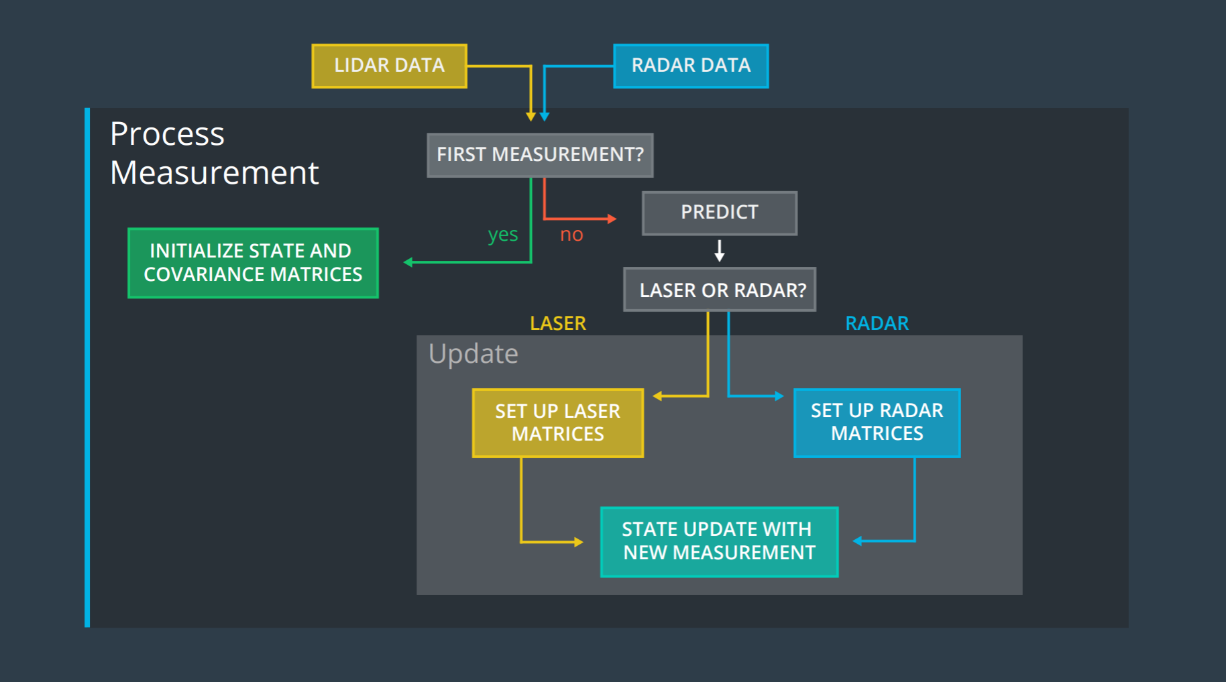
Kalman Filters don’t need to only be used with sequential measurements from one sensor. It can be used to combine measurements from different sensors.

We can use LIDAR and RADAR sensors together to overcome each of their individual weaknesses to estimate things like pedestrian location, heading, and speed.

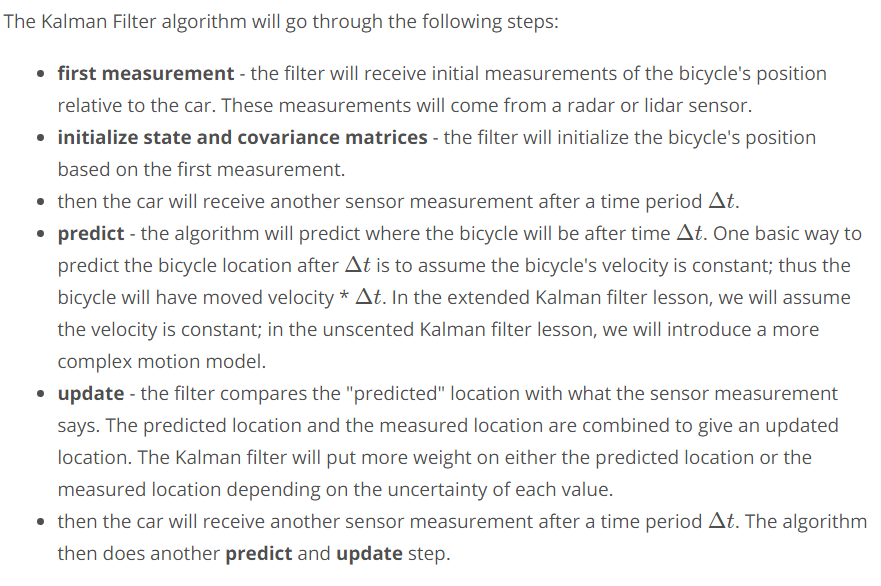
**Kalman Filter Based Sensor Fusion**

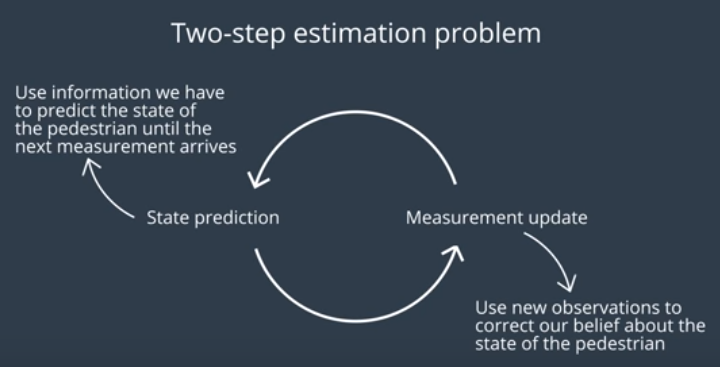


**Extended Kalman Filter**: A Kalman Filter with the capability of handling more complex motion and measurement models.

Process:

1. Sensor Info is used to estimate the state of an object in motion. The state is represented by a 2D position and a 2D velocity.
2. With each new measurement from the sensor, the estimation function is triggered to perform 2 steps of State Prediction and the Measurement Update
3. Prediction step provides a prediction of the state and its covariance. This is done by taking a lapse time between the current and previous observations.
4. Measurement Update step depends on sensor type. If the current measurement is Laser Data, a standard Kalman filter measurement step can be done. However if it is RADAR data involves a nonlinear measurement function. To handle non-linearity’s, we use different tweaks to handle the measurement step like extended **Kalman Filter Equations**.





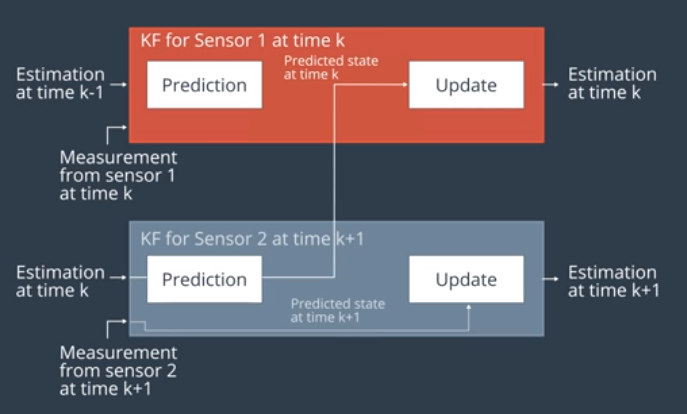
Always starts with initial state prediction, and is an endless loop of prediction and measurement steps.

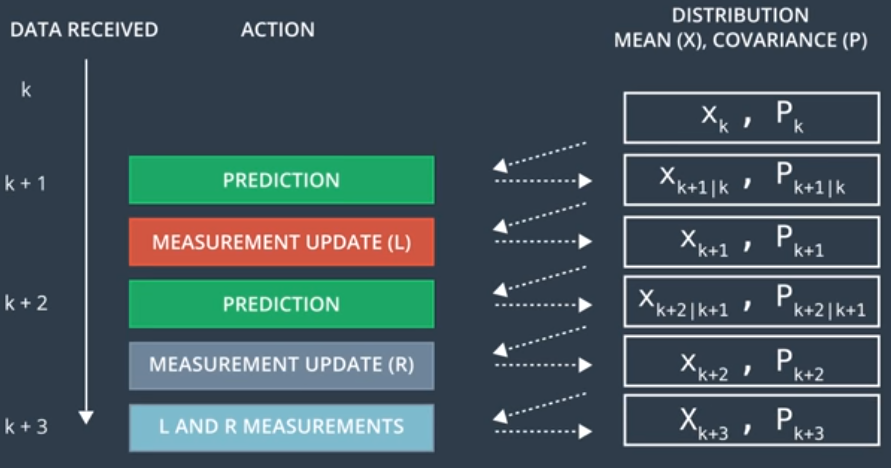
**What Happens when there are 2 sensors measuring the same pedestrian?**

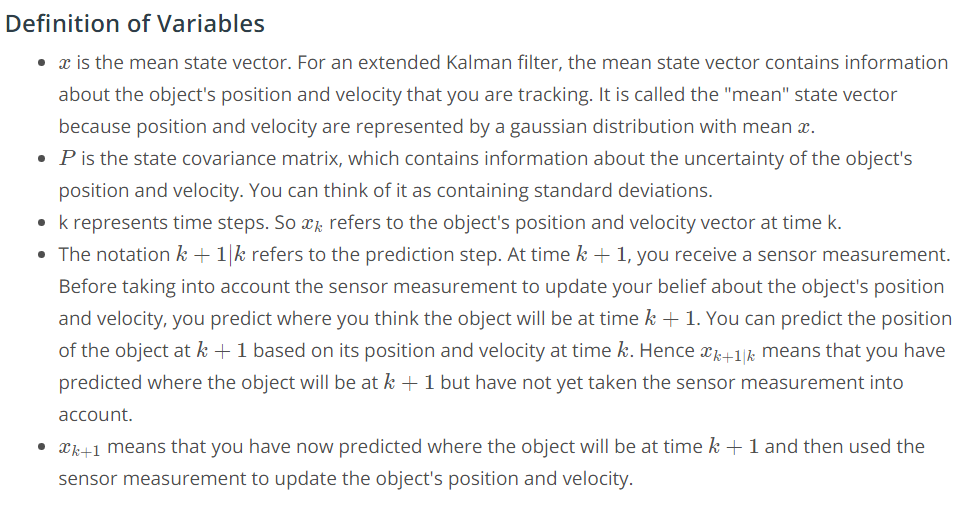
One way to handle both sensors is to keep the same process but to have each sensor have its own predict and measurement scheme.

This means the belief about the pedestrian’s position and velocity is updated asynchronously each time the measurement is received regardless of the source sensor.

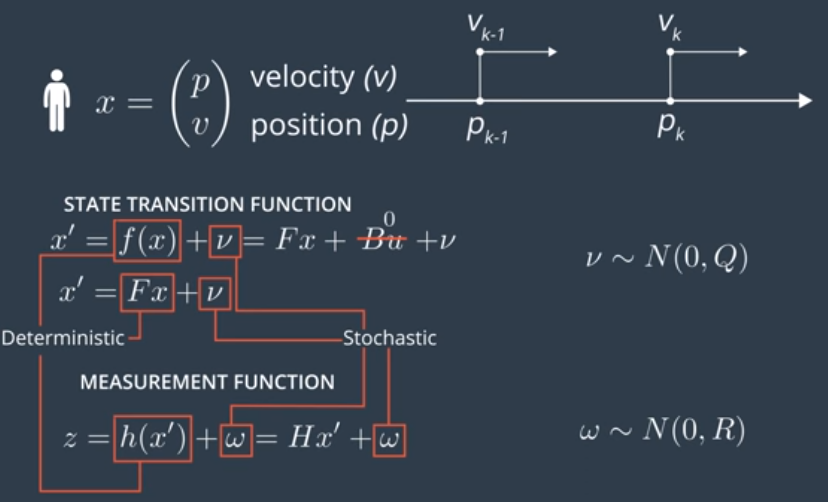
Although the state prediction method is the same for both sensors, the measurement update is different for LIDAR and RADAR. LIDAR measurements are in Cartesian co-ordinates and RADAR measurements are in Polar Co-ordinates.

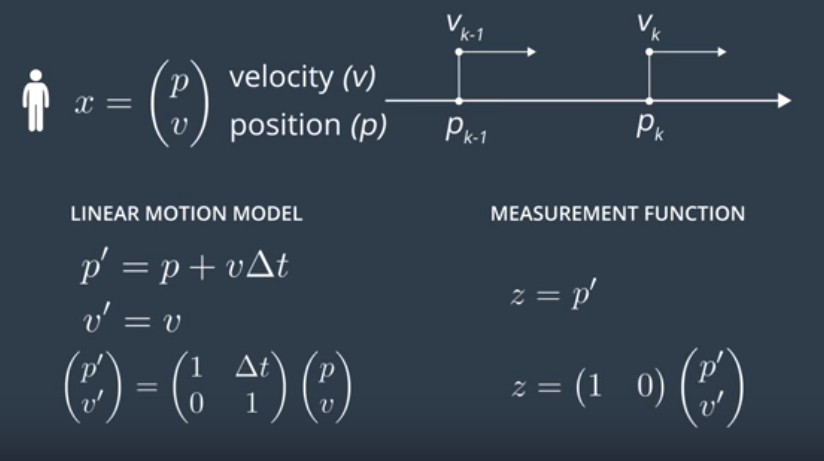


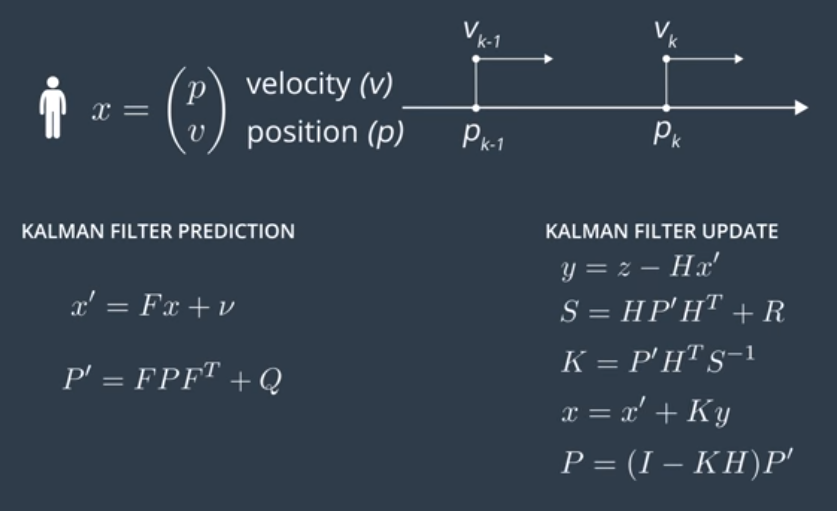


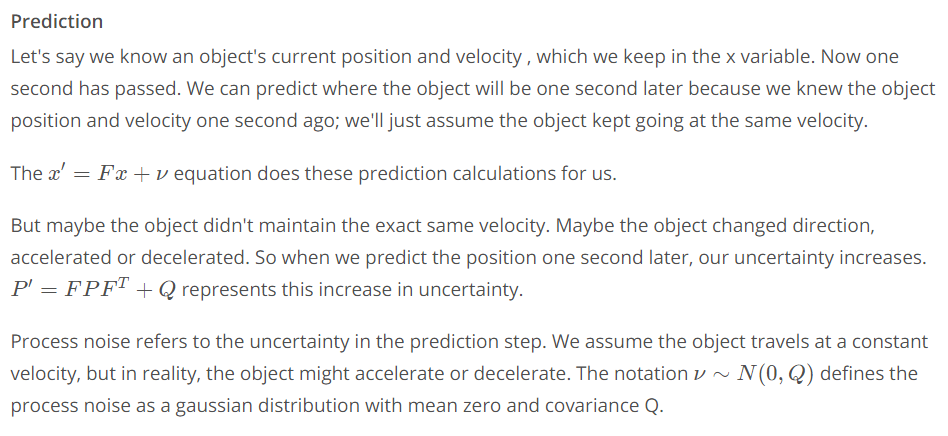


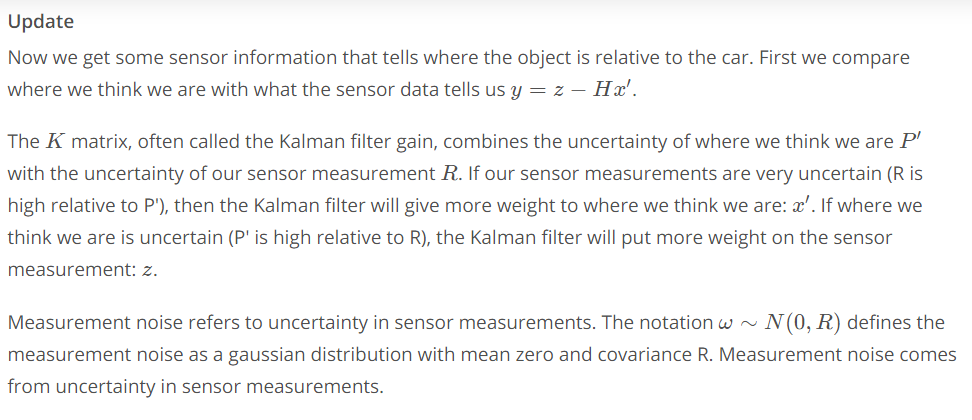
If both LIDAR and RADAR data come at the same time (K + 3), we should Predict the state to k+3 then use either one of the sensor to update. Afterwards we should predict the state K+3 again and update with the other sensors measurement. (Since both predictions for K+3 are the same, it can be a Prediction, Update, Update).

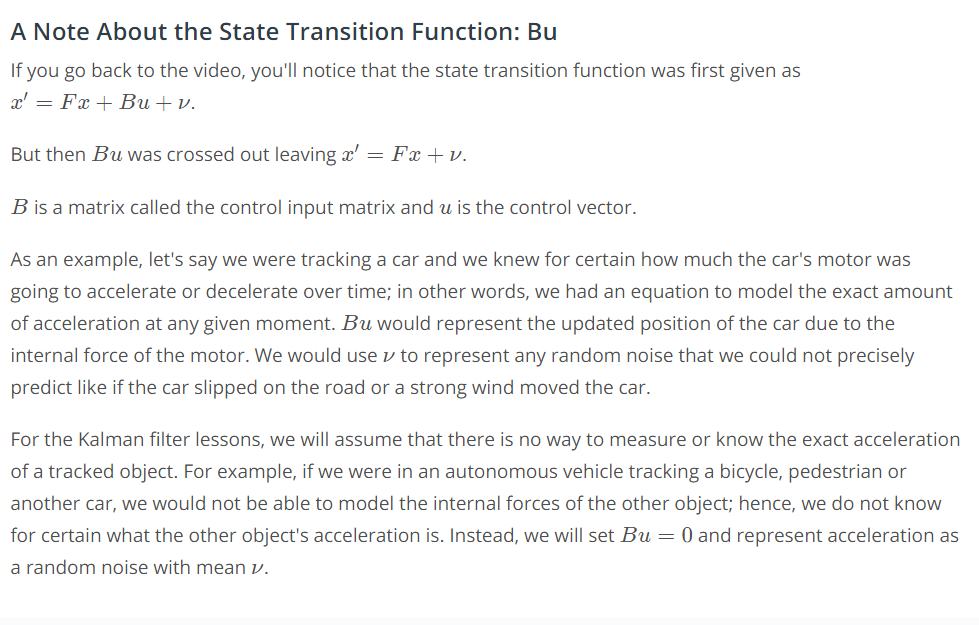












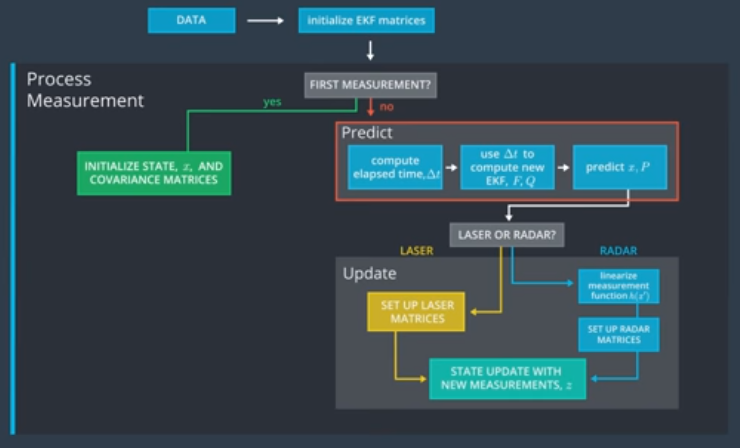
Process Noise, u, can be set to 0 because of it is random process that has a mean of 0. The process noises covariance is added to the Covariance prediction.

**Noise**  
There are 2 types of noise we consider, Process Noise from the prediction step, and Measurement Noise from the measurement step.

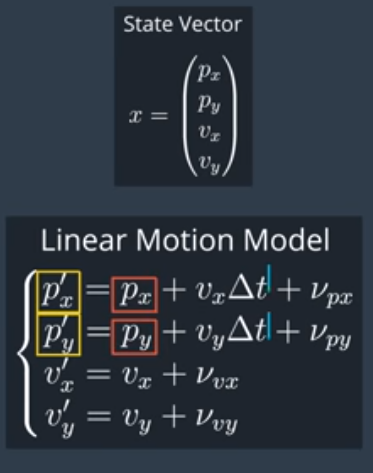
**Measurement Noise**: Uncertainty that is associated with the measurement accuracy of the sensor of an object in a state.

**Process Noise**: Uncertainty that comes from predicting the future measurement of an object. Ex: Uncertainty that a pedestrian will accelerate when we use the model of predicting constant velocity. (acceleration is any change in velocity like direction)

**2 Dimensional Equation**

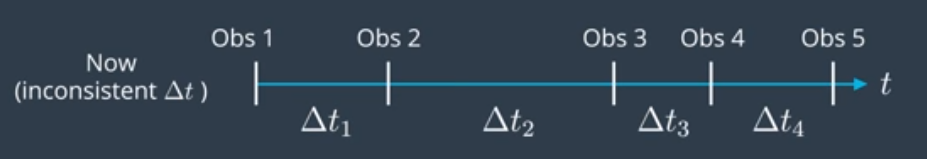


**Constant Velocity Model**



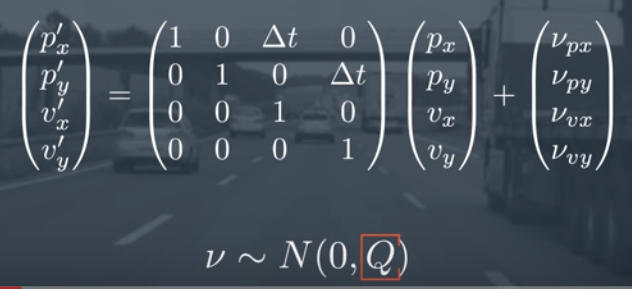


The time between measurements is now different every time. The delta T, is not constant. The time between 2 observations is not constant.



Now we need the Process Covariance matrix (Q) to calculate the stochastic part of the state transition function.

**How Acceleration is Expressed in the equation**

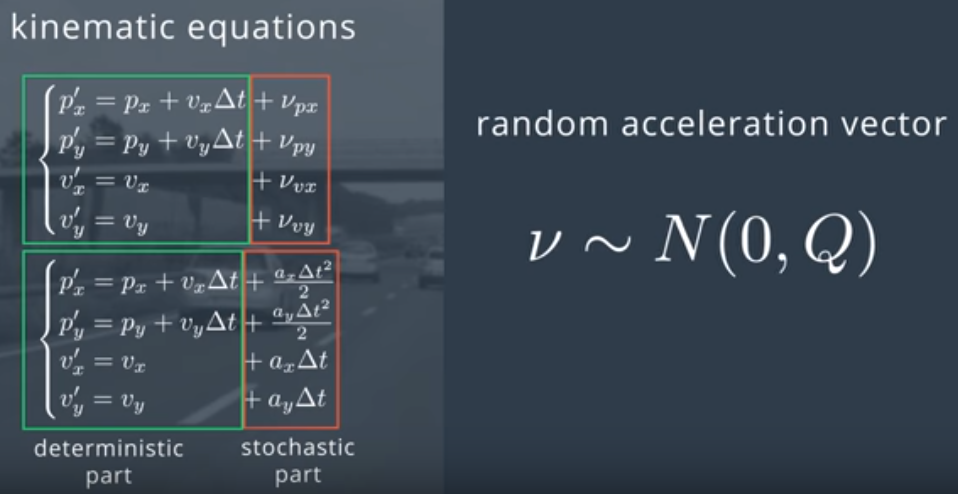


The State transition equation describes the prediction of the next state. Acceleration is represented in these kinematic equations.

If we have 2 consecutive observations of the same pedestrians position with initial and final velocities.

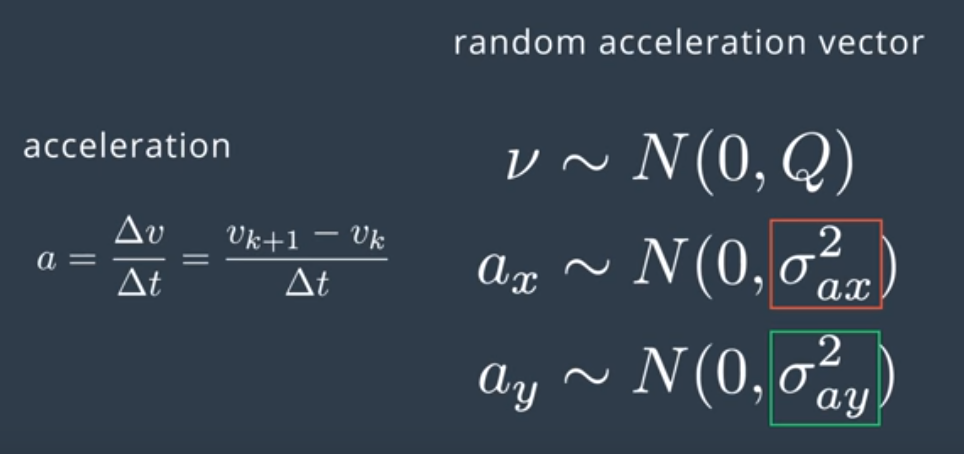
From the kinematic equations we can derive the current position and speed of the pedestrian as a function of previous state variables, including the change in velocity (acceleration).

The acceleration is then included as the random noise v vector, which makes up the stochastic part of the state transition function. The model of constant velocity is the deterministic part of the equation.

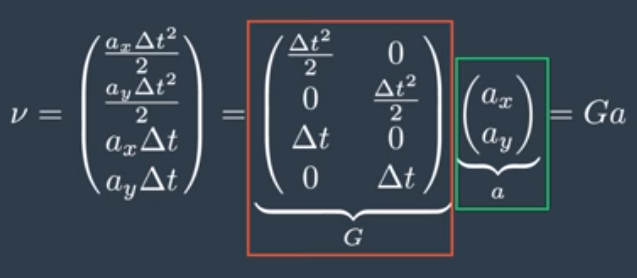


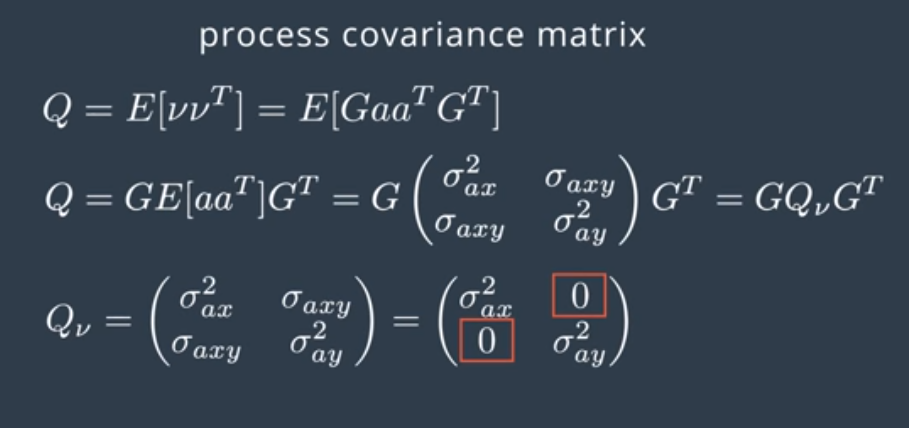
The noise is stochastic with a mean of 0 and a covariance matrix of Q.

Each Kalman Filter Step Calculates delta T.

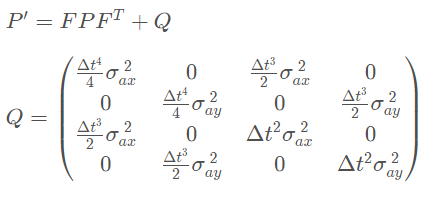


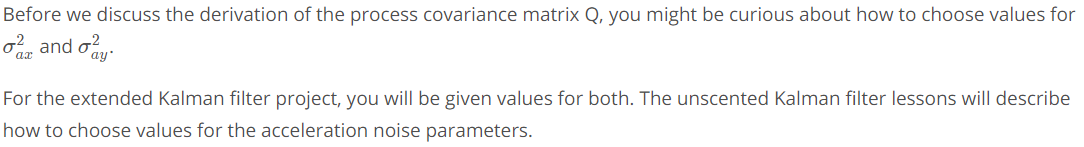
The noise can be decomposed into 2 components (4x2 matrix and a 2x1 matrix with the random components)





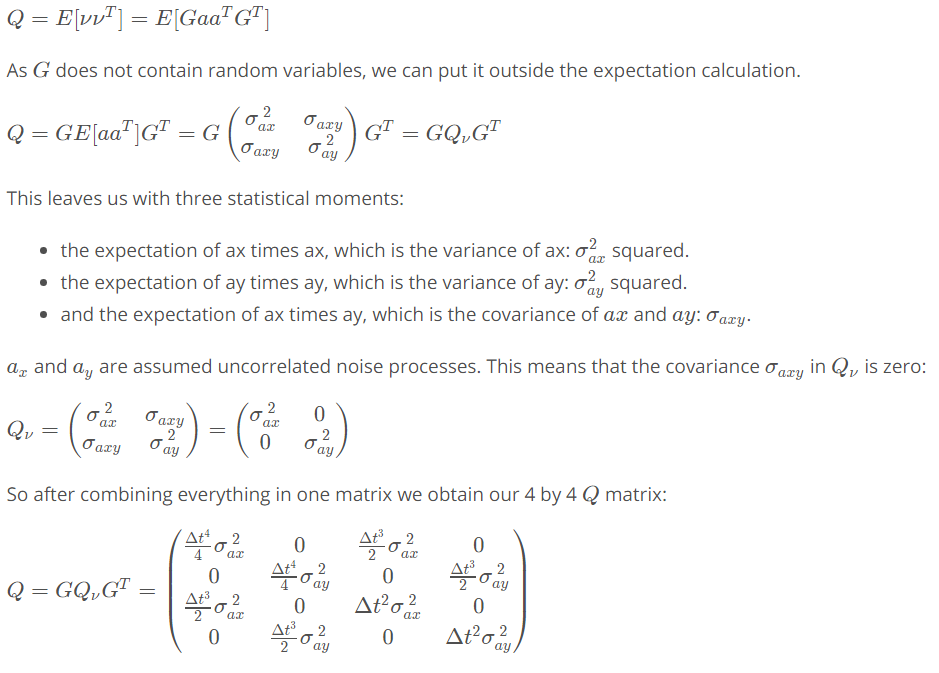
**Calculating the Process Covariance Matrix (Q)**





Because the state vector only tracks position and velocity, acceleration is modeled as a random noise. The calculation for Q includes the Delta T because the longer time goes on, the more uncertain we become about our position and velocity.

The matrix value Q, is defined as the Expected Value of the noise vector v times the noise vector v­T.

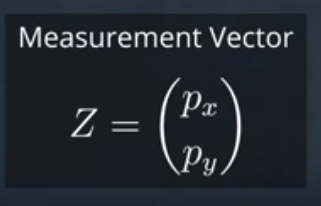


**Laser Measurements**

Now that we have a motion model that accounts for noise, we can move on to the next step of considering designing the measurement model for the Laser sensor. Gaining the measurement model for the sensor involves finding the **Measurement vector** z, the **Measurement Matrix** H, and the **covariance matrix** R.

The Laser sensor outputs a point cloud but for simplicity, in the project we will assume that the point cloud data has already been analyzed with object detection, and we have the measurements of the objects position (px,py).

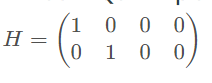
That will make our measurement vector:



Because we also know the state vector, We can calculate the find the Measurement Matrix H.

|  |  |
| --- | --- |
|  |  |

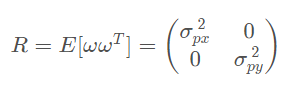
H is the projection of a 4D state to a 2D observation state. It projects the belief about the object’s current state into the measurement space of the sensor. (For the LIDAR sensor it we are saying that we are discarding velocity information since the LIDAR only measures position). The prime means that the state is after the prediction step.

This means that H is : 

**Covariance Matrix R**

Now the covariance matrix R, represents the uncertainty in our sensor measurements. R is a square matrix that is the same length as the number of measurement parameters.

That means that R is a 2x2 matrix in this case due to the measurement vector being 2x1.



Generally parameters for random noise measurement matrix will be provided by the sensor manufacturer. The 0’s in the matrix mean the noise is uncorrelated between x and y directions.

**Radar Measurements**

How will radar data improve predictions?

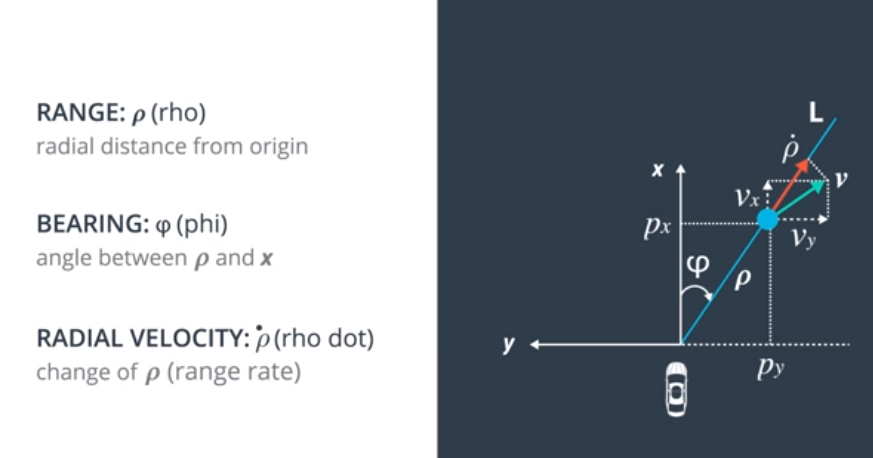
Although LIDAR data gains position information with high accuracy, we can’t directly observe any information on the objects velocity.

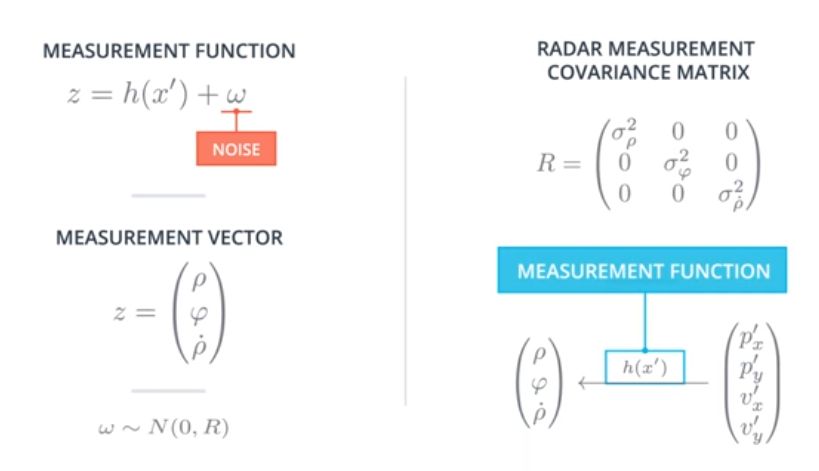
RADAR can directly measure the velocity (Radial Velocity) of an object using the Doppler Effect.

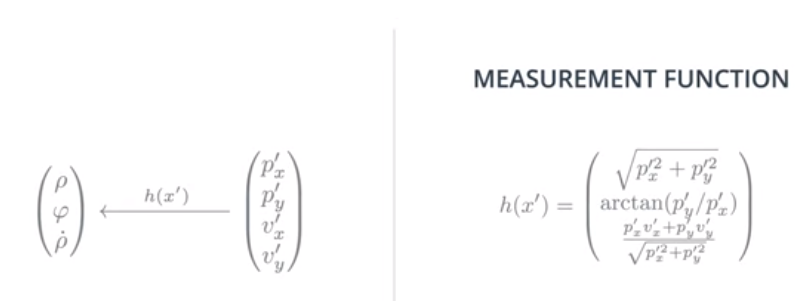
Radial velocity is the component of velocity moving away or towards the sensor.

Combining both sensors gains the strengths of both sensors.

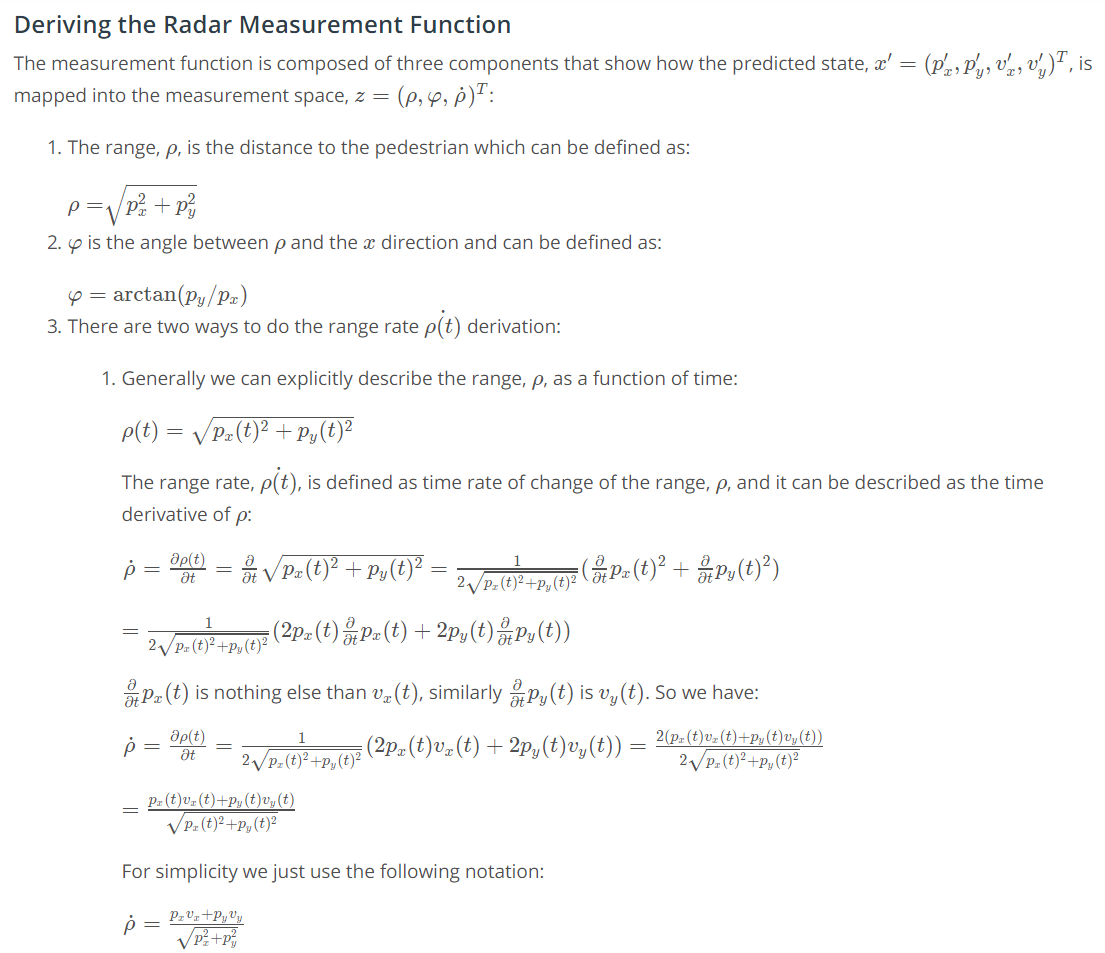
The state transition function is the exact same for both sensors, they both have the same linear motion model and process noise. However the RADAR sees the world in polar co-ordinates so some changes need to be made to the measurement step.

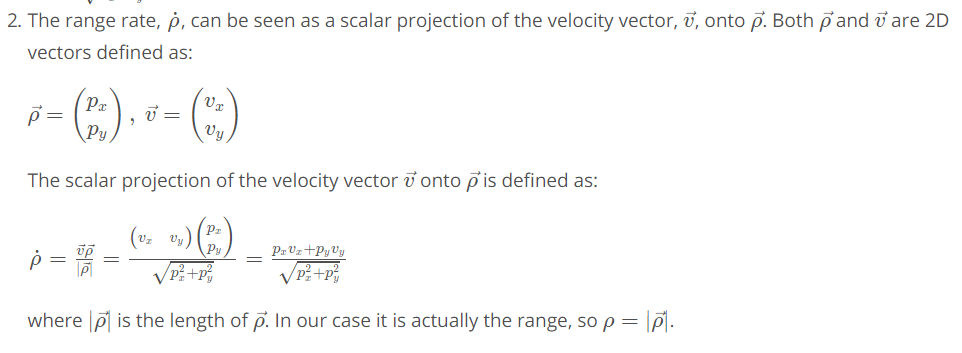






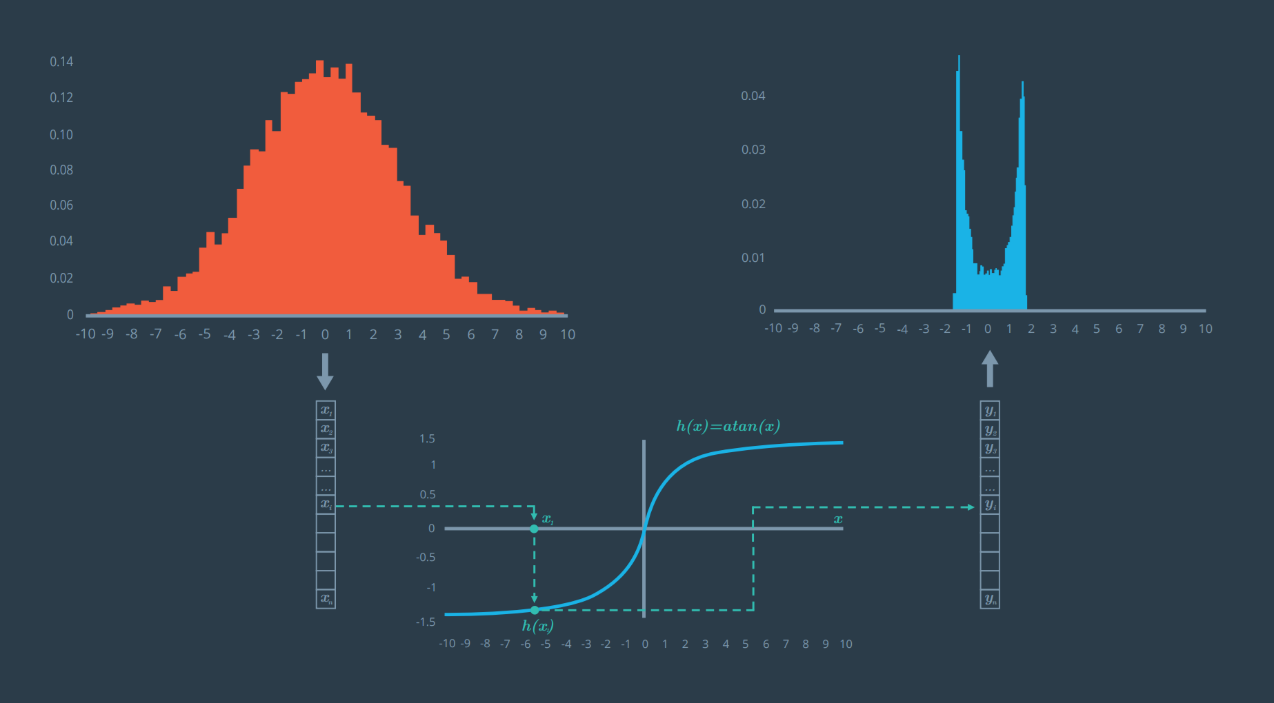
Looking at the measurement function that has been derived, we can see that it is a **non-linear function**.





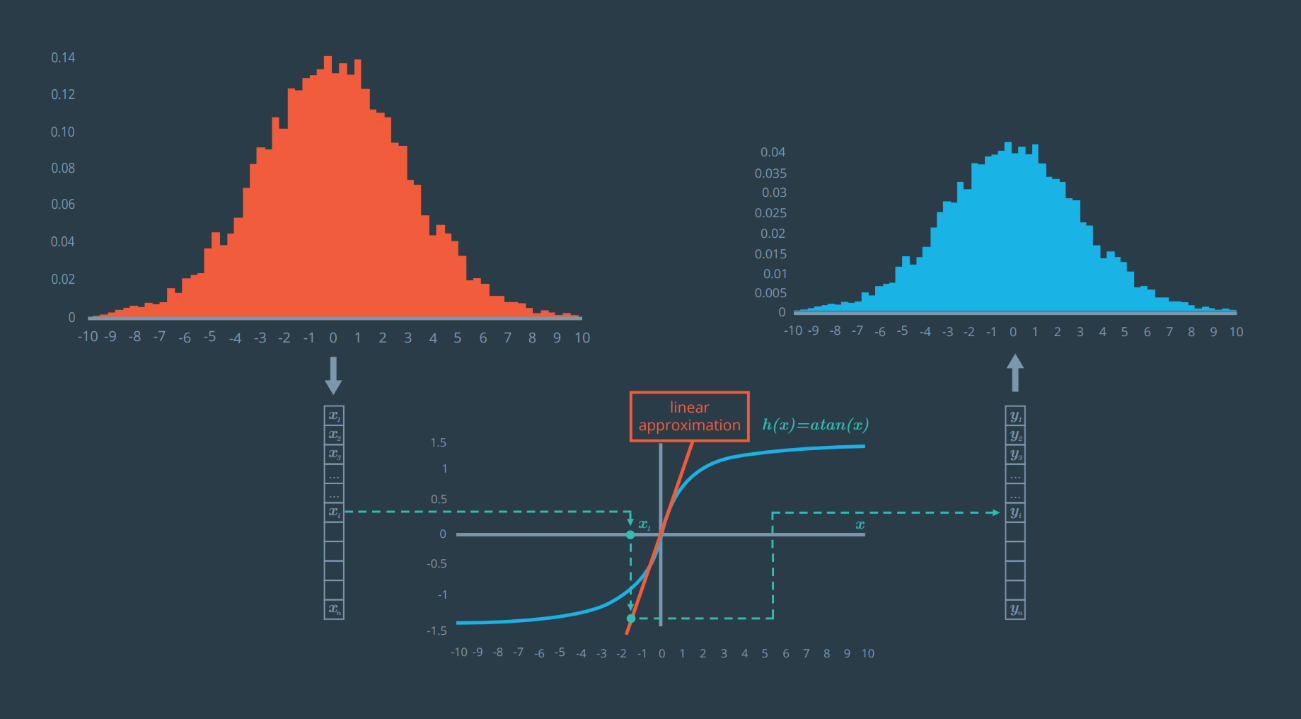
H(x) is non-linear which will not work with our Kalman Filter Equations we derived so far.

This is because the state representation after the transform of H(x) is applied is not a Gaussian anymore due to the functions non-linearities.



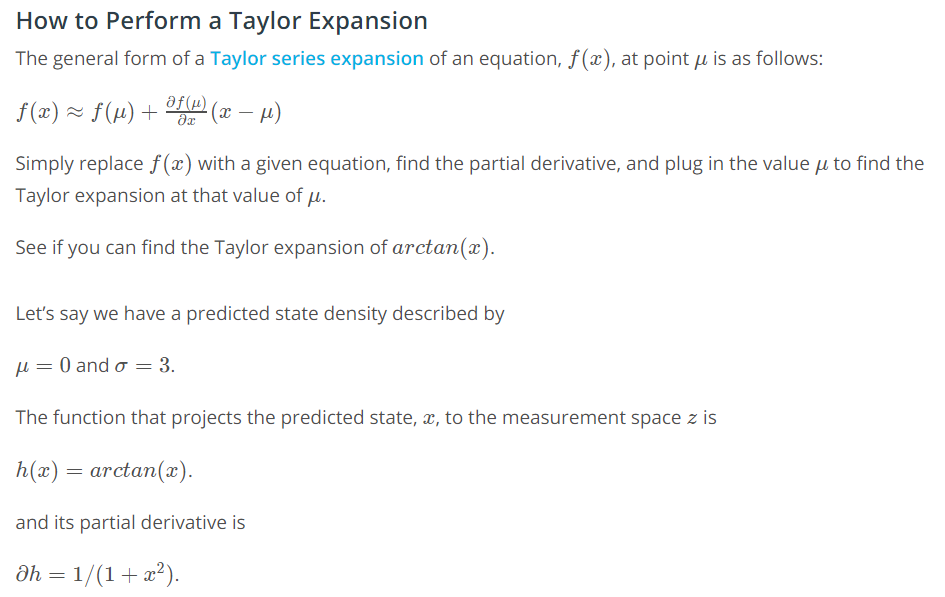
**Linearization**

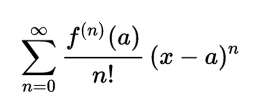
Linearizing a function means to take a approximation in a region that the function is a line. This is done by using the Taylor series expansion about a small region to find the functions linear approximation.



A Taylor series expansion is important in that it means that any function can be represented as an infinite sum of terms that are calculated from the values of the functions derivatives at a single point. (Special case where the point is 0 is called a Maclaurin series).

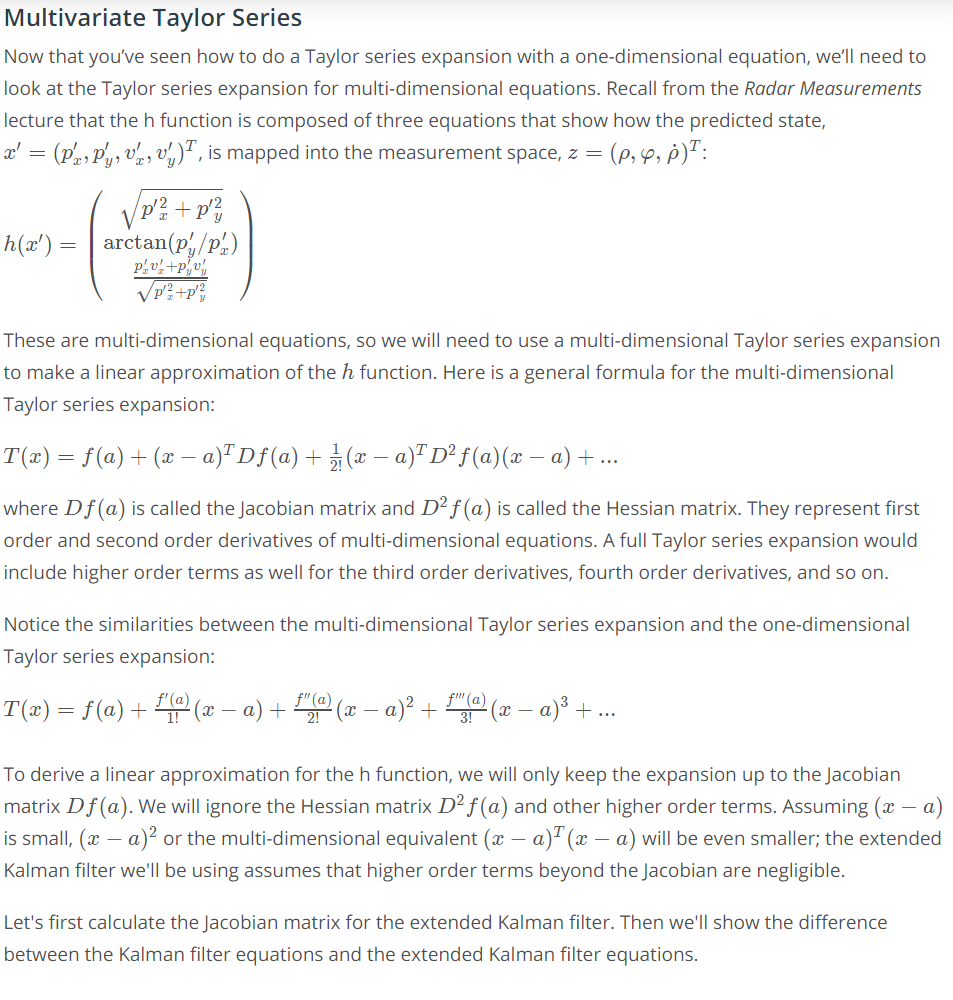
A function can be approximated by using a finite number of terms in the Taylor series, where using only the terms of degree 0 and 1 would result in an equation for a line. Approximating non-linear functions as lines is useful in linearizing a function about the single point we are creating the series around.





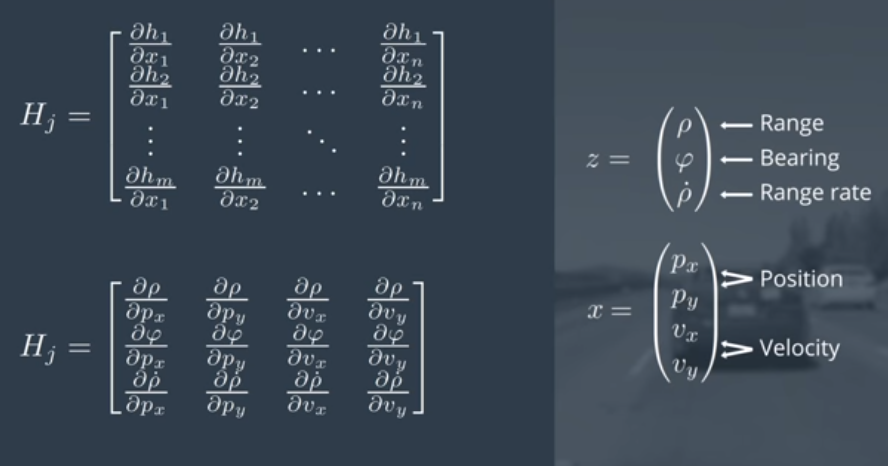
Now that we see Taylor Series Expansion in one Dimension, we need to know Multivariate Taylor Series Expansion to linearize the radars H(x).

Note: Linearization is only useful in a small region that is similar to a line. If the region is not similar, it is a bad approximation. This means that we are working with the assumption that (x-a) is small, but (x-a)2 is even smaller, and that all other higher terms are small enough to be negligible.

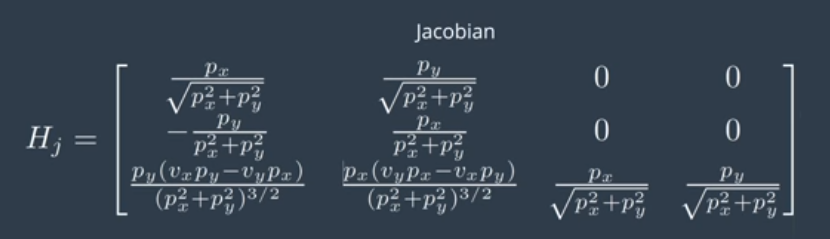


**Jacobian Matrix**

A matrix containing all of the partial derivatives.

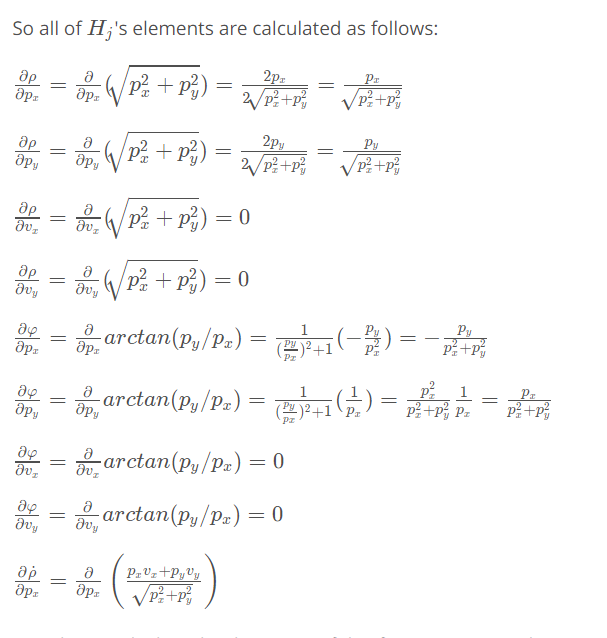


**Result:**

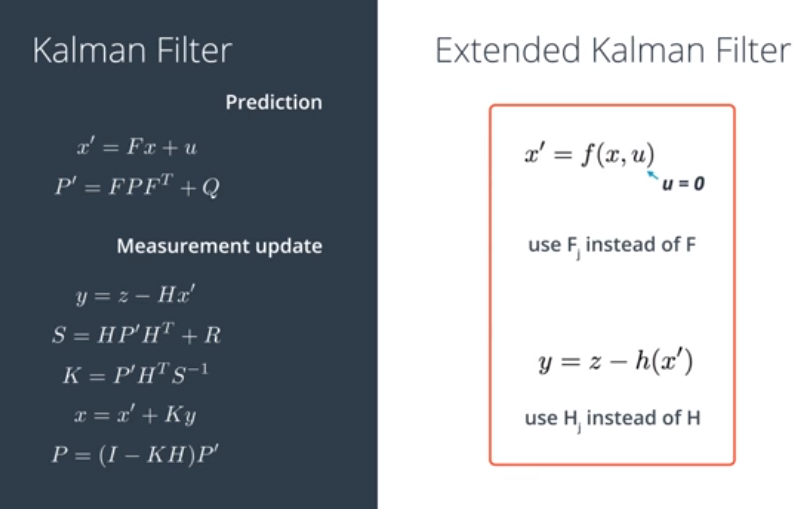


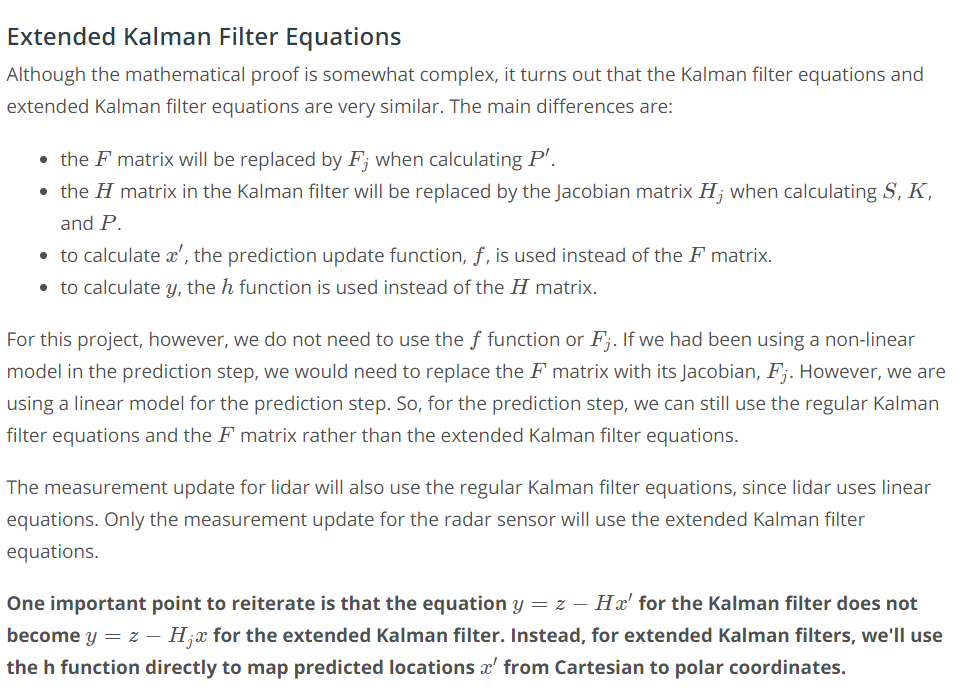
In our implementation, we will create a function that can take the current predicted state, and compute its Jacobian.

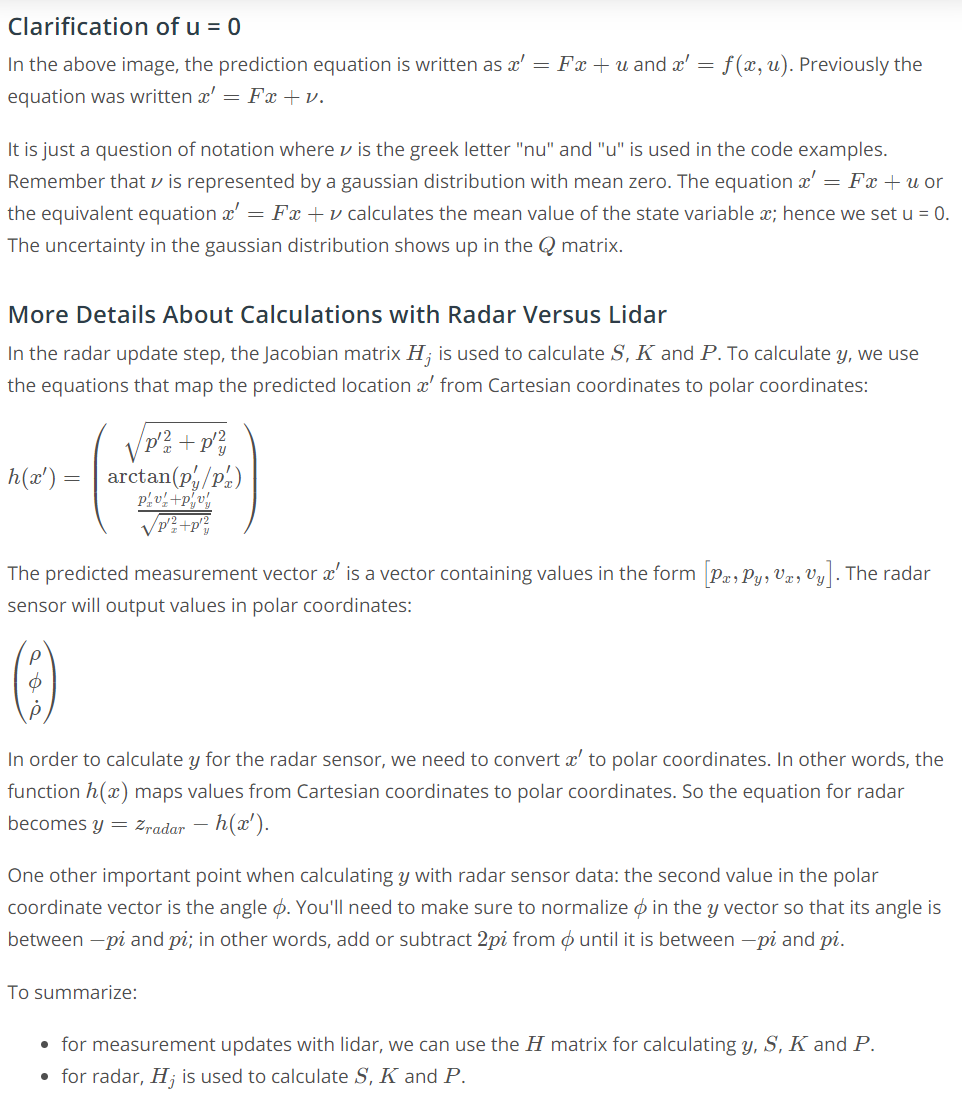
**Calculation:**



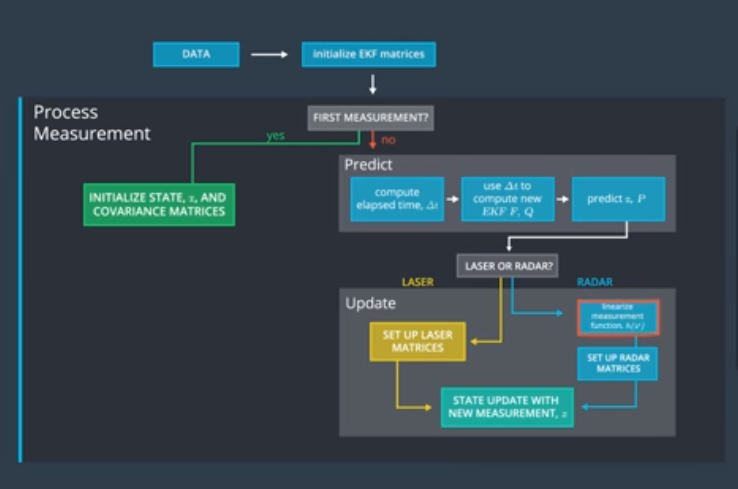
**Extended Kalman Filter Generalization**







Note: If an extended Kalman filter were used on linear functions, the result would be the exact same as a standard Kalman Filter. This is because the Jacobians are the same.



**Evaluating the Performance of a Kalman Filter**

How far the estimated results are from the true results can be tracked using many performance metrics. The most common one used for Kalman Filters is Root Mean Squared Error.

