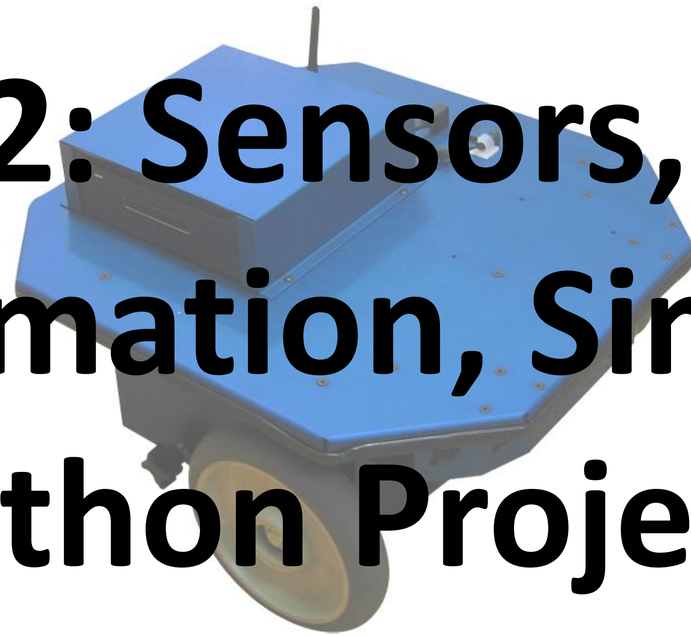
COMP 551: Advanced Robotics Lab



Lec02: Sensors, Pose Estimation, Simple Python Projects

James McLurkin
Rice University
jmclurkin@rice.edu

Sensors

Sensors, sensors, everywhere

We’re surrounded by sensors

You can sense anything and everything.

For example, let’s say you want to sense obstacles:

Lever Switch

IR Range Finder

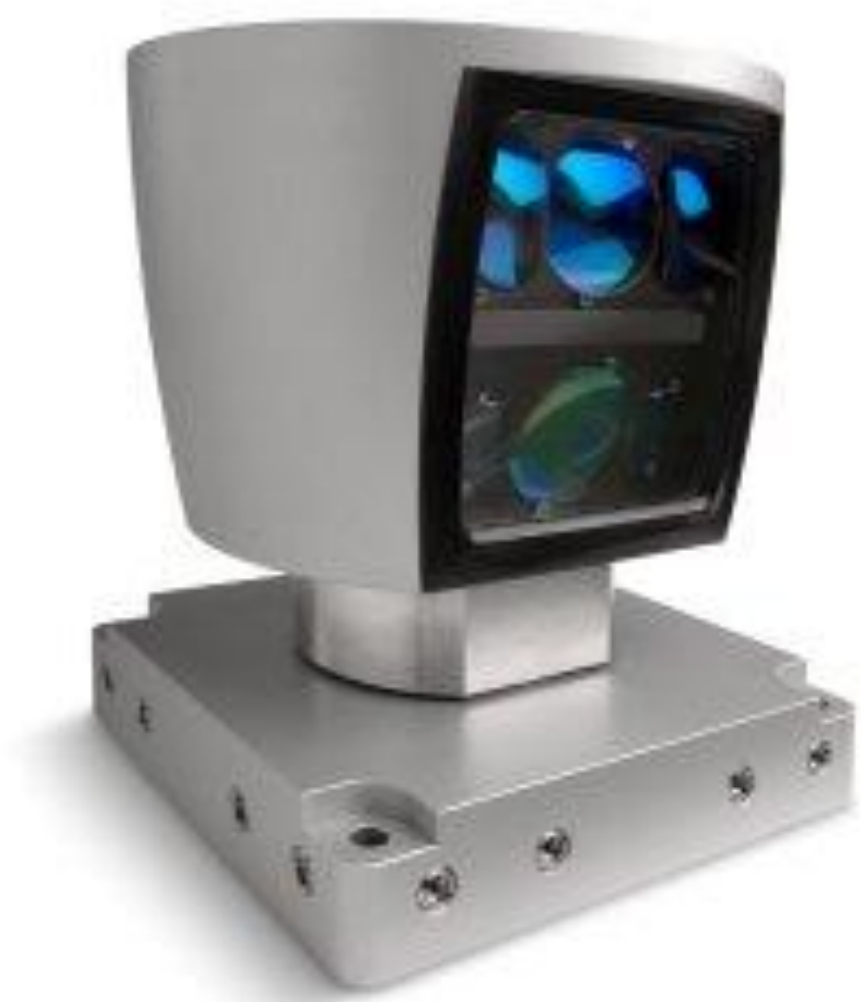
Sonar

Radar

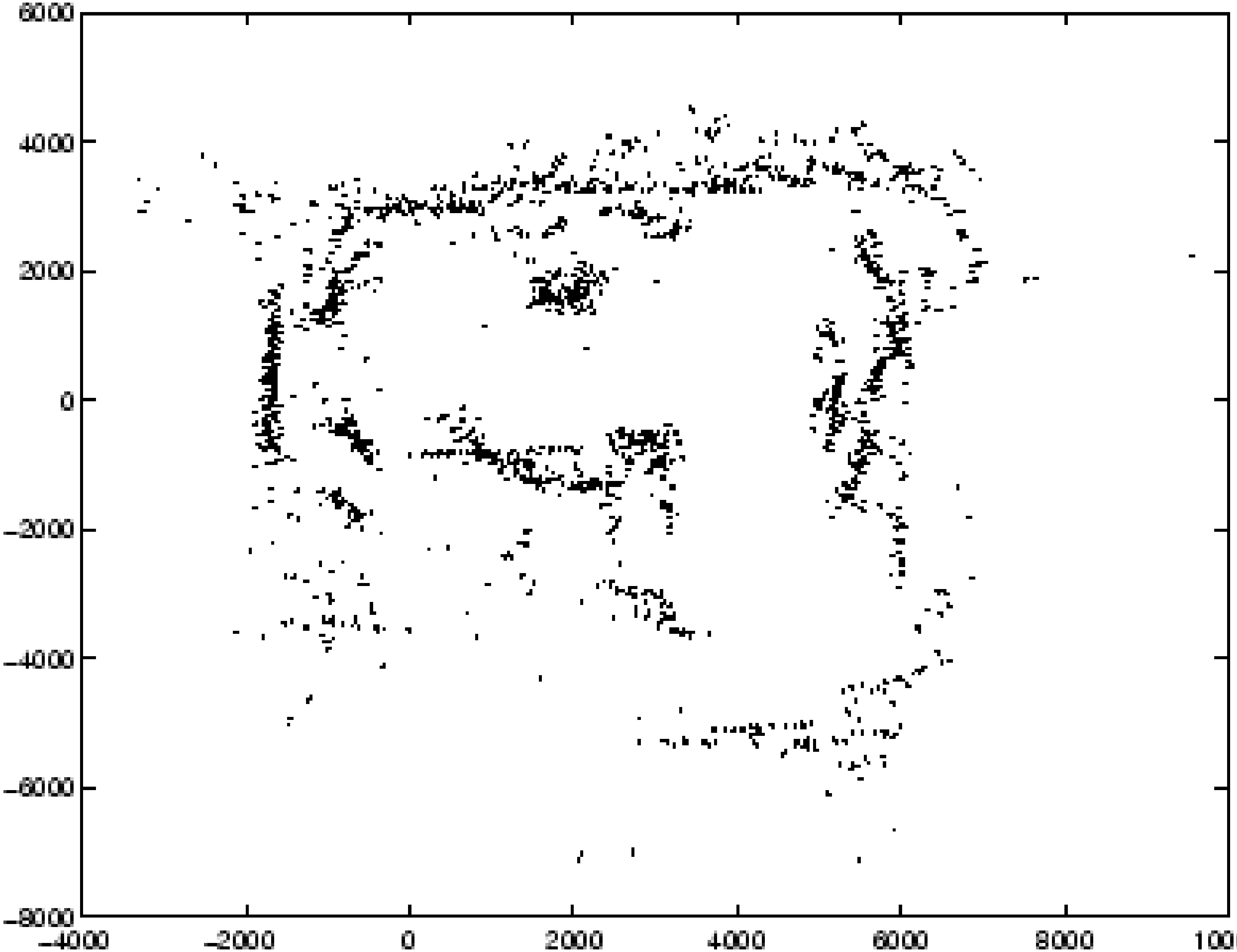
Lidar

Lidar (smaller)

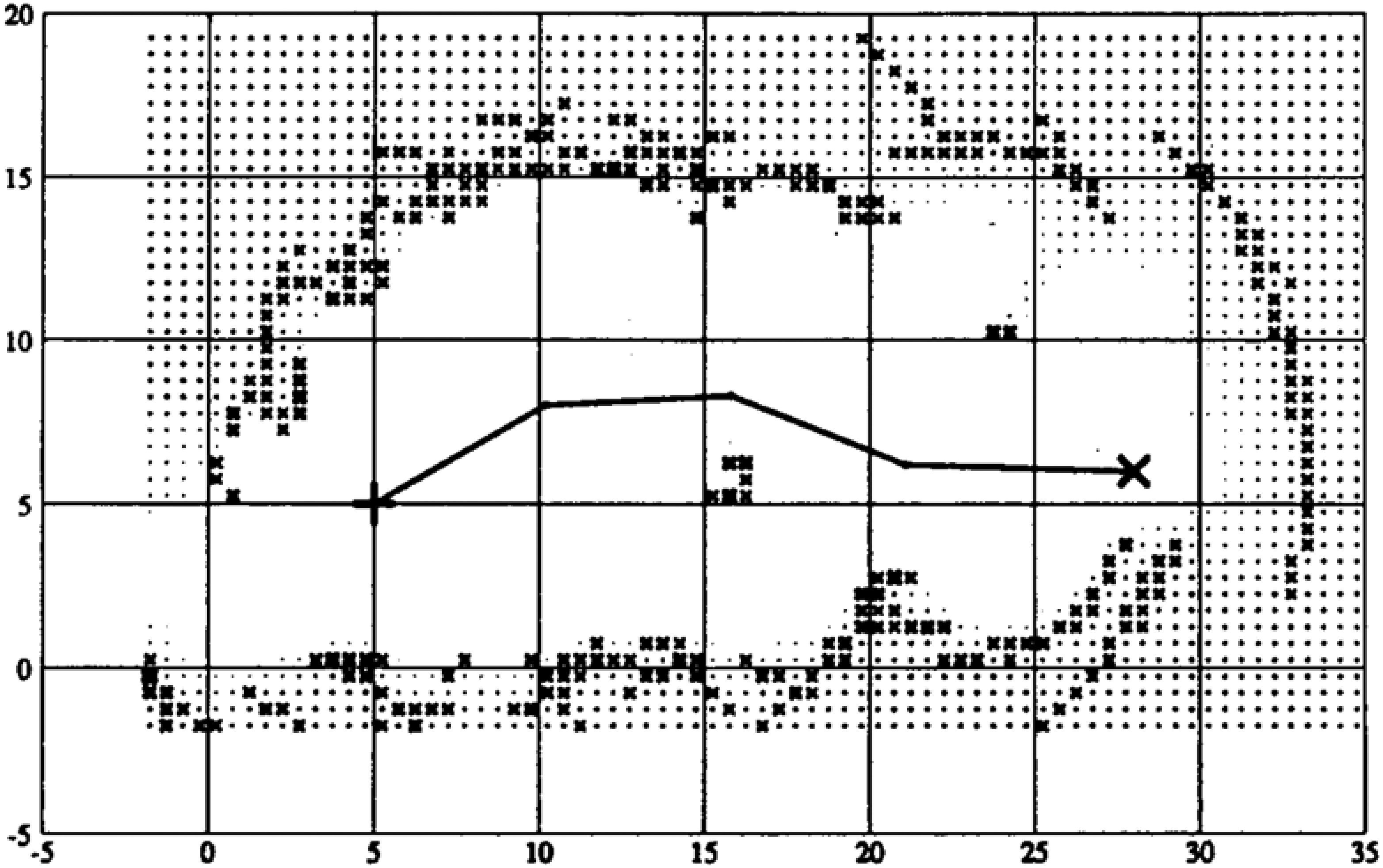
Lidar (insane)



Map from Sonar



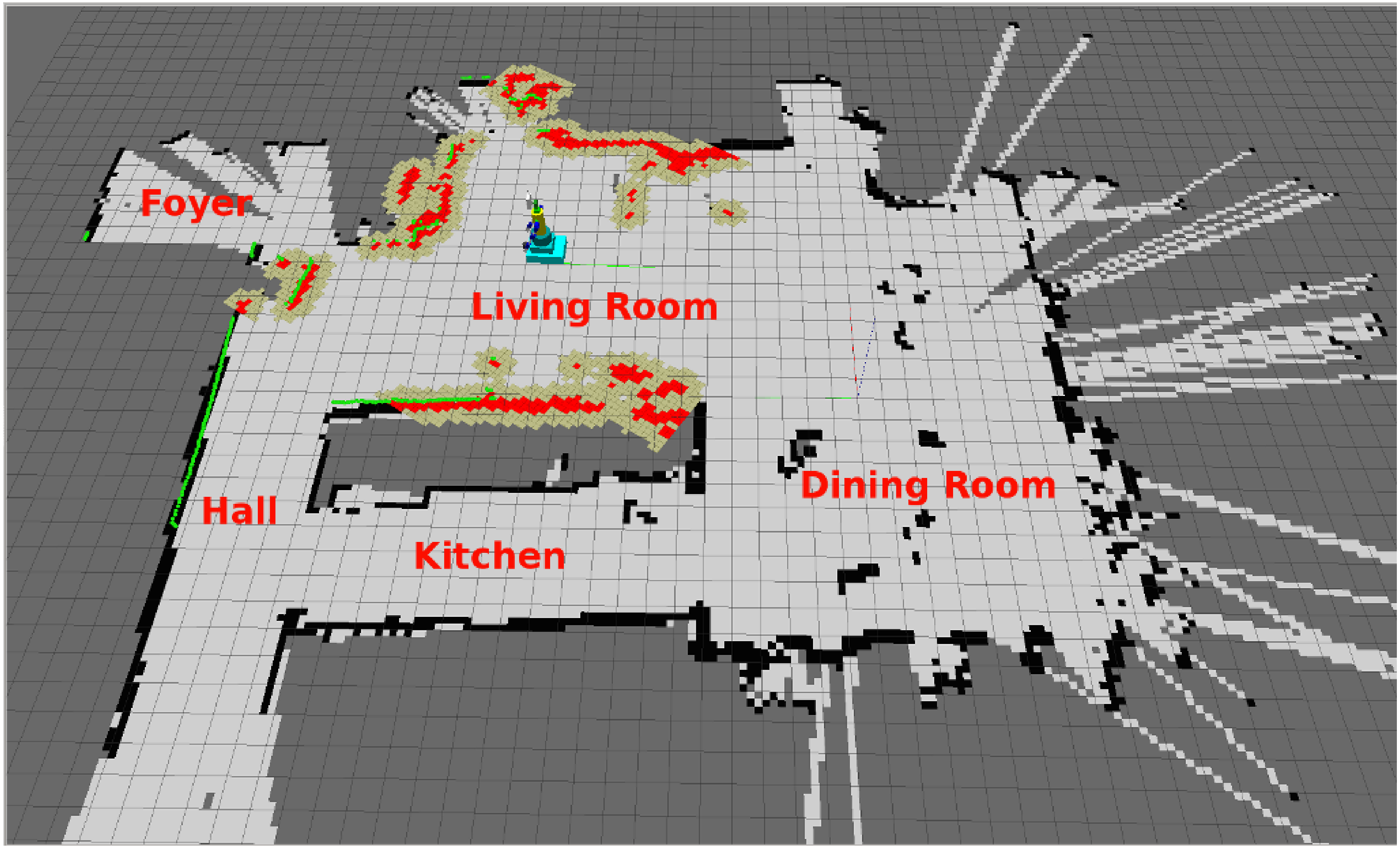
Map from Sonar



(e)

Map from Lidar

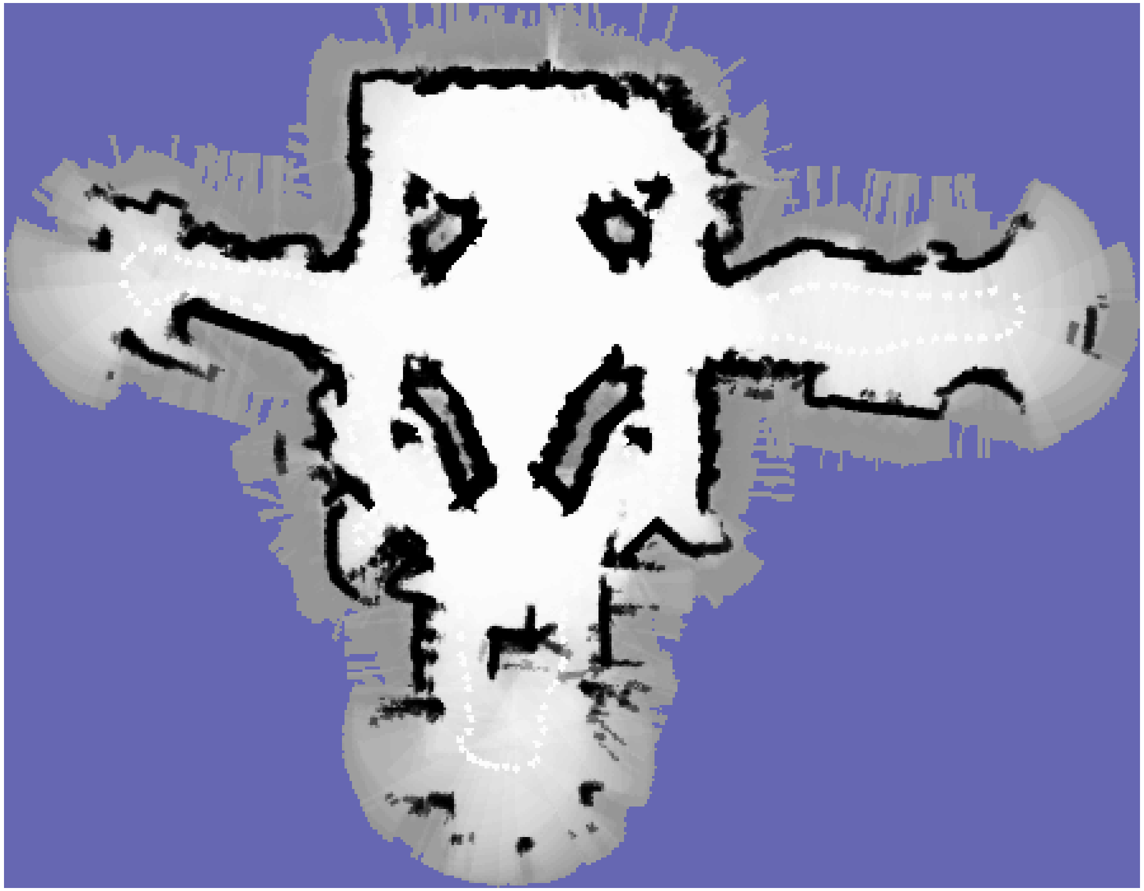
Map from Lidar



Occupancy Grid Mapping

Use laser scanner to detect obstacles

Use sensor model and “better pose” to produce a map



Particle Filter Localization (MCL)

Produce lots of estimates of current position

Keep the good ones

Particle Filter Localization (MCL)

Produce lots of estimates of current position

Keep the good ones

SLAM in Large-Scale Cyclic Environments Using the Atlas Framework

Michael Bosse, Paul Newman,
John Leonard, Seth Teller

International Journal of Robotics Research
February, 2004

The Sensor Model

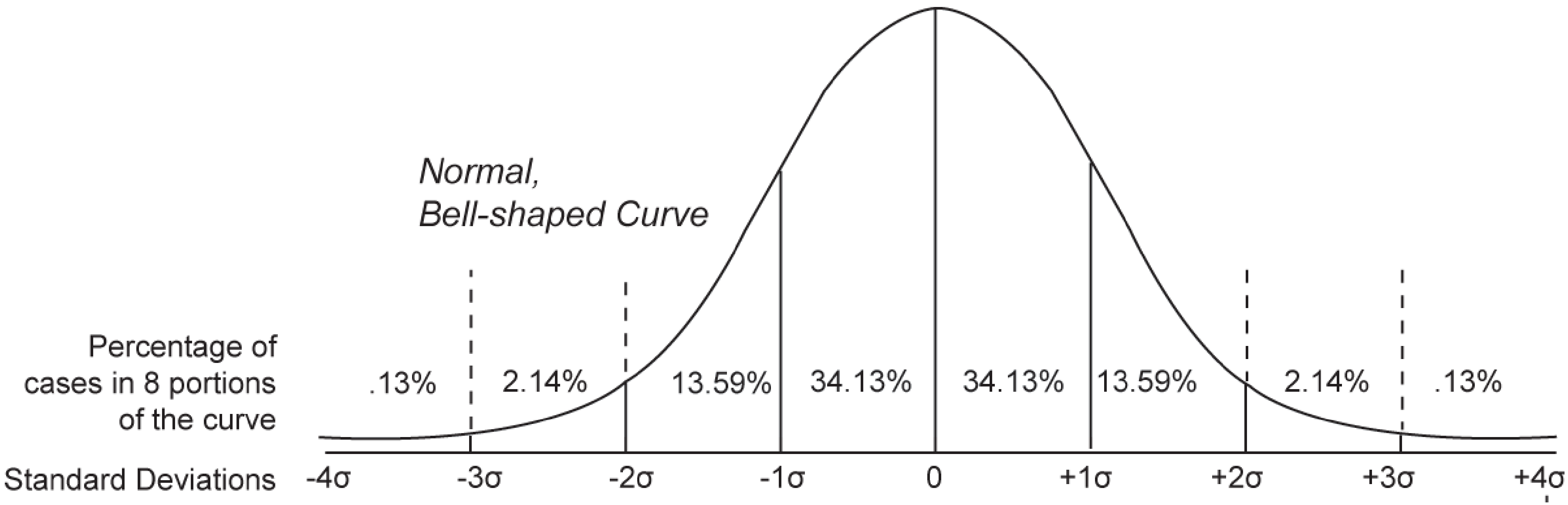
There are many ways to sense any physical quantity

Each approach has trade-offs in cost, complexity, and accuracy

We need to understand these basic parameters of the ***sensor model*** in order to select the right sensors for the job

We often represent this model as a ***normal distribution…***

Normal Distribution



Looks complex, but can be represented with two parameters

- This is convenient for math
- It can be easily manipulated

Surprisingly accurate

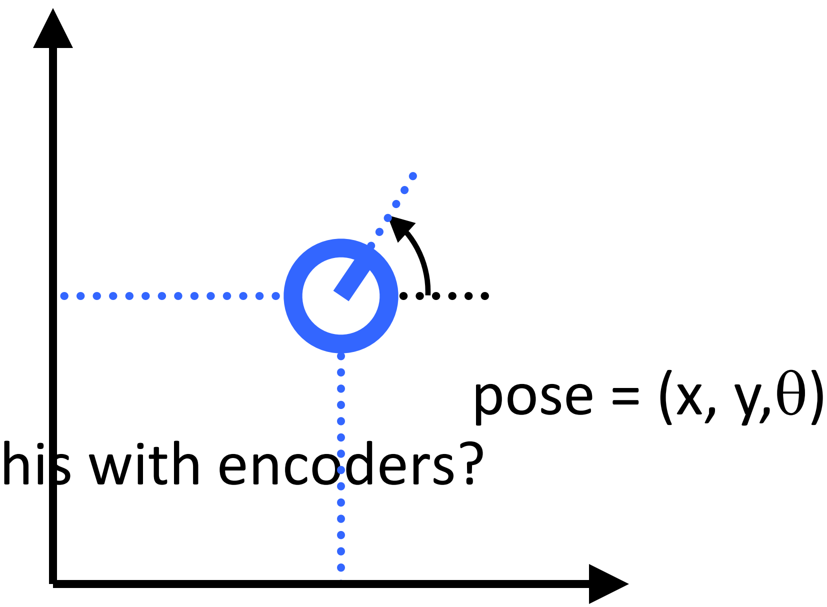
- Even when the underlying physical phenomena is not normal

Using Sensors: Odometry

Pose Estimation

We can describe the robot’s *pose* in an external reference frame:

This is useful for getting the robots around in the world



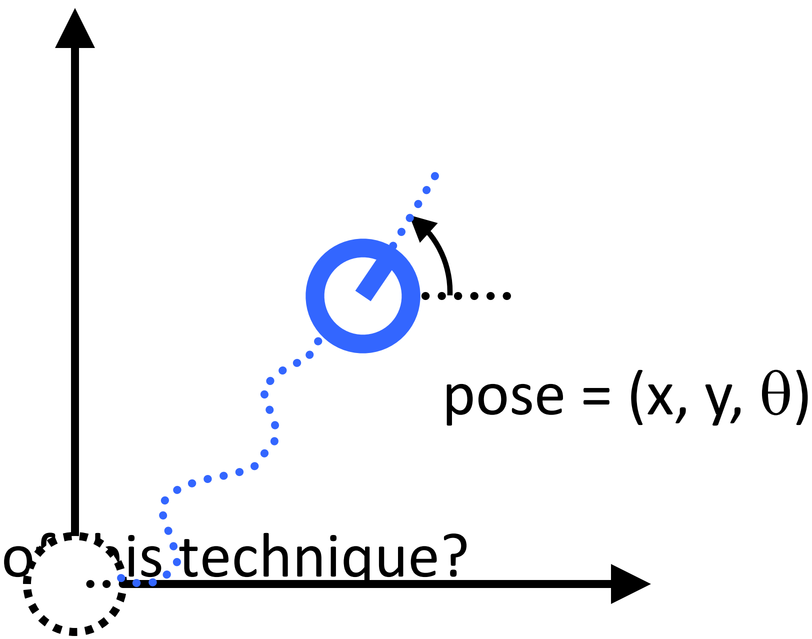
How can we compute this with encoders?

Pose Estimation

We assume the robot starts at (0, 0, 0)

We compute incremental changes to the pose using the encoders

This is called *dead reckoning*



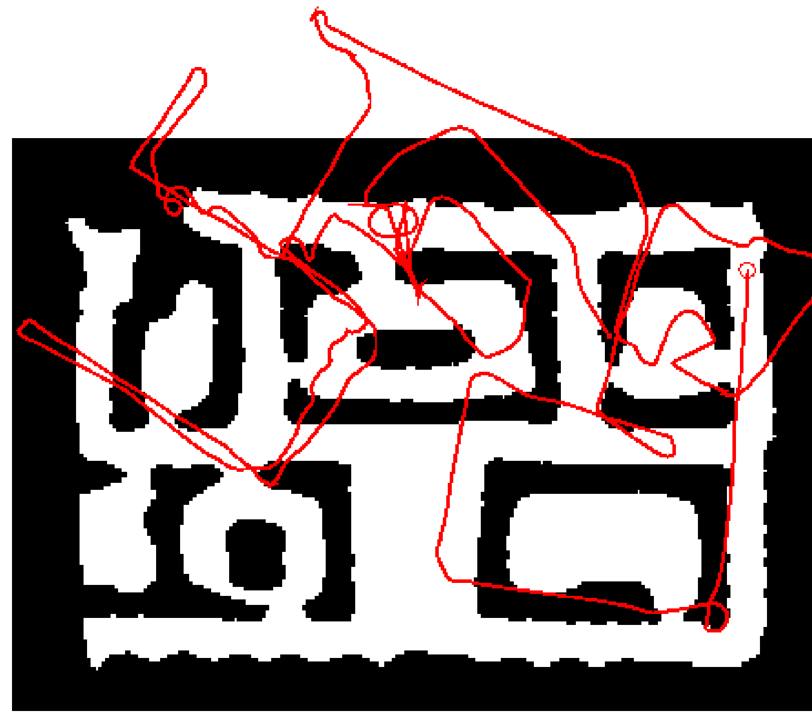
What is the drawback of this technique?

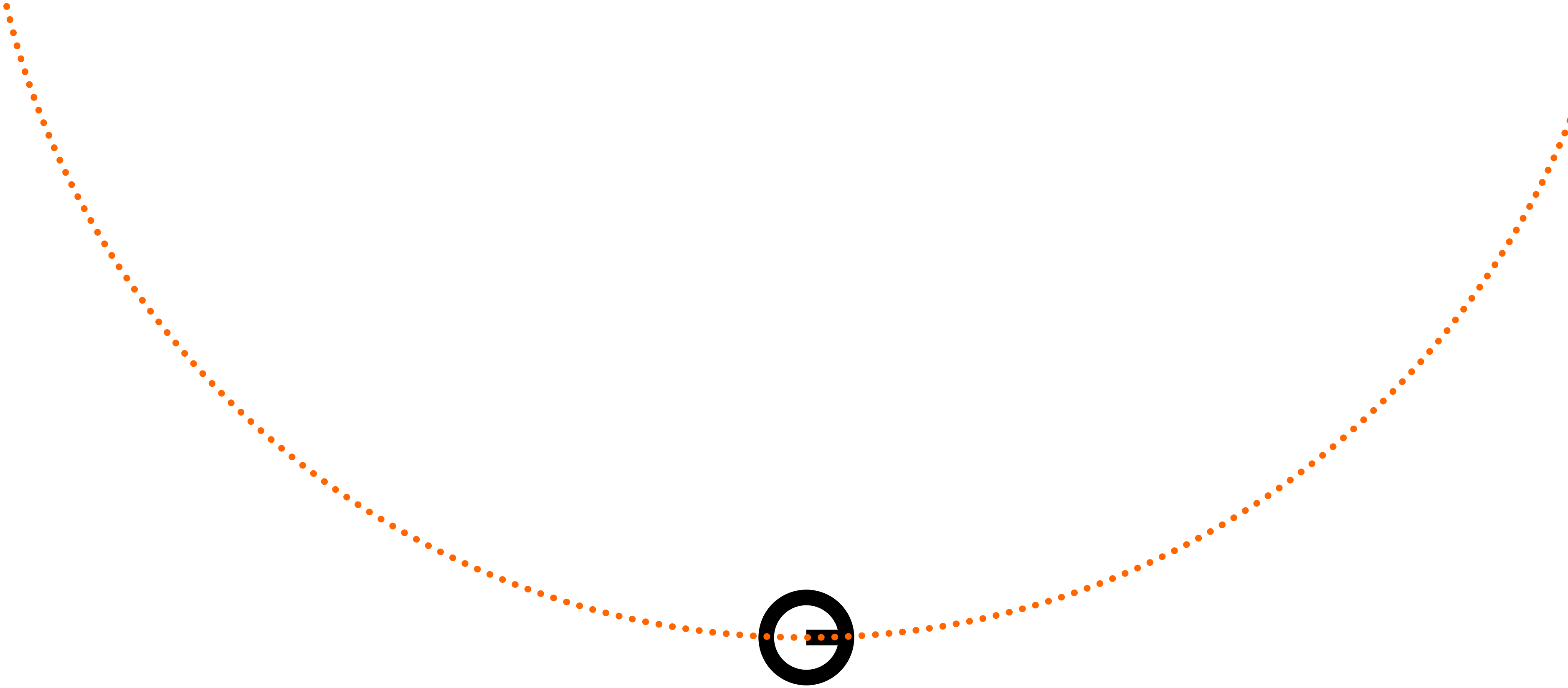
Wheel Slippage

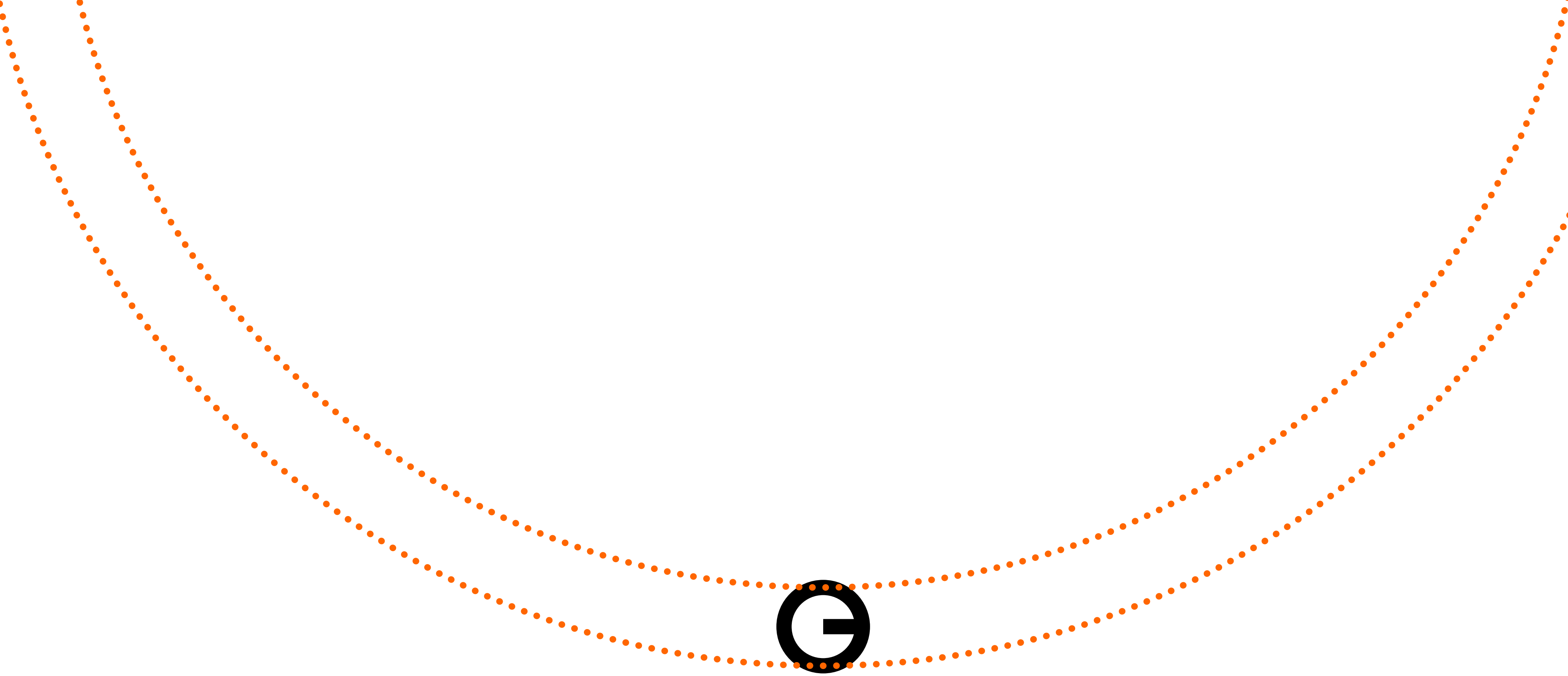
Wheels are always slipping, therefore Dead Reckoning is always accruing errors

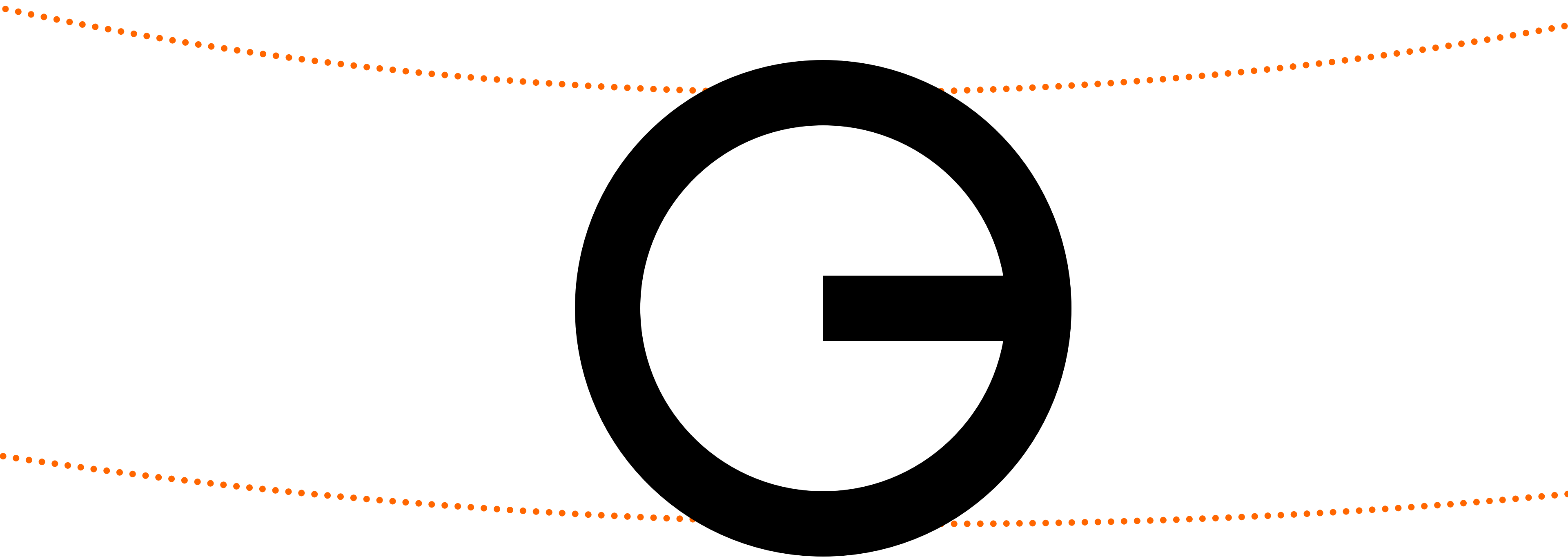
Even worse, the errors *integrate*, i.e. they increase little by little but have no bound. Soon the robot has **no idea** where it is.

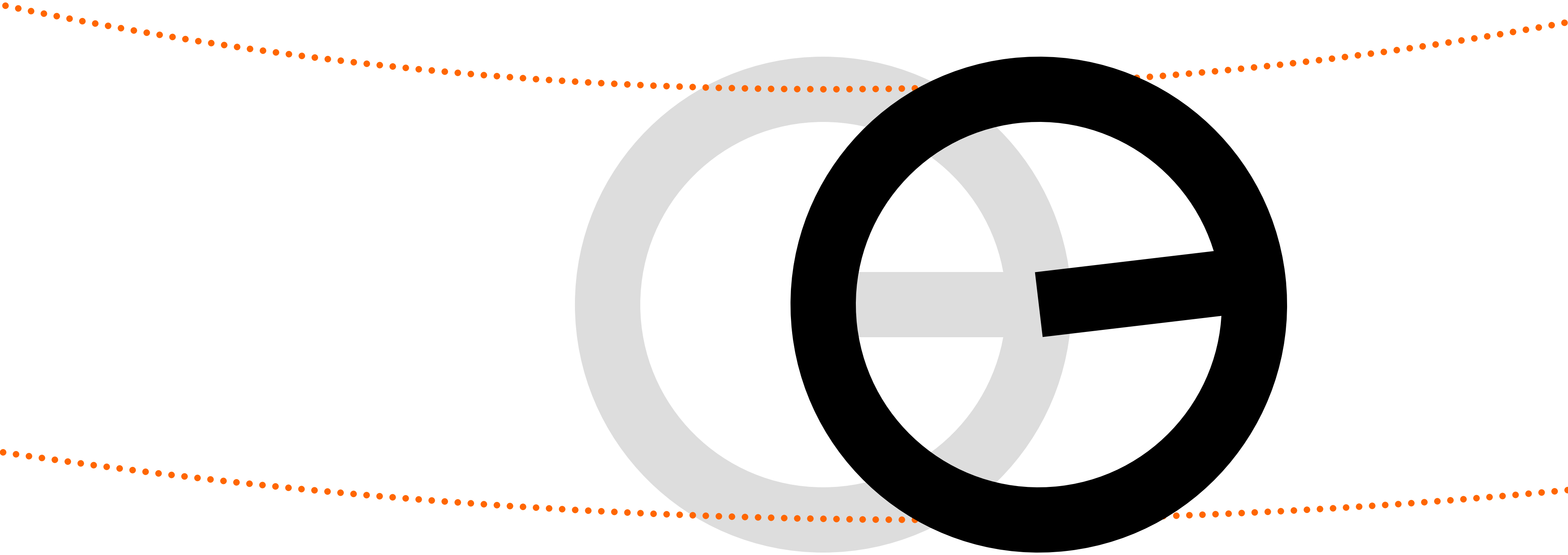
We can do better, right?

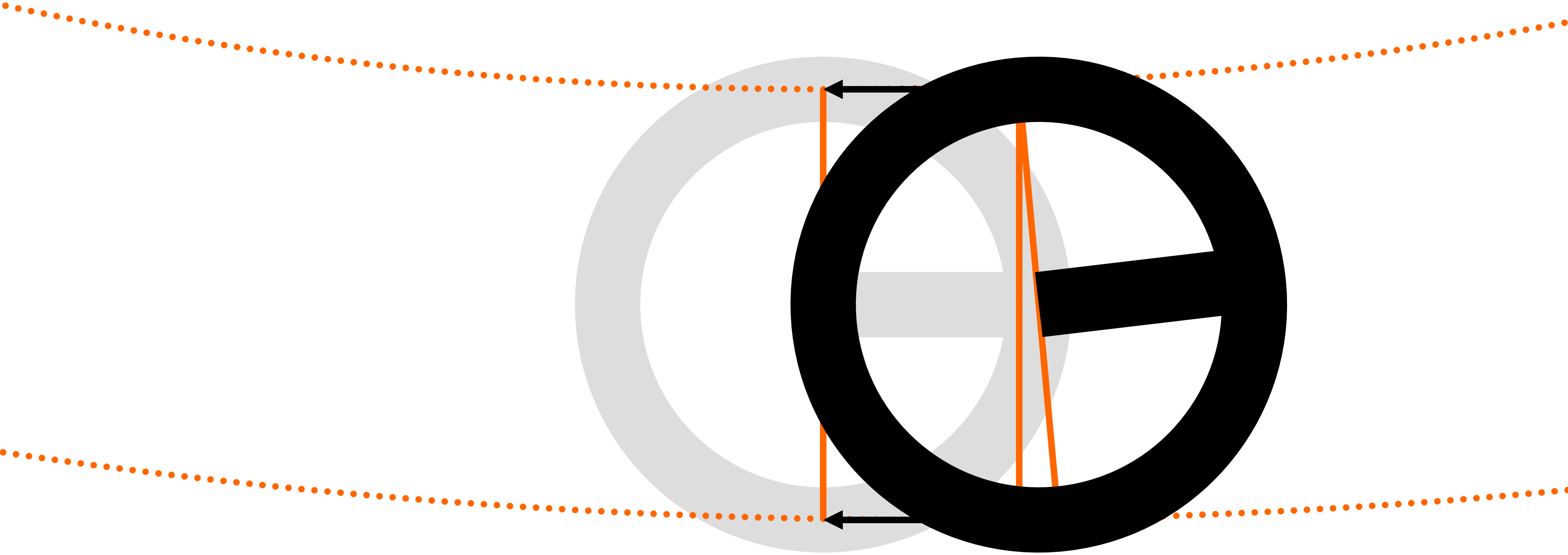




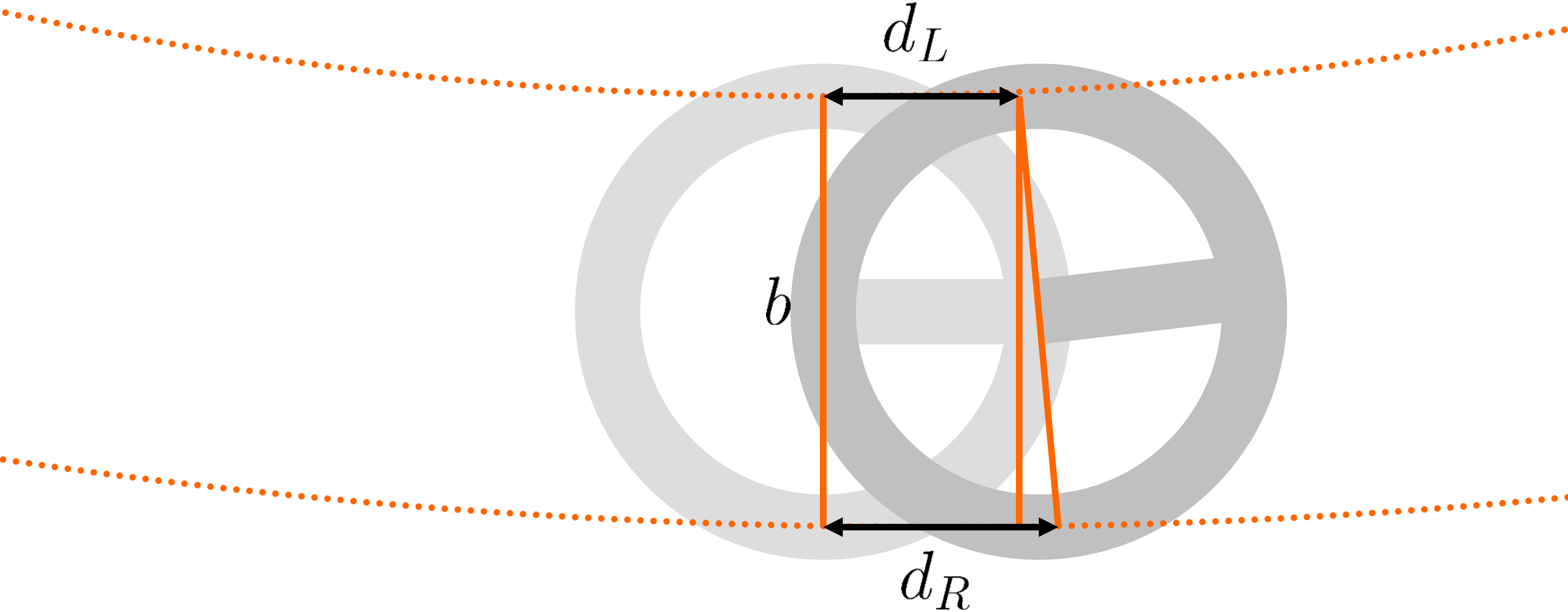




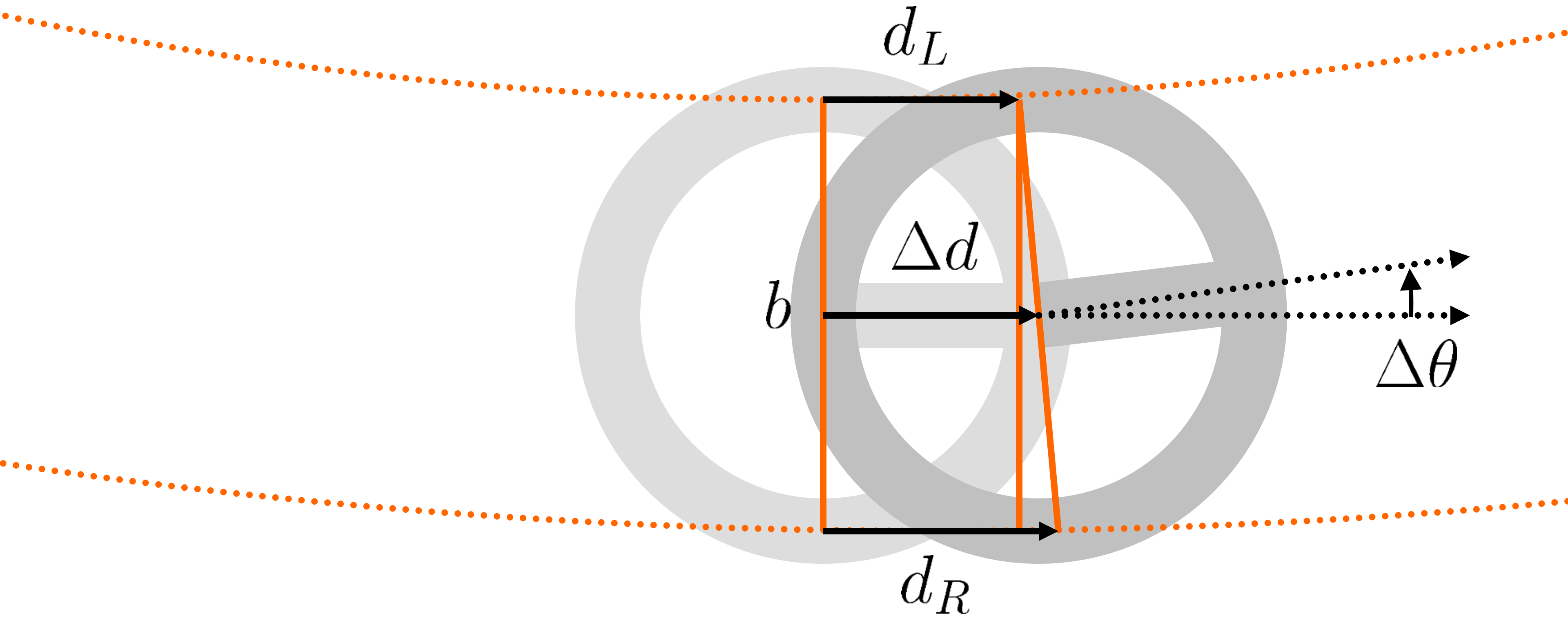




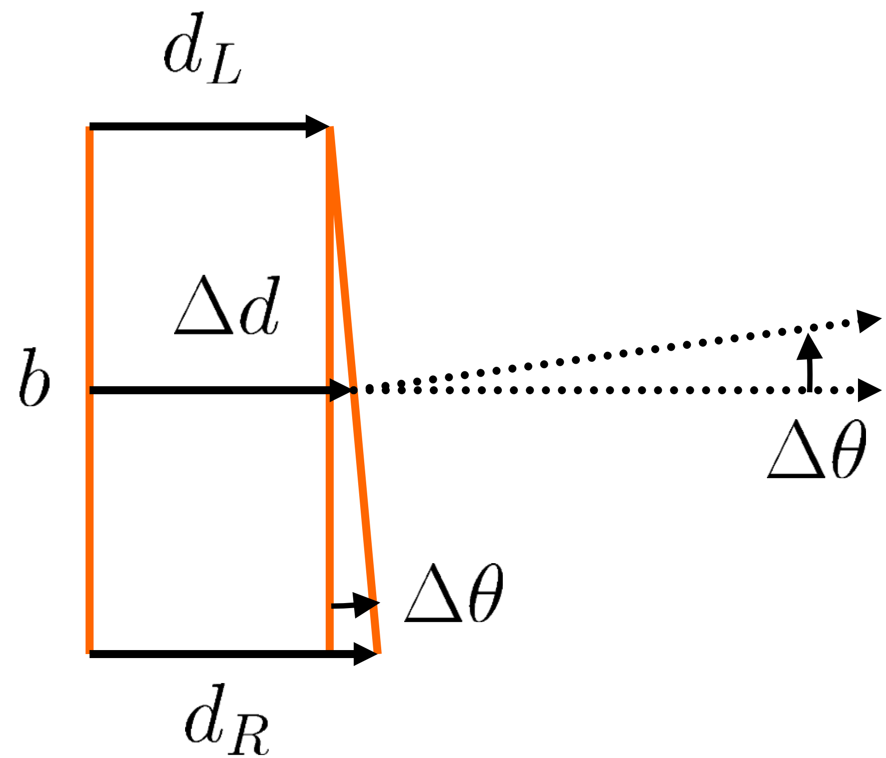
Parameters we know



Parameters we want to compute



One Update Step



[The rest is homework]

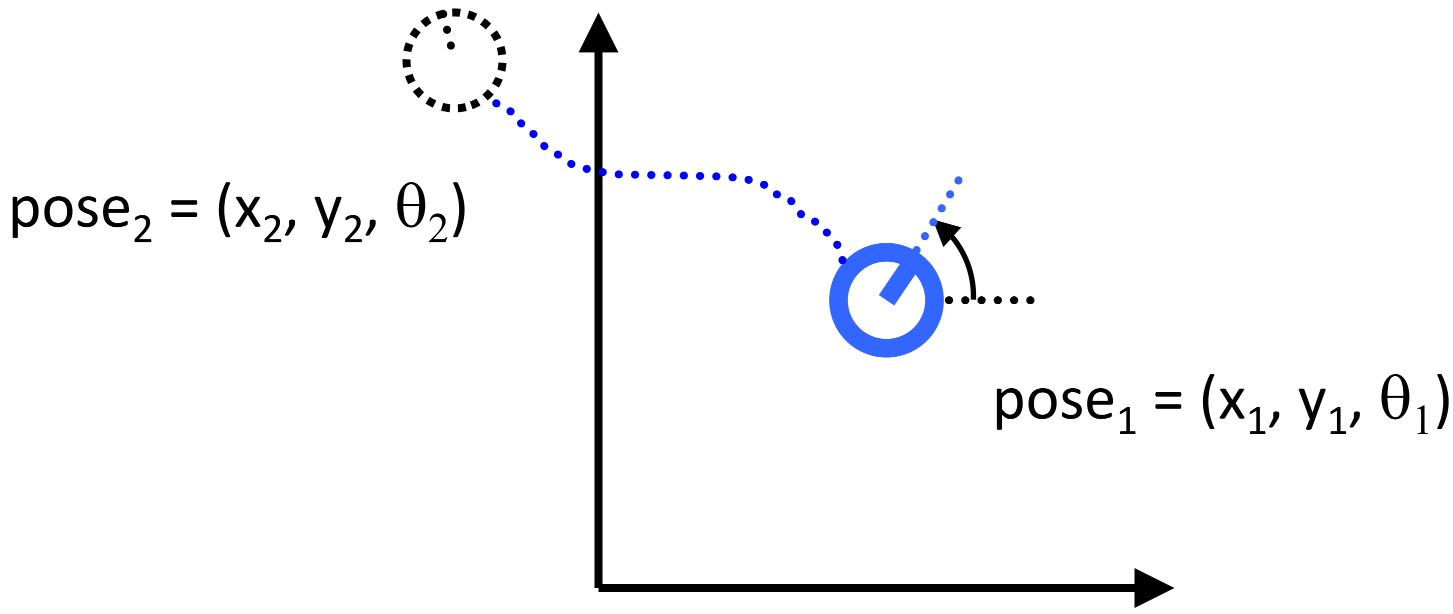
Using Odometry: Waypoint Navigation

Waypoint Navigation

We assume that we know our current pose

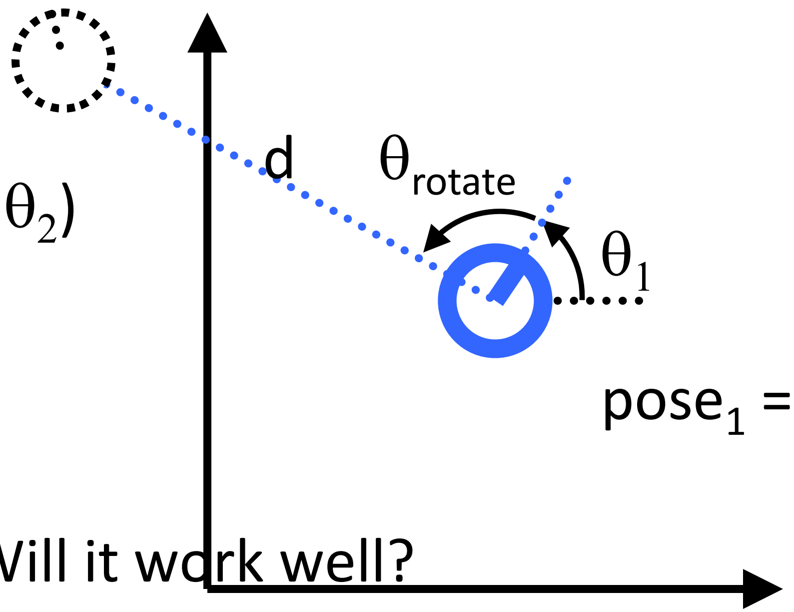
How can we get from one point to another?

We don't want to specify θ_2, just (x_2, y_2)



Waypoint Navigation

1. Compute θ_{rotate} and d
2. Rotate, then move a fixed distance

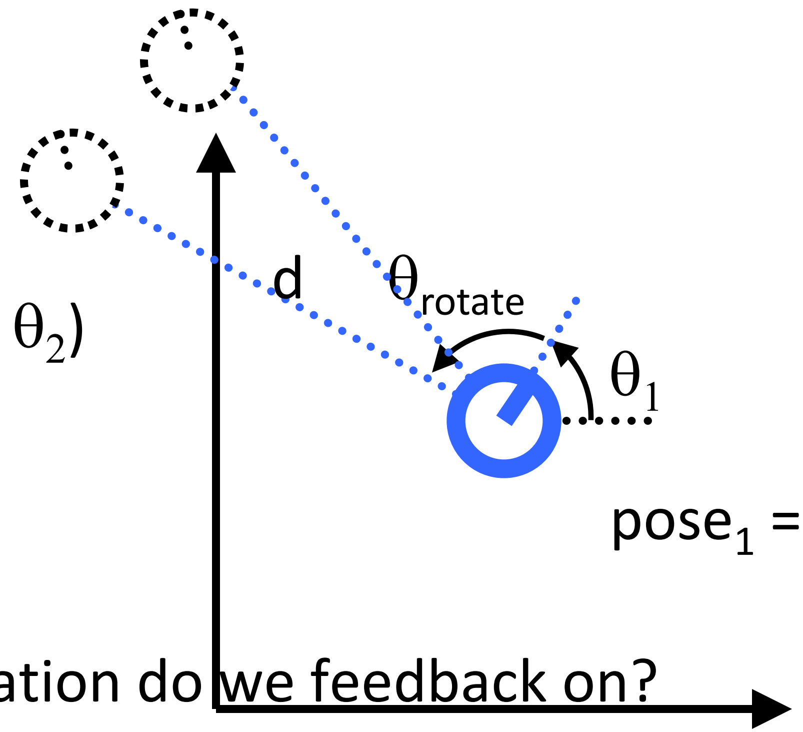


Will this work? Will it work well?

No

Small angular errors will be magnified over long d

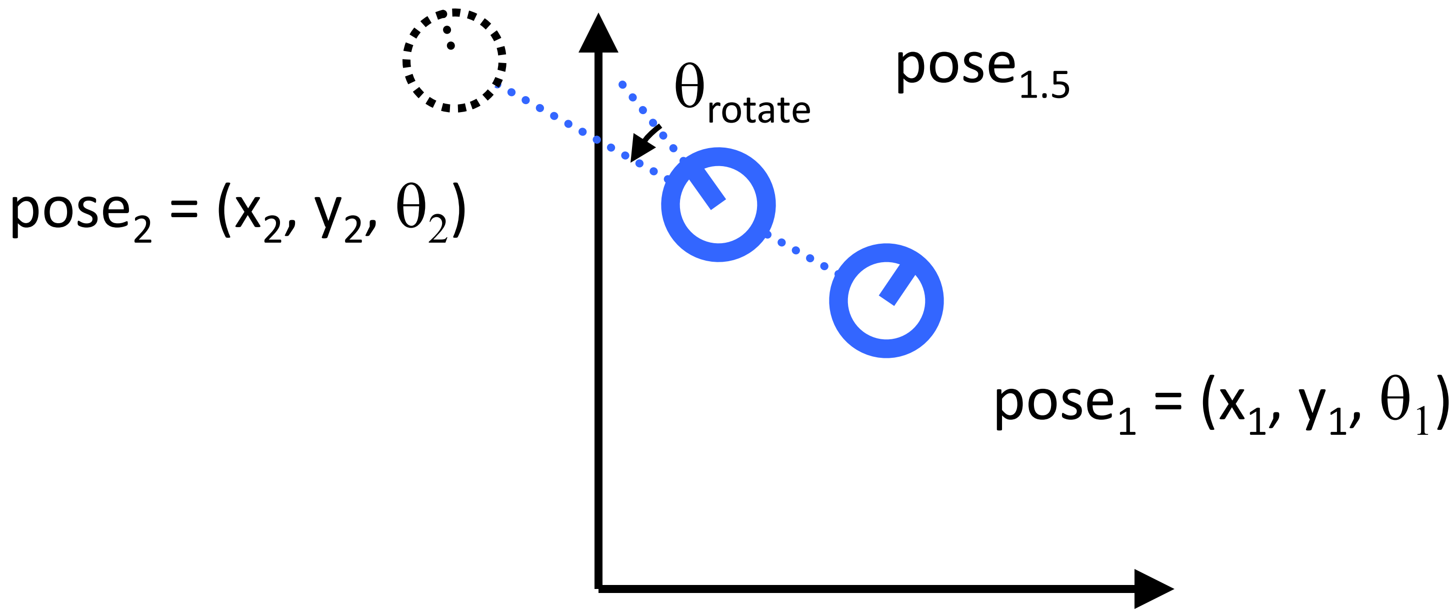
But we know how to deal with errors: a feedback controller



But what information do we feedback on?

Waypoint Navigation with a controller

1. While d > ε:
   2. Compute θ_{rotate} and d
   3. Rotate while translating



Topic 5:

The "r-one" robot

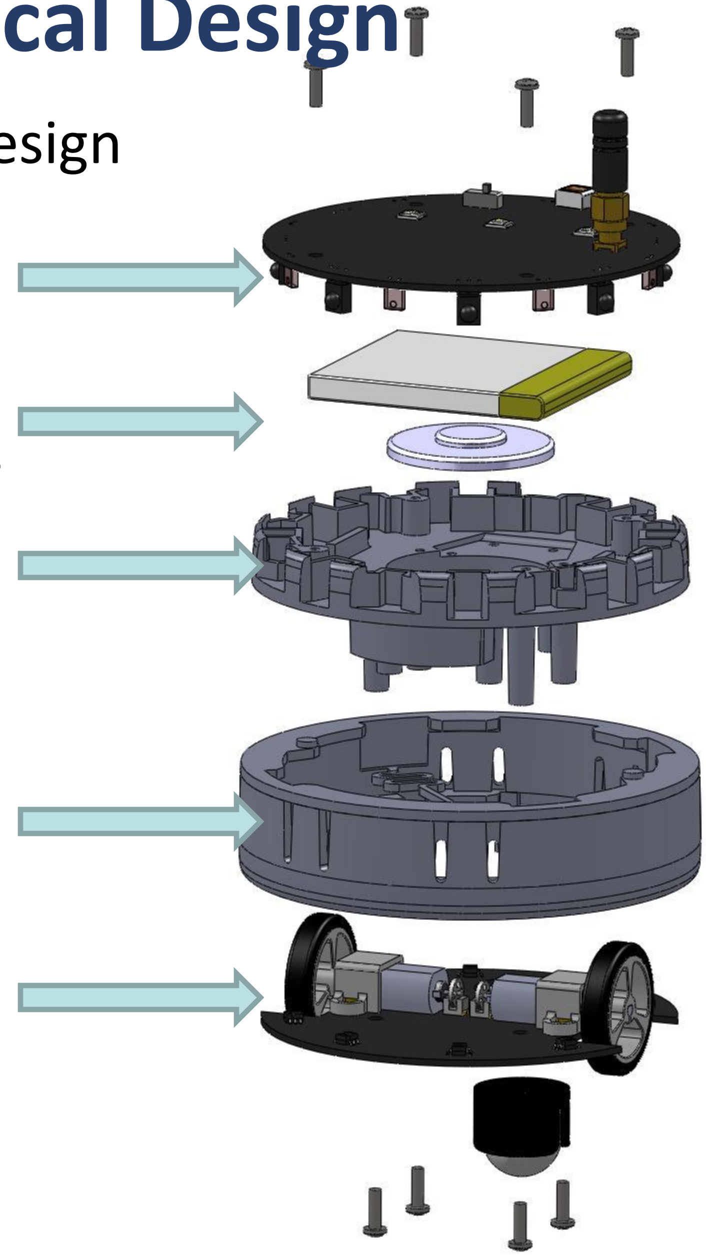
Sensors and Actuators on the r-one Robot


63
T0
GND
3.3V
IR7
J6
GND
3.3V
VBAT
IR0
FR
IR6
T1
FL
Expansion
T6
QEIOA
QEIOB
PWMOA
PWMOB
IR1
IR
Beacon
Micro SD
open source
hardware
T2
IR5
RST
BR
R
G
B
IR2
Heartbeat
System
Charge
Power
T5
BL
RX
TX
IR4
T3
IR3
T4

r-one Robot: Mechanical Design

Absurdly simple (read: elegant) design

- 2 circuit boards
- 3 plastic parts
- 7 off-the-shelf parts
- only 14 screws!



The r-one Sensors and Specifications

McLurkin [DARS 2010]

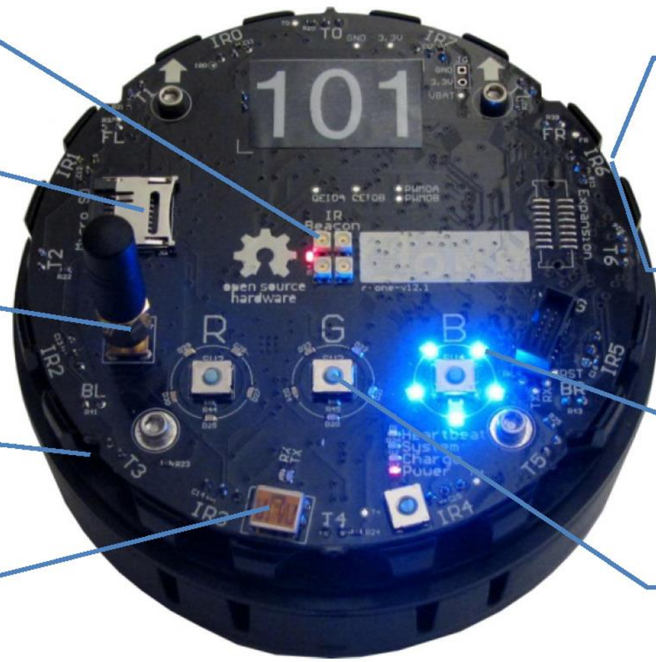
IR Beacon for Global Localization

Integrated SD card for Data Logging

2.4 GHz Radio
2 Mb/s Data Rate

4 Analog Light Sensors

USB Serial Port



8 IR Transmitters
8 IR Receivers for Communication and Localization

IR Obstacle Detection

15 LEDs in Red, Green, and Blue

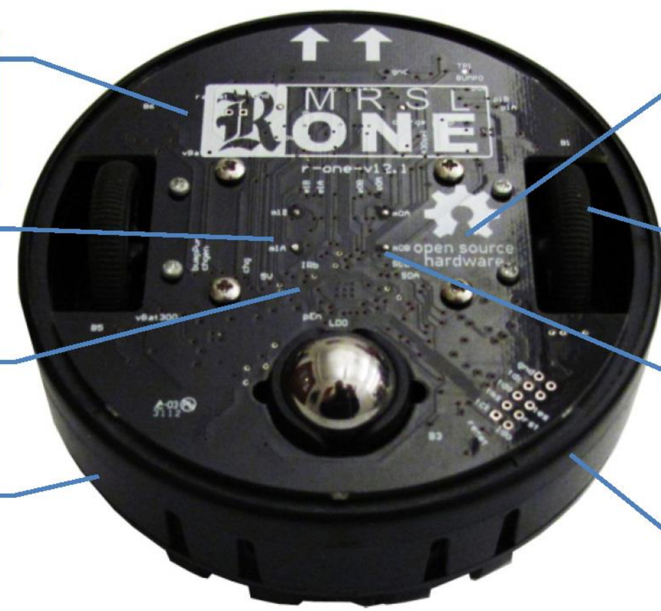
3 User Mode Buttons
1 On/Off Button with Airplane Mode

3 Chassis Components,
2 Circuit Boards,
12 Total Screws!

3-Axis Gyro and 3D Accelerometer

MIDI 2 Sound

Injection-Molded 8-bit Bump Skirt



Quadrature Encoders:
0.0625 mm/tick

Max Speed: 300 mm/s
5 Hour Running Time

50MHz ARM CPU and Texas Instruments Stellaris Processor

110 mm Diameter
230 grams total weight