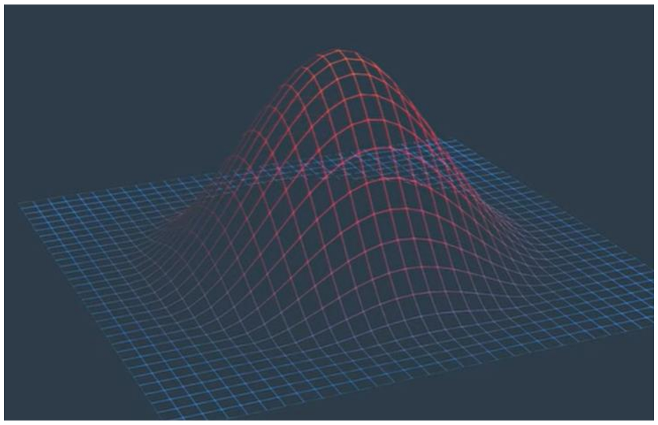
# MATRICES AND TRANSFORMATION OF STATE

Kalman Filter Land

**High dimensional Gaussian**



Explain how this works, I have to talk about high-dimensional Gaussians.

It is called multivariate Gaussians

1- The Mean is now a vector with one element for each of the dimensions
2- The Variance square is replaced by what`s called a Covariance and it`s matrix With D rows and D columns if the dimensionality of the estimated is D  and the formula is something you have to get used to but you never get seethe again.

$$\begin{pmatrix} \mu_0 \\ .. \\ .. \\ .. \\ \mu_D \end{pmatrix} \qquad \Sigma \begin{pmatrix} \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots \end{pmatrix}$$

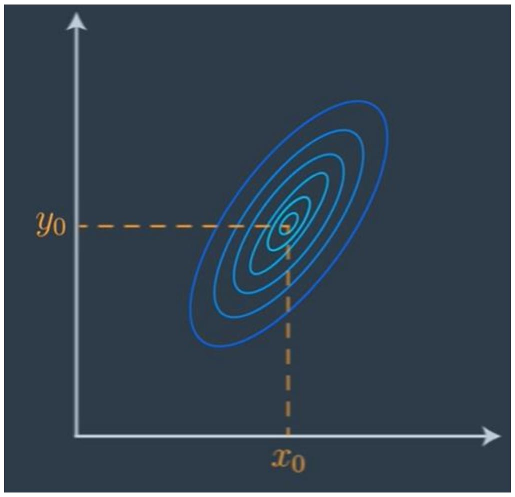
$$2\pi^{-\frac{D}{2}} \ |\Sigma|^{\frac{1}{2}} \exp -\frac{1}{2}(x-\mu)^T \ \Sigma^{-1}(x-\mu)$$

Let me explain it to you more intuitively.

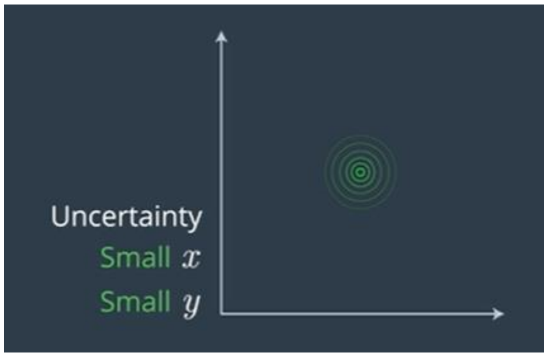
A two-dimensional Gaussian is defined over that space and it is possible to draw **Contour line** of the gaussian might look like this

## MATRICES AND TRANSFORMATION OF STATE

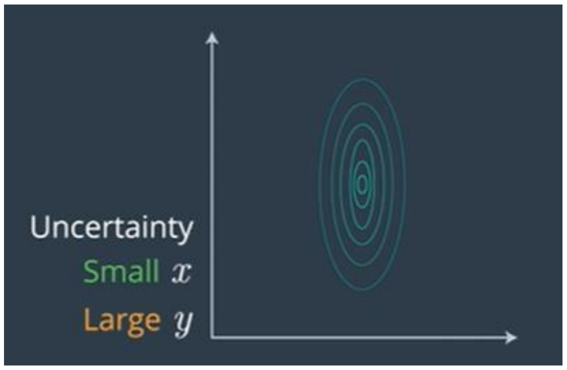
The **mean of gaussian is this x_0, y_0** pair and the **Covariance now defines the spread** of the Gaussian as indicated by these contour lines.



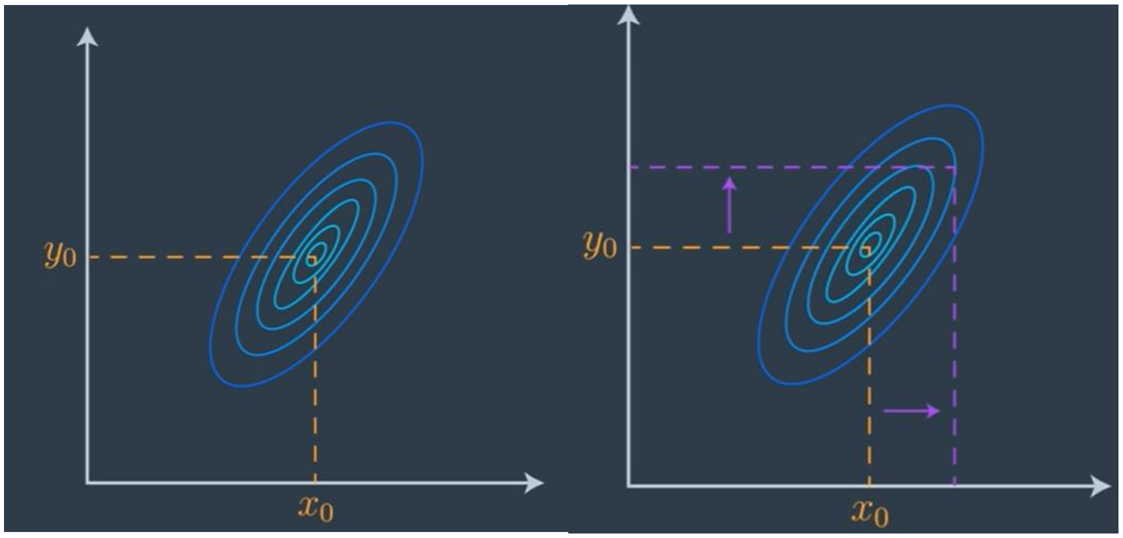
A **Gaussian with small amounts of uncertainty** might look like



It might be possible to have a fairly small uncertainty in one dimensional but a hug uncertainty in the other.



The uncertainty in the x dimension is small and the y dimension is large and when the Gaussian is titled as shown over here, then uncertainty of x and y is correlated which mean if I get information about x that actually sits over here



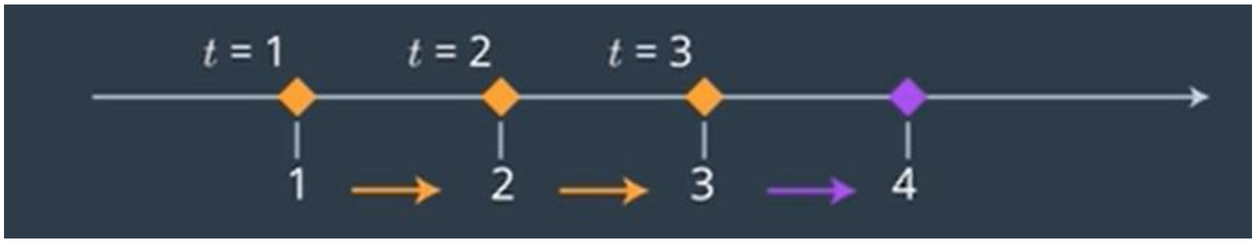
They would make  me believe that y probably sits somewhere over here, that`s called correction.

I can explain to you the entire effect of estimating velocity and using it in filtering using gaussians like this and it becomes really simple

The problem i`m going to choose is a one-dimensional motion explain example

Let`s assume t =1 we see option over here ……………………. , then I would assume that t equal = 4 the objects sits over here

And the reason why you would assume this is even though we just see these different discrete location's you can infer from where there`s actually velocity that drives the object to right side to the point over here, how does the Kalman filter addresses ?

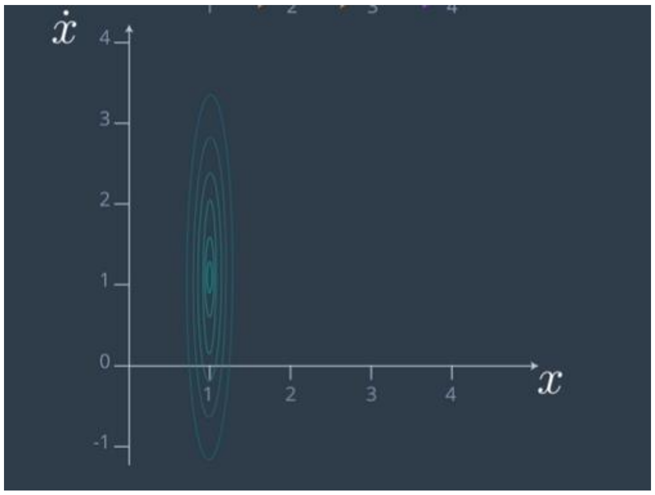


**<u>Kalman Filter prediction</u>**

Address problem in Kalman filter, we are going to build a two -dimensional estimate.

1- One for the location and one for the velocity denoted X  but it can it also be 0. It can be negative or it can be positive

**<u>if initially, I know my location but not me velocity, then I represented Gaussian as elongated around the correct location.</u>**



But really , really broad in the space of velocities.

Now let`s look at the predictions step.

In the prediction step, I don`t know my velocity. So I can`t possibly predict for location I want I saw.

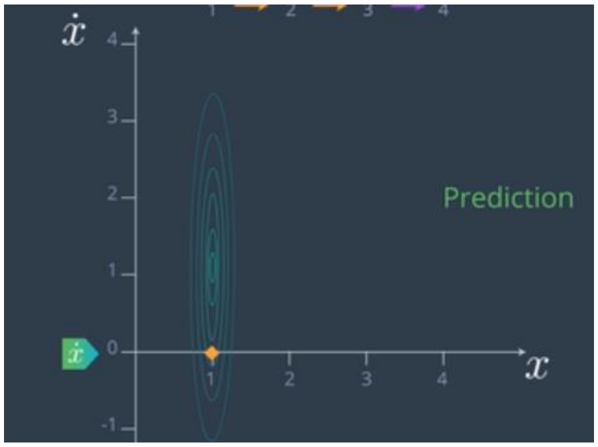
But miraculously they have some interesting correction.

1- Pick a point on the distribution

Let me assume the velocity is 0. Of Course in practice I don`t know the velocity but let me assume for a moment the velocity is 0.

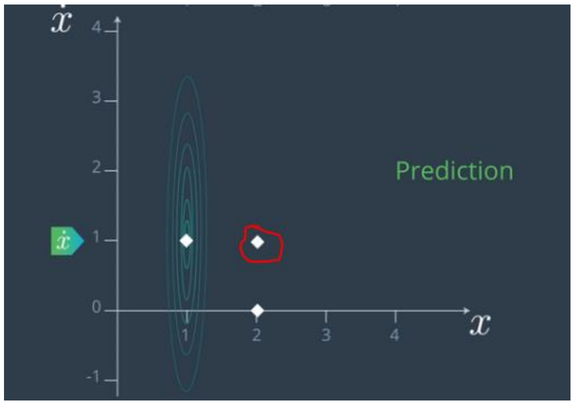
Where would my posterior be after prediction.

We know we start location one so the velocity is 0 , so my location be here.



Now let`s change my belief of the velocity and pick  different one.

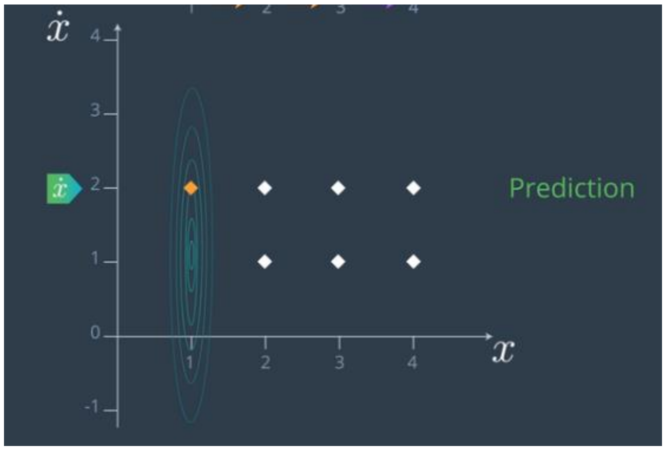
Let`s ay the velocity is one where would my prediction be one timestep later,  starting at location one and velocity 1



Another prediction

Let`s me consider the velocity is 2which means this is our starting point.

And let me ask you where we would expect among those choices to be most plausible prediction to be?



This model assumes that in the absence of more knowledge diversity shouldn`t really change.

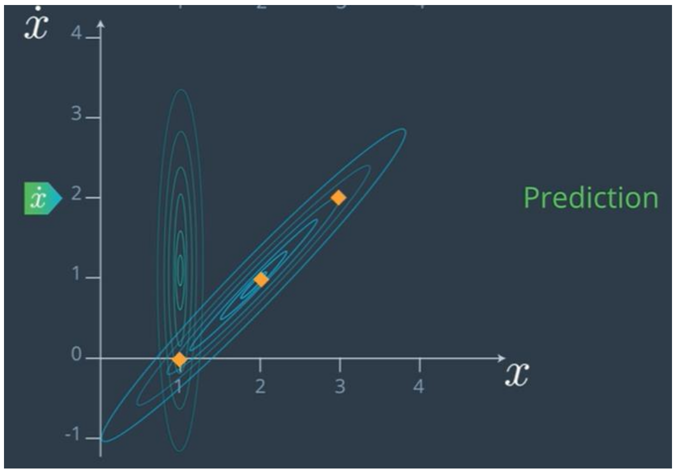
## Correction

The equation for calculating the next position x should read as an addition: x' = x + \dot{x} \Delta t$x' = x + \dot{x} \Delta t$

### *Kalman Filter*

You will find that all these possibilities on the gaussian over here link to a Gaussian that`s just like this

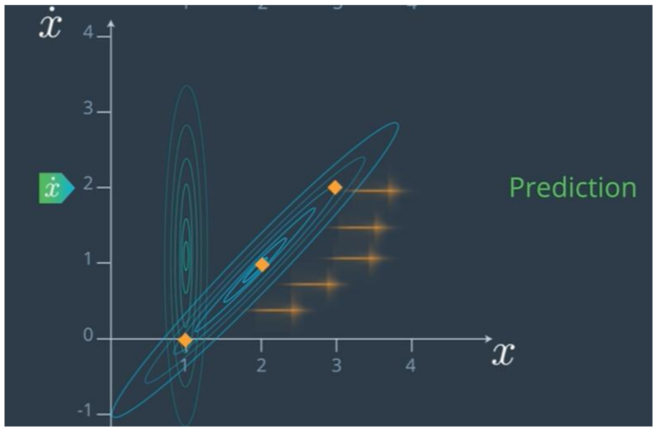
## MATRICES AND TRANSFORMATION OF STATE



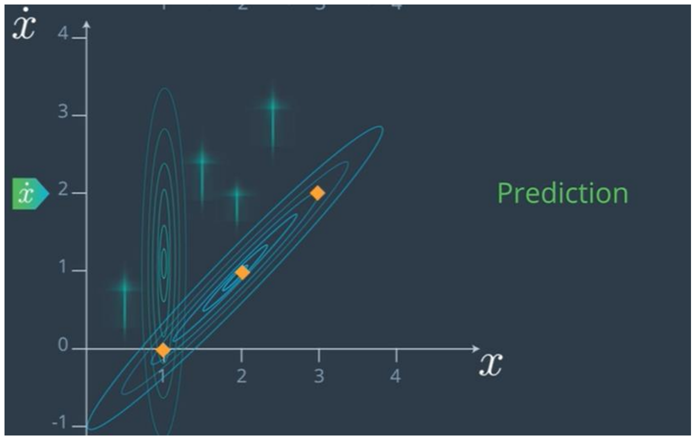
This is really interesting two-dimensional Gaussian

Clearly if I it was to project this Gaussian uncertainty into the space of possible location's, I can`t predit a thing .

It`s impossible to predict where the object is  , and the reason is I don`t know the velocity



Also, clearly, if I project this gaussian into the space of x_dot , then it`s impossible to say what to say what the velocity is.

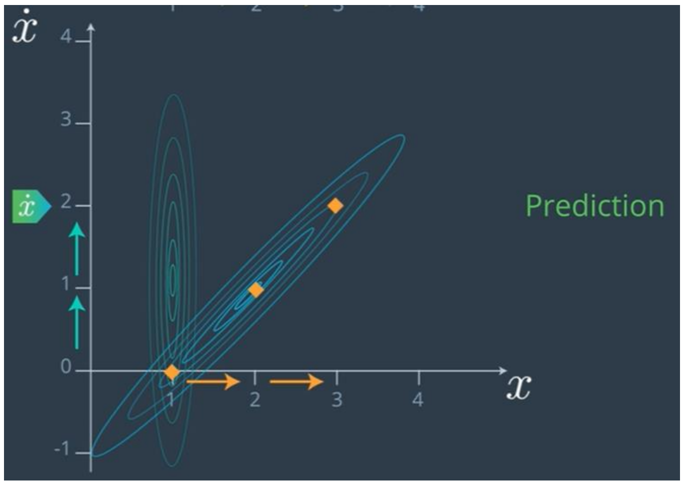


## MATRICES AND TRANSFORMATION OF STATE

A single observation or single prediction is insufficient to make that observation.

However, what we know is our location is correlated to the velocity.

**The faster I move the further on the right is the location, and this Gaussian express this.**



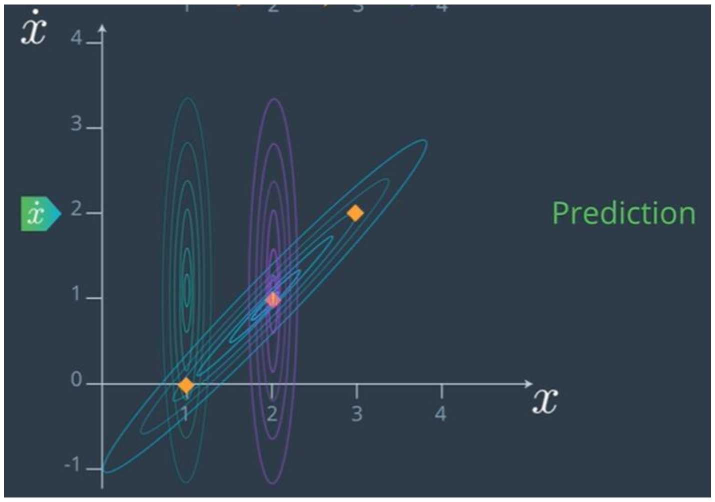
**<u>If I for example figured out that my velocity was 2 and then able under this Gaussian to really nail that my location is 3</u>**

**This is really remarkable you still haven`t figured where you are , we haven`t figured out how fast we are moving, but we have learned so much about the relation of these two things with this titled Gaussian.**

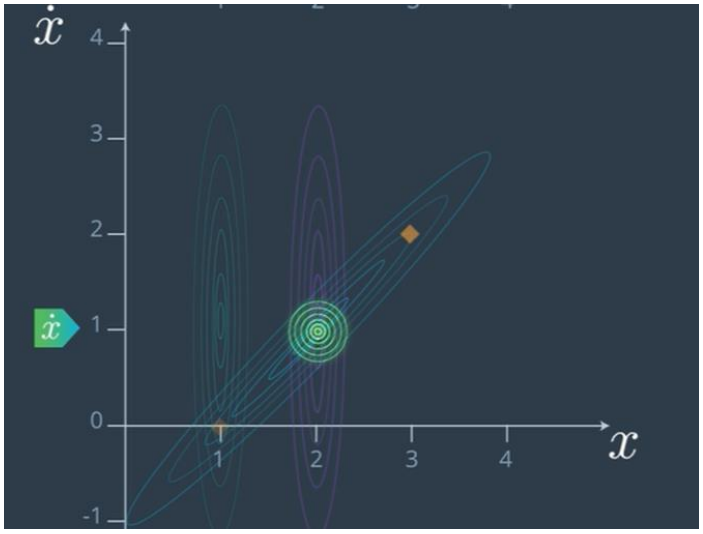
**To understand how powerful this is, let`s now fall in the second observation at time t = 2.**

**This observation tells us nothing about the velocity and only something about the location.**

**So, if I were to draw this as Gaussian, it`s a Gaussian just like this, which is sending about location but not about the velocity.**



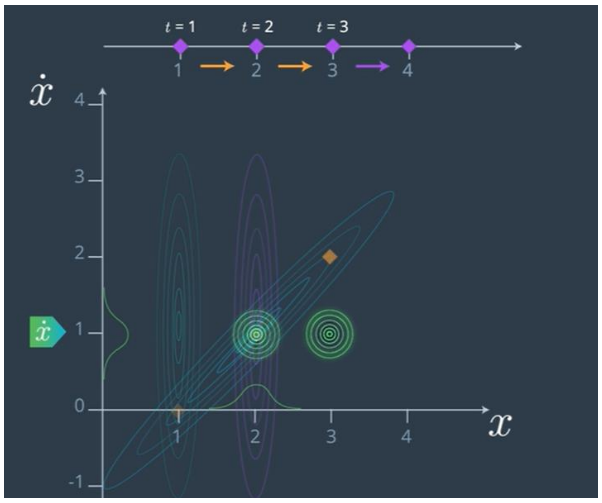
**But if I now multiply my prior from the perdition step with the measurement probability, then miraculously, I get the gaussian that sits right over here.**



**And then Gaussian now has a really good estimate what my velocity is and a really good estimate where I am.**

**If I take this Gaussian and predict one step forward, and I find myself right over here**

**This is exactly the effect we have over here.**

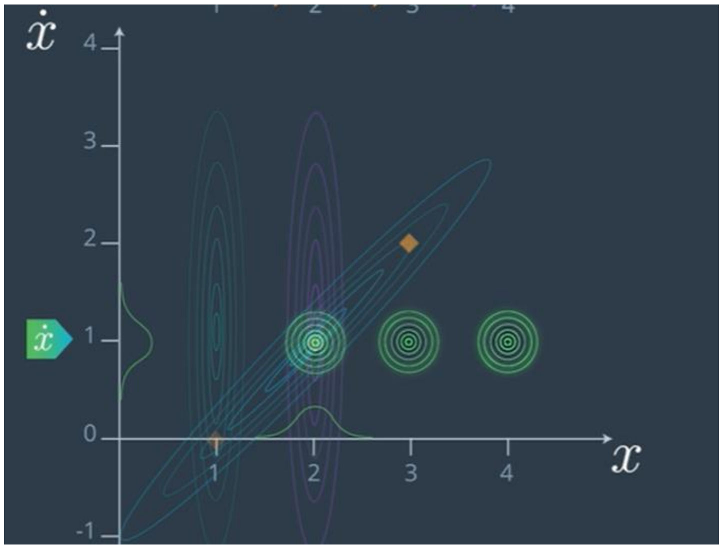


**As I opted this, I get a Gaussian like this,**

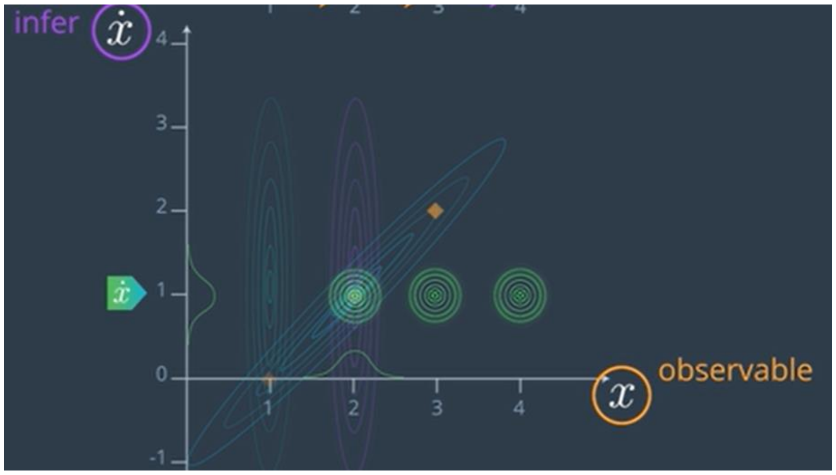
**I predict right over here. Think bout this**

## MATRICES AND TRANSFORMATION OF STATE

**This is a really deep insight in how Kalman filter work.**



**In particular, we have only been able to observe one variable, we have been able to multiple observation to infer this other variable.**



**The way we have been able to infer this is that there`s a set of physical equations which say that**

1- **My location after time step is my old location + my velocity**

$$x' = x\Delta t\dot{x}$$

**This set of equation has been able to <u>propagate constraints</u> from subsequent measurements back to this unobserved variable velocity, so we are able to estimate the velocity as well.**

**This is really key to understanding Kalman filter.**

## MATRICES AND TRANSFORMATION OF STATE

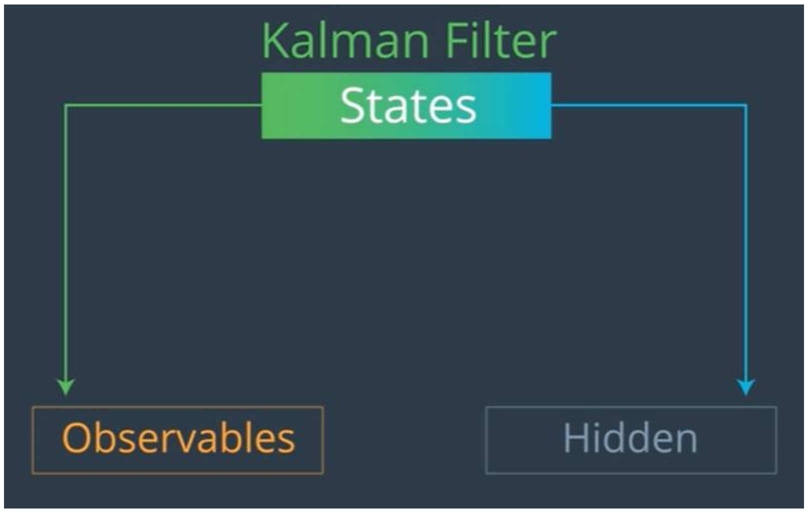
It is key to understand how google self-driving car estimate location other cars, and is able to make predictions even if it`s unable to measure velocity directly.

So, there`s a big lesson here

The variables of the a Kalman filter, they often called states because they reflect states of the physical world like where`s the other car and how fast it`s moving.

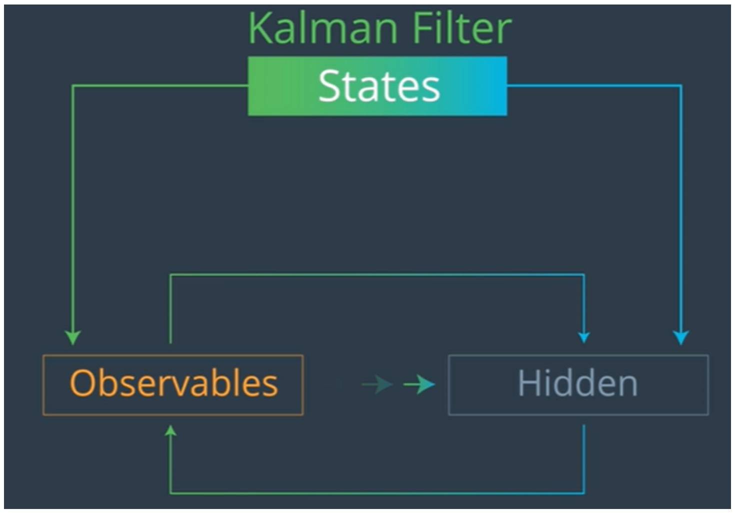
They separate into two subsets

1- the observables, like the momentary location
2- the hidden, which in our example is the velocity which I can never directly observe.



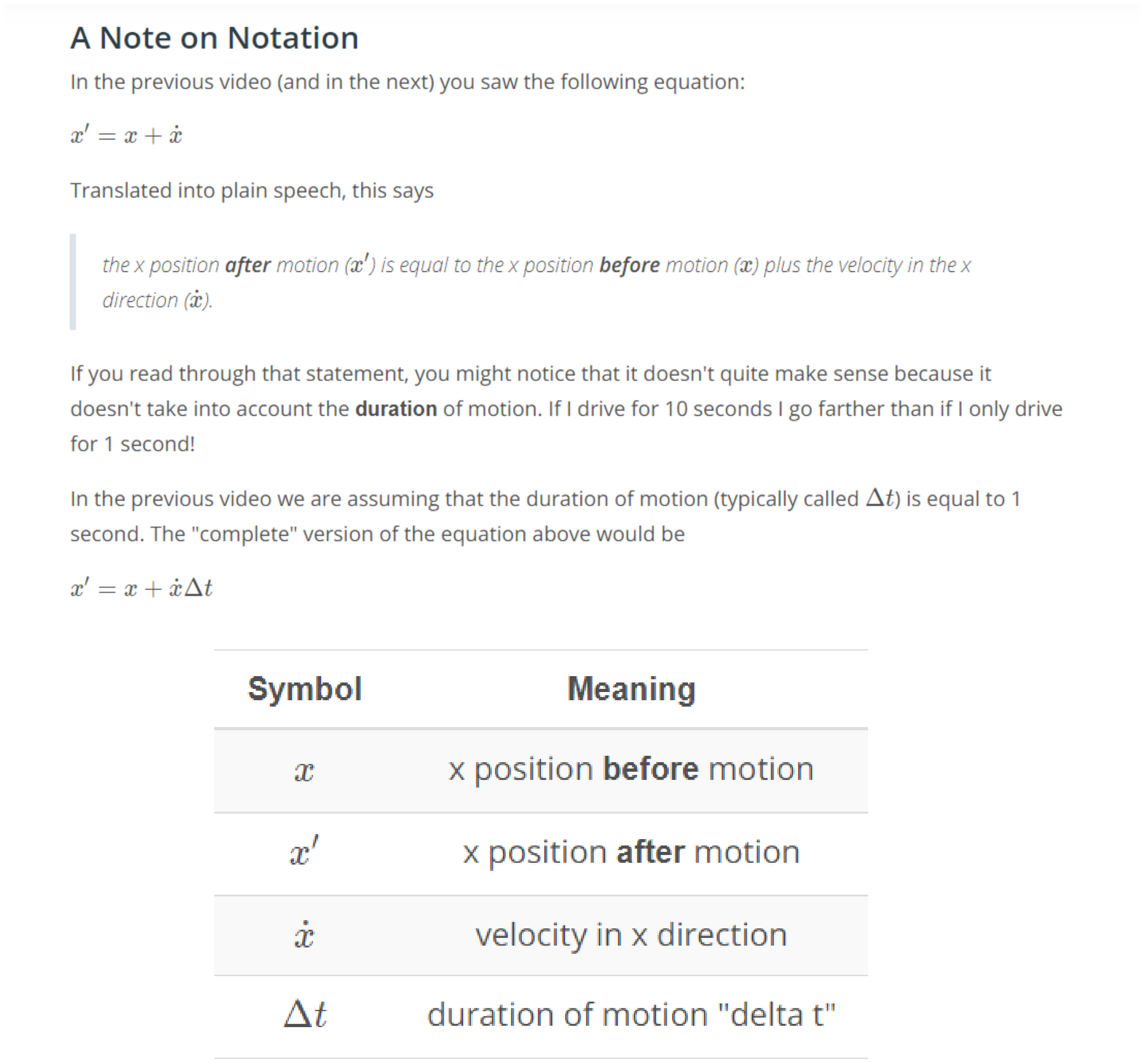
But because those two thing interact, subsequent observations of the observed variables give us information about these hidden variables.

So we can also estimate what these hidden variables are.



So from multiple observations of the places of the object,  the location we can estimate how fast it`s moving

**This is actually true for all the different but because Kalman filter happened to be very efficient to calculate, when you have a problem like this you tend to often use just the Kalman filter.**



## A Note on Notation

In the previous video (and in the next) you saw the following equation:

$x' = x + \dot{x}$

Translated into plain speech, this says

> *the x position **after** motion ($x'$) is equal to the x position **before** motion ($x$) plus the velocity in the x direction ($\dot{x}$).*

If you read through that statement, you might notice that it doesn't quite make sense because it doesn't take into account the **duration** of motion. If I drive for 10 seconds I go farther than if I only drive for 1 second!

In the previous video we are assuming that the duration of motion (typically called $\Delta t$) is equal to 1 second. The "complete" version of the equation above would be

$x' = x + \dot{x}\Delta t$

| Symbol | Meaning |
|---|---|
| $x$ | x position **before** motion |
| $x'$ | x position **after** motion |
| $\dot{x}$ | velocity in x direction |
| $\Delta t$ | duration of motion "delta t" |

## MATRICES AND TRANSFORMATION OF STATE

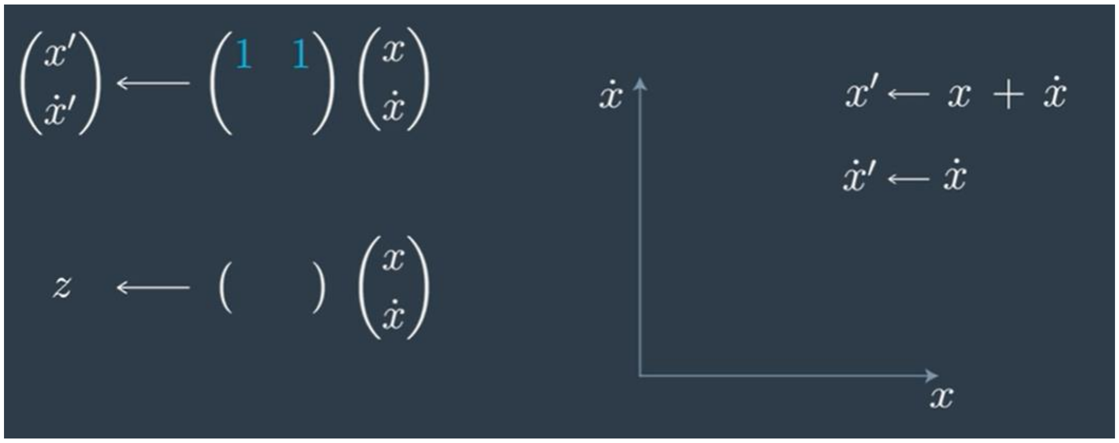
**Kalman Filter Design**

**Design Kalman filter**

1- **state translation function**
2- **Measurement function**

**So, Let`s give you those for our example of the 1D motion of an abject.**

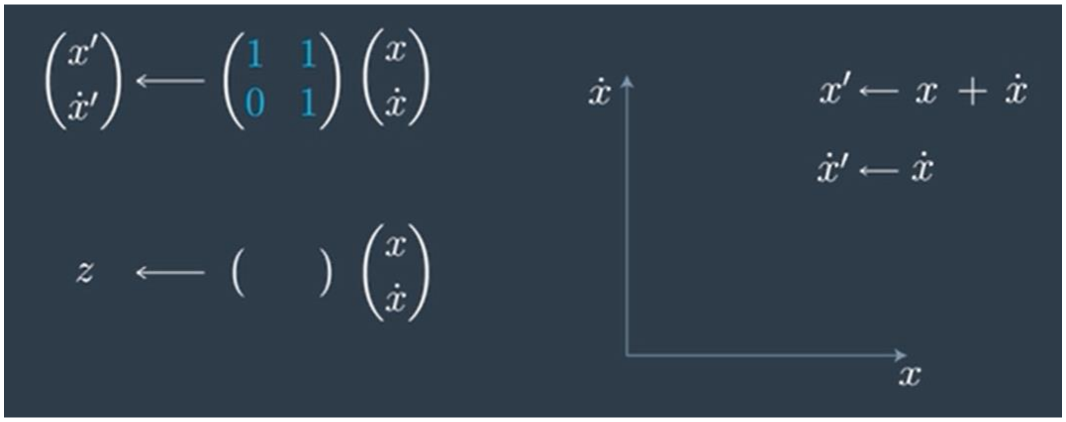
**The new location = old location + velocity to in this matrix**


$\begin{pmatrix} x' \\ \dot{x}' \end{pmatrix} \longleftarrow \begin{pmatrix} 1 & 1 \\ & \end{pmatrix} \begin{pmatrix} x \\ \dot{x} \end{pmatrix}$
$z \longleftarrow \begin{pmatrix} & \end{pmatrix} \begin{pmatrix} x \\ \dot{x} \end{pmatrix}$
$\dot{x}$
$x$
$x' \leftarrow x + \dot{x}$
$\dot{x}' \leftarrow \dot{x}$

**The new velocity should just be**

**The new velocity the old velocity because the 0 over here, and a 1 over here.**

**If we multiply this matrix with this vector, this is result**


$\begin{pmatrix} x' \\ \dot{x}' \end{pmatrix} \longleftarrow \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ \dot{x} \end{pmatrix}$
$z \longleftarrow \begin{pmatrix} & \end{pmatrix} \begin{pmatrix} x \\ \dot{x} \end{pmatrix}$
$\dot{x}$
$x$
$x' \leftarrow x + \dot{x}$
$\dot{x}' \leftarrow \dot{x}$

**For measurement, you only observe the first component of the place not the velocity, and that uses a vector or matrix like this.**

$$z \longleftarrow \begin{pmatrix} 1 & 0 \end{pmatrix} \begin{pmatrix} x \\ \dot{x} \end{pmatrix}$$

## MATRICES AND TRANSFORMATION OF STATE

**So, this matrix will be called F, and the H actual update equations for a Kalman filter involved.**

**Update**

**I give them to you, but please don`t memorize them and I won`t prove them for you,**

1- **There`s prediction step where I take my best estimate x multiplied with the state transition matrix F and I add whatever the motion that give me new x.**

**I also I have a covariance that  characters my uncertainty this update as follow, where T is transpose.**

**There`s also a measurement update  step where we use the measurement z.**

**We compare the measurement with our prediction where H is the measurement function that maps the state to measurement. (we call this the error).**

**The error is mapped into a matrix S which is obtained by projecting the system uncertainty into the measurement space using the measurement function protection + a matrix R the character is a measurement noise.**

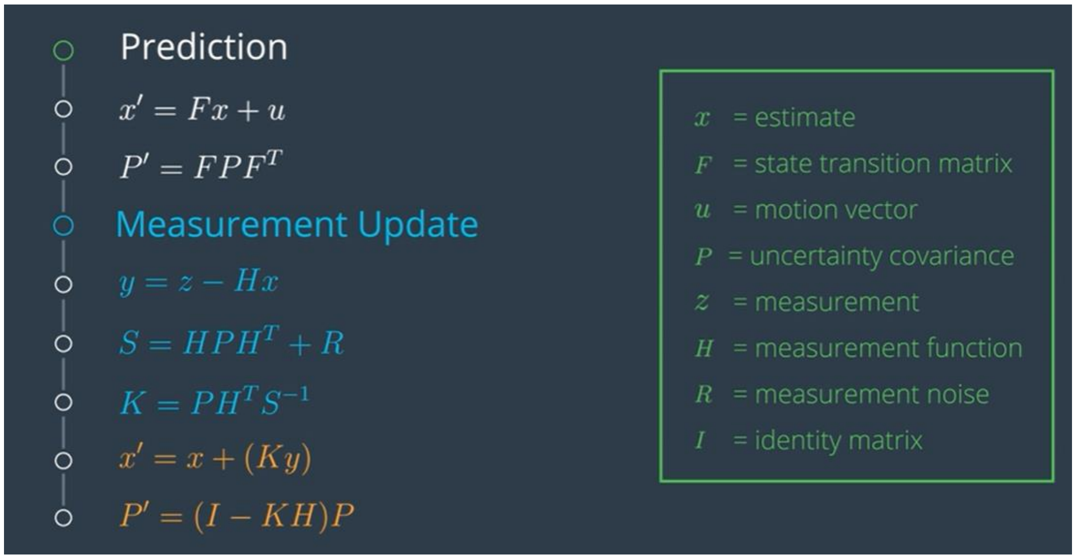
**Then is then mapped into a variable called K which is often called common gain where we invert the matrix s then finally,**

**We actually update our estimate and our uncertainty using what ought to be the most cryptic equation that you have seen in a long time.**

**Now, vote this down so that you have a complete definition,  but this is something you should not memorize.**

**If you really wish understand ,this math happen just a generalization of the math I gave you higher dimensional spaces.**

**There`s set of linear algebraic equations that implement the Kalman filter in higher dimensions.**



Prediction

$x' = Fx + u$

$P' = FPF^T$

Measurement Update

$y = z - Hx$

$S = HPH^T + R$

$K = PH^T S^{-1}$

$x' = x + (Ky)$

$P' = (I - KH)P$

| | |
|---|---|
| $x$ | = estimate |
| $F$ | = state transition matrix |
| $u$ | = motion vector |
| $P$ | = uncertainty covariance |
| $z$ | = measurement |
| $H$ | = measurement function |
| $R$ | = measurement noise |
| $I$ | = identity matrix |

### A note on time

In the above equation we see that x' = x + velocity, but we know that the complete equation is `x' = x + delta-t*velocity`. In this example, Sebastian assumes that delta t = 1 (that a time step is always one), which is allows him to simplify this equation.

## What You've Learned

So far, you've really done a lot in this section of the course. You've:

- Programmed a **histogram filter**
- Learned about the sense/move cycle in a Kalman Filter
- Implemented a 1D Kalman Fillter
- Learned how to represent a car's motion and **state** in a **vector** that could be transformed

And you've just learned a but about 2D and multidimensional Kalman Filters.

## Linear Algebra and the Kalman Equations

Now, Sebastian said "not to worry" about the complex Kalman equations that he's written, but this course is meant to be a good foundation for real-world programming challenges. So, the rest of this lesson will be about how to really understand these equations. Along the way, you'll learn a bit about how to approach a challenge like this and learn a lot of linear algebra that will allow you to read any equations like the Kalman filter equations with relative ease!

If you are already comfortable with linear algebra, feel free to skip around this section, otherwise, let's learn more about these equations and the powerful tool: linear algebra!

As Sebastian has repeatedly said, it is not important to memorize equations, but as a self-driving car engineering, it will be important for you to find be able and use the equations you need when you need to use them.
So imagine that at some point in the future you are working on the traffic protection team for some self-driving car company and you are trying to build a prototype of some object tracking code.

## Kalman Filter Equations Fx versus Bu

Consider this specific Kalman filter equation: $x' = Fx + Bu$ $x'=Fx+Bu$.
This equation is the move function that updates your beliefs in between sensor measurements. Fx models motion based on velocity, acceleration, angular velocity, etc of the object you are tracking.

B is called the control matrix and u is the control vector. Bu measures extra forces on the object you are tracking. An example would be if a robot was receiving direct commands to move in a specific direction, and you knew what those commands were and when they occurred. Like if you told your robot to move backwards 10 feet, you could model this with the Bu term.

When you take the self-driving car engineer nanodegree, you'll use Kalman filters to track objects that are moving around your vehicle like other cars, pedestrians, bicyclists, etc. In those cases, you would ignore $Bu$ $Bu$ because you do not have control over the movement of other objects. The Kalman filter equation becomes $x' = Fx$.

## MATRICES AND TRANSFORMATION OF STATE

<u>**Simplyfing Kalman Filter:**</u>

$$\hat{x}_{k|k-1} = F_k \hat{x}_{k-1|k-1} + B_k u_k$$

1- **Remove hat while the hat does have physical significance it indicates that the variable is an estimate.**
2- **It`s not telling us to do anything mathematically with these equations.**

$$x_{k|k-1} = F_k x_{k-1|k-1} + B_k u_k$$

1- **We are going to take all x_k given k-1 and we are going to replace that with x prime**
2- **Not only is this easier to look at it also consistent with the notation you have been using with Sebastian.**

$$x' = F_k x + B_k u_k$$

**Finally we are just going to get rid of these lone k`s remove subscript they are cluttering things up and i`d rather just look at the equation like this.**

$$x' = Fx + Bu$$

**Remember it`s some math here just looks simpler and look like all call Kalman filter equation.**

**Convert this equation to >>>>**

## MATRICES AND TRANSFORMATION OF STATE

$$\boxed{\hat{\mathbf{x}}_{k|k-1} = \mathbf{F}_k\hat{\mathbf{x}}_{k-1|k-1} + \mathbf{B}_k\mathbf{u}_k} \longrightarrow x' = Fx + Bu$$

$$\boxed{\mathbf{P}_{k|k-1} = \mathbf{F}_k\mathbf{P}_{k-1|k-1}\mathbf{F}_k^{\mathrm{T}} + \mathbf{Q}_k} \longrightarrow P' = FPF^T + Q$$

$$\boxed{\tilde{\mathbf{y}}_k = \mathbf{z}_k - \mathbf{H}_k\hat{\mathbf{x}}_{k|k-1}} \longrightarrow y = z - Hx'$$

$$\boxed{\mathbf{S}_k = \mathbf{R}_k + \mathbf{H}_k\mathbf{P}_{k|k-1}\mathbf{H}_k^{\mathrm{T}}} \longrightarrow S = R + HP'H^T$$

$$\boxed{\mathbf{K}_k = \mathbf{P}_{k|k-1}\mathbf{H}_k^{\mathrm{T}}\mathbf{S}_k^{-1}} \longrightarrow K = P'H^TS^{-1}$$

$$\boxed{\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k\tilde{\mathbf{y}}_k} \longrightarrow x = x' + Ky$$

$$\boxed{\mathbf{P}_{k|k} = (\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)\mathbf{P}_{k|k-1}} \longrightarrow P = (I - KH)P'$$

$$\boxed{\tilde{\mathbf{y}}_{k|k} = \mathbf{z}_k - \mathbf{H}_k\hat{\mathbf{x}}_{k|k}} \longrightarrow y = z - Hx$$

**In fact, we can actually use certain aspects of these equations right now to highlight what we are going to cover in the rest lesson**

**1-* Let`s focus on the difference between lower case and upper case there`s actually meaning there.**

**Generally lower case variables indicate vector while uppercase variables indicate matrices**

$$\begin{array}{|c|l}\hline x' & = Fx + Bu \\ P' & = FPF^T + Q \\ \hline \end{array}$$

**We will also talk about addition and multiplication of vectors and matrices**

**Special matrix identity**

## Representing State with Matrices

### The State Vector

You just learned how to represent a self-driving car's state using a motion model. It turns out that matrices provide a very convenient and compact form for representing a vehicle's state.

Let's go back to the constant velocity motion model:

$distance = velocity \times time$

The vehicle's state is represented by the two variables distance and velocity. If you were going to store these two variables in Python, you'd probably use a list like this:

```
state = [distance, velocity]
```

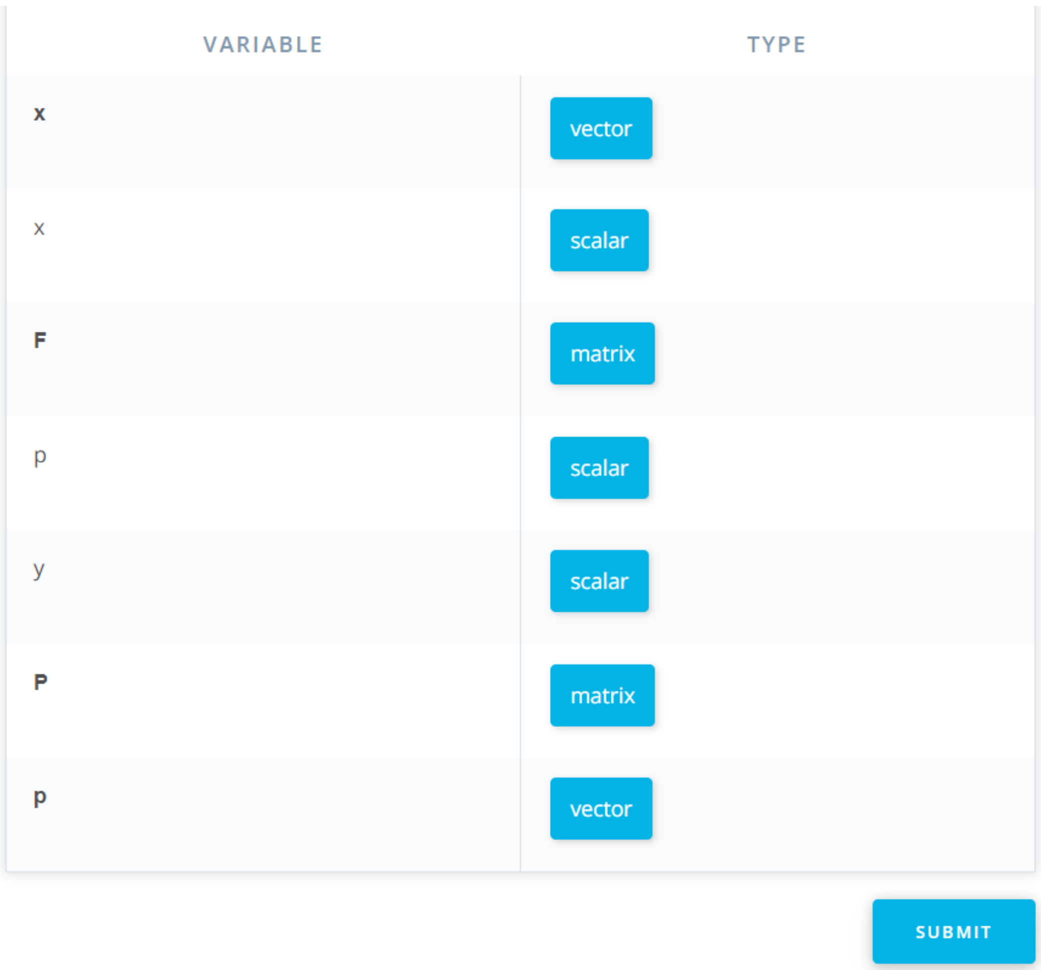
That Python code looks a lot like a mathematical concept called a vector. A vector is essentially a list where each element in the list contains some information.

You could imagine that a state vector could have even more information. In a two-dimensional world, the state could have a $distance_x$, $distance_y$, $velocity_x$ and a $velocity_y$.

In Python, the list would look like this:

```
state = [distance_x, distance_y, velocity_x, velocity_y]
```

## MATRICES AND TRANSFORMATION OF STATE



| VARIABLE | TYPE |
|---|---|
| $\mathbf{x}$ | vector |
| $x$ | scalar |
| $\mathbf{F}$ | matrix |
| $p$ | scalar |
| $y$ | scalar |
| $\mathbf{P}$ | matrix |
| $\mathbf{p}$ | vector |

## Kalman Equation Reference

We're just including this here in case you want to refer back to the Kalman Filter equations at any time. Feel free to move along :)

### Variable Definitions

$\hat{\mathbf{x}}$ - state vector

$\mathbf{F}$ - state transition matrix

$\mathbf{P}$ - error covariance matrix

$\mathbf{Q}$ - process noise covariance matrix

$\mathbf{R}$ - measurement noise covariance matrix

$\mathbf{S}$ - intermediate matrix for calculating Kalman gain

$\mathbf{H}$ - observation matrix

$\mathbf{K}$ - Kalman gain

$\tilde{\mathbf{y}}$ - difference between predicted state and measured state

$\mathbf{z}$ - measurement vector (lidar data or radar data, etc.)

$\mathbf{I}$ - Identity matrix

## MATRICES AND TRANSFORMATION OF STATE

**Prediction Step Equations**

PREDICT STATE VECTOR AND ERROR COVARIANCE MATRIX

$$\hat{\mathbf{x}}_{\mathbf{k|k-1}} = \mathbf{F_k}\hat{\mathbf{x}}_{\mathbf{k-1|k-1}}$$

$$\mathbf{P_{k|k-1}} = \mathbf{F_k P_{k-1|k-1} F_k^T} + \mathbf{Q_k}$$

**Update Step Equations**

KALMAN GAIN

$$\mathbf{S_k} = \mathbf{H_k P_{k|k-1} H_k^T} + \mathbf{R_k}$$

$$\mathbf{K_k} = \mathbf{P_{k|k-1} H_k^T S_k}^{-1}$$

UPDATE STATE VECTOR AND ERROR COVARIANCE MATRIX

$$\tilde{\mathbf{y}}_{\mathbf{k}} = \mathbf{z_k} - \mathbf{H_k}\hat{\mathbf{x}}_{\mathbf{k|k-1}}$$

$$\hat{\mathbf{x}}_{\mathbf{k|k}} = \hat{\mathbf{x}}_{\mathbf{k|k-1}} + \mathbf{K_k}\tilde{\mathbf{y}}_{\mathbf{k}}$$

$$\mathbf{P_{k|k}} = (\mathbf{I} - \mathbf{K_k H_k})\mathbf{P_{k|k-1}}$$

Or an even more complex model could include information about the turning angle of the vehicle and the turning rate:

```
state = [distance_x, distance_y, velocity_x, velocity_y, angle, angle_rate]
```

### State Vector in a One-Dimensional World

For now, consider the one-dimensional model with just distance and velocity.

```
state = [distance, velocity]
```

How did you calculate the distance and velocity of the vehicle over time when the velocity was constant? There were two different equations.

$$\begin{cases} distance = velocity \times time \\ velocity = velocity \end{cases}$$

Now, think about these equations in terms of state. For convenience, you can represent distance as $x$, velocity as $v$, and time as $t$.

#### Initial State

When the vehicle first starts moving, you can consider that $t = t_0$, $x = x_0$ and $v = v_0$. So the state vector is $state_0 = [x_0, v_0]$.

#### First Time Step

What about after a certain amount of time has passed and now you are at a time $t_1$?

At $t_1$, the state vector is:

$state_1 = [x_1, v_1]$

## Updating State with Matrix Algebra

If you look back at the equation that updates the distance, you'll notice that distance depends on the previous distance, the initial velocity, and how much time has elapsed since the distance formula was updated. You end up with a generic function:

$$x_{t+1} = x_t + v_0 \times (t_{t+1} - t_t)$$

You can also write $t_{t+1} - t_t$ as:

$$\Delta t$$

For a constant velocity model, the generic velocity equation becomes: $v_{t+1} = v_t$

How could you combine the x and v equations into one matrix algebra expression? The matrix algebra would look like this:

$$\begin{bmatrix} x_{t+1} \\ v_{t+1} \end{bmatrix} = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \times \begin{bmatrix} x_t \\ v_t \end{bmatrix}$$

Don't worry if you're not sure what this expression means or how to multiply these matrices. You will learn how in this lesson.

## Notation

The matrix algebra equation you just saw is actually one part of the Kalman filter update equation.

Traditionally, the matrix operation:

$$\begin{bmatrix} x_{t+1} \\ v_{t+1} \end{bmatrix} = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \times \begin{bmatrix} x_t \\ v_t \end{bmatrix}$$

is represented by this notation for Kalman filters: $\mathbf{\hat{x}_{k|k-1}} = \mathbf{F}\mathbf{\hat{x}_{k-1|k-1}}$

where $\mathbf{\hat{x}}$ is the state vector and $\mathbf{F}$ is the matrix

According to the model formulas,

$x_1 = x_0 + v_0 \times (t_1 - t_0)$

and since velocity is constant:

$v_1 = v_0$.

#### Second Time Step

Then after the next time step $t_2$, the state vector is: $state_2 = [x_2, v_2]$

where

$x_2 = x_1 + v_0 \times (t_2 - t_1)$

and since velocity is constant $v_2 = v_0$.

### A Better Way

The math so far is not too hard, right? You have a distance equation and a velocity equation. You plug in the previous state into each equation, the time lapse, and you get the new velocity and the new distance.

But imagine what will happen as your self-driving car model gets more complex. What happens when you have to take into account an x-direction, y-direction, x and y velocities, and steering angle and angular velocity? Or what about an even more complex model like a drone or helicopter that also has a z-direction?

Instead of updating your equations one by one, you can actually use vectors and matrices to do all of the calculations in just one step.

where $\hat{\mathbf{x}}$ is the state vector and $\mathbf{F}$ is the matrix

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

$\mathbf{k-1}$ is the previous step and $\mathbf{k}$ is the current step.

You will see in the next part of the lesson why the notation contains $\mathbf{k-1|k-1}$ and $\mathbf{k|k-1}$ .

This notation can get a little bit confusing. For example, what is the difference between $x$ and $\hat{\mathbf{x}}$?

The regular $x$ would usually represent distance along the x-axis; on the other hand, the bold $\hat{\mathbf{x}}$ indicates a vector. In the one-dimensional case being discussed here, the $\hat{\mathbf{x}}$ vector contains two variables: distance along the x-axis and velocity; hence $\hat{\mathbf{x}} = \begin{bmatrix} x \\ v \end{bmatrix}$.

Why is there a capitalized bold $\mathbf{F}$ instead of $\mathbf{f}$? The capitalized, bold $\mathbf{F}$ tells you that this variable is a matrix.

## Coding

### Vectors in Python

In the following exercises, you will work on coding vectors in Python.

Assume that you have a state vector

$$\mathbf{x_0}$$

representing the x position, y position, velocity in the x direction, and velocity in the y direction of a car that is driving in front of your vehicle. You are tracking the other vehicle.

Currently, the other vehicle is 5 meters ahead of you along your x-axis, 2 meters to your left along your y-axis, driving 10 m/s in the x direction and 0 m/s in the y-direction. How would you represent this in a Python list where the vector contains `<x, y, vx, vy>` in exactly that order?