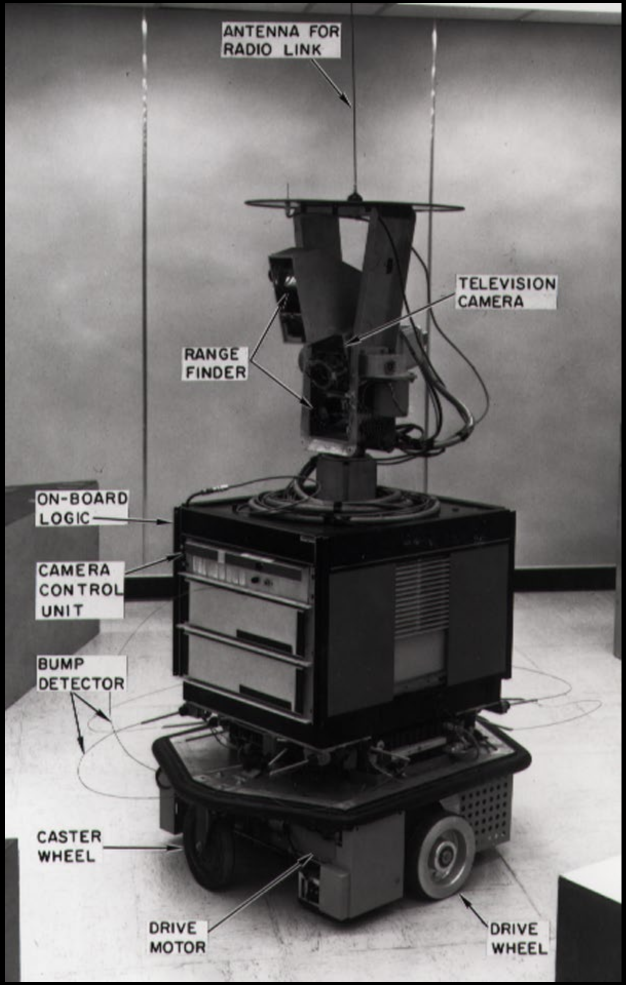
Prof. Kirstin Hagelskjær Petersen
kirstin@cornell.edu

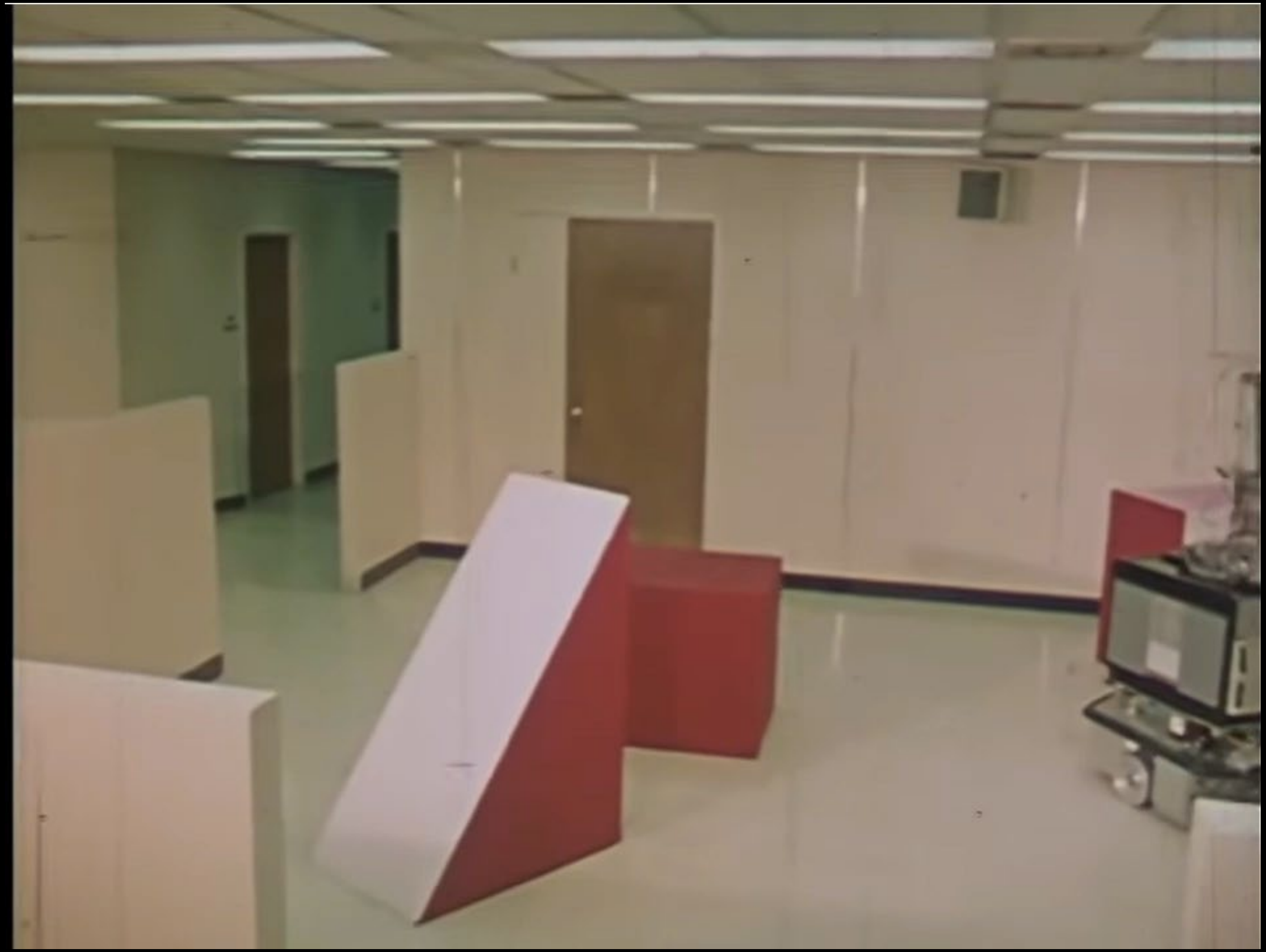
ECE 4160/5160
MAE 4910/5910

Fast Robots
Sensors

- Intro to sensors
- Distance sensors
- Odometry and dead reckoning
- Lab 2

History

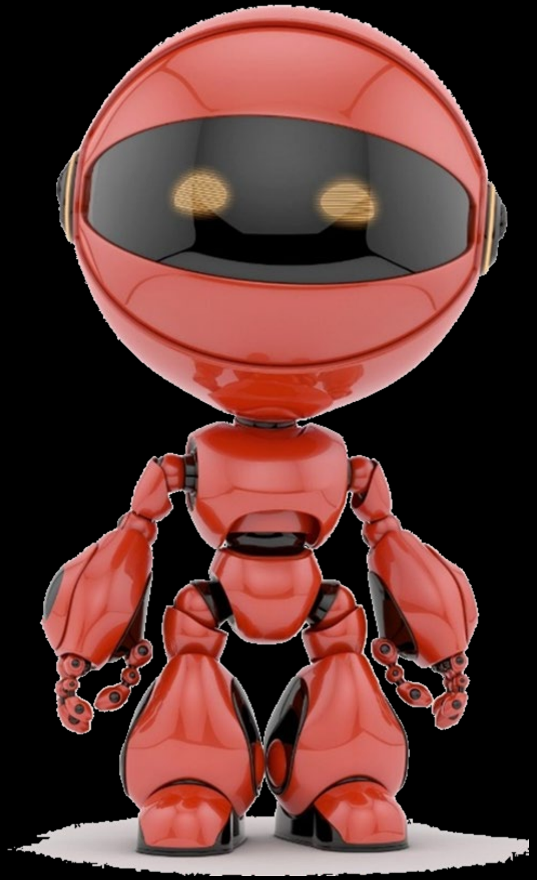




Shakey: Experiments in Robot Planning and Learning (1972), SRI

Sensor Classification

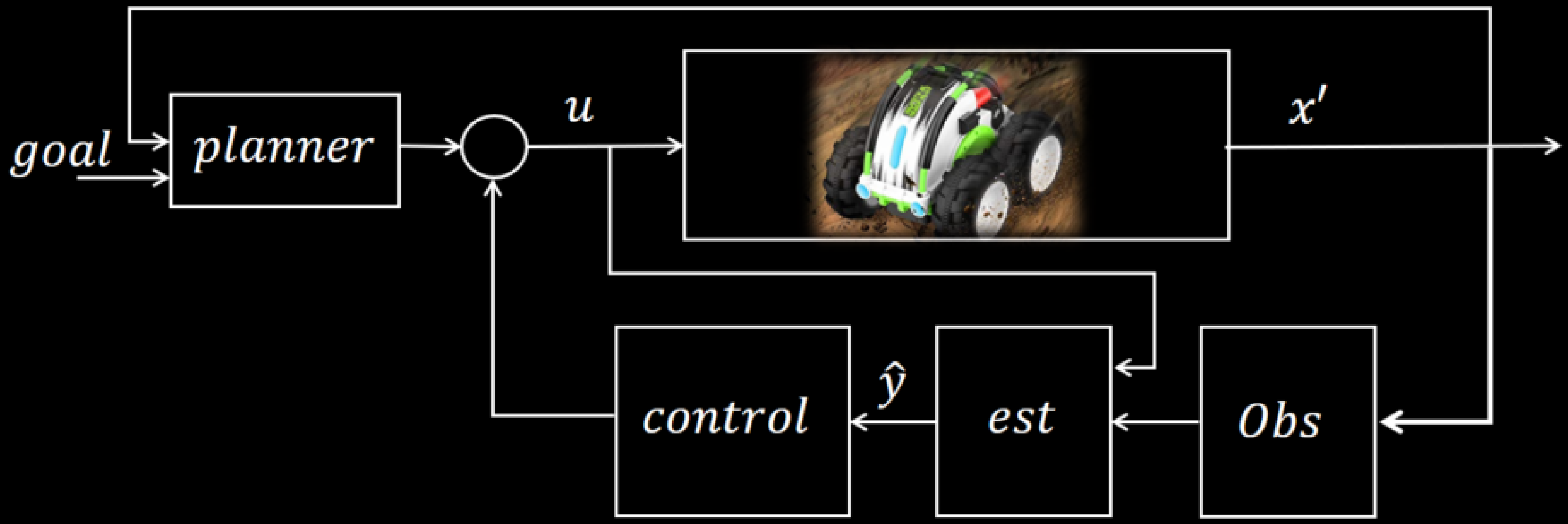
- **Proprioceptive**
  - Motor speed, wheel load, joint angles, battery voltage
- **Exteroceptive**
  - distance measurements, light intensity, sound amplitude
- **Passive Sensors**
  - Measure ambient environmental energy
  - E.g. temperature probes, microphones, light sensors
- **Active Sensors**
  - Senses reaction to emitted energy
  - E.g. wheel quadrature encoders, ultrasonic sensors, laser rangefinders



Classification

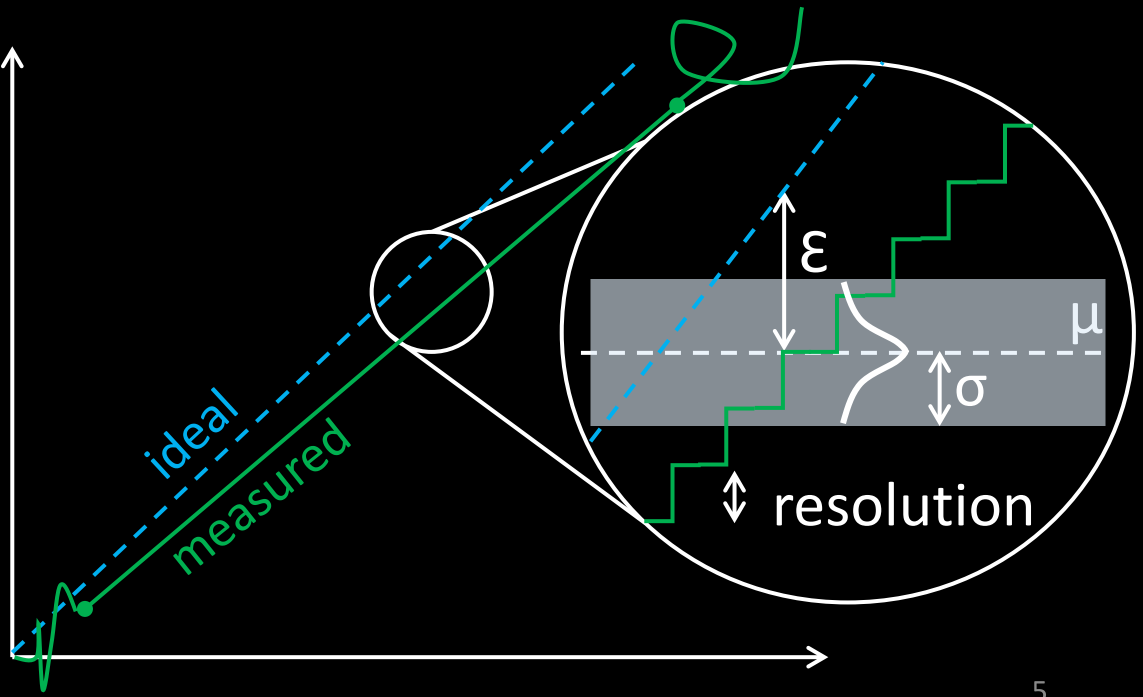
Type	Sensor	Prop/Exte	Passive/Active
Tactile (contact/closeness)	Contact switches, bumpers, Break beams, proximity Capacitive	Exteroceptive Exteroceptive Exteroceptive	Passive Active Both
Wheel/motor	Brush encoders Potentiometers Optical encoders Magnetic/inductive/capacitive encoders	Proprioceptive Proprioceptive Proprioceptive Proprioceptive	Passive Passive Active Active
Active ranging	Reflectivitiy sensors, ultrasonic, laser rangefinders, optical triangulation, etc.	Exteroceptive	Active
Heading	Compass Gyroscopes	Exteroceptive Proprioceptive	Passive Passive
Ground based beacons	GPS, RF, reflective beacons	Exteroceptive	Active
Motion/speed	Doppler radar, sound	Exteroceptive	Active
Vision	CCD/CMOS	Exteroceptive	Passive

Sensor Characteristics



Name some examples

- Dynamic Range
- Range
- Resolution
- Linearity
- Bandwidth / Sampling Frequency
- Sensitivity
- Cross-sensitivity
- Accuracy
- Precision
- Error
  - Systematic
  - Random
    - Power consumption
      - Size, price, etc…



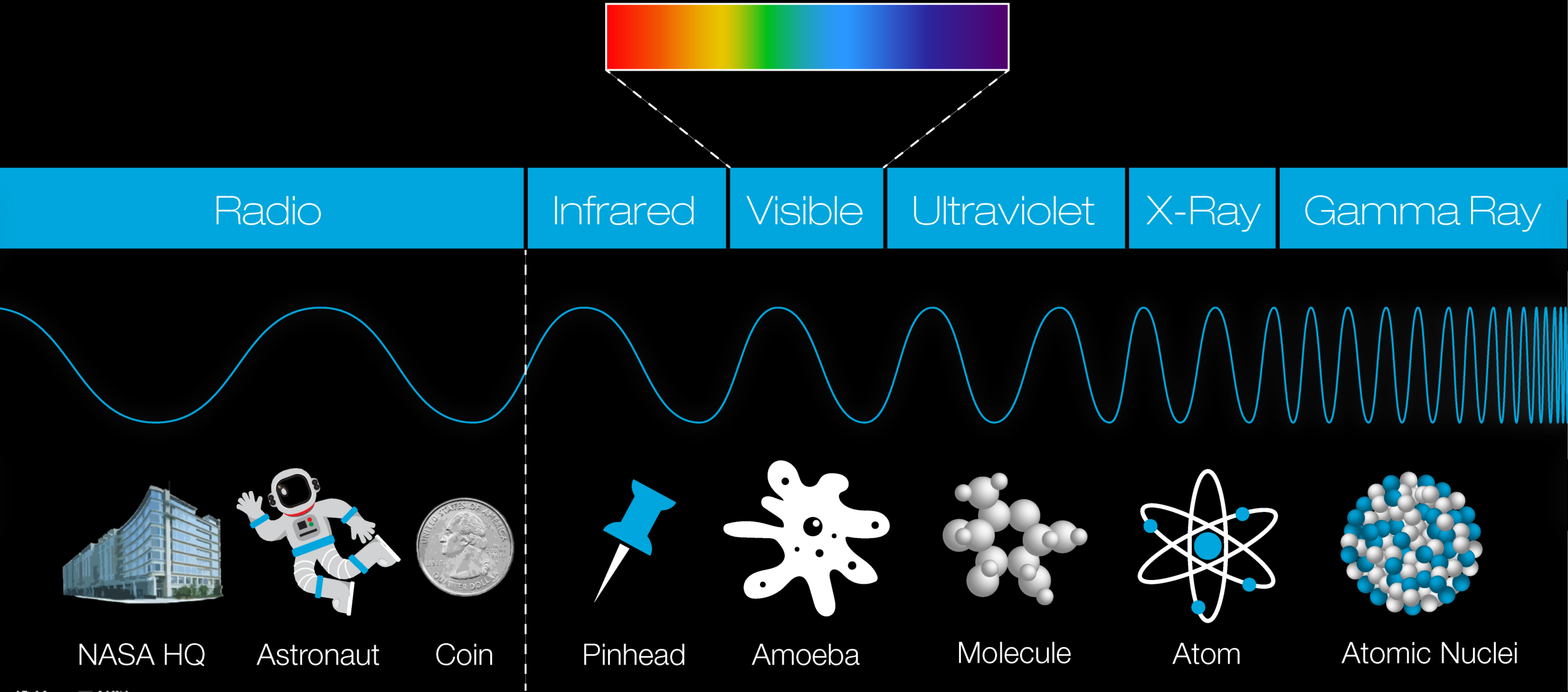
Prof. Kirstin Hagelskjær Petersen
kirstin@cornell.edu

Distance Sensors

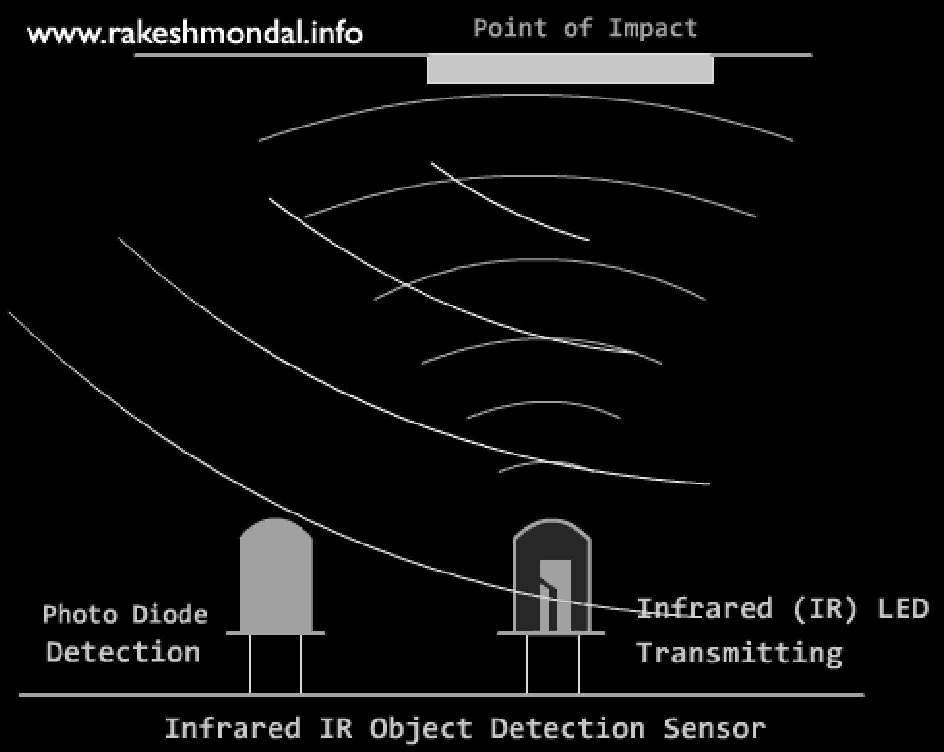
DIY-level Distance Sensors

Technology	Application	Pros	Cons
Amplitude-based IR	<10cm	• ~ 0.5 USD • Small form factor	• Depends on target reflectivity • Does not work in high ambient light
IR triangulation	<1m	• Insensitive to surface color/texture/ambient light	• ~ 10 USD • Does not work in high ambient light • Bulky (1.75" × 0.75" × 0.53") • Low sample rate (26Hz)
IR Time of Flight	0.1 - 4m	• High sample rate (4kHz) • Small form factor • Insensitive to surface color/texture/ambient light	• ~ 6.5 USD • Complicated processing • Low sampling frequency: 7-30Hz
Ultrasonic	0.2 – 10m	• Low cost • Insensitive to ambient light and surface color • Works in rain and fog	• ~4 USD • Complicated processing • Resolution trade off with max range • Output depends on surface/geometry/humidity • Bulky, sample time (tens of milliseconds) • Hard to achieve a narrow FoV

The Electromagnetic Spectrum



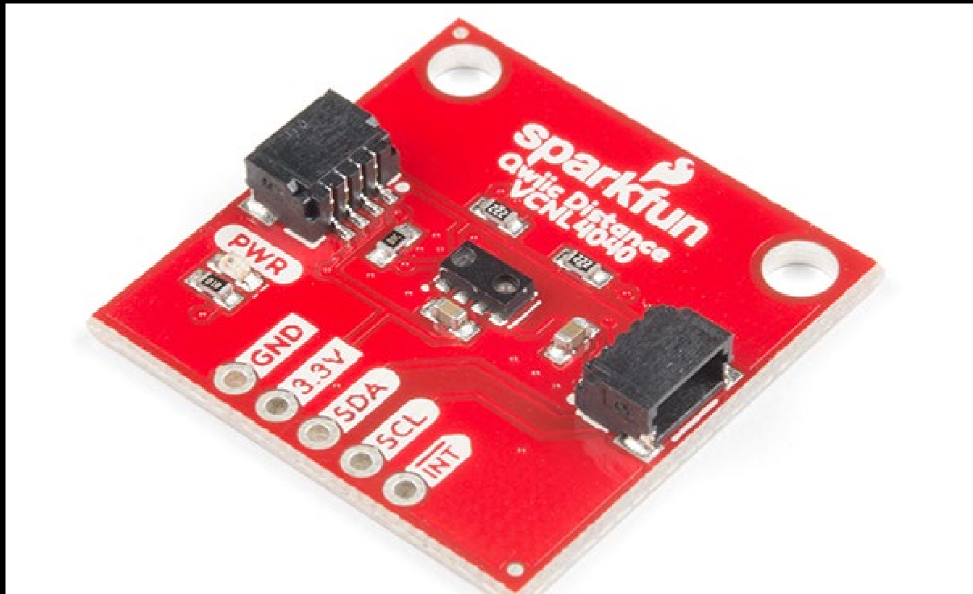
Amplitude –Based IR Distance Sensors



- Very cheap
- Very simple circuitry
- Works reasonably well for
  - Object detection
  - Break beam sensors
  - Classifying greyscale intensity at a fixed distance
  - Short-range distance sensor
- Range <0.5m
- Sensitive to surface color, texture, and ambient light

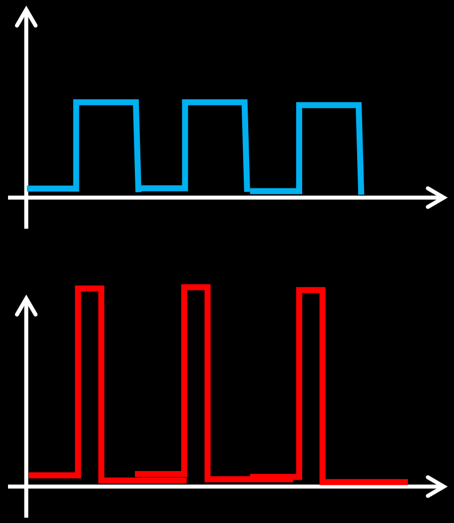
VCNL4040

- $3.34
- Range 20cm
- Ambient light sensor
- Programmable DC



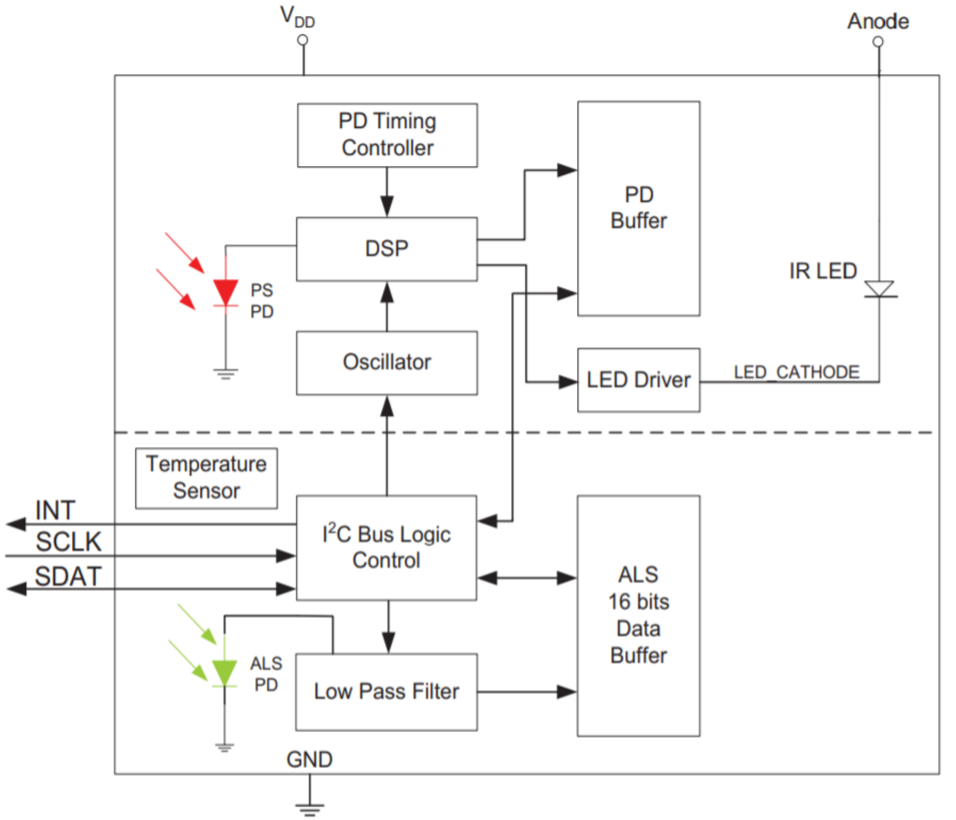
Amplitude –Based IR Distance Sensors

- Very cheap
- Very simple circuitry
- Works reasonably well for
  - Object detection
  - Break beam sensors
  - Classifying greyscale intensity at a fixed distance
  - Short-range distance sensor
- Sensitive to surface color, texture, and ambient light

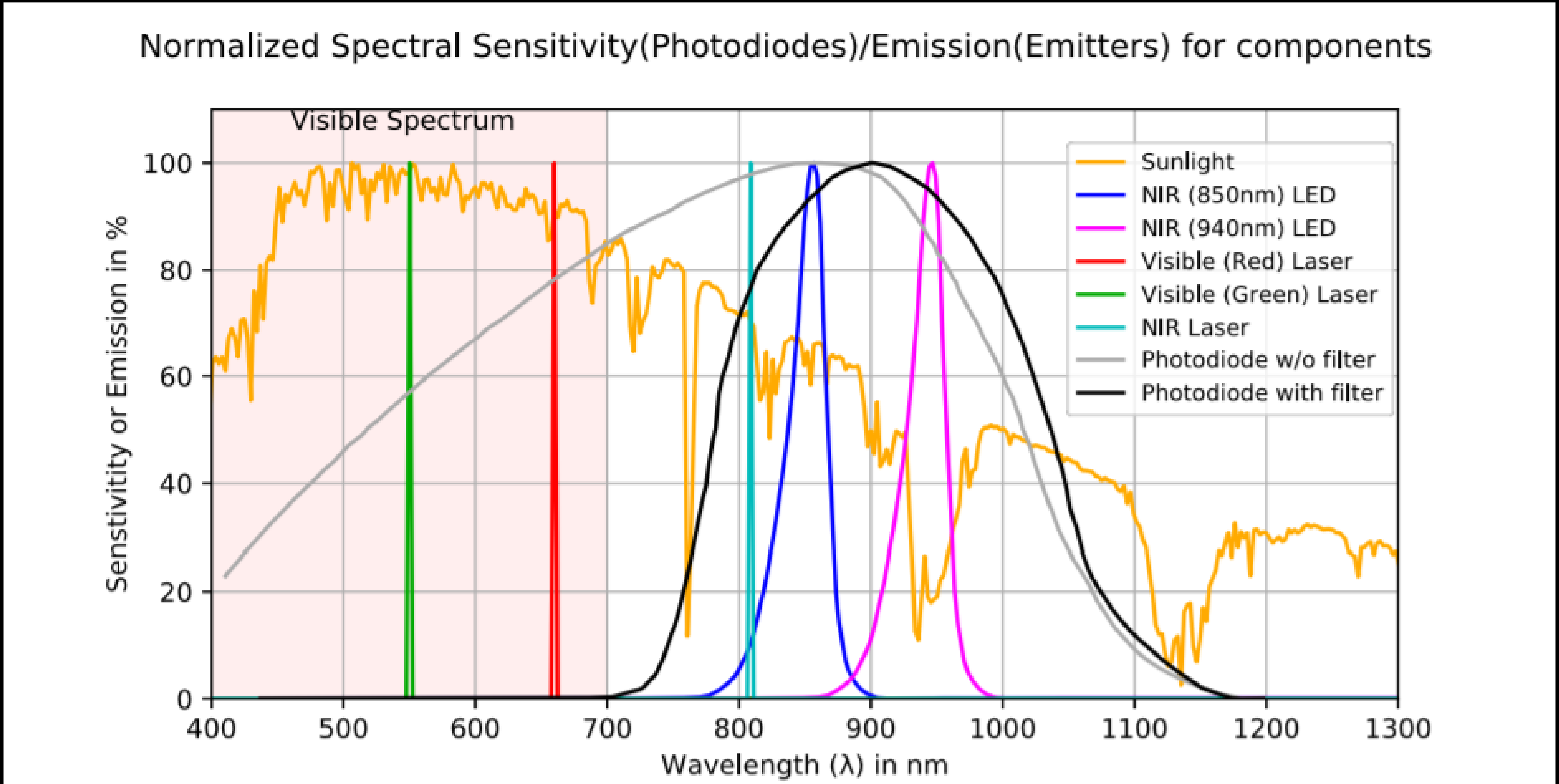


VCNL4040

- $3.34
- Range 20cm
- Ambient light sensor
- Programmable DC

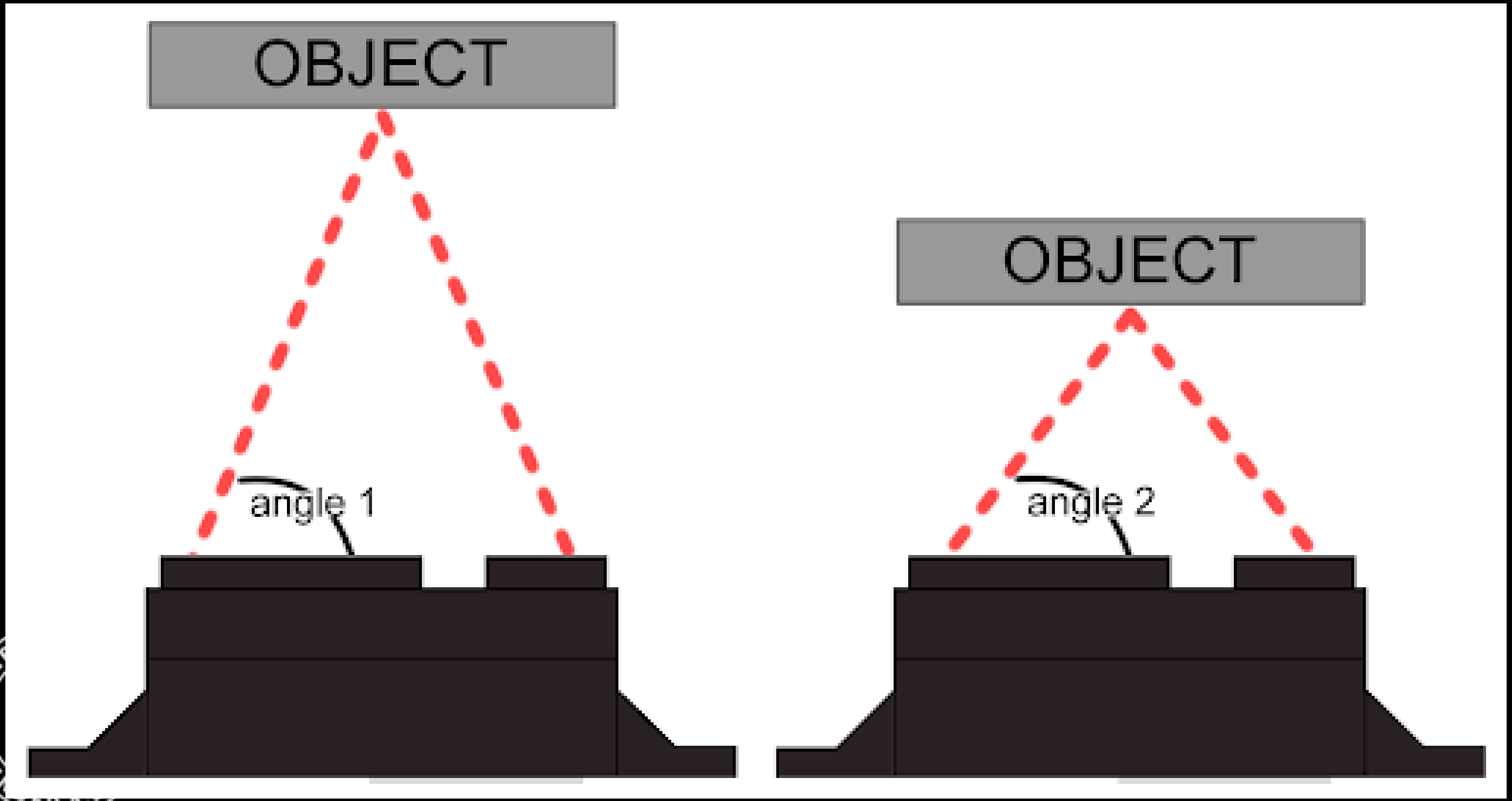


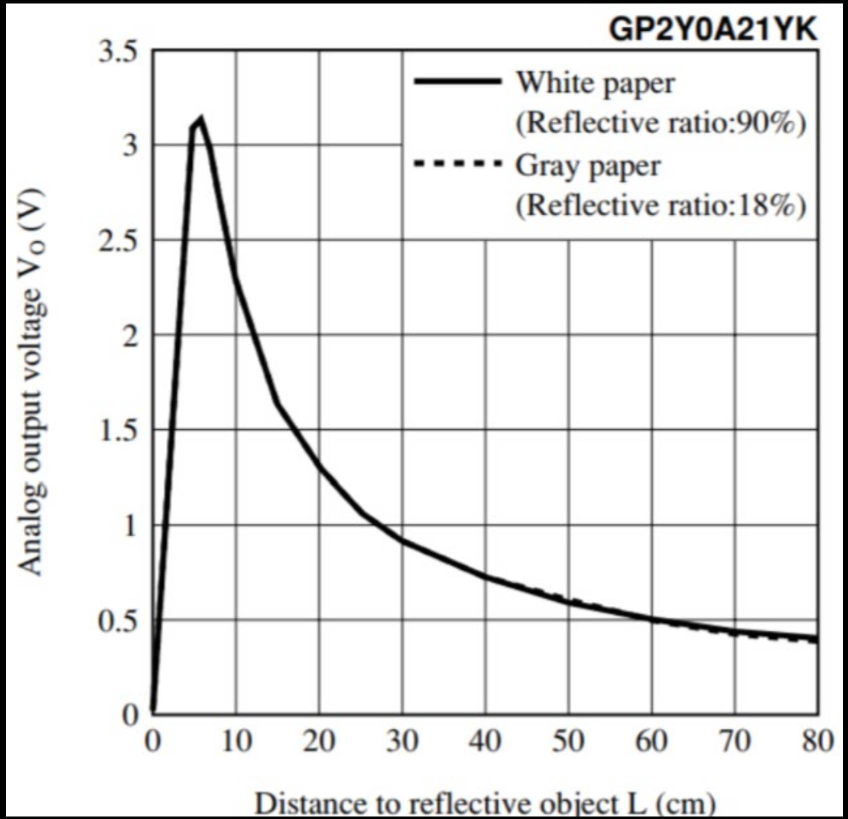
Amplitude –Based IR Distance Sensors



Triangulation–Based IR Distance Sensors

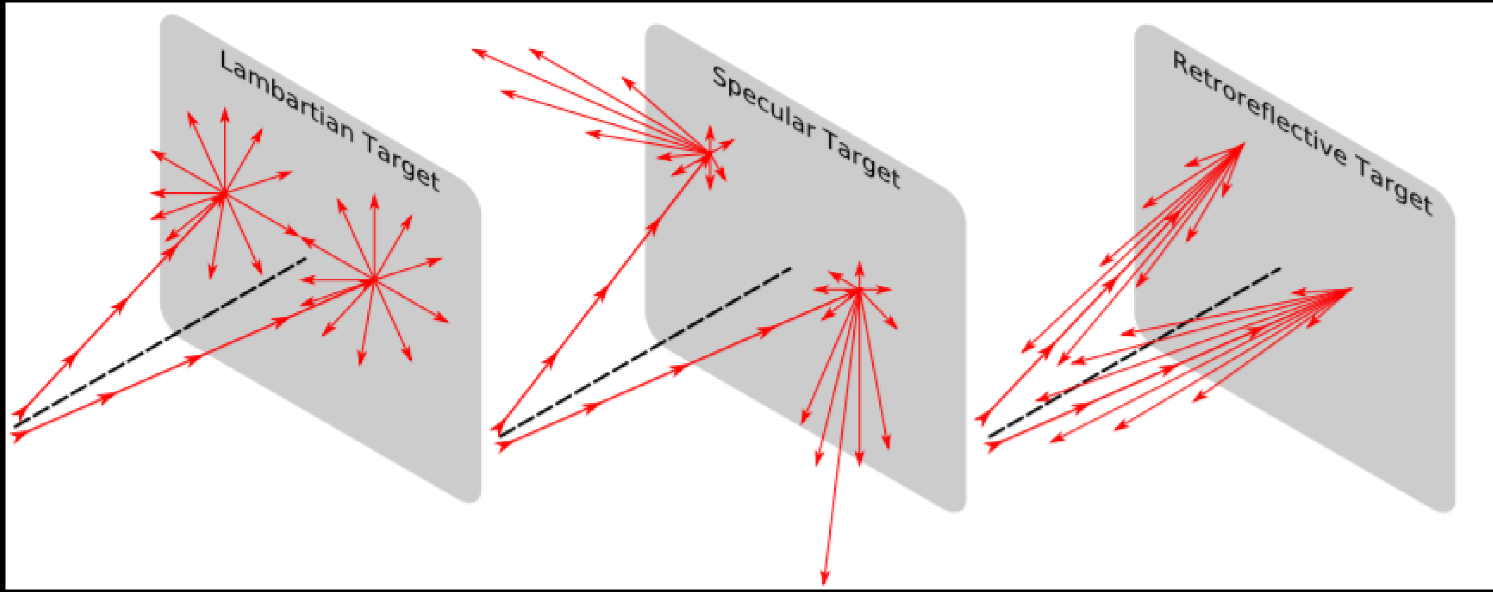
- Very simple circuitry
- Less sensitive to color, texture, ambient light
- Medium range (0.05 - 1 m)
- Cost 5-25 USD

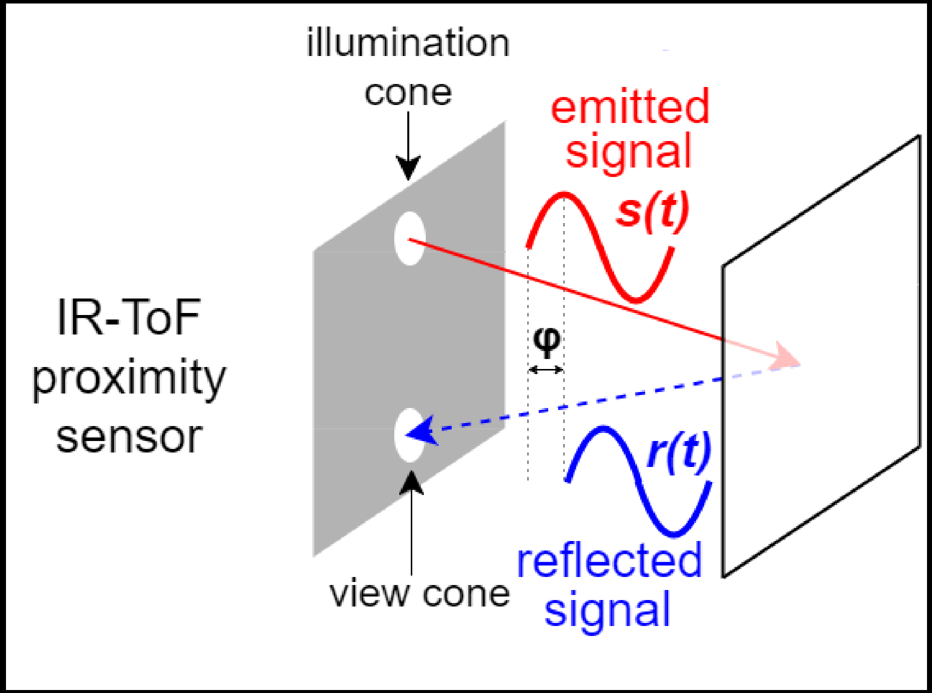




Time of Flight IR Sensor

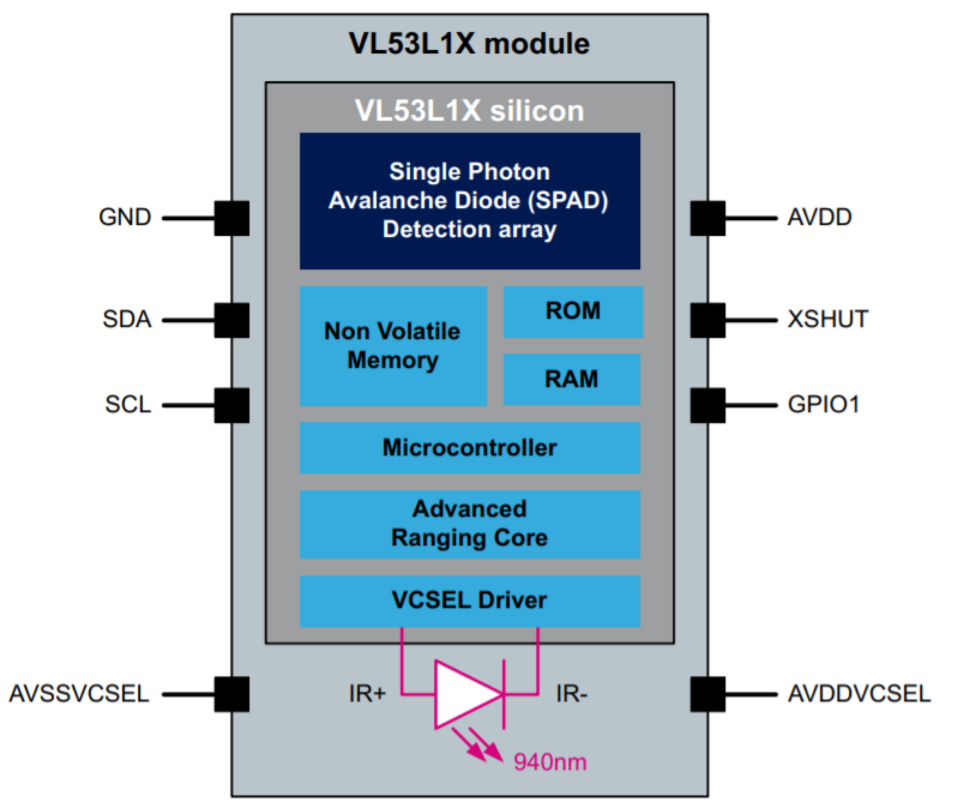
- Emit a pulse modulated signal, record time *t* until return!
  - r = t*c/2
  - c = speed of light = 299,792,458 m/s
- Mostly insensitive to texture, color, ambient light

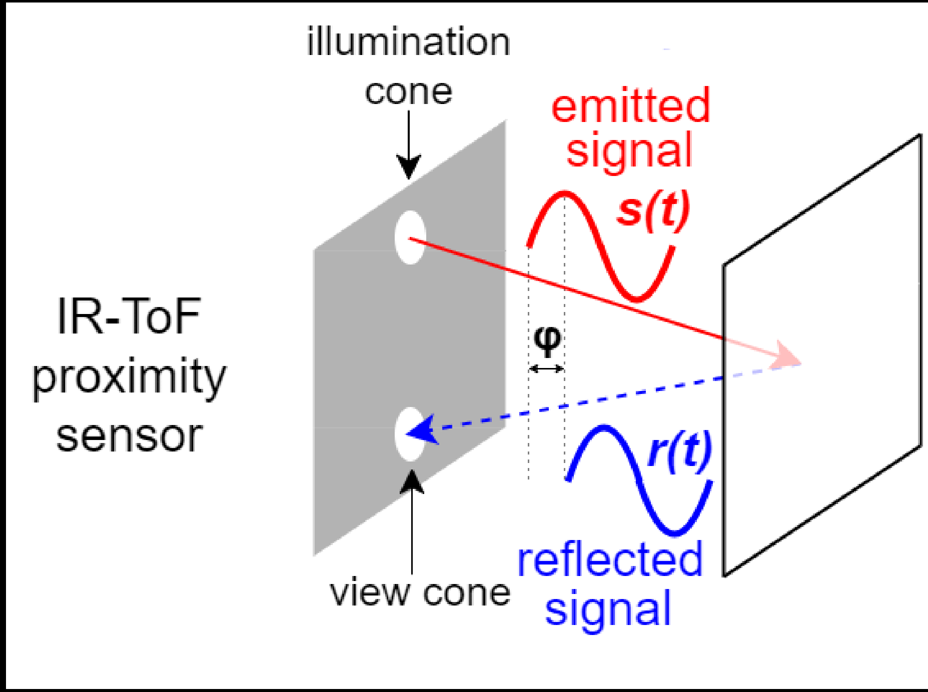




Time of Flight IR Sensor

- Emit a pulse modulated signal, record time *t* until return!
  - r = t*c/2
  - c = speed of light = 299,792,458 m/s
- Mostly insensitive to texture, color, ambient light
- Outputs (Distance in mm, return signal rate, ambient signal rate, range status)



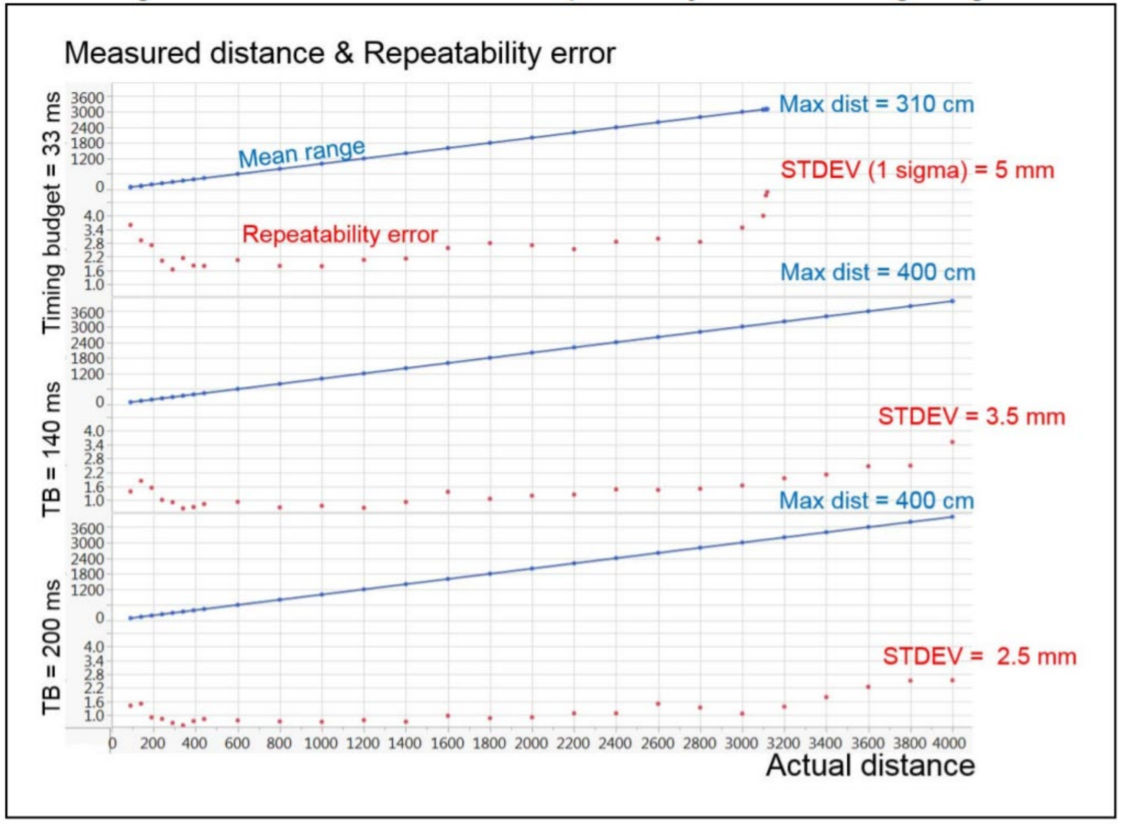


Time of Flight IR Sensor

- Emit a pulse modulated signal, record time *t* until return!
  - r = t*c/2
  - c = speed of light = 299,792,458 m/s
- Mostly insensitive to texture, color, ambient light
- Outputs (Distance in mm, return signal rate, ambient signal rate, range status)
- Programmable FOV
- Distance mode (~1, 2, 4m)
- Timing budget
  - 20ms: short distance mode (0.05 - 1.3m)
  - 33ms: all distance modes (0.05 - 3.6m)
  - 140ms: improve reliability errors

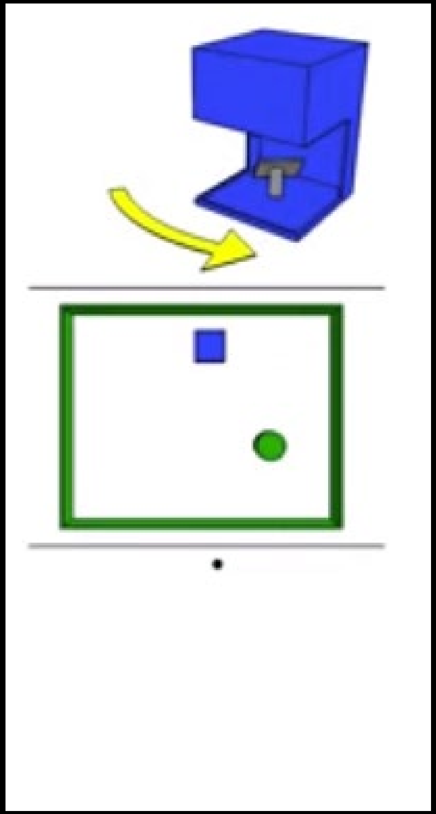
- Newest developments
  - ToF Imager (64 pixels)

Figure 6. Maximum distance and repeatability error vs. timing budget

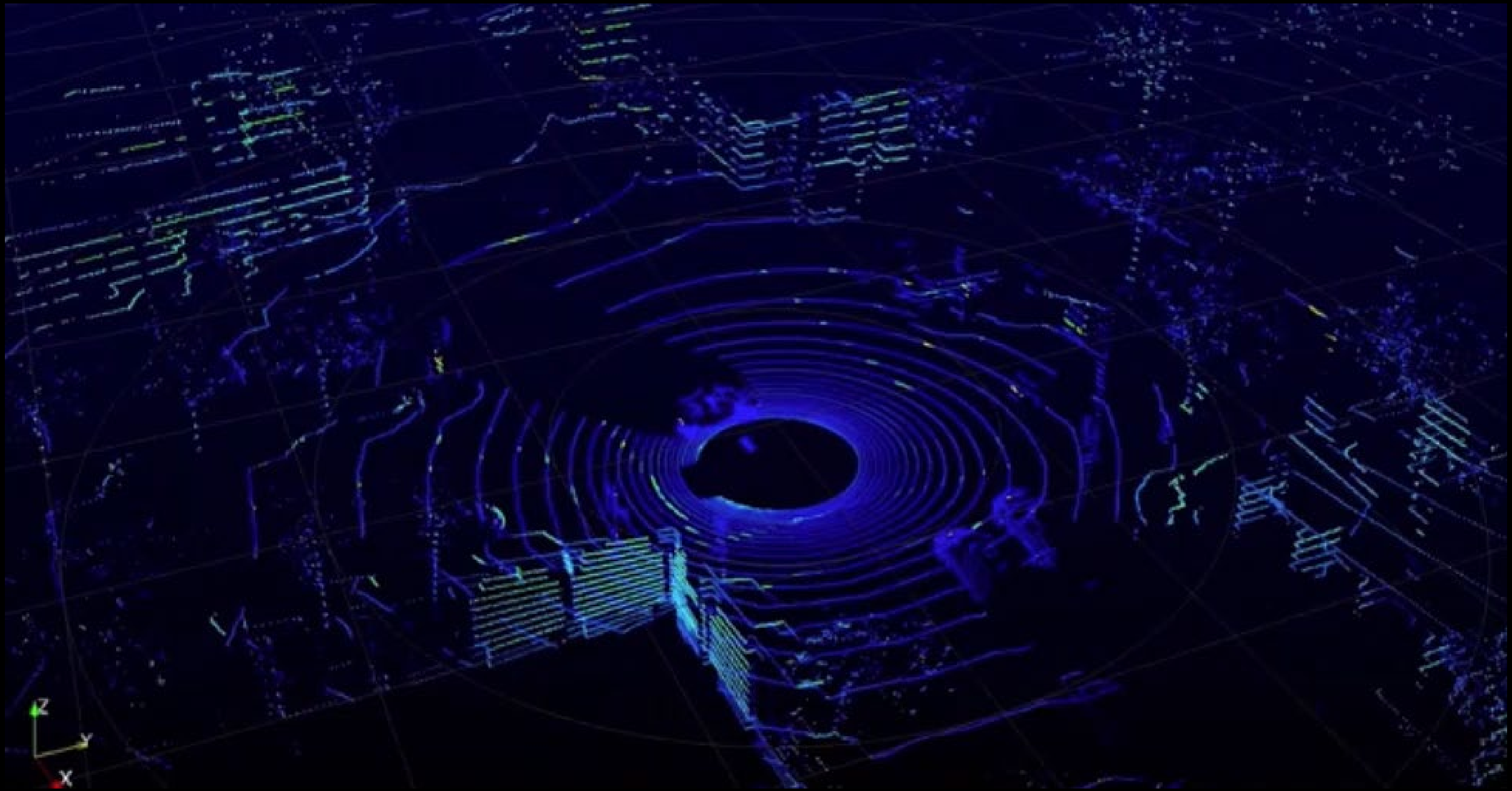


Light Detection and Ranging Sensors

- Most common sensors on autonomous cars and robots
- Single points, line scans, full 3D
- $$$

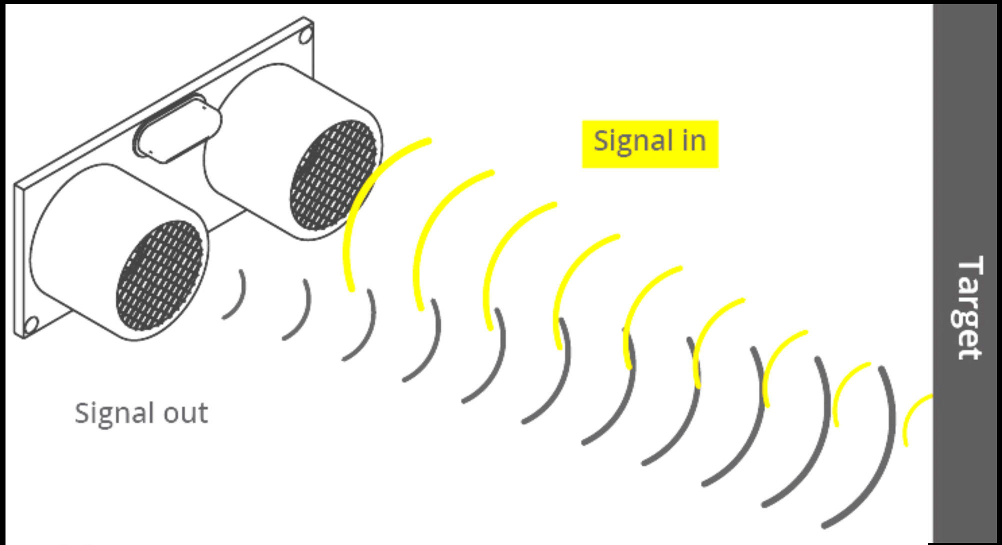


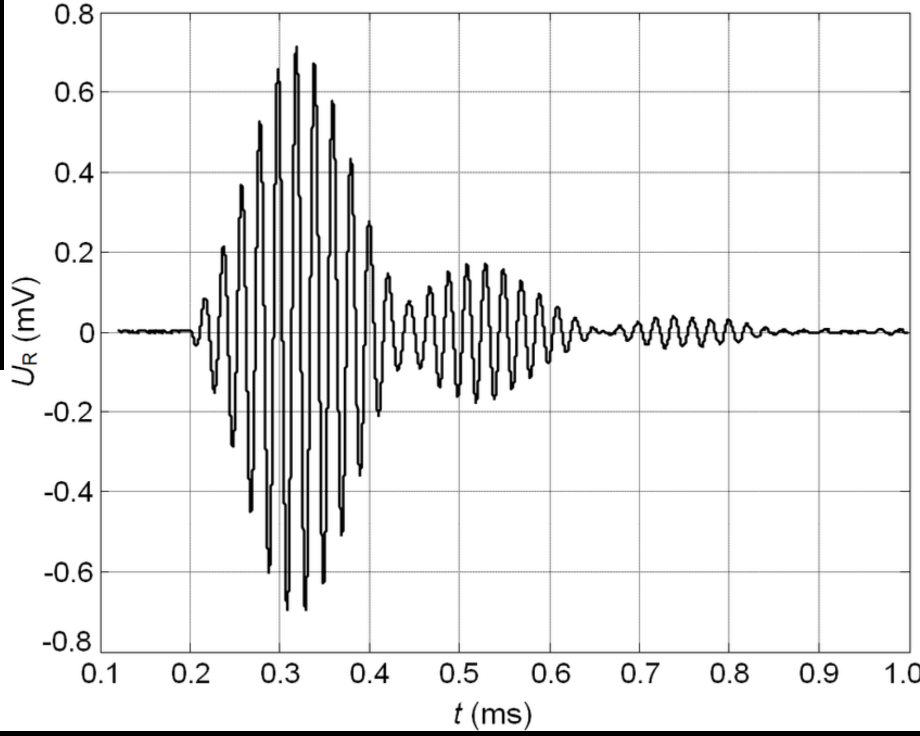
What does the color represent?

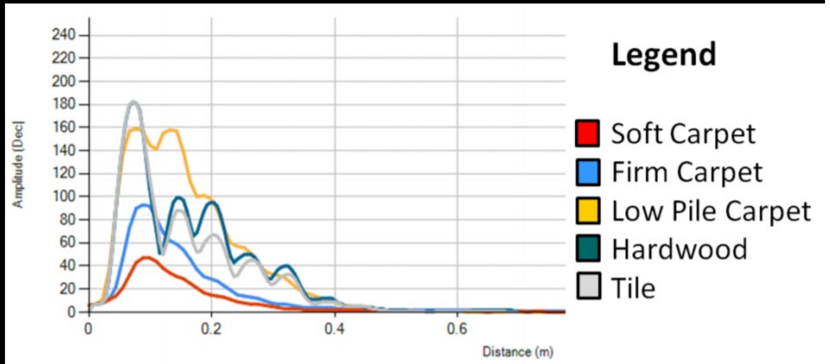


Ultrasound (Time of Flight) Distance Sensors

- Measure the reflections of an emitted sound wave
  - $r = t * c_{sound}/2$
  - $c_{sound} = 343$ m/s
- Frequency versus resolution and range
  - 58kHz: cm resolution, range < 11m
  - 300kHz: mm resolution, range < 0.3m
- Cost is low (Sparkfun module: 4-12 USD)
- Insensitive to color, texture, glass, fog, dust, etc.
- Sensitive to humidity, temperature, audible noise, and geometry

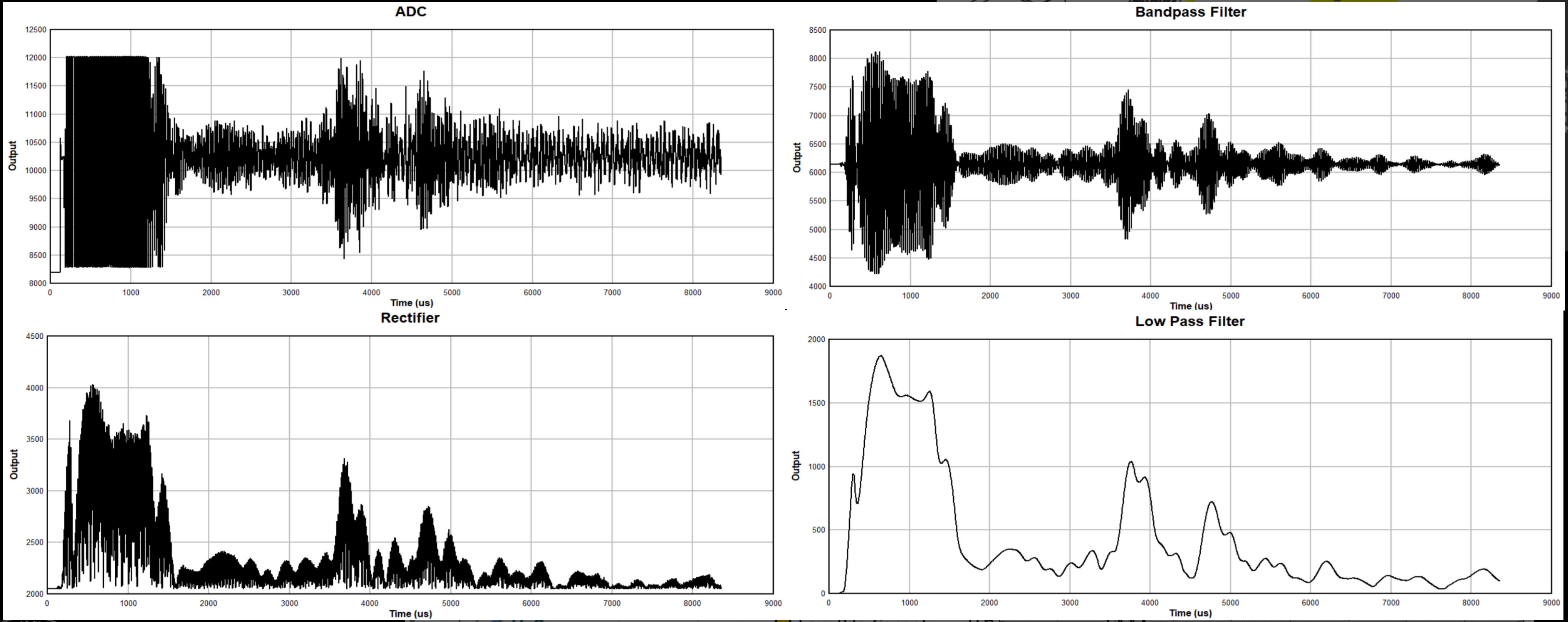


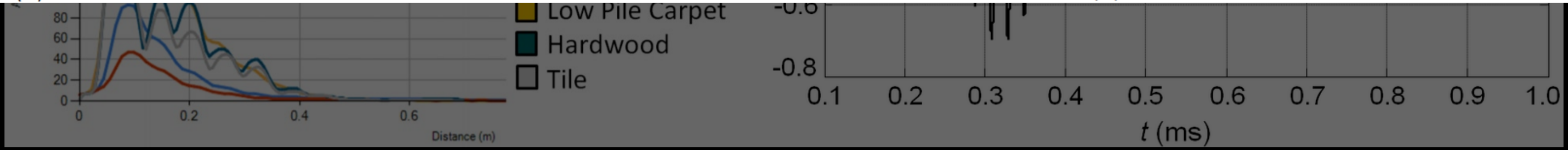




Ultrasound (Time of Flight) Distance Sensors

- Measure the reflections of an emitted sound wave





DIY-level Distance Sensors

Technology	Application	Pros	Cons
Amplitude-based IR	<10cm	• ~ 0.5 USD • Small form factor	• Depends on target reflectivity • Does not work in high ambient light
IR triangulation	<1m	• Insensitive to surface color/texture/ambient light	• ~ 10 USD • Does not work in high ambient light • Bulky (1.75″ × 0.75″ × 0.53″) • Low sample rate (26Hz)
IR Time of Flight	0.1 - 4m	• Small form factor • Insensitive to surface color/texture/ambient light	• ~ 6.5 USD • Complicated processing • Low sampling frequency: 7-30Hz
Ultrasonic	0.2 – 10m	• Low cost • Insensitive to ambient light and surface color • Works in rain and fog	• ~4 USD • Complicated processing • Resolution trade off with max range • Output depends on surface/geometry/humidity • Bulky, sample time (tens of milliseconds) • Hard to achieve a narrow FoV

Prof. Kirstin Hagelskjær Petersen
kirstin@cornell.edu

ODOMETRY SENSORS

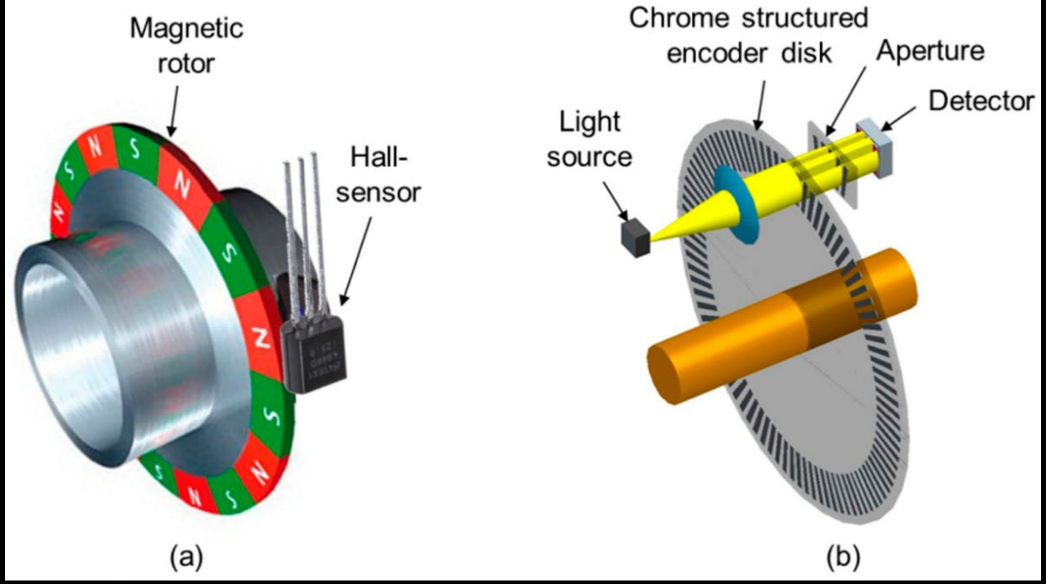
(the process of inferring your position by the integration of speed)

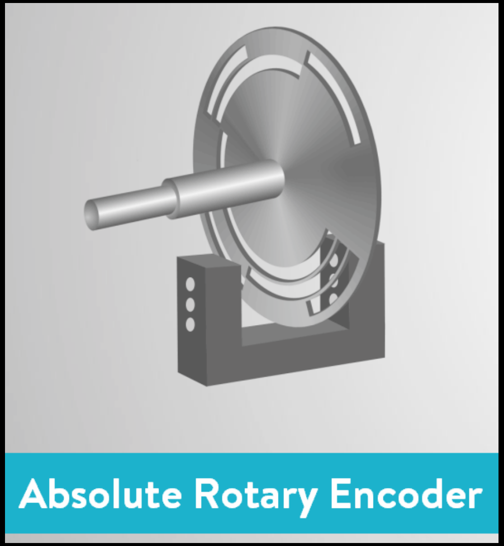
- Wheel encoders
- IMU
- Optical flow

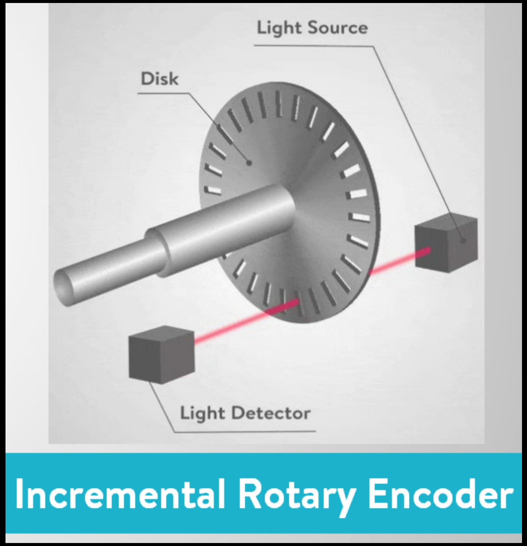
Encoders

- Technology
  - Magnetic
  - Optical
  - Inductive, Capacitive, Laser
- Rotary (shaft) Encoders
  - Absolute Rotary Encoders (angular position)
  - Incremental Rotary Encoders (distance, speed, position)

How to add encoders to your robot?

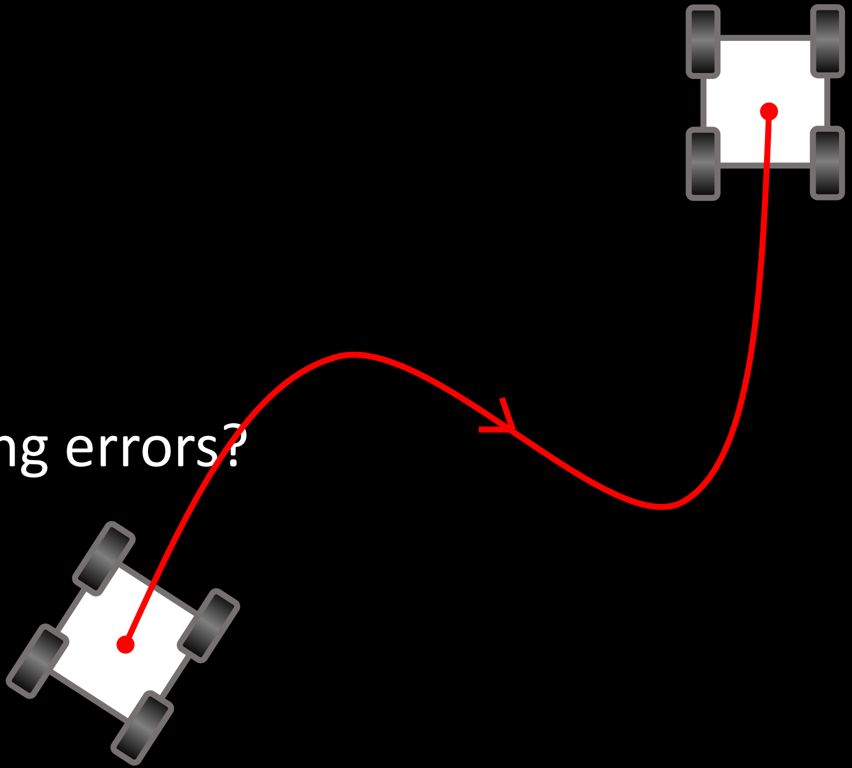






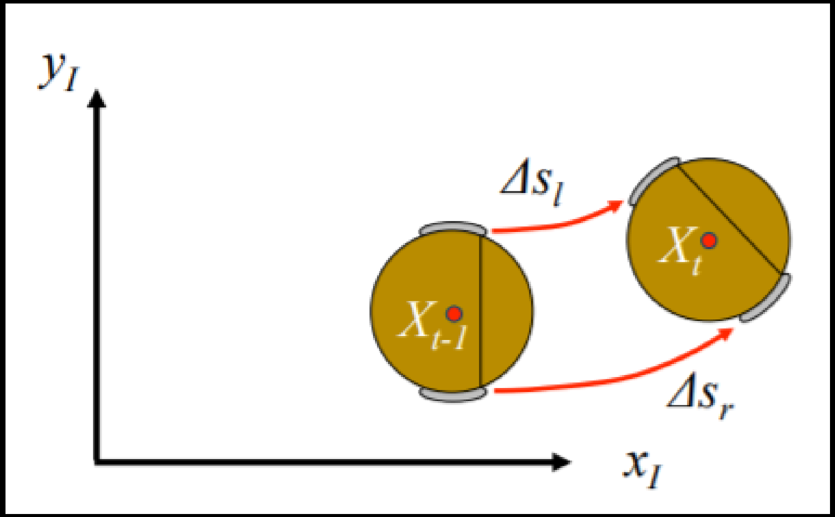
Dead Reckoning

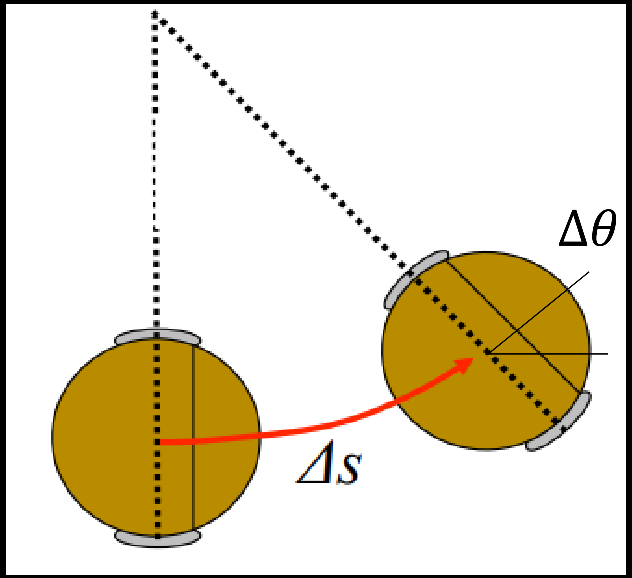
- Map the present state and wheel encoder measurements to the new robot state
  - $X_t = f(X_{t-1}, U_{t-1})$
  - Pro: Easy to implement
  - Con: Errors integrate and grow unbounded
- ***Sources of error?***
  - Limited resolution during integration
  - Unequal wheel diameter
  - Variation in the contact point of the wheel
  - Variable friction > slipping
  - Drift or noise in sensors
- How do wheel rotation errors propagate into positioning errors?



Modeling Motion

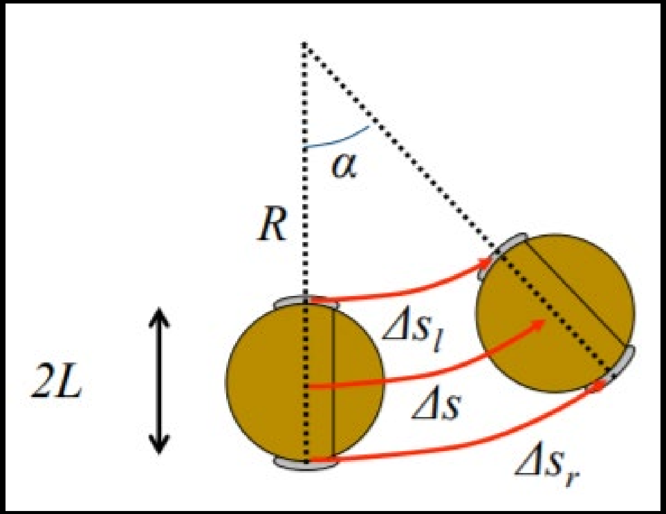
- Start at pose X_{t-1}, move right/left wheel by $\triangle s_r$ and $\triangle s_l$, what is pose X_t?
- Model the change in angle $\Delta\theta$ and the distance travelled Δs
  - (assume that the robot is travelling on a circular arc of constant radius)





Modeling Motion

- Start at pose X_{t-1}, move right/left wheel by $\triangle s_r$ and $\triangle s_l$, what is pose X_t?
- Model the change in angle $\Delta\theta$ and the distance travelled Δs
  - (assume that the robot is travelling on a circular arc of constant radius)



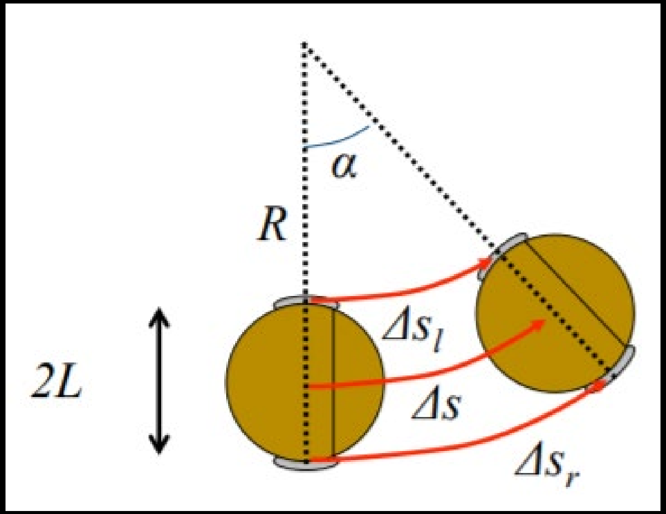
For circular arcs:

- (1) $\Delta s_l = R\alpha$
- (2) $\Delta s_r = (R + 2L)\alpha$
- (3) $\Delta s = (R + L)\alpha$

- Use (1) and (2) to compute (4):
  - $L\alpha = \frac{(\Delta s_r - R\alpha)}{2}$
  - $= \frac{\Delta s_r}{2} - \frac{\Delta s_l}{2}$
- Insert into (3): $\Delta s = \Delta s_l + \frac{\Delta s_r}{2} - \frac{\Delta s_l}{2} = \frac{\Delta s_l + \Delta s_r}{2}$

Modeling Motion

- Start at pose X_{t-1}, move right/left wheel by $\triangle s_r$ and $\triangle s_l$, what is pose X_t?
- Model the change in angle $\Delta\theta$ and the distance travelled Δs
  - (assume that the robot is travelling on a circular arc of constant radius)



For circular arcs:

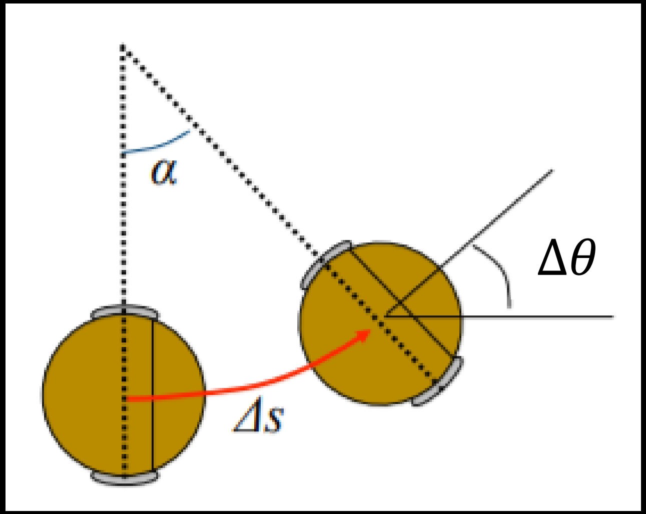
- (1) $\Delta s_l = R\alpha$
- (2) $\Delta s_r = (R + 2L)\alpha$
- (3) $\Delta s = (R + L)\alpha$

- (4) $\Delta s = \frac{\Delta s_l + \Delta s_r}{2}$

- (or note that the distance traveled by the robot center, is simply the avg distance traveled by each wheel)

Modeling Motion

- Start at pose X_{t-1}, move right/left wheel by $\triangle\, s_r$ and $\triangle\, s_l$, what is pose X_t?
- Model the change in angle $\Delta\theta$ and the distance travelled Δs
  - (assume that the robot is travelling on a circular arc of constant radius)



For circular arcs:

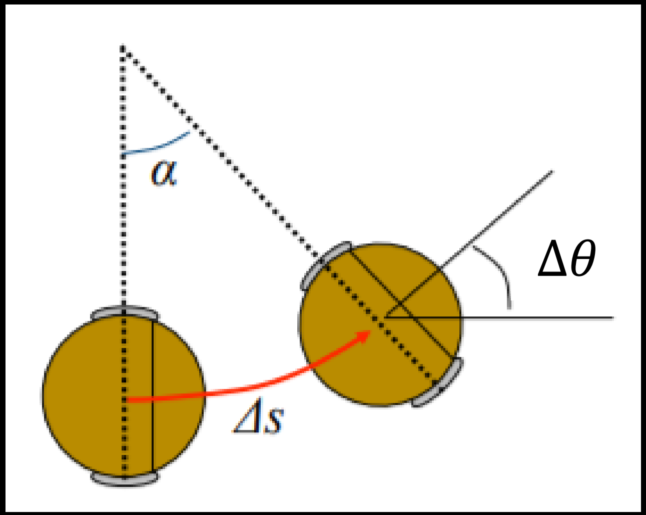
- (1) $\Delta s_l = R\alpha$
- (2) $\Delta s_r = (R + 2L)\alpha$
- (3) $\Delta s = (R + L)\alpha$

- (4) $\Delta s = \frac{\Delta s_l + \Delta s_r}{2}$

- The change in angle, $\Delta\theta$:
  - $\Delta\theta = \alpha$

Modeling Motion

- Start at pose X_{t-1}, move right/left wheel by $\triangle s_r$ and $\triangle s_l$, what is pose X_t?
- Model the change in angle $\Delta\theta$ and the distance travelled Δs
  - (assume that the robot is travelling on a circular arc of constant radius)



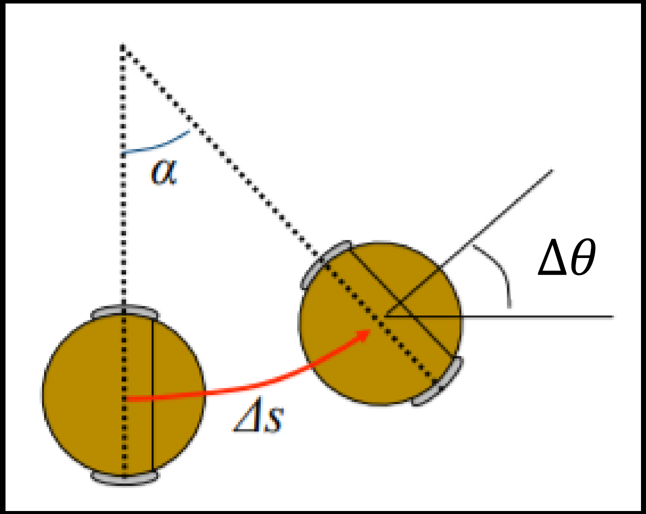
For circular arcs:

- (1) $\Delta s_l = R\alpha$
- (2) $\Delta s_r = (R + 2L)\alpha$
- (3) $\Delta s = (R + L)\alpha$
- (4) $\Delta s = \frac{\Delta s_l + \Delta s_r}{2}$

- Use α in (1) and (2):
  - $\frac{\Delta s_l}{R} = \frac{\Delta s_r}{R+2L} \quad \leftrightarrow \quad (R + 2L)\Delta s_l = R(\Delta s_r)$
  - $\leftrightarrow \quad 2L\Delta s_l = \mathrm{R}(\Delta s_r - \Delta s_l)$
  - $\leftrightarrow \quad R = \frac{2L\Delta s_l}{\Delta s_r - \Delta s_l}$

Modeling Motion

- Start at pose X_{t-1}, move right/left wheel by $\triangle s_r$ and $\triangle s_l$, what is pose X_t?
- Model the change in angle $\Delta\theta$ and the distance travelled Δs
  - (assume that the robot is travelling on a circular arc of constant radius)

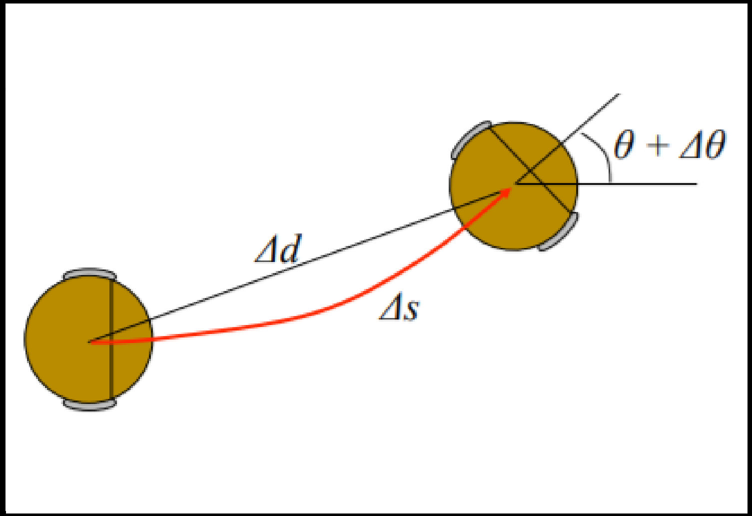


For circular arcs:

- (1) $\Delta s_l = R\alpha$
- (2) $\Delta s_r = (R + 2L)\alpha$
- (3) $\Delta s = (R + L)\alpha$
- (4) $\Delta s = \frac{\Delta s_l + \Delta s_r}{2}$
- (5) $R = \frac{2L\Delta s_l}{\Delta s_r - \Delta s_l}$
- Use (5) in (1) :
  - $\alpha = \frac{\Delta s_l}{R} = \frac{\Delta s_l(\Delta s_r - \Delta s_l)}{2L\Delta s_l} = \frac{\Delta s_r - \Delta s_l}{2L} = \Delta\theta$

Modeling Motion

- Start at pose X_{t-1}, move right/left wheel by $\triangle s_r$ and $\triangle s_l$, what is pose X_t?
- Model the change in angle $\Delta\theta$ and the distance travelled Δs
  - (assume that the robot is travelling on a circular arc of constant radius)
  - (assume that the motion is small, $\Delta d \approx \Delta s$)



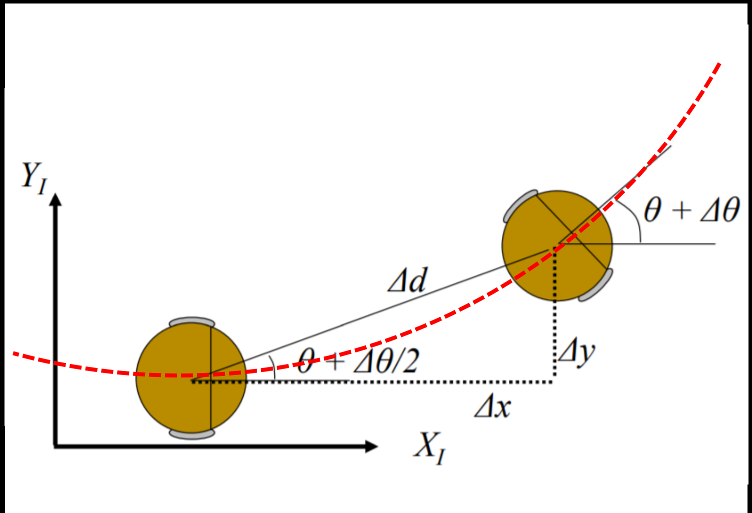
For circular arcs:

- (1) $\Delta s_l = R\alpha$
- (2) $\Delta s_r = (R + 2L)\alpha$
- (3) $\Delta s = (R + L)\alpha$

- (4) $\Delta s = \frac{\Delta s_l + \Delta s_r}{2}$
- (5) $R = \frac{2L\Delta s_l}{\Delta s_r - \Delta s_l}$
- (6) $\Delta\theta = \frac{\Delta s_r - \Delta s_l}{2L}$

Modeling Motion

- Start at pose X_{t-1}, move right/left wheel by $\triangle s_r$ and $\triangle s_l$, what is pose X_t?
- Model the change in angle $\Delta\theta$ and the distance travelled Δs
  - (assume that the robot is travelling on a circular arc of constant radius)
  - (assume that the motion is small, $\Delta d \approx \Delta s$)

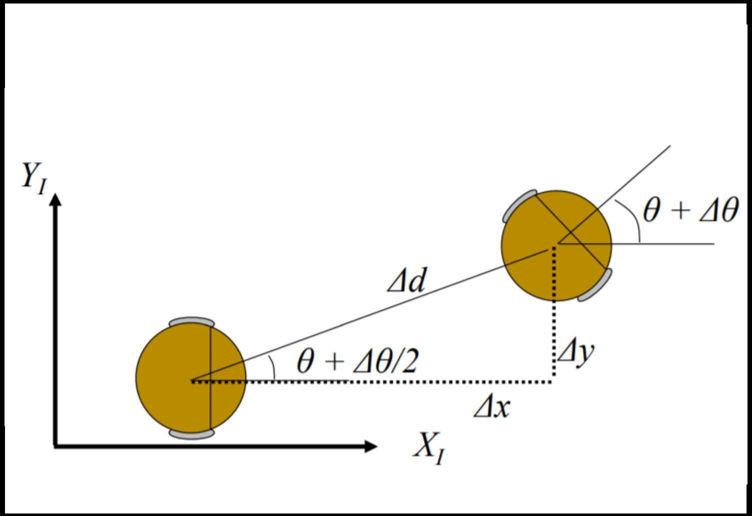


For circular arcs:

- (1) $\Delta s_l = R\alpha$
- (2) $\Delta s_r = (R + 2L)\alpha$
- (3) $\Delta s = (R + L)\alpha$
- (4) $\Delta s = \frac{\Delta s_l + \Delta s_r}{2}$
- (5) $R = \frac{2L\Delta s_l}{\Delta s_r - \Delta s_l}$
- (6) $\Delta\theta = \frac{\Delta s_r - \Delta s_l}{2L}$
- (7) $\Delta x = \Delta s\ cos(\theta + \Delta\theta/2)$
- (8) $\Delta y = \Delta s\ sin(\theta + \Delta\theta/2)$

Modeling Motion

- Start at pose X_{t-1}, move right/left wheel by $\triangle s_r$ and $\triangle s_l$, what is pose X_t?
- Model the change in angle $\Delta\theta$ and the distance travelled Δs
  - (assume that the robot is travelling on a circular arc of constant radius)
  - (assume that the motion is small, $\Delta d \approx \Delta s$)

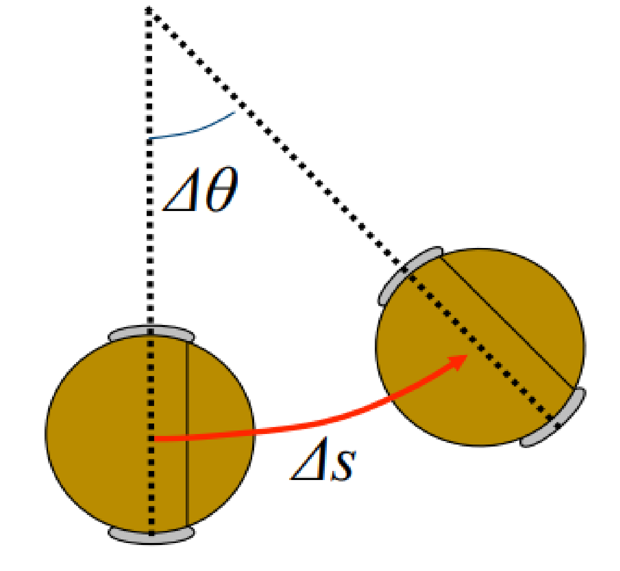


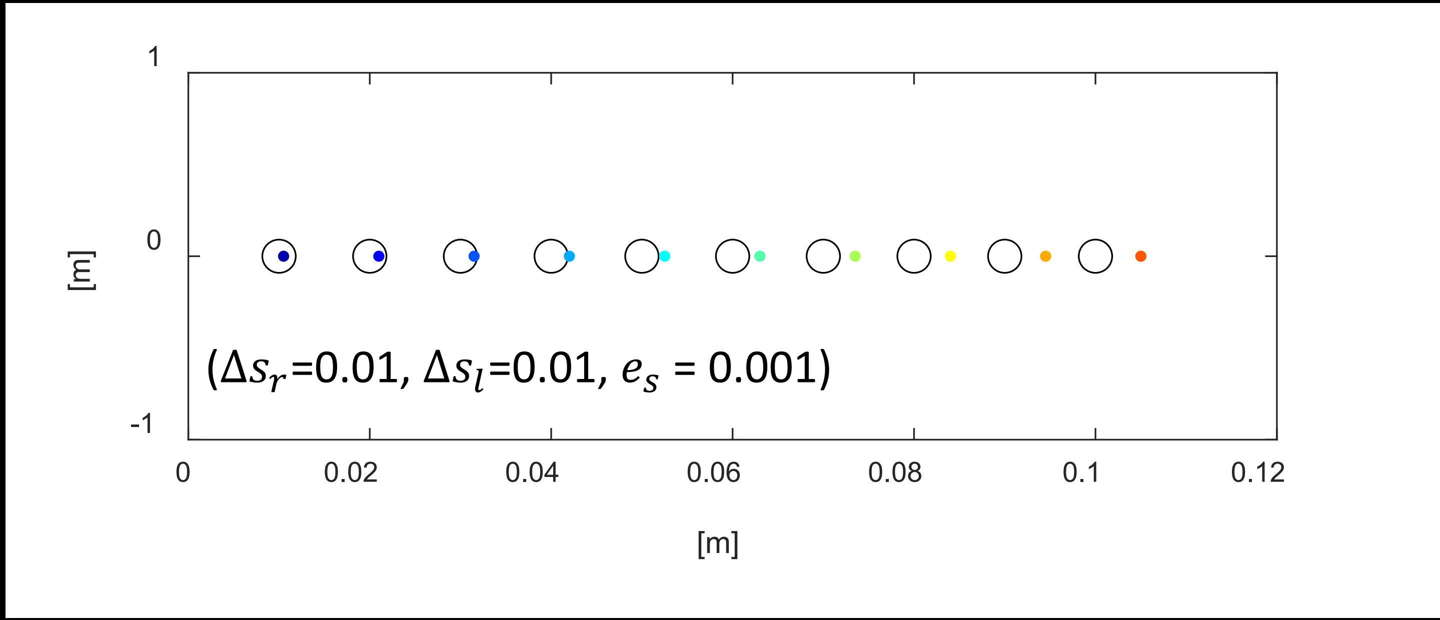
- (4) $\Delta s = \frac{\Delta s_l + \Delta s_r}{2}$
- (6) $\Delta \theta = \frac{\Delta s_r - \Delta s_l}{2L}$
- (7) $\Delta x = \Delta s\ cos(\theta + \Delta\theta/2)$
- (8) $\Delta y = \Delta s\ sin(\theta + \Delta\theta/2)$
- $X_t = f(x, y, \theta, \Delta s_r, \Delta s_l)$
- $X_t = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} + \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta\theta \end{bmatrix}$

$$\boxed{= \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} + \begin{bmatrix} \frac{\Delta s_l + \Delta s_r}{2} cos\left(\theta + \frac{\Delta s_r - \Delta s_l}{4L}\right) \\ \frac{\Delta s_l + \Delta s_r}{2} sin\left(\theta + \frac{\Delta s_r - \Delta s_l}{4L}\right) \\ \frac{\Delta s_r - \Delta s_l}{2L} \end{bmatrix}}$$

Modeling Motion

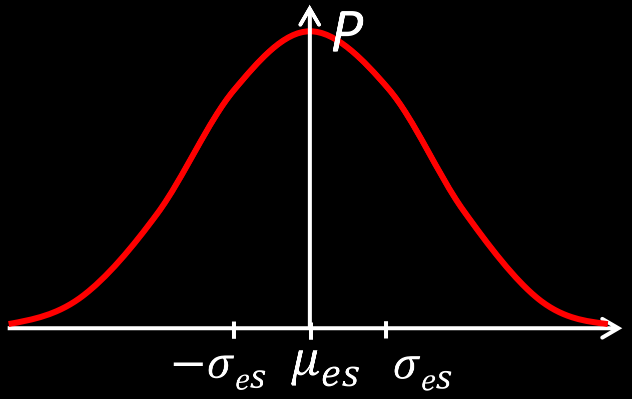
- How do wheel rotation errors propagate into positioning errors?
  - $\Delta s = d + e_s$
    - $\Delta x = \frac{\Delta s_l + \Delta s_r + e_s}{2} cos\left(\theta + \frac{\Delta s_r - \Delta s_l}{4L}\right)$
    - $\Delta y = \frac{\Delta s_l + \Delta s_r + e_s}{2} sin\left(\theta + \frac{\Delta s_r - \Delta s_l}{4L}\right)$

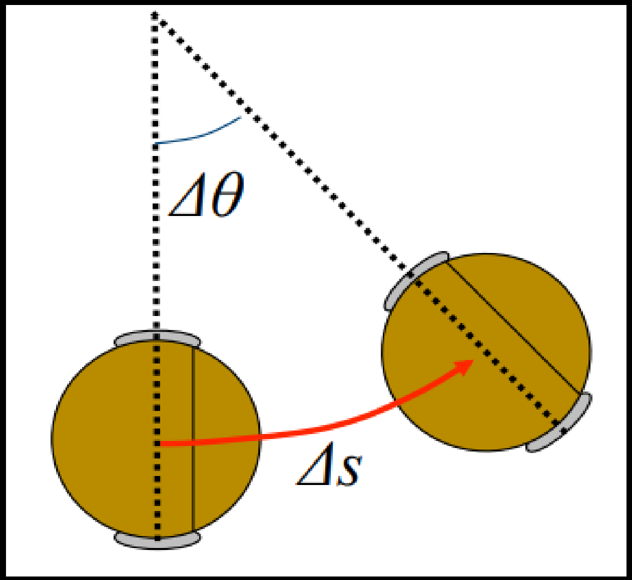


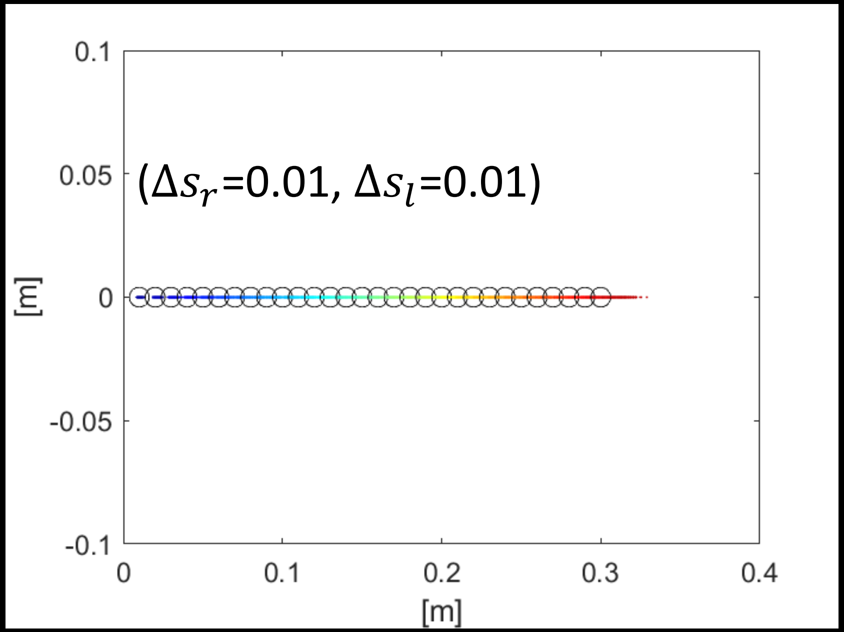


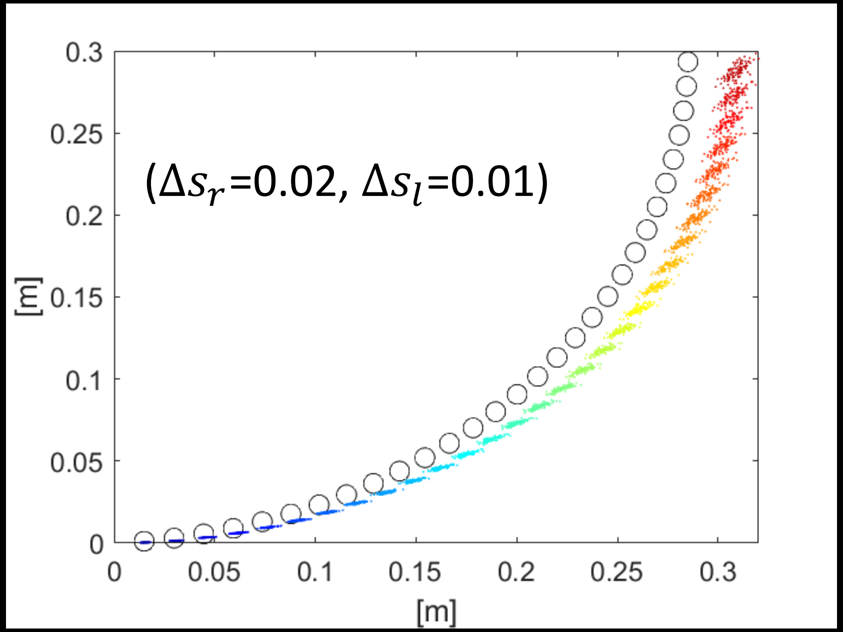
Modeling Motion

- How do wheel rotation errors propagate into positioning errors?
  - $\Delta s = d + e_s$
    - $\Delta x = \frac{\Delta s_l + \Delta s_r + e_s}{2} cos\left(\theta + \frac{\Delta s_r - \Delta s_l}{4L}\right)$
    - $\Delta y = \frac{\Delta s_l + \Delta s_r + e_s}{2} sin\left(\theta + \frac{\Delta s_r - \Delta s_l}{4L}\right)$
    - e_s (μ_{es}= 1mm, σ_{es}= 2mm)



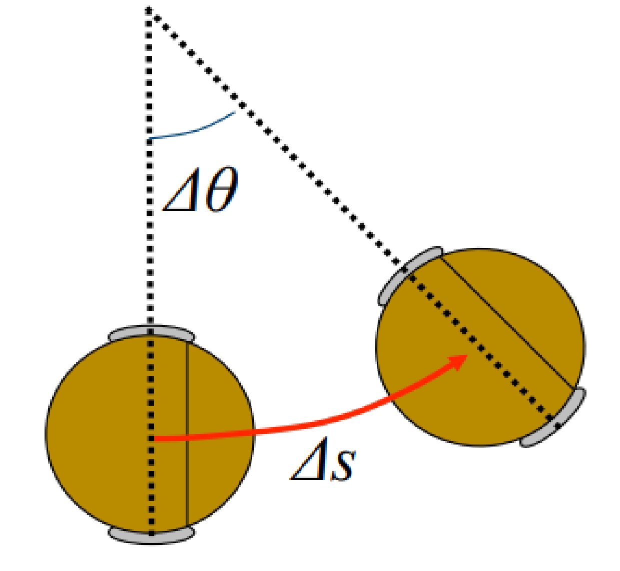


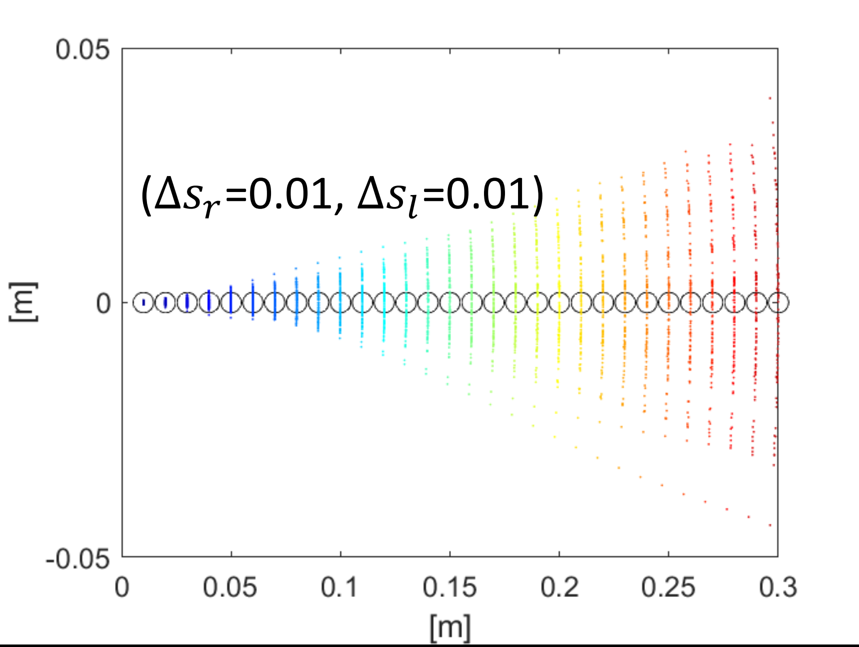


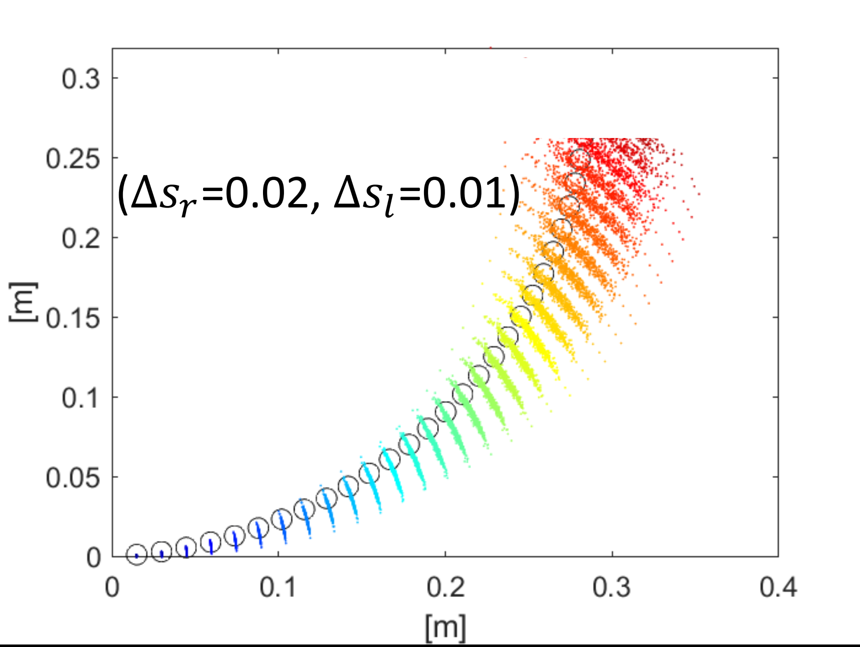


Modeling Motion

- How do wheel rotation errors propagate into positioning errors?
  - $\Delta\theta = \beta + e_\theta$, e_θ = 1°
    - $\Delta x = \Delta s \; cos\left(\theta + \frac{\beta}{2} + e_\theta\right) \quad = 0.1000m$
    - $\Delta y = \Delta s \; sin\left(\theta + \frac{\beta + e_\theta}{2} + e_\theta\right) = 0.0175m$
    - e_θ ($\mu_{e\theta}$= 0°, $\sigma_{e\theta}$= 1°)







Sources and References

- http://www.cs.cmu.edu/~rasc/Download/AMRobots4.pdf
- https://www.ti.com/lit/ug/sbau305b/sbau305b.pdf?ts=1599417595209&ref_url=https%253A%252F%252Fwww.google.com%252F
- https://hmc.edu/lair/ARW/ARW-Lecture01-Odometry.pdf
- Matlab Tech Talks on Sensor Fusion (https://www.youtube.com/watch?v=6qV3YjFppuc)

Introduction to Lab 2

- Last 10min of class

Lab 2 Bluetooth

- *NB: The lab has changed slightly from last year!*
- Good example from last year
  - Owen Deng: https://qd39l.github.io/fast-robots/lab2.html
- Lab 2
  - https://cei-lab.github.io/FastRobots-2023/Lab2.html
  - Prelab (setup VM/Jupyter and Artemis, read through the code base)
  - Tasks
    - Change the MAC address/UUID
    - Send/receive ECHO
    - Get_Time_Millis()
    - Notification handler
    - Get_Temp_5s (T:1500|C:24|T:2657|C:24.5 …etc)
    - Get_Temp_Rapid
    - Consider Artemis storage