Mobile Robot Programming Laboratory

Lab 3

Thursday Week 3

http://www.andrew.cmu.edu/course/16-362-862

Agenda Today

- Administrative Issues
- Estimation vs Prediction
- WMR Modeling
- Stuff Worth Knowing
- Concurrent Programming
- Overview of Lab 3

Agenda Today

Carnegie Mellon
THE ROBOTICS INSTITUTE

- **Administrative Issues**
- Estimation vs Prediction
- WMR Modeling
- Stuff Worth Knowing
- Concurrent Programming
- Overview of Lab 3

Agenda Today

- Administrative Issues
- **Estimation vs Prediction**
- WMR Modeling
- Stuff Worth Knowing
- Concurrent Programming
- Overview of Lab 3

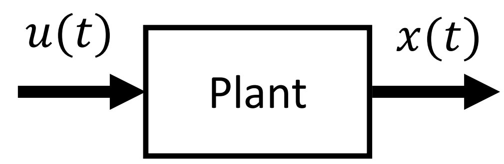
Prediction

- System model relates

  state: $x = [x\ y\ \theta]$ to…

  inputs: $u = [V \omega]$

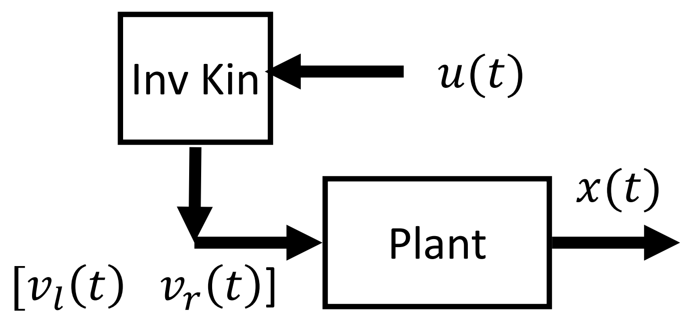
  That you are about to send to the robot.



- Relationship is a differential equation.

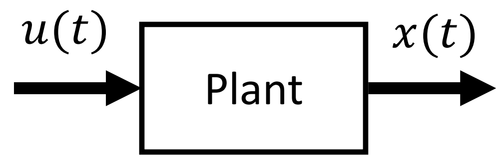
$$\dot{x} = f(x, u)$$

- Sometimes you also need **inverse** kinematics
  - what wheel velocities should I command to get $u = [V \omega]$ used in first bullet?



Prediction

- In prediction, knowing $\dot{x}$ is not enough.
- You want to know x - where the robot will go if you command some $u(t)$.
- In a computer, integrate in discrete time.



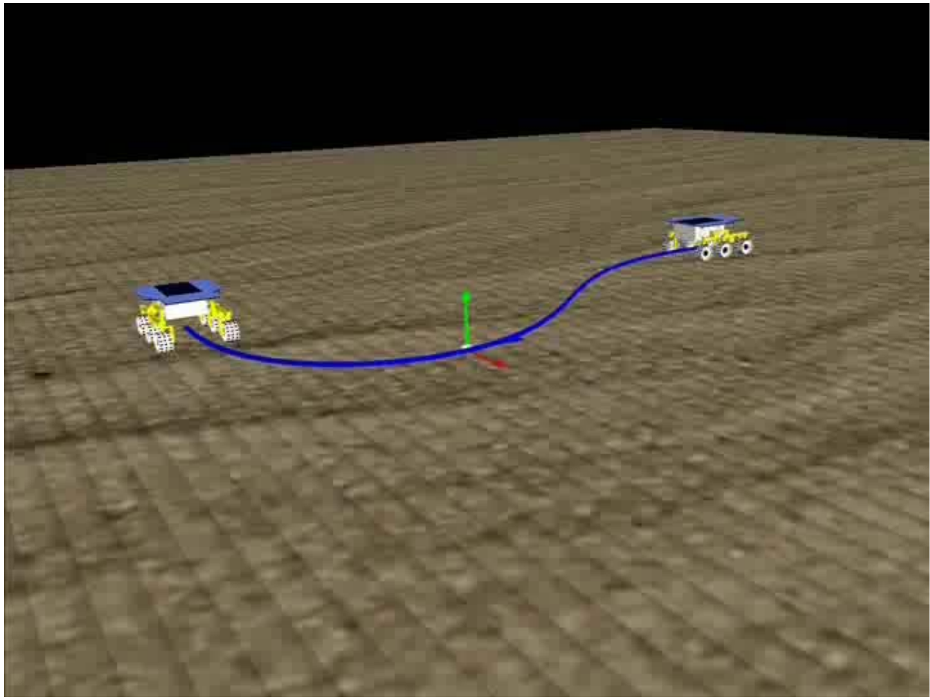
$$\dot{x} = f(x, u)$$

$$x = \int f(x, u) dt$$

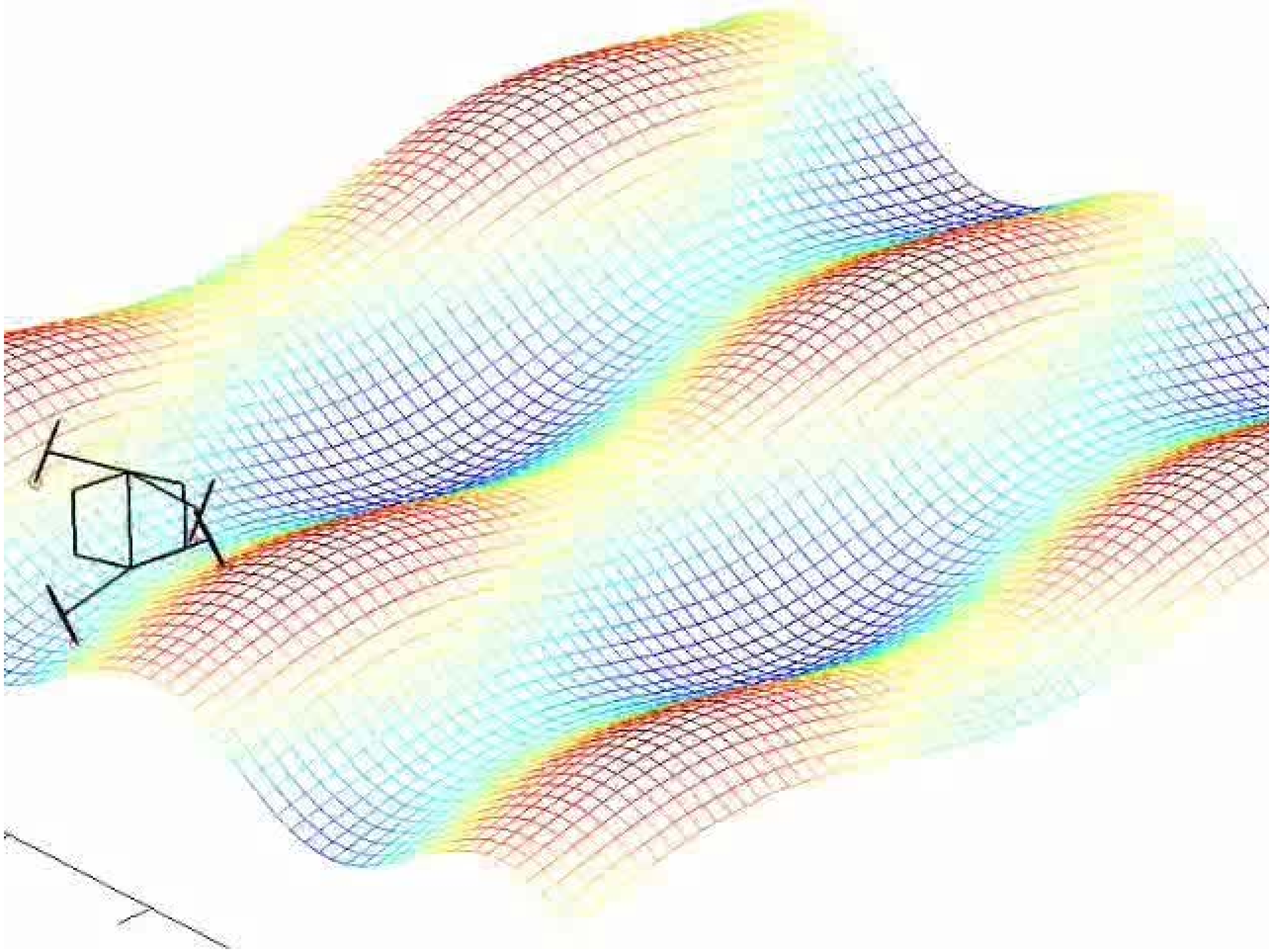
$$x = \sum f(x, u) \Delta t$$

Terrain Following in 3D

- Prediction in 3D requires modeling of:
  - Suspension articulation
  - Terrain Contact
  - Wheel Slip



Zoë Experiment



Estimation

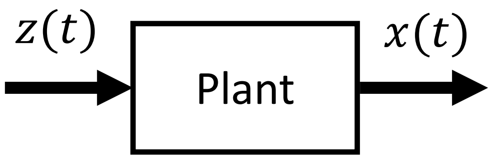
Carnegie Mellon
THE ROBOTICS INSTITUTE

- System model relates

  state: $x = [x \; y \; \theta]$ to…

  measurements: $z = [V \omega]$

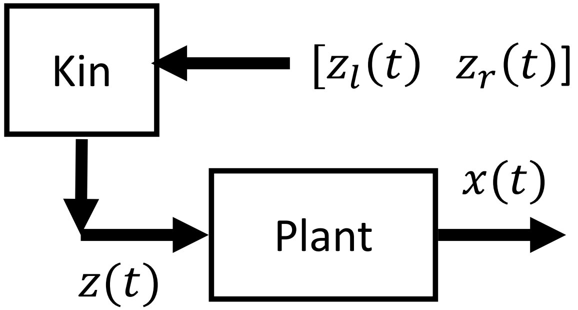
  That you just received from the robot.



- Relationship is a differential equation.

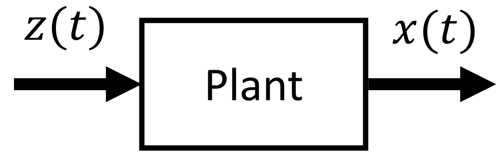
$$\dot{x} = f(x, z)$$

- Sometimes you also need **forward** kinematics
  - What speeds $z = [V \omega]$ are consistent with the wheel speeds I am measuring now.



Estimation

- In estimation, knowing $\dot{x}$ is not enough.
- You want to know x - where the robot is now based on the measurement history $z(t)$.
- In a computer, integrate in discrete time.



$$\dot{x} = f(x, z)$$

$$x = \int f(x, z)dt$$

$$x = \sum f(x, z)\,\Delta t$$

Comparison

1) The math is identical.
2) You decide what the symbols mean.
3) If written generically, you need to write the code only once.

- Prediction

$$\dot{x} = f(x, u)$$

$$x = \sum f(x, u)\,\Delta t$$

- Estimation

$$\dot{x} = f(x, z)$$

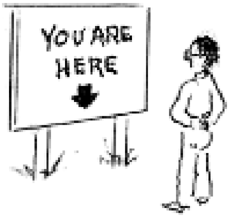
$$x = \sum f(x, z)\,\Delta t$$

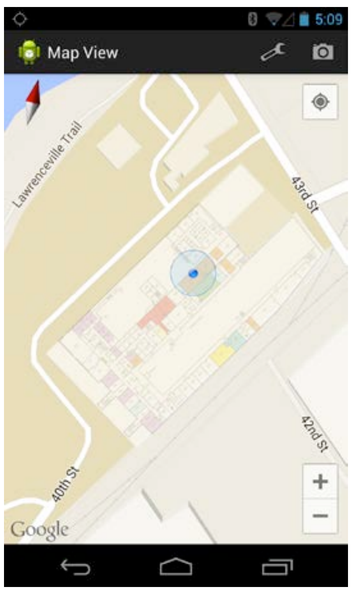
Material Handling AGV Guidance

Personal Inertial Navigation

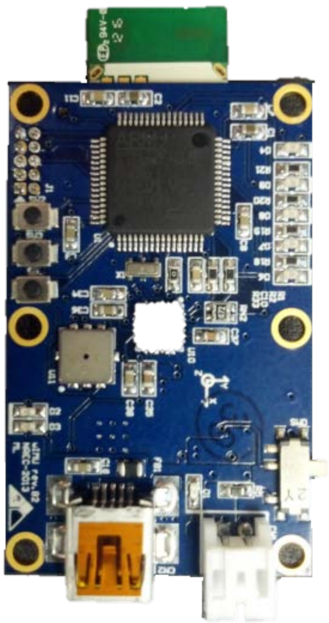
Carnegie Mellon THE ROBOTICS INSTITUTE

- Indoor “GPS” for airports, malls.
- Entire Inertial Navigation System runs on a microcontroller.





Cell Phone User Interface



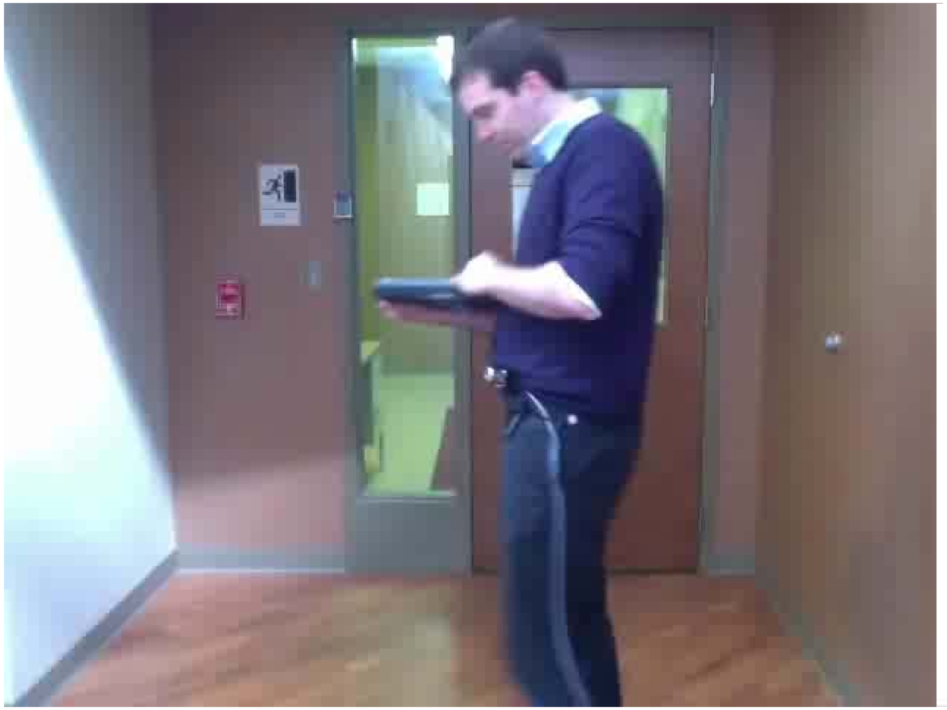
Board Layout

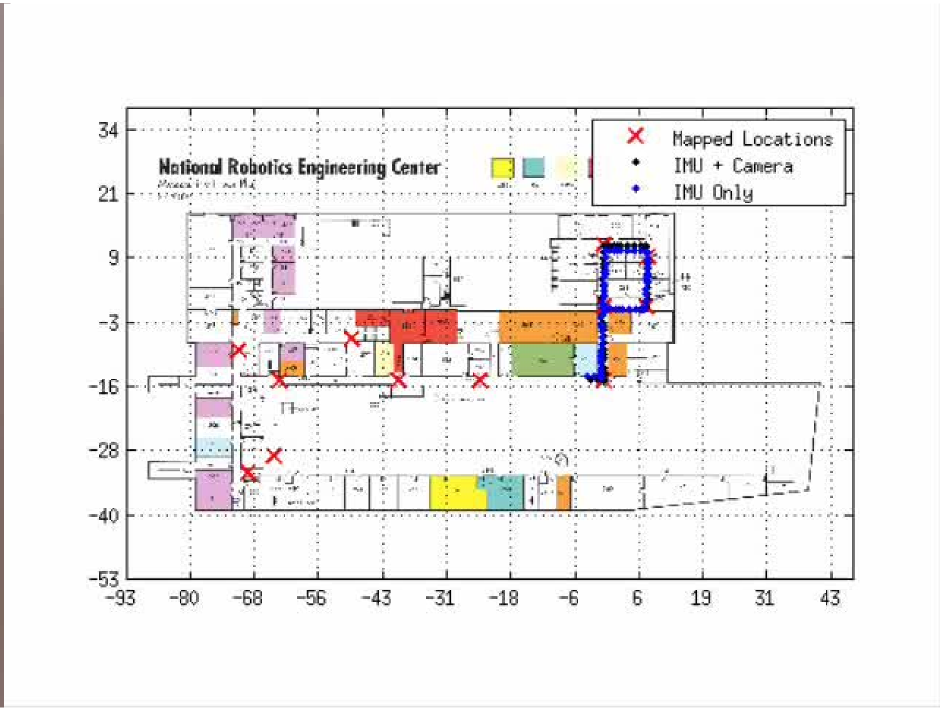
Laced Into Shoe

User Interface

Wireless IMU

Personal Navigation





Agenda Today

- Administrative Issues
- Estimation vs Prediction
- **WMR Modeling**
- Stuff Worth Knowing
- Concurrent Programming
- Overview of Lab 3

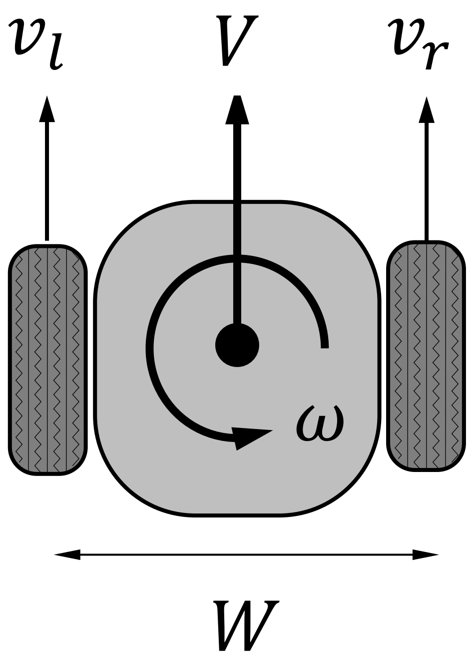
Implementing "Odometry"

- Step 1: Differentiate encoders

$$v_l = \frac{ds_l}{dt}$$
$$v_r = \frac{ds_r}{dt}$$

- Step 2: Find linear and angular velocity of rigid body (called "robot").
  - Inverse Kinematics

$$V = \frac{(v_r + v_l)}{2}$$
$$\omega = \frac{(v_r - v_l)}{W}$$



Recall Path definitions

- Linear velocity

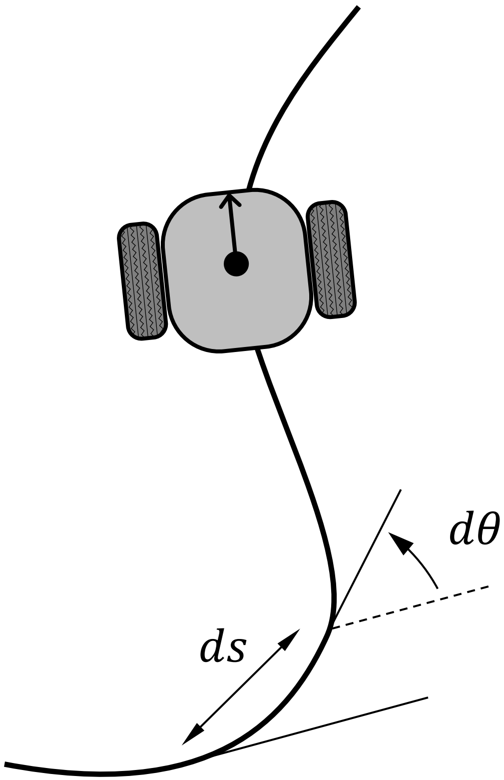
$$V = \frac{ds}{dt} \rightarrow \; ds = V\,dt$$

- Angular velocity

$$\omega = \frac{d\theta}{dt} \rightarrow \; d\theta = \omega dt$$

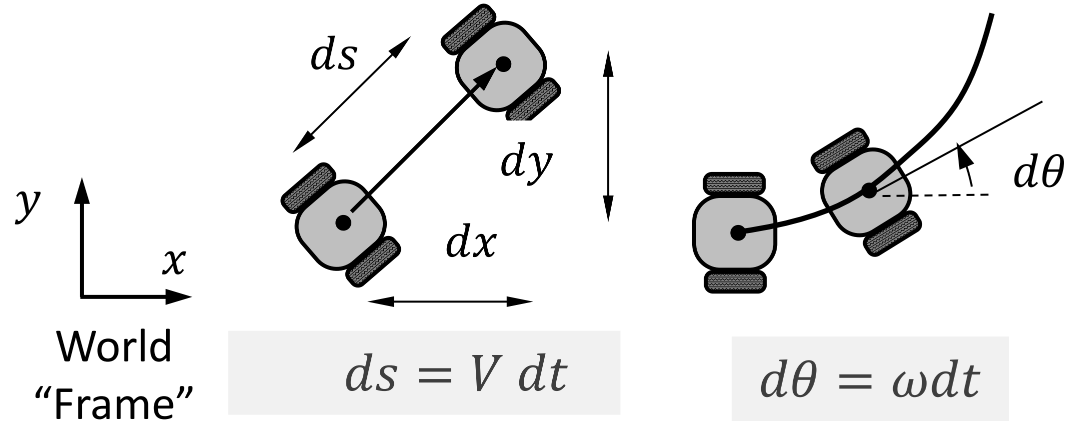
- Curvature

$$\kappa = \frac{d\theta}{ds} \rightarrow \; d\theta = \kappa ds$$



Implementing "Odometry"

- Step 3: Compute incremental rotation and translation (see opposite)



$$ds = V\,dt \qquad d\theta = \omega dt$$

- Step 4: Project translation onto the world frame:

$$dx = cos(\theta)ds$$
$$dy = sin(\theta)ds$$

Implementing “Odometry”

- Step 5: Add it all up.
- It matters (sometimes alot) if you update theta first or last.
- “Midpoint” algorithm (sort of..)
  - $\theta = \theta + \omega dt/2$
  - Do x, y
  - $\theta = \theta + \omega dt/2$

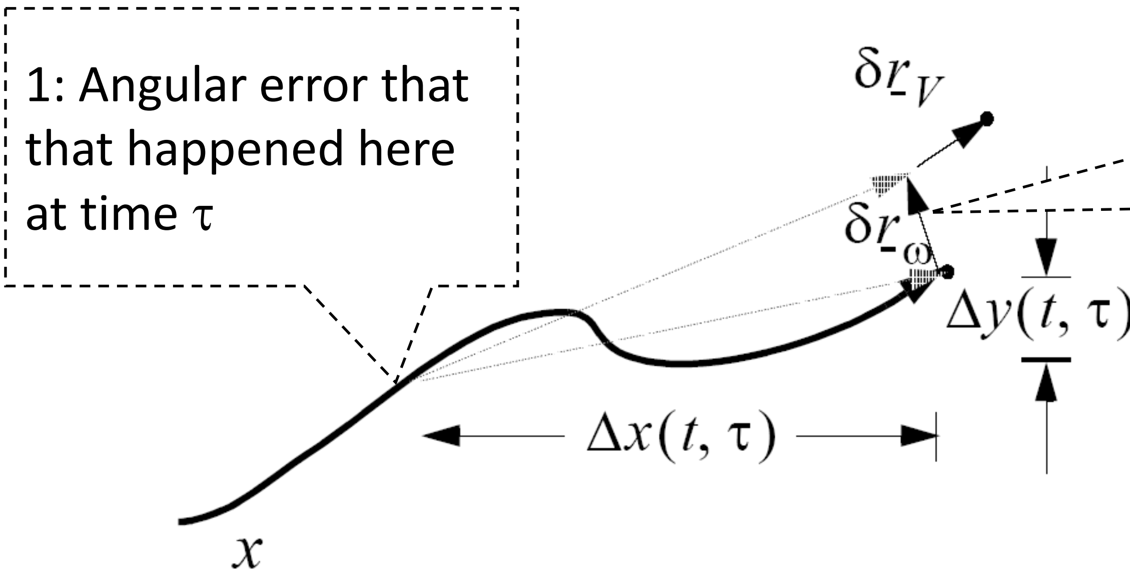
```
x=x(t0)
y=y(t0)
θ=θ (t0)
while( t < tf)
    θ = θ + ωdt
    x = x + Vcos(θ)dt
    y = y + Vsin(θ)dt
end
```

This is most of what you need for this lab

Agenda Today

- Administrative Issues
- Estimation vs Prediction
- WMR Modeling
- **Stuff Worth Knowing**
- Concurrent Programming
- Overview of Lab 3

Eg: Error Propagation



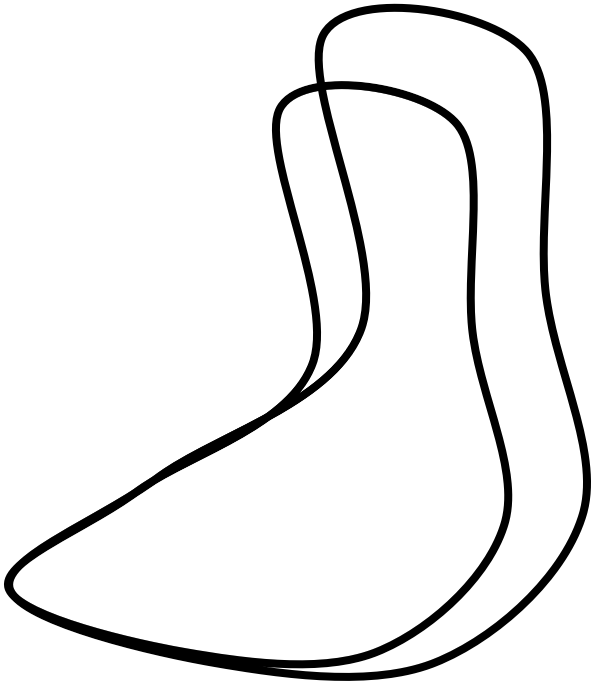
$$d\begin{bmatrix}\delta x(t)\\ \delta y(t)\\ \delta\theta(t)\end{bmatrix} = \begin{bmatrix}c\theta & -\Delta y(t,\tau)\\ s\theta & \Delta x(t,\tau)\\ 0 & 1\end{bmatrix}\begin{bmatrix}\delta V(\tau)\\ \delta\omega(\tau)\end{bmatrix}d\tau$$

- Error propagation process simply adds up the impact of <u>every</u> historical error to produce the present error.
- Dead reckoning never forgets an error.

Insights: Path Independence

Carnegie Mellon
THE ROBOTICS INSTITUTE

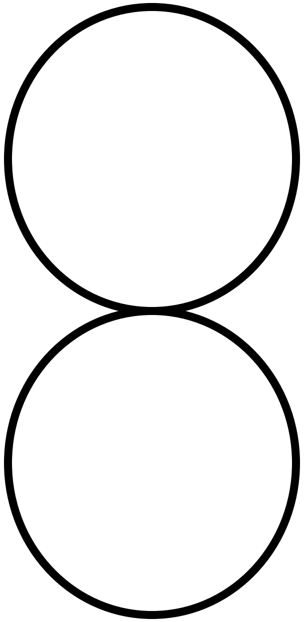
- Some errors (e.g. velocity scale errors) vanish on closed trajectories.
- So closed trajectories are not the way to test your encoder scale factor.



Insights: Symmetry

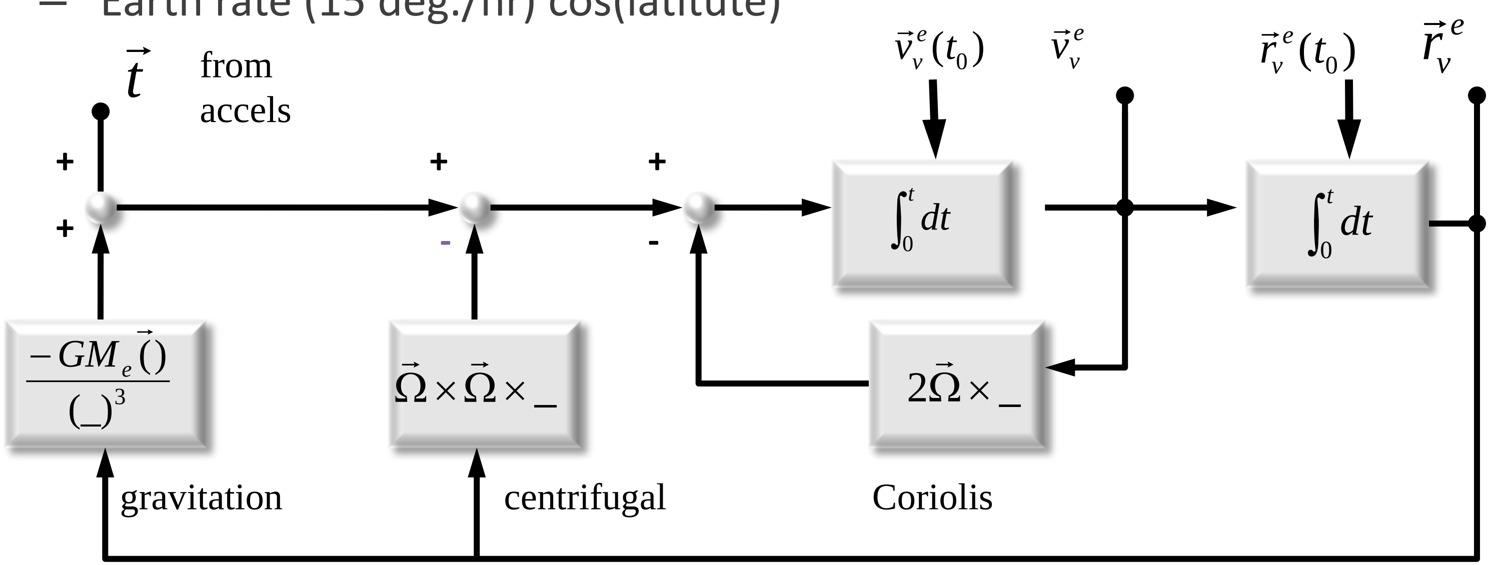
- Some errors (e.g. gyro bias) vanish at the centroid of the trajectory.

- The wrong way to assess gyro bias.



INS: Pragmatic Issues

- Need good initial conditions for:
  - Position, attitude, heading
  - These are almost never easy to get, heading is especially hard.
- Must remove from accelerometers:
  - Effect of gravity
  - Centrifugal and Coriolis forces
- Must remove from gyros:
  - Earth rate (15 deg./hr) cos(latitute)



Perturbative INS Error Analysis

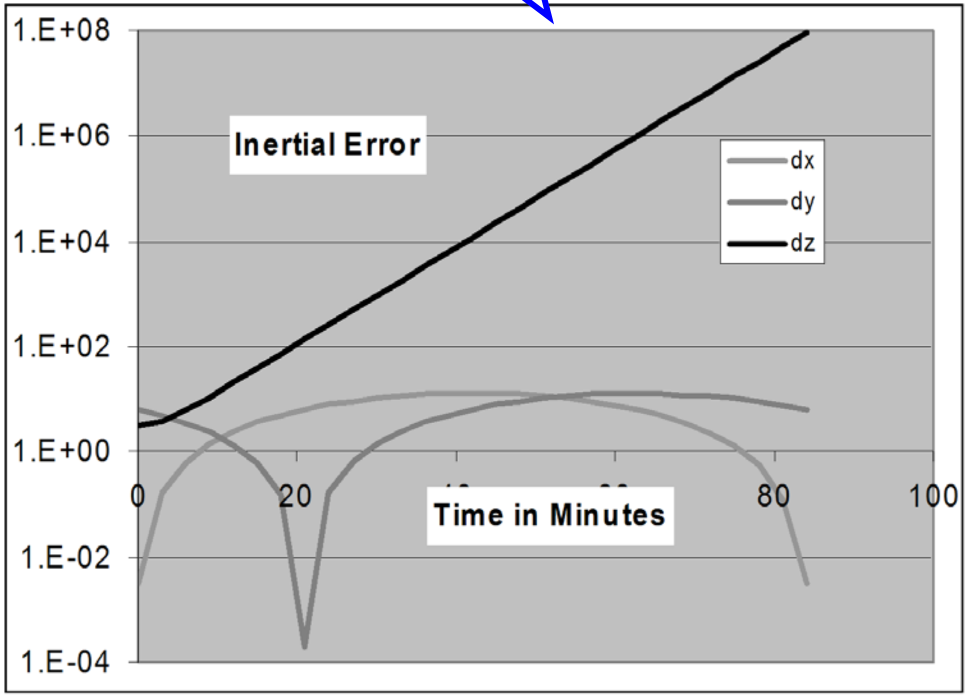
- If the accelerometer biases are constant, the solutions are:

$$\delta x = \frac{\delta t_x}{g_0 / R_0}\left[1 - \cos\left(\sqrt{\frac{g_0}{R_0}}t\right)\right]$$

$$\delta y = \frac{\delta t_y}{g_0 / R_0}\left[1 - \sin\left(\sqrt{\frac{g_0}{R_0}}t\right)\right]$$

$$\delta z = \frac{\delta t_z}{2g_0 / R_0}\left[\cosh\left(\sqrt{\frac{2g_0}{R_0}}t\right)\right]$$

Error magnitudes for 1 micro-g biases.



- Gravity field is a mixed blessing !!

Agenda Today

- Administrative Issues
- Estimation vs Prediction
- WMR Modeling
- Stuff Worth Knowing
- **Concurrent Programming**
- Overview of Lab 3

Consistency

- Most control algorithms perform better on, and many require, consistent data.
  - You would not change a variable by a small random amount in the middle of an algorithm.
  - Almost any algorithm will break or perform erratically. Robots will do random, even dangerous stuff
- Suppose there is no latency in sensor data, encoder data comes in at 50 Hz, robot is moving, and your code to pick up an object looks like so:
  - myPose = estimatePose(lastMessage) % first time
  - objPose = locateObject(myPose)
  - myPose = estimatePose(lastMessage) % second time
  - myTraj = computeTrajectory(objPose,myPose)

Consistency

- This algorithm will always underestimate the distance to the object because the robot moved between looking and doing.
  - Any robot always does move. Its up to you to deal with it in a principled manner.
  - You can at least use state estimates that are consistent in time (occur at the same time or roughly the same time).
- If the variables involved are logical rather than signals, inconsistency is a major major safety issue.
- Inconsistency is bad and the badness is proportional to:
  - The delay between updates
  - The speed of the robot
  - The sensitivity of the algorithm to inconsistency.

Concurrency

- YOU must control when changes to signals are allowed to enter your algorithm.
- Yet, any concurrent (multiprocessor, multiprocess, multi-thread) system does not control, out of the box, precisely when shared data is accessed.
  - So, it does change variables by random amounts in the middle of an algorithm.
- Even ROS is doing this under the hood when you access the LatestMessage struct.
- Moral:
  - DO NOT read the ROS lastMessage more than once in your "main loop" (the highest level loop of your entire algorithm).
  - Update, using incoming data (encoder, lidar, other) the robot and world state estimate however often you like in a concurrent thread/process/processor **BUT** introduce it **once at the top** of your main loop and use that state estimate everywhere else until your code returns to the top of main loop again.
- Mind these rules and save yourself sleepless nights in October and November chasing bugs that come and go randomly.

Concurrency - Callbacks

- Callbacks are not executed in your main thread. Hence concurrency issues…
- Doing odometry every time an encoder message arrives at 50 Hz is a good idea because it improves accuracy of the state estimate.
- BUT, passing those updates to your main algorithm at random times is a bad idea.
- You can use a semaphore or synchronous code but its pointless because your code is way slower than 50Hz anyway.

Concurrency - Callbacks

- Makes more sense to simply process encoder data as fast as you can and **wait** if there is no new data when you need it.
  - Works when your code runs at 400Hz because it waits for new data (and runs at 50Hz)
  - Works when your code runs at 10 Hz (because it uses the latest encoder update)
    - BTW you could queue the updates and process all of them. That is slightly more accurate.
- Optimal solution is to do odometry at 50Hz but update state in the main loop once per iteration.
  - However doing odometry as fast as the main loop using the latest encoder data at that time is good enough for this course.

Dealing with Latency

- Everything about concurrency above matters even without sensor data latency.
- Latency is a second, compounding issue.
  - Encoder messages are always late with respect to reality.
  - The latency of the real data cannot be eliminated.
  - So, your state estimate is always where the robot "was".
- It is possible to use prediction to remove the latency error at the cost of introducing prediction error.
  - Compute DRState as usual every cycle
  - Use PrState where PrState = DRState plus (V,w) times the latency.
- In addition to incoming measurements (encoders, lidar), there are equivalent latency issues for the commands going out to the robot.

Agenda Today

- Administrative Issues
- Estimation vs Prediction
- WMR Modeling
- Stuff Worth Knowing
- Concurrent Programming
- **Overview of Lab 3**

Preparation

- Did everyone read lab 3 before today?

Lab 3 Preparation

- Click for Lab3 Writeup