Kalman Filters: An Overviews

Adapted from “How a Kalman filter works, in pictures” by Tim Babb
http://www.bzarg.com/p/how-a-kalman-filter-works-in-pictures/

What exactly is a Kalman filter?

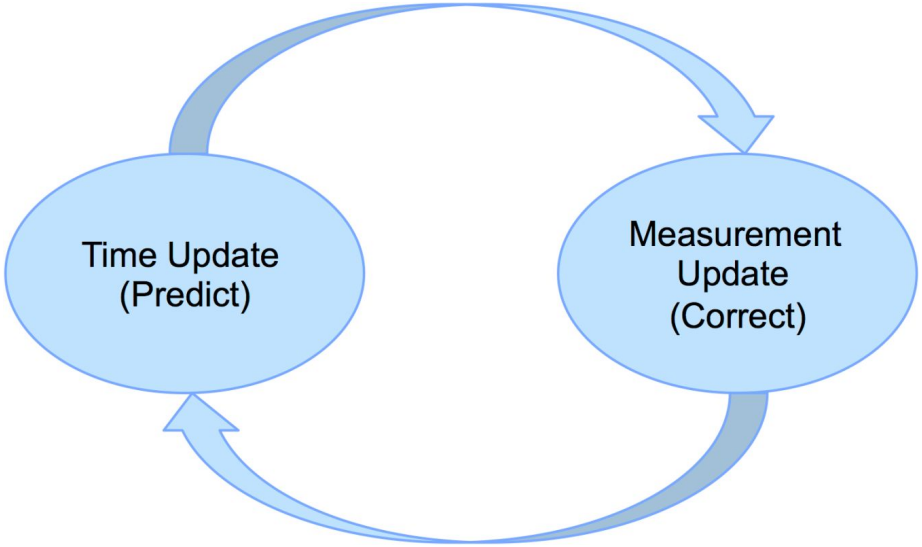
It’s essentially tracking algorithm that recursively tries to follow an un-observable quantity (“state”) using related signal observations (measurements). It attempts to mitigate measurement noise, but also projects observations into a state estimate, which is the basis the Kalman filter and its extensions

You can use a Kalman filter in any place where you have **uncertain information** about some dynamic system, and you can make an **educated guess** about what the system is going to do next.

Kalman filters are ideal for systems which are **continuously changing**. They have the advantage that they are light on memory (they don’t need to keep any history other than the previous state), and they are very fast, making them well suited for real time problems and embedded systems.

Kalman Filter Summary

- Time update (a priori estimates): Project state and covariance forward to the next time step, that is, predict.
- Measurement update (a posteriori estimates): Update with a (noisy) measurement of the process, that is, correct.



What can we do with a Kalman filter?

What can we do with a Kalman filter?

What can we do with a Kalman filter?

We'll say our robot has a state x_k, which is just a position and a velocity:

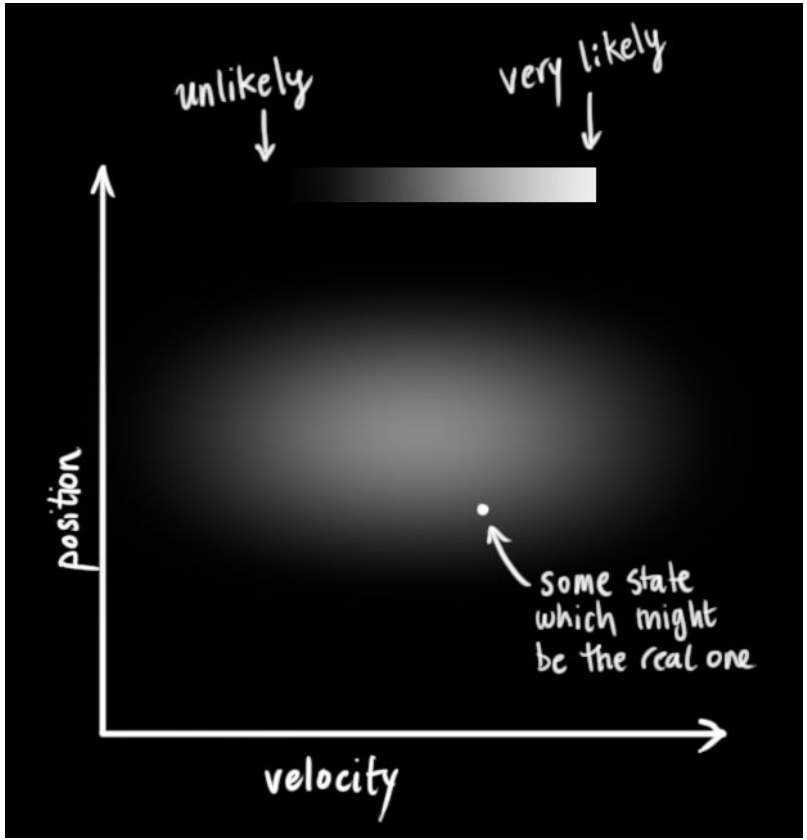
$$\vec{x} = \begin{bmatrix} p \\ v \end{bmatrix}$$

What can we do with a Kalman filter?

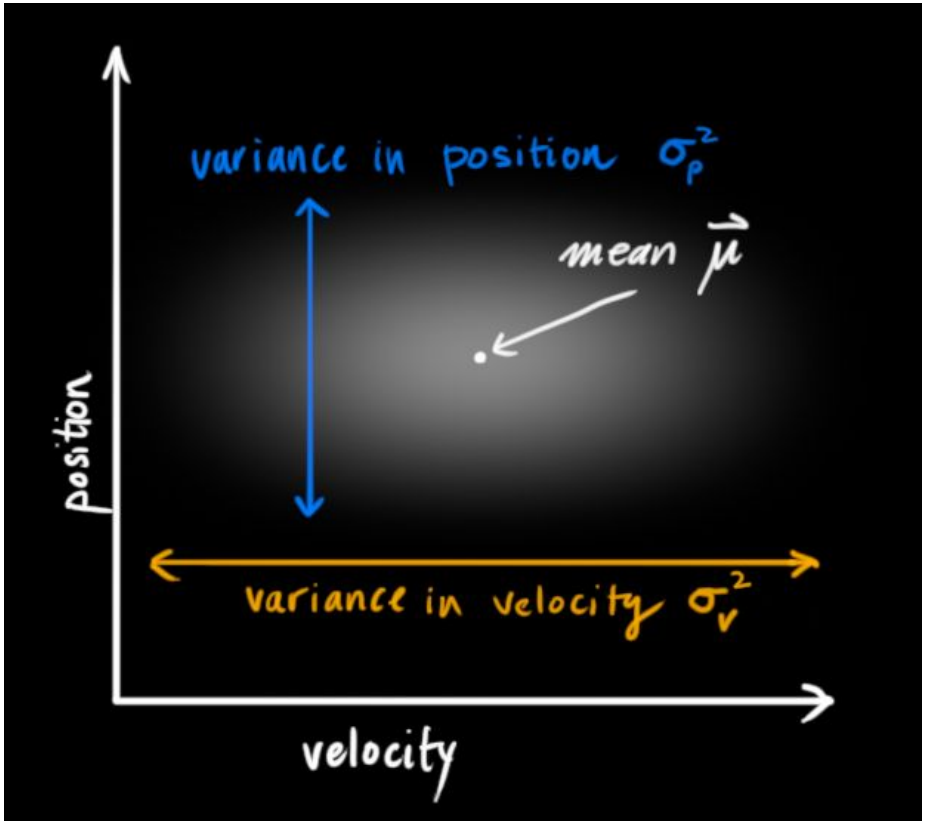
Our robot also has a GPS sensor, which is accurate to about 10 meters, which is good, but it needs to know its location more precisely than 10 meters.

What can we do with a Kalman filter?

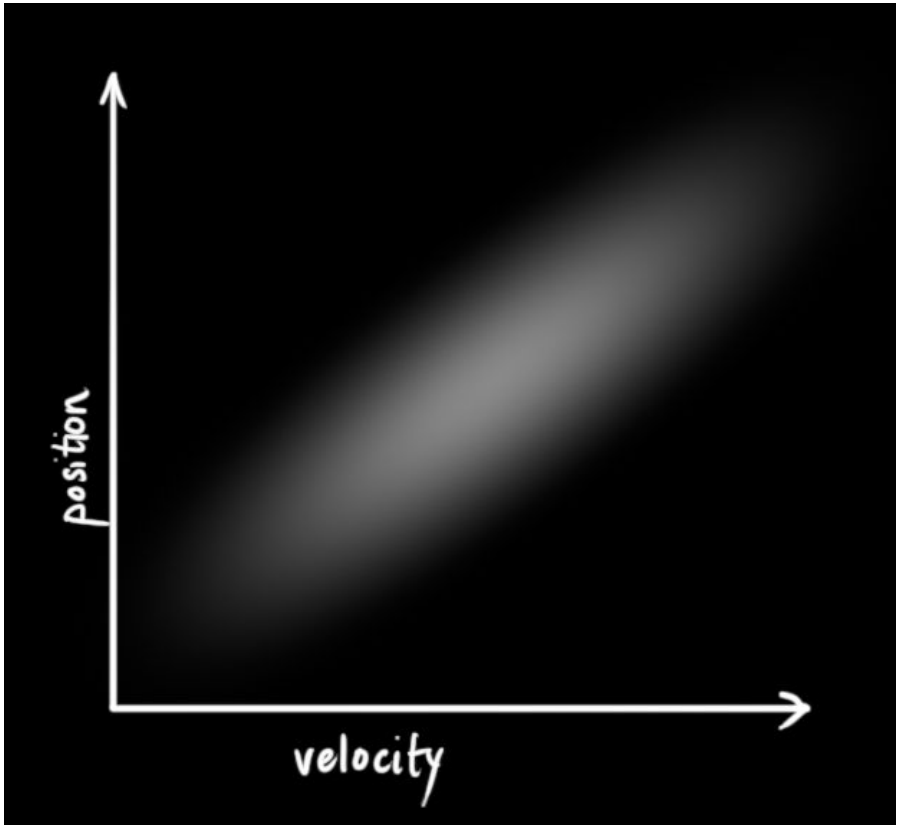
How a Kalman filter sees your problem



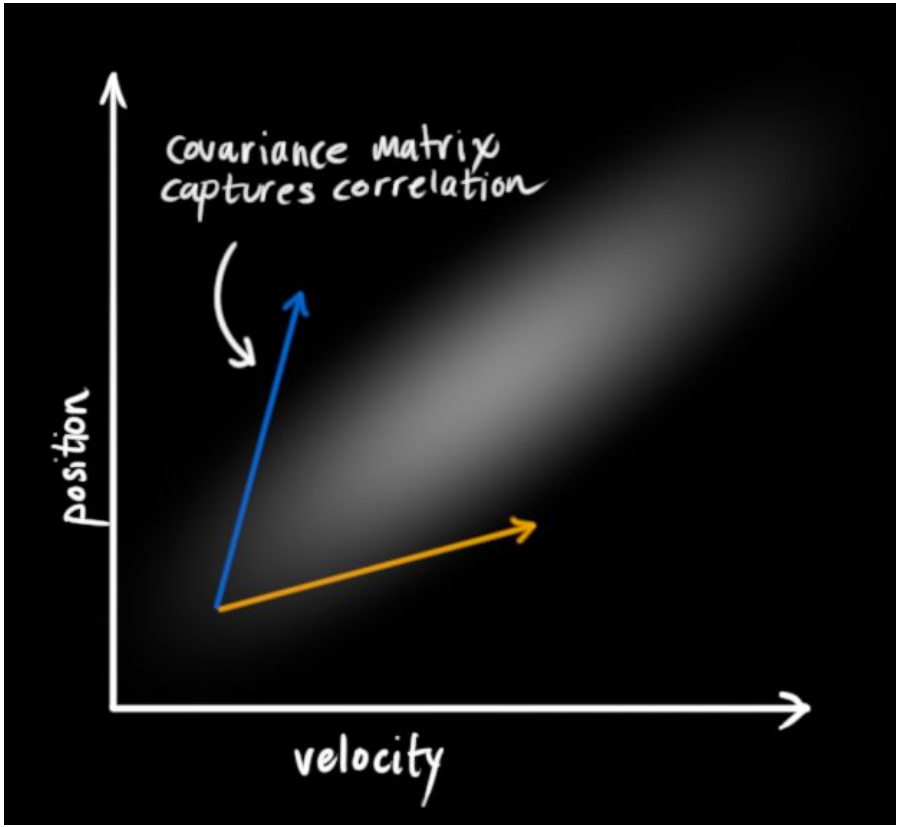
How a Kalman filter sees your problem



A more realistic example



A more realistic example



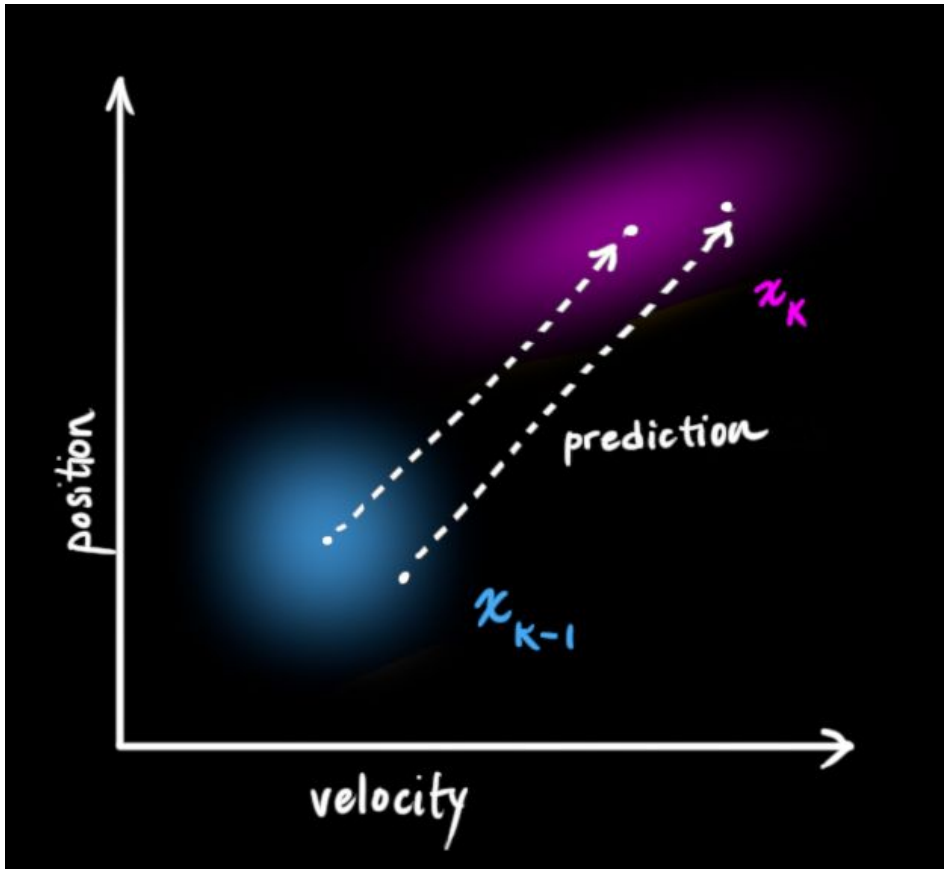
Describing the problem with matrices

We're modeling our knowledge about the state as a Gaussian blob, so we need two pieces of information at time *k*: We'll call our best estimate $\hat{\mathbf{x}}_k$, and its covariance matrix $\mathbf{P}_k$.

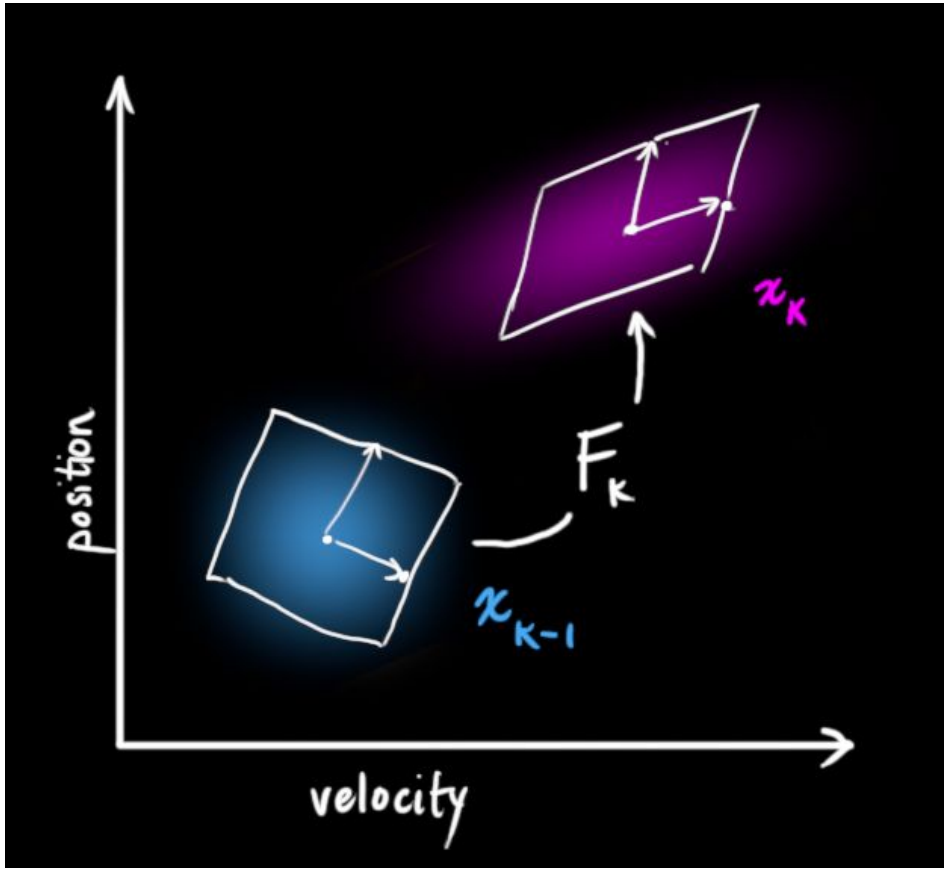
$$\hat{\mathbf{x}}_k = \begin{bmatrix} \text{position} \\ \text{velocity} \end{bmatrix}$$

$$\mathbf{P}_k = \begin{bmatrix} \Sigma_{pp} & \Sigma_{pv} \\ \Sigma_{vp} & \Sigma_{vv} \end{bmatrix}$$

Next, we need some way to look at the **current state** (at time **k-1**) and **predict the next state** at time **k**



We represent this prediction step with a matrix, **F_k (the system state matrix)**



What does this look like in matrix notation?

What does this look like in matrix notation?

Let’s use a basic kinematic formula:

$$\begin{aligned} p_k &= p_{k-1} + \Delta t v_{k-1} \\ v_k &= v_{k-1} \end{aligned}$$

What does this look like in matrix notation?

We'll use a really basic kinematic formula:

$$\begin{aligned} p_k &= p_{k-1} + \Delta t v_{k-1} \\ v_k &= v_{k-1} \end{aligned}$$

In other words:

$$\begin{aligned} \hat{\mathbf{x}}_k &= \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \hat{\mathbf{x}}_{k-1} \\ &= \mathbf{F}_k \hat{\mathbf{x}}_{k-1} \end{aligned}$$

What does this look like in matrix notation?

We'll use a really basic kinematic formula:

$$\begin{aligned} p_k &= p_{k-1} + \Delta t v_{k-1} \\ v_k &= v_{k-1} \end{aligned}$$

In other words:

$$\begin{aligned} \hat{\mathbf{x}}_k &= \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \hat{\mathbf{x}}_{k-1} \\ &= \mathbf{F}_k \hat{\mathbf{x}}_{k-1} \end{aligned}$$

Using a linear algebra identity:

$$\begin{aligned} Cov(x) &= \Sigma \\ Cov(\mathbf{A}x) &= \mathbf{A}\Sigma\mathbf{A}^T \end{aligned}$$

What does this look like in matrix notation?

We'll use a really basic kinematic formula:

$$\begin{aligned} p_k &= p_{k-1} + \Delta t v_{k-1} \\ v_k &= v_{k-1} \end{aligned}$$

In other words:

$$\begin{aligned} \hat{\mathbf{x}}_k &= \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \hat{\mathbf{x}}_{k-1} \\ &= \mathbf{F}_k \hat{\mathbf{x}}_{k-1} \end{aligned}$$

Using a linear algebra identity:

$$\begin{aligned} Cov(x) &= \Sigma \\ Cov(\mathbf{A}x) &= \mathbf{A}\Sigma\mathbf{A}^T \end{aligned}$$

And combining:

$$\begin{aligned} \hat{\mathbf{x}}_k &= \mathbf{F}_k \hat{\mathbf{x}}_{k-1} \\ \mathbf{P}_k &= \mathbf{F}_\mathbf{k} \mathbf{P}_{k-1} \mathbf{F}_k^T \end{aligned}$$

External Influence

External Influence

Let’s say we know the expected acceleration a due to the throttle setting or control commands.

From basic kinematics we get:

$$p_k = p_{k-1} + \Delta t v_{k-1} + \frac{1}{2} a \Delta t^2$$
$$v_k = v_{k-1} + a \Delta t$$

External Influence

Let's say we know the expected acceleration *a* due to the throttle setting or control commands.

From basic kinematics we get:

$$
\begin{aligned}
p_k &= p_{k-1} + \Delta t v_{k-1} + \frac{1}{2} a \Delta t^2 \\
v_k &= v_{k-1} + a \Delta t
\end{aligned}
$$

Which becomes:

$$
\begin{aligned}
\hat{\mathbf{x}}_k &= \mathbf{F}_k \hat{\mathbf{x}}_{k-1} + \begin{bmatrix} \frac{\Delta t^2}{2} \\ \Delta t \end{bmatrix} a \\
&= \mathbf{F}_k \hat{\mathbf{x}}_{k-1} + \mathbf{B}_k \vec{\mathbf{u}}_k
\end{aligned}
$$

External Influence

Let's say we know the expected acceleration a due to the throttle setting or control commands.

From basic kinematics we get:

$$p_k = p_{k-1} + \Delta t v_{k-1} + \frac{1}{2} a \Delta t^2$$
$$v_k = v_{k-1} + a \Delta t$$

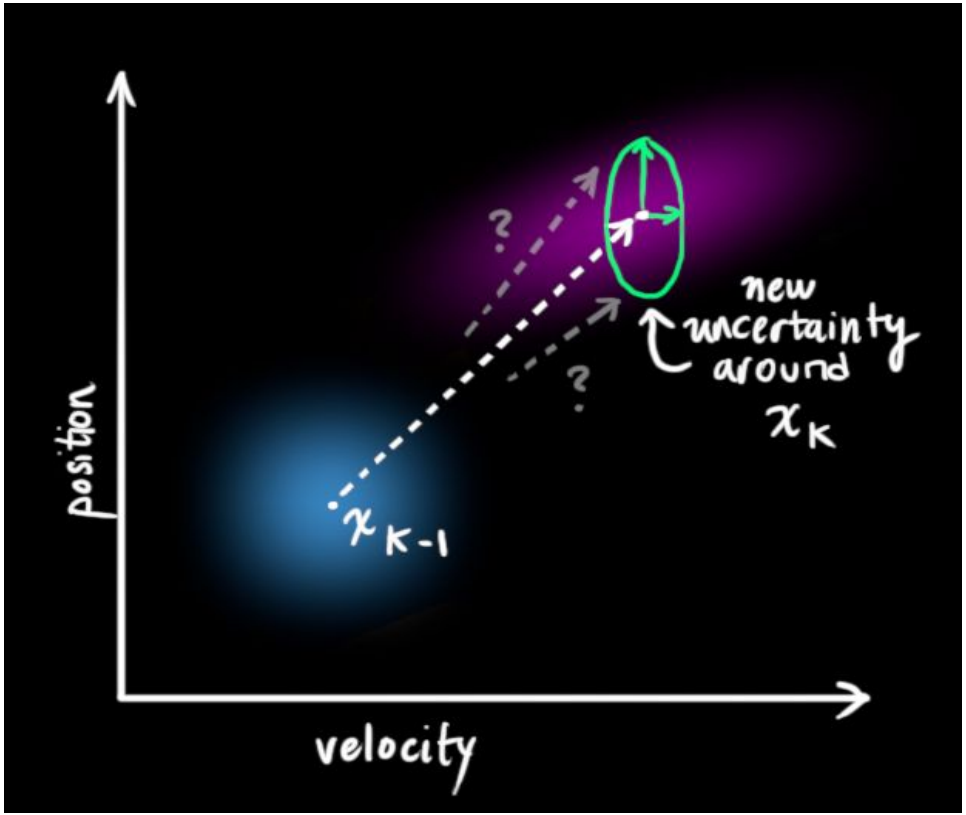
Which becomes:

$$\hat{\mathbf{x}}_k = \mathbf{F}_k \hat{\mathbf{x}}_{k-1} + \begin{bmatrix} \frac{\Delta t^2}{2} \\ \Delta t \end{bmatrix} a$$
$$= \mathbf{F}_k \hat{\mathbf{x}}_{k-1} + \mathbf{B}_k \vec{\mathbf{u}}_k$$

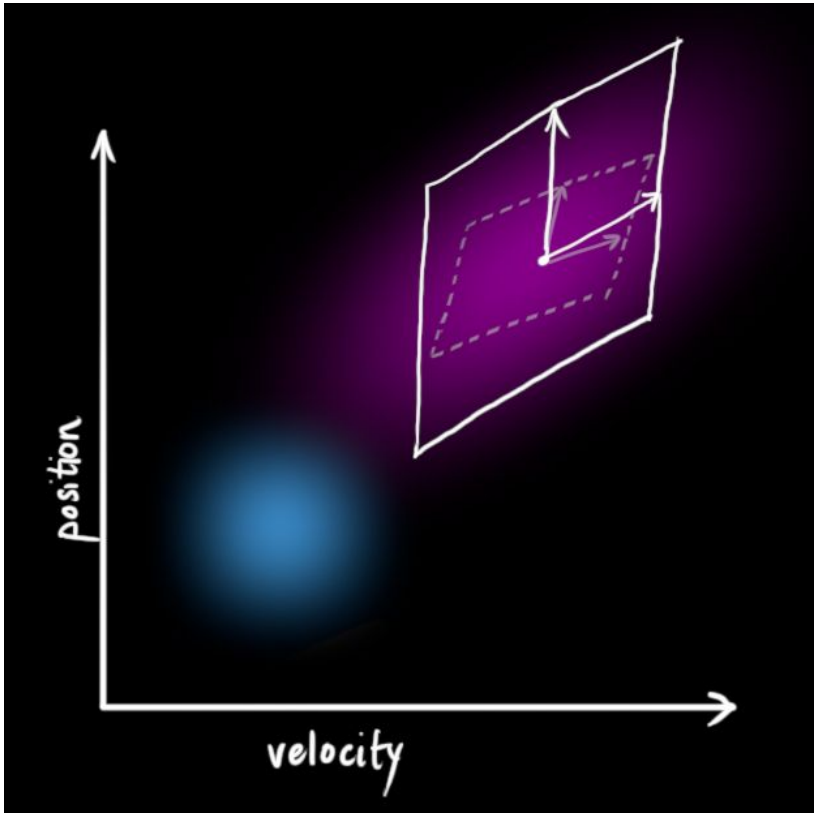
Where $\mathbf{B}_k$ is called the **control matrix** and $\mathbf{u}_k$ the **control vector.** (For very simple systems with no external influence, you could omit these).

External Uncertainty

External Uncertainty



External Uncertainty



External Uncertainty

We get the expanded covariance by simply **adding** $\mathbf{Q}_k$, giving our complete expression for the **prediction step**:

$$\hat{\mathbf{x}}_k = \mathbf{F}_k\hat{\mathbf{x}}_{k-1} + \mathbf{B}_k\vec{\mathbf{u}_k}$$
$$\mathbf{P}_k = \mathbf{F_k}\mathbf{P}_{k-1}\mathbf{F}_k^T + \mathbf{Q}_k$$

External Uncertainty

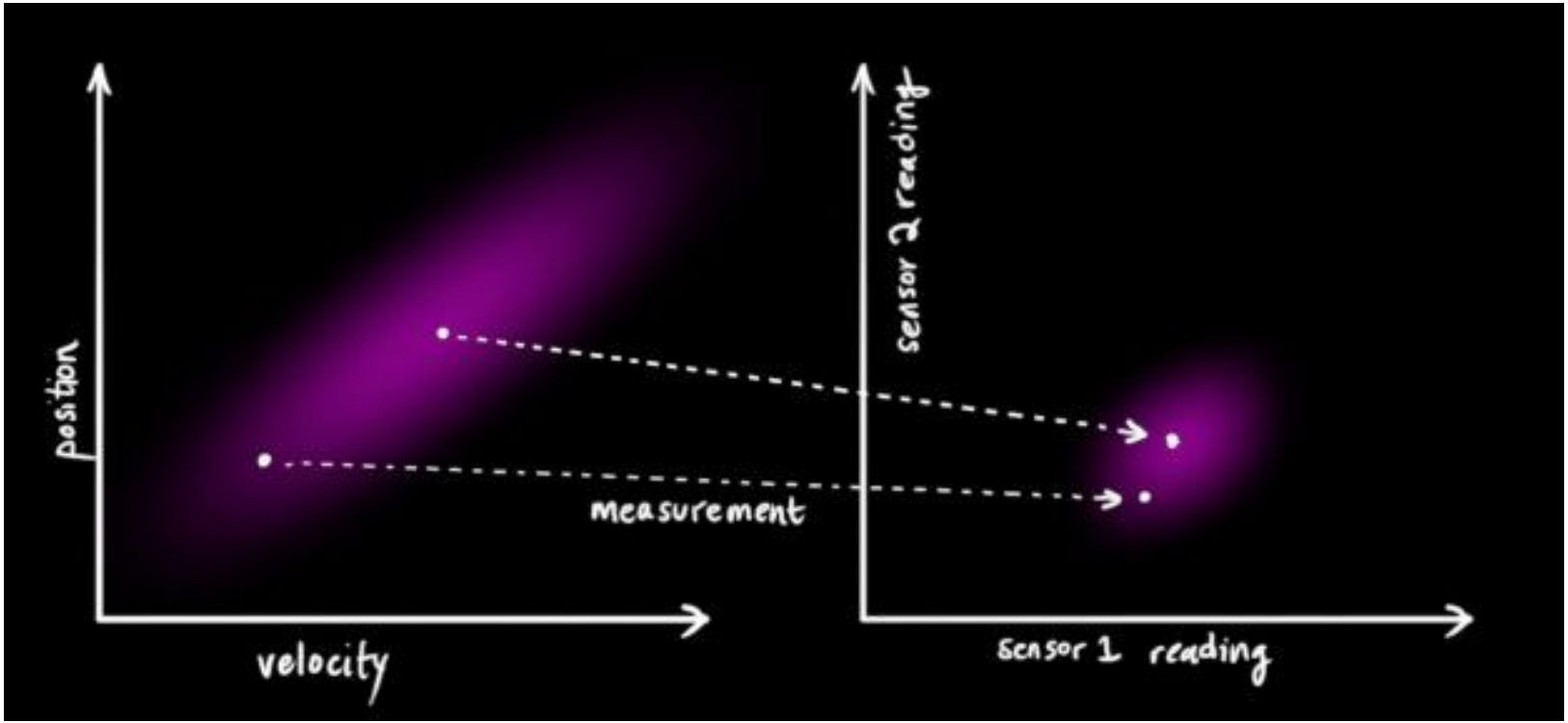
We get the expanded covariance by simply **adding** Q_k, giving our complete expression for the **prediction step**:

$$\hat{\mathbf{x}}_k = \mathbf{F}_k\hat{\mathbf{x}}_{k-1} + \mathbf{B}_k\vec{\mathbf{u}_k}$$
$$\mathbf{P}_k = \mathbf{F_k}\mathbf{P}_{k-1}\mathbf{F}_k^T + \mathbf{Q}_k$$

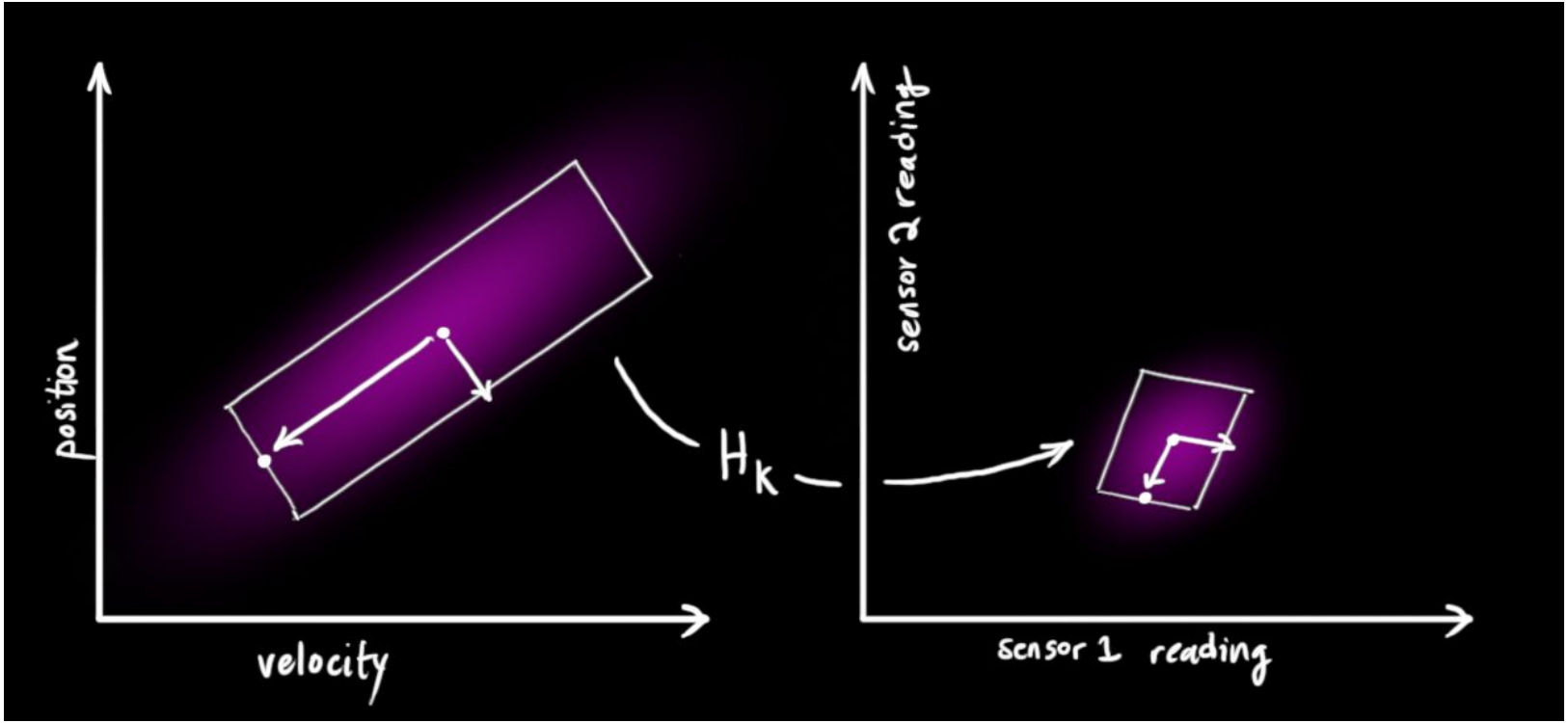
In other words, the **new best estimate** is a **prediction** made from **previous best estimate**, plus a **correction** for **known external influences**.

And the **new uncertainty** is **predicted** from the **old uncertainty**, with some **additional uncertainty from the environment**.

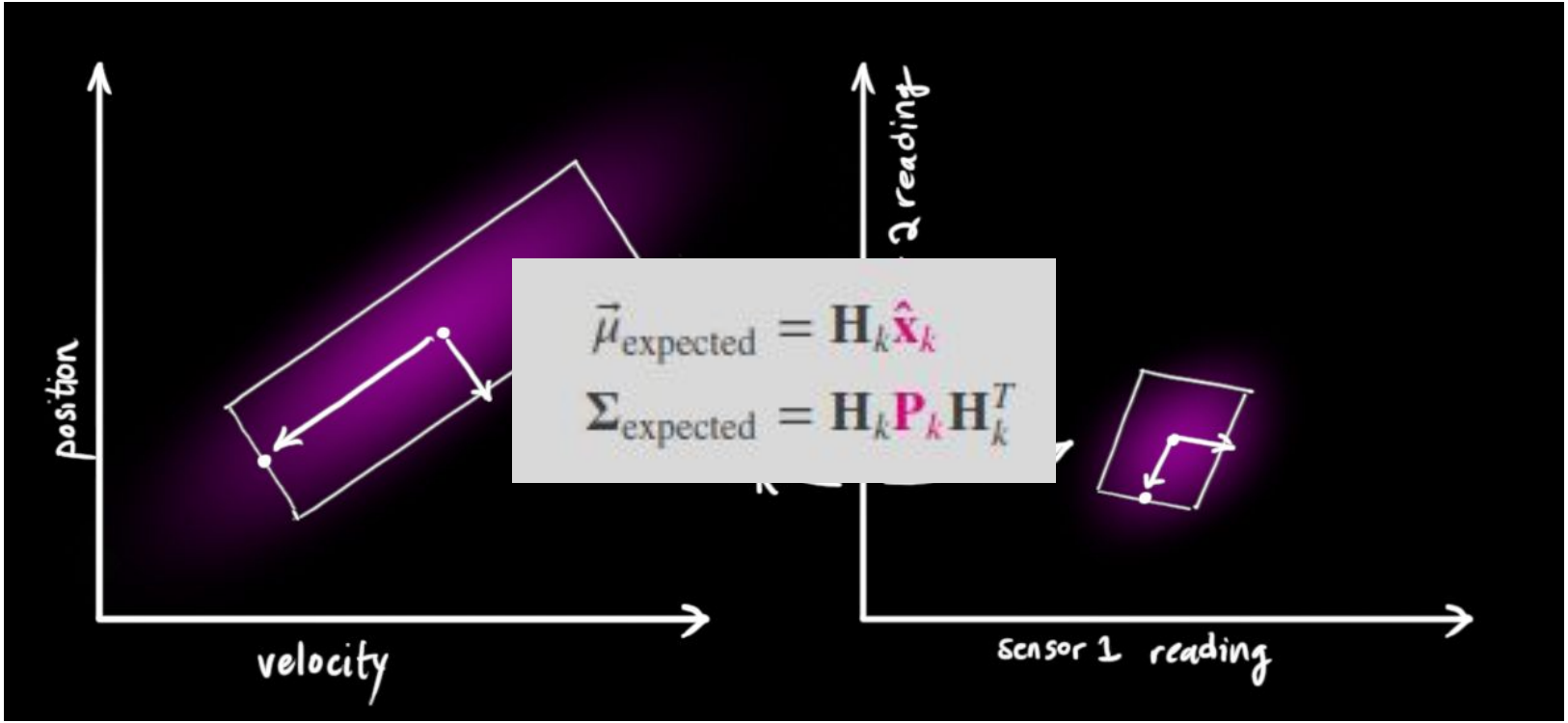
Refining our estimate with measurements



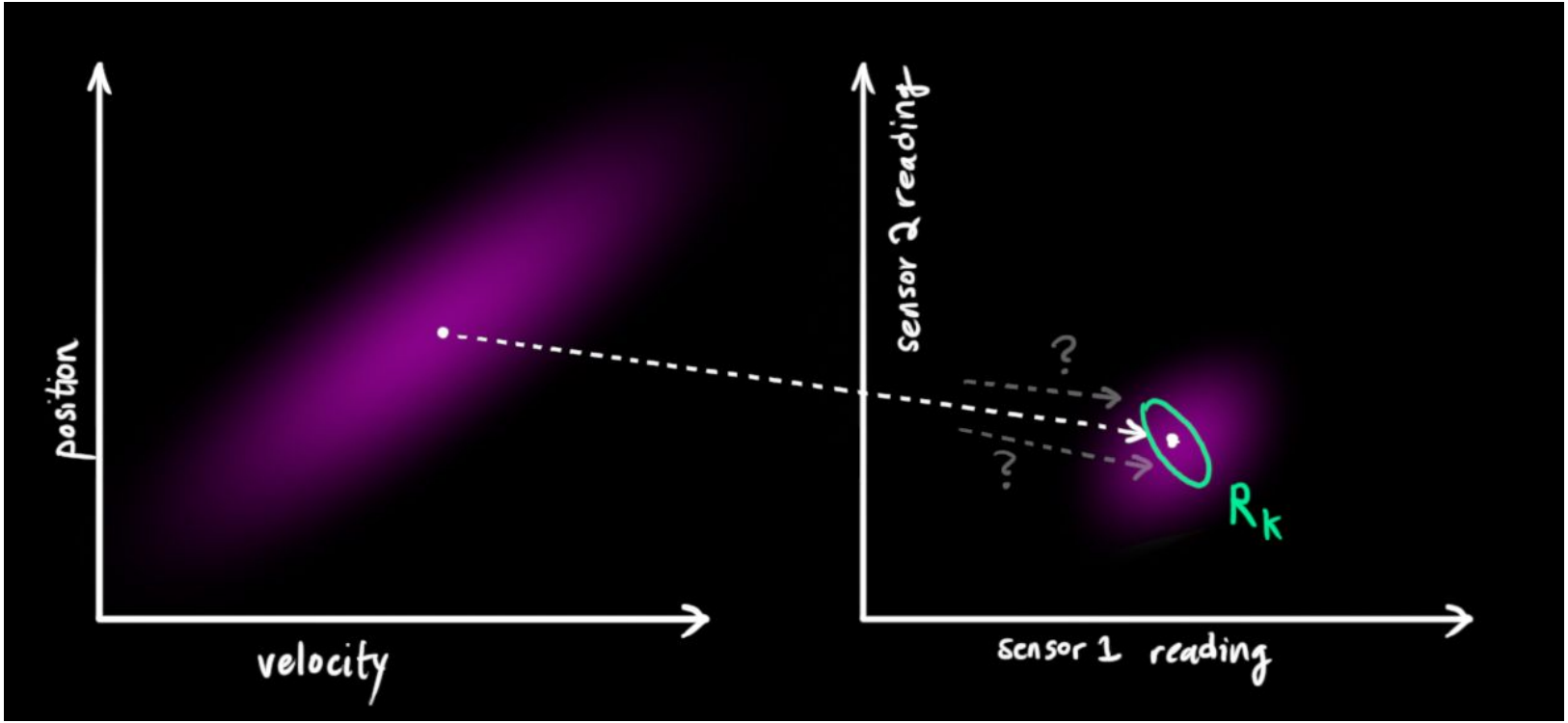
Refining our estimate with measurements



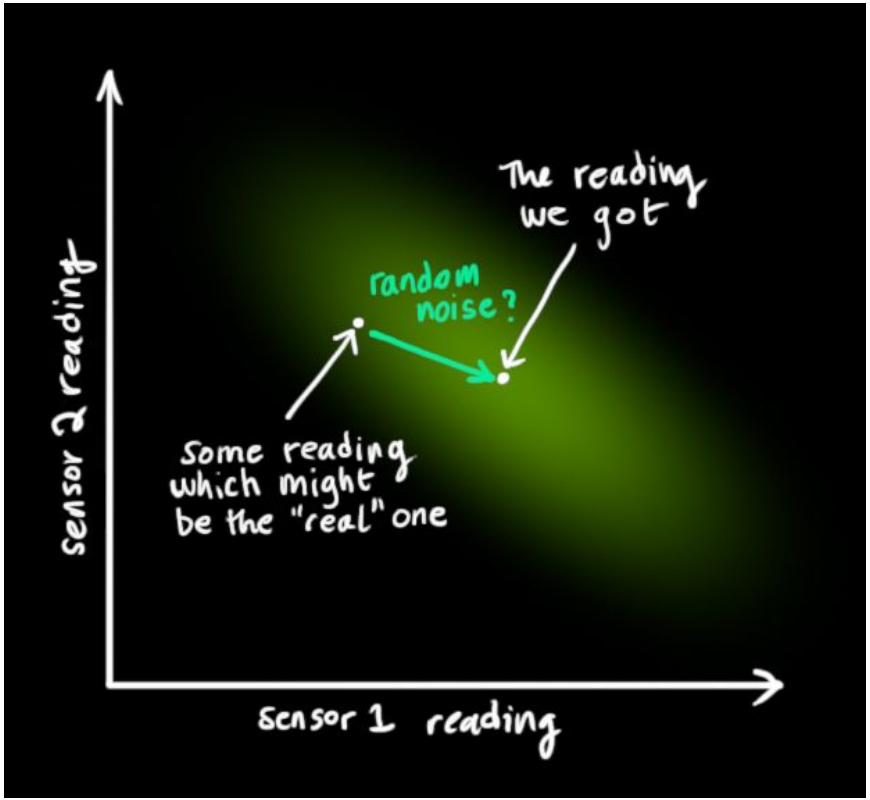
And in matrix notation



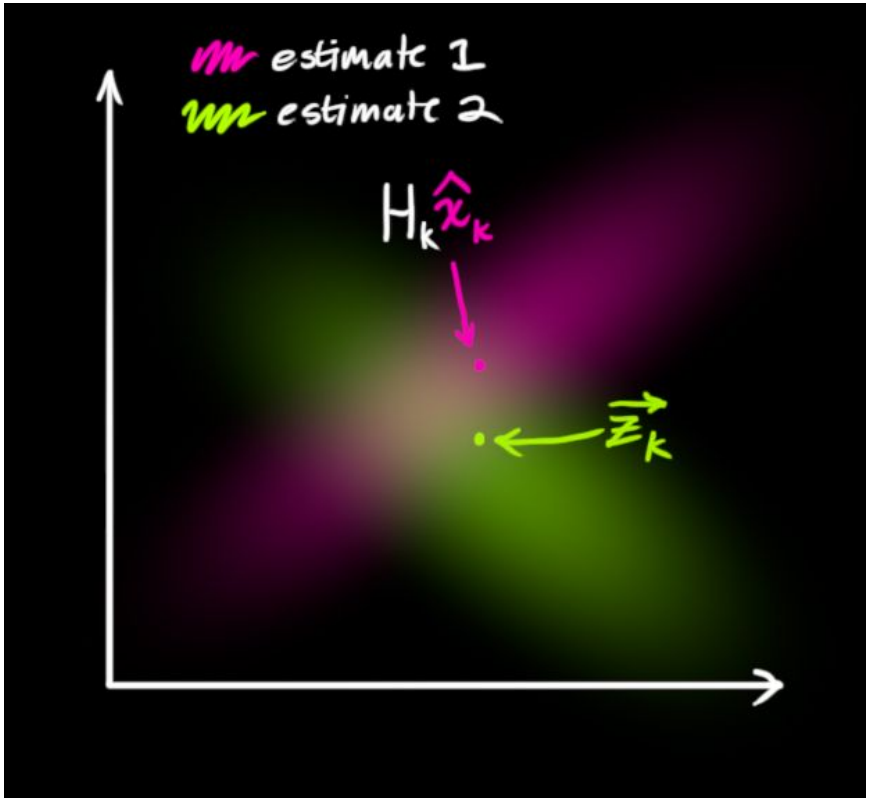
Dealing with sensor noise



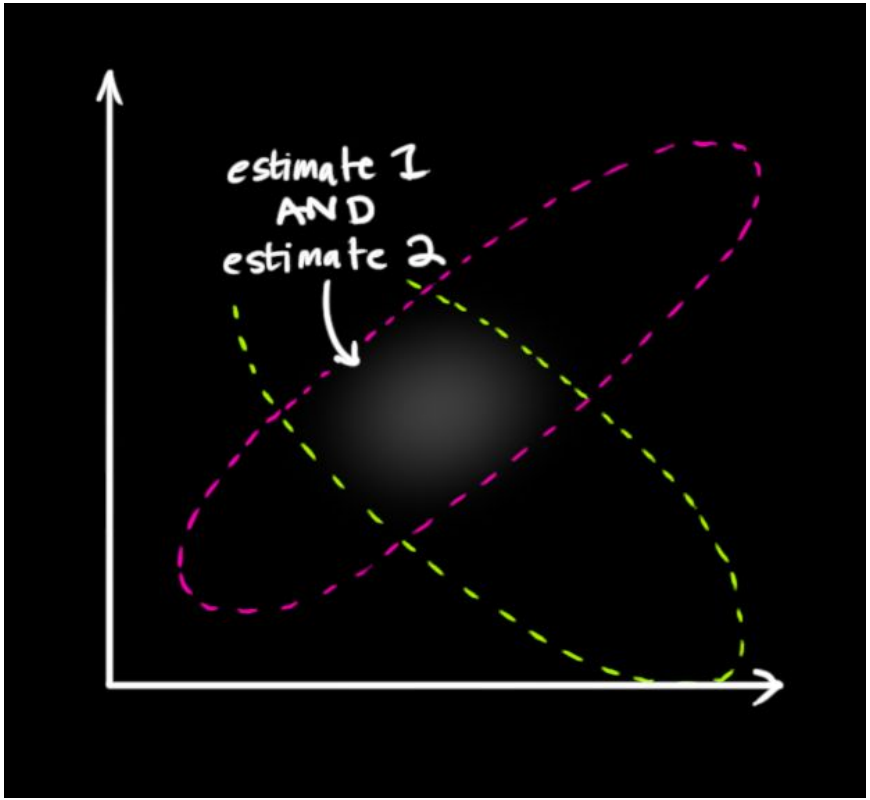
Dealing with sensor noise



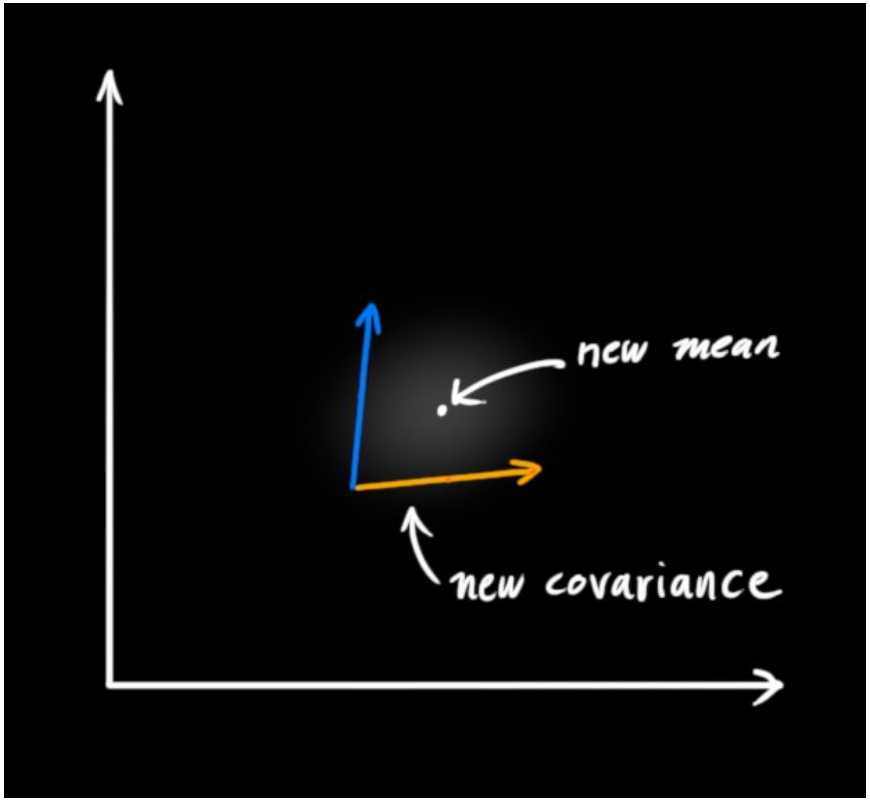
Now we have two Gaussian blobs



So what's the most likely state?

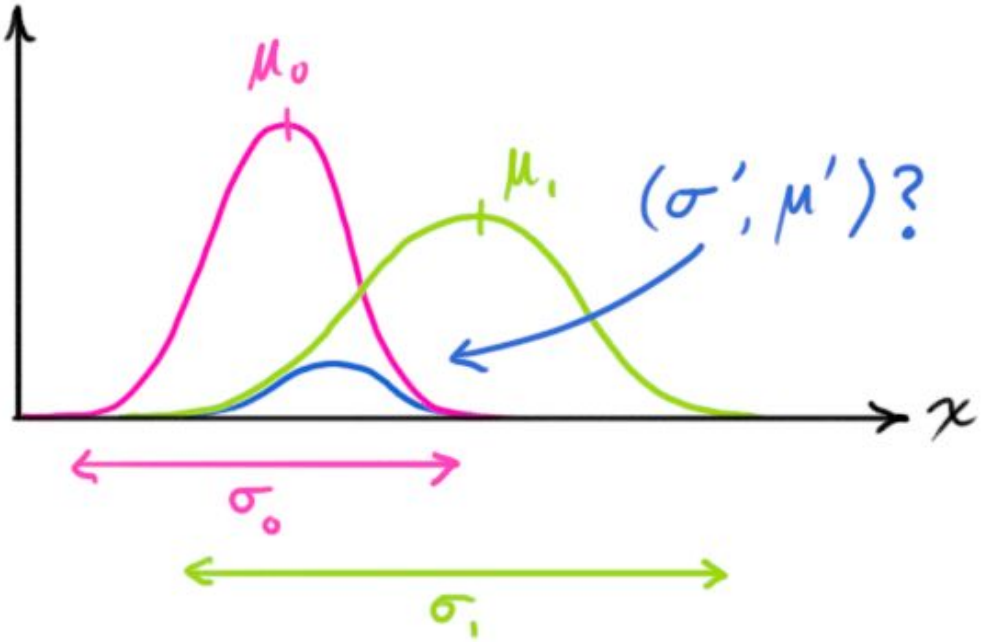


What we're left with



Combining 1D Gaussians

Combining 1D Gaussians



$$\mathcal{N}(x, \mu_0, \sigma_0) \cdot \mathcal{N}(x, \mu_1, \sigma_1) \stackrel{?}{=} \mathcal{N}(x, \mu', \sigma')$$

Questions???

Resources

1. How a Kalman filter works, in pictures
   http://www.bzarg.com/p/how-a-kalman-filter-works-in-pictures/
2. A Kalman Filtering Tutorial for Undergraduate Students
   http://aircconline.com/ijcses/V8N1/8117ijcses01.pdf
3. Kalman Filter
   https://en.wikipedia.org/wiki/Kalman_filter
4. Understanding Kalman Filters
   https://www.mathworks.com/videos/series/understanding-kalman-filters.html

Backup

Equation Summary

Substituting and doing some algebra (being careful to renormalize, so that the total probability is 1) to obtain:

$$\mu' = \mu_0 + \frac{\sigma_0^2(\mu_1 - \mu_0)}{\sigma_0^2 + \sigma_1^2}$$

$$\sigma'^2 = \sigma_0^2 - \frac{\sigma_0^4}{\sigma_0^2 + \sigma_1^2}$$

We can simplify by factoring out a little piece and calling it **k (the Kalman gain):**

$$\mathbf{k} = \frac{\sigma_0^2}{\sigma_0^2 + \sigma_1^2}$$

$$\mu' = \mu_0 + \mathbf{k}(\mu_1 - \mu_0)$$

$$\sigma'^2 = \sigma_0^2 - \mathbf{k}\sigma_0^2$$

What does it look like in matrix form?

$$\mathbf{K} = \Sigma_0(\Sigma_0 + \Sigma_1)^{-1}$$

$$\vec{\mu}' = \vec{\mu_0} + \mathbf{K}(\vec{\mu_1} - \vec{\mu_0})$$

$$\mathbf{\Sigma}' = \Sigma_0 - \mathbf{K}\Sigma_0$$

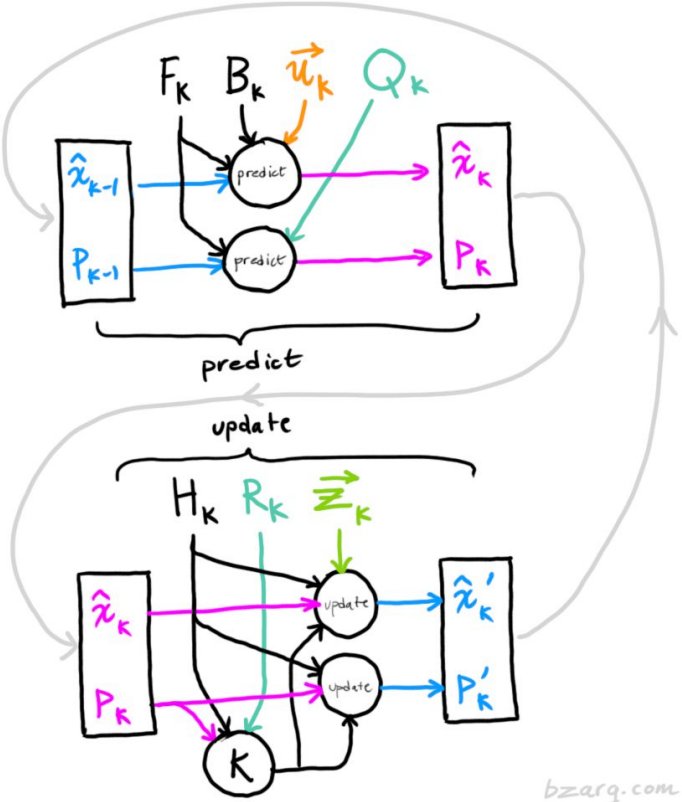
Putting it all together

$$\hat{\mathbf{x}}_k' = \hat{\mathbf{x}}_k + \mathbf{K}'(\vec{\mathbf{z}_k} - \mathbf{H}_k\hat{\mathbf{x}}_k)$$

$$\mathbf{P}_k' = \mathbf{P}_k - \mathbf{K}'\mathbf{H}_k\mathbf{P}_k$$

$$\mathbf{K}' = \mathbf{P}_k\mathbf{H}_k^T(\mathbf{H}_k\mathbf{P}_k\mathbf{H}_k^T + \mathbf{R}_k)^{-1}$$

Kalman Filter Information Flow



bzarg.com