Bachelor Seminar : Kalman Filter

Overview []

1. Introduction
   - Examples?
2. g-h-Filter
3. HMM
4. Kalman-Filter
5. Outlook: Extended / Unscented Kalman Filter
6. End.

Introduction []

- Assume we track sth from above (2D)
- vel+pos -> predict []
- now we get new sensor data - what to do? []
- result: weighted mean value []
- Concept: Combine inaccurate measures and worldmodel-based predictions to achieve better accuracy.
- later: variance / imprecision part of the model []
- result: new mean & variance []

G-H-Filter []

- this mechanism is the gh-filter
- State variables: []
  - $\hat{x}_k$ like position
  - $\hat{v}_k$ like velocity
- Input:
  - Measures x_k
- World Model:
  - constant velocity
- **Predict**: $\hat{x}_k$ in Δt becomes $\hat{x}_k = \hat{x}_{k-1} + \Delta t \cdot \hat{v}_{k-1}$ []
- The Update-step with assumedly inaccurate measure x_k uses the **Residual** $\hat{r}_k = x_k - \hat{x}_k$ scaled by **Parameters** g **and** h to correct predictions: []
  - $\hat{x}_k = \hat{x}_{k-1} + \Delta t \cdot \hat{v}_{k-1} + g \cdot \hat{r}_k$
  - $\hat{v}_k = \hat{v}_{k-1} + \frac{h}{\Delta t} \cdot \hat{r}_k$

Notes []

- G-H-Filter is also used to *predict*: after update comes next predict already, with time-constant used.
  - many sensors (radar, etc) measure in constant intervals.

Choice of g and h []

- [] Small g, small(no) h -> noise reduced, but lagging
- [] Large g, no h -> no noise reduction.. no filtering
- [] no g, larger h -> total divergence
- [] decent choices -> decent tracking and great noise reduction
- [] last choice = Benedict-Bordner. Reduces transient error given g

Disadvantages

- Acceleration not part of the model

HMM

Markov Model (probability theory)

- stochastic model for randomly changing systems

Markov Chain []

for fully observable systems

- State-Space S : **countable** and often finite
- Observed Variable $x_k \in S$ at step k
- Markov Chain = chain $x_k \in S, k \in \{1, 2, ...\}$
- [] [] **markov property** : future state(s) depend only on current
- Instead of hidden state variable, the probability for each state is stored
- [] Transition-probabilities: Matrix F_k between states
  - per Timestep k
  - or globally, if *time-homogeneous*
- If S is finite -> directed graph, nodes in S, edges the probability
  - simple **Dynamic Bayesian Network**
- [] **EXAMPLE** [] [] [] []
- other example: PageRank algorithm by Google

Hidden Markov Model []

for partially observable systems!

- Not the state variables $x \in S$ are observable, just **output tokens** z that depend on the state []
  - [] **emission probabilities**: probability-dist of observed variable (tokens) over hidden states $p(z|x)$
    - interesting for filtering: **a-posteriori** - $p(x|z)$ get state from emission
  - sequence of tokens/emissions gives only **some** information.

examples

- pattern recognition (speech, handwriting, gestures)
- reinforcement learning
- bioinformatics

Kalman-Filter

Theory

From HMM []

- [] The model of the Kalman-Filter is closely related to the HMM
- [] only in continuous space and adding **control** and **process noise**
- state now as Propability Distributions with mean x and (co)variance P
- state transition now operates on mean statespace directly, not on probabilities of states

From G-H-Filter []

- now uses *one* state vector for all
- allows *any* linear operation on state vars
- adds **control** vector & model

- adds **observation-model** - doesn't assume observations of same unit as state
- g and h are now time-dependent and functions of (co)variances of both measurement and prior state -> now called **kalman-gain** : scales each component of the innovation individually

World-Model []

- The **Prediction-Matrix (State-transition model)** $F_k \in \mathbb{R}^{n\times n}$
  - converts $\hat{x}_k$ to $\hat{x}_{k+1}$. This can be used to realize position & velocity as in the g-h-filter, but also *Acceleration* or rather derivations of any degree.
  - The Prediction-Matrix also converts P_k to P_{k+1}
- **Control Vector** u_k : deterministic and known influence on the state
- The **Control-Matrix (Control-input model)** B_k converts the control-vector u_k to the units of the state variable.
  - *z.B. Voltagelevel on Engine leading to increased velocity*
- Some **Process Noise** is assumed in every step. mean 0, cov Q_k. Only cov used

Observation-Model []

- **Measurement / Sensor Reading** z_k (generally not of the same unit as x_k)
- The **Observation model** H_k converts the state-variable to the units of the measurement-vector (in general the sensors use different units than the kalman filter's state model)
  - *For example a gps measures position but not velocity, some in feet others in meters…*
- **Observation noise** = imprecision of sensors with **Covariance** R_k, only this used

[] (diagram with formulas)

Kalman Filter Process

0. Past state []

1. Prediction []

- Changes estimated filter state analog to assumed real change

2. Update []

Innovation (Residual)

- mean (in observe-space) - observation minus assumed obs from prediction
- cov (in observe-space) - sensorCov minus assumed cov from prediction
- [] The **Kalman-Gain** K_k scales the Residual in proportion to the prediction **like g and h** only dynamically, proportional to increase in precision ($cov^{-1}) by new sensor readings

New best estimates []

- new mean analog to g-h-filter only with kalman gain
- new covariance

Overview of Variables in Diagram []

1-D example: In 1-D with state estimate $(x_{k|k-1}, Var(x_{k|k-1}))$ comes measurement $(y_k, Var(y_k))$. Weigh both with the **inverse Variance** (higher Var -> lower Precision) to recieve:

Notes

- Kalman Filter is a common **Sensor Fusion** Algorithm
- Kalman Filter is optimal linear filter, assumed
  1. the model perfectly matches the real system
  2. the entering noise is white (uncorrelated)
  3. the covariances of the noise are exactly known