# Global Positioning System, Kalman Filters

Prof. Dr. Daniel Göhring

December 4, 2017

FOR INTERNAL USE ONLY

# Table of contents

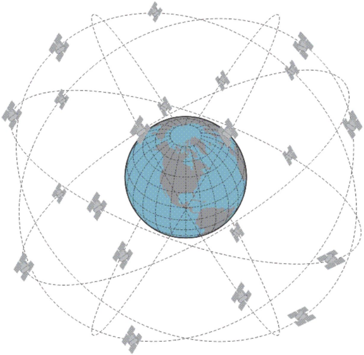
1 Global Positioning System
- Basics
- Model

2 Bayesian Filtering
- Basics
- Principle

3 Kalman Filtering
- Kalman Model
- Example

# GPS

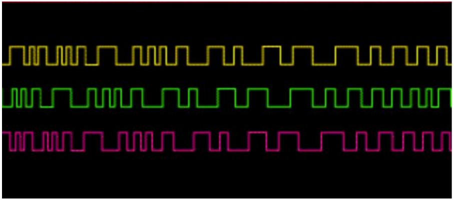
- originally named NAVSTAR GPS
- 1973: 24-hour real-time, high-accuracy navigation for moving vehicles in three dimensions, secure, passive, and global



Source:https://www.e-education.psu.edu/geog862/node/1768

# Rough idea

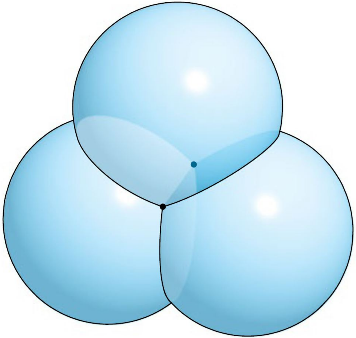
- At least 4 satellites are necessary for accurate localization.
- Satellites send a signal (pseudorandom code, PRC).



- The time it takes for that signal from satellite to receiver (shift w.r.t. receiver's PRC) is used to localize.
- Almanac data is data that describes the orbital courses of the satellites, it does not need to be precise.
- Ephemeris data is data that tells the GPS receiver where each GPS satellite should be at any time throughout the day. Up-to-date data is needed.

# Geometric interpretation

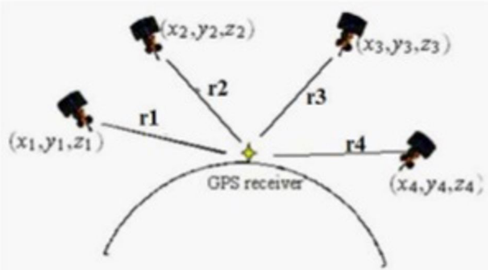
- Assume 3 satellites, given a certain radius (a known time difference w.r.t. the receiver), they intersect in two points:



- One of the two points is usually too far off the earth's surface, so it will be disregarded.

# Geometric interpretation

- Problem: Receiver's clock is low-cost and too inaccurate, we solve this but need a 4. satellite.



# Position calculation

- Define $d$ to be the difference between the earth-bound receiver clock and the synchronized time on the (now four) satellite clocks. Denote the location of satellite $i$ by $(X_i, Y_i, Z_i)$. $c$ is the speed of light. $t_i$ is the measured time difference between satellite $i$ and the receiver. Then the true intersection point $(x, y, z)$ satisfies:

$$r_1(x,y,z) = \sqrt{(x-X_1)^2 + (y-Y_1)^2 + (z-Z_1)^2} = c(t_1 - d) \quad (1)$$

$$r_2(x,y,z) = \sqrt{(x-X_2)^2 + (y-Y_2)^2 + (z-Z_2)^2} = c(t_2 - d) \quad (2)$$

$$r_3(x,y,z) = \sqrt{(x-X_3)^2 + (y-Y_3)^2 + (z-Z_3)^2} = c(t_3 - d) \quad (3)$$

$$r_4(x,y,z) = \sqrt{(x-X_4)^2 + (y-Y_4)^2 + (z-Z_4)^2} = c(t_4 - d) \quad (4)$$

# Position calculation

- Geometrically speaking, four spheres may not have a common intersection point, but they will if the radii are expanded or contracted by the right common amount $c \cdot d$. The equations can be equivalently written as:

$$r_1^2 = c(t_1 - d)^2 = (x - X_1)^2 + (y - Y_1)^2 + (z - Z_1)^2 \quad (5)$$
$$r_2^2 = c(t_2 - d)^2 = (x - X_2)^2 + (y - Y_2)^2 + (z - Z_2)^2 \quad (6)$$
$$r_3^2 = c(t_3 - d)^2 = (x - X_3)^2 + (y - Y_3)^2 + (z - Z_3)^2 \quad (7)$$
$$r_4^2 = c(t_4 - d)^2 = (x - X_4)^2 + (y - Y_4)^2 + (z - Z_4)^2 \quad (8)$$

# Position calculation

- Now we want to eliminate the quadratic terms of $x, y, z$. After multiplying out the squared terms and subtracting the three lower equations from the first, we end up with three new equations of the form $d = a_1 x + a_2 y + a_3 z + a_4$:

$$
\begin{aligned}
r_1^2 - r_2^2 = \quad & c((t_1 - d)^2 - (t_2 - d)^2) \\
= \quad & 2x(X_2 - X_1) + X_1^2 - X_2^2 \\
& +2y(Y_2 - Y_1) + Y_1^2 - Y_2^2 \\
& +2z(Z_2 - Z_1) + Z_1^2 - Z_2^2 \qquad (9) \\
r_1^1 - r_3^2 = \quad & 2x(X_3 - X_1) + X_1^2 - X_3^2 \\
& +\ldots \qquad (10) \\
r_1^1 - r_4^2 = \quad & 2x(X_4 - X_1) + X_1^2 - X_4^2 \\
& +\ldots \qquad (11)
\end{aligned}
$$

# Calculating Time Delay

- Using the 3 equations (9,10,11) above for 4 given satellite positions we end up with 3 equations (not linear) with 4 unknown variables $x, y, z, d$.
- This leads to solutions for $x, y, z$ with respect to free parameter $d$, as $\mathrm{x} = a_1 d + a_2$, and so on for $y$ and $z$. Putting these $x, y, z$ into equation (1), we can calculate $d$ solving a quadratic equation and then we calculate $x, y, z$.
- But with more than 4 satellites, there is usually no single point where all spheres intersect. Therefore we have to use approximations, e.g., least squares or gradient descent.

# Example

Given 4 satellites:

$$X_1 = 1, Y_1 = 0, Z_1 = 0, t_1 = 1.5 \tag{12}$$
$$X_2 = 0, Y_2 = 2, Z_2 = 0, t_2 = 2.5 \tag{13}$$
$$X_3 = 0, Y_3 = 0, Z_3 = 3, t_3 = 3.5 \tag{14}$$
$$X_4 = 1, Y_4 = 1, Z_4 = 1, t_4 = \sqrt{3} + 0.5 \approx 2.232 \tag{15}$$

, assume for simplicity $c = 1$.

# Example

Now apply equations (5 - 8) for to the values of the 4 satellites:

$$
\begin{aligned}
(1.5-d)^2-(x-1)^2 &= (y-0)^2+(z-0)^2 \\
-3d+1.25+2x &= x^2+y^2+z^2-d^2 \quad (16)\\
-5d+2.25+4y &= x^2+y^2+z^2-d^2 \quad (17)\\
-7d+3.25+6y &= x^2+y^2+z^2-d^2 \quad (18)\\
-4.46d+1.98+2x+2y+2z &= x^2+y^2+z^2-d^2 \quad (19)
\end{aligned}
$$

# Example

Subtract each of the last three equations from the first one (diagonalization):

$$-1 + 2d + 2x - 4y = 0 \tag{20}$$
$$-2 + 4d + 2x - 6z = 0 \tag{21}$$
$$-0.73 + 1.46d - 2y - 2z = 0 \tag{22}$$

After solving this linear equation, we get:

$$x \approx -0.52d + 0.26 \tag{23}$$
$$y \approx 0.24d - 0.12 \tag{24}$$
$$z \approx 0.49d - 0.245 \tag{25}$$

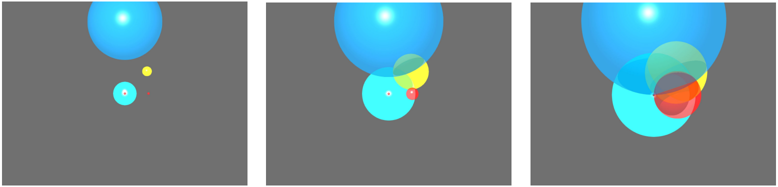
# Example

Apply the solutions of $x, y, z$ to equation (5):

$$(1.5 - d)^2 = (-0.52d + 0.26 - 1)^2 + (0.24d - 0.12)^2 + (0.49d - 0.245)^2 \tag{26}$$

$$\begin{aligned} 0 &= d^2 - 8.6d + 4.08 && (27) \\ d_{1,2} &= 4.3 \pm \sqrt{14.41} && (28) \\ d_1 &= 0.5 && (29) \\ d_2 &= 8 \rightarrow \text{reject, too far away} && (30) \\ &\rightarrow x = 0, y = 0, z = 0 && (31) \end{aligned}$$

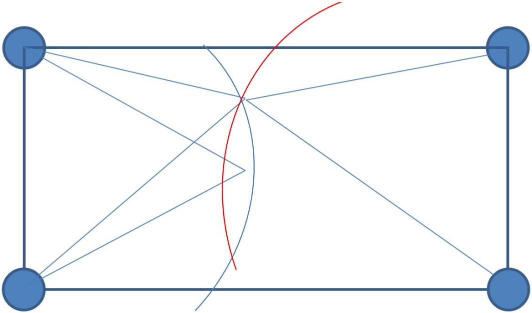
# Example



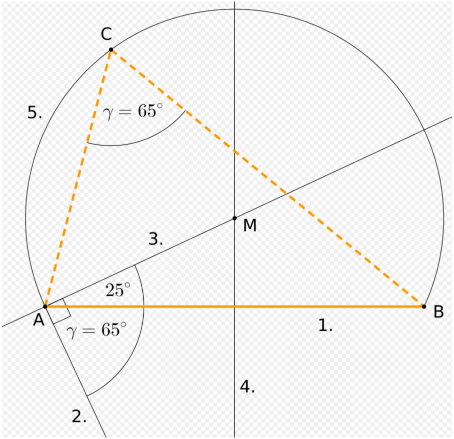
Satellite signals over time. Red: 1, green: 2, blue: 3, yellow: 4, bright spot in the center corresponds to real position.

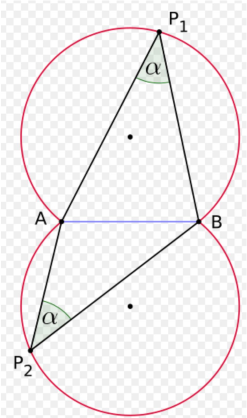
Link - https://youtu.be/xixajrhn3T4

# Pseudo - GPS: Four Light Posts

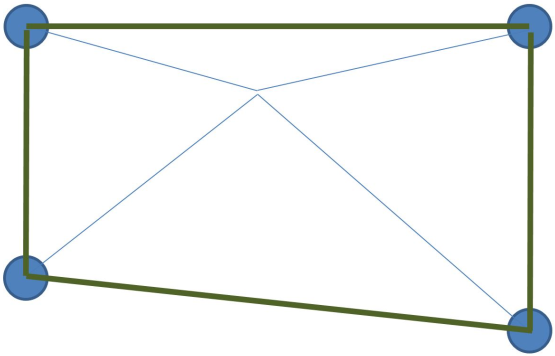


# Beacon Localization



# Four Light Posts, a Second Option

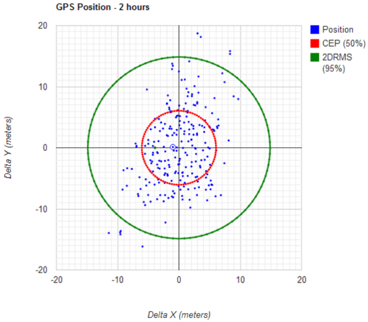


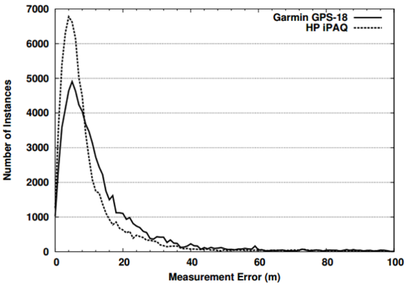
# Kalman Filters

- Widely used for state estimation Part of Bayesian filter family
- Handle uncertainty by assuming Gaussian distribution modeling state and covariances
- Optimal for time discrete, linear processes, i.e., they minimize the quadratic error of state estimation
- Extensions exist for non-linear processes (Extended, Unscented Kalmanfilters)

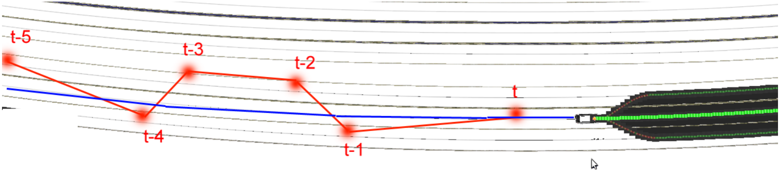
# Introductory Example

- state estimation (speed and velocity) of an autonomous car using GPS (x,y coordinates). Examples:



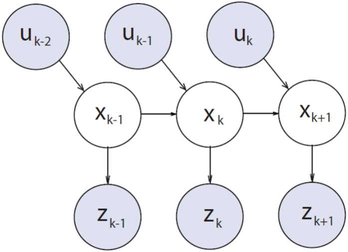
# Introductory Example




- car state: position, orientation, velocity
- sensory data: position (GPS)
- control params: acceleration, steering angle
- combine sensory data with physical model for state estimation

# Hidden-Markov-Model

- State variables x are "hidden" or "latent"
- Actions u and sensor readings z are known
- $x$, $u$, and $z$: n-dimensional state vectors
- conditional independency: $x_k$ depends on $u_{k-1}$, $x_{k-1}$



# Bayes Theorem

- Kalman Filters are part of Bayesian Filter family
- Assumption:

$$\overbrace{p(x|Z)}^{\text{posterior}} = \frac{\overbrace{p(Z|x)}^{\text{likelihood}}\,\overbrace{p(x)}^{\text{prior}}}{\underbrace{p(Z)}_{\text{normalizing constant}}}$$

- where x is the state vector and Z denotes the sensor readings

# Bayesian Filters

- 2 steps:
  - I. prediction (over time)
  - II. update (measurements)
- I. prediction:

$$Bel^{-}(x_t) \leftarrow \int p(x_t|x_{t-1}, u_{t-1}) Bel(x_{t-1}) dx_{t-1}$$

- II. update:

$$Bel(x_t) \leftarrow \eta p(z_t|x_t) Bel^{-}(x_t)$$

# Kalman-Model

- Process-model:
  - A: state transition model (matrix), xt: state vector at time t, B: control input model, ut: control input vector, wt: white noise
  - Q: process noise model

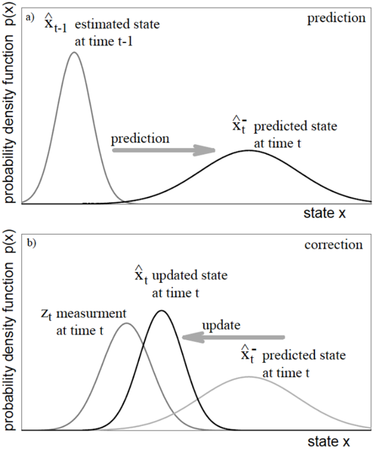
$$
\begin{aligned}
x_t &= Ax_{t-1} + Bu_{t-1} + \omega_{t-1} \\
\omega_{t-1} &\sim N(0, Q)
\end{aligned}
$$

- Sensor-model:
  - H: observation model, vt: white noise with covariance R, R: sensor noise model

$$
\begin{aligned}
z_t &= Hx_t + v_t \\
v_t &\sim N(0, R)
\end{aligned}
$$

# Kalman-Filter Principle

- I. prediction (propagation)
- II. update (correction)



# Prediction

- predicting state vector $\hat{x}_t^-$ and covariance matrix $P_t^-$

*z: sensory data vector*
*u: input vector*

$$
\begin{aligned}
\hat{x}_t^- &\triangleq E[x_t | u_{t-1}, ..., z_0] \\
\text{Markov} \quad &= E[Ax_{t-1} + Bu_{t-1} + w_{t-1} | u_{t-1}, ..., z_0] \\
&= A\hat{x}_{t-1} + Bu_{t-1}
\end{aligned}
$$

$$
\begin{aligned}
P_t^- &\triangleq E[(x_t - E[x_t])(x_t - E[x_t])^T | u_{t-1}, ..., z_0] \\
\text{Markov} \quad &= AE[(x_t - E[x_t])(x_t - E[x_t])^T]A^T + E[w_t w_t^T] \\
&= AP_{t-1}A^T + Q
\end{aligned}
$$

# Update

- the difference between expected measurement $\hat{z}_t = H\hat{x}_t^-$ and real measurement is called innovation, its covariance matrix $S$ tells, how "confident" the innovation is:

$$S_t = (R + HP_t^- H^T)$$

- Kalmangain $K$ determines, how much the innovation will be considered for the new state estimate:

$$K_t = P_t^- H^T S_t^{-1}$$

- finally, calculation of a posteriori state estimate x and covariance matrix $P$:

$$\begin{aligned} \hat{x}_t &= \hat{x}_t^- + K_t(z_t - \hat{z}_t) \\ P_t &= P_t^- - K_t S_t K_t^T \end{aligned}$$

# In a Nutshell

Prediction Step

A: propagation matrix,
B: input matrix

^$x_t^-$: a priori state estimate

$$\hat{x}_t^- = A\hat{x}_{t-1} + Bu_{t-1}$$

^$x_{t-1}$: a post. state est. at t-1
$u_{t-1}$: input vector

$$P_t^- = AP_{t-1}A^T + Q$$

$P_t^-$: a priori error cov. estimate
Q: process covariance

K: Kalman gain

$$K_t = P_t^- H^T S_t^{-1}$$

S: innovation covariance

H: sensory data transform. matrix

$$S_t = (R + HP_t^- H^T)$$

R: sensor covariance

Update Step

^$x_t$: a posteriori state estimate

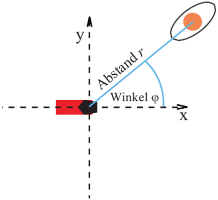
$$\hat{x}_t = \hat{x}_t^- + K_t(z_t - \hat{z}_t)$$

$$P_t = P_t^- - K_t S_t K_t^T$$

$P_t$: a posteriori error cov. estimate

# Application Example

- ball tracking
- state variables position and velocity
- module execution freq.: 30 Hz
- egocentric coordinate frame



# Application Example

- propagation of ball position and velocity:

$$\hat{x}_t^- = A\hat{x}_{t-1} + Bu_{t-1}$$

$$\begin{pmatrix} s_{x_t} \\ s_{y_t} \\ v_{x_t} \\ v_{y_t} \end{pmatrix} = \begin{pmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & exp^{\gamma \Delta t} & 0 \\ 0 & 0 & 0 & exp^{\gamma \Delta t} \end{pmatrix} \times \begin{pmatrix} s_{x_{t-1}} \\ s_{y_{t-1}} \\ v_{x_{t-1}} \\ v_{y_{t-1}} \end{pmatrix}$$

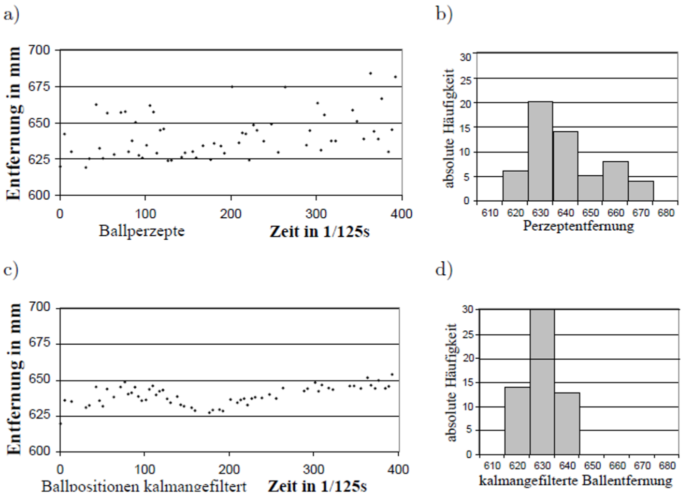
- robot motion $u_x$, $u_y$, $u_\theta$ *(not completely linear)*

$$\begin{pmatrix} s'_x \\ s'_y \\ v'_x \\ v'_y \end{pmatrix} = \begin{pmatrix} \cos(u_\theta) & -\sin(u_\theta) & 0 & 0 \\ \sin(u_\theta) & \cos(u_\theta) & 0 & 0 \\ 0 & 0 & \cos(u_\theta) & -\sin(u_\theta) \\ 0 & 0 & \sin(u_\theta) & \cos(u_\theta) \end{pmatrix} \times \begin{pmatrix} s_x \\ s_y \\ v_x \\ v_y \end{pmatrix} + \begin{pmatrix} u_x \\ u_y \\ 0 \\ 0 \end{pmatrix}$$

# Acquisition of Covariance Matrices

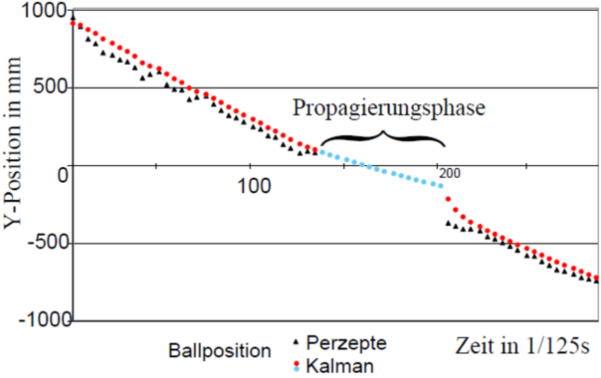
- Process noise covariance matrix $Q$ can be generated by letting the robot move a certain distance for many times from a determined starting point for e.g. 5 seconds. Then measure covariance matrix over $x, y, \theta$ over the robots final positions
- Sensor noise covariance matrix $R$ is generated by letting the robot walk on the spot and measure the distance vectors to a ball. $R$ is their covariance matrix. Be aware: $R$ depends on distance, too. Sensory data transform matrix $H$ can be derived from the mean distance vector.

# Experimental Data



# What if no Updates Available

- If no updates available, Kalman-Filters rely on predictions only
- Error matrix P increases because of process noise



# Outlook: Extended Forms

- Extended Kalman-Filter
  - use Taylor series for Gaussian state estimation
  - state transition and observation models differentiable functions (not necessarily linear)
  - used for GPS
- Unscented Kalman-Filter
  - Uses sigma-points for state estimation

# Literature


- [1] Marcus A. Brubaker, University of Toronto, Canada
- [2] S. Thrun, W. Burgard, D. Fox, Probabilistic Robotics, MIT Press, 2005.
- [3] Michael Coyle, Calculating your own GPS accuracy, http://blog.oplopanax.ca/2012/11/calculating-gps-accuracy/
- [4] Jeffrey Hemmes, Douglas Thain, Christian Poellabauer: Cooperative Localization in GPS-Limited Urban Environments,