CDS 270-2: Lecture 4-3
Road Detection and Tracking

(Application example for Extended Kalman Filtering and Moving Horizon Estimation)

Lars B. Cremean

Fri. 4/21/2006

Overview

Goals

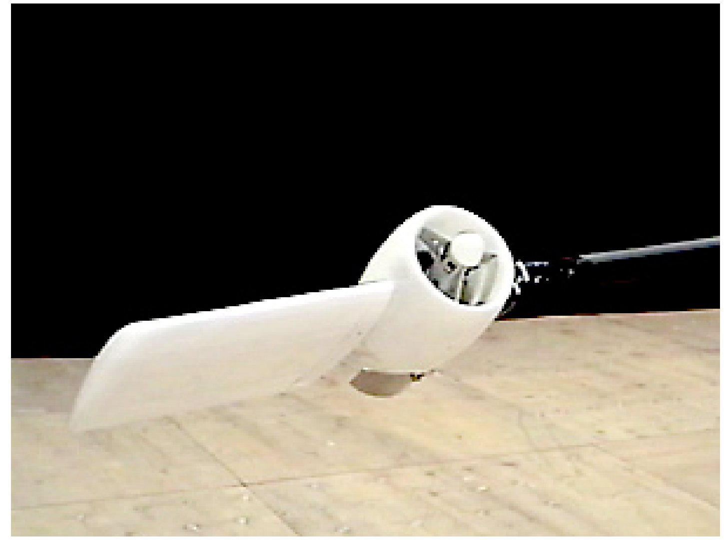
- Motivate use of the extended Kalman filter and moving horizon estimation with a practical example (road tracking).
- Derive implementations of EKF and MHE for road tracking
- Present live and simulated results from application of both approaches

Outline

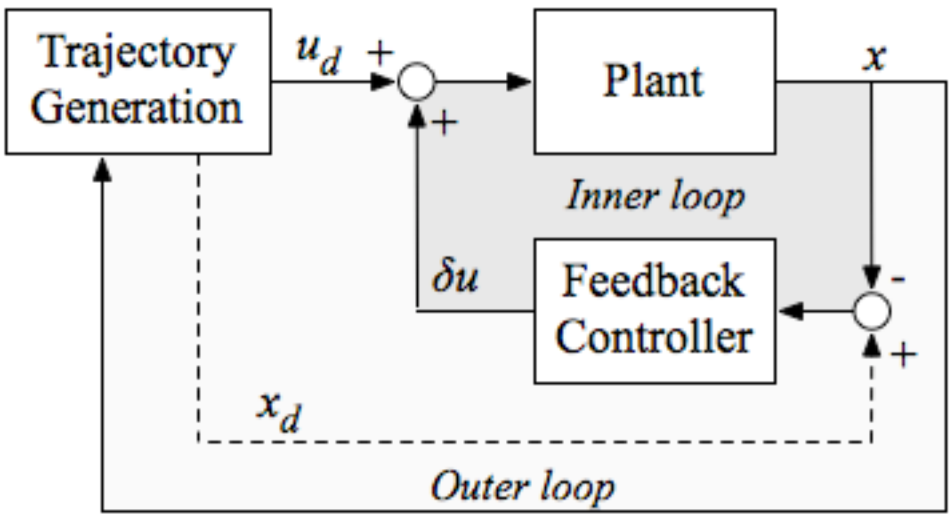
1. Motivation for road tracking
2. Road tracking problem statement / define approach
3. Derive dynamic equations and connect EKF concepts
4. Results: EKF road tracking and following on Alice
5. Present moving horizon estimation alternative
6. MHE implementation and simulation results (preliminary)

Context: nested feedback control

- "Two degree of freedom" controller design (cf. Mark Milam thesis)
- A.k.a. inner-outer loop
- Early application of receding horizon control techniques to fast-dynamics processes (orig. developed in context of chemical processes)



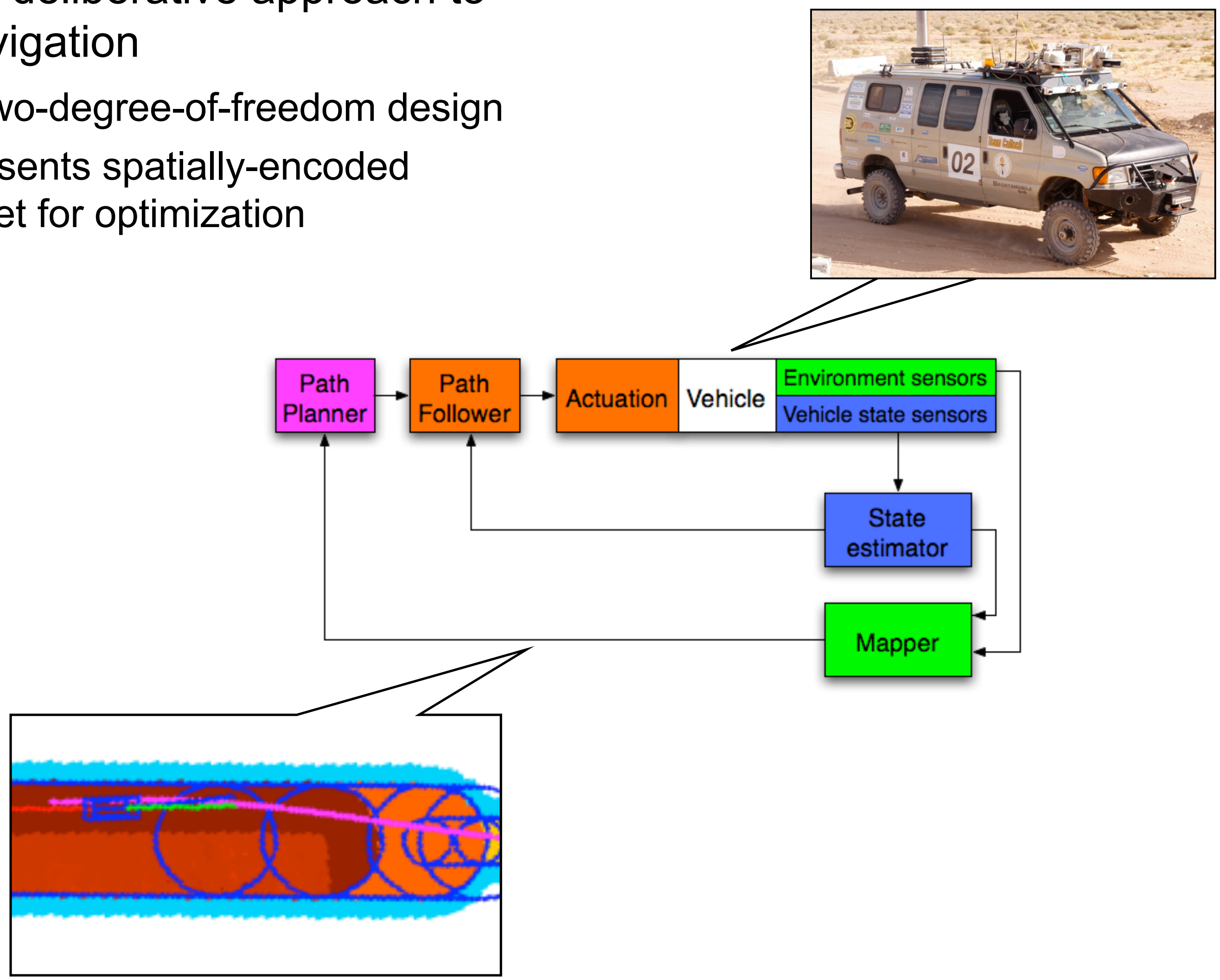
Caltech ducted fan



Context: autonomous navigation

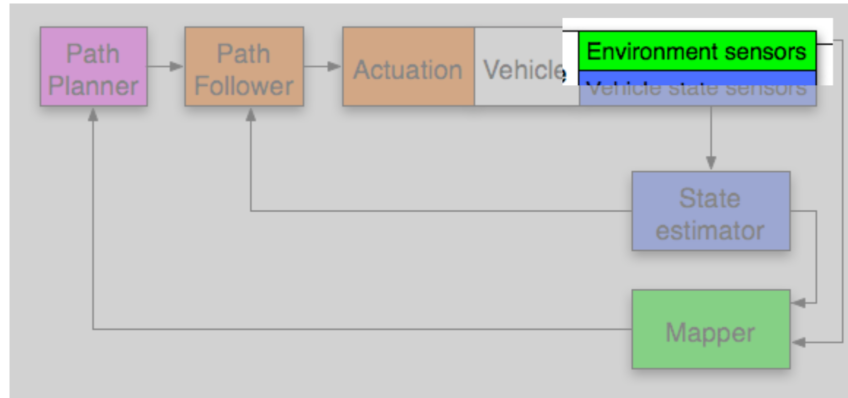
Traditional deliberative approach to robotic navigation

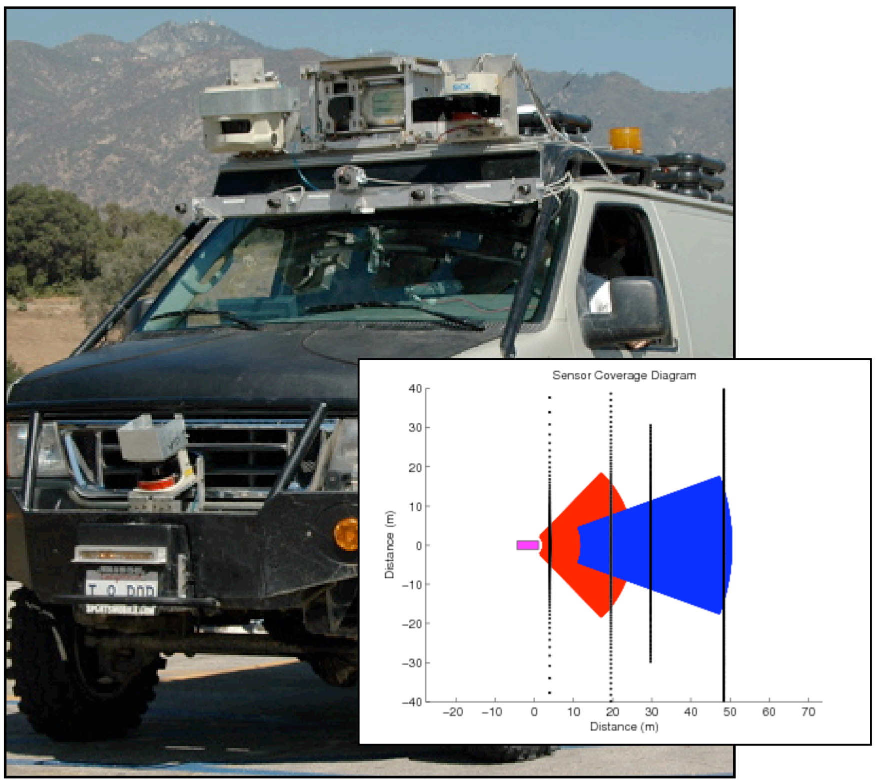
- Follows two-degree-of-freedom design
- Map represents spatially-encoded constraint set for optimization



Sensor suite

- Combination of LADAR and stereovision range sensors used
- All sensors fixed (not gimbaled)





<u>Environment sensors</u>

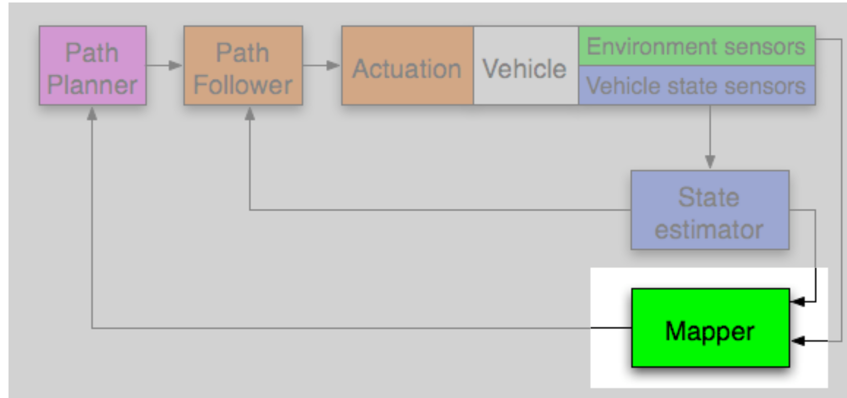
- 5 fixed single-axis LADAR units, oriented horizontally
  - → 4 pitched downward at different amounts for sweeping ground elevation
  - → 1 horizontal for aiding road boundary detection
- 5 fixed Firewire cameras
  - → 2 B/W stereovision pairs
  - → 1 color camera for road detection

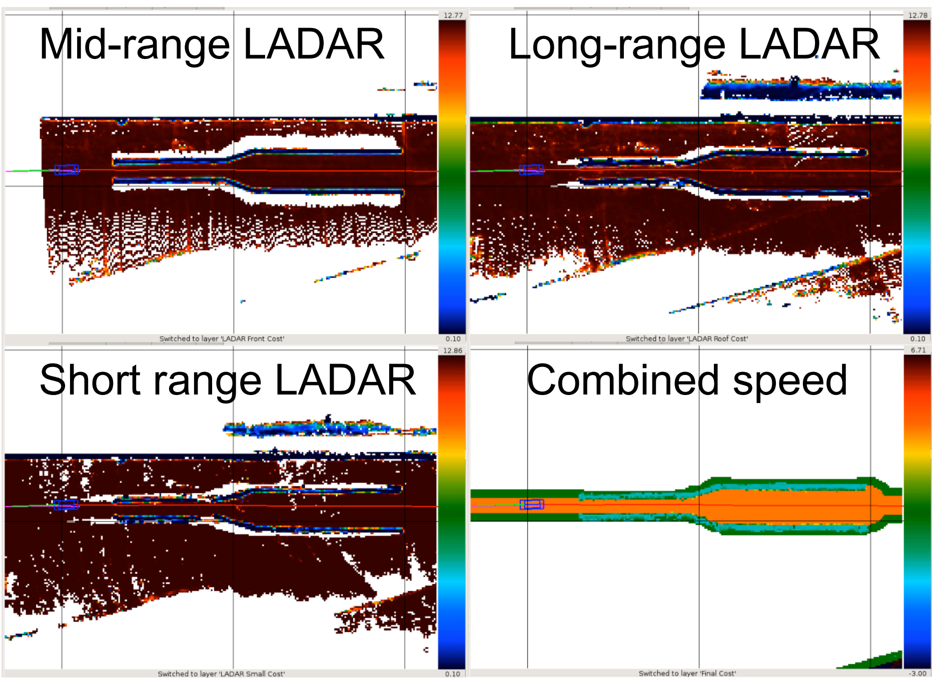
Perception and mapping

Mapping accomplished with layered approach:

1. RDDF (a priori data) layer
2. Elevation-based speed layer
3. Road finding layer (speed bonus)

Layers combined in *speed map*





Goal: Model-based road estimation

- Produce best estimate of parameterized road centerline
- Does not depend on grid-based mapping framework
- Can be converted to map if road boundaries are tracked

Alice: Road finding implementation

- Road following using integrated camera and LADAR data implemented for race
- Tracked vanishing point by grouping of dominant orientations (Rasmussen 2004)
- Boundaries of road estimated from horizontal LADAR and integrated with vanishing point data (Rasmussen 2006)
- Vanishing point updated by particle filter
- Assumes straight line parameterization for road

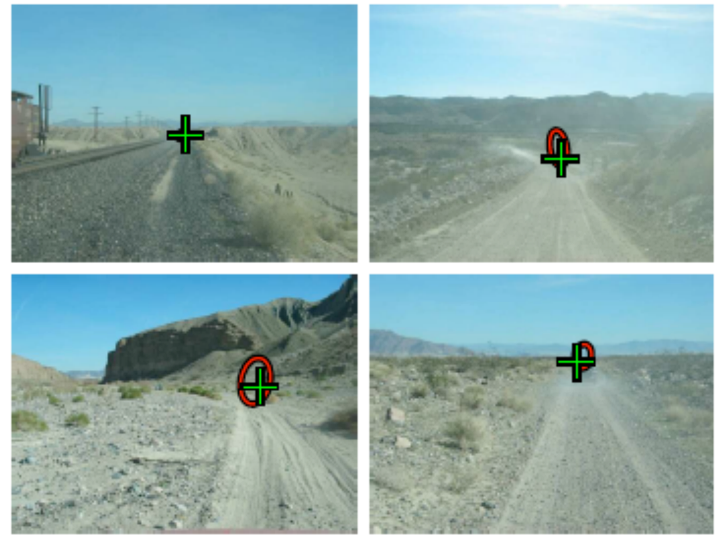
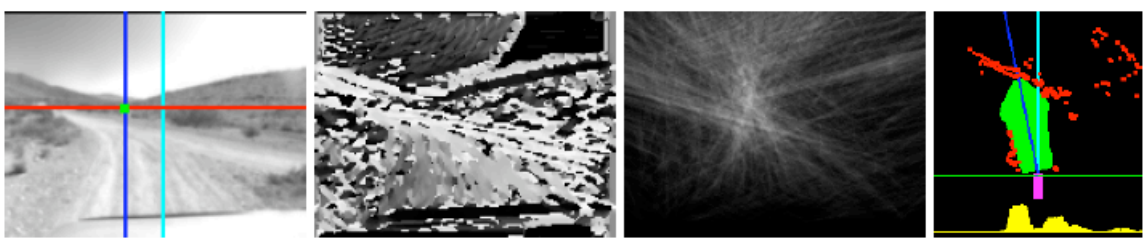


Image from (Rasmussen 2004)



(a) (b) (c) (d)

From (Cremean et al. 2004)

Figure 11: Sample road follower output on test run: (a) Computed road direction & Alice direction (blue, cyan lines respectively–same in (d)); (b) Dominant orientations ($[0, \pi]$ angle proportional to intensity); (c) Vanishing point votes $\mathbf{I}_{VP}$; (d) Overhead view of estimated road region (green area): Alice is purple rectangle, bumper ladar returns are red points, ladar-derived obstacle function $h(x)$ is shown in yellow at bottom.

Road detection, tracking and following

Motivation

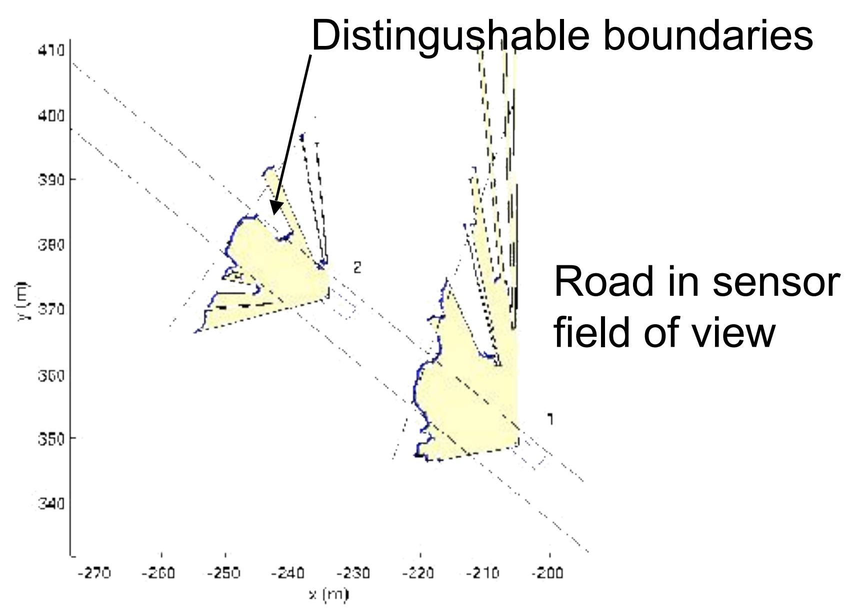
- Unimproved roads are a common presence in desert driving (95% of 2005 Grand Challenge race, e.g.)
- Lots of research in **image processing** for detecting roads
  - Typically frame-by-frame (no dynamics)
  - Sensitive to lighting conditions
  - No direct range information
- Improved on this state-of-the-art with processing of LADAR range images

Problem statement**:**

Provide and update a model-based estimate of the road, given LADAR range image sequences and sensor pose information

Parameterization**:**

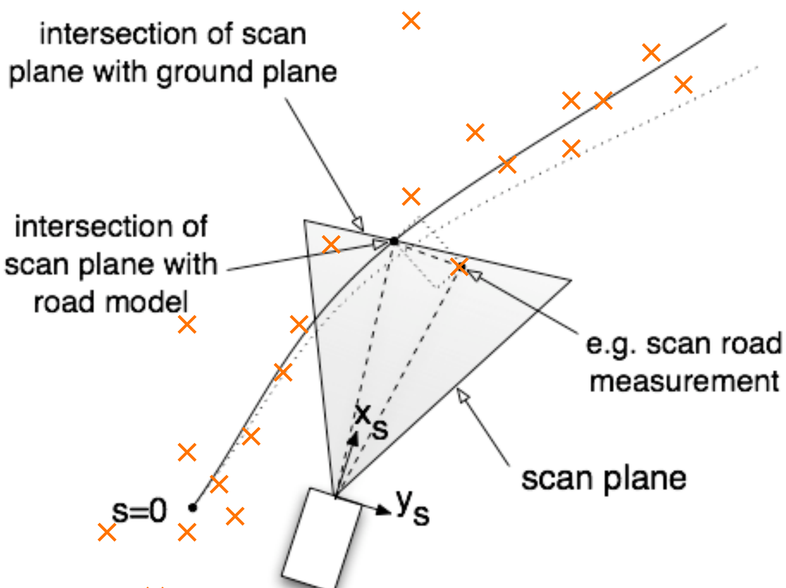
- Road heading, curvature, curvature rate (clothoid model), width

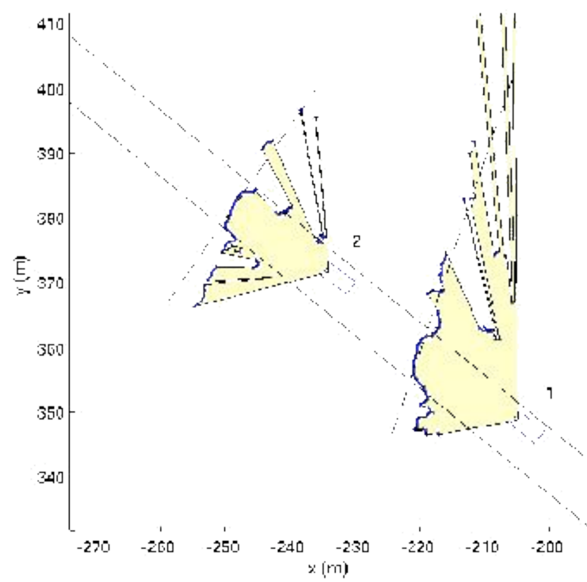


Road modeling

• Sensor used: single axis laser detection and ranging (LADAR)

• Features are extracted from sequences of range images that correspond to estimated road centerline location





Road parameterization by curvature

- Parameterization: Used clothoid model (piecewise constant rate of change of curvature)
- Including heading and excluding curvature rate improved results significantly

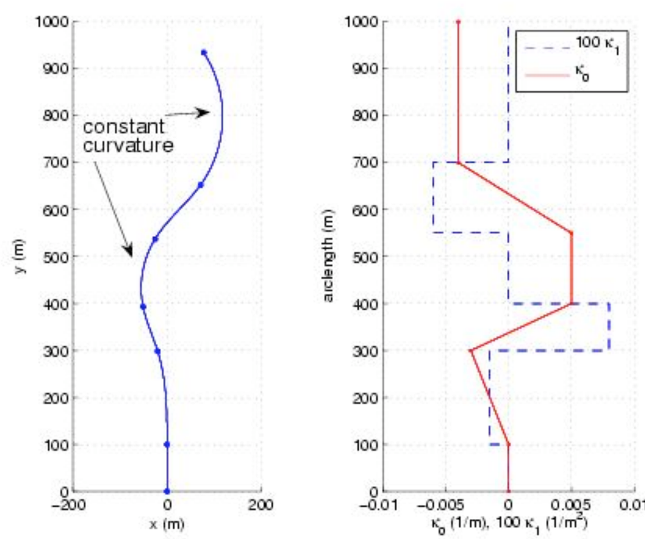
$$
\begin{aligned}
\kappa(s) &= \kappa_0 + \kappa_1 s \\
\theta(s) &= \theta_0(s) + \int_0^s \theta(\tau)\, d\tau \\
x(s) &= x_0(s) + \int_0^s \cos\theta(\tau)\, d\tau \\
y(s) &= y_0(s) + \int_0^s \sin\theta(\tau)\, d\tau
\end{aligned}
$$

Road parameter estimation

- Extended Kalman filter implemented to track road heading and curvature parameters
- Location of road centerline used as input in Kalman filter update model

$$
\begin{bmatrix} \dot{\kappa}_0 \\ \dot{\kappa}_1 \end{bmatrix} = \begin{bmatrix} 0 & v(t) \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \kappa_0 \\ \kappa_1 \end{bmatrix} + \begin{bmatrix} 0 \\ w(t) \end{bmatrix}
$$

$$
y(t) = \begin{bmatrix} \frac{1}{2}x_m^2 & \frac{1}{6}x_m^3 \end{bmatrix} \begin{bmatrix} \kappa_0 \\ \kappa_1 \end{bmatrix} + \nu(t)
$$



Kalman Filter C++ implementation

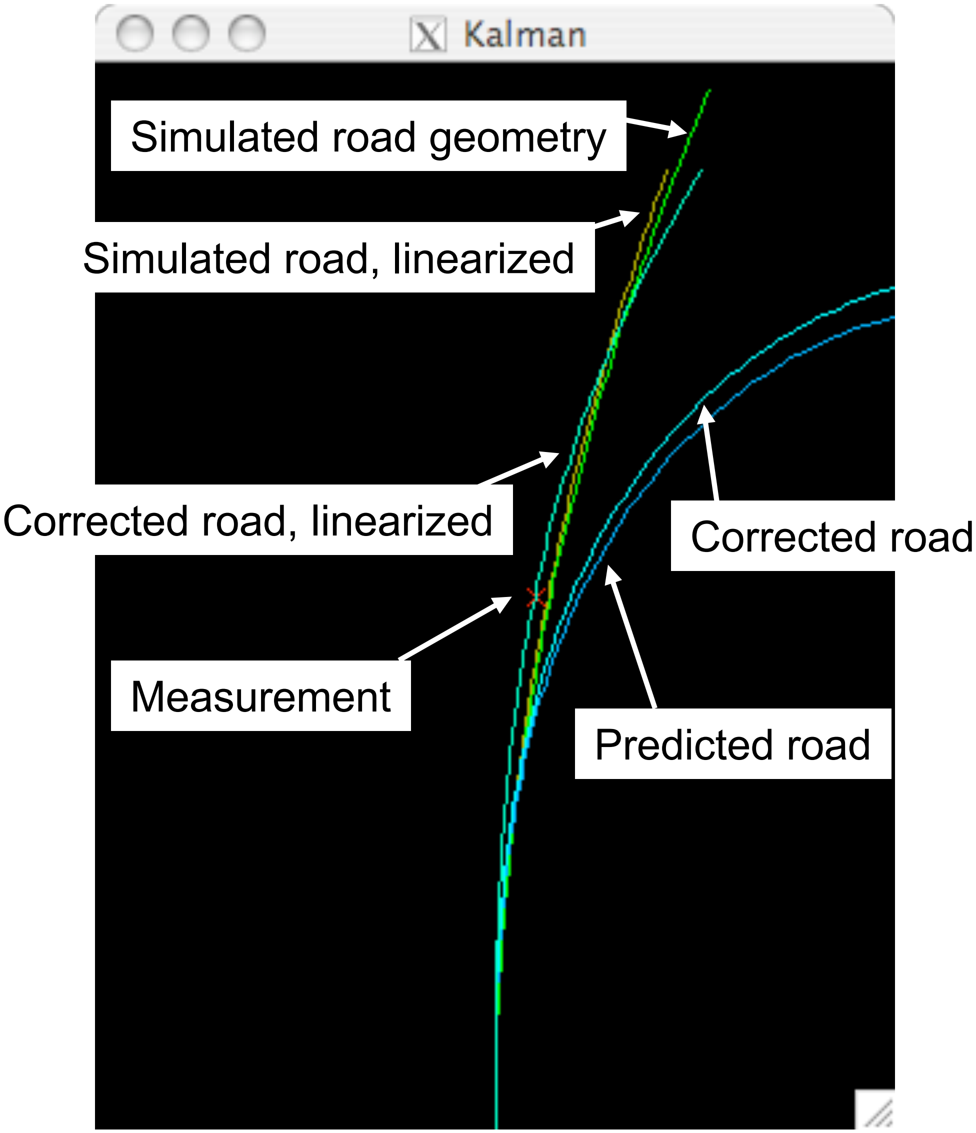
```
 /* cvCreateKalman args: dimension of state, meas, control vectors */
CvKalman* kalman = cvCreateKalman( 2, 1, 0 );

/* set measurement matrix to [1 0] */
cvSetIdentity( kalman->measurement_matrix, cvRealScalar(1) );
/* set process covariance matrix */
cvSetIdentity( kalman->process_noise_cov, cvRealScalar(0.0) );
cvSetReal2D(kalman->process_noise_cov, 0, 0, PROC_CURVATURE_COV);
cvSetReal2D(kalman->process_noise_cov, 1, 1, PROC_CURVRATE_COV);
/* set measurement noise variance to something (scalar) */
cvSetIdentity(kalman->measurement_noise_cov,
cvRealScalar(MEAS_NOISE_COV));
/* initialize posteriori error estimate covariance matrix */
cvSetIdentity( kalman->error_cov_post, cvRealScalar(1.0e-2));

 /* predict point position (2nd arg is control vector) */
const CvMat* predicted = cvKalmanPredict( kalman, NULL );

 /* adjust Kalman filter state estimate. */
const CvMat* corrected = cvKalmanCorrect( kalman, measurement );
```

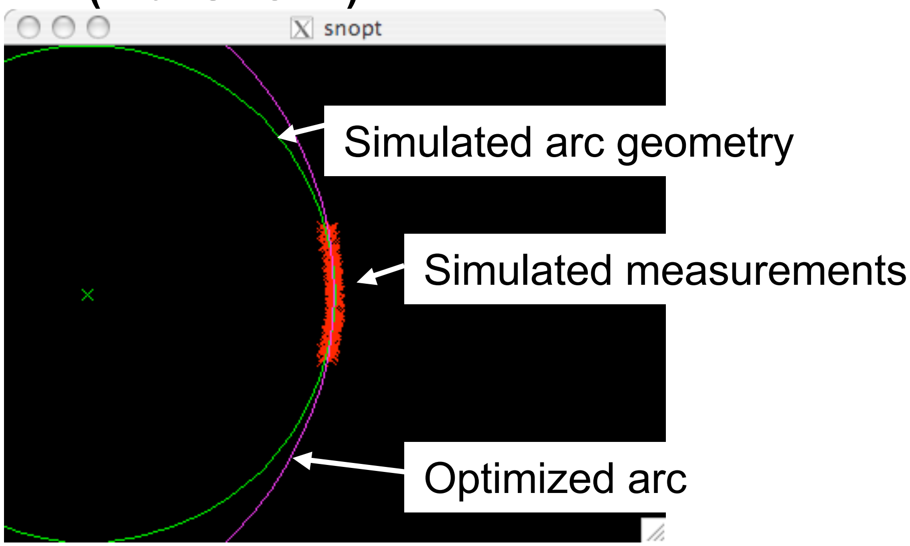
Road following EKF results (sim + live)



- Video: fully autonomous demonstration of road following using data only from one roof-mounted LADAR unit (Cremean, Murray 2006)
- 10 minute run at ~10 mph average speed
- Video sped up ~5X
- Model-based estimation method successfully tracks road heading and curvature

Moving Horizon Estimation (in progress)

Batch state estimation via optimization (with SNOPT)



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

Find initial state and process noise sequence such that

1. measurement data is matched (first term small);
2. process noise not larger than expected (second term small);
3. initial state not too far away from initial guess (third term small).

(from Wednesday lecture)

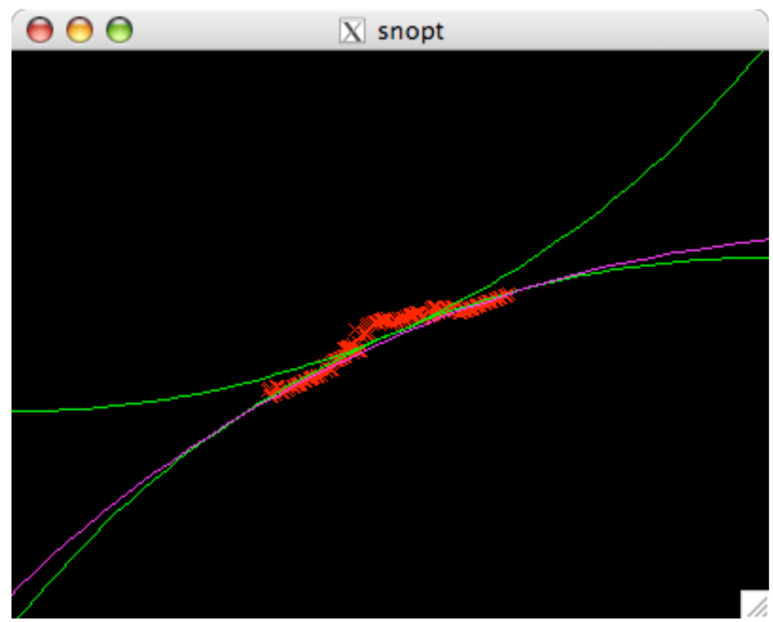
Recursive batch state estimation (no arrival cost yet)

1. Solve

$$\{z^*, \{\hat{w}_k\}_{T-N}^{T-1}\} = \arg \min_{z\in\mathcal{R}_{T-N}, w_k} \left(\sum_{k=T-N}^{T-1} L_k(w_k, v_k) + Z_{T-N}(z) \right)$$

subject to dynamics and $x_k \in \mathbb{X}_k$, $w_k \in \mathbb{W}_k$, $v_k \in \mathbb{V}_k$.

(Use (S)QP, NPSOL, NTG,. . .)



References

(See lecture page for references:
http://www.cds.caltech.edu/~murray/wiki/index.php?title=Alice:_Road_Following)

Reading

- Cf. (extended) Kalman filter reading materials from Monday 4/17/06 lecture.
- Cf. Moving Horizon Estimation reading materials from Wednesday 4/19/06 lecture.
- Model-Based Estimation of Off-Highway Road Geometry using Single-Axis LADAR and Inertial Sensing, Lars B. Cremean and Richard M. Murray, To appear, *Proc. of 2006 International Conference on Robotics and Automation*

Additional Resources

- The Open Source Computer Vision (OpenCV) Library has a well-documented C++ class that implements the Kalman filter, and a host of image processing tools. See http://www.intel.com/technology/computing/opencv/ and http://opencvlibrary.sourceforge.net/
- An Octave (MATLAB clone) toolbox for implementing nonlinear receding horizon control and moving horizon estimation is available at http://jbrwww.che.wisc.edu/home/tenny/nmpc/. The release is somewhat old (2003); be sure to get the CVS version to avoid some known bugs.
- For numerical solving, SNOPT is commercial software, but CDS cluster has a license for it. NPSOL is another option. SNOPT 5.3 user manual available at http://citeseer.ist.psu.edu/gill99users.html.