1

Advanced Video Analysis and Imaging (5LSH0), Module 7

Object Level Content Analysis
Part II: Tracking

Jungong Han

For questions: (a.pourtaherian@tue.nl)

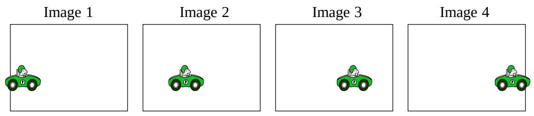
2

Outline

* Introduction & Background
* Mean-Shift Algorithm
  – theory & applications
  – mean-shift object tracking
  – implementation
* Particle-Filter Algorithm
  – theory & applications
  – particle-filter object tracking
  – implementation
* Results and Comparisons

3

The Tracking Problem (1)



* Can we estimate the position of the object?
* Can we estimate its speed?
* Can we predict future positions?
* We can!
* Given Sequence of Images
* Find center of moving object

4

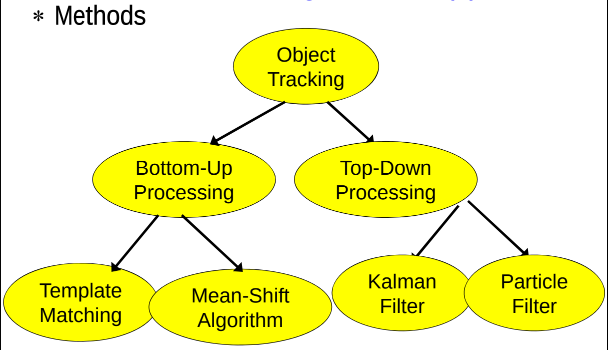
The Tracking Problem (2)

* A straightforward solution
  – we assume: we can find object in the first frame
  – we track across multiple images
* Is this easy?
  – is a fundamental problem in computer vision
* Some of the main challenges
  – objects with many degrees of freedom
  – occlusion or scale
  – multiple objects and background clutter
  – camera may move

5

The Tracking Problem (3)

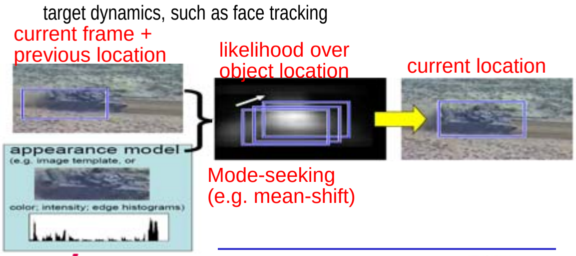
* Methods



6

The Tracking Problem (4)

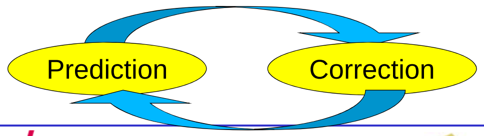
* Bottom-Up Method (template matching, mean-shift)
  – target representation is reliable
  – target localization is possible
  – suitable to the case that relies more on target representation than on target dynamics, such as face tracking



The Tracking Problem (5)

7

- Top-Down Method (kalman filter, particle filter)
  - prediction
    - estimate future state: obtain "a-priori" estimation
  - correction
    - measure the prediction based on the current observations: obtain "a-posterior" estimation
  - better for the situation where the target motion is more reliable, like aerial video surveillance



Outline

8

- Introduction & Background
- Mean-Shift Algorithm
  - theory & applications
  - mean-shift object tracking
  - implementation
- Particle-Filter Algorithm
  - theory & applications
  - particle-filter object tracking
  - implementation
- Results and Comparisons

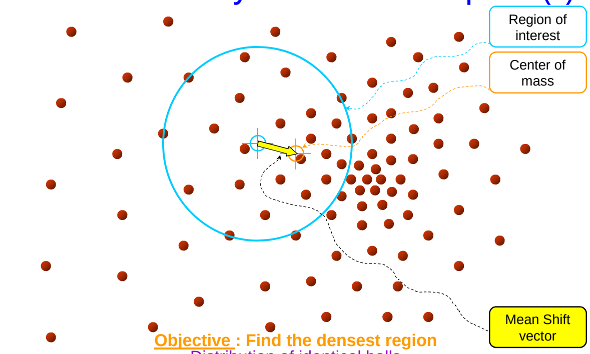
Mean-Shift Algorithm: History

9

- K. Fukunaga and L. Hostetler. "The Estimation of the Gradient of a Density Function, with Applications in Pattern Recognition" IEEE Trans. IT, 1975
- Y. Cheng. "Mean Shift, Mode Seeking, and Clustering" IEEE Trans. PAMI, 1995
- D. Comaniciu, P Meer. "Robust analysis of feature spaces: color image segmentation" CVPR,1997
- GR. Bradski. "Computer vision face tracking for use in a perceptual user interface". Intel Technology Journal 1998

Mean-Shift Theory: Intuitive Description (1)

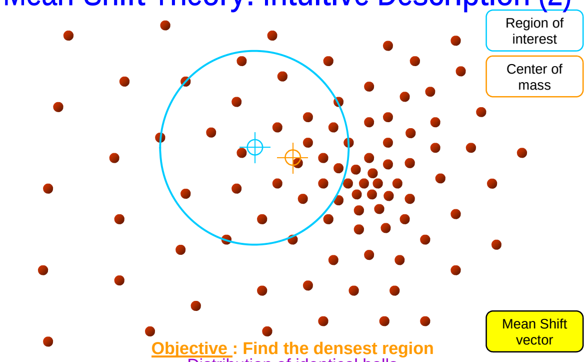
10



Objective : Find the densest region
Distribution of identical balls

Mean-Shift Theory: Intuitive Description (2)

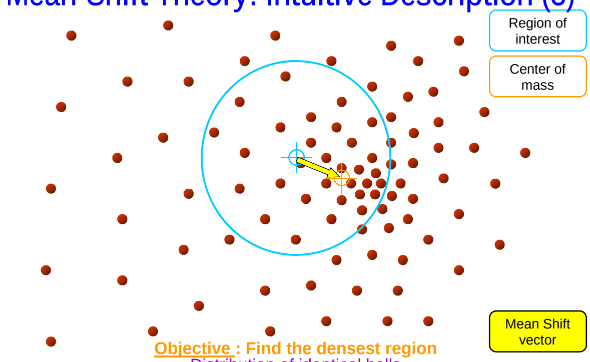
11



Objective : Find the densest region
Distribution of identical balls

Mean-Shift Theory: Intuitive Description (3)

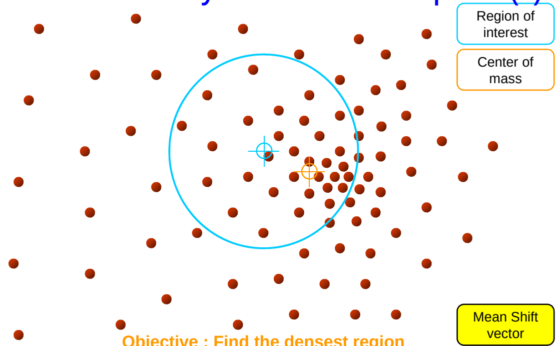
12



Objective : Find the densest region
Distribution of identical balls

Mean-Shift Theory: Intuitive Description (4)

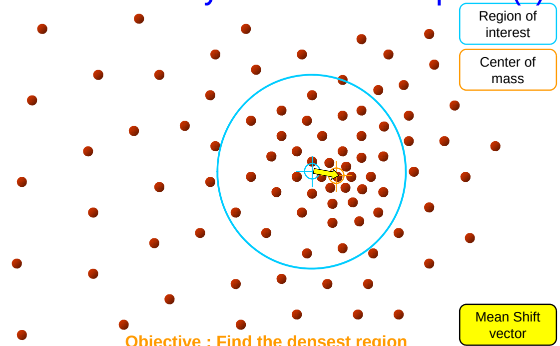
13



Objective : Find the densest region
Distribution of identical balls

Mean-Shift Theory: Intuitive Description (5)

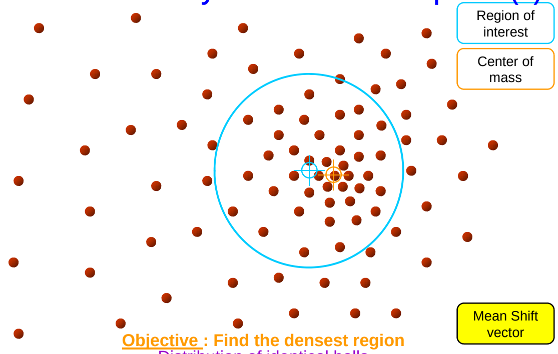
14



Objective : Find the densest region
Distribution of identical balls

Mean-Shift Theory: Intuitive Description (6)

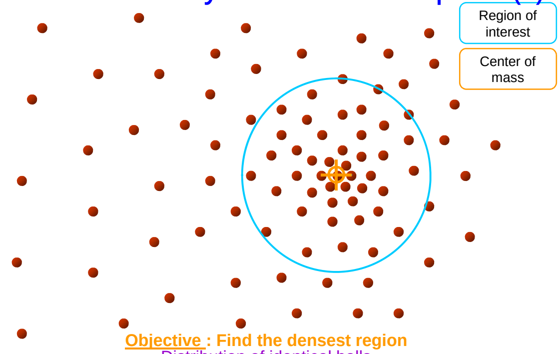
15



Objective : Find the densest region
Distribution of identical balls

Mean-Shift Theory: Intuitive Description (7)

16

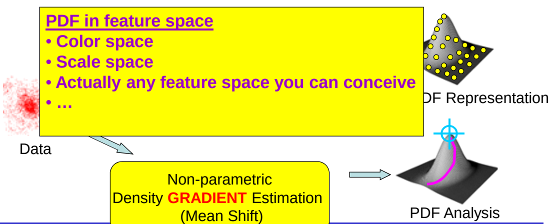


Objective : Find the densest region
Distribution of identical balls

MS: What is Mean Shift ?

17

A tool for:
Finding modes in a set of data samples, manifesting an underlying probability density function (PDF) in R^N

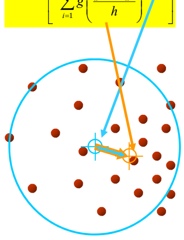


MS: Mean Shift Vector

18

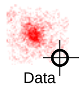
- Given & Task
  - data points and initial mean of the data
  - exact location of the mean of the data
- Mean Shift Vector
  - n: number of the points in the kernel
  - x: initial mean location
  - x_i: data point
  - $g(x)$: kernel function; h: kernel radius
- Properties
  - points towards the direction of gradient
  - towards the direction of the maximal increase in the density

$$\mathbf{m}(\mathbf{x}) = \left[\frac{\sum_{i=1}^{n} \mathbf{x}_i g\left(\frac{\|\mathbf{x}-\mathbf{x}_i\|^2}{h} \right)}{\sum_{i=1}^{n} g\left(\frac{\|\mathbf{x}-\mathbf{x}_i\|^2}{h} \right)} \right] - \mathbf{x}$$



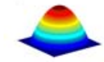
MS: Kernel Density Estimation
19

$P(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} K(\mathbf{x}-\mathbf{x}_i)$ A function of some finite number of data points $x_1 \ldots x_n$

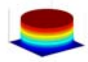


Examples:

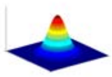
Epanechnikov Kernel $K_E(\mathbf{x}) = \begin{cases} c\left(1-\|\mathbf{x}\|^2\right) & \|\mathbf{x}\| \le 1 \\ 0 & \text{otherwise} \end{cases}$



Uniform Kernel $K_U(\mathbf{x}) = \begin{cases} c & \|\mathbf{x}\| \le 1 \\ 0 & \text{otherwise} \end{cases}$



Normal Kernel $K_N(\mathbf{x}) = c \cdot \exp\left(-\frac{1}{2}\|\mathbf{x}\|^2\right)$



MS: Kernel Density $\underbrace{\text{Estimation}}_{\textit{Gradient}}$
20

$$\nabla P(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} \nabla K(\mathbf{x}-\mathbf{x}_i)$$

Give up estimating the PDF !
Estimate **ONLY** the gradient

Using the Kernel form: $K(\mathbf{x}-\mathbf{x}_i) = ck\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2\right)$

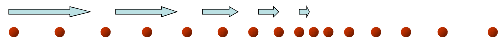
Size of window

We get :

$$\nabla P(\mathbf{x}) = \frac{c}{n}\sum_{i=1}^{n} \nabla k_i = \frac{c}{n}\left[\sum_{i=1}^{n} g_i\right] \cdot \left[\frac{\sum_{i=1}^{n} \mathbf{x}_i g_i}{\sum_{i=1}^{n} g_i} - \mathbf{x}\right]$$

$$\mathbf{g}(\mathbf{x}) = -k'(\mathbf{x})$$

MS: Strengths & Weaknesses
21



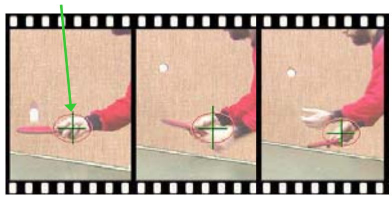
Strengths:
- application independent tool
- suitable for real data analysis
- does not assume any prior shape (e.g. elliptical) on data clusters
- can handle arbitrary feature spaces
- only ONE parameter to choose

Weaknesses:
- the window size (bandwidth selection) is not trivial
- inappropriate window size can cause modes to be merged, or generate additional “shallow” modes ➔ Use adaptive window size

MS: Object Tracking (1)
22

General Framework: Target Representation

Choose a reference model in the current frame → Choose a feature space → Represent the model in the chosen feature space



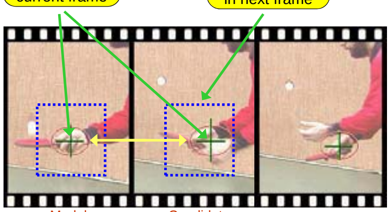
… Current frame → …

Some pictures are from Yaron Ukrainitz

MS: Object Tracking (2)
23

General Framework: Target Localization

Start from the position of the model in the current frame → Search in the model's neighborhood in next frame → Find best candidate by maximizing a similarity func.



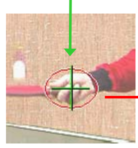
… Current frame → …

Repeat the same process in the next pair of frames

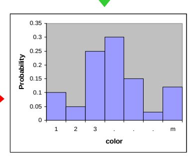
MS: Object Tracking (3)
24

Target Representation

Choose a reference target model → Choose a feature space → Represent the model by its PDF in the feature space



Quantized Color Space



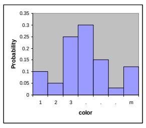
MS: Object Tracking (4)

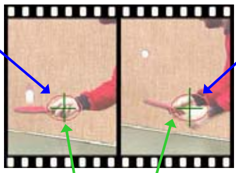
25

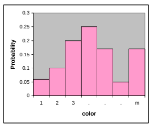
PDF Representation

Target Model
(centered at 0)

Target Candidate
(centered at y)







$\vec{q} = \{q_u\}_{u=1..m} \quad \sum_{u=1}^{m} q_u = 1$

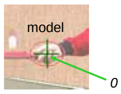
$\vec{p}(y) = \{p_u(y)\}_{u=1..m} \quad \sum_{u=1}^{m} p_u = 1$

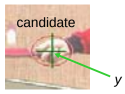
Similarity Function: $f(y) = f[\vec{q}, \vec{p}(y)]$

MS: Object Tracking (5)

26

Finding the PDF of the target model





$\{x_i\}_{i=1..n}$ Target pixel locations

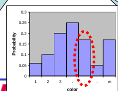
$k(x)$ A monotonically decreasing kernel

$b(x)$ The color bin index (1..*m*) of pixel *x*

Probability of feature u in model

$$q_u = C \sum_{b(x_i)=u} \underbrace{k\left(\|x_i\|^2\right)}_{\text{Pixel weight}}$$

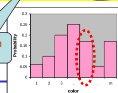
Normalization factor



Probability of feature u in candidate

$$p_u(y) = C_h \sum_{b(x_i)=u} \underbrace{k\left(\left\|\frac{y-x_i}{h}\right\|^2\right)}_{\text{Pixel weight}}$$

Normalization factor



MS: Object Tracking (6)

27

Similarity Function

Target model: $\vec{q} = (q_1, \ldots, q_m)$

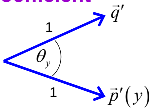
Target candidate: $\vec{p}(y) = (p_1(y), \ldots, p_m(y))$

Similarity function: $f(y) = f[\vec{p}(y), \vec{q}] = ?$

The Bhattacharyya Coefficient

$\vec{q}' = \left(\sqrt{q_1}, \ldots, \sqrt{q_m}\right)$

$\vec{p}'(y) = \left(\sqrt{p_1(y)}, \ldots, \sqrt{p_m(y)}\right)$



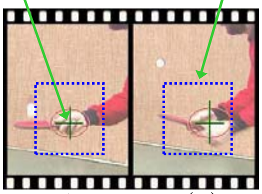
$$f(y) = \cos\theta_y = \frac{p'(y)^T q'}{\|p'(y)\| \cdot \|q'\|} = \sum_{u=1}^{m} \sqrt{p_u(y) q_u}$$

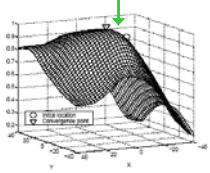
MS: Object Tracking (7)

28

Target Localization Algorithm

Start from the position of the model in the current frame → Search in the model's neighborhood in next frame → Find best candidate by maximizing a similarity func.





$\vec{q}$ $\quad \vec{p}(y)$ $\quad f[\vec{p}(y), \vec{q}]$

MS: Object Tracking (8)

29

Approximating the Similarity Function

$f(y) = \sum_{u=1}^{m} \sqrt{p_u(y) q_u}$

Model location: y_0
Candidate location: y

Linear approx. (around y_0)

$$f(y) \approx \underbrace{\frac{1}{2}\sum_{u=1}^{m}\sqrt{p_u(y_0) q_u}}_{\text{Independent of } y} + \frac{1}{2}\sum_{u=1}^{m} \boxed{p_u(y)} \boxed{\sqrt{\frac{q_u}{p_u(y_0)}}}$$

$p_u(y) = C_h \sum_{b(x_i)=u} \boxed{k\left(\left\|\frac{y-x_i}{h}\right\|^2\right)}$

$$\frac{C_h}{2} \sum_{i=1}^{n} \boxed{w_i} k\left(\left\|\frac{y-x_i}{h}\right\|^2\right)$$

Density estimate! (as a function of *y*)

MS: Object Tracking (9)

30

Maximizing the Similarity Function

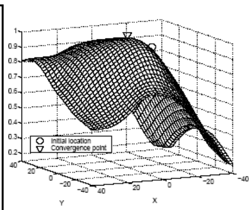
The mode of $\frac{C_h}{2} \sum_{i=1}^{n} w_i k\left(\left\|\frac{y-x_i}{h}\right\|^2\right)$ = sought maximum

Important Assumption:

The target representation provides sufficient discrimination

↓

One mode in the searched neighborhood



MS: Implementation

31

1. Initialize the location of the target in the current frame with $\hat{\mathbf{y}}_0$, compute $\{\hat{p}_u(\hat{\mathbf{y}}_0)\}_{u=1\ldots m}$, and evaluate

$$\rho[\hat{\mathbf{p}}(\hat{\mathbf{y}}_0), \hat{\mathbf{q}}] = \sum_{u=1}^{m} \sqrt{\hat{p}_u(\hat{\mathbf{y}}_0)\hat{q}_u} .$$

2. Derive the weights $\{w_i\}_{i=1\ldots n_h}$ according to (10).
3. Find the next location of the target candidate according to (11).
4. Compute $\{\hat{p}_u(\hat{\mathbf{y}}_1)\}_{u=1\ldots m}$, and evaluate

$$\rho[\hat{\mathbf{p}}(\hat{\mathbf{y}}_1), \hat{\mathbf{q}}] = \sum_{u=1}^{m} \sqrt{\hat{p}_u(\hat{\mathbf{y}}_1)\hat{q}_u} .$$

5. While $\rho[\hat{\mathbf{p}}(\hat{\mathbf{y}}_1), \hat{\mathbf{q}}] < \rho[\hat{\mathbf{p}}(\hat{\mathbf{y}}_0), \hat{\mathbf{q}}]$
   Do $\hat{\mathbf{y}}_1 \leftarrow \frac{1}{2}(\hat{\mathbf{y}}_0 + \hat{\mathbf{y}}_1)$
   Evaluate $\rho[\hat{\mathbf{p}}(\hat{\mathbf{y}}_1), \hat{\mathbf{q}}]$
6. If $\|\hat{\mathbf{y}}_1 - \hat{\mathbf{y}}_0\| < \epsilon$ Stop.
   Otherwise Set $\hat{\mathbf{y}}_0 \leftarrow \hat{\mathbf{y}}_1$ and go to Step 2.

MS: Conclusion and Reference

32

* Advantages
  – good color histogram model and distance measure
  – the mean shift usually converged at 4 to 6 iterations – fast
* Disadvantages
  – sometimes get stuck at local maximum
  – scale of the kernel/target is always changed
* References
  – D. Comaniciu, V. Ramesh and P. Meer. “Kernel-Based Object Tracking”, IEEE PAMI, 2003.

Outline

33

* Introduction & Background
* Mean-Shift Algorithm
  – theory & applications
  – mean-shift object tracking
  – implementation
* Particle-Filter Algorithm
  – theory & applications
  – particle-filter object tracking
  – implementation
* Results and Comparisons

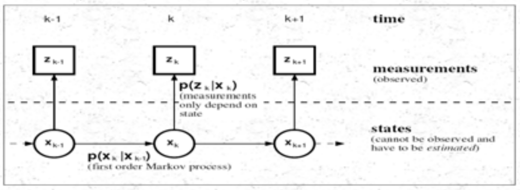
Particle-Filter: History

34

* First attempts – simulations of growing polymers
  – M. N. Rosenbluth and A.W. Rosenbluth. “Monte Carlo calculation of the average extension of molecular chains,” Journal of Chemical Physics,1956
* First application in signal processing
  – N. J. Gordon, D. J. Salmond, and A. F. M. Smith. “Novel approach to nonlinear/non-Gaussian Bayesian state estimation,” IEE Proceedings-F, 1993
* Application in object tracking
  – K. Nummiaro, E. Koller-Meier and L. Van Gool. “An Adaptive Color-Based Particle Filter,” Image and Vision Computing, 2002

Particle-Filter: Theory (1)

35



* Assumptions
  – the observations are conditionally independent given the state

$P(z_{k+1}|x_{k+1},z_{1:k}) = P(z_{k+1}|x_{k+1})$

  – Markov process: $P(x_{k+1}|x_k,x_{k-1...}) = P(x_{k+1}|x_k)$

Particle-Filter: Theory (2)

36

* Objective
  – we seek estimates of x_k based on all available measurements up to time k
  – we attempt to construct/compute the posterior pdf of the state given all measurements: $p(x_k|z_{1:k})$
* Method
  – Recursive filter, which consists of two steps

Prediction Step: $p(x_{k-1}|z_{1:k-1}) \longrightarrow p(x_k|z_{1:k-1})$

Update Step: $p(x_k|z_{1:k-1}),z_k \longrightarrow p(x_k|z_{1:k})$

PF: General Prediction-Update (1)
37

* Assume that $p(x_{k-1}|z_{1:k-1})$ is available at time *k-1*
* Prediction step:

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

This is the prior of the state x_k at time k without knowledge of the measurement z_k, i.e. the probability given only previous measurements

* Update step: (compute posterior pdf from predicted prior pdf and new measurement)

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

PF: General Prediction-Update (2)
38

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

?

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

PF: Structure of the Update Equation
39

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

$$\boxed{\text{posterior} = \frac{\text{likelihood}\cdot\text{prior}}{\text{evidence}}}$$

* Prior: given by prediction equation
* Likelihood: given by observation model
* Evidence: the normalizing constant in the denominator

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

PF Theory: Problem Statement
40

* This theoretically allows an optimal Bayesian solution
* Problem: only a conceptual solution
* Solutions
  - kalman filter: parametric method
    - assume prior $p(x)$, observation $p(z|x)$ and posterior $p(x|z)$ are a normal distribution
    - mean μ_k and covariance matrix Σ_k can be derived analytically
    - sequentially update μ_k and Σ_k for each time step k
  - particle filter: non-parametric method

PF Theory: Monte Carlo Sampling
41

* Monte Carlo Sampling (Particle filter)

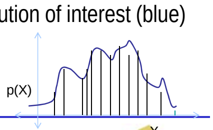
If we cannot solve the integrals required for a Bayesian recursive filter, we represent the posterior probabilities by a set of randomly chosen weighted samples

"randomly chosen" = "Monte Carlo"

increasing number of samples = convergence to true pdf

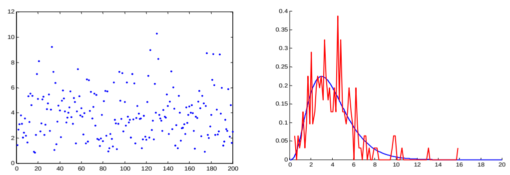
* What is a particle?
  - $p(x)$: continuous probability distribution of interest (blue)

$$p(x) \approx \sum_{i=1}^{n} w^i \delta(x - x^i)$$



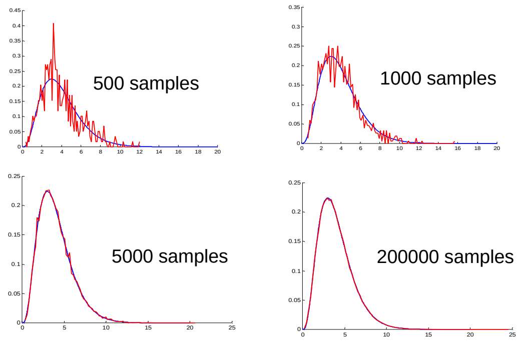
PF Theory: Random Samples and PDF (1)
42

* Take $p(x)$=Gamma function
* Generate some random samples
* Plot histogram and basic approximation to pdf



200 samples

PF Theory: Random Samples and PDF (2)
43



PF Theory: Importance Sampling (1)
44

- Unfortunately it is often not possible to sample directly from the posterior distribution, but we can use importance sampling
- Let $p(x)$ be a pdf from which it is difficult to draw samples
- Let $x^i \sim q(x), i=1, \ldots, N$, be samples that are easily generated from a proposal pdf q, which is called an *importance density*
- Then approximation to the density p is given by

$$p(x) \approx \sum_{i=1}^{n} w^i \delta(x - x^i) \quad \text{where} \quad w^i = \frac{p(x^i)}{q(x^i)}$$

PF Theory: Importance Sampling (2)
45

- We obtain a recursive estimate of the importance weights

$$\boxed{w_k^i = w_{k-1}^i \frac{p(z_k \mid z_{1:k-1}, x_{1:k}^i)\, p(x_k^i \mid x_{1:k-1}^i)}{q(x_k^i \mid x_{1:k-1}^i, z_{1:k})}}$$

- Use Markov assumption

$$\boxed{w_k^i = w_{k-1}^i \frac{p(z_k \mid x_k^i)\, p(x_k^i \mid x_{k-1}^i)}{q(x_k^i \mid x_{k-1}^i, z_k)}}$$

- Choose the proposal distribution

$$q(x_k \mid x_{1:k-1}, z_{1:k}) = p(x_k \mid x_{k-1})$$

- We obtain then

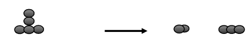
$$w_k^i = w_{k-1}^i \frac{p(z_k \mid x_k^i)\, p(x_k^i \mid x_{k-1}^i)}{q(x_k^i \mid x_{1:k-1}^i, z_{1:k})} = w_{k-1}^i\, p(z_k \mid x_k^i)$$

PF Theory: Degeneracy Problem
46

- After a few iterations, most particles have negligible weight
- The weight is concentrated on a few particles only
- Resampling

Without resampling:

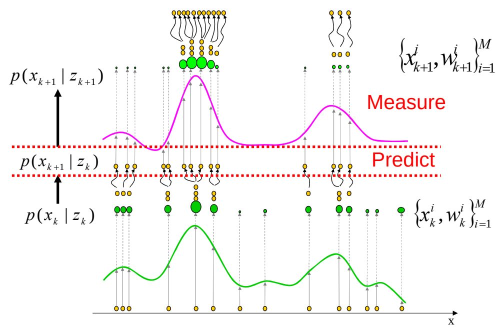
With resampling:



PF Theory: Algorithm Implementation (1)
47

- Step 1: Generate samples to represent the initial probability
- Step 2: Using the prior equation, predict the next state
- Step 3: Using the observation, get the weights for the states computed
- Step 4: Resample it so as to have the uniformly distributed current state omitting the least-significant representation
- Step 5: Continue steps 2 through 4, till all the observations are exhausted

PF Theory: Algorithm Implementation (2)
48



PF Object Tracking: An Example (1)

* What we want to compute

$$\underbrace{p(x_k \mid z_{1:k})}_{\text{current object state}} = \alpha \underbrace{p(z_k \mid x_k)}_{\text{observation model}} \int_{x_{k-1}} \underbrace{p(x_k \mid x_{k-1})}_{\text{transition model}} \underbrace{p(x_{k-1} \mid z_{1:k-1})}_{\text{previous object state}} dx_{k-1}$$

$$p(x_k \mid z_{1:k}) \approx \alpha p(z_k \mid x_k) \sum_i w_k{}^i p(x_k \mid x_{k-1}{}^i)$$

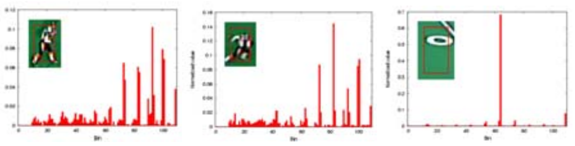
* Prior distribution: $p(x_0)$
  – describes initial distribution of object states
* Transition Model: $p(x_k|x_{k-1})$
  – specifies how objects move between frames
* Observation model: $p(z_k|x_k)$

PF Object Tracking: An Example (2)

* The prior distribution: $p(x_0)$
  – could be based on an object detector
* The transition Model: $p(x_k|x_{k-1})$
  – a simple model: a Gaussian window around current state
  – a better model: consider previous states for velocity and acceleration

PF Object Tracking: An Example (3)

* The observation model: $p(z_k|x_k)$
  – use a simple HSV/RGB/YUV histogram-based model

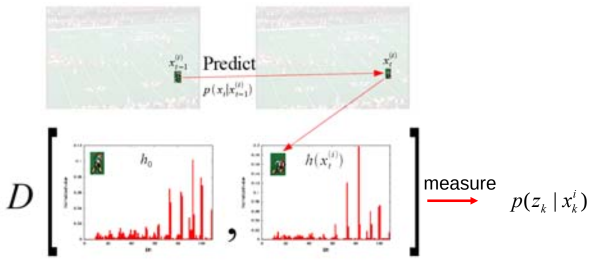


  – likelihood is based on a distance metric D between histograms h_0 and $h(x_t)$

$$p(z_k \mid x_k) \propto e^{-\lambda D^2[h_0, h(x_k)]}$$

PF Object Tracking: An Example (4)

* Particles for object tracking
  – for a given particle at time k-1, we



PF Object Tracking: An Example (5)

* Particles for object tracking
  – each particle is weighted by its likelihood
  – most likely particle represents the object state at time t

PF Object Tracking: An Example (6)

Given the sample set S_{t-1} and the target model

$q = C \sum_{i=1}^{I} k\left(\frac{\|x_i\|}{a}\right) \delta\left[h(x_i) - u\right]$,

perform the following steps:

(1) Select N samples from set S_{t-1} with probability $\pi_{t-1}^{(n)}$:

(a) calculate the normalized cumulative probabilities c'_{t-1}

$c_{t-1}^{(0)} = 0$

$c_{t-1}^{(n)} = c_{t-1}^{(n-1)} + \pi_{t-1}^{(n)}$

$c'^{(n)}_{t-1} = \frac{c_{t-1}^{(n)}}{c_{t-1}^{(N)}}$

(b) generate a uniformly distributed random number $r \in [0, 1]$

(c) find, by binary search, the smallest j for which $c'^{(j)}_{t-1} \geq r$

(d) set $s'^{(n)}_{t-1} = s_{t-1}^{(j)}$

(2) Propagate each sample from the set S'_{t-1} by a linear stochastic differential equation:

$s_{t-1}^{(n)} = A s'^{(n)}_{t-1} + w_{t-1}^{(n)}$

where $w_{t-1}^{(n)}$ is a multivariate Gaussian random variable

(3) Observe the color distribution

(a) calculate the color distribution

$p_{s_t^{(n)}}^{(u)} = C \sum_{i=1}^{I} k\left(\frac{\|s_t^{(n)} - x_i\|}{a}\right) \delta\left[h(x_i) - u\right]$

for each sample of the set S_t

(b) calculate the Bhattacharyya coefficient for each sample of the set S_t

$\rho[p_{s_t^{(n)}}, q] = \sum_{u=1}^{m} \sqrt{p_{s_t^{(n)}}^{(u)} q^{(u)}}$

(c) weight each sample of the set S_t

$\pi_t^{(n)} = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(1-\rho[p_{s_t^{(n)}}, q])}{2\sigma^2}}$

(4) Estimate the mean state of the set S_t

$E[S_t] = \sum_{n=1}^{N} \pi_t^{(n)} s_t^{(n)}$

Fig. 5. Tracking algorithm

PF: Conclusion and Reference

- Advantages
  - able to represent arbitrary density
  - converging to true posterior even for non-Gaussian and nonlinear system
  - results with high-accuracy
- Disadvantages
  - complexity grows exponentially in the dimensions
- References
  - K. Nummiaro, E. Koller-Meier and L. Van Gool, "An adaptive color-based particle filter", *Image and Vision Computing, pp. 99-110,* 2002.

Outline

- Introduction & Background
- Mean-Shift Algorithm
  - theory & applications
  - mean-shift object tracking
  - implementation
- Particle-Filter Algorithm
  - theory & applications
  - particle-filter object tracking
  - implementation
- Results and Comparisons

Object Tracking: Comparison (1)

Sequence	Tracker	Particle amount	Color similarity between target and background	Background noise in selected blob	Correct rate	Occlusion exists in sequence	Solving occlusion
1-case1	MS	x	Medium	High	Failed	No	x
	PF	800	Medium	High	93%	No	x
	PF	200	Medium	High	90%	No	x
1-case2	MS	x	High	Low	Failed	No	x
	PF	800	High	Low	88%	No	x
	PF	200	High	Low	80%	No	x
1-case3	MS	x	Low	Low	100%	No	x
	PF	100	Low	Low	100%	No	x
2-case1	MS	x	Low	Medium	100%	No	x
	PF	800	Low	Medium	100%	No	x
2-case2	MS	x	Low	Medium	<50% 100%	Yes	Failed
	PF	100	Low	Medium	89% 98%	Yes	Success
3	MS	x	Medium	Low	100%	No	x
	PF	200	Medium	Low	100%	No	x
4	MS	x	Low	Low	<50% <50%	Yes	Failed
	PF	200	Low	Low	100% 100%	Yes	Success
5	MS	x	High	High	90% 70%	Yes	Failed
	PF	200	High	High	<50% <50%	Yes	Failed
6	MS	x	Low	Low	98%	No	x
	PF	200	Low	Low	<50%	No	x
7	MS	x	Low	High	99%	No	x
	PF	200	Low	High	<50%	No	x

Object Tracking: Comparison (2)

- For situations that both methods are available, the particle filter's accuracy is often higher than mean shift tracking
- Particle filter is better than Mean shift when the occlusion occurs. But both of them is unable to solve complete occlusion problem
- Sometimes the blob includes lots of noise e.g. target in sequence 7, the particle filter does not work. However, the mean shift can still keep on tracking

Object Tracking: Comparison (3)

- Particle filter has memory effect which is different with mean shift tracking
  - if the target has a sudden movement the blob needs some time to catch and follow it
  - if the target suddenly stops, the blob will go on moving along the previous direction for some time before stop
- The running speed of particle filter-based algorithm is much slower than that of the mean shift algorithm
- If the particle filter loses the target, sometimes it is possible to capture the target object again, but Mean shift cannot

Object Tracking: Comparison (4)

- If the particle filter already successfully tracks the target, it is almost impossible to make it more accurate by only increasing the particle number
- Particle filter has dynamical behaviour that the blob oscillates around the real target during tracking while similar phenomenon can hardly be observed with mean shift