2D Image Processing

Bayes filter implementation: Kalman filter

Prof. Didier Stricker
Dr. Gabriele Bleser
Kaiserlautern University

http://ags.cs.uni-kl.de/

DFKI – Deutsches Forschungszentrum für Künstliche Intelligenz

http://av.dfki.de

Some slides based on G. Panin and K. Smith
EXAM: Oral exam from **28th August to 1 September 2017**

Please register and get an appointment, i.e. send a mail to: **keonna.cunningham@dfki.de**
Sensor Fusion: Inertial Sensors + Vision
Outline

- Recap: Bayes filter
- Some facts about Gaussians
- Kalman filter algorithm

Some slides are based on G. Panin, S. Thrun, and K. Smith
Summary: Bayes filter framework

- **Given:**
  - Stream of measurements $z_{1:t}$ and control data $u_{1:t}$
  - **Measurement model** $p(z_t|x_t)$
  - **Motion/Dynamic model** $p(x_t|x_{t-1}, u_t)$
  - **Prior/Initial probability of the system state** $p(x_0)$

- **Wanted:**
  - **Estimate of the state** $x_t$ of a dynamical system
  - The posterior of the state is also called **belief**: $\text{bel}(x_t) = p(x_t|u_{1:t}, z_{1:t})$
Summary: Bayes filters

- Probabilistic tool for recursively estimating the state of a dynamical system from noisy measurements and control inputs.

- Based on probabilistic concepts such as the Bayes theorem, Theorem of Total Probability (marginalization), and conditional independence.

- Make a Markov assumption according to which the state is a complete summary of the past. In real-world problems, this assumption is usually an approximation!
**Markov assumption**

\[
p(x_t | x_{0:t-1}, z_{1:t-1}, u_{1:t}) = p(x_t | x_{t-1}, u_t) \\
p(z_t | x_{0:t}, z_{1:t-1}, u_{1:t}) = p(z_t | x_t)
\]

**Underlying Assumptions**
- **Static world** (future is independent from past, given current state)
- **Independent noise**
- **Perfect model, no approximation errors**
Markov assumption revisited

Reality: sources of error and uncertainty

- Environment dynamics
- Approximate computation
- Inaccurate models
- Sensor limitations
- Random effects
Bayes update rule

\[ bel(x_t) = \eta p(z_t | x_t) \int p(x_t | x_{t-1}, u_t) \, bel(x_{t-1}) \, dx_{t-1} \]

- Provides a general concept
- Can in the presented form only be implemented for simple estimation problems, requires either…or…
  - closed form solutions for multiplication and integral
  - restriction to finite state spaces
- What is missing to be able to use this:
  - Concrete representations for probability density functions
  - Implementable and tractable filter approximations
  - Applicability to continuous estimation problems
Representations of PDFs

- Example: Model the probability distribution of faces appearing in frame $t$
Representations of PDFs

- Example: Model the probability distribution of faces appearing in frame \( t \)

\[ p(x) \]
Representations of PDFs

- Example: Model the probability distribution of faces appearing in frame $t$
Representations of PDFs

- Dirac

\[ p(x) = \begin{cases} 
1 & \text{if } x = \mu \\
0 & \text{otherwise} 
\end{cases} \]

One hypothesis, no uncertainty
MAP=Maximum a posteriori: we look only for the maximum
Representations of PDFs

- Gaussian

\[ p(x) = \eta \exp \left\{ -\frac{1}{2} (x - \mu)^T \Sigma^{-1} (x - \mu) \right\} \]

One hypothesis + uncertainty
Representations of PDFs

- Mixture of Gaussians

\[ p(x) = \eta \sum_{m=1}^{M} w_m \exp \left\{ -\frac{1}{2} (x - \mu_m)^T \Sigma^{-1}_m (x - \mu_m) \right\} \]

Fixed number, \( M \), of modes + uncertainties
Representations of PDFs (non-parametric)

- Set of discrete samples (particles) \( \{x_t^{(n)}, n = 1, \ldots, N\} \)

\[
p(x) \approx \eta \sum_{n=1}^{N} \delta(x_t - x_t^{(n)})
\]

Any number and shape of modes
Representations of PDFs (non-parametric)

- Set of weighted particles

\[ p(x) \approx \sum_{n=1}^{N} w_t^{(n)} \delta(x_t - x_t^{(n)}) \]

\[ \{x_t^{(n)}, w_t^{(n)}\}_{n=1}^{N} \]

\[ w_t^{(n)} \in [0,1] \]

\[ \sum_{n} w_t^{(n)} = 1 \]

Any number and shape of modes
Filter methods (rules-of-thumb)

- Different characteristics in terms of:
  - Computational efficiency
  - Accuracy of the approximation
  - Ease of implementation

- Bayes Filter
  - Linear Gaussian models

- Kalman Filter
  - Nonlinear models, Gaussian noises

- Unscented Kalman Filter

- Extended Kalman Filter

- Kalman Filter banks
  - (Non)linear models, Gaussian noises, multi-modal

- Particle Filter
  - Highly nonlinear models, non-Gaussian noises, multi-modal
Kalman filter

- **Published in 1960**

- **Used for many problems**
  - Guidance
  - Navigation
  - Autopilots
  - Radar
  - Satellite
  - Weather forecasting
Univariate Gaussian

\[ p(x) = (2\pi\sigma^2)^{\frac{1}{2}} \exp\left\{ -\frac{1}{2} \frac{(x - \mu)^2}{\sigma^2} \right\} := N(X; \mu, \sigma^2) \]

Exponentiation of a quadratic function
Multivariate Gaussian

- Normal distribution over a vector \(\rightarrow\) generalization of univariate normal distribution to higher dimensions

\[
p(x) = \frac{1}{\det(2\pi \Sigma)^{\frac{1}{2}}} \exp\left\{-\frac{1}{2} (x - \mu)^T \Sigma^{-1} (x - \mu)\right\}
\]

\[
:= N(X; \mu, \Sigma)
\]

Mean vector \((n \times 1)\)  
Covariance matrix \((n \times n)\)  
(positive semi-definite, symmetric matrix)

\[
\mu = E[X] = \int x \, p(x) \, dx
\]

\[
\text{Cov}(X) = E\left( (X - E(X))(X - E(X))^T \right) = E(XX^T) - E(X)E(X)^T
\]

\[
= \text{Cov}(X, X)
\]
Matlab example: Multivariate Gaussian
Matlab example: Samples drawn from a multivariate Gaussian
Properties of Gaussians

- Affine functions of Gaussians (linear in random variable) are again Gaussians with the following properties:

  **Univariate case:** $\mu, X, Y, \sigma, a, b$ are scalars
  \[
  \begin{align*}
  X &\sim N(\mu, \sigma^2) \\
  Y &= aX + b
  \end{align*}
  \[
  \Rightarrow 
  Y \sim N(a\mu + b, a^2\sigma^2)
  
  **Multivariate case:** $\mu, X, Y, B$ are vectors, $\Sigma, A$ are matrices
  \[
  \begin{align*}
  X &\sim N(\mu, \Sigma) \\
  Y &= AX + B
  \end{align*}
  \[
  \Rightarrow 
  Y \sim N(A\mu + B, A\Sigma A^T)
  \]
Properties of Gaussians

- Let $X \sim N(0, I)$ be standard normally distributed
- Let $Y = AX + b$, with $A \in \mathbb{R}^{n \times n}$ invertible and $b \in \mathbb{R}^n$

\[
p(Y = y) = \frac{p(X = A^{-1}(y - b))}{\det(A)}
\]

$N(b, AA^T)(y) = N(0, I)(A^{-1}(y - b))/\det(A)$

- E.g. when halving a univariate Gaussian random variable, it is not only “compressed” by the factor 2, but also higher, since the values are more dense

Note: $\det A = \sqrt{\det AA^T}$
Matlab example: Two Gaussians with different variance (and mean)
Handling Gaussians

- How to generate samples $Y \sim N(\mu, C)$ with mean $\mu$ and covariance $C$ from $X \sim N(0, I)$ (standard normal distribution)?

  - **Univariate case:**
    $$Y = \sqrt{C}X + \mu$$

  - **Multivariate case:** $C \in \mathbb{R}^{n \times n}$ with Cholesky decomposition $C = LL^T$ and $\mu \in \mathbb{R}^n$
    $$Y = LX + \mu$$
Matlab example: General Gaussian vs. transformed normal distribution
Drawing confidence regions of Gaussians (ex. 2D).

\[ X \sim N(\mu, \Sigma), n\text{-dimensional (here } n = 2) \]
\[ p(x) \propto \eta \exp \left\{ -\frac{1}{2} (x - \mu)^T \Sigma^{-1} (x - \mu) \right\} \]

Shape: draw a scaled and shifted circle

Overall scale of the ellipse determines enclosed probability mass \( \rightarrow \) confidence region

Matlab:
\texttt{chi2inv(\text{conf, } n)} provides squared scale

Quadratic function, induces an ellipse

Basically standardization \( \rightarrow \chi_n^2 \) distributed
Matlab example: Confidence region
Tracking with Kalman filters: Gaussians!

\[
\overline{\text{bel}}(x_t) = \int p(x_t|x_{t-1},u_t) \text{bel}(x_{t-1}) \, dx_{t-1}
\]

\[
\text{bel}(x_t) = \eta \, p(z_t|x_t) \overline{\text{bel}}(x_t)
\]

All PDFs are assumed Gaussian

→ closed-form solutions for integral and product of Gaussians
→ tractable Bayes filter implementation for continuous spaces
→ optimal (minimum variance) estimator for linear Gaussian systems
Product of Gaussians (fusion formula)

- Is again a Gaussian (though no longer normalized)
- **Univariate case:**
  \[ X_1 \sim N(\mu_1, \sigma_1^2) \quad X_2 \sim N(\mu_2, \sigma_2^2) \quad \Rightarrow \quad p(X_1) \cdot p(X_2) \sim N \left( \frac{\sigma_2^2}{\sigma_1^2 + \sigma_2^2} \mu_1 + \frac{\sigma_1^2}{\sigma_1^2 + \sigma_2^2} \mu_2, \frac{\sigma_1^2 \sigma_2^2}{\sigma_1^2 + \sigma_2^2} \right) \]

- **Multivariate case:**
  \[ X_1 \sim N(\mu_1, \Sigma_1) \quad X_2 \sim N(\mu_2, \Sigma_2) \quad \Rightarrow \quad p(X_1) \cdot p(X_2) \sim N(\mu_1 + K(\mu_2 - \mu_1), \Sigma_1 - K\Sigma K^T) \quad K = \Sigma_1 \left( \Sigma_1 + \Sigma_2 \right)^{-1} \]

Recursive formulation

\[ N \left( \mu_1 + k(\mu_2 - \mu_1)\sigma_1^2 - k\sigma_1^2 \right) \quad k = \frac{\sigma_1^2}{\sigma_1^2 + \sigma_2^2} \]
Fusion formula derived

\[ e^a e^b = e^{a+b} \]

\[ X_1 \sim N(\mu_1, \sigma_1^2) \quad X_2 \sim N(\mu_2, \sigma_2^2) \]
\[ \Rightarrow p(X_1) \cdot p(X_2) = \eta \cdot \exp \left\{ -\frac{1}{2} \frac{(x - \mu_1)^2}{\sigma_1^2} - \frac{1}{2} \frac{(x - \mu_2)^2}{\sigma_2^2} \right\} \]

\[ \frac{\partial}{\partial x} \left\{ -\frac{1}{2} \frac{(x - \mu_1)^2}{\sigma_1^2} - \frac{1}{2} \frac{(x - \mu_2)^2}{\sigma_2^2} \right\} = \frac{x - \mu_1}{\sigma_1^2} + \frac{x - \mu_2}{\sigma_2^2} = 0 \quad \text{(for new } \mu := x) \]

Maximum of quadratic function \( \Rightarrow \) set 1\(^{\text{st}}\) derivative to 0 \( \Rightarrow \)
Mean of resulting distribution

\[ (\mu - \mu_1)\sigma_2^2 + (\mu - \mu_2)\sigma_1^2 = 0 \]
\[ \mu (\sigma_1^2 + \sigma_2^2) = \mu_1 \sigma_2^2 + \mu_2 \sigma_1^2 \]
\[ \mu = \frac{\mu_1 \sigma_2^2 + \mu_2 \sigma_1^2}{\sigma_1^2 + \sigma_2^2} \]

\[ \frac{\partial^2}{\partial x^2} \left\{ -\frac{1}{2} \frac{(x - \mu_1)^2}{\sigma_1^2} - \frac{1}{2} \frac{(x - \mu_2)^2}{\sigma_2^2} \right\} = \sigma_1^{-2} + \sigma_2^{-2} \]

Curvature of quadratic function \( \Rightarrow \) 2\(^{\text{nd}}\) derivative \( \Rightarrow \) Inverse of covariance

\[ \sigma = \frac{1}{\sigma_1^{-2} + \sigma_2^{-2}} = \frac{\sigma_1^2 \sigma_2^2}{\sigma_1^2 + \sigma_2^2} \]
Kalman filter models (state-space model)

- **Motion model:**
  - Linear stochastic difference equation in $x$
  - Evolution of $x_t$ based on previous state $x_{t-1}$ and control input $u_t$

\[
x_t = A_t x_{t-1} + B_t u_t + \varepsilon_t \iff p(x_t \mid x_{t-1}, u_t) = N(x_t; A_t x_{t-1} + B_t u_t, R_t)
\]

\[
\varepsilon_t \sim N(0, R_t)
\]

- **Measurement model:**
  - Linear stochastic equation in $x$
  - Describes, how measurements $z_t$ are related to the state

\[
z_t = C_t x_t + \delta_t \iff p(z_t \mid x_t) = N(z_t; C_t x_t, Q_t)
\]

\[
\delta_t \sim N(0, Q_t)
\]
Components of a Kalman Filter

$A_t$ Matrix $(n \times n)$ that describes how the state evolves from $t - 1$ to $t$ without control input or noise.

$B_t$ Matrix $n \times l$ that describes how the control input $u_t$ changes the state from $t - 1$ to $t$.

$C_t$ Matrix $(k \times n)$ that describes how to map the state $x_t$ to an observation $z_t$.

$\varepsilon_t$ Random variables representing the process and measurement noise that are assumed to be independent and normally distributed with covariance $R_t$ and $Q_t$ respectively.
Kalman filter update: prediction/time update

Initial belief

\[ \text{bel}(x_{t-1}) = N(x_{t-1}; \mu_{t-1}, \Sigma_{t-1}) \]

Prediction

\[ \overline{\text{bel}}(x_t) = \int N(x_t; A_t x_{t-1} + B_t u_t, R_t) \text{bel}(x_{t-1}) \, dx_{t-1} \]

Gaussian

1D-case

\[ \overline{\text{bel}}(x_t) = \begin{cases} \bar{\mu}_t = a_t \mu_{t-1} + b_t u_t \\ \bar{\sigma}_t^2 = a_t^2 \sigma_t^2 + \sigma_{\epsilon,t}^2 \end{cases} \]

Process noise covariance

\[ \begin{cases} \bar{\mu}_t = A_t \mu_{t-1} + B_t u_t \\ \bar{\Sigma}_t = A_t \Sigma_{t-1} A_t^T + R_t \end{cases} \]
Kalman filter update: correction/measurement update

Initial belief \( \text{Prediction} \) Measurement

\[ \text{bel}(x_{t}) \propto N(z_{t}; C_{t}x_{t}, Q_{t})\overline{\text{bel}}(x_{t-1}) \]

Gaussian fusion formula including linear functions!

**1D-case**

\[
\text{bel}(x_{i}) = \begin{cases} 
\mu_{i} = \overline{\mu}_{i} + k_{i}(z_{i} - c_{i}\overline{\mu}_{i}) \\
\sigma_{i}^{2} = (1 - k_{i}c_{i})\overline{\sigma}_{i}^{2}
\end{cases},
\text{with } k_{i} = \frac{\overline{\sigma}_{i}^{2}c_{i}}{\overline{\sigma}_{i}^{2}c_{i}^{2} + \overline{\sigma}_{\delta_{i}}^{2}}
\]

\[
\text{bel}(x_{i}) = \begin{cases} 
\mu_{i} = \overline{\mu}_{i} + K_{i}(z_{i} - C_{i}\overline{\mu}_{i}) \\
\Sigma_{i} = (I - K_{i}C_{i})\overline{\Sigma}_{i}
\end{cases},
\text{with } K_{i} = \overline{\Sigma}_{i}C_{i}^{T}(C_{i}\overline{\Sigma}_{i}C_{i}^{T} + Q_{i})^{-1}
\]

Uncertainty decreases

Measurement noise covariance
**Measurement update derivation**

\[
bel(x_t) = \eta \quad p(z_t \mid x_t) \quad \Rightarrow \quad bel(x_t) \\
\sim N(z_t; C_t x_t, Q_t) \quad \sim N(x_t; \mu_t, \Sigma_t) \\
\Rightarrow \\
bel(x_t) = \eta \exp \left\{ -\frac{1}{2} (z_t - C_t x_t)^T Q_t^{-1} (z_t - C_t x_t) \right\} \exp \left\{ -\frac{1}{2} (x_t - \mu_t)^T \Sigma_t^{-1} (x_t - \mu_t) \right\}
\]

- Find new \( \mu \) and \( \Sigma \) from 1\(^{st}\) and 2\(^{nd}\) derivative of quadratic function in exponent \( \Rightarrow \) cf. fusion formula

\[
bel(x_t) = \left\{ \begin{array}{l}
\mu_t = \mu_t + K_t (z_t - C_t \mu_t) \\
\Sigma_t = (I - K_t C_t) \Sigma_t
\end{array} \right.
\] with \( K_t = \Sigma_t C_t^T (C_t \Sigma_t C_t^T + Q_t)^{-1} \)

Whole derivation given, e.g., in Probabilistic Robotics by Thrun
Time update derivation

\[
\overline{\text{bel}}(x_t) = \int p(x_t | u_t, x_{t-1}) \, \text{bel}(x_{t-1}) \, dx_{t-1}
\]

\[
\downarrow \\
\sim N(x_t; A_t x_{t-1} + B_t u_t, R_t) \sim N(x_{t-1}; \mu_{t-1}, \Sigma_{t-1})
\]

\[
\downarrow \\
\overline{\text{bel}}(x_t) = \eta \int \exp \left\{ -\frac{1}{2} (x_t - A_t x_{t-1} - B_t u_t)^T R_t^{-1} (x_t - A_t x_{t-1} - B_t u_t) \right\}
\]

\[
\exp \left\{ -\frac{1}{2} (x_{t-1} - \mu_{t-1})^T \Sigma_{t-1}^{-1} (x_{t-1} - \mu_{t-1}) \right\} \, dx_{t-1}
\]

\[
\overline{\text{bel}}(x_t) = \left\{ \begin{array}{l}
\overline{\mu}_t = A_t \mu_{t-1} + B_t u_t \\
\overline{\Sigma}_t = A_t \Sigma_{t-1} A_t^T + R_t
\end{array} \right.
\]

Move the part of the exponent that is independent of \(x_{t-1}\) out of the integral and subsume integral in normalizer. Find new \(\mu\) and \(\Sigma\) from 1\textsuperscript{st} and 2\textsuperscript{nd} derivative of quadratic function in exponent (as before).

Whole derivation given, e.g., in Probabilistic Robotics by Thrun
Kalman filter algorithm (same structure as Bayes filter)

1. Kalman_filter($\mu_{t-1}, \Sigma_{t-1}, u_t, z_t$):

2. Prediction/time update:
   3. $\bar{\mu}_t = A_t \mu_{t-1} + B_t u_t$
   4. $\bar{\Sigma}_t = A_t \Sigma_{t-1} A_t^T + R_t$

5. Correction/measurement update:
   6. $K_t = \bar{\Sigma}_t C_t^T (C_t \bar{\Sigma}_t C_t^T + Q_t)^{-1}$
   7. $\hat{\mu}_t = \bar{\mu}_t + K_t (z_t - C_t \bar{\mu}_t)$
   8. $\Sigma_t = (I - K_t C_t) \bar{\Sigma}_t$
   9. return $\mu_t, \Sigma_t$
Correction/Measurement update

\[ \text{bel}(x_t) = \begin{cases} \mu_t = \bar{\mu}_t + K_t(z_t - C_t \bar{\mu}_t) \\ \Sigma_t = (I - K_t C_t) \bar{\Sigma}_t \end{cases} \]

with

\[ K_t = \bar{\Sigma}_t C_t^T (C_t \bar{\Sigma}_t C_t^T + Q_t)^{-1} \]

Kalman gain:
how much the innovation is taken into account ➔ minimizes the posterior state covariance

Covariance decrease

Innovation

Innovation covariance
Kalman filter illustration (2D)

- Notation differs throughout literature

- In the following illustration by Kevin Smith:
  - Motion model without control input (often not given in visual tracking)
  - State covariance, measurement and process noise covariances are denoted by different symbols
  - Often used notation:
    - \( \hat{X}_{t|t-1}, P_{t|t-1} \)
      - \( \hat{X} \) indicates estimate (rather than true state)
      - \( P \) often used to denote state covariance
      - Estimate at time \( t \) given \( t-1 \) (before including the measurement)
Kalman filter example

- **State vector**

\[ x_t = \begin{pmatrix} x \\ y \\ \dot{x} \\ \dot{y} \end{pmatrix} \]

- **Measurement**

\[ z_t = \begin{pmatrix} x \\ y \end{pmatrix} \]

\[ x_{0|0}, P_{0|0} \]
Kalman filter: initial conditions

- Initial state

\[
\begin{pmatrix}
  x_0 \\
  y_0 \\
  \dot{x}_0 \\
  \dot{y}_0
\end{pmatrix}
\]

\[
\begin{pmatrix}
  L \\
  L \\
  L \\
  L
\end{pmatrix}
\]

\[
x_{0|0}, P_{0|0}
\]
Kalman filter: predict mean (const. velocity model)

- Prediction from the motion model (here without control)
- Update the mean

\[ \hat{\mathbf{x}}_{t|t-1} = \mathbf{F}_t \hat{\mathbf{x}}_{t-1|t-1} \]

- State transition matrix

\[ \mathbf{F}_t = \begin{pmatrix} 1 & \Delta t \\ 1 & \Delta t \\ 1 & 1 \end{pmatrix} \]
Kalman filter: predict covariance

\[ P_{1|0} = F_1 P_{0|0} F_1^T + Q_0 \]

- Prediction from the motion model (here without control)

- Update covariance

\[ P_{t|t-1} = F_t P_{t-1|t-1} F_t^T + Q_{t-1} \]
Kalman filter: predict covariance

- If we would only predict

\[ x_{0|0}, P_{0|0} \]

Predict: \[ \hat{x}_{t|t-1}, P_{t|t-1} \]

Correct: \[ x_{t|t}, P_{t|t} \]
Kalman filter: measurement

- Receive a noisy measurement (observation)
  \[ z_1 = \begin{pmatrix} x \\ y \end{pmatrix} \]
- Measurement matrix
  \[ H_t = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \]
Kalman filter: predicted measurement

- Predicted measurement

\[ z_p = H_t x_t \]

\[
\begin{pmatrix}
  x_p \\
  y_p \\
\end{pmatrix} =
\begin{pmatrix}
  1 & 0 & 0 & 0 \\
  0 & 1 & 0 & 0 \\
\end{pmatrix}
\begin{pmatrix}
  x \\
  y \\
  \dot{x} \\
  \dot{y} \\
\end{pmatrix}
\]

\[ z_p, R_p \]

\[ z_t, R \]
Kalman filter: innovation

- Innovation (residual) and innovation covariance:

\[ \tilde{y}_t = z_t - H_t \hat{x}_{t|t-1} \]

\[ S_t = H_t P_{t|t-1} H_t^T + R_t \]

\[ x_{0|0}, P_{0|0} \]

\[ \hat{x}_{t|t-1}, P_{t|t-1} \]

\[ z_t, R \]

\[ \tilde{y}_t, S_t \]

predict

\[ \begin{align*}
\tilde{y}_t &= z_t - H_t \hat{x}_{t|t-1} \\
S_t &= H_t P_{t|t-1} H_t^T + R_t
\end{align*} \]

correct
Kalman gain specifies how much the correction considers the prediction or the measurement.

\[
K_t = P_{t|t-1}H_t^T S_t^{-1}
\]
Kalman filter: measurement update

- Correct the prediction using the measurement

\[
\hat{x}_{t|t} = \hat{x}_{t|t-1} + K_t \tilde{y}_k
\]

\[
P_{t|t} = (I - K_t H_t) P_{t|t-1}
\]

Includes innovation covariance

Predicted covariance

State prediction

Innovation

\(x_{0|0}, P_{0|0}\)

\(\hat{x}_{t|t-1}, P_{t|t-1}\)

Predict

Correct
- Predict, measure, correct cycle iteratively estimates the state at each time step

\[ x_{0|0}, P_{0|0}, \hat{x}_{t|t-1}, P_{t|t-1}, z_t, R, \tilde{y}_t, S_t \]
Summary: Kalman filter

- **Pros 😊**
  - Efficient
  - Gaussian densities are easy to work with
  - Optimal solution for linear Gaussian systems (minimal variance)
  - Well established method

- **Cons 😞**
  - Restricted to Gaussian densities
  - Uni-modal distribution: single hypothesis
  - Only linear, continuous models

- **Readings:**
**General purpose (translational) motion models**

- **Unconstrained models without control input** (holds for \( n = 1 \ldots 3D \) case)

- **Notation:** \( x, p, v, a, j, \varepsilon \to n \)-vectors and \( I, 0, T := \Delta t^2 \frac{2}{2} I, T^3 := \Delta t^3 \frac{3}{3} I \to n \times n \) diagonal matrices

<table>
<thead>
<tr>
<th>State</th>
<th>Motion model</th>
<th>Assumptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x = p )</td>
<td>( x_{t+\Delta t} = x_t + T \ell_t )</td>
<td>(also Brownian motion)</td>
</tr>
<tr>
<td>( \ell )</td>
<td>( \ell_{t+\Delta t} = (T^2 \ell_t^T \ell_t) \ell_t )</td>
<td>Const. position, white noise velocity ( \ell_t = v_t )</td>
</tr>
<tr>
<td>( \ell )</td>
<td>( \ell_{t+\Delta t} = \left( \begin{array}{c} 0 \ 1 \ 0 \ 0 \end{array} \right) \ell_t + \left( \begin{array}{c} T^3 \ T^2 \ T^3 \ T^3 \end{array} \right) \ell_t )</td>
<td>Const. acceleration, white noise jerk ( \ell_t = j_t )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>State</th>
<th>Motion model</th>
<th>Assumptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x = p )</td>
<td>( x_{t+\Delta t} = x_t + T \ell_t )</td>
<td>(also Brownian motion)</td>
</tr>
<tr>
<td>( \ell )</td>
<td>( \ell_{t+\Delta t} = \left( \begin{array}{c} 0 \ 1 \ 0 \ 0 \end{array} \right) \ell_t + \left( \begin{array}{c} T^3 \ T^2 \ T^3 \ T^3 \end{array} \right) \ell_t )</td>
<td>Const. position, white noise velocity ( \ell_t = v_t )</td>
</tr>
<tr>
<td>( \ell )</td>
<td>( \ell_{t+\Delta t} = \left( \begin{array}{c} 0 \ 1 \ 0 \ 0 \end{array} \right) \ell_t + \left( \begin{array}{c} T^3 \ T^2 \ T^3 \ T^3 \end{array} \right) \ell_t )</td>
<td>Const. acceleration, white noise jerk ( \ell_t = j_t )</td>
</tr>
</tbody>
</table>
### General purpose (translational) motion models

<table>
<thead>
<tr>
<th>State</th>
<th>Motion model</th>
<th>Assumptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathbb{F}$</td>
<td>$x_{t+\Delta t} = x_t + T(u_t + \epsilon_t)$</td>
<td>Const. velocity, velocity as noisy input $v_t = u_t + \epsilon_t$</td>
</tr>
<tr>
<td>$\mathbb{F}$</td>
<td>$x_{t+\Delta t} = \left(\begin{array}{c} \dot{x}_t \ v_t \end{array}\right) + \left(\begin{array}{c} T^2 \ T \end{array}\right) \left(\begin{array}{c} u_t + \epsilon_t \ \epsilon_t \end{array}\right)$</td>
<td>Const. acceleration, acceleration as noisy input $a_t = u_t + \epsilon_t$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>State</th>
<th>Motion model</th>
<th>Assumptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathbb{F}$</td>
<td>$x_{t+\Delta t} = x_t + T(u_t + \epsilon_t)$</td>
<td>Const. velocity, velocity as noisy input $v_t = u_t + \epsilon_t$</td>
</tr>
<tr>
<td>$\mathbb{F}$</td>
<td>$x_{t+\Delta t} = \left(\begin{array}{c} \dot{x}_t \ v_t \end{array}\right) + \left(\begin{array}{c} T^2 \ T \end{array}\right) \left(\begin{array}{c} u_t + \epsilon_t \ \epsilon_t \end{array}\right)$</td>
<td>Const. acceleration, acceleration as noisy input $a_t = u_t + \epsilon_t$</td>
</tr>
</tbody>
</table>

- ☺ **Smaller state, more efficient**
- ☹ **Control input enters estimate immediately, not filtered**
General purpose (translational) motion models

Ballistic trajectory (gravity modelled as noisy control input)

- The model does not have to be physically motivated
- Basically, we are free to model whatever we want

Constrained Brownian motion (in 1D)
Thank you!