# NCS Lecture 5:
# Kalman Filtering and Sensor Fusion


**Richard M. Murray**


**18 March 2008**

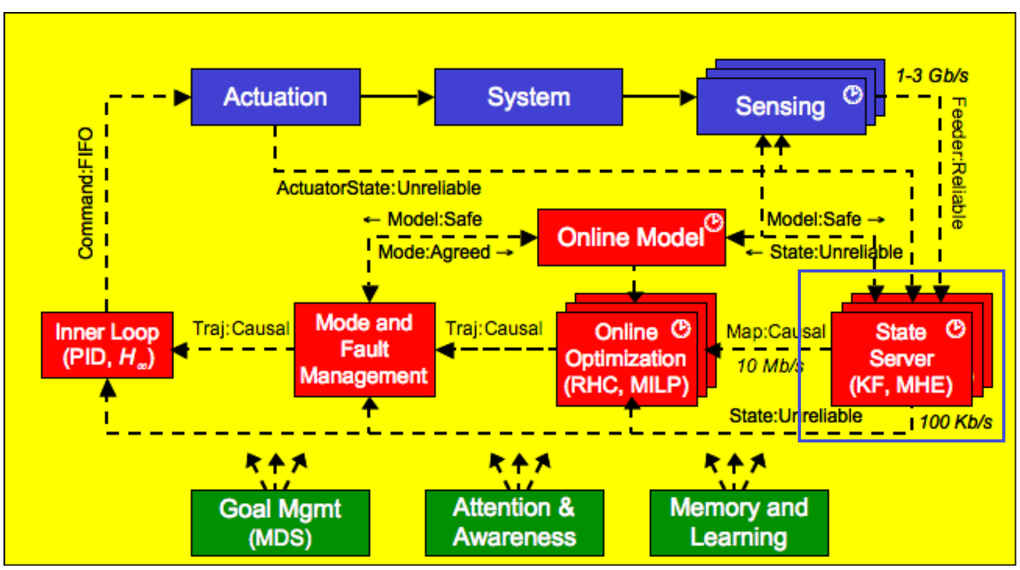
**Goals:**

- Review the Kalman filtering problem for state estimation and sensor fusion
- Describes extensions to KF: information filters, moving horizon estimation

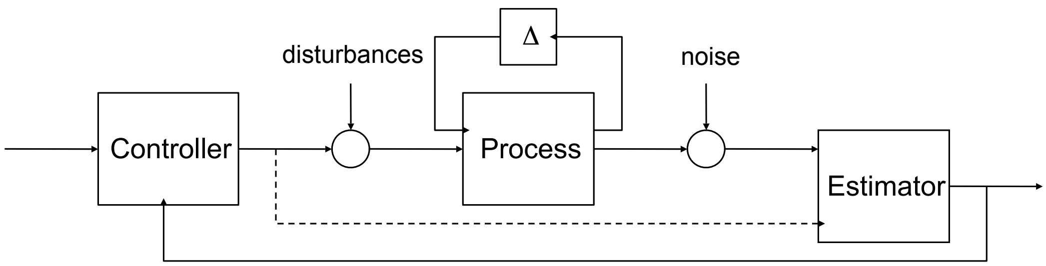
**Reading:**

- OBC08, Chapter 4 - Kalman filtering
- OBC08, Chapter 5 - Sensor fusion

## Networked Control Systems



# The State Estimation Problem




### Problem Setup

- Given a dynamical system with noise and uncertainty, estimate the state

$$\dot{x} = Ax + Bu + Fv \qquad \dot{\hat{x}} = \alpha(\hat{x}, y, u) \quad \leftarrow \text{estimator}$$
$$y = Cx + Du + Gw \quad \longrightarrow \quad \lim_{t\to\infty} E\{x - \hat{x}\} = 0$$

expected value

- $\hat{x}$ is called the *estimate* of $x$

### Discrete-time systems

$$x[k+1] = Ax[k] + Bu[k] + Fv[k]$$
$$y[k] = Cx[k] + w[k],$$

estimator gain

$$\hat{x}[k+1] = \underbrace{A\hat{x}[k] + Bu[k]}_{\text{prediction}} + \underbrace{L(y[k] - C\hat{x}[k])}_{\text{correction}}$$



# Optimal Estimation

### System description

$$x[k+1] = Ax[k] + Bu[k] + Fv[k]$$
$$y[k] = Cx[k] + w[k],$$

$$E\{v[k]\} = 0$$
$$E\{v[k]v[j]^T\} = \begin{cases} 0 & k \neq j \\ R_v & k = j \end{cases}$$

- Disturbances and noise are multi-variable Gaussians with covariance $R_v$, $R_w$

**Problem statement:** Find the estimate that minimizes the mean square error $E\{(x[k] - \hat{x}[k])(x[k] - \hat{x}[k])^T\}$

### Proposition

- For Gaussian noise, optimal estimate is the expectation of the random process $x$ given the *constraint* of the observed output:

$$\hat{x}[k] = E\{X[k] \,|\, Y[l], l \leq k\}$$

- Can think of this as a *least squares* problem: given all previous $y[k]$, find the estimate $\hat{x}[k]$ that satisfies the dynamics and minimizes the square error with the measured data.

# Kalman Filter

**Thm** (Kalman, 1961) The observer gain *L* that minimizes the mean square error is given by

$$L[k] = AP[k]C^T(R_w + CP[k]C^T)^{-1}$$

where

$$P[k+1] = (A - LC)P[k](A - LC)^T + R_v + LR_wL^T$$
$$P_0 = E\{X(0)X^T(0)\}.$$

**Proof** (easy version). Let $P[k] = E\{(\hat{x}[k] - x[k])(\hat{x}[k] - x[k])^T\}$. By definition,

$$P[k+1] = E\{x[k+1]x[k+1]^T\}$$
$$= AP[k]A^T - AP[k]C^TL^T - LCA^T + L(R_w + CP[k]C^T)L^T.$$

Letting $R_\epsilon = (R_w + CP[k]C^T)$,

$$P[k+1] = AP[k]A^T + \big(L - AP[k]C^TR_\epsilon^{-1}\big)R_\epsilon\big(L - AP[k]C^TR_\epsilon^{-1}\big)^T$$
$$- AP[k]C^TR_\epsilon^{-1}CP[k]^TA^T + R_w.$$

to minimize covariance, choose $L = AP[k]C^TR_\epsilon^{-1}$



# Kalman Filtering with Intermittent Data

**Kalman filter has "predictor-corrector" form**

$$\hat{x}[k+1] = A\hat{x}[k] + Bu[k] + L(y[k] - C\hat{x}[k])$$

$$P[k+1] = \underbrace{AP[k]A^T + R_w}_{\text{prediction}} - \underbrace{AP[k]C^TR_\epsilon^{-1}CP[k]^TA^T}_{\text{correction}}$$

- Key idea: updated prediction on each iteration; apply correction when data arrives

**Alternative formulation**

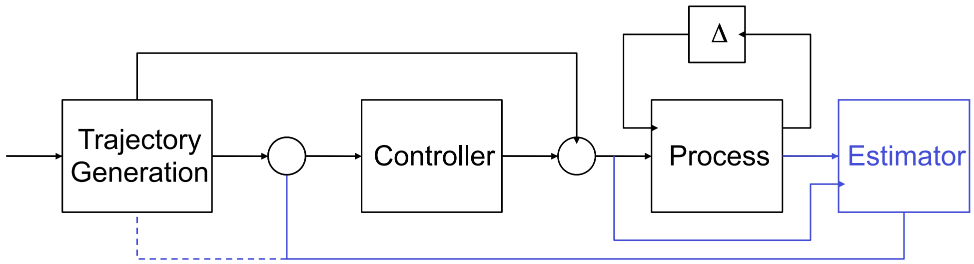
- Prediction:

$$\hat{x}[k+1|k] = A\hat{x}[k|k] + Bu[k]$$
$$P[k+1|k] = AP[k|k]A^T + FR_v[k]F^T$$

- Correction:

$$\hat{x}[k|k] = \hat{x}[k|k-1] + L[k](y[k] - C\hat{x}[k|k-1])$$
$$P[k|k] = P[k|k-1] - P[k|k-1]C^T(CP[k|k-1]C^T + R_w[k])^{-1}CP[k|k-1]$$

# Kalman-Bucy Filter




**System dynamics: linear process + Gaussian white noise**

$$\dot{x} = Ax + Bu + Fv \qquad E\{v(s)v^T(t)\} = Q(t)\delta(t-s)$$
$$y = Cx + w \qquad E\{w(s)w^T(t)\} = R(t)\delta(t-s)$$

Time-varying statistics OK

**Estimator: prediction + correction**

$$\dot{\hat{x}} = A\hat{x} + Bu + L(y - C\hat{x}) \qquad L(t) = P(t)C^T R^{-1}$$
$$P(t) = E\{(x(t) - \hat{x}(t))(x(t) - \hat{x}(t))^T\}$$

**Covariance update**

$$\dot{P} = AP + PA^T - PC^T R^{-1}(t)CP + FQ(t)F^T$$
$$P(0) = E\{x(0)x^T(0)\}$$



# Variation #1: Sensor Fusion

**What happens if we have redundant sensors?**

- Kalman filter "fuses" data measurements according to covariance

$$\dot{\hat{x}} = A\hat{x} + Bu + L(y - C\hat{x}) \qquad L(t) = P(t)C^T R^{-1}$$
$$P(t) = E\{(x(t) - \hat{x}(t))(x(t) - \hat{x}(t))^T\}$$

- Assume R is diagonal, expand out gain:

$$L(t) = P(t)C^T \begin{bmatrix} R_{11}^{-1} & & \\ & \ddots & \\ & & R_{nn}^{-1} \end{bmatrix}$$

$R_{11}$ large $\Rightarrow$ smaller effect

$P$ small $\Rightarrow$ don't rely on sensors (state is accurate)

$$\dot{P} = AP + PA^T - PC^T R^{-1}(t)CP + FQ(t)F^T$$

Process disturbances

$R$ small decreases uncertainty

$P$ evolves according to nominal dynamics

- Steady state (ARE): optimal balance of dynamics and uncertainty

# Variation #2: Extended Kalman Filter (EKF)

Consider a *nonlinear* system

$$\dot{x} = f(x, u, v) \qquad x \in \mathbb{R}^n, u \in \mathbb{R}^m$$
$$y = Cx + w \qquad v, w \text{ Gaussian white noise processes with covariance matrices } Q \text{ and } R.$$

Form estimator using nonlinear model + linear feedback

$$\dot{\hat{x}} = f(\hat{x}, u, 0) + L(y - C\hat{x})$$

Compute estimator gain based on linearization at current estimated state:

$$\dot{\hat{x}} = f(\hat{x}, u, 0) + L(y - C\hat{x}) \qquad L = PC^T R^{-1}$$
$$\dot{P} = (\tilde{A} - LC)P + P(\tilde{A} - LC)^T + \tilde{F}Q\tilde{F}^T + LRL^T \qquad P(t_0) = E\{x(t_0)x^T(t_0)\}$$

$$\tilde{A} = \left.\frac{\partial F}{\partial e}\right|_{(0,\hat{x},u,0)} = \left.\frac{\partial f}{\partial x}\right|_{(\hat{x},u,0)}$$
$$\tilde{F} = \left.\frac{\partial F}{\partial v}\right|_{(0,\hat{x},u,0)} = \left.\frac{\partial f}{\partial v}\right|_{(\hat{x},u,0)}$$

- Little formal theory, but works very well as long as estimated state is close
- Very important for tracking problems (might operate far from equilibrium)

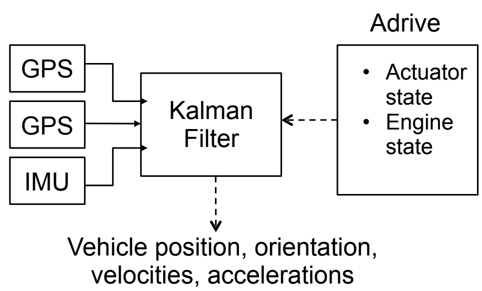


# Example: GPS + IMU localization in Alice

**Nonlinear dynamics (simplified)**

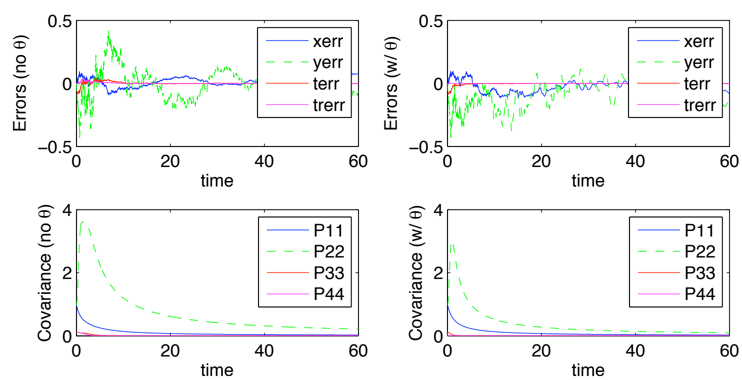
$$\dot{x} = \cos\theta\, v$$
$$\dot{y} = \sin\theta\, v$$
$$\ddot{\theta} = \frac{\dot{v}}{\ell}\tan\phi + \frac{v}{\ell}\frac{\dot{\phi}}{\cos^2\phi}$$

- Measure x, y and $\dot{\theta}$




**Results**

- If only x and y are measured, get larger errors in state estimate
- Adding angular rate measurement improves performance (right)



## Variation #3: Parameter Estimation

Suppose dynamics depend on unknown parameter $\xi$

$$\begin{aligned} \dot{x} &= A(\xi)x + B(\xi)u + Fv \qquad \xi \in \mathbb{R}^p \\ y &= C(\xi)x + w \end{aligned}$$

Rewrite dynamics using added state $\xi$

$$\begin{aligned} \dot{x} &= A(\xi)x + B(\xi)u + Fv \\ \dot{\xi} &= 0 \end{aligned}$$

Now use extended Kalman filter to estimate state and parameter:

$$\frac{d}{dt}\begin{bmatrix} x \\ \xi \end{bmatrix} = \overbrace{\begin{bmatrix} A(\xi) & 0 \\ 0 & 0 \end{bmatrix}\begin{bmatrix} x \\ \xi \end{bmatrix} + \begin{bmatrix} B(\xi) \\ 0 \end{bmatrix} u}^{f(\left[\begin{smallmatrix} x \\ \xi \end{smallmatrix}\right],u,v)} + \begin{bmatrix} F \\ 0 \end{bmatrix} v$$

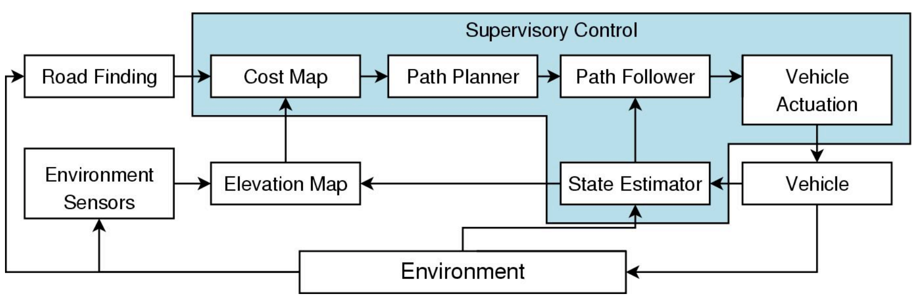
$$y = \underbrace{C(\xi)x + w}_{h(\left[\begin{smallmatrix} x \\ \xi \end{smallmatrix}\right],w)}$$

Example: estimate wheel base, sensor locations on Alice



## Example: Autonomous Driving

Cremean et al, 2006
*J. Field Robotics*



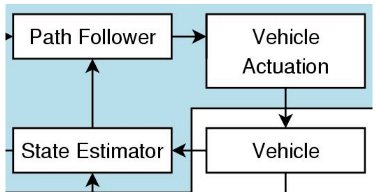

**Computing**

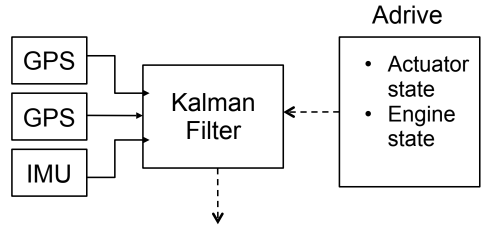
- 6 Dell 750 PowerEdge Servers (P4, 3GHz)
- 1 IBM Quad Core AMD64 (fast!)
- 1 Gb/s switched ethernet


**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**

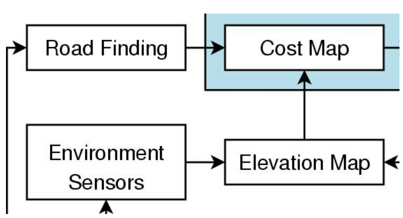
- Broadcast current vehicle state to all modules that require it (many)
- Timing of state signal is critical - use to calibrate sensor readings
- Quality of state estimate is critical: use to place terrain features in global map
- Issue: GPS jumps
  - Can get 20-100 cm jumps as satellites change positions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

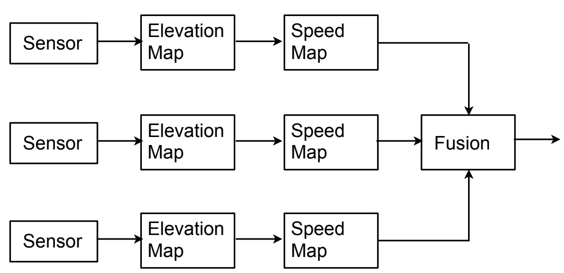
**Astate**

- 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:**

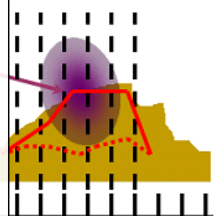
Propagation Equations:

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

Update Equations:

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

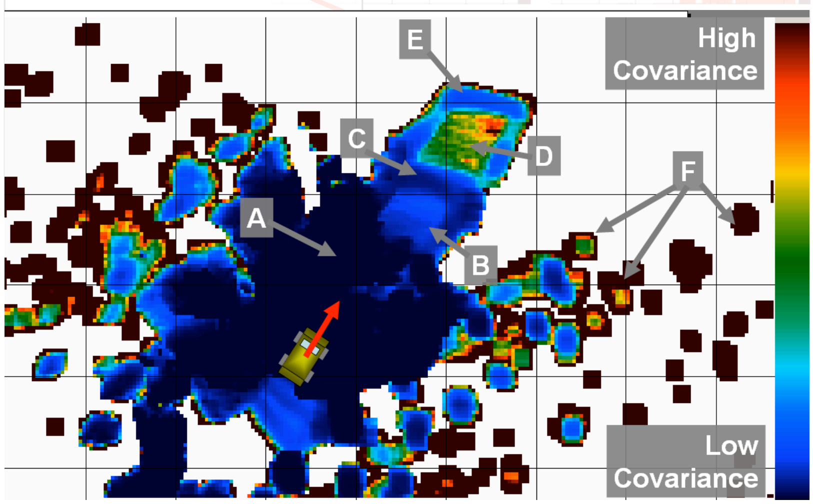
$$P_{i,j}(k+1|k+1) = \frac{P_{i,j}(k+1|k)R}{P_{i,j}(k+1|k) + R}$$





# The Results – Accurate Elevation and Covariance




**Covariance Map:**

**A:** All sensors

**B:** Just LADARs

**C:** LADAR in place

# Extension: Information Filter

**Idea: rewrite Kalman filter in terms of inverse covariance**

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

**Resulting update equations become *linear*:**

$$\hat{X}[k|k-1] = (1 - \Gamma[k]F^T)A^{-T}\hat{X}[k-1|k-1] + I[k|k-1]Bu$$
$$I[k|k-1] = M[k] - \Gamma[k]\Sigma[k]\Gamma^T[k]$$
$$I[k|k] = I[k|k-1] + \sum_{i=1}^{q} \Omega_i[k]$$
$$\hat{Z}[k|k] = \hat{Z}[k|k-1] + \sum_{i=1}^{q} \Psi_i[k]$$

$$M[k] = A^{-T}P^{-1}[k-1|k-1]A^{-1}$$
$$\Gamma[k] = M[k]F\sigma^{-1}[k]$$
$$\Sigma[k] = F^T M[k] F + R_v^{-1}$$

**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

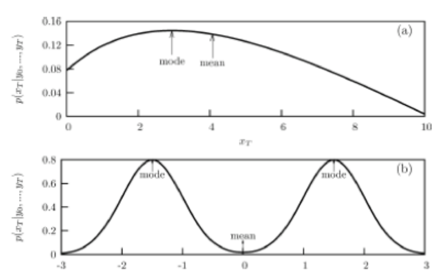
**System description:**

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

**The problem:** *Given* the data

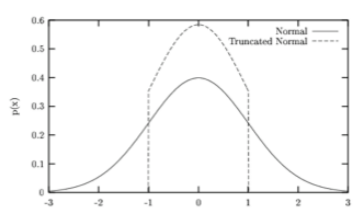
$$Y_k = \{y_i : 0 \le i \le k\},$$

*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**

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

**Special case: Gaussian noise**

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

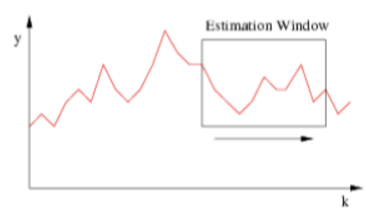
- Log of the probabilities sum of squares for noise terms
- Note: switched use of w and v from Friedland (and course notes)



# Extension: Moving Horizon Estimation

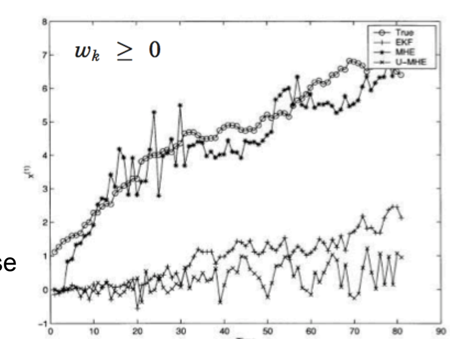
**Key idea: estimate over a finite window in the past**

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




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

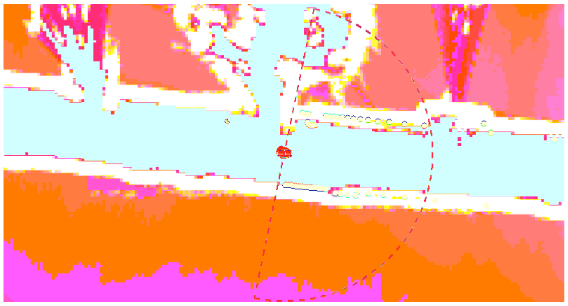
$$
\begin{aligned}
x_{1,k+1} &= 0.99 x_{1,k} + 0.2 x_{2,k} \\
x_{2,k+1} &= -0.1 x_{1,k} + \frac{0.5 x_{2,k}}{1 + x_{2,k}^2} + w_k \\
y_k &= x_{1,k} - 3 x_{2,k} + v_k
\end{aligned}
$$

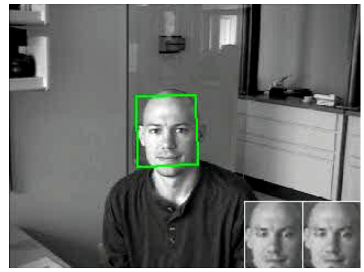



- EKF handles nonlinearity, but assumes noise is zero mean => misses positive drift

# 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



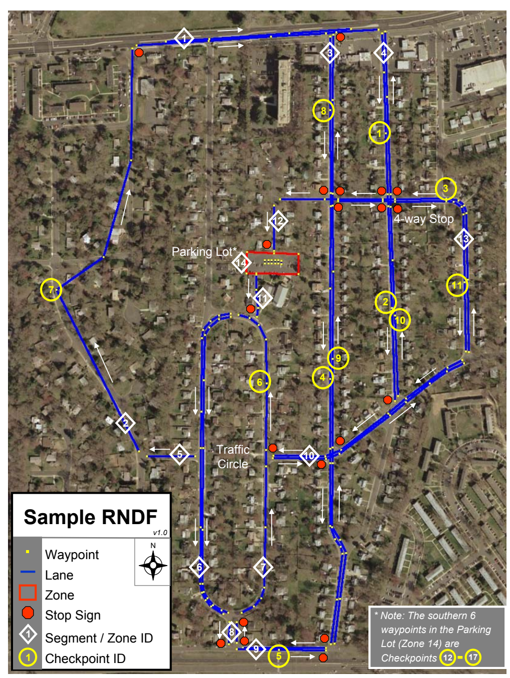


**Remarks**
- Can handle nonlinear, non-Gaussian processes
- Very computationally intensive; typically need to exploit problem structure
- Being explored in many application areas (eg, SLAM in robotics)
- Lots of current debate about information filters versus MHE versus particle filters



# 2007 Urban Challenge - 3 November 2007

**Autonomous Urban Driving**
- 60 mile course, less than 6 hours
- City streets, obeying traffic rules
- Follow cars, maintain safe distance
- Pull around stopped, moving vehicles
- Stop and go through intersections
- Navigate in parking lots (w/ other cars)
- U turns, traffic merges, replanning
- Prizes: $2M, $500K, $250K




## Sensing and Decision Making

**Video from 29 Jun 06 field test**

- Front and side views from Tosin
- Rendered at 320x240, 15 Hz
- Manually synchronize

**Some challenges**

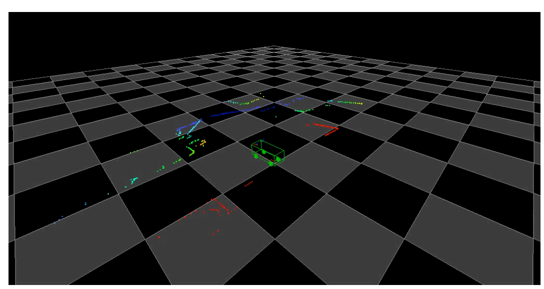
- Moving obstacle detection, separation, tracking and prediction
- Decision-making
- Lane markings (w/ shadows)

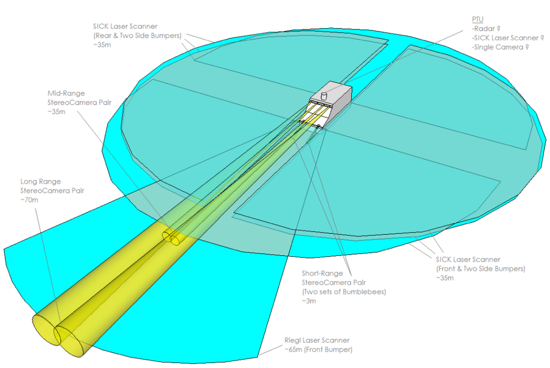


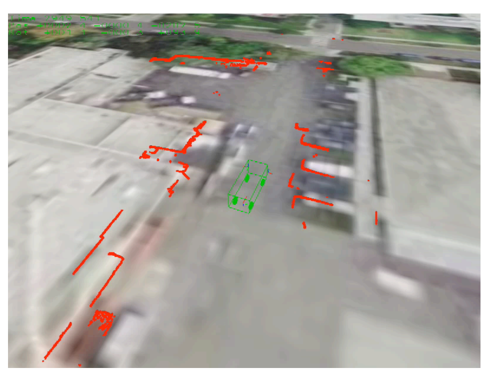
## Sensing System

**Sensing hardware**

- 6 horizontal LADAR (overlapping)
- 1 pushbroom LADAR; 1 sweeping (PTU)
- 3 stereo pairs (color; 640x480 @ ~10 Hz)
- 2 road finding cameras (B&W)
- 2 RADAR units (PTU mounted)
- 10 blade cPCI high speed computing



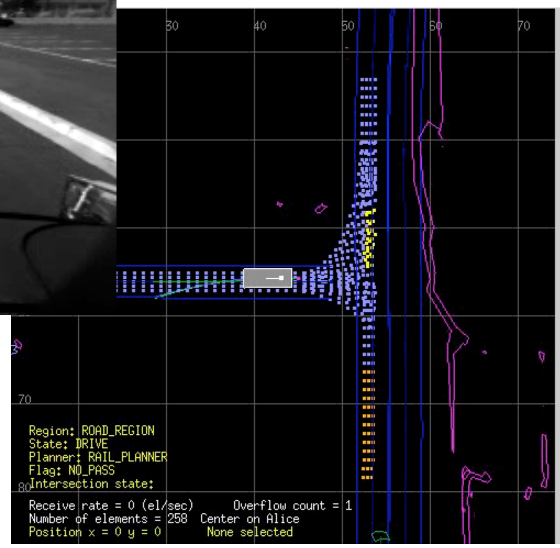


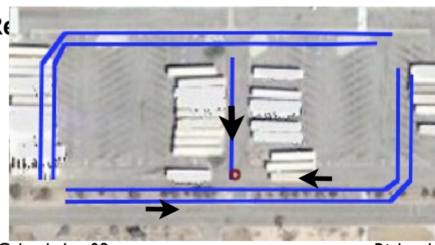



# 2007 National Qualifying Event

**Merging test**

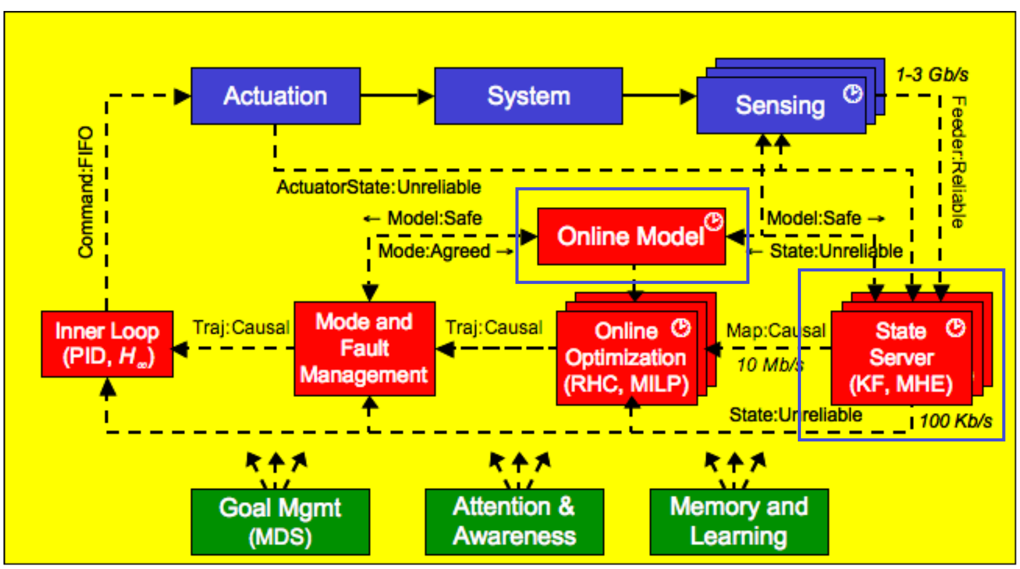
- 10-12 cars circling past inters'n
- Count "perfect runs" in 30 min





# Networked Control Systems




**Next: effects of the *network*...**