ENGR-E 511; ENGR-E 399

Machine Learning for Signal Processing

Module 13:

Kalman Filtering

Minje Kim

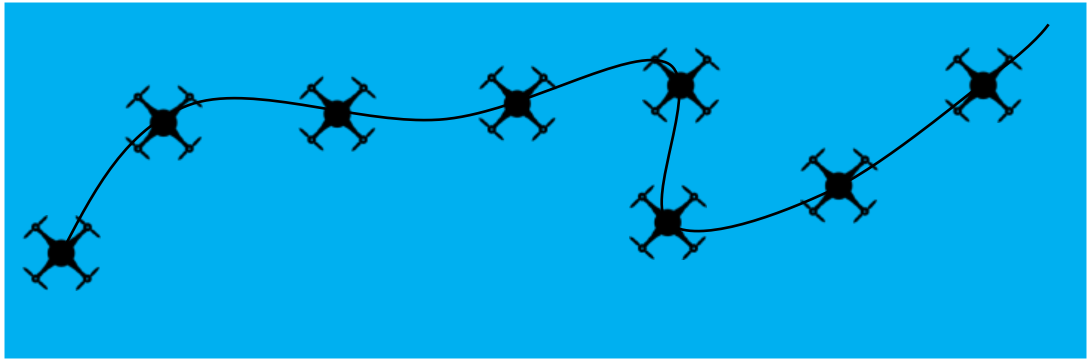
Department of Intelligent Systems Engineering

Email: minje@indiana.edu

Website: http://minjekim.com

Research Group: http://saige.sice.indiana.edu

Meeting Request: http://doodle.com/minje



INDIANA UNIVERSITY

SCHOOL OF INFORMATICS, COMPUTING, AND ENGINEERING

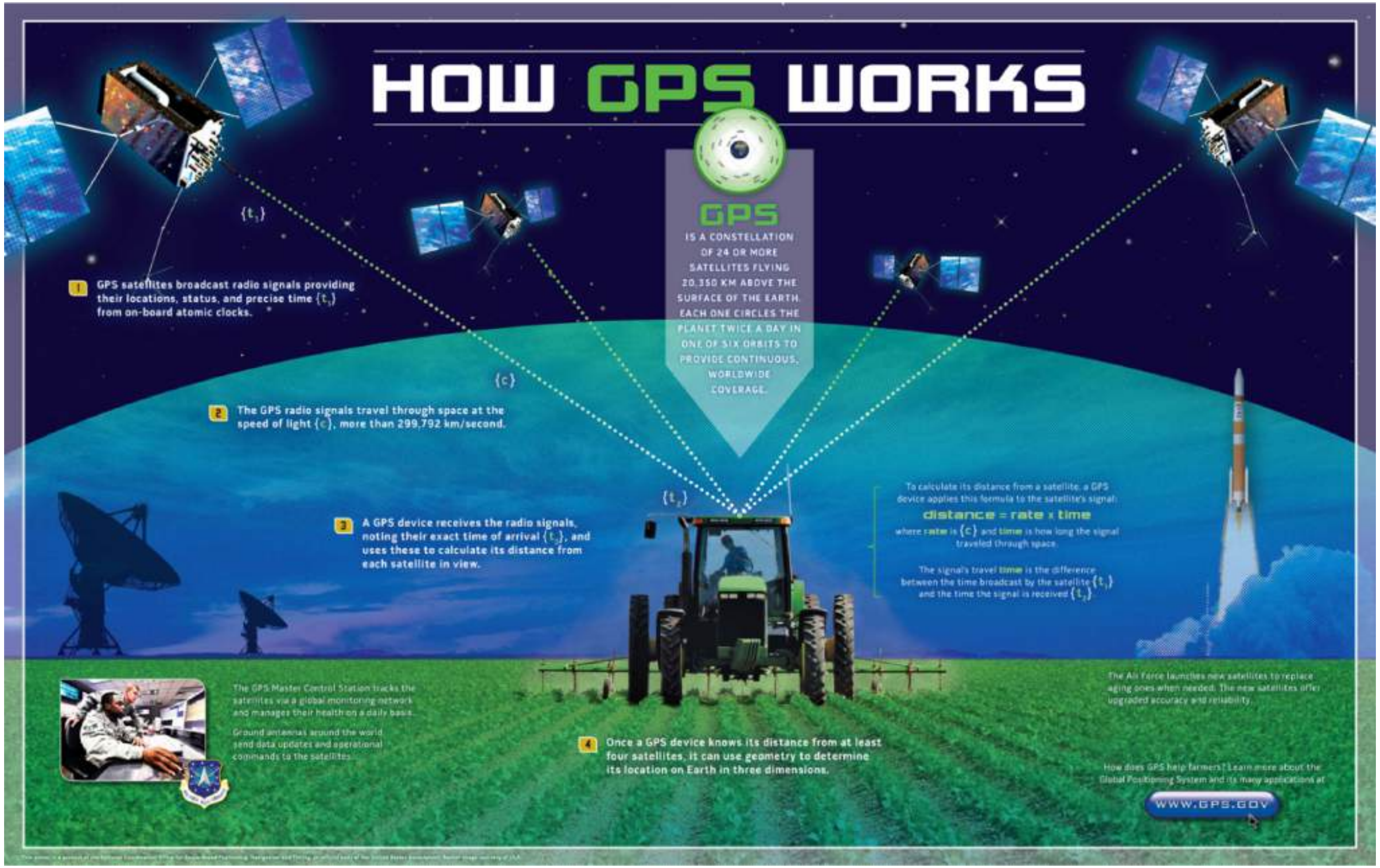
Online Transportation Network

- Track your drivers

- You are a CTO of a start-up company
  - Good luck
- Your company is specialized in online transportation network services
  - Good luck with your competition
- You distributed an app to your drivers (as well as to your customers)
  - Using that, you want to track their location
- What you want to do:
  - Track down the drivers using their GPS information
  - Match your customer and the closest driver
- But,
  - The GPS information is noisy

Online Transportation Network

- Global Positioning System (GPS)

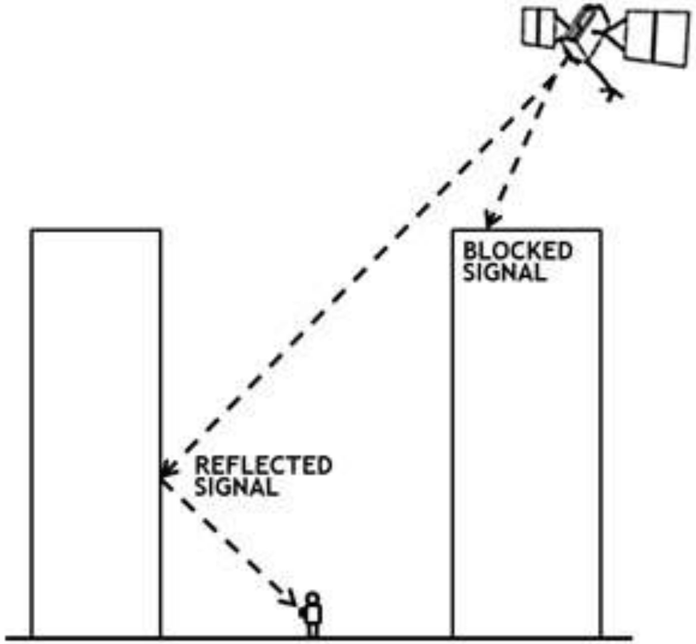


http://www.gps.gov/multimedia/poster/poster-web.pdf

Online Transportation Network

- Denoising a GPS signal

- "How accurate is GPS?
  - For example, GPS-enabled smartphones are typically accurate to within a 4.9 m (16 ft.) radius under open sky
  - However, their accuracy worsens near buildings, bridges, and trees"
- You need to **filter** out this measurement **noise**
  - How?
    - Using some *a priori* knowledge about the location
  - What kind of a priori knowledge?
    - Previously reported location
- So, smoothing using the previous time frame?
  - Sounds like what?
  - HMM?



http://www.gps.gov/systems/gps/performance/accuracy/

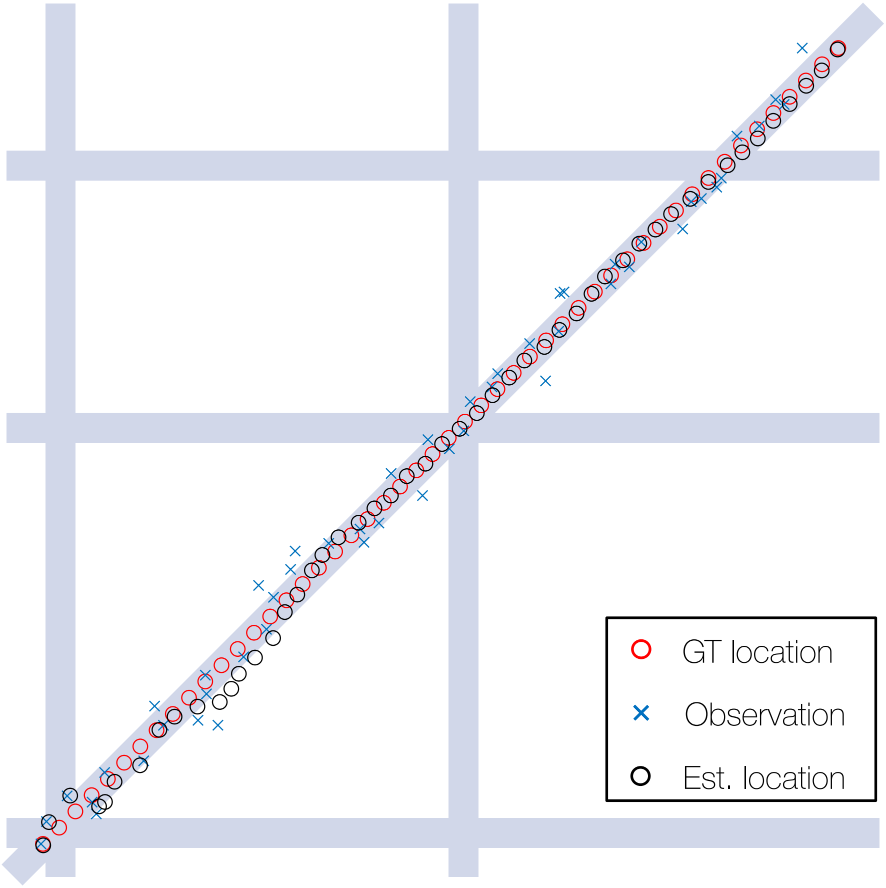
Online Transportation Network
- Denoising a GPS signal using HMM?

- Why does it not work?
  - HMM assumes discrete hidden states
  - Are your hidden states discrete?
- You observe a continuous real-world signal
  - There's no guarantee that your hidden states are discrete
- In our case
  - I feel that there are four hidden states:
    - Ground-truth longitude
    - Ground-truth latitude
    - Ground-truth speed along longitude
    - Ground-truth speed along latitude
- Our goal
  - Infer the states at every given time stamp from the noisy observations

Online Transportation Network

- The Kalman filter

- We're chasing a car
  - With a steady speed
- The only information available:
  - Noisy location info
  - Doesn't-make-sense-off-road locations
- Our goal is to denoise this and recover the smooth movement
- I'm going to use a technique called **Kalman Filtering**



Kalman Filter

- Some history

- “In 1959, when NASA first tasked its Centers to explore the problems of navigating to the Moon, Schmidt saw the potential for making major theoretical extensions to the **Kalman linear filter.** The result was a state-estimation algorithm (in simple terms a procedure for solving a problem) called the **Kalman-Schmidt filter.** By early 1961, Schmidt and John White had demonstrated that a **computer built with this filter, combined with optical measurements of the stars and data about the motion of the spacecraft, could provide the accuracy needed for a successful insertion into orbit around the Moon.** … **The Kalman-Schmidt filter was embedded in the Apollo navigation computer and ultimately into all air navigation systems**, and laid the foundation for Ames' future leadership in flight and air traffic research.”

https://www.nasa.gov/centers/ames/news/releases/2004/moon/apollo_ames_atmos.html

Kalman Filter without Control

- Steady speed

- We're not doing rocket science
  - So, longitude and latitude are enough to track our driver
- Kalman filtering blends the noisy observation and the a priori knowledge about the location
- First, the transformation from the hidden state to the observation is simple

Observed longitude

Longitude

$$\boldsymbol{z}_1(t) = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_1(t) \\ \boldsymbol{x}_2(t) \end{bmatrix} + \boldsymbol{v}_1(t)$$

Measurement noise

Speed along longitude

Observed latitude

Latitude

$$\boldsymbol{z}_2(t) = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_3(t) \\ \boldsymbol{x}_4(t) \end{bmatrix} + \boldsymbol{v}_2(t)$$

Measurement noise

Speed along latitude

Kalman Filter without Control

- Steady speed (no control)

- Since I'm a big fan of linear algebra..

$$\begin{bmatrix} \boldsymbol{z}_1(t) \\ \boldsymbol{z}_2(t) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_1(t) \\ \boldsymbol{x}_2(t) \\ \boldsymbol{x}_3(t) \\ \boldsymbol{x}_4(t) \end{bmatrix} + \begin{bmatrix} \boldsymbol{v}_1(t) \\ \boldsymbol{v}_2(t) \end{bmatrix}$$

  - Or, $\boldsymbol{z}_t = \boldsymbol{H}\boldsymbol{x}_t + \boldsymbol{v}_t$

- It might be reasonable to assume that the measurement noise is following a Gaussian distribution $\mathcal{N}(\boldsymbol{v}_t; \boldsymbol{0}, \boldsymbol{R})$

$$\boldsymbol{\mu_v} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \qquad \boldsymbol{R} = \begin{bmatrix} \sigma_{\boldsymbol{v}}^2 & 0 \\ 0 & \sigma_{\boldsymbol{v}}^2 \end{bmatrix}$$

- This procedure corresponds to emission probabilities in HMM

Kalman Filter without Control

- Steady speed (no control)

- Then, what about the transition rule?
  - Here you go $$\begin{bmatrix} \boldsymbol{x}_1(t+1) \\ \boldsymbol{x}_2(t+1) \\ \boldsymbol{x}_3(t+1) \\ \boldsymbol{x}_4(t+1) \end{bmatrix} = \begin{bmatrix} 1 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_1(t) \\ \boldsymbol{x}_2(t) \\ \boldsymbol{x}_3(t) \\ \boldsymbol{x}_4(t) \end{bmatrix} + \begin{bmatrix} \boldsymbol{w}_1(t) \\ \boldsymbol{w}_2(t) \\ \boldsymbol{w}_3(t) \\ \boldsymbol{w}_4(t) \end{bmatrix}$$
  - In English
    - The longitude at time $t+1$ is [the longitude at time t] plus [the velocity at time t times the time difference]
    - The latitude at time $t+1$ is [the latitude at time t] plus [the velocity at time t times the time difference]
    - And the velocity doesn't change over time
    - Plus **process noise** $$\boldsymbol{w}_t \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{Q}), \quad \boldsymbol{Q} = \begin{bmatrix} \sigma^2_{\boldsymbol{w}} & 0 \\ 0 & \sigma^2_{\boldsymbol{w}} \end{bmatrix}$$
      - How noisy the prediction would be
      - e.g. The driver tries to keep the speed, but there are up and down
- In linear algebra $\boldsymbol{x}_{t+1} = \boldsymbol{F}\boldsymbol{x}_t + \boldsymbol{w}_t$

Kalman Filter without Control

- Prediction

$$\begin{bmatrix} x_1(t+1) \\ x_2(t+1) \\ x_3(t+1) \\ x_4(t+1) \end{bmatrix} = \begin{bmatrix} 1 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \\ x_4(t) \end{bmatrix} + \begin{bmatrix} w_1(t) \\ w_2(t) \\ w_3(t) \\ w_4(t) \end{bmatrix}$$

F Q

- The first time frame

$$\hat{x}_1 = \begin{bmatrix} z_1(1) \\ 0.002\frac{\text{mile}}{\text{sec}} \\ z_2(2) \\ 0.002\frac{\text{mile}}{\text{sec}} \end{bmatrix}$$

- The second time frame

$$\hat{x}_2 = F\hat{x}_1 + w_1 \qquad \hat{x}_2 \sim \mathcal{N}(F\hat{x}_1, P_1) \qquad P_1 = Q$$

- The third time frame

$$\hat{x}_3 = F\hat{x}_2 + w_2 \qquad \hat{x}_3 \sim \mathcal{N}(F\hat{x}_2, P_2) \qquad P_2 = FP_1F^\top + Q$$

- The (*n*+*1*)-th time frame

$$\hat{x}_{t+1} = F\hat{x}_t + w_t \qquad \hat{x}_{t+1} \sim \mathcal{N}(F\hat{x}_t, P_t) \qquad P_t = FP_{t-1}F^\top + Q$$

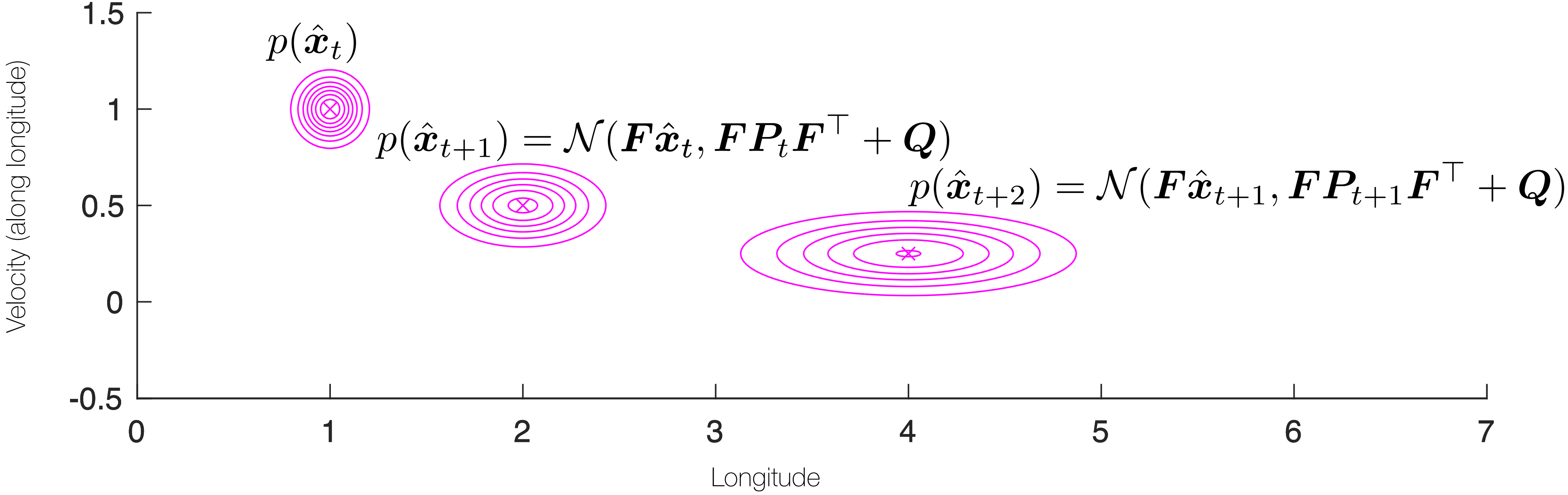
- Covariance matrix after a linear transform (zero mean)

$$E\left[Fx(Fx)^\top\right] = E\left[Fxx^\top F^\top\right] = FE\left[xx^\top\right]F^\top = FPF^\top$$

Kalman Filter without Control

- Gaussian after linear transformation

- An evolution example (toy data, different from our GPS data) $\boldsymbol{F} = \begin{bmatrix} 2 & 0 \\ 0 & .5 \end{bmatrix}$



Kalman Filter without Control

- Fusion of prediction and measurement

$$\begin{bmatrix} \boldsymbol{z}_1(t) \\ \boldsymbol{z}_2(t) \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{\boldsymbol{H}} \begin{bmatrix} \boldsymbol{x}_1(t) \\ \boldsymbol{x}_2(t) \\ \boldsymbol{x}_3(t) \\ \boldsymbol{x}_4(t) \end{bmatrix} + \underbrace{\begin{bmatrix} \boldsymbol{v}_1(t) \\ \boldsymbol{v}_2(t) \end{bmatrix}}_{\boldsymbol{R}}$$

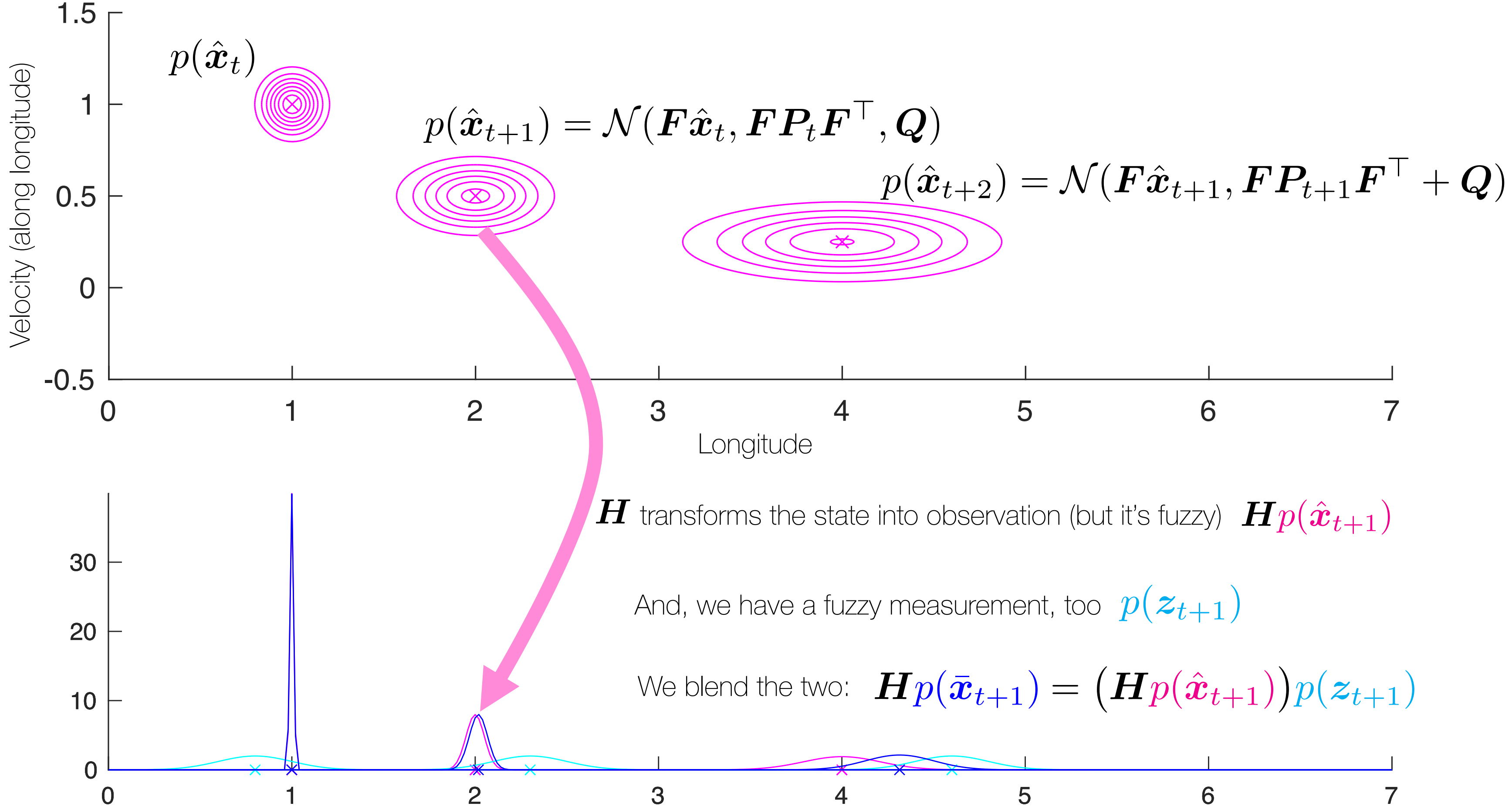
$$\begin{bmatrix} \boldsymbol{x}_1(t+1) \\ \boldsymbol{x}_2(t+1) \\ \boldsymbol{x}_3(t+1) \\ \boldsymbol{x}_4(t+1) \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\boldsymbol{F}} \begin{bmatrix} \boldsymbol{x}_1(t) \\ \boldsymbol{x}_2(t) \\ \boldsymbol{x}_3(t) \\ \boldsymbol{x}_4(t) \end{bmatrix} + \underbrace{\begin{bmatrix} \boldsymbol{w}_1(t) \\ \boldsymbol{w}_2(t) \\ \boldsymbol{w}_3(t) \\ \boldsymbol{w}_4(t) \end{bmatrix}}_{\boldsymbol{Q}}$$

- So far we saw predictions
  - Whose distributions can be fuzzier as time goes by
- But, we can make use of measurements, too
- If both the measurement and the prediction are correct
  - $\boldsymbol{z}_t = \boldsymbol{H}\hat{\boldsymbol{x}}_t$
- If $\boldsymbol{z}_t$ and $\boldsymbol{H}\hat{\boldsymbol{x}}_t$ are different from each other
  - We don't know which one is more correct
  - We need to think what the actual state would be
- In Kalman filtering we blend the measurement and prediction
  - For that we need to denote the consolidated prediction separately $\bar{\boldsymbol{x}}_t$

$$p(\boldsymbol{H}\bar{\boldsymbol{x}}_t) \approx p(\boldsymbol{z}_t) \cdot p(\boldsymbol{H}\hat{\boldsymbol{x}}_t) = \mathcal{N}(\boldsymbol{z}_t, \boldsymbol{R}) \cdot \mathcal{N}(\boldsymbol{H}\hat{\boldsymbol{x}}_t, \boldsymbol{HQH}^\top)$$
$$= \mathcal{N}(?, ?)$$

Kalman Filter without Control

- Fusion of prediction and measurement



Kalman Filter without Control

- Fusion of prediction and measurement

- Eventually, it's a product of two Gaussians

$$\boldsymbol{H}\bar{\boldsymbol{x}} \sim \mathcal{N}(\boldsymbol{H}\hat{\boldsymbol{x}}_t, \boldsymbol{H}\boldsymbol{P}_t\boldsymbol{H}^\top)\mathcal{N}(\boldsymbol{z}_t, \boldsymbol{R})$$

- Product of two Gaussians?
  - A simple 1-d case $\mathcal{N}(\bar{\mu}, \bar{\sigma}) = \mathcal{N}(\mu_1, \sigma_1)\mathcal{N}(\mu_2, \sigma_2)$

$$\bar{\mu} = \mu_1 + \frac{\sigma_1^2(\mu_2 - \mu_1)}{\sigma_1^2 + \sigma_2^2} \qquad \bar{\sigma}^2 = \sigma_1^2 - \frac{\sigma_1^4}{\sigma_1^2 + \sigma_2^2}$$

  - For multidimensional Gaussians in general

$$\boldsymbol{K} = \boldsymbol{\Sigma}_1(\boldsymbol{\Sigma}_1 + \boldsymbol{\Sigma}_2)^{-1} \qquad \bar{\boldsymbol{\mu}} = \boldsymbol{\mu}_1 + \boldsymbol{K}(\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1) \qquad \bar{\boldsymbol{\Sigma}} = \boldsymbol{\Sigma}_1 - \boldsymbol{K}\boldsymbol{\Sigma}_1$$

- Therefore, in our case

$$\boldsymbol{K} = \boldsymbol{H}\boldsymbol{P}_t\boldsymbol{H}^\top(\boldsymbol{H}\boldsymbol{P}_t\boldsymbol{H}^\top + \boldsymbol{R})^{-1}$$

$$\boldsymbol{H}\bar{\boldsymbol{x}}_t \sim \mathcal{N}\big(\boldsymbol{H}\hat{\boldsymbol{x}}_t + \boldsymbol{K}(\boldsymbol{z}_t - \boldsymbol{H}\hat{\boldsymbol{x}}_t), \boldsymbol{H}\boldsymbol{P}_t\boldsymbol{H}^\top - \boldsymbol{K}\boldsymbol{H}\boldsymbol{P}_t\boldsymbol{H}^\top\big)$$

  - Or, if we take off $\boldsymbol{H}$ from all terms

$$\boldsymbol{K}' = \boldsymbol{P}_t\boldsymbol{H}^\top(\boldsymbol{H}\boldsymbol{P}_t\boldsymbol{H}^\top + \boldsymbol{R})^{-1}$$

$$\bar{\boldsymbol{x}}_t \sim \mathcal{N}\big(\hat{\boldsymbol{x}}_t + \boldsymbol{K}'(\boldsymbol{z}_t - \boldsymbol{H}\hat{\boldsymbol{x}}_t), \boldsymbol{P}_t - \boldsymbol{K}'\boldsymbol{H}\boldsymbol{P}_t\big)$$

Kalman Filter without Control

- The algorithm

- At time frame t
- Do the prediction for the hidden states using
  - The consolidated hidden state at time t-1
  - And the transition matrix

$$\hat{\boldsymbol{x}}_t = \boldsymbol{F}\bar{\boldsymbol{x}}_{t-1}$$

- Update the running covariance matrix of the prediction, too

$$\hat{\boldsymbol{P}}_t = \boldsymbol{F}\bar{\boldsymbol{P}}_{t-1}\boldsymbol{F}^\top + \boldsymbol{Q}$$

- Pick up your measurement $\boldsymbol{z}_t$
- Update your **Kalman gain**

$$\boldsymbol{K}' = \hat{\boldsymbol{P}}_t\boldsymbol{H}^\top(\boldsymbol{H}\hat{\boldsymbol{P}}_t\boldsymbol{H}^\top + \boldsymbol{R})^{-1}$$

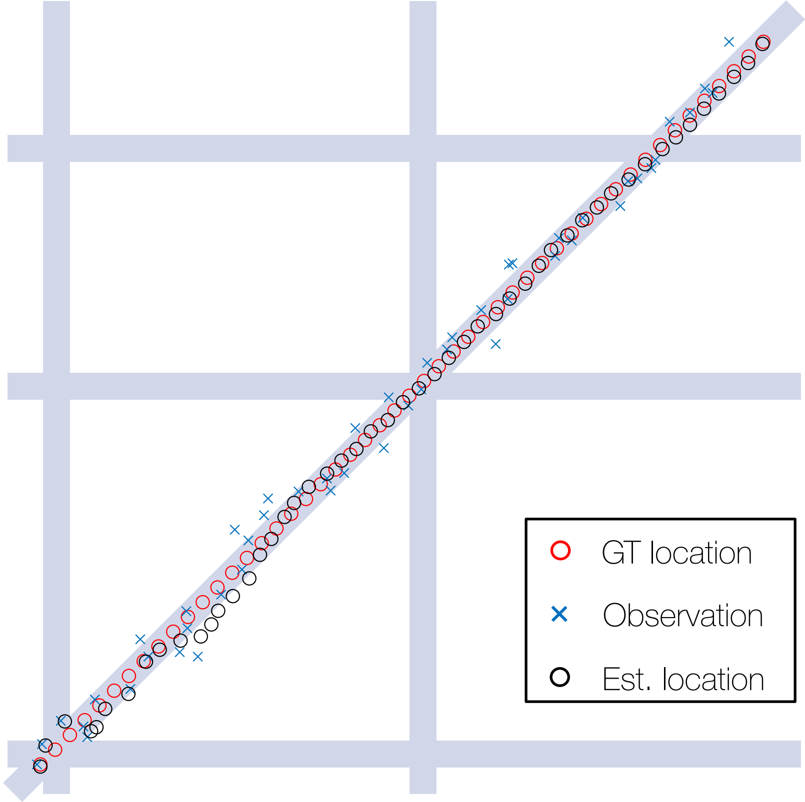
- Get the consolidated hidden states

$$\bar{\boldsymbol{x}}_t = \hat{\boldsymbol{x}}_t + \boldsymbol{K}'(\boldsymbol{z}_t - \boldsymbol{H}\hat{\boldsymbol{x}}_t)$$

$$\bar{\boldsymbol{P}}_t = \hat{\boldsymbol{P}}_t - \boldsymbol{K}'\boldsymbol{H}\hat{\boldsymbol{P}}_t$$

Kalman Filter without Control

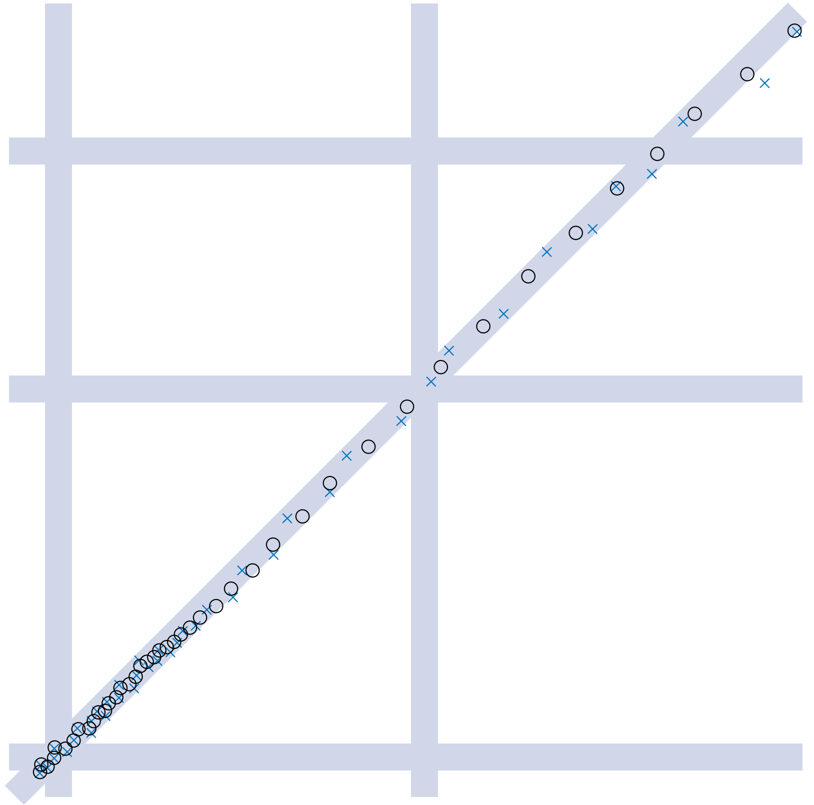
- It works



Kalman Filter with Control

- What if the car accelerates?

- We still want to track the car
  - But the physical model we've used is not good enough anymore



Kalman Filter with Control

- Control input to handle acceleration

- New transformation rules (emission)

Observed longitude

Observed latitude

$$
\begin{bmatrix} z_1(t) \\ z_2(t) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \begin{array}{c} \\ \\ \\ \\ \\ \\ \end{array}\begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \\ x_4(t) \\ x_5(t) \\ x_6(t) \end{bmatrix} \begin{array}{l} \text{Longitude} \\ \text{Velocity along longitude} \\ \textbf{Acceleration along longitude} \\ \text{Latitude} \\ \text{Speed along latitude} \\ \textbf{Acceleration along latitude} \end{array} + \begin{bmatrix} v_1(t) \\ v_2(t) \end{bmatrix}
$$

Measurement noise

- New prediction rules

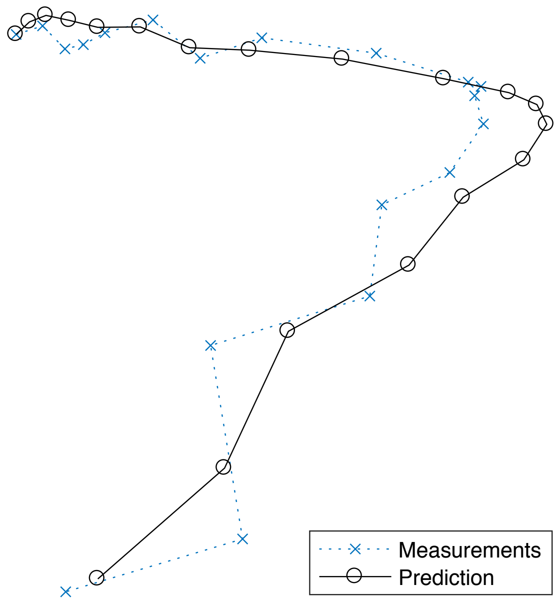
$$
\begin{bmatrix} x_1(t+1) \\ x_2(t+1) \\ x_3(t+1) \\ x_4(t+1) \\ x_5(t+1) \\ x_6(t+1) \end{bmatrix} = \begin{bmatrix} 1 & \Delta t & \frac{(\Delta t)^2}{2} & 0 & 0 & 0 \\ 0 & 1 & \Delta t & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & \Delta t & \frac{(\Delta t)^2}{2} \\ 0 & 0 & 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \\ x_4(t) \\ x_5(t) \\ x_6(t) \end{bmatrix} + \begin{bmatrix} w_1(t) \\ w_2(t) \\ w_3(t) \\ w_4(t) \\ w_5(t) \\ w_6(t) \end{bmatrix}
$$

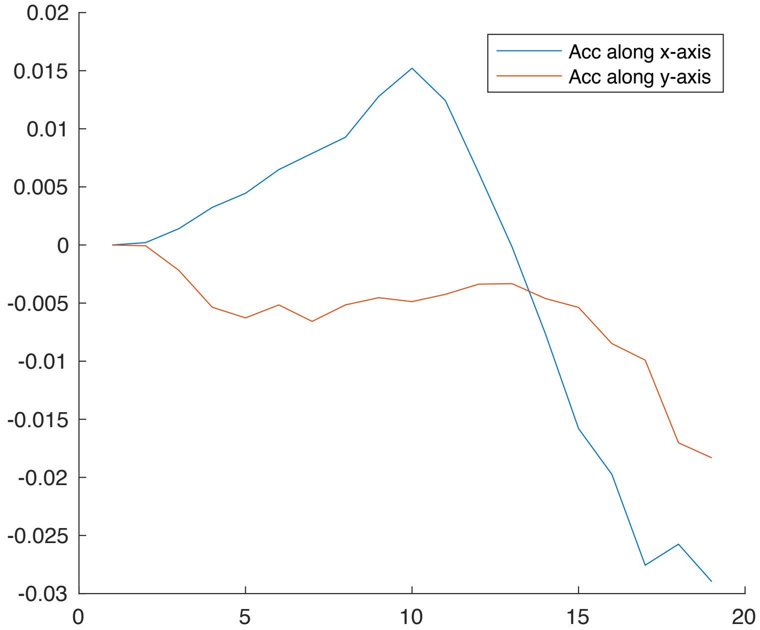
$$z_t = Hx_t + v_t \qquad x_{t+1} = Fx_t + w_t$$

- In other words
  - The only difference is the newly defined transformation and transition matrices

Kalman Filter with Control
- Handwriting on touch screens

- You're developing a handwritten digit recognition system
  - Running on your device with a touch screen
  - And a screen protection filter is applied

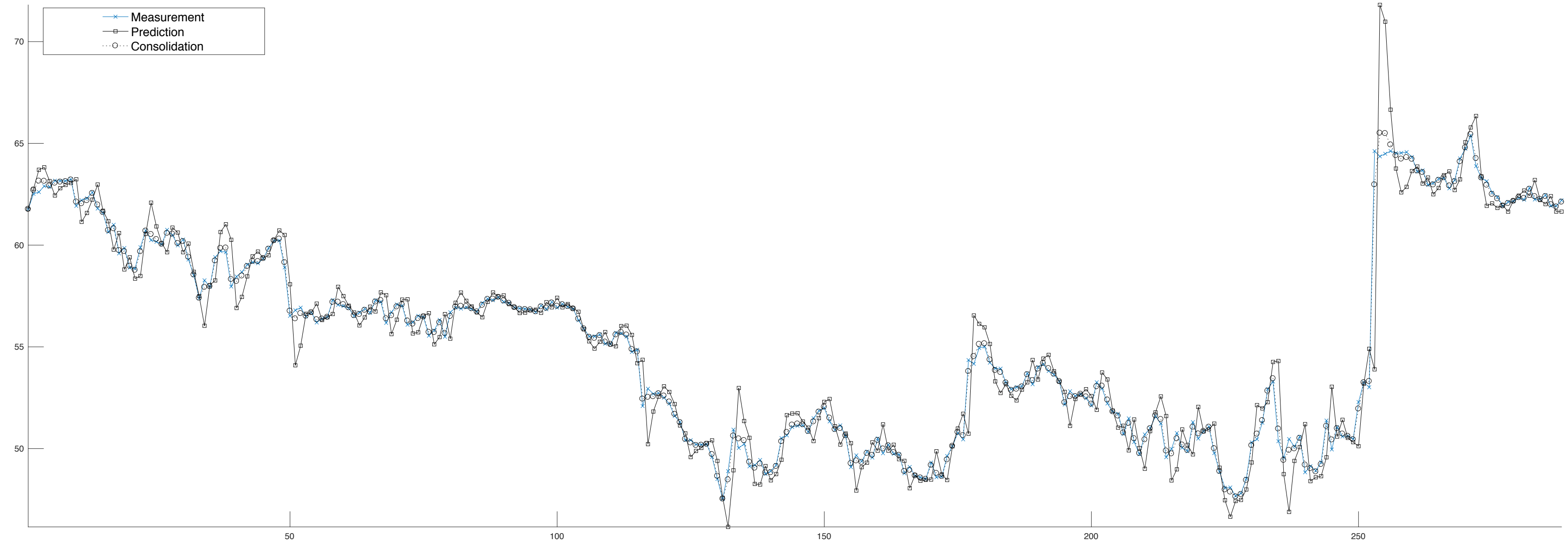




Kalman Filter with Control

- Stock market prediction

- Stock price of a company during 2016 and 2017
  - Accuracy of the prediction: 52.26% (not very impressive)
  - Maybe the stock price is not following the simple Gaussian assumption



Reading

- R. Faragher, "Understanding the Basis of the Kalman Filter Via a Simple and Intuitive Derivation [Lecture Notes]," in *IEEE Signal Processing Magazine*, vol. 29, no. 5, pp. 128-132, Sept. 2012. doi: 10.1109/MSP.2012.2203621
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6279585&isnumber=6279563

- http://www.bzarg.com/p/how-a-kalman-filter-works-in-pictures/

Thank You!