Tracking

COS 429

Princeton University

Tracking

- Feature tracking
- Object tracking

Tracking

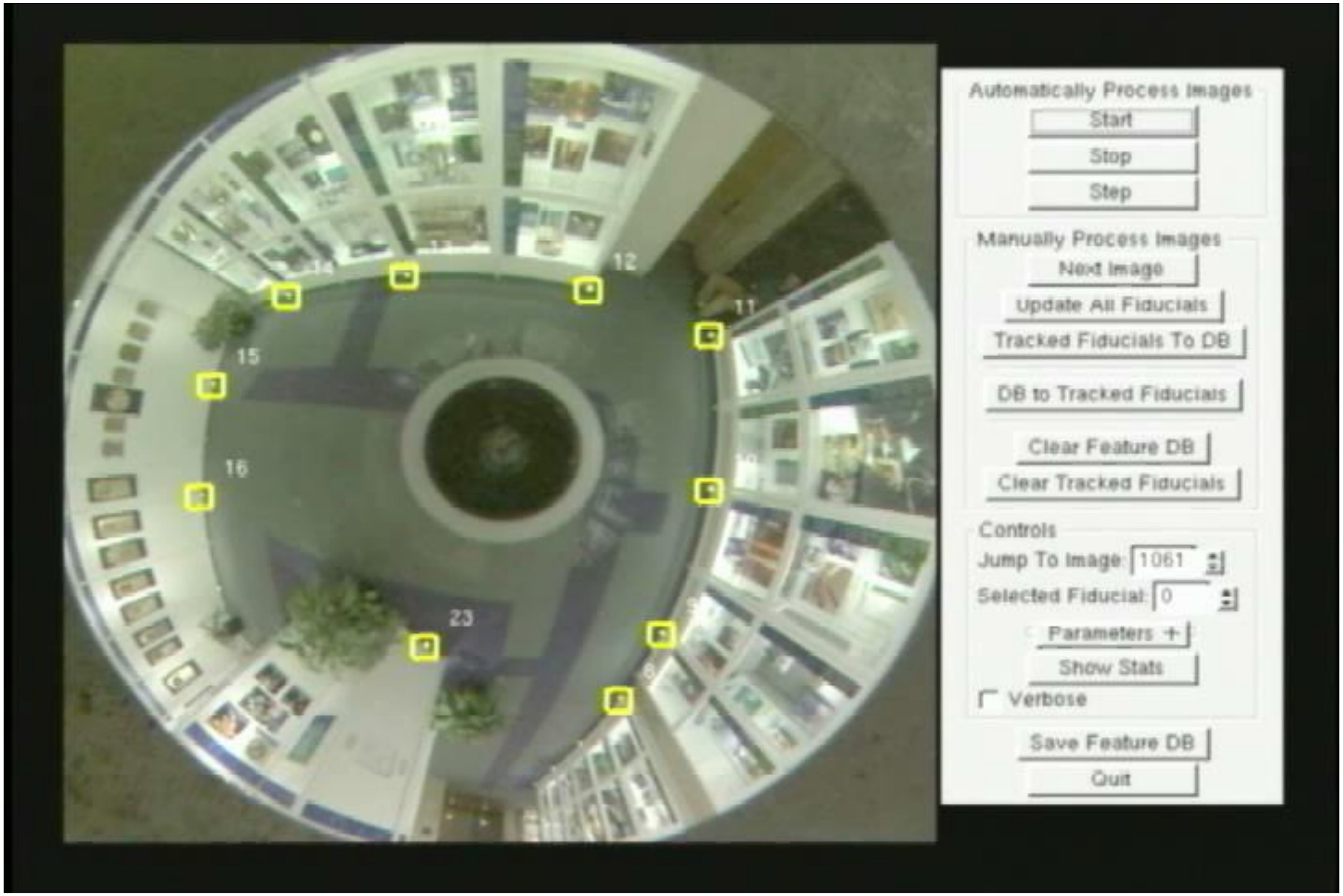
- Feature tracking ←
- Object tracking

Feature Tracking

- Given sequence of images
- Find feature correspondences

Applications?

Feature Tracking Example



Sea of Images

Feature Tracking Example

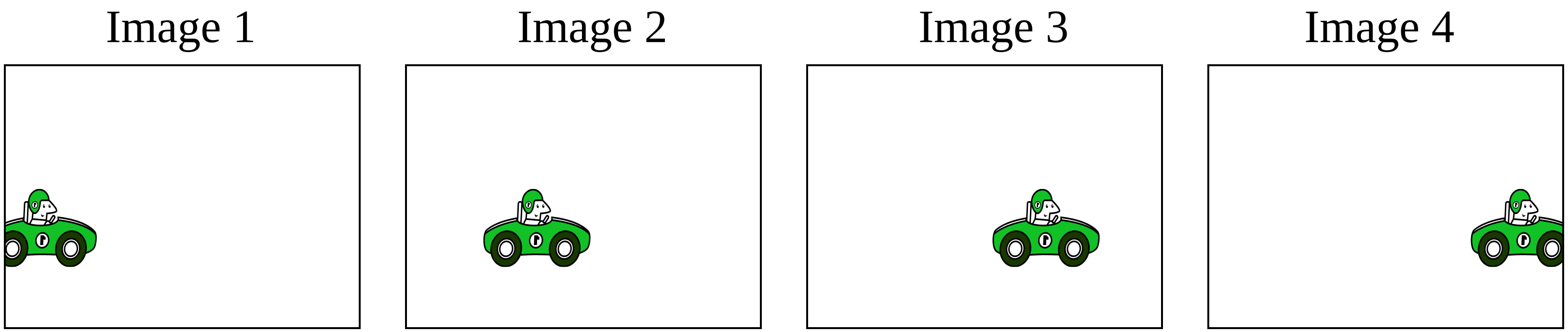
Tracking

- Feature tracking
- Object tracking ←

Object Tracking

- Given sequence of images
- Track moving foreground objects

Object Tracking



- Can we estimate the position of the object?
- Can we predict future positions?

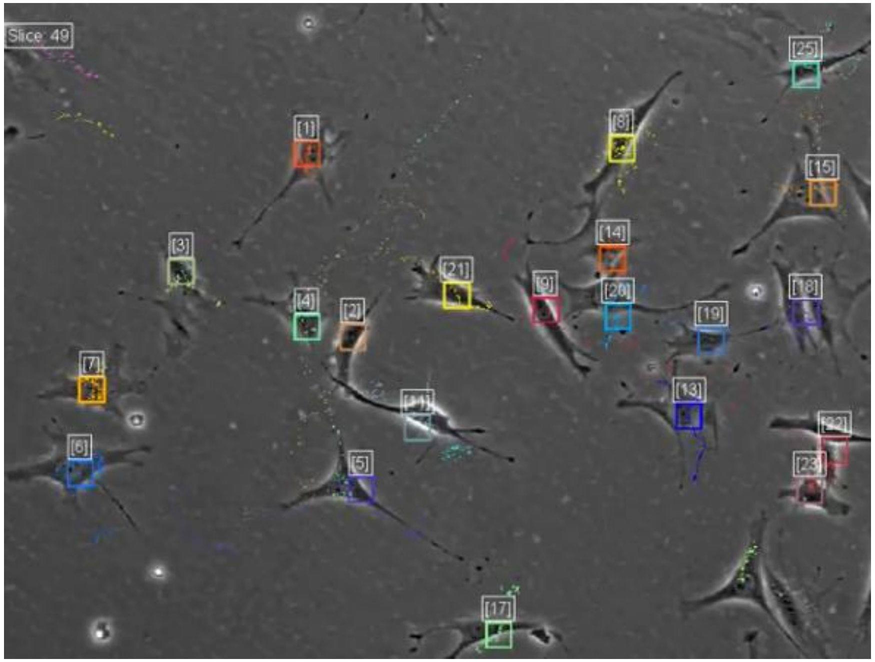
Applications?

Applications

- Astronomy, biology, etc.
- Surveillance
- Activity analysis
- Gesture recognition
- etc.

Applications

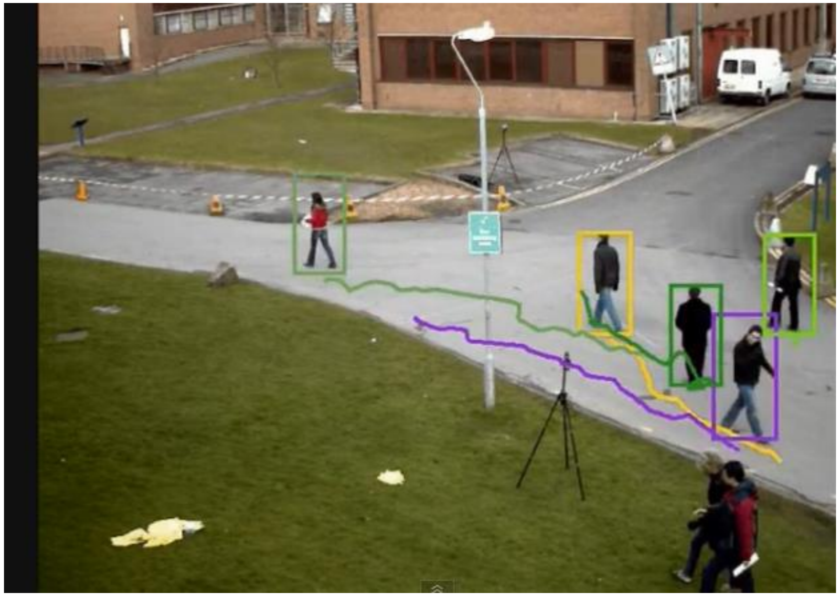
- Astronomy, biology, etc.
- Surveillance
- Activity analysis
- Gesture recognition
- etc.



http://www.youtube.com/watch?v=SLYgvHzAm2w

Applications

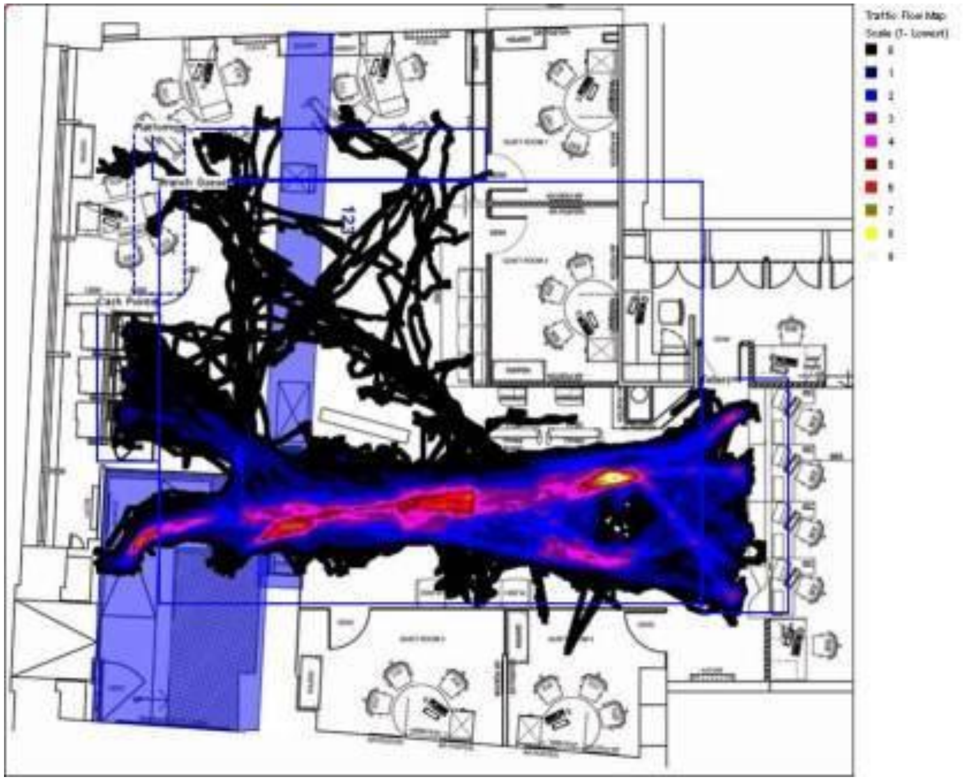
- Astronomy, biology, etc.
- Surveillance
- Activity analysis
- Gesture recognition
- etc.



http://www.youtube.com/watch?v=bY8qGk45WxM

Applications

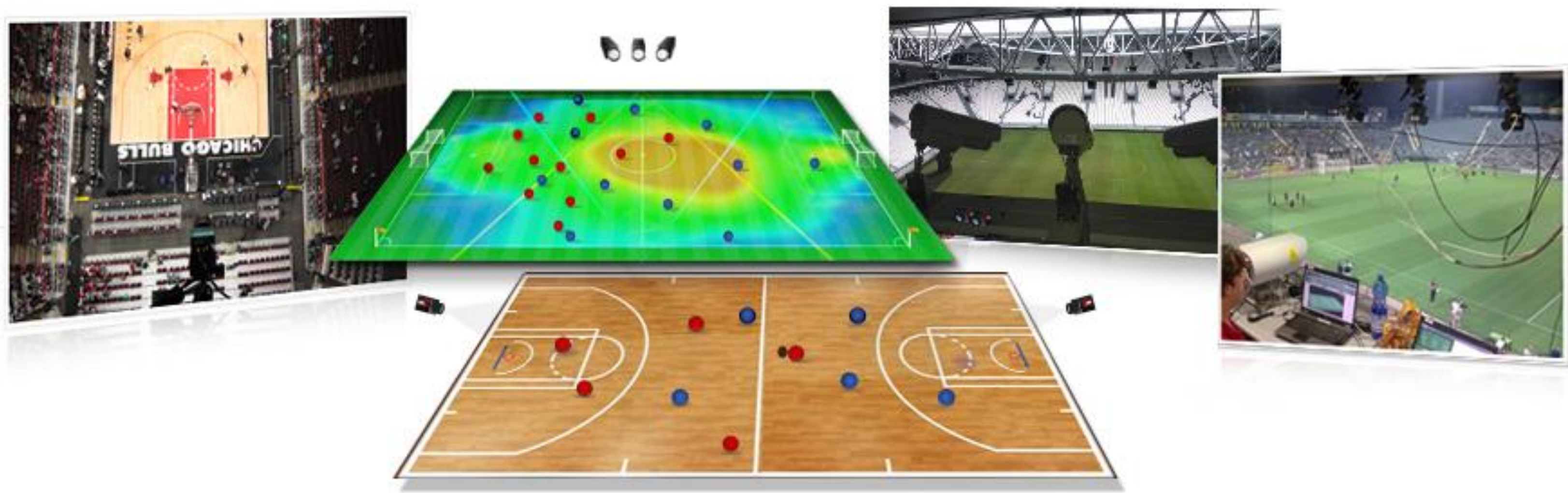
- Astronomy, biology, etc.
- Surveillance
- Activity analysis
- Gesture recognition
- etc.



wavestore

Applications

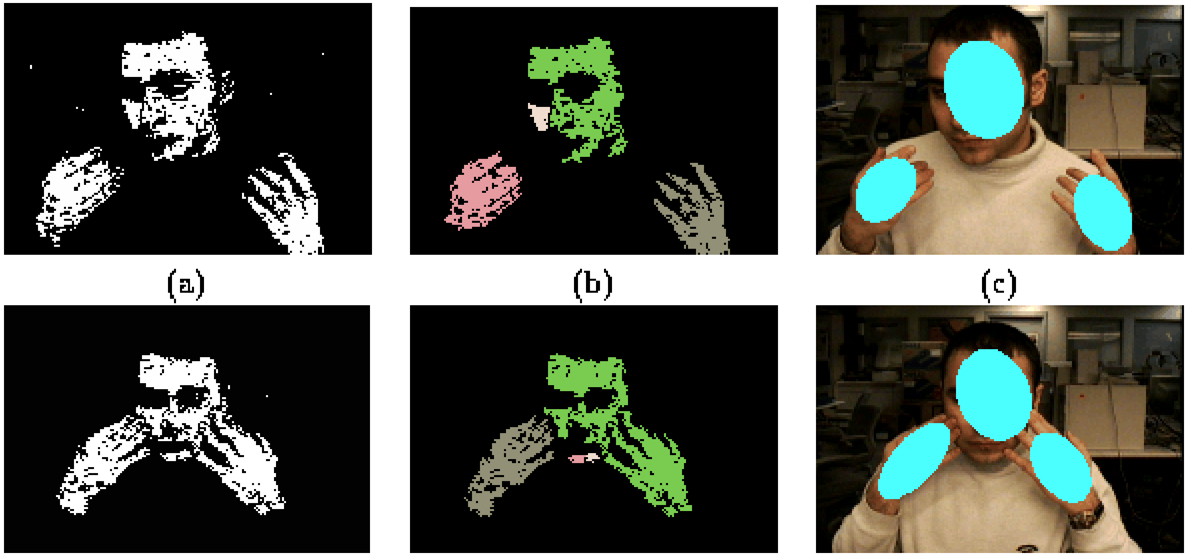
- Astronomy, biology, etc.
- Surveillance
- Activity analysis
- Gesture recognition



SportVU

Applications

- Astronomy, biology, etc.
- Security surveillance
- Activity analysis
- Gesture recognition
- etc.

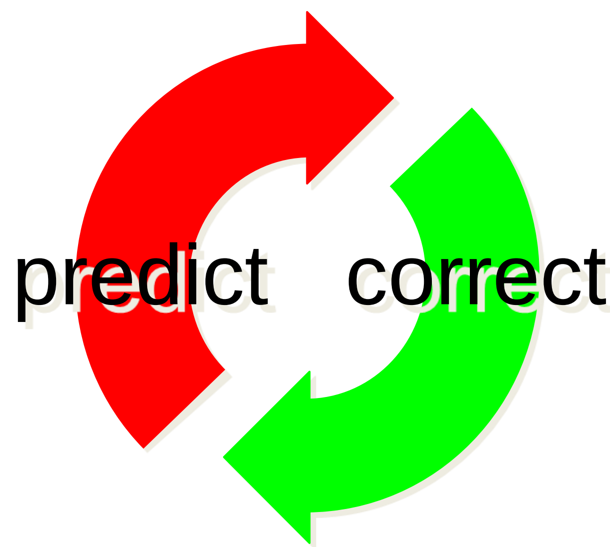


http://www.cs.columbia.edu/~jebara/htmlpapers/ARL/node21.html

Methods?

General Strategy

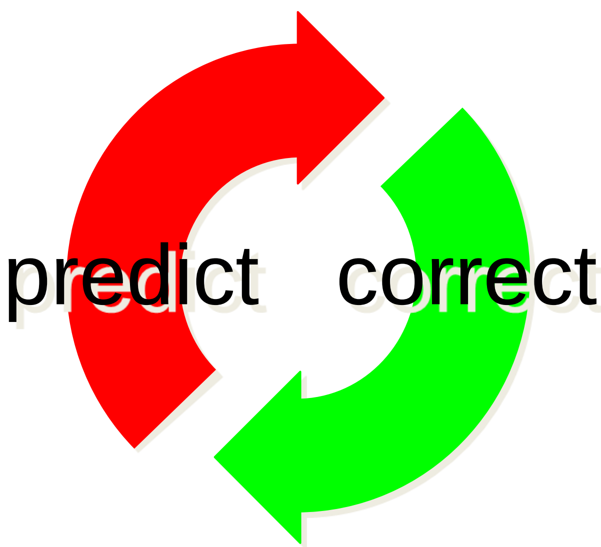
- Initialize *model* in the first frame
- Given model estimate for frame *t*-1:
  - *Predict* for frame *t*
    - Use *dynamics model* of how the image changes
  - *Correct* for frame *t*
    - Use foreground estimation in current frame to update model



Kristen Grauman

General Strategy

- Initialize *model* in the first frame
- Given model estimate for frame *t*-1:
  - *Predict* for frame *t*
    - Use *dynamics model* of how the image changes
  - *Correct* for frame *t*
    - Use foreground estimation in current frame to update model



Kristen Grauman

Outline

- Feature tracking
- Object tracking
  - Foreground estimation
  - Model update

Foreground Estimation

Image at time t:
$I(x, y, t)$
$\Downarrow$

Foreground at time t:
F(x, y, t)

How?

Slide credit: Birgi Tamersoy

Background Subtraction

Image at time t: $I(x, y, t)$ $\Downarrow$

Background at time t: $B(x, y, t)$ $\Downarrow$

$|$

$-$

$| > Th$

1. Estimate the background for time t.
2. Subtract the estimated background from the input frame.
3. Apply a threshold, Th, to the absolute difference to get the **foreground mask**.

Slide credit: Birgi Tamersoy

Background Subtraction

Image at time t:
$I(x, y, t)$

Foreground at time t:
F(x, y, t)

1. Estimate the background for time t.
2. Subtract the estimated background from the input frame.
3. Apply a threshold, Th, to the absolute difference to get the **foreground mask**.

Slide credit: Birgi Tamersoy

Background Subtraction

Image at time t:
$I(x, y, t)$
$\Downarrow$

Background at time t:
$B(x, y, t)$
$\Downarrow$

How estimate the background?

Background = Previous Frame

- Background is estimated to be the previous frame. Background subtraction equation then becomes:

$$B(x, y, t) = I(x, y, t-1)$$
$$\Downarrow$$
$$|I(x, y, t) - I(x, y, t-1)| > Th$$

- Depending on the object structure, speed, frame rate and global threshold, this approach may or may **not** be useful (usually **not**).

$|$

$-$

$| > Th$

Slide credit: Birgi Tamersoy

Background = Previous Frame

$Th = 25$

$Th = 50$

$Th = 100$

$Th = 200$

Slide credit: Birgi Tamersoy

Background = Mean Filter

- In this case the background is the mean of the previous n frames:

$$B(x, y, t) = \frac{1}{n}\sum_{i=0}^{n-1} I(x, y, t-i)$$
$$\Downarrow$$
$$|I(x, y, t) - \frac{1}{n}\sum_{i=0}^{n-1} I(x, y, t-i)| > Th$$

- For $n = 10$:

Estimated Background

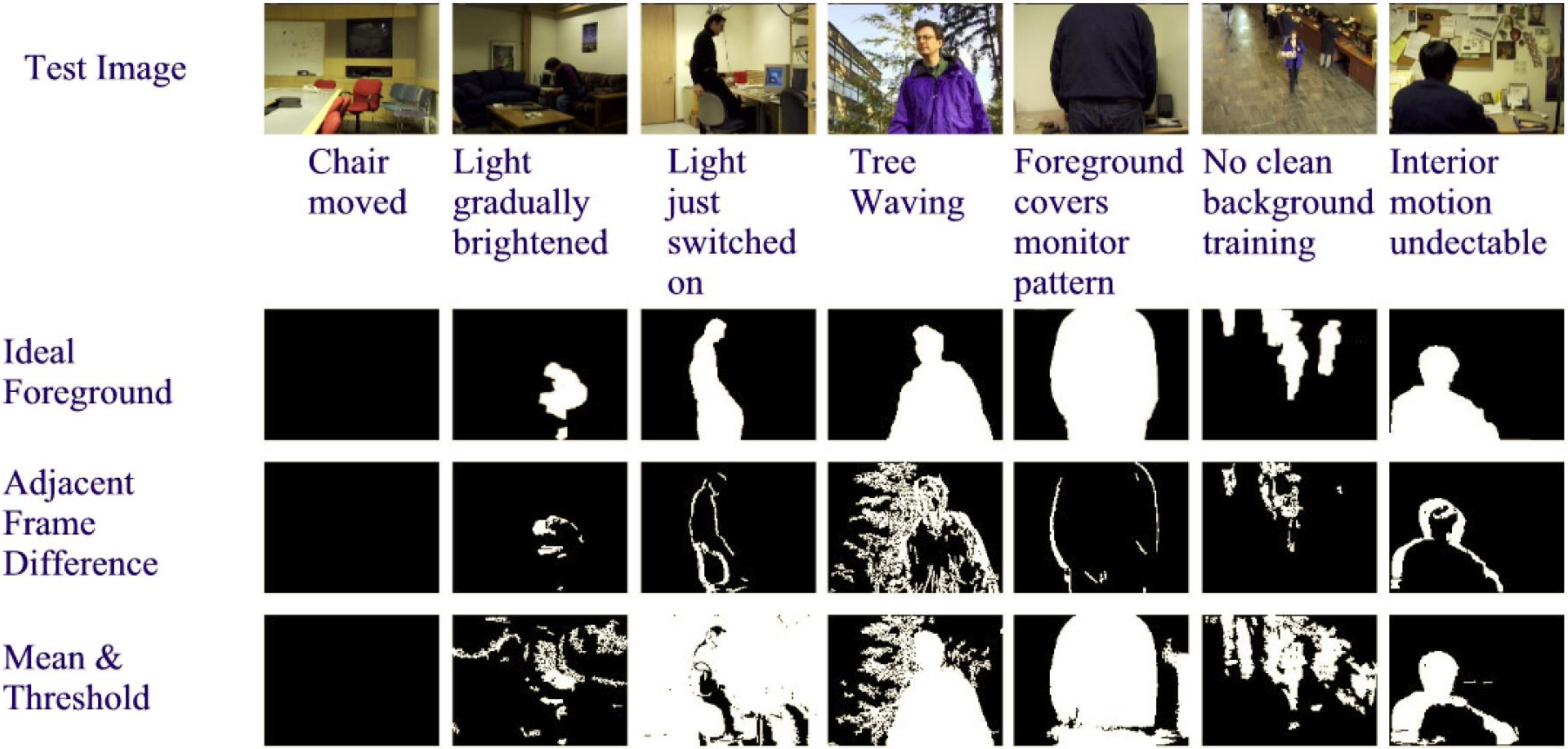
Foreground Mask

Slide credit: Birgi Tamersoy

Background = Mean Filter

- When won’t this work?

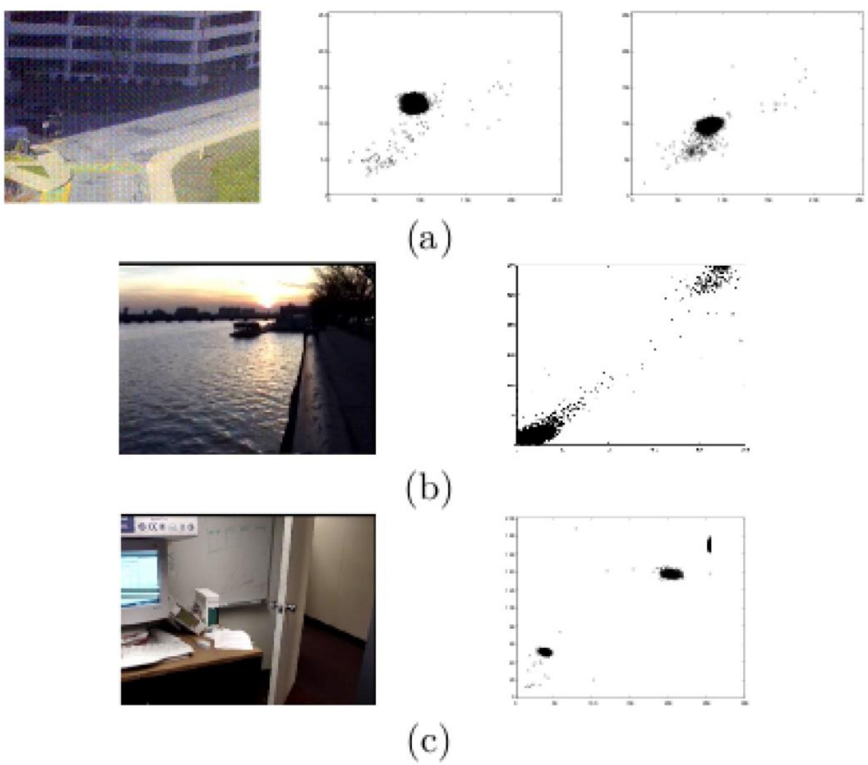
Background = Mean Filter

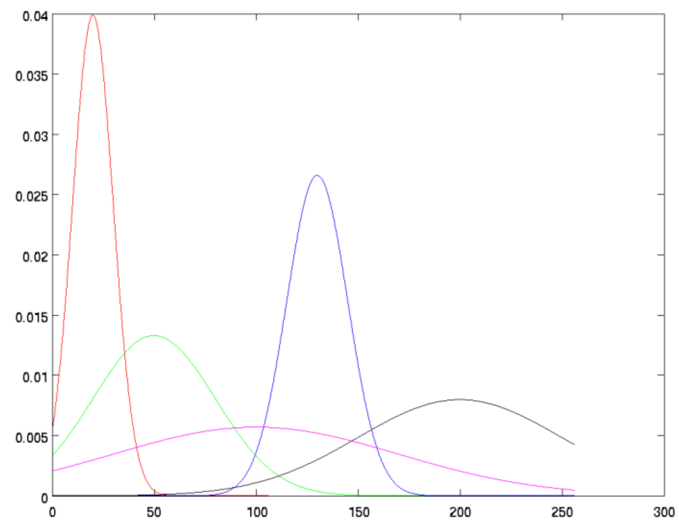


- Toyama et al. 1999

Background = Mixture Model

- Model each background pixel with a *mixture* of Gaussians; update parameters over time.





Adaptive Background Mixture Models for Real-Time Tracking, Chris Stauer & W.E.L. Grimson

Background = Median Filter

- Assuming that the background is more likely to appear in a scene, we can use the median of the previous n frames as the background model:

$$B(x, y, t) = median\{I(x, y, t-i)\}$$
$$\Downarrow$$
$$|I(x, y, t) - median\{I(x, y, t-i)\}| > Th \text{ where } i \in \{0, \ldots, n-1\}.$$

- For $n = 10$:

Estimated Background

Foreground Mask

Slide credit: Birgi Tamersoy

Comparison

Mean

Median

Alyosha Efros, CMU

Background Subtraction Result

-

=

Alyosha Efros, CMU

Background Subtraction

Advantages:

- Extremely easy to implement and use!
- All pretty fast.
- Corresponding background models need not be constant, they change over time.

Disadvantages:

- Accuracy of frame differencing depends on object speed and frame rate
- Median background model: relatively high memory requirements.
- Setting global threshold Th…

How could this approach be better?

Slide credit: Birgi Tamersoy

Discriminative Models

- Use adaptive models of both foreground and background to estimate p(foreground)
  - Color histograms
  - Segmentation algorithms
  - etc.
- Potential problem: poor separation of foreground and background causes poor update to discriminative models
  - Drift

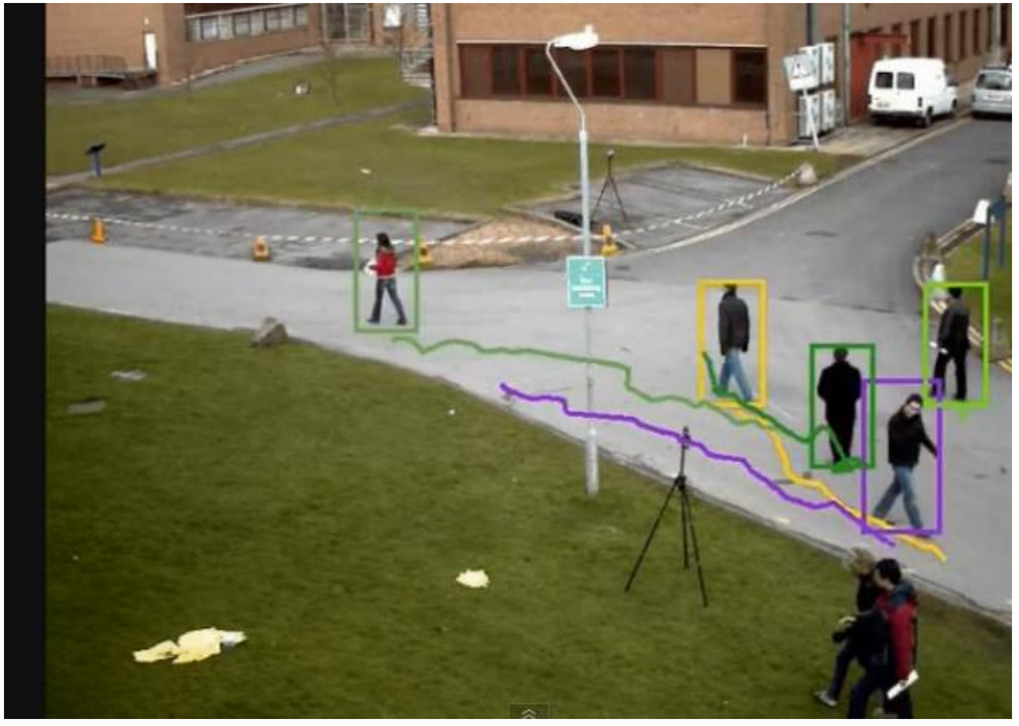
Outline

- Feature tracking
- Object tracking
  - Foreground estimation
  - Model update

Object Tracking

- Given Sequence of Images
- Track centers of moving foreground objects

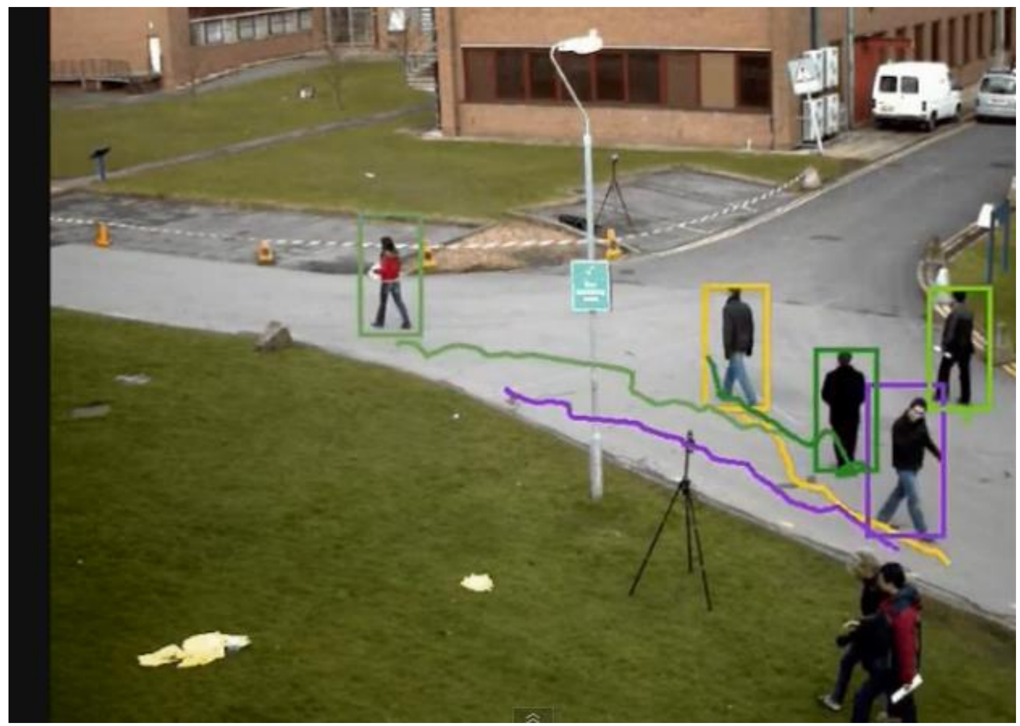
Why isn't foreground estimation enough?



http://www.youtube.com/watch?v=bY8qGk45WxM

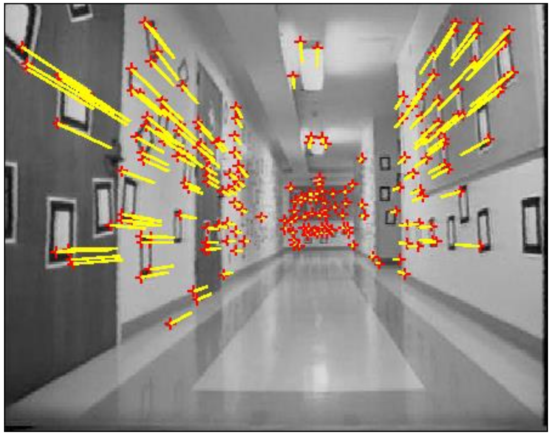
Object Tracking

- Use *model* of object motion to compensate for noisy foreground estimation, handle occlusions, keep track of multiple foreground objects, etc.



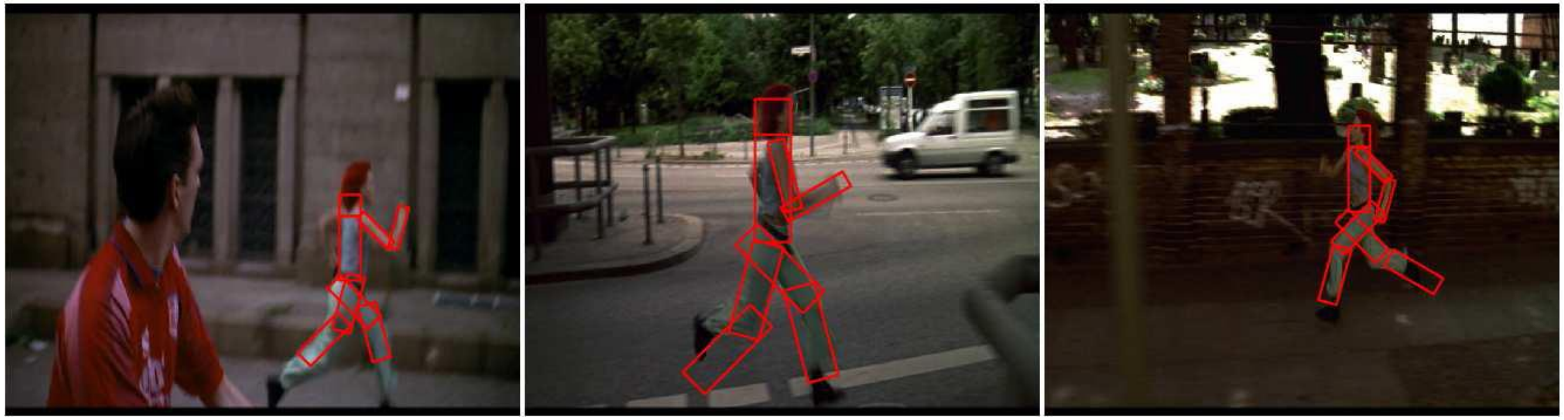
http://www.youtube.com/watch?v=bY8qGk45WxM

Example Object Models



points

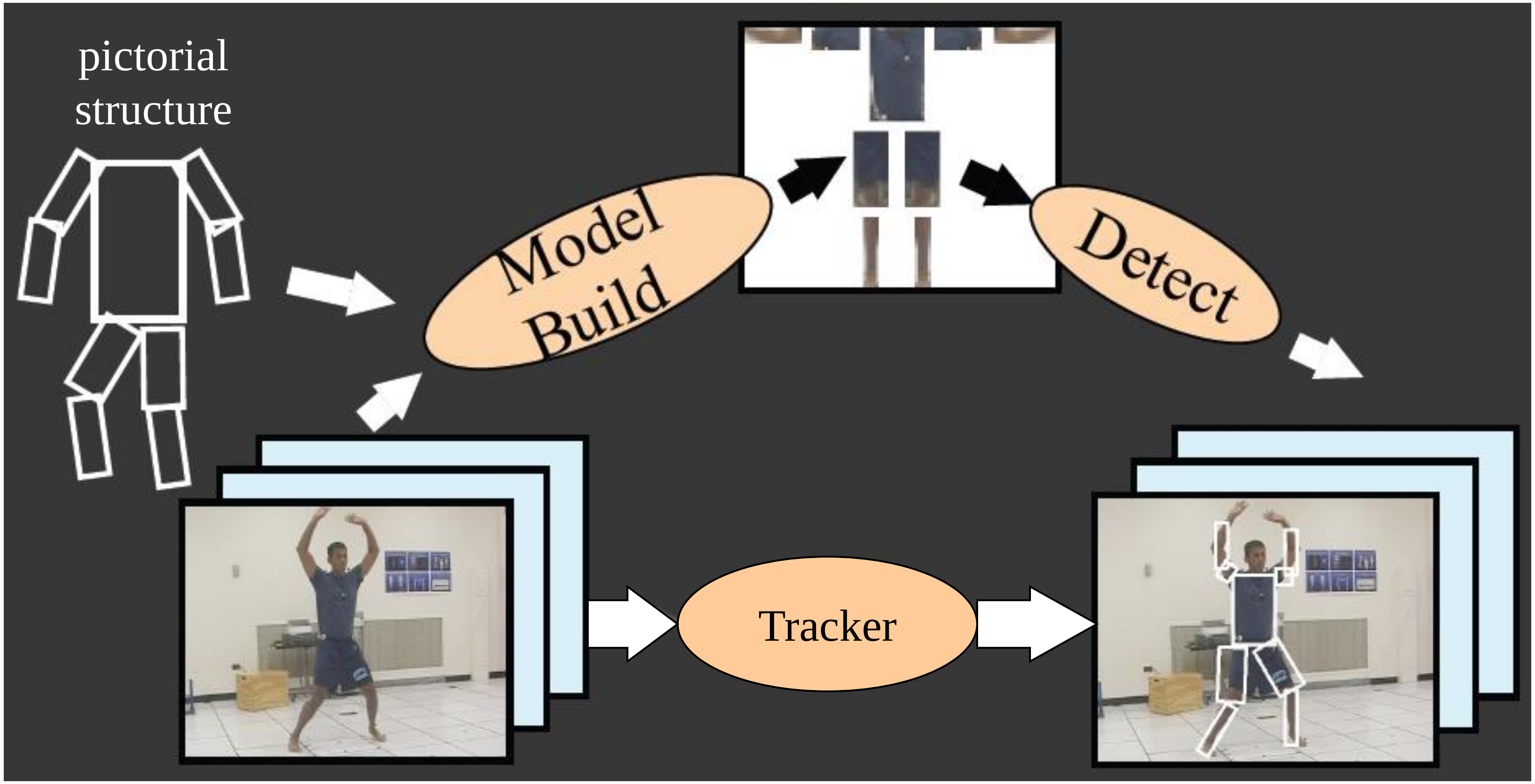
curves



Part-based structures

Kristen Grauman

Example Object Models



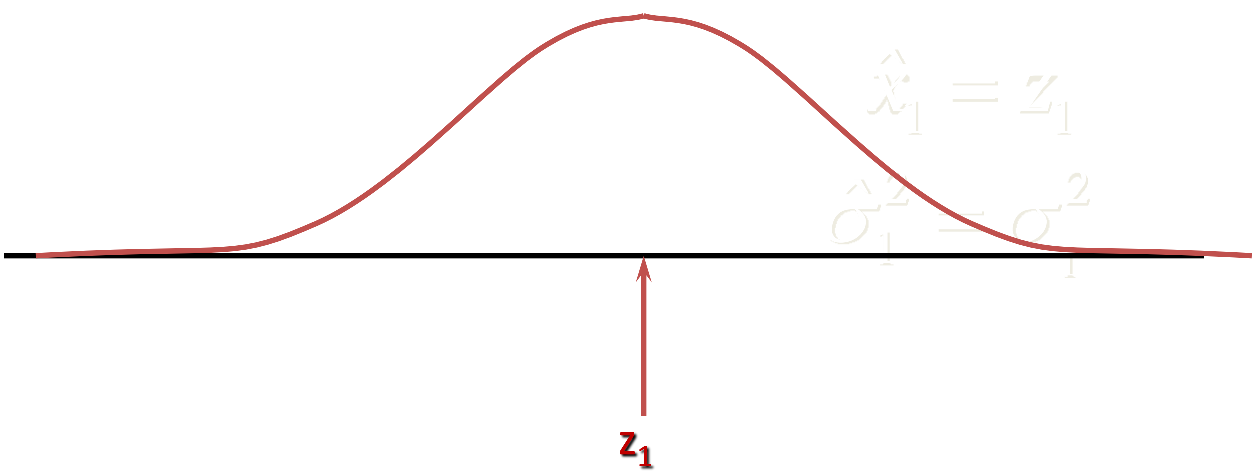
D. Ramanan, D. Forsyth, and A. Zisserman. Tracking People by Learning their Appearance. PAMI 2007.

Model Update

- Continually update probabilistic parameters of the object model based on observations (foreground estimates)

Simple Example

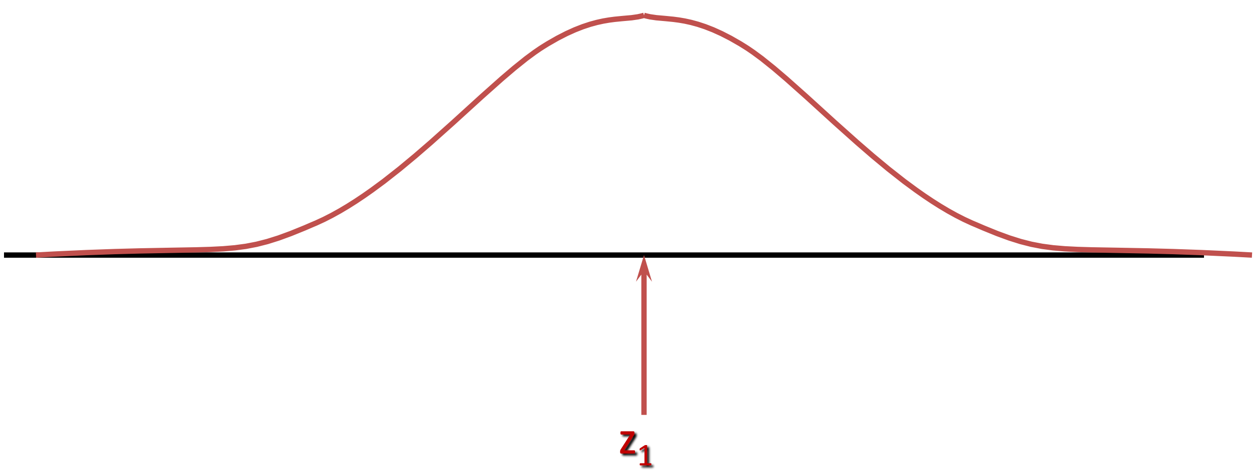
- Measurement of a single point z_1
- Variance σ_1^2 (uncertainty σ_1)
  - Assuming Gaussian distribution



Simple Example

- Measurement of a single point z_1, variance σ_1^2
  - Assuming Gaussian distribution

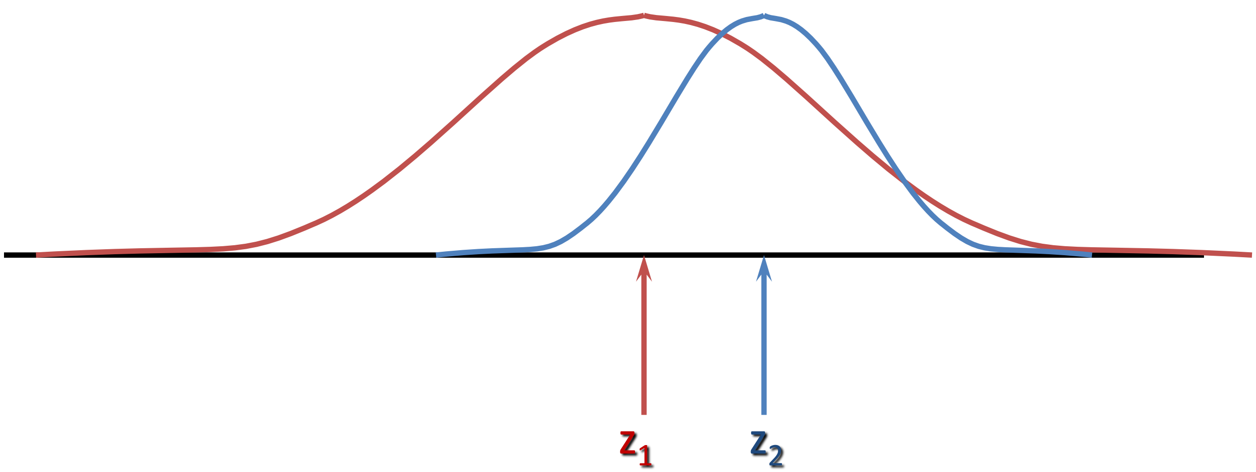
- Best estimate of true position: $\hat{x}_1 = z_1$
- Uncertainty in best estimate: $\hat{\sigma}_1^2 = \sigma_1^2$



Simple Example

- Second measurement z_2, variance σ_2^2

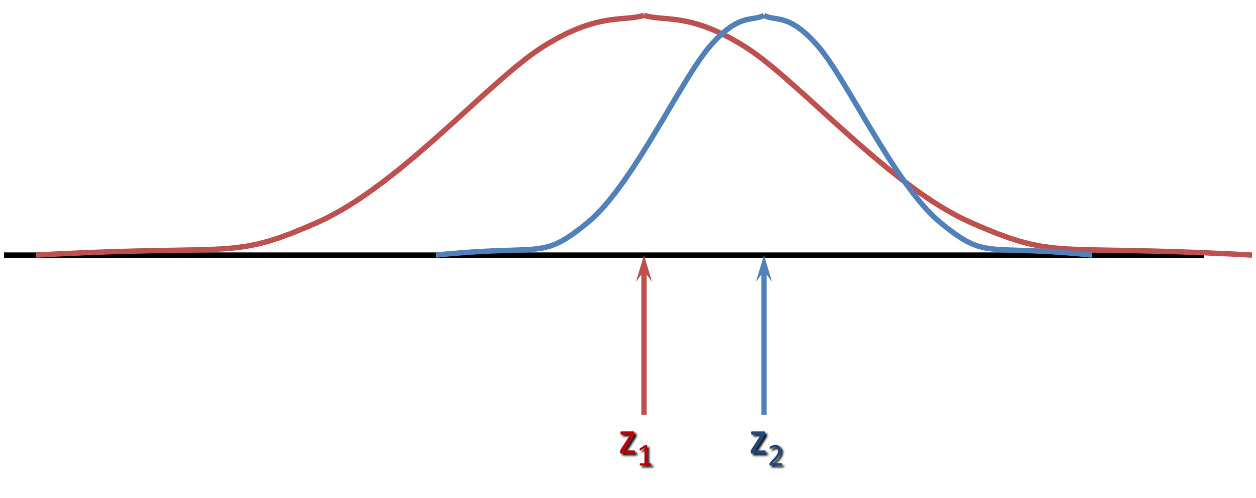
- Best estimate of true position?
- Uncertainty in best estimate?



Simple Example

- Second measurement z_2, variance σ_2^2

- Best estimate of true position?
- Uncertainty in best estimate?



Simple Example

- 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

$$\hat{\sigma}_2^2 = \frac{1}{\frac{1}{\hat{\sigma}_1^2} + \frac{1}{\sigma_2^2}}$$

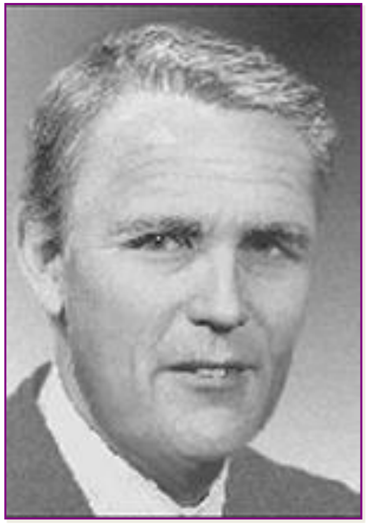
Possible Update Strategy

- 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

Kalman Filter

- Assume measurement errors are Gaussian
- Assume model parameters are Gaussian
- Assume model is linear

- Kalman filter provides optimal update strategy



Rudolf Emil Kalman

Kalman Filter Terminology

- System model:

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

- $\hat{x}_k$ is estimate of state $\hat{x}$ with covariance *P*
- The matrix Φ_k is *state transition matrix*
- The vector ξ_k represents *additive noise*, assumed to have covariance *Q*

Kalman Filter Terminology

- 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

Kalman Filter Update

- Predict new state

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

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

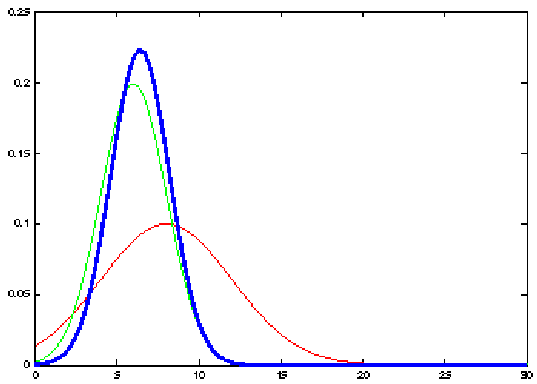
- Correct model based on new measurements

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

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

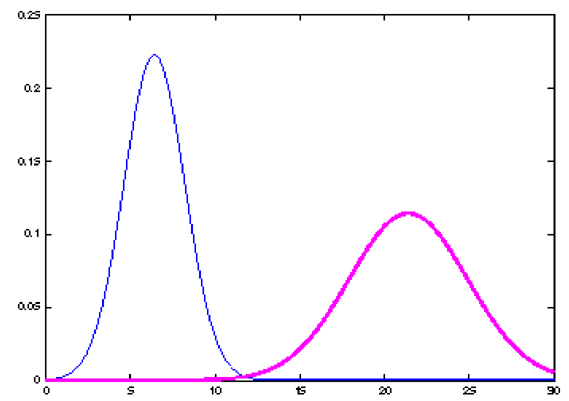
The Prediction-Correction-Cycle

Prediction



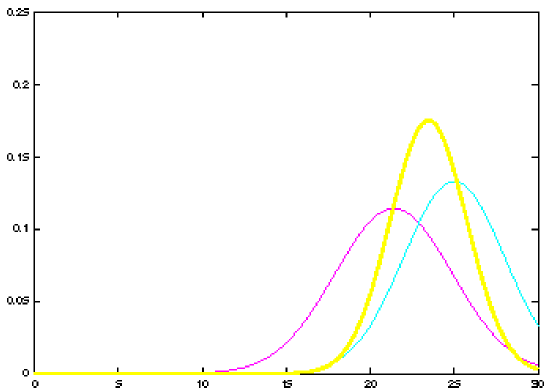
$$\overline{bel}(x_t) = \begin{cases} \bar{\mu}_t = a_t \mu_{t-1} + b_t u_t \\ \bar{\sigma}_t^2 = a_t^2 \sigma_t^2 + \sigma_{act,t}^2 \end{cases}$$

$$\overline{bel}(x_t) = \begin{cases} \bar{\mu}_t = A_t \mu_{t-1} + B_t u_t \\ \bar{\Sigma}_t = A_t \Sigma_{t-1} A_t^T + R_t \end{cases}$$



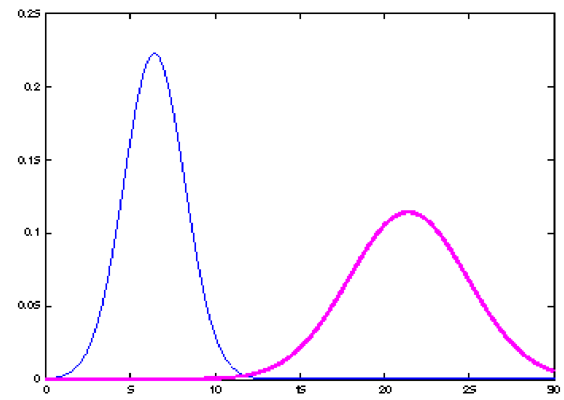
Thrun & Košecká

The Prediction-Correction-Cycle



$$bel(x_t) = \begin{cases} \mu_t = \bar{\mu}_t + K_t(z_t - \bar{\mu}_t) \\ \sigma_t^2 = (1 - K_t)\bar{\sigma}_t^2 \end{cases}, K_t = \frac{\bar{\sigma}_t^2}{\bar{\sigma}_t^2 + \bar{\sigma}_{obs,t}^2}$$

$$bel(x_t) = \begin{cases} \mu_t = \bar{\mu}_t + K_t(z_t - C_t\bar{\mu}_t) \\ \Sigma_t = (I - K_t C_t)\bar{\Sigma}_t \end{cases}, K_t = \bar{\Sigma}_t C_t^T (C_t \bar{\Sigma}_t C_t^T + Q_t)^{-1}$$



Correction

Thrun & Košecká

The Prediction-Correction-Cycle

Prediction

$$bel(x_t) = \begin{cases} \mu_t = \bar{\mu}_t + K_t(z_t - \bar{\mu}_t) \\ \sigma_t^2 = (1 - K_t)\bar{\sigma}_t^2 \end{cases}, K_t = \frac{\bar{\sigma}_t^2}{\bar{\sigma}_t^2 + \bar{\sigma}_{obs,t}^2}$$

$$\overline{bel}(x_t) = \begin{cases} \bar{\mu}_t = a_t \mu_{t-1} + b_t u_t \\ \bar{\sigma}_t^2 = a_t^2 \sigma_t^2 + \sigma_{act,t}^2 \end{cases}$$

$$bel(x_t) = \begin{cases} \mu_t = \bar{\mu}_t + K_t(z_t - C_t \bar{\mu}_t) \\ \Sigma_t = (I - K_t C_t)\bar{\Sigma}_t \end{cases}, K_t = \bar{\Sigma}_t C_t^T (C_t \bar{\Sigma}_t C_t^T + Q_t)^{-1}$$

$$\overline{bel}(x_t) = \begin{cases} \bar{\mu}_t = A_t \mu_{t-1} + B_t u_t \\ \bar{\Sigma}_t = A_t \Sigma_{t-1} A_t^T + R_t \end{cases}$$

Correction

Thrun & Košecká

Kalman Filter Summary

- *Highly efficient*: Polynomial in measurement dimensionality k and state dimensionality n:

$$O(k^{2.376} + n^2)$$

- *Optimal for linear Gaussian systems!*

Thrun & Košecká

Kalman Filter

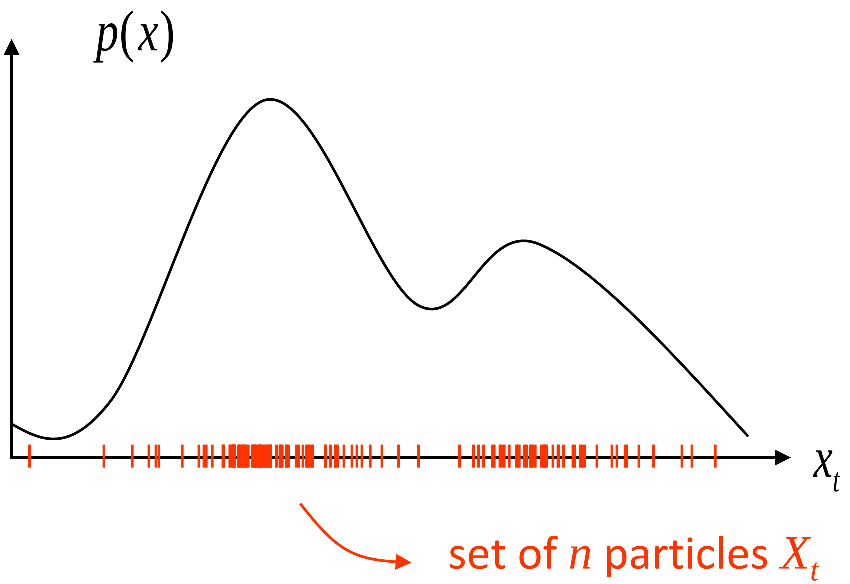
- Problems:

  What if model of motion is not linear?

  What if it is not even parametric?

Particle Filter

- Basic idea: model is represented by a population of particles (X_t)



Thrun & Košecká

Particle Filter Algorithm

Initialization:

$X_0 \leftarrow$ n particles $x_0^{[i]} \sim p(x_0)$

particleFilters(X_{t-1}) {

for i=1 to n

$x_t^{[i]} \sim \mathrm{p}(x_t \mid x_{t-1}{}^{[i]})$ (prediction)

$w_t^{[i]} = \mathrm{p}(z_t \mid x_t^{[i]})$ (importance weights)

endfor

for i=1 to n

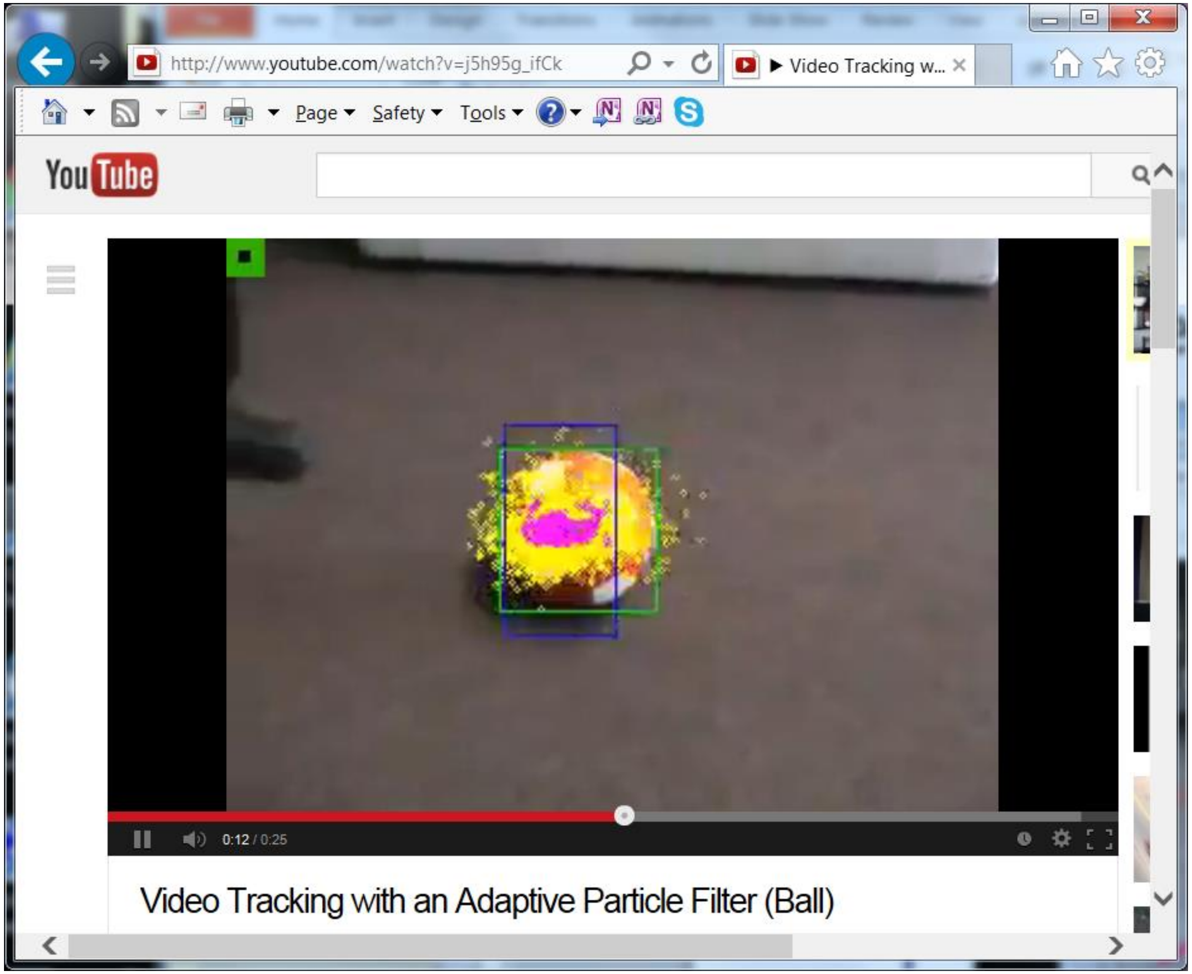
include $x_t^{[i]}$ in X_t with probability $\propto w_t^{[i]}$ (resampling)

endfor

}

Thrun & Košecká

Particle Filter Example

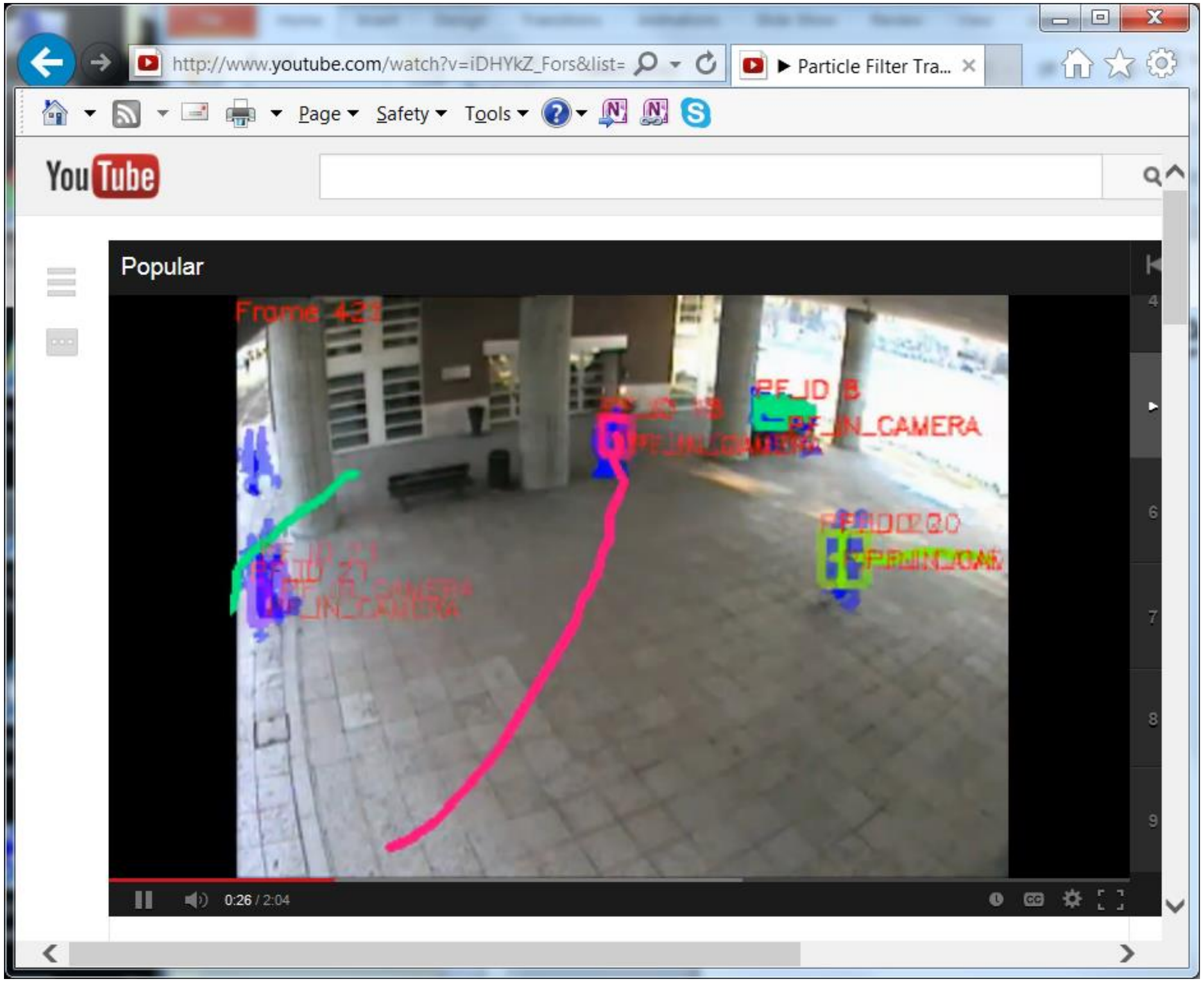


http://www.youtube.com/watch?v=j5h95g_ifCk

Particle Filter Example

http://www.youtube.com/watch?v=wCMk-pHzScE

Particle Filter Example



http://www.youtube.com/watch?v=iDHYkZ_Fors&list=TLctNoIqoeYKI_Mjy-26deFuPWKddjMI4P

Kalman Filter

- Estimates state of a system
  - Position
  - Velocity
  - Many other continuous state variables possible
- KF maintains
  - Mean vector for the state
  - Covariance matrix of state uncertainty
- Implements
  - Time update = prediction
  - Measurement update
- Standard Kalman filter is linear-Gaussian
  - Linear system dynamics, linear sensor model
  - Additive Gaussian noise (independent)
  - Nonlinear extensions: extended KF, unscented KF: linearize

Particle Filter

- Estimates state of a system
  - Position
  - Velocity
  - Many other continuous and discrete state variables possible
- KF maintains
  - ~~Mean vector for the state~~
  - ~~Covariance matrix of state uncertainty~~

  set of particles (example states)
- Implements
  - Time update = prediction = predictive sampling
  - Measurement update = resampling, importance weights
- ~~Standard Kalman filter is linear-Gaussian~~
  - ~~Linear system dynamics, linear sensor model~~
  - ~~Additive Gaussian noise (independent)~~
  - ~~Nonlinear extensions: extended KF, unscented KF: linearize~~

fully nonlinear

easy to implement

Thrun & Košecká

Summary

- Feature tracking
- Object tracking
  - Foreground estimation
  - Model update
    - Kalman filter
    - Particle filter