CSCI 1108

Sensors

Characterizing / Modeling

Announcements

- Today's Topics
  - Sensors
  - How to model/characterize a sensor
  - Using sensors (sampling, debouncing, etc)

- Accelerometers
- Biometric Sensors
- Cameras & Vision Sensors
- Contact & Proximity Sensors
- Current & Voltage Sensors
- Encoders & Disks
- Force Sensors
- Gas Sensors
- Gyroscopes
- Inclination & Tilt Sensors
- Inertia Measurement Units
- Infrared & Light Sensors
- LIDAR, Laser Scanners & Rangefinders
- Linear & Rotary Resistors
- Localization & GPS
- Magnetic Sensors / Compass
- Pressure Sensors
- Real-Time Clocks
- Sound Sensors
- Stretch & Bend Sensors
- Temperature & Humidity Sensors
- Thermal Array Sensors
- Ultrasonic Range Finders

Scanning Laser Range Finders and LIDAR

Ultrasonnic range finder

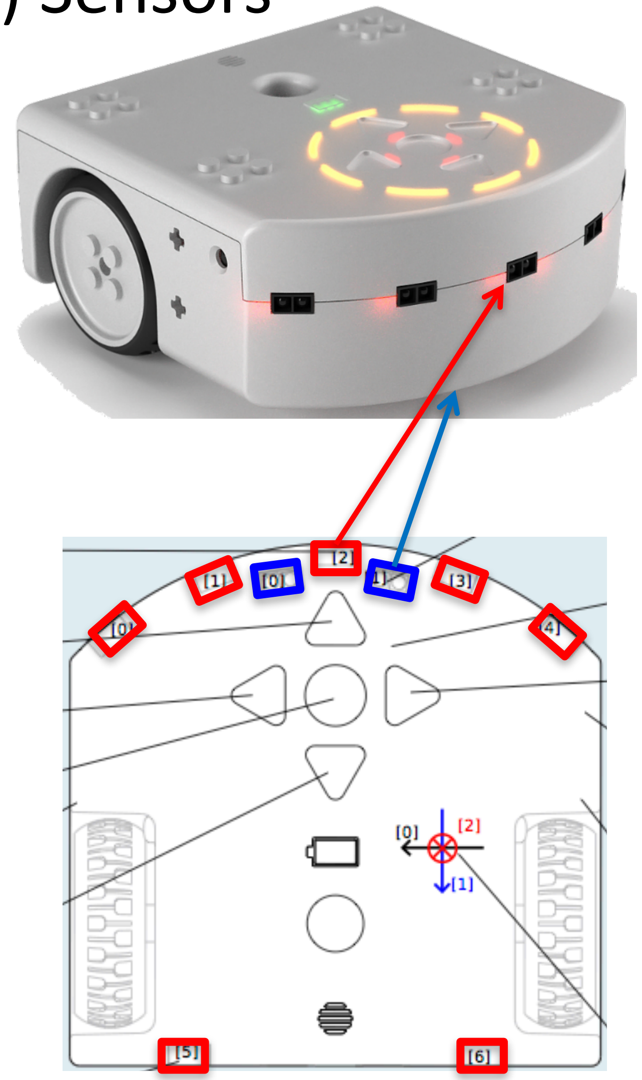
Gyroscope

Camera

Characterizing sensors

Example: Proximity (prox) Sensors

- 7 horizontal proximity sensors
  - Measures distance to objects using infra-red light
  - 5 in front and 2 in rear
  - Range: 0 (nothing) to 4000+ (object very close)
  - Values stored in **`prox.horizontal[0:6]`**
- 2 ground proximity sensors
  - Measures light from the ground
    - *ambient* (surrounding light)
    - *reflected* (received infra-red light emitted by sensor)
    - *delta* (difference between ambient and reflected)
  - 2 in front
  - Response ranges: 0 (no light) to 1023 (full light)
  - Values stored in array
    - **`prox.ground.ambient[0:1]`**
    - **`prox.ground.reflected[0:1]`**
    - **`prox.ground.delta[0:1]`**



Sensors are Imperfect

- Sensors have two kinds of errors
  - *Bias*: a systemic deviation from the true value
    - E.g., a clock that runs fast, or
    - A thermostat that thinks its warmer than it is.
  - *Variability*: random deviation from the true value
    - E.g., static on the radio and
    - Flickering low-oil sensor
- **Key Ideas:**
  - No matter how good a sensor is, it is imperfect
  - Imperfect sensors introduce *uncertainty*
  - Our programs have to deal with the uncertainty

Models

- Sensors, like many devices, are complicated
- **Idea:** To use sensors (easily), we need a model of the sensor
- **Def:** A *model* is a simplified description of a complicated object that describes how the object will behave
- **Questions:**
  – What properties should we include in the model?
  – How do we create a model of the sensor?

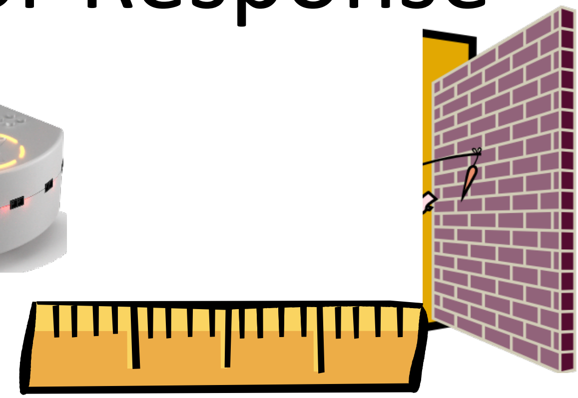
How to Model/ Characterize a Sensor

1. Identify the sensor we want to model
2. Identify the sensor property we want to model
3. Identify the possible variables of the property
4. Fix all but one of the variables
5. Create a sequence of known ``actual'' inputs where the
   - One variable is varied and
   - All other variables are fixed
6. Perform a sequence of measurements (*multiple times)* on the inputs
7. Tabulate the results and compute aggregates if appropriate (average, median, variance, etc)
8. Plot the results
9. Repeat steps 4 – 8, allowing a different variable to vary each time
10. Analyze the plot(s) to model the sensor

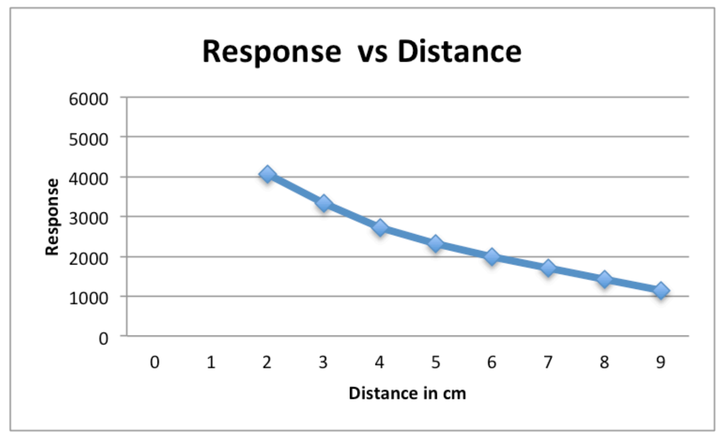
Questions:

1. How do we get the "measured" values?
2. How do we get the "actual" values?
3. How do we ensure all other variables are fixed?

Example: Proximity Sensor Response[1]



1. Sensor: Horizontal Proximity Sensor
2. Property: Response
3. Variables to consider:
   - Distance to target
   - Target size
   - Target material
   - Target shape
4. Fix all variables except "Distance to Target"
5. Create a sequence of known inputs
6. Perform a sequence of measurements for each input
7. Tabulate the results and compute means
8. Plot the results
9. Repeat steps 4 - 8
10. Analyze the plot to derive the sensor model



Inputl (cm)	2	3	4	5	6	7	8	9
Response 1	4051	3343	2735	2311	1973	1708	1421	1145
Response 2	4056	3340	2734	2320	1983	1697	1426	1152
Response 3	4062	3347	2721	2307	1981	1702	1408	1138
Average	**4056**	**3343**	**2730**	**2313**	**1979**	**1702**	**1418**	**1145**

Making Use of the Results[2]

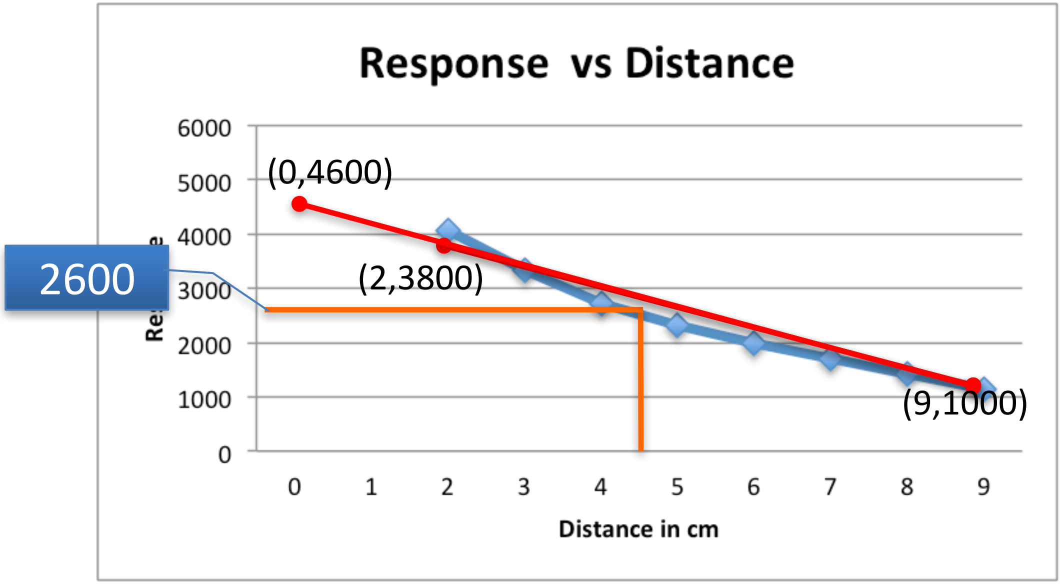
- General observation(s)
  - Response decreases as distance increases
  - Useful for visual interpolation
- Create a linear model
  - Draw a linear approximation
  - Compute slope (*m*) and intercept (*b*) of the line
  - Plug into equation of a line
- Then what?

$$m = \frac{rise}{run} = \frac{y_2 - y_1}{x_2 - x_1} = \frac{1000 - 3800}{9 - 2} \cong -400$$

$$x = 2, y = 3800$$

$$y = mx + b \Rightarrow 3800 = -400 \times 2 + b$$

$$b = 4600$$



$$y = mx + b \quad \Rightarrow \quad y = -400x + 4600$$

Using Sensors

- **Idea:** Perform sensor readings when events occur
  - Checking a sensor's reading is called *polling* the sensor
- It is the program's responsibility to *interpret* the sensor reading, i.e.,
  - Translate the value returned by the sensor into meaningful information
- A simple way to assign meaning is to use *thresholds*

Thresholds

- We are typically not interested in what the value of a sensor reading is.
- We are typically interested
  - when that value changes, or
  - when that value reaches a specific threshold
- For example,
  - We don’t care if the car ahead of us is 50 meters away or 150 meters away.
  - We do care if
    - the car is getting closer, or
    - the car is less than 5 meters away!

Thresholds (cont.)

- **Def:** A *threshold* is a fixed constant such that an event is triggered when a measurement from a sensor returns a value that is above (or below) the constant.

- Examples:
  - Object too close:
    - if distance < threshold, stop
  - Loud sound occurs:
    - if sound level > threshold, start moving
  - Black line detected:
    - If light level > threshold, move right, else move left

What are the thresholds here?

```
onevent prox
  if prox.horizontal[2] > 1000 then
    motor.left.target = 0
    motor.right.target = 0
  elseif prox.horizontal[4] > 1000 then
    motor.left.target = -100
    motor.right.target = 100
  elseif prox.horizontal[0] > 1000 then
    motor.left.target = 100
    motor.right.target = -100
  else
    motor.left.target = 100
    motor.right.target = 100
  end
```

But … How often should sensors be polled?

Polling Frequency

- Polling Frequency depends on
  - The response time of the sensor
  - The rate at which the environment changes
- Response time dictates the maximum useful polling rate
- The rate of change dictates the minimum rate needed to ensure that no events are missed

- **Question:** What if the maximum useful rate is less than the minimum required rate?

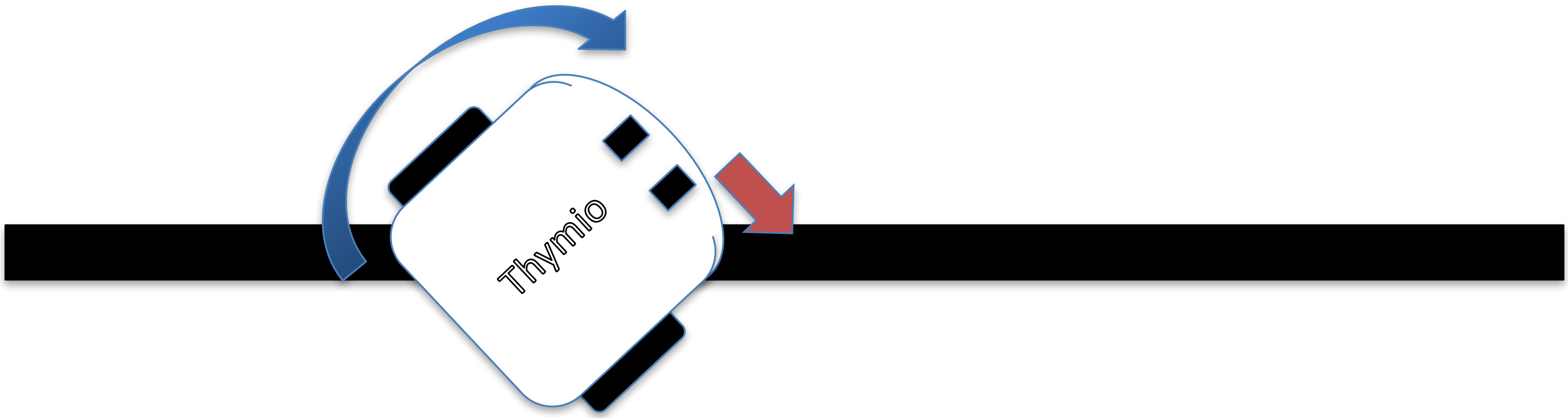
Polling Frequency vs. Response Time

- Observation: There is no point in polling the sensor quickly if its response time is slow
  - Are we there yet? How about now? Now? Now?
- Polling the sensor too quickly does not hurt, but wastes CPU resources
- Our sensors have a fast response time (mostly)

- When the response time is slow, our programs need to take this into account

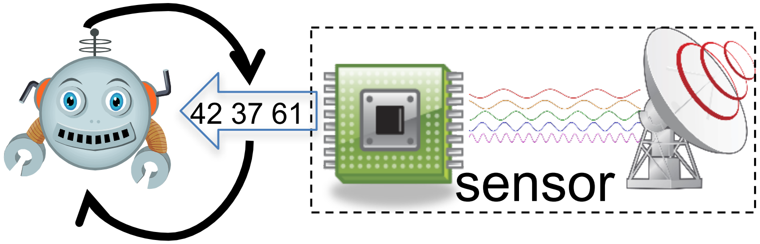
When Response Time Matters

- In Follow-The-Line
  - The angular velocity of the light sensor is quite fast
  - This could cause the sensor to move over the black line too quickly to pick it up
  - This would result in the robot losing the line
- How do we ensure that the robot does not miss the line?



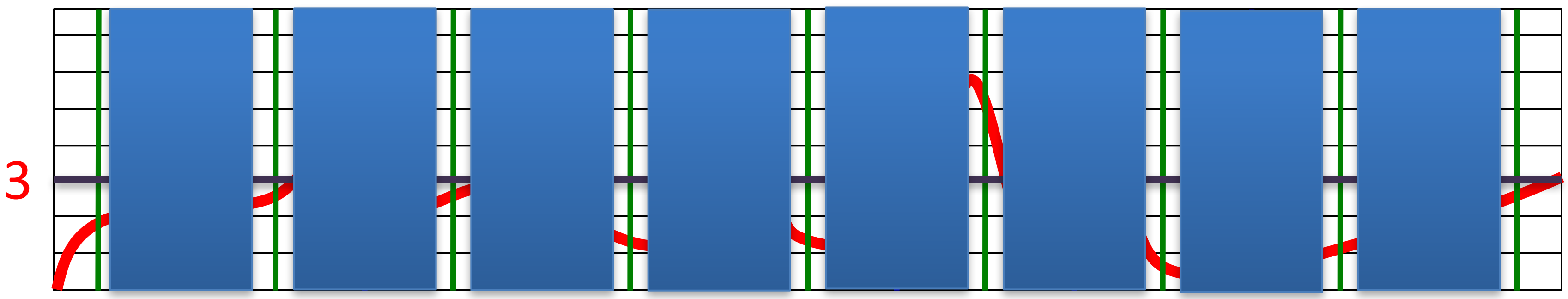
Sampling

- Sensors must be polled (sampled) for values
- The *sampling rate* is the frequency of the polls
- A higher rate means we are
  - Less likely to miss a change in inputs
  - Using more CPU time to poll the sensor
- If the rate is too high, there is no time to do anything else



Another Example

When is the signal at least 3?



S1 2 2.5 2.5 1.25 1.25 6 0.5 1 2.5

S2 2.25 5 2.5 4 1.25 5 0.5 2

S* 2 2.25 2.5 5 2.5 2.5 1.25 4 1.25 1.25 6 5 0.5 0.5 1 2 2.5

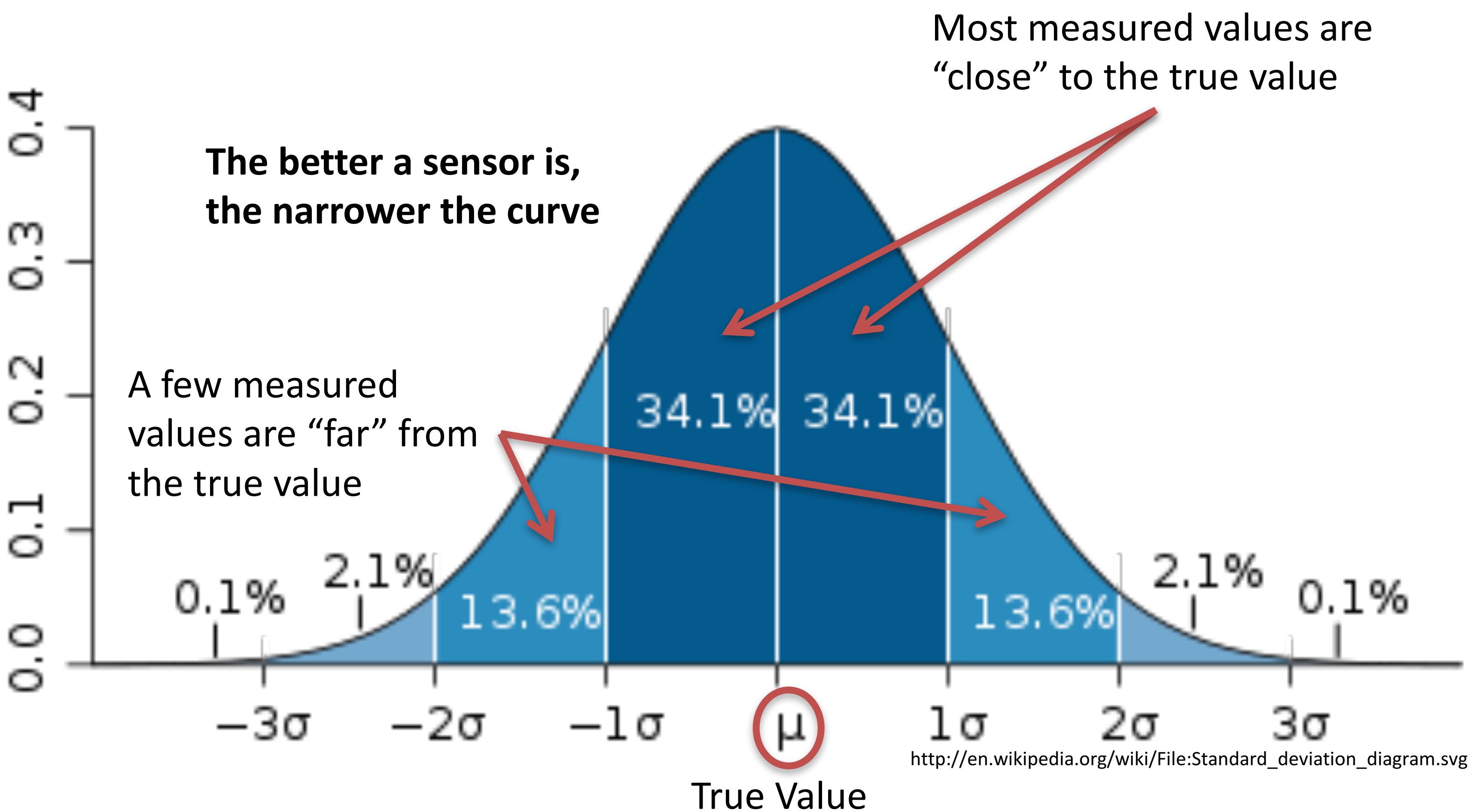
What If We Do Detect a Change?

- Suppose the sensor returns a different value.
- Does this mean that the environment has changed?
- Are you sure?

Variability of Sensors

- Problem: All sensors have some variability
  - A sensor reading randomly deviates from the true value
- A single sensor reading may not report the true value or even close to the true value
- Multiple sensor readings may report different values for the same true value
- The reported values will be *distributed* around the true value
  - Most readings will be "close" to the true value, assuming the bias is 0

Normal Distribution



http://en.wikipedia.org/wiki/File:Standard_deviation_diagram.svg

Dealing with Variability

- **Key Idea:** Want to aggregate the sensor data
  - Get multiple “second opinions”
- Approach:
  - Take a number of sensor readings (polls)
    - More is better
  - Combine readings for a more accurate measurement
    - Average (mean)
    - Median
    - Mode
- Trade-off:
  - Get a more accurate measurement
  - Costs more time to perform
- **Question:** Is taking multiple readings all at once useful?

Sensor Debouncing

- Key Idea: Need to filter the data from a sensor
- Approach:
  - Take a number of samples (polls)
    - More is better
  - Combine samples for a more precise measurement
    - Average (mean)
    - Median
    - Mode
- Trade-off:
  - Get a more precise measurement
  - Costs more time to perform