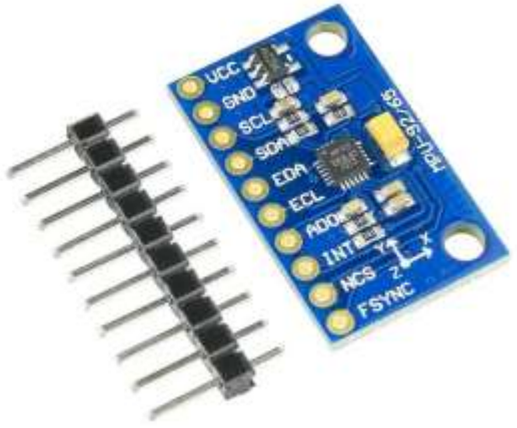
Inertial Motion Tracking using IMUs

ILLINOIS
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

Inertial Measurement Unit (IMU)

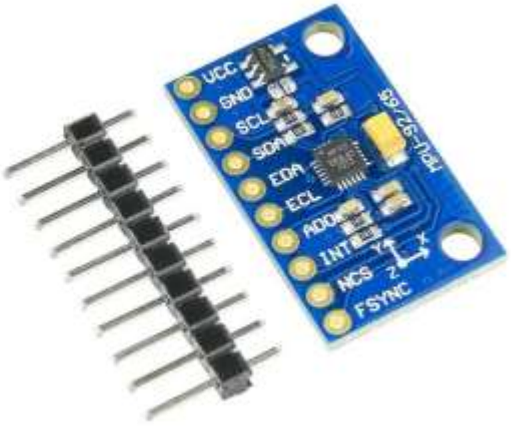


Accelerometer

Gyroscope

Magnetometer

Inertial Measurement Unit (IMU)

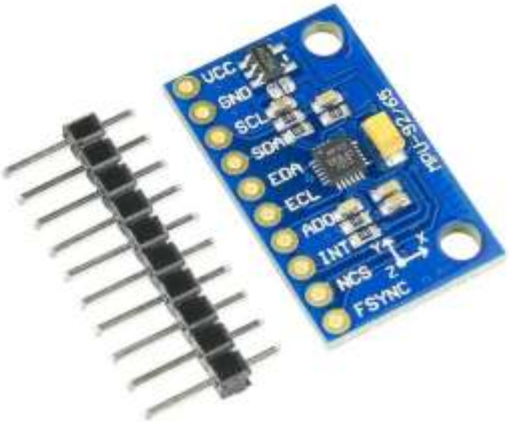


Accelerometer

Gyroscope

Magnetometer

Inertial Measurement Unit (IMU)



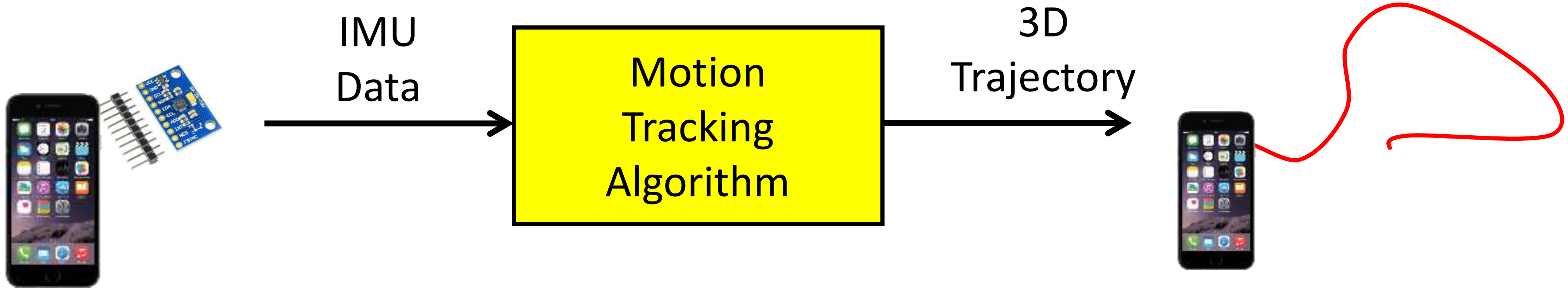
Accelerometer

Gyroscope

Magnetometer

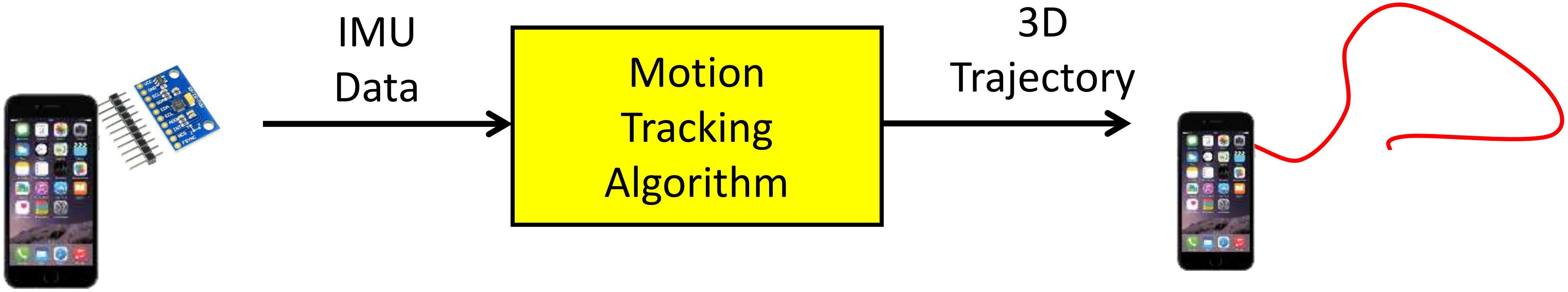
Wide applications in motion tracking

Lot of work in inertial motion tracking



Open problem in mobile computing

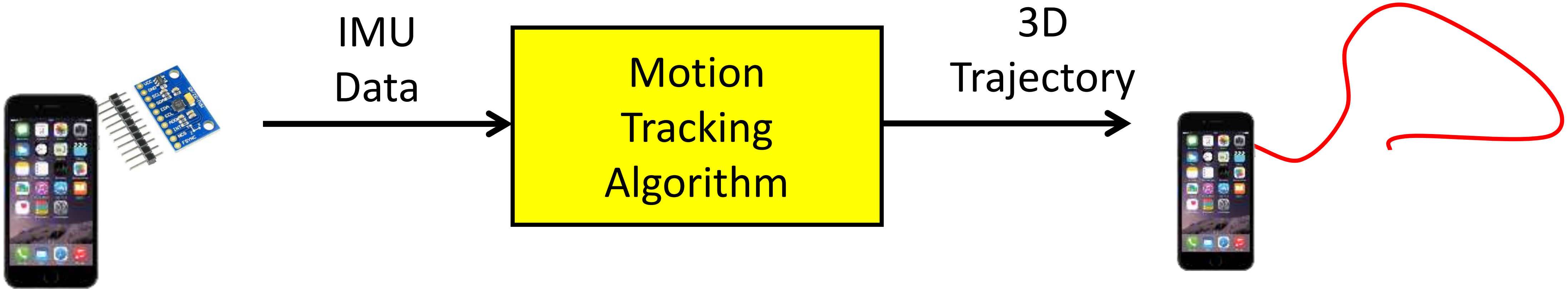
Lot of work in inertial motion tracking



Open problem in mobile computing

No one has the solution … but people making progress

Lot of work in inertial motion tracking



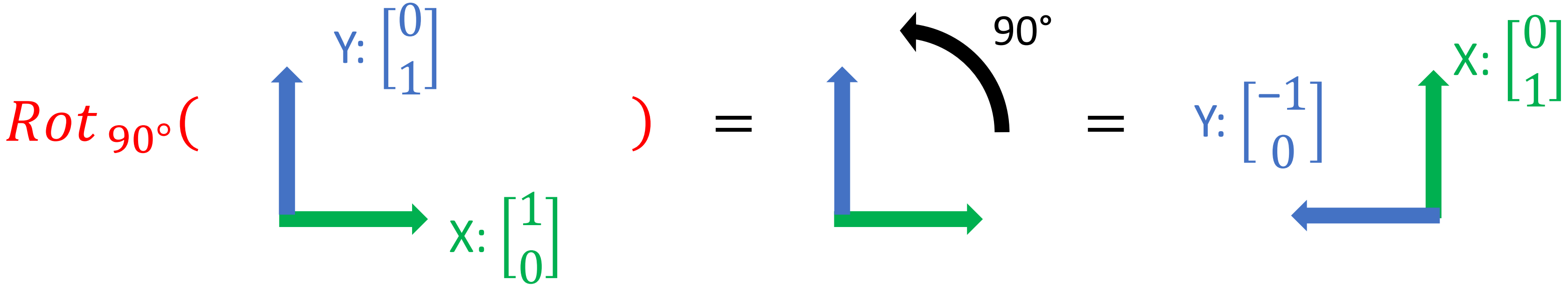
Open problem in mobile computing

No one has the solution … but people making progress

Let’s understand what’s the real difficulty here …

One Prerequisite Slide: Rotation Matrices

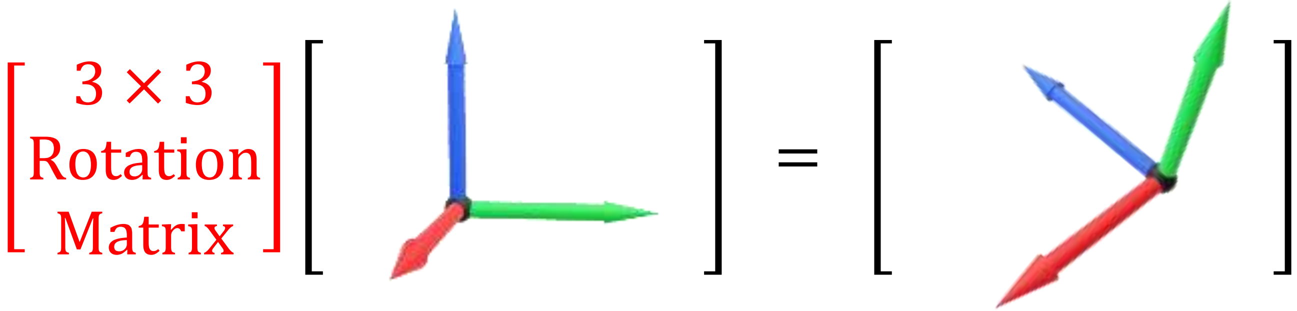
- Rotation is a function



- Mathematically, rotation is a matrix

$$\begin{bmatrix}\cos 90° & -\sin 90° \\ \sin 90° & \cos 90°\end{bmatrix}\begin{bmatrix}1 & 0 \\ 0 & 1\end{bmatrix} = \begin{bmatrix}0 & -1 \\ 1 & 0\end{bmatrix}$$

- Same for 3D Rotation



MUSE: Our Goal is 3D Localization

Initial Location

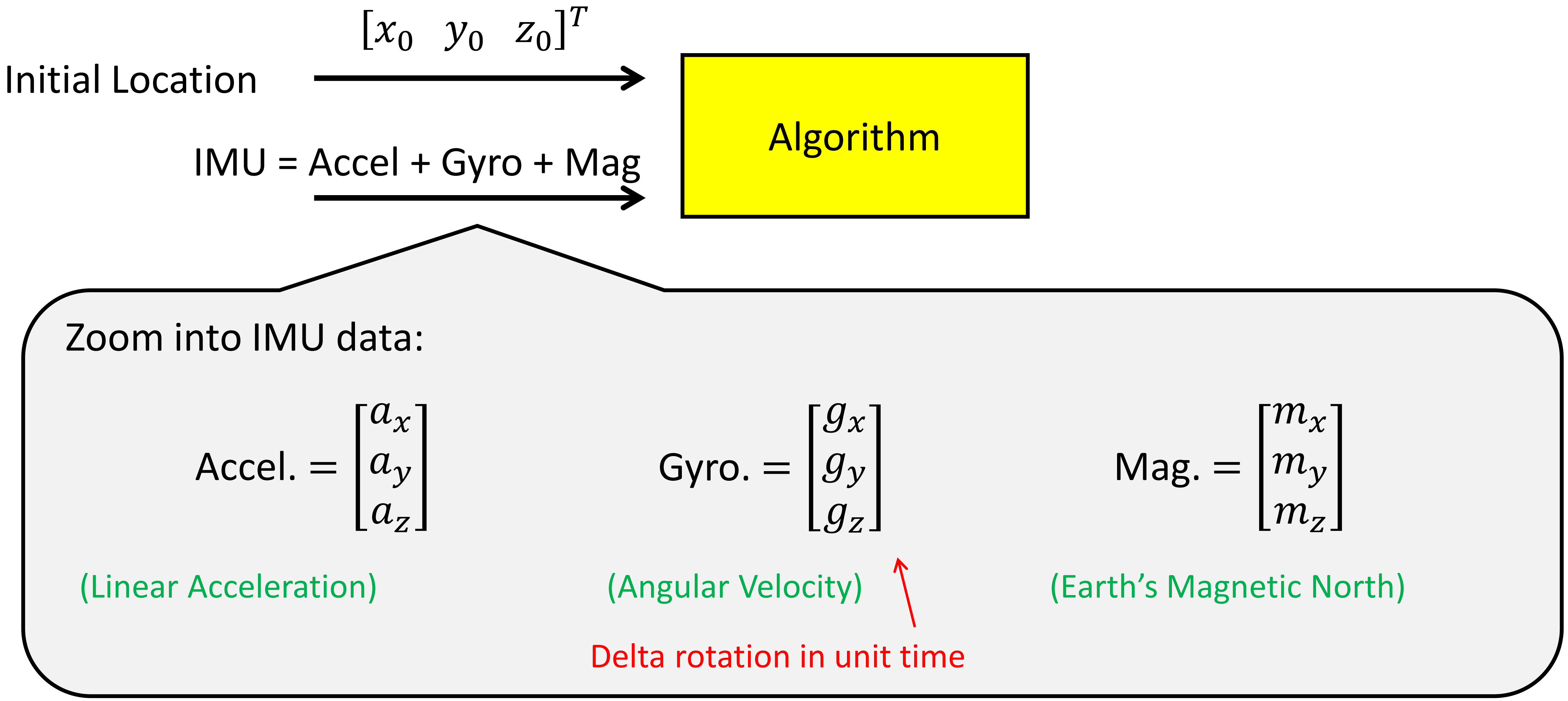
IMU = Accel + Gyro + Mag

Algorithm

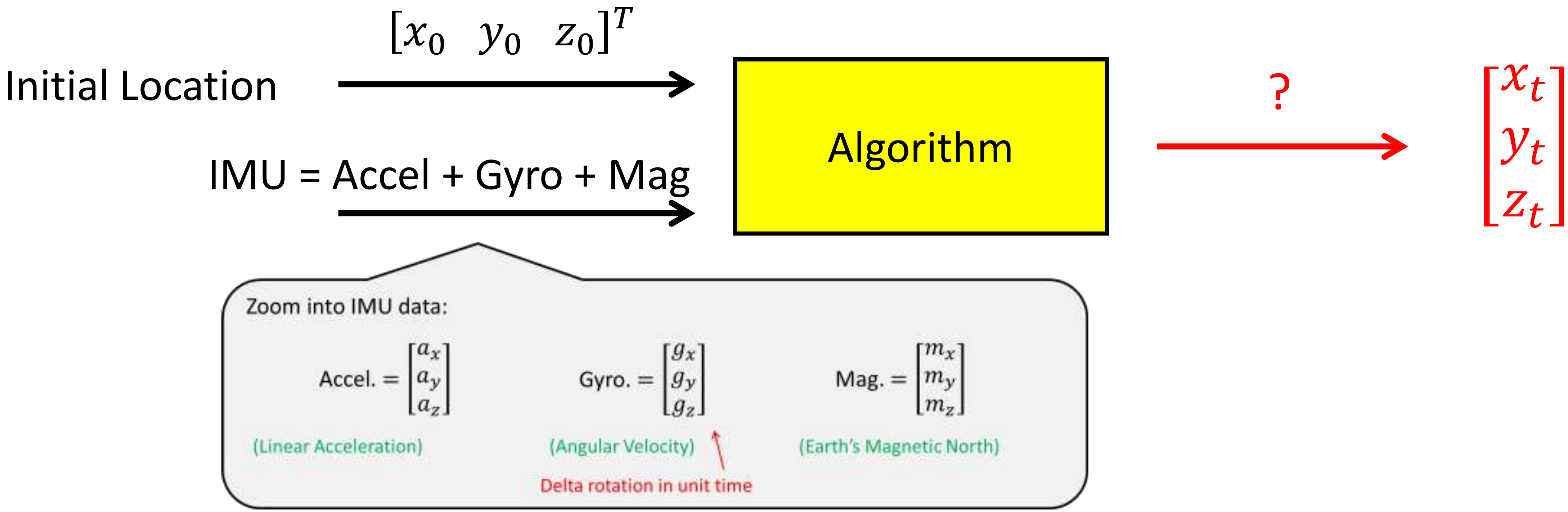
3D Trajectory

Map

Let's Understand the Inputs



Can we solve localization with these inputs?



One possibility is:

Initial Location $\xrightarrow{[x_0 \quad y_0 \quad z_0]^T}$

IMU $\xrightarrow{\text{Accel., Gyro., Mag.}}$

$$\begin{bmatrix} x_0 \\ y_0 \\ z_0 \end{bmatrix} + \iint_0^t (Accel.)\, dt^2 \longrightarrow \begin{bmatrix} x_t \\ y_t \\ z_t \end{bmatrix}$$

Zoom into IMU data:

$\text{Accel.} = \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix}$ (Linear Acceleration)

$\text{Gyro.} = \begin{bmatrix} g_x \\ g_y \\ g_z \end{bmatrix}$ (Angular Velocity)

Delta rotation in unit time

$\text{Mag.} = \begin{bmatrix} m_x \\ m_y \\ m_z \end{bmatrix}$ (Earth's Magnetic North)

But there is one BIG problem: $\text{Accel.} = \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix}$ is in local reference frame

But there is one BIG problem: Accel. $= \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix}$ is in local reference frame

But there is one BIG problem: Accel. $= \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix}$ is in local reference frame

As an analogy

Chicago Airport →

Accelerating →

a_x: Frontwards

a_y: Rightwards

a_z: Upwards

→ 3D Trajectory

Map

But there is one BIG problem: Accel. $= \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix}$ is in local reference frame

As an analogy

Chicago Airport →

Accelerating →

a_x: Frontwards

a_y: Rightwards

a_z: Upwards

What is
- Frontwards
- Rightwards
- Upwards

w.r.t. the earth?

3D Trajectory →

Map

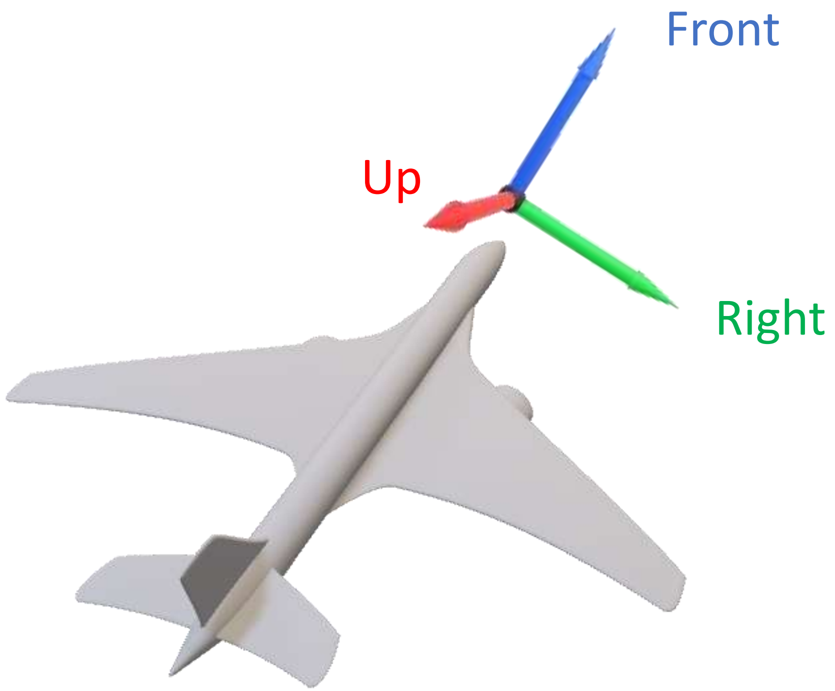
This makes orientation necessary

This makes orientation necessary

- What is 3D orientation?

Orientation is the **3D rotation needed** to make:

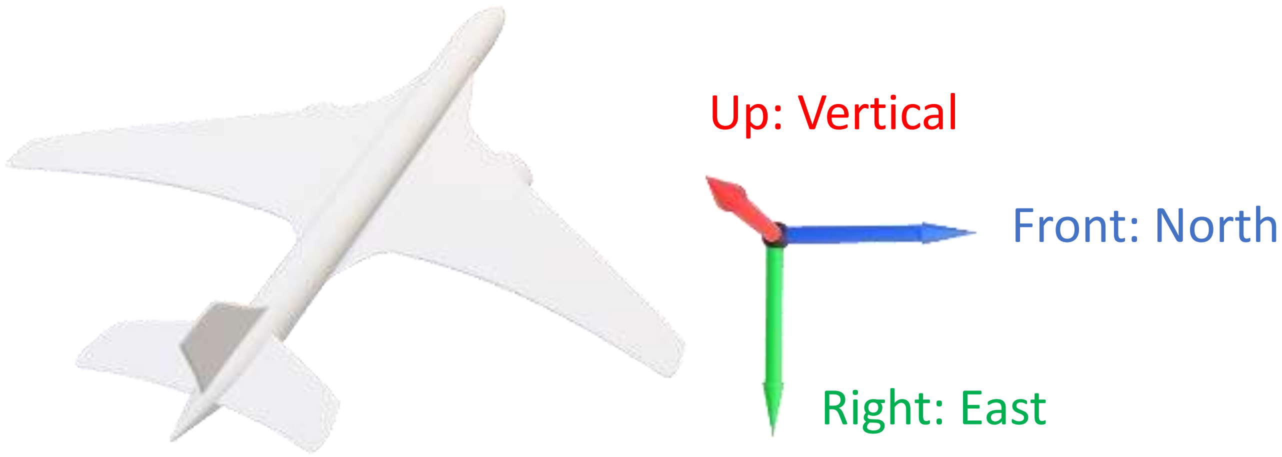
Plane's Own $\begin{bmatrix} \text{Frontwards} \\ \text{Rightwards} \\ \text{Upwards} \end{bmatrix} \rightarrow \begin{bmatrix} \text{Northwards} \\ \text{Eastwards} \\ \text{Vertical} \end{bmatrix}$



This makes orientation necessary

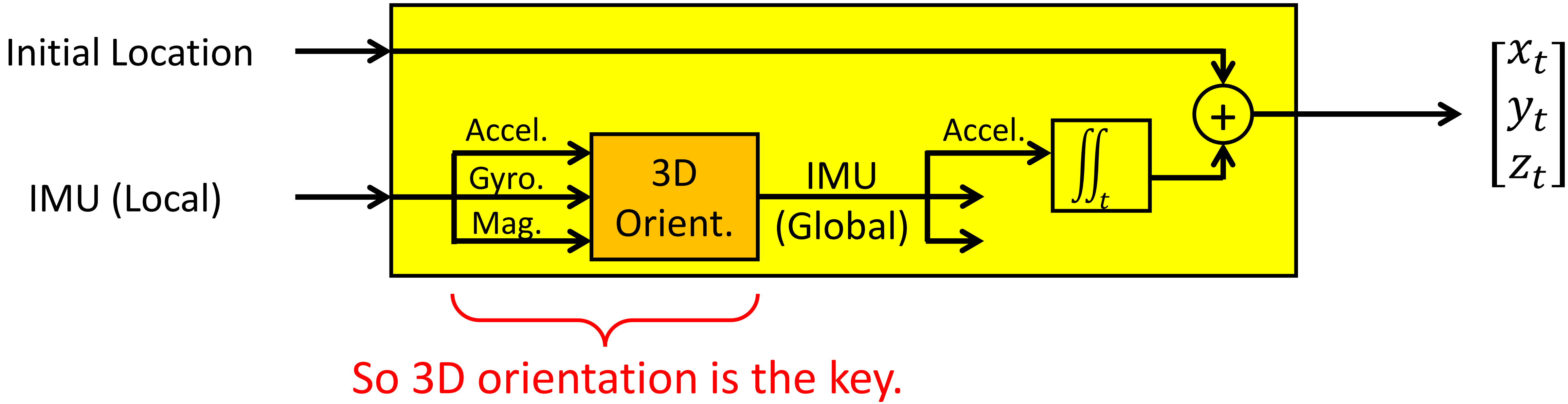
- What is 3D orientation? Orientation is the **3D rotation needed** to make:

$$\text{Plane's Own} \begin{bmatrix} \text{Frontwards} \\ \text{Rightwards} \\ \text{Upwards} \end{bmatrix} \rightarrow \begin{bmatrix} \text{Northwards} \\ \text{Eastwards} \\ \text{Vertical} \end{bmatrix}$$



So, how will orientation solve the problem?

What we need to do is:



So how to get 3D orientation?

2 Main Opportunities: 1. Gravity 2. Magnetic North

Both measurable by IMU

Key idea: What rotation is needed such that

1) **Gravity** is exactly in my **downward** direction
2) **North** is exactly in my **frontward** direction

So how to get 3D orientation?

Key idea: What rotation is needed such that

1) **Gravity** is exactly in my **downward** direction
2) **North** is exactly in my **frontward** direction

So how to get 3D orientation?

Key idea: What rotation is needed such that

1) **Gravity** is exactly in my **downward** direction
2) **North** is exactly in my **frontward** direction

$$\begin{bmatrix} 3 \times 3 \\ \text{Rotation} \\ \text{Matrix} \end{bmatrix} \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -9.8 \end{bmatrix}$$

Tilt is determined (2 out of 3 DoFs)

Gravity says a lot about orientation, but not sufficient

Gravity

Downward

So how to get 3D orientation?

Key idea: What rotation is needed such that

1) **Gravity** is exactly in my **downward** direction
2) **North** is exactly in my **frontward** direction

$$\begin{bmatrix} 3\times 3 \\ \text{Rotation} \\ \text{Matrix} \end{bmatrix} \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -9.8 \end{bmatrix}$$

Tilt + Heading is determined (all 3 DoFs)

$$\begin{bmatrix} 3\times 3 \\ \text{Rotation} \\ \text{Matrix} \end{bmatrix} \begin{bmatrix} a_x & m_x \\ a_y & m_y \\ a_z & m_z \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 100 \\ -9.8 & 0 \end{bmatrix}$$

My orientation

North

Frontward

So is Gravity + North enough to get 3D Orientation?

Only when object is **static** … but not otherwise. **Why?**

So is Gravity + North enough to get 3D Orientation?

Only when object is **static** … but not otherwise. **Why?**

Because any motion of the object will reflect in the accelerometer … thereby polluting the gravity estimate

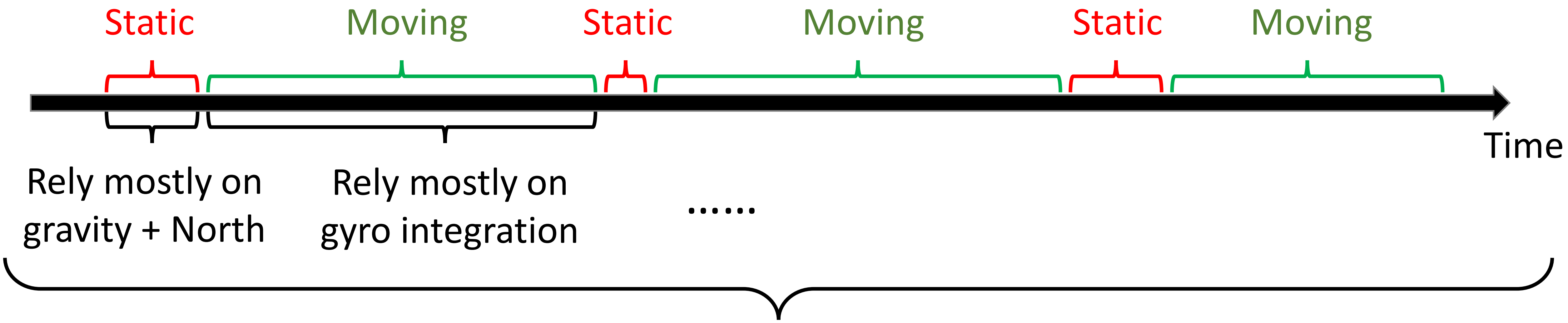
So how to get 3D orientation? (Another idea)

Another Idea for Orientation: Integrate angular velocity from gyro

Initial Orientation + $\int_0^t (Gyro.)\, dt$ = New Orientation (at time t)

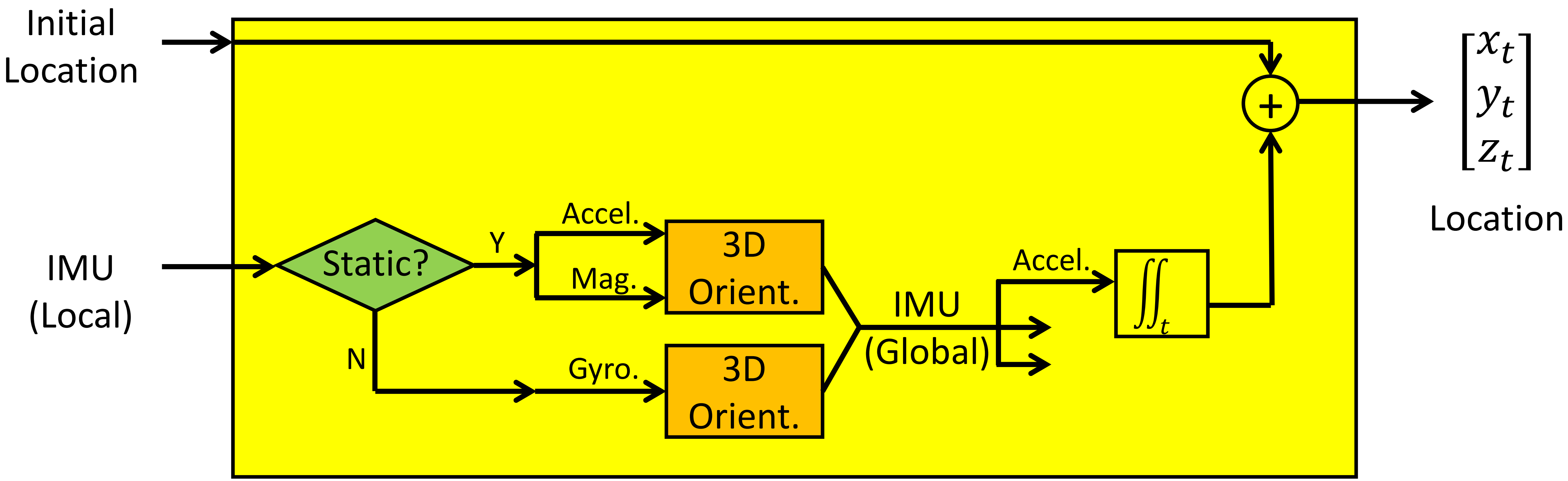
But gyro drifts, so only useful in short time scales

State of the art today: **Sensor Fusion**

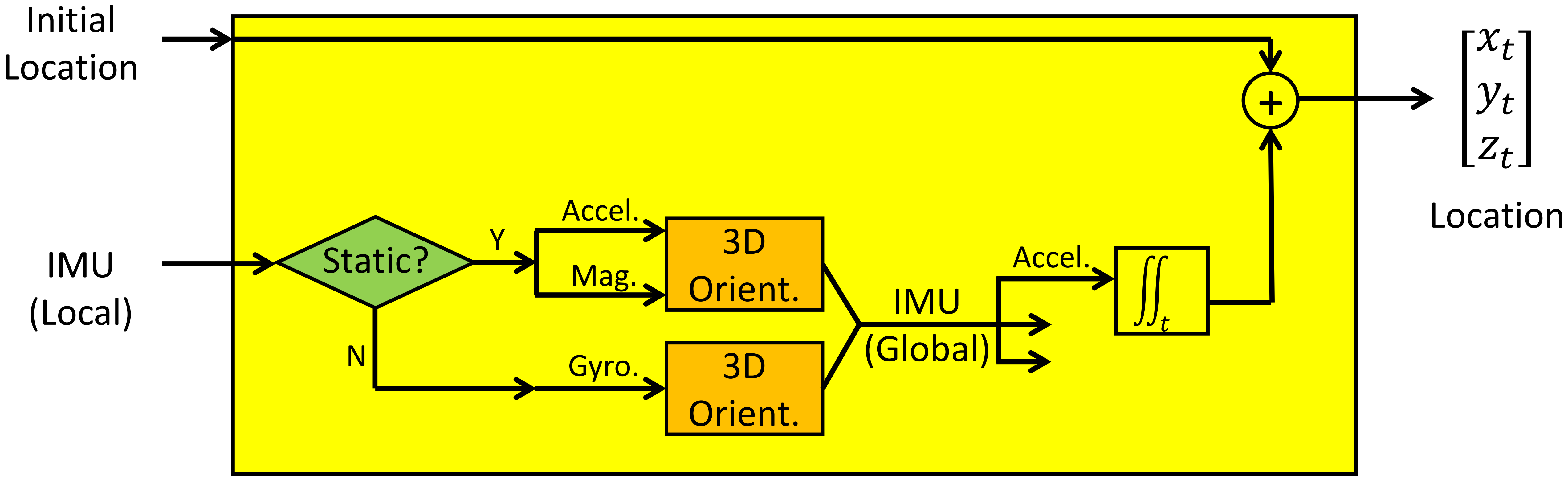


Always know 3D orientation

Getting back to our goal



Getting back to our goal



Main take away: Gravity is the main anchor for 3D orientation

But what if object is not often static

But what if object is not often static

But what if object is not often static

No good solution today …

Your job to solve the problem …

Questions ?