Gaussian Filters: Kalman Filter

Corina Barbalata

Mechanical and Industrial Engineering

iCORE Laboratory

cbarbalata@lsu.edu

Readings for this class

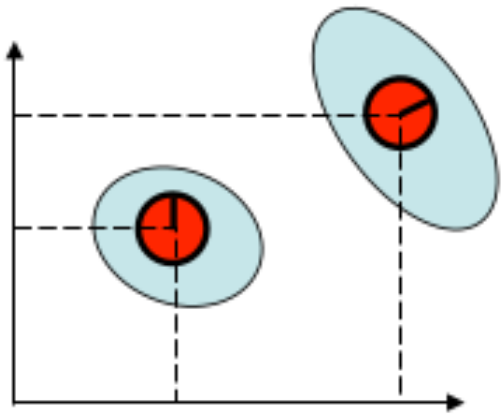
- Chapter 3: Gaussian Filters (pages 33-48) from *Probabilistic Robotics*, Sebastian Thrun

Gaussian Filters

- Gaussian filters constitute the earliest tractable implementations of the Bayes filter for continuous spaces.
- They are also by far the most popular family of techniques to date.

Gaussian Filters

- **Hypothesis**: Uncertainty in robot pose can be represented through a Gaussian Random Vector (GRV)



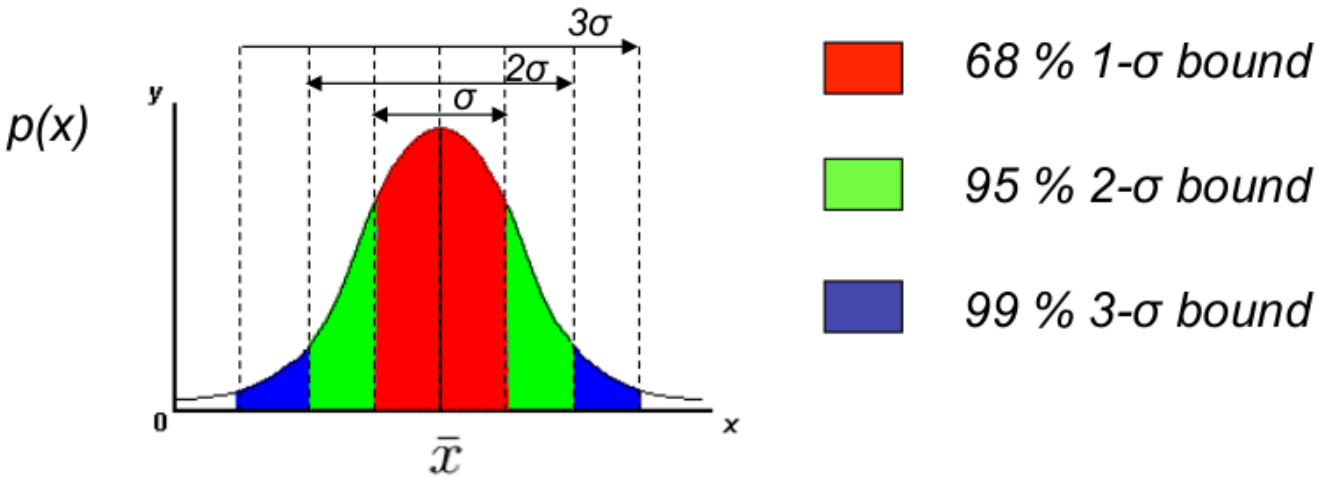
Position corrupted with Random Gaussian noise

- **Why the Gaussian hypothesis is so important?**
  - **Simplicity**: With only two parameters the PDF is completely defined.
  - **Computational efficiency**: Make some problems tractable.
  - **Central Limit Theorem (CLT)**: The sum of a sequence of n independent random variables tends to a Gaussian PDF as $n \to \infty$

Gaussian Random Variables

- The PDF of a (scalar) Gaussian or a normal random variable is

$$p(x) = \mathcal{N}(x; \bar{x}, \sigma^2) \triangleq \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\bar{x})^2}{2\sigma^2}}$$



Gaussian Random Vectors (GRV)

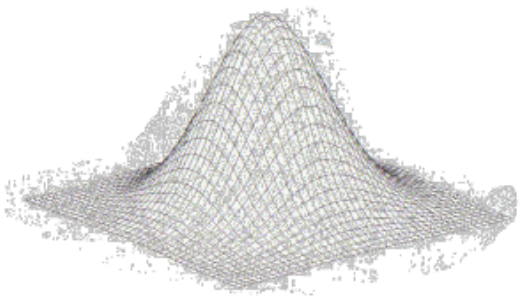
- A vector-valued Gaussian random variable has the density:

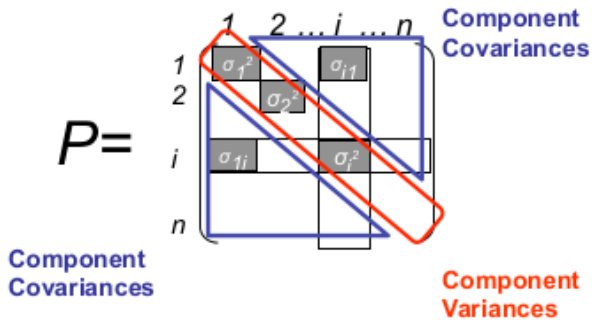
$$\mathcal{N}(X;\bar{X},\sigma^2) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(X-\bar{X})^2}{2\sigma^2}}$$

where

$$\bar{X} = E[X], \quad \text{the expected value vecor/ mean vector}$$

$$P = E[(X-\bar{X})(X-\bar{X})^T], \quad \text{the Covariance matrix}$$





σ_i^2: Variance of component i

σ_{ij}: Covariance between component i & j

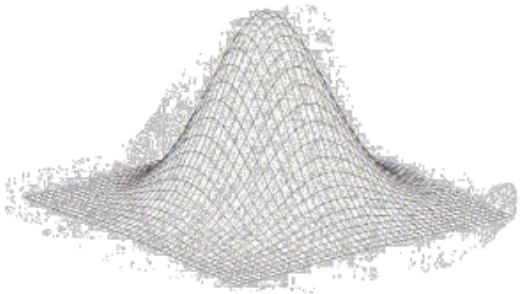
The covariance matrix

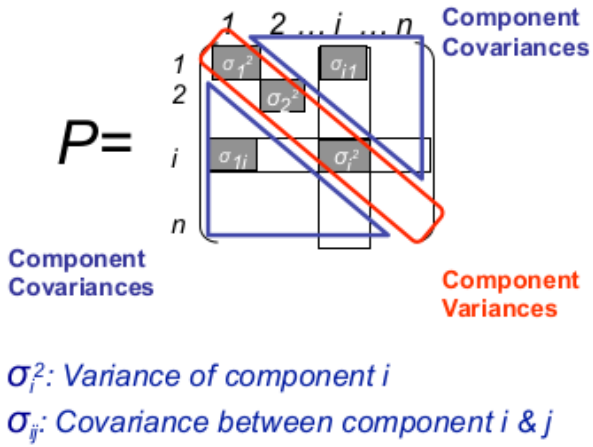
$$P = E[(X - \bar{X})(X - \bar{X})^T] =$$

$$\begin{bmatrix} (X_1 - EX_1)^2 & (X_1 - EX_1)(X_2 - EX_2) & \dots & (X_1 - EX_1)(X_n - EX_n) \\ (X_2 - EX_2)(X_1 - EX_1) & (X_2 - EX_2)^2 & \dots & (X_2 - EX_2)(X_n - EX_n) \\ \vdots & \vdots & \vdots & \vdots \\ (X_n - EX_n)(X_1 - EX_1) & (X_n - EX_n)(X_2 - EX_2) & \dots & (X_n - EX_n)^2 \end{bmatrix}$$

$$= \begin{bmatrix} \mathrm{Var}(X_1) & \mathrm{Cov}(X_1, X_2) & \dots & \mathrm{Cov}(X_1, X_n) \\ \mathrm{Cov}(X_2, X_1) & \mathrm{Var}(X_2) & \dots & \mathrm{Cov}(X_2, X_n) \\ \vdots & \vdots & \vdots & \vdots \\ \mathrm{Cov}(X_n, X_1) & \mathrm{Cov}(X_n X_2) & \dots & \mathrm{Var}(X_n) \end{bmatrix}.$$

Gaussian Random Vectors (GRV)



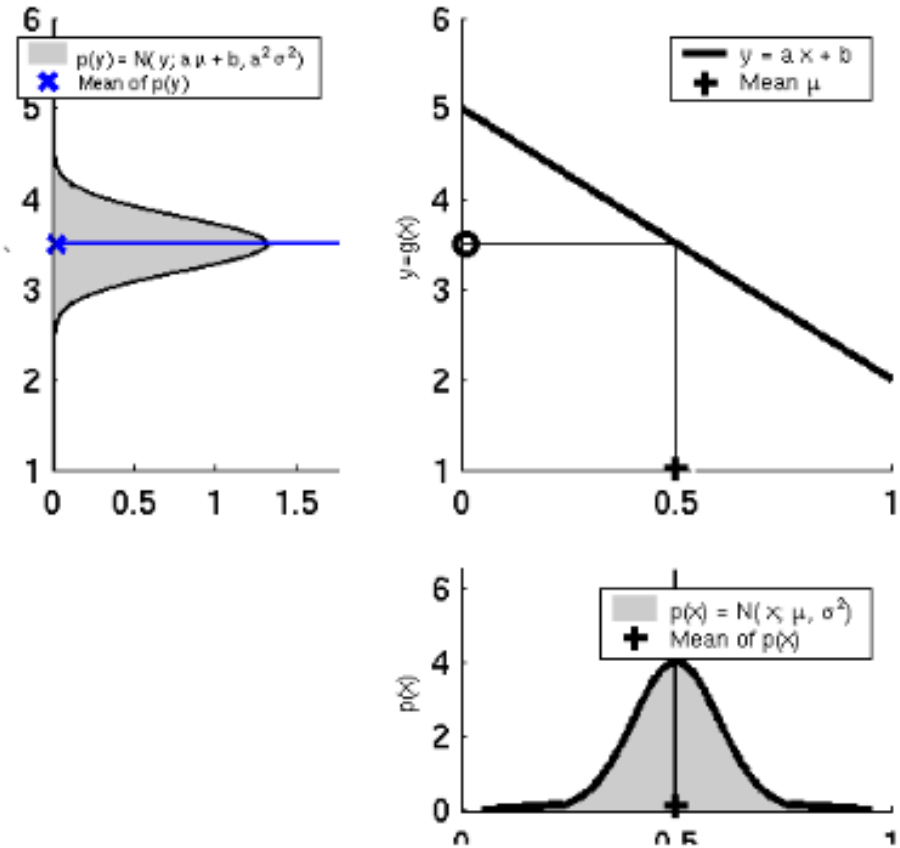


- $P = P^T$ (symetric matrix) $\sigma_{ij} = \sigma_{ji}$
- P diagonal $\Rightarrow$ uncorrelated $\Rightarrow$ independent

Normally, independence doen't mean uncorrelation. But for **GRV** **independence $\Leftrightarrow$ uncorrelation**

Properties: Linear transformations of GRVs

- We stay in the "Gaussian world" as long as we start with Gaussians and perform only linear transformations.



Properties: Linear transformations of GRVs

Scalar

$$\left.\begin{array}{l} X \sim N(\mu,\sigma^2) \\ Y = aX + b \end{array}\right\} \Rightarrow \quad Y \sim N(a\mu + b, a^2\sigma^2)$$

$$\left.\begin{array}{l} X_1 \sim N(\mu_1,\sigma_1^{\,2}) \\ X_2 \sim N(\mu_2,\sigma_2^{\,2}) \\ Z = aX_1 + bX_2 \end{array}\right\} \Rightarrow Z \sim N\left(a\mu_1 + b\mu_2, a^2\sigma_1^2 + 2ab\sigma_{12}^{\;2} + b^2\sigma_2^2\right)$$

$$\left.\begin{array}{l} X_1 \sim N(\mu_1,\sigma_1^{\,2}) \\ X_2 \sim N(\mu_2,\sigma_2^{\,2}) \end{array}\right\} \Rightarrow p(X_1)\cdot p(X_2) \sim N\left(\frac{\sigma_2^{\,2}}{\sigma_1^{\,2}+\sigma_2^{\,2}}\mu_1 + \frac{\sigma_1^{\,2}}{\sigma_1^{\,2}+\sigma_2^{\,2}}\mu_2, \quad \frac{1}{\sigma_1^{\,-2}+\sigma_2^{\,-2}}\right)$$

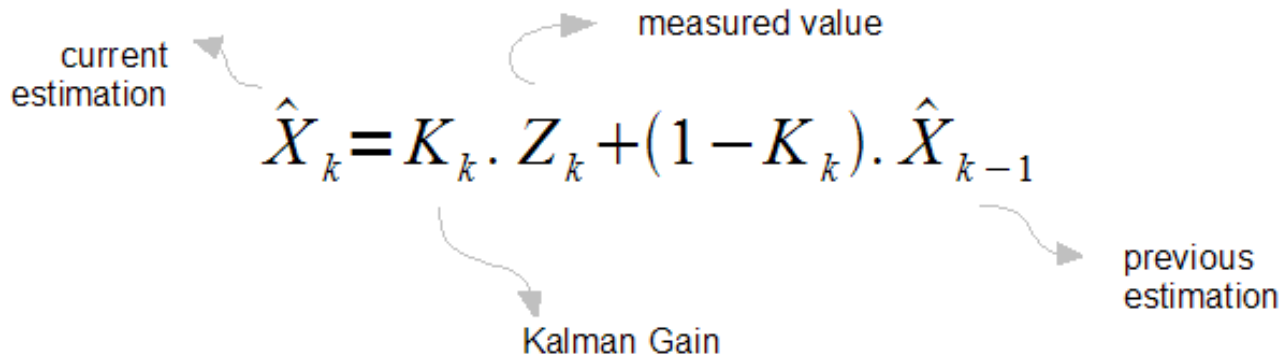
Vectorial

$$\left.\begin{array}{l} X \sim N(\mu,P) \\ Y = AX + B \end{array}\right\} \quad \Rightarrow \quad Y \sim N(A\mu + B, APA^T)$$

$$\left.\begin{array}{l} X_1 \sim N(\mu_1,P_1) \\ X_2 \sim N(\mu_2,P_2) \\ Z = AX_1 + BX_2 \end{array}\right\} \Rightarrow Z \sim N\left(A\mu_1 + B\mu_2, AP_1A^T + AP_{12}B^T + BP_{21}A^T + BP_2B^T\right)$$

$$\left.\begin{array}{l} X_1 \sim N(\mu_1,P_1) \\ X_2 \sim N(\mu_2,P_2) \end{array}\right\} \Rightarrow p(X_1)\cdot p(X_2) \sim N\left(\frac{P_2}{P_1+P_2}\mu_1 + \frac{P_1}{P_1+P_2}\mu_2, \quad \frac{1}{P_1^{-1}+P_2^{-1}}\right)$$

Simplistic view of Kalman Filtering

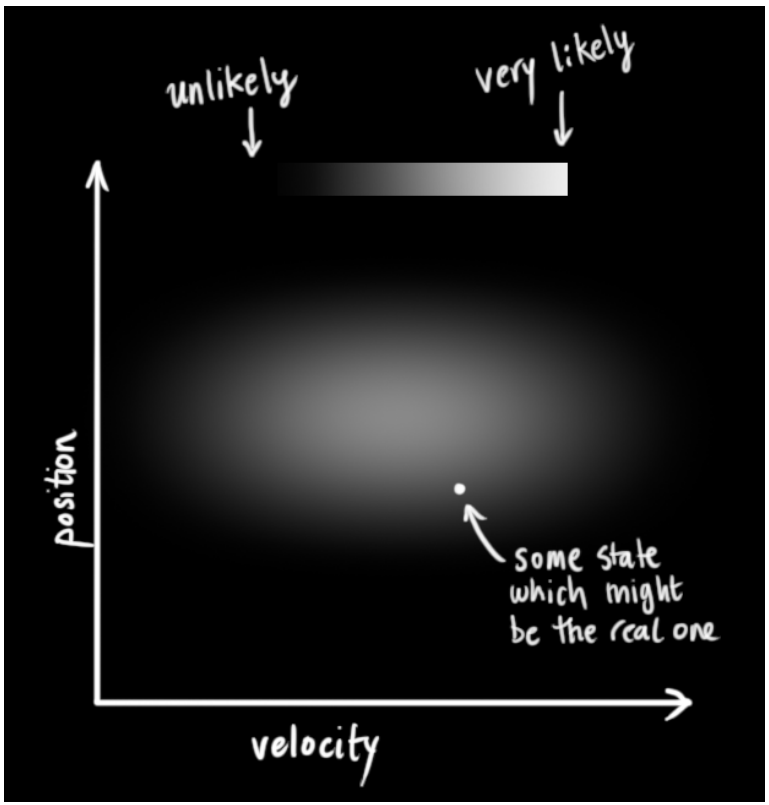


- GOAL: find $\hat{X}_k$ that estimates the signal X, for each consequent k.
- ASSUMPTIONS: uncertainty/noise in the measurement value Z_k
- UNKNOWN: the Kalman Gain K_k

How Kalman Filter interprets the problem?

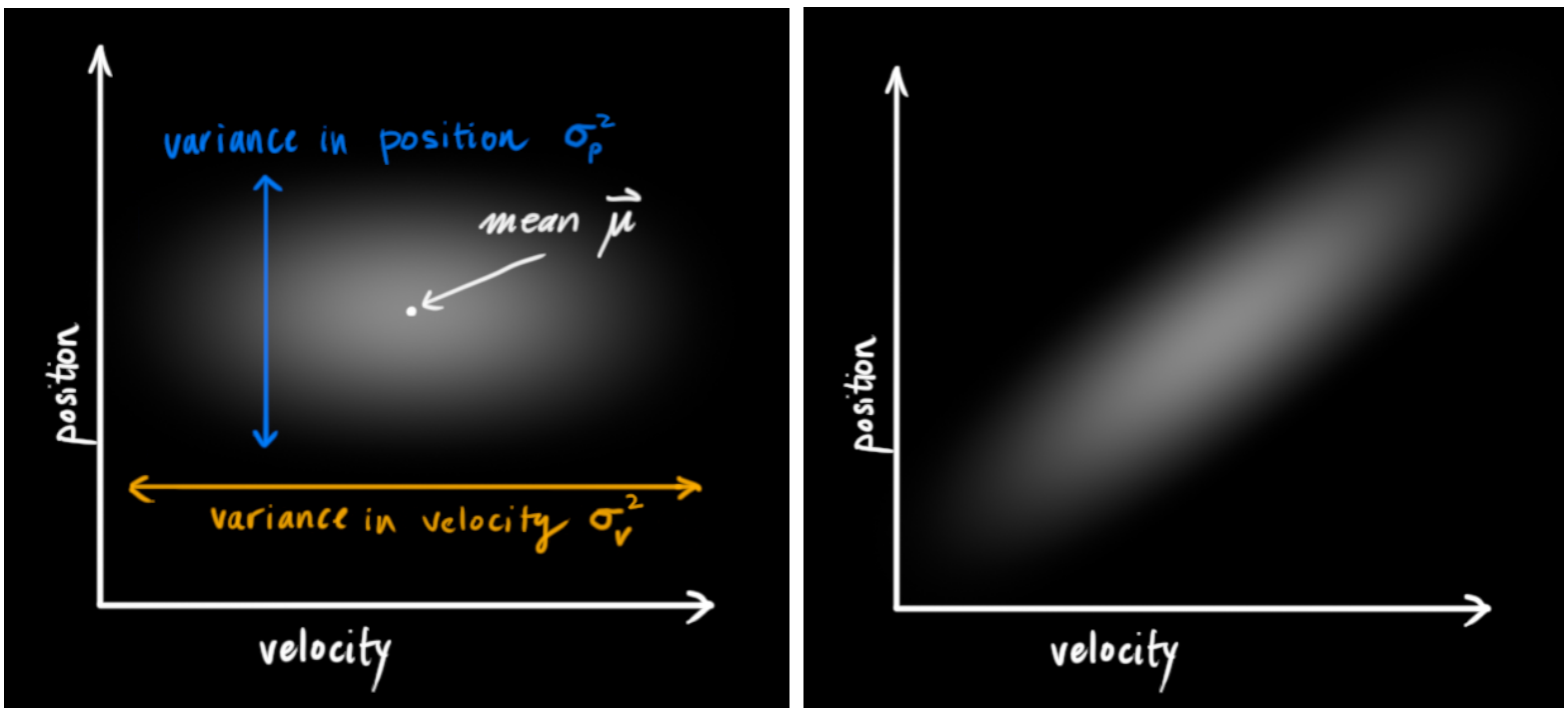
$$x = [p, v]^T$$

- We don't know exactly, precisely the position and velocity of the robot.
- A hole range, some more likely than others.



How Kalman Filter interprets the problem?

- Assumption that both variables are random and Gaussian distributed.
- Each variable has a mean value μ, which is the center of the random distribution (and its most likely state), and a variance σ^2, which is the uncertainty.



(a) uncorrelated (b) correlated

The Covariance Matrix

Correlation between states is very important to keep track of as it gives more information: one measurement tells something about what the others can be.

Kalman filter gets you as much information as possible from uncertain measurements.

This correlation is captured by the COVARIANCE matrix.

Kalman Filter using Matrices

- Best estimate

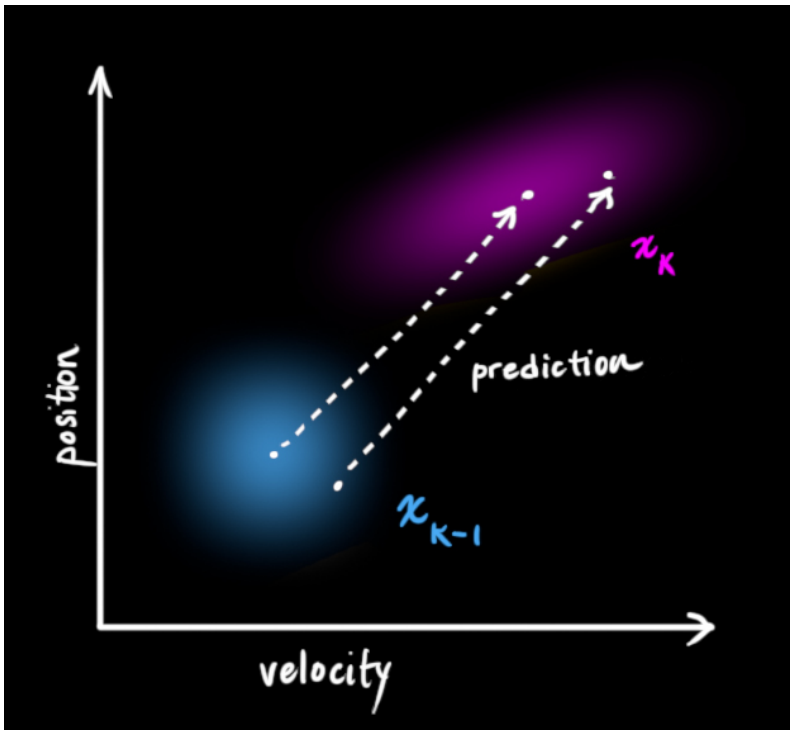
$$\hat{x}_k = [\text{position},\ \text{velocity}]$$

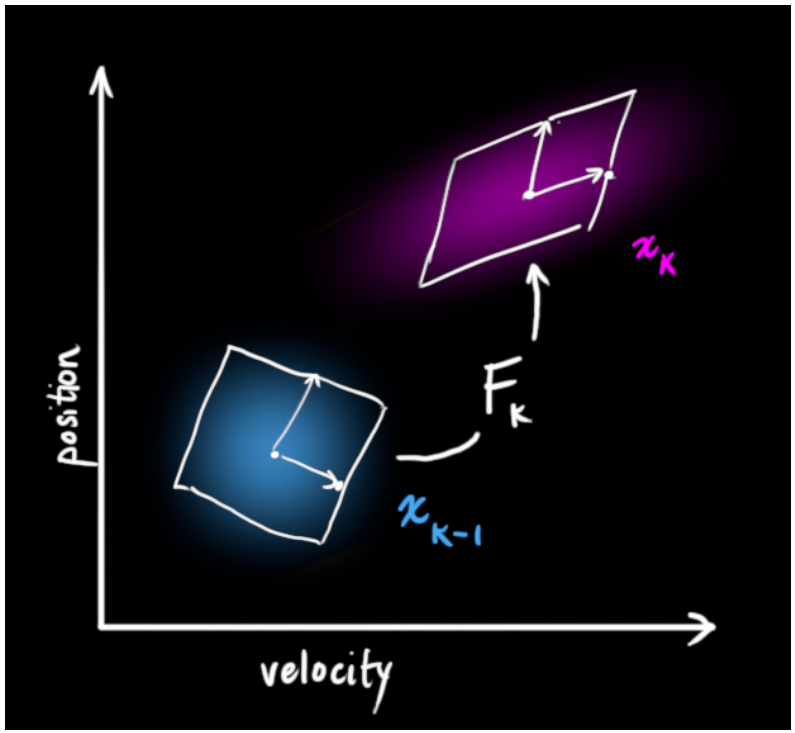
- Its covariance matrix:

$$P_k = \begin{bmatrix} \Sigma_{pp} & \Sigma_{pv} \\ \Sigma_{pv} & \Sigma_{vv} \end{bmatrix}$$

Predicting the next state

F_k is the prediction matrix. Takes every point in our original estimate and moves it to a new predicted location, which is where the system would move if that original estimate was the right one.





Predicting the next state

$$p_k = p_{k-1} + \Delta t v_{k-1}$$

$$v_k = v_{k-1}$$

Leading to

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

If we multiply every point in a distribution by a matrix F, then what happens to its covariance matrix?
Based on the basic properties of covariance matrices:

$$P_k = F_k P_{k-1} F_k^T$$

External influences

There might be some changes that aren't related to the state itself, modelled through the control input u_k.

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

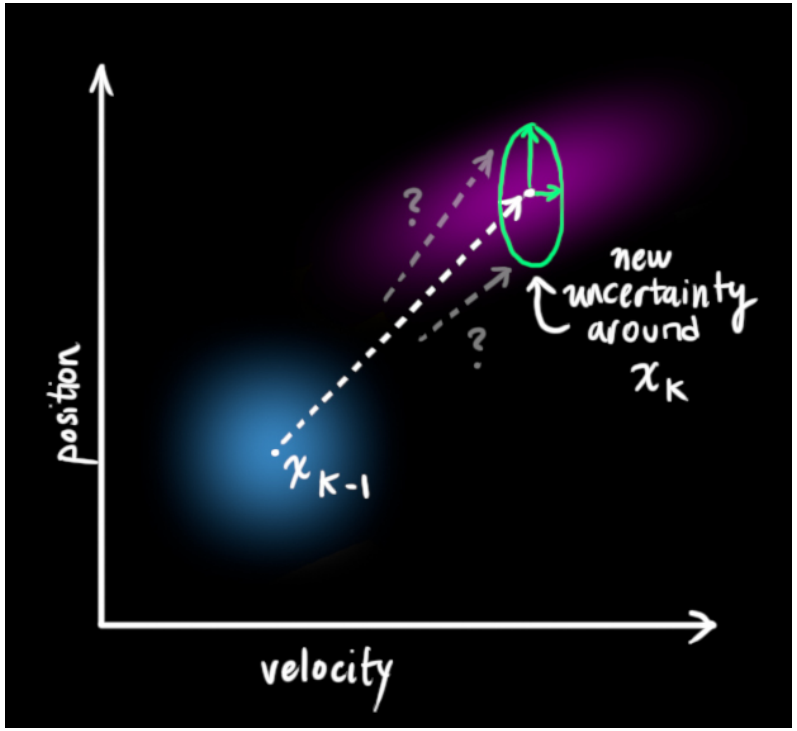
$$p_k = p_{k-1} + \Delta t v_{k-1} + \frac{1}{2} a \Delta t^2$$

$$v_k = v_{k-1} + a \Delta t$$

$$\hat{x}_k = F_k \hat{x}_{k-1} + \begin{bmatrix} \frac{\Delta t^2}{2} \\ \Delta t \end{bmatrix} a = F_k \hat{x}_{k-1} + B_k u_k$$

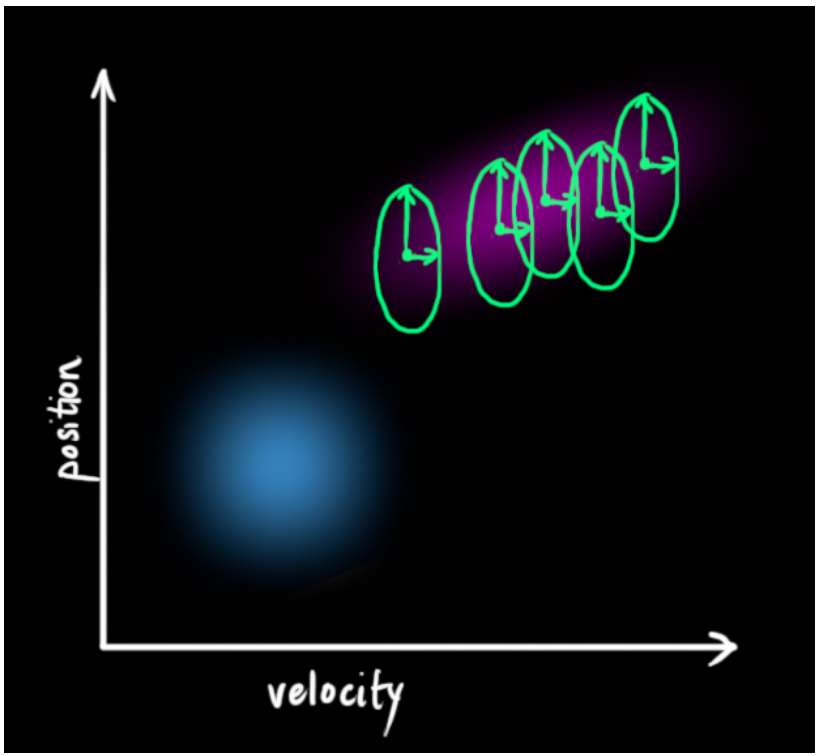
External uncertainty

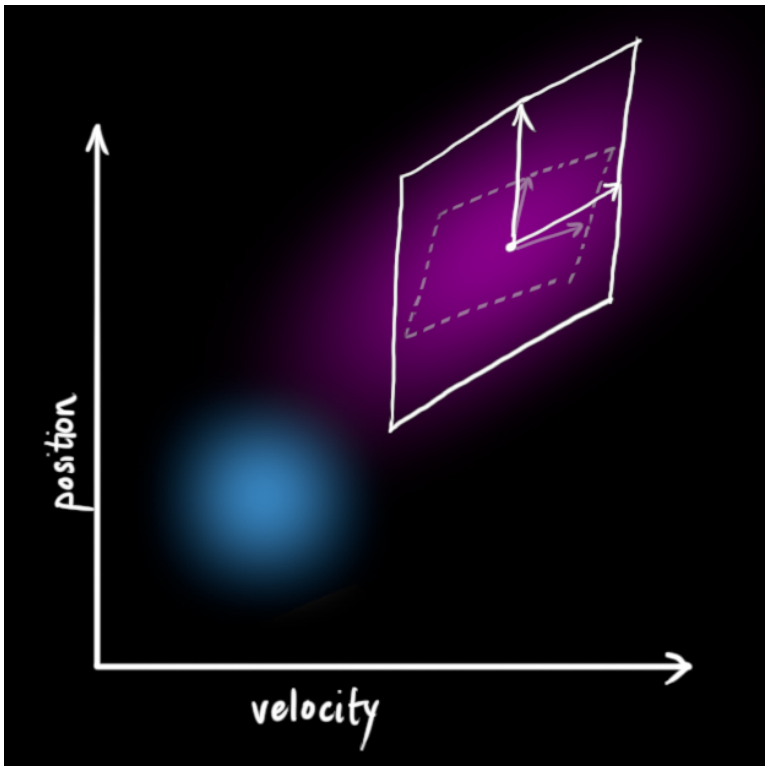
We can model the uncertainty associated with the ”world” (i.e. things we aren’t keeping track of) by adding some new uncertainty after every prediction step:



External uncertainty

We'll say that each point in $\hat{x}_{k-1}$ is moved to somewhere inside a Gaussian blob with covariance Q_k. This produces a new Gaussian blob, with a different covariance (but the same mean).





Prediction step completed

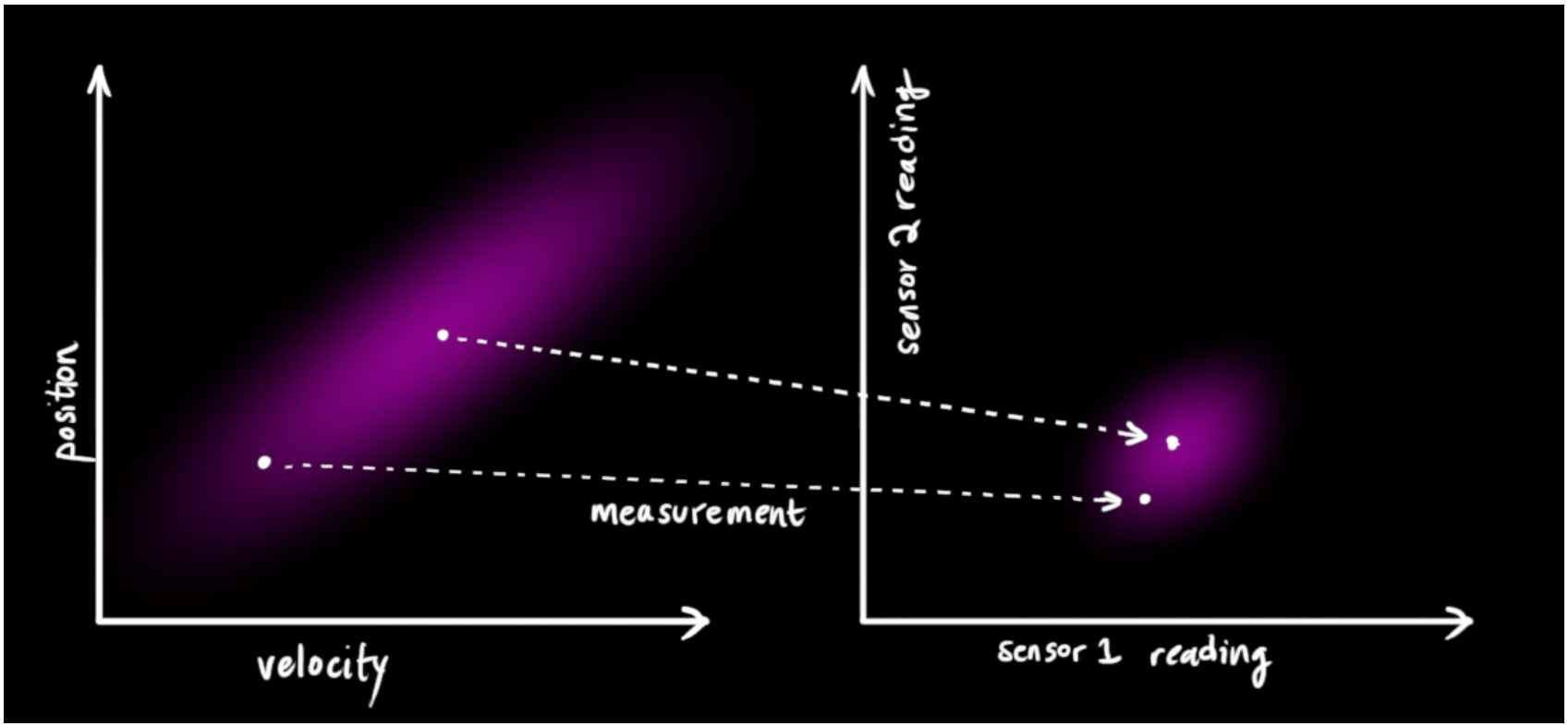
We get the expanded covariance by simply adding Q_k, giving our complete expression for the prediction step:

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

$$P_k = F_k P_{k-1} F_k^T + Q_k$$

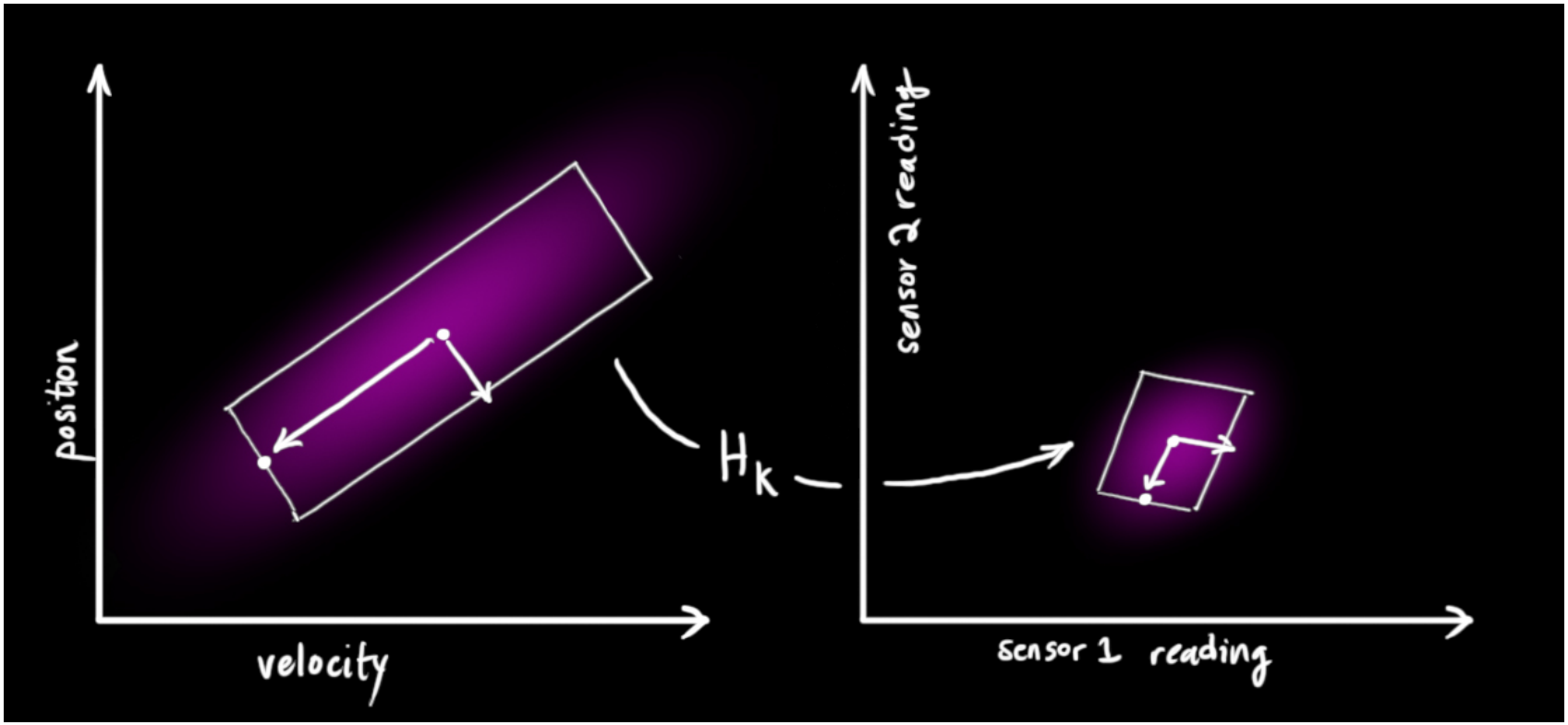
Refining estimates with measurements

We might have several sensors which give us information about the state of our system. Each sensor tells us something indirect about the state, in other words, the sensors operate on a state and produce a set of readings.



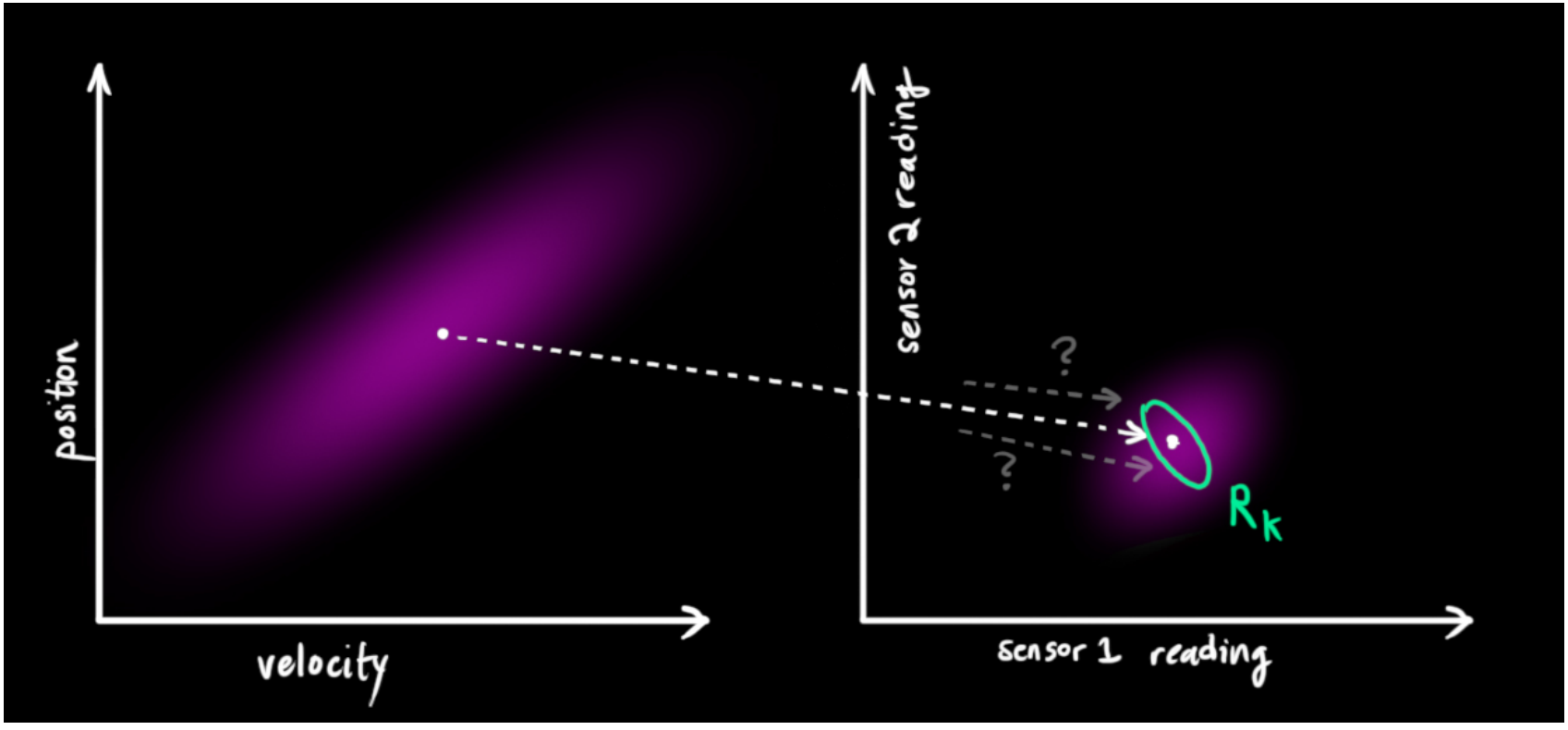
Refining estimates with measurements

We need to map from the state to the sensor measurement space.
We'll model the transformation with a matrix, H_k



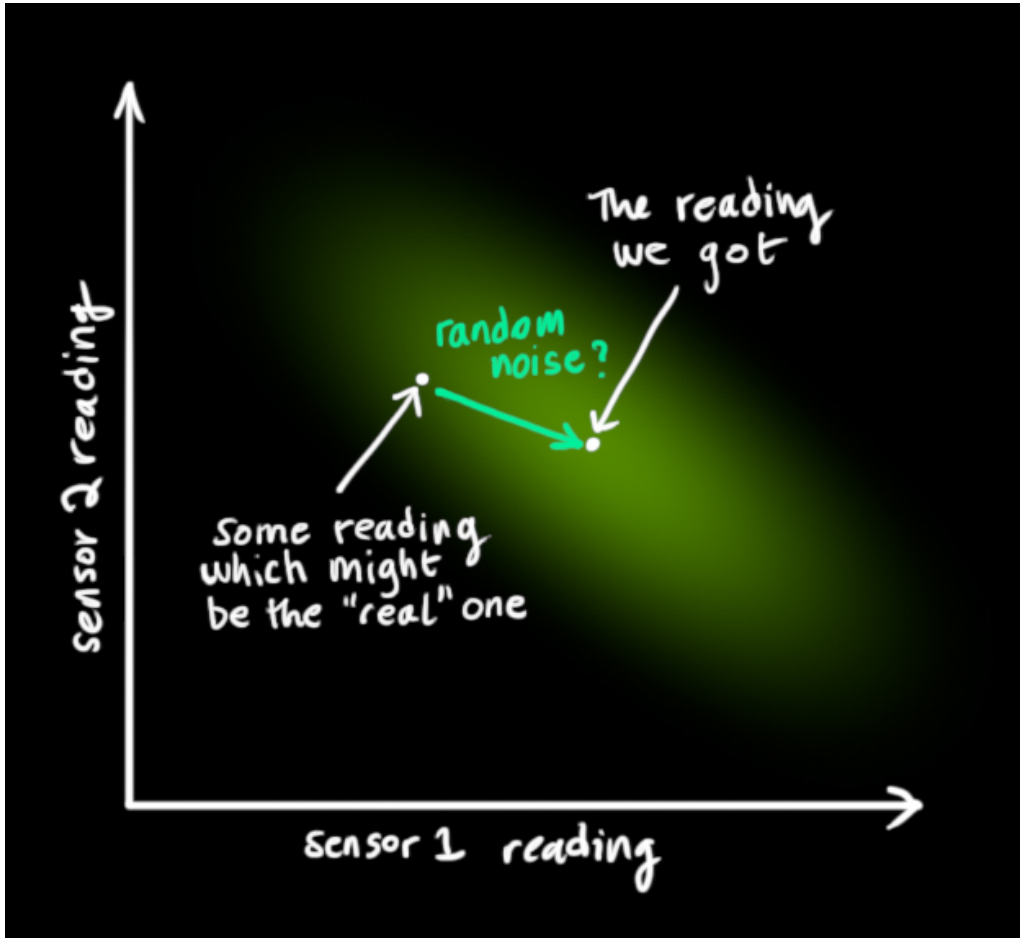
Refining estimates with measurements

Our sensors are at least ”somewhat” unreliable, and every state in our original estimate might result in a range of sensor readings.



Refining estimates with measurements

From each reading we observe, we might guess that our system was in a particular state. But because there is uncertainty, some states are more likely than others to have produced the reading we saw:

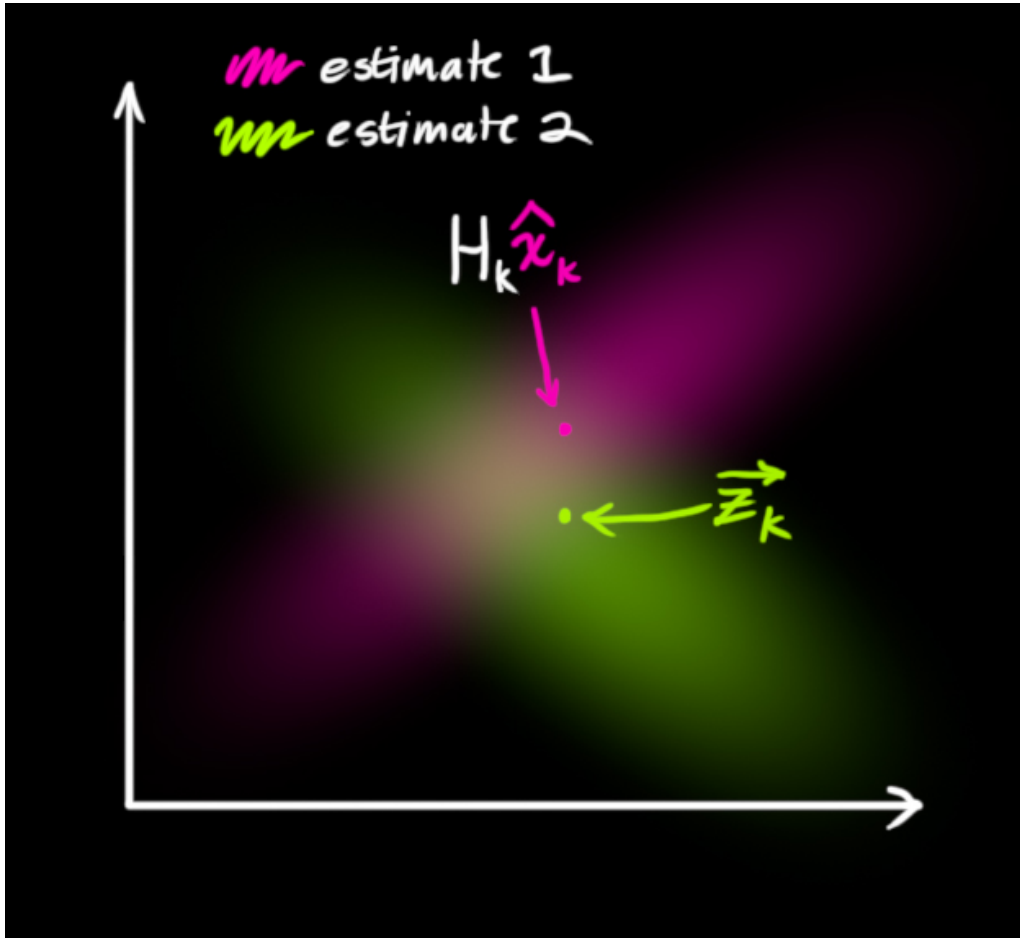


Refining estimates with measurements

We'll call the covariance of this uncertainty (i.e. of the sensor noise) R_k. The distribution has a mean equal to the reading we observed, which we'll call z_k.

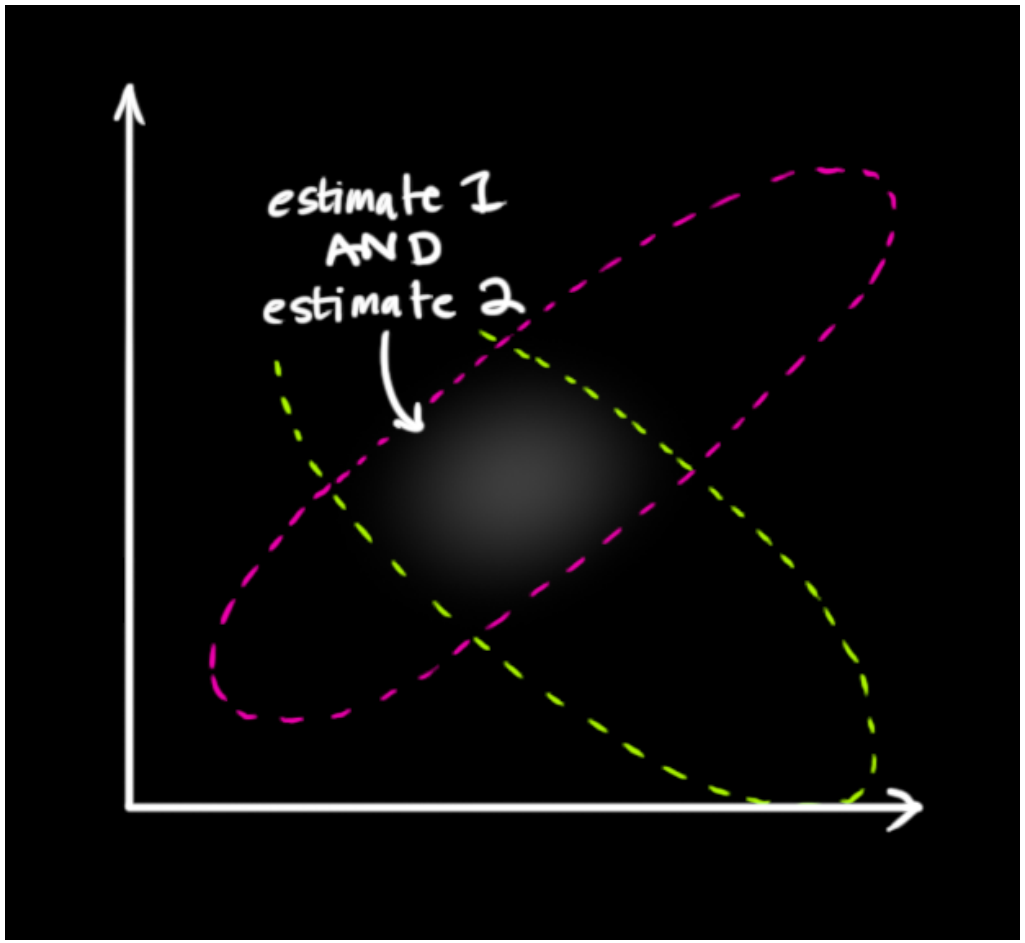
Refining estimates with measurements

Now we have two Gaussian blobs: One surrounding the mean of our transformed prediction, and one surrounding the actual sensor reading we got.



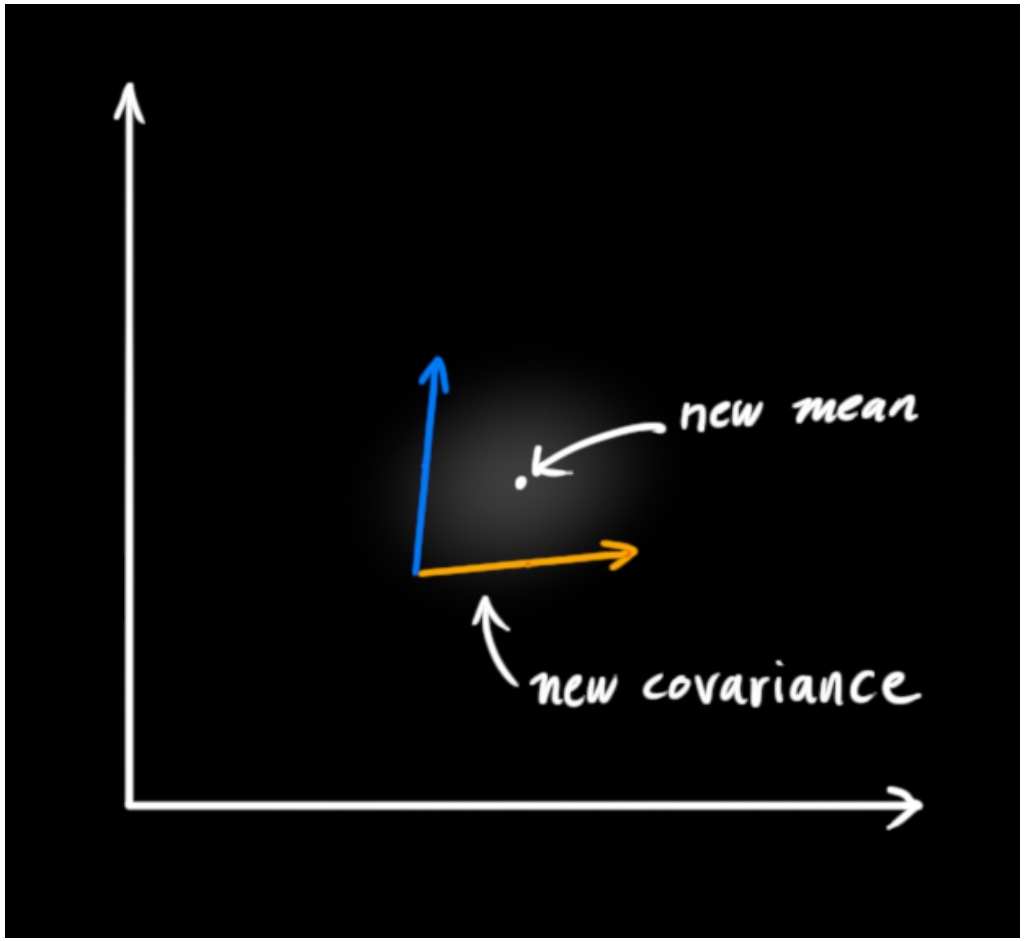
Refining estimates with measurements

We must try to reconcile our guess about the readings we'd see based on the predicted state (pink) with a different guess based on our sensor readings (green) that we actually observed.



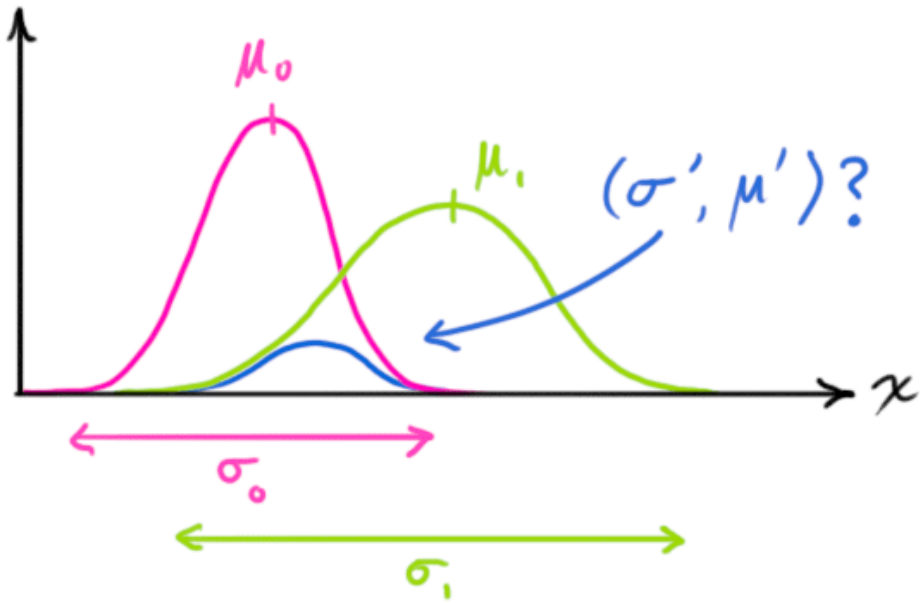
The Kalman Gain

When you multiply two Gaussian blobs with separate means and covariance matrices, you get a new Gaussian blob with its own mean and covariance matrix!



The Kalman Gain

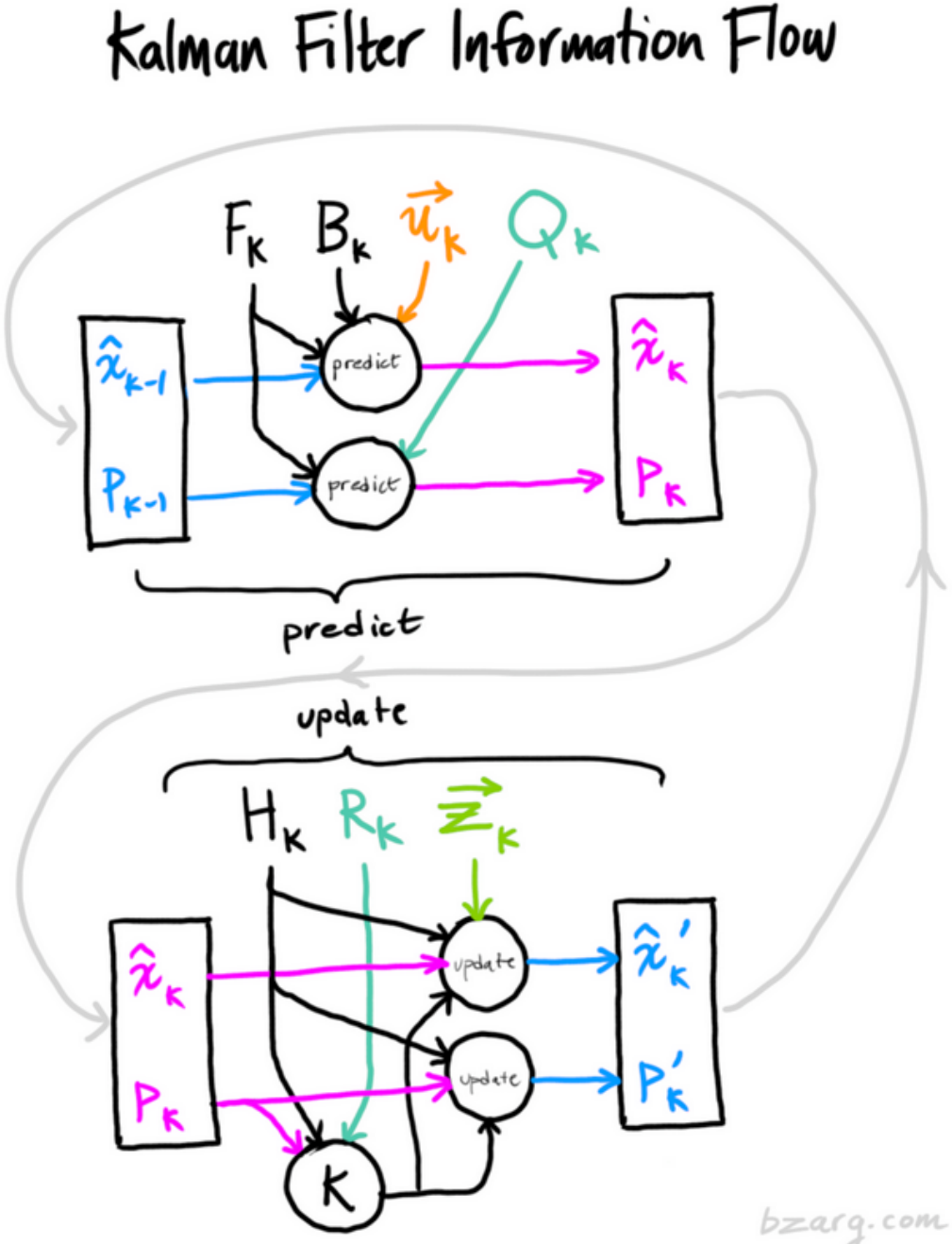
Multiplying the two blobs, we will get the new Gaussian blob and we can define a term named "Kalman gain"



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

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

$$\vec{\mu}' = \vec{\mu_0} + \mathbf{K}(\vec{\mu_1} - \vec{\mu_0})$$
$$\Sigma' = \Sigma_0 - \mathbf{K}\Sigma_0$$


Kalman Filter Information Flow
F_k B_k $\vec{u}_k$ Q_k
$\hat{x}_{k-1}$
P_{k-1}
predict
predict
$\hat{x}_k$
P_k
predict
update
H_k R_k $\vec{z}_k$
$\hat{x}_k$
P_k
update
update
K
$\hat{x}'_k$
P'_k
bzarg.com

Kalman Filter Algorithm: The Prediction

State prediction (predicts where we are going to be):

$$\hat{x}_k^- = F_k x_{k-1} + B_k u_k$$

Covariance Prediction (predicts how much error):

$$P_K^- = F_k P_{k-1} F_k^T + Q_k$$

Kalman Filter Algorithm: The Update

Innovation (compare reality against prediction)

$$\tilde{y}_k = z_k - H_k \hat{x}_k^-$$

Innovation Covariance (compare real error against prediction)

$$S_k = H_k P_K^- H_k^T + R_k$$

Kalman Gain (moderate the prediction)

$$K_k = P_K^- H_k^T S_k^{-1}$$

State Update (new estimate of where we are)

$$\hat{x}_k = \hat{x}_k^- + K_k \tilde{y}_k$$

Covariance Update (new estimate of error)

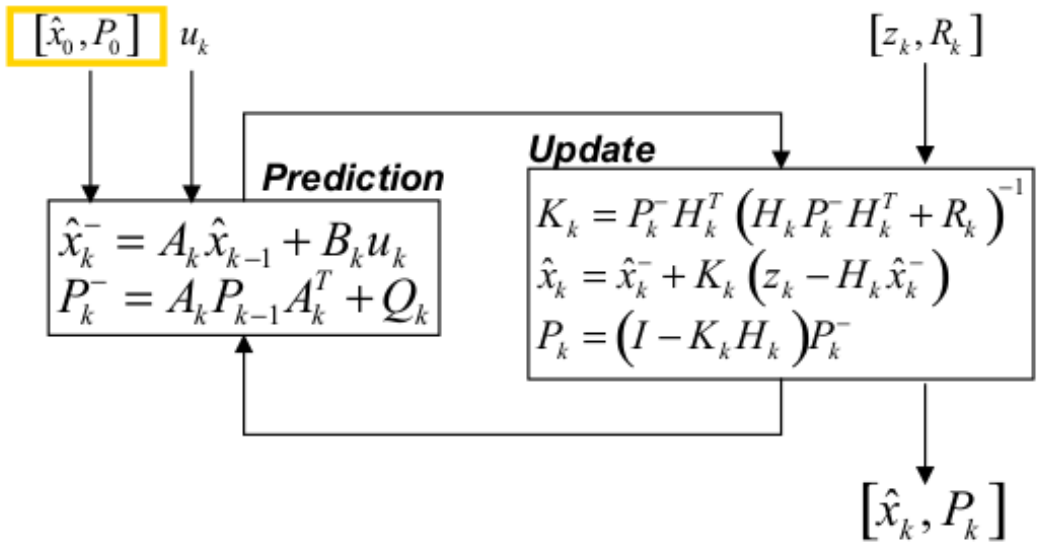
$$P_k = (I - K_k H_k) P_k^-$$

Monobot with constant velocity model and position fixes

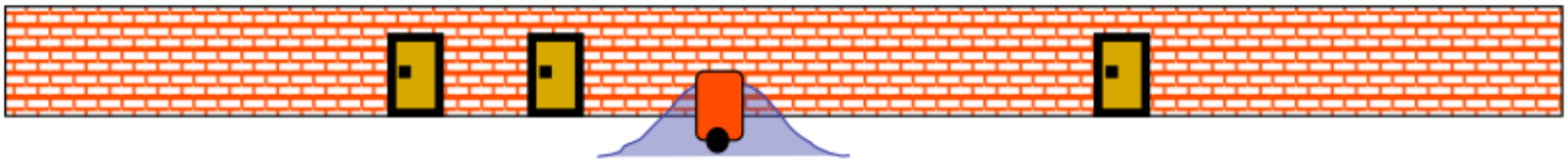
1. Initial position is known with some uncertainty

$$\mathbf{x_k} = \begin{bmatrix} x_k \\ \dot{x}_k \end{bmatrix}$$

$$z_k = x_{k,measured}$$



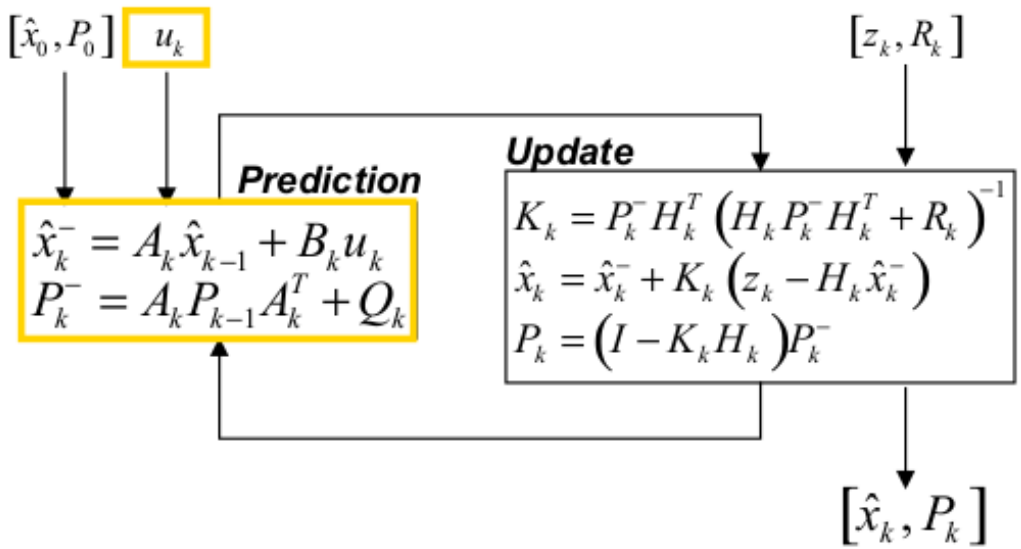
Monobot with constant velocity model and position fixes



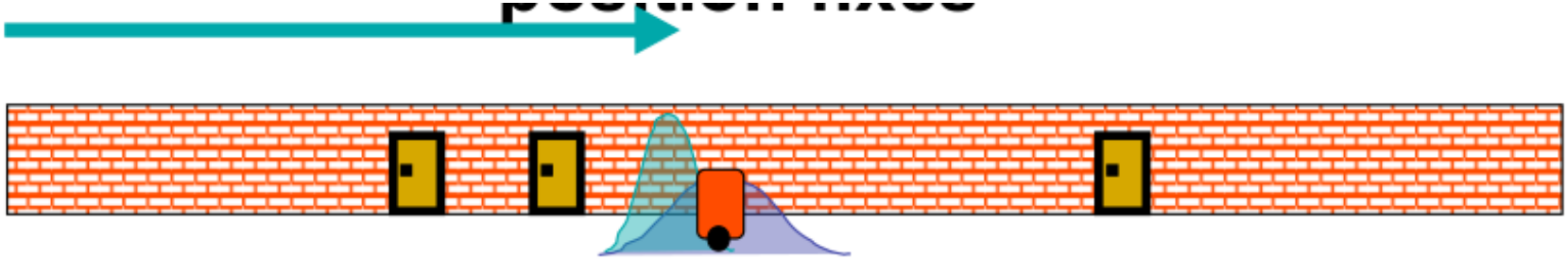
1. Initial position is known with some uncertainty
2. The robot Moves and the model is used to predict the new position

$$\mathbf{x_k} = \begin{bmatrix} x_k \\ \dot{x}_k \end{bmatrix}$$

$$z_k = x_{k,measured}$$



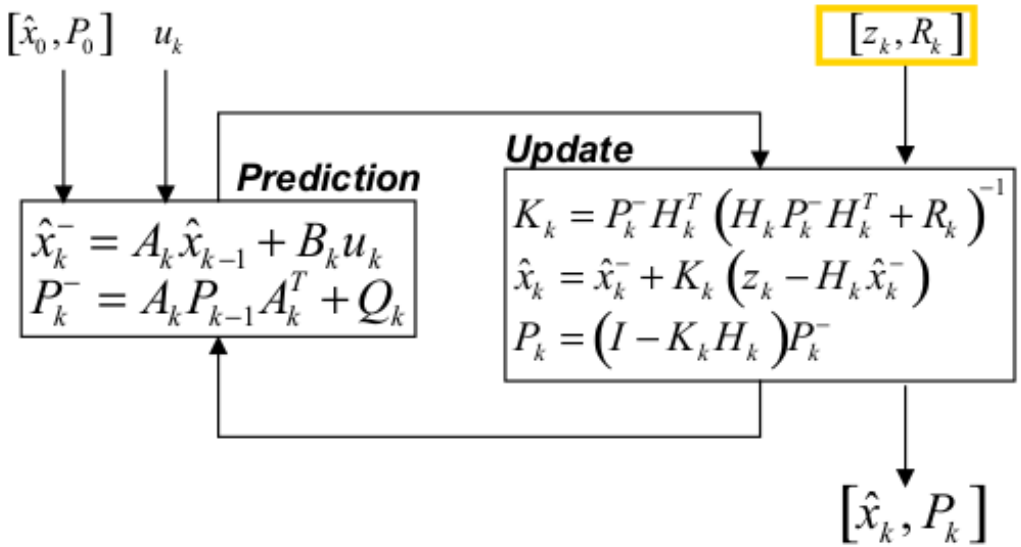
Monobot with constant velocity model and position fixes



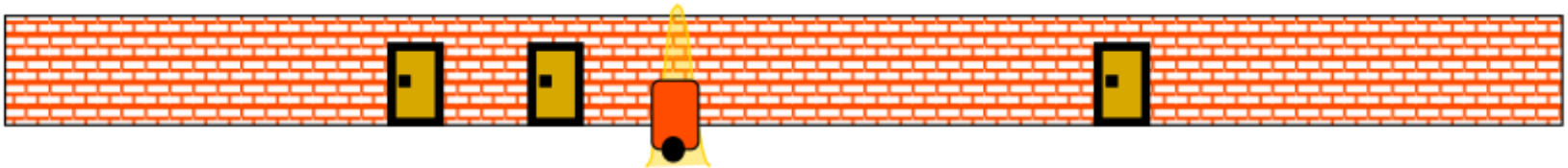
1. Initial position is known with some uncertainty
2. The robot Moves and the model is used to predict the new position
3. The robot position is sensed

$$\mathbf{x_k} = \begin{bmatrix} x_k \\ \dot{x}_k \end{bmatrix}$$

$$z_k = x_{k,measured}$$



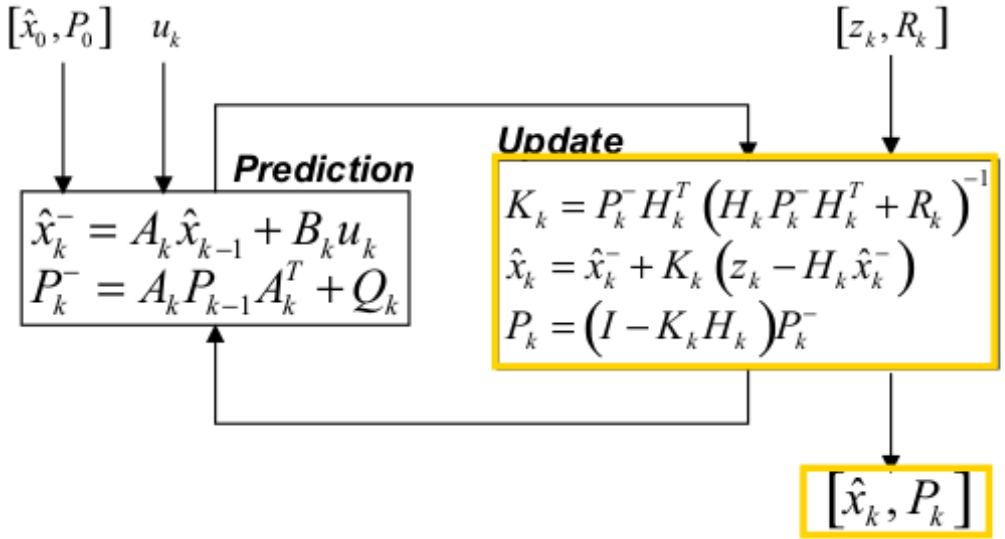
Monobot with constant velocity model and position fixes



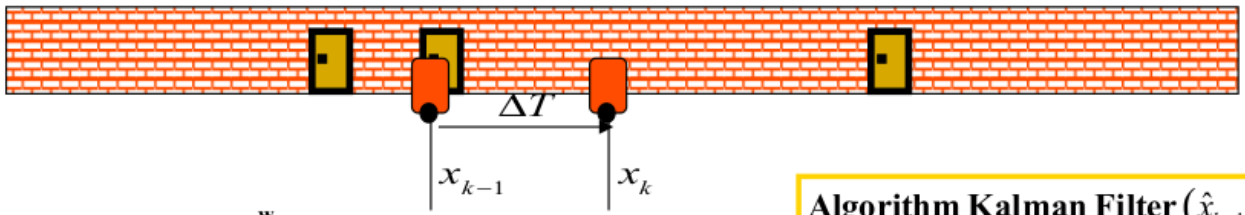
1. Initial position is known with some uncertainty
2. The robot Moves and the model is used to predict the new position
3. The robot position is sensed
4. The measured position is used to update the robot position

$$\mathbf{x_k} = \begin{bmatrix} x_k \\ \dot{x}_k \end{bmatrix}$$

$$z_k = x_{k,measured}$$



Monobot with constant velocity model and position fixes



$$\underbrace{\begin{bmatrix} x_k \\ \dot{x}_k \end{bmatrix}}_{\mathbf{x_k}} = \underbrace{\begin{bmatrix} 1 & \Delta T \\ 0 & 1 \end{bmatrix}}_{\mathbf{A_k}} \underbrace{\begin{bmatrix} x_{k-1} \\ \dot{x}_{k-1} \end{bmatrix}}_{\mathbf{x_{k-1}}} + \overbrace{\begin{bmatrix} \frac{\Delta T^2}{2} \\ \Delta T \end{bmatrix} w_k}^{\mathbf{w_k}} \qquad \mathbf{B_k = u_k = 0}$$

$$z_k = \underbrace{\begin{bmatrix} 1 & 0 \end{bmatrix}}_{\mathbf{H_k}} \underbrace{\begin{bmatrix} x_k \\ \dot{x}_k \end{bmatrix}}_{\mathbf{x_k}} + \underbrace{v_k}_{\mathbf{v_k}}$$

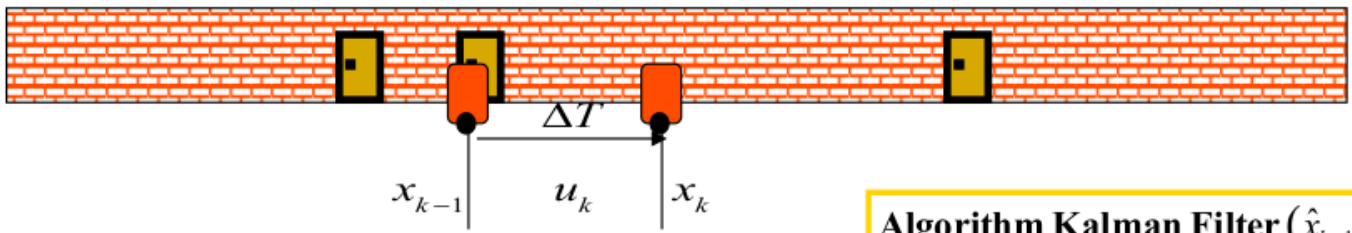
Algorithm Kalman Filter $(\hat{x}_{k-1}, P_{k-1}, u_k, z_k)$

$$\hat{x}_k^- = A_k \hat{x}_{k-1} + B_k u_k$$
$$P_k^- = A_k P_{k-1} A_k^T + Q_k$$
$$K_k = P_k^- H_k^T \left(H_k P_k^- H_k^T + R_k \right)^{-1}$$
$$\hat{x}_k = \hat{x}_k^- + K_k \left(z_k - H_k \hat{x}_k^- \right)$$
$$P_k = \left(I - K_k H_k \right) P_k^-$$
$$return \left(\hat{x}_k, P_k \right)$$

$$Q_k = \mathrm{Cov}\left(\mathbf{w_k}\right) = E\left[\mathbf{w_k}\,\mathbf{w_k^T}\right] = \begin{bmatrix} \frac{\Delta T^2}{2} \\ \Delta T \end{bmatrix} w_k w_k^T \begin{bmatrix} \frac{\Delta T^2}{2} & \Delta T \end{bmatrix} = \begin{bmatrix} \frac{\Delta T^4}{4} & \frac{\Delta T^3}{2} \\ \frac{\Delta T^3}{2} & \Delta T^2 \end{bmatrix} \sigma_w^2$$

$$R_k = Cov\left(\mathbf{v_k}\right) = E\left[v_k v_k^T\right] = \sigma_v^2$$

Monobot with constant velocity model and position fixes



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

$$z_k = x_k + v_k$$

Sonar noise

Odometry measurement

Odometry noise

Algorithm Kalman Filter $(\hat{x}_{k-1}, P_{k-1}, u_k, z_k)$

$\hat{x}_k^- = A_k \hat{x}_{k-1} + B_k u_k$

$P_k^- = A_k P_{k-1} A_k^T + Q_k$

$K_k = P_k^- H_k^T \left(H_k P_k^- H_k^T + R_k\right)^{-1}$

$\hat{x}_k = \hat{x}_k^- + K_k \left(z_k - H_k \hat{x}_k^-\right)$

$P_k = \left(I - K_k H_k\right) P_k^-$

$return\left(\hat{x}_k, P_k\right)$

Kalman Filter: Summary

- Highly efficient: Polynomial in measurement dimensionality k and state dimensionality n: $O(k^{2.376} + n^2)$
- Optimal for linear Gaussian systems!
- Most robotics systems are nonlinear

Questions?