Tracking

Components of a Tracking System

- Object model including position,
  and optionally size, appearance, velocity
- Predict locations of objects in new frame
- Search in new frame
- Data association
- Update model

- Also useful: background model

Face Tracking

Tracking Challenges

- Occlusion
- Similar foreground/background
- Camera motion
- Brightness changes:
  - Overall
  - Foreground object
- Update foreground model?
  - No: can't deal with brightness changes
  - Yes: "tracker drift"

Object Model

- Position
- Dynamic model: velocity, acceleration, etc.
- Appearance model:
  - None (object = anything that's not background)
  - Color / hue histogram
  - Template
- Shape model:
  - None (point): "feature tracking"
  - Ellipse / rectangle: "blob tracking"
  - Full outline: snakes, etc.

Prediction

- Simple approximations:
  - Window around last position
  - Window, updated by velocity on last frame
- Prediction uncertainty (together with object size) gives search window
- Kalman filter (later today)

Search

- Search strategy depends on appearance model
  - If we're using a template, look for it…
  - If using foreground color histograms, lookup pixels to evaluate probability
- If no appearance model: "background subtraction"
  - In simplest case, take |frame – background|
  - More generally: estimate probability that a given pixel is foreground given background model
  - Compare to threshold

Background Models

- Major decision: adapt over time?
  - If no, does not respond to changing conditions, but appropriate for short videos
  - If yes, danger of still foreground objects being treated as background
- Non-adaptive methods:
  - Single frame obtained @ beginning (or mean/median/mode of *n* frames)

Single-Frame Statistical Model

- Alternative: mean + std. dev. per pixel
  - Gathered over many frames before tracking begins
  - Allows for noise, small motion, small changes in brightness, etc.
  - Permits estimation of probability:

$$p_{bg}(i) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(i-mean)^2}{2\sigma^2}}$$

Gaussian Mixture Model

- Background statistics often not unimodal
  - Trees in the wind, shadows, etc.
- Solution: mixture of Gaussians @ each pixel
  - Compute using expectation maximization
- Still allows computing $p_{background}$
- Usual difficulties with determining # of clusters
  - In practice, 3-5 observed to be a good number

Adaptive Background Models

- Previous frame
- Mean/median of previous *n* frames
- Rolling average of previous frames
- For GMM, update *closest* Gaussian
- In all cases, must be careful not to update pixels determined to be foreground

Dealing with Camera Motion

- If camera is moving, can eliminate its effects: "image stabilization" or "dominant motion estimation"
- Image alignment using feature matching, optical flow, phase correlation, etc.
  - Must use robust methods to not get confused by motion of foreground object(s)
  - Often done using EM
  - Implicitly segments the image: layered motion

Data Association

- For tracking multiple objects, need to maintain identities across frames
- For feature tracking, simplest option is nearest neighbor (after prediction)
  - Can include threshold to permit occlusion
  - More robust: nearest neighbor in both directions
  - Can also include "soft", probabilistic assignment

EM for Blob Tracking

- Represent each object as a "blob": Gaussian with mean μ and covariance Σ
  - Often illustrated as ellipse or rectangle
- Dynamic model for mean position (and sometimes, but rarely, covariance)

EM for Blob Tracking

- At each frame, predict new positions of blobs
  - Implicitly limits region of interest (ROI) to a few standard deviations around predicted position
- Construct “strength image”
  - Thresholded result of background subtraction
  - (Output of feature tracker)
- Run EM on probability image to update positions

EM Tracking Demo

Mean Shift Tracker

- Applied when we have foreground color histogram
- EM-like loop, except "strength image" recomputed on each iteration
  - Let m = model's normalized color histogram
  - Let d = data color histogram (i.e., histogram of ROI)
  - Set $w_p = \sqrt{m_p / d_p}$
  - Compute *mean shift*:

$$\vec{\mu}_{j,new} - \vec{\mu}_j = \frac{\sum_p w_p G_j(\vec{x}_p - \vec{\mu}_j, \Sigma_j)\,(\vec{x}_p - \vec{\mu}_j)}{\sum_p w_p G_j(\vec{x}_p - \vec{\mu}_j, \Sigma_j)}$$

Updating Model?

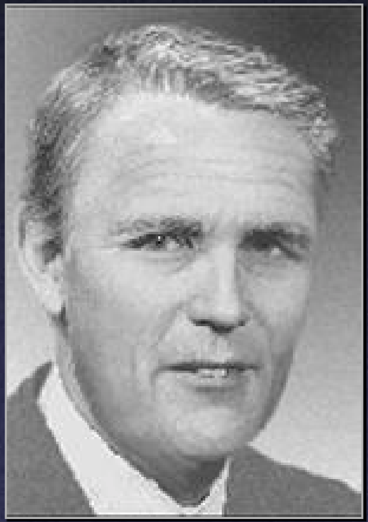
- With histogram-based trackers, easy for part of background to get into foreground histogram
  - Leads to "tracker drift": tracker locks onto different object (or, more typically, part of background)
  - Result: often better to stay with constant histogram, or adapt *very slowly* over time

Updating Model?

- Biggest motivation for adaptation: lighting changes
  - But using only Hue is usually insensitive to those…
  - BIG CAVEAT: be sure to undo gamma when converting to HSV space
  - ANOTHER BIG CAVEAT: be sure to discard pixels with low S or V

Kalman Filtering

- Assume that results of experiment (i.e., tracking) are noisy measurements of system state
- Model of how system evolves
- Optimal combination of system model and observations
- Prediction / correction framework



Rudolf Emil Kalman

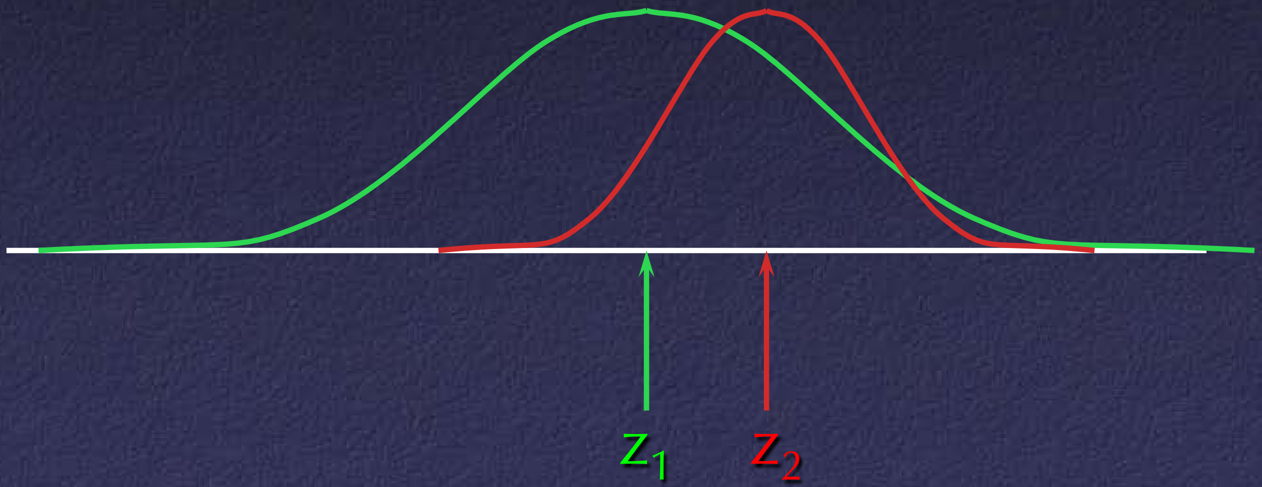
Acknowledgment: much of the following material is based on the SIGGRAPH 2001 course by Greg Welch and Gary Bishop (UNC)

Simple Example

- Measurement of a single point z_1
- Variance σ_1^2 (uncertainty σ_1)
  - Assuming Gaussian distribution
- Best estimate of true position $\hat{x}_1 = z_1$
- Uncertainty in best estimate $\hat{\sigma}_1^2 = \sigma_1^2$

Simple Example

- Second measurement z_2, variance σ_2^2
- Best estimate of true position?



Simple Example

- Second measurement z_2, variance $\sigma_2{}^2$
- Best estimate of true position: weighted average

$$\hat{x}_2 = \frac{\frac{1}{\sigma_1^2} z_1 + \frac{1}{\sigma_2^2} z_2}{\frac{1}{\sigma_1^2} + \frac{1}{\sigma_2^2}}$$

$$= \hat{x}_1 + \frac{\hat{\sigma}_1^2}{\hat{\sigma}_1^2 + \sigma_2^2}\left(z_2 - \hat{x}_1\right)$$

- Uncertainty in best estimate $\hat{\sigma}_2^2 = \dfrac{1}{\frac{1}{\hat{\sigma}_1^2} + \frac{1}{\sigma_2^2}}$

Online Weighted Average

- Combine successive measurements into constantly-improving estimate
- Uncertainty decreases over time
- Only need to keep current measurement, last estimate of state and uncertainty

Terminology

- In this example, position is *state* (in general, any vector)
- State can be assumed to evolve over time according to a *system model* or *process model* (in this example, "nothing changes")
- Measurements (possibly incomplete, possibly noisy) according to a *measurement model*
- Best estimate of state $\hat{x}$ with covariance P

Linear Models

- For "standard" Kalman filtering, everything must be linear
- System model:

$$x_k = \Phi_{k-1} x_{k-1} + \xi_{k-1}$$

- The matrix Φ_k is *state transition matrix*
- The vector ξ_k represents *additive noise*, assumed to have covariance Q

Linear Models

- Measurement model:

$$z_k = H_k x_k + \mu_k$$

- Matrix H is *measurement matrix*
- The vector μ is *measurement noise*, assumed to have covariance R

PV Model

- Suppose we wish to incorporate velocity

$$\mathbf{x}_k = \begin{bmatrix} x \\ {dx}/{dt} \end{bmatrix}$$

$$\Phi_k = \begin{bmatrix} 1 & \Delta t_k \\ 0 & 1 \end{bmatrix}$$

$$H = \begin{bmatrix} 1 & 0 \end{bmatrix}$$

Prediction/Correction

- Predict new state

$$x'_k = \Phi_{k-1}\hat{x}_{k-1}$$

$$P'_k = \Phi_{k-1}P_{k-1}\Phi^{\mathrm{T}}_{k-1} + Q_{k-1}$$

- Correct to take new measurements into account

$$\hat{x}_k = x'_k + K_k\left(z_k - H_k x'_k\right)$$

$$P_k = \left(I - K_k H_k\right)P'_k$$

- Important point: no need to invert measurement matrix H

Kalman Gain

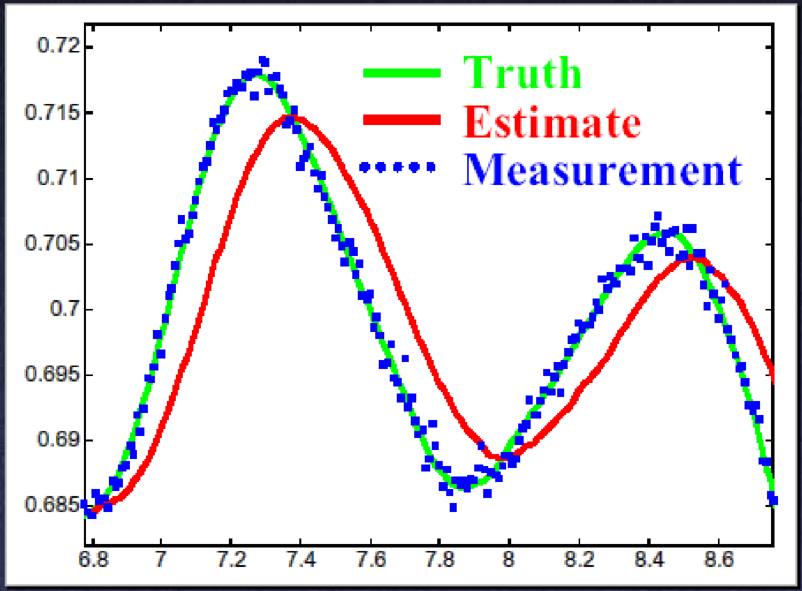
- Weighting of process model vs. measurements

$$K_k = P'_k H_k^{\mathrm{T}} \left(H_k P'_k H_k^{\mathrm{T}} + R_k \right)^{-1}$$

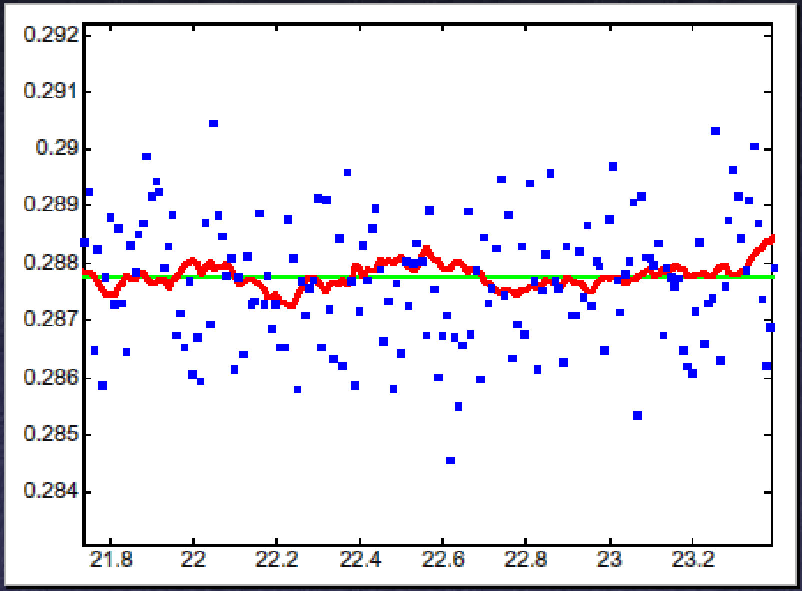
- Compare to what we saw earlier:

$$\frac{\sigma_1^2}{\sigma_1^2 + \sigma_2^2}$$

Results: Position-Only Model



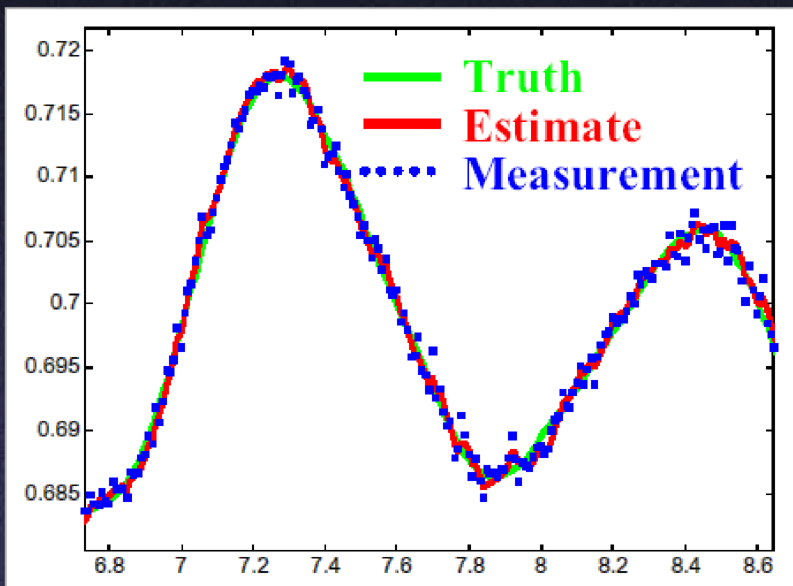
Moving



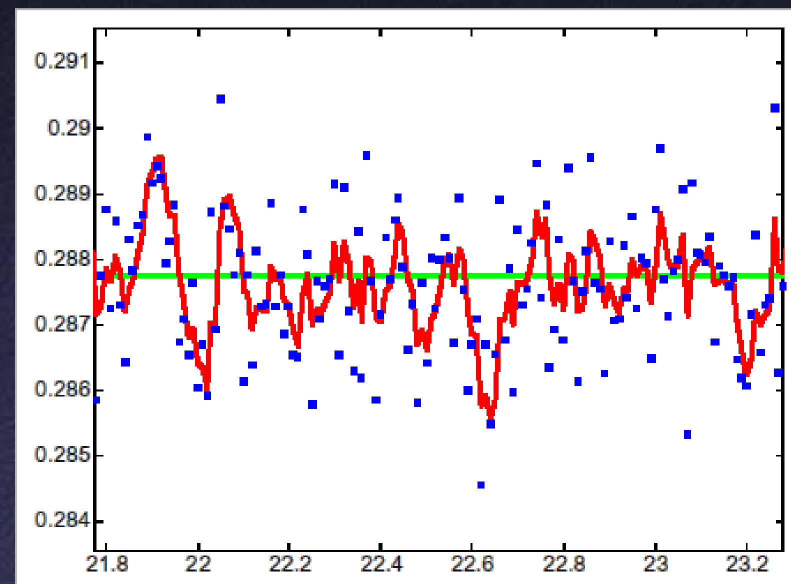
Still

[Welch & Bishop]

Results: Position-Velocity Model



Moving



Still

[Welch & Bishop]

Extension: Multiple Models

- Simultaneously run many KFs with different system models
- Estimate probability that each KF is correct
- Final estimate: weighted average

Probability Estimation

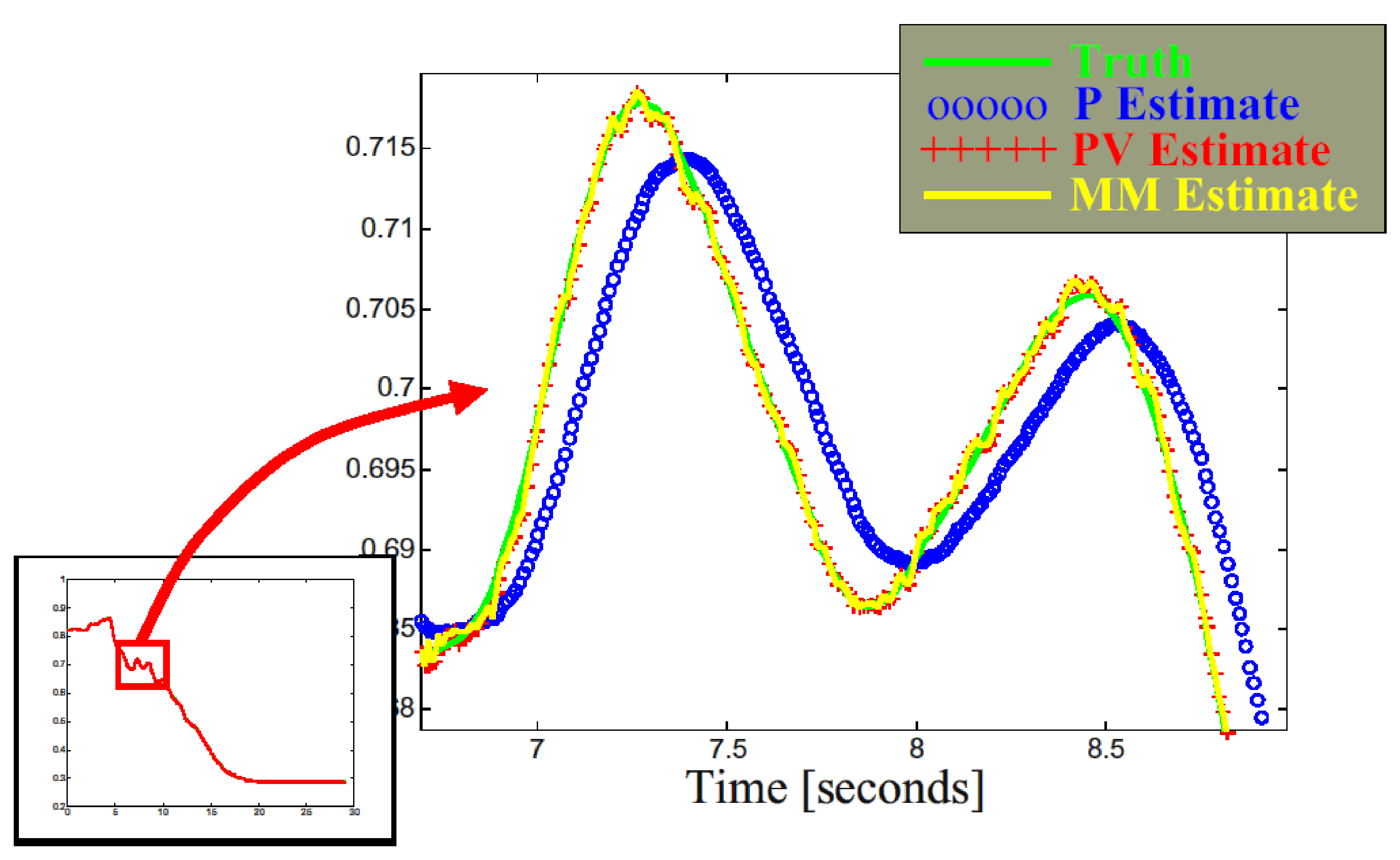
- Given some Kalman filter, the probability of a measurement z_k is just *n*-dimensional Gaussian

$$p = \frac{1}{\left(2\pi\,|\,C\,|\right)^{n/2}} e^{-\frac{1}{2}\left(z_k - H_k x_k'\right)^{\mathrm{T}} C^{-1} \left(z_k - H_k x_k'\right)^{\mathrm{T}}}$$

where

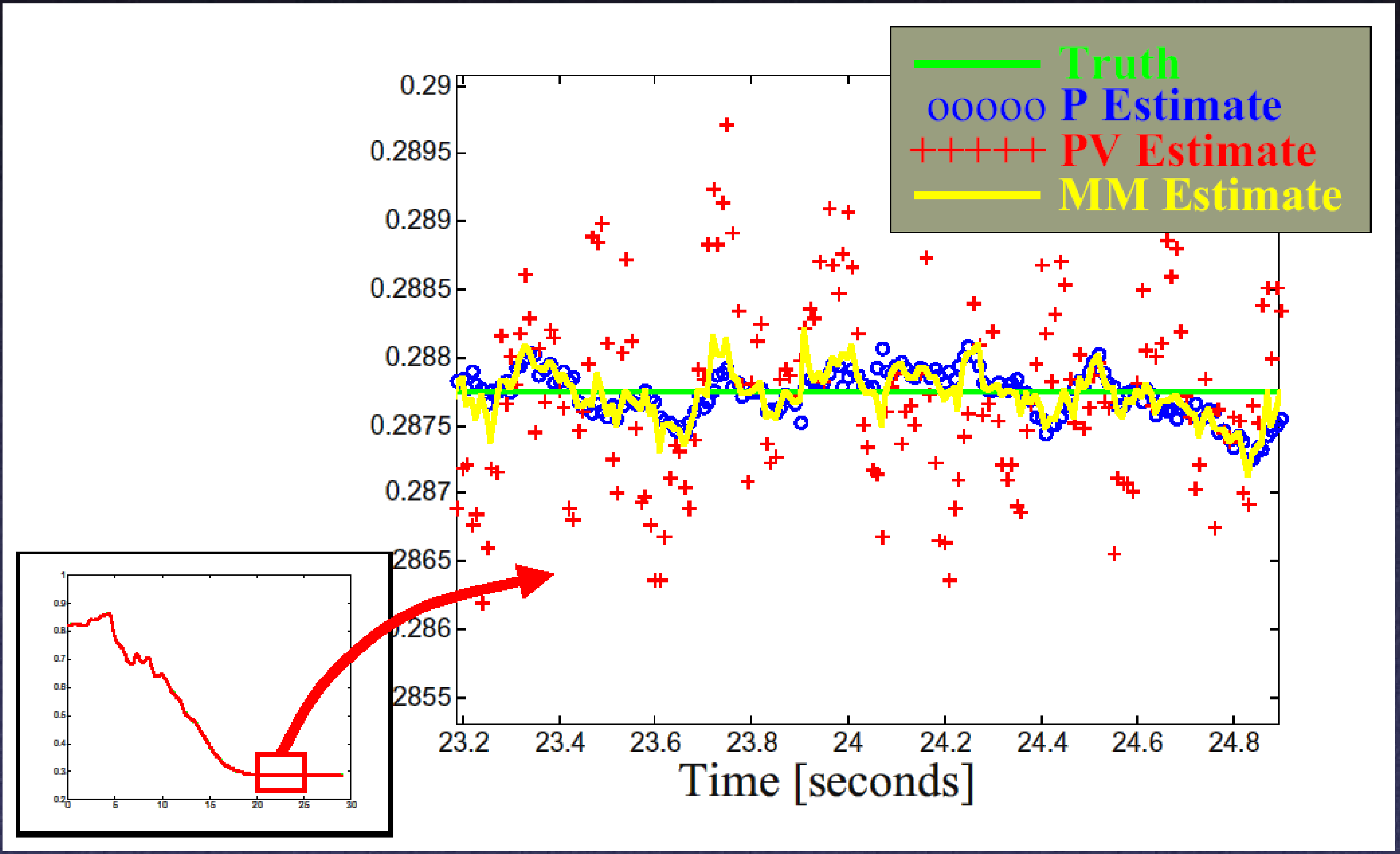
$$C = HPH^{\mathrm{T}} + R$$

Results: Multiple Models



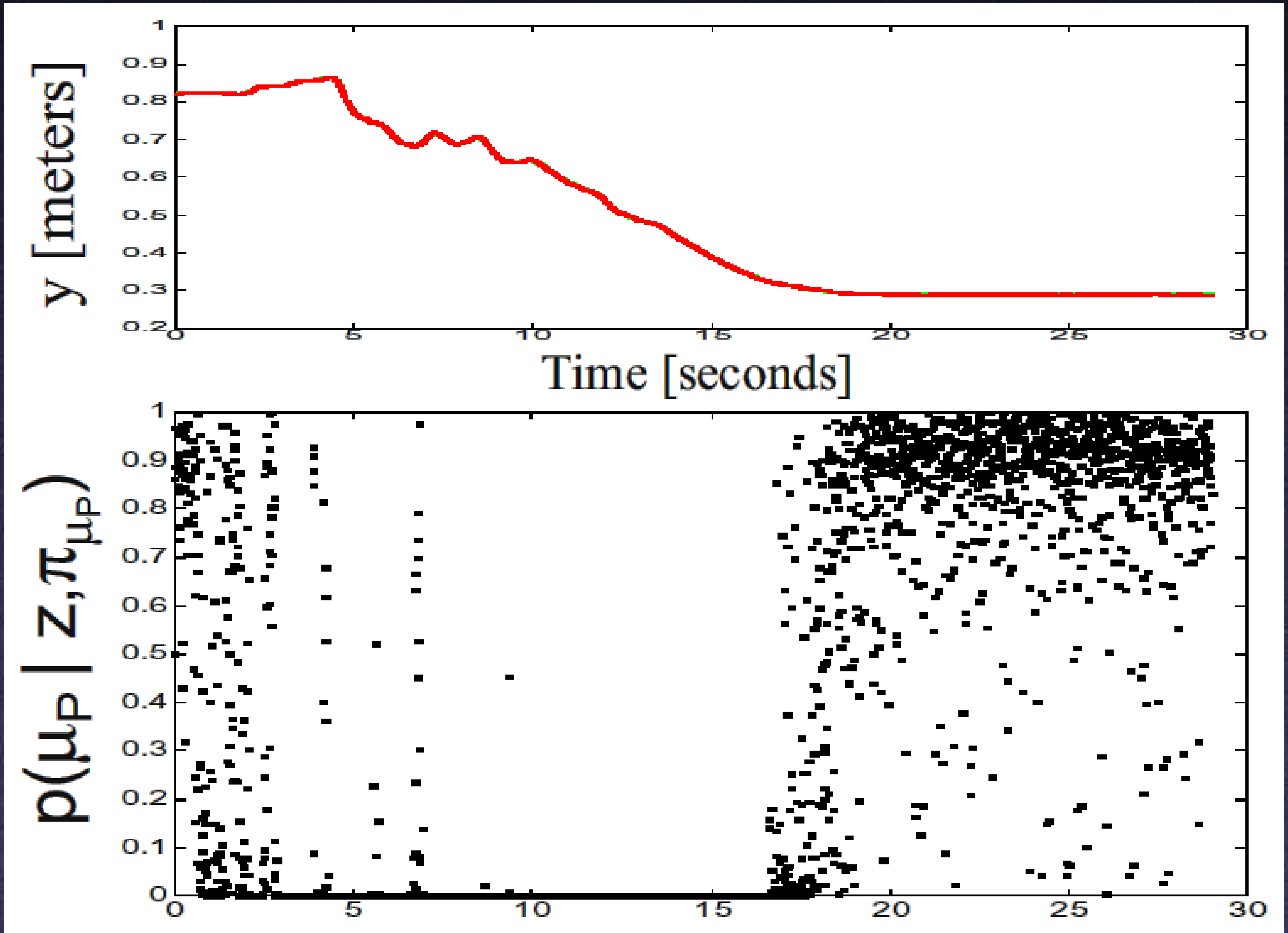
[Welch & Bishop]

Results: Multiple Models



[Welch & Bishop]

Results: Multiple Models

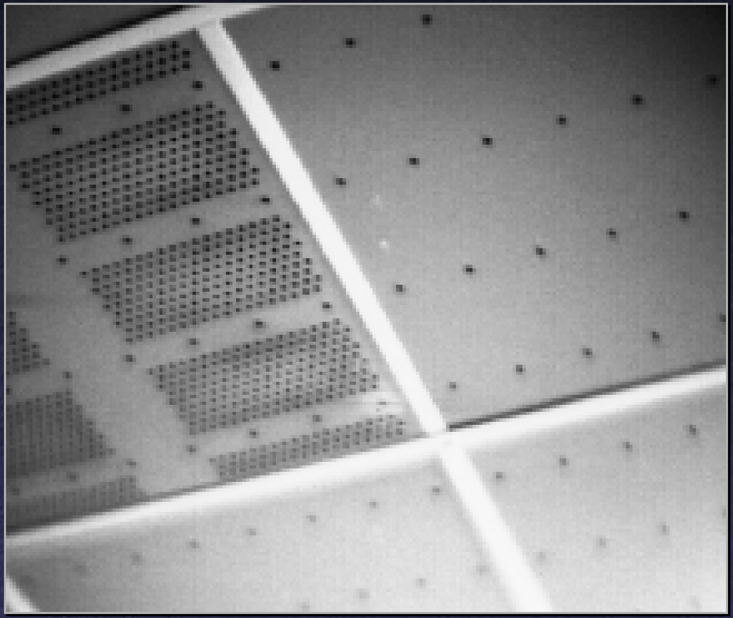


[Welch & Bishop]

Extension: SCAAT

- *H* can be different at different time steps
  - Different sensors, types of measurements
  - Sometimes measure only part of state
- Single Constraint At A Time (SCAAT)
  - Incorporate results from one sensor at once
  - Alternative: wait until you have measurements from enough sensors to know complete state (MCAAT)
  - MCAAT equations often more complex, but sometimes necessary for initialization

UNC HiBall



- 6 cameras, looking at LEDs on ceiling
- LEDs flash over time

[Welch & Bishop]

Extension: Nonlinearity (EKF)

- HiBall state model has nonlinear degrees of freedom (rotations)
- Extended Kalman Filter allows nonlinearities by:
  - Using general functions instead of matrices
  - Linearizing functions to project forward
  - Like 1st order Taylor series expansion
  - Only have to evaluate Jacobians (partial derivatives), not invert process/measurement functions

Other Extensions

- On-line noise estimation
- Using known system input (e.g. actuators)
- Using information from both past and future
- Non-Gaussian noise and particle filtering