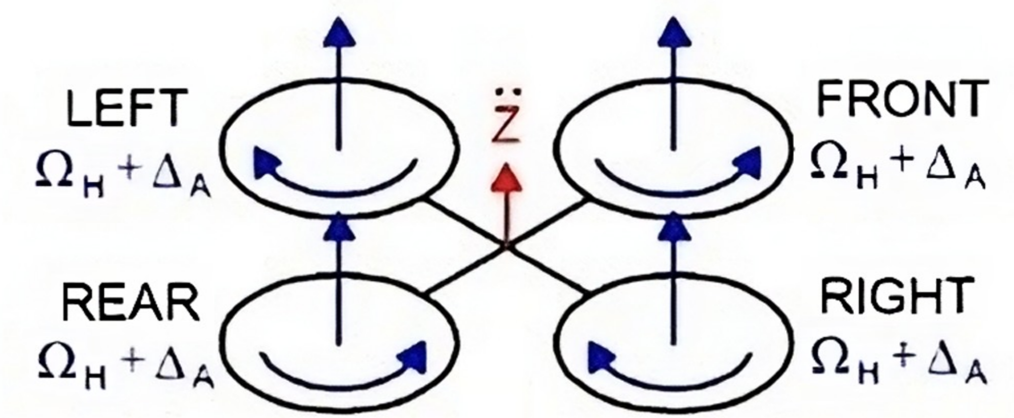


Autonomous Aerial Vehicles: Quadcopters

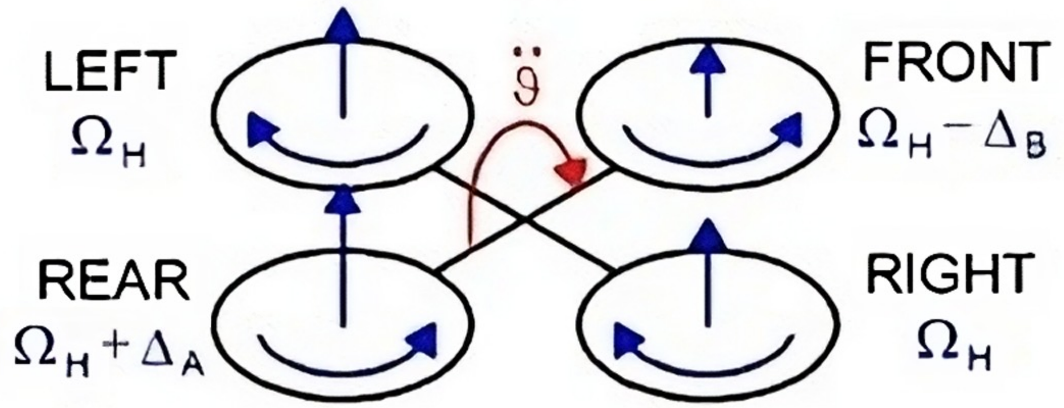



LEFT
$\Omega_H + \Delta_A$
$\ddot{Z}$
FRONT
$\Omega_H + \Delta_A$
REAR
$\Omega_H + \Delta_A$
RIGHT
$\Omega_H + \Delta_A$

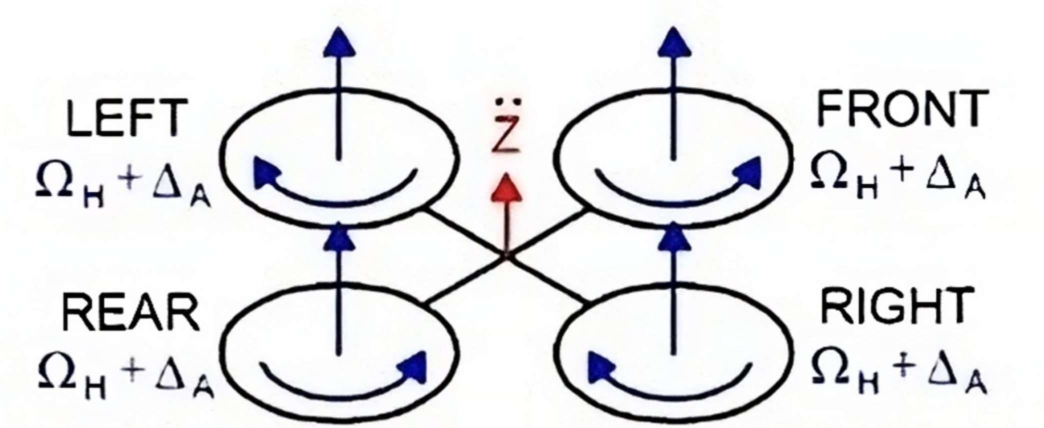
Throttle



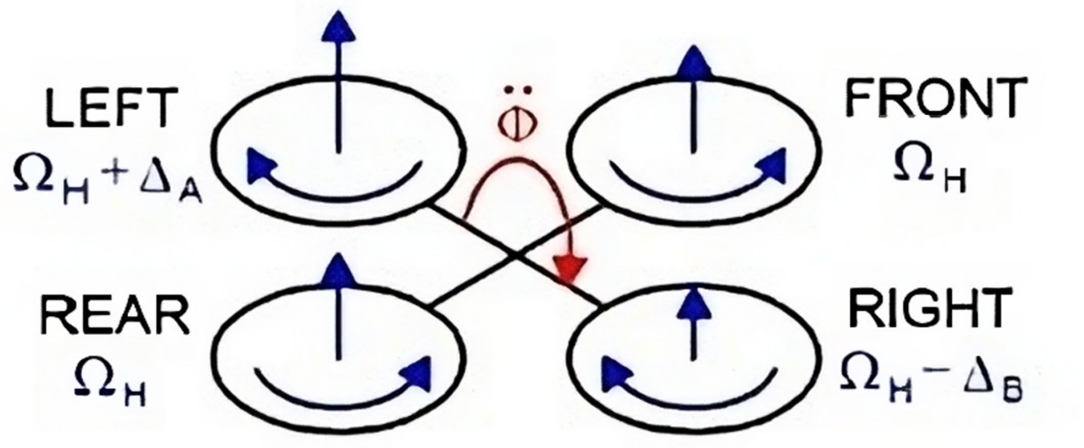
Throttle



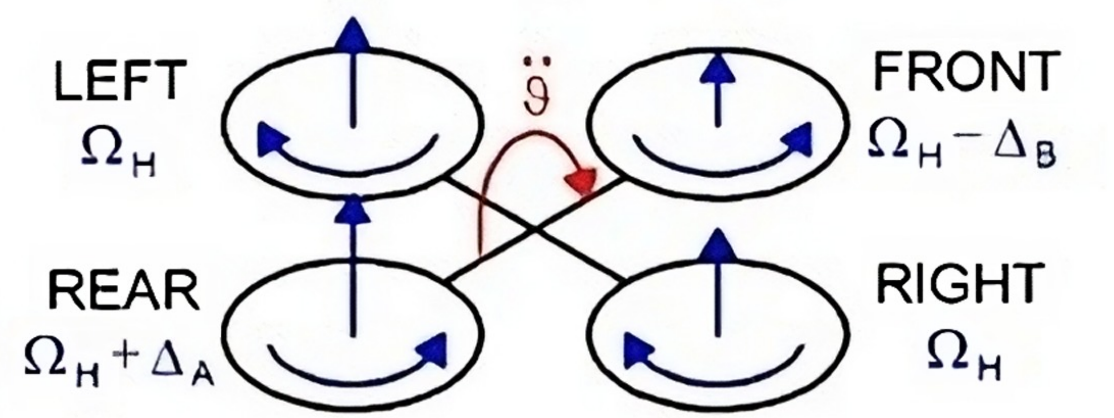
Pitch



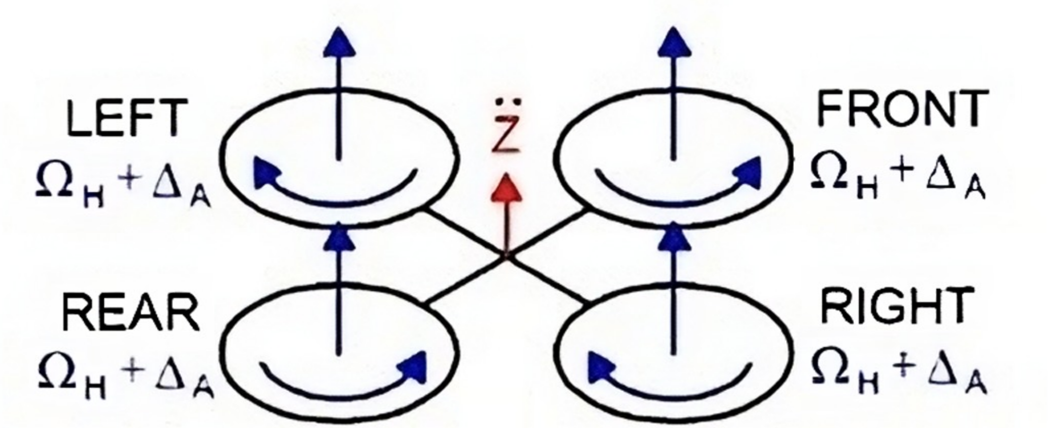
Throttle



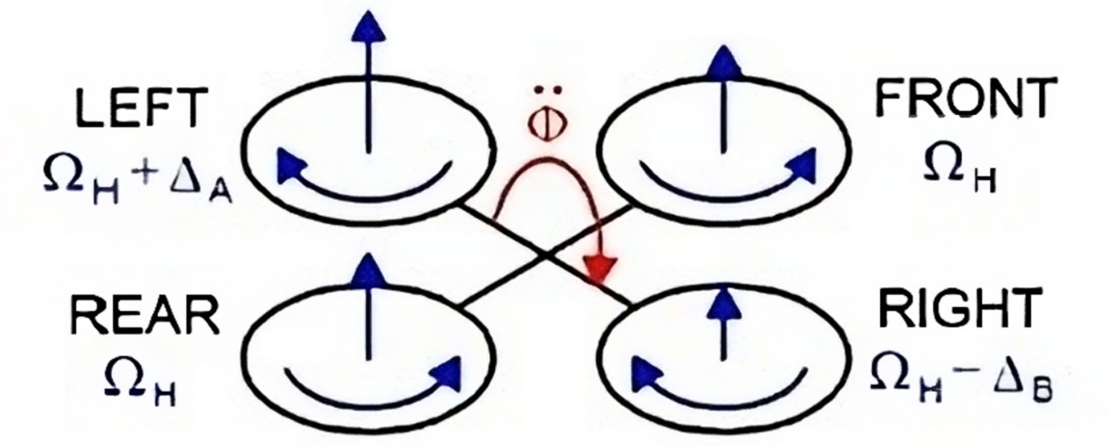
Roll



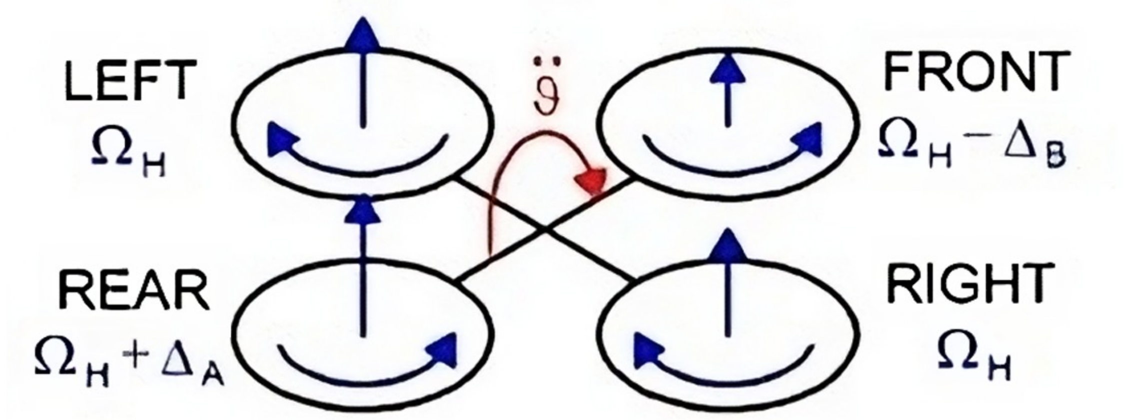
Pitch



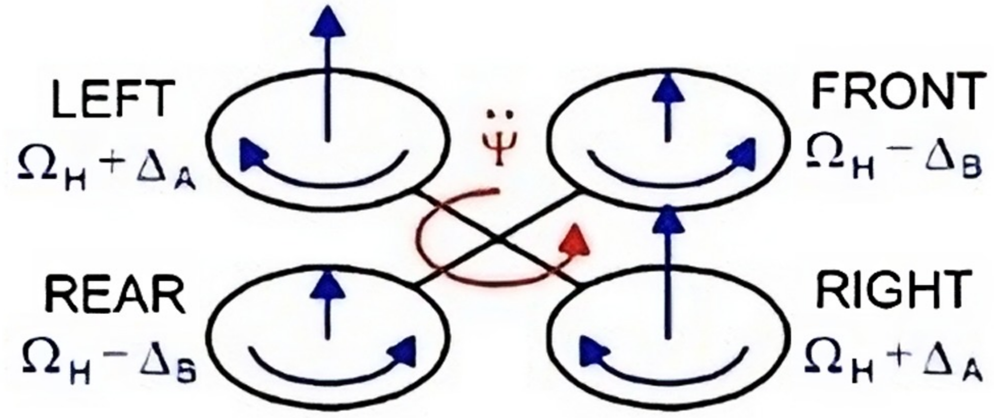
Throttle



Roll



Pitch



Yaw

All this is also applicable to underwater vehicles!

Hydrus, by Advanced Navigation, is an underwater quadrotor.