# Bayesian Selection of Non-Faulty Sensors

Kevin Ni and Greg Pottie
University of California, Los Angeles

# Motivation

- Sensor networks have been used in environmental monitoring
- We often see sensors failing throughout the deployment of the network
- Faults may be due to power supply problems, problematic wiring, or other unforeseen issues
- We aim to detect these faults without the need for human inference during the deployment to cue action to fix the problem in some way

# Design methodology

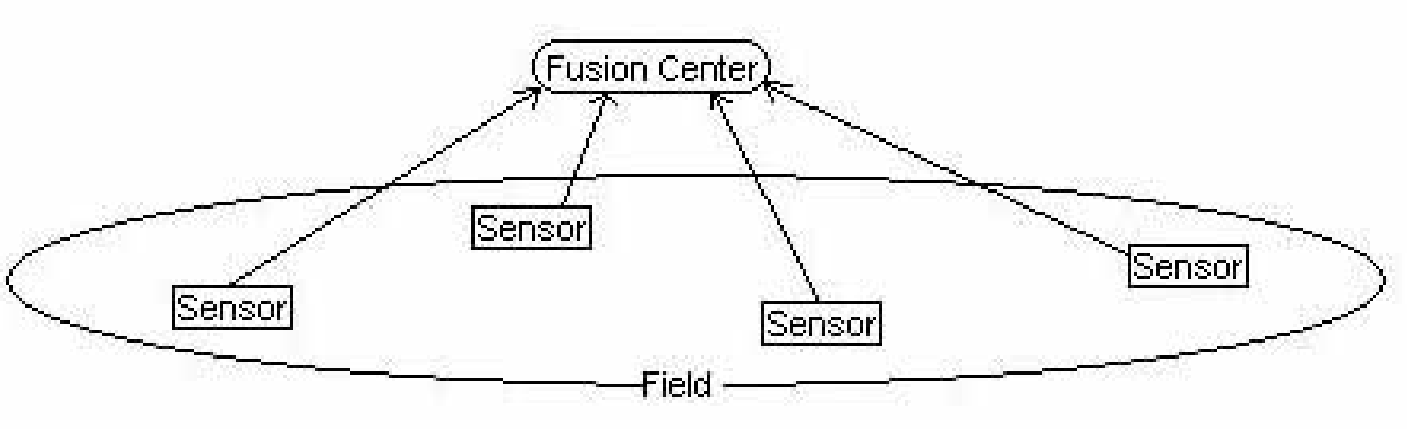
- In order to determine sensor faults we must consider four issues.
  - We must determine a model of normal sensor behavior
  - We must model or restrict ourselves to a model of the physical phenomenon
  - Determine whether or not a sensor's behavior conforms with it's expected behavior as dictated by our models
  - Remedy problems or classify behavior as faulty or acceptable, and update models accordingly

# Approach overview

- We seek to resolve at least some of the issues listed in the previous slide
- Currently we lack a fully trusted node for ground truth
- We resort to using an agreement problem combined with a Bayesian selection method
- We use a Bayesian method so that we can include prior knowledge; it also provides for a easy updating mechanism

# Problem formulation and assumptions

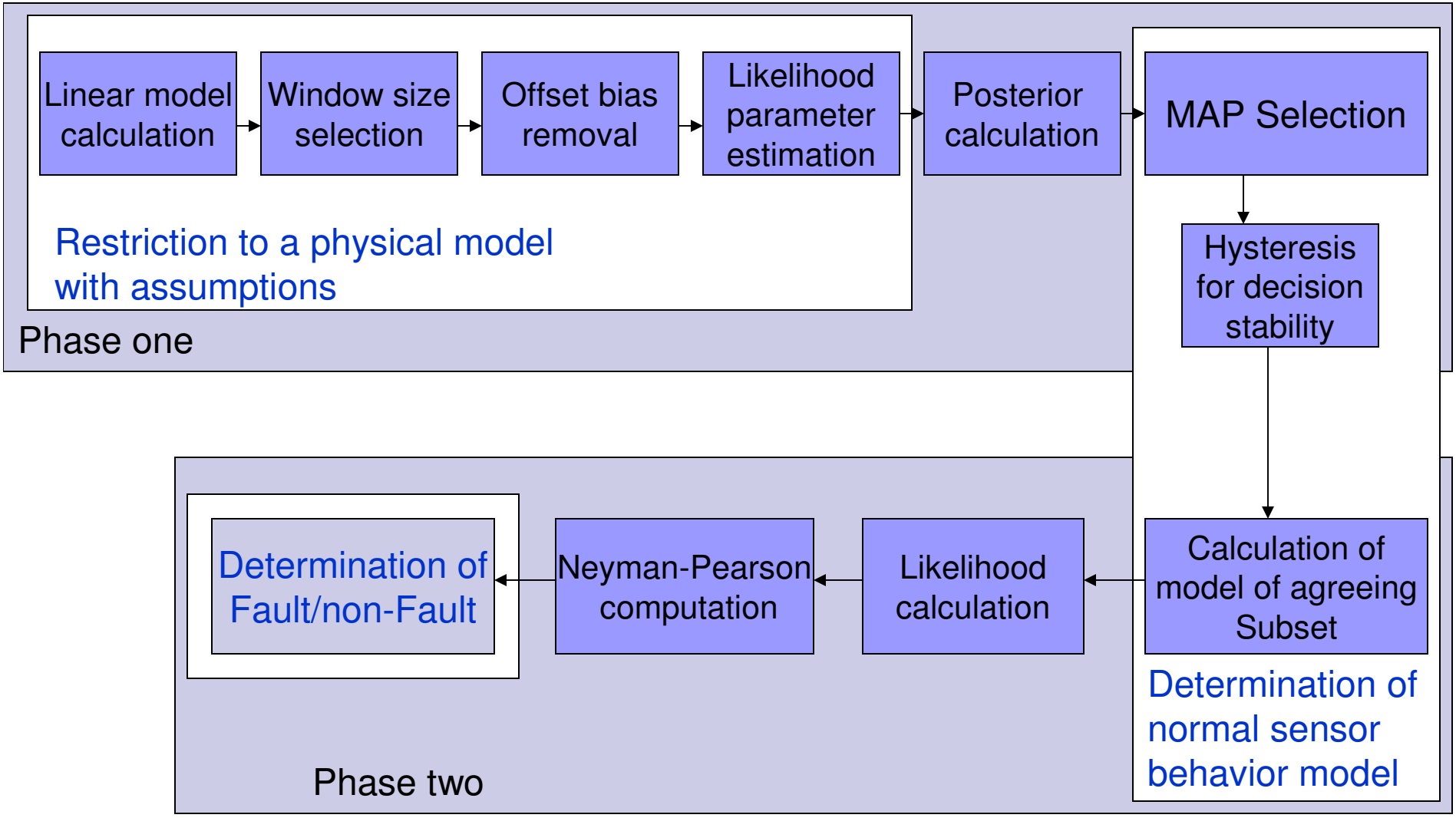
- We have a set of $K$ sensors deployed across a field as depicted in the figure below
- Smoothly varying field
- Initially, we consider only data trends in judging sensor quality
  - Expect that sensor values move with similar trends
- Will then consider detection assuming similar data values
  - Sensor data is clustered and all measuring the same values



# Further assumptions

- Phenomenon smooth in time to allow for use of a linear model
- Faults are relatively persistent
- Gaussian noise model
- No large data gaps
- At least $\frac{K}{2}$ are not faulty at any given time
- All faulty sensors do not behave as a single correlated process.

# System Flow


Linear model calculation
Window size selection
Offset bias removal
Likelihood parameter estimation
Posterior calculation
MAP Selection
Restriction to a physical model with assumptions
Phase one
Hysteresis for decision stability
Determination of Fault/non-Fault
Neyman-Pearson computation
Likelihood calculation
Calculation of model of agreeing Subset
Determination of normal sensor behavior model
Phase two

# Phase one: Bayesian detection method to select subset of agreeing sensors

# Phase one: Selection of agreeing subset

- Frame the problem as a Maximum a-posteriori probability (MAP) problem:

$$\hat{\vec{\phi}} = arg\max_{all\vec{\phi}}(P(\vec{\phi}|\vec{D},\xi))$$

- Select the subset, $\vec{\phi}$ , with the highest posterior probability of being reliable, given our data.
- Search all $\vec{\phi}$ of size $\frac{K}{2}$ since this is the minimum expected to be non-faulty
- $\phi_i \in \{0, 1\}$ represents exclusion and inclusion in the set
- $P(\vec{\phi}|\vec{D},\xi)$ is the posterior probability
- $\vec{D}$ represents new data
- $\xi$ represents other background information

- This is a non-polynomial search, but still tractable in reasonably sized networks

# Phase one: Implementation

- Recall Bayes rule:

$$P(\vec{\phi}|\vec{D},\xi) = \frac{f(\vec{D}|\vec{\phi},\xi)P(\vec{\phi}|\xi)}{f(\vec{D}|\xi)}$$

- We initially set the prior distribution $P(\vec{\phi}|\xi)$ to a uniform distribution implying that we do not know what sensors are reliable at system start
- Priors are updated using the previous algorithm iteration's posterior distribution $P(\vec{\phi}|\vec{D},\xi)$
- Gaussian assumption gives a Gaussian likelihood: $f(\vec{D}|\vec{\phi},\xi)$
- The marginal is given by: $f(\vec{D}|\xi) = \sum_{all\ \vec{\phi}} f(\vec{D}|\vec{\phi},\xi)P(\vec{\phi}|\xi)$

# Phase one: Window size selection

- In order to estimate the parameters of the likelihood, we use linear models to model sensor data over a particular window of past data, *M.*
- Want to choose a good window size that for each sensor:
  - Interpolates data well
  - Predicts the next point well
- We can adaptively select the *M*, within a given range, such that we choose the window size that has the smallest maximum mean square error across all sensors.

# Phase one: Offset bias removal

- Without a strong model of the phenomenon, we assumed that temperature changes are similar across the field
- We remove offsets in sensor data by subtracting the bias that was derived from the linear model for each sensor from the sensor data.

- Note, under the data trends only test, cases exist where faulty sensors may have similar trends with large offsets.

- To test distance between sensor data, i.e. offsets, we impose the assumption that sensor data is clustered and measuring similar data values.
- We can then apply our algorithm to sensor data by removing the trend component derived from the model from the sensor data, (as in the trend only test).
- If a sensor is marked as faulty in either test, then it is faulty overall.

# Phase one: Likelihood parameter estimation

- The likelihood function $f(\vec{D}|\vec{\phi},\xi)$ may be influenced by many factors in the form of background information.
- $f(\vec{D}|\vec{\phi},\xi)$ is a joint Gaussian distribution.
- For a subset, $\vec{\phi}$ , we only include parameter values for sensors included in that subset in the likelihood function.

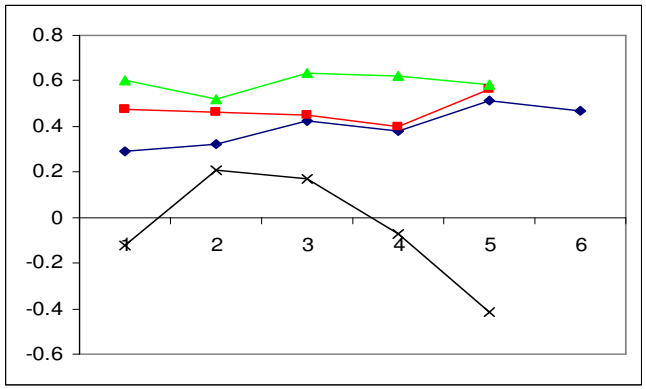
- To calculate the likelihood, we must determine $\vec{\mu}$ and $\Lambda$

- Since we assumed local linearity, we use linear models for interpolation and estimation.
- We can use a model derived using data from all other sensors as the expected value, $\vec{\mu}$. With this we can also calculate the covariance $\Lambda$ .

- With the likelihood parameters, we can now evaluate our MAP criterion

# Phase one: An example

- For an example of a Bayesian selection step, we will compare two subsets where one set includes a faulty sensor. We set the priors to have equal likelihood that these sets are in agreement.
- $set_1$ = green and red
- $set_2$ = red and black




We estimate the covariance matrices of these sets as:

| | | | |
|---|---|---|---|
| 0.01039 | 0.00803 | 0.007033 | -0.02535 |
| 0.00803 | 0.007033 | -0.02535 | 0.10819 |

So the likelihood values are:

$f(D|set_1)$ = 9.7547 and $f(D|set_2)$ =2.6778

So, using uniform priors: $p(set_i)$ = 1/6, (6 possible subsets of size K/2 with K=4) and plugging into Bayes rule we get:

$p(set_1|D)$ = 0.362706 and $p(set_2|D)$ = 0.099568

MAP would select $set_1$ over $set_2$ in this case, which is what we expect

These values of $p(set_1|D)$ and $p(set_2|D)$ are to be used as priors in the next iteration

# Phase one: Hysteresis for decision

- Want to ensure the stability of decisions
- MAP selection may “flip-flop” from subset $\vec{\phi}_n$ to subset $\vec{\phi}_m$ .
- After a brief moment, MAP switches back to $\vec{\phi}_n$ from $\vec{\phi}_m$ .
- Want to avoid any difficulty so we include a lag in our decision.

- Note, we are only comparing MAP decisions, and not our final agreeing subset decisions.

# Phase two: Classification of all sensors as either faulty or not

# Recall: Design methodology

- Originally we stated four issues for detection
  - We must determine a model of normal sensor behavior
  - We must model or restrict ourselves to a model of the physical phenomenon
  - Determine whether or not a sensor's behavior conforms with it's expected behavior as dictated by our models
  - Remedy problems or classify behavior as faulty or acceptable, and update models accordingly
- We have selected a sensor subset that we use to determine a model of expected sensor behavior
- Our physical phenomenon model has been limited by our assumptions

# Phase two: Judging sensors

- Now we must judge the remaining sensors
- Use likelihood function, $f(\vec{D}|\vec{\phi},\xi)$ , which we can normalize, as a natural extension to the Bayesian approach for our metric.
- We assume a Gaussian likelihood function for simplicity

- We can evaluate the likelihood for each sensor in relation to the model for expected sensor behavior

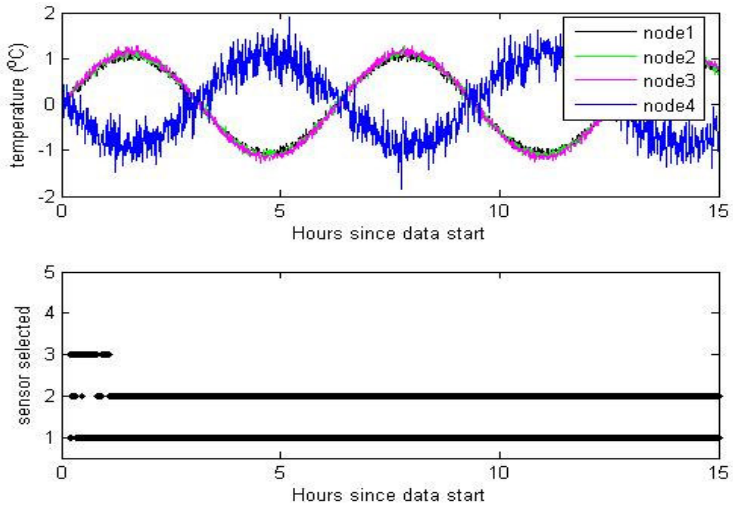
- To smooth wild variations we use the previous *M* samples for a moving average of the likelihood

# Phase two: Judging sensors

- We want to maximize our chances of detection, $P_D$, of a faulty sensor while keeping the false detection, $P_{FA}$, below a certain level.
- We apply a simple Neyman-Pearson test to select faulty sensors at each iteration assuming the distribution on the likelihood for "good" sensors is Gaussian
- In our experiments we set our threshold of $P_{FA}$ to be 0.05

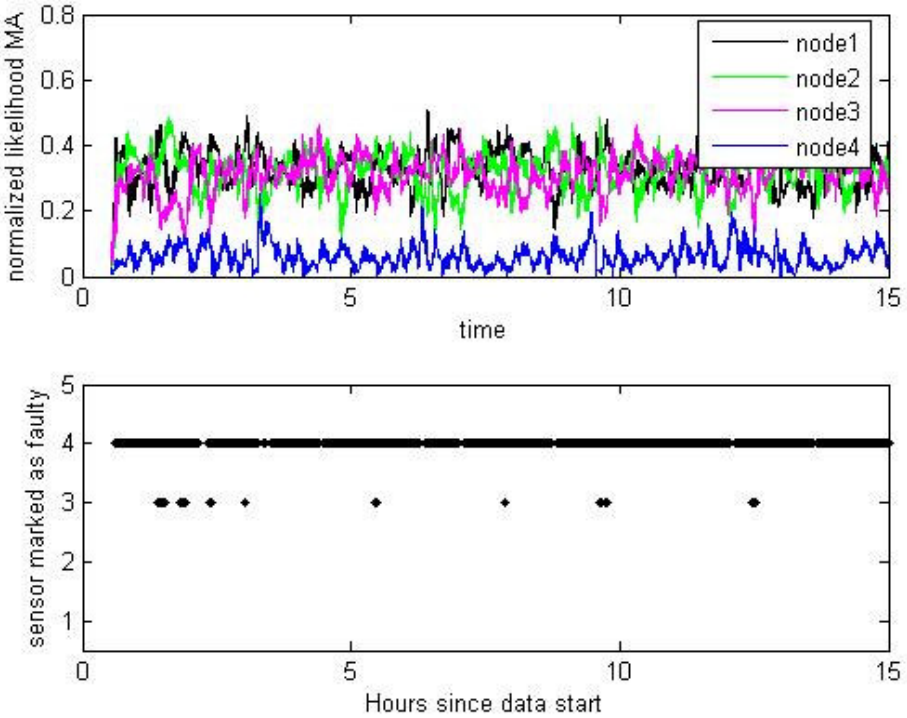
# Results: Simulated data with faulty sensor

- First we would like to see how well our method performs when all sensors are working well, and all assumptions hold relatively true
- We have four sensors measuring similar data, but with slightly different amplitudes.
- Sensor 4 from we make to be clearly faulty.
- We see what sensors are included in the “agreeing” subset



# Results: Simulated Data with faulty sensor

- Each good sensor has remained under the designed $P_{FA}$ in all cases
- In the trend only consideration, we note some of the missed detection is from having the slopes both equal zero at the peaks for faulty and non-faulty sensors
- We see that we increase our fault detection rate to 99.34% when we include the offset, this is because the good sensors tend to match with the assumption of measuring similar data.

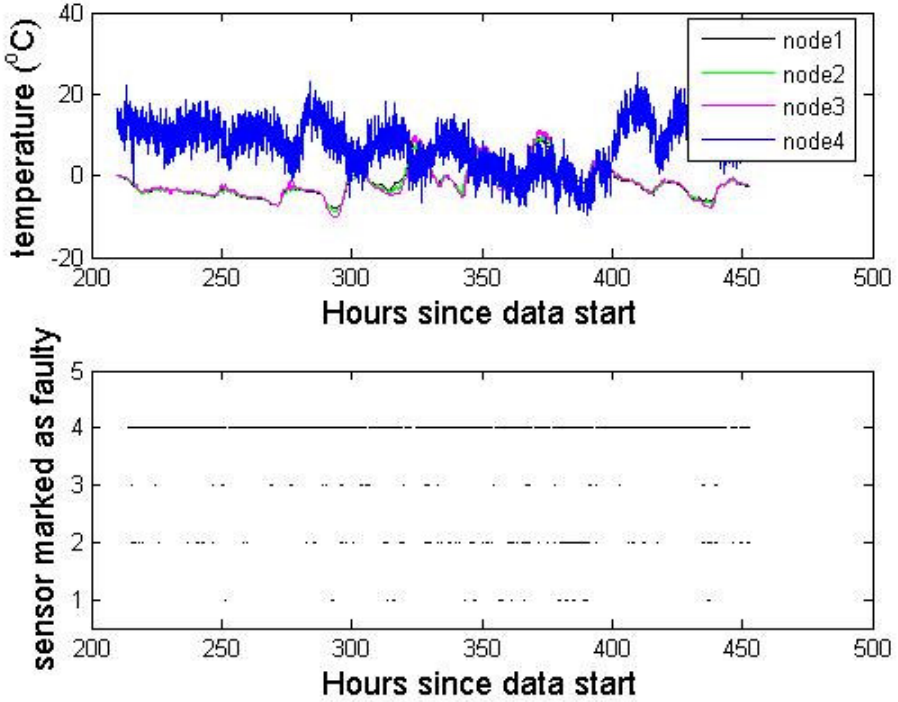



| Proportion of sensors marked as faulty | | |
|---|---|---|
| Sensor | Trend only | Trend+offset |
| 1 | 0 | 0 |
| 2 | 0 | 0.0089 |
| 3 | 0.0189 | 0.0234 |
| 4 | 0.8201 | 0.9934 |

# Results: Cold Air Drainage Data

- We applied our algorithm to data collected in the field
- Sensors are deployed at James Reserve in California measuring temperature and other data.
- One of the sensors is clearly faulty and measuring incorrect data.

- We can see how frequently each sensor is marked as faulty



# Results: Cold Air Drainage Data

- When considering trend only:
  - The $P_{FA}$ exceeded design tolerance for sensor 2. The other two good sensors are within design specifications. Probably since this sensor is not usually in the agreeing subset
  - Overall $P_{FA}$ rates were higher than simulated cases. This indicates inaccuracy in our models.
  - Sensor detection rate was lower than simulated case, but this is expected
- When considering trend and offset together:
  - False detection rate jumps
  - Detection rate for sensor 4 is very high, and close to 100%, though not as high as in the simulated case.
  - False detection rate is higher likely due to the fact not all sensors are measuring the exact same data.

| **Proportion of sensors marked as faulty** | | |
|---|---|---|
| Sensor | Trend only | Trend+offset |
| 1 | 0.0186 | 0.0297 |
| 2 | 0.0796 | 0.1478 |
| 3 | 0.0194 | 0.0803 |
| 4 | 0.7589 | 0.9793 |

# Recall: Design methodology

- Originally we stated four issues for detection
  - We must determine a model of normal sensor behavior
  - We must model or restrict ourselves to a model of the physical phenomenon
  - Determine whether or not a sensor's behavior conforms with it's expected behavior as dictated by our models
  - Remedy problems or classify behavior as faulty or acceptable, and update models accordingly
- We have taken a first attempt at some of these problems
- We seek to improve our performance through further work in a few key areas
- This will eventually allow us to incorporate the final step

# Conclusion and Future work

- We have proposed a framework to detect sensor faults online using a Bayesian MAP detection approach
- We seek to better model data and fault modes in order to reduce false alarm rates
- We also will improve our Bayesian framework in order to incorporate better prior knowledge on the model parameters.
  - We seek to include long term data or fault behavior in current decisions.
- This will allow us to relax many assumptions and allow for data that is expected to have different offsets and trends.

# Thank you!