Mean Shift

Theory and Applications

Slide credit: Yaron Ukrainitz & Bernard Sarel

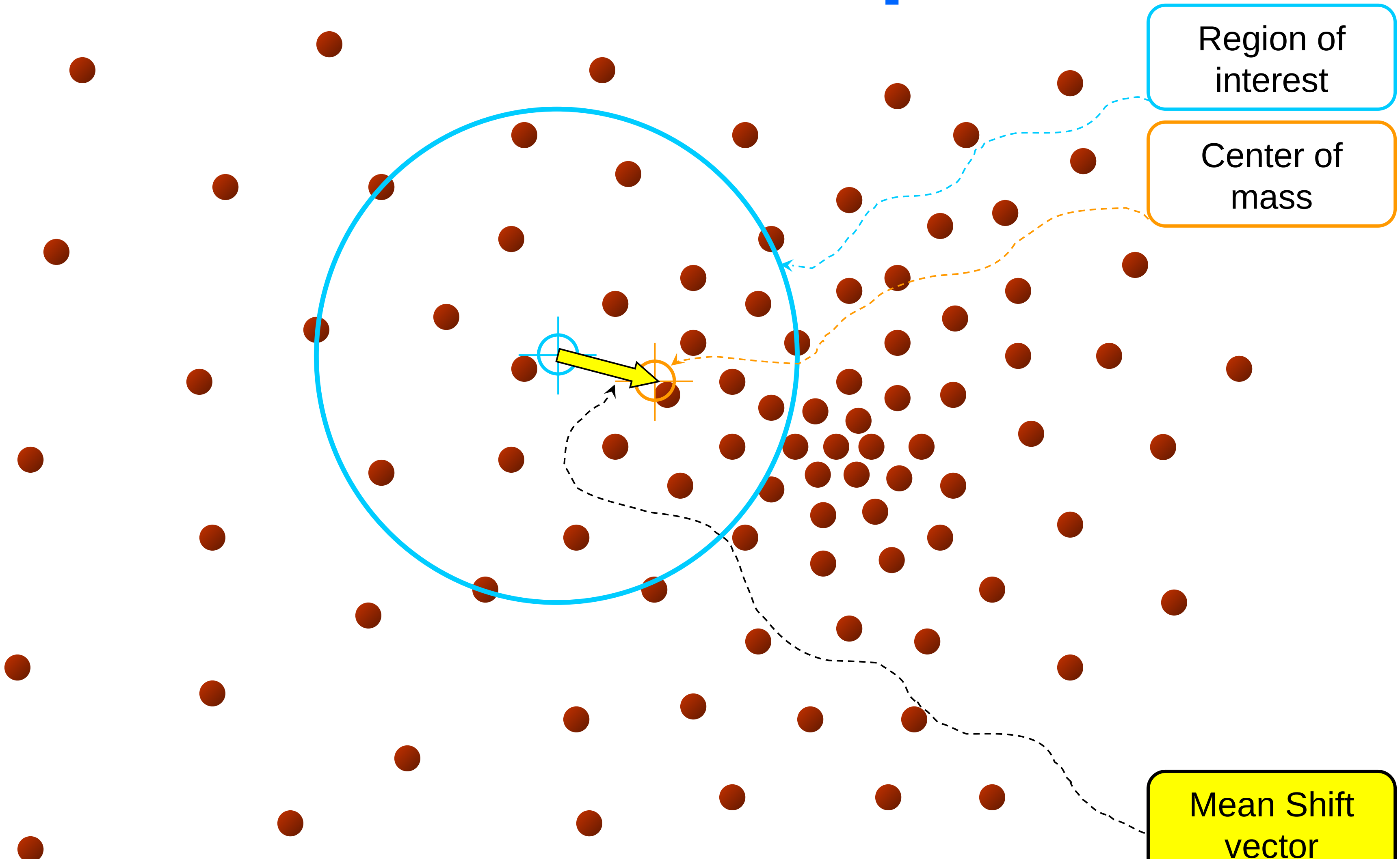
Agenda

- **Mean Shift Theory**
  - What is Mean Shift ?
  - Density Estimation Methods
  - Deriving the Mean Shift
  - Mean shift properties

- **Applications**
  - Clustering
  - Discontinuity Preserving Smoothing
  - Object Contour Detection
  - Segmentation
  - Object Tracking

Mean Shift Theory

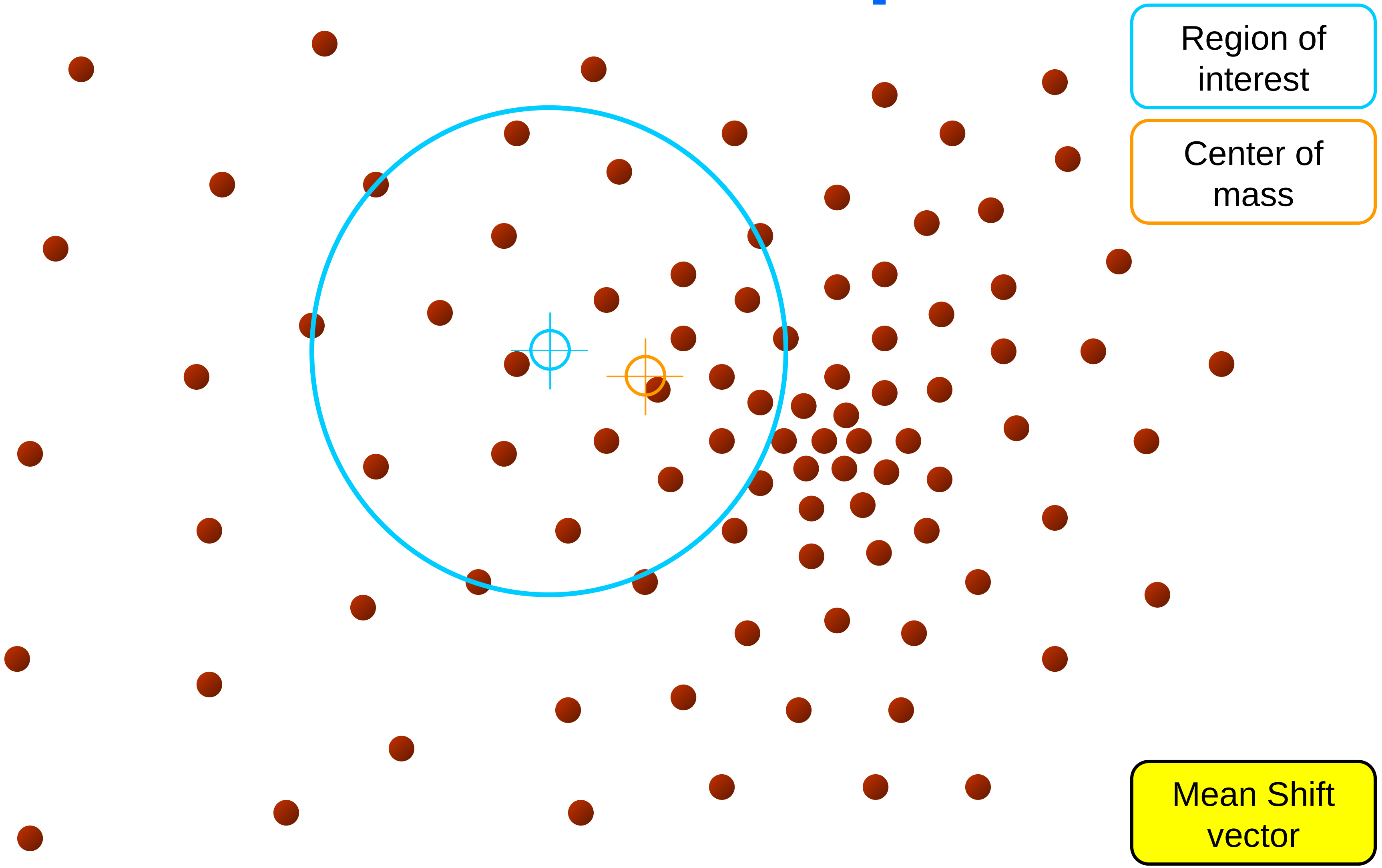
Intuitive Description



Objective : **Find the densest region**

Distribution of identical billiard balls

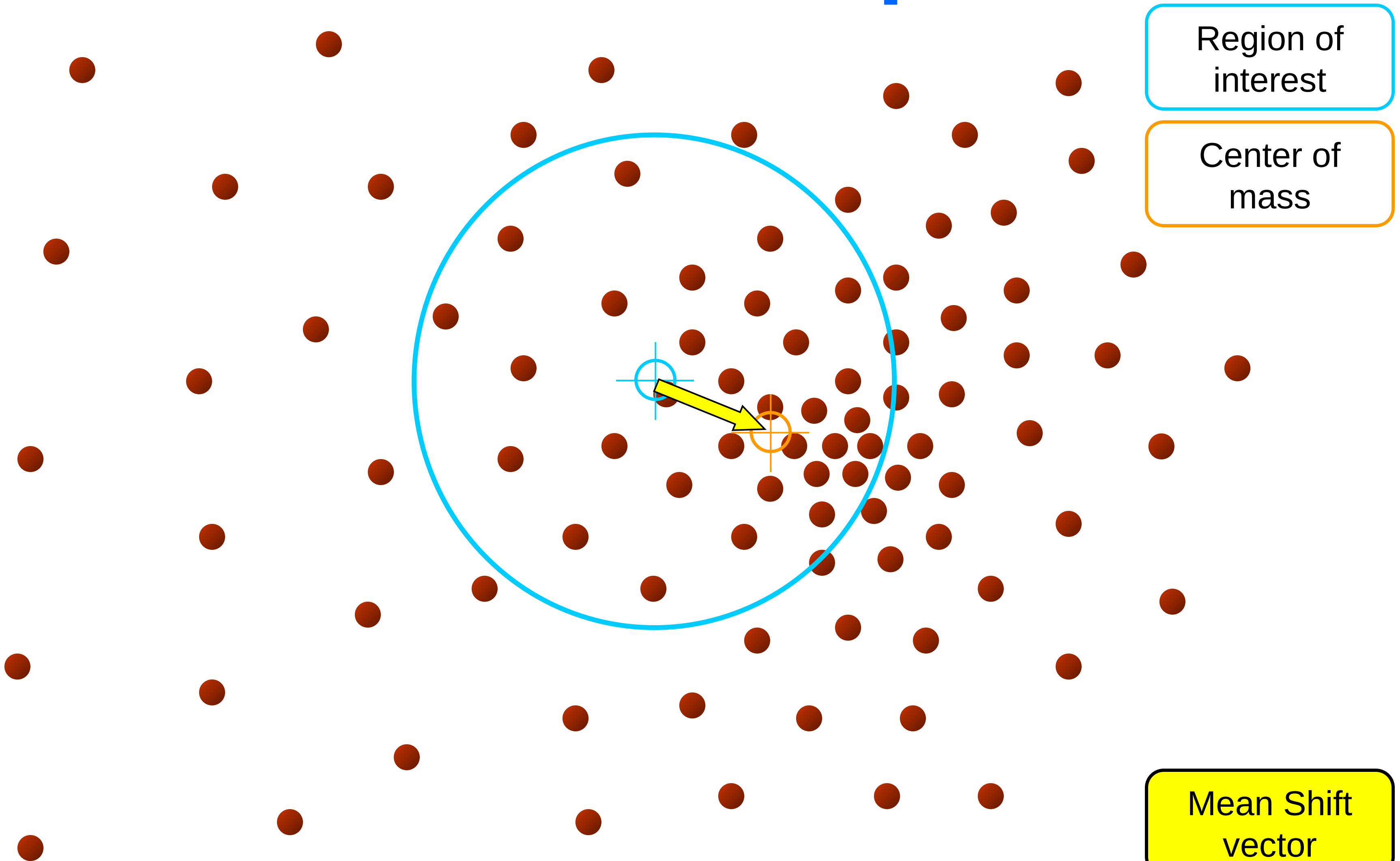
Intuitive Description



Objective : Find the densest region

Distribution of identical billiard balls

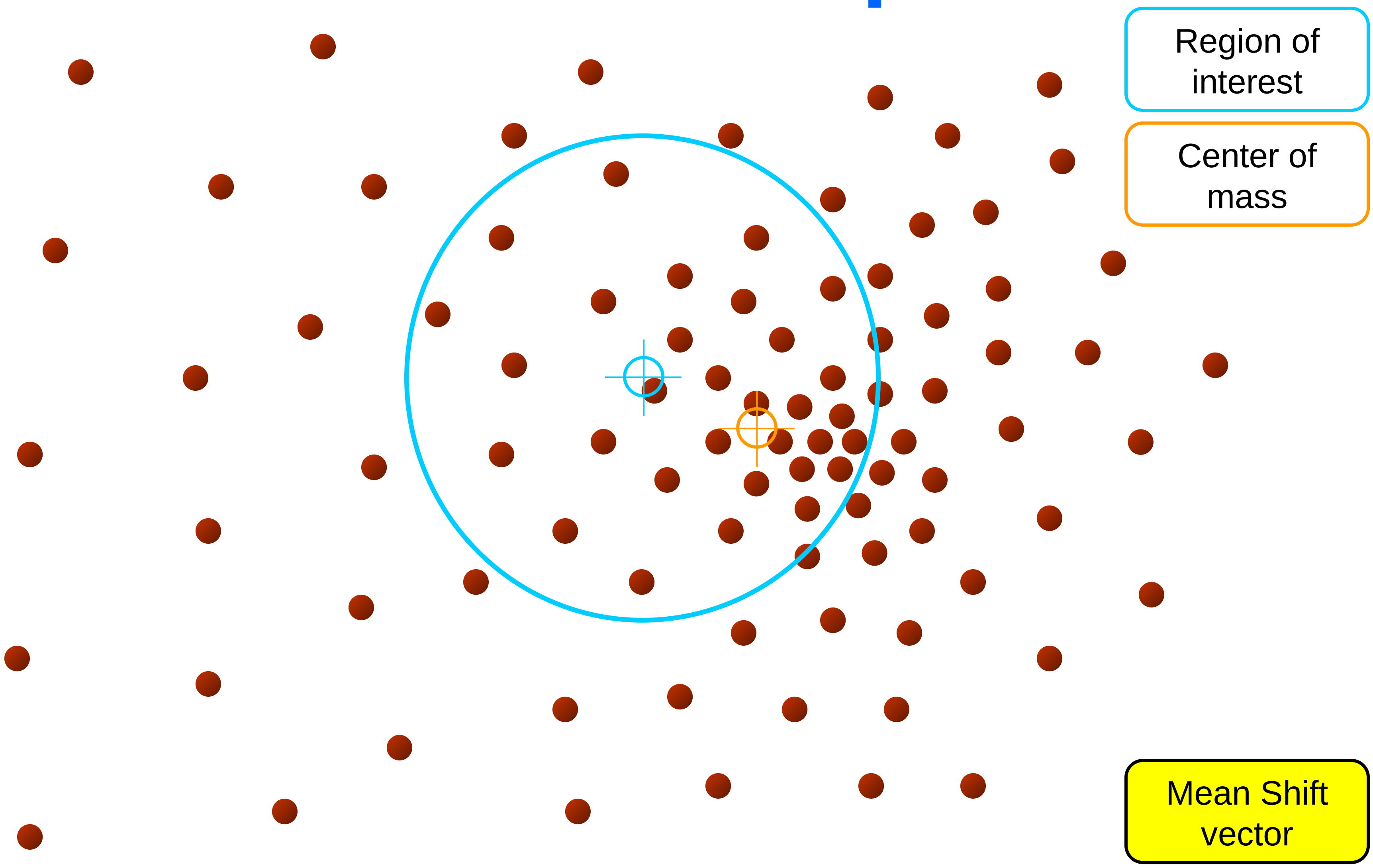
Intuitive Description



Objective : Find the densest region

Distribution of identical billiard balls

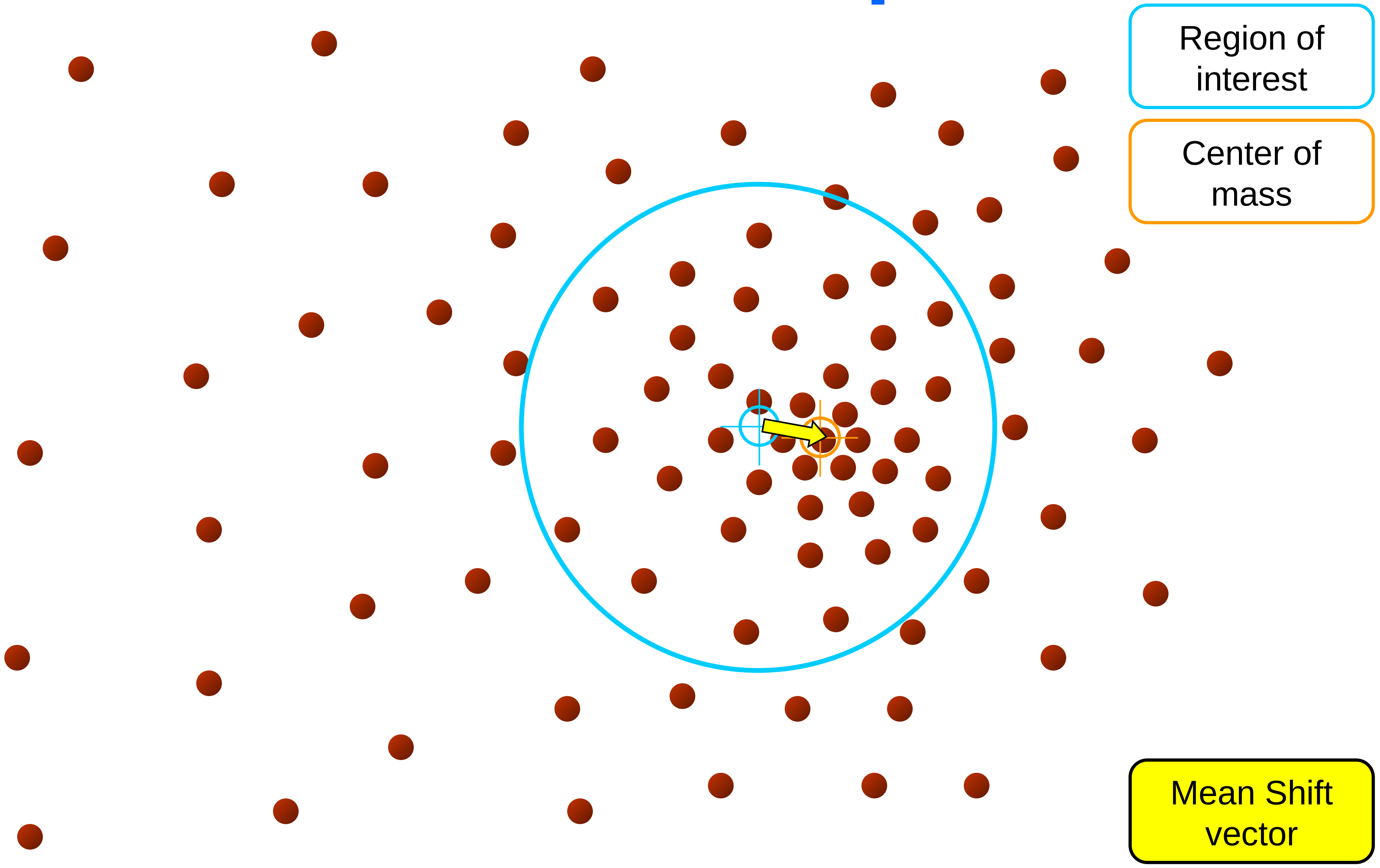
Intuitive Description



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

Distribution of identical billiard balls

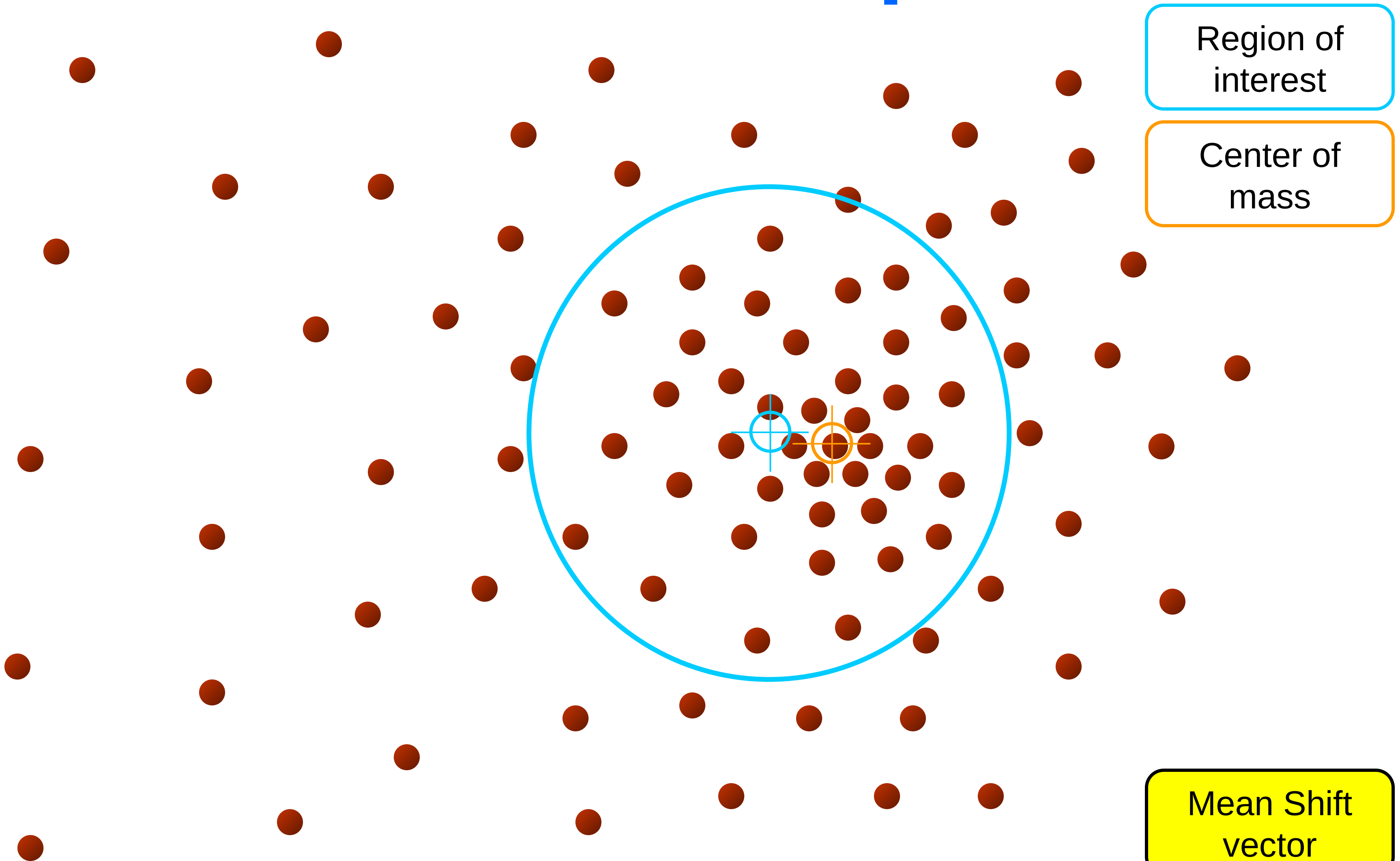
Intuitive Description



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

Distribution of identical billiard balls

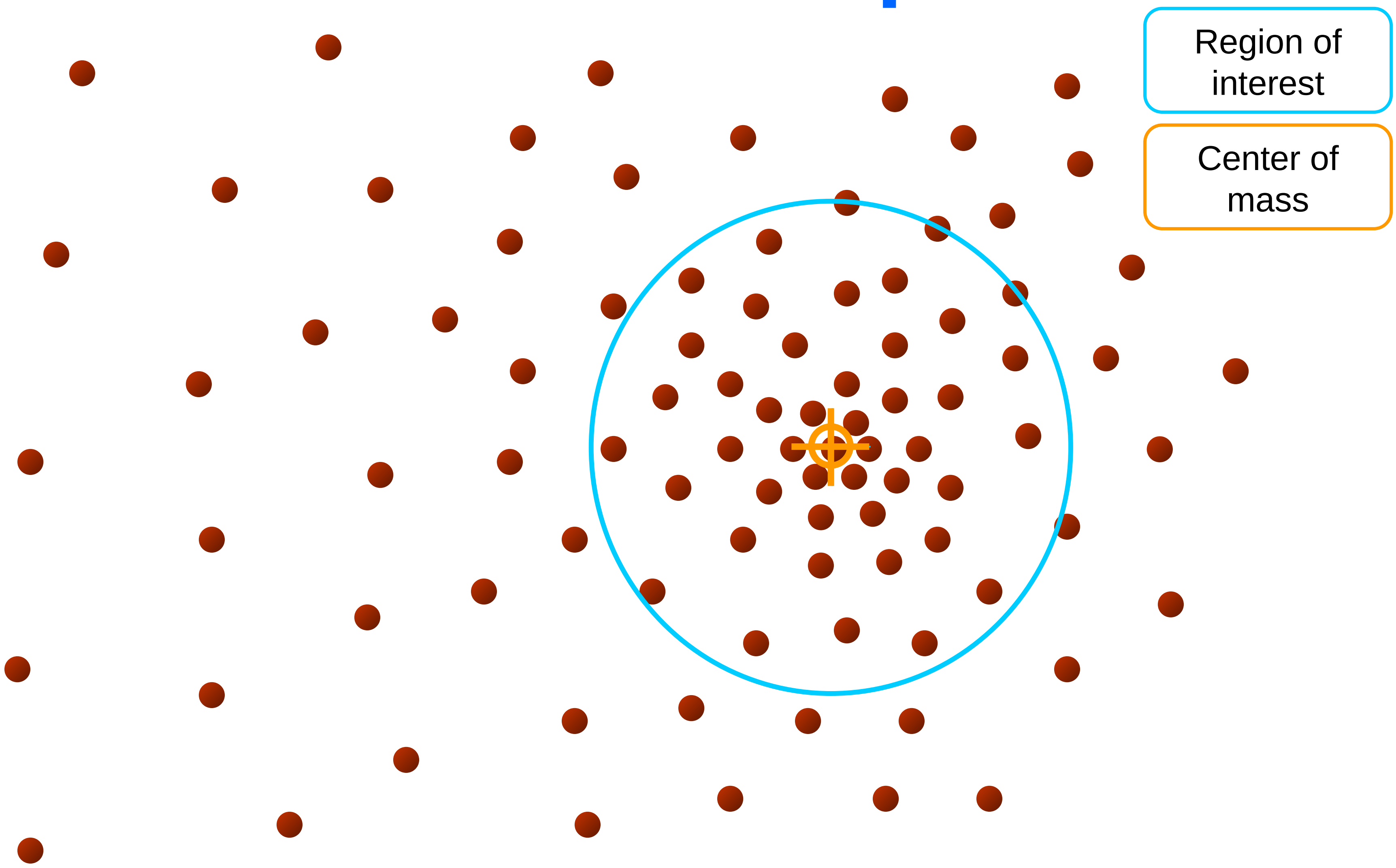
Intuitive Description



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

Distribution of identical billiard balls

Intuitive Description



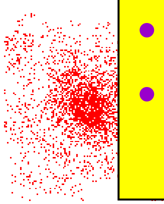
<u>Objective</u> : Find the densest region

Distribution of identical billiard balls

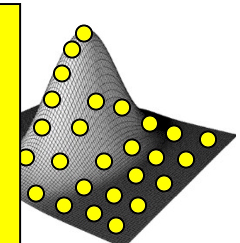
What is Mean Shift ?

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

<u>PDF in feature space</u>
- Color space
- Scale space
- Actually any feature space you can conceive
- …

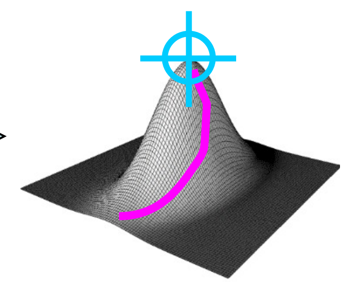


Data



DF Representation

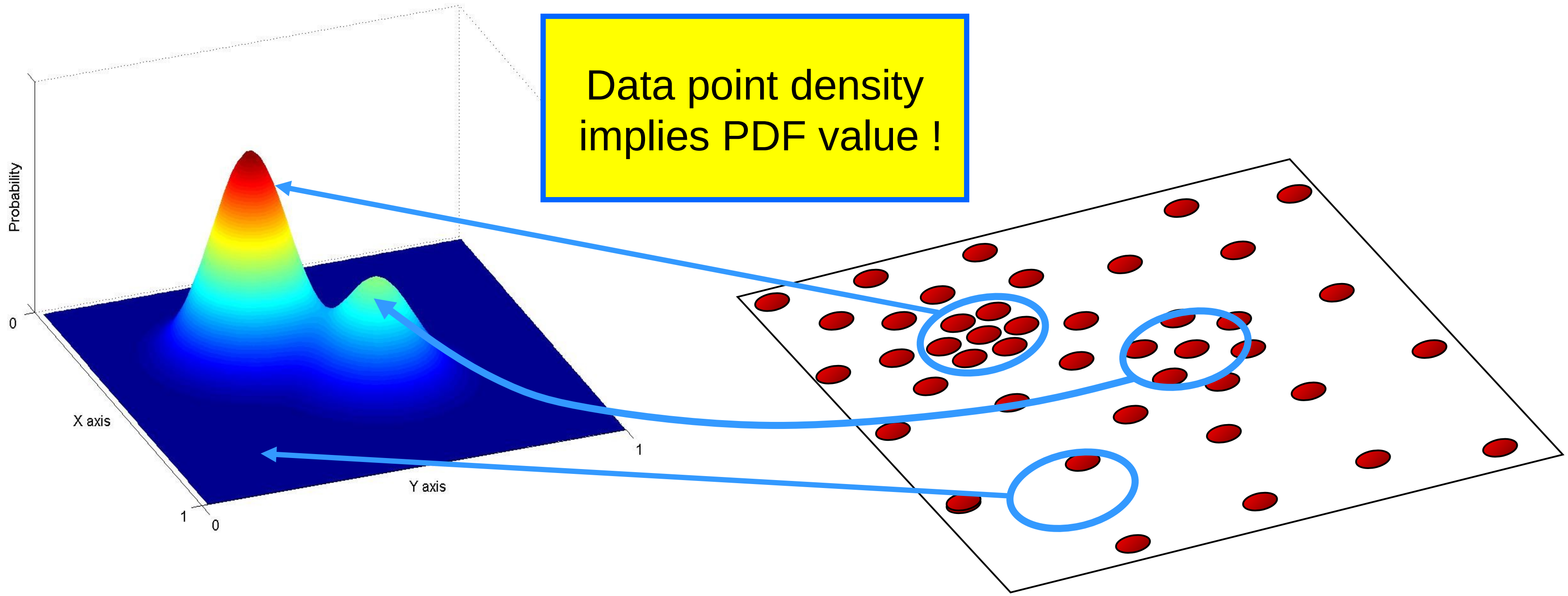
Non-parametric
Density **GRADIENT** Estimation
(Mean Shift)



PDF Analysis

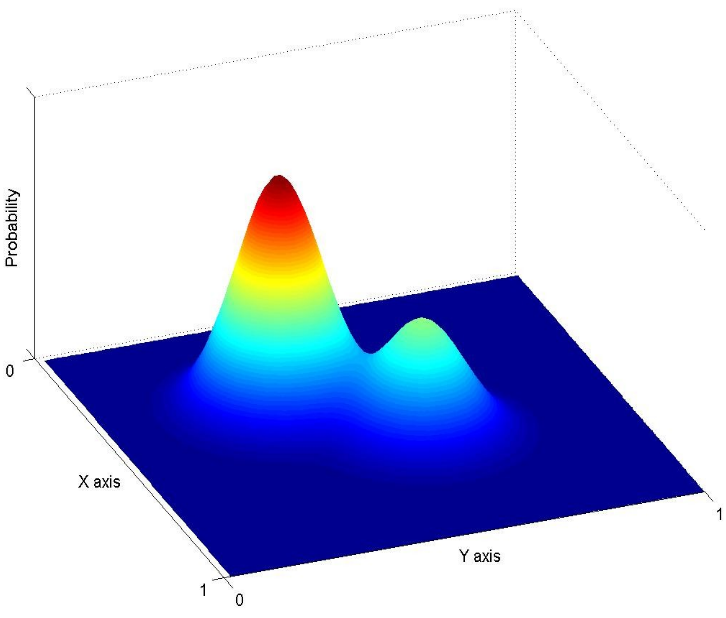
Non-Parametric Density Estimation

Assumption : The data points are sampled from an underlying PDF

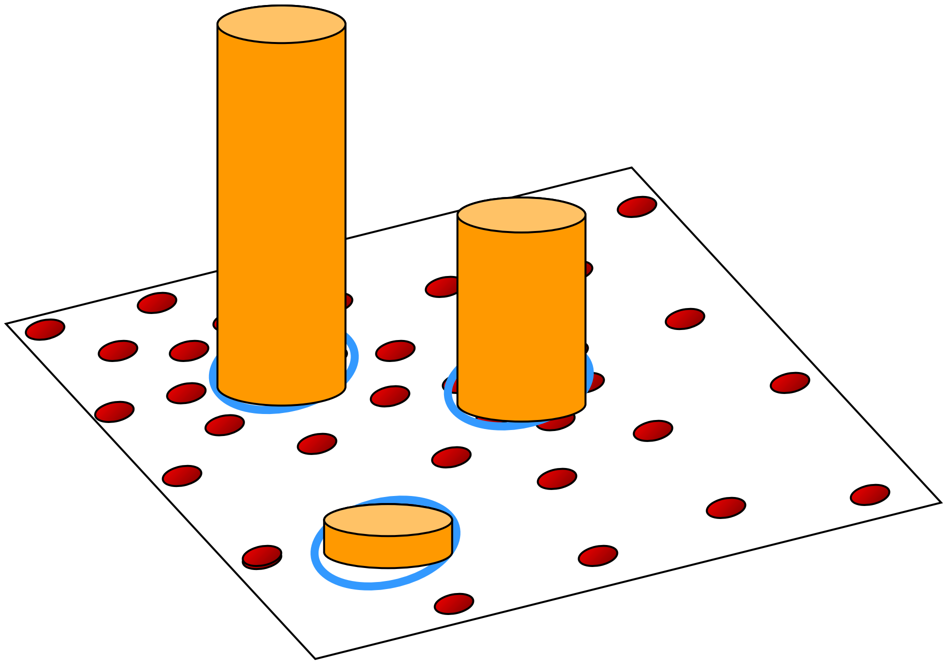


Assumed Underlying PDF **Real Data Samples**

Non-Parametric Density Estimation

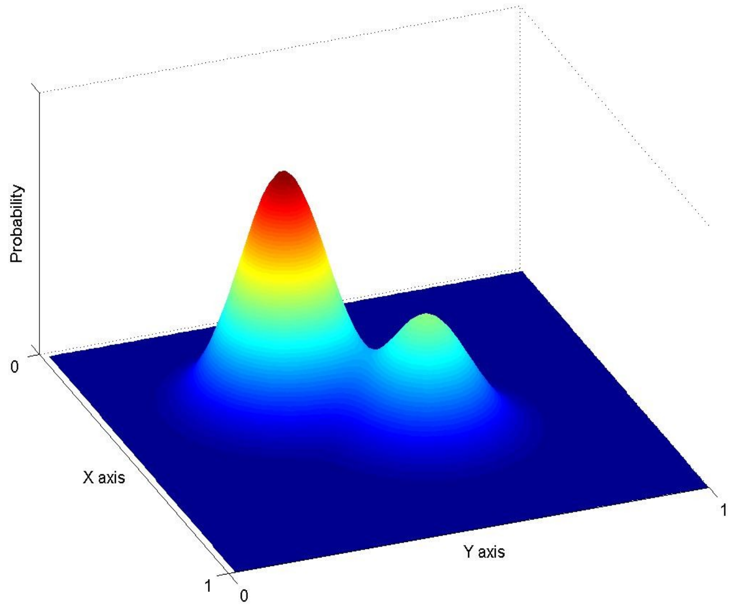


Assumed Underlying PDF

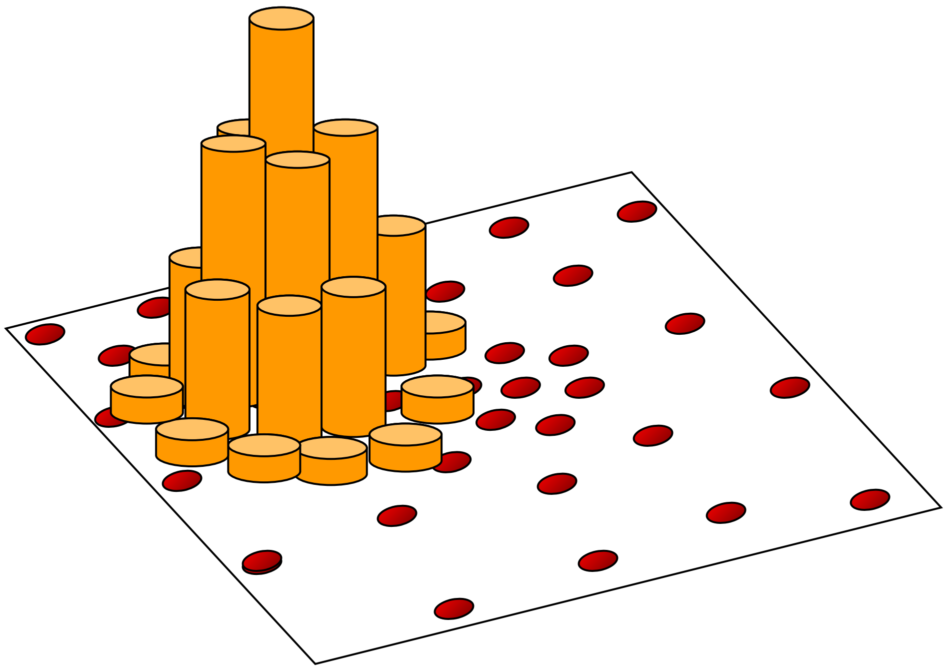


Real Data Samples

Non-Pa?rametric Density Estimation



Assumed Underlying PDF

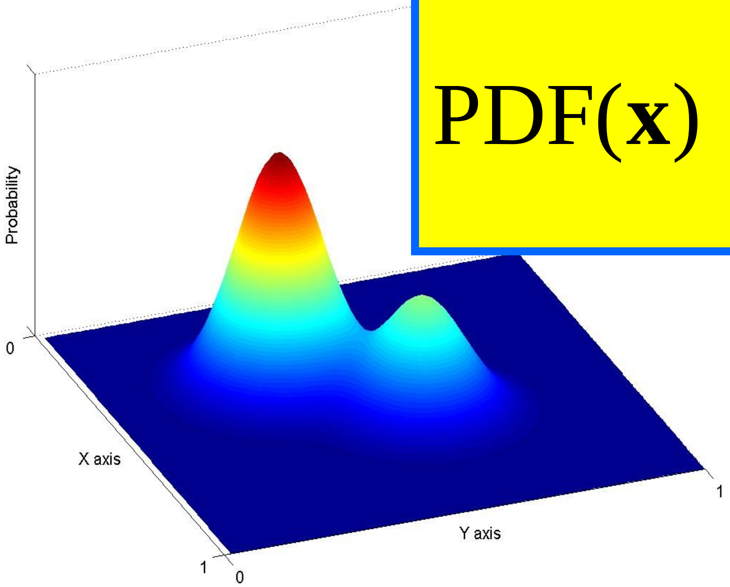


Real Data Samples

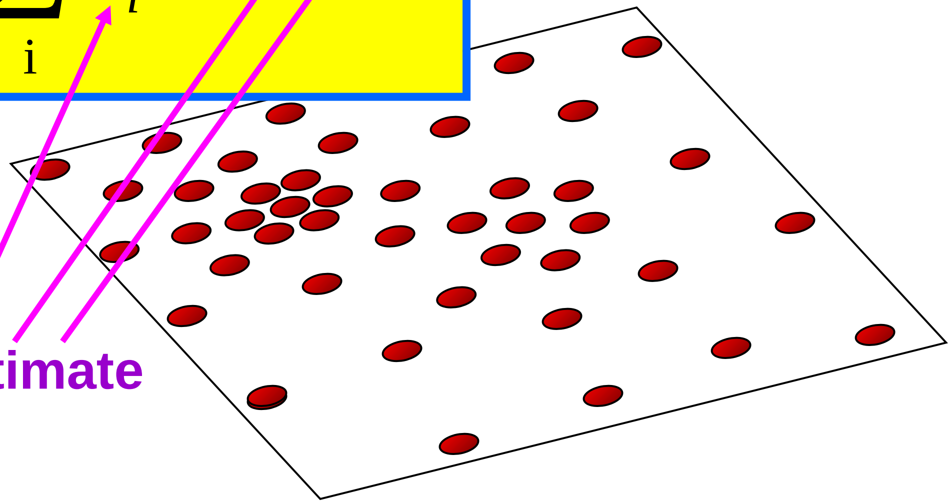
Parametric Density Estimation

<u>Assumption</u> : The data points are sampled from an underlying PDF

$$\mathrm{PDF}(\mathbf{x}) = \sum_{\mathrm{i}} c_i \cdot e^{-\frac{(\mathbf{x}-\mathbf{u}_i)^2}{2\sigma_i^2}}$$



Estimate



Assumed Underlying PDF

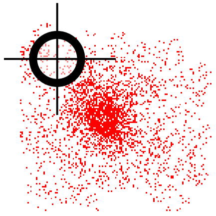
Real Data Samples

Kernel Density Estimation

Parzen Windows - Function Forms

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

A function of some finite number of data points $x_1 \ldots x_n$



Data

In practice one uses the forms:

$$K(\mathbf{x}) = c\prod_{i=1}^{d} k(x_i)$$

or

$$K(\mathbf{x}) = ck\left(\|\mathbf{x}\|\right)$$

Same function on each dimension

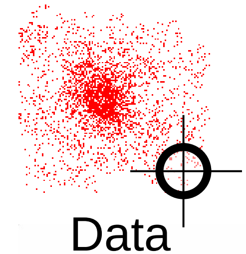
Function of vector length only

Kernel Density Estimation

Various Kernels

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

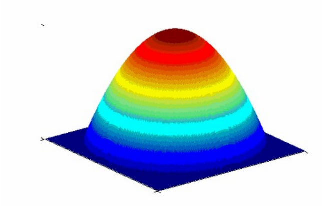
A function of some finite number of data points $x_1 \ldots x_n$



<u>Examples:</u>

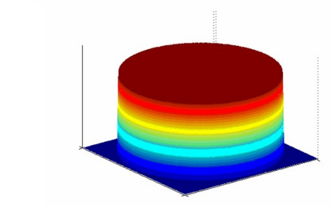
- Epanechnikov Kernel

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



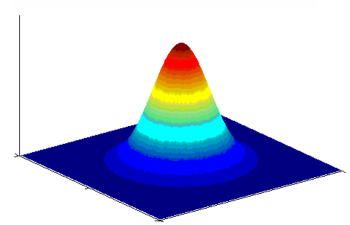
- Uniform Kernel

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



- Normal Kernel

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



Kernel Density Estimation

Gradient

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

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 :

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

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

Computing The Mean Shift

Kernel Density Estimation

Gradient

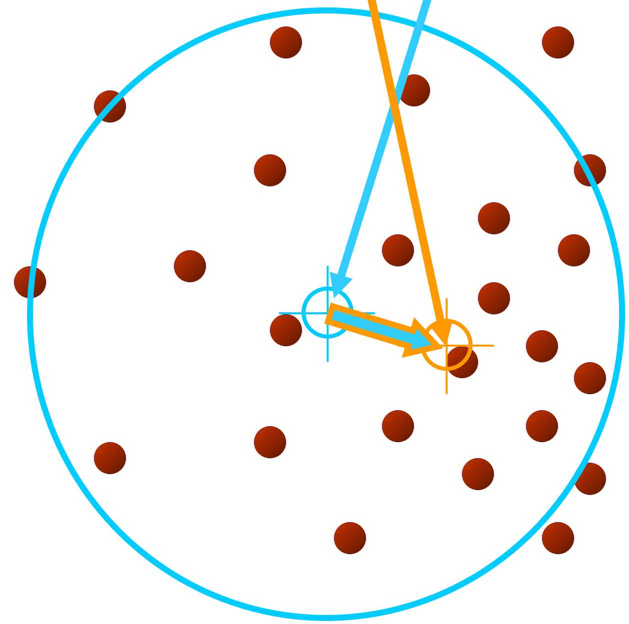
$$\nabla P(\mathbf{x}) = \frac{c}{n}\sum_{i=1}^{n}\nabla k_i = \frac{c}{n}\left[\sum_{i=1}^{n} g_i\right] \cdot \left[\frac{\sum_{i=1}^{n}\mathbf{x}_i g_i}{\sum_{i=1}^{n} g_i} - \mathbf{x}\right]$$

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

Computing The Mean Shift

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

Yet another Kernel density estimation !



Simple Mean Shift procedure:

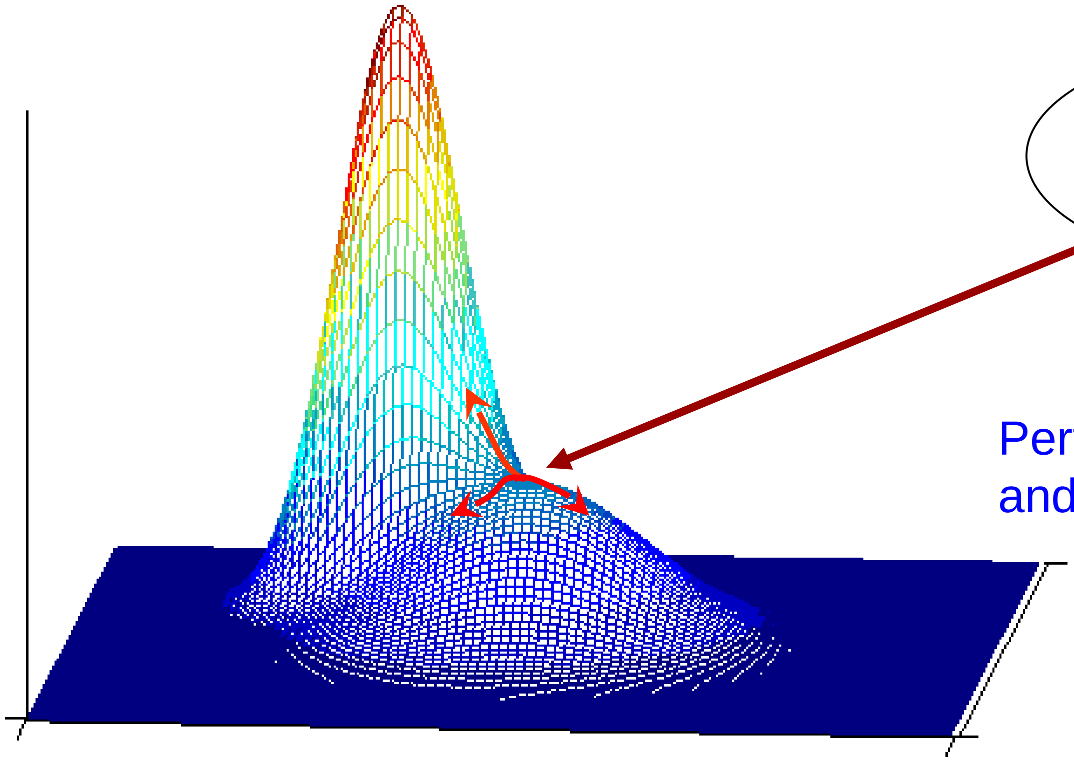
- Compute mean shift vector

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

- Translate the Kernel window by **m(x)**

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

Mean Shift Mode Detection



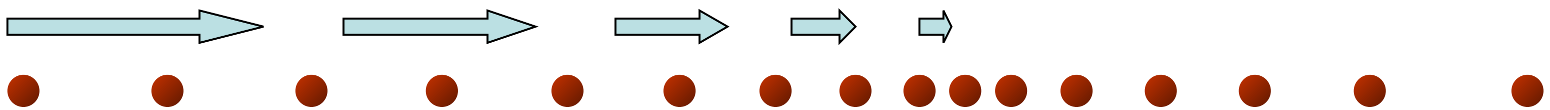
What happens if we reach a saddle point ?

Perturb the mode position and check if we return back

Updated Mean Shift Procedure:

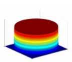
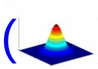
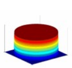
- Find all modes using the Simple Mean Shift Procedure
- Prune modes by perturbing them (find saddle points and plateaus)
- Prune nearby – take highest mode in the window

Mean Shift Properties

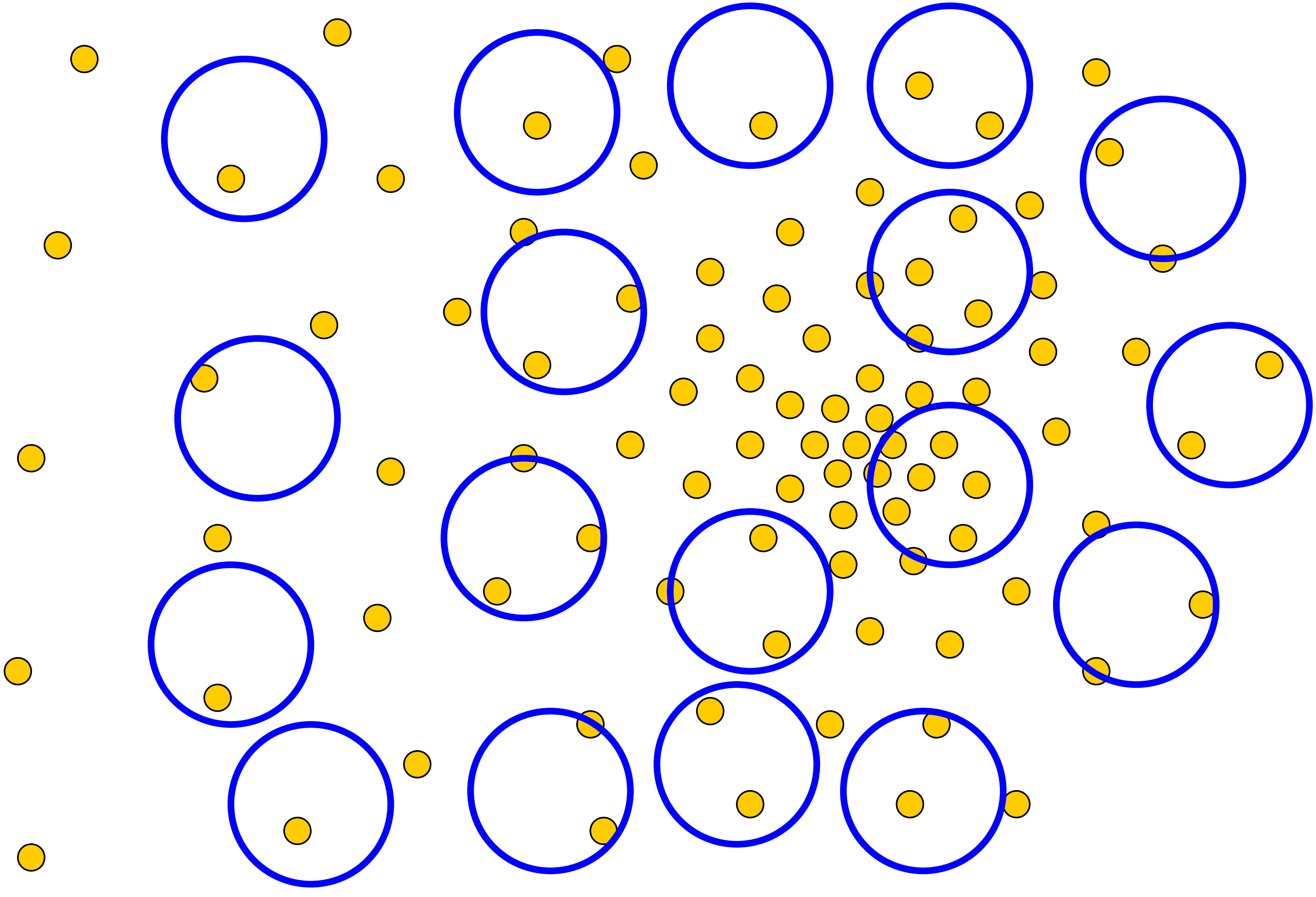


- Automatic convergence speed – the mean shift vector size depends on the gradient itself.
- Near maxima, the steps are small and refined

Adaptive Gradient Ascent

- Convergence is guaranteed for infinitesimal steps only ➔ infinitely convergent, (therefore set a lower bound)
- For Uniform Kernel (), convergence is achieved in a finite number of steps
- Normal Kernel () exhibits a smooth trajectory, but is slower than Uniform Kernel ().

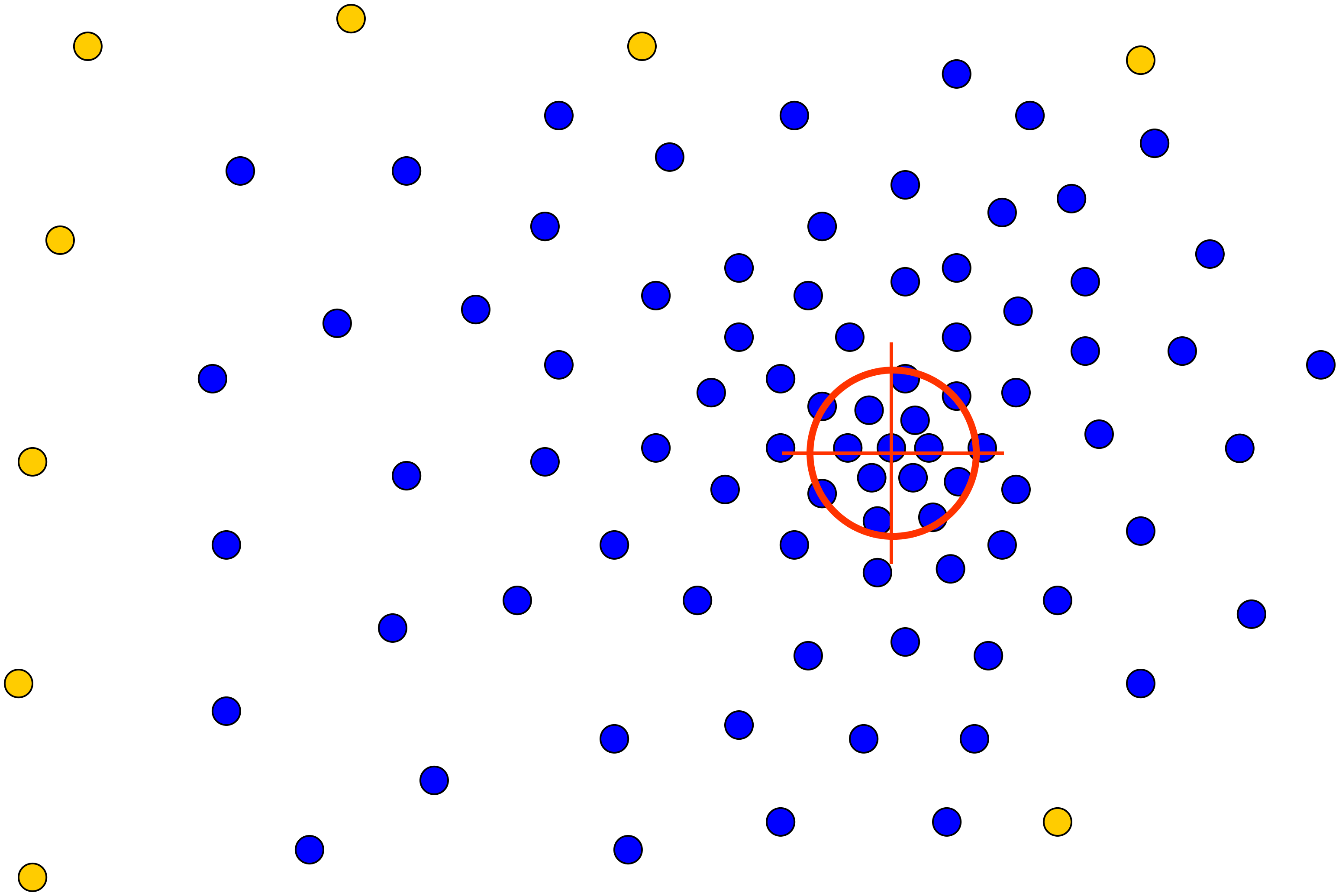
Real Modality Analysis



Tessellate the space with windows

Run the procedure in parallel

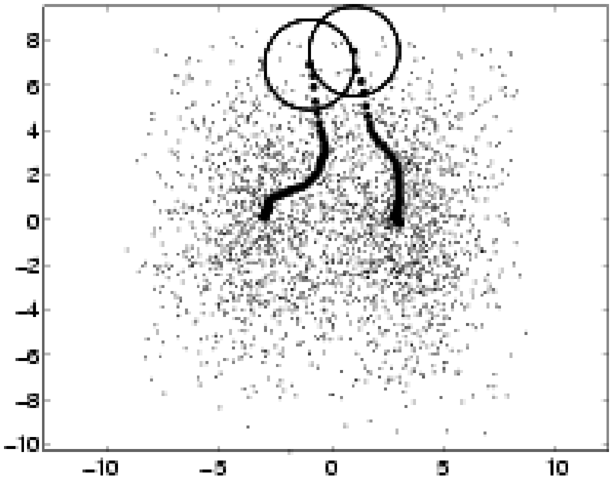
Real Modality Analysis



The blue data points were traversed by the windows towards the mode

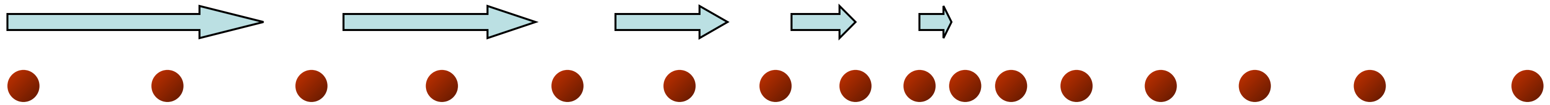
Real Modality Analysis

An example



Window tracks signify the steepest ascent directions

Mean Shift Strengths & Weaknesses



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
- *h* (window size) has a physical meaning, unlike K-Means

Weaknesses :

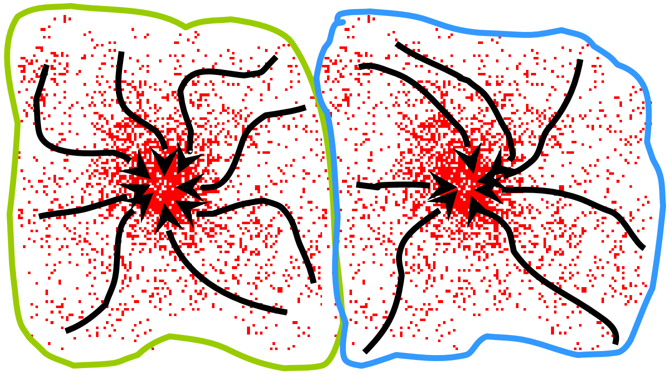
- The window size (bandwidth selection) is not trivial
- Inappropriate window size can cause modes to be merged, or generate additional “shallow” modes ➔ Use adaptive window size

Mean Shift Applications

Clustering

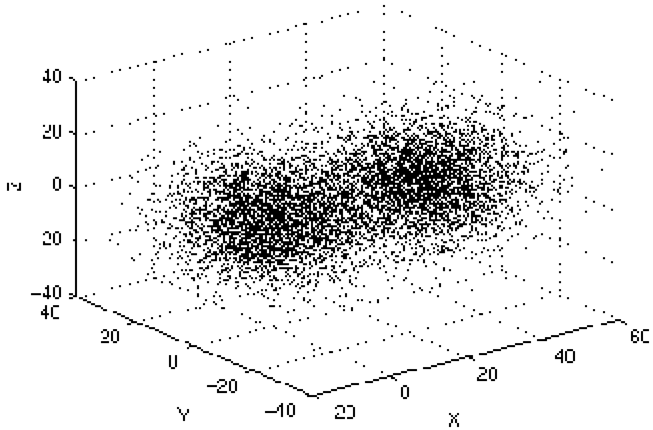
Cluster : All data points in the ***attraction basin*** of a mode

Attraction basin : the region for which all trajectories lead to the same mode



Clustering

Synthetic Examples



Simple Modal Structures

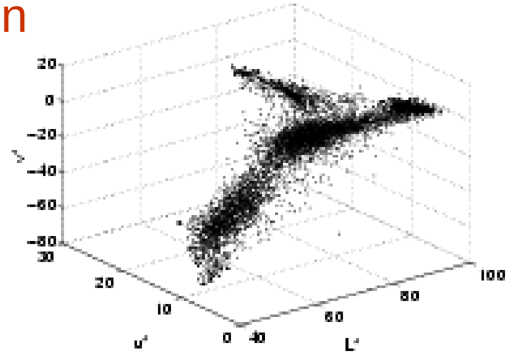
Complex Modal Structures

Clustering

Real Example

Feature space:
L*u*v representation

'nitial window
 enters

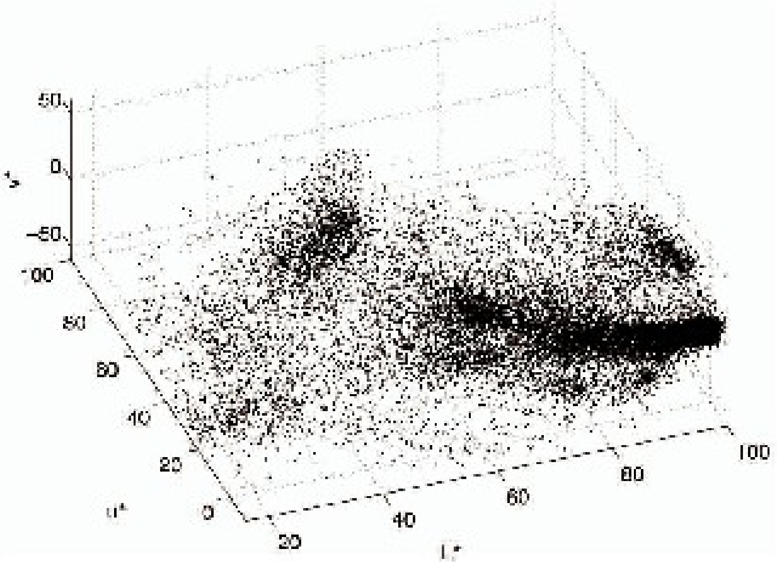


(a)

M

pruning

Clustering
Real Example

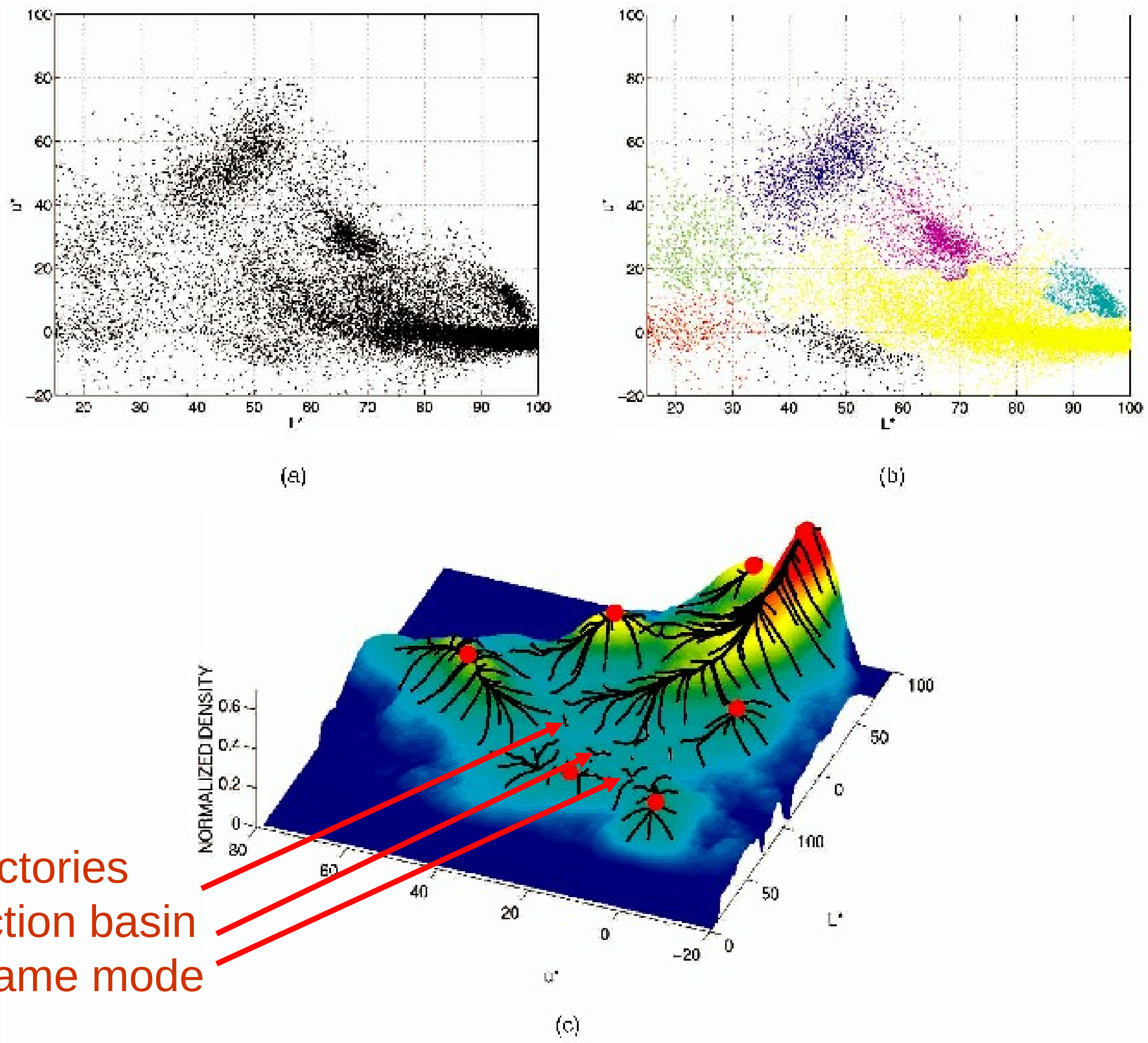


L*u*v space representation

Clustering

Real Example

2D (L*u) space representation



Final clusters

Not all trajectories in the attraction basin reach the same mode

Discontinuity Preserving Smoothing

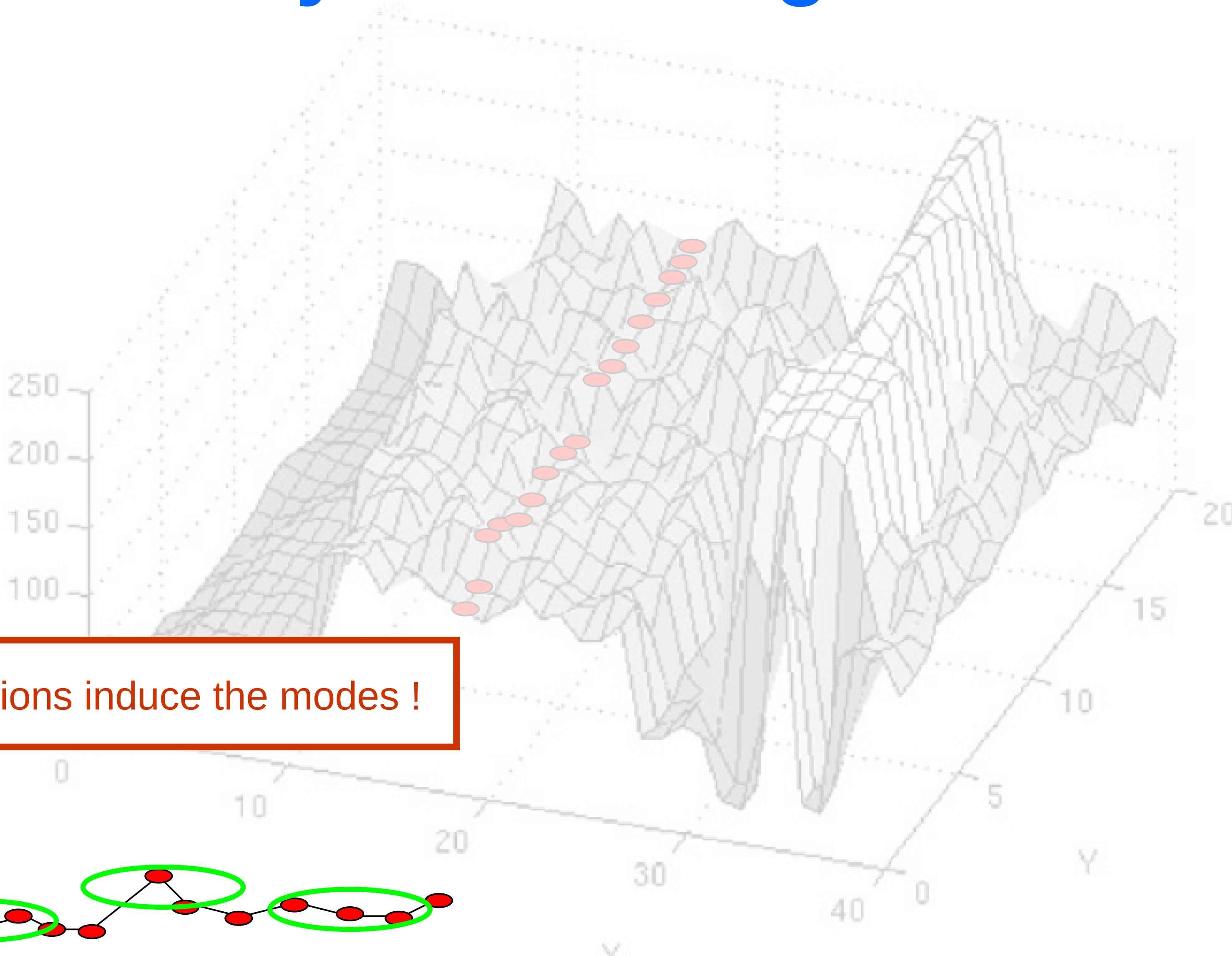
Feature space : Joint domain = spatial coordinates + color space

$$K(\mathbf{x}) = C \cdot k_s\left(\left\|\frac{\mathbf{x}^s}{h_s}\right\|\right) \cdot k_r\left(\left\|\frac{\mathbf{x}^r}{h_r}\right\|\right)$$

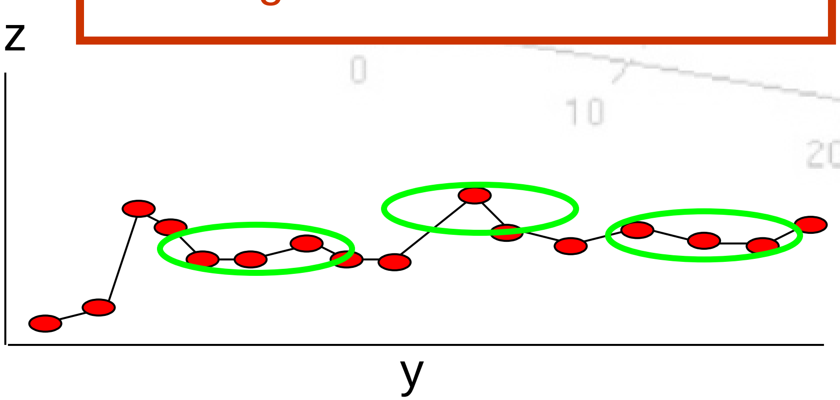
Meaning : treat the image as data points in the spatial and gray level domain

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

Discontinuity Preserving Smoothing

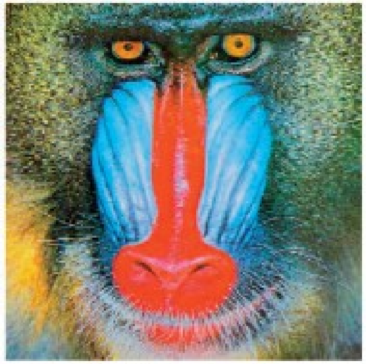


Flat regions induce the modes !

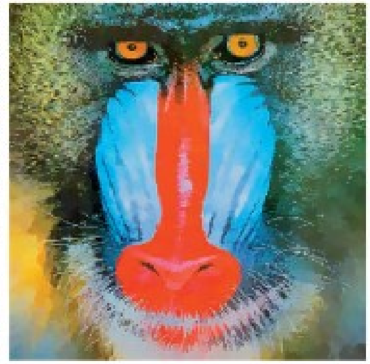


Discontinuity Preserving Smoothing

The effect of window size in spatial and range spaces

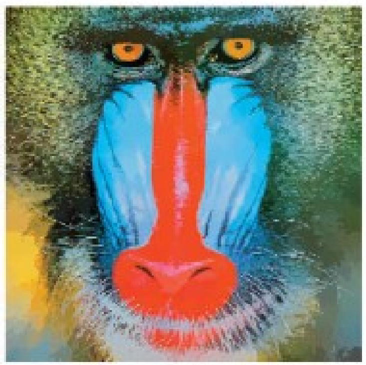


Original



$(h_s, h_r) = (8, 8)$

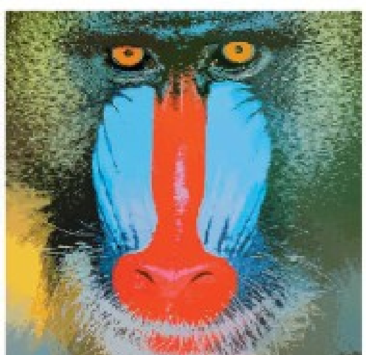
$(h_s, h_r) = (8, 16)$



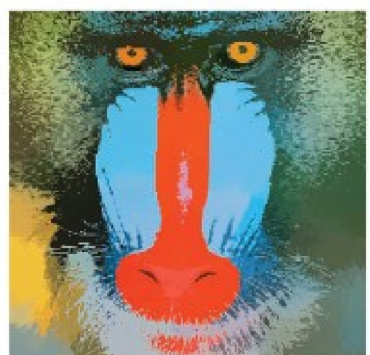
$(h_s, h_r) = (16, 4)$

$(h_s, h_r) = (16, 8)$

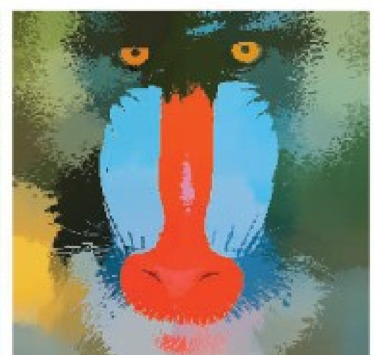
$(h_s, h_r) = (16, 16)$



$(h_s, h_r) = (32, 4)$



$(h_s, h_r) = (32, 8)$



$(h_s, h_r) = (32, 16)$

Discontinuity Preserving Smoothing

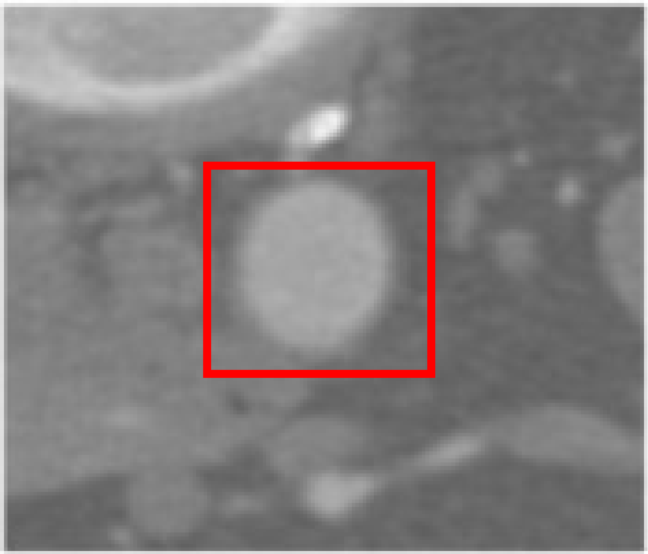
Example

Discontinuity Preserving Smoothing

Example

Object Contour Detection

Ray Propagation



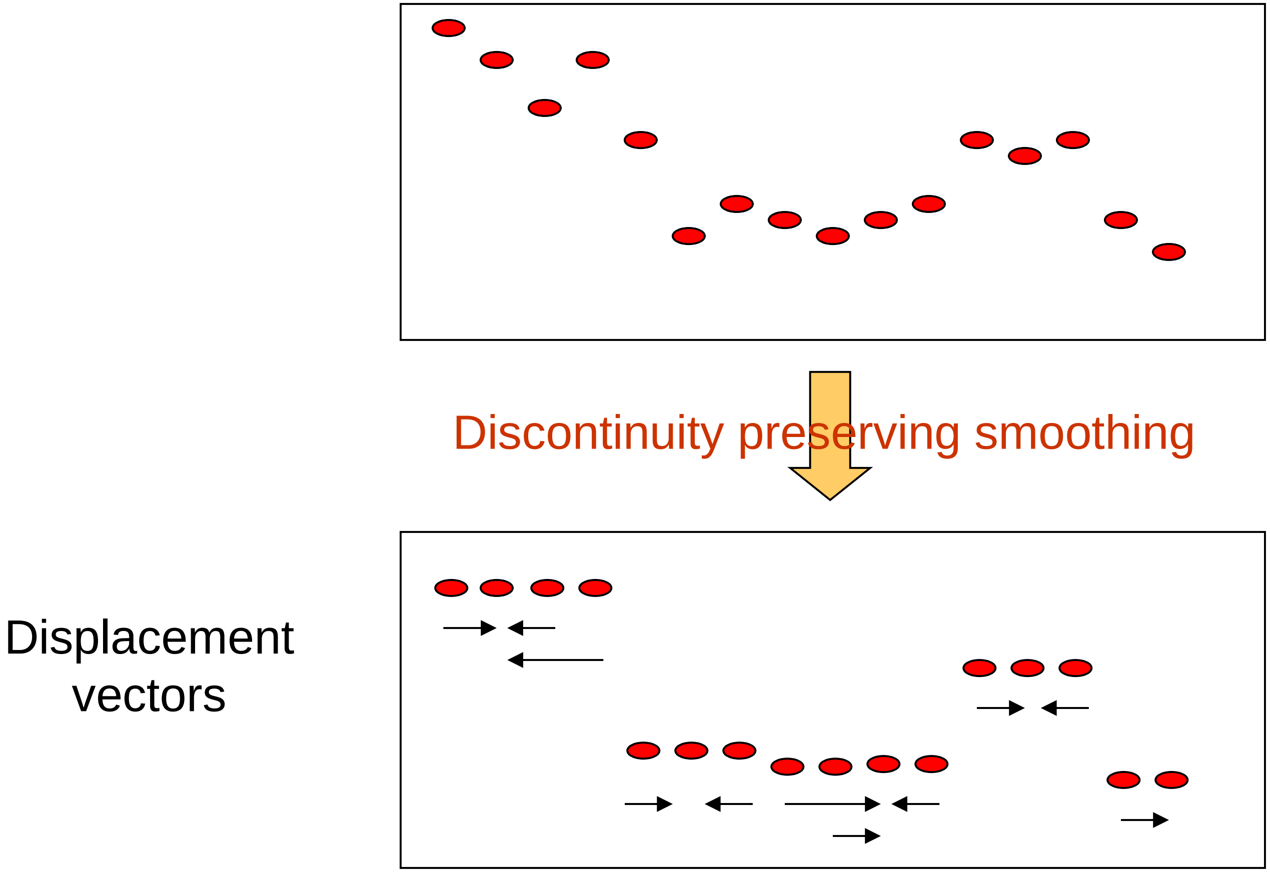
Accurately segment various objects (rounded in nature) in medical images

Vessel Detection by Mean Shift Based Ray Propagation, by Tek, Comaniciu, Williams

Object Contour Detection

Ray Propagation

Use displacement data to guide ray propagation



Vessel Detection by Mean Shift Based Ray Propagation, by Tek, Comaniciu, Williams

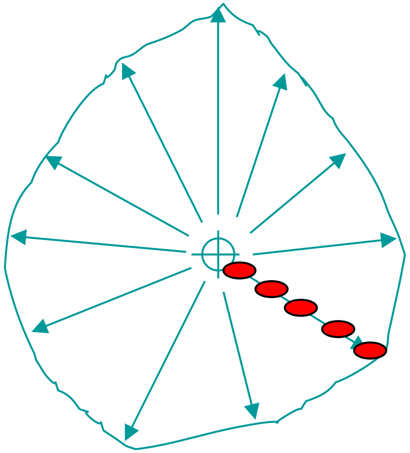
Object Contour Detection

Ray Propagation

Speed function

$$\frac{\partial Ray}{\partial t}(s,t) = Speed(x,y)\cdot \vec{N}$$

Normal to the contour

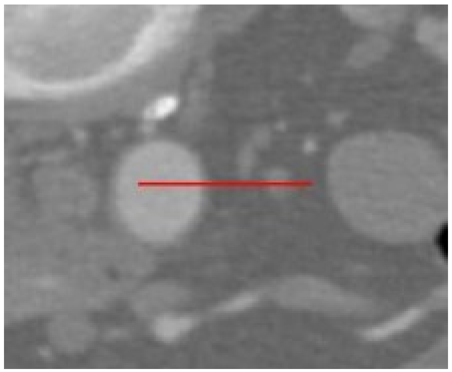


Curvature

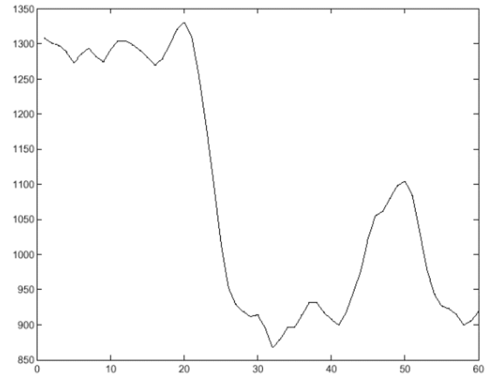
$$Speed(x,y) = \alpha f\left(\nabla disp(x,y)\right) + \beta\kappa(x,y)$$

Object Contour Detection

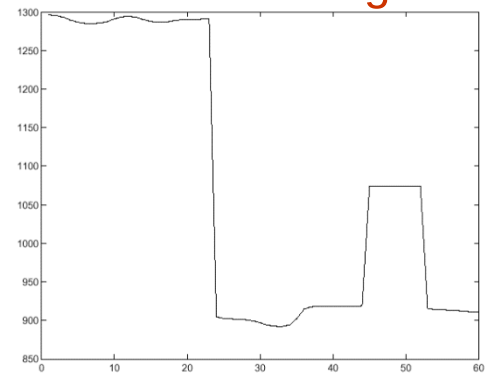
Original image



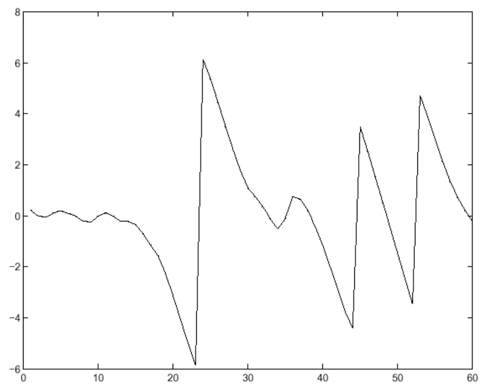
Gray levels along red line



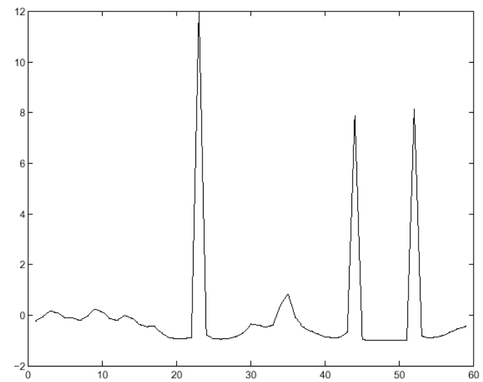
Gray levels after smoothing



Displacement vectors



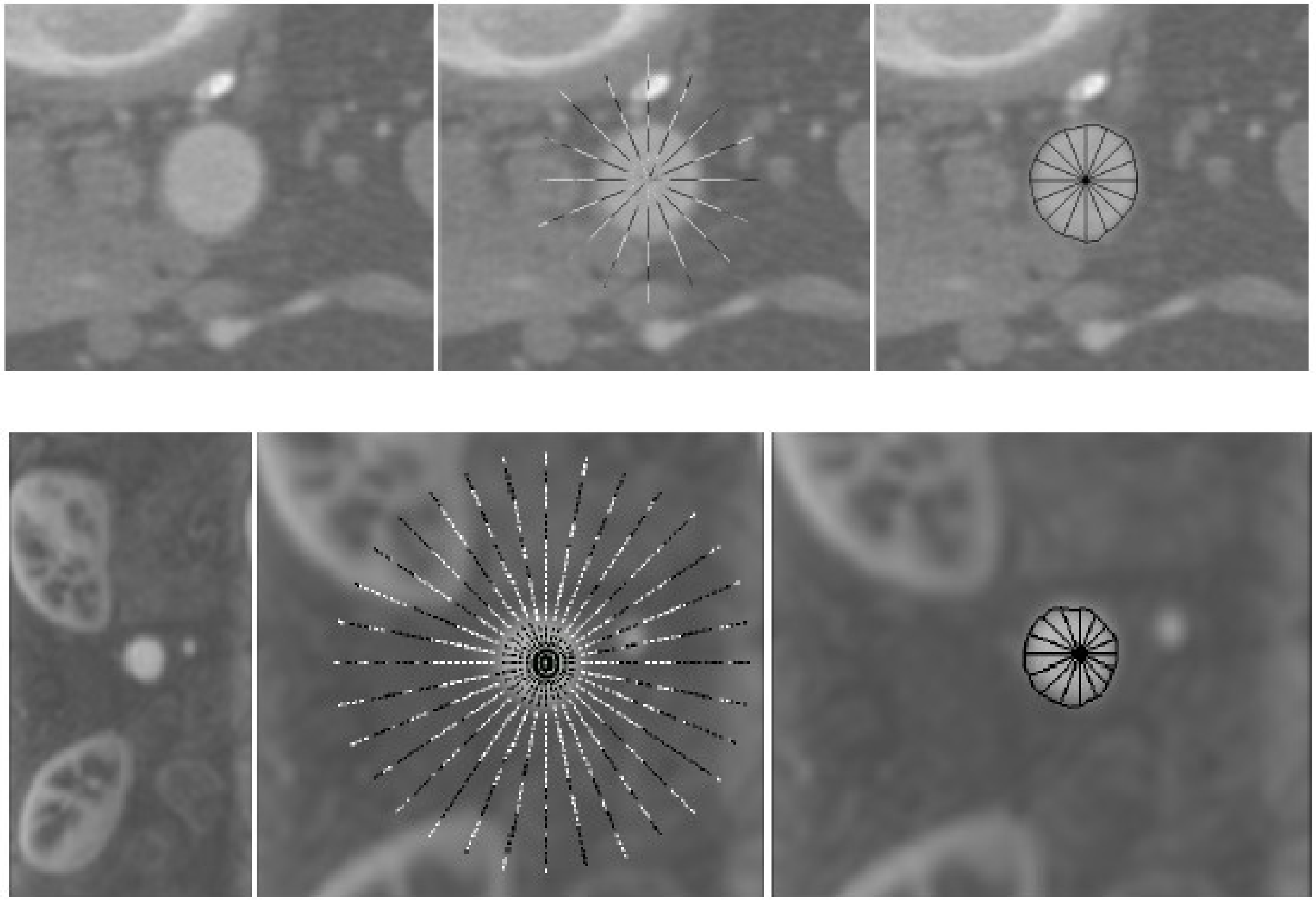
Displacement vectors' derivative



$$Speed(x,y) = \alpha f\left(\nabla disp(x,y)\right) + \beta\kappa(x,y)$$

Object Contour Detection

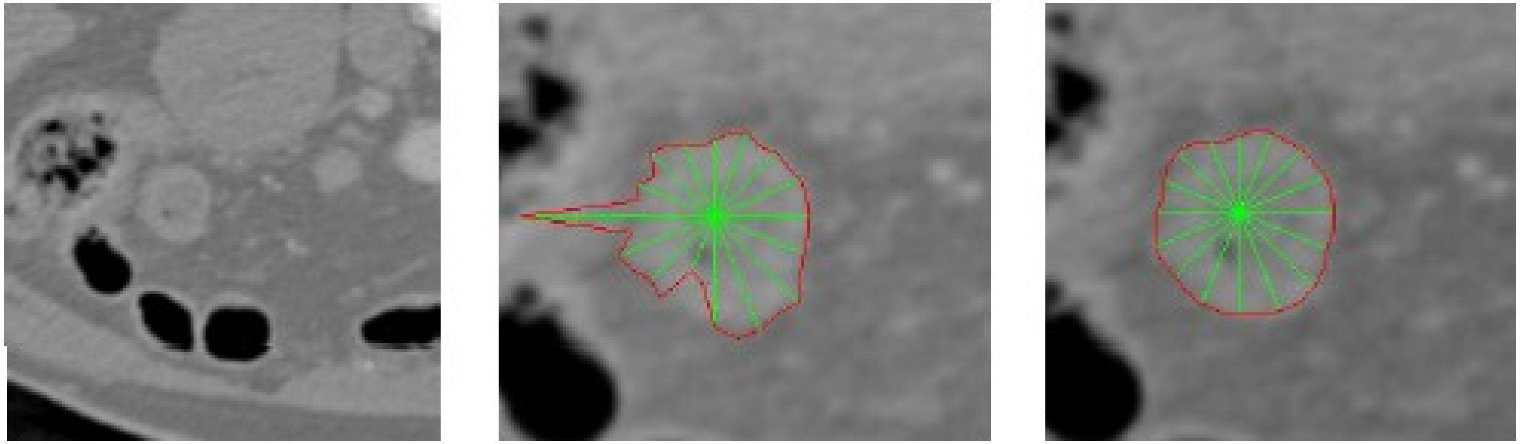
Example



Object Contour Detection

Example

Importance of smoothing by curvature



Segmentation

Segment = Cluster,   or Cluster of Clusters

Algorithm:

- Run Filtering (*discontinuity preserving smoothing*)
- Cluster the clusters which are closer than window size

Segmentation
Example

…when feature space is only gray levels…

Segmentation

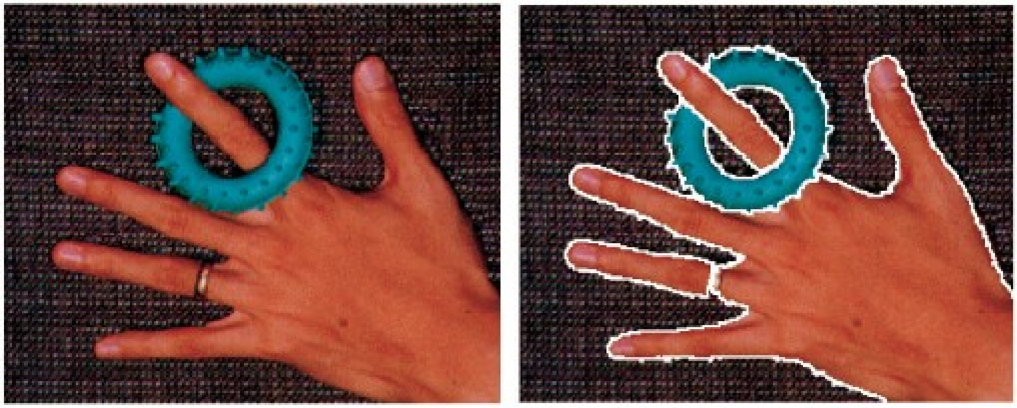
Example

Segmentation

Example

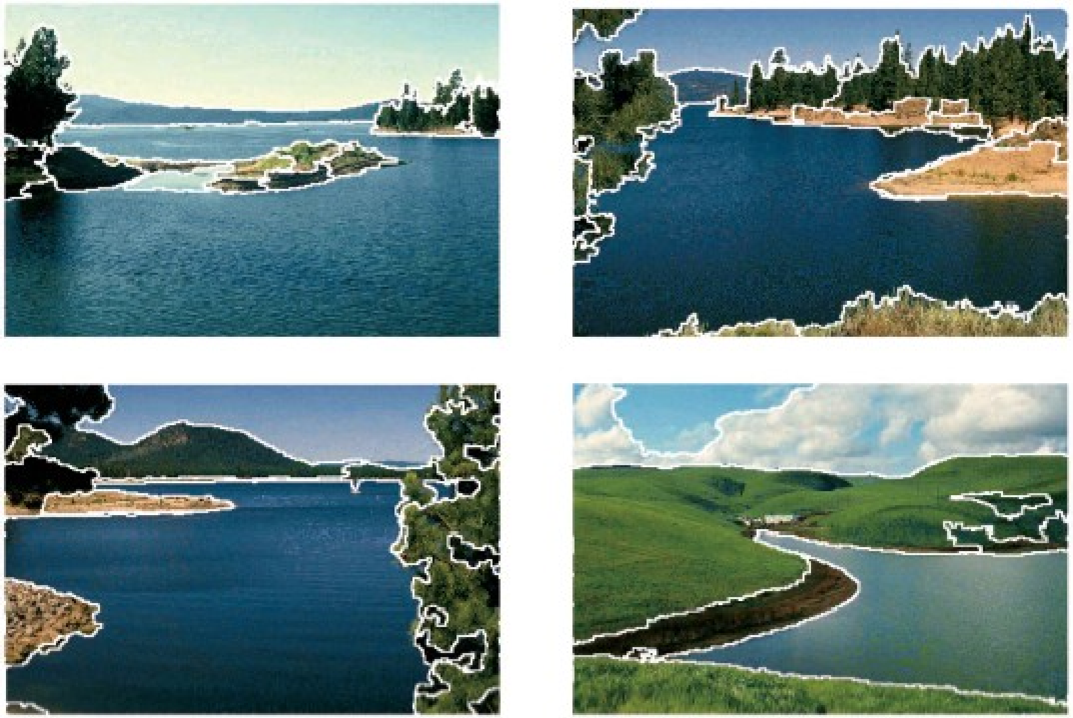
Segmentation

Example



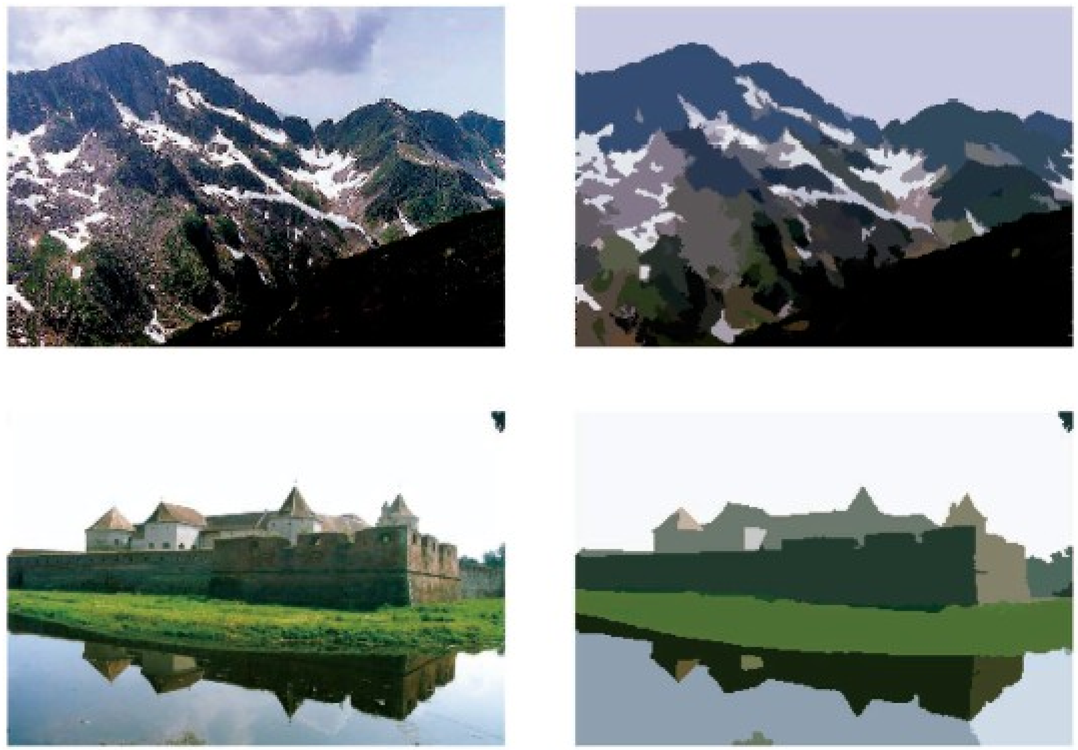
Segmentation

Example



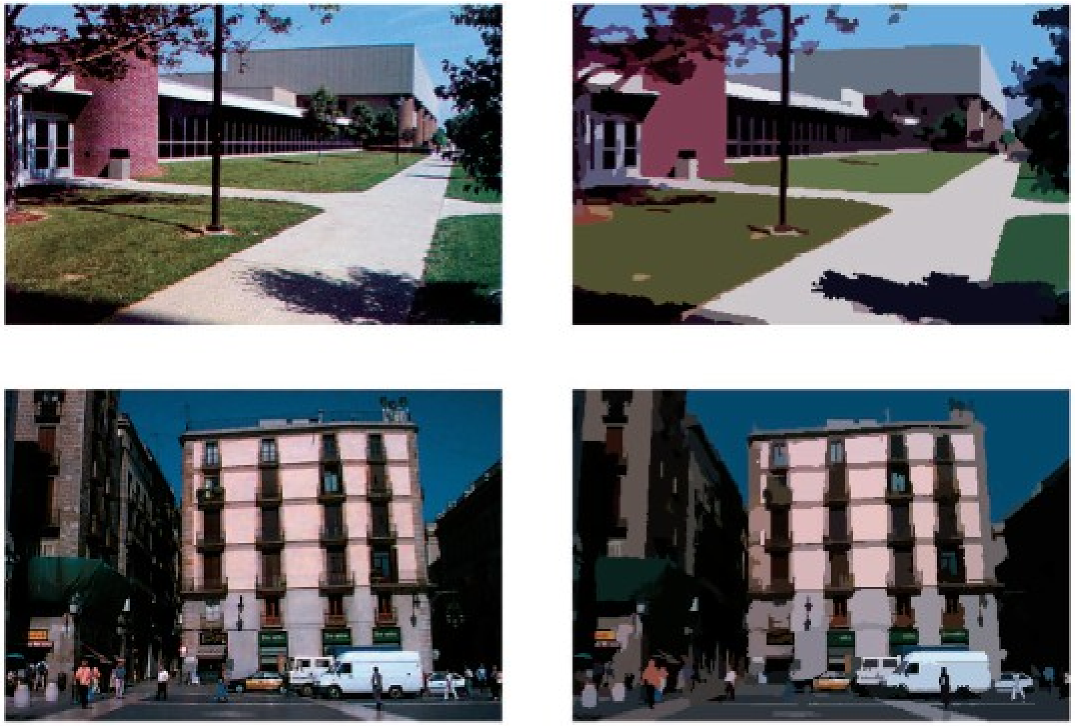
Segmentation

Example

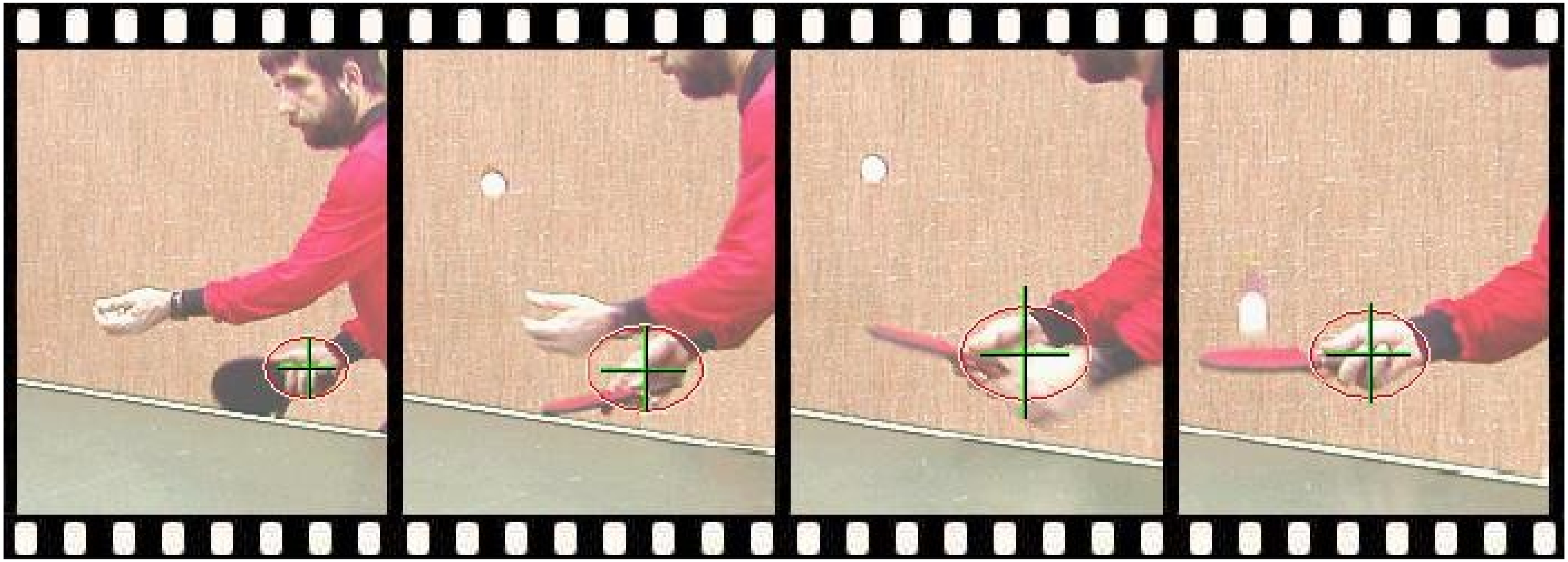


Segmentation

Example



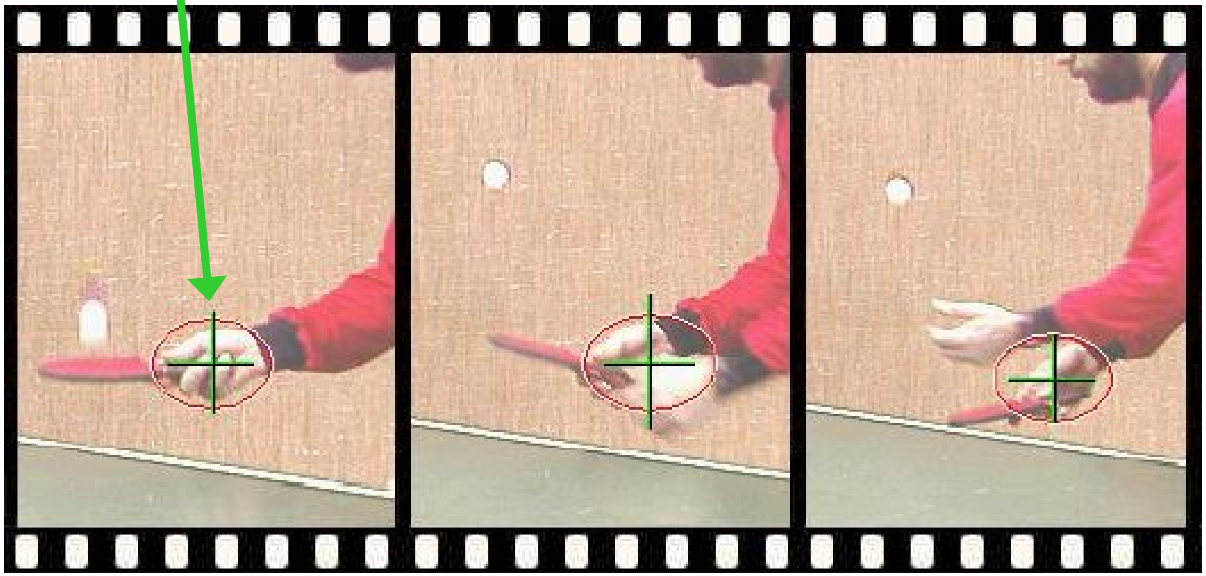
Non-Rigid Object Tracking



Mean-Shift Object Tracking

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 → …

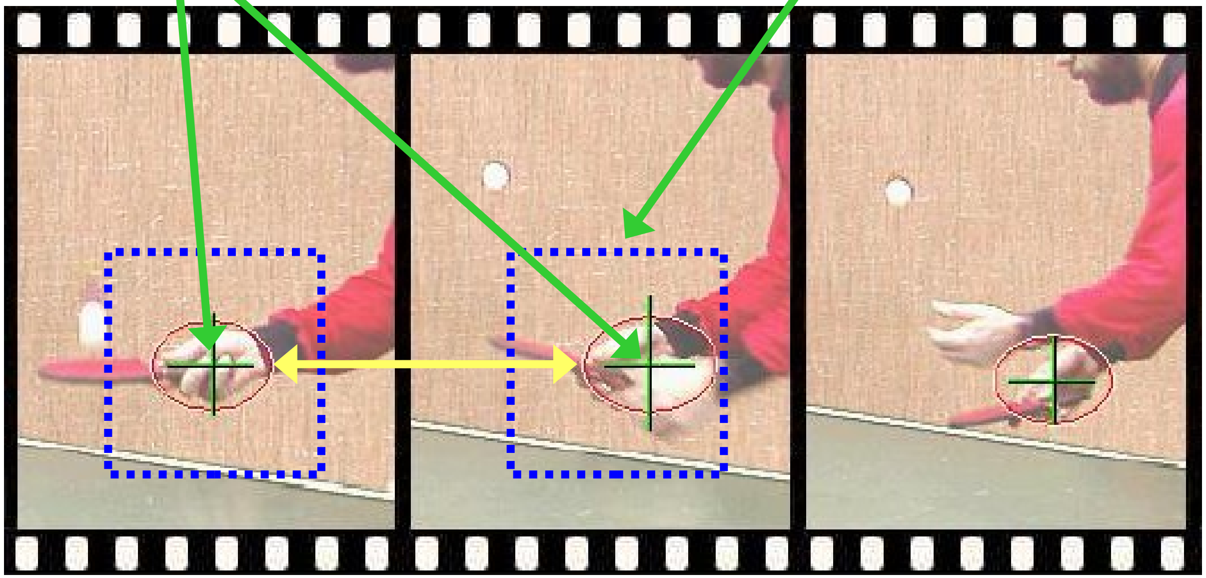
Mean-Shift Object Tracking

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.



Model

Candidate

Repeat the same process in the next pair of frames

… Current frame …

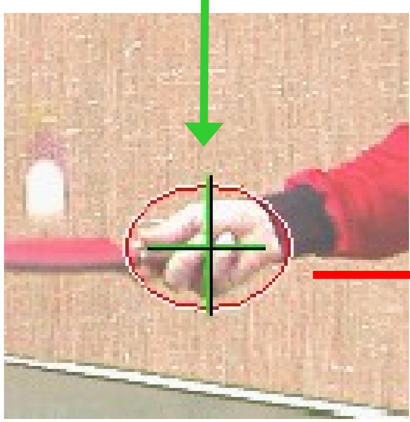
Mean-Shift Object Tracking

Target Representation

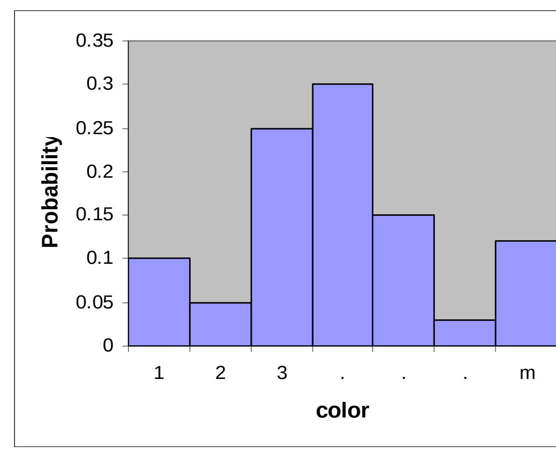
Choose a reference target model

Choose a feature space

Represent the model by its PDF in the feature space



Quantized Color Space

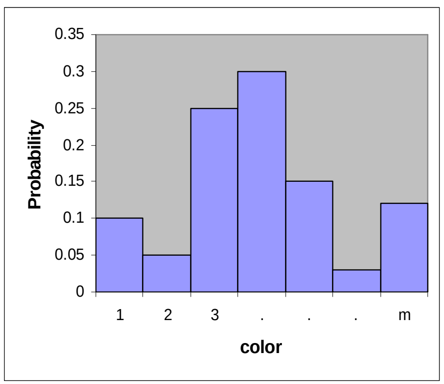


Kernel Based Object Tracking, by Comaniniu, Ramesh, Meer

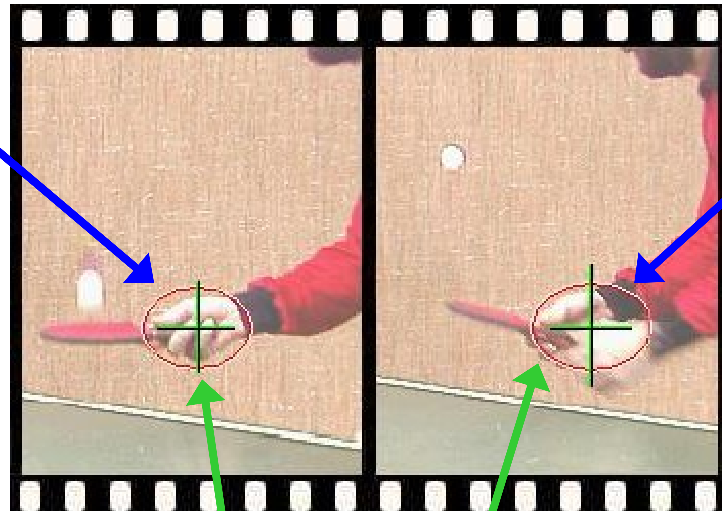
Mean-Shift Object Tracking

PDF Representation

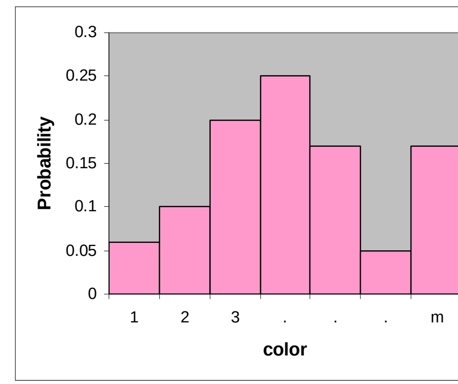
Target Model
(centered at 0)



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



Target Candidate
(centered at y)



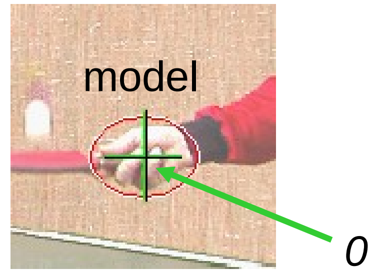
$$\vec{p}(y) = \left\{ p_u(y) \right\}_{u=1..m} \qquad \sum_{u=1}^{m} p_u = 1$$

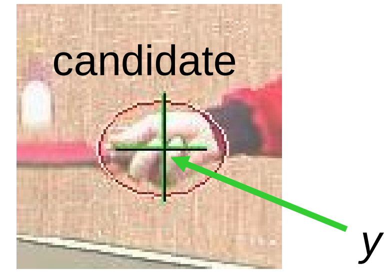
Similarity Function:

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

Mean-Shift Object Tracking

Finding the PDF of the target model





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

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

- Peripheral pixels are affected by occlusion and background interference

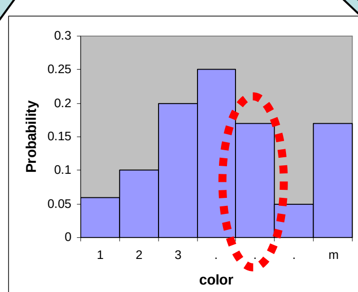
$b(x)$ The color bin index (1..*m*) of pixel *x*

Probability of feature u in model

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

Normalization factor

Pixel weight

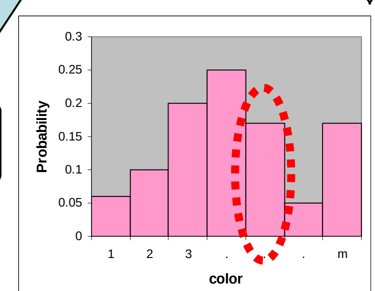


Probability of feature u in candidate

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

Normalization factor

Pixel weight



Mean-Shift Object Tracking

Similarity Function

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

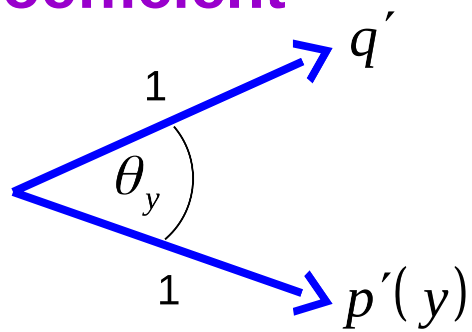
Target candidate: $p(y) = \left(p_1(y), \ldots, p_m(y) \right)$

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

The Bhattacharyya Coefficient

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

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

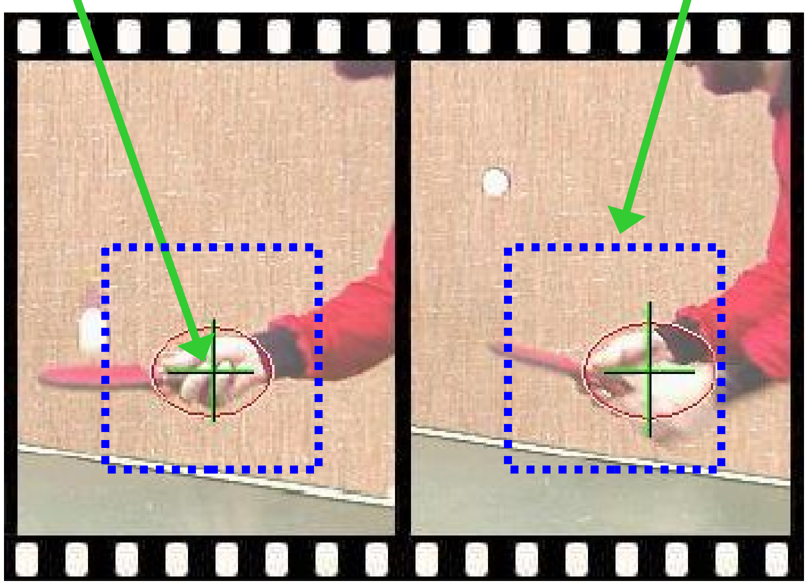


$$f(y) = \cos\theta_y = \frac{p'(y)^T q'}{\left\| p'(y) \right\| \cdot \left\| q' \right\|} = \sum_{u=1}^{m} \sqrt{p_u(y)\, q_u}$$

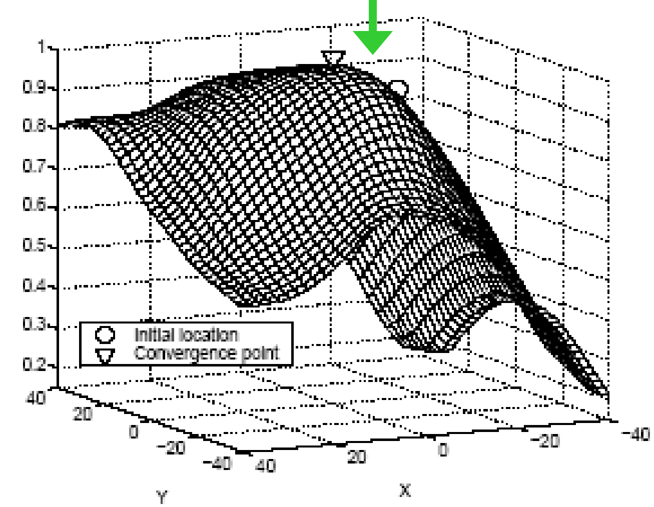
Mean-Shift Object Tracking

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.



q $\qquad\qquad$ $p(y)$



$$f\left[p(y), q\right]$$

Mean-Shift Object Tracking

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)

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

Independent of y

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

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

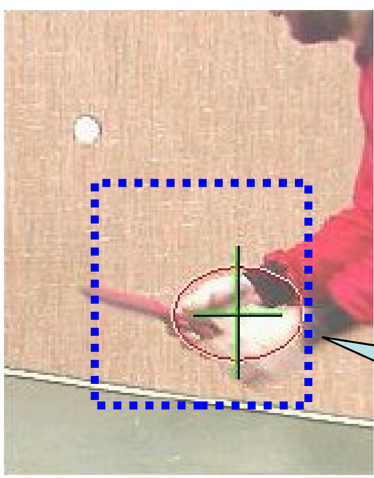
Density estimate!
(as a function of y)

Mean-Shift Object Tracking

Maximizing the Similarity Function

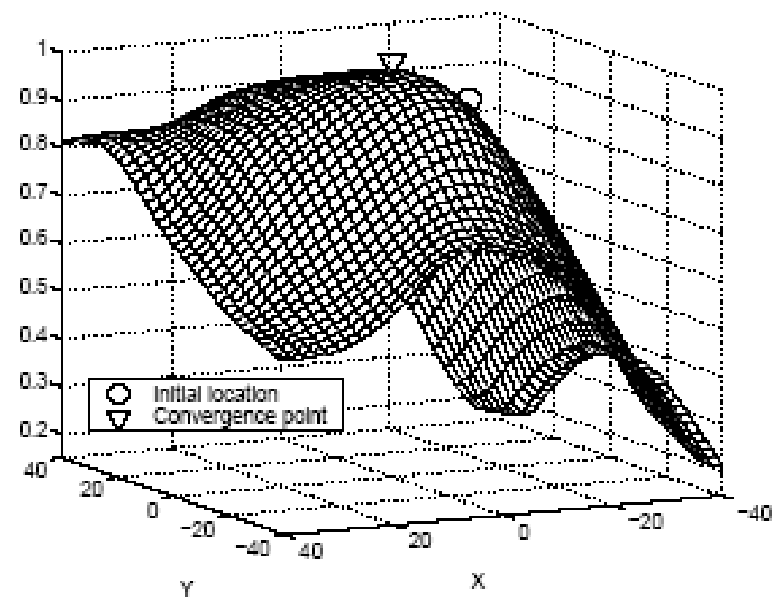
The mode of $\frac{C_h}{2}\sum_{i=1}^{n} w_i k\left(\left\|\frac{y - x_i}{h}\right\|^2\right)$ = sought maximum

Important Assumption:



The target representation provides sufficient discrimination

One mode in the searched neighborhood



Mean-Shift Object Tracking

Applying Mean-Shift

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

Original Mean-Shift:

Find mode of $c\sum_{i=1}^{n} k\left(\left\|\frac{y-x_i}{h}\right\|^2\right)$ using $y_1=\frac{\sum_{i=1}^{n} x_i g\left(\left\|\frac{y_0-x_i}{h}\right\|^2\right)}{\sum_{i=1}^{n} g\left(\left\|\frac{y_0-x_i}{h}\right\|^2\right)}$

Extended Mean-Shift:

Find mode of $c\sum_{i=1}^{n} \boxed{w_i} k\left(\left\|\frac{y-x_i}{h}\right\|^2\right)$ using $y_1=\frac{\sum_{i=1}^{n} x_i \boxed{w_i} g\left(\left\|\frac{y_0-x_i}{h}\right\|^2\right)}{\sum_{i=1}^{n} \boxed{w_i} g\left(\left\|\frac{y_0-x_i}{h}\right\|^2\right)}$

Mean-Shift Object Tracking

About Kernels and Profiles

A special class of radially symmetric kernels:

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

The profile of kernel *K*

$$k'(x) = -g(x)$$

Extended Mean-Shift:

Find mode of $c\sum_{i=1}^{n} w_i k\left(\left\|\frac{y - x_i}{h}\right\|^2\right)$ using $y_1 = \frac{\sum_{i=1}^{n} x_i w_i g\left(\left\|\frac{y_0 - x_i}{h}\right\|^2\right)}{\sum_{i=1}^{n} w_i g\left(\left\|\frac{y_0 - x_i}{h}\right\|^2\right)}$

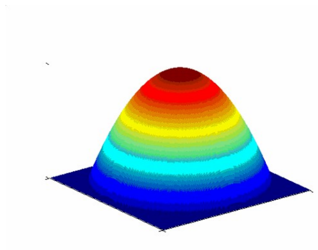
Mean-Shift Object Tracking

Choosing the Kernel

A special class of radially symmetric kernels:

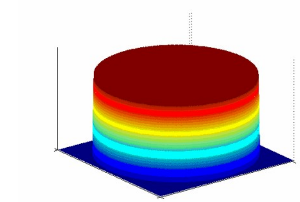
$$K(x) = ck\left(\|x\|^2\right)$$

Epanechnikov kernel:



$$k(x) = \begin{cases} 1 - x & \text{if } \|x\| \le 1 \\ 0 & \text{otherwise} \end{cases}$$

Uniform kernel:



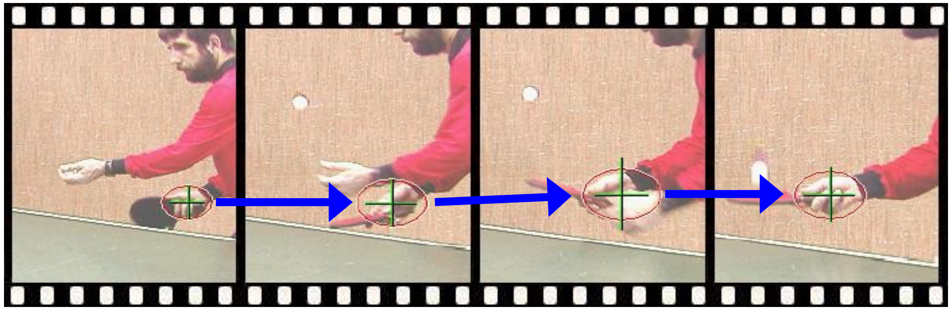
$$g(x) = -k(x) = \begin{cases} 1 & \text{if } \|x\| \le 1 \\ 0 & \text{otherwise} \end{cases}$$

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

Mean-Shift Object Tracking
Adaptive Scale

Problem:

The scale of the target changes in time → The scale (h) of the kernel must be adapted



Solution:

Run localization 3 times with different h → Choose h that achieves maximum similarity

Mean-Shift Object Tracking

Results

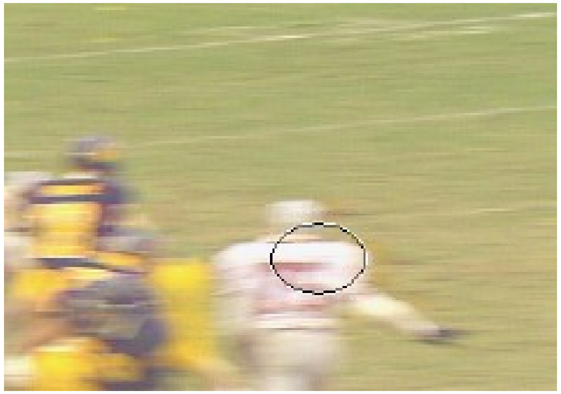
Feature space: 16×16×16 quantized RGB
Target: manually selected on 1st frame
Average mean-shift iterations: 4

Mean-Shift Object Tracking

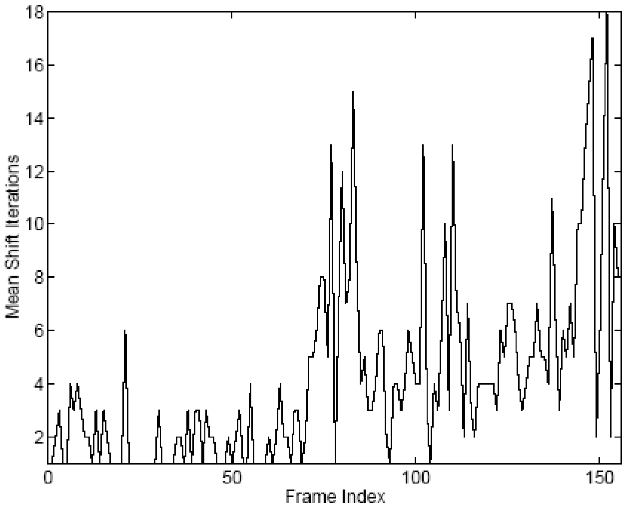
Results

Partial occlusion

Distraction



Motion blur



Mean-Shift Object Tracking
Results

Mean-Shift Object Tracking

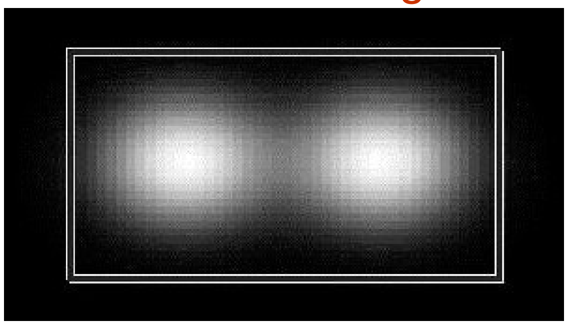
Results

Feature space: 128×128 quantized RG

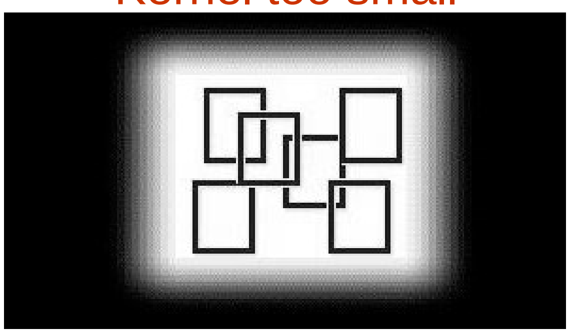
Mean-Shift Object Tracking

The Scale Selection Problem

Kernel too big



Kernel too small



Poor localization → *h* mustn't get too big or too small!

Problem:

In uniformly colored regions, similarity is invariant to *h* → Smaller *h* may achieve better similarity → Nothing keeps *h* from shrinking too small!

Tracking Through Scale Space

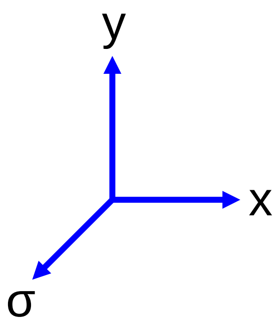
Motivation

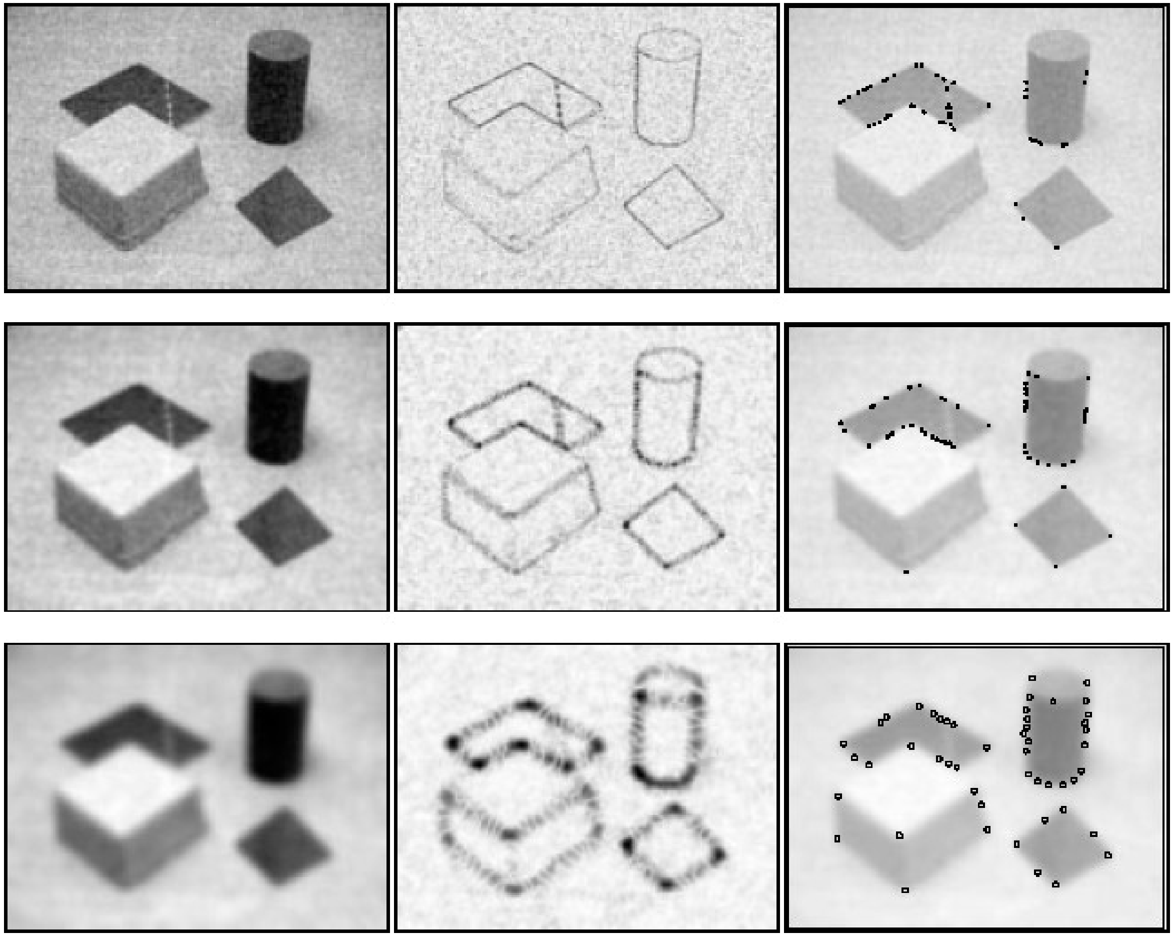
Spatial localization for several scales → Simultaneous localization in space and scale

Previous method — This method

Lindeberg's Theory

Selecting the best scale for describing image features





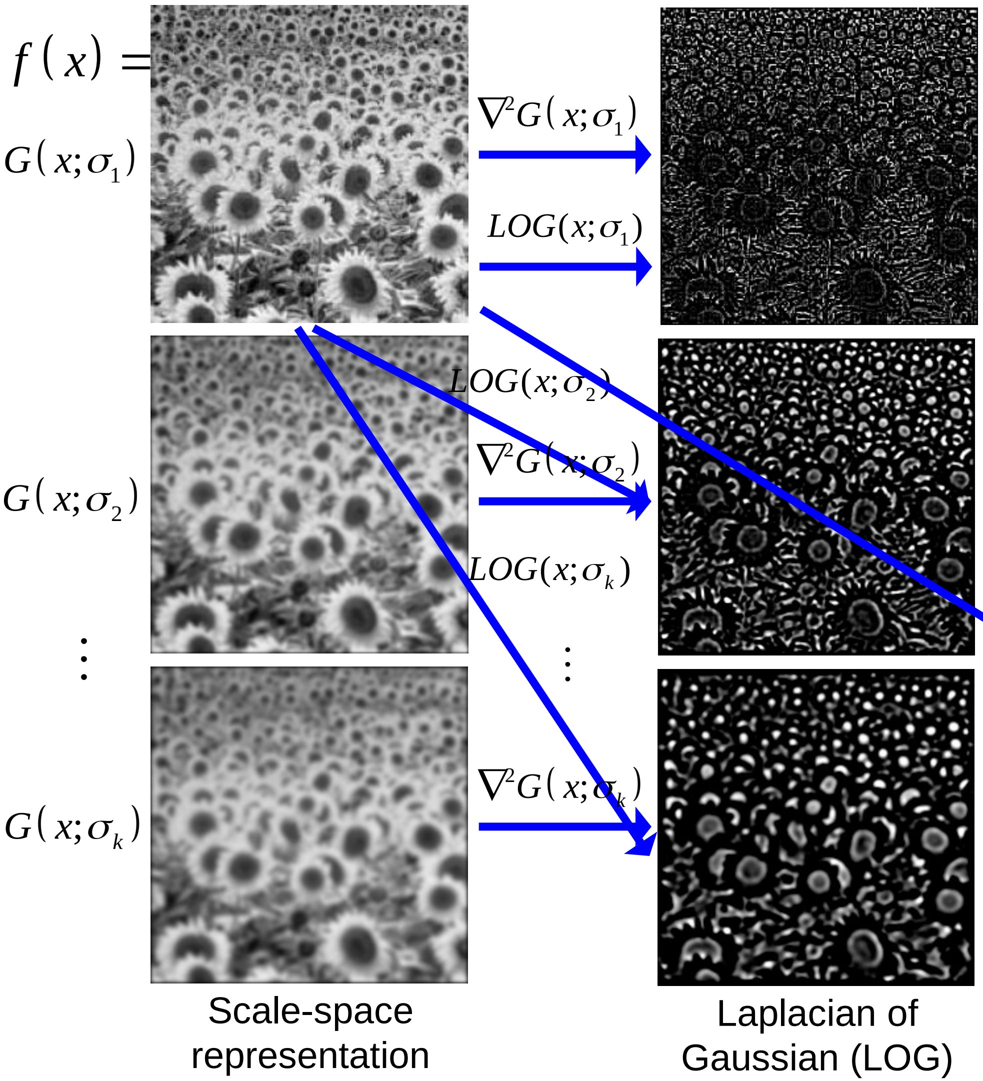
Scale-space representation

Differential operator applied

50 strongest responses

Lindeberg's Theory

The Laplacian operator for selecting blob-like features



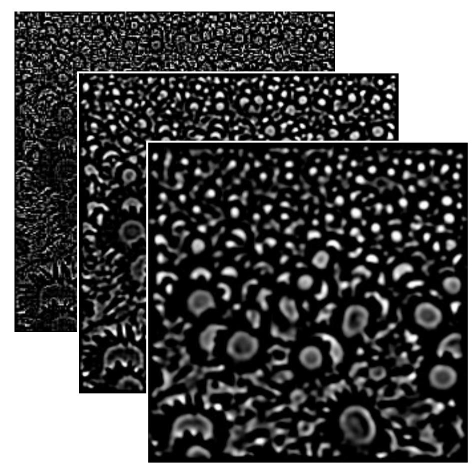
2D LOG filter with scale σ

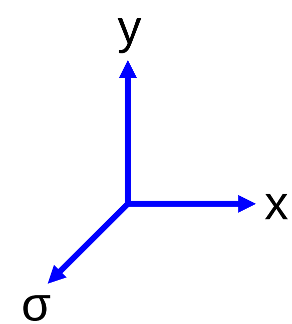
$$LOG(x;\sigma) = \frac{2\sigma^2 - \|x\|^2}{2\pi\sigma^6} e^{\frac{-\|x\|^2}{2\sigma^2}}$$

3D scale-space representation

$$\forall x \in f,\ \forall \sigma_{1..k}:$$

$$L(x,\sigma) = LOG(x;\sigma) * f(x)$$





Best features are at (x,σ) that maximize L

Lindeberg's Theory

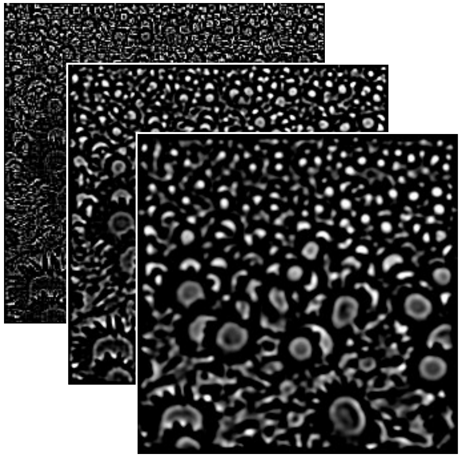
Multi-Scale Feature Selection Process

Original Image

$f(x)$

Convolve

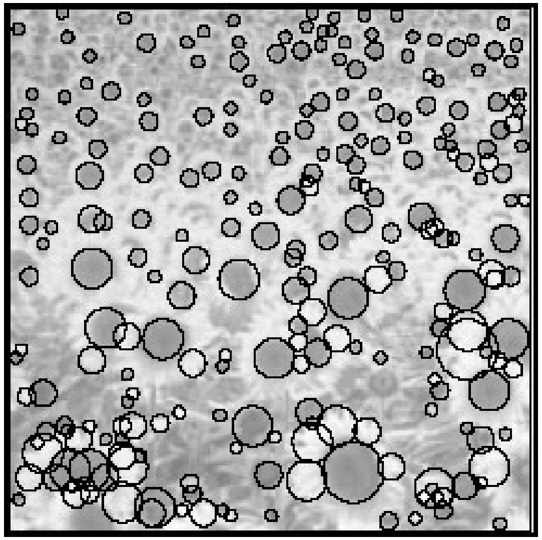
3D scale-space function



$$L(x,\sigma) = LOG(x;\sigma) \ast f(x)$$

Maximize

250 strongest responses
(Large circle = large scale)



Tracking Through Scale Space

Approximating LOG using DOG

$$LOG(x;\sigma) \approx DOG(x;\sigma) = G(x;\sigma) - G(x;1.6\sigma)$$

2D LOG filter with scale σ

2D DOG filter with scale σ

2D Gaussian with μ=0 and scale σ

2D Gaussian with μ=0 and scale 1.6σ

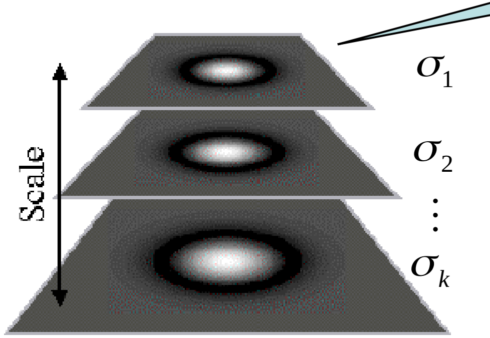
Why DOG?

- Gaussian pyramids are created faster
- Gaussian can be used as a mean-shift kernel

DOG filters at multiple scales

3D spatial kernel

$$K(x,\sigma) =$$



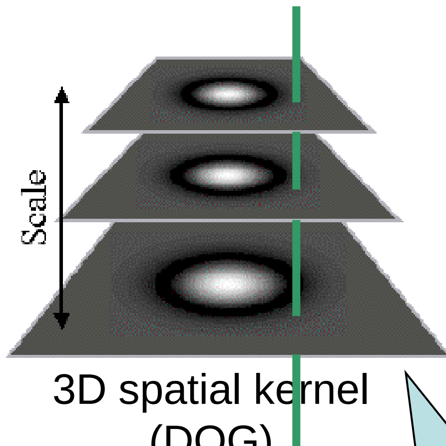
→ Scale-space filter bank

Tracking Through Scale Space

Using Lindeberg's Theory

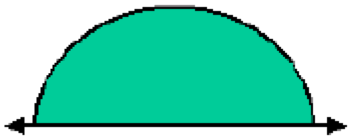
Weight image

$\ast$



3D spatial kernel (DOG)

$\ast$



1D scale kernel (Epanechnikov)

$= E(x, \sigma)$

3D scale-space representation

Centered at current location and scale

Modes are blobs in the scale-space neighborhood

Need a mean-shift procedure that finds local modes in $E(x,\sigma)$

Recall:

Model: $q = (q_1, \ldots, q_m)$ at y_0

Candidate: $p(y) = (p_1(y), \ldots, p_m(y))$

Color bin: $b(x)$

Pixel weight: $w(x) = \sqrt{\dfrac{q_{b(x)}}{p_{b(x)}(y_0)}}$

The likelihood that each candidate pixel belongs to the target

Tracking Through Scale Space

Example

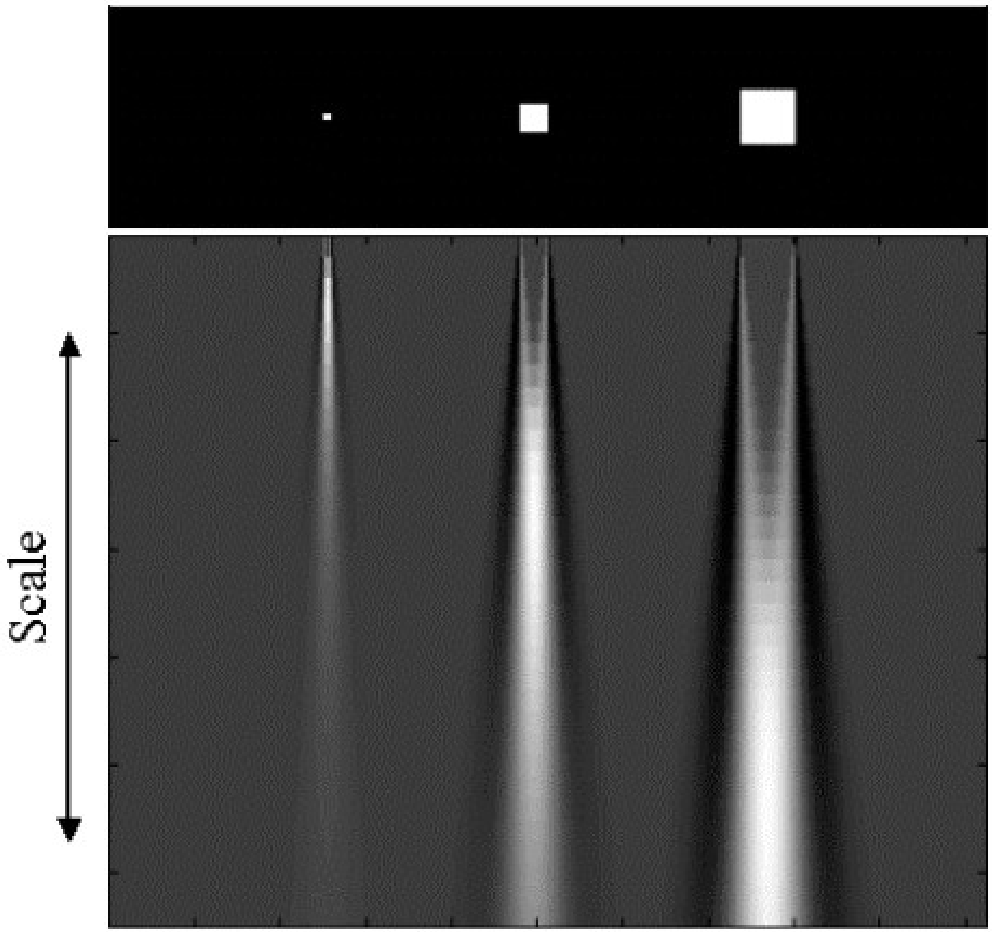


Image of 3 blobs

A slice through the 3D scale-space representation

Tracking Through Scale Space

Applying Mean-Shift

Use interleaved spatial/scale mean-shift

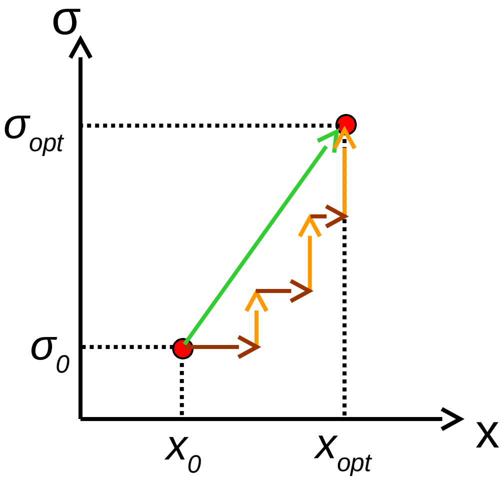
Spatial stage:

Fix σ and look for the best *x*

Scale stage:

Fix x and look for the best σ

Iterate stages until convergence of *x* and σ



Tracking Through Scale Space

Results

Fixed-scale

± 10% scale adaptation

Tracking through scale space

Thank You