Particle Filters and Applications in Computer Vision

Désiré Sidibé

Maître de Conférences - Université de Bourgogne
LE2I - UMR CNRS 5158
dro-desire.sidibe@u-bourgogne.fr

April 6th 2011

Outline

Outline

Introduction

Particle Filters : two words

- **Filter** : a procedure that estimates parameters (state) of a system.
- **Particles** : a set of randomly chosen weighted samples used to approximate a pdf.

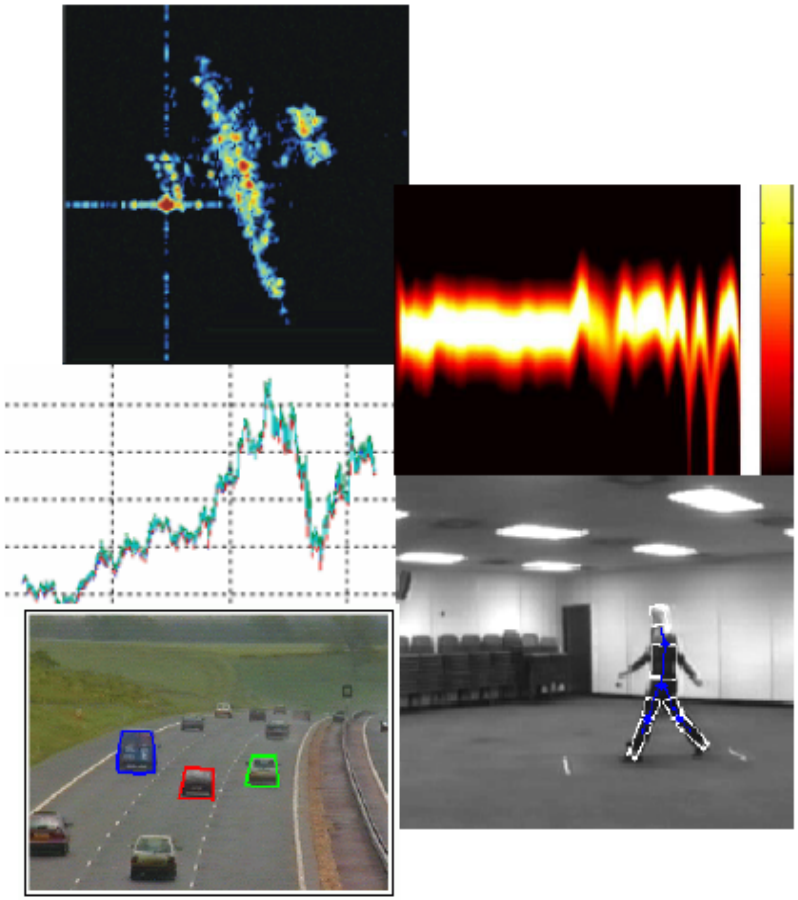
Estimation

Estimation *is the process by which we infer the value of a quantity of interest,* **x**, *by processing data that is in some way dependent on* **x**.

Introduction

State estimation or filtering has many applications

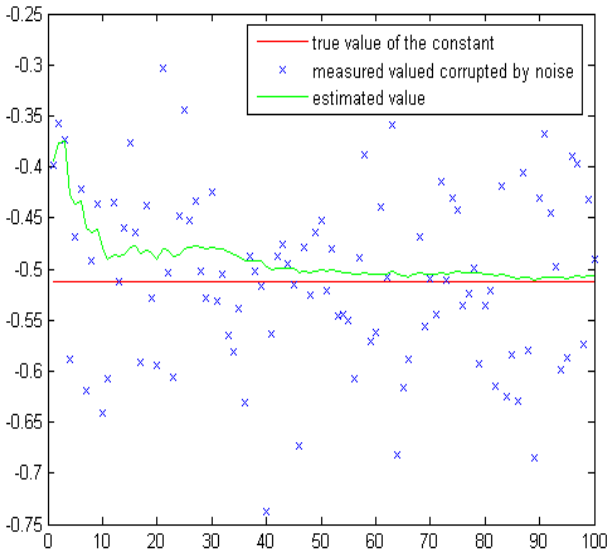
- Estimating communication signals from noisy measurements
- Predicting economical data
- Tracking of aircraft positions from radar
- Tracking of people or cars in surveillance videos
- Mobile robotics
- etc.



Introduction

Goals of this presentation :

- State the problem
- Introduce the key ideas
- Show examples of applications in computer vision



Outline

The problem

Find the *best* estimate $\hat{\mathbf{x}}$ for a parameter $\mathbf{x}$ given a set of k measurements $\mathbf{Z}_{1:k} = \{\mathbf{z}_1, \ldots, \mathbf{z}_k\}$.

- State estimation is based on probability theroy.
- One of the *'most important'* results in probability theory is Bayes Rule :

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

$$\text{posterior} = \frac{\text{likelihood} \times \text{prior}}{\text{evidence}}$$

=> simple but powerful !

Introduction

Using Bayes rule, the *posterior* pdf is given by

$$\begin{aligned} p(\mathbf{x}|\mathbf{Z}_{1:k}) &= \frac{p(\mathbf{Z}_{1:k}|\mathbf{x})p(\mathbf{x})}{p(\mathbf{Z}_{1:k})} \\ &\propto \eta \times p(\mathbf{Z}_{1:k}|\mathbf{x})p(\mathbf{x}) \end{aligned}$$

$p(\mathbf{x}|\mathbf{Z}_{1:k})$: the **posterior** pdf
$p(\mathbf{Z}_{1:k}|\mathbf{x})$: the **likelihood** function
$p(\mathbf{x})$: the **prior** distribution

"'*The probability of any event is the ratio between the value at which an expectation depending on the happening of the event ought to be computed, and the value of the thing expected upon its happening.*"'
Thomas Bayes (1702-1761)

Maximum Likelihood Estimation

If we have no prior knowledge about the parameter $\mathbf{x}$:

$$p(\mathbf{x}|\mathbf{Z}_{1:k}) \propto \eta \times p(\mathbf{Z}_{1:k}|\mathbf{x})$$

Then, the estimate $\hat{\mathbf{x}}$ is given by the value of $\mathbf{x}$ which maximizes the likelihood function :

$$\hat{\mathbf{x}}_{ML} = \arg\max_{\mathbf{x}} p(\mathbf{Z}_{1:k}|\mathbf{x})$$

Maximum A-Posteriori Estimation

In many cases we have some prior knowledge on $\mathbf{x}$ represented by $p(\mathbf{x})$:

$$p(\mathbf{x}|\mathbf{Z}_{1:k}) \propto \eta \times p(\mathbf{Z}_{1:k}|\mathbf{x})p(\mathbf{x})$$

Then, the estimate $\hat{\mathbf{x}}$ is given by the value of $\mathbf{x}$ which maximizes the posterior distribution :

$$\hat{\mathbf{x}}_{MAP} = \arg\max_{\mathbf{x}} p(\mathbf{Z}_{1:k}|\mathbf{x})p(\mathbf{x})$$

MMSE Estimation

Another key technique for estimating the value of a random variable $\mathbf{x}$ is the minimum mean squared error estimation :

$$\hat{\mathbf{x}}_{MMSE} = \arg\min_{\hat{\mathbf{x}}} \mathbb{E}\{(\hat{\mathbf{x}} - \mathbf{x})^T(\hat{\mathbf{x}} - \mathbf{x})|\mathbf{Z}_{1:k}\}$$

It is easy to show that :

$$\hat{\mathbf{x}}_{MMSE} = \mathbb{E}\{\mathbf{x}|\mathbf{Z}_{1:k}\}$$

Recursive Estimation

If the measurements are obtained sequentially over time, we could use the previous estimate of $\mathbf{x}$ as the prior knowledge.

$$p(\mathbf{x}|\mathbf{Z}_{1:k}) = \frac{p(\mathbf{Z}_{1:k}|\mathbf{x})p(\mathbf{x}|\mathbf{Z}_{1:k-1})}{p(\mathbf{z}_k|\mathbf{Z}_{1:k-1})}$$

Introduction

We can distinguish three estimation problems :

- **Filtering** : estimate the *current* state $\mathbf{x}(k)$ given all the data availabe up to and including $\mathbf{z}(k)$.
- **Smoothing** : estimate some *past* value of $\mathbf{x}(l)$, $l < k$, given all the data availabe up to and including $\mathbf{z}(k)$.
- **Prediction** : estimate some *future* value of $\mathbf{x}(l)$, $l > k$, given all the data availabe up to and including $\mathbf{z}(k)$.

Outline

What Is A Particle Filter ?

Recursive filter

- Sequential update of previous estimate
  $\Rightarrow$ faster online data processing as opposed to batch processing

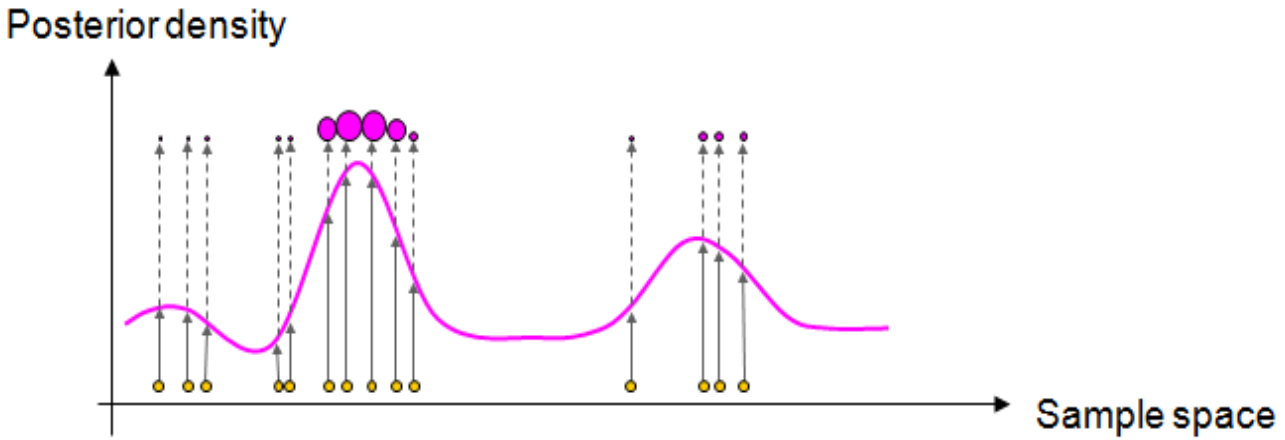
Approximate method

- It is impossible to obtain analytic solutions to non-linear and non-Gaussian estimation problems
  $\Rightarrow$ particle filters provide approximate solutions

What Is A Particle Filter ?

Particle filter is :

- a technique for implementing recursive Bayesian filter by Monte Carlo sampling.
- the idea is to represent the posterior density by a set of random particles with associated weights ;
- and compute estimates based on these samples and weights



Outline

Visual Tracking

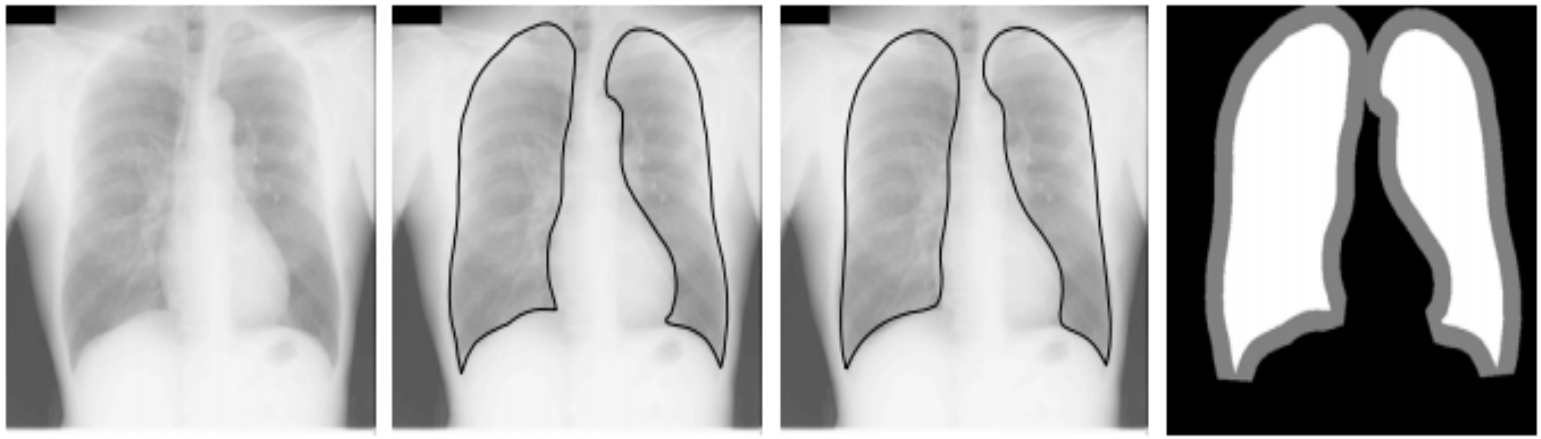
- Particle filters were first introduced to the computer vision community by the famous CONDENSATION paper of Isard and Blake.
  $\Rightarrow$ real-time contour tracking

M. Isard and A. Blake, *CONDENSATION - conditional density propagation for visual tracking*, Int. J. Computer Vision, 29, 1, 5–28, (1998)

- A lot of work has been published on contour extraction and tracking since then.

Medical Image Analysis

- Particle filters have been widely used in medical image analysis
  $\Rightarrow$ contour tracking, image segmentation

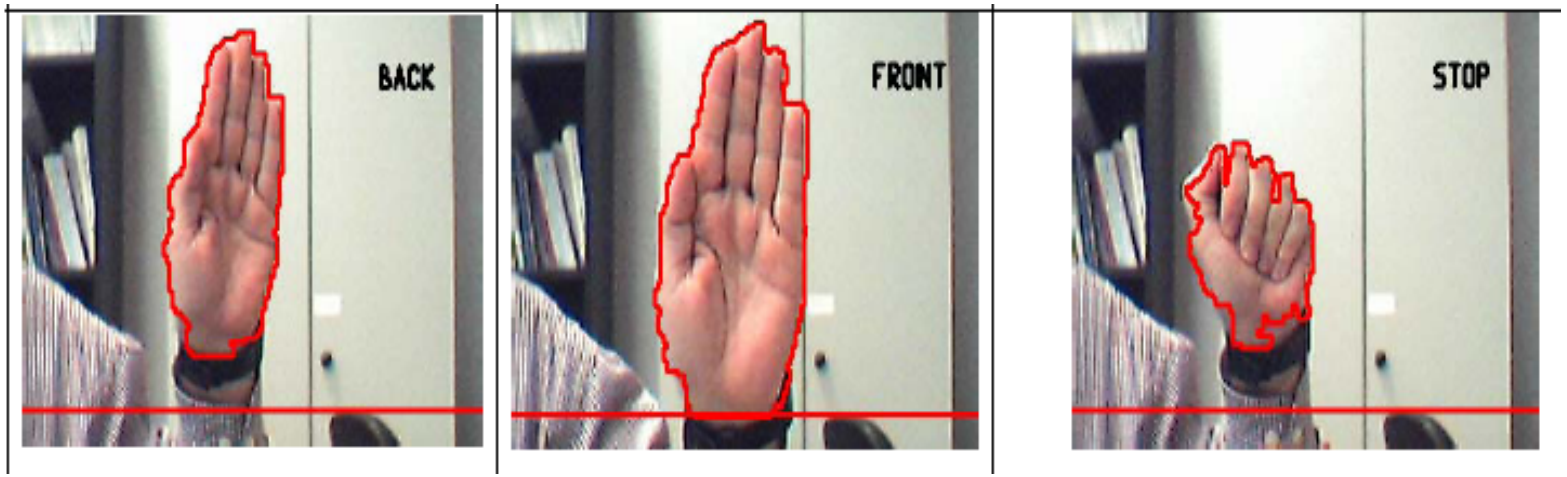


I. Smal et al., *Multiple Object Tracking in Molecular Bioimaging by Rao-Blackwellized Marginal Particle Filtering*, Medical Image Analysis, 12 (6), pp. 764–777, (2008)

M. de Bruijne and M. Nielsen, *Shape Particle Filtering for Image Segmentation*, In Proc. MICCAI, pp. 168–175, (2004)

Human Computer Interaction

- Particle filters are used in HCI for tracking
  $\Rightarrow$ face and hand tracking can be coupled with facial expression and gesture recognition



M. Bray et al., *Smart particle filtering for 3D hand tracking*, in Proc. AFGR, pp. 675–680, (2004)

M. F. Ho et al., *A multi-view vision-based hand motion capturing system*, Pattern Recognition, 44 (2), pp ; 443–453 (2011)

Image Restoration

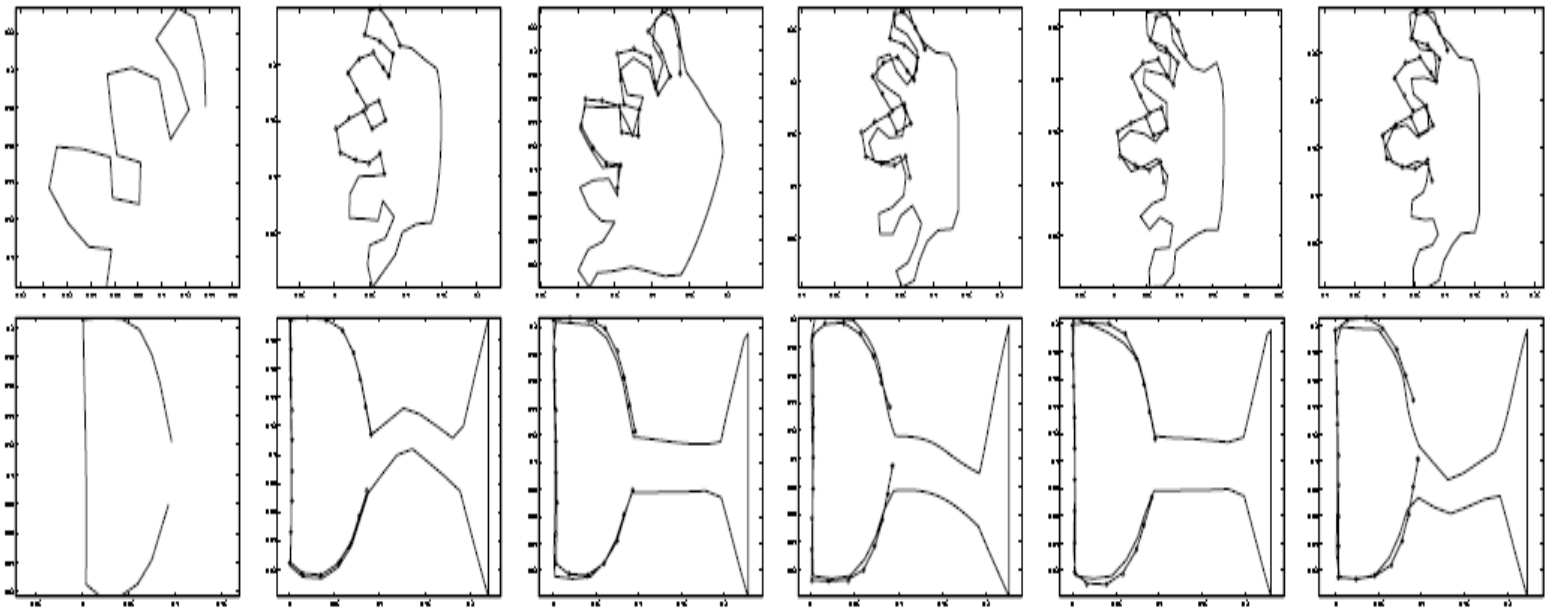
- Particle filters have been used for image enhancement and denoising

N. Azzabou, N. Paragios, and F. Guichard, *Image Reconstruction Using Particle Filters and Multiple Hypotheses Testing*, IEEE Trans. on Image Processing, 19, 5, 1181–1190, (2010)

L. Xia, *Image restoration based on particle filters*, in Proc. ICSP, pp. 1084–1087, (2004)

Shape correpondence

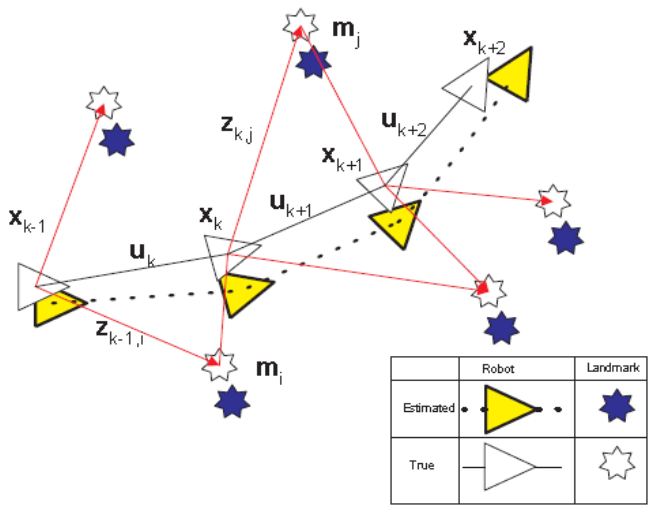
- Particle filters can also be used for shape correspondence



R. Lakaemper and M. Sobel, *Correspondences between parts of shapes with particle filters*, in Proc. CVPR, pp. 1–8, (2008)

Robot Navigation

- Particle filters are now used as a key technique for localization and mapping problems in mobile robot navigation.



The Simultaneous Localisation and Map Building (SLAM) problem requires that the probability distribution $P(\mathbf{x}_k; \mathbf{m}|\mathbf{Z}_{0:k}; \mathbf{U}_{0:k}; \mathbf{x}_0)$ be computed for all times k.

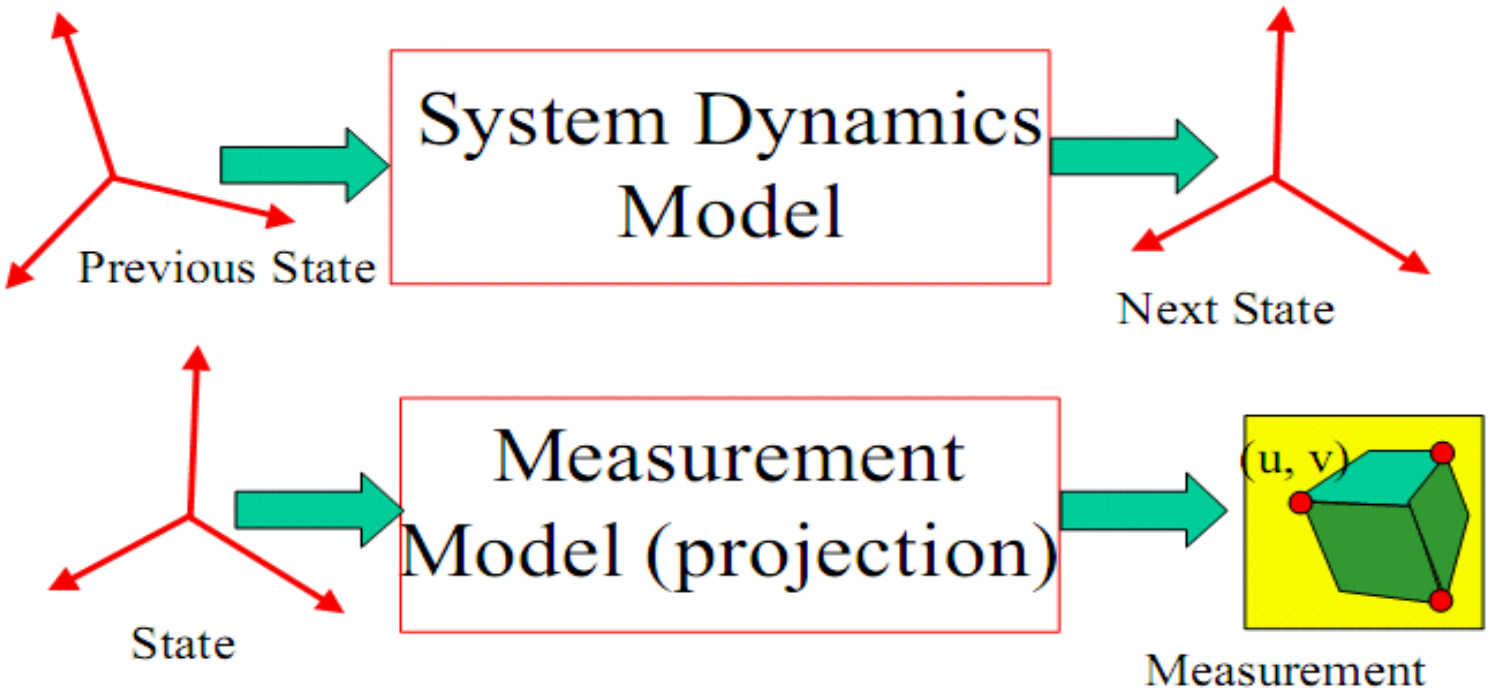
S. Thrun, W. Burgard and D. Fox, *Probabilistic Robotics*, The MIT Press, (2006)

Outline

Introduction

Recursive filtering involves a description of our a priori knowledge about the world :

- **System dynamics model** : how does system state evolves over time ?
- **Measurement model** : how are measurements related to system state ?



Notations

- System state at time k : $\mathbf{x}(k)$
- Measurement at time k : $\mathbf{z}(k)$
- All measurements up to time k : $\mathbf{Z}_{1:k} = \{\mathbf{z}(1), \ldots, \mathbf{z}(k)\}$
- System error sources at time k : $w(k)$
- Measurement error sources at time k : $v(k)$
- Control input at time k : $u(k)$

Models equations

$$\mathbf{x}_k = f(\mathbf{x}_{k-1}, u_{k-1}) + w_{k-1}, \tag{1}$$

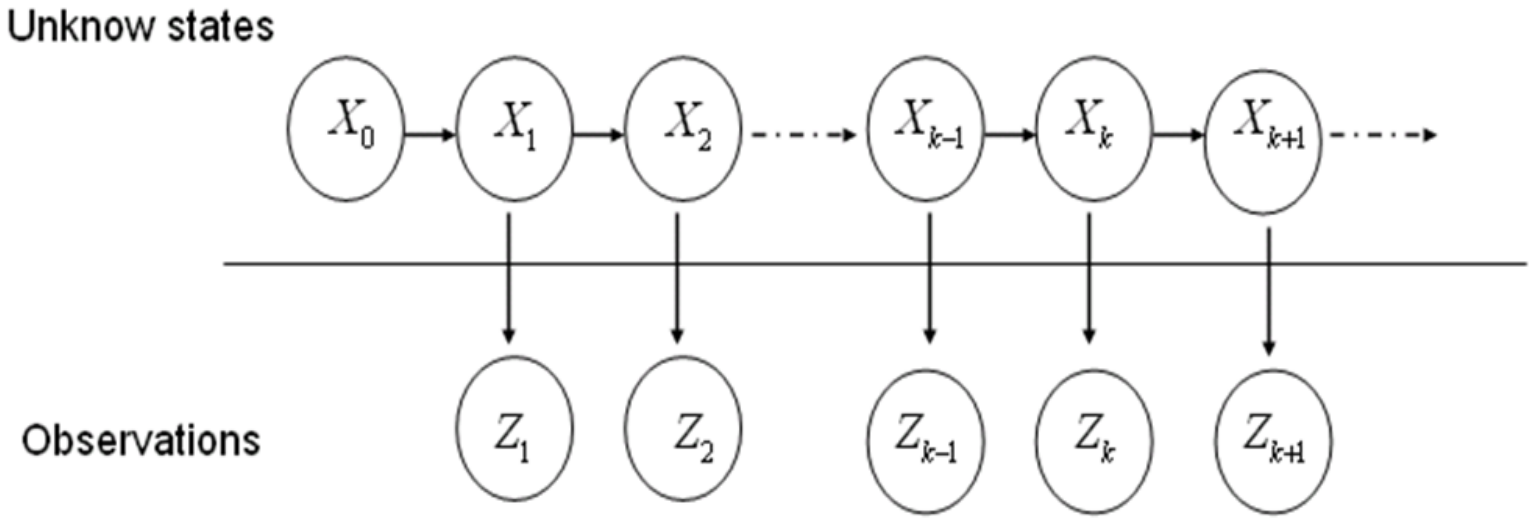
$$\mathbf{z}_k = h(\mathbf{x}_k) + v_k. \tag{2}$$

General Bayesian Framework

General model

- We have a system described by some internal state parameters **X**
- We also have a set of measurements **Z** from some devices.
- There is a clock
  - at each tick, the state changes (system state equation)
  - at each tick, we get a new observation (measurement equation)
- System state is unkown ('hidden')
- We want to recover the state components from observations
  => **state estimation problem**.

General Bayesian Framework



$$P(X_k|Z_{1:k}) = ?$$

- We want to get $P(\mathbf{X}_k|\mathbf{Z}_{1:k})$

Independence assumptions

- Hidden Markov Model

$$P(\mathbf{X}_k|\mathbf{X}_0,\ldots,\mathbf{X}_{k-1}) = P(\mathbf{X}_k|\mathbf{X}_{k-1}) \tag{3}$$

- Observations are conditionally independent given the state

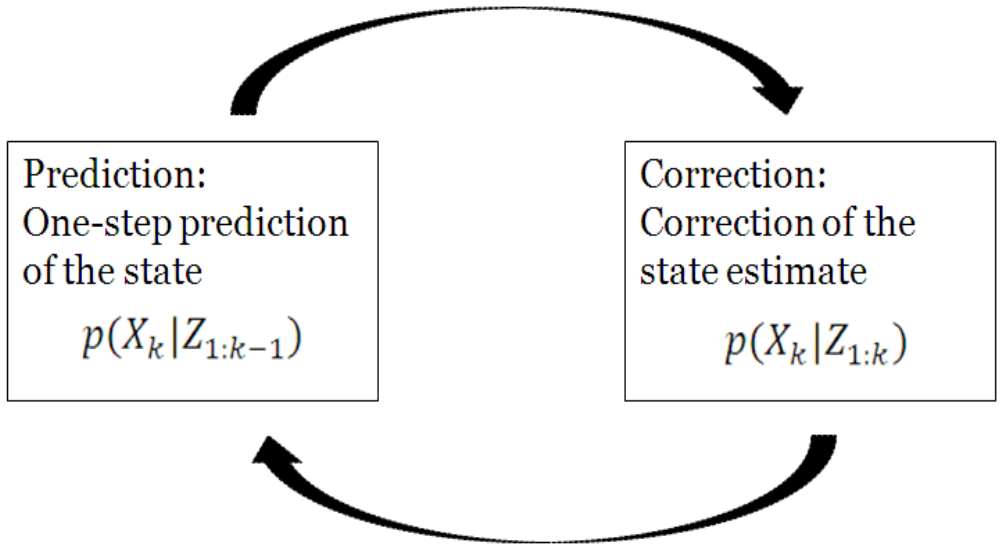
$$P(\mathbf{Z}_k,\mathbf{Z}_i,\ldots,\mathbf{Z}_j|\mathbf{X}_k) = P(\mathbf{Z}_k|\mathbf{X}_k)P(\mathbf{Z}_i,\ldots,\mathbf{Z}_j|\mathbf{X}_k) \tag{4}$$

General Bayesian Framework

Recursive estimation implies two main steps :

- **Prediction** : given measurements $\mathbf{z}_1, \ldots, \mathbf{z}_{k-1}$, what state $\mathbf{X}_k$ can we predict for the k-th tick ? We need to obtain a representation of $P(\mathbf{X}_k | \mathbf{Z}_1 = \mathbf{z}_1, \ldots, \mathbf{Z}_{k-1} = \mathbf{z}_{k-1})$.

- **Correction** : given the relevant measurement at time k, $\mathbf{z}_k$, we need to compute a representation of $P(\mathbf{X}_k | \mathbf{Z}_1 = \mathbf{z}_1, \ldots, \mathbf{Z}_k = \mathbf{z}_k)$.



General Bayesian Framework

We assume that the initial pdf $P(\mathbf{X}_0)$ is known
Then given a representation of $P(\mathbf{X}_{k-1}|\mathbf{z}_1, \ldots, \mathbf{z}_{k-1})$, we have :

Prediction

$$P(\mathbf{X}_k|\mathbf{z}_1, \ldots, \mathbf{z}_{k-1}) = \int P(\mathbf{X}_k|\mathbf{X}_{k-1}) P(\mathbf{X}_{k-1}|\mathbf{z}_1, \ldots, \mathbf{z}_{k-1}) d\mathbf{X}_{k-1}$$

Correction

$$P(\mathbf{X}_k|\mathbf{z}_1, \ldots, \mathbf{z}_k) = \frac{P(\mathbf{z}_k|\mathbf{X}_k) P(\mathbf{X}_k|\mathbf{z}_1, \ldots, \mathbf{z}_{k-1})}{\int P(\mathbf{z}_k|\mathbf{X}_k) P(\mathbf{X}_k|\mathbf{z}_1, \ldots, \mathbf{z}_{k-1}) d\mathbf{X}_k}$$

Computing these pdfs is impossible in practice ! (nasty integrals)

Prediction

$$
\begin{aligned}
P(\mathbf{X}_k|\mathbf{Z}_{1:k-1}) &= \int P(\mathbf{X}_k, \mathbf{X}_{k-1}|\mathbf{Z}_{1:k-1}) d\mathbf{X}_{k-1} \\
&= \int P(\mathbf{X}_k|\mathbf{X}_{k-1}, \mathbf{Z}_{1:k-1}) P(\mathbf{X}_{k-1}|\mathbf{Z}_{1:k-1}) d\mathbf{X}_{k-1} \\
&= \int P(\mathbf{X}_k|\mathbf{X}_{k-1}) P(\mathbf{X}_{k-1}|\mathbf{Z}_{1:k-1}) d\mathbf{X}_{k-1}
\end{aligned}
$$

Correction

$$
\begin{aligned}
P(\mathbf{X}_k|\mathbf{Z}_{1:k}) &= \frac{P(\mathbf{Z}_{1:k}|\mathbf{X}_k)P(\mathbf{X}_k)}{P(\mathbf{Z}_{1:k})} \\
&= \frac{P(\mathbf{z}_k, \mathbf{Z}_{1:k-1}|\mathbf{X}_k)P(\mathbf{X}_k)}{P(\mathbf{z}_k, \mathbf{Z}_{1:k-1})} \\
&= \frac{P(\mathbf{z}_k|\mathbf{X}_k, \mathbf{Z}_{1:k-1})P(\mathbf{Z}_{1:k-1}|\mathbf{X}_k)P(\mathbf{X}_k)}{P(\mathbf{z}_k|\mathbf{Z}_{1:k-1})P(\mathbf{Z}_{1:k-1})} \\
&= \frac{P(\mathbf{z}_k|\mathbf{Z}_{1:k-1}, \mathbf{X}_k)P(\mathbf{X}_k|\mathbf{Z}_{1:k-1})P(\mathbf{Z}_{1:k-1})P(\mathbf{X}_k)}{P(\mathbf{z}_k|\mathbf{Z}_{1:k-1})P(\mathbf{Z}_{1:k-1})P(\mathbf{X}_k)} \\
&= \frac{P(\mathbf{z}_k|\mathbf{X}_k)P(\mathbf{X}_k|\mathbf{Z}_{1:k-1})}{P(\mathbf{z}_k|\mathbf{Z}_{1:k-1})}
\end{aligned}
$$

Outline

Kalman Filters

If we restrict attention to linear dynamic systems and linear measurement models, both with additive Gaussian noise
=> all the densities are Gaussians and we can avoide the question of solving the various integrals.

Model of the system

State equation :

$$X_t = A_t X_{t-1} + B_t U_{t-1} + w_t$$

Measurement equation :

$$Z_t = C_t X_t + v_t$$

with $w_t \sim \mathcal{N}(0, Q_t)$, $v_t \sim \mathcal{N}(0, R_t)$ and $w_t \perp v_t$.

Discrete Kalman Filters

Importance of Gaussian distribution

Since the system is linear and the distributions are Gaussian, we need only to know

- the conditional mean $\widehat{X}_k = \mathbb{E}[X_k|Z_{1:k}, U_{0:k-1}]$
- the conditional covariance $P_k = Cov(X_k, X_k|Z_{1:k}, U_{0:k-1})$

Recursivity

Assume that we know $P(x_k|Z_{1:k}, U_{0:k-1})$ at some time k

1. **Prediction** : what can we say about x_{k+1} before we get the measurement Z_{k+1} ?

$$P(x_{k+1}|Z_{1:k}, U_{0:k})$$

2. **Correction** : how can we improve our knowledge of x_{k+1} given the actual measurement z_{k+1} ?

$$P(x_{k+1}|Z_{1:k+1}, U_{0:k})$$

Assume we know $\hat{x}_k$ and P_k.

Prediction step

- From the state equation :

$$\begin{aligned} \hat{x}_{k+1}^{-} &= \mathbb{E}[x_{k+1}|Z_{1:k}, U_{0:k}] \\ &= A_k \hat{x}_k + B_k u_k. \end{aligned}$$

- From the prior estimate error covariance

$$\begin{aligned} P_{k+1}^{-} &= \mathbb{E}[(x_{k+1} - \hat{x}_{k+1}^{-})(x_{k+1} - \hat{x}_{k+1}^{-})^T] \\ &= A_k P_k A_k^T + Q_k. \end{aligned}$$

Correction step

- Use the measurement equation to make a prediction of the observation

$$z_{k+1}^{-} = C_{k+1}\hat{x}_{k+1}^{-}.$$

- Given, the actual observation z_{k+1}, correct the state prediction

$$\hat{x}_{k+1} = \hat{x}_{k+1}^{-} + K_{k+1}(z_{k+1} - z_{k+1}^{-}).$$

  - $(z_{k+1} - z_{k+1}^{-})$ is called the measurement *innovation* or the *residual*.
  - K_{k+1} is called the *Kalman gain* and is chosen in order to minimize the posterior MMSE $P_{k+1} = \mathbb{E}[(x_{k+1} - \hat{x}_{k+1})(x_{k+1} - \hat{x}_{k+1})^T]$.

Optimal Kalman Gain

The a posteriori error covariance matrix is

$$P_{k+1} = \mathbb{E}[(x_{k+1} - \hat{x}_{k+1})(x_{k+1} - \hat{x}_{k+1})^T],$$

which after a few calculations gives

$$P_{k+1} = (I - K_{k+1}C_{k+1})P^-_{k+1}(I - K_{k+1}C_{k+1})^T + K_{k+1}R_{k+1}K^T_{k+1}.$$

Taking the trace of P_{k+1} and setting its derivative w.r.t. K_{k+1} equals to zero, we obtain the optimal Kalman gain

$$K_{k+1} = P^-_{k+1}C^T_{k+1}[C_{k+1}P^-_{k+1}C^T_{k+1} + R_{k+1}]^{-1}.$$

The posterior covariance matrix can then be simplified as

$$P_{k+1} = (I - K_{k+1}C_{k+1})P^-_{k+1}.$$

Kalman Filters

Algorithm Kalman-Filter(μ_{t-1}, Σ_{t-1}, u_t, z_t)

Prediction

1. $\mu_t^- = A_t \mu_{t-1} + B_t u_t$
2. $\Sigma_t^- = A_t \Sigma_{t-1} A_t^T + Q_t$

Correction

3. $K_t = \Sigma_t^- C_t^T (C_t \Sigma_t^- C_t^T + R_t)^{-1}$
4. $\mu_t = \mu_t^- + K_t (z_t - C_t \mu_t^-)$
5. $\Sigma_t = (I - K_t C_t) \Sigma_t^-$

Return μ_t and Σ_t

KF recursively predicts and updates the mean μ_t and the covariance matrix Σ_t of the Gaussian pdf.

Discrete Kalman Filters

In most of our applications, the matrices A_k, B_k and C_k do not vary over time. So, the Kalman filter equations are simplified as :

Prediction step

$$\hat{x}_{k+1}^{-} = A\hat{x}_k + Bu_k$$

$$P_{k+1}^{-} = AP_kA^T + Q_k$$

Correction step

$$K_{k+1} = P_{k+1}^{-}C^T[CP_{k+1}^{-}C^T + R_{k+1}]^{-1}$$

$$\hat{x}_{k+1} = \hat{x}_{k+1}^{-} + K_{k+1}(z_{k+1} - C\hat{x}_{k+1}^{-})$$

$$P_{k+1} = (I - K_{k+1}C)P_{k+1}^{-}$$

Discrete Kalman Filter Algorithm

1. Start : $\hat{x}_0^-$, P_0^-
2. Compute Kalman gain :

$$K_k = P_k^- C^T (C P_k^- C^T + R_k)^{-1}$$

3. Compute state estimate :

$$\hat{x}_k = \hat{x}_k^- + K_k (z_k - C\hat{x}_k^-)$$

4. Update error covariance :

$$P_k = (I - K_k C) P_k^-$$

5. Update prediction and predicted error covariance :

$$\hat{x}_{k+1}^- = A\hat{x}_k + Bu_k$$

$$P_{k+1}^- = AP_k A^T + Q_k$$

6. Do $k = k + 1$ and go to step 2.

Discrete Kalman Filter Example

Estimation of a scalar random constant

Let's assume that we have the ability to make measurements of the constant, but the measurement are corrupted by a white noise.

System equation

$$\begin{aligned} x_k &= Ax_{k-1} + Bu_k + w_k \\ &= x_{k-1} + w_k \end{aligned}$$

Measurement equation

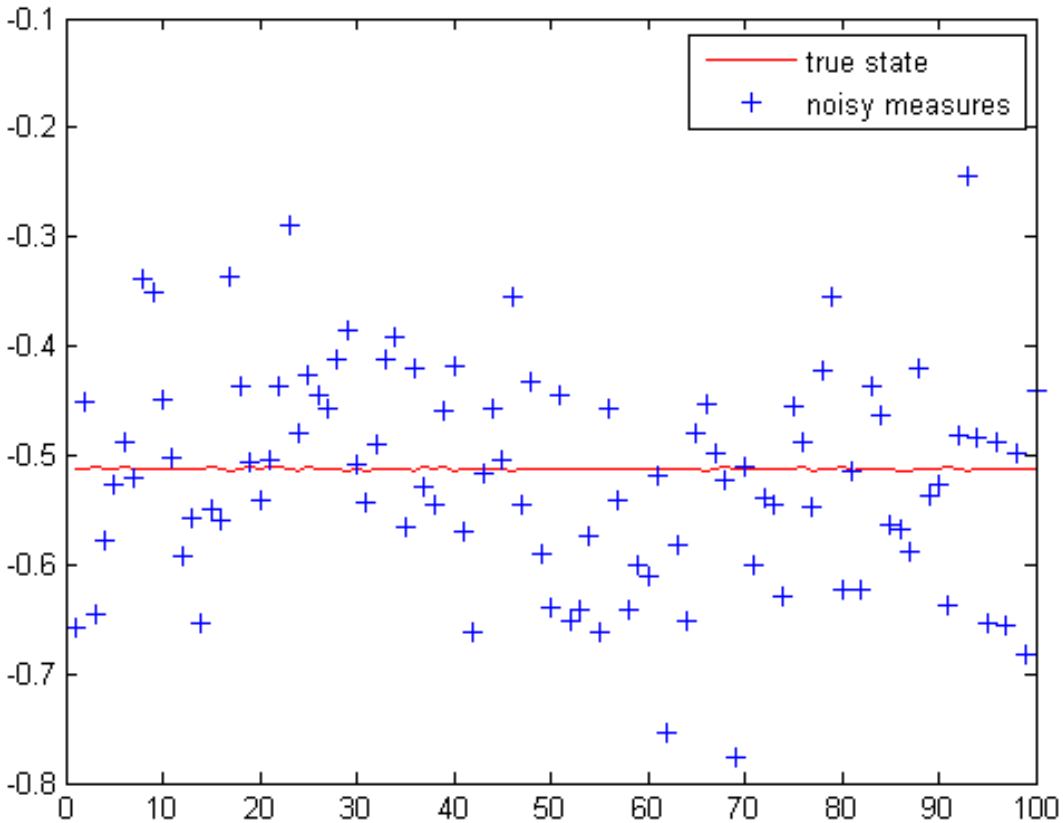
$$\begin{aligned} z_k &= Cx_k + v_k \\ &= x_k + v_k \end{aligned}$$

Discrete Kalman Filter Example

```
% the state-space equations
A = 1; % transition matrix
B = 0; % input matrix: no control input
C = 1; % measurement matrix: noisy measurement is the state
% calculation of noise covariance matrices
R = Meas_Noise^2; % measurement noise cov
Q = Process_Noise^2;  % process noise cov
% initialization
x = -0.51234;  % true state
% simulation
x_out = zeros(duration,1);
y_out = zeros(duration,1);
Process_Noise = Process_Noise * randn(duration,1);
Meas_Noise = Meas_Noise * randn(duration,1);
for t=1:duration,
    x_out(t) = A*x + Process_Noise(t);
    y_out(t) = C*x + Meas_Noise(t);
end
```

Discrete Kalman Filter Example

We randomly chose a scalar constant $z = -0.51234$
We assume a very small process noise : $Q_k = 1e-5$
We assume that measurements are corrupted by a 0.1 noise $R_k = 0.01$

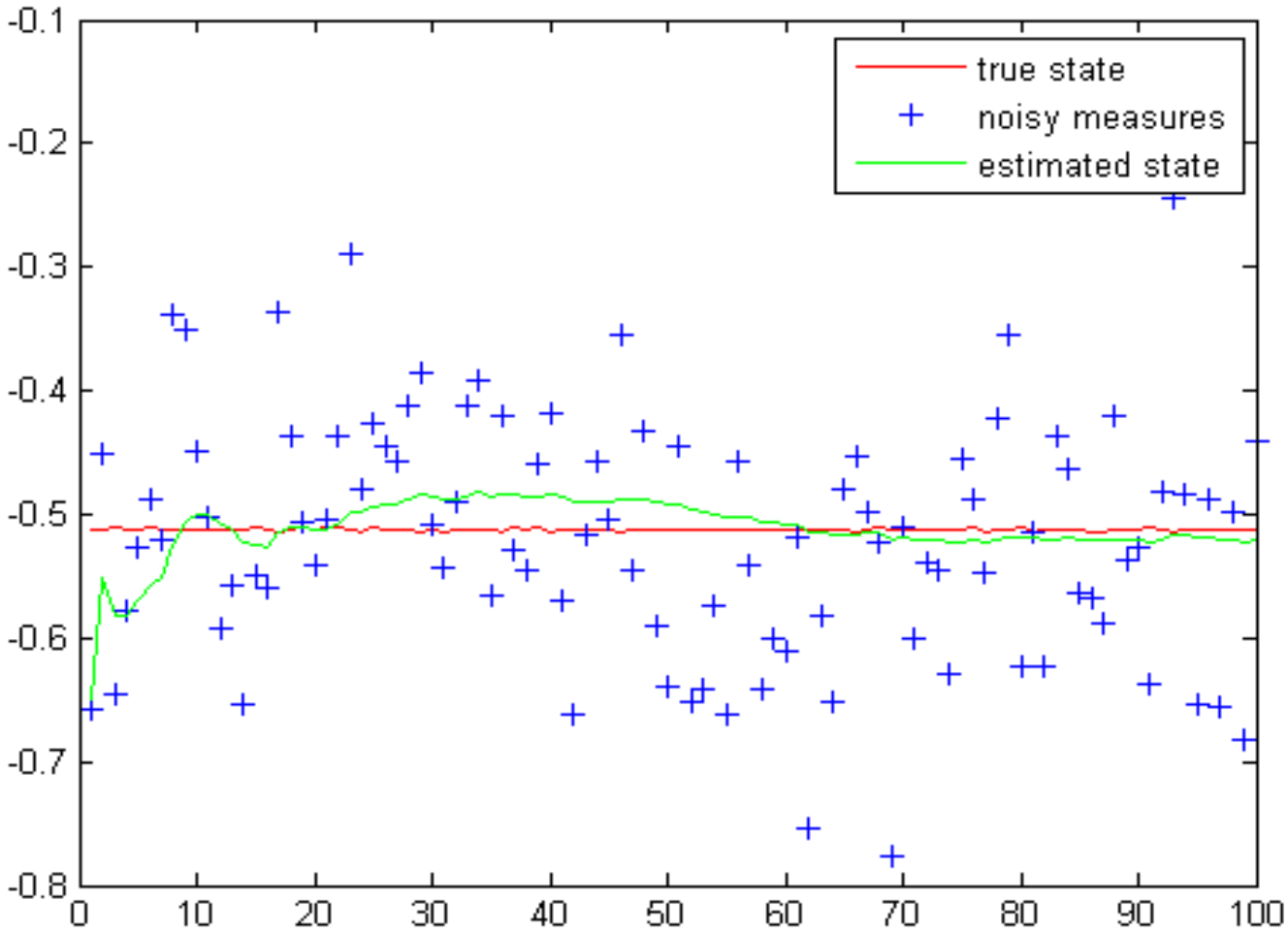


Discrete Kalman Filter Code

```
function [x_out]=discrete_kalman_filter(y, u, x0, P0, A,B,C,Q,R)
T = size(y, 1); %number of observations
[n,m] = size(P0); I = eye(n,m); % form identity matrix
xhat = x0; P = P0; %initialization
x_out = [];
for k=1:T,
   % compute Kalman Gain
   K = P*C'* inv(C*P*C' + R);
   % estimate state
   innovation = y(k) - C*xhat; % innovation vector
   xhat = xhat + K*innovation;
   x_out = [x_out; xhat'];
   % update covariance matrice
   P = (I - K*C)*P;
   % predict next state and covariance
   xhat = A*xhat + B*u;
   P = A*P*A' + Q;
end
```

Discrete Kalman Filter Example

Given that $Q_k = 1e-5$ and $R_k = 0.01$, we initialize the filter with $\hat{x}_0 = 0$ and $P_0 = 1$.



Kalman Filters

Extensions of the Kalman Filter

- For *non-linear* dynamic models with Gaussian noises : Extended KF ; Unscented KF.
- For *non-linear, non-Gaussian* distributions : Particle filters

Outline

Particle Filters

Why particle filters ?

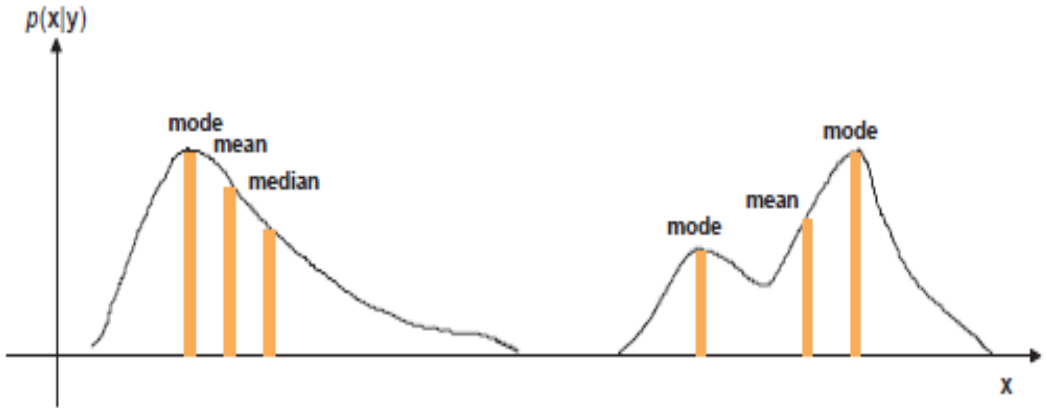
- Kalman filter is limited to linear system with Gaussian noise
- Extended KF and Unscented KF are limited to Gaussian distributions
- Many practical problems show non linear dynamics and non Gaussian noises
- We often find multi-modal distributions

Key Idea

Find an approximate solution using a complex model rather than an exact solution using a simplified model.

Particle Filters

Muti-modal distribution



A Gaussian distribution is unimodal and verifies : mode = mean = median
For a multimodal distribution, the mode, mean and median do not coincide.
=> Kalman Filter, clearly, is not suitable for multi-modal distributions.

Particle Filters

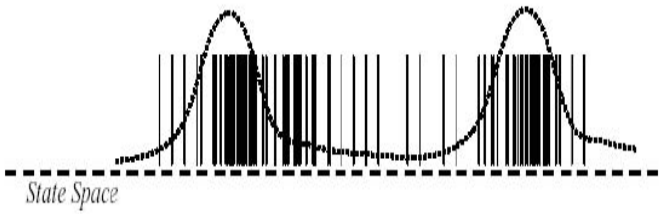
Particle Filters (PF) are a family of methods often called :

- condensation algorithms
- sequential Monte Carlo methods
- bootstrap filters
- etc

PF is based on representing the posterior pdf by a set of randomly chosen weighted samples.
"randomly chosen" ≡ "Monte carlo"

Basic Principle

- Represent the pdf as a set of weighted samples
- For $N \rightarrow \infty$, PF converges (almost surely) to the true pdf
- For $N \rightarrow \infty$, PF approaches optimal Bayesian estimate
- Regions of high density
  - Many particles
  - Large weight of particles



$\{\mathbf{x}_{0:k}^i\}$: set of samples.
$\{w_k^i\}$ associated weights, normalized to $\sum_i w_k^i = 1$

$$p(\mathbf{x}|\mathbf{Z}_{1:k}) = \sum_{i=1}^{N} w_k^i \delta(\mathbf{x}_{0:k} - \mathbf{x}_{0:k}^i)$$

SIS : Sequential Importance Sampling

- The discrete approximation is good if the samples are drawn from the posterior pdf $p(\mathbf{x}_k|\mathbf{Z}_{1:k})$.
  - But we don't have an explicit representation of $p(.)$!

- KEY IDEA : we can sample from another *importance density* $q(.)$.
  - Our approximation is still correct, up to normalization, if the particles are weighted :

$$w_k^i \propto \frac{p(\mathbf{x}_{0:k}^i|\mathbf{Z}_{1:k})}{q(\mathbf{x}_{0:k}^i|\mathbf{Z}_{1:k})}$$

  - We can choose $q(.)$ freely !

SIS : Sequential Importance Sampling

Sample from prior belief q(x) (for instance, the uniform distribution)

Compute importance weights, w(x) = p(x) /q(x)

Resample particles according to importance weights to get p(x)
Samples with high weights chosen many times; density reflects pdf

SIS : Sequential Importance Sampling

Pseudo-code

ALGORITHM 1 : SIS Particle Filter

$[\{\mathbf{x}_k^i, w_k^i\}_{i=1}^N] = \text{SIS}\ [\{\mathbf{x}_{k-1}^i, w_{k-1}^i\}_{i=1}^N, \mathbf{z}_k]$

- For $i = 1 : N$
  - Draw $\mathbf{x}_k^i \sim q(\mathbf{x}_k|\mathbf{x}_{k-1}^i, \mathbf{z}_k)$
  - Update weights according to

$$w_k^i \propto w_{k-1}^i \frac{p(\mathbf{z}_k|\mathbf{x}_k^i)p(\mathbf{x}_k^i|\mathbf{x}_{k-1}^i)}{q(\mathbf{x}_k^i|\mathbf{x}_{k-1}^i, \mathbf{z}_k)}$$

- End For
- Normalize weights to $\sum_{i=1}^N w_k^i = 1$

SIS : Sequential Importance Sampling

Degeneracy problem : after a few iterations, most particles have negligible weights ; the weights are concentrated on a few particles only !

Solutions :

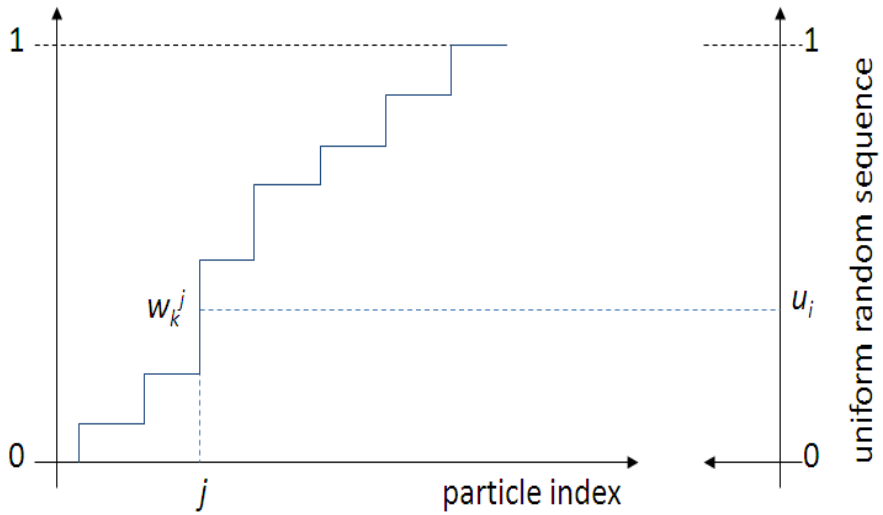
1. *Brute force* : many, many samples ! Of course, impractical.
2. *Optimal importance density* :
   $q(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{z}_k)_{opt} = p(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{z}_k)$
   and $w_k^i = w_{k-1}^i \int p(\mathbf{z}_k|\mathbf{x}'_k) p(\mathbf{x}'_k|\mathbf{x}_{k-1}^i) d\mathbf{x}'_k$
   But evaluating an integral over the entire state can rarely be done.

   An alternative convenient choice is to take $q(\mathbf{x}_k|\mathbf{x}_{k-1}^i, \mathbf{z}_k) = p(\mathbf{x}_k|\mathbf{x}_{k-1}^i)$
   $\hookrightarrow$ easy to implement, but does not take measurements into account.
3. *Resampling*

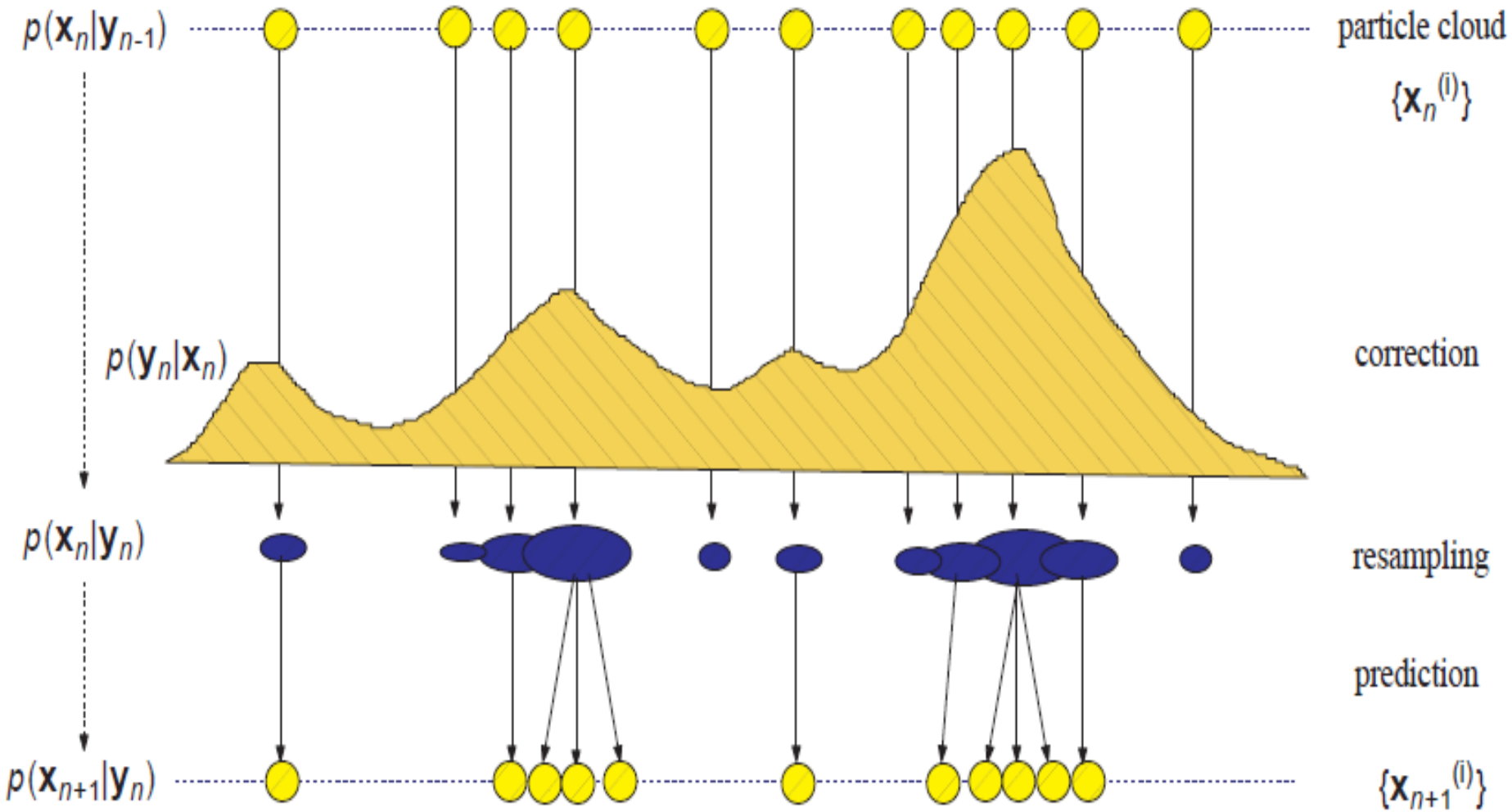
SIS : Sequential Importance Sampling

Resampling : eliminate particles with small associated weights and to concentrate on particles with high weights.

- Sample N times *with replacement* from the set of particles $\{\mathbf{x}_{0:k}^{i}\}$ according to importance weights :
  $\Rightarrow$ "*Survival of the fittest*"



Sampling and Resampling



Generic Particle Filter Pseudo-Code

ALGORITHM 2 : Generic Particle Filter

$[\{\mathbf{x}_k^i, w_k^i\}_{i=1}^N] = \text{PF}\ [\{\mathbf{x}_{k-1}^i, w_{k-1}^i\}_{i=1}^N, \mathbf{z}_k]$

- FOR $i = 1 : N$
  - Draw $\mathbf{x}_k^i \sim q(\mathbf{x}_k|\mathbf{x}_{k-1}^i, \mathbf{z}_k)$
  - Update weights according to $w_k^i \propto w_{k-1}^i \frac{p(\mathbf{z}_k|\mathbf{x}_k^i)p(\mathbf{x}_k^i|\mathbf{x}_{k-1}^i)}{q(\mathbf{x}_k^i|\mathbf{x}_{k-1}^i,\mathbf{z}_k)}$
- END FOR
- Normalize weights to $\sum_{i=1}^N w_k^i = 1$
- Calculate degeneracy measure : $\widehat{N_{eff}} = \frac{1}{\sum_{i=1}^N (w_k^i)^2}$
- IF $\widehat{N_{eff}} < N_T$
  - Resample
- END IF

Estimation form particles

Any estimate of a function $f(\mathbf{x}_k)$ can be obtained from the discrete pdf approximation :

$$\mathbb{E}[f(\mathbf{x}_k)] = \frac{1}{N}\sum_{i=1}^{N} w_k^i f(\mathbf{x}_k^i)$$

- Mean : $\mathbb{E}[\mathbf{x}_k] = \frac{1}{N}\sum_{i=1}^{N} w_k^i \mathbf{x}_k^i$
- MAP estimate : particle with largest weight
- Robust mean : mean within window around MAP estimate

A simple example

We want to estimate a parameter **x** whose evolution is governed by :

$$x_{t+1} = 1 + \sin(4.10^{-2}\pi t) + 0.5x_t + w_t,$$

with $w_t \rightsquigarrow Gamma(3, 2)$.

We cannot measure x directly but have access to a set of measurements :

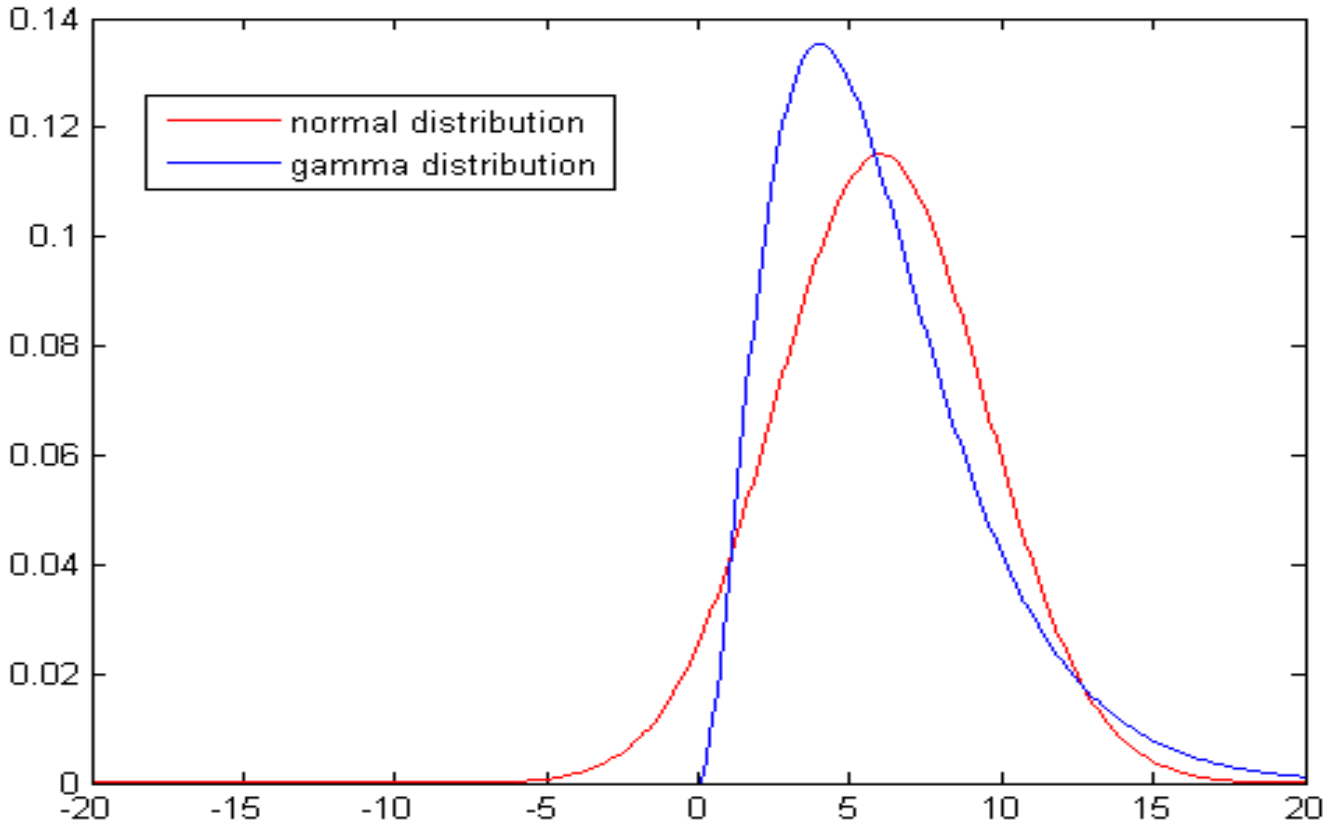
$$z_t = \begin{cases} 0.2x_t^2 + v_t & \text{if} \quad t \leq 30 \\ 0.5x_t - 2 + v_t & \text{if} \quad t > 30 \end{cases},$$

with $v_t \rightsquigarrow \mathcal{N}(0, \sigma_m)$.

Particle Filters : Example

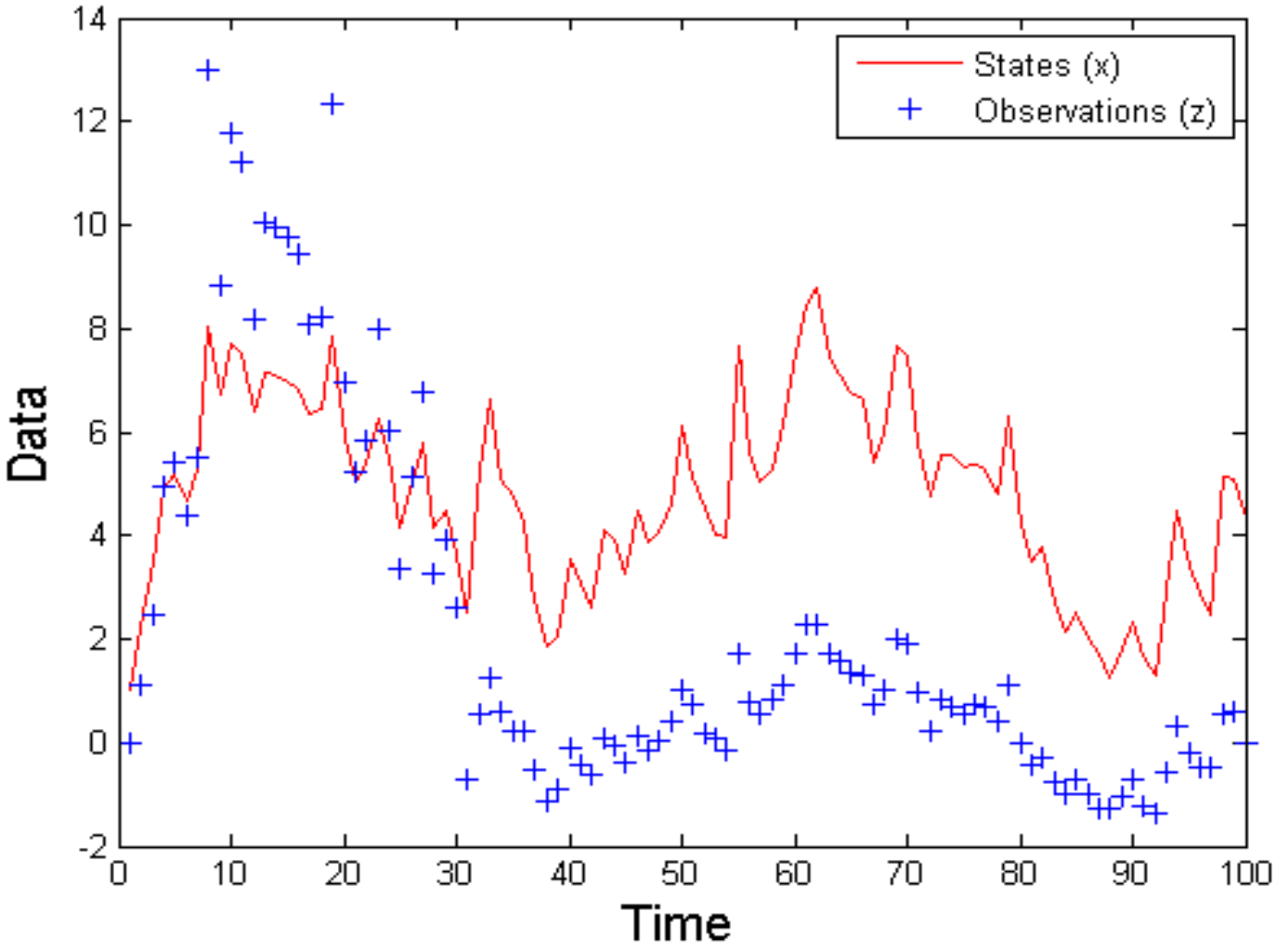
A simple example

Gaussian and Gamma distributions : same mean and variance.



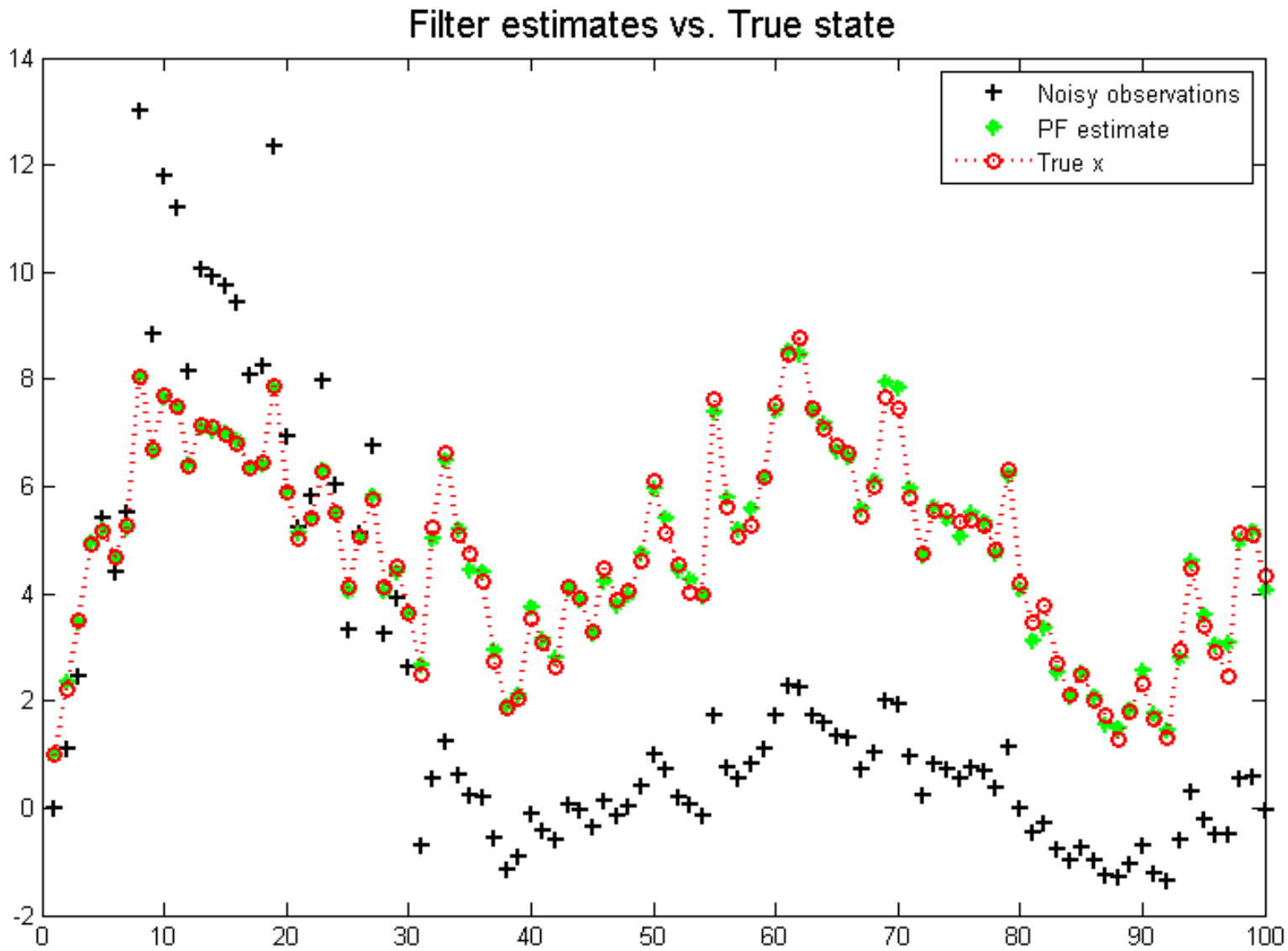
Particle Filters : Example

A simple example



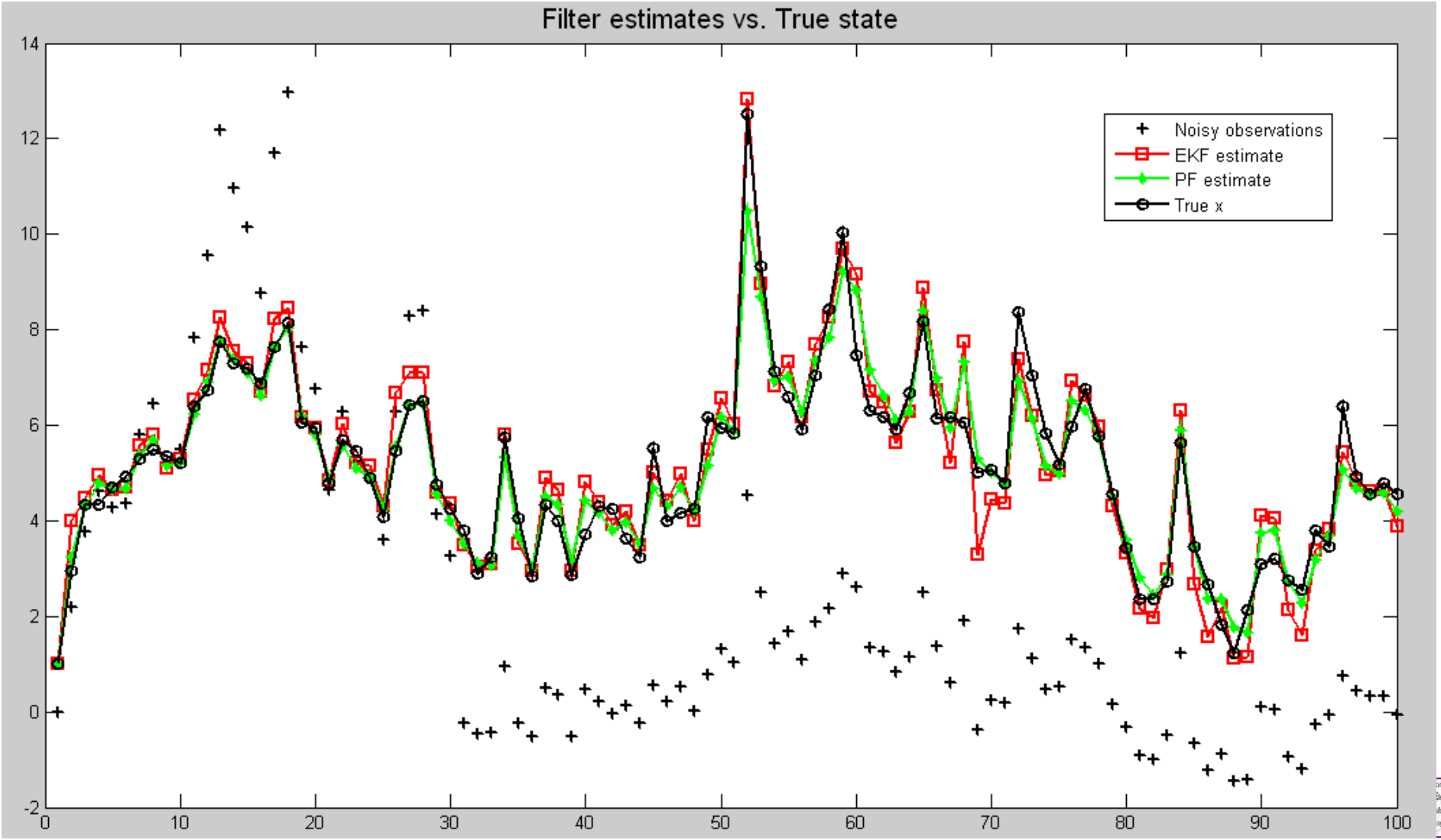
Particle Filters : Example

A simple example



Particle Filters : Example

Comparison with the Extended KF



Outline

What is tracking ?

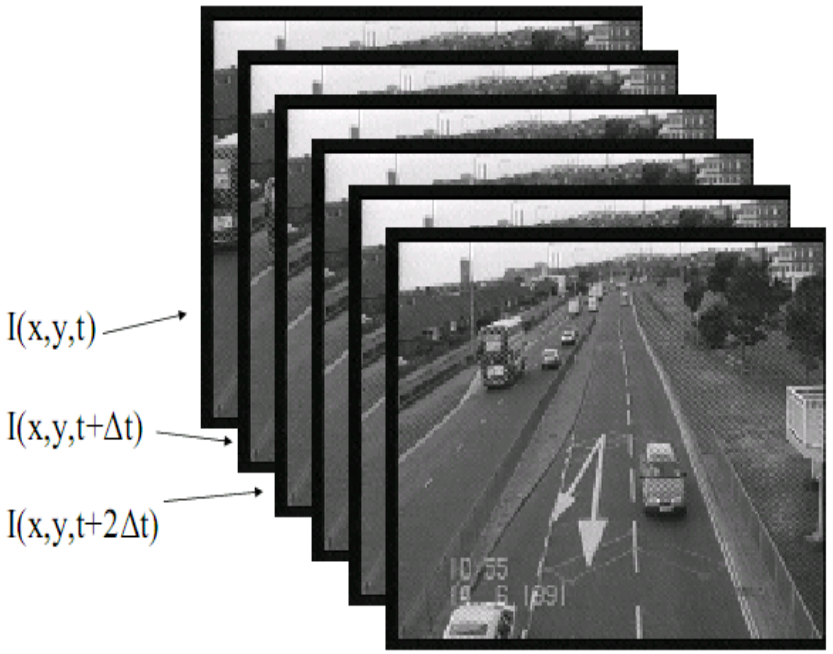
Problem statement

Given a sequence of images

- Find the trajectories of objects over time

Fundamental problem

- Surveillance
- Vehicle tracking
- Human-computer interaction
- Motion capture
- Robotics
- etc.



What is tracking ?

Why track ?

- Segmentation is hard !
- Detection and recognition are expensive
- If we can **predict** where the object will be in next frame, then we need less work detecting or recognizing the object.

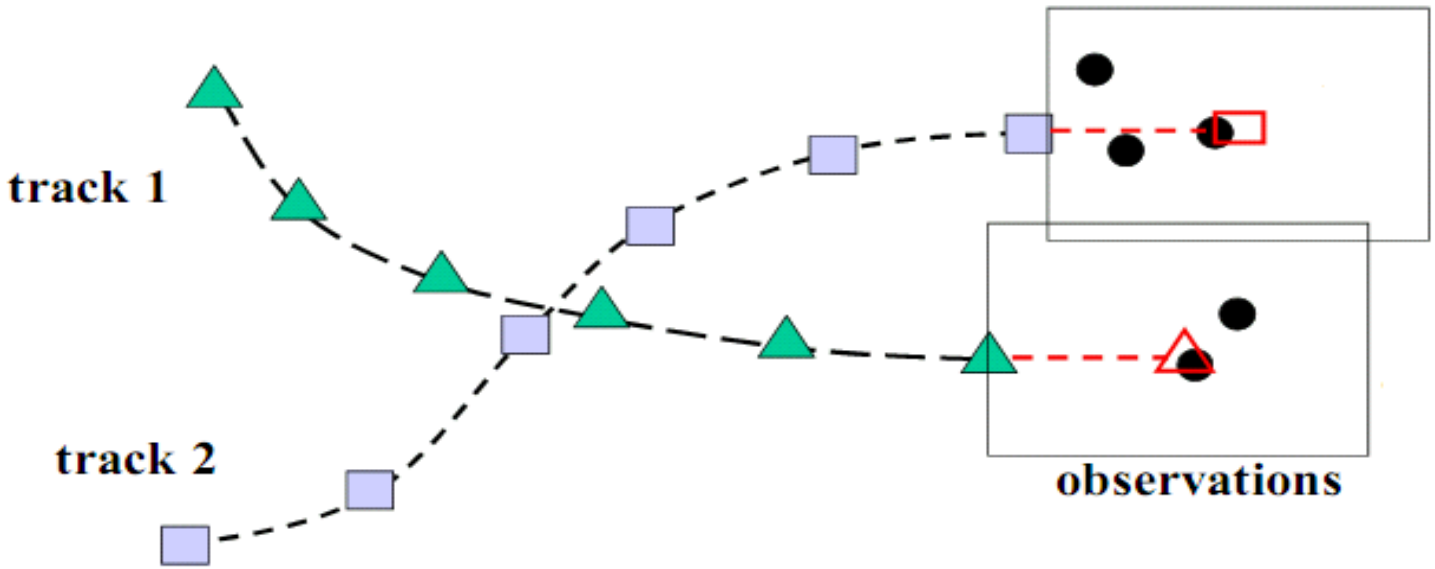
Three main steps

- Prediction
- Matching (data association)
- Correction (updating)

What is tracking ?

Tracking involves two basic problems :

- **Motion** : predicts the limited search region in which the tracked object is most likely to be in next frame.
- **Matching** : identifies the object within the designed search region.

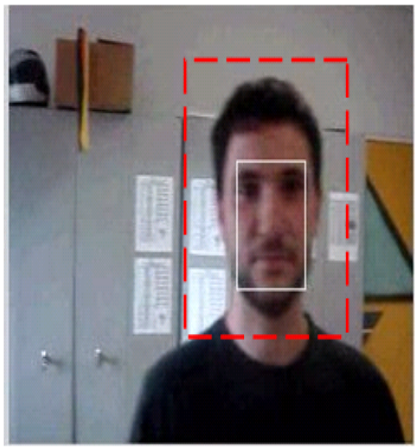


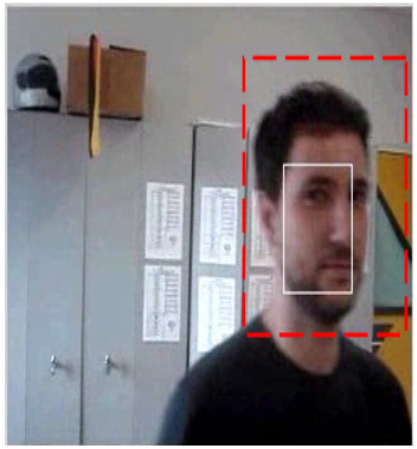
What is tracking ?

Example of Face Tracking

- Detection involves searching in entire image
  => Can take some time !
- Though efficient solutions exist like Adaboost classifiers, e.g. Viola and Jones detector for face detection.

- Making prediction, reduces the search area
  => template matching can do the job efficiently !





Implicit Assumptions

- Physical cameras do not move instantly from one viewpoint to another.
- Objects do not teleport between places around the scene.
- Relative position between camera and scene changes incrementally.
- We can model objects motion.

Designing a Visual Tracker

What is the state ?

- pose and motion (position, velocity, acceleration, ...)
- shape (size, deformation, articulation, ...)
- appearance (subspace, colors, ...)

Which image properties should we use ?

- intensity, color, texture, motion, edges, ...
- template, adaptive appearance, region statistics

What might simplify the problem ?

- known/stationary background
- use of color (e.g., skin)
- multiple cameras (often 2 or 3)
- manual initialization
- prior knowledge of the number of objects and their types
- limited (or no) occlusion

Outline

Tracking Methods

Deterministic methods

Iteratively search for a local maxima of a similarity measure between a template of the target and the current frame.

Examples :

- KLT tracker
- Mean-Shift tracker

Probabilistic methods

Use a state-space representation of the moving object to model its underlying dynamics.
=> The tracking problem is viewed as a Bayesian inference problem.

Examples :

- Kalman filter
- Particle filter

Tracking Methods : Background subtraction

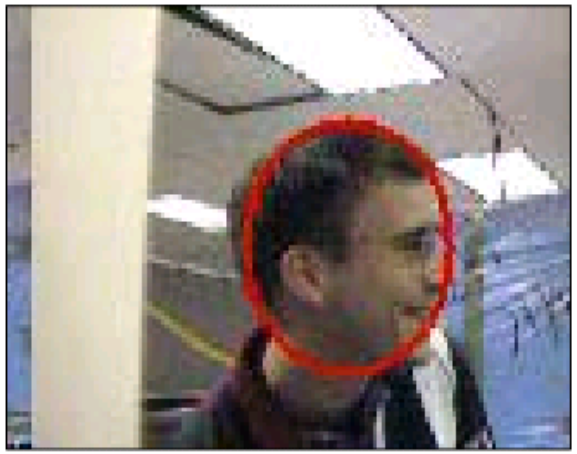
Pros

- Simple and easy implementation
- Detect all changes

Cons

- Detect unwanted changes
- Difficult to quantify motion
- Difficult to deal with changing background

Tracking Methods : Regions tracking

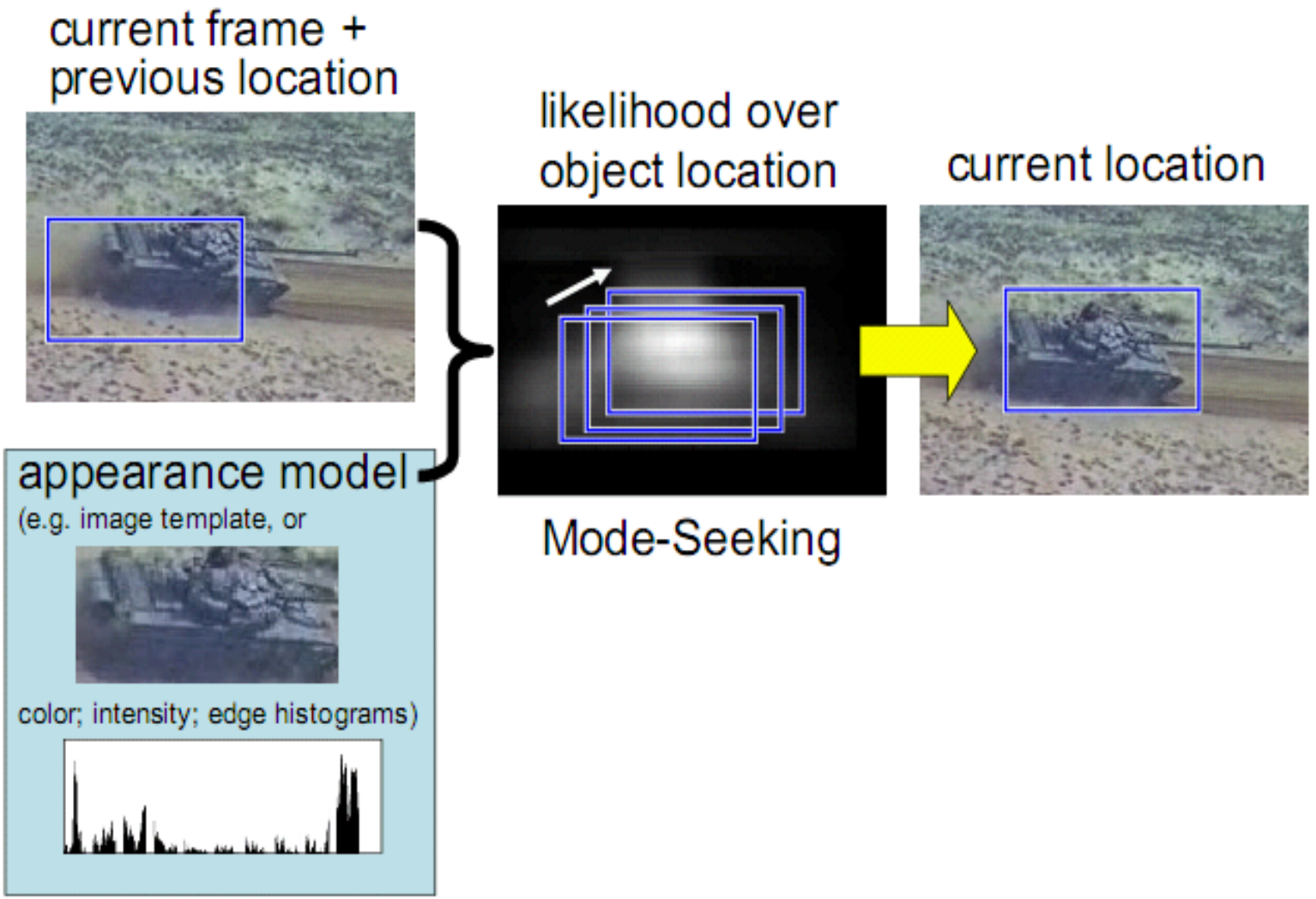


State : 2D image location, size of ellipse, velocity
Appearance model : color histogram, elliptical contour
Estimation : search over discrete locations and size, or density estimation

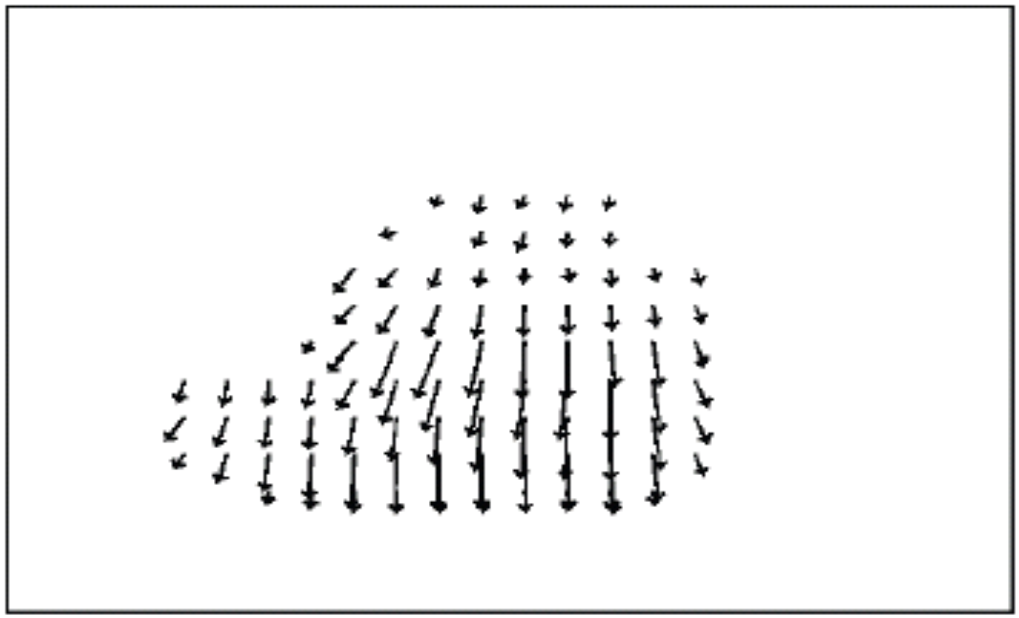
Pros/Cons

- fast
- good for large variation in shape/appearance
- limited fidelity in pose estimation

Tracking Methods : Appearance-Based Tracking



Tracking Methods : Motion-Based Tracking



We can better understand the scene if we can find the motions of the objects/cameras.

Pros/Cons

- based on brightness constancy
- good for small motion

Tracking Methods : Probabilistic Tracking

- Explicit modeling of the system dynamics : how does the object move from frame to frame ?
- Modeling of states uncertainty : how do we acount for system and measurement noises ?

Pros/Cons

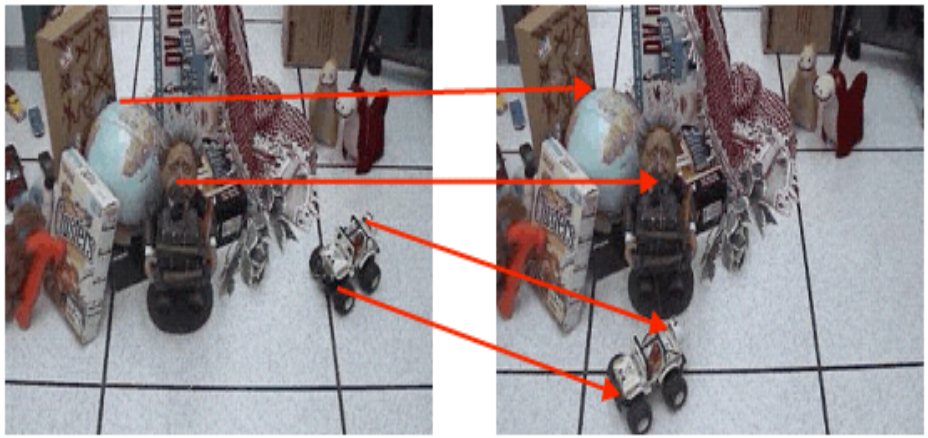
- give good results for complex motion
- robust to occlusion
- can be slow

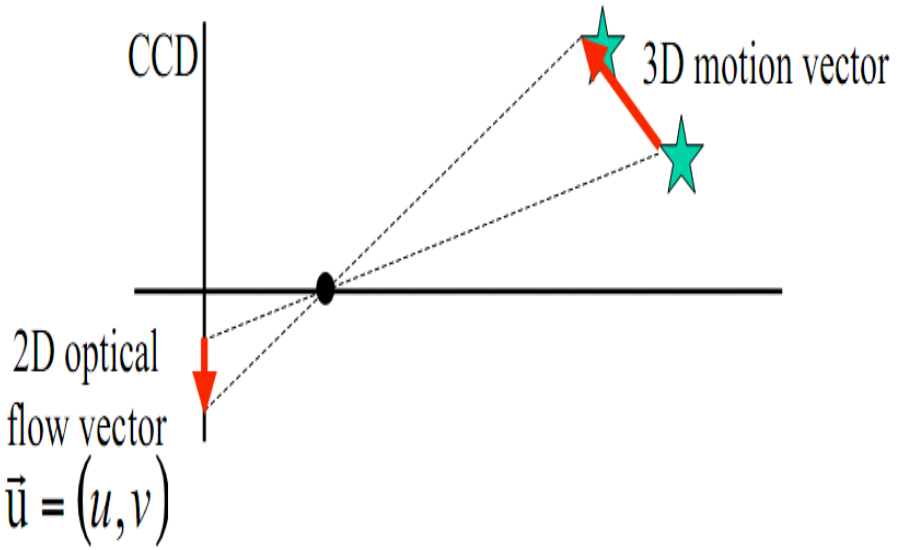
Deterministic Methods

Two popular tracking methods are :

- KLT Tracker (1984-1990) for feature points tracking
- Mean Shift Tracker (2000) for blob/region tracking

Optical Flow and Motion Field

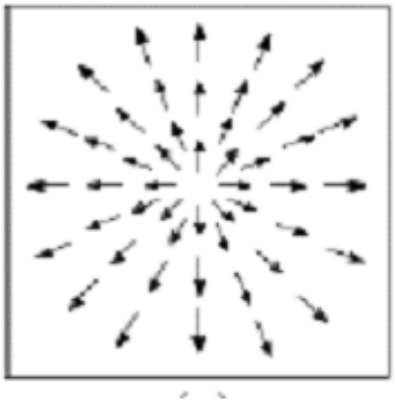




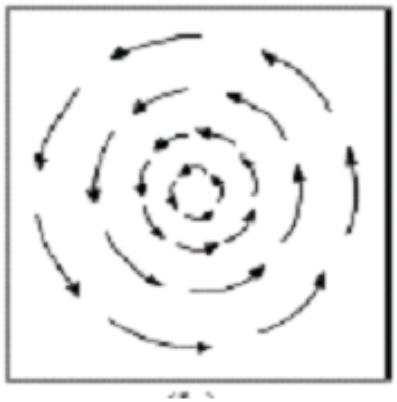
Where did each pixel in I_1 go to in I_2 ?

- Motion Field = Real word 3D motion
- Optical Flow Field = Projection of the motion field onto the 2D image

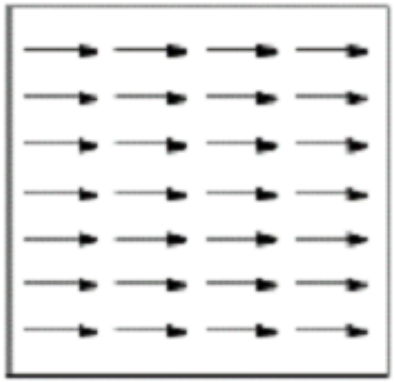
Examples of Motion fields



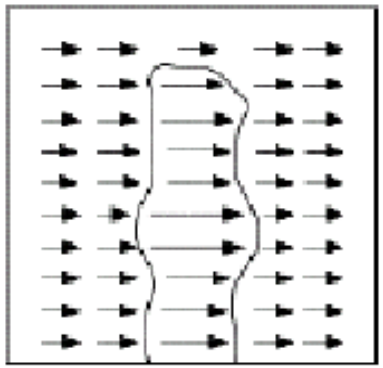
Forward motion



Rotation



Horizontal translation



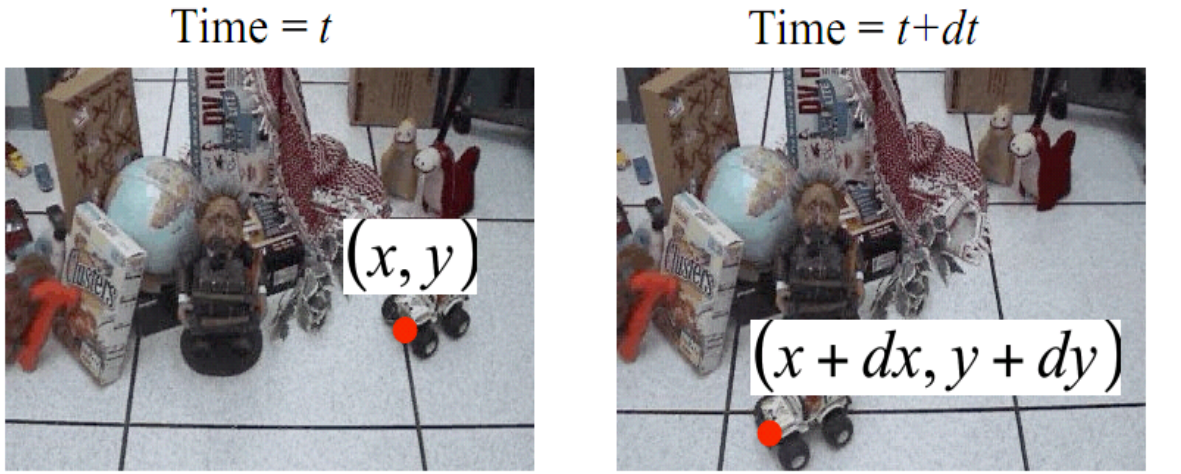
Closer objects appear to move faster!!

Estimating Optical Flow

Goal

Find for each pixel a velocity vector $\vec{\mathbf{u}} = (u, v)$, which says :

- how quickly is the pixel moving across the image
- in which direction it is moving



$$I(x, y, t) = I(x + dx, y + dy, t + dt)$$

Zelnik-Manor et al., PAMI 2000

=> Important assumption : image **intensity I is constant**

Brightness Constancy Equation

$$\begin{aligned} I(x,y,t) &= I(x+dx, y+dy, t+dt) \\ &= I(x,y,t) + \frac{\partial I}{\partial x}dx + \frac{\partial I}{\partial y}dy + \frac{\partial I}{\partial t}dt \end{aligned}$$

Which gives

$$\frac{\partial I}{\partial x}dx + \frac{\partial I}{\partial y}dy + \frac{\partial I}{\partial t}dt = 0$$

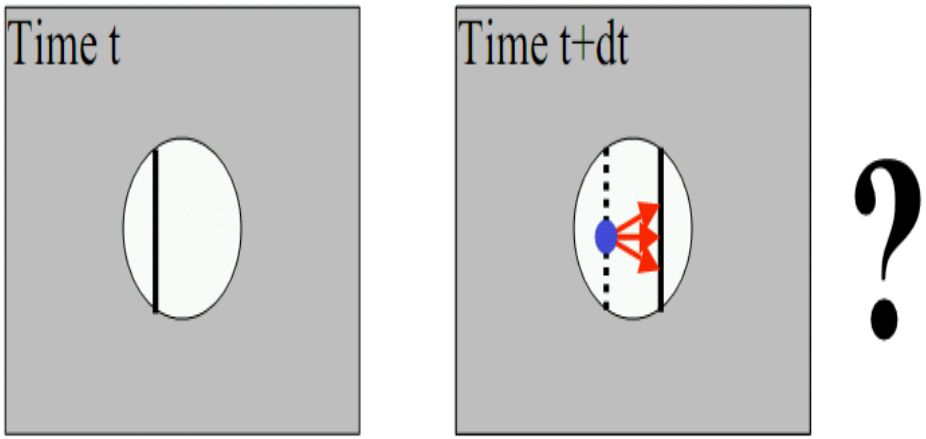
Finally, if we divide by dt and denote $u = \frac{dx}{dy}$ and $v = \frac{dy}{dt}$, we have :

$$\boxed{I_x u + I_y v = -I_t}$$

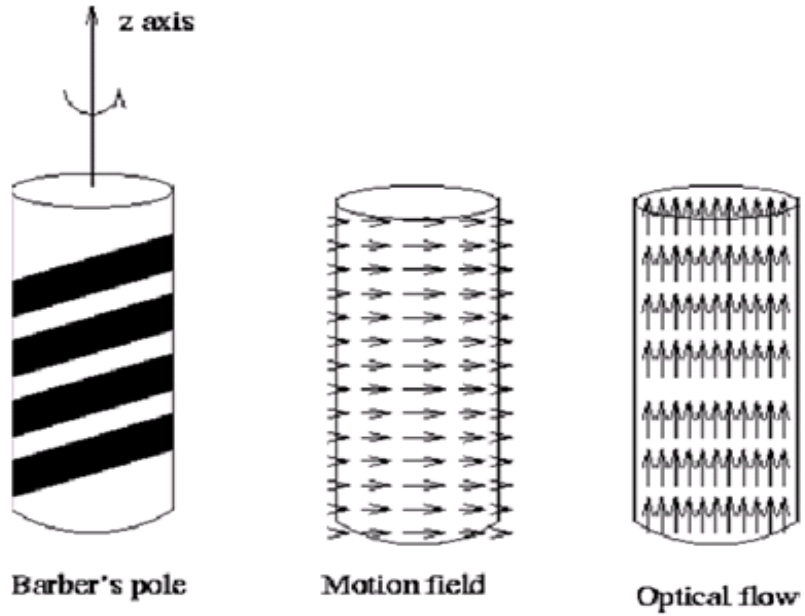
Problem : one equation, two unknows ! (under-determined system)

Estimating Optical Flow

The Aperture Problem



Where did the blue point move to ?



The aperture problem in practice

Brightness constancy equation provides only the normal component of the motion flow.
=> We need additional constraints

Local smoothness

$$I_x u + I_y v = -I_t \implies [Ix \quad Iy] \begin{bmatrix} u \\ v \end{bmatrix} = -I_t$$

If we assume constant velocity (u, v) in small neighborhood

$$\begin{bmatrix} I_{x1} & I_{y1} \\ I_{x2} & I_{y2} \\ I_{x3} & I_{y3} \\ \vdots & \vdots \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} I_{t1} \\ I_{t2} \\ I_{t3} \\ \vdots \end{bmatrix}$$

We get an over-determined system

$$\boxed{\mathbf{Au} = \mathbf{b}}$$

which we can solve using LLS (Linear Least Squares) method.

Lucas Kanade

We want to find **u** that minimizes $||\mathbf{Au} - \mathbf{b}||^2$
LLS solution is given by

$$\hat{\mathbf{u}} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b}$$

So, we need the matrix $\mathbf{A}^T\mathbf{A}$ to be invertible.

$$\mathbf{A}^T\mathbf{A} = \begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix}$$

=> summation over all pixels in a $K \times K$ window (e.g., 5×5).

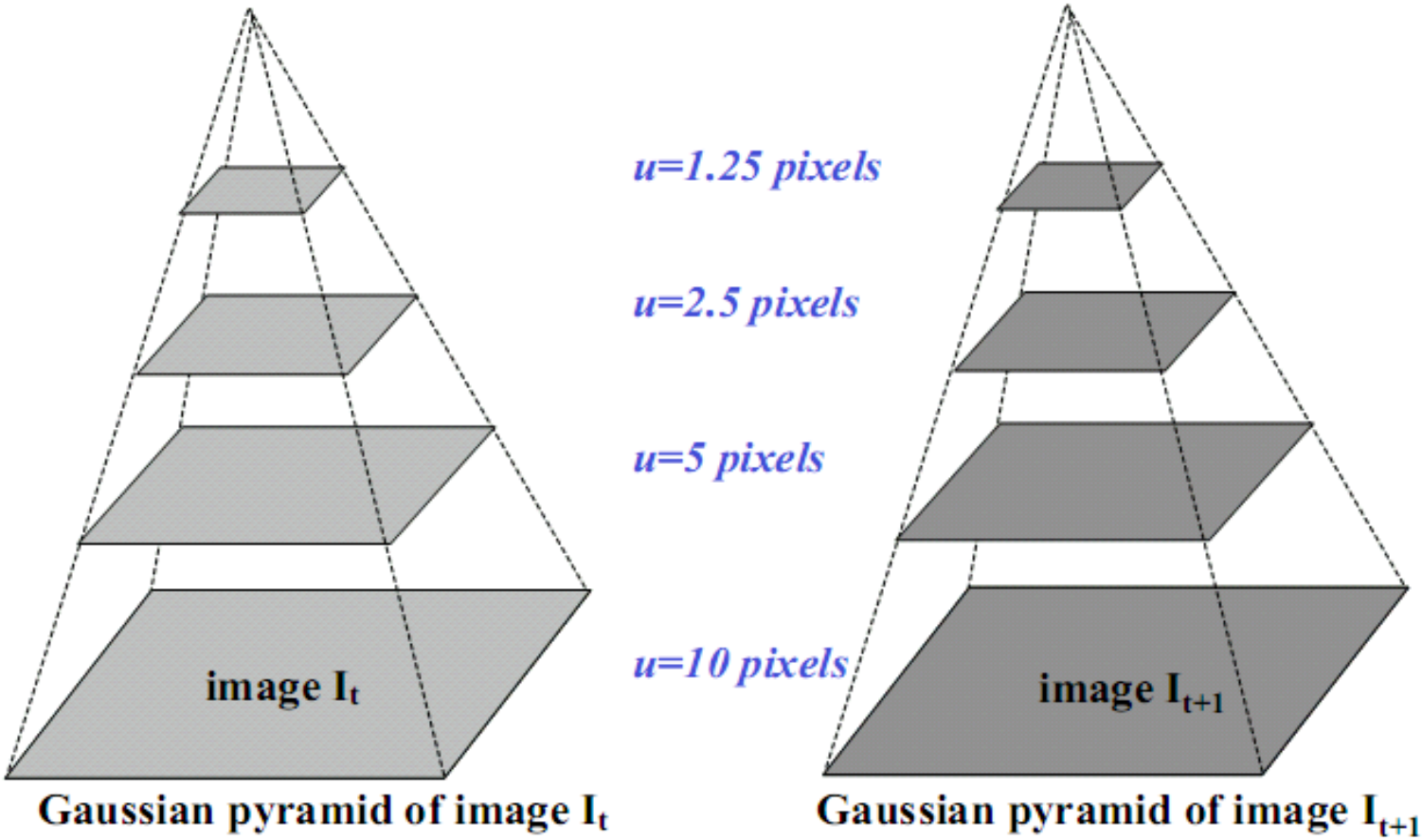
Good Features to Track

- Edge $\rightarrow$ $\mathbf{A}^T\mathbf{A}$ becomes singular
- Homogeneous region $\rightarrow$ low gradients ; $\mathbf{A}^T\mathbf{A} \approx 0$
- **High texture** $\rightarrow$ Good ! $\mathbf{A}^T\mathbf{A}$ has two large eigenvalues.

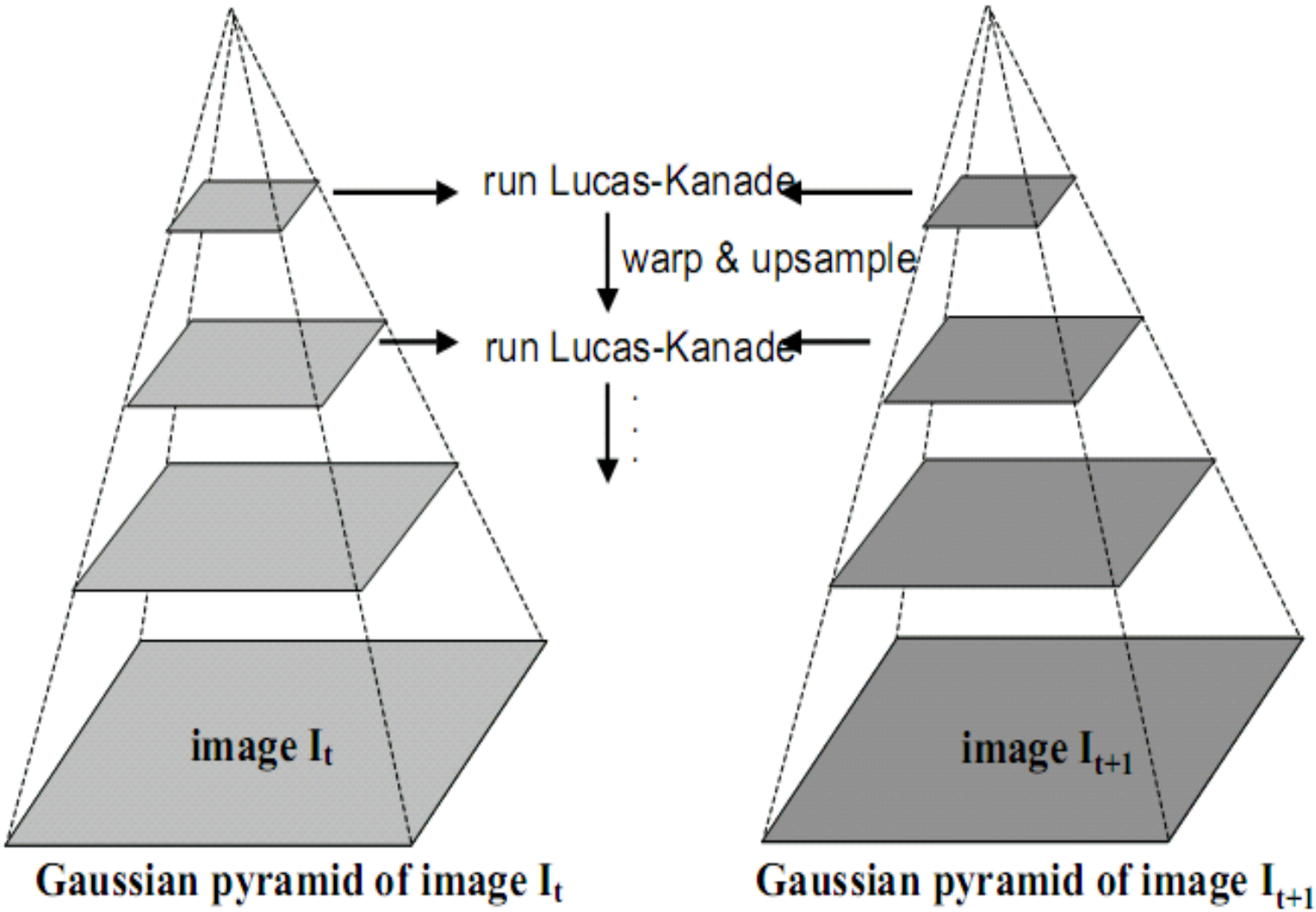
KLT Tracker

KLT assumes that the motion is small, if not the case Taylor expansion doesn't hold !

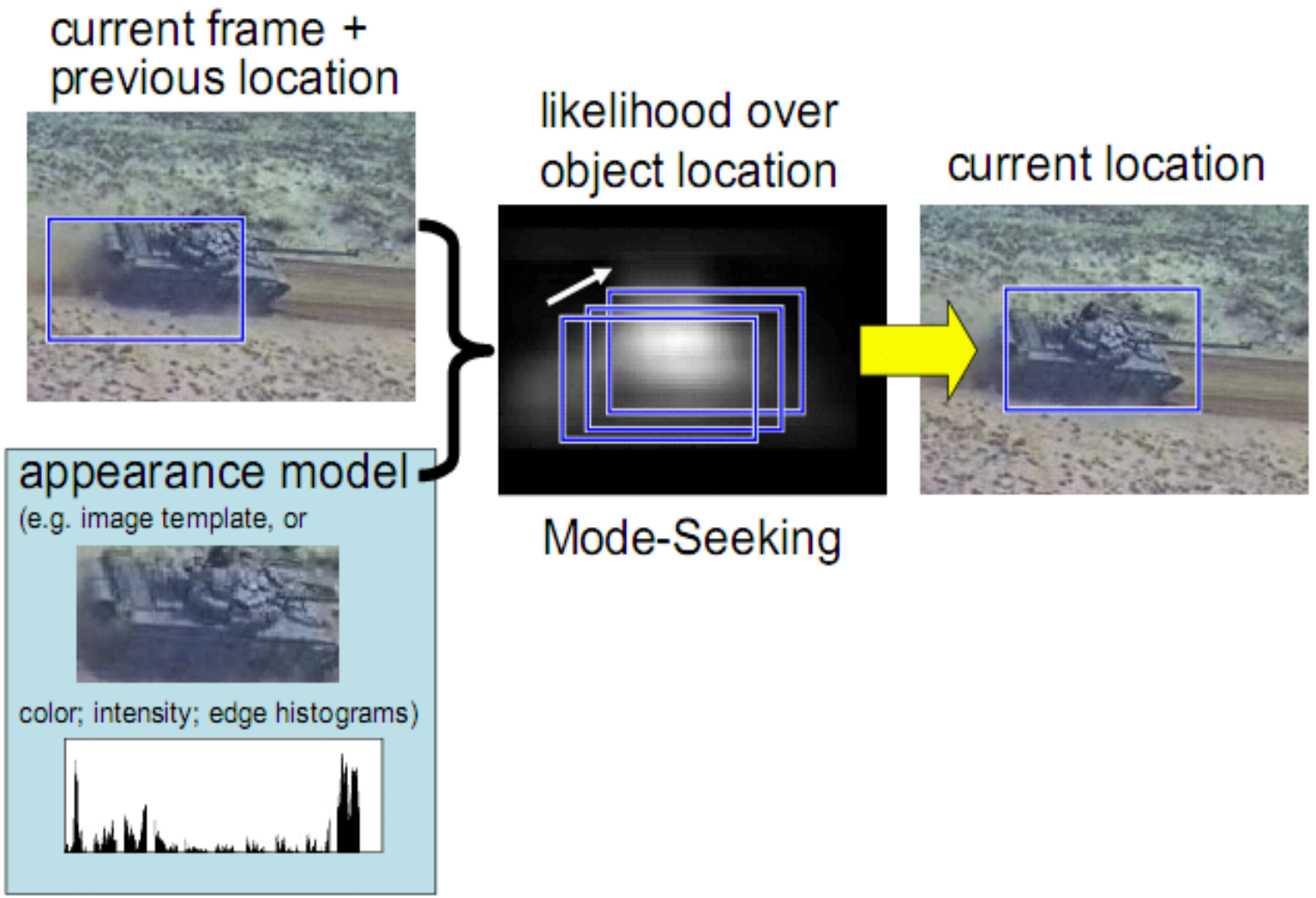
=> multi-scale estimation



Multi-scale estimation



Appearance-Based Tracking



Mean-Shift Tracker

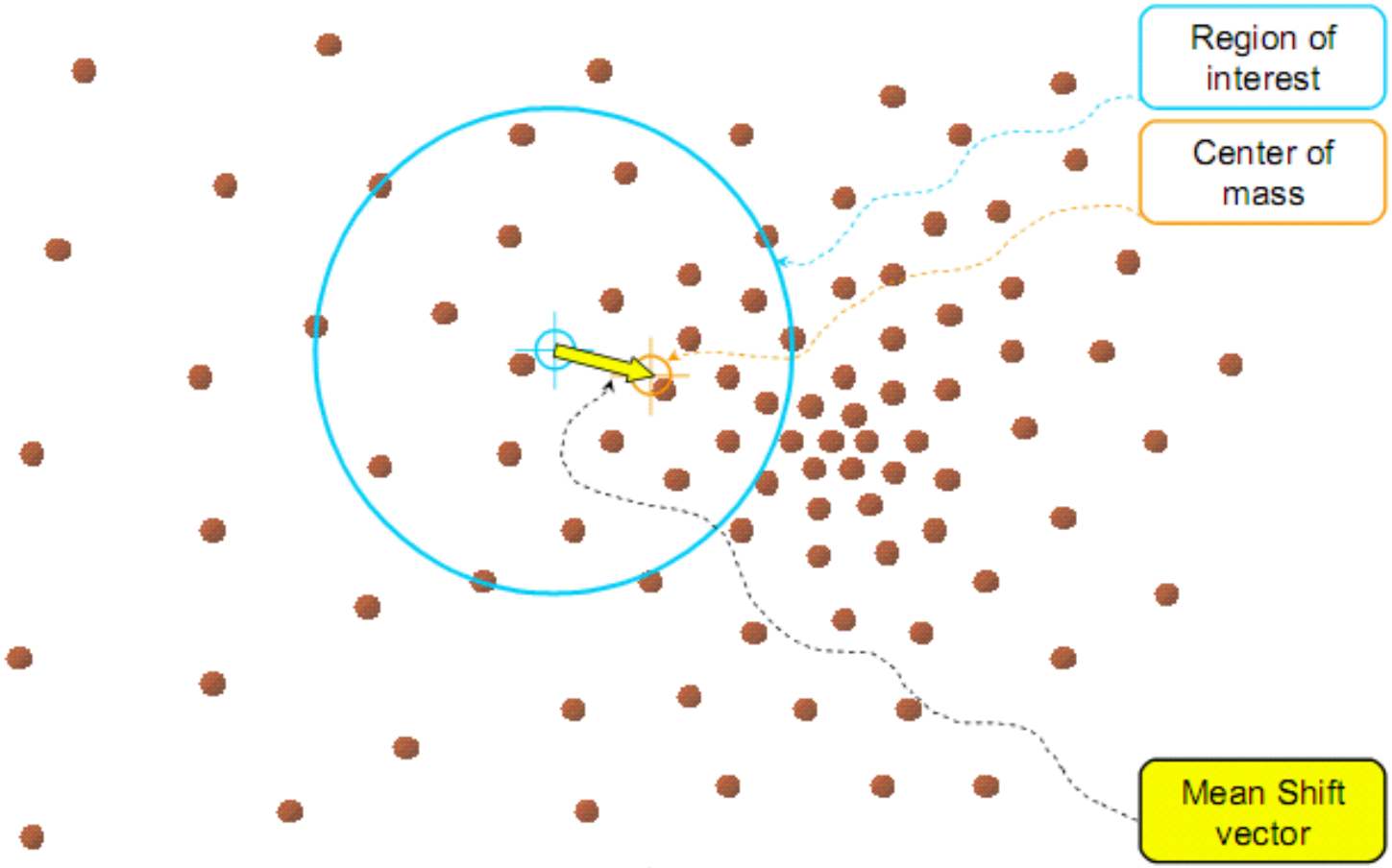
Motivations

- Track non-rigid objects, for which it is hard to specify an explicit 2D parametric motion model.
- Appearances of non-rigid objects can be modeled by color distribution

Mean-Shift

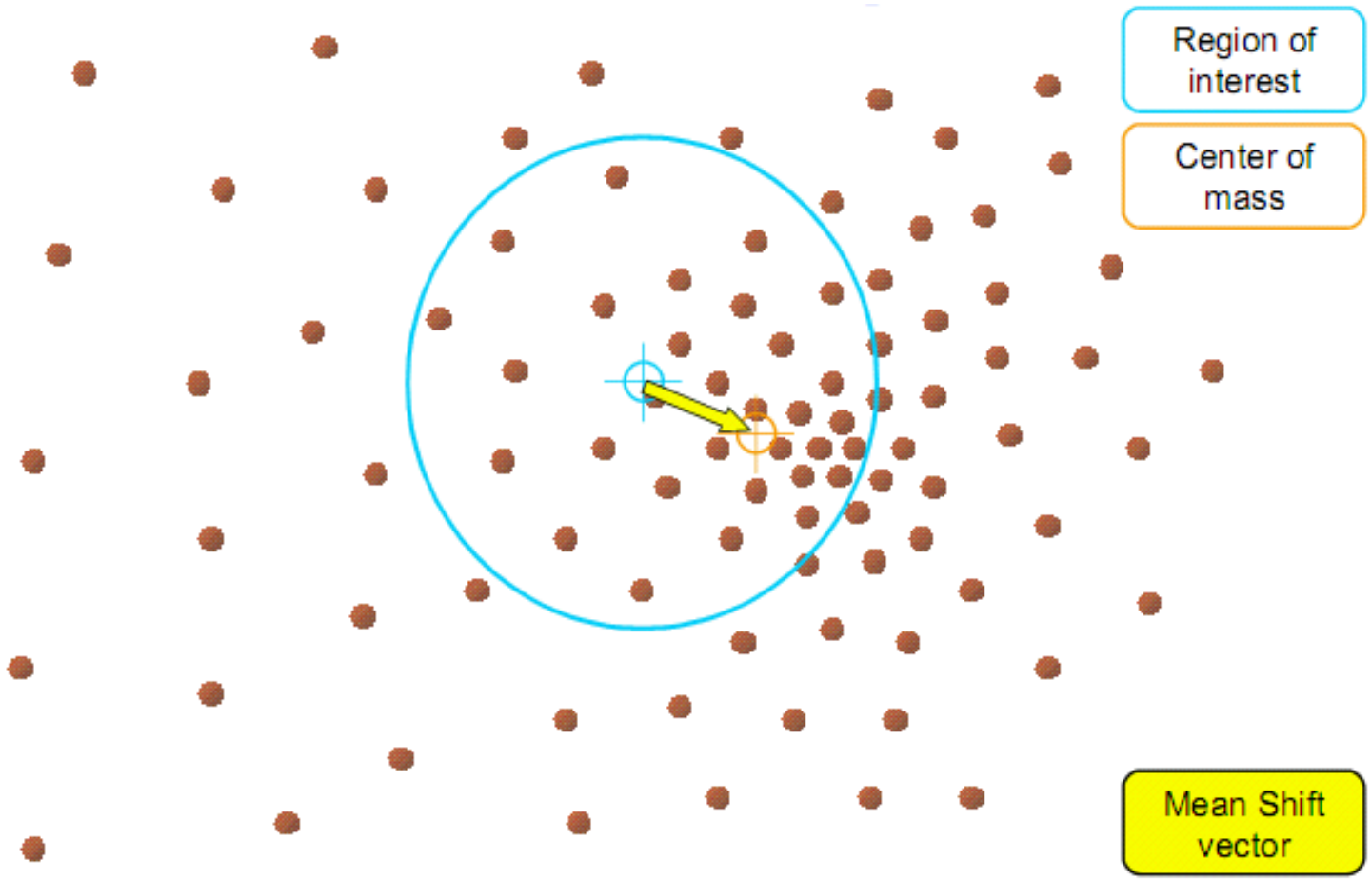
- Mean-shift algorithm is an efficient approach to tracking objects whose appearance is modeled by histograms (not limited to color).
- It finds modes of the appearance's distribution.

Intuitive Description



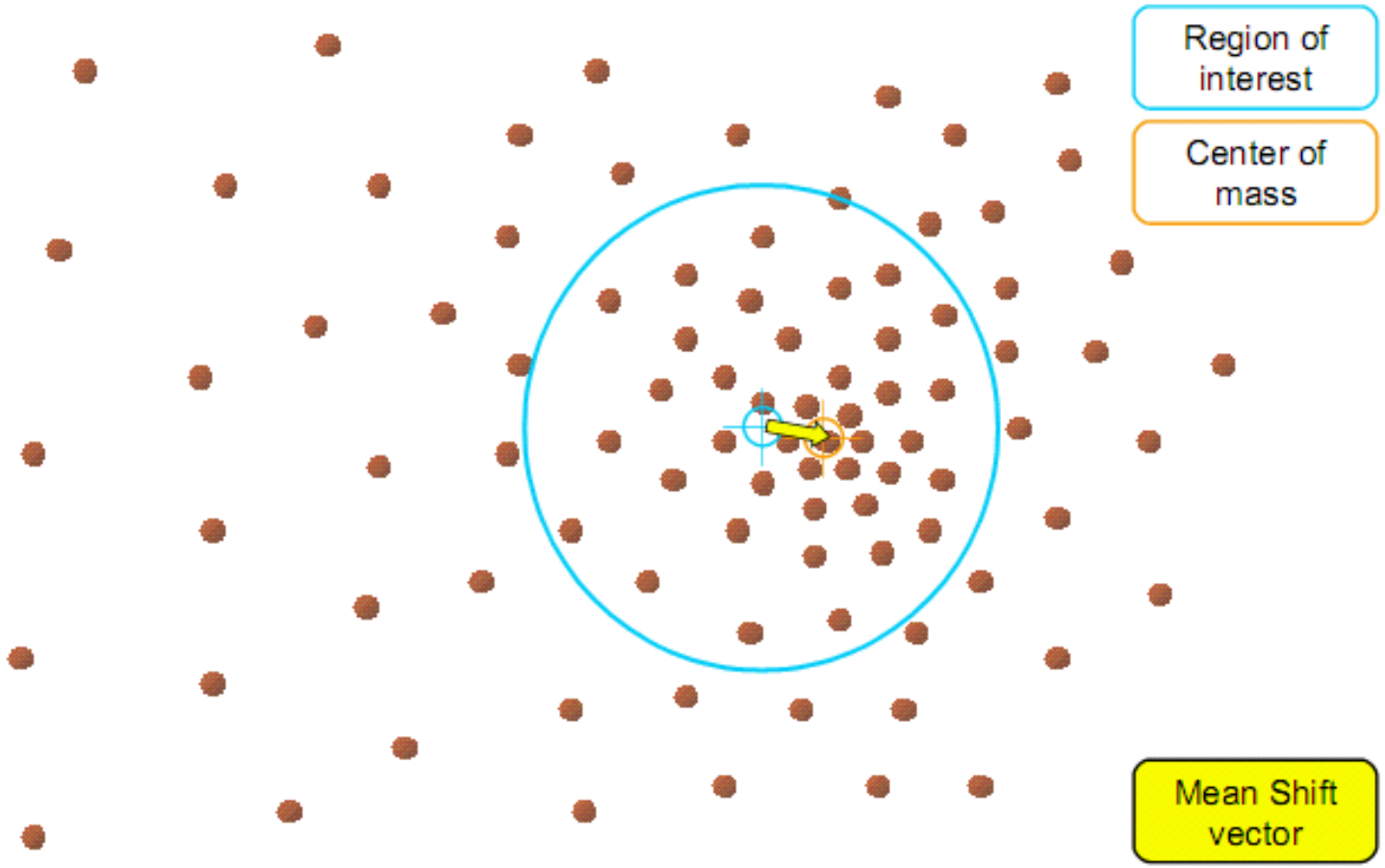
Objective : Find the densest region

Intuitive Description



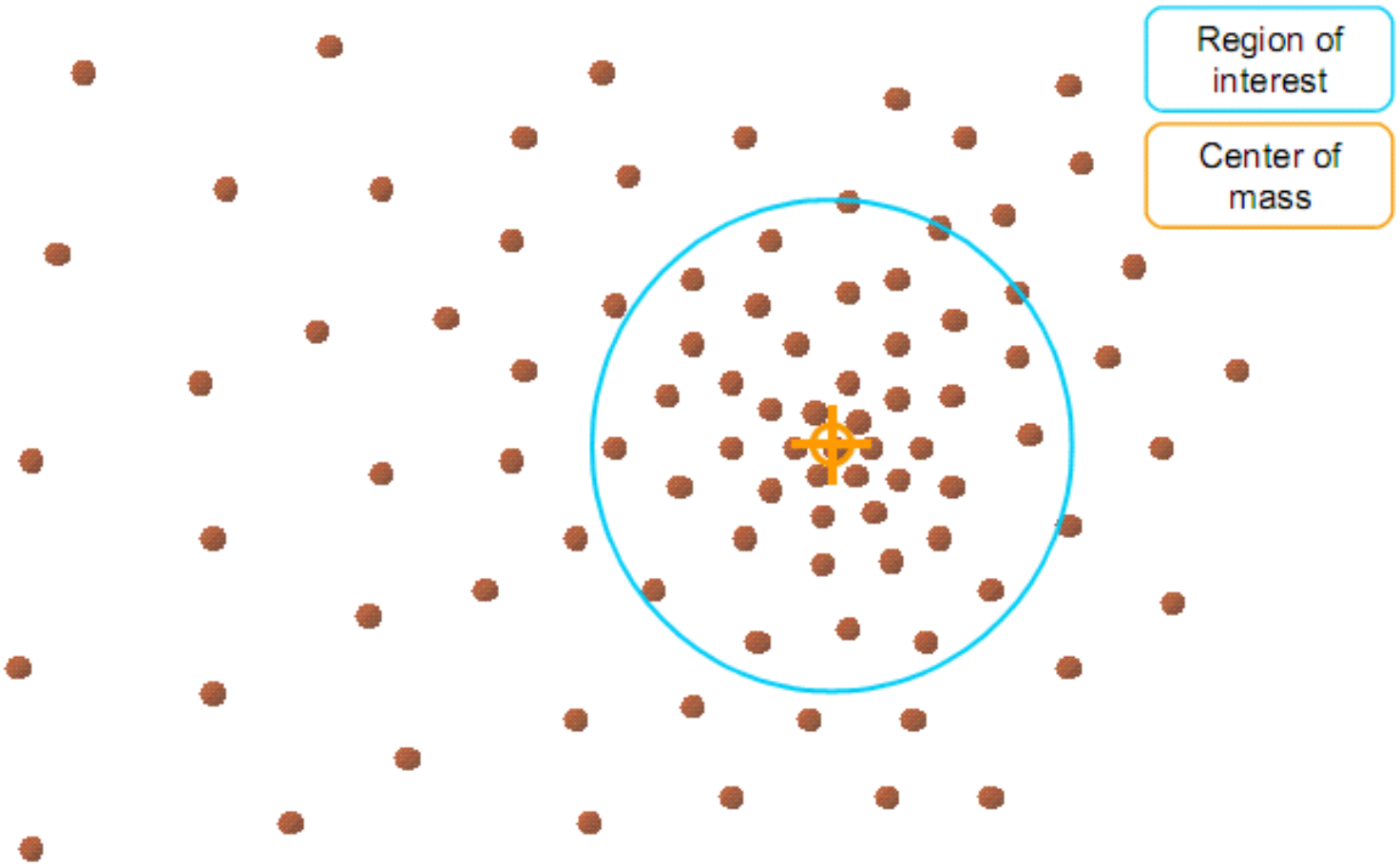
Objective : Find the densest region

Intuitive Description



Objective : Find the densest region

Intuitive Description



Objective : Find the densest region

Mean-Shift : Theory

Instead of estimating the pdf

$$P(X) = \frac{1}{n}\sum_{i=1}^{n} K(X - X_i)$$

Only estimate the gradient

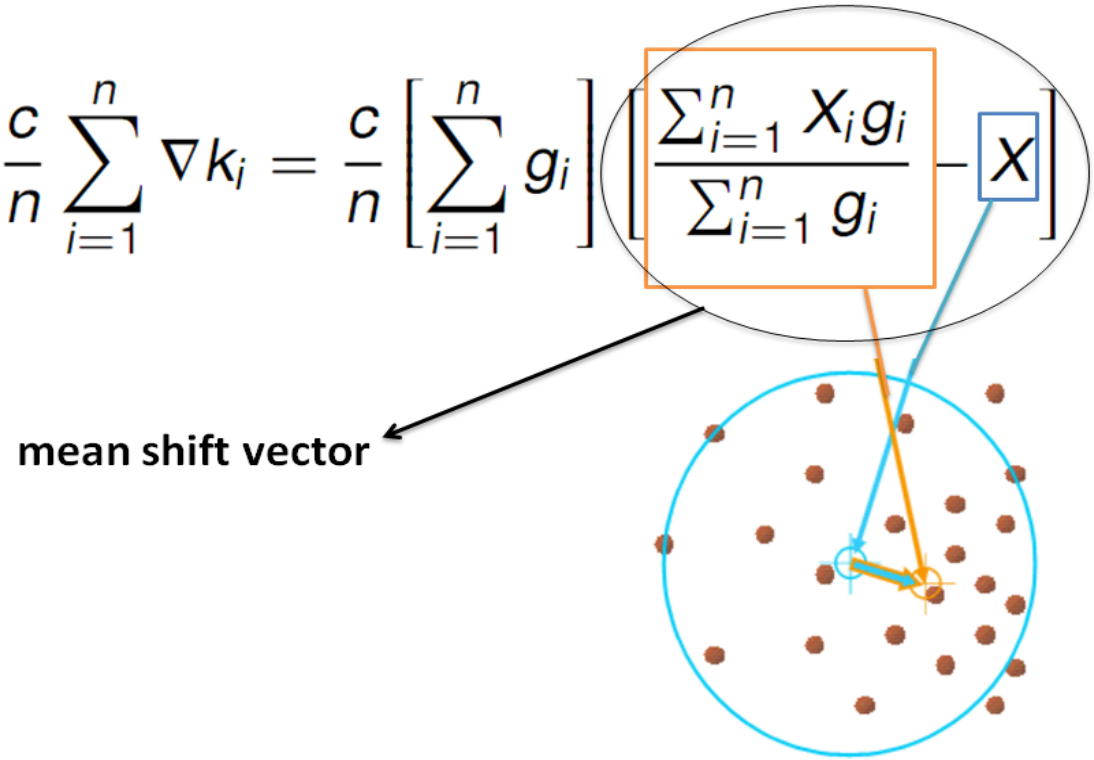
$$\nabla P(X) = \frac{1}{n}\sum_{i=1}^{n} \nabla K(X - X_i)$$

Using the kernel form : $K(X - X_i) = ck\left(\left\|\frac{X-X_i}{h}\right\|^2\right)$, we get

$$\nabla P(X) = \frac{c}{n}\sum_{i=1}^{n} \nabla k_i = \frac{c}{n}\left[\sum_{i=1}^{n} g_i\right]\left[\frac{\sum_{i=1}^{n} X_i g_i}{\sum_{i=1}^{n} g_i} - X\right]$$

Mean-Shift : Theory

$$\nabla P(X) = \frac{c}{n}\sum_{i=1}^{n}\nabla k_i = \frac{c}{n}\left[\sum_{i=1}^{n} g_i\right]\left[\frac{\sum_{i=1}^{n} x_i g_i}{\sum_{i=1}^{n} g_i} - X\right]$$



Simple Mean-Shift procedure :

- Compute the mean shift vector $\mathbf{m}(\mathbf{x})$
- Translate the Kernel window by $\mathbf{m}(\mathbf{x})$
- Iterate until convergence

Mean-Shift : Tracking

Some Results

Partial occlusion

Distraction

Motion blur

Comaniciu et al., PAMI 2002

- The method is quite effective
- But not scale invariant
- Collins (CVPR 2003) has proposed an scale-space adapted mean-shift

Outline

A Tracking example

The state is defined by the position and the velocity of the object

$$X_t = \begin{pmatrix} x_t \\ y_t \\ \dot{x}_t \\ \dot{y}_t \end{pmatrix}$$

If we use a first order AR model, then the state-space equations are given by :

- State equation

$$X_t = AX_{t-1} + v_{t-1}$$

- Measurement equation

$$Z_t = BX_t + w_t$$

where

$$A = \begin{pmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \text{ and } B = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

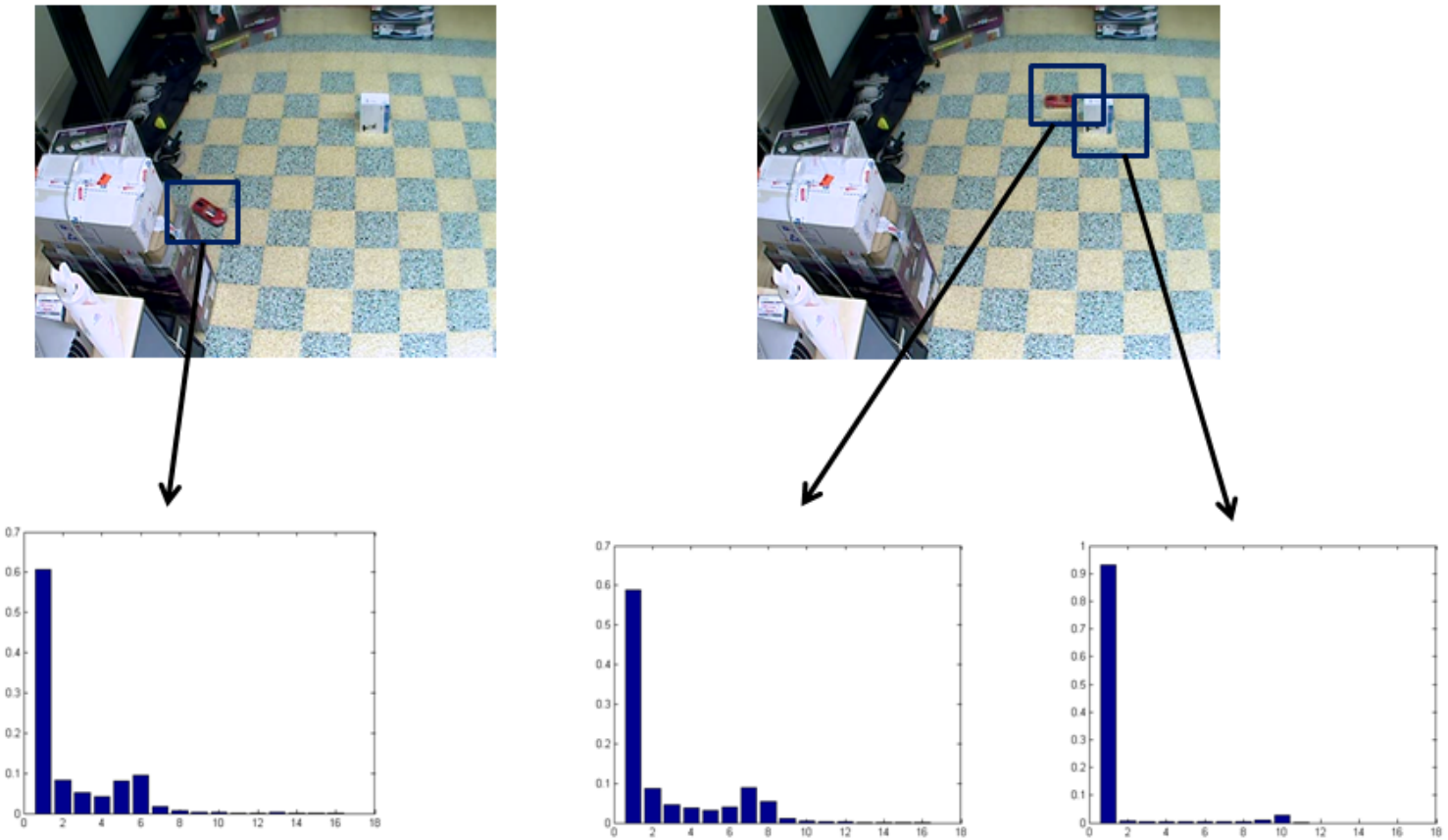
and

$$v_t \rightarrow \mathcal{N}(0, Q) \qquad Q = \begin{pmatrix} q_x^2 & 0 & 0 & 0 \\ 0 & q_y^2 & 0 & 0 \\ 0 & 0 & q_{\dot{x}}^2 & 0 \\ 0 & 0 & 0 & q_{\dot{y}}^2 \end{pmatrix}$$

$$w_t \rightarrow \mathcal{N}(0, R) \qquad R = \begin{pmatrix} r_x^2 & 0 \\ 0 & r_y^2 \end{pmatrix}$$

Particle Filters for Visual Tracking

The likelihood function is based on similarity between image features : one can use color, gradient or any other distinctive feature



- Since color is a distinctive and quite robust feature, each particle is dscribed by the color distribution of its local image patch, $p(\mathbf{x}) = \{p_u(\mathbf{x})\}_{u=1,\ldots,m}$ given by :

$$p_u(\mathbf{x}) = C \sum_{i=1}^{N_p} k(\left\| \frac{\mathbf{x}_i - \mathbf{x}}{h} \right\|^2) \delta[b(\mathbf{x}_i) - u],$$

where C is a normalizer, δ is the Kronecker function, k is a kernel with bandwith h, N_p is the number of pixels in the region and $b(\mathbf{x}_i)$ is a function that assigns one of the m-bins to a given color at location $\mathbf{x}_i$. The kernel k is used to consider spatial information by lowering the contribution of farther pixels.

- The similarity between two color distributions, $p(\mathbf{x}_t)$ and $p^*(\mathbf{x}_0)$, can be defined as the Bhattacharyya distance

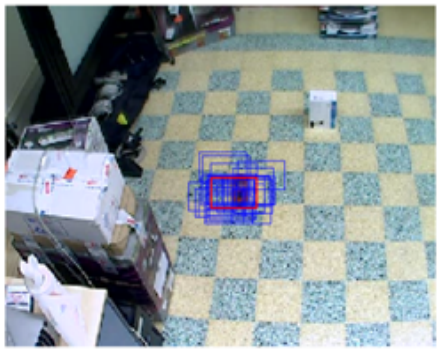
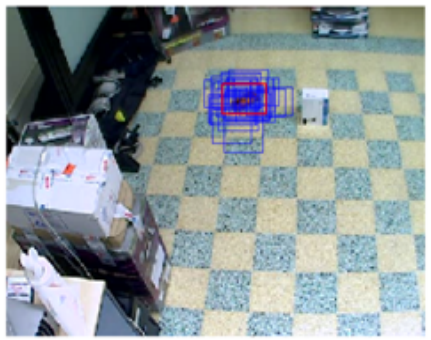
$$\rho[p^*, p(\mathbf{x}_t)] = \left[1 - \sum_{u=1}^{m} \sqrt{p_u^*(\mathbf{x}_0) p_u(\mathbf{x}_t)}\right]^{\frac{1}{2}}$$

- Each sample $\mathbf{x}_t^i$ is assigned an importance weight which corresponds to the likelihood that $\mathbf{x}_t^i$ is the true location of the object. The weights are given by the observation likelihood :

$$w_t^i = p(\mathbf{z}_t|\mathbf{x}_t^i) \propto e^{-\lambda \rho[p^*, p(\mathbf{x}_t^i)]^2}.$$

Particle Filters for Visual Tracking

For every frame, the weighted mean of the particles (MMSE estimate) or the particle with largest weight (MAP estimate) can be used as an estimate of the object's current state.


Tracking algorithm

- Initialize the state vector for the first frame and get a reference color model
- Generate a set of N particles
- For each new frame
  - Find the predicted state of each particle using the state equation
  - Compute histogram distance between the reference color model and the predicted one
  - Weight each particle based on similarity between color models
  - Select the state of the target based on the weighted particles (MMSE or MAP)
  - Sample the particles for next iteration

Outline

Pros and Cons of PF

+ estimation of full pdfs
+ non Gaussian and multi-modal distributions
+ non linear dynamic systems
+ can be parallelized

- degeneracy problem
- high number of particles needed
- can be computationally expensive
- choice of importance density is crucial

Personal Notes

- The principal is quite simple (basically Bayes rule and Markov models)
- Though I did not talk about the math (convergence properties, complexity issues, etc)
- Writing a code that do what is expected can be tricky !
- But works well when you succeed !

Challenges in Visual Tracking

- **Changing appearance**
  - Discriminant and adapting features
  - Optimal combination of multiple feature modalities
  - Binary classification approach

- **Quantitative evaluation**
  - There is a need for a standard a complete dataset.

- **Non conventional sensors**
  - Tracking beyond the visible spectrum
  - Tracking in omnidirectional image sequences is still an open problem

References

A. Doucet, N. De Freitas and D. Gordon (2001).

Sequential Monte Carlo Methods in Practice.

Springer Verlag.

M. S. Arulampalam and S. Maskell and N. Gordon and T. Clapp (2002).

A Tutorial on Particle Filters for Online Nonlinear/Non-Gaussian Bayesian Tracking.

IEEE Transactions on Signal Processing, 50(2), pp : 174-188.

M. Isard and A. Blake (1998).

CONDENSATION - conditional density propagation for visual tracking.

Int. J. Computer Vision, 29(1), pp : 5–28.