CDS 110b: Lecture 8-2
KF Extensions and Applications

Richard M. Murray

27 February 2008

Goals

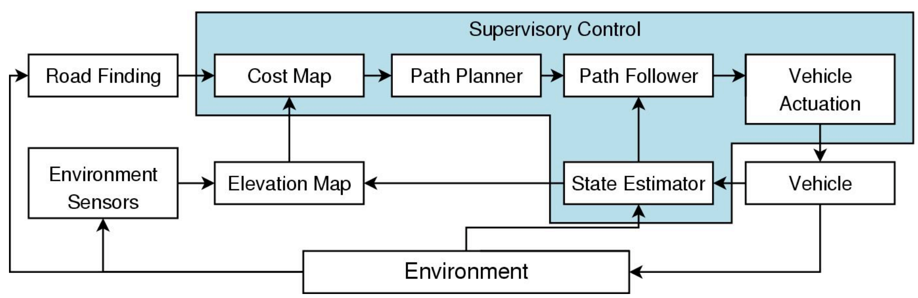
- Describe variations and examples of sensor fusion: Alice + information form
- Describe nonlinear, non-Gaussian extensions to KF: MHE + particle filters

Reading (optional)

- Cremean et al, "Alice: A Networked Control System for Autonomous Desert Driving", *J. Field Robotics,* 2006. (Available at http://gc.caltech.edu/public)
- Ben Grochalsky's thesis on information filter (link on web page)
- CDS 270-2 web page (MHE, particle filters)

Cremean et al, 2006
J. Field Robotics

Application: Autonomous Driving



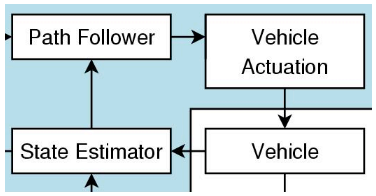
Computing

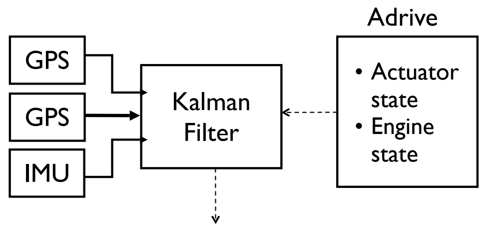
- 6 Dell 750 PowerEdge Servers (P4, 3GHz)
- 1 IBM Quad Core AMD64 (fast!)
- 1 Gb/s switched ethernet

Sensing

- 5 cameras: 2 stereo pairs, roadfinding
- 5 LADARs: long, med*2, short, bumper
- 2 GPS units + 1 IMU (LN 200)
- 0.5-1 Gb/s raw data rates

State Estimation





Vehicle position, orientation, velocities, accelerations

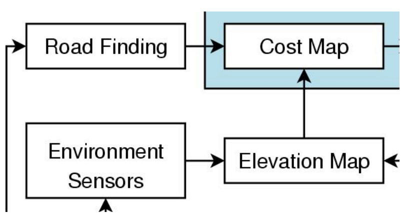
State estimation: astate

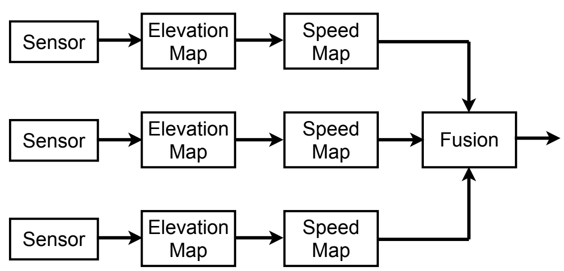
- Broadcast current vehicle state to all modules that require it (many)
- Timing of state signal is critical - use to calibrate sensor readings
- Quality of state estimate is critical: use to place terrain features in global map
- Issue: GPS jumps
  - Can get 20-100 cm jumps as satellites change positions
  - Maintain continuity of state at same time as insuring best accuracy

Astate

- HW: 2 GPS units (2-10 Hz update), 1 inertial measurement unit (gryo, accel @ 400 Hz)
- In: actuator commands, actuator values, engine state
- Out: time-tagged position, orientation, velocities, accelerations
- Use vehicle wheel speed + brake command/position to check if at rest

Terrain Estimation





Sensor processing

- Construct local elevation based on measurements and state estimate
- Compute speed based on gradients

Sensor fusion

- Combine individual speed maps
- Process “missing data” cells

Road finding

- Identify regions with road features
- Increase allowable speed along roads

LadarFeeder, StereoFeeder

- HW: LADAR (serial), stereo (firewire)
- In: Vehicle state
- Out: Speed map (deltas)
- Multiple computers to maintain speed

FusionMapper

- In: Sensor speed maps (deltas)
- Output: fused speed map
- Run on quadcore AMD64

Example: Kalman Filtering for Terrain (Gillula)

KF Framework:
- State to estimate is elevation of each cell
- Elevation is static – so no time updates!

Kalman Filtering:

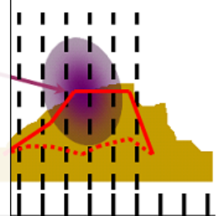
Propagation Equations:

$$\begin{aligned}\hat{z}_{i,j}(k+1|k) &= \hat{z}_{i,j}(k|k)\\ P_{i,j}(k+1|k) &= P_{i,j}(k|k)\end{aligned}$$

Update Equations:

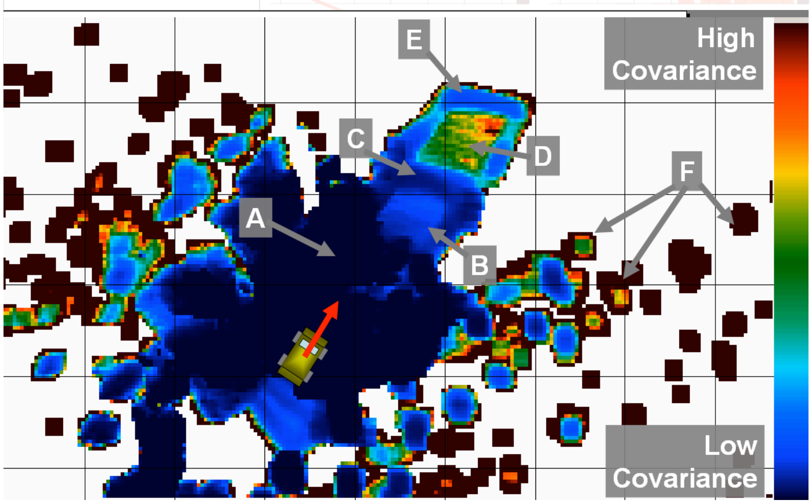
$$\hat{z}_{i,j}(k+1|k+1) = \frac{R\hat{z}_{i,j}(k+1|k) + P_{i,j}(k+1|k)z_m}{P_{i,j}(k+1|k)+R}$$

$$P_{i,j}(k+1|k+1) = \frac{P_{i,j}(k+1|k)R}{P_{i,j}(k+1|k)+R}$$



The Results – Accurate Elevation and Covariance

Elevation Map:
- Individual sensors
- Fused map



Covariance Map:

A: All sensors

B: Just LADARs

C: LADAR in place

D: Long-range LADAR

E: LADAR in place

F: Sparse measurements

Extension: Information Filter

Idea: rewrite Kalman filter in terms of inverse covariance

$$
\begin{aligned}
I[k|k] &:= P^{-1}[k|k], & \hat{Z}[k|k] &:= P^{-1}[k|k]\hat{X}[k|k] \\
\Omega_i[k] &:= C_i^T R_{W_i}^{-1}[k] C_i, & \Psi_i[k] &:= C_i^T R_{W_i}^{-1}[k] C_i \hat{X}[k|k]
\end{aligned}
$$

Resulting update equations become *linear*:

$$
\begin{aligned}
\hat{X}[k|k-1] &= (1 - \Gamma[k]F^T)A^{-T}\hat{X}[k-1|k-1] + I[k|k-1]Bu \\
I[k|k-1] &= M[k] - \Gamma[k]\Sigma[k]\Gamma^T[k]
\end{aligned}
$$

$$
\boxed{
\begin{aligned}
I[k|k] &= I[k|k-1] + \sum_{i=1}^{q} \Omega_i[k] \\
\hat{Z}[k|k] &= \hat{Z}[k|k-1] + \sum_{i=1}^{q} \Psi_i[k]
\end{aligned}
}
\qquad
\begin{aligned}
M[k] &= A^{-T}P^{-1}[k-1|k-1]A^{-1} \\
\Gamma[k] &= M[k]F\sigma^{-1}[k] \\
\Sigma[k] &= F^T M[k] F + R_v^{-1}
\end{aligned}
$$

Remarks

- Information form allows simple addition for correction step: “additional measurements add information”
- Sensor fusion: each additional sensor increases the information
- Multi-rate sensing: whenever new information arrives, add it to the scaled estimate, information matrix; no date => prediction update only
- Derivation of the information filter is non-trivial; not easy to derive from Kalman filter

Henrik Sandberg, 2005

Extension: Moving Horizon Estimation

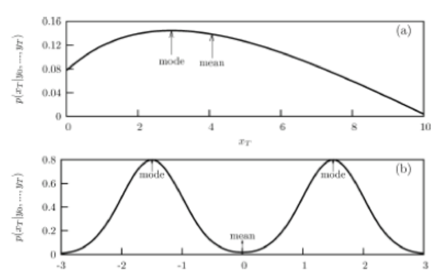
System description:

$$
\begin{aligned}
x_{k+1} &= f_k(x_k, w_k) \\
y_k &= h_k(x_k) + v_k
\end{aligned}
\qquad x_k \in \mathbb{X}_k, \quad w_k \in \mathbb{W}_k, \quad v_k \in \mathbb{V}_k.
$$

The problem: *Given* the data

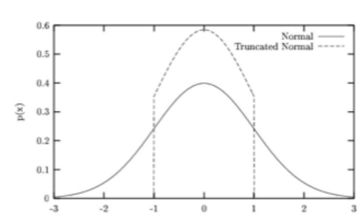
$$Y_k = \{y_i : 0 \le i \le k\},$$

find the “best” (to be defined) estimate $\hat{x}_{k+m}$ of x_{k+m}.
($m = 0$ filtering, $m > 0$ prediction, and $m < 0$ smoothing.



Pose as optimization problem:

$$\{\hat{x}_0, \ldots, \hat{x}_T\} = \arg \max_{\{x_0, \ldots, x_T\}} p(x_0, \ldots, x_T | Y_{T-1})$$



Remarks:

- Basic idea is to compute out the “noise” that is required for data to be consistent with model and penalize noise based on how well it fits its distribution

Extension: Moving Horizon Estimation

Solution: write out probability and maximize

$$
\begin{aligned}
\arg \max_{\{x_0,\ldots,x_T\}} & \; p(x_0,\ldots,x_T \mid y_0,\ldots,y_{T-1}) \\
&= \arg \max_{\{x_0,\ldots,x_T\}} p_{x_0}(x_0) \prod_{k=0}^{T-1} p_{v_k}(y_k - h(x_k)) p(x_{k+1} \mid x_k) \\
&= \arg \max_{\{x_0,\ldots,x_T\}} \sum_{k=0}^{T-1} \log p_{v_k}(y_k - h_k(x_k)) + \log p(x_{k+1} \mid x_k) + \log p_{x_0}(x_0)
\end{aligned}
$$

Special case: Gaussian noise

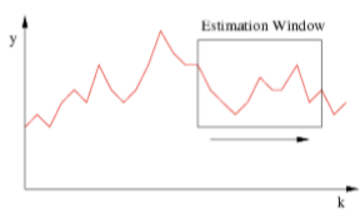
$$
\min_{x_0,\{w_0,\ldots,w_{T-1}\}} \sum_{k=0}^{T-1} \|y_k - h_k(x_k)\|^2_{R_k^{-1}} + \|w_k\|^2_{Q_k^{-1}} + \|x_0 - \bar{x}_0\|^2_{P_0^{-1}}
$$

- Log of the probabilities sum of squares for noise terms
- Note: switched use of w and v from Friedland (and course notes)

Extension: Moving Horizon Estimation

Key idea: estimate over a finite window in the past

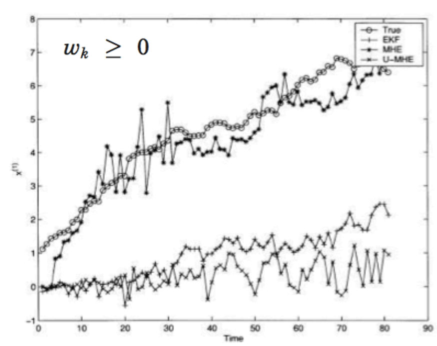
$$
\begin{aligned}
\Phi_T^* &= \min_{x_0,\{w_k\}_{k=0}^{T-1}} \left(\sum_{k=T-N}^{T-1} L_k(w_k, v_k) + \sum_{k=0}^{T-N-1} L_k(w_k, v_k) + \Gamma(x_0) \right) \\
&= \min_{z \in \mathcal{R}_{T-N}, \{w_k\}_{k=T-N}^{T-1}} \left(\sum_{k=T-N}^{T-1} L_k(w_k, v_k) + Z_{T-N}(z) \right).
\end{aligned}
$$



Example (Rao et al, 2003): nonlinear model with positive disturbances

$$
\begin{aligned}
x_{1,k+1} &= 0.99 x_{1,k} + 0.2 x_{2,k} \\
x_{2,k+1} &= -0.1 x_{1,k} + \frac{0.5 x_{2,k}}{1 + x_{2,k}^2} + w_k \\
y_k &= x_{1,k} - 3 x_{2,k} + v_k
\end{aligned}
$$

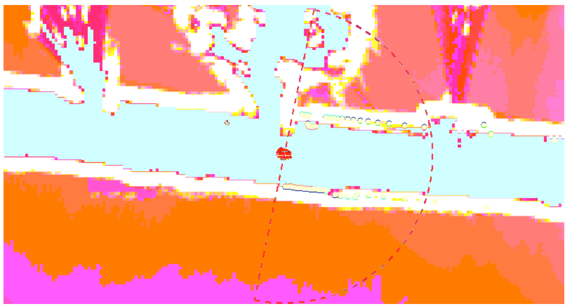
- EKF handles nonlinearity, but assumes noise is zero mean => misses positive drift

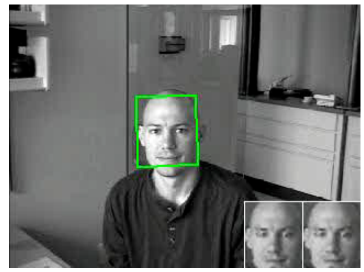


Extension: Particle Filters

Sequential Monte Carlo

- Rough idea: keep track of many possible states of the system via individual "particles"
- Propogate each particle (state estimate + noise) via the system model with noise
- Truncate those particles that are particularly unlikely, redistribute weights

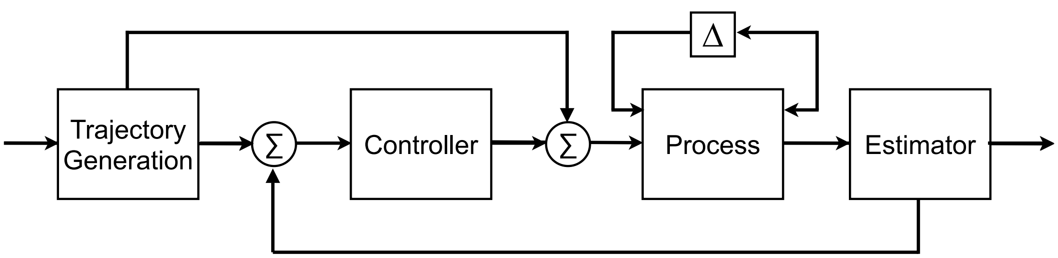




Remarks

- Can handle nonlinear, non-Gaussian processes
- Very computationally intensive; typically need to exploit problem structure
- Being explored in many application areas (eg, SLAM in robotics)
- Lots of current debate about information filters versus MHE versus particle filters

Modern Control System Design



Traditional Control System: controller + process

- Corresponds to "inner loop" of most control system designs

Modern Control System: optimization-based design + robust analysis

- Replace reference with reference trajectory (Weeks 1-4)
- Replace process output with estimated output (Weeks 5-8)
- Replace "inner loop" controller with robust controller (Week 9-10 + CDS 212/213)