ELEC 2885:
Image Processing and Computer Vision

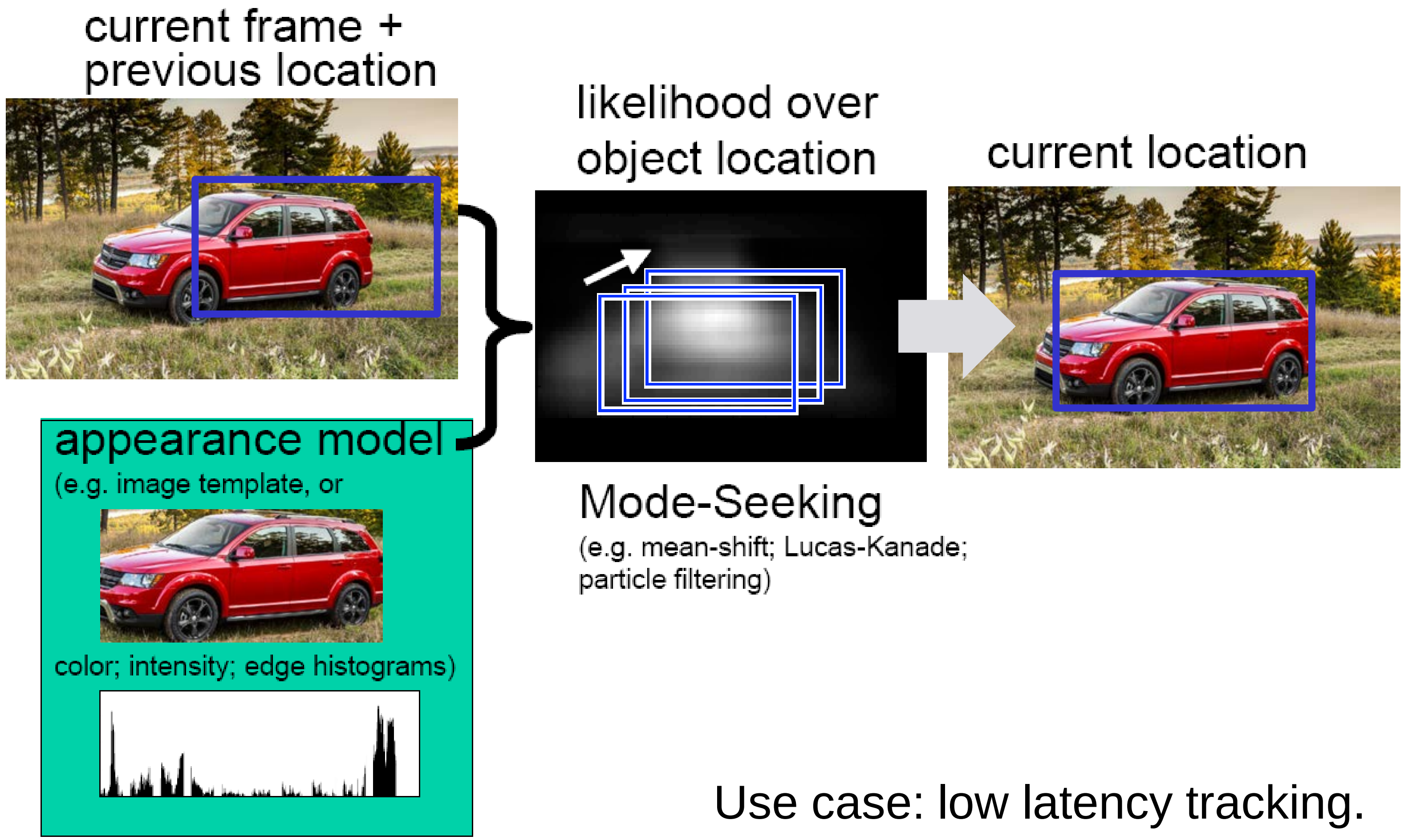
christophe.devleeschouwer@uclouvain.be

Recursive tracking: track2detect

- Introduction
- Template matching: Lucas-Kanade method
- Kernel-based tracking: Mean-shift
- Bayesian recursive estimation: Kalman and particle filters

Credits: part of the slides borrowed from
- Robert Collins, http://www.cse.psu.edu/~rtc12/CSE598G/LKintro_6pp.pdf
- Yaron Ukrainitz and Bernard Sarel: www.wisdom.weizmann.ac.il/~deniss/vision_spring04/files/mean_shift/mean_shift.ppt

Introduction: appearance-based recursive tracking



Use case: low latency tracking.

Tracking

- Introduction
- **Template matching: Lucas-Kanade method**
- Kernel-based tracking: Mean-shift
- Bayesian recursive estimation: Kalman and particle filters

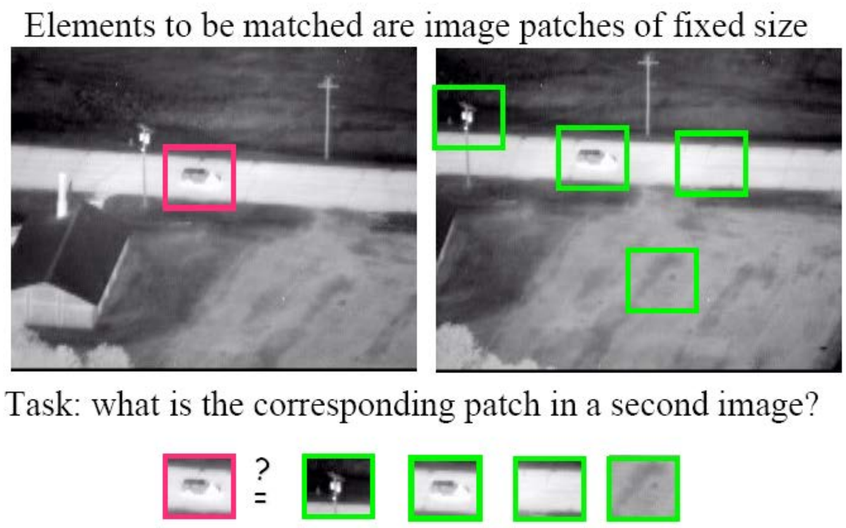
Template matching

- Assumptions:
  - A snapshot of object from first frame can be used to describe appearance along the sequence.
  - Object will look nearly identical in the new image.
  - Movement is nearly pure 2D translation.

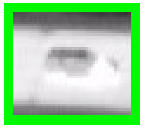
The last two are very restrictive. We will relax them later on.

- Search problem:

  Given an intensity patch element in the previous image, search for the corresponding patch in the current image.



Comparing windows/patches

 $\stackrel{?}{=}$ 

f g

Some possible measures:

$$\stackrel{?}{=} \quad \max_{[i,j]\in R} |f(i,j) - g(i,j)|$$

$$\sum_{[i,j]\in R} |f(i,j) - g(i,j)|$$

Sum of squared difference.

$$SSD = \sum_{[i,j]\in R} (f(i,j) - g(i,j))^2$$

$$C_{fg} = \sum_{[i,j]\in R} f(i,j)g(i,j)$$

Most popular

Substract mean value of template!

Efficient computation: Lucas-Kanade method

- Motivation:
  - Want a more efficient method than explicit search over some large parameter set
  - If we have a good estimate of object position, a gradient descent strategy sounds relevant.

- Fonction to minimize, in case of a simple translation of template:

$$E(u,v) = \sum_{x,y} \underbrace{w(x,y)}_{\text{Window function}} \Big[\underbrace{I(x+u, y+v)}_{\text{Shifted intensity}} - \underbrace{T(x,y)}_{\text{template}} \Big]^2$$

Why can it help ?

Mathematical derivations

$$E(u,v) = \sum [I(x+u, y+v) - T(x,y)]^2$$

$$\approx \sum [I(x,y) + uI_x(x,y) + vI_y(x,y) - T(x,y)]^2 \quad \text{First order approx}$$

$$= \sum [uI_x(x,y) + vI_y(x,y) + D(x,y)]^2$$

Take partial derivs and set to zero

$$\frac{\delta E}{du} = \sum [uI_x(x,y) + vI_y(x,y) + D(x,y)]\, I_x(x,y) = 0$$

$$\frac{\delta E}{dv} = \sum [uI_x(x,y) + vI_y(x,y) + D(x,y)]\, I_y(x,y) = 0$$

Form matrix equation

$$\sum \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = -\sum \begin{bmatrix} I_x D \\ I_y D \end{bmatrix}$$

- **Defines how to update parameters (u,v).**
- **Repeat iteratively because the computation of (u,v) relies on 1st order approximation of E(u,v).**

Main issues: handling changes of appearance

- Recursive update of template:

  Once the target has been located in a new frame, just extract a new template, centered at that location. *Problem: template drift!*

- Template deformation.

  - SSD is computed between the template T and the image warped back onto the coordinate frame of the template:

  $$\sum_{\mathbf{x}} \left[I(\mathbf{W}(\mathbf{x};\mathbf{p})) - T(\mathbf{x}) \right]^2$$

  - Here, W(x;p) denotes the parameterized set of allowed warps. For example, in case of translations, we have:

  $$\mathbf{W}(\mathbf{x};\mathbf{p}) = \begin{pmatrix} x + p_1 \\ y + p_2 \end{pmatrix}$$

  - Wrapping requires interpolating I at sub-pixel locations…

More about Lucas-Kanade: https://www.ri.cmu.edu/pub_files/pub3/baker_simon_2002_3/baker_simon_2002_3.pdf

Tracking

- Introduction
- Template matching: Lucas-Kanade method
- **Kernel-based tracking: Mean-shift**
- Bayesian recursive estimation: Kalman and particle filters

Kernel-based tracking

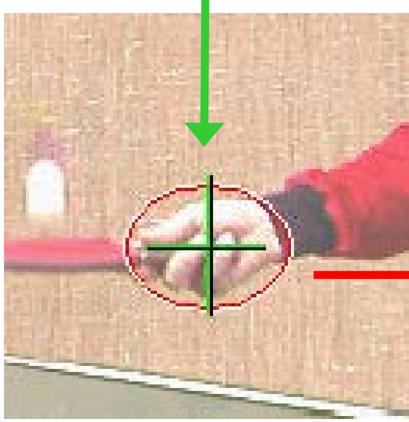
- Motivation:
  - Template-based tracking only able to track small-sized windows, due to the constraint on template deformation.
- Tracking deformable and non-rigid objects:
  - What about defining appearance in a feature space, e.g. based on an histogram?

Example of target representation: target model + feature space

Choose a reference target model

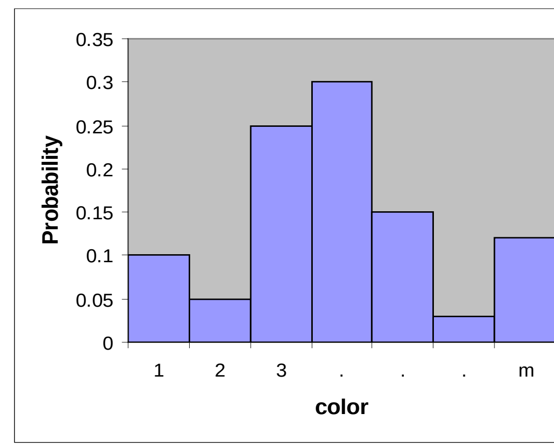
Choose a feature space

Represent the model by its PDF in the feature space



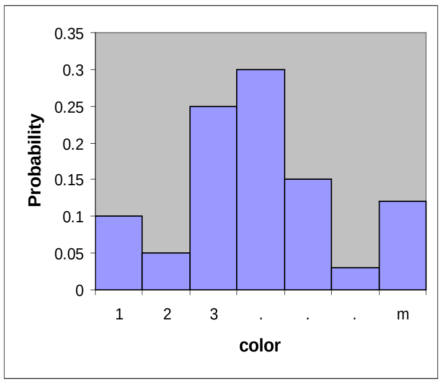
Quantized Color Space

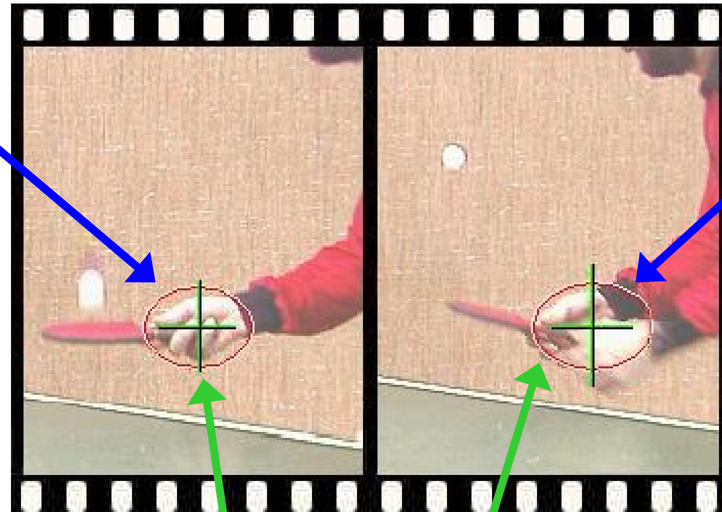
e.g. 16x16x16 quantized RGB



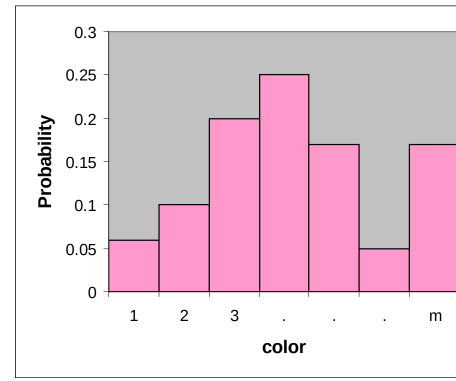
Kernel Based Object Tracking, by Comaniniu, Ramesh, Meer

Target Model
(centered at 0)





Target Candidate
(centered at y)



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

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

Similarity Function:

$$f(y) = f\left[\vec{q}, \vec{p}(y)\right]$$

Similarity function

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

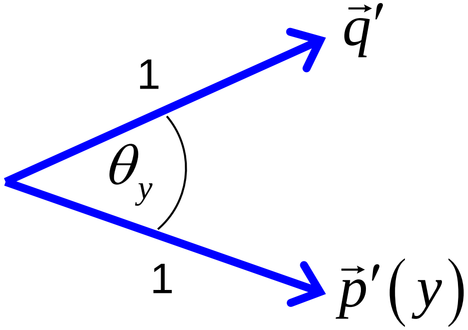
Target candidate: $\vec{p}\left(y\right) = \left(p_1\left(y\right), \ldots, p_m\left(y\right)\right)$

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

The Bhattacharyya Coefficient

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

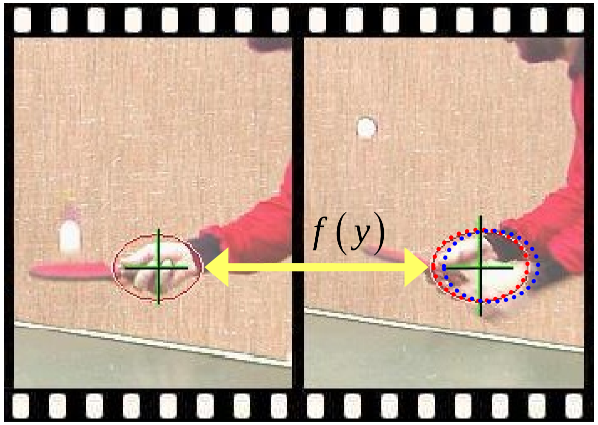
$$\vec{p}'\left(y\right) = \left(\sqrt{p_1\left(y\right)}, \ldots, \sqrt{p_m\left(y\right)}\right)$$



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

Key issue for target localization using gradient descent:
Smoothness of Similarity Function

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



Problem:

Target is represented by color info only → Spatial info is lost → Large similarity variations for adjacent locations →

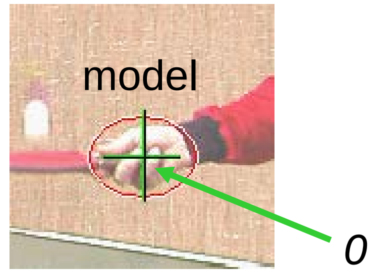
f is not smooth → **Gradient-based optimizations are not robust**

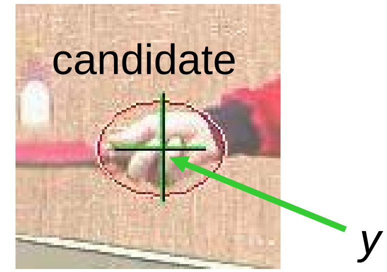
Solution:

Mask the target with an isotropic kernel in the spatial domain → $f(y)$ becomes smooth in y

Kernel-based PDF estimation

$\{x_i\}_{i=1..n}$ Pixel offsets around target center





$k(x)$ A differentiable, isotropic, convex, monotonically decreasing kernel:

Reduces the importance of peripheral pixels since they are affected by occlusion and background interference + make *f(y)* smoother.

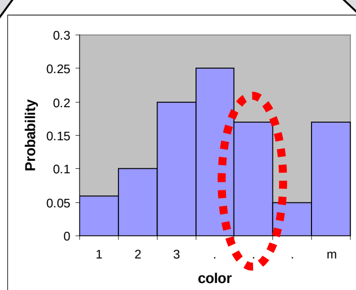
$b(x)$ The color bin index (1..*m*) of pixel *x*

Probability of color u in model

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

Normalization factor

Pixel weight



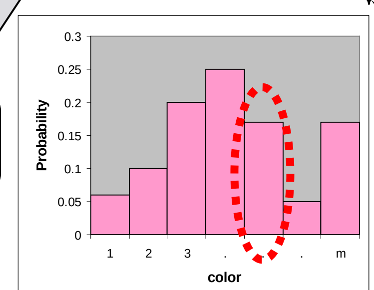
Probability of color u in candidate

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

Normalization factor

Pixel weight

Size of window might adapt to the object Scale.



Kernel-based Object Tracking

Approximating the Similarity Function

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

Initial location: y_0

Candidate location: y

Linear approx. around $p_u(y_0)$

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

Independent of y

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

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

Density estimate! (as a function of y)

$$\boxed{\sum_{u=1}^{m} \sqrt{\frac{q_u}{p_u(y_0)}} \cdot \delta(b(\mathbf{x}_i) - u) = \sqrt{\frac{q_{b(\mathbf{x}_i)}}{p_{b(\mathbf{x}_i)}(y_0)}}}$$

Size of window, might adapt to the object scale

For a given value of h:
Kernel Density Estimation

$$\underbrace{\text{Kernel Density Estimation}}_{\textbf{Gradient}}$$

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

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

<u>Iterative solution:</u>

Search for $\boldsymbol{y_1}$ that sets gradient to zero, based on what we observe in $\boldsymbol{y_0}$.

$$\sum_{i=1}^{n} w_i (\mathbf{y}_1 - \mathbf{x_i}) k'\left(\left\|\frac{\mathbf{y}_0-\mathbf{x_i}}{h}\right\|^2\right) = 0$$

$$\Rightarrow \mathbf{y}_1 = \frac{\sum_{i=1}^{n} w_i \mathbf{x_i} k'\left(\left\|\frac{\mathbf{y}_0-\mathbf{x_i}}{h}\right\|^2\right)}{\sum_{i=1}^{n} w_i k'\left(\left\|\frac{\mathbf{y}_0-\mathbf{x_i}}{h}\right\|^2\right)}$$

Kernel-based tracking

Choosing the Kernel

A special class of radially symmetric kernels:

$$K(x) = ck\left(\|x\|^2\right)$$

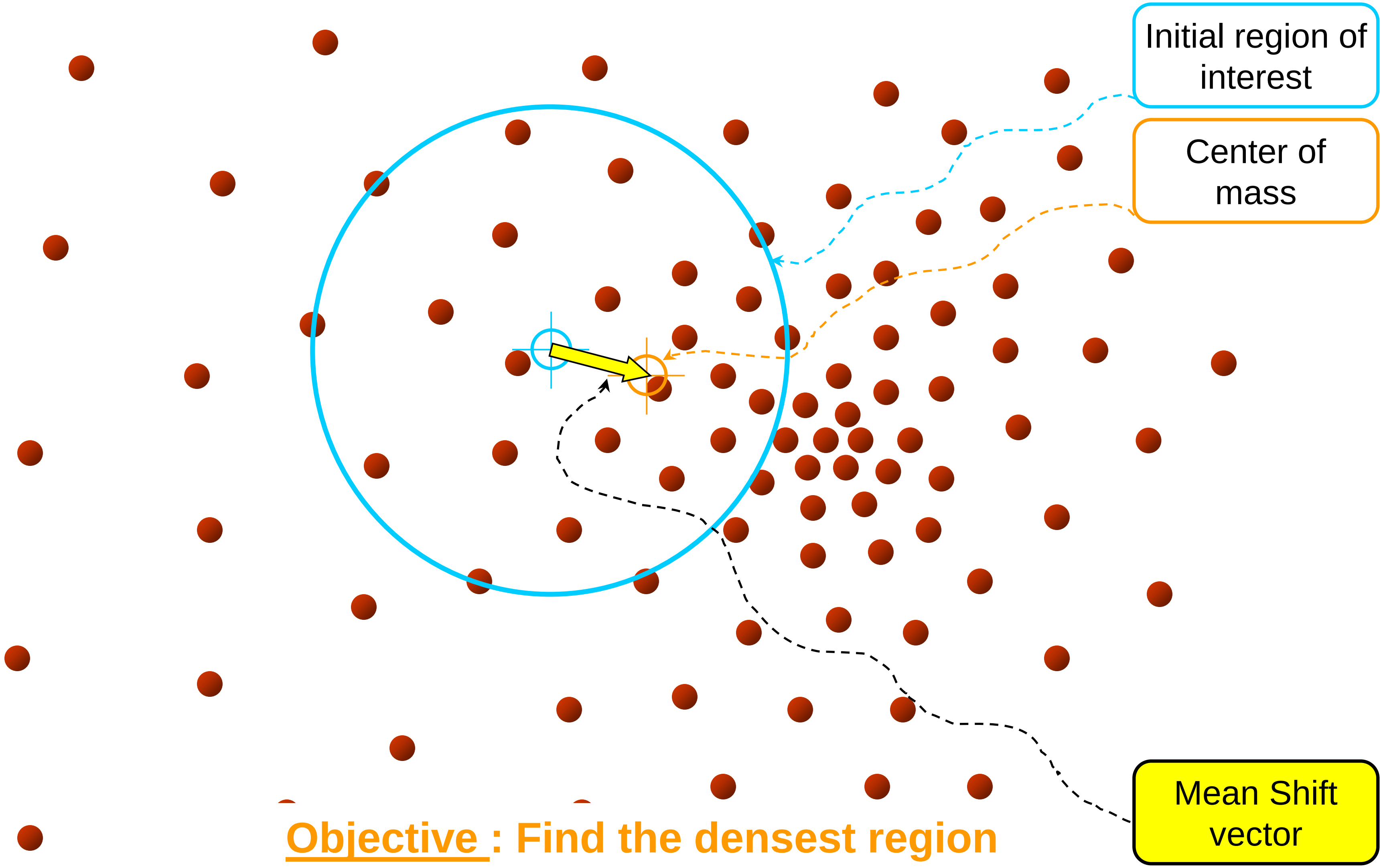
$$= \begin{cases} 1 & \text{if } \|\bullet\| \le 1 \\ 0 & \text{otherwise} \end{cases}$$

$$\boxed{y_1 = \frac{\sum_{i=1}^{n} x_i w_i g\left(\left\|\frac{y_0 - x_i}{h}\right\|^2\right)}{\sum_{i=1}^{n} w_i g\left(\left\|\frac{y_0 - x_i}{h}\right\|^2\right)} \quad \Longrightarrow \quad y_1 = \frac{\sum_{i=1}^{n} x_i w_i}{\sum_{i=1}^{n} w_i}}$$

Interpretation:
Center of mass of w_i in a window centered at y_0 and of radius h.

Note:
the result is also known as
mean-shift algorithm,
a tool for non-parametric PDF maximization

Intuitive Description

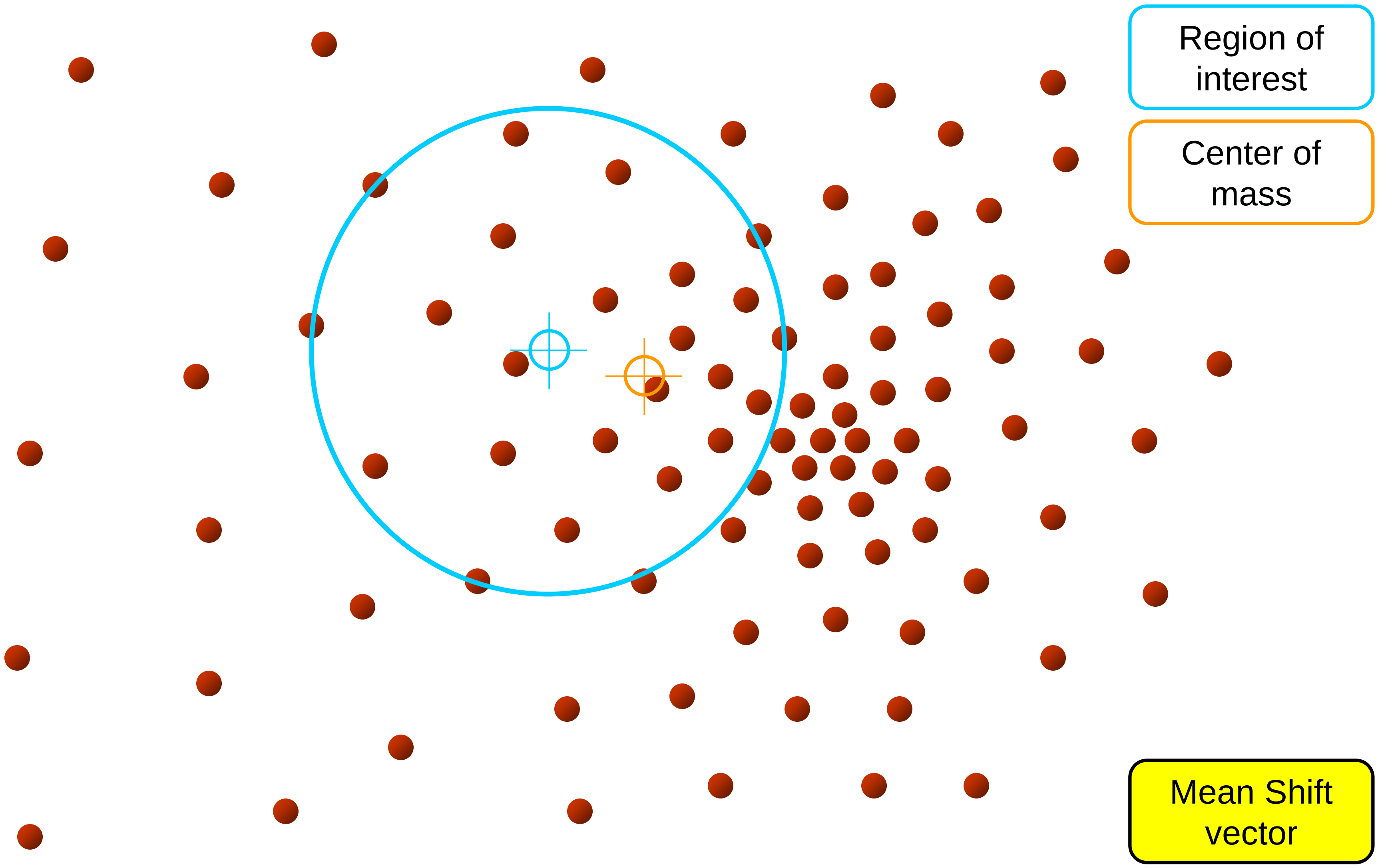


Objective : Find the densest region

Distribution of identical billiard balls

(equivalent to our problem, where sampling is regular and a large w_i corresponds to a larger density of balls).

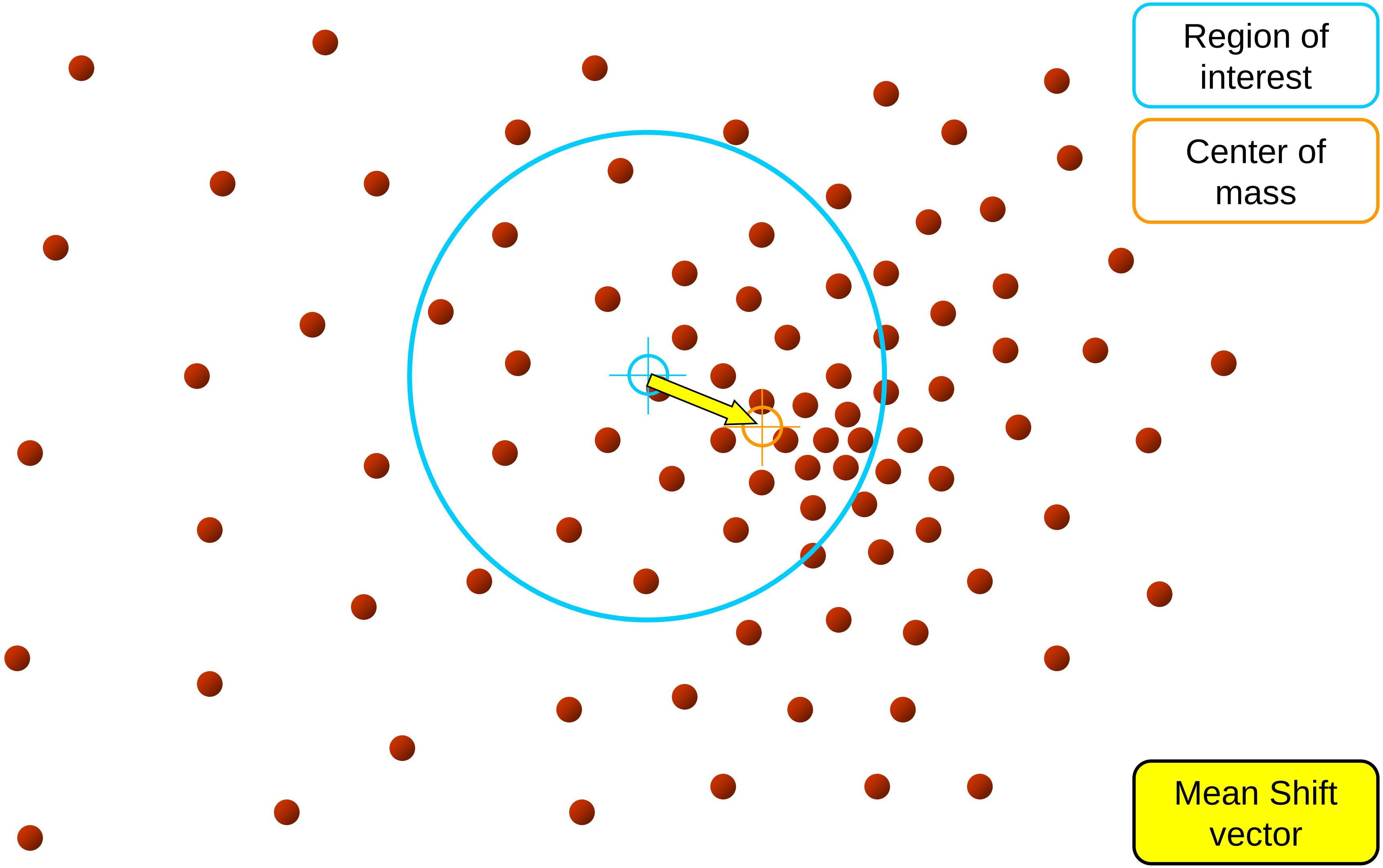
Intuitive Description



<u>Objective</u> : Find the densest region

Distribution of identical billiard balls

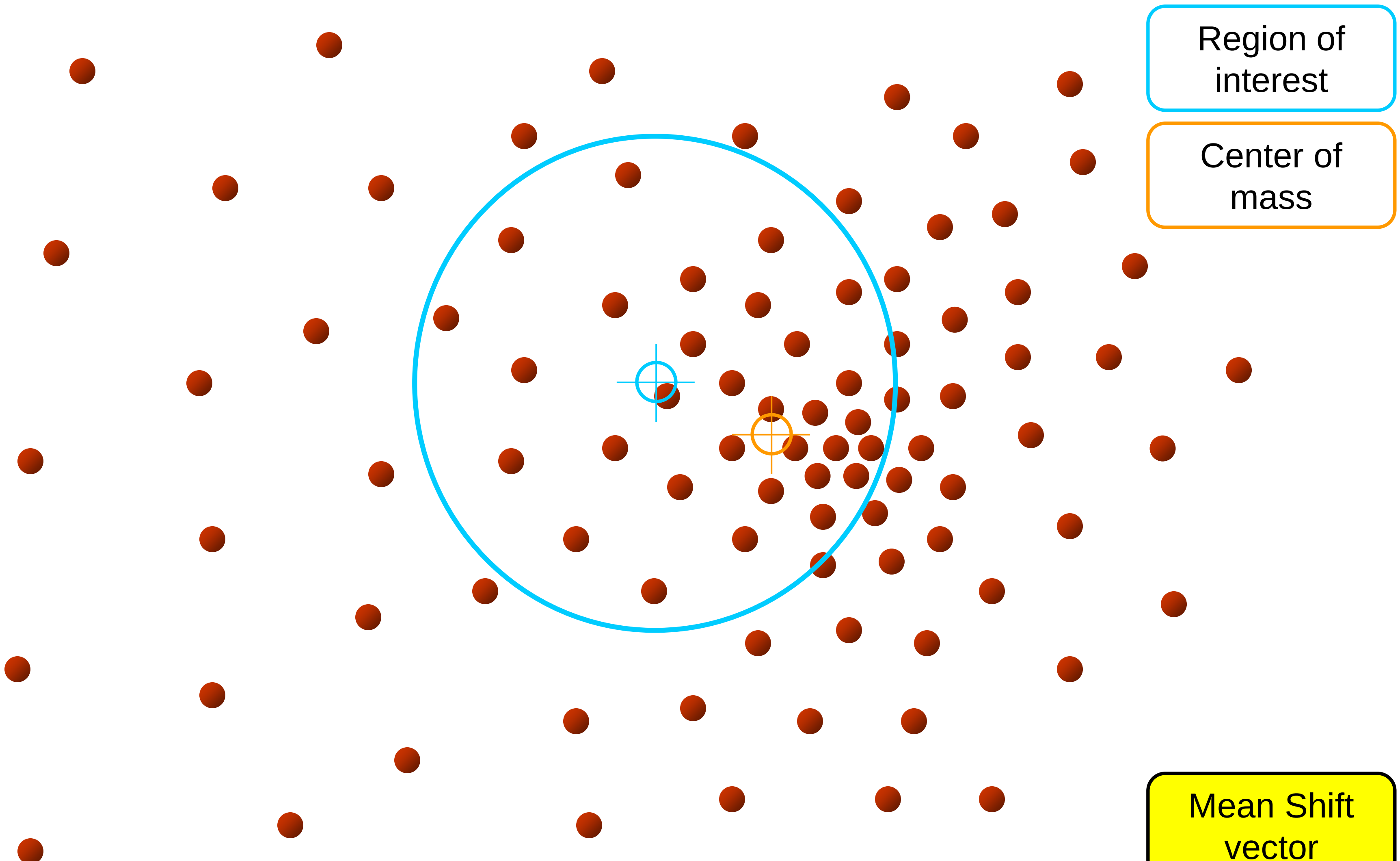
Intuitive Description



Objective : Find the densest region

Distribution of identical billiard balls

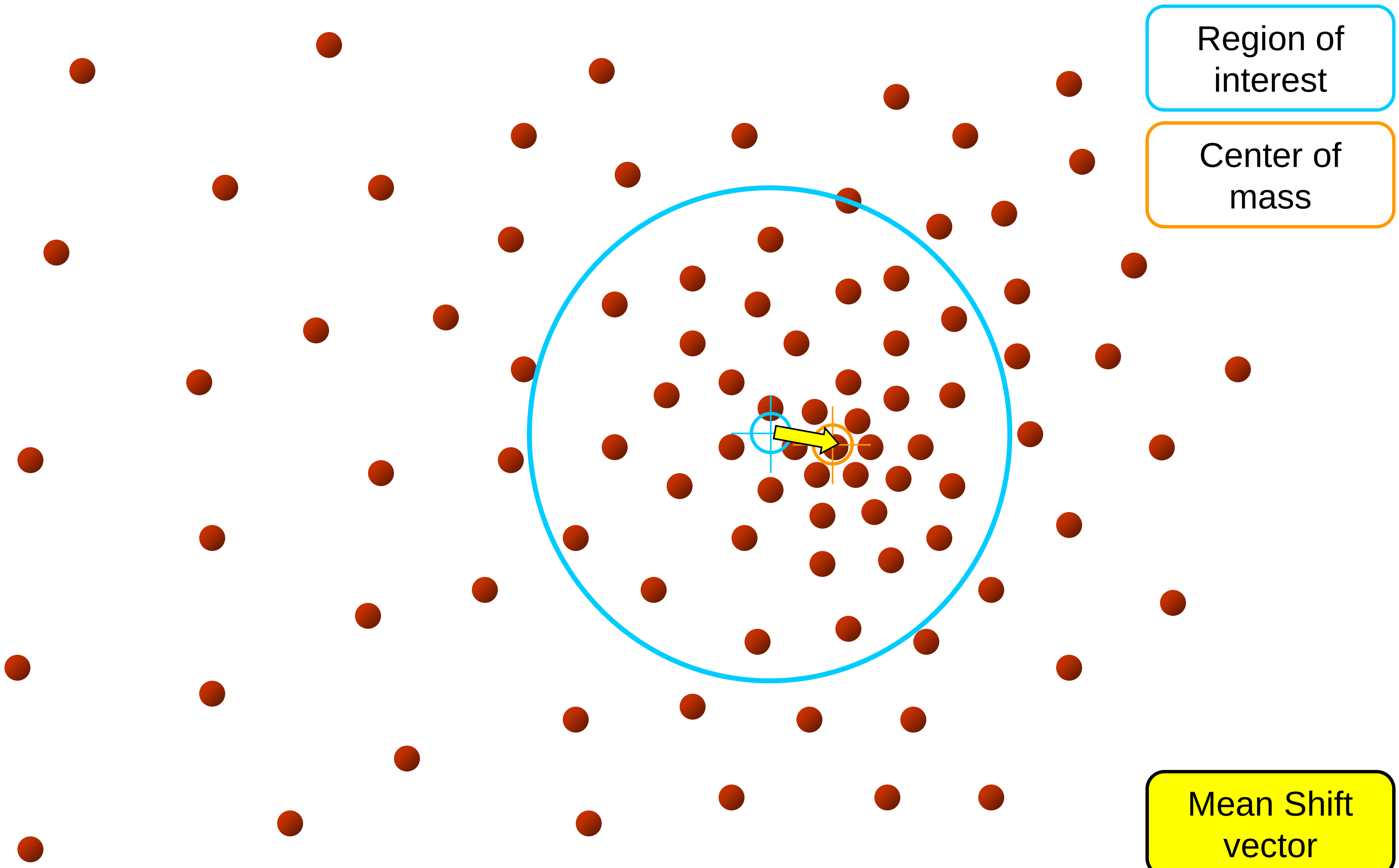
Intuitive Description



Objective : Find the densest region

Distribution of identical billiard balls

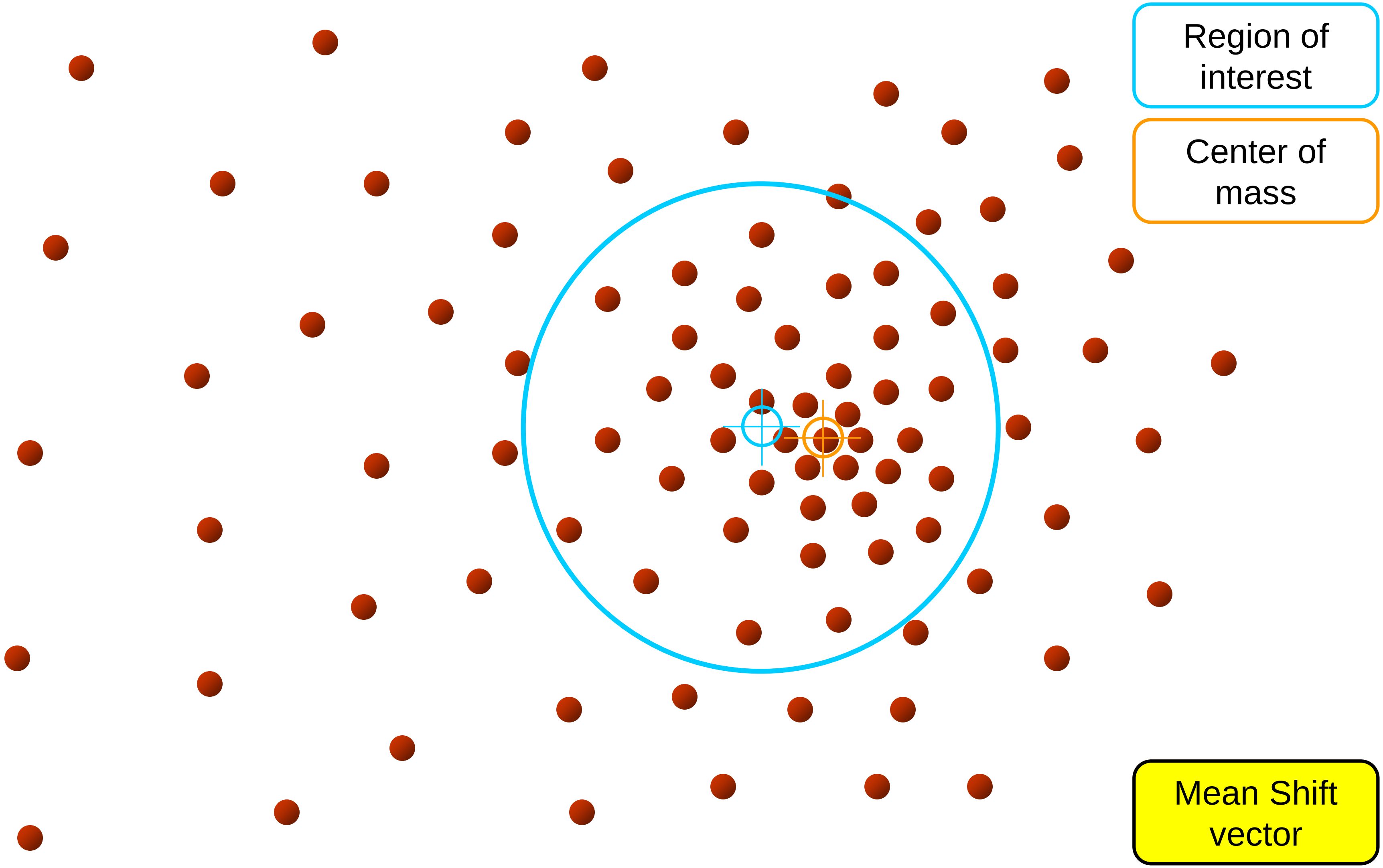
Intuitive Description



<u>Objective</u> : Find the densest region

Distribution of identical billiard balls

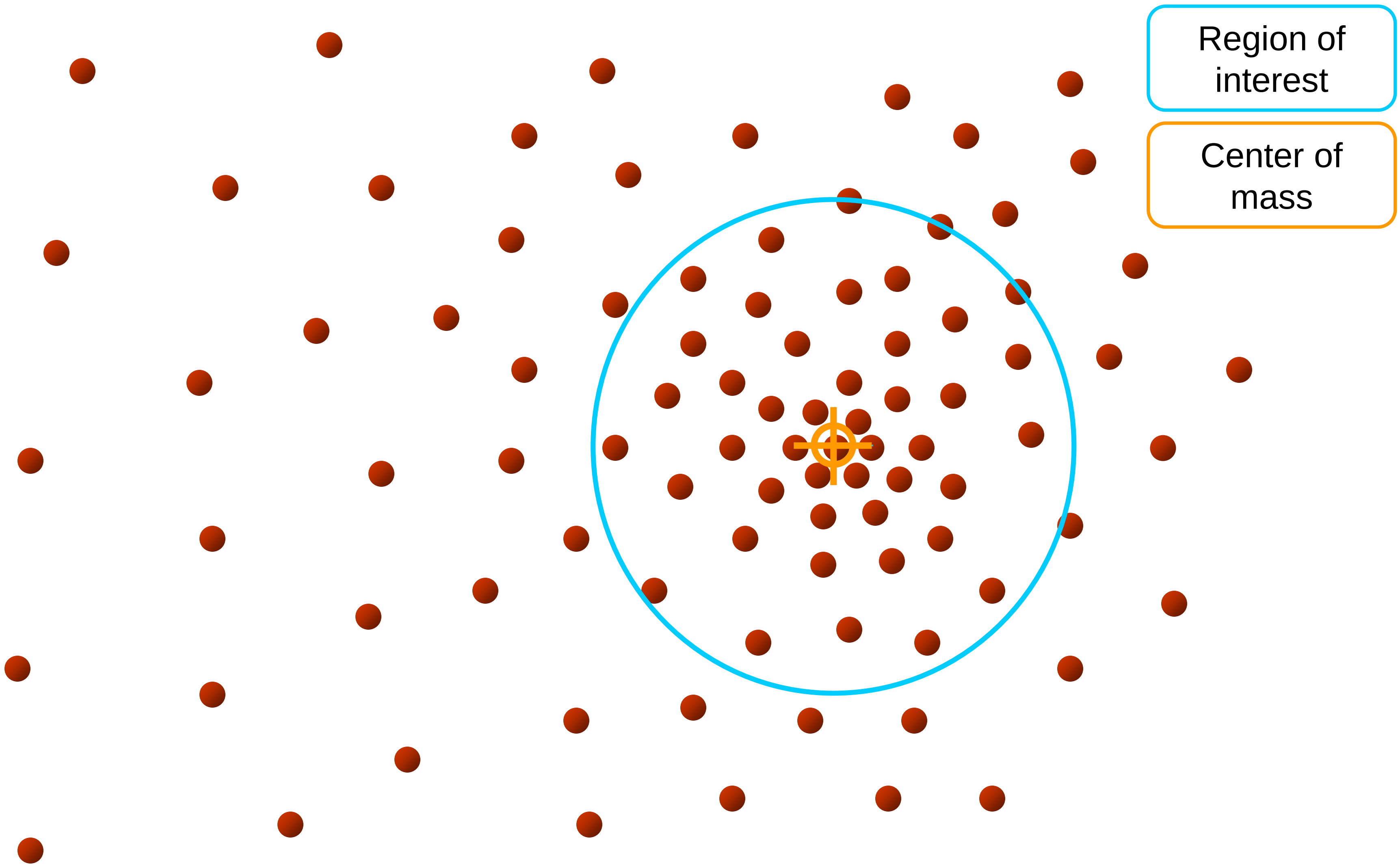
Intuitive Description



<u>Objective</u> : Find the densest region

Distribution of identical billiard balls

Intuitive Description



Objective : Find the densest region

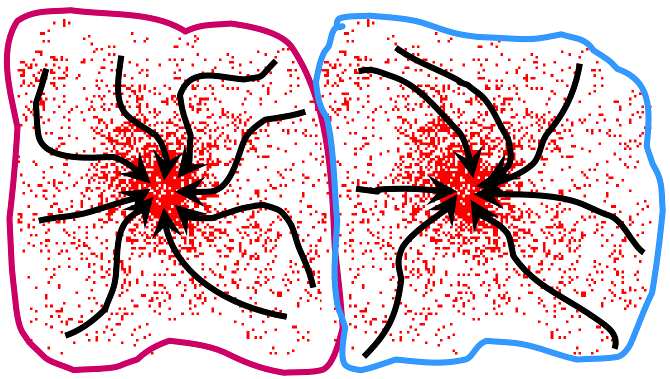
Distribution of identical billiard balls

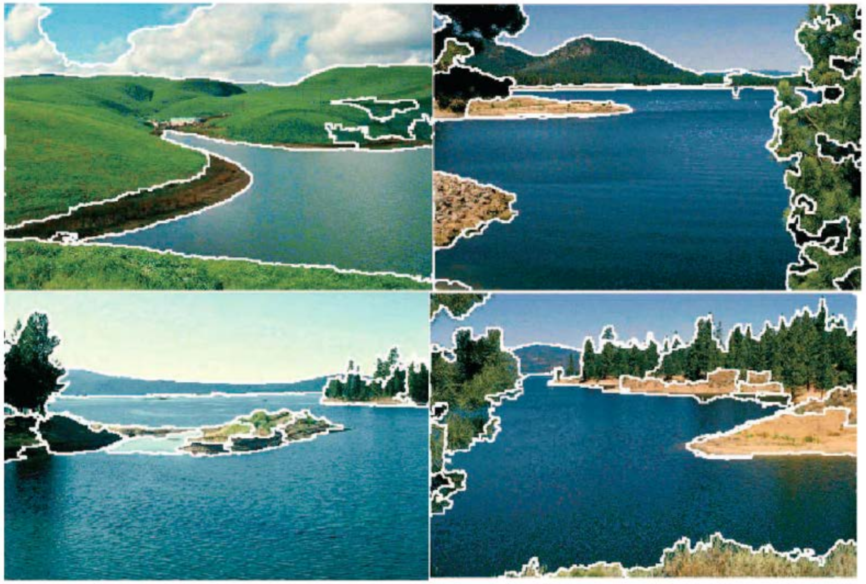
Examples of mean-shift applications

Clustering & segmentation

<u>Cluster</u> : All data points in the ***attraction basin*** of a mode

<u>Attraction basin</u> : the region for which all trajectories lead to the same mode





Segmentations of images obtained using the mean shift algorithm. *This figure was originally published as Figure 10 of "Mean Shift: A Robust Approach Toward Feature Space Analysis," by D. Comaniciu and P. Meer, IEEE Transactions on Pattern Analysis and Machine Intelligence, 2002 © IEEE, 2002.*

Mean Shift : A robust Approach Toward Feature Space Analysis, by Comaniciu, Meer

Tracking

- Introduction
- Template matching: Lucas-Kanade method
- Kernel-based tracking: Mean-shift
- **Bayesian recursive estimation: Kalman and particle filters**

Motivation

- Traditional approaches only remember the best solution at each step of the recursive tracking mechanism.
  - Risk to propagate the bad decision to subsequent frames.

- Better approach: estimate target state probability distribution !

Bayesian recursive estimation is implemented with particle filters:

  - Naturally and elegantly keep track of multiple hypotheses about target location.
  - Particles with small probabilities tend to disappear, but will however survive for some frames, thereby exploring alternative hypotheses.
  - Particles with high probabilities define the tracking decision.

Bayesian recursive framework

- State-space approach:
  - State variable $\mathcal{X}_k$: e.g. target position and velocity in state-space at time k

    $$\mathcal{X}_k = [x, y, z, \dot{x}, \dot{y}, \dot{z}]^T$$

  - Observation $\mathcal{Y}_k$: measurements obtained from processing camera image data
  - Set of all observations: $\mathcal{Y}_{1:k} = [\mathcal{Y}_1, \ldots, \mathcal{Y}_k]$
  - System dynamics (transition) equation: $\mathcal{X}_k = g(\mathcal{X}_{k-1}, v_{k-1})$

    \+ **Observation model** : $Y_k = h(X_k, n_k)$

Note: Markov process.

Aim: given all data $\mathcal{Y}_{1:k}$, compute posterior PDF $p(\mathcal{X}_k|\mathcal{Y}_{1:k})$

⇨ Bayesian filtering problem

- **Bayesian filtering solution**: if posterior PDF $p(\mathcal{X}_{k-1}|\mathcal{Y}_{1:k-1})$ known at time $k-1$, compute current posterior PDF as follows:

Marginalization of the joint conditional probability.

$$\text{Predict: } p(\mathcal{X}_k|\mathcal{Y}_{1:k-1}) = \int p(\mathcal{X}_k|\mathcal{X}_{k-1})\, p(\mathcal{X}_{k-1}|\mathcal{Y}_{1:k-1})\, \mathrm{d}\mathcal{X}_{k-1}$$

$$\text{Update: } p(\mathcal{X}_k|\mathcal{Y}_{1:k}) \propto p(\mathcal{Y}_k|\mathcal{X}_k)\, p(\mathcal{X}_k|\mathcal{Y}_{1:k-1})$$

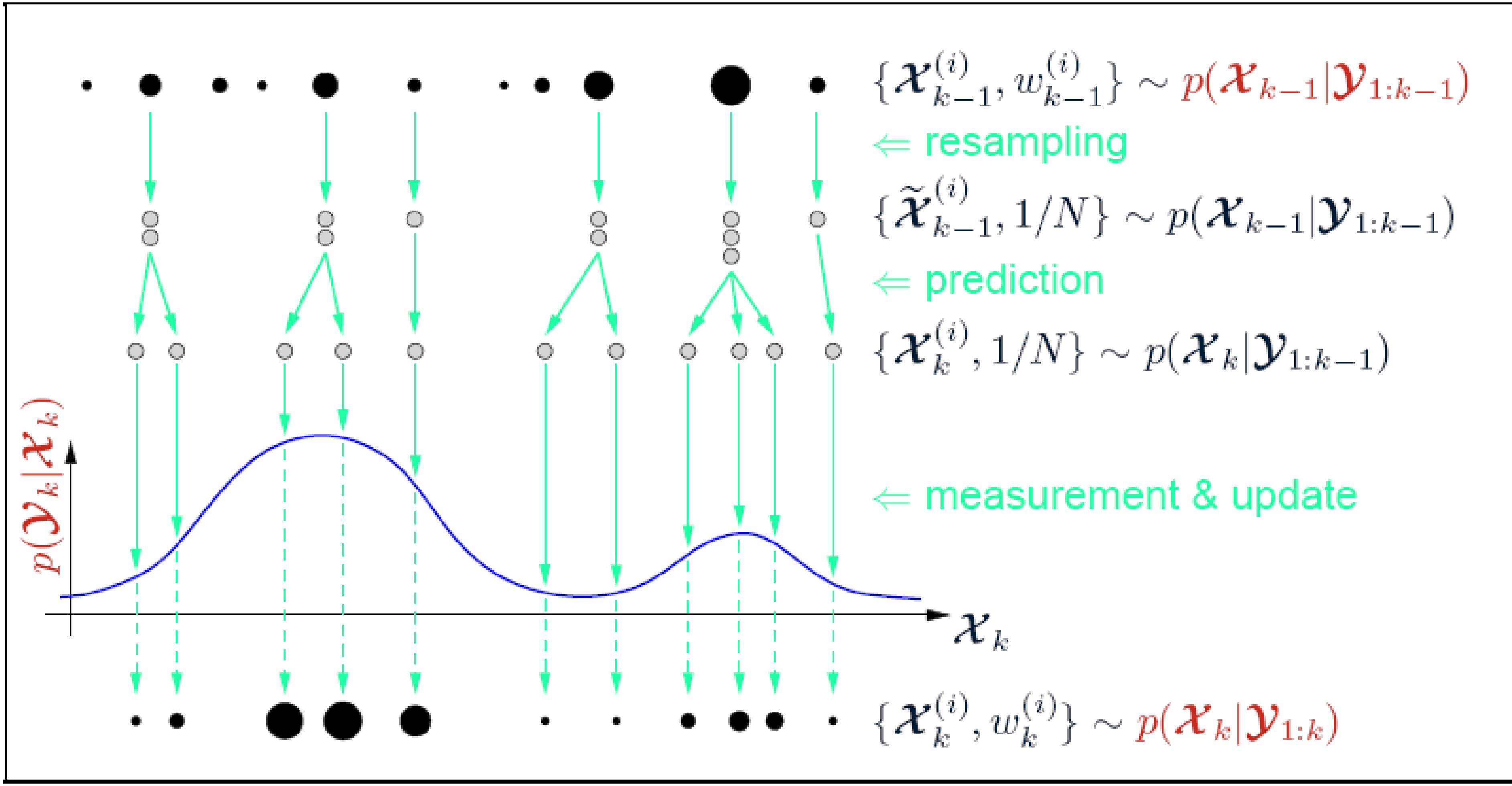
Note: observations are independent, conditionally on the state.

where $p(\mathcal{Y}_k|\mathcal{X}_k)$ is the likelihood function (measurement PDF)

- **Problem**: usually no closed-form solutions available for many natural dynamic models

- **Current approximations**: Kalman filter, extended Kalman filter, Gaussian sum methods, grid-based methods, etc.
  ⇨ Sequential Monte Carlo methods, i.e. Particle Filters (PF)

BRF: Symbolic representation



More about particle filters

- See the Appendix or reading list.

  Thanks to Jacek Czyz !

Appendix: Particle filters

Overview

1. Bayesian Recursive Estimation
   - State-space models
   - Bayesian Recursive estimation/filtering (BRF)
   - Particle filter solution to BRF

2. Example: Tracking with particle filters

3. Summary

Estimation

Estimation theory deals with estimating the values of parameters/quantities based on some measured data.

In estimation theory, it is assumed that the desired information is embedded into a noisy signal. Noise adds uncertainty and if there was no uncertainty then there would be no need for estimation.

In **Recursive Estimation**, measured data arrive sequentially and it is assumed that the unknown quantity is dynamic and follows a evolution model.

Bayesian Recursive Est. : the state-space model

To make the estimation of the unknown parameters , two models are required: a **dynamic model** (evolution of the unknown system state) and an **observation model** (relates the parameters to the measurements)

- State vector $\mathbf{x}_t$ contains (internal) info about the system at time t
- Observation vector $\mathbf{z}_t$ measured at time t
- State evolution $\mathbf{x}_t = \mathbf{f}_t(\mathbf{x}_{t-1}) + \mathbf{w}_{t-1}$
- Observation/measurement model: relates the noisy measurement to the state $\mathbf{z}_t = \mathbf{h}_t(\mathbf{x}_t) + \mathbf{n}_t$
- state evolution is stochastic: $\Leftrightarrow p(\mathbf{x}_t|\mathbf{x}_{t-1})$
- observ. model is stochastic: $\Leftrightarrow p(\mathbf{z}_t|\mathbf{x}_t)$

Bayesian Rec. Estimation : the state-space model II

Our purpose

- Find $p(\mathbf{x}_t|\mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_t) = p(\mathbf{x}_t|\mathbf{z}_{1:t})$
- Recursively: Given $p(\mathbf{x}_t|\mathbf{z}_{1:t})$, find $p(\mathbf{x}_{t+1}|\mathbf{z}_{1:t+1})$
- $p(\mathbf{x}_t|\mathbf{z}_{1:t})$ provides estimates of state and accuracy.
  e.g.

$$\bar{\mathbf{x}}_t = \int \mathbf{x}_t p(\mathbf{x}_t|\mathbf{z}_{1:t}) d\mathbf{x}_t$$

Bayesian Recursive Estimation

Under hypotheses that observations are mutually *conditionally* independent, and $\mathbf{x}_t$ is a markov process, solution to our problem is given by
(see Isard and Blake, CONDENSATION - conditional density propagation for visual tracking, 1998)

- Prediction step:

$$p(\mathbf{x}_{t+1}|\mathbf{z}_{1:t}) = \int p(\mathbf{x}_{t+1}|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{z}_{1:t})d\mathbf{x}_t$$

- Update step: measurement $\mathbf{z}_{t+1}$ becomes available

$$p(\mathbf{x}_{t+1}|\mathbf{z}_{1:t+1}) = k_t p(\mathbf{z}_{t+1}|\mathbf{x}_{t+1})p(\mathbf{x}_{t+1}|\mathbf{z}_{1:t})$$

The repetition of the two steps propagates $p(\mathbf{x}_t|\mathbf{z}_{1:t})$ in time

Bayesian Recursive Filtering: Kalman Filter

When state and observation models are linear and process and observation noises are Gaussian, the recursive solution to the BRF is the Kalman filter.

$$\mathbf{x}_t = F_t \mathbf{x}_{t-1} + \mathbf{w}_{t-1}$$

$$\mathbf{z}_t = H_t \mathbf{x}_t + \mathbf{n}_t$$

where $\mathbf{w}_t \sim \mathcal{N}(0, R_t)$ and $\mathbf{n}_t \sim \mathcal{N}(0, Q_t)$.
See Welch and Bishop, An Introduction to the Kalman Filter, 2006.

The recursive Kalman solution is optimal if assumptions hold.
In the non-linear/non-Gaussian case, only sub-optimal solutions exists.

Particle filters

- Particle filters give an approx solution to BRF
- $p(\mathbf{x}_t|\mathbf{z}_{1:t})$ is represented by N weighted particles

$$p(\mathbf{x}_t|\mathbf{z}_{1:t}) \approx \sum_i w_t^{(i)} \delta(\mathbf{x}_t - \mathbf{x}_t^{(i)})$$

- each particle/sample simulates a trajectory in the state-space.
- trajectories that explain the data well are kept. The others are discarded.

Having a set of samples $\{\mathbf{x}_{t-1}^{(i)}, w_{t-1}^{(i)}\} \sim p(\mathbf{x}_{t-1}|\mathbf{z}_{1:t-1})$, look for a new set $\{\mathbf{x}_t^{(i)}, w_t^{(i)}\} \sim p(\mathbf{x}_t|\mathbf{z}_{1:t})$,

Prediction

Simulate the prediction step

- To sample $\{\mathbf{x}_{t-1}^{(i)}, w_{t-1}^{(i)}\}$, apply the state evolution step i.e. draw samples from $p(\mathbf{x}_t|\mathbf{x}_{t-1}^{(i)})$
  - Deterministic drift coming from $\mathbf{f}_t()$ – same for all particles
  - Stochastic drift coming from $\mathbf{w}_{t-1}$ – different for each particles

- The new samples $\{\mathbf{x}_t^{(i)}, w_{t-1}^{(i)}\}$ follow the pdf $p(\mathbf{x}_t|\mathbf{z}_{1:t-1})$

Update

Weight the particles according to the obs model. Particles that explain well the observation receive a large weight.

Update step

$$p(\mathbf{x}_t|\mathbf{z}_{1:t}) \propto p(\mathbf{z}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{z}_{1:t-1})$$

$$\{\mathbf{x}_t^{(i)}, w_{t-1}^{(i)}\} \sim p(\mathbf{x}_t|\mathbf{z}_{1:t-1})$$

$$\{\mathbf{x}_t^{(i)}, w_t^{(i)}\} \sim p(\mathbf{x}_t|\mathbf{z}_{1:t})$$

$$w_t^{(i)} \propto w_{t-1}^{(i)} . p(\mathbf{z}_t|\mathbf{x}_t^{(i)})$$

Resampling

Degeneracy Problem: after a few iterations, all but one particle have negligible weight.

Solution : resample the particle set at each time step to

- obtain uniform weights, so that $w_{t-1}^{(i)} = \frac{1}{N} \; \forall i$
- duplicate many times particles with high weight (put comp. power on promising trajectories/hypotheses)
- delete particles with negligible weights (avoid unnecessary computations on unlikely trajectories)

Overview

1. Bayesian tracking
   - State-space models
   - Bayesian Recursive estimation/filtering (BRF)
   - Particle solution to BRF

2. Example : tracking with particle filters

3. Summary

Example: tracking

Tracking: estimate the successive positions of an object given a sequence of images.

It is a recursive estimation: position is dynamic – current position depends on previous position

Typically, the quantity to estimate: $\mathbf{x} = (x, y, \dot{x}, \dot{y}, H_x, H_y)^T$

We assume successive observations are independent (conditionnally on the state)

State equation

- State vector : $\mathbf{x} = (x, y, \dot{x}, \dot{y}, H_x, H_y)^T$
- Linear state model for $\mathbf{x}_t$ (time step t) (1st order lin. model)

$$\mathbf{x}_t = A\mathbf{x}_{t-1} + \mathbf{w}_{t-1}$$

where A is (6 x 6) matrix, $\mathbf{w}_{t-1}$ Gaussian 6-dim. noise vector.

$$\begin{pmatrix} x_t \\ y_t \\ \dot{x}_t \\ \dot{y}_t \\ H_{xt} \\ H_{yt} \end{pmatrix} = \begin{pmatrix} 1 & 0 & \Delta t & 0 & 0 & 0 \\ 0 & 1 & 0 & \Delta t & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_{t-1} \\ y_{t-1} \\ \dot{x}_{t-1} \\ \dot{y}_{t-1} \\ H_{xt-1} \\ H_{yt-1} \end{pmatrix} + \begin{pmatrix} w_{1t-1} \\ w_{2t-1} \\ w_{3t-1} \\ w_{4t-1} \\ w_{5t-1} \\ w_{6t-1} \end{pmatrix}$$

Measurement equation

Measurement equation/likelihood relates the state to the observation.

$$\mathbf{z}_t = \mathbf{h}_t(\mathbf{x}_t) + \mathbf{n}_t \Leftrightarrow p(\mathbf{z}_t|\mathbf{x}_t)$$

For PF we need to specify $p(\mathbf{z}_t|\mathbf{x}_t)$

e.g.

$$p(\mathbf{z}_t|\mathbf{x}_t) \propto \exp\left(-\alpha\|\mathbf{g}_{\mathbf{x}_t}(\mathbf{z}_t) - \mathbf{t}_t\|^2\right)$$

where $\mathbf{t}_t$ is a template of the object to track.

$\mathbf{g}_{\mathbf{x}_t}(\mathbf{z}_t)$ is a feature extraction function applied on $\mathbf{z}_t$ at the location specified by the state vector (e.g. color histogram, filter response, etc.)

α is a design parameter

Examples of feature extraction function that can be used for tracking

- $\mathbf{g}_{\mathbf{x}_t}$ extracts color histograms from $\mathbf{z}_t$ in region described by $\mathbf{x}_t$
  Demonstration in 'football' and 'coca' sequences.

- $\mathbf{g}_{\mathbf{x}_t}$ computes learned filter responses for face detection.
  Demonstration in 'face' sequences.

PF for tracking

Summary: One step of the PF algorithm

- From
  1. the particles from previous frame $\mathbf{x}_{t-1}^{(i)}$
  2. System equation $\mathbf{x}_t = A\mathbf{x}_{t-1} + \mathbf{w}_{t-1}$

  Predict next positions of particles $\mathbf{x}_t^{(i)}$

  Note: need to draw from $\mathbf{w}_{t-1}$ otherwise particles evolve identically

- Then Weight the particle $\mathbf{x}_t^{(i)}$ with $w_t^{(i)} \propto \exp\left(\alpha \|\mathbf{g}_{\mathbf{x}}(\mathbf{z}_t) - \mathbf{t}\|^2\right)$

- Estimate the mean state $E[\mathbf{x}_t] = \sum_i w_t^{(i)} \mathbf{x}_t^{(i)}$

- Resample: particles with big weight generate many particles (Prepare N particles for next frame).

Particle filter: issues

PF's give a technique for solving the BRF problem.
The difficult part is to define the BRF problem, i.e.

- Determine what should contain the state $\mathbf{x}$
- Determine the dynamics of $\mathbf{x}_t$, i.e. determine (learn) $p(\mathbf{x}_t|\mathbf{x}_{t-1})$
- Detemine the observation model i.e. determine (learn) $p(\mathbf{z}_t|\mathbf{x}_t)$

Particle filter: Conclusion

- PF's offer a general framework for solving Bayesian Recursive Estimation in the state-space formalism (Kalman).
- PF's estimate $p(\mathbf{x}_t|\mathbf{z}_{1:t})$
- PF's represent $p(\mathbf{x}|\mathbf{z}_{1:t})$ as a set of particles $\mathbf{x}^{(i)}$
- The problem is solved in three steps
  - Prediction: use the state equation to compute state vector one time step ahead wrt current position
  - Update: give more weight to $\mathbf{x}^{(i)}$'s that correspond to observation model $p(\mathbf{z}_t|\mathbf{x}_t)$ and current observation
  - Resample to get equal weights
- iterate the three steps as more observations become available