

Lecture 14

The Kalman Filter

CS 3630



Context

	Space	Distribution
Particle Filter	Continuous	Multimodal
Kalman Filter	Continuous	Unimodal

What is a Kalman Filter and What Can It Do?



A Kalman filter is an **optimal estimator**

- invented in 1960 – [helped us land on the moon and more!](#)
- infers parameters of interest from *indirect, inaccurate* and *uncertain* observations.
- *recursive* so that new measurements can be processed as they arrive.

Optimal in what sense?

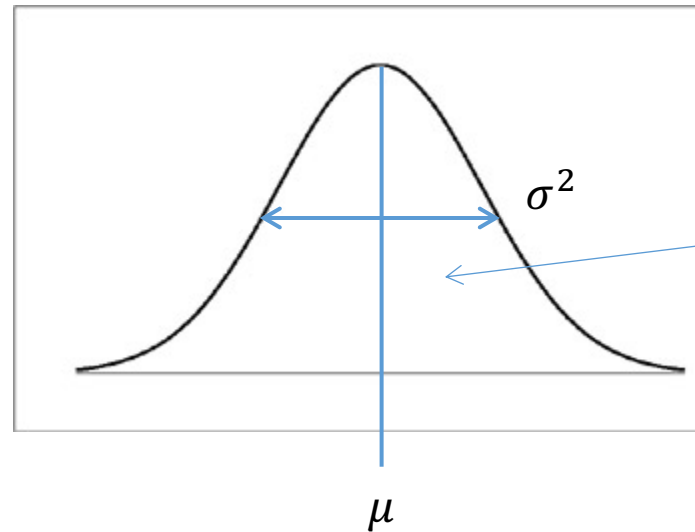
If all noise is Gaussian, the Kalman filter **minimizes the mean squared error** of the estimated parameters

Why do we keep using the word “Filter”?

- Optimal estimates from noisy data => “filtering out” the noise.
- Also “projects” these measurements onto the **state estimate**.

Some basics...

Gaussian distribution

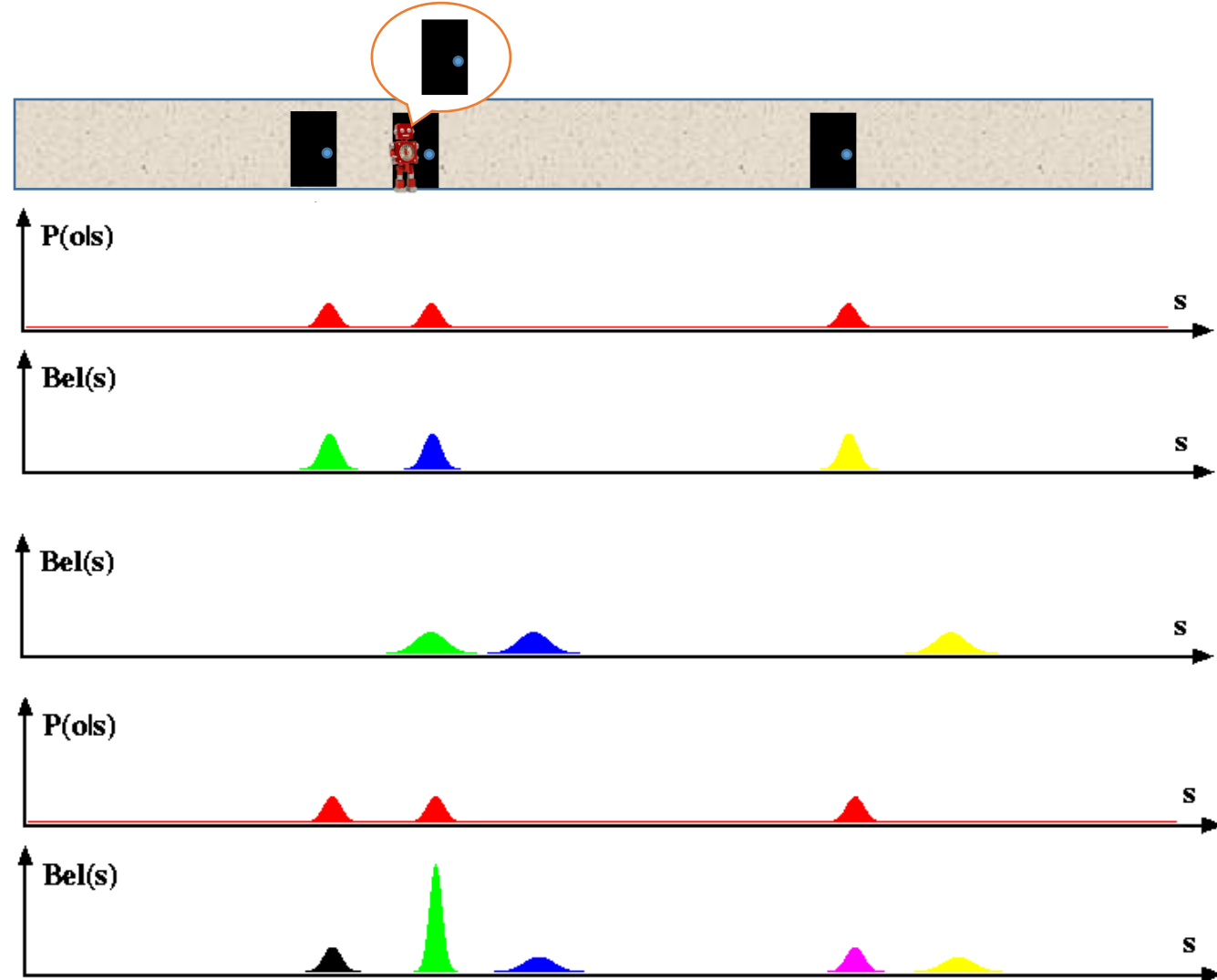


Area under the
curve sums to 1

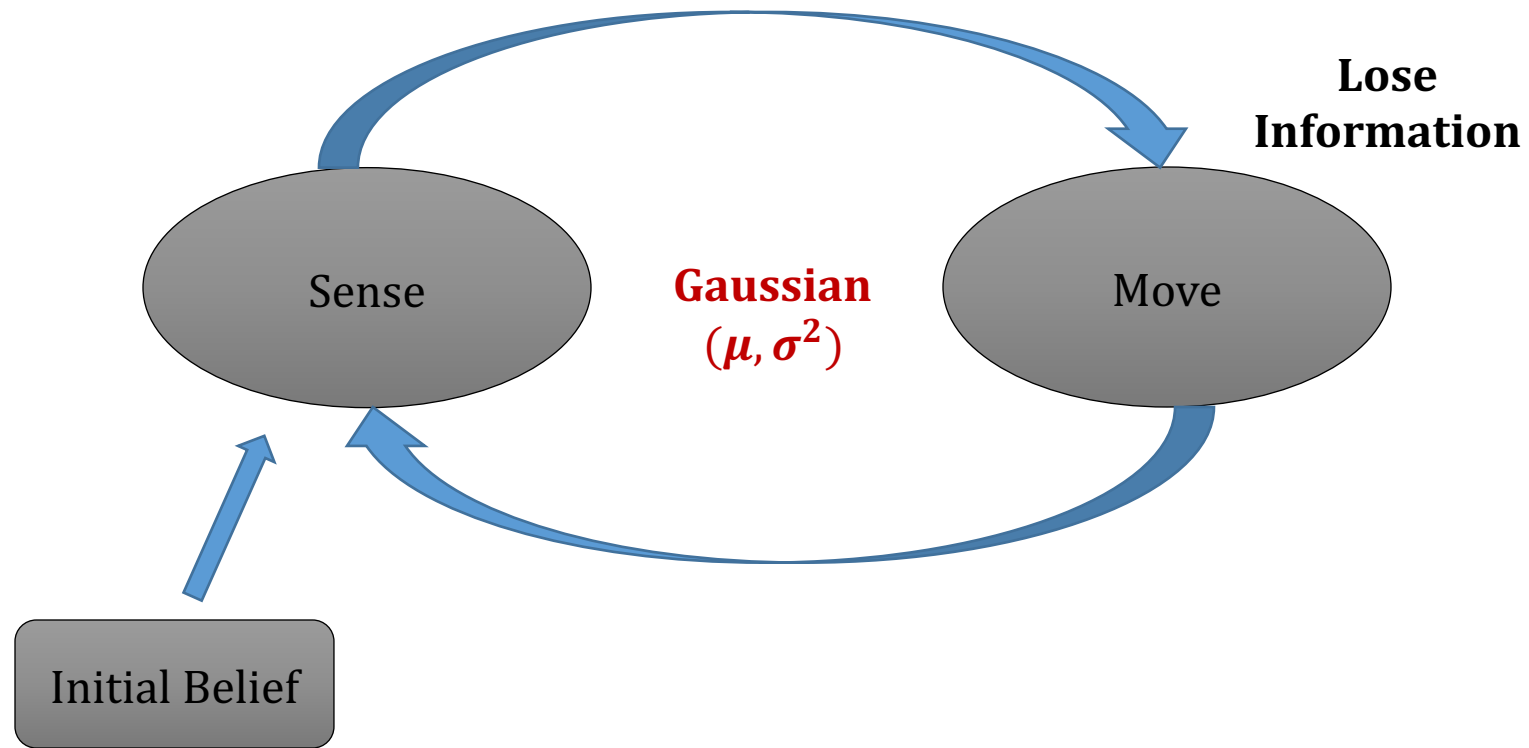
$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

(in 1D for now)

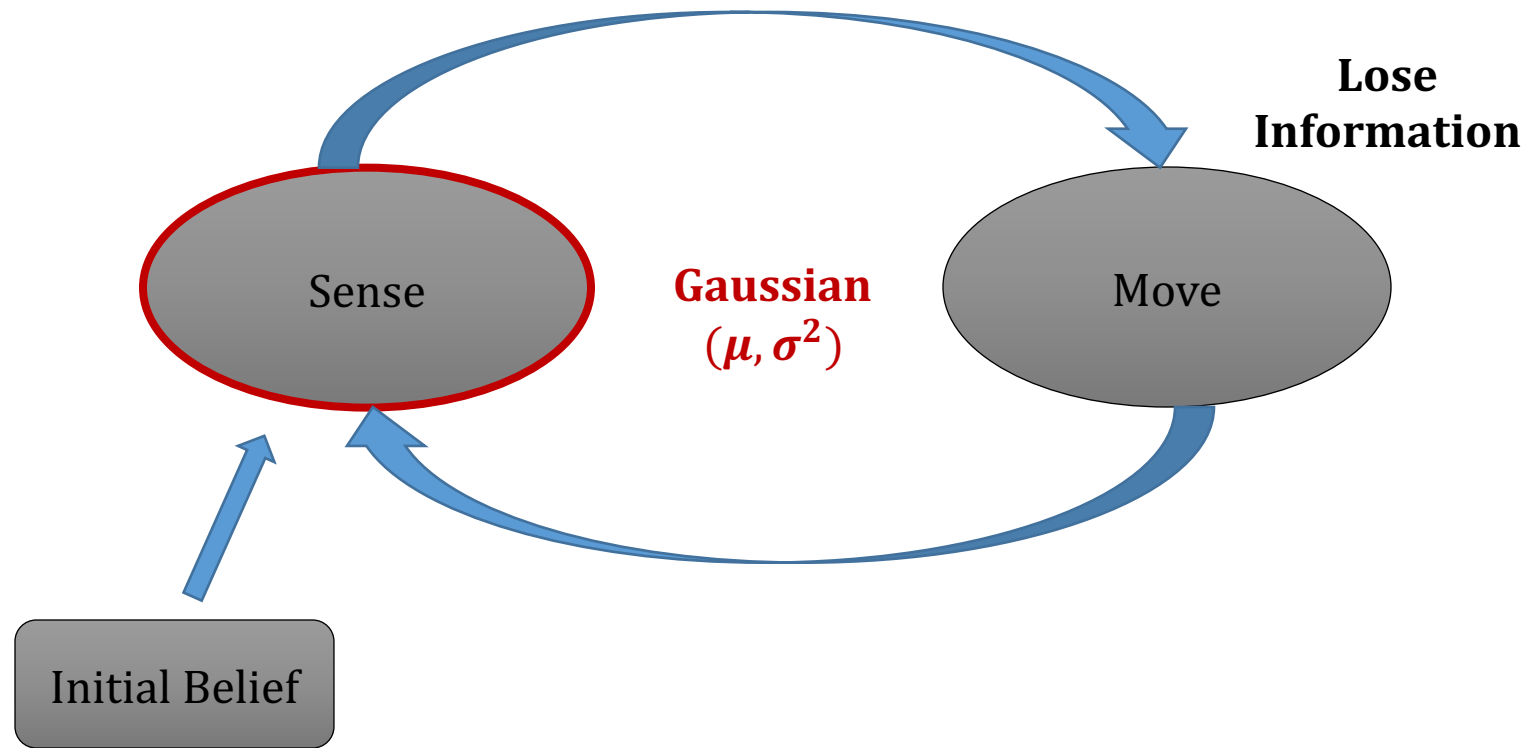
Similar to this example, but with a unimodal distribution



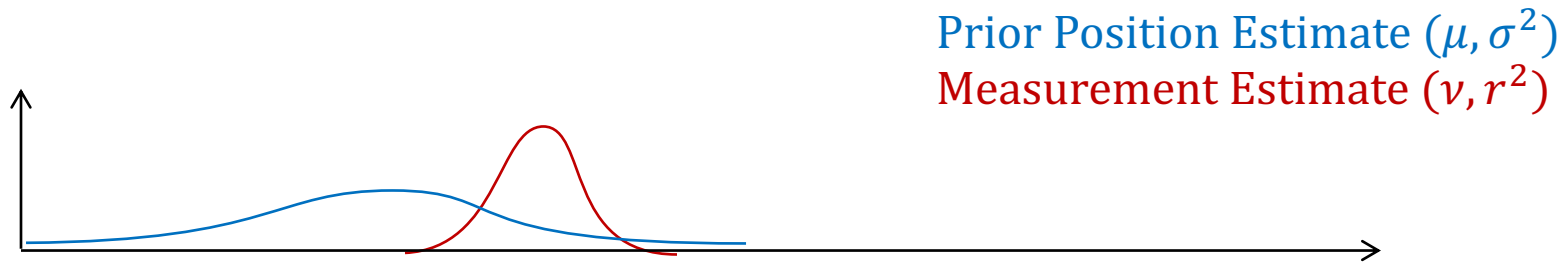
Kalman Filter



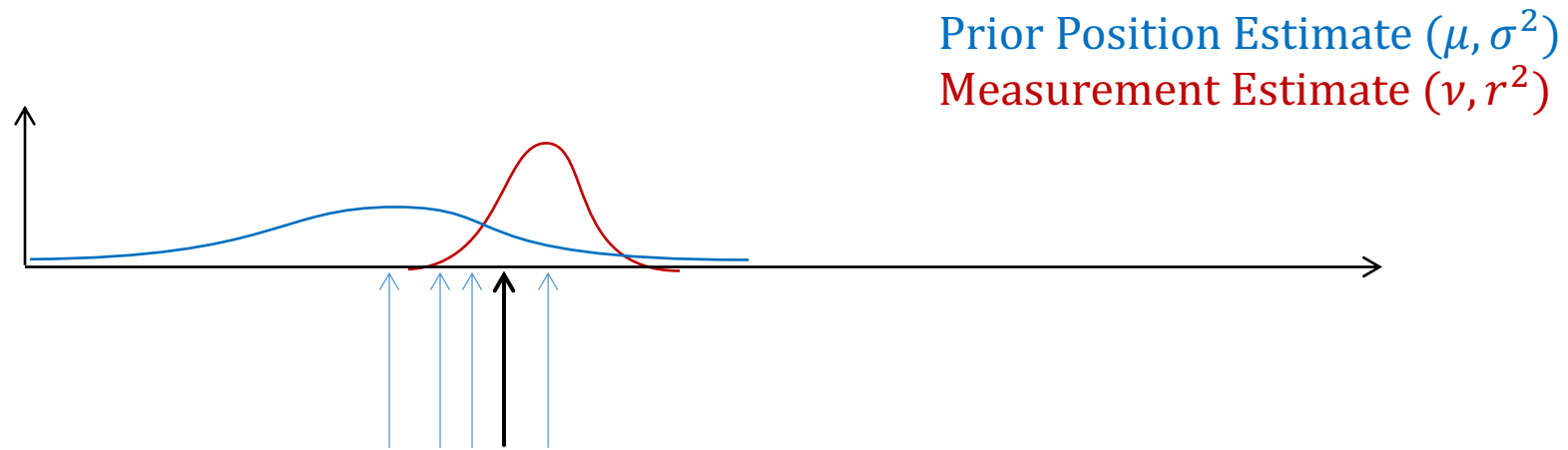
Kalman Filter



Measurement (sensing) Update Example

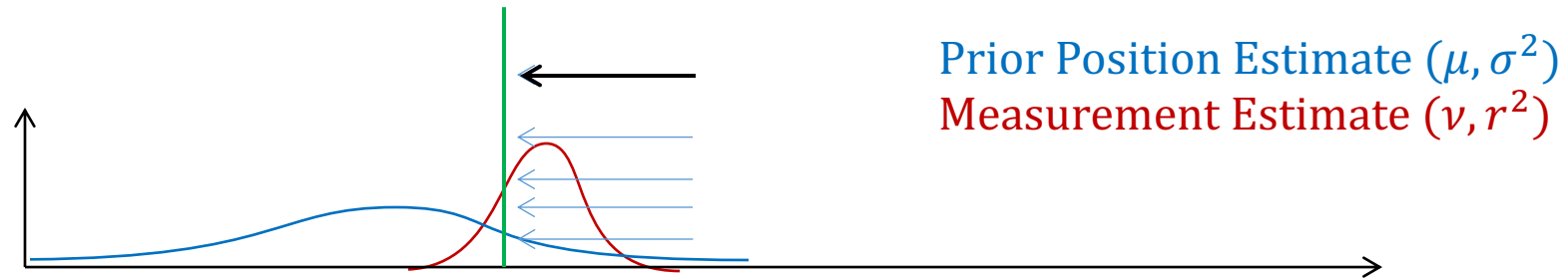


Measurement Update Example



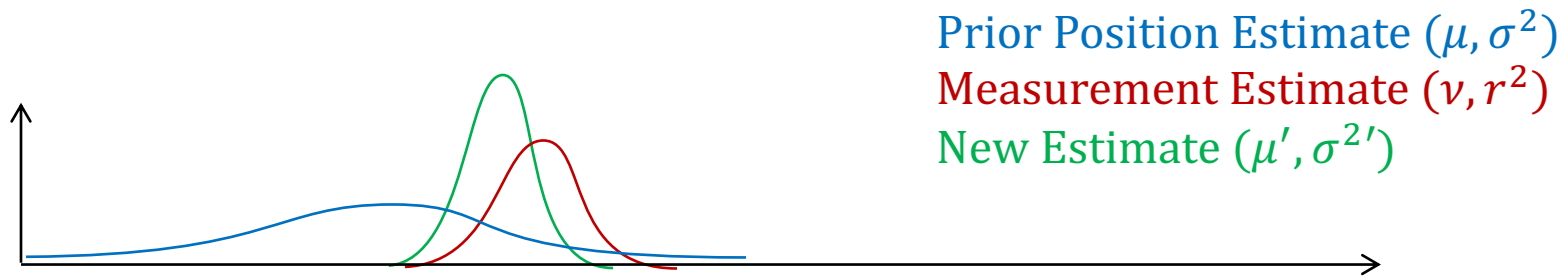
Where is the new mean μ' ?

Measurement Update Example

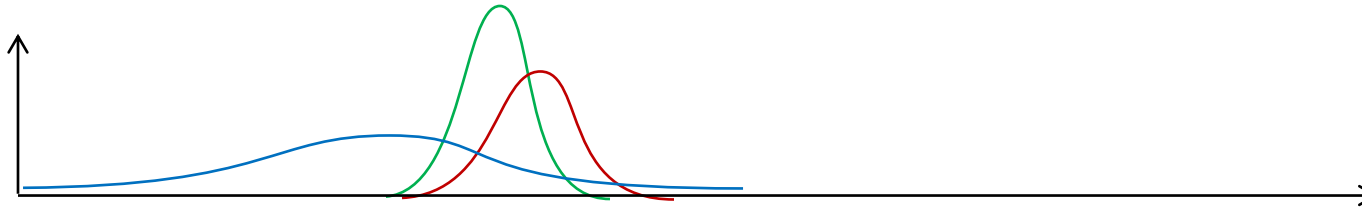


What is the new variance $\sigma^{2'}$?

Measurement Update Example



Equivalently...



Prior Estimate (μ, σ^2)

$\leftarrow p(x)$

Measurement Likelihood (v, r^2)

$\leftarrow p(z|x)$

New (Posterior) Estimate (μ', σ'^2)

$\leftarrow p(x|z)$

Markov Localization

- Perception (or sensing) model: represents likelihood that robot senses a particular reading at a particular position (related to our discussion last class)

$$P(x) = \alpha \underbrace{P(z|x)} P(x)$$

multiplying Gaussian PDFs together

Probability that robot will perceive z , given that the robot is in position x , times the likelihood the robot is in position x

- Action (or motion) model: represents movements of robot

$$P(x) = \sum P(x|a, x')P(x')$$

Probability that action a from position x' moves the robot to position x , times the likelihood the robot is in position x' , summed over all possible ways robot could have reached position x

Multiplying Two Gaussians

Prior Position Estimate (μ, σ^2)
Measurement Estimate (ν, r^2)

Equation for Gaussian: $f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$

[Link to complete derivation](#)

1. The product of two Gaussians is: $\exp \left[\left(\frac{-1}{2} \right) \left(\frac{(x-\mu)^2}{\sigma^2} + \frac{(x-\nu)^2}{r^2} \right) \right]$ (ignoring the constant)
2. To find the mean, need to maximize this function (i.e., use derivative)
3. $\frac{d}{dx} = 2 \left(\frac{x-\mu}{\sigma^2} \right) + 2 \left(\frac{x-\nu}{r^2} \right)$
4. Set equal to 0 and solve for x
5. $x = \frac{\mu r^2 + \nu \sigma^2}{r^2 + \sigma^2} = \left(\frac{r^2}{r^2 + \sigma^2} \right) \mu + \left(\frac{\sigma^2}{r^2 + \sigma^2} \right) \nu$
6. ...something similar for the variance: $\sigma^{2'} = \frac{\sigma^2 r^2}{\sigma^2 + r^2} \Rightarrow \frac{1}{\sigma^{2'}} = \frac{1}{r^2} + \frac{1}{\sigma^2}$

Weighted combination

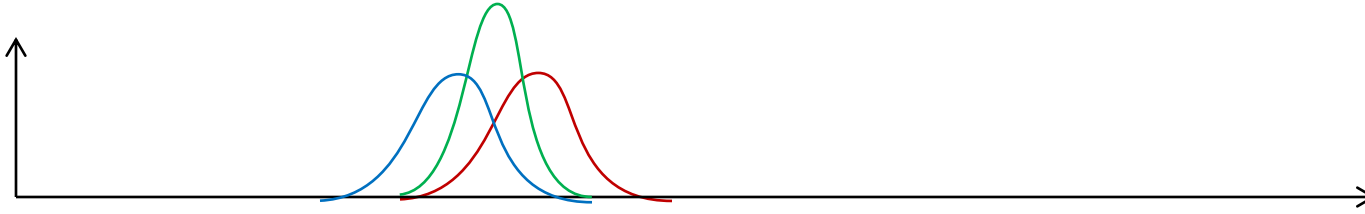
Additive information

NOTE: Note that the product of two normal random variables is NOT normal, but the product of their PDFs is proportional to the PDF of another normal.

Prior Position Estimate (10,4)

Measurement Estimate (12, 4)

Example



$$\mu' = \frac{\mu r^2 + v \sigma^2}{r^2 + \sigma^2}$$

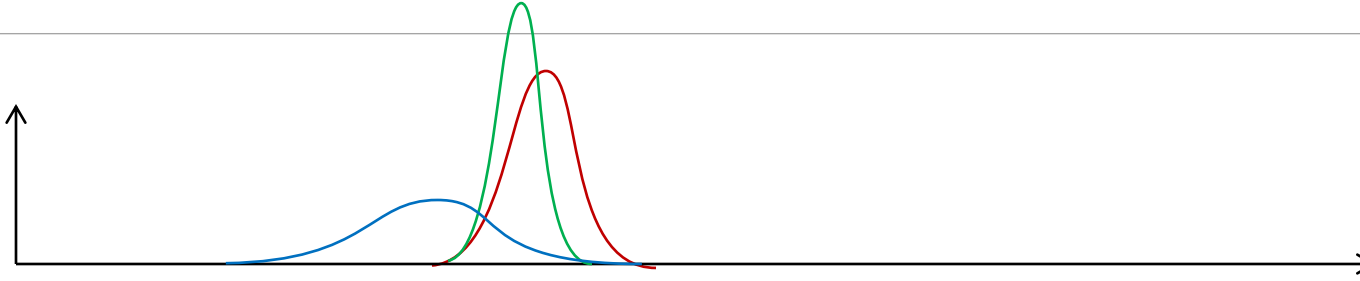
$$\sigma^{2'} = \frac{\sigma^2 r^2}{\sigma^2 + r^2}$$

$$\begin{aligned}\mu' &= 11 \\ \sigma^{2'} &= 2\end{aligned}$$

Prior Position Estimate (10, 8)

Measurement Estimate (12, 4)

Example



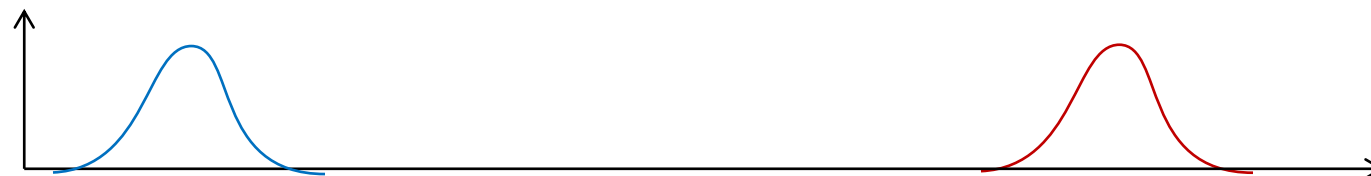
$$\mu' = \frac{\mu r^2 + v \sigma^2}{r^2 + \sigma^2}$$

$$\sigma^{2'} = \frac{\sigma^2 r^2}{\sigma^2 + r^2}$$

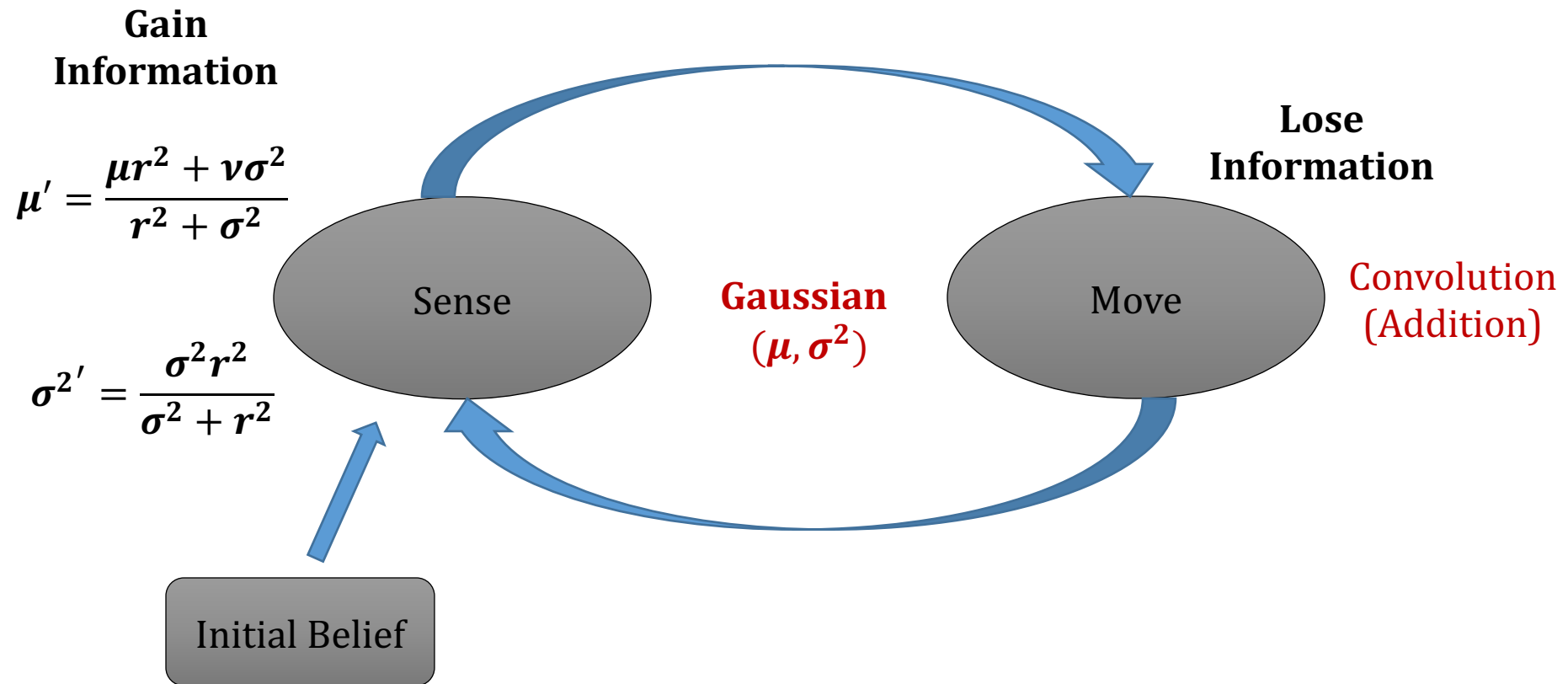
$$\mu' = 11.33$$

$$\sigma^{2'} = 2.66$$

What will happen here?

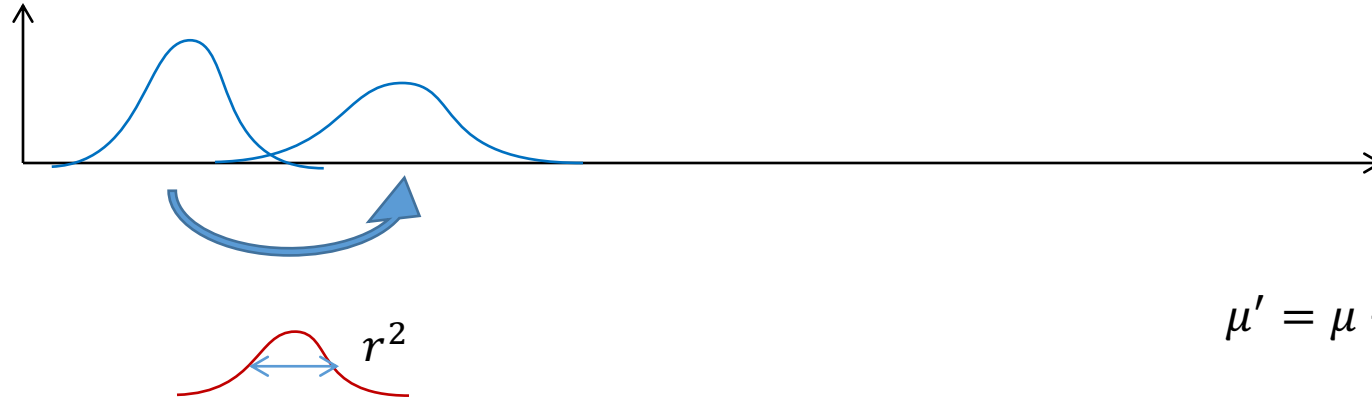


Kalman Filter



Motion Update

- For motion u



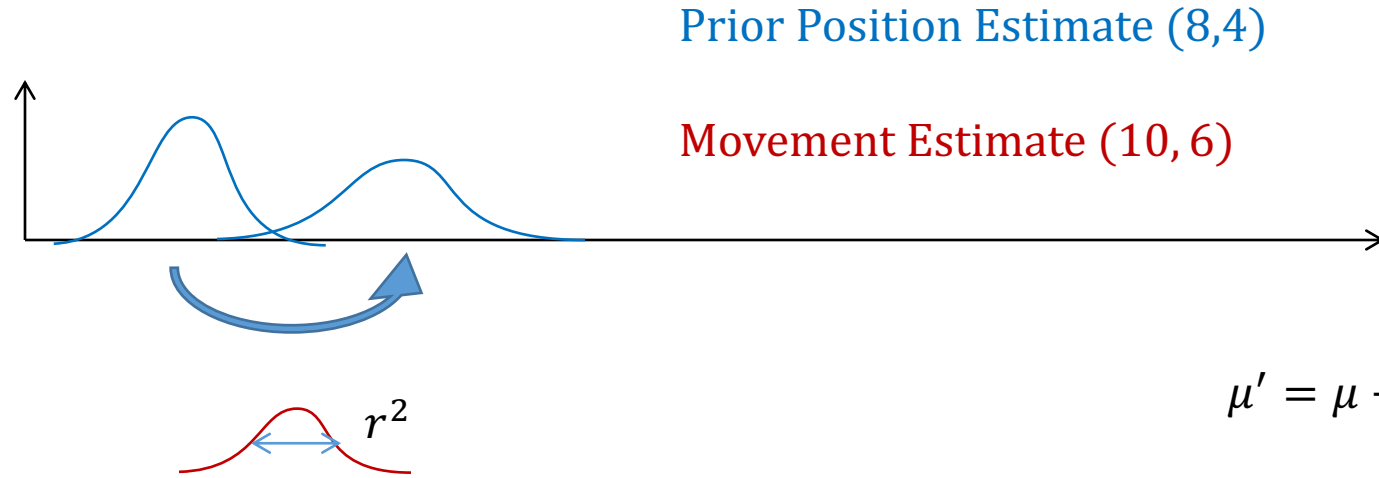
$$\mu' = \mu + u$$

$$\sigma^{2'} = \sigma^2 + r^2$$

Model of motion noise

Example

- For motion u



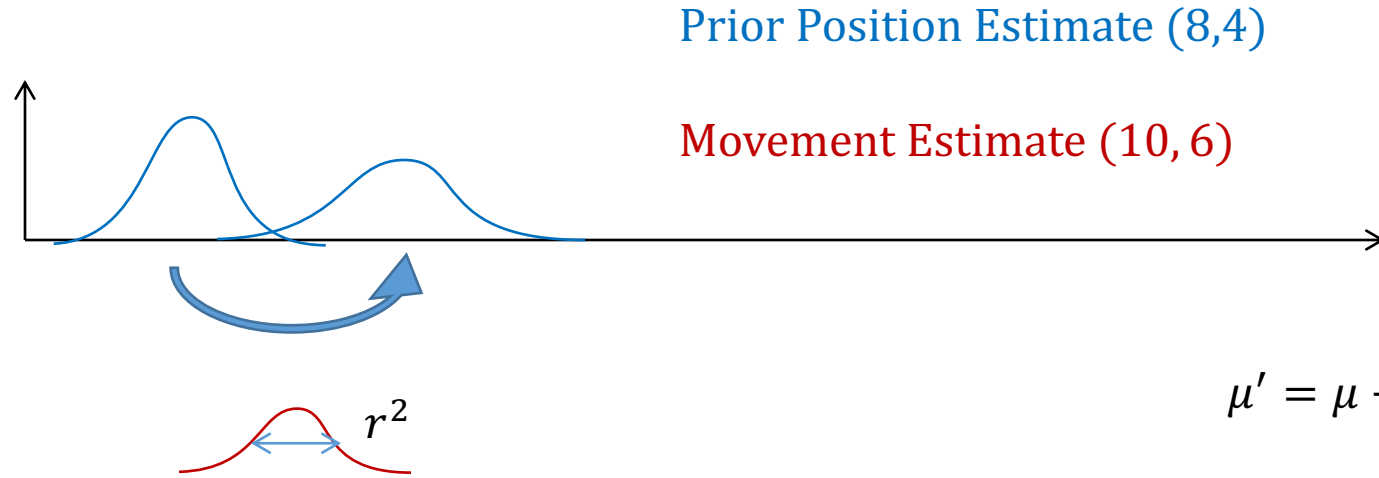
$$\mu' = \mu + u$$

$$\sigma^{2'} = \sigma^2 + r^2$$

Model of motion noise

Example

- For motion u



$$\mu' = \mu + u$$

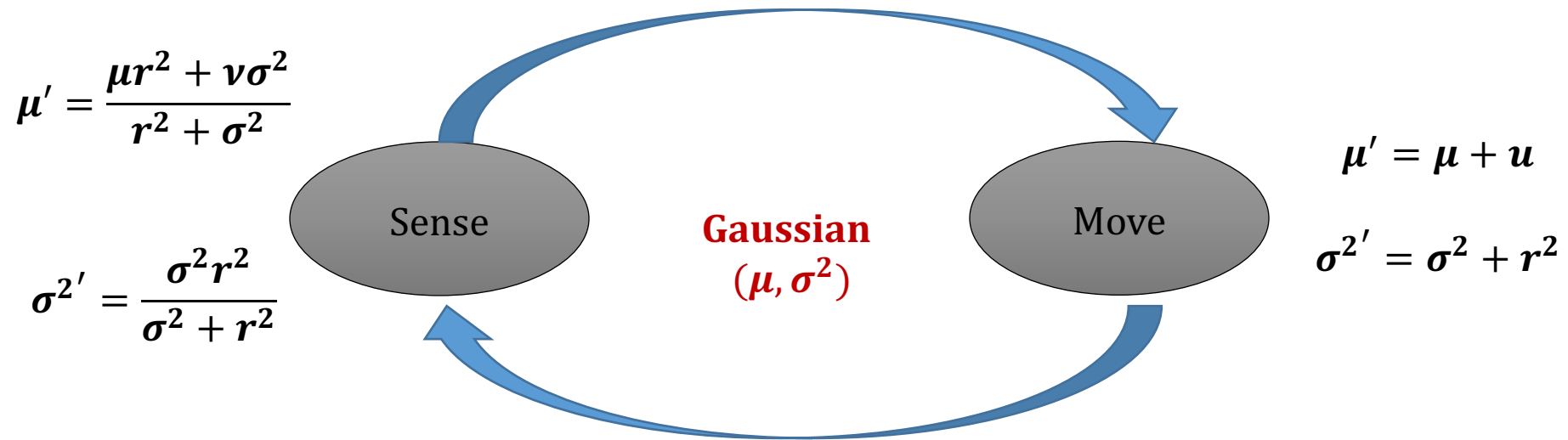
$$\sigma^{2'} = \sigma^2 + r^2$$

Model of motion noise

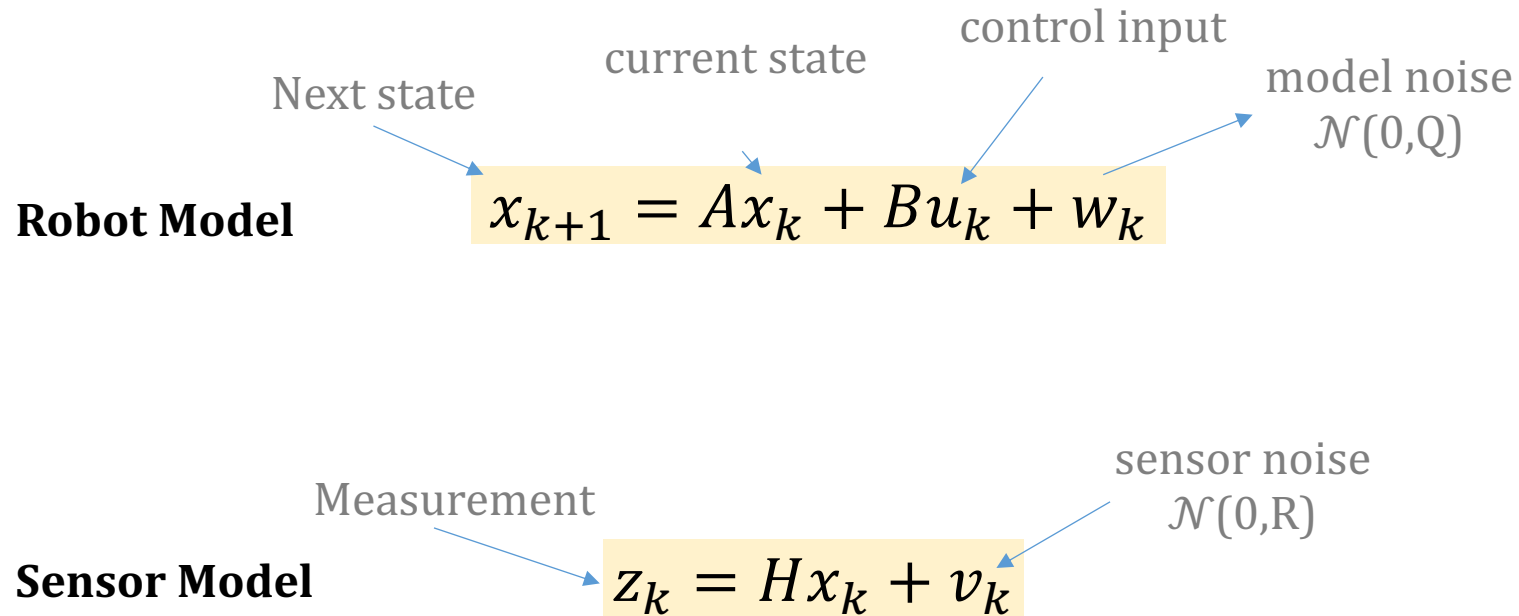
$$\mu' = 18$$

$$\sigma^{2'} = 10$$

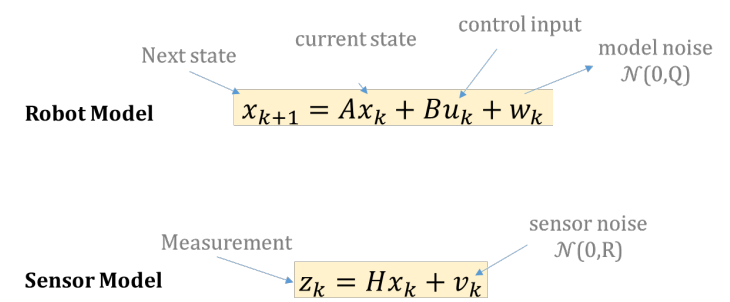
1D Kalman Filter



Now let's apply the same idea to a 2 dimensional problem...



Kalman Filter



The Kalman Filter is a two-step process:

1. *Time Update* or *Prediction* step:

predicted state
and covariance

$$\begin{aligned}\bar{x}_k &= Ax_{k-1} + Bu_k \\ \bar{P}_k &= AP_{k-1}A^T + Q\end{aligned}$$

2. *Measurement Update*:

Kalman gain

updated state
and covariance

$$\begin{aligned}K_k &= \bar{P}_k H^T (H \bar{P}_k H^T + R)^{-1} \\ \hat{x}_k &= \bar{x}_k + K_k (z_k - H \bar{x}_k) \\ P_k &= (I - K_k H) \bar{P}_k\end{aligned}$$

Proportional to
1. sensor accuracy
2. prediction covariance

Note:

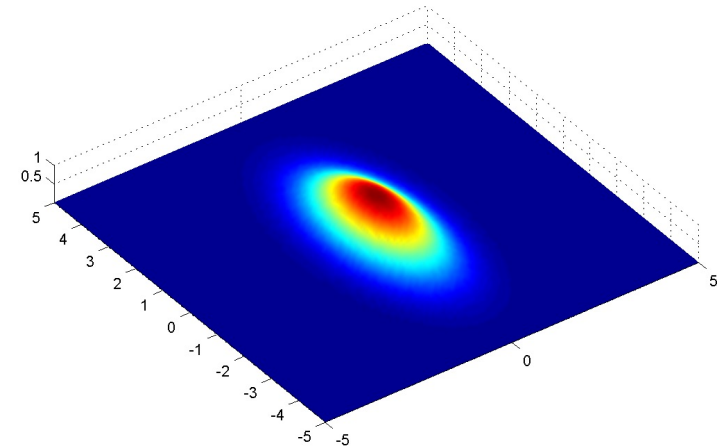
$\bar{x}_k \Rightarrow$ the state after the **Time Update**

$\hat{x}_k \Rightarrow$ the state after a **Measurement Update**.

Kalman Filter in Multiple Dimensions

- $\mu \rightarrow \begin{bmatrix} \mu_0 \\ \mu_1 \\ \dots \end{bmatrix}$ (we will call this state x)

- $\sigma^2 \rightarrow \Sigma = \begin{bmatrix} \sigma_1^2 & \sigma_{21} & \dots \\ \sigma_{12} & \dots & \dots \\ \dots & \dots & \dots \end{bmatrix}$ covariance matrix

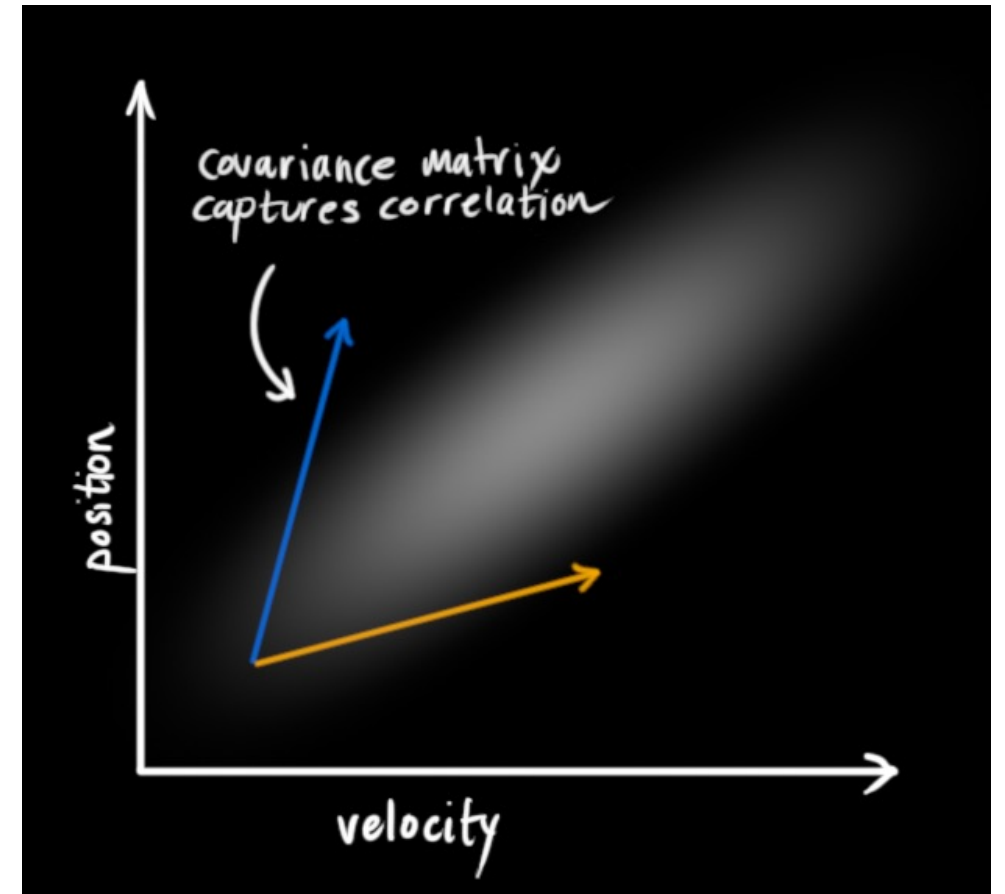


2D Gaussian

Covariance Matrix

- Along the diagonal, σ_i^2 are variances.
- Off-diagonal $\sigma_{i,j}$ are essentially correlations

$$\begin{bmatrix} \sigma_1^2 & \sigma_{1,2} & & \sigma_{1,N} \\ \sigma_{2,1} & \sigma_2^2 & & \\ & & \ddots & \vdots \\ \sigma_{N,1} & & \dots & \sigma_N^2 \end{bmatrix}$$



Source: [How a Kalman filter works, in pictures | Bzarg](#)

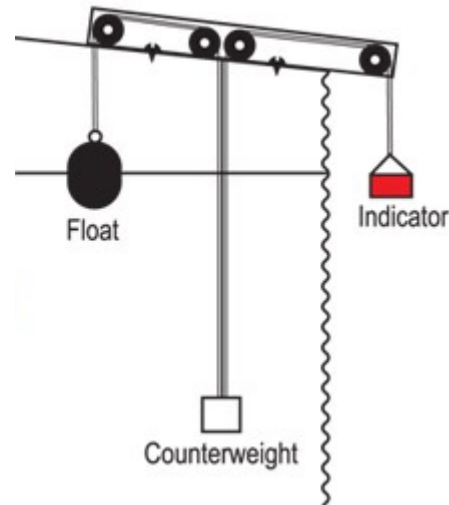
Implementing a Kalman Filter

1. Break your problem down to the mathematical basics.
2. Model the **state** process, how do you measure state?
3. How does the model behave over **time**?
4. Model the **measurement** process, what are your sensors measuring and what do they tell you about the state?
5. Model the **noise** for both the state and measurement process. The base Kalman filter assumes Gaussian (white) noise.
6. Test the filter, is it behaving as it should?
7. Refine filter. Try to change the noise parameters. If necessary go back to state process and refine there.

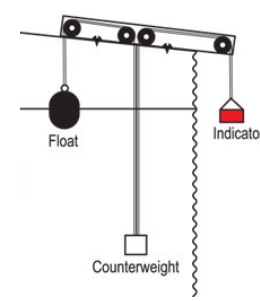
Example Problem

Predicting Water Level

- We want to estimate how much water is in a tank
- We have noisy measurements from a float
- The water level in the tank may be changing (filling, emptying)
- Let's assume the water fill rate is constant



Model the State



- State vector at time k :

$$\mathbf{x}_k = \begin{bmatrix} l_k \\ f_k \end{bmatrix}$$

← water level

← flow rate ($f_k = \frac{dl_k}{dt}$)

- How does the state change over time? How do we calculate current state based on previous state?
 - We could just say the state stays the same, $\mathbf{x}_k = \mathbf{x}_{k-1}$, but that's not very accurate...
 - A more accurate model would say that the current state changes in some way relative to previous state:

$$\mathbf{x}_k = A\mathbf{x}_{k-1}$$

- Let's apply this to our example:

$$\begin{bmatrix} l_k \\ f_k \end{bmatrix} = \begin{bmatrix} ? & ? \\ ? & ? \end{bmatrix} \begin{bmatrix} l_{k-1} \\ f_{k-1} \end{bmatrix}$$

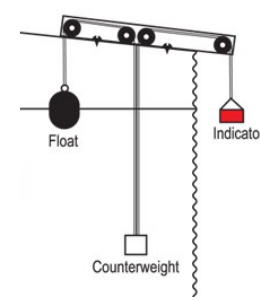
Water level is defined by previous water level plus the fill rate:

$$l_k = l_{k-1} + tf_{k-1}$$

Fill rate is constant:

$$f_k = f_{k-1}$$

Model the State



- State vector at time k :

$$\mathbf{x}_k = \begin{bmatrix} l_k \\ f_k \end{bmatrix}$$

water level

flow rate ($f_k = \frac{dl_k}{dt}$)

- How does the state change over time? How do we calculate current state based on previous state?
 - We could just say the state stays the same, $\mathbf{x}_k = \mathbf{x}_{k-1}$, but that's not very accurate...
 - A more accurate model would say that the current state changes in some way relative to previous state:

$$\mathbf{x}_k = A\mathbf{x}_{k-1}$$

- Let's apply this to our example:

$$\begin{bmatrix} l_k \\ f_k \end{bmatrix} = \begin{bmatrix} 1 & t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} l_{k-1} \\ f_{k-1} \end{bmatrix}$$

Water level is defined by previous water level plus the fill rate:

$$l_k = l_{k-1} + tf_{k-1}$$

Fill rate is constant:

$$f_k = f_{k-1}$$

Adding Uncertainty

- In addition to the robot's state, we need to model the covariance as a measure of how reliable we believe the state estimate at timestep k to be

$$P_k = \begin{bmatrix} \sigma_{ll} & \sigma_{lf} \\ \sigma_{fl} & \sigma_{ff} \end{bmatrix}$$

- In part, the value of P is determined by our *process noise model*, Q
 - Q -> represents changes to the state due to events that we can't model
 - Q is low -> state model is relatively accurate.
 - Q is high -> can't predict changes to the state just based on x_{k-1} (and later, u_k)

Model Measurement Process (Sensing)

- What are your sensors measuring and what do they tell you about the state?
- In the ideal world, we could directly sense all of the state information: $z_k = x_k$
- In reality, we typically can only sense part of the state $z_k = Hx_k$
 - We use H to define the relationship between what we are sensing and the actual state
- Also, our measurements are usually noisy, so $z_k = Hx_k + v_k$ for some measure of noise v_k
- Our domain allows us to measure only the water level, but not the fill rate. So our actual sensor model is:

$$z_k = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} l_k \\ f_k \end{bmatrix} + v_k$$

What we have so far...

Time/State (motion) model:

$$x_k = Ax_{k-1} + w_k$$

$$\begin{bmatrix} l_k \\ f_k \end{bmatrix} = \begin{bmatrix} 1 & t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} l_{k-1} \\ f_{k-1} \end{bmatrix} + w_k$$

$\swarrow \mathcal{N}(0, Q)$

Measurement:

$$z_k = Hx_k + v_k$$

$$z_k = [1 \quad 0] \begin{bmatrix} l_k \\ f_k \end{bmatrix} + v_k$$

$\swarrow \mathcal{N}(0, r)$

Time Update (Motion)

$$\bar{x}_k = Ax_{k-1}$$

$$\bar{P}_k = AP_{k-1}A^T + Q$$

Measurement Update

$$K_k = \bar{P}_k H^T (H \bar{P}_k H^T + R)^{-1}$$

$$\hat{x}_k = \bar{x}_k + K_k(z_k - H\bar{x}_k)$$

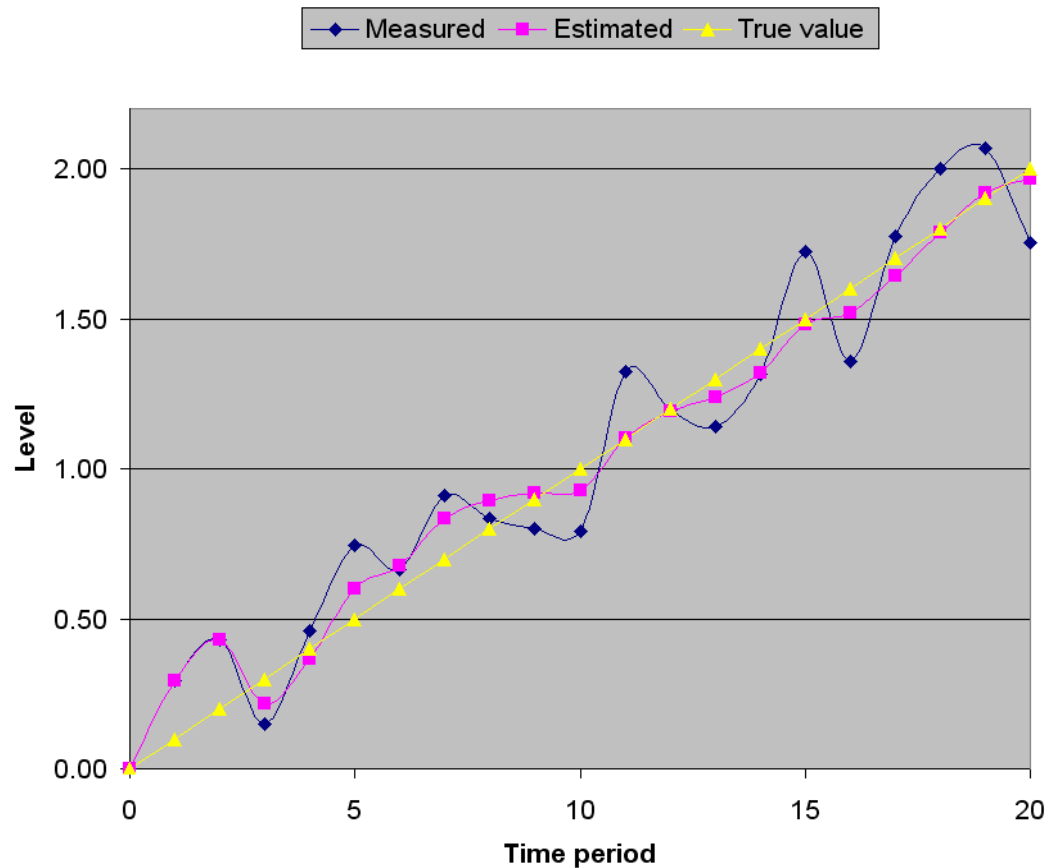
$$P_k = (I - K_k H) \bar{P}_k$$

Let's see what happens when we apply these formulas...

- First, we need to define some of the variables.
 - $t = 1$ (we'll calculate the water level at 1Hz)
 - $r = .1$ (measurement noise covariance)
 - $Q = \begin{bmatrix} 0 & 0 \\ 0 & 0.00001 \end{bmatrix}$ (very little process noise covariance)
- Let's assume that we have no idea what the initial state of the water tank is, so:
 - $x_0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$
 - $P_0 = \begin{bmatrix} 1000 & 0 \\ 0 & 1000 \end{bmatrix}$
 - *this indicates:*
 - we're really unsure about l_0 and f_0*
 - we are assuming there is no correlation between them*

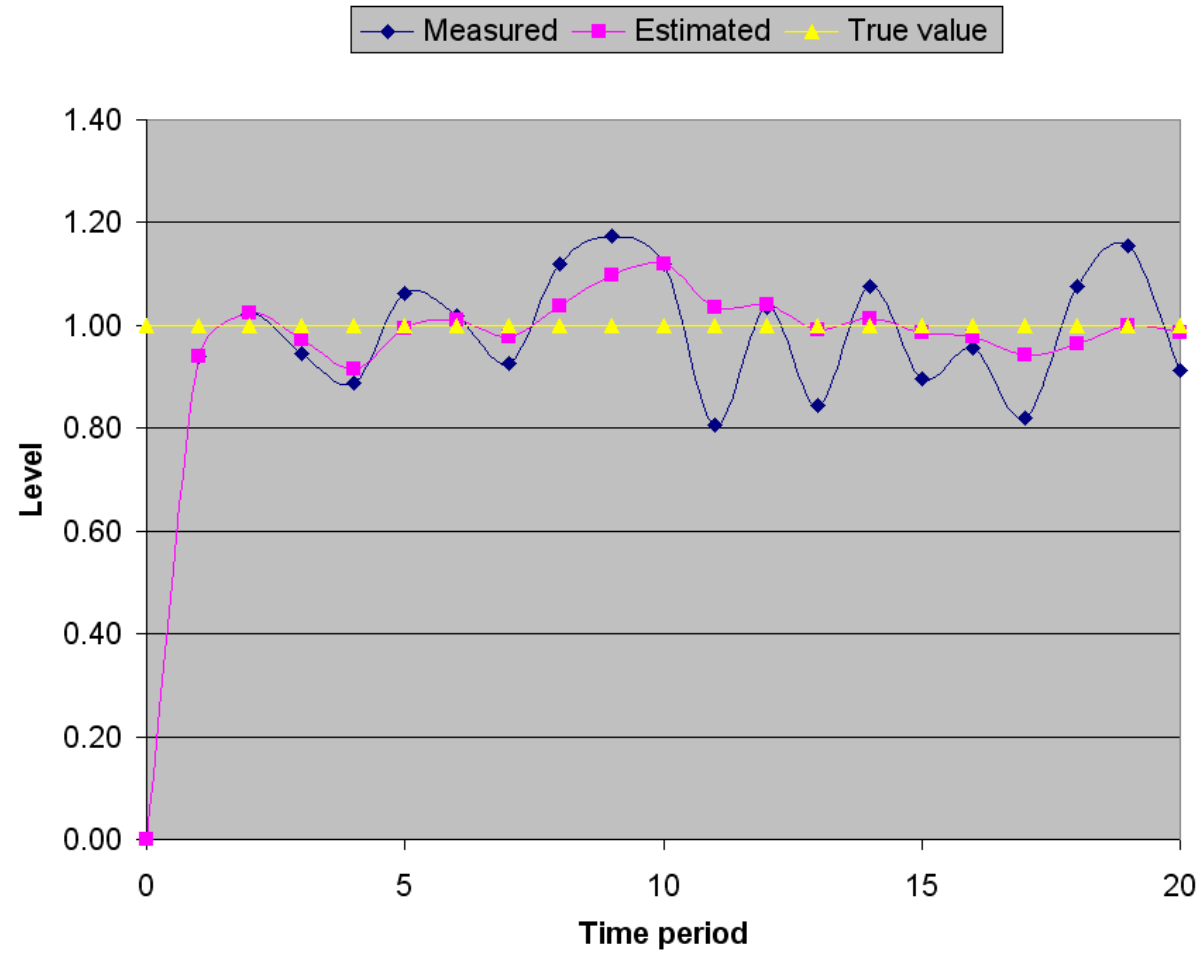
Example 1: Constant fill rate

- Run model on data with a constant fill rate of 0.1, with actual measurement noise $r = 0.3$



The Kalman filter figures out the fill rate even with a bad initial value and no ability to directly sense the fill rate.

Example 2: Fill rate of 0



Let's go back over the formulas...

- And fill in some things that were missing from our example.

Time (Motion) Update

- We had modeled state as $x_k = Ax_{k-1} + w_k$
- A more complete model would be:

$$x_k = Ax_{k-1} + Bu_k + w_k$$

- This is because the current state depends not only on the previous state, but also on any actions/control inputs from the environment.
 - u_k represents the control input at time step k
 - B tells us how to relate that to the state
 - In the water tank example, it would be like having a valve we could use to turn the water on and off.

Kalman Filter

The Kalman Filter is a two-step process:

1. In the *Time Update*, or *Prediction* step, the filter uses the previous state and the current control input to estimate the new state and its uncertainty.

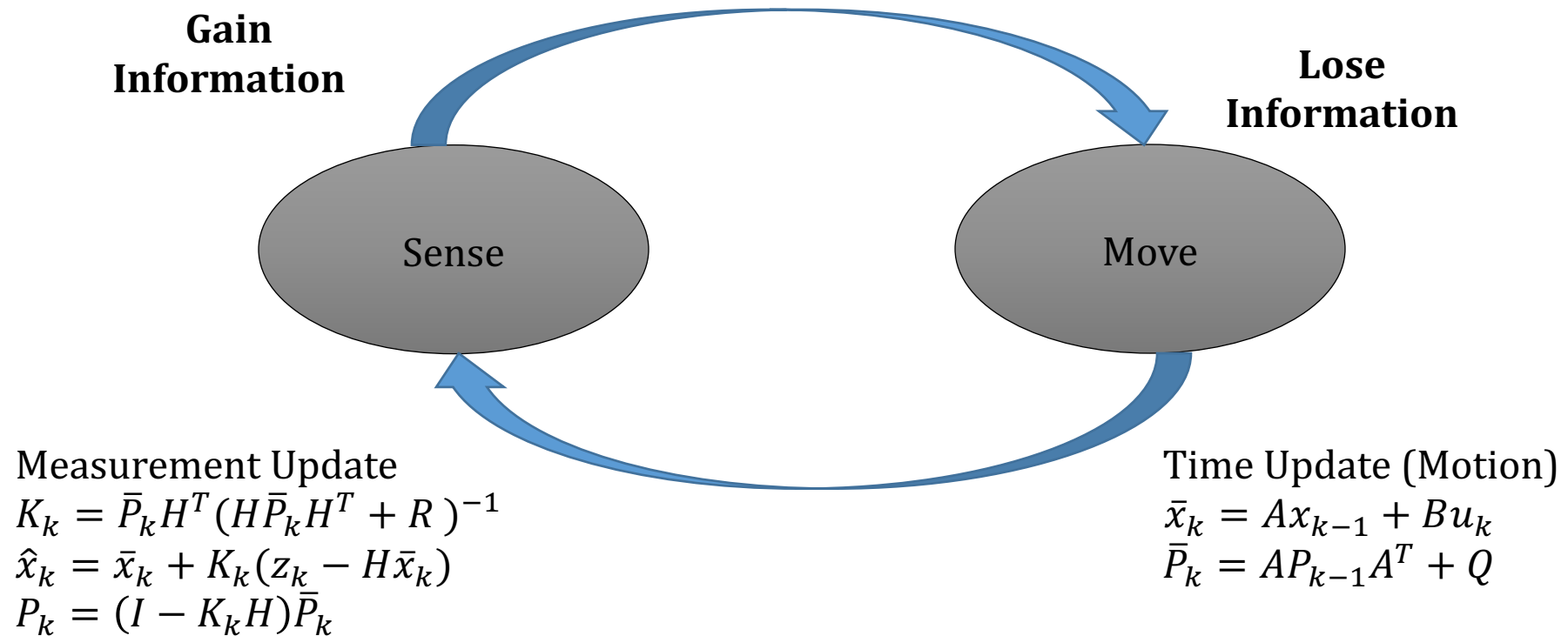
$$\begin{aligned}\bar{x}_k &= Ax_{k-1} + Bu_k \\ \bar{P}_k &= AP_{k-1}A^T + Q\end{aligned}$$

2. In the *Measurement Update* step, the filter compares observations from the sensors to its estimate and refines the state/uncertainty values.

$$\begin{aligned}K_k &= \bar{P}_k H^T (H \bar{P}_k H^T + R)^{-1} \\ \hat{x}_k &= \bar{x}_k + K_k (z_k - H \bar{x}_k) \\ P_k &= (I - K_k H) \bar{P}_k\end{aligned}$$

Note that to distinguish the state and probability values after each step, from here out we will use \bar{x}_k to represent the state after the Time Update and \hat{x}_k to represent the state after a Measurement Update.

Sense-Move Loop



Kalman Filter (measurement update)

Diagram illustrating the measurement update step of the Kalman Filter. The equation is shown in red:

$$\hat{x}_k = \bar{x}_k + K_k (z_k - H\bar{x}_k)$$

Annotations and arrows indicate the components of the equation:

- Actual sensor reading**: Points to z_k .
- Predicted sensor reading**: Points to $H\bar{x}_k$.
- Kalman gain**: Points to K_k .
- Correction based on measurement**: Points to the term $(z_k - H\bar{x}_k)$.
- Estimated state based on prior state & control input**: Points to \bar{x}_k .

Modeling Noise

- \hat{x}_k =state estimate
 - P =uncertainty covariance
 - A =state transition matrix
 - B =control input function
 - u =control input
 - Q = process/motion noise
-
- K = Kalman gain
 - z = measurement
 - H =measurement function
 - R = measurement noise
-
- I =identity matrix

terms that help model
noise are in blue

Time Update (Motion)

$$\begin{aligned}\bar{x}_k &= Ax_{k-1} + Bu_k \\ \bar{P}_k &= AP_{k-1}A^T + Q\end{aligned}$$

Measurement Update

$$\begin{aligned}K_k &= \bar{P}_k H^T (H \bar{P}_k H^T + R)^{-1} \\ \hat{x}_k &= \bar{x}_k + K_k (z_k - H \bar{x}_k) \\ P_k &= (I - K_k H) \bar{P}_k\end{aligned}$$


Note how the Kalman Gain is a measure of how
informative the current measurement is.

Modeling Noise – Simplified Example

- Let's look at a simplified example to study the noise parameters more closely... State and measurement formulation:

$$x_k = l_k = x_{k-1} + w_k$$

$$z_k = l_k + v_k$$


 $\mathcal{N}(0,r)$

- What does this say?
 - We're now modeling state only in terms of water level
 - Our model assumes the new state is always equal to the old state, $\mathbf{A}=\mathbf{1}$ (yes, this is not a good model, we will see how it works)
 - Our measurement input still lets us observe the current water level, $\mathbf{H}=\mathbf{1}$

Simplified Example Full Formulation

Time update

$$\bar{x}_k = x_{k-1}$$

$$\bar{p}_k = p_{k-1} + q$$

Measurement update

$$K_k = \bar{p}_k(\bar{p}_k + r)^{-1}$$

$$\hat{x}_k = \bar{x}_{k-1} + K_k(z_k - H\bar{x}_k)$$

$$p_k = (1 - K_k) \bar{p}_k$$

Time Update (Motion)

$$\bar{x}_k = Ax_{k-1} + Bu_k$$

$$\bar{P}_k = AP_{k-1}A^T + Q$$

Measurement Update

$$K_k = \bar{P}_k H^T (H \bar{P}_k H^T + R)^{-1}$$

$$\hat{x}_k = \bar{x}_k + K(z_k - H\bar{x}_k)$$

$$P_k = (I - K_k H) \bar{P}_k$$

Modeling Noise – Simplified Example

- Now let's run our new model with different estimates for noise and see what happens.

Scenario 1

- True level of the water in the tank $l_k = 1$
- Initial guess of the state $x_0 = 0, p_0 = 1000$
- Model noise as $r = 0.1, q = 0.0001$ (this tells the algorithm we think the sensors have 10% noise, but we believe our model to be accurate)
- Let's assume we get a first reading of $z_k = 0.9$ What happens when we run the Kalman filter?

Time update

$$\bar{x}_k = 0$$

$$\bar{p}_k = 1000 + 0.0001$$

Measurement update

$$K_k = 1000.0001(1000.0001 + 0.1)^{-1} = 0.9999$$

$$\hat{x}_k = 0 + 0.9999(0.9 - 0) = 0.8999$$

$$p_k = (1 - 0.9999)1000.0001 = 0.1$$

Time Update (Motion)

$$\bar{x}_k = Ax_{k-1} + Bu_k$$

$$\bar{P}_k = AP_{k-1}A^T + Q$$

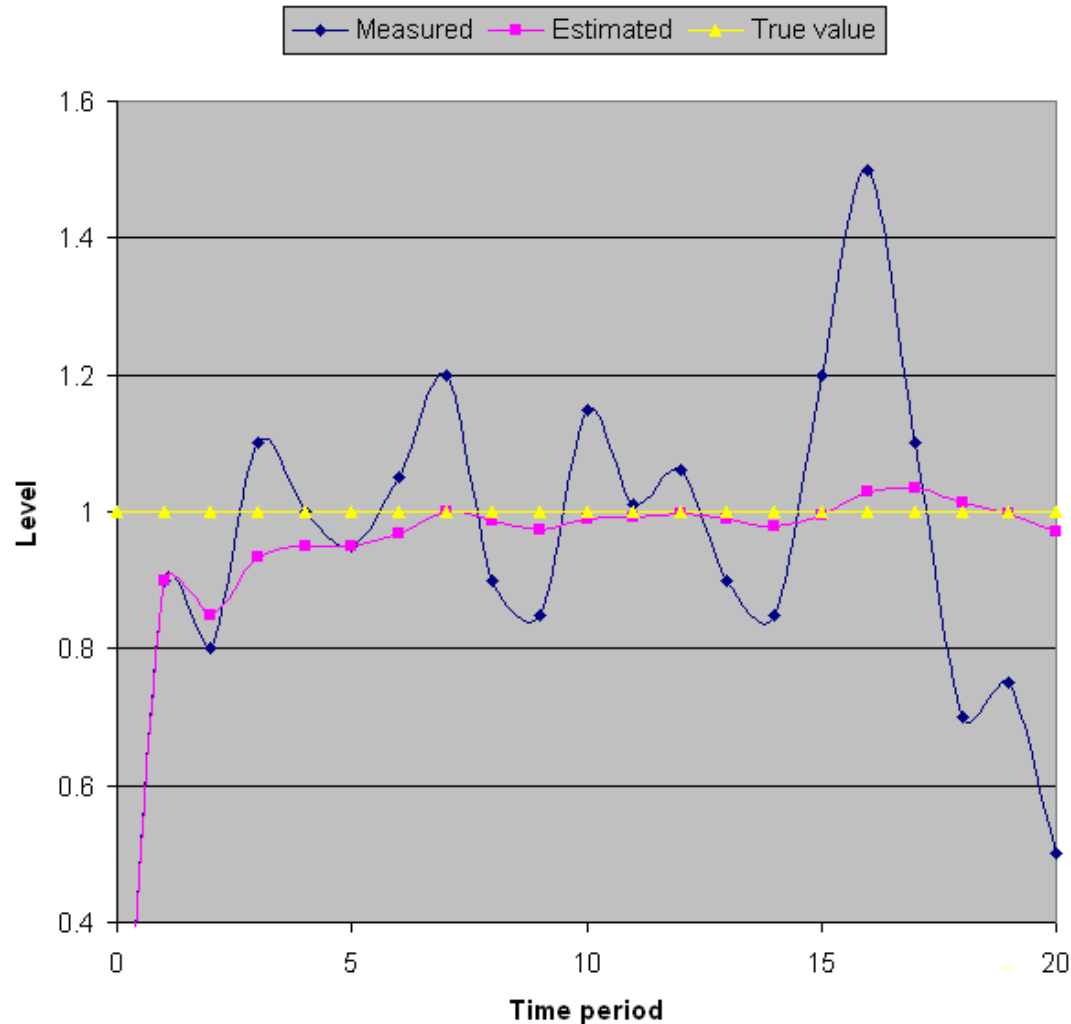
Measurement Update

$$K_k = \bar{P}_k H^T (H \bar{P}_k H^T + R)^{-1}$$

$$\hat{x}_k = \bar{x}_k + K_k(z_k - H\bar{x}_k)$$

$$P_k = (I - K_k H)\bar{P}_k$$

Scenario 1



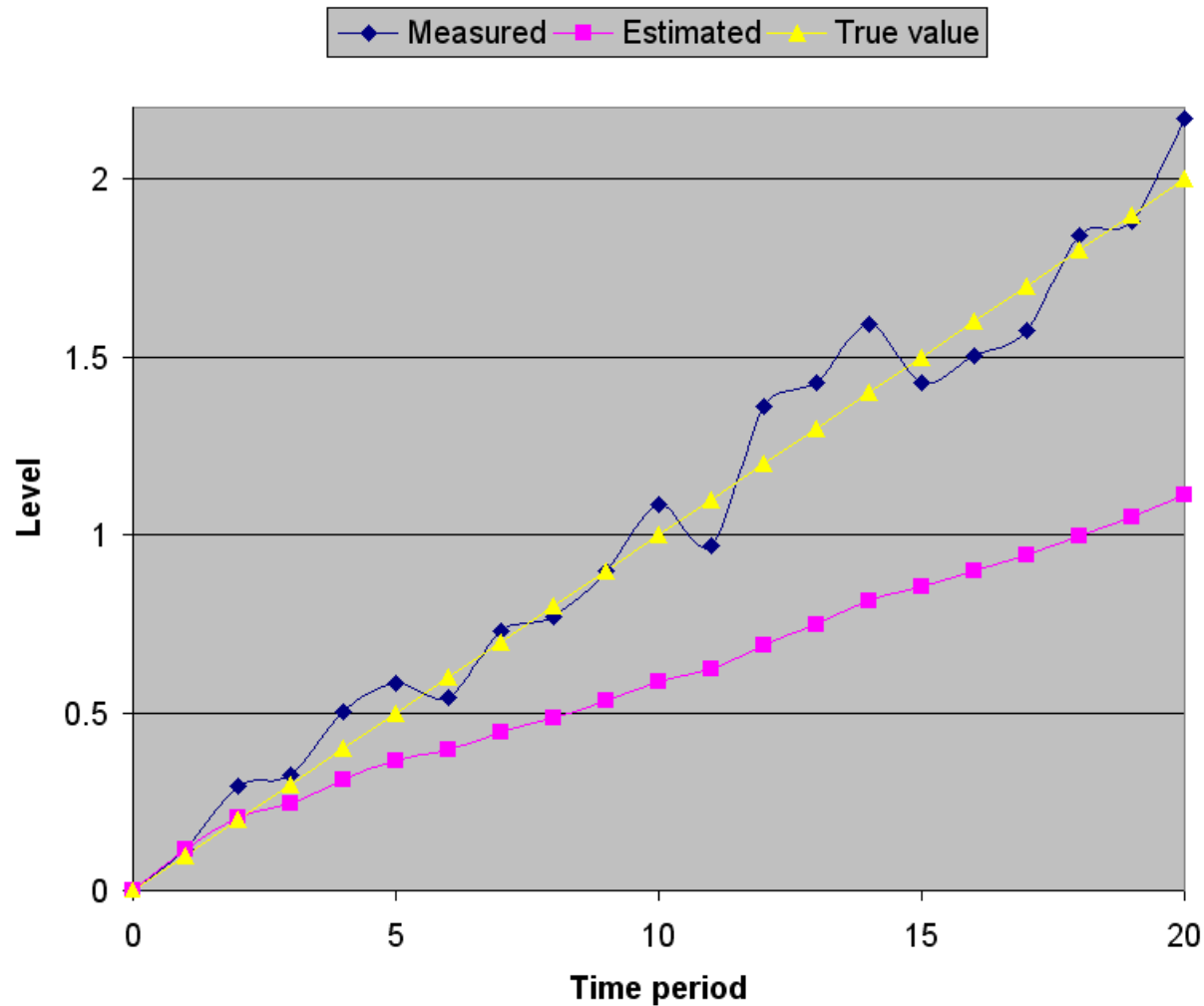
We get an estimate close to the true value, even with significant noise in the sensor readings.

... But this is partly because our (simple) model matches what was happening... let's see what happens when we change this.

Scenario 2

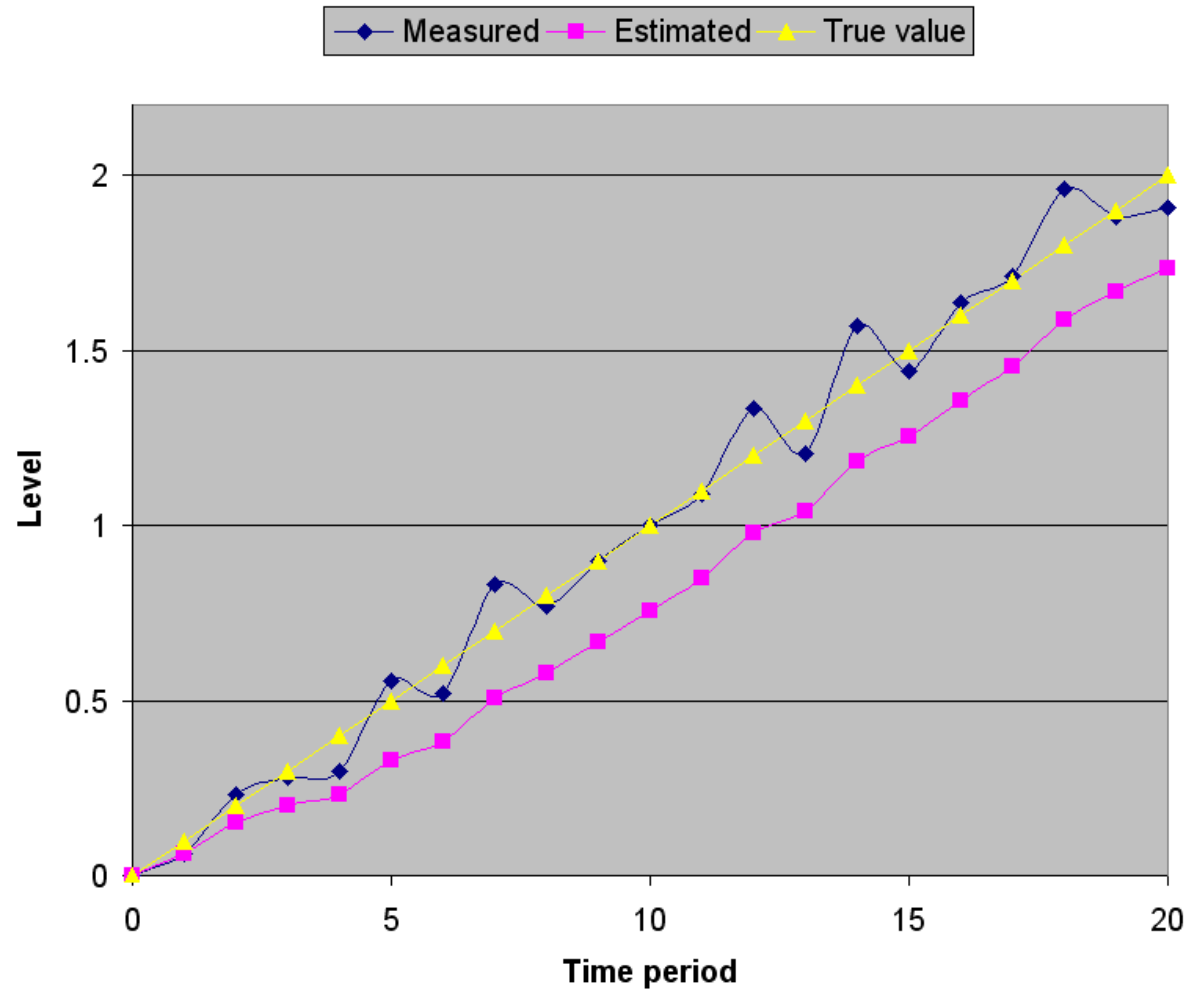
- We're going to keep the same model, but now what is *actually* happening to the tank is that it is filling with water and the water level is rising.

Scenario 2



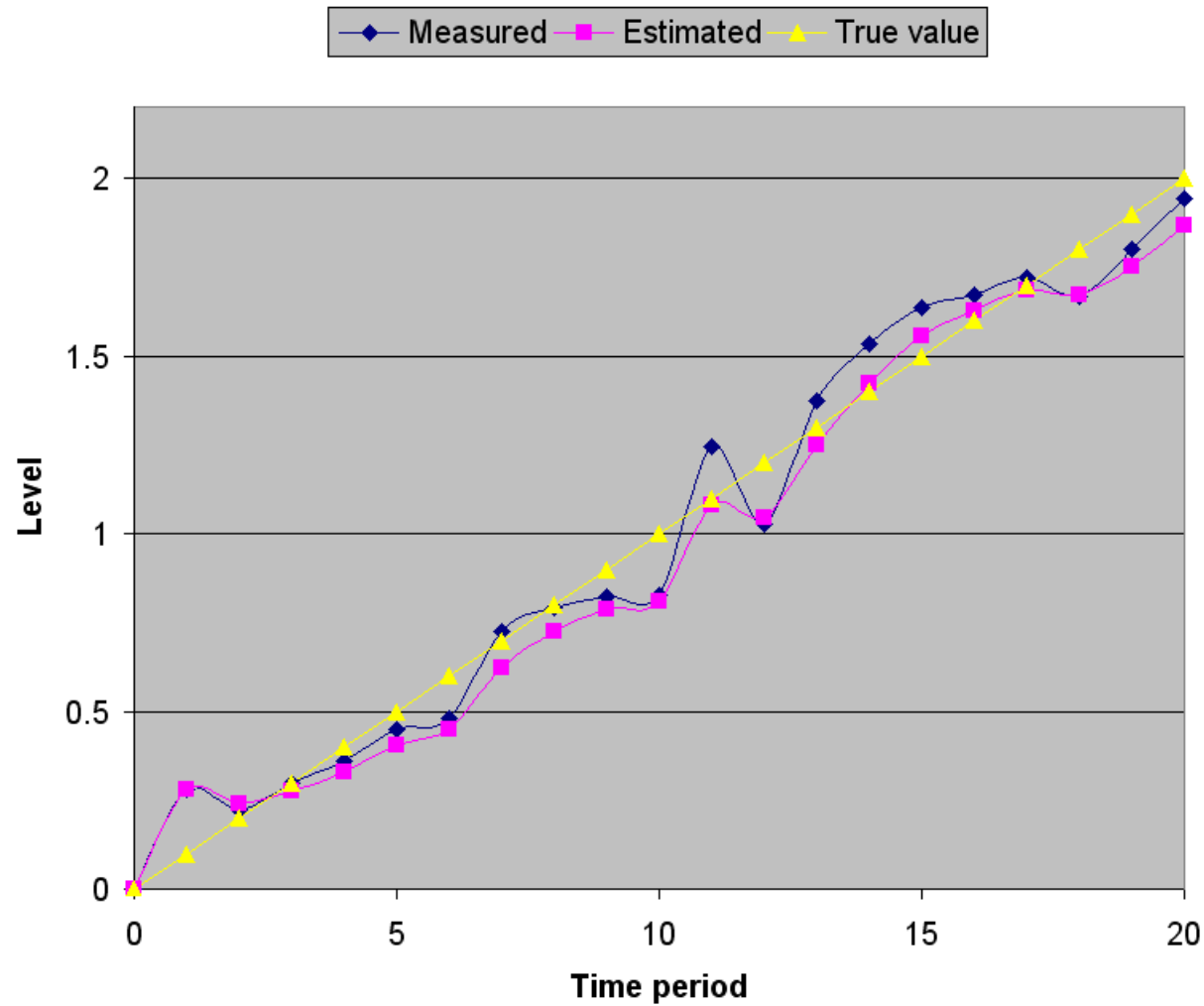
This time, we *think* our model is good ($q = 0.0001$), but actually it's very inaccurate and we are unable to model the data.

Scenario 2



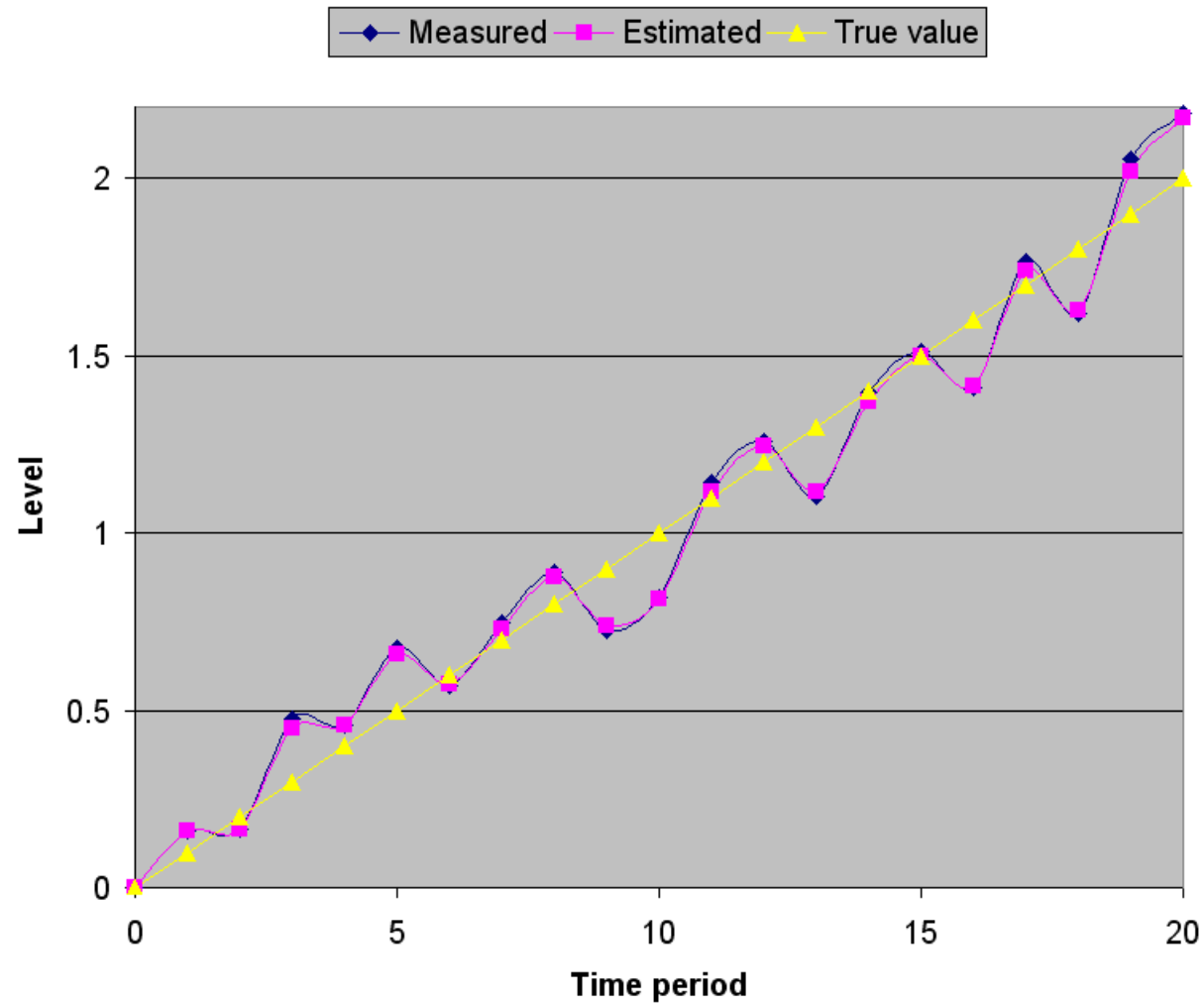
Here is the same exact model but with $q = 0.01$

Scenario 2



Now $q = 0.1$

Scenario 2



Now $q = 1$

Conclusions?

- We can have a bad model but still get the Kalman filter to produce reasonable estimates by trusting the model less and the measurements more (i.e. increasing q)
- But if we trust the measurements too much, then we're not really doing filtering anymore...

Kalman Filter Assumptions

- Linear system model
- Gaussian distribution
- White noise

For Mobile Robots

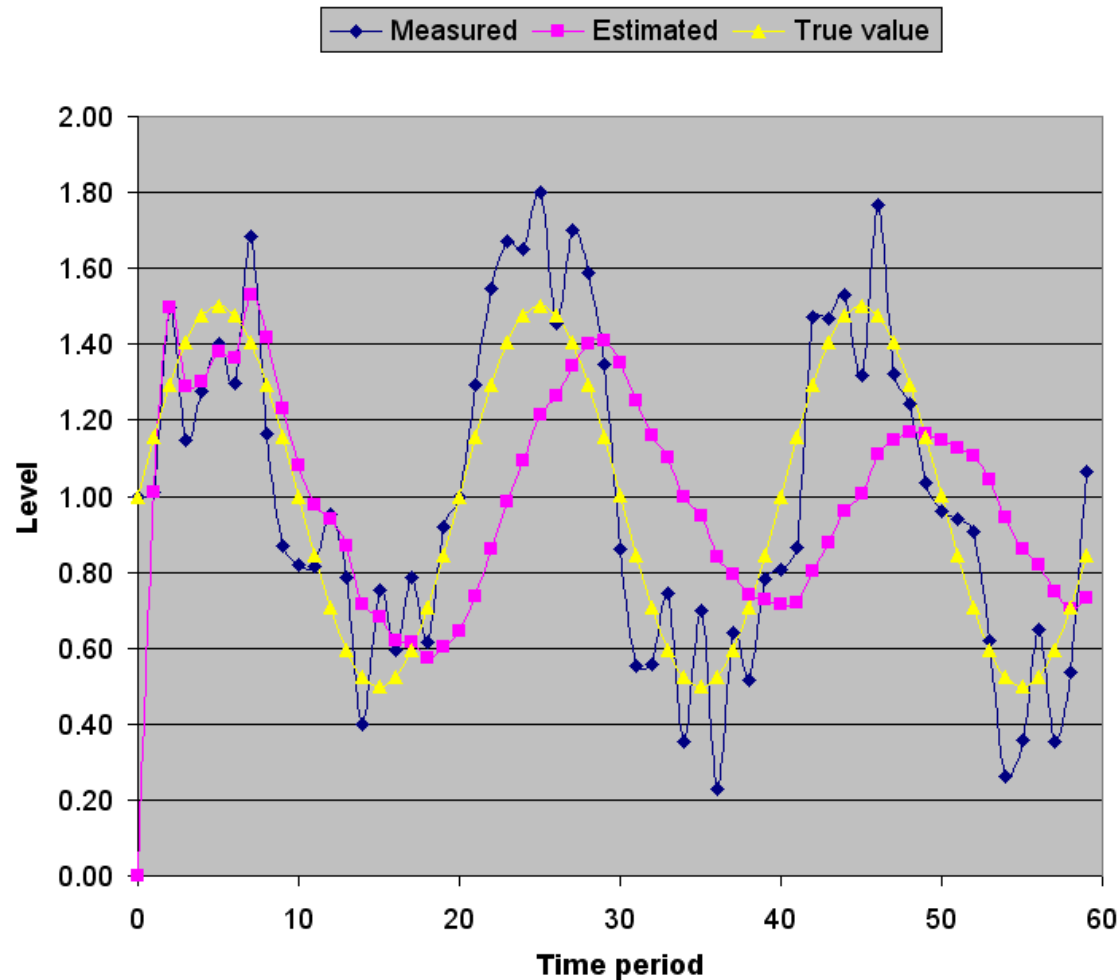
- State vector for robot moving in 2D
 - $x_k = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix}$
 - The covariance matrix P is 3x3
- Measurement updates
 - Odometry
 - Laser data
 - Vision features
 - GPS
 - ...
- Problem: Mobile robot dynamics are NOT linear (we can't formulate $x_k = Ax_{k-1}$)

Problems with the Kalman Filter

Linear Model Assumption

- Many systems of interest are highly non-linear (such as mobile robots)
- In order to model such systems, a linear process model must be generated out of the non-linear system dynamics
- The Extended Kalman filter allows the state propagation equations and the sensor models can be linearized about the current state estimate

Example of trying to model non-linear data in Water Tank Domain



Example in which water level is changing in a wave pattern within the tank (Kalman estimate is smoother than the measurement noise, but is “behind” the real data)

Extended Kalman Filter (EKF)

Time Update

$$\begin{aligned}\bar{x}_k &= f(x_{k-1}, u_k) \\ \bar{P}_k &= A P_{k-1} A^T + Q\end{aligned}$$

Measurement Update

$$\begin{aligned}K_k &= \bar{P}_k H^T (H \bar{P}_k H^T + R)^{-1} \\ \hat{x}_k &= \bar{x}_k + K_k (z_k - h(\bar{x}_k)) \\ P_k &= (I - K_k H) \bar{P}_k\end{aligned}$$

- Where A and H are the Jacobian of f and h , respectively, with respect to x .
- Process noise w is drawn from $N(0, Q)$, with covariance matrix Q .
- Measurement noise v is drawn from $N(0, R)$, with covariance matrix R .

What if the noise is NOT Gaussian?

Given only the mean and standard deviation of noise, the Kalman filter is the best linear estimator. Non-linear estimators may be better.

Why is Kalman Filtering so popular?

- Good results in practice due to optimality and structure.
- Convenient form for online real time processing.
- Easy to formulate and implement, given a basic understanding.

Comparison

- **Kalman Filter**

- Continuous
- Unimodal
- Harder to implement
- More efficient
- Requires a good starting guess of where the robot is to work well

- **Particle Filter**

- Continuous
- Multimodal
- Easier to implement
- Less efficient
- Does not require an accurate prior estimate

Resources

- Corke book: Appendix H
- Excellent online tutorials:
 - <https://www.bzarg.com/p/how-a-kalman-filter-works-in-pictures/>
 - <https://www.kalmanfilter.net/background.html>
- Extended Kalman Filter (advanced topic for those interested)
 - <https://simondlevy.github.io/ekf-tutorial/>