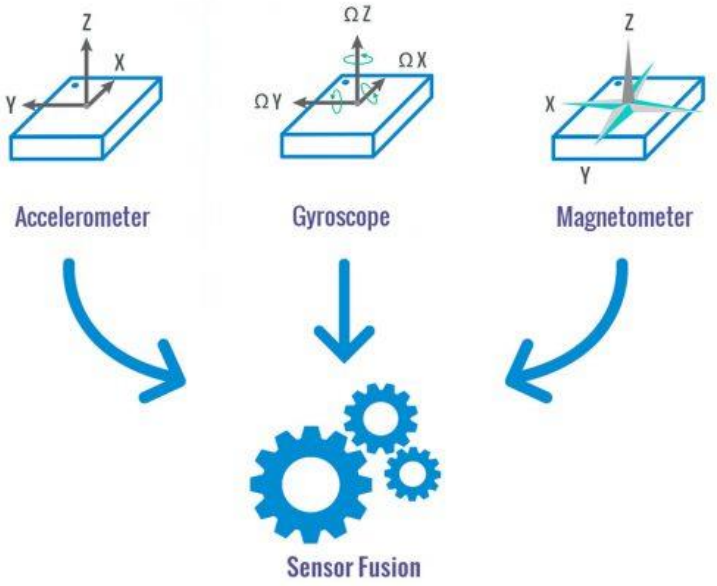

Sensor Fusion with Kalman Filters
Sam Shue, PhD
THE WILLIAM STATES LEE COLLEGE OF ENGINEERING

# Demo Code and Slides:

In various engineering applications, we use sensors to get information about the environment.

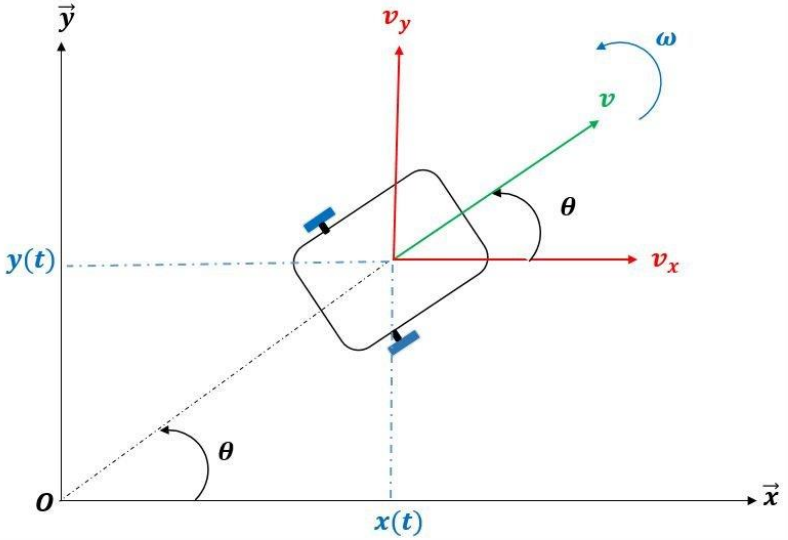
Often, these sensors provide similar information about a certain “state” we are attempting to measure.



Consider the problem of trying to determine the x,y position and x,y velocities of a robot in its environment.

The robot is equipped with 2 sensors:

- Accelerometer
- GPS



The GPS measures an X,Y coordinate directly but does it somewhat inaccurately.

The Accelerometer measures acceleration in the X and Y directions accurately but needs to be differentiated to determine position and velocity.

Both of these sensors give information about the “state” of robot – the properties I’m interested in measuring:

- X, Y Position
- X, Y Velocity

To measure the state of the robot, I can create a set of equations that utilize each sensor.

Accelerometer:

$$v_{x\_robot} = a_{x\_accel} \cdot \Delta t$$
$$x_{robot} = v_{x\_robot} \cdot \Delta t + a_{x\_accel} \cdot \Delta t^2$$
$$v_{y\_robot} = a_{y\_accel} \cdot \Delta t$$
$$y_{robot} = v_{y\_robot} \cdot \Delta t + a_{y\_accel} \cdot \Delta t^2$$

GPS:

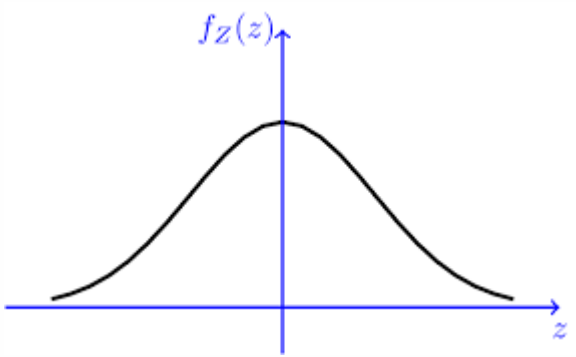
$$x_{robot} = x_{GPS}$$
$$y_{robot} = y_{GPS}$$

Now that we have the state information, why not just average the values?

A simple averaging technique could work, but the Kalman filter will provide better results!

The Kalman filter will consider the accuracies of each sensor and combine them to get an optimal estimation.

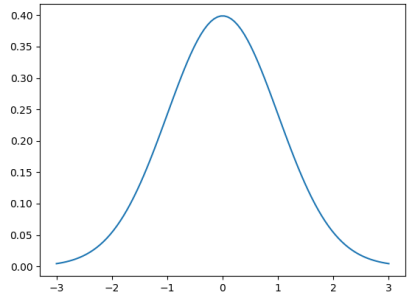
The Kalman filter models each sensor as a Gaussian random variable – meaning that the inaccuracies are modeled as a “Bell Curve”




This means that the noise associated with each measurement is Gaussian distributed.

The Gaussian probability distribution function is parametric.

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




This means the distribution is modeled by a set of parameters:

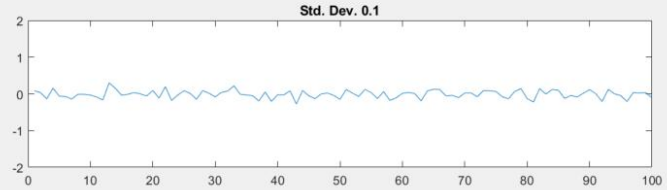
- Sigma – which represents the standard deviation
- Mean – which is the average value of the signal

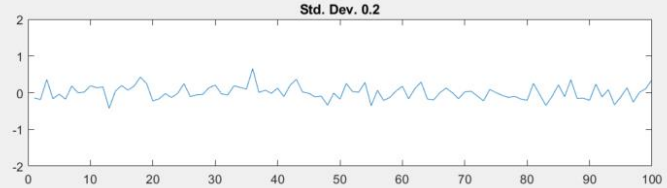
The standard deviation describes how dispersed the data is in relation to mean.

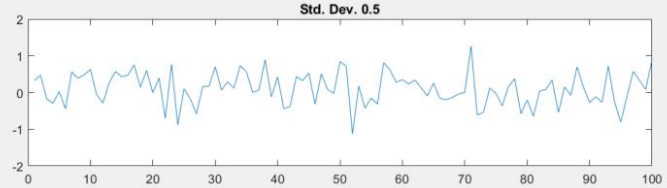
Standard deviation is the square root of the variance.

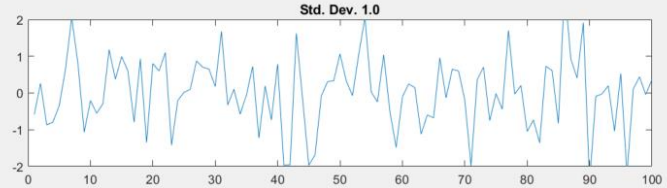
Variance is the average value of the square of the difference between each sample and the mean, given a population.

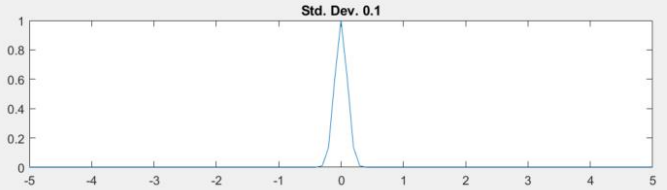
$$\sigma^2 = \frac{\sum(xi - \bar{x})^2}{N}$$

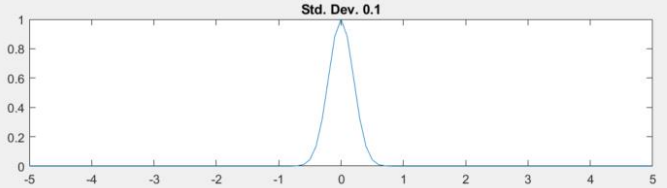

Std. Dev. 0.1

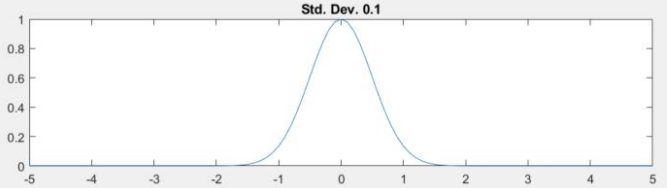

Std. Dev. 0.2

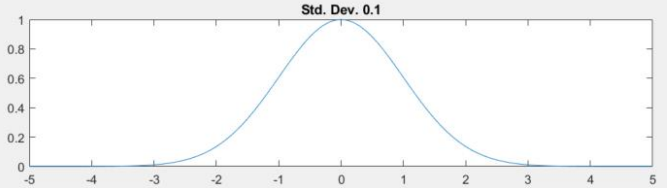

Std. Dev. 0.5


Std. Dev. 1.0


Std. Dev. 0.1

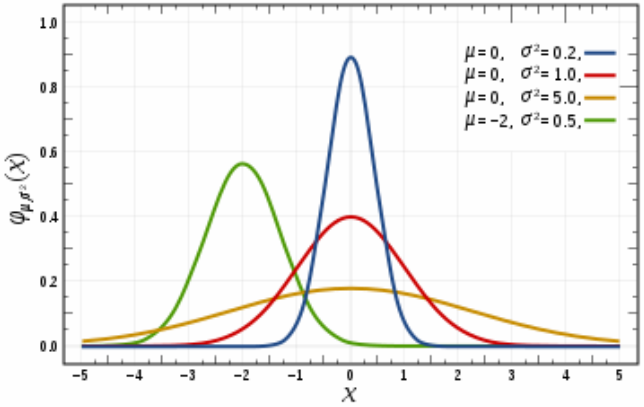

Std. Dev. 0.1


Std. Dev. 0.1

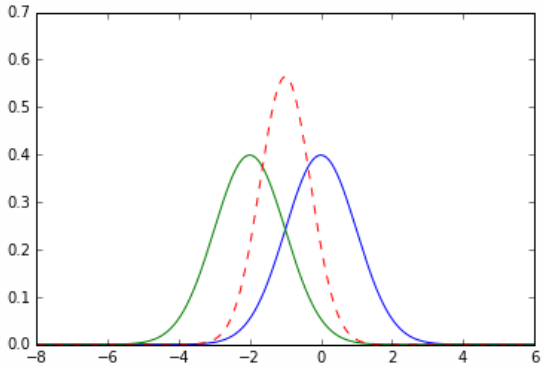

Std. Dev. 0.1

The wider the Gaussian distribution is, the more uncertain we are about any one sample.

The more narrow and “peakier” the Gaussian distribution is, the more certain we are about any one sample.



Gaussians PDFs exhibit an interesting behavior when multiplied:



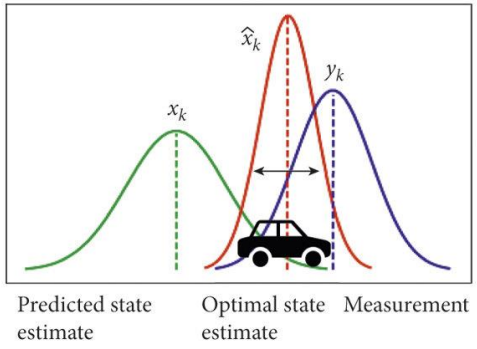

The result is yet another Gaussian PDF! This one being more certain than the combination of the two that were multiplied together.

Many types of signals tend to have Gaussian-distributed noise, or at least very close fit.

Assuming that each input is Gaussian-distributed, the Kalman filter can fuse sensor data together, weighting each input according to its certainty.

The Kalman filter has two parts:

- The “Prediction”, where the state is updated by the system dynamics
- The “Measurement”, where the predicted state is corrected by measurements from a sensor



To use a Kalman filter, 5 things are required:

- State Vector
- Covariance Matrix
- Linear Model of the System (and Control Input)
- Linear Model of the Measurement
- Variance values for the system model and sensor

The state vector is a vector containing the properties of the system you are trying to estimate.

In this example, we want to estimate the robot's x, y position and velocities:

$$x = \begin{bmatrix} p_x \\ v_x \\ p_y \\ v_y \end{bmatrix}$$

The Covariance matrix tracks the variances and covariances of each value in the state vector.

The diagonal of the covariance matrix contains the variance of each state variable. This value will indicate how certain we are of each estimate.

In this example:

$$P = \begin{bmatrix} var(p_x) & cov(p_x v_x) & cov(p_x p_y) & cov(p_x v_y) \\ cov(p_x v_x) & var(v_x) & cov(v_x p_y) & cov(v_x v_y) \\ cov(p_x p_y) & cov(v_x p_y) & var(p_y) & cov(p_y v_y) \\ cov(p_x v_y) & cov(v_x v_y) & cov(p_y v_y) & var(v_y) \end{bmatrix}$$

This matrix contains the kinematic equations, because when multiplied by x, it produces the results of the kinematic equations.

$$F = \begin{bmatrix} 1 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The Linear Model of the system contains the system's dynamics.

In this example, it will contain the basic kinematic equations.

$$x_{k|k-1} = F \cdot x_{k-1|k-1} = \begin{bmatrix} 1 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} p_x \\ v_x \\ p_y \\ v_y \end{bmatrix} = \begin{bmatrix} 1 \cdot p_x + \Delta t \cdot v_x + 0 \cdot p_y + 0 \cdot v_y \\ 0 \cdot p_x + 1 \cdot v_x + 0 \cdot p_y + 0 \cdot v_y \\ 0 \cdot p_x + 0 \cdot v_x + 1 \cdot p_y + \Delta t \cdot v_y \\ 0 \cdot p_x + 0 \cdot v_x + 0 \cdot p_y + 1 \cdot v_y \end{bmatrix}$$

$$x_{k|k-1} = \begin{bmatrix} p_x + \Delta t \cdot v_x \\ v_x \\ p_y + \Delta t \cdot v_y \\ v_y \end{bmatrix}$$

This gives us a "prediction" of the state after the next time step.

Next, we need to propagate the variances through the system.

$$P_{k|k-1} = F \cdot P_{k-1|k-1} \cdot F^T$$

The process of the prediction step will spread the variances.

Think, if we are predicting how much the robot will move based on an uncertain velocity estimation, shouldn't that make the position estimate more uncertain?

Now that we've made our predictions, we can now correct them based on a sensor measurement.

Our measurement comes from the GPS, which will provide us with an X,Y position.

$$z = \begin{bmatrix} x \\ y \end{bmatrix}$$

In order to correct my estimate, I need to generate an error term.

Since my measurement is in the form of (x,y), I will need to transform the state vector into that same form so that I can subtract them from each other.

To do this, we'll define a matrix, H, which will map the state space into the measurement space:

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Multiplying H by x will create vector that contains just the x and y components of the state vector.

This can then be subtracted from the measurement to get the error term, y

$$y = Hx_{k|k-1} - z$$

I can then use the error term to make corrections to my state, x.

But how do I determine how much to correct x? By using our confidence in both the measurement and the current state estimate after prediction.

Our confidence in the current state estimate, x, is represented by the covariance matrix, P.

We also have a matrix, R, which is used to hold the variance of the measurement, z.

Since z is a vector of length 2, R will be a 2x2 covariance matrix:

$$R = \begin{bmatrix} var(x) & cov(x,y) \\ cov(x,y) & var(y) \end{bmatrix}$$

Using this matrix, we can compute the “Kalman Gain”, which will indicate how x should change based on our confidence in the current state and the measurement:

$$K = P_{k|k-1}H^T(HP_{k|k-1}H^T + R)^{-1}$$

Using the Kalman Gain, we can update x:

$$x_{k|k} = x_{k|k-1} + Ky$$

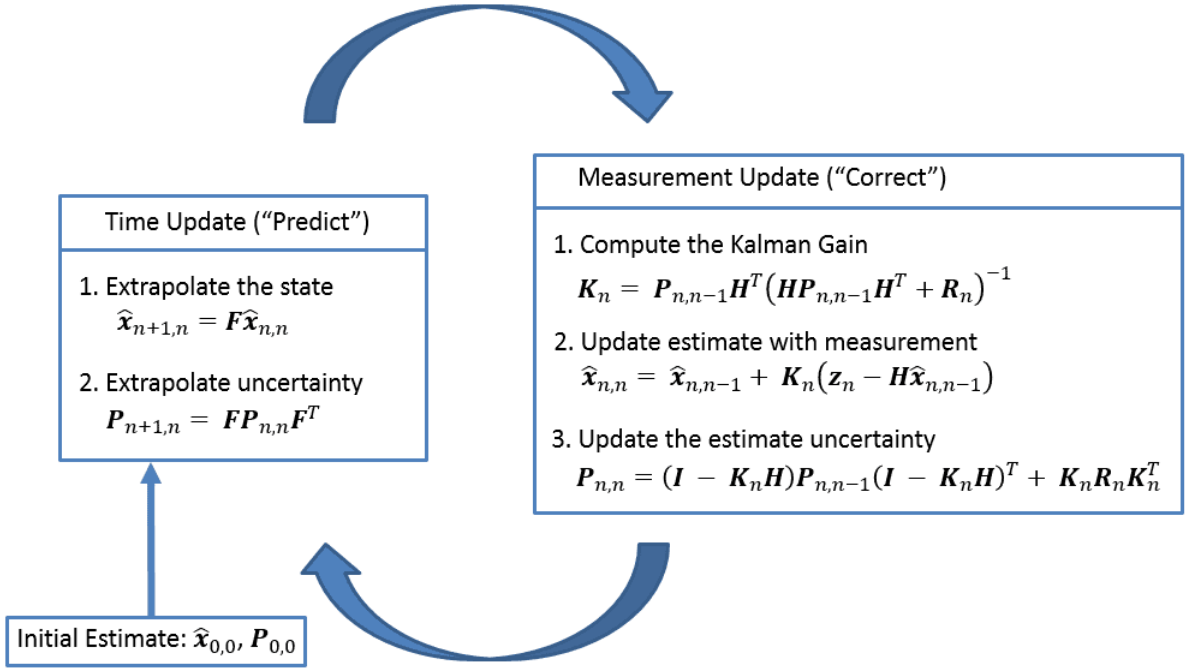
By multiplying K by y, it essentially indicates how much of the error to use to modify x weighted by the variance.

We also update P as well, based on K and the uncertainty of the measurement noise:

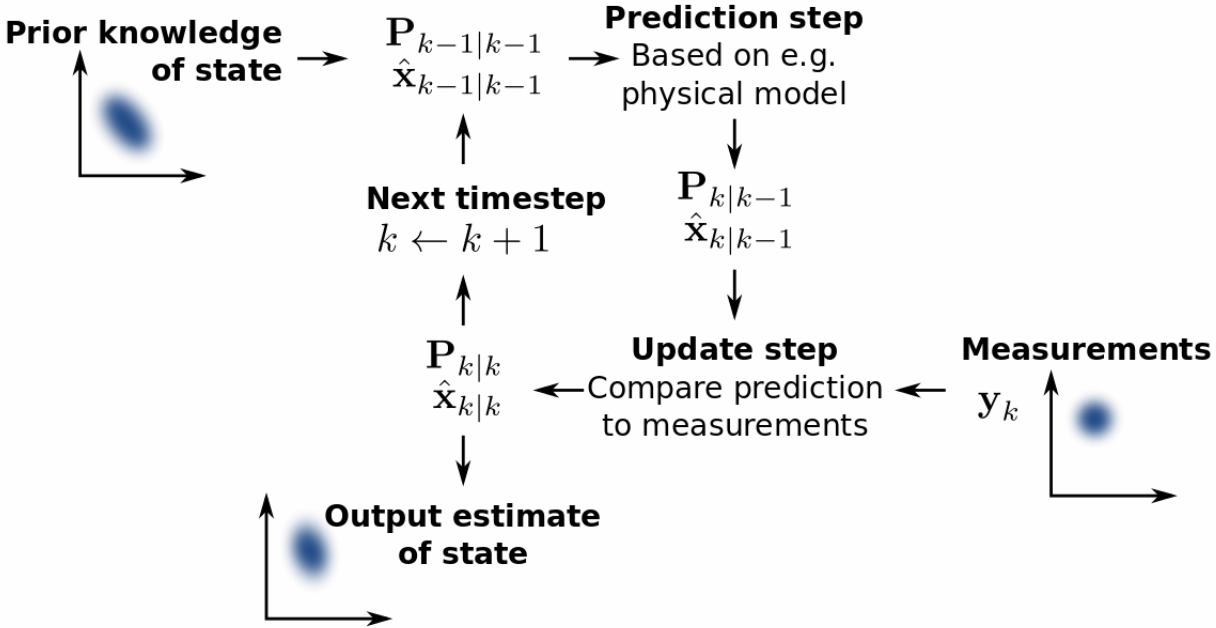
$$P_{k|k} = (I - KH)P_{k|k-1}(I - KH)^T + KRK^T$$

**This essentially applies the multiplication of two Gaussian PDFs**

And we're done! We can then iterate on this process each time the system receives a new measurement!



The process can be visually illustrated as:



But… this example only used the GPS, what happened to the accelerometer?

Where is the sensor fusion?

We can incorporate additional sensors in the measurement model.

We can also use the accelerometer as a “control” input for the prediction stage.

Control inputs are represented by a vector, u:

$$u = \begin{bmatrix} a_x \\ a_y \end{bmatrix}$$

Which contains the acceleration along the x and y axes.

We then multiply u by B, which transforms the control input into the state space:

$$B = \begin{bmatrix} \Delta t^2 & 0 \\ \Delta t & 0 \\ 0 & \Delta t^2 \\ 0 & \Delta t \end{bmatrix}$$

My prediction step now changes to:

$$x_{k|k-1} = F \cdot x_{k-1|k-1} + B \cdot u$$

The control input also introduces its own noise. This noise is represented by the covariance matrix, Q.

$$Q = F \cdot \begin{bmatrix} var(a_x) \\ var(a_y) \end{bmatrix} \cdot F^T$$

Using Q, when update the prediction step for the covariance matrix:

$$x_{k|k-1} = F \cdot x_{k-1|k-1} + B \cdot u$$

Now, the prediction step updates the x position based on not just the velocity, but also the acceleration. The acceleration also updates the velocity as well.

The rest of the process doesn't change.

Now we're fusing sensors! Assuming the noise is actually 0-mean Gaussian distributed, this method gives us the optimal estimate of the system.

(Unfortunately, the noise is often NOT 0-mean Gaussian, so we don't actually get optimal estimates)

THE WILLIAM STATES LEE COLLEGE OF ENGINEERING

# Thank You for Coming!

# **Questions?**