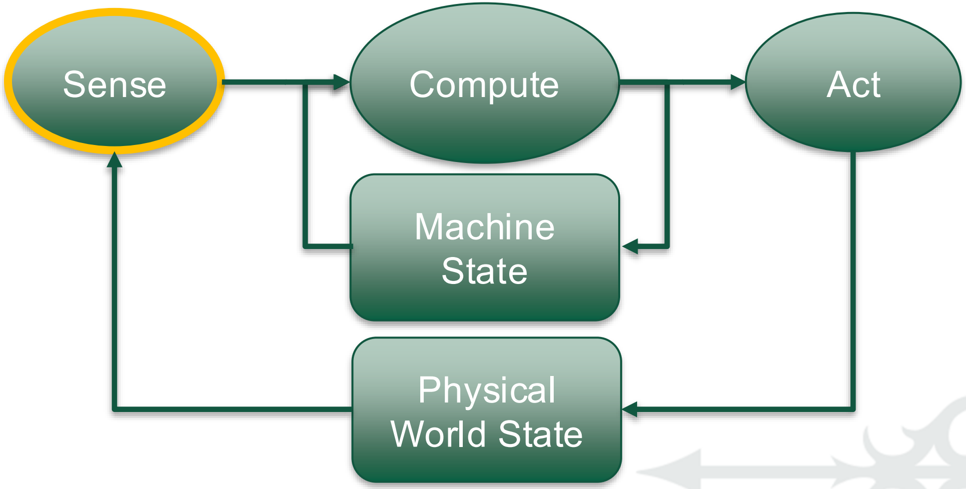
Sensors and Noise Management

CSCI 420-02 Robotics

WILLIAM & MARY
CHARTERED 1693


Sense
Compute
Act
Machine State
Physical World State

What is a sensor?

- Transduce energy into measurement
- Observe a physical phenomenon in physical units
  - Are the units aligned?
- Provides a window into the world or robot

What is i

- Example: Accelero

https://www.sparkfun.com/sparkfun-triple-axis-accelerometer-breakout-adxl345.html

SPECIFICATIONS

T_A = 25°C, V_S = 2.5 V, $V_{DD\ I/O}$ = 1.8 V, acceleration = 0 *g*, C_S = 1 μF tantalum, C_{IO} = 0.1 μF, unless otherwise noted.

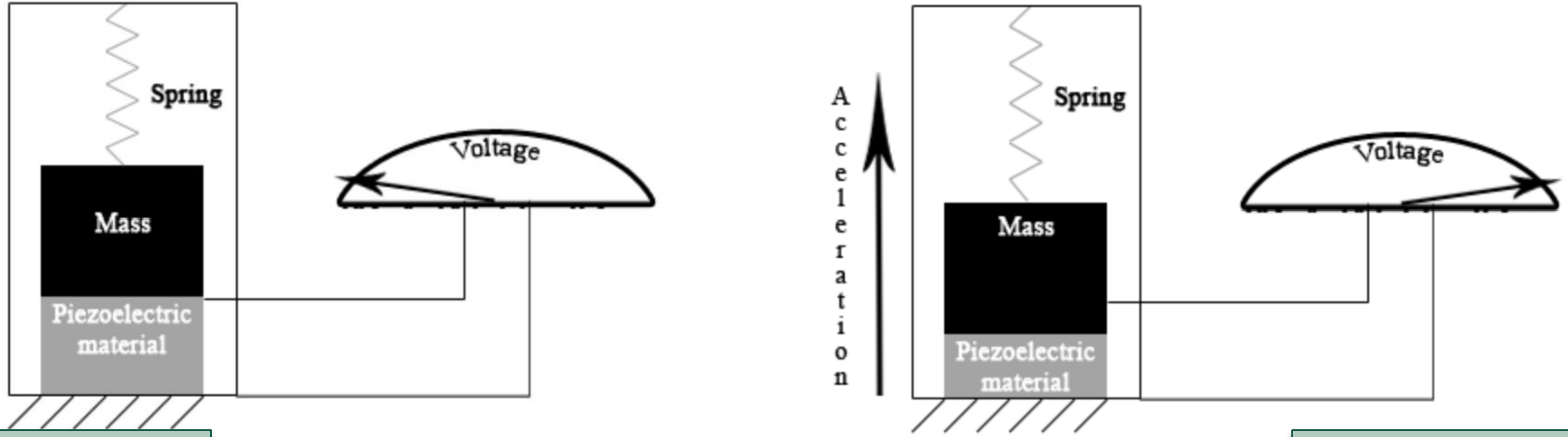
Table 1. Specifications[1]

Parameter	Test Conditions	Min	Typ	Max	Unit
SENSOR INPUT	Each axis				
Measurement Range	User selectable		±2, ±4, ±8, ±16		*g*
Nonlinearity	Percentage of full scale		±0.5		%
Inter-Axis Alignment Error			±0.1		Degrees
Cross-Axis Sensitivity[2]			±1		%
OUTPUT RESOLUTION	Each axis				
All *g* Ranges	10-bit resolution		10		Bits
±2 *g* Range	Full resolution		10		Bits
±4 *g* Range	Full resolution		11		Bits
±8 *g* Range	Full resolution		12		Bits
±16 *g* Range	Full resolution		13		Bits
SENSITIVITY	Each axis				
Sensitivity at X_{OUT}, Y_{OUT}, Z_{OUT}	±2 *g*, 10-bit or full resolution	232	256	286	LSB/*g*
Scale Factor at X_{OUT}, Y_{OUT}, Z_{OUT}	±2 *g*, 10-bit or full resolution	3.5	3.9	4.3	m*g*/LSB
Sensitivity at X_{OUT}, Y_{OUT}, Z_{OUT}	±4 *g*, 10-bit resolution	116	128	143	LSB/*g*
Scale Factor at X_{OUT}, Y_{OUT}, Z_{OUT}	±4 *g*, 10-bit resolution	7.0	7.8	8.6	m*g*/LSB
Sensitivity at X_{OUT}, Y_{OUT}, Z_{OUT}	±8 *g*, 10-bit resolution	58	64	71	LSB/*g*
Scale Factor at X_{OUT}, Y_{OUT}, Z_{OUT}	±8 *g*, 10-bit resolution	14.0	15.6	17.2	m*g*/LSB
Sensitivity at X_{OUT}, Y_{OUT}, Z_{OUT}	±16 *g*, 10-bit resolution	29	32	36	LSB/*g*
Scale Factor at X_{OUT}, Y_{OUT}, Z_{OUT}	±16 *g*, 10-bit resolution	28.1	31.2	34.3	m*g*/LSB
Sensitivity Change Due to Temperature			±0.01		%/°C
0 *g* BIAS LEVEL	Each axis				
0 *g* Output for X_{OUT}, Y_{OUT}		−150	±40	+150	m*g*
0 *g* Output for Z_{OUT}		−250	±80	+250	m*g*
0 *g* Offset vs. Temperature for x-, y-Axes			±0.8		m*g*/°C
0 *g* Offset vs. Temperature for z-Axis			±4.5		m*g*/°C
NOISE PERFORMANCE					
Noise (x-, y-Axes)	Data rate = 100 Hz for ±2 *g*, 10-bit or full resolution		<1.0		LSB rms
Noise (z-Axis)	Data rate = 100 Hz for ±2 *g*, 10-bit or full resolution		<1.5		LSB rms
OUTPUT DATA RATE AND BANDWIDTH	User selectable				
Measurement Rate[3]		6.25		3200	Hz
SELF-TEST[4]	Data rate ≥ 100 Hz, 2.0 V ≤ V_S ≤ 3.6 V				
Output Change in x-Axis		0.20		2.10	*g*
Output Change in y-Axis		−2.10		−0.20	*g*
Output Change in z-Axis		0.30		3.40	*g*
POWER SUPPLY					
Operating Voltage Range (V_S)		2.0	2.5	3.6	V
Interface Voltage Range ($V_{DD\ I/O}$)	V_S ≤ 2.5 V	1.7	1.8	V_S	V
	V_S ≥ 2.5 V	2.0	2.5	V_S	V
Supply Current	Data rate > 100 Hz		145		μA
	Data rate < 10 Hz		40		μA
Standby Mode Leakage Current			0.1	2	μA
Turn-On Time[5]	Data rate = 3200 Hz		1.4		ms
TEMPERATURE					
Operating Temperature Range		−40		+85	°C
WEIGHT					
Device Weight			20		mg

What is in a sensor?

- Example: Accelerometer

https://www.sparkfun.com/sparkfun-triple-axis-accelerometer-breakout-adxl345.html



Phenomenon → Measured

Acceleration to mechanical stress to electrical potential

https://learn.sparkfun.com/tutorials/accelerometer-basics/all

Sensors in ROS

- Sensor Node Implementations
- Sensor Msgs
  - BatteryState
  - Image
  - LaserScan
  - PointCloud
  - Temperature

Image

This is a ROS message definition.

Source

```
# This message contains an uncompressed image
# (0, 0) is at top-left corner of image

std_msgs/Header header # Header timestamp should be acquisition time of image
                             # Header frame_id should be optical frame of camera
                             # origin of frame should be optical center of cameara
                             # +x should point to the right in the image
                             # +y should point down in the image
                             # +z should point into to plane of the image
                             # If the frame_id here and the frame_id of the CameraInfo
                             # message associated with the image conflict
                             # the behavior is undefined

uint32 height                # image height, that is, number of rows
uint32 width                 # image width, that is, number of columns

# The legal values for encoding are in file include/sensor_msgs/image_encodings.hpp
# If you want to standardize a new string format, join
# ros-users@lists.ros.org and send an email proposing a new encoding.

string encoding       # Encoding of pixels -- channel meaning, ordering, size
                      # taken from the list of strings in include/sensor_msgs/image_encodings.hpp

uint8 is_bigendian    # is this data bigendian?
uint32 step           # Full row length in bytes
uint8[] data          # actual matrix data, size is (step * rows)
```

Types of Sensors

- Inward vs Outward
  - Proprioceptive: measures the system
  - Exteroceptive: measures the world
- Active vs Passive
  - Passive: receives energy from world
  - Active: sends energy into world

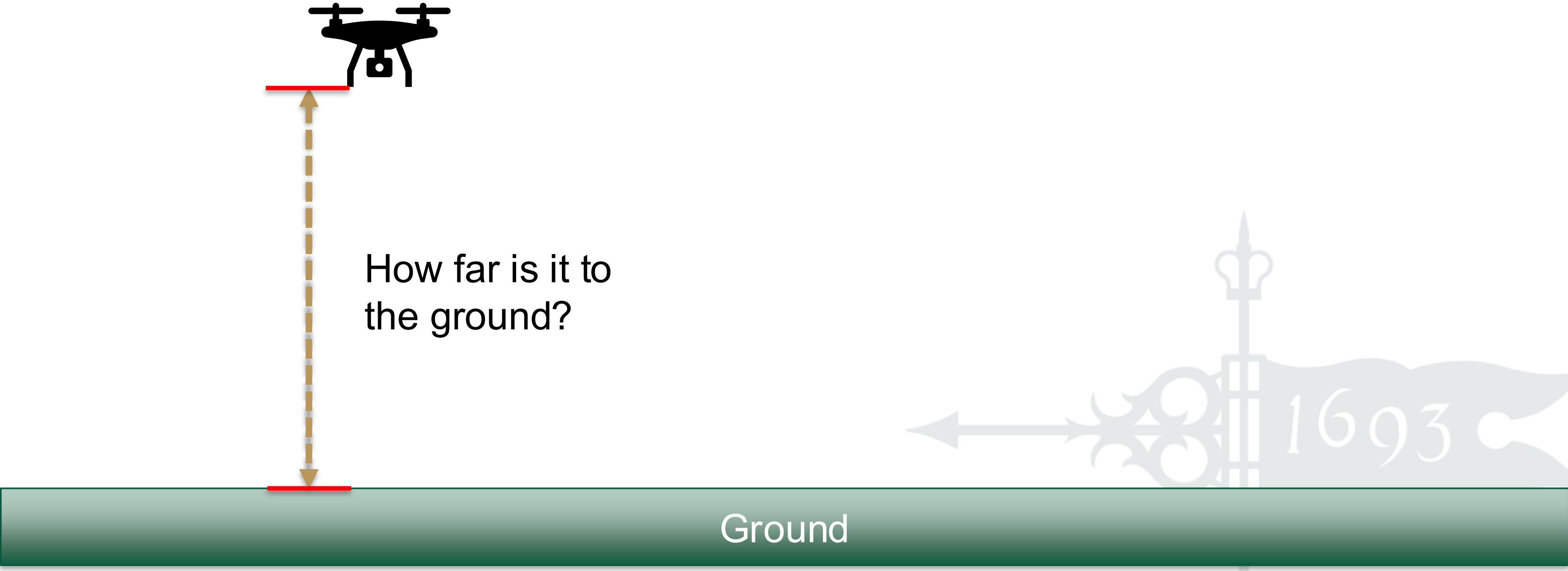
Types of Sensors

- Exteroceptive
  - Passive: Camera, Compass
  - Active: LiDAR, Radar, Ultrasonic
- Proprioceptive:
  - Passive: IMU, Encoder
  - Reference: Global Navigation Satellite System

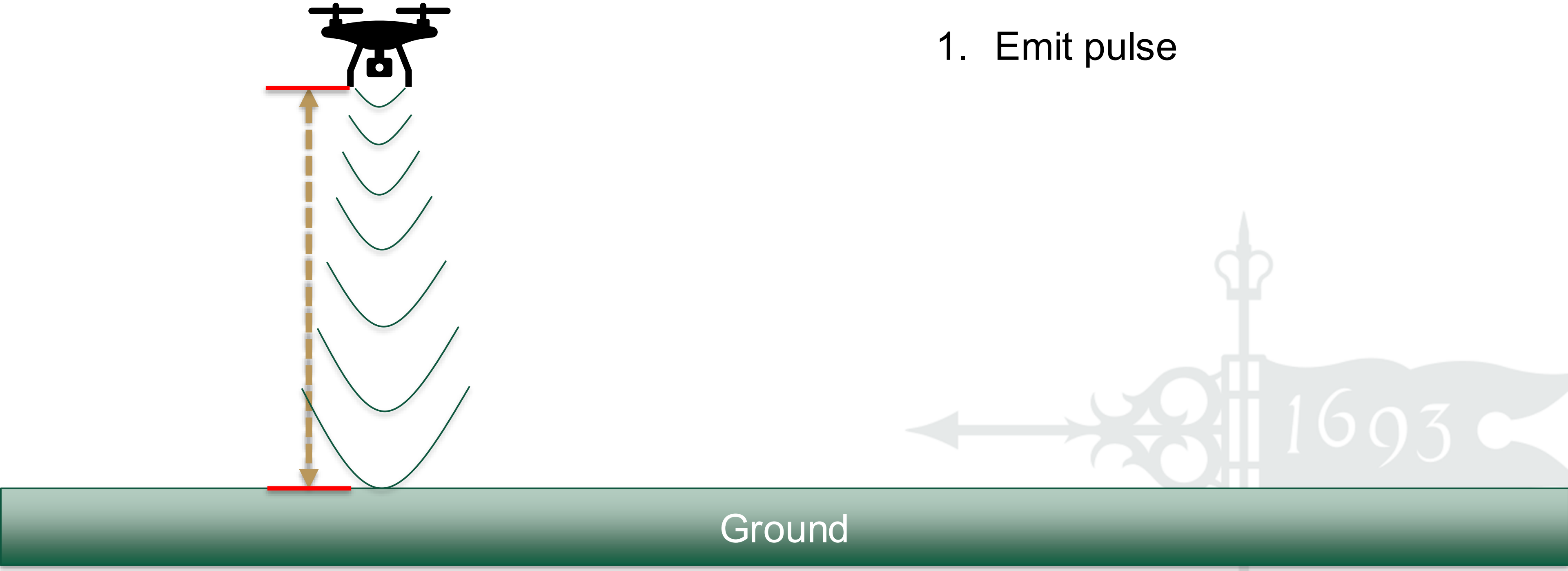
How does our drone sense?

- Altitude
  - Lab 2 uses the pressure sensor
  - What else could we use?
    - GPS
    - Range finder
      - LiDAR
      - Radar
      - Ultrasonic

How do range finders work?

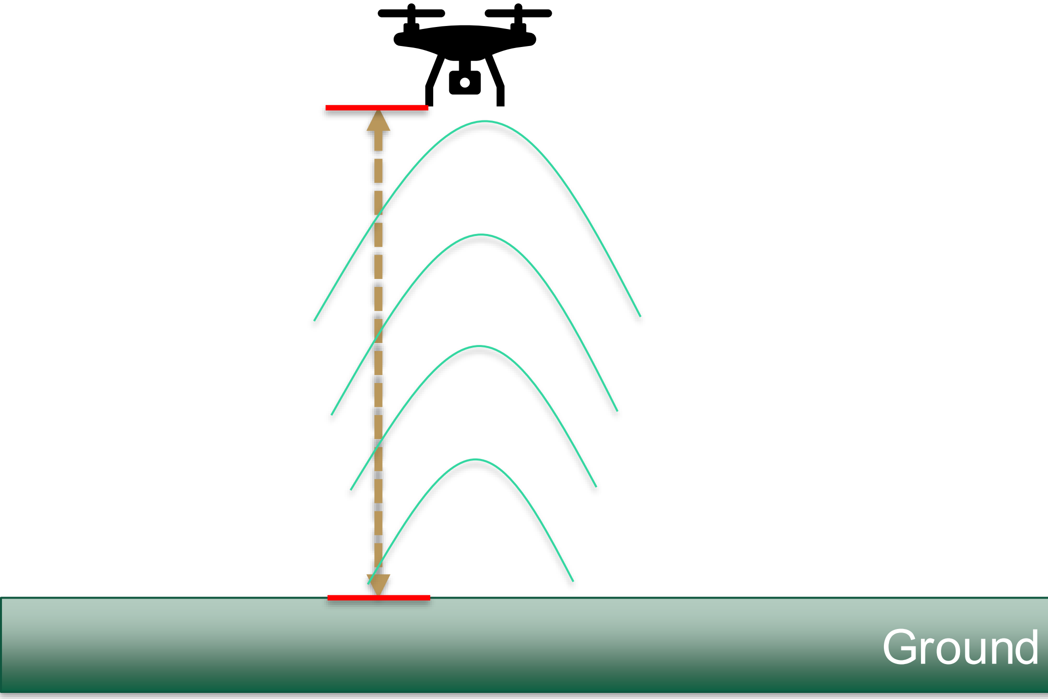


How do range finders work?



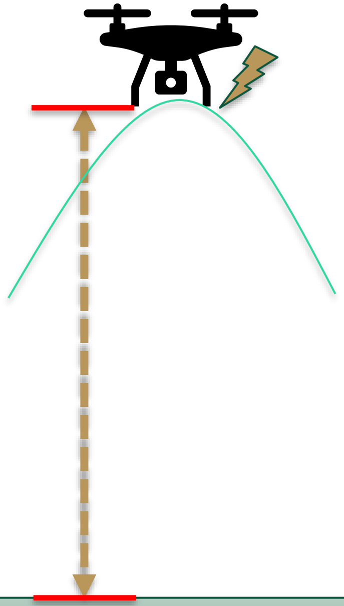
1. Emit pulse

How do range finders work?



1. Emit pulse
2. Pulse reflects (echo)

How do range finders work?

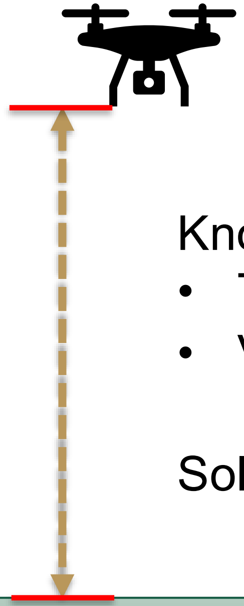


1. Emit pulse
2. Pulse reflects (echo)
3. Pulse received by sensor

What do we measure?

Ground

How do range finders work?



1. Emit pulse
2. Pulse reflects (echo)
3. Pulse received by sensor

Known:

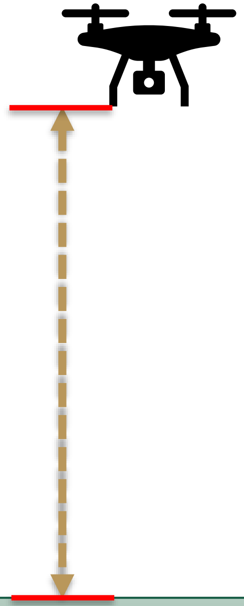
- Time between pulse sent and received $t_{diff} = t_{start} - t_{end}$
- Velocity of signal (v_p)

Solve:

$$d_{ground} = \frac{1}{2} d_{travel} = \frac{1}{2}(t_{diff} \cdot v_p)$$

Ground

How do range finders work?



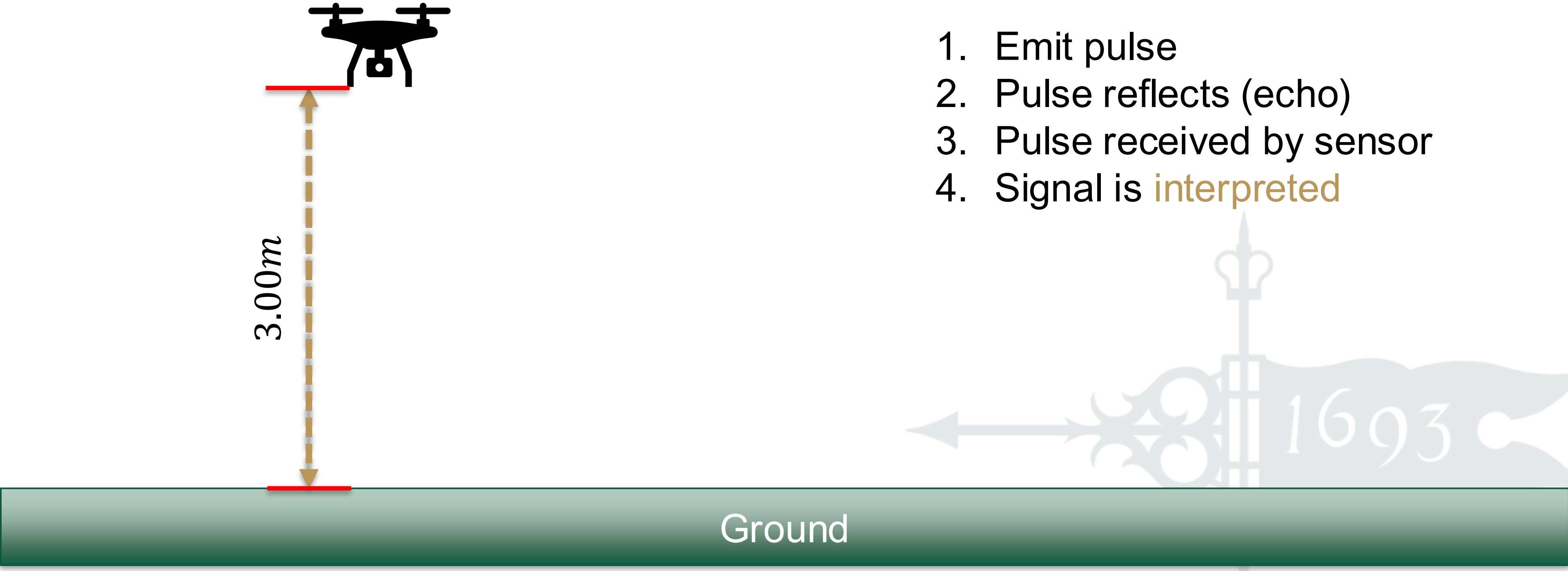
Solve: $d_{ground} = \frac{1}{2} d_{travel} = \frac{1}{2}(t_{diff} \cdot v_p)$

Example: Ultrasonic

- $v_p = 343\frac{m}{s}$
- $t_{diff} = 17.5ms$
- $d_{ground} = \frac{1}{2}\left(17.5ms \cdot \frac{343m}{s}\right)$
- $d_{ground} = \frac{1}{2}\left(17.5ms \cdot \frac{1s}{1000ms} \cdot \frac{343m}{s}\right)$
- $d_{ground} = \frac{1}{2}\left(\frac{17.5\cancel{ms}\cdot 1\cancel{s}\cdot 343m}{1000\cancel{ms}\cdot \cancel{1s}}\right)$
- $d_{ground} = 3.00m$

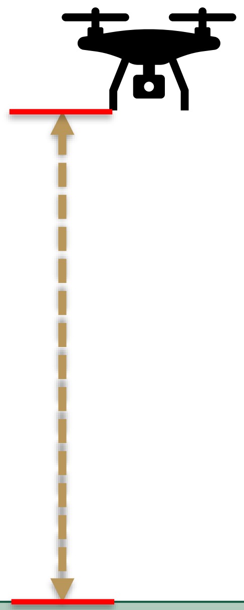
Ground

How do range finders work?



1. Emit pulse
2. Pulse reflects (echo)
3. Pulse received by sensor
4. Signal is interpreted

How do range finders work?



Solve: $d_{ground} = \frac{1}{2} d_{travel} = \frac{1}{2} (t_{diff} \cdot v_p)$

Example: Ultrasonic

- $v_p = 343 \frac{m}{s}$
- $t_{diff} = 17.5 ms$
- $d_{ground} = 3.00 m$

Leaky Abstractions:

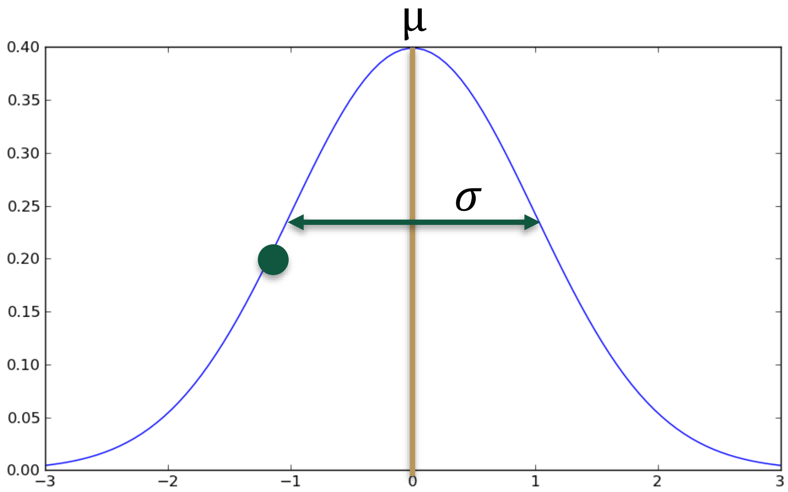
- Speed is $\frac{343m}{s}$ when:
  - 20°C, dry, sea level
- Ground is flat, level
- Drone is stationary relative to the ground

Sensors can be inaccurate

- Each reading is a single sample

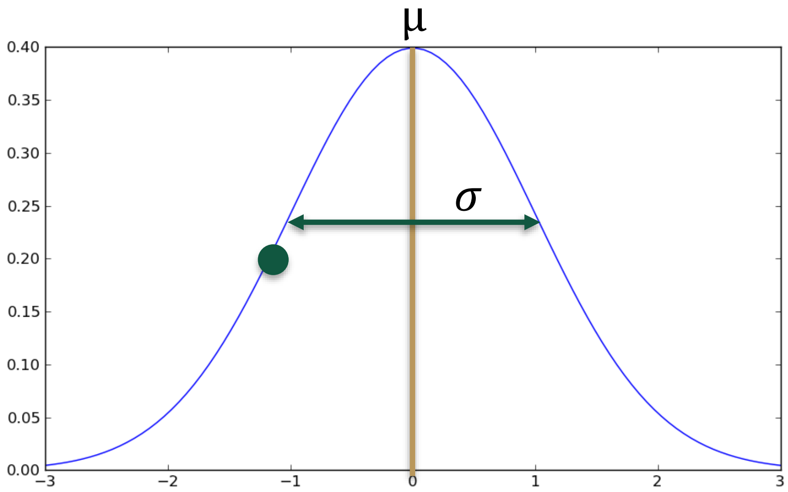
Sensors can be inaccurate

- Each reading is a single sample
- From a distribution



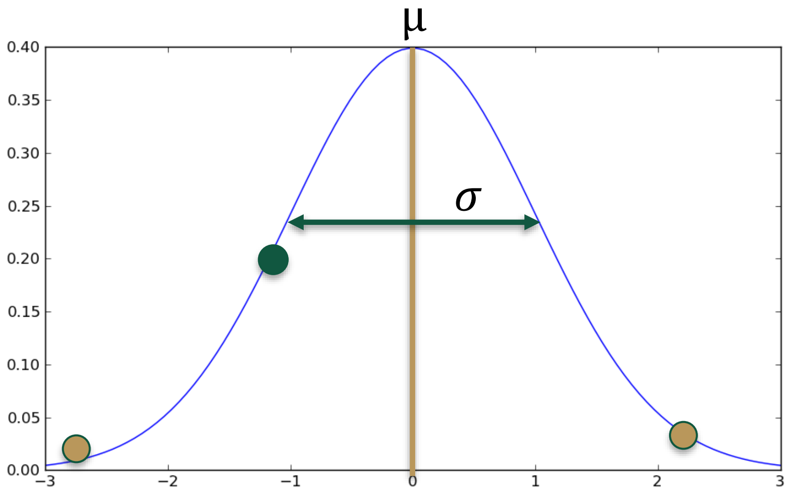
Sensors can be inaccurate

- Each reading is a single sample
- From a distribution



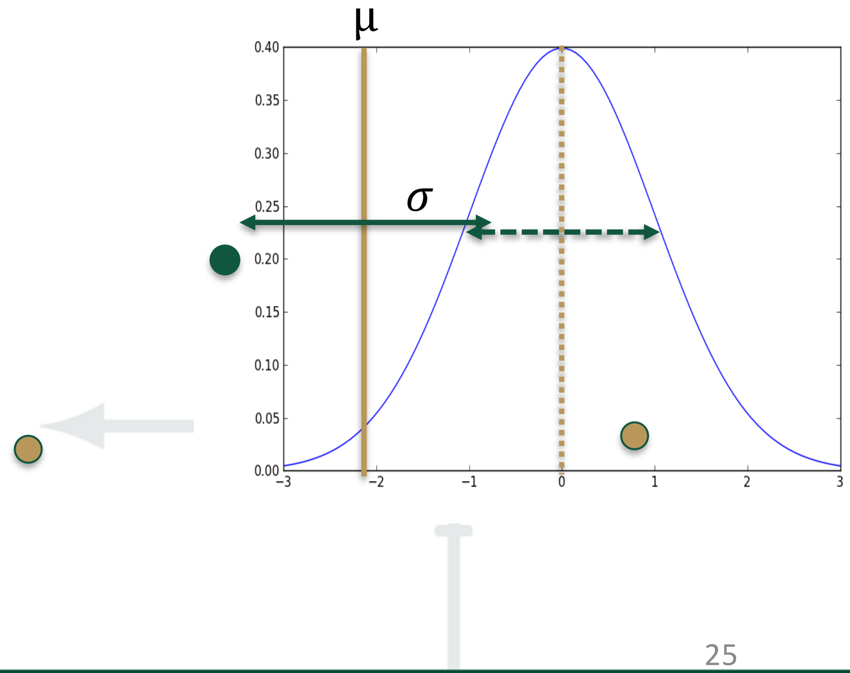
Sensors can be inaccurate

- Each reading is a single sample
- From a distribution
  - Can have outliers



Sensors can be inaccurate

- Each reading is a single sample
- From a distribution
  - Can have outliers
  - Can shift



Managing noise

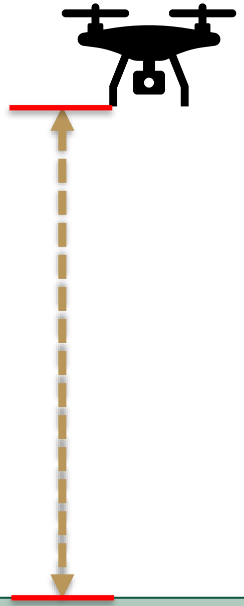
- Calibration
- Filtering
- Fusion

Calibration

- Shift due to environmental assumption
- Need to adjust for new context
- To Calibrate:
  - Conduct new *standardized* tests
  - Recompute constants and errors
  - Redefine model parameters

How to calibrate?

Solve: $d_{ground} = \frac{1}{2} d_{travel} = \frac{1}{2}(t_{diff} \cdot v_p)$



Example: Ultrasonic

- $v_p = 343\frac{m}{s}$
- $t_{diff} = 17.5ms$
- $d_{ground} = 3.00m$

Known:

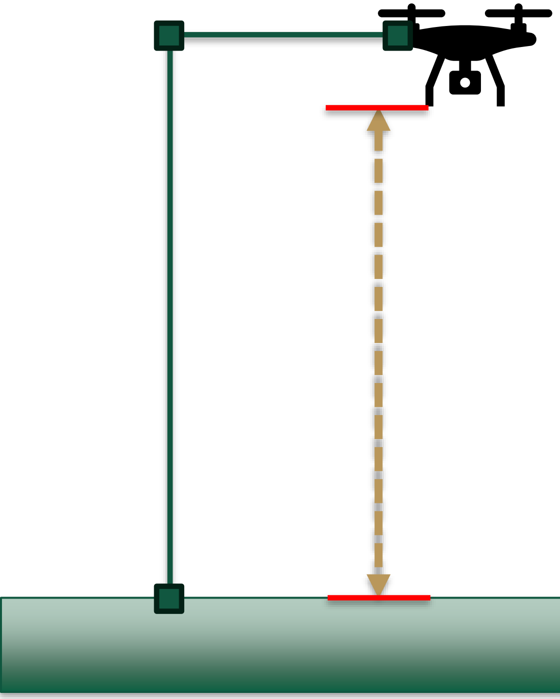
- Time
- Velocity?

Leaky Abstractions:

- Speed is $\frac{343m}{s}$ when:
  - **20°C, dry, sea level**
- Ground is flat, level
- Drone is stationary relative to the ground

Ground

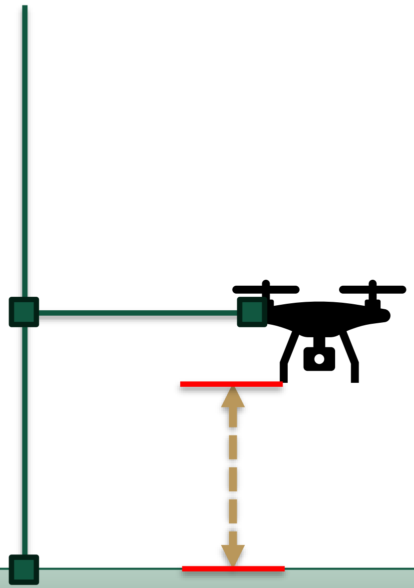
How do we find velocity?

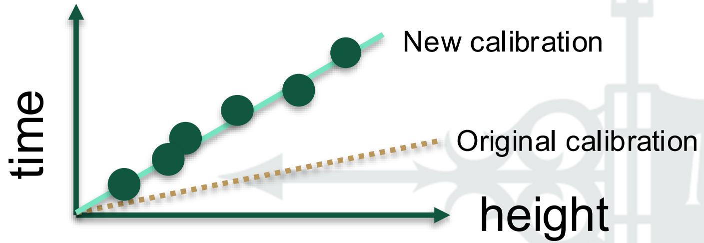


- Conduct new *standardized* tests
  - Mount drone at fixed, known height
  - Measure time to compute velocity

How do we find velocity?

- Conduct new *standardized* tests
  - Mount drone at fixed, known height
  - Measure time to compute velocity
  - Repeat for many heights





Ground

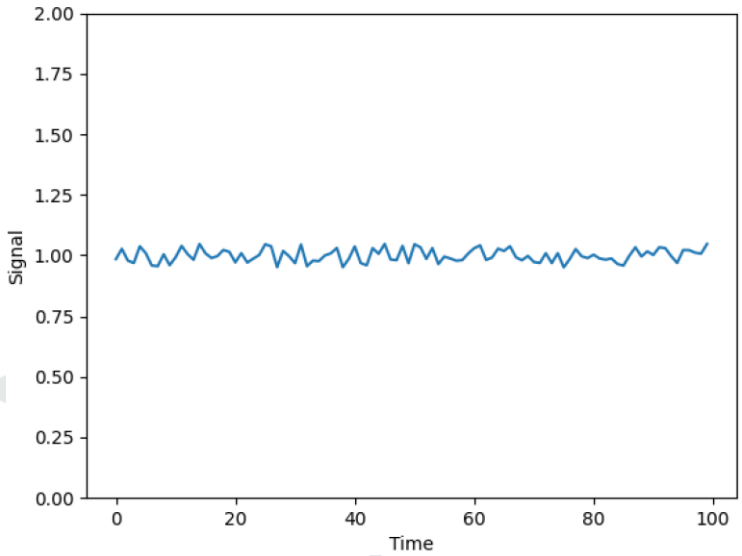
Recalibrate based on parameters

Altitude Meters (m)	Temperature Celsius (°C)	Speed of Sound m/s
0 (sea level)	21	344
3048 (10k feet)	-4.8	328
6096 (20k feet)	-24.6	316
9144 (30k feet)	-44.4	303

You may be able to look up parameters based on your context!

Filtering

- Sensors are noisy
  - Physical disturbances
  - Unaccounted for variables



Filtering

- Signal Processing
  - High Pass
  - Low Pass
  - Band Pass
- Most noise is a different frequency than the signal!

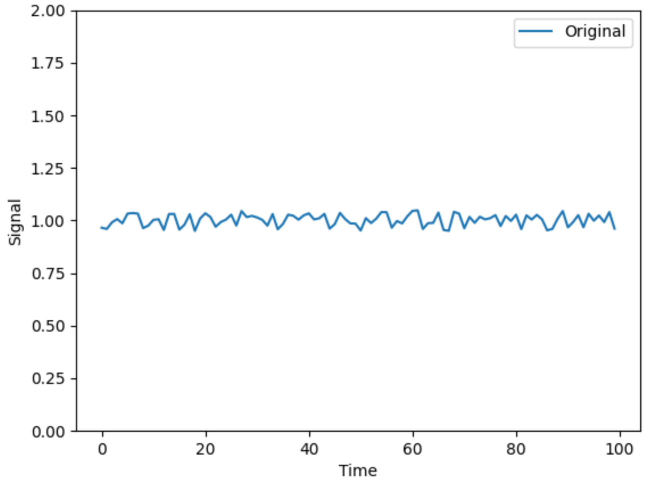
Low Pass Filter: Smoothing

- Moving Average or Window Filter

Low Pass Filter: Smoothing

- Moving Average or Window Filter
  - $y_t = \frac{x_t + \cdots + x_{t-n}}{n}$

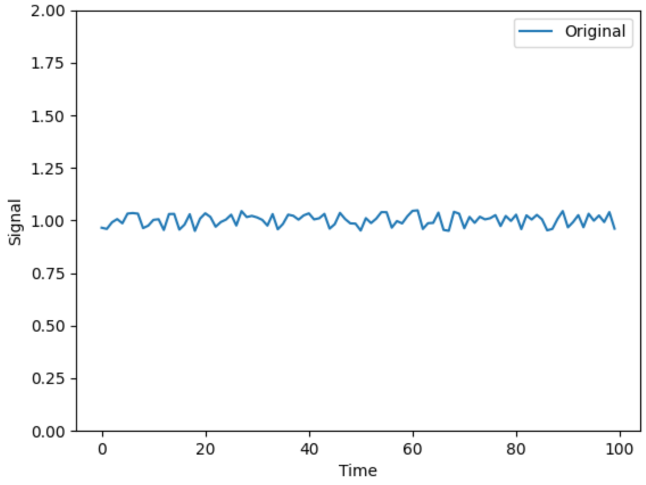
x	1.01	0.99	1.03	1.00	0.98	0.99
y						



Low Pass Filter: Smoothing

- Moving Average or Window Filter
  - $y_t = \frac{x_t + \cdots + x_{t-n}}{n}$

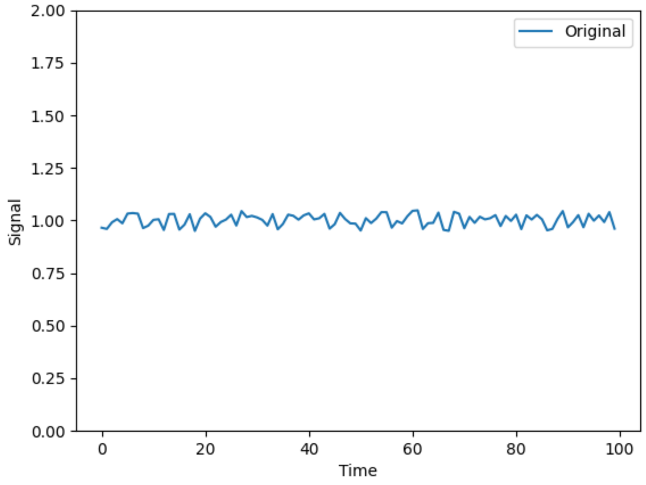
x	$\boxed{1.01}$	0.99	1.03	1.00	0.98	0.99
y	1.01					



Low Pass Filter: Smoothing

- Moving Average or Window Filter
  - $y_t = \frac{x_t + \cdots + x_{t-n}}{n}$

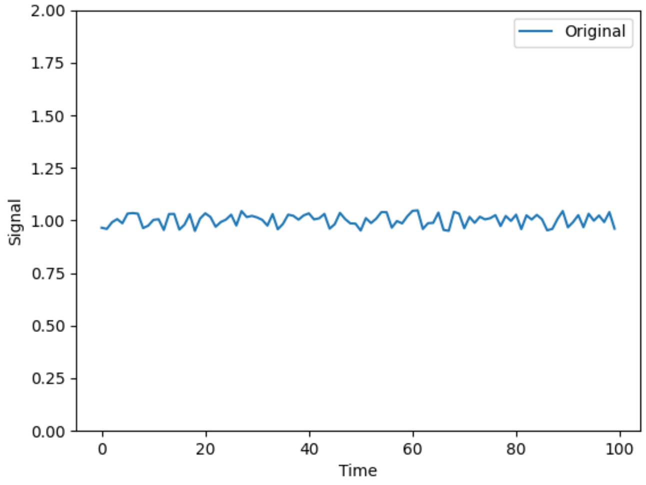
x	1.01	0.99	1.03	1.00	0.98	0.99
y	1.01	1.00				



Low Pass Filter: Smoothing

- Moving Average or Window Filter
  - $y_t = \frac{x_t + \cdots + x_{t-n}}{n}$

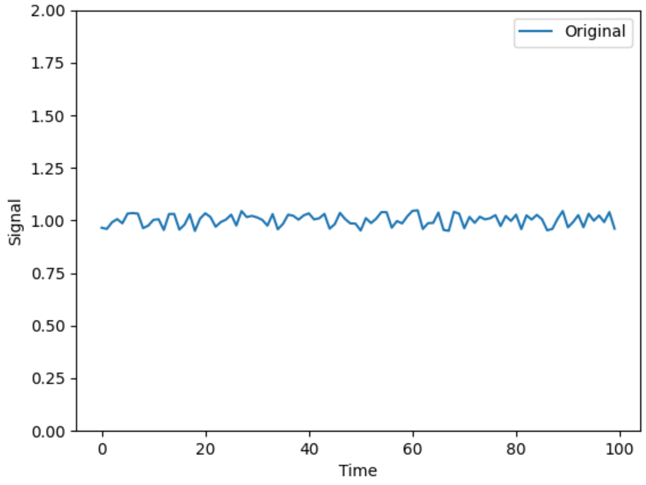
x	1.01	0.99	1.03	1.00	0.98	0.99
y	1.01	1.00	1.01			



Low Pass Filter: Smoothing

- Moving Average or Window Filter
  - $y_t = \frac{x_t + \cdots + x_{t-n}}{n}$

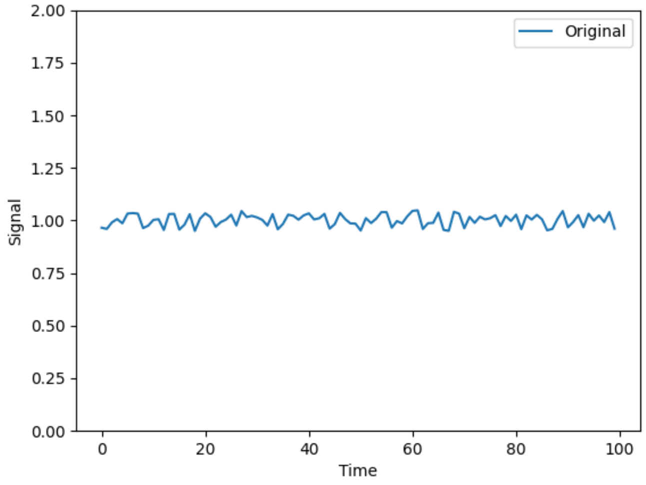
x	1.01	0.99	1.03	1.00	0.98	0.99
y	1.01	1.00	1.01			



Low Pass Filter: Smoothing

- Moving Average or Window Filter
  - $y_t = \frac{x_t + \cdots + x_{t-n}}{n}$

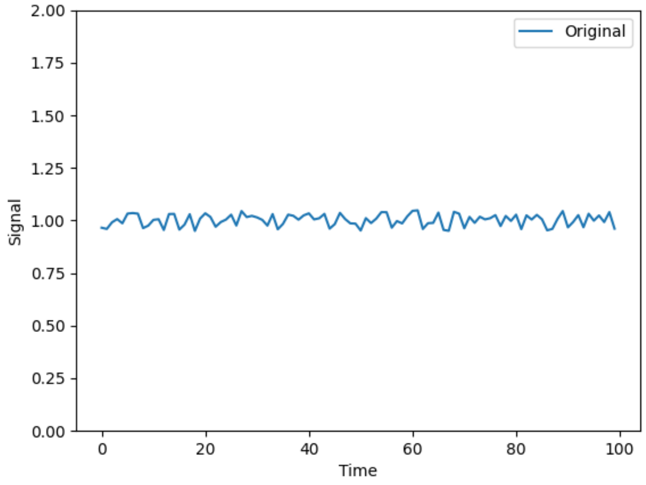
x	1.01	0.99	1.03	1.00	0.98	0.99
y	1.01	1.00	1.01	1.01		



Low Pass Filter: Smoothing

- Moving Average or Window Filter
  - $y_t = \frac{x_t + \cdots + x_{t-n}}{n}$

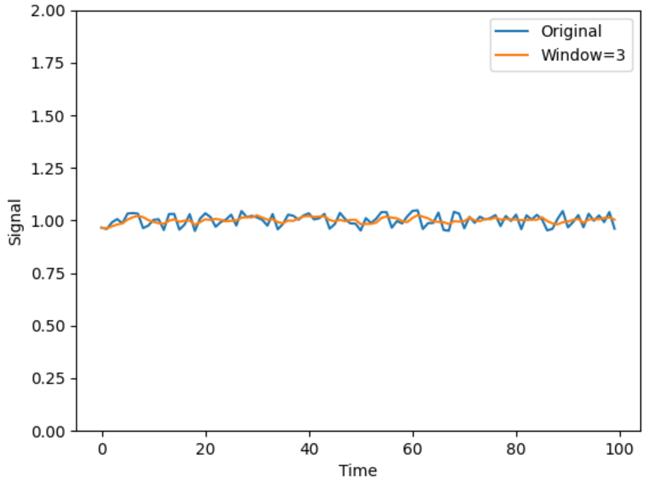
x	1.01	0.99	1.03	1.00	0.98	0.99
y	1.01	1.00	1.01	1.01	1.00	



Low Pass Filter: Smoothing

- Moving Average or Window Filter
  - $y_t = \frac{x_t + \cdots + x_{t-n}}{n}$

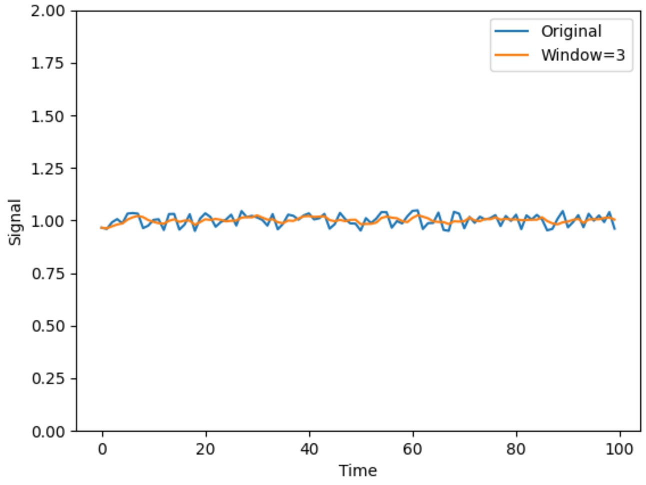
x	1.01	0.99	1.03	1.00	0.98	0.99
y	1.01	1.00	1.01	1.01	1.00	0.99



Low Pass Filter: Smoothing

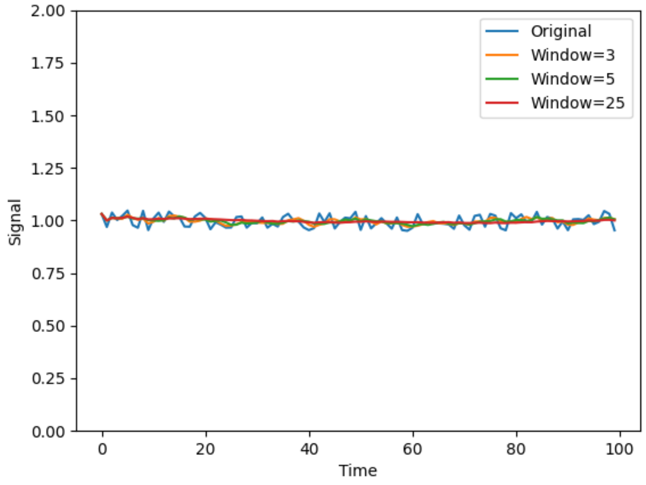
- Moving Average or Window Filter
  - $y_t = \frac{x_t + \cdots + x_{t-n}}{n}$

x	1.01	0.99	1.03	$\boxed{1.00 \quad 0.98 \quad 0.99}$		
y	1.01	1.00	1.01	1.01	1.00	0.99



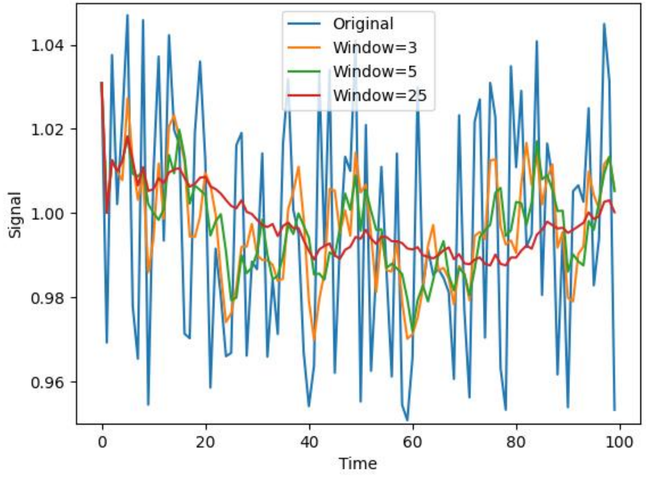
Low Pass Filter: Smoothing

- Moving Average or Window Filter
  - $y_t = \frac{x_t + \cdots + x_{t-n}}{n}$
  - Larger Window is More Smoothing



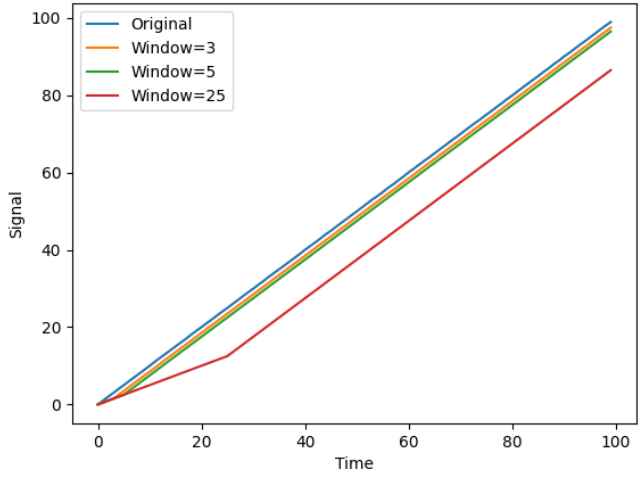
Low Pass Filter: Smoothing

- Moving Average or Window Filter
  - $y_t = \frac{x_t + \cdots + x_{t-n}}{n}$
  - Larger Window is More Smoothing



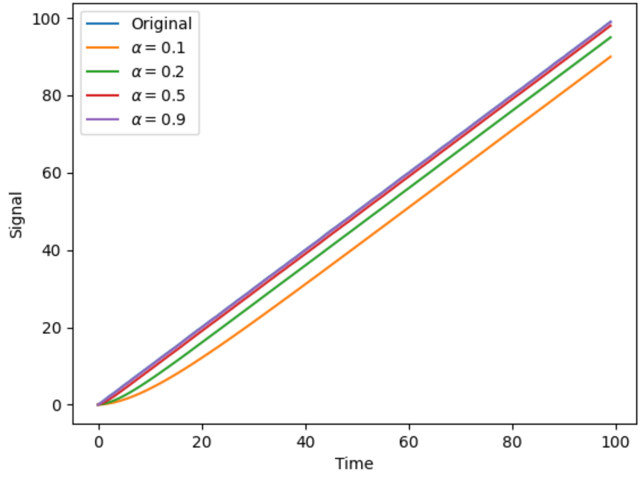
Low Pass Filter: Smoothing

- Moving Average or Window Filter
  - $y_t = \frac{x_t + \cdots + x_{t-n}}{n}$
  - Larger Window is More Smoothing
  - Larger Window is More Lag



Low Pass Filter: Smoothing

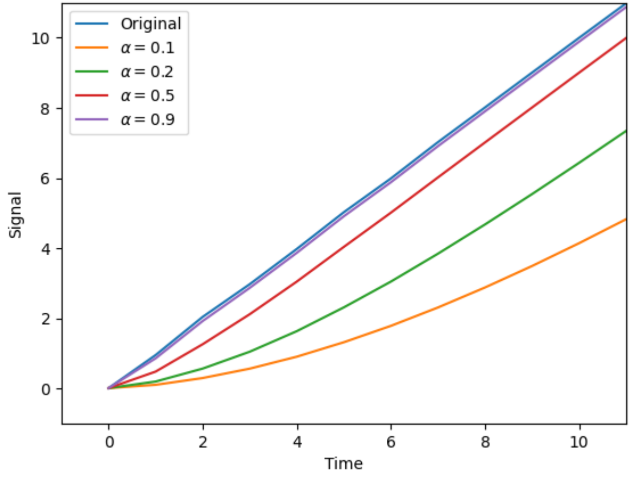
- Moving Average or Window Filter
  - $y_t = \frac{x_t + \cdots + x_{t-n}}{n}$
- Generalize by weighting
  - Linear
  - Exponential decay
  - Important to tune!
    - α closer to 1, less filtering
    - α closer to 0, more filtering



$$y_t = (1 - \alpha) \cdot y_{t-1} + \alpha \cdot x_t$$

Low Pass Filter: Smoothing

- Moving Average or Window Filter
  - $y_t = \frac{x_t + \cdots + x_{t-n}}{n}$
- Generalize by weighting
  - Linear
  - Exponential decay
  - Important to tune!
    - α closer to 1, less filtering
    - α closer to 0, more filtering



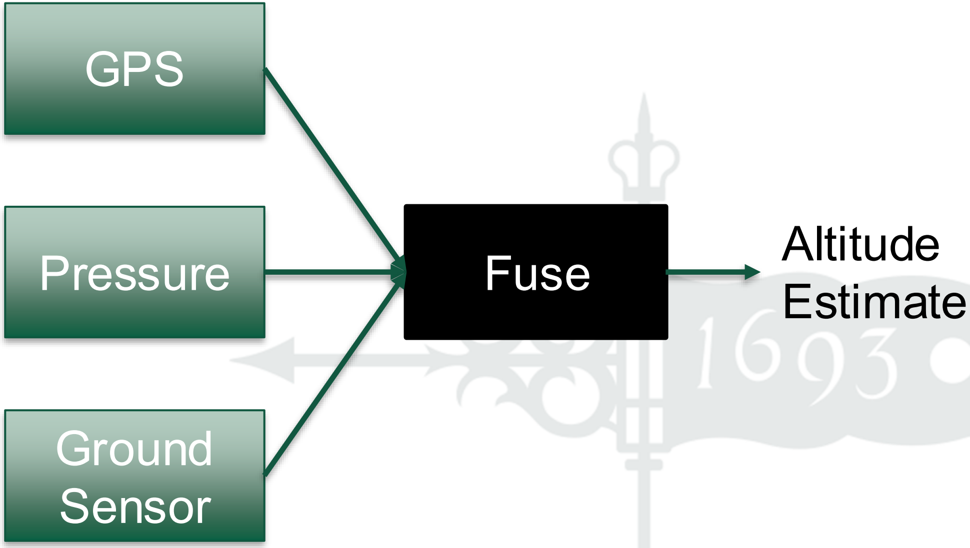
$$y_t = (1 - \alpha) \cdot y_{t-1} + \alpha \cdot x_t$$

Sensor Fusion

- Use multiple sources of data:
  - More data means less noise
  - Different operating profiles
    - Redundancy
      - One sensor fails, operation continues
    - Robustness
      - Camera requires ambient light, LiDAR does not

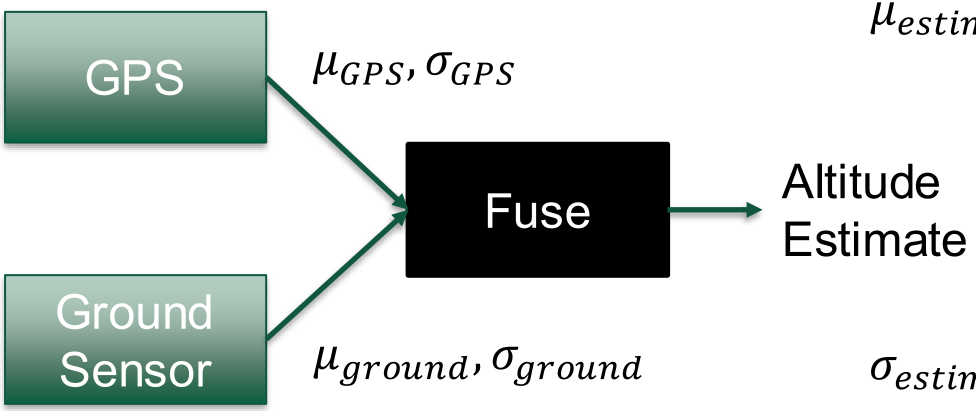
Sensor Fusion

- How do we fuse?
  - Average
    - Weight by
      - uncertainty
      - reliability
      - conditions
  - Kalman Filter



Sensor Fusion

- Fusing Like Data: Weight by Uncertainty

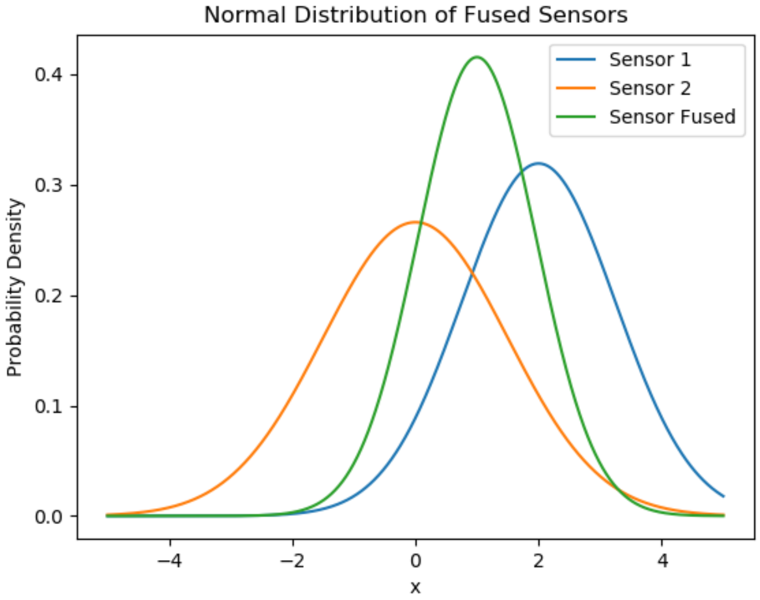


$$\mu_{estimate} = \frac{\sigma_{ground}^2 \mu_{GPS} + \sigma_{GPS}^2 \mu_{ground}}{\sigma_{ground}^2 + \sigma_{GPS}^2}$$

$$\sigma_{estimate} = \sqrt{\frac{1}{\frac{1}{\sigma_{GPS}^2} + \frac{1}{\sigma_{ground}^2}}}$$

Sensor Fusion

- Fusing Like Data: Weight by Uncertainty



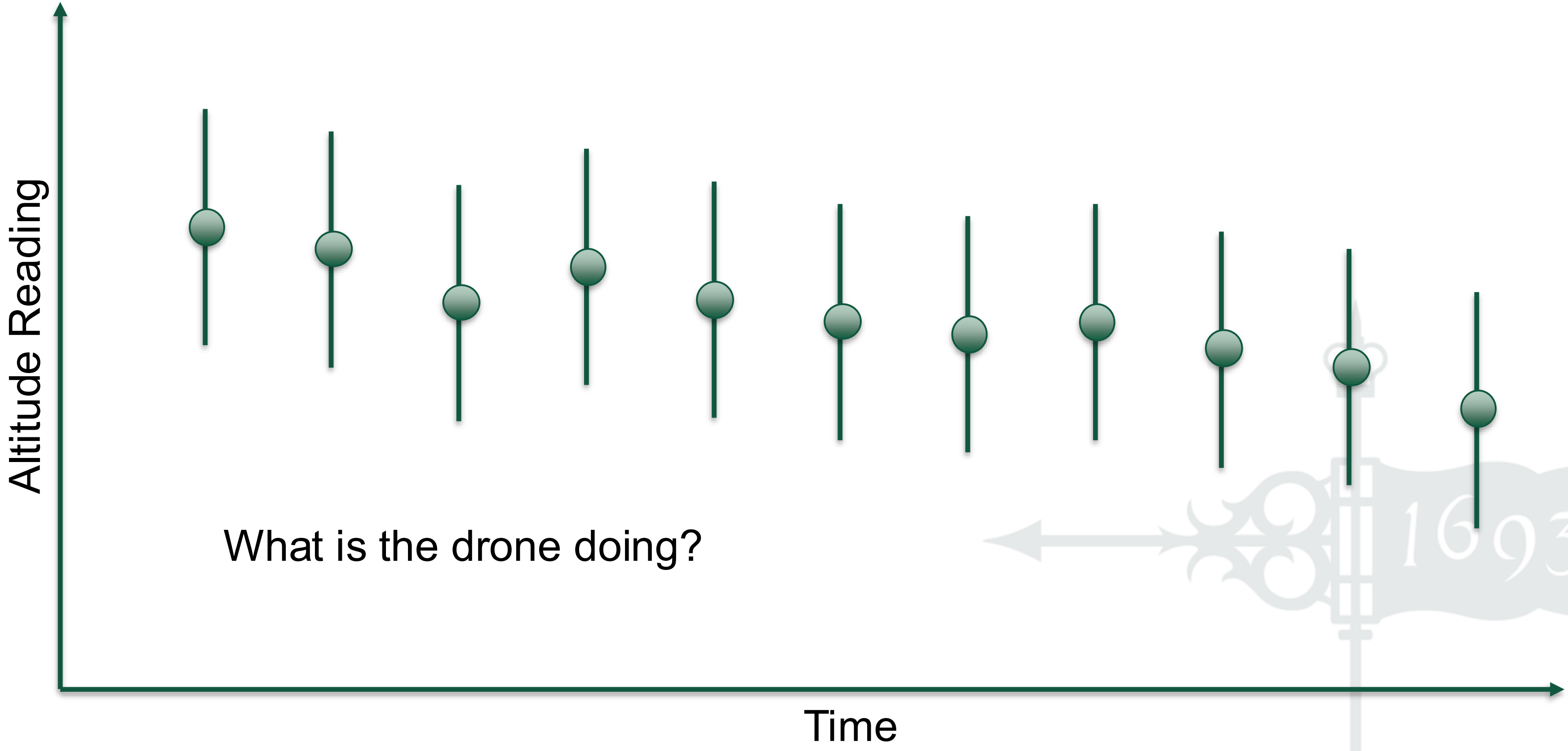
$$\mu_{estimate} = \frac{\sigma_{ground}^2 \mu_{GPS} + \sigma_{GPS}^2 \mu_{ground}}{\sigma_{ground}^2 + \sigma_{GPS}^2}$$

ltitude
stima

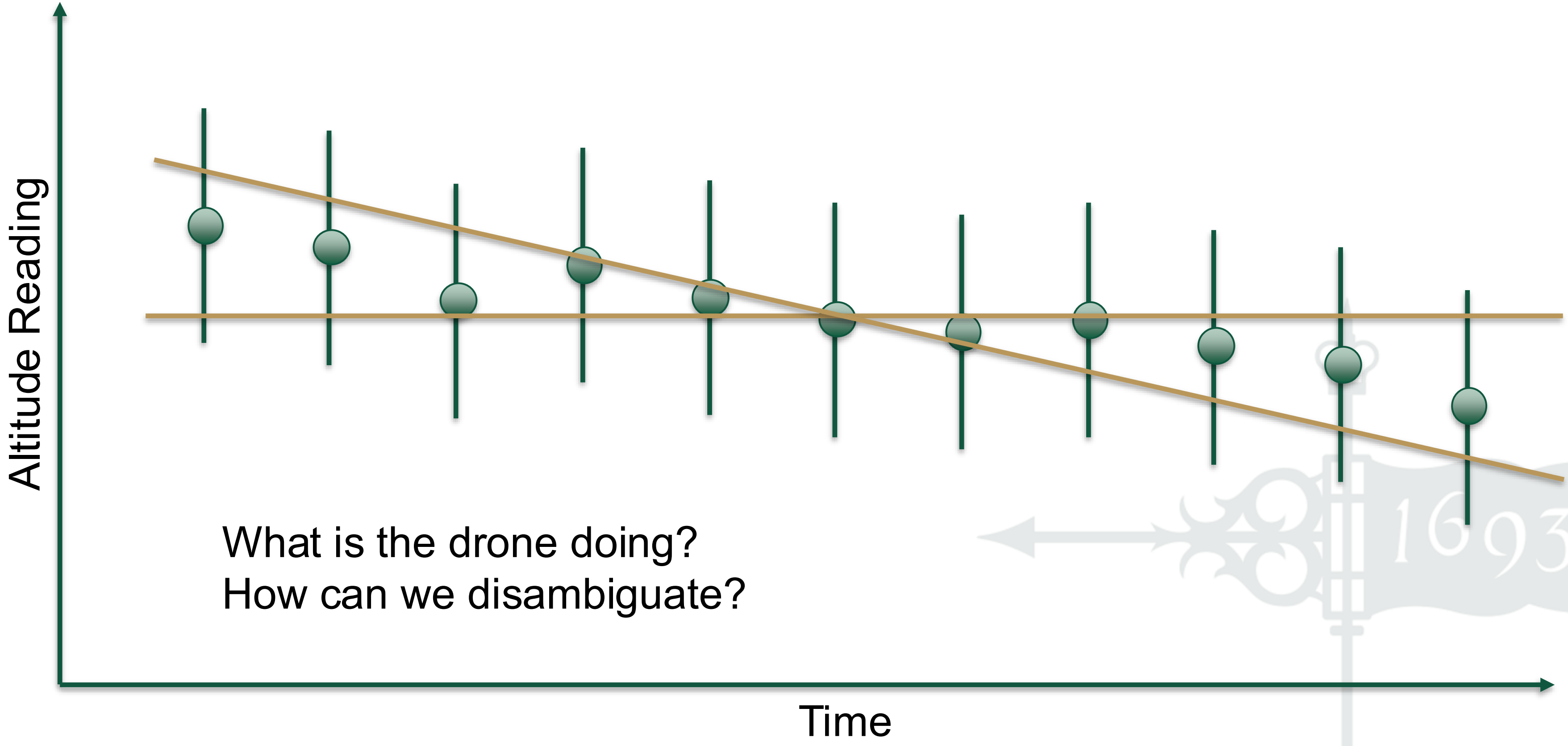
Mean is between estimates
Uncertainty is lower than either

$$\sigma_{estimate} = \sqrt{\frac{1}{\frac{1}{\sigma_{GPS}^2} + \frac{1}{\sigma_{ground}^2}}}$$

Sensor Fusion



Sensor Fusion

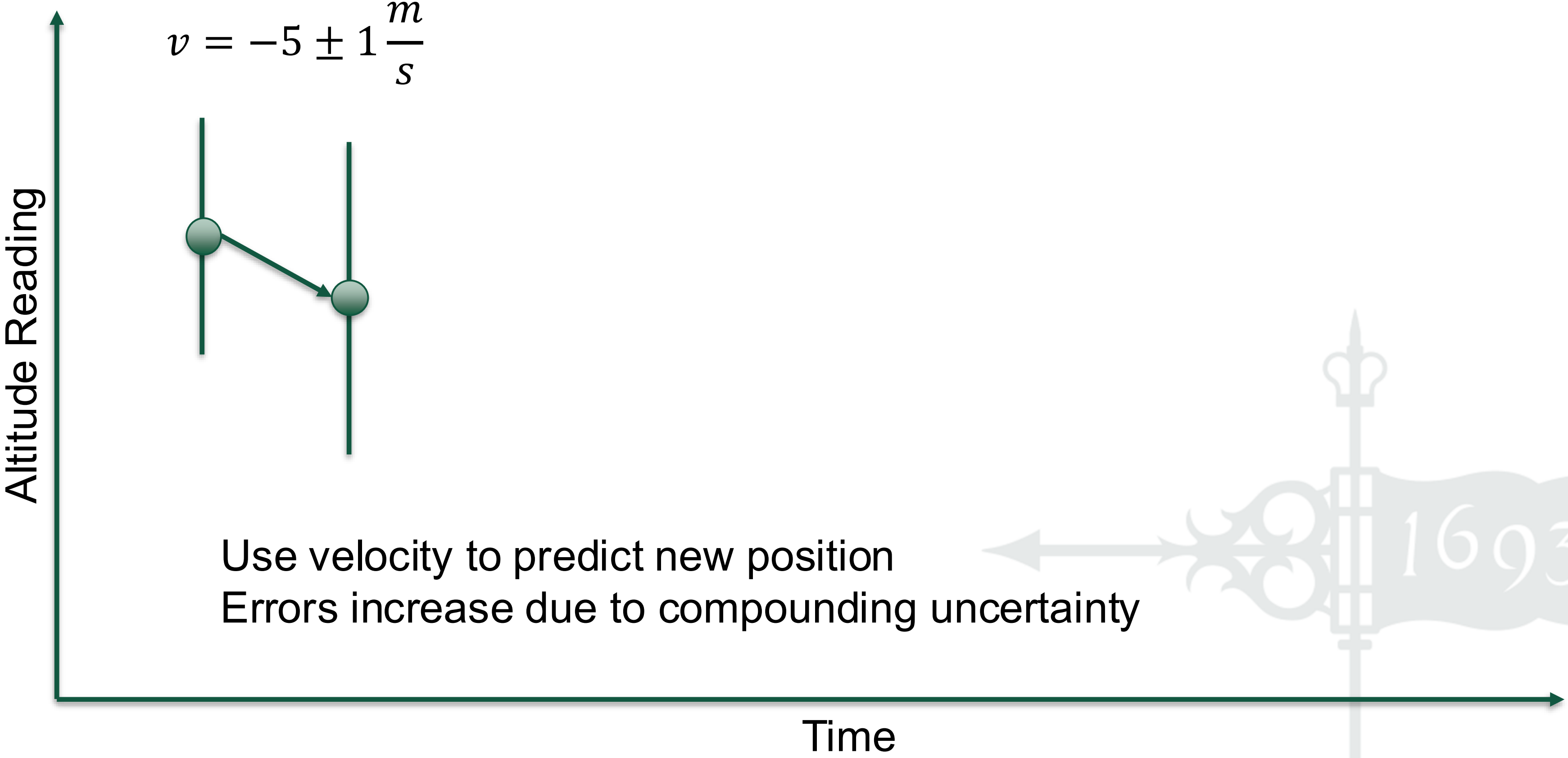


What is the drone doing?
How can we disambiguate?

Sensor Fusion: Kalman

- Fusing disparate data: Kalman Filter
  - Canonical example:
    - Fuse position + velocity to estimate position

Sensor Fusion

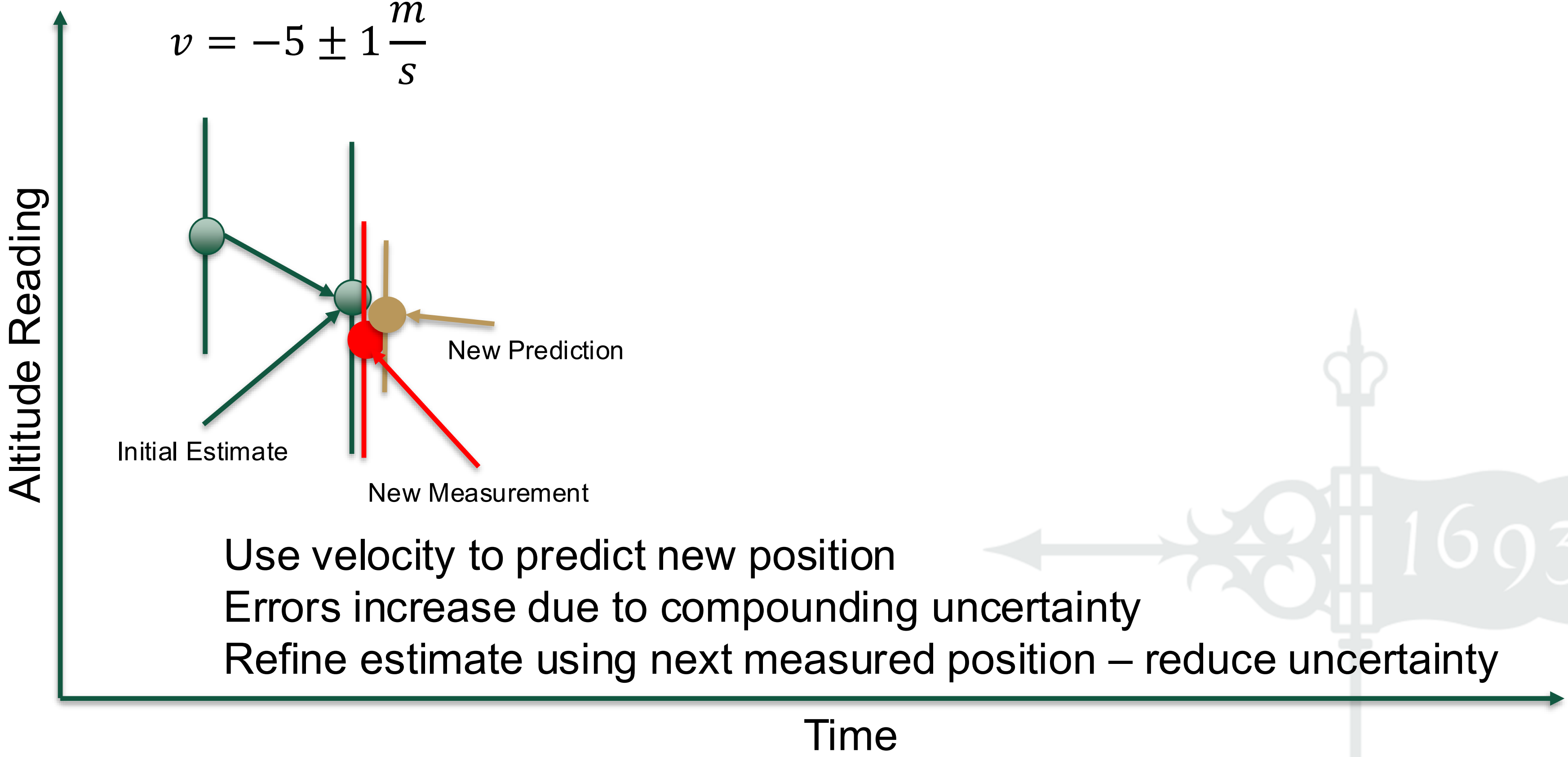


$$v = -5 \pm 1\,\frac{m}{s}$$

Use velocity to predict new position
Errors increase due to compounding uncertainty

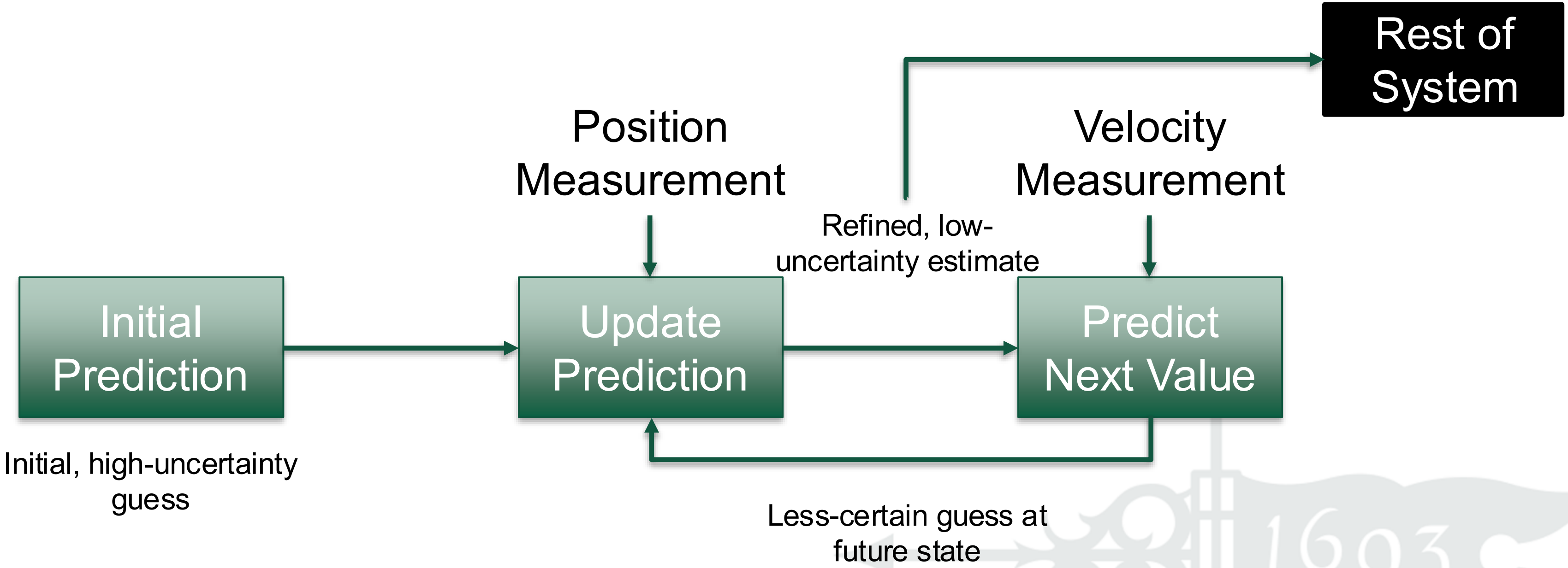
Sensor Fusion

$$v = -5 \pm 1\,\frac{m}{s}$$



Use velocity to predict new position
Errors increase due to compounding uncertainty
Refine estimate using next measured position – reduce uncertainty

Sensor Fusion: Kalman



You'll develop a version of this in Lab 4
See lab documents for equations

Sensors and Noise Handling

- Sensors capture robot and world state
- We often measure something other than what we want to know
- All sensors have noise, imperfections, uncertainty
  - Calibration
  - Filtering
  - Fusion