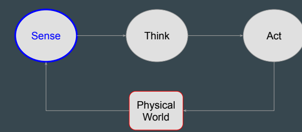
CS4501
Robotics for Soft Eng
...

Sensors and Noise Management

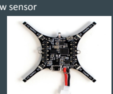
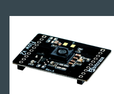


Sensor

- Transduces energy into measure
- Measures a physical quantity (light, force, speed, …)
- Provides window into the world and robot

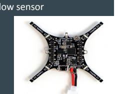
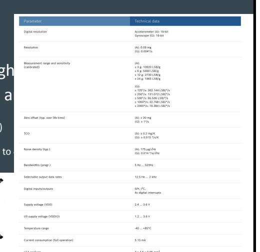
Sensor

- Transduces energy
- Measures a physical quantity (light, force, speed, …)
- Provides window into the world and oneself
    - BMI088: 3 axis accelerometer / gyroscope ()
    - BMP388: high precision pressure sensor
    - VL53L1x ToF sensor to measure distance up to 4 meters
    - PMW3901 optical flow sensor

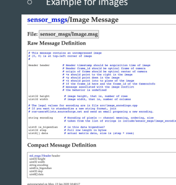
Sensor

- Transduces energy
- Measures a physical quantity (ligh
- Provides window into the world a
    - BMI088: 3 axis accelerometer / gyroscope ()
    - BMP388: high precision pressure sensor
    - VL53L1x ToF sensor to measure distance up to
    - PMW3901 optical flow sensor

Sensors in ROS

- Component support for
    - Range finders
    - Cameras
    - Audio
    - Force
    - Pose
    - Power
    - …
- Sensor messages
- Sensor messages
    - Example for images



Sensor Classification

- Proprioceptive (internal state) - sense of itself
  - Measures values internally to the system
  - Battery level, wheel position, gyro
- Exteroceptive (external state) - sense the world
  - Observations of environment
  - Compass, cameras, lidars

Sensor Classification

- Proprioceptive (internal state)
  - Measures values internally to the system
  - Battery level, wheel position, gyro
- Exteroceptive (external state)
  - Observations of environment
  - Compass, cameras, lidars

- Active (emits energy)
  - Optical encoder
  - Radar
- Passive (passively receives energy)
  - Camera
  - Bump

Sensor Classification

- Proprioceptive (internal state)
  - Measures values internally to the system
  - Battery level, wheel position, gyro
- Exteroceptive (external state)
  - Observations of environment
  - Compass

- Active (emits energy)
  - Optical encoder
  - Radar
- Passive (passively receives energy)
  - Camera
  - Bump

Question: Examples of inter, external, active, passive in your body?

Sensor Classification

- Proprioceptive (internal state)
  - Measures values internally to the system
  - Battery level, wheel position, gyro
- Exteroceptive (external state)
  - Observations of environment
  - Compass

- Active (emits energy)
  - Optical encoder
  - Radar
- Passive (passively receives energy)
  - Camera
  - Bump

Question: how the Crazyflie sensors?

- BMI088: 3 axis accelerometer / gyroscope (): Pr/Pa
- BMP388: high precision pressure sensor: E/Pa
- VL53L1x ToF sensor to measure distance up to 4 meters: E/Ac
- PMW3901 optical flow sensor: E/Pa

Sensor Classification

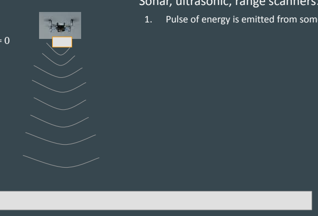
- Proprioceptive (internal state)
  - Measures values internally to the system
  - Battery level, wheel position, gyro
- Exteroceptive (external state)
  - Observations of environment
  - Compass

- Active (emits energy)
  - Optical encoder
  - Radar
- Passive (passively receives energy)
  - Camera
  - Bump

Question: how the Crazyflie sensors?

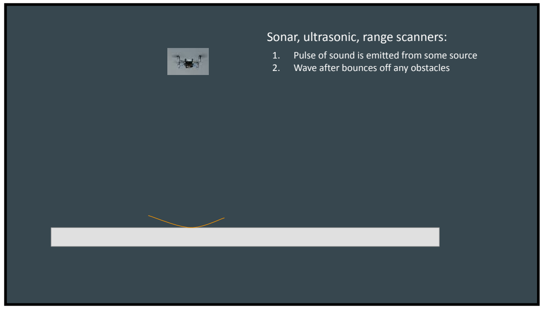
- BMI088: 3 axis accelerometer / gyroscope (): Pr/Pa
- BMP388: high precision pressure sensor: E/Pa
- VL53L1x ToF sensor to measure distance up to 4 meters: E/Ac
- PMW3901 optical flow sensor: E/Pa

T = 0

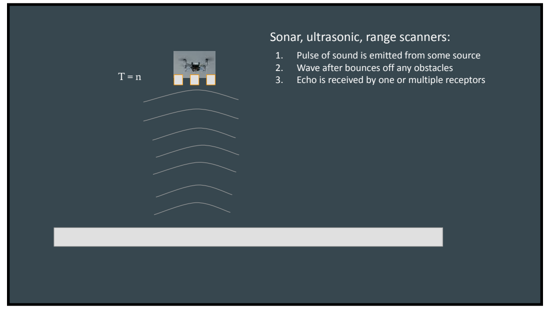


Sonar, ultrasonic, range scanners:

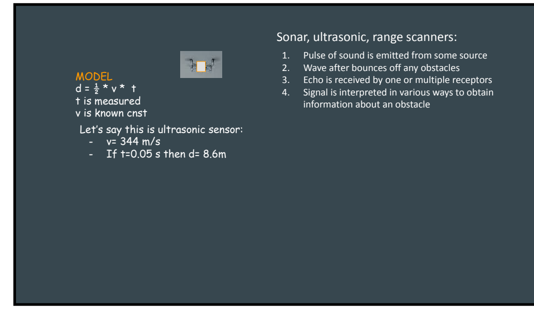
1. Pulse of energy is emitted from some source



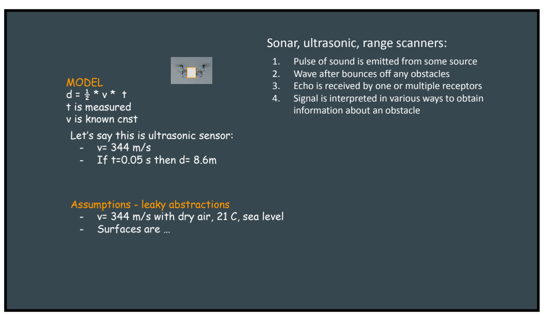
Sonar, ultrasonic, range scanners:
1. Pulse of sound is emitted from some source
2. Wave after bounces off any obstacles



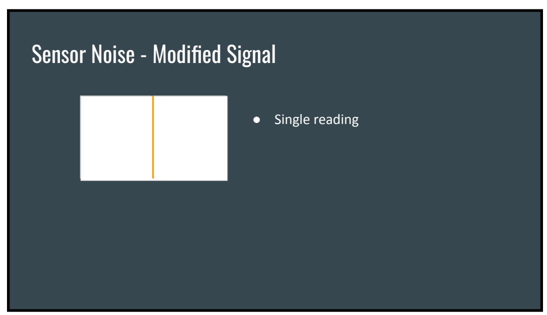
T = n
Sonar, ultrasonic, range scanners:
1. Pulse of sound is emitted from some source
2. Wave after bounces off any obstacles
3. Echo is received by one or multiple receptors



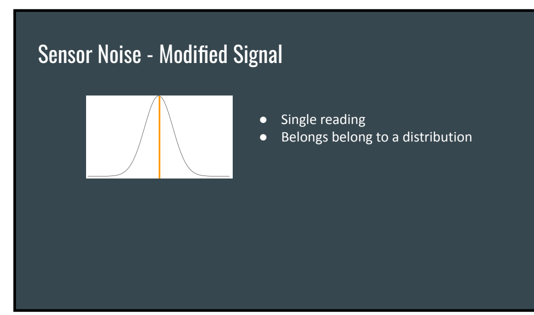
MODEL
d = ½ * v * t
t is measured
v is known cnst
Let's say this is ultrasonic sensor:
- v= 344 m/s
- If t=0.05 s then d= 8.6m
Sonar, ultrasonic, range scanners:
1. Pulse of sound is emitted from some source
2. Wave after bounces off any obstacles
3. Echo is received by one or multiple receptors
4. Signal is interpreted in various ways to obtain information about an obstacle



MODEL
d = ½ * v * t
t is measured
v is known cnst
Let's say this is ultrasonic sensor:
- v= 344 m/s
- If t=0.05 s then d= 8.6m
Assumptions - leaky abstractions
- v= 344 m/s with dry air, 21 C, sea level
- Surfaces are ...
Sonar, ultrasonic, range scanners:
1. Pulse of sound is emitted from some source
2. Wave after bounces off any obstacles
3. Echo is received by one or multiple receptors
4. Signal is interpreted in various ways to obtain information about an obstacle

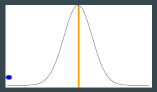


Sensor Noise - Modified Signal
Single reading



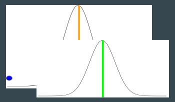
Sensor Noise - Modified Signal
Single reading
Belongs belong to a distribution

Sensor Noise - Modified Signal



- Single reading
- Belong belong to a distribution
- Outliers

Sensor Noise - Modified Signal



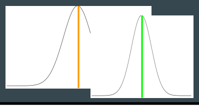
- Single reading
- Belong belong to a distribution
- Outliers
- Shifts

Managing Sensor Noise

- Calibration
- Filtering
- Fusing

Calibration

- Shifts in distribution due to environmental assumptions
- Adjusting sensor for more accurate physical measurements within context
- Process
  a. Conduct standardized tests
  b. Recompute constants and error estimates
  c. Redefine model parameters



Calibration Problem

MODEL
d = ½ * v * t
t is measured
v is known cnst

Let's say this is ultrasonic sensor:
- v= 344 m/s
- If t=0.05 seconds, then d= 8.6m

Assumptions
- v= 344 m/s with dry air, 21 C, sea level
- Surfaces are …
- When assumptions break, measures are off!

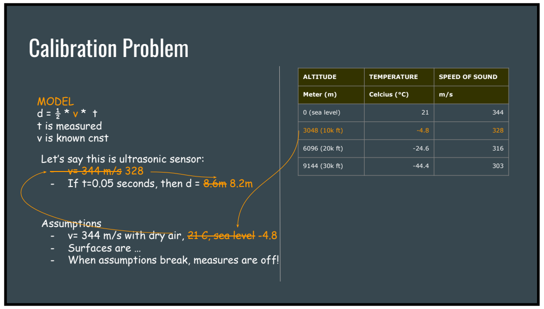
Calibration Problem

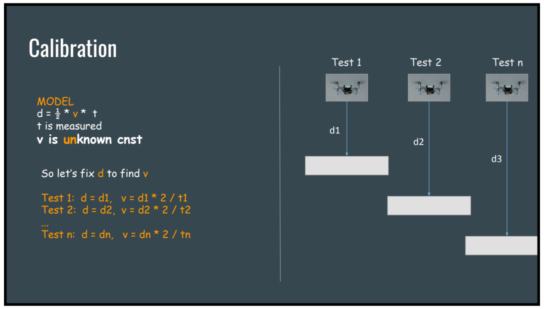
MODEL
d = ½ * v * t
t is measured
v is known cnst

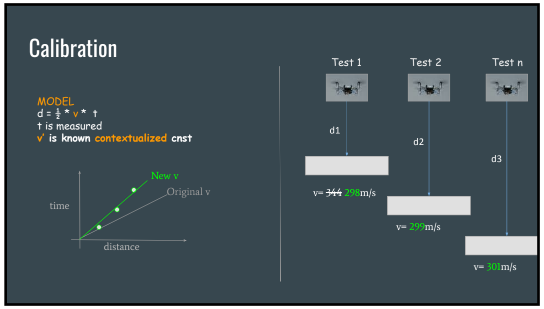
Let's say this is ultrasonic sensor:
- v= 344 m/s
- If t=0.05 seconds, then d= 8.6m

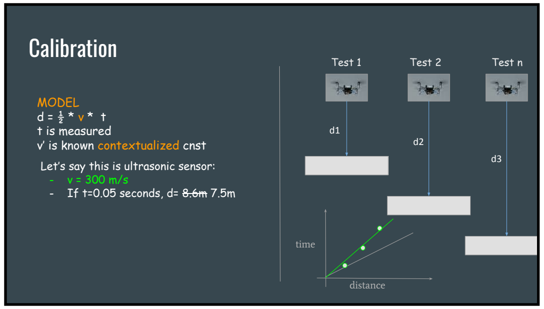
Assumptions
- v= 344 m/s with dry air, ~~21 C, sea level~~ -4.8
- Surfaces are …
- When assumptions break, measures are off!

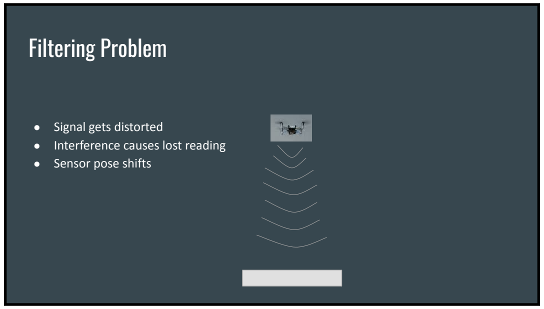
ALTITUDE	TEMPERATURE	SPEED OF SOUND
Meter (m)	Celcius (°C)	m/s
0 (sea level)	21	344
3048 (10k ft)	-4.8	328
6096 (20k ft)	-24.6	316
9144 (30k ft)	-44.4	303

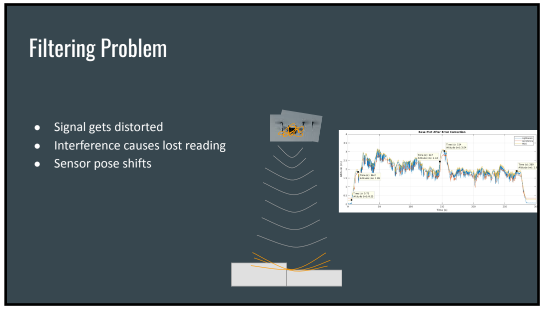

Calibration Problem
MODEL
d = ½ * v * t
t is measured
v is known cnst
Let's say this is ultrasonic sensor:
- v= 344 m/s 328
- If t=0.05 seconds, then d = 8.6m 8.2m
Assumptions
- v= 344 m/s with dry air, 21 C, sea level -4.8
- Surfaces are ...
- When assumptions break, measures are off!
ALTITUDE | TEMPERATURE | SPEED OF SOUND
Meter (m) | Celcius (°C) | m/s
0 (sea level) | 21 | 344
3048 (10k ft) | -4.8 | 328
6096 (20k ft) | -24.6 | 316
9144 (30k ft) | -44.4 | 303


Calibration
MODEL
d = ½ * v * t
t is measured
v is unknown cnst
So let's fix d to find v
Test 1: d = d1, v = d1 * 2 / t1
Test 2: d = d2, v = d2 * 2 / t2
...
Test n: d = dn, v = dn * 2 / tn
Test 1
Test 2
Test n
d1
d2
d3


Calibration
MODEL
d = ½ * v * t
t is measured
v' is known contextualized cnst
time
distance
New v
Original v
Test 1
Test 2
Test n
d1
d2
d3
v= 344 298m/s
v= 299m/s
v= 301m/s

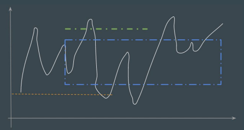

Calibration
MODEL
d = ½ * v * t
t is measured
v' is known contextualized cnst
Let's say this is ultrasonic sensor:
- v = 300 m/s
- If t=0.05 seconds, d= 8.6m 7.5m
Test 1
Test 2
Test n
d1
d2
d3
time
distance


Filtering Problem
● Signal gets distorted
● Interference causes lost reading
● Sensor pose shifts


Filtering Problem
● Signal gets distorted
● Interference causes lost reading
● Sensor pose shifts

Basic Filters from Signal Processing

- Low-pass filters
- High-pass filters
- Band filters



Filtering as smoothing

- Moving averages

$$Y_t = (X_t + X_{t-1} + X_{t-2} + \dots + X_{t-window}) / windowSize$$

Filtering as smoothing

- Moving averages

$$Y_t = (X_t + X_{t-1} + X_{t-2} + \dots + X_{t-window}) / windowSize$$

… 64 65 64 78 65 66 67 64 …

64

windowSize = 3

Filtering as smoothing

- Moving averages

$$Y_t = (X_t + X_{t-1} + X_{t-2} + \dots + X_{t-window}) / windowSize$$

… 64 65 64 78 65 66 67 64 …

64 69

windowSize = 3

Filtering as smoothing

- Moving averages

$$Y_t = (X_t + X_{t-1} + X_{t-2} + \dots + X_{t-window}) / windowSize$$

… 64 65 64 78 65 66 67 64 …

64 69 69

windowSize = 3

Filtering as smoothing

- Moving averages

$$Y_t = (X_t + X_{t-1} + X_{t-2} + \dots + X_{t-window}) / windowSize$$

… 64 65 64 78 65 66 67 64 …

64 69 69 70

windowSize = 3

Filtering as smoothing

- Moving averages

  Yt = (Xt + Xt-1 + Xt-2 + … + Xt-window) / windowSize

… 64 65 64 78 65 66 67 64 …

67

windowSize = 5

Larger windows stronger smoothing

Filtering as smoothing

- Moving averages

  Yt = (Xt + Xt-1 + Xt-2 + … + Xt-window) / windowSize

- Generalized with weights for decaying

  Yt = a Xt + a1 Xt-1 + a2 Xt-2 + a3 Xt-3 + … where a+a1+a2+a… = 1

Filtering as smoothing

- Moving averages

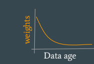
  Yt = (Xt + Xt-1 + Xt-2 + … + Xt-window) / windowSize

- Generalized with weights for decaying

  Yt = a Xt + a1 Xt-1 + a2 Xt-2 + a3 Xt-3 + … where a+a1+a2+a… = 1

- Generalized with exponential weights for decaying

  Yt = α [Xt + (1 - α) Xt-1 + (1 - α)*2 Xt-2 + (1 - α)*3 Xt-3 + …]

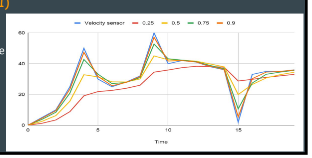


Filtering as smoothing

- Generalized with exponential weights for decaying

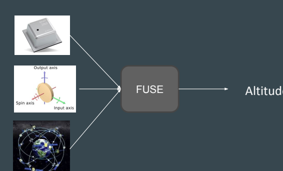
  Yt = α Xt + (1 - α) Xt-1 + (1 - α)*2 Xt-2 + (1 - α)*3 Xt-3 + …
  Or efficiently approximated
  Yt = Yt-1 + α (Xt - Yt-1)

- Selection of α is crucial
  - α closer to 0: closer to last value
  - α closer to 1: no filtering

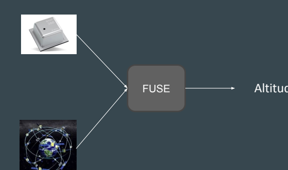


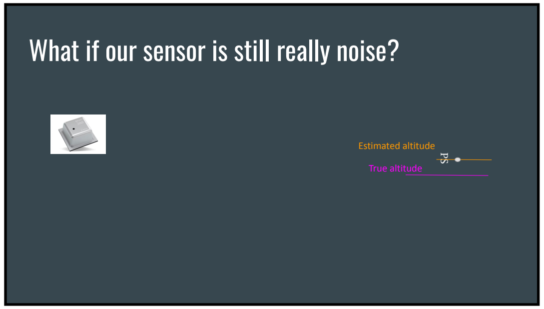
What if our sensor is still really noise?

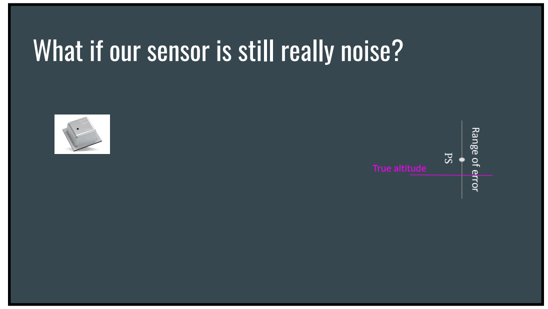
Add sensors with complementary attributes and fuse them

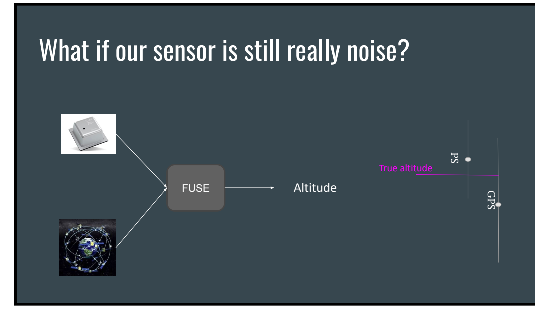


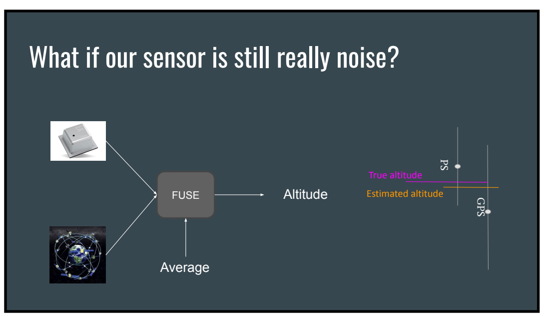
What if our sensor is still really noise?

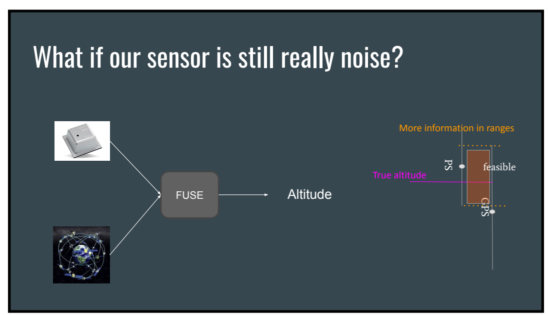


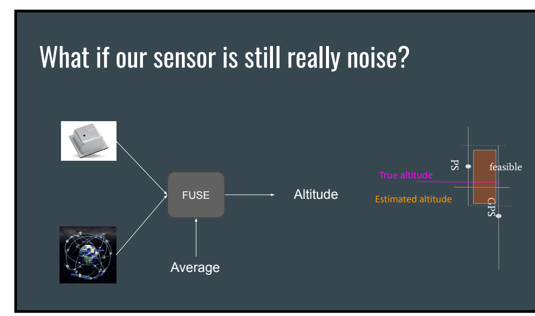

What if our sensor is still really noise?
Estimated altitude
True altitude
PS

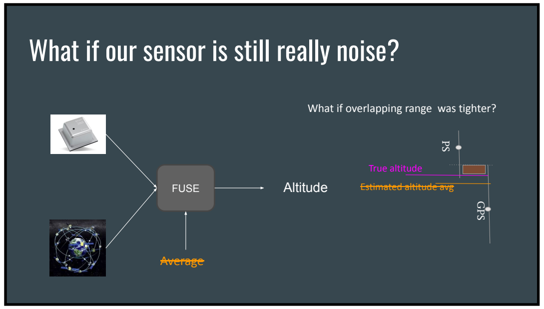

What if our sensor is still really noise?
True altitude
PS
Range of error

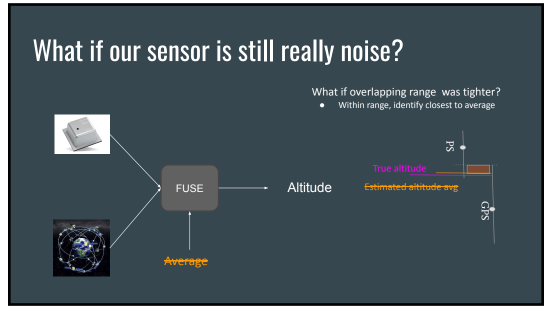

What if our sensor is still really noise?
FUSE
Altitude
True altitude
PS
GPS

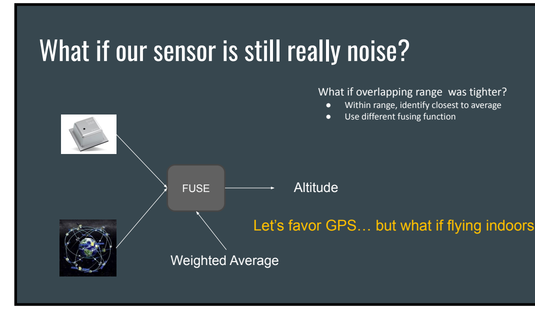

What if our sensor is still really noise?
FUSE
Altitude
Average
True altitude
Estimated altitude
PS
GPS

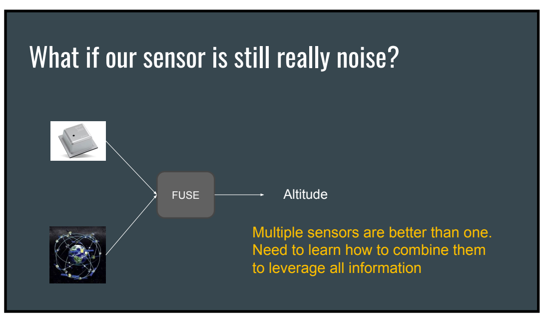

What if our sensor is still really noise?
FUSE
Altitude
More information in ranges
True altitude
PS
feasible
GPS

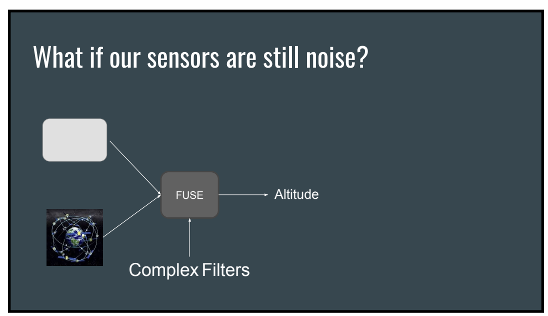

What if our sensor is still really noise?
FUSE
Altitude
Average
True altitude
Estimated altitude
PS
feasible
GPS

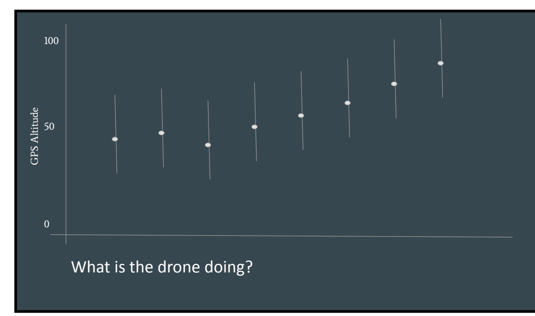

What if our sensor is still really noise?
What if overlapping range was tighter?
PS
True altitude
Estimated altitude avg
GPS
FUSE
Altitude
Average

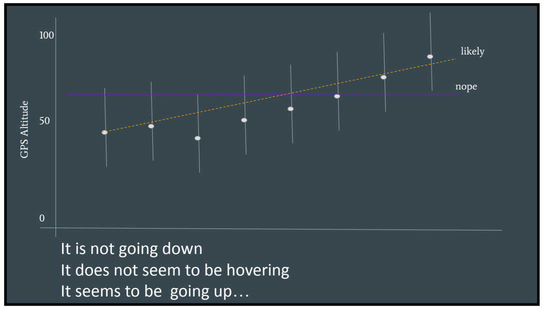

What if our sensor is still really noise?
What if overlapping range was tighter?
Within range, identify closest to average
PS
True altitude
Estimated altitude avg
GPS
FUSE
Altitude
Average

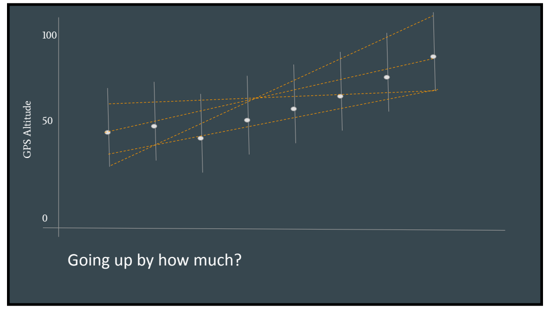

What if our sensor is still really noise?
What if overlapping range was tighter?
Within range, identify closest to average
Use different fusing function
FUSE
Altitude
Let's favor GPS… but what if flying indoors?
Weighted Average

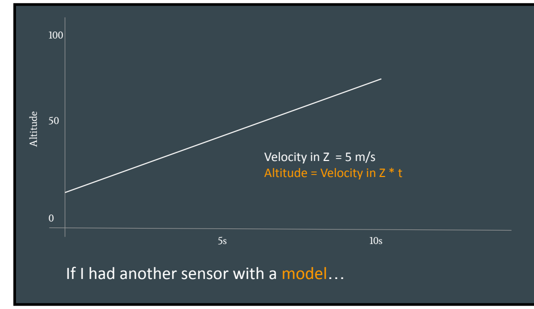

What if our sensor is still really noise?
FUSE
Altitude
Multiple sensors are better than one.
Need to learn how to combine them
to leverage all information

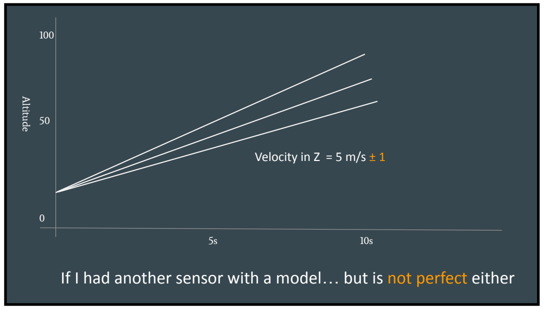

What if our sensors are still noise?
FUSE
Altitude
Complex Filters

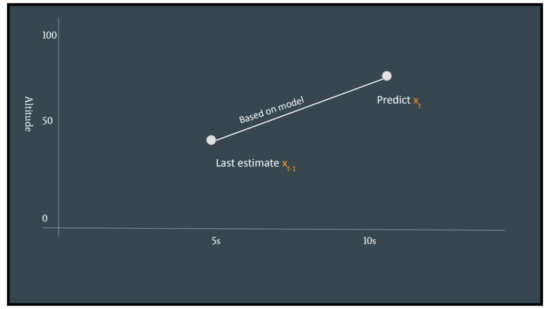

100
50
0
GPS Altitude
What is the drone doing?

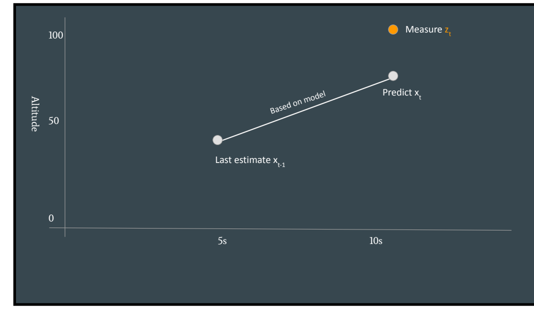

100
likely
nope
GPS Altitude
50
0
It is not going down
It does not seem to be hovering
It seems to be going up…

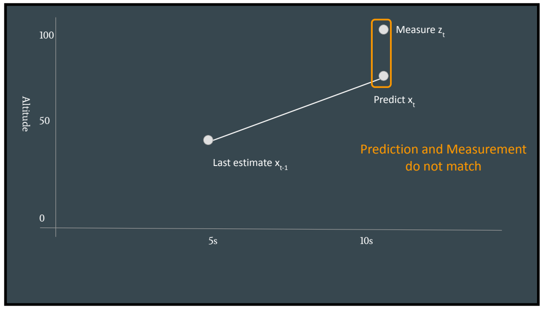

100
GPS Altitude
50
0
Going up by how much?

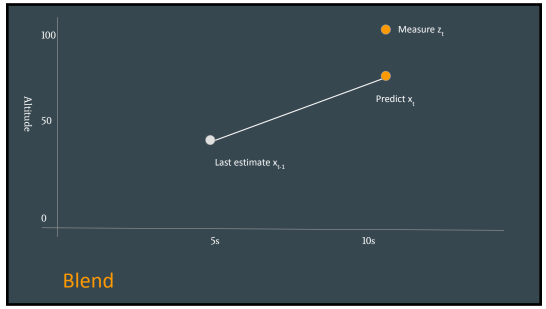

100
Altitude
50
0
Velocity in Z = 5 m/s
Altitude = Velocity in Z * t
5s
10s
If I had another sensor with a model…

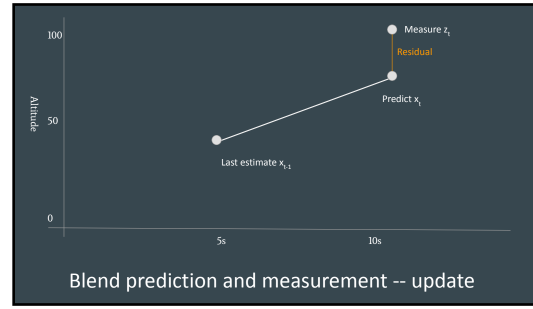

100
Altitude
50
0
Velocity in Z = 5 m/s ± 1
5s
10s
If I had another sensor with a model… but is not perfect either

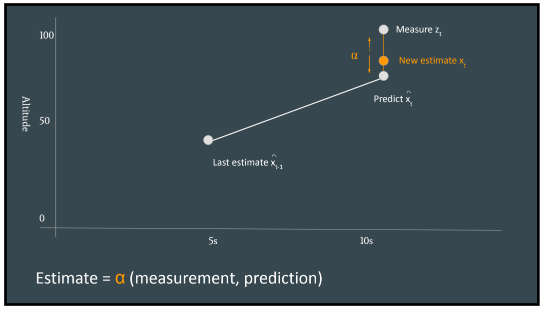

100
Altitude
50
0
Based on model
Predict x_t
Last estimate x_{t-1}
5s
10s

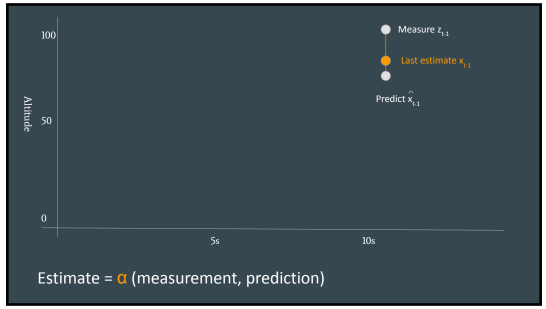

Measure z_t
100
Altitude
50
0
Based on model
Predict x_t
Last estimate x_{t-1}
5s
10s

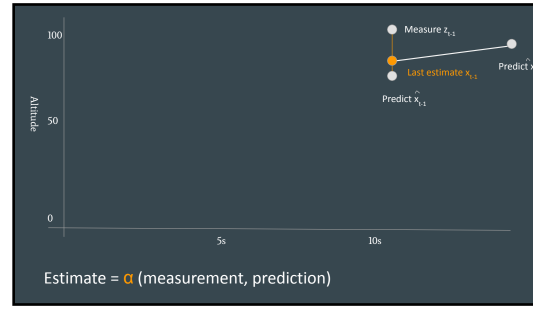

100
50
0
Altitude
Measure z_t
Predict x_t
Last estimate x_{t-1}
Prediction and Measurement do not match
5s
10s

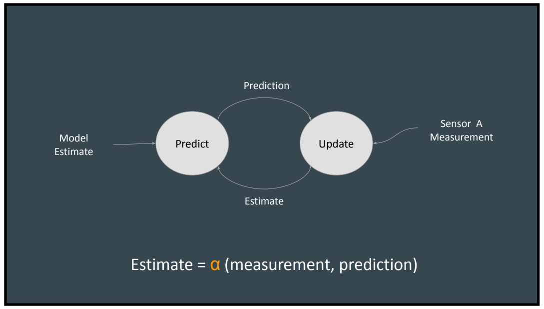

100
50
0
Altitude
Measure z_t
Predict x_t
Last estimate x_{t-1}
5s
10s
Blend

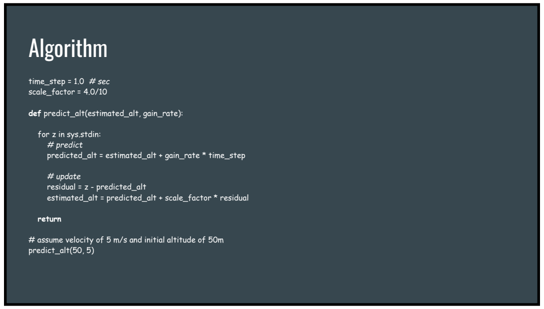

100
50
0
Altitude
Measure z_t
Residual
Predict x_t
Last estimate x_{t-1}
5s
10s
Blend prediction and measurement -- update

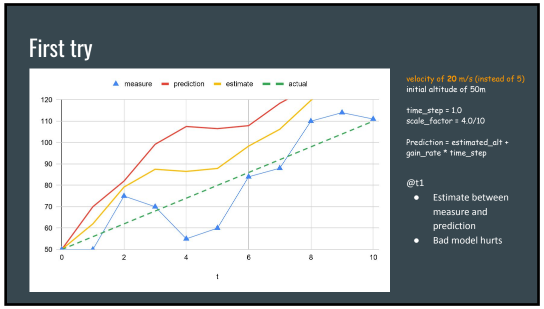

100
50
0
Altitude
Measure z_t
α
New estimate x_t
Predict $\hat{x}_t$
Last estimate $\hat{x}_{t-1}$
5s
10s
Estimate = α (measurement, prediction)

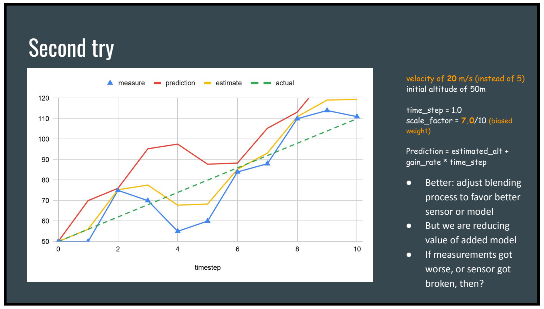

100
50
0
Altitude
Measure z_{t-1}
Last estimate x_{t-1}
Predict $\hat{x}_{t-1}$
5s
10s
Estimate = α (measurement, prediction)

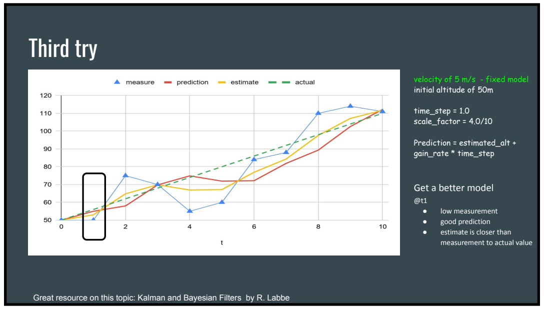

100
50
0
Altitude
Measure z_{t-1}
Predict $\hat{x}_t$
Last estimate x_{t-1}
Predict $\hat{x}_{t-1}$
5s
10s
Estimate = α (measurement, prediction)

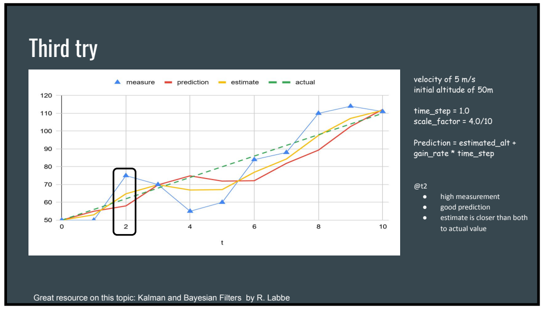

Prediction
Model Estimate
Predict
Update
Sensor A Measurement
Estimate
Estimate = α (measurement, prediction)


Algorithm
time_step = 1.0 # sec
scale_factor = 4.0/10
def predict_alt(estimated_alt, gain_rate):
for z in sys.stdin:
# predict
predicted_alt = estimated_alt + gain_rate * time_step
# update
residual = z - predicted_alt
estimated_alt = predicted_alt + scale_factor * residual
return
# assume velocity of 5 m/s and initial altitude of 50m
predict_alt(50, 5)

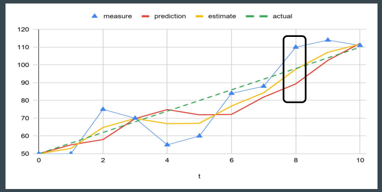

First try
measure
prediction
estimate
actual
t
velocity of 20 m/s (instead of 5)
initial altitude of 50m
time_step = 1.0
scale_factor = 4.0/10
Prediction = estimated_alt + gain_rate * time_step
@t1
Estimate between measure and prediction
Bad model hurts


Second try
measure
prediction
estimate
actual
timestep
velocity of 20 m/s (instead of 5)
initial altitude of 50m
time_step = 1.0
scale_factor = 7.0/10 (biased weight)
Prediction = estimated_alt + gain_rate * time_step
Better: adjust blending process to favor better sensor or model
But we are reducing value of added model
If measurements got worse, or sensor got broken, then?


Third try
measure
prediction
estimate
actual
t
velocity of 5 m/s - fixed model
initial altitude of 50m
time_step = 1.0
scale_factor = 4.0/10
Prediction = estimated_alt + gain_rate * time_step
Get a better model
@t1
low measurement
good prediction
estimate is closer than measurement to actual value
Great resource on this topic: Kalman and Bayesian Filters by R. Labbe


Third try
measure
prediction
estimate
actual
t
velocity of 5 m/s
initial altitude of 50m
time_step = 1.0
scale_factor = 4.0/10
Prediction = estimated_alt + gain_rate * time_step
@t2
high measurement
good prediction
estimate is closer than both to actual value
Great resource on this topic: Kalman and Bayesian Filters by R. Labbe

Third try



velocity of 5 m/s
initial altitude of 50m

time_step = 1.0
scale_factor = 4.0/10

Prediction = estimated_alt + gain_rate * time_step

@t8

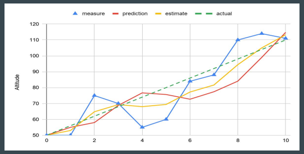
- High measurement
- Low prediction
- Estimate is closer to actual value

Great resource on this topic: Kalman and Bayesian Filters by R. Labbe

OR Revise Algorithm for adjusting model

```
time_step = 1.0
scale_factor= 4./10
gain_scale = 1./3

def predict_alt(estimated_alt, gain_rate):

    for z in sys.stdin:
        # predict
        predicted_alt = estimated_alt + gain_rate * time_step

        # update
        residual = z - predicted_alt
        # dynamically adjust gain rate according to residual changes
        gain_rate = gain_rate + gain_scale  * (residual/time_step)
        estimated_alt  =  predicted_alt  +  scale_factor * residual

    return

# assume velocity of 5 m/s and initial altitude of 50m
predict_alt(50, 5)
```

Final try



velocity of ~~5 m/s~~ Dynamic based on residual size
initial altitude of 50m

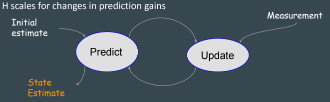
time_step = 1.0
scale_factor = 4.0/10 # H
gain_scale = 0.3 # G: How strongly to adjust predictions

Performance close as with tuned model

Much nicer… still a few magic blending numbers for G and H

G-H Filter (alpha - beta Filter)

- Predict next value & rate of change based on
  - Current estimate
  - Predict of how it will change
- Choose new scaled estimate between prediction and measurement
  - G scales measurements
  - H scales for changes in prediction gains



- Basis for Kalman filter

Takeaways

- Sensors capture data about robot and world state
- We cannot rely on sensors for perfect data
- To manage noise
  - Calibration
  - Fusion
  - Filtering