Univerza v Ljubljani

Advanced CV methods
Mean Shift tracking

Matej Kristan

Laboratorij za Umetne Vizualne Spoznavne Sisteme,
Fakulteta za računalništvo in informatiko,
Univerza v Ljubljani

A toy-example – detector

- An imperfect detector says whether a selected region might contain a target or not.

“Detector” usually fires correctly, but sometimes incorrectly

But more often it fires correctly…

Observe: densely-populated areas contain a target with a high probability

Today’s topic:

Finding the “most probable” position.

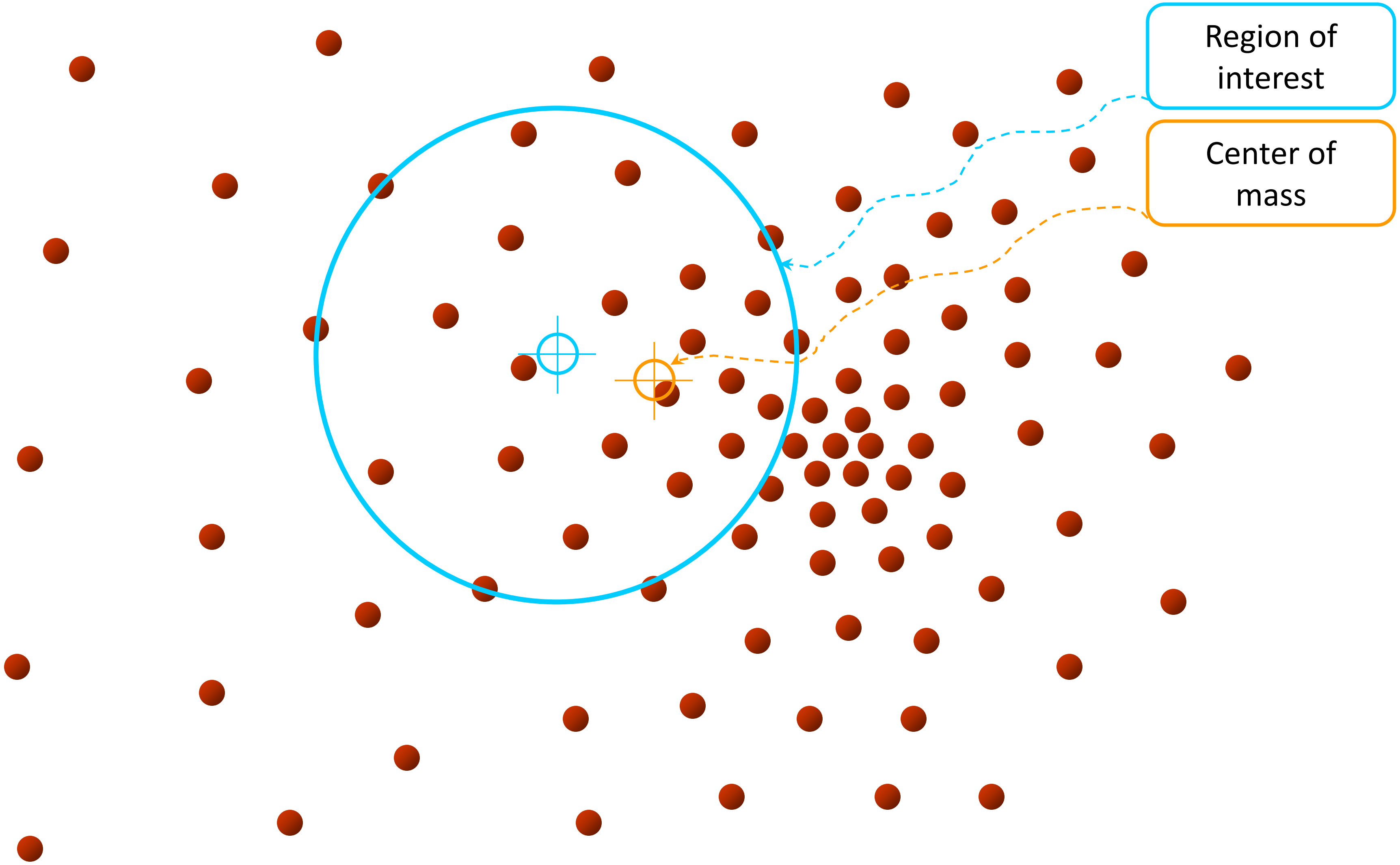
Outline

1. The theory behind the Mean Shift algorithm
2. Tracker based on the Mean Shift algorithm

Advanced Topics in Computer Vision

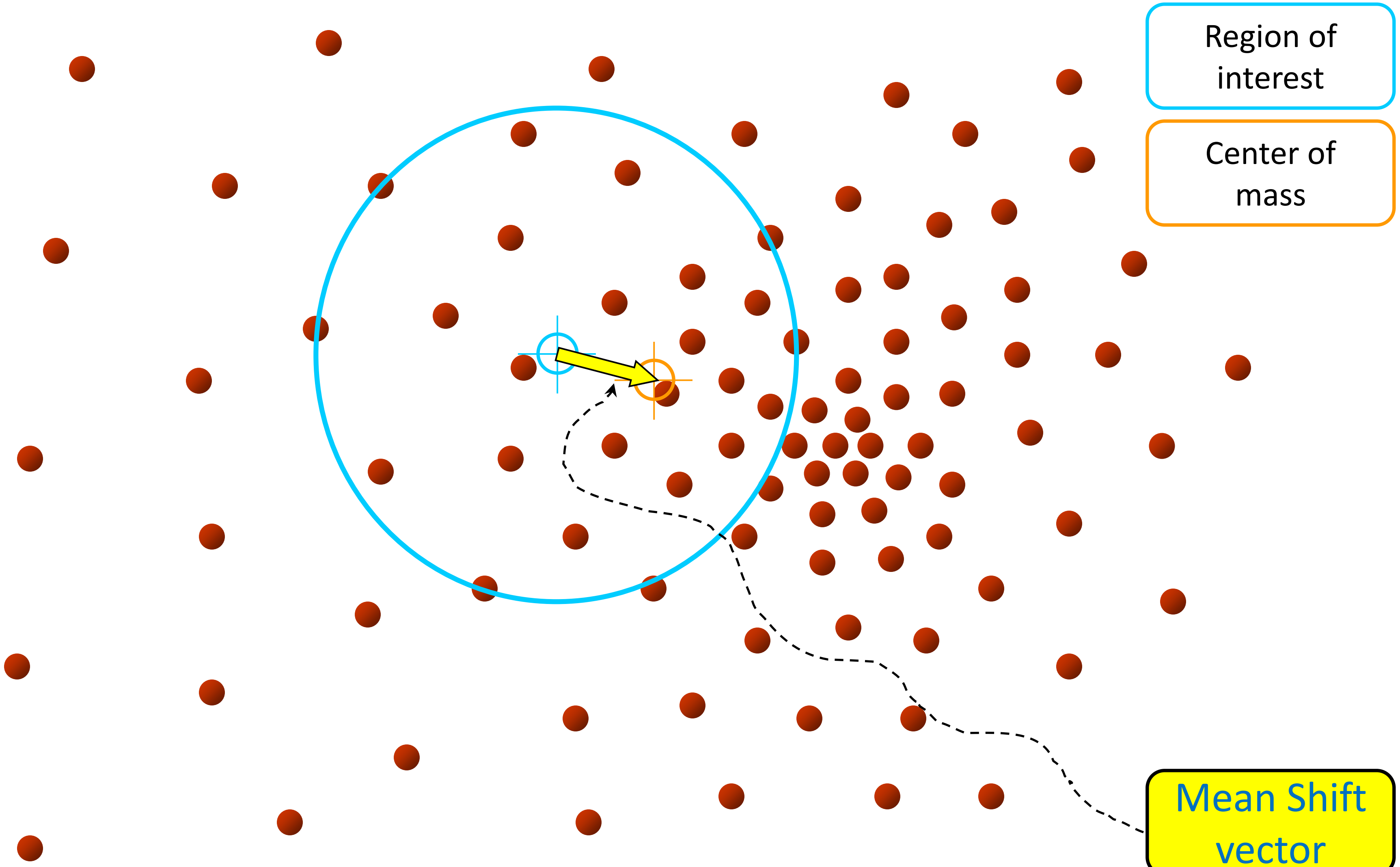
THE MEAN SHIFT THEORY

Intuitive description



Objective: Find the densest region
Distribution of “detections”

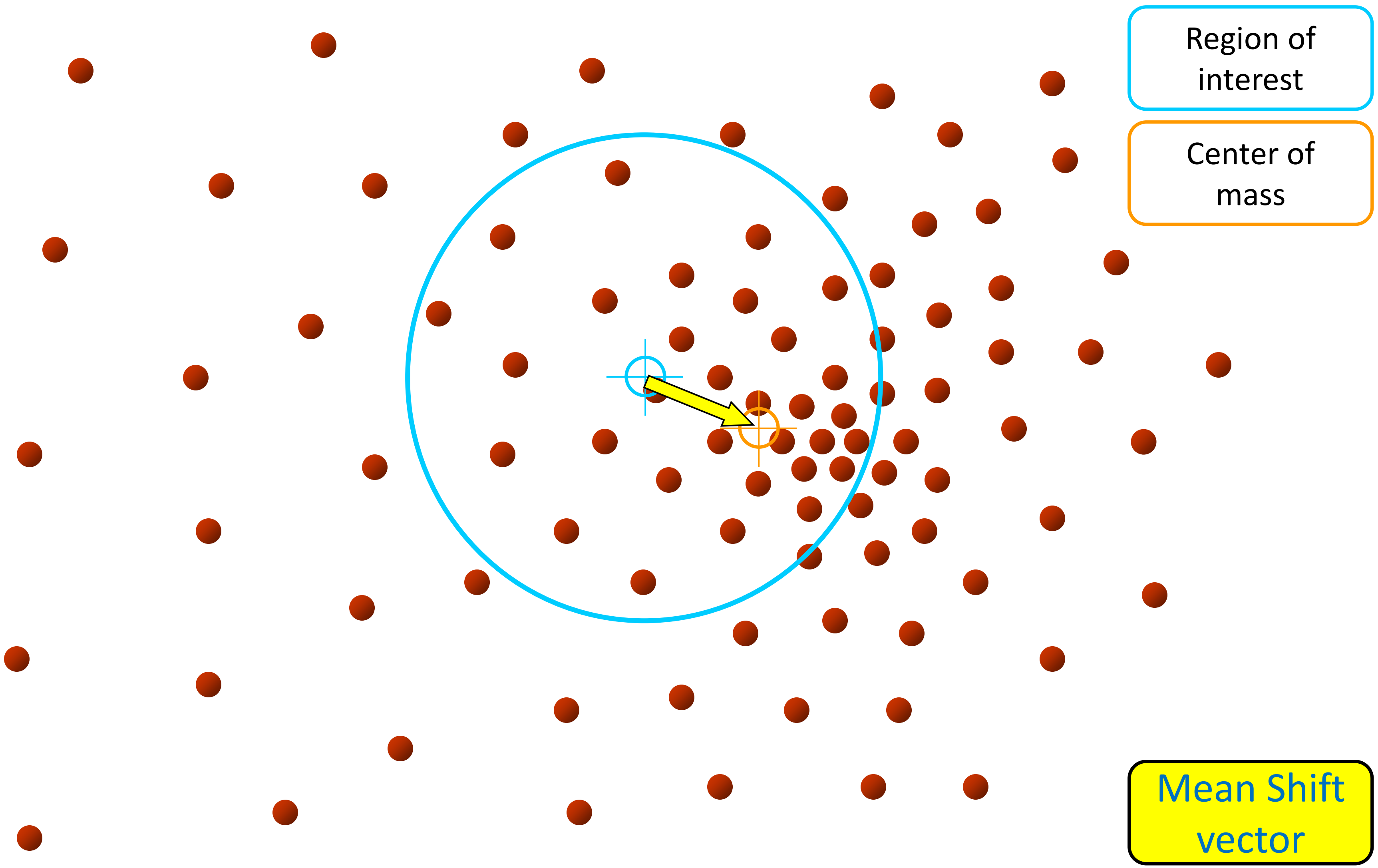
Intuitive description



Objective: Find the densest region

Distribution of “detections”

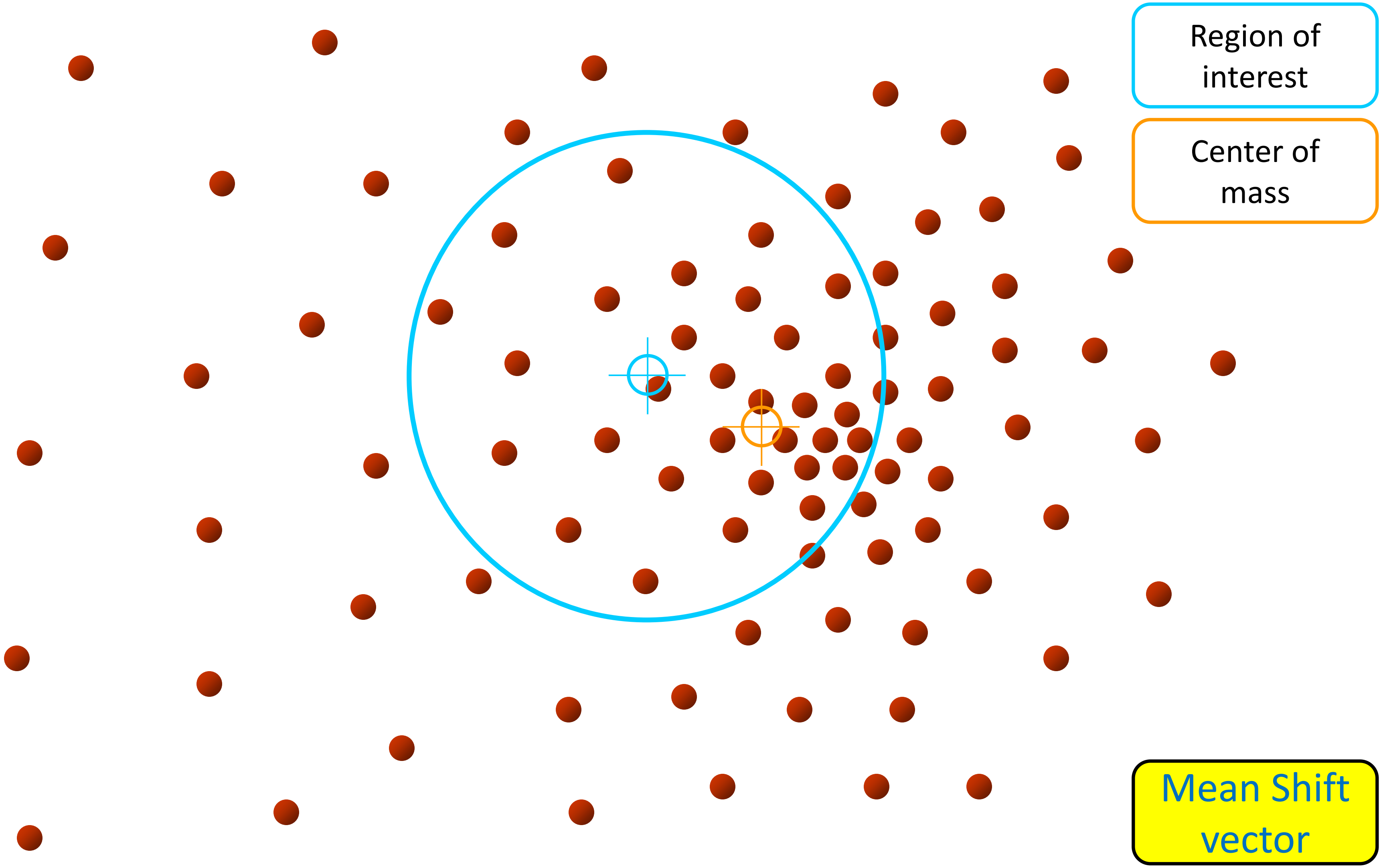
Intuitive description



Objective: Find the densest region

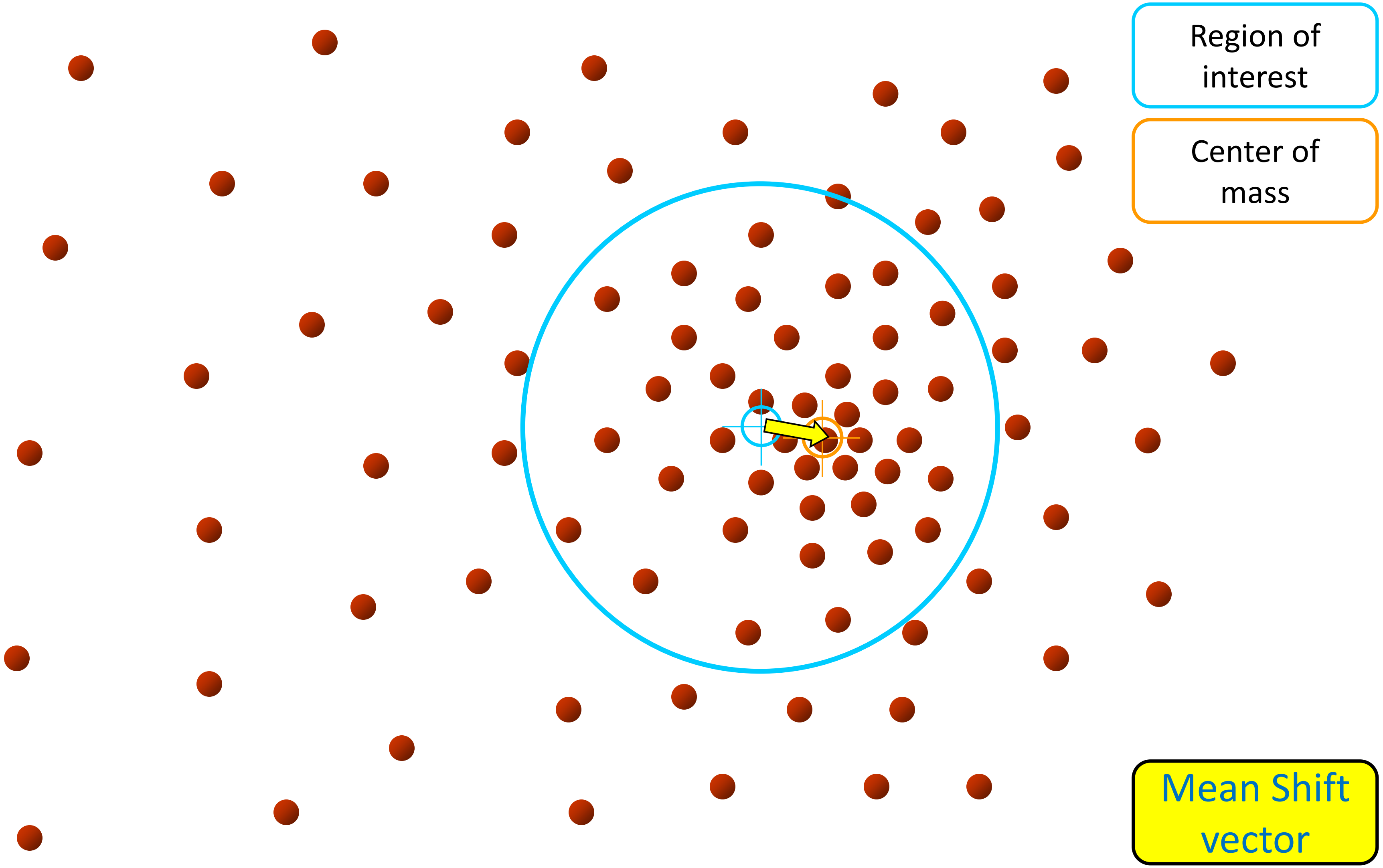
Distribution of “detections”

Intuitive description



Objective: Find the densest region
Distribution of "detections"

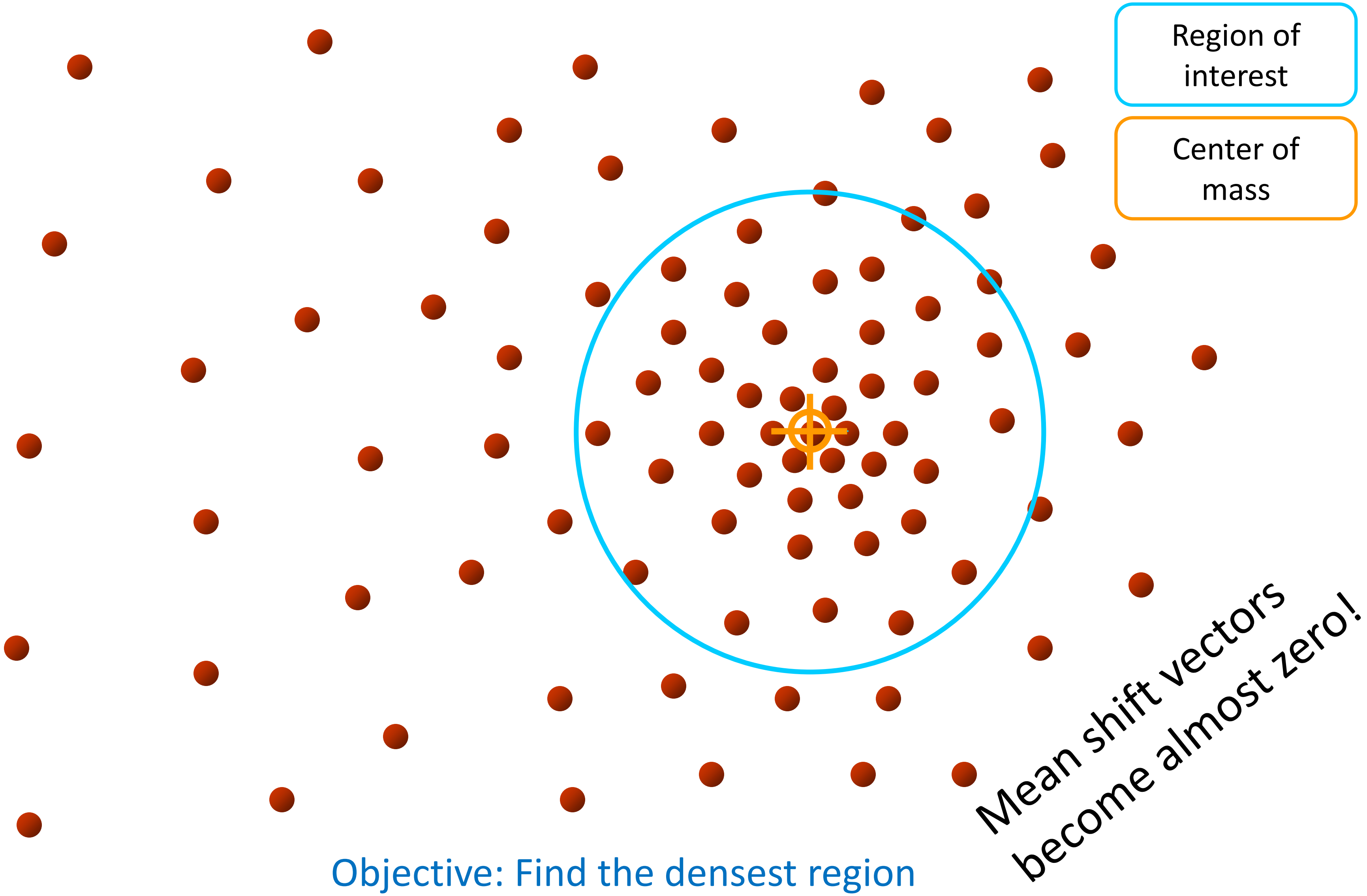
Intuitive description



Objective: Find the densest region

Distribution of “detections”

Intuitive description



Objective: Find the densest region
Distribution of "detections"

Mean shift in a nutshell

- Estimate mean: $x^{(k)}$

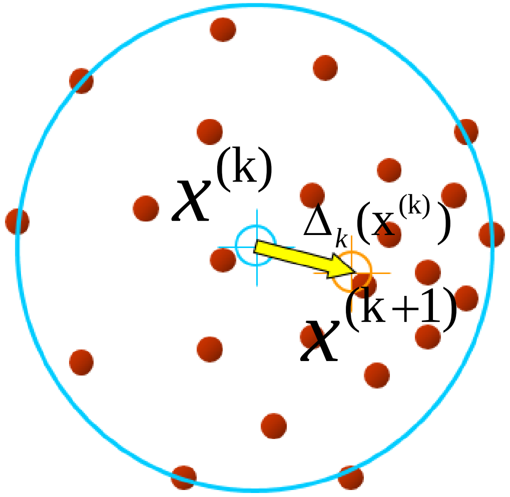
  Estimate the mean from the data in the neighborhood.

- Estimate the shift: $\Delta_k(x^{(k)})$

  Estimate the shift as the vector from the current mean to the estimated one.

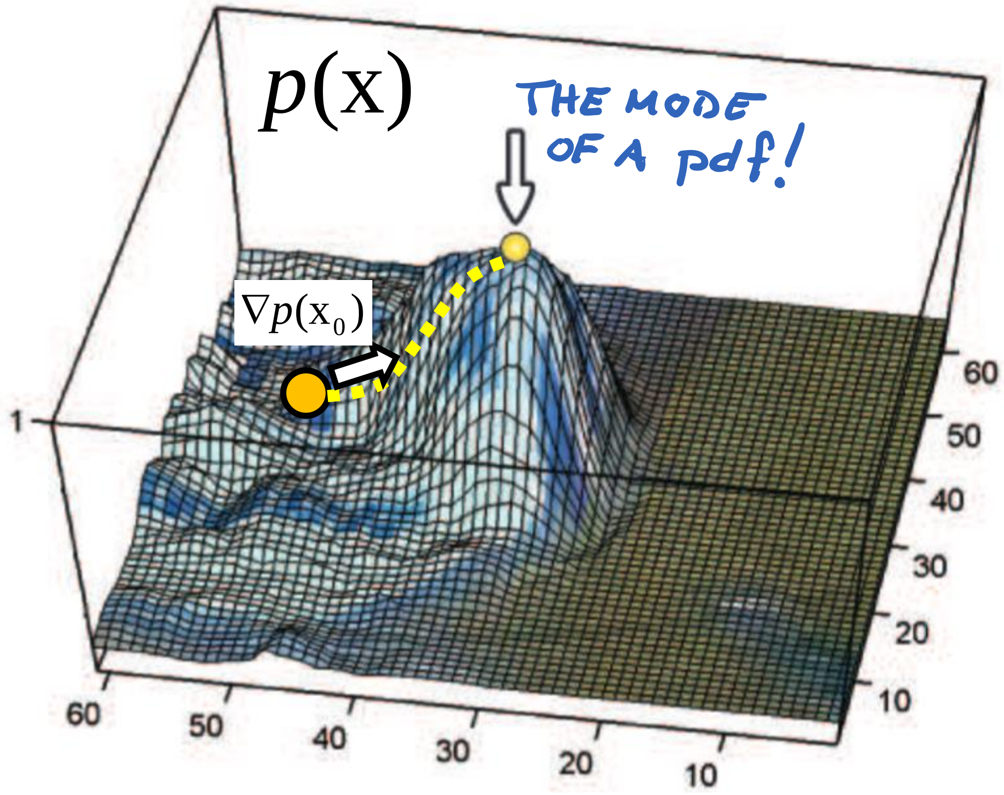
$$x^{(k+1)} = x^{(k)} + \Delta_k(x^{(k)})$$

The mean shift vector



What is a Mean Shift? (maths)

- A way to find the modes of a probability density functions (pdf) – *a gradient ascent on pdf!*



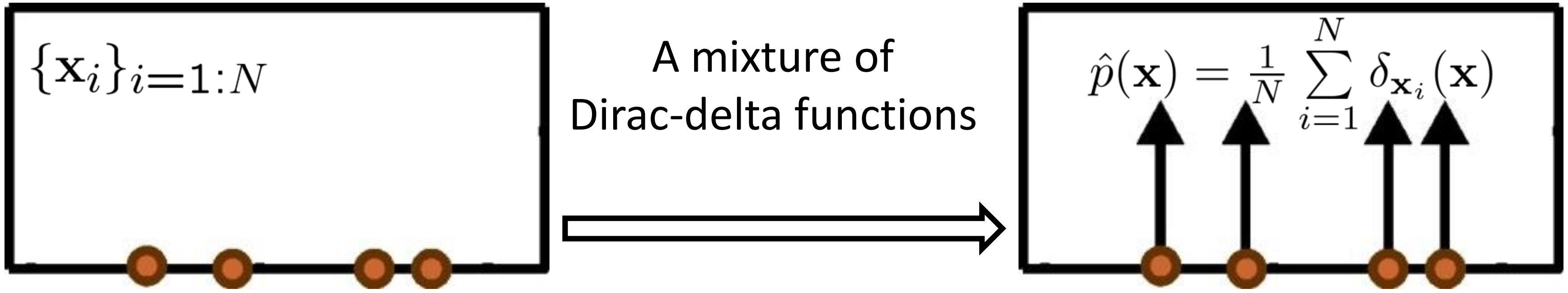
$p(\mathrm{x})$ ... a probability density function.

$\nabla p(\mathrm{x}_0)$ ... a gradient of a $p(x)$ evaluated at x_0.

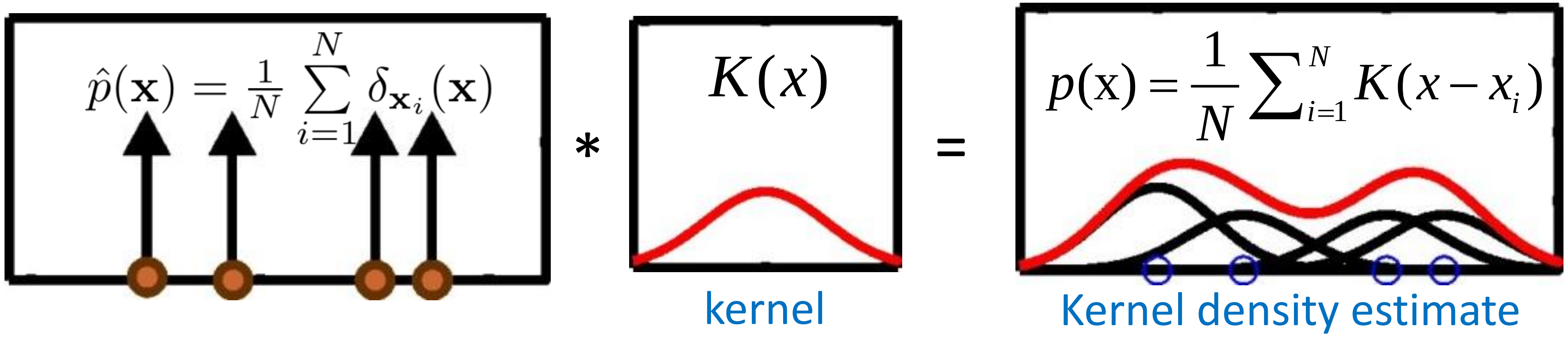
- We will apply it to nonparametric pdfs.

Kernel density estimation (KDE)

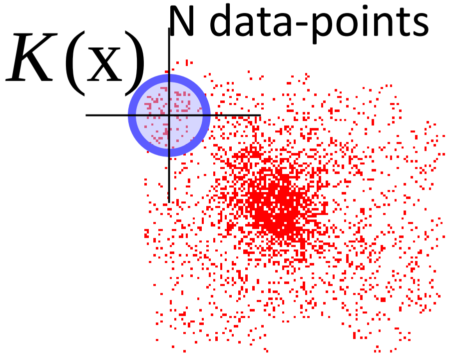
- The data samples are already an estimate of a pdf!

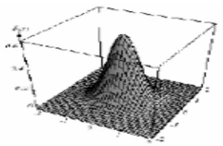


- Usually we assume a smooth pdf:

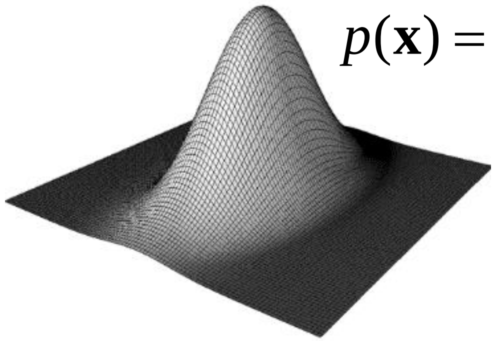


Kernel density estimation (KDE)





The KDE

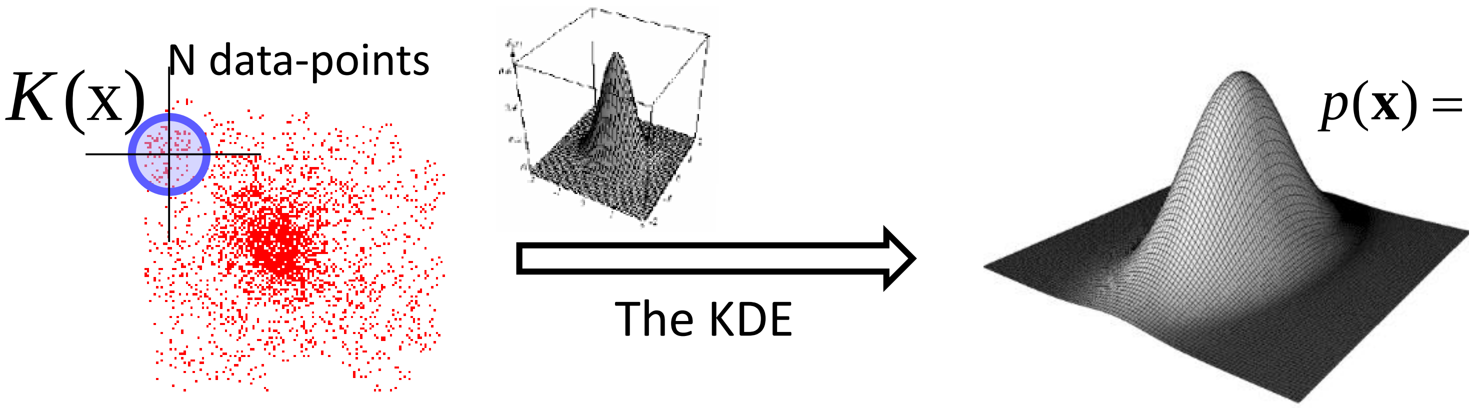


$$p(\mathbf{x}) = \frac{1}{N}\sum_{i=1}^{N} K(\mathbf{x} - \mathbf{x}_i)$$

Kernel Properties:

- Normalized $$\int_{R^d} K(\mathbf{x})d\mathbf{x} = 1$$
- Symmetric $$\int_{R^d} \mathbf{x}K(\mathbf{x})d\mathbf{x} = 0$$
- Homoscedastic $$\int_{R^d} \mathbf{x}\mathbf{x}^T K(\mathbf{x})d\mathbf{x} = c\mathbf{I}$$

Examples of kernels

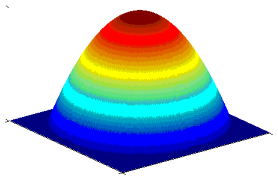


$$p(\mathbf{x}) = \frac{1}{N}\sum_{i=1}^{N} K(\mathbf{x} - \mathbf{x}_i)$$

Examples:

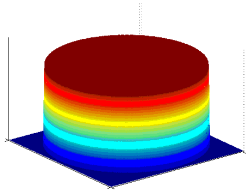
- Epanechnikov Kernel

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



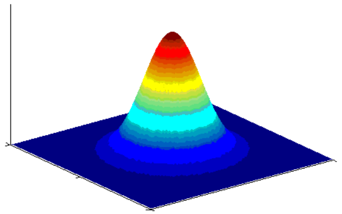
- Uniform Kernel

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



- Normal Kernel

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

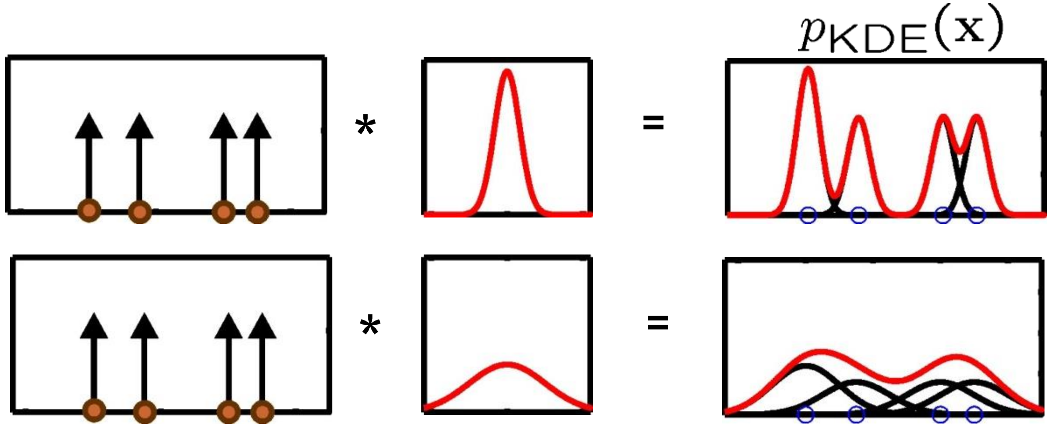


“Nonparametric” with parameter $\boldsymbol{h}$?

- A note on the kernel size – the bandwidth h

small h

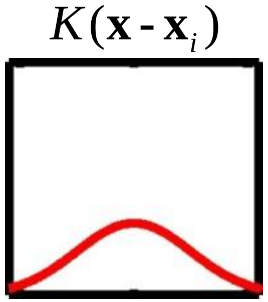
large h



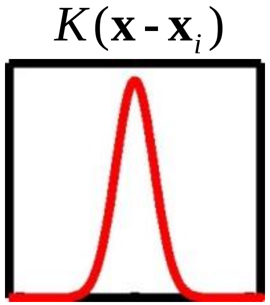
For advanced approaches for bandwidth estimation see: Kristan, et al., *Multivariate Online Kernel Density Estimation with Gaussian Kernels*, Pattern Recognition 2011

- We will use the following definition:

$$K(\mathbf{x}-\mathbf{x}_i) = c \cdot k\left(\left\| \frac{\mathbf{x}-\mathbf{x}_i}{\boxed{h}} \right\|^2 \right)$$



large h



small h

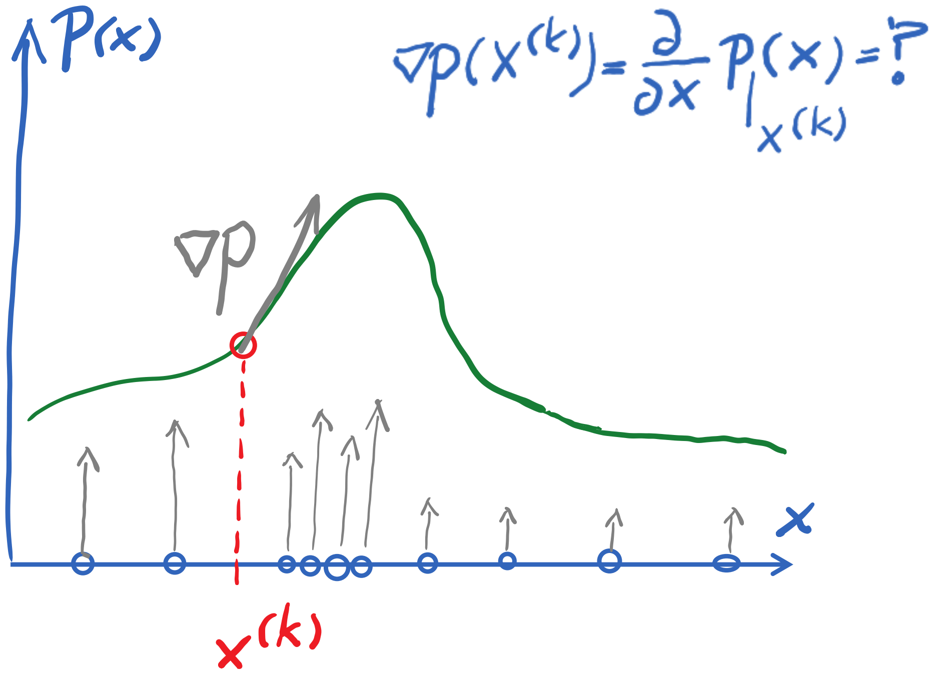
Gradient ascent on a KDE

- The KDE calculated from weighted data

$$p(\mathrm{x}) = \sum_{i=1}^{N} w_i K(\mathrm{x} - \mathrm{x}_i) \quad , \quad \sum_{i=1}^{N} w_i = 1 \, , \quad K(\mathrm{x} - \mathrm{x}_i) = \mathrm{ck}\left(\left\| \frac{x - x_i}{h} \right\|^2\right)$$

- Goal: Climb the mode!
- Approach:
  - Iteratively solve

$$\nabla p(\mathrm{x}^{(\mathrm{k})}) \equiv 0 \longrightarrow x^{(\mathrm{k}+1)}$$



Gradient ascent on a KDE

- The density model: $p(\mathrm{x}) = c\sum_{i=1}^{N} w_i \,\mathrm{k}\left(\left\|\frac{x-x_i}{h}\right\|^2\right)$
- The partial derivative (the gradient):

1. $$\nabla p(\mathrm{x}) = \frac{\partial}{\partial x} p(\mathrm{x}) = c\sum_{i=1}^{N} w_i \frac{\partial}{\partial x}\mathrm{k}\left(\left\|\frac{x-x_i}{h}\right\|^2\right)$$

2. $$\frac{\partial}{\partial x}\mathrm{k}\left(\left\|\frac{x-x_i}{h}\right\|^2\right) = -\frac{2}{h^2}(\mathrm{x}-\mathrm{x}_i)\,\mathrm{g}\left(\left\|\frac{x-x_i}{h}\right\|^2\right) \quad , \quad \text{where } g(\mathrm{r}) = -\mathrm{k}'(\mathrm{r})$$

3. $$\nabla p(\mathrm{x}) = \frac{2c}{h^2}\left[\sum_{i=1}^{N} w_i x_i g\left(\left\|\frac{x-x_i}{h}\right\|^2\right) - x\sum_{i=1}^{N} w_i g\left(\left\|\frac{x-x_i}{h}\right\|^2\right)\right]$$

Gradient ascent on a KDE

- Setting the partial derivative to zero $\frac{\partial}{\partial x} p(x) \equiv 0$ gives:

$$0 = \frac{2c}{h^2}\left[\sum_{i=1}^{N} w_i x_i g\left(\left\|\frac{x - x_i}{h}\right\|^2\right) - x\sum_{i=1}^{N} w_i g\left(\left\|\frac{x - x_i}{h}\right\|^2\right)\right]$$

- Expressing the x:

$$x = \frac{\sum_{i=1}^{N} x_i w_i g(\left\|\frac{x-x_i}{h}\right\|^2)}{\sum_{i=1}^{N} w_i g(\left\|\frac{x-x_i}{h}\right\|^2)}$$

Problem: *x is on the left-hand as well as the right-hand side.*

Solution: apply iterations.

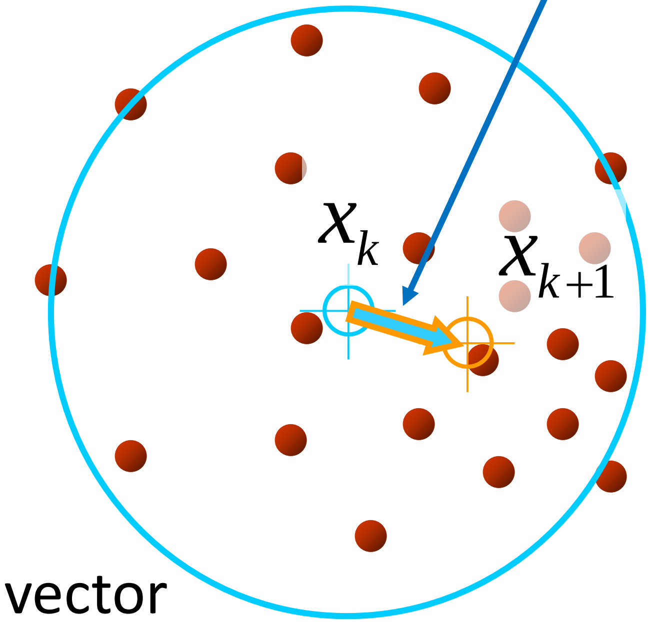
Gradient ascent on a KDE

- Iterative approach:
  - Plug $x^{(k)}$ to the right-hand side
  - Get a new estimate $x^{(k+1)}$

$$x^{(k+1)} = \frac{\sum_{i=1}^{N} x_i w_i g\left(\left\|\frac{x^{(k)} - x_i}{h}\right\|^2\right)}{\sum_{i=1}^{N} w_i g\left(\left\|\frac{x^{(k)} - x_i}{h}\right\|^2\right)}$$

You can show this at home!

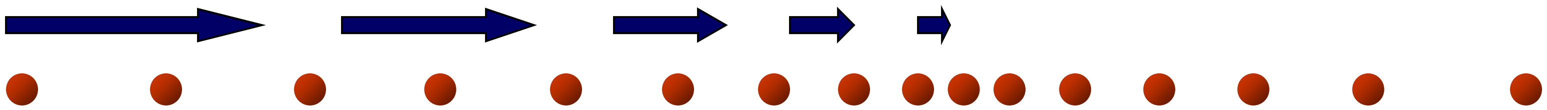
$$\frac{\partial}{\partial x} p(x) \propto m(x^{(k)})$$

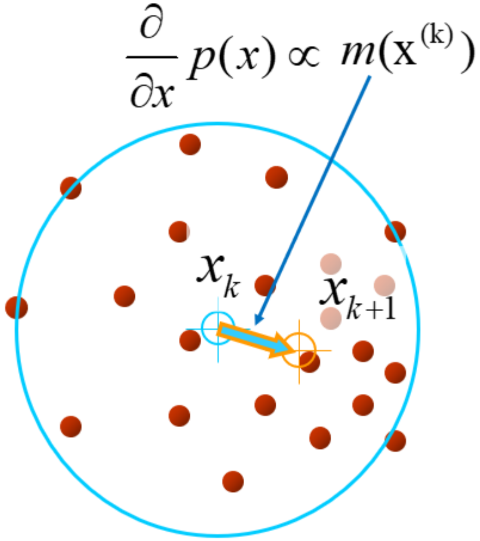


$m^{(k)} = x^{(k+1)} - x^{(k)}$ … The mean shift vector

The mean shift vector is proportional to the gradient on the pdf!

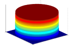
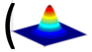
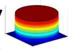
Mean Shift properties



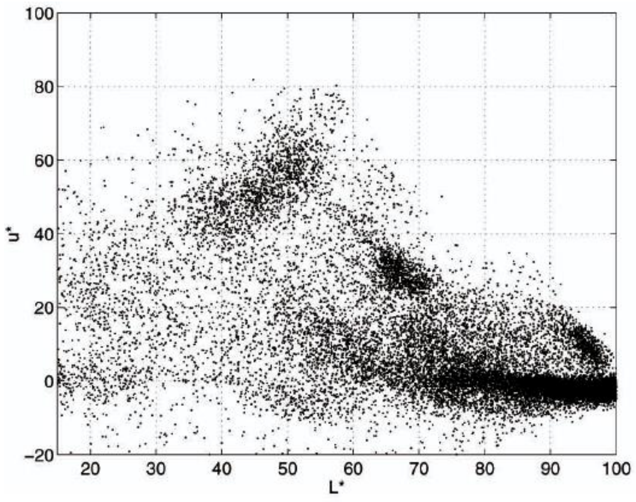


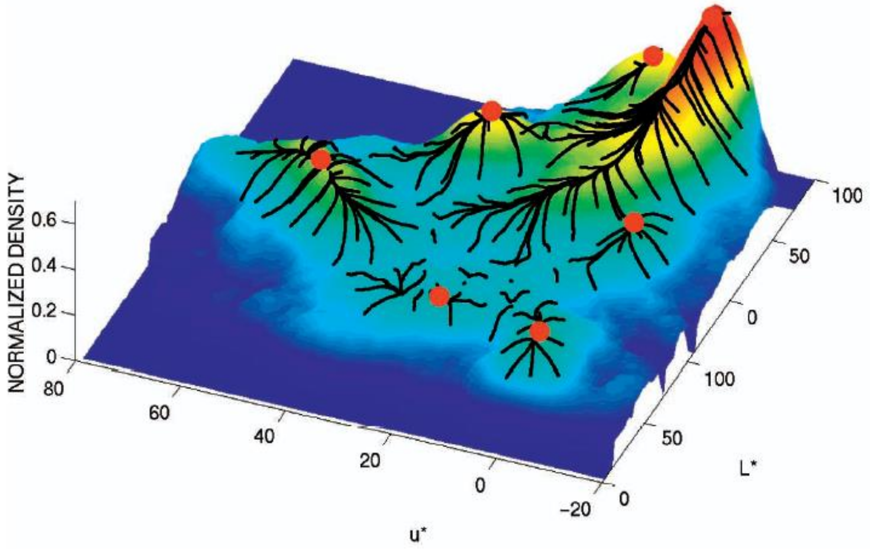
- Automatic convergence speed – the mean shift vector size depends on the gradient itself.
- Near maxima, the steps are small and refined

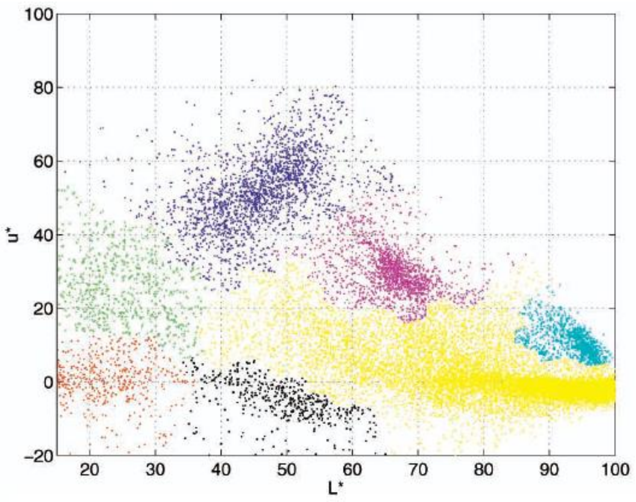
Adaptive Gradient Ascent

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

Mean-shift cluster discovery





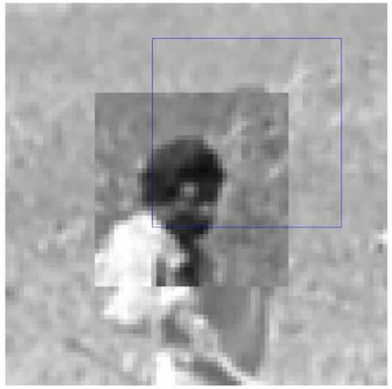


Previously at ACVM…

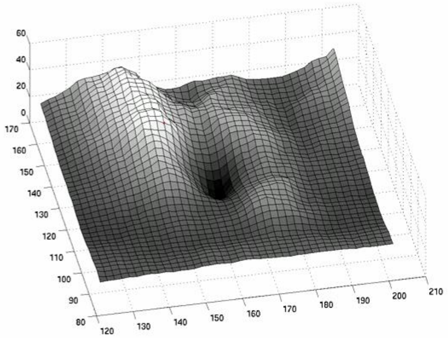
Patch tracking as incremental image registration

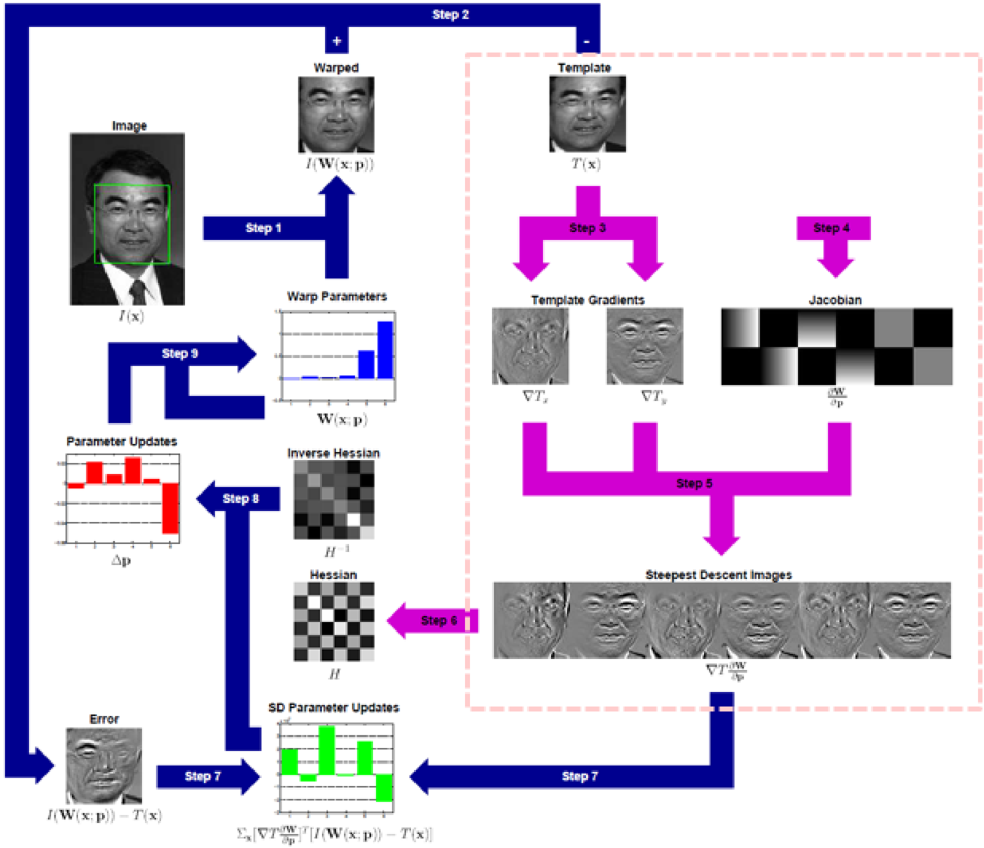
- Iteratively improve warp parameters to match template $T(x)$

$$E(\Delta p) = \sum_{x} \left(I(W(x; p + \Delta p)) - T(x) \right)^2$$



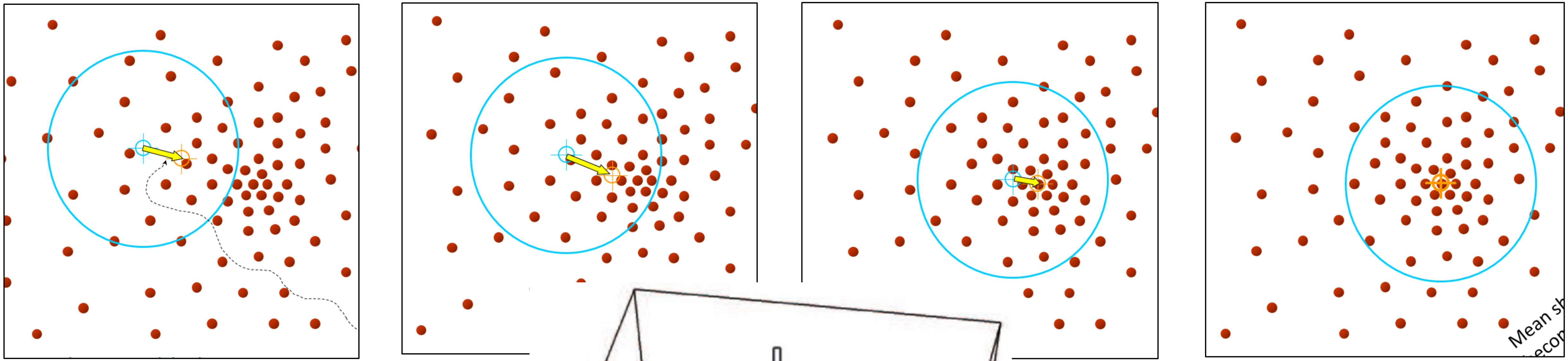
$E(p)$

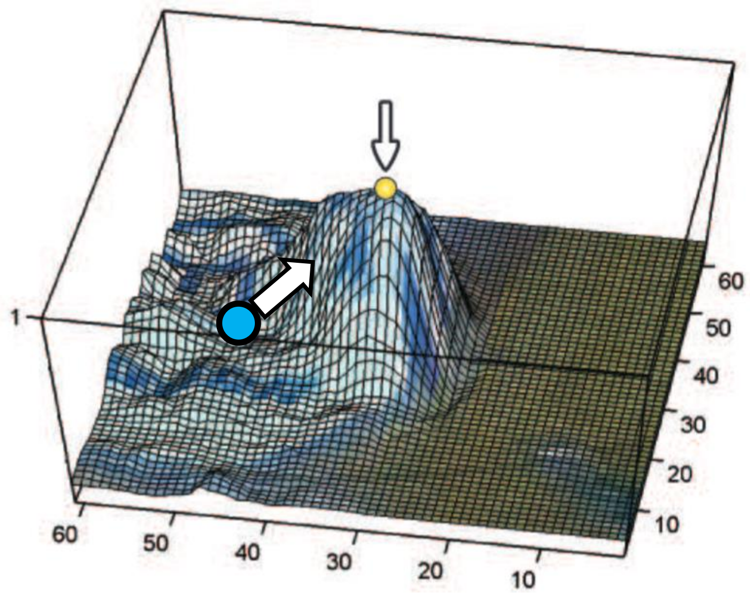




Previously at ACVM…

- Mode detection by Mean Shift:

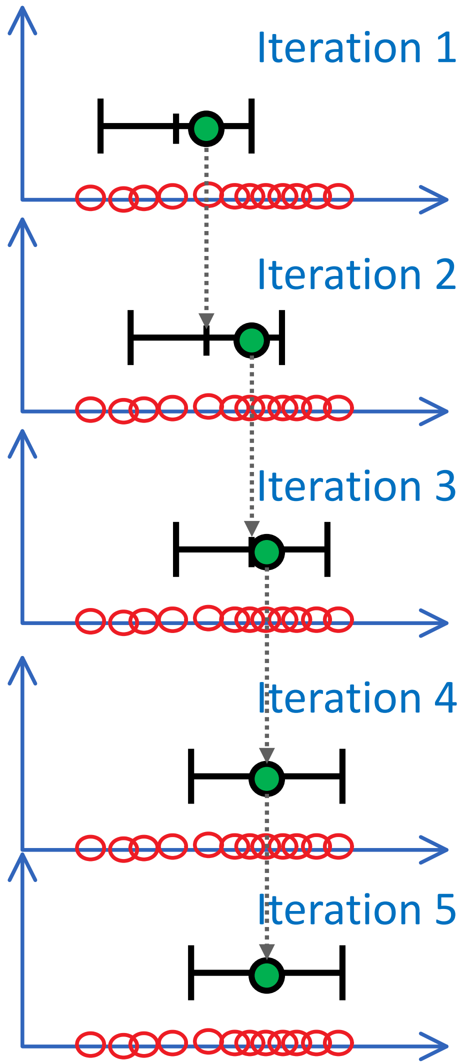




$$x^{(k+1)} = \frac{\sum_{i=1}^{n} x_i}{n}$$

Mean Shift == gradient ascent

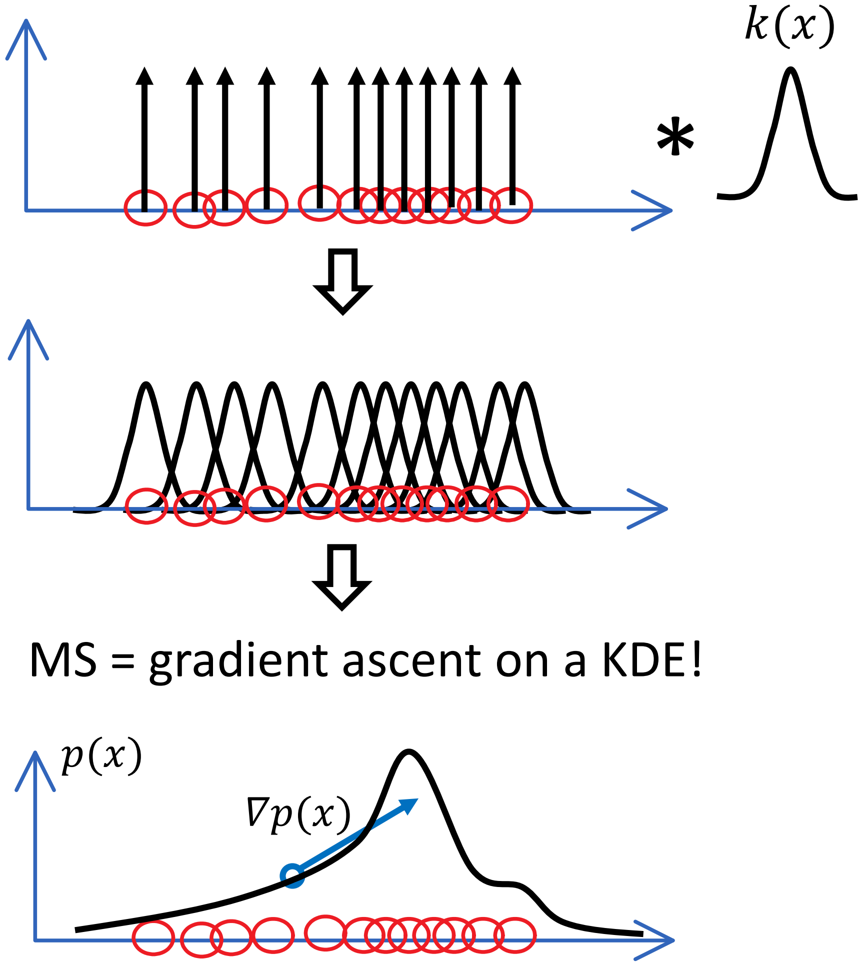
- Mean Shift: Iterative approach to finding densely populated regions



$$x^{(k+1)} = \frac{\sum_{i=1}^{N} x_i w_i g\left(\left\|\frac{x^{(k)} - x_i}{h}\right\|^2\right)}{\sum_{i=1}^{N} w_i g\left(\left\|\frac{x^{(k)} - x_i}{h}\right\|^2\right)}$$

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

Nonparametric pdf: KDE



Advanced Computer Vision Methods

MEAN SHIFT TRACKER

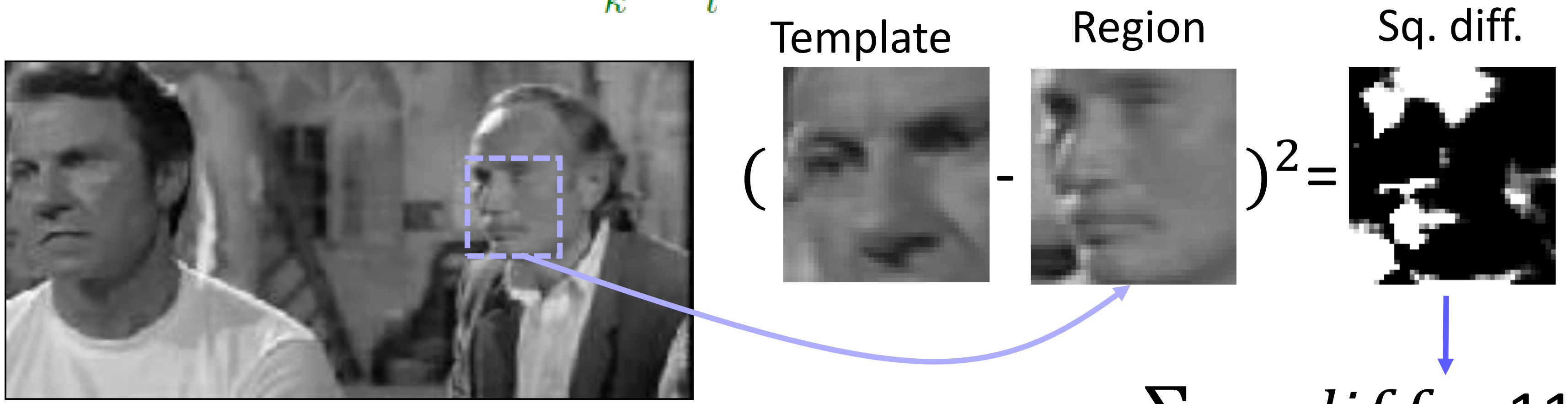
Mean Shift tracking example

- Tracking using color!

Recall the similarity measure in LK

- Quantify the similarity between the visual model and the target region
- Sum of squared differences

$$ssd(x,y) = \sum_k \sum_l (T(k,l) - I(x+k, y+l))^2$$

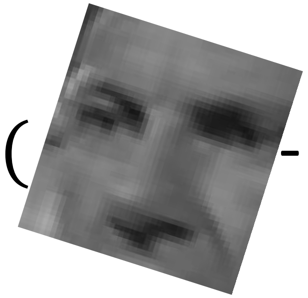


$\sum sq.diff$ =113563

Problems with SSD

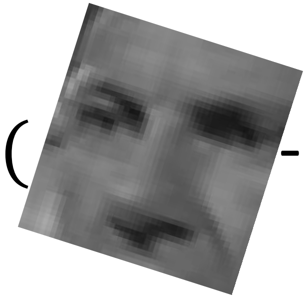
- Assume we are interested only in position and size
- What happens when the object slightly rotates?

Template Region

( - )^2= Large number

(...good)

Template Region

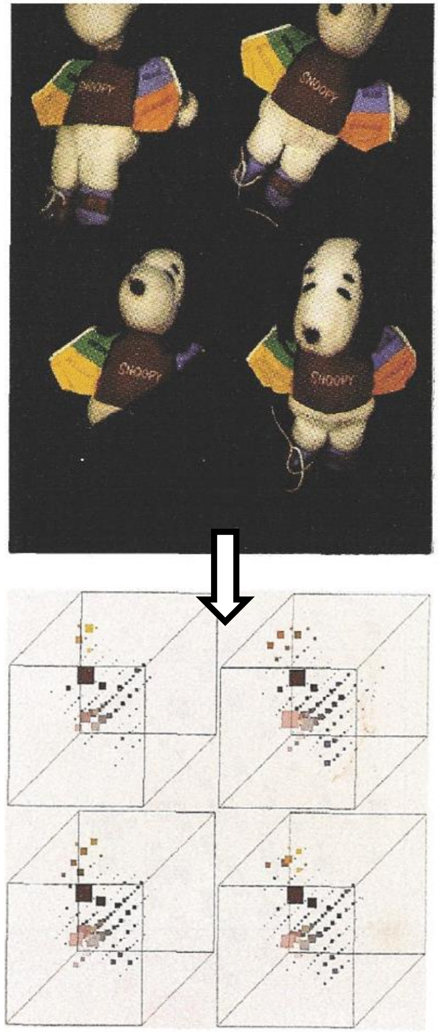
( - )^2= Large number as well

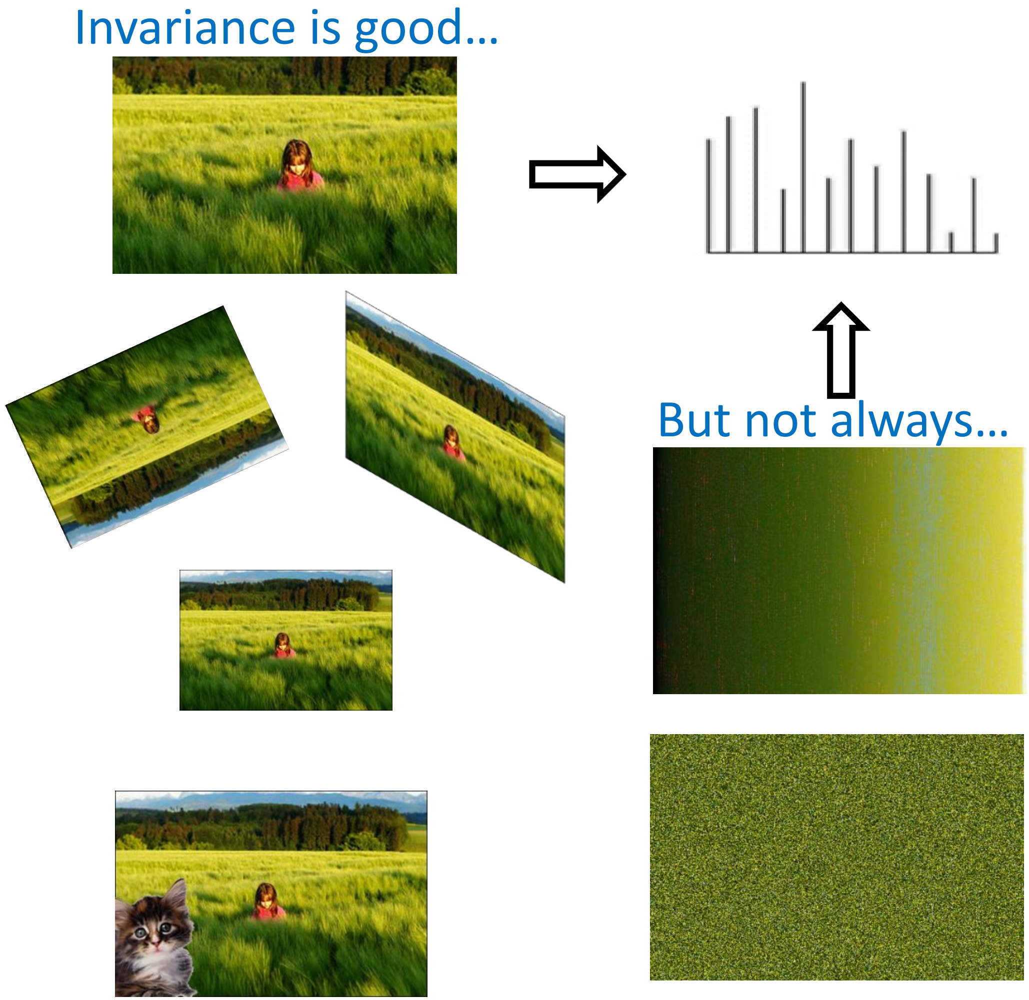
(...bad)

Require a visual model
that is invariant to rotation...

Color histograms

- Invariant to rotation, scale, partial occlusion, etc.



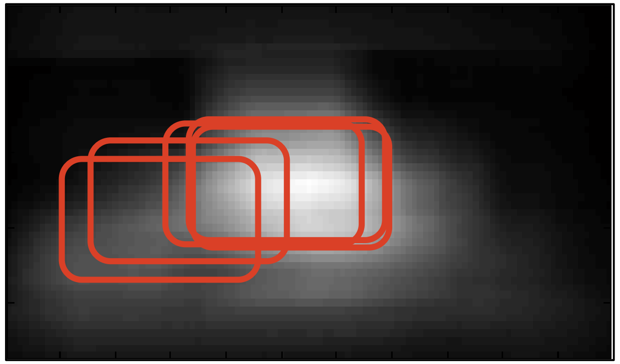


Mean Shift tracking: Intuition

- A highly cited paper[1]

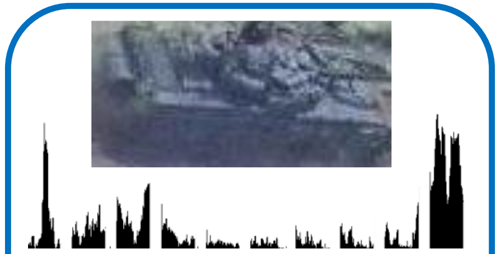
Start at previous estimate

Similarity to template



New estimate

Visual model



Color histograms, edge histograms, etc.

1. *“Could calculate”* for each window the similarity to the visual model.
2. Move locally (within window) to position with max similarity.

(NOTE: it’s not *really* done like that!
It’s done WITHOUT directly computing similarity!)

[1]*Mean Shift : A robust Approach Toward Feature Space Analysis, by Comaniciu, Meer, TPAMI, 2002*

Target representation – histograms

Choose a reference target region

Choose a feature space

Represent the target image by its PDF in the feature space

$$q = \{q_u\}_{u=1..m}$$

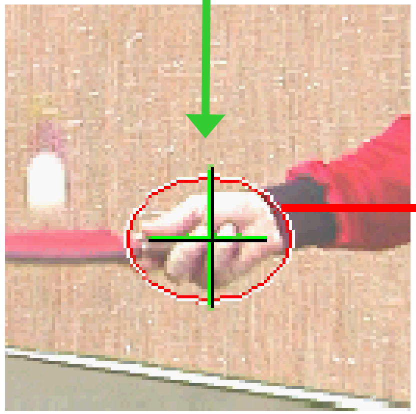
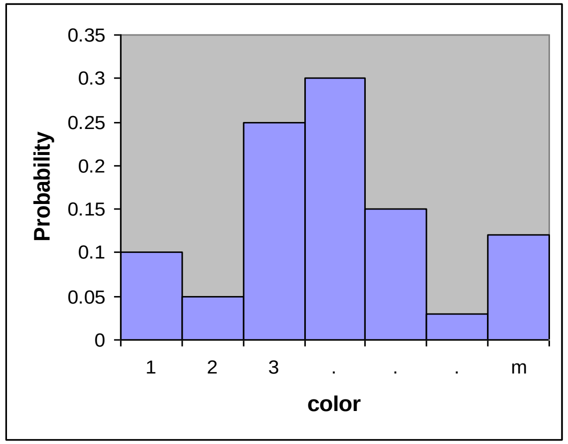


Image: I

Quantize the Color Space into m colors

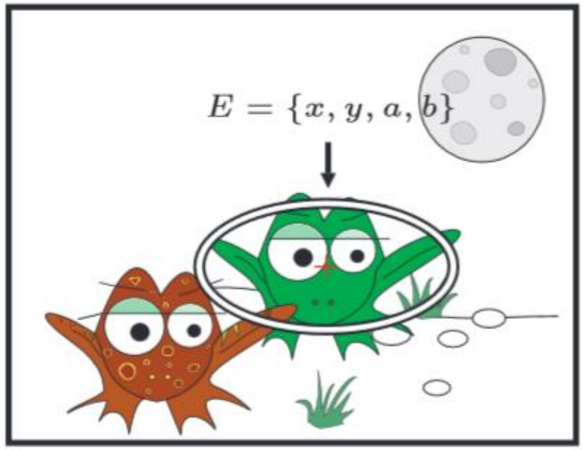
$$b = \text{floor}[I * \frac{m-1}{255}]$$

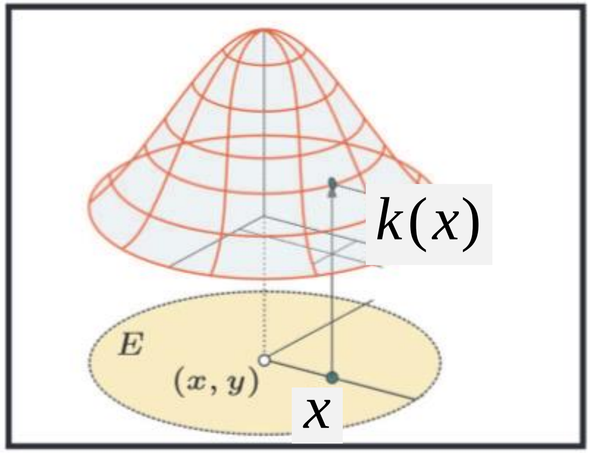


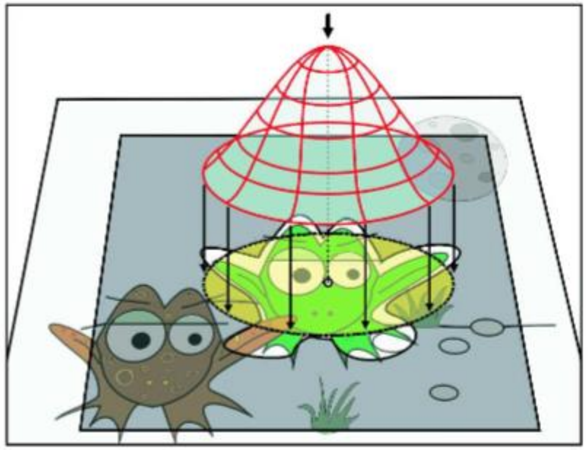
$$\sum_{u=1}^{m} q_u = 1$$

A weighted visual model

- Assign higher weights to the pixel colors closer to center



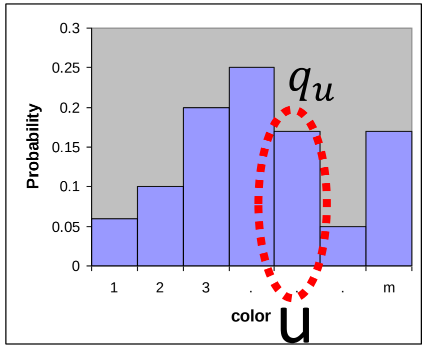




$\{x_i\}_{i=1..n}$ ... Target pixel locations

$k(x)$ ... Smooth, decreasing kernel

$u_i = b(x_i)$ ... Color bin index (u={1...m}) of pixel x_i



$q = \{q_u\}_{u=1..m}$

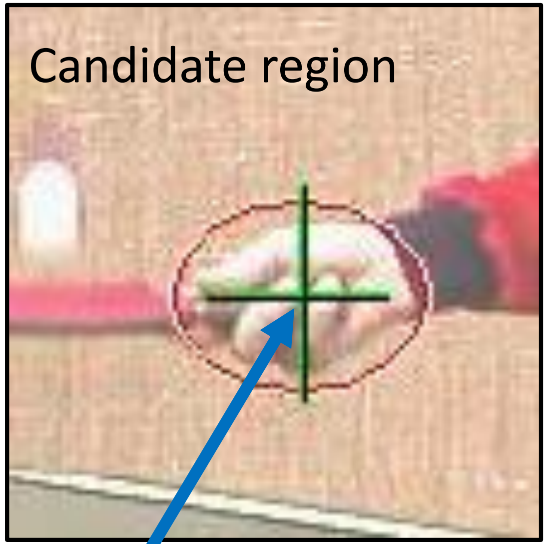
$$q_u = C\sum_{i=1}^{N} k\left(\|x_i\|^2\right)\delta_u(b(x_i))$$

Pixel weight

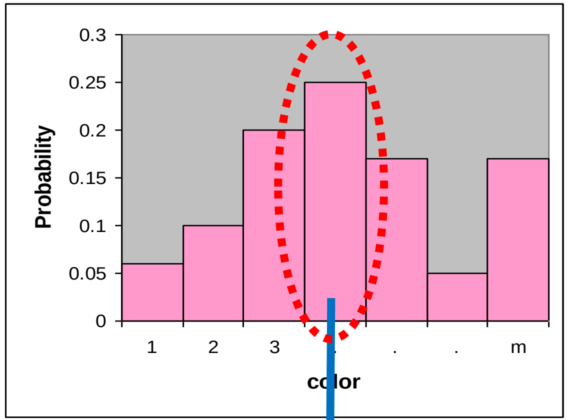
Normalization constant: $C = \sum_i k(\|x_i\|^2)$

The target “candidate”

- Want to check whether this region contains the target
- We use the same kernel, but with different bandwidth *h*



x



$$p(\mathrm{x}) = \{ p_u(\mathrm{x}) \}_{u=1:m}$$

Probability of feature *u* in candidate

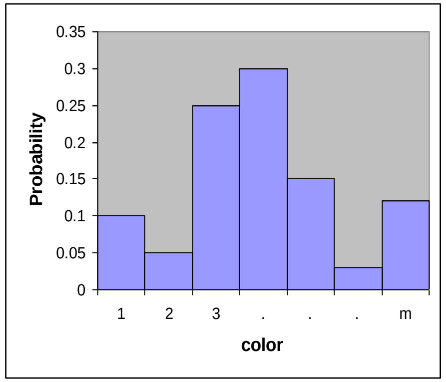
$$p_u(x) = C_h \sum_{i=1}^{N} \underbrace{k\left(\left\| \frac{x - x_i}{h} \right\|^2 \right)}_{\text{Pixel weighting}} \delta_u(\mathrm{b}(\mathrm{x}_i))$$

Normalization factor

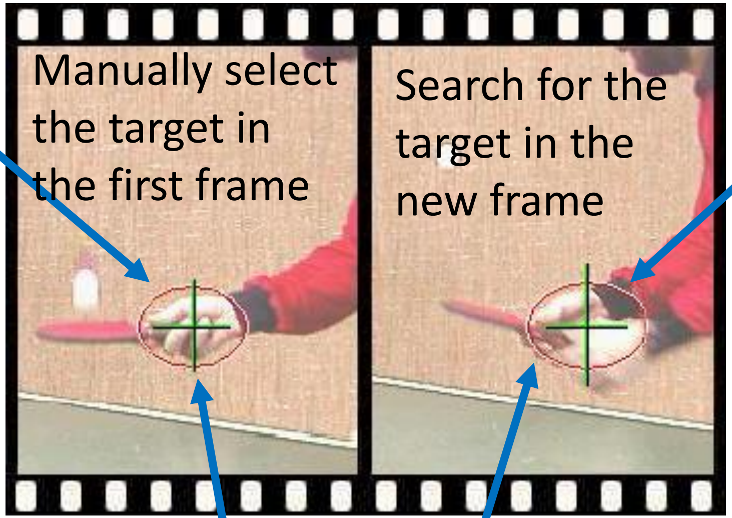
Pixel weighting

Histogram similarity measure

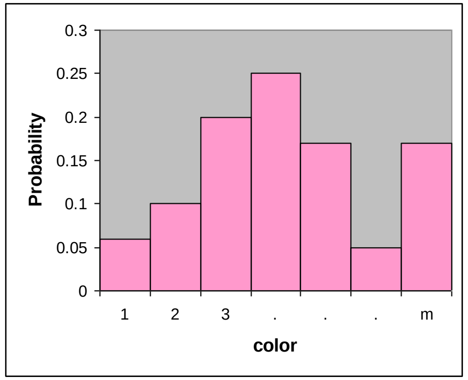
Target Model



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



Target Candidate
(centered at x)



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

Similarity function: $\rho(x) = \rho\left[q, p(x)\right]$

Similarity measure for histograms

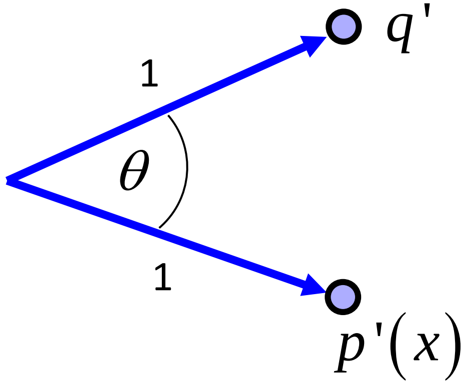
- The Bhattacharyya measure (related to Hellinger distance)
  - Similarity between distributions *q* and *p*

$$q' = q^{\frac{1}{2}} = \left(\sqrt{q_1}, \ldots, \sqrt{q_m}\right)$$

$$p'(x) = p^{\frac{1}{2}} = \left(\sqrt{p_1(x)}, \ldots, \sqrt{p_m(x)}\right)$$

$$\rho(x) = p'(x)^T q' = \cos\theta$$

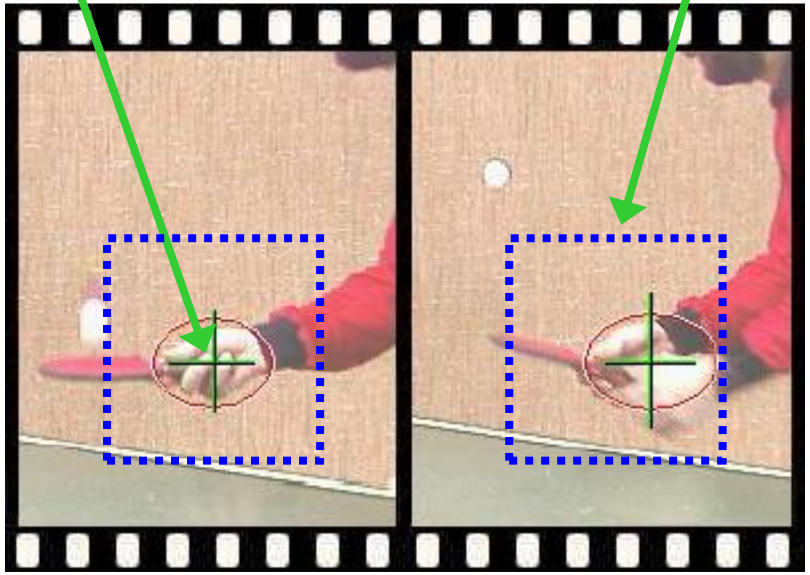
$$\rho(x) = \sum_{u=1}^{m} \sqrt{p_u(x) q_u}$$



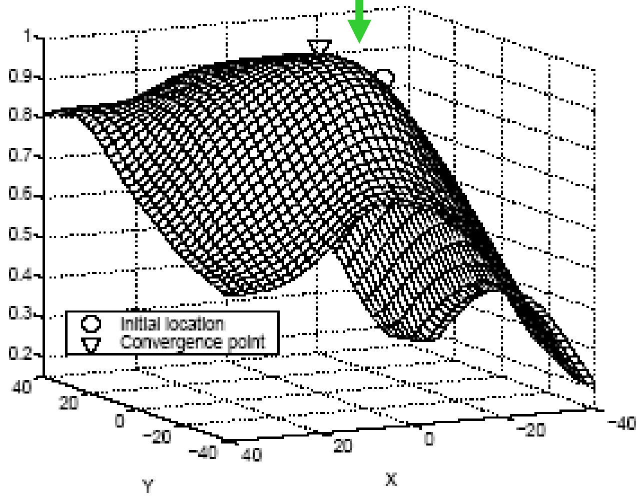
- Note: The similarity function $\rho(x)$ will be spatially smooth since the histograms are extracted by a spatially smooth kernel!

Localization by histogram similarity

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 histogram similarity



q $\quad\quad p(x)$



$$\rho(\mathrm{x}) \equiv \left(p(x), q \right)$$

The catch: how to perform localization quickly?

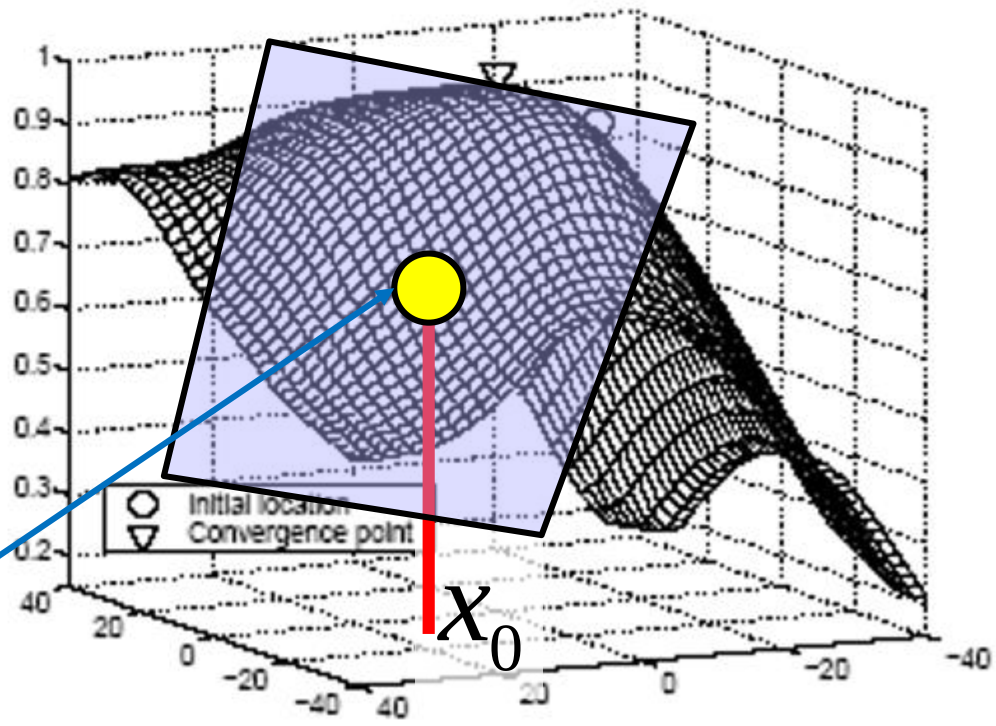
Gradient ascent on similarity

- Iterative approach to maximization
- Start at some x_0, estimate gradient, move to x_1

Apporach:

1. Linearize $\rho(x)$ at $p(x_0)$!
2. Then maximize the linearized version w.r.t. the position $\boldsymbol{x}$.

$\rho\left(p\left(x\right),q\right)$



$\rho\left(p\left(x_0\right),q\right)$

Linearization of similarity function

- Linearize $\rho(p(x_0) + \delta, q)$ at $p(x_0)$:

$$\rho(\mathrm{p}(\mathrm{x}_0) + \delta, \mathrm{q}) = \rho(\mathrm{p}(\mathrm{x}_0), \mathrm{q}) + \nabla \rho_{\mathrm{x}_0}^T \delta$$

- Reparameterize: $\mathrm{p}(\mathrm{x}) = \mathrm{p}(\mathrm{x}_0) + \delta$

$$\rho(\mathrm{p}(\mathrm{x}), \mathrm{q}) = \rho(\mathrm{p}(\mathrm{x}_0), \mathrm{q}) - \nabla \rho_{\mathrm{x}_0}^T \mathrm{p}(\mathrm{x}_0) + \nabla \rho_{\mathrm{x}_0}^T \mathrm{p}(\mathrm{x})$$

does not depend on ***x***!

- Can maximize $\rho(p(x), q)$ by only considering the last term, i.e.,:

$$x^* = \arg\max_x \rho(\mathrm{p}(\mathrm{x}_0), \mathrm{q}) = \arg\max_x \boxed{\nabla \rho_{x_0}^T p(\mathrm{x})}$$

Let's calculate this term

Maximization of $\nabla \rho_{x_0}^T p(\mathrm{x})$

- This is our cost function: $E(\mathrm{x}) = \nabla \rho_{x_0}^T p(\mathrm{x})$ $\qquad \rho(x) = \sum_{u=1}^{m} \sqrt{p_u(x) q_u}$

$$\boldsymbol{p} = [p_1, p_2, \dots, p_u, \dots, p_m]^T$$

$$\nabla \rho_{x_0}^T = \frac{\partial}{\partial \mathbf{p}} \left(\sum_{u=1}^{m} p_u^{\frac{1}{2}}(\mathrm{x}_0) \mathrm{q}_u^{\frac{1}{2}} \right) = \frac{1}{2} \left[\sqrt{\frac{\mathrm{q}_1}{p_1(\mathrm{x}_0)}}, \dots, \sqrt{\frac{\mathrm{q}_u}{p_u(\mathrm{x}_0)}}, \dots, \sqrt{\frac{\mathrm{q}_m}{p_m(\mathrm{x}_0)}} \right]^T$$

- Plugging the gradient $\nabla \rho_{x_0}^T$ in the cost function gives

$$E(\mathrm{x}) = \nabla \rho_{x_0}^T p(\mathrm{x}) = \boxed{\frac{1}{2} \sum_{u=1}^{m} p_u(\mathrm{x}) \sqrt{\frac{\mathrm{q}_u}{p_u(\mathrm{x}_0)}}}$$

This is what we want to maximize w.r.t. ***x***!

Maximization of $\nabla \rho_{x_0}^T p(\mathrm{x})$

- Cost function: $\mathrm{E}(\mathrm{x}) = \nabla \rho_{x_0}^T p(\mathrm{x}) = \frac{1}{2} \sum_{u=1}^{m} p_u(\mathrm{x}) \sqrt{\frac{\mathrm{q}_u}{p_u(\mathrm{x}_0)}}$
- Recall definition: $p_u(x) = C_h \sum_{i=1}^{N} k\left(\left\| \frac{x - x_i}{h} \right\|^2 \right) \delta_u(\mathrm{b}(\mathrm{x}_i))$
- With some manipulation, we can rewrite the cost:

$$\mathrm{E}(\mathrm{x}) = \frac{1}{2} C_h \sum_{i=1}^{N} w_i k\left(\left\| \frac{x - x_i}{h} \right\|^2 \right) \qquad w_i = \sqrt{\frac{\mathrm{q}_{b(\mathrm{x}_i)}}{p_{b(\mathrm{x}_i)}(\mathrm{x}_0)}}$$

$$x^* = \arg\max_x E(\mathrm{x})$$

Note: *$E(x)$ is a KDE and we can find the mode by applying Mean Shift iterations!*

Maximization by Mean Shift

- This is the rewritten cost function:

$$E(\mathrm{x}) = \frac{1}{2} C_h \sum_{i=1}^{N} w_i k\left(\left\| \frac{x - x_i}{h} \right\|^2 \right) \qquad w_i = \sqrt{\frac{\mathrm{q}_{b(\mathrm{x}_i)}}{p_{b(\mathrm{x}_i)}(\mathrm{x}_0)}}$$

- Apply Mean Shift iterations:

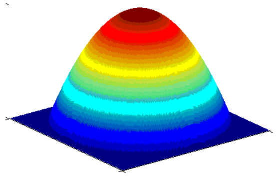
$$x^{(\mathrm{k}+1)} = \frac{\sum_{i=1}^{N} x_i w_i g\left(\left\| \frac{x^{(\mathrm{k})} - x_i}{h} \right\|^2 \right)}{\sum_{i=1}^{N} w_i g\left(\left\| \frac{x^{(\mathrm{k})} - x_i}{h} \right\|^2 \right)} \qquad g(x) = -k'(x)$$

Simplification of Mean Shift

$$x^{(k+1)} = \frac{\sum_{i=1}^{N} x_i w_i g\left(\left\|\frac{x^{(k)} - x_i}{h}\right\|^2\right)}{\sum_{i=1}^{N} w_i g\left(\left\|\frac{x^{(k)} - x_i}{h}\right\|^2\right)}$$

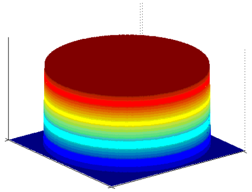
$$g(y) = -\tfrac{\partial}{\partial y} k(y)$$

Epanechnikov kernel $k(y)$:



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

Derivative $g(y)$ is the Uniform kernel:



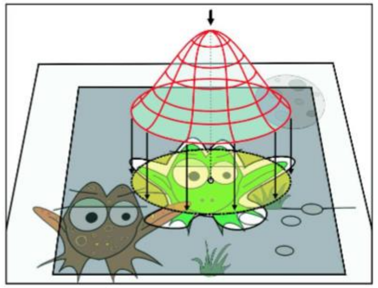
$$g(y) = -\tfrac{\partial}{\partial y} k(y) = \begin{cases} 1 & \text{if } \|y\| \le 1 \\ 0 & \text{otherwise} \end{cases}$$

$$x^{(k+1)} = \frac{\sum_{i=1}^{N} x_i w_i g\left(\left\|\frac{x^{(k)} - x_i}{h}\right\|^2\right)}{\sum_{i=1}^{N} w_i g\left(\left\|\frac{x^{(k)} - x_i}{h}\right\|^2\right)}$$

MS simplifies

$$x^{(k+1)} = \frac{\sum_{i=1}^{n} x_i w_i}{\sum_{i=1}^{n} w_i}, \qquad w_i = \sqrt{\frac{q_{b(x_i)}}{p_{b(x_i)}(x_0)}}$$

The MS tracking in a nutshell



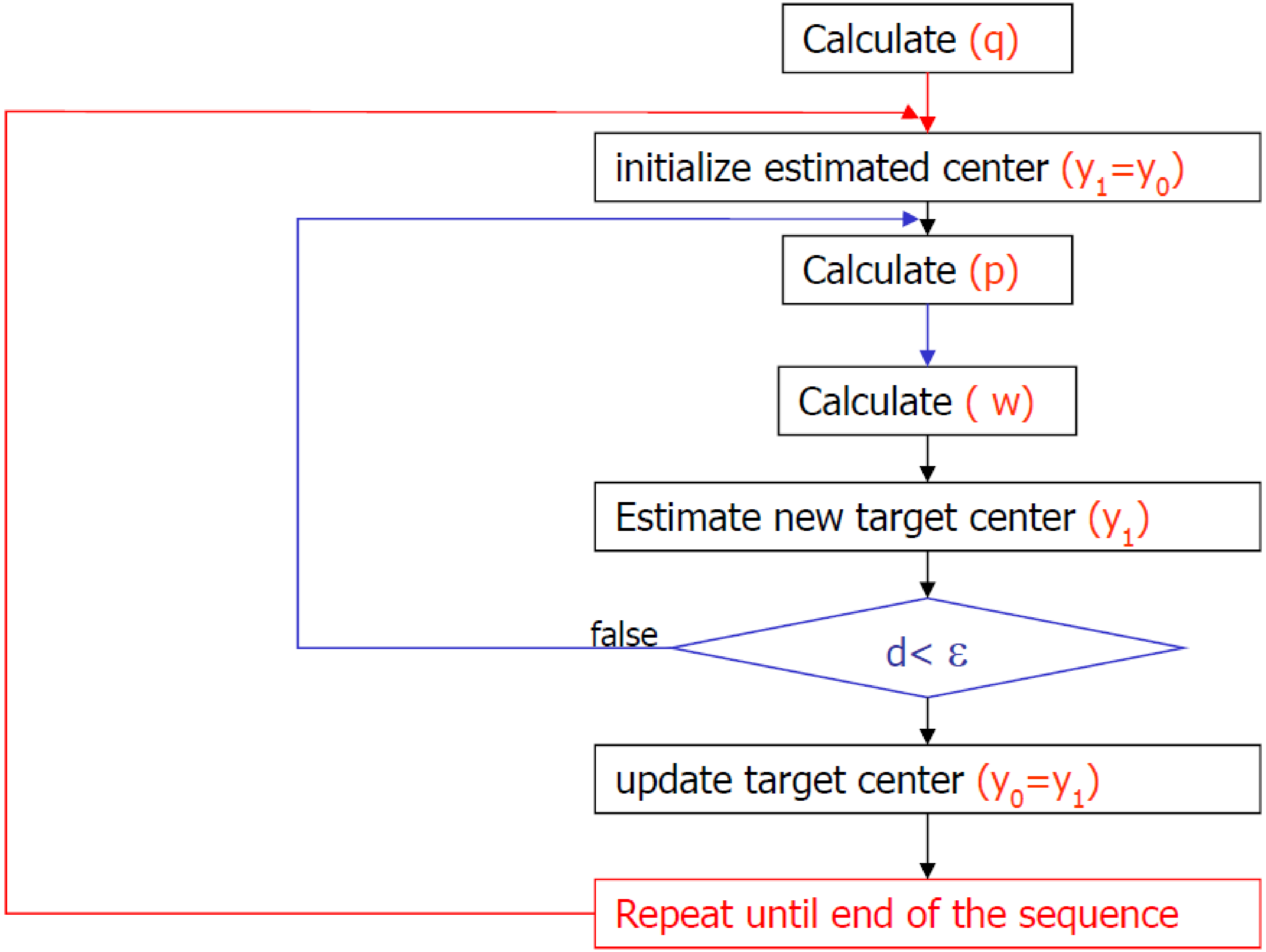
- Initialize target model (histogram) $\boldsymbol{q}$.
  - Note: use a smooth kernel, e.g., Epanechnikov
- New frame: start at some location
  1. Extract the histogram $\boldsymbol{p}$ at the current location using the Epanechnikov kernel
  2. For each pixel in the bounding box calculate the weight:

$$w_i = \sqrt{\frac{q_{b(x_i)}}{p_{b(x_i)}}}$$

  3. Calculate the new position by: $x^{(k+1)} = \frac{\sum_{i=1}^{n} x_i w_i}{\sum_{i=1}^{n} w_i}$

  - Iterate 1-3 until convergence

The tracking algorithm



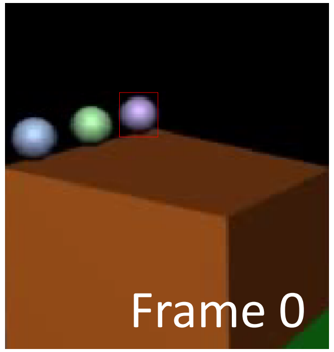
D. Comaniciu, V. Ramesh, P. Meer: Kernel-Based Object Tracking, TPAMI, 2003

Advanced Computer Vision Methods

MEAN SHIFT TRACKING STEPS ILLUSTRATED

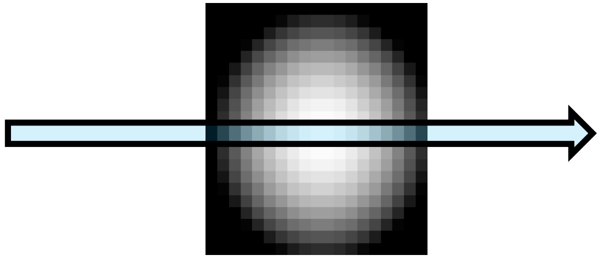
A single iteration within a time-step

Initialization:

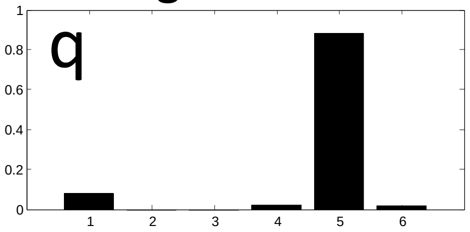


Cut out the target image

Kernel



Histogram extracted using the Kernel



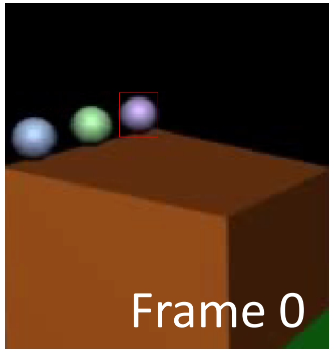
$$q_u = C\sum_{i=1}^{N} k\left(\left\|x_i\right\|^2\right)\delta_u(\mathrm{b}(\mathrm{x}_i))$$

Implementation of histogram extraction:

- Go over all pixels in the cut out image.
- For each pixel compute the histogram bin from its color.
- Look up the weight of the pixel coordinate in the Kernel image.
- Increment the content of histogram bin by the weight.
- Normalize the histogram to make the sum of all cells equal to one.
  (i.e., divide each cell by sum of all cells)

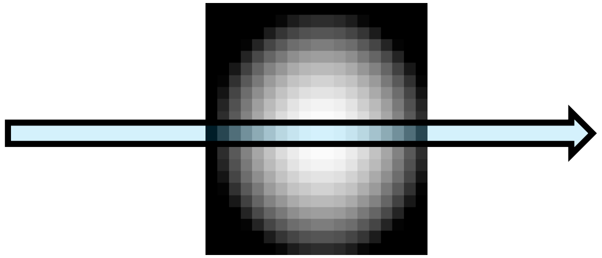
A single iteration within a time-step

Initialization:

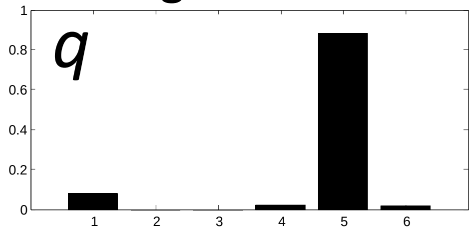


Cut out the target image

Kernel



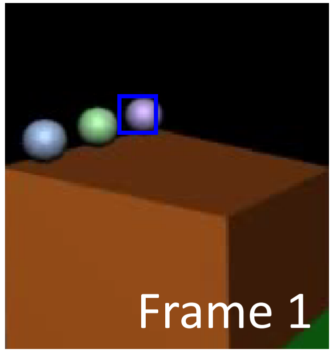
Histogram extracted using the Kernel



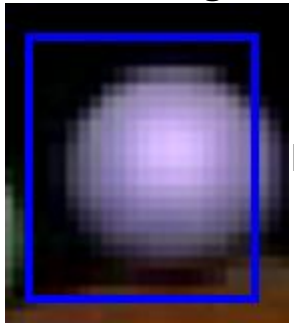
$$q_u = C\sum_{i=1}^{N} k\left(\left\|x_i\right\|^2\right)\delta_u(\mathrm{b}(\mathrm{x}_i))$$

$$x^{(k+1)} = \frac{\sum_{i=1}^{N} x_i w_i g\left(\left\|\frac{x^{(k)} - x_i}{h}\right\|^2\right)}{\sum_{i=1}^{N} w_i g\left(\left\|\frac{x^{(k)} - x_i}{h}\right\|^2\right)}$$

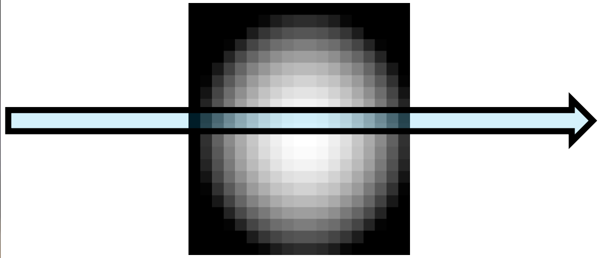
Tracking in Frame 1: *iteratively re-localize the target by MS (step 1b)*



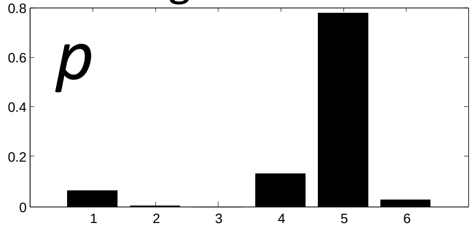
Cut out the target image



Kernel



Histogram extracted using the Kernel

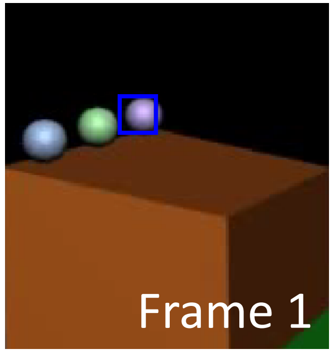


$$w_i = \sqrt{\frac{q_{b(\mathrm{x}_i)}}{p_{b(\mathrm{x}_i)}(\mathrm{x}_0)}}$$

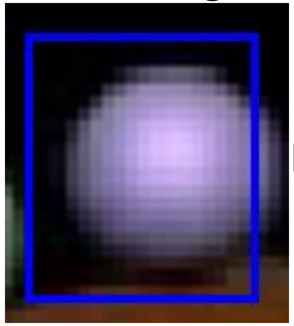
- The current estimate of the target position is the position from previous time-step
- Cut out the image from the current estimate (bounding box)
- Calculate the weighted histogram p using the Kernel

A single iteration within a time-step

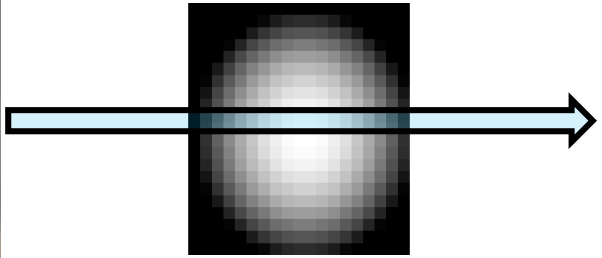
Tracking in Frame 1: *iteratively re-localize the target by MS (step 2c)*



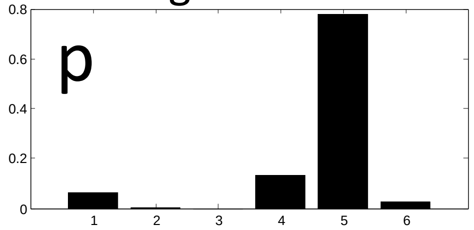
Cut out the target image



Kernel

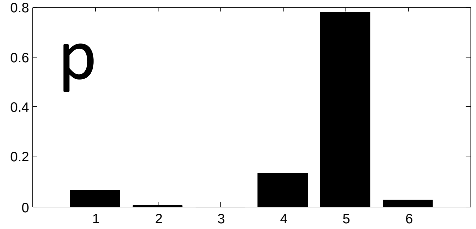


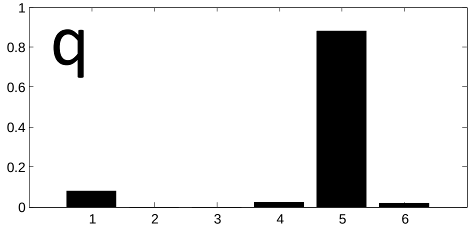
Histogram extracted using the Kernel



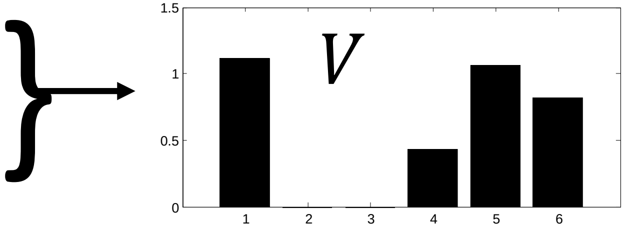
$$x^{(k+1)} = \frac{\sum_{i=1}^{N} x_i w_i g\left(\left\| \frac{x^{(k)} - x_i}{h} \right\|^2\right)}{\sum_{i=1}^{N} w_i g\left(\left\| \frac{x^{(k)} - x_i}{h} \right\|^2\right)}$$

Calculate the weight of each color bin from the target and candidate histogram:





$$v_u = \sqrt{\frac{q_u}{p_u + eps}}$$



$$w_i = \sqrt{\frac{q_{b(x_i)}}{p_{b(x_i)}(x_0)}}$$

eps is some small number for numerical stability, i.e., 1e-3 … 1e-10.

!! Source of many errors – don’t set eps too small!

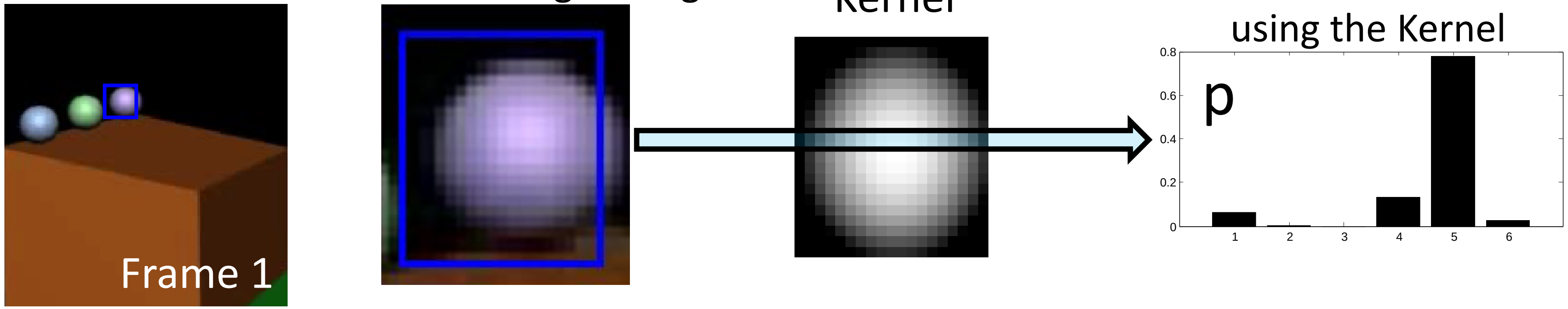
A single iteration within a time-step

Tracking in Frame 1: *iteratively re-localize the target by MS (step 2c)*

Cut out the target image

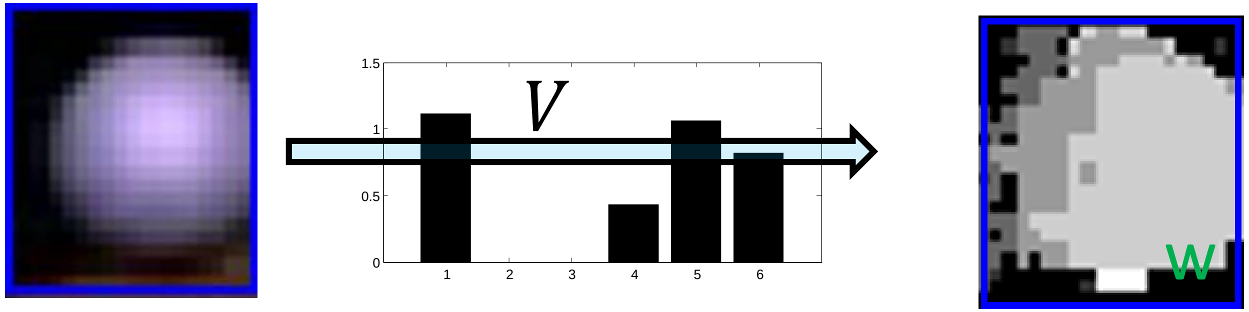
Kernel

Histogram extracted using the Kernel



Back project the weight histogram V into the image:

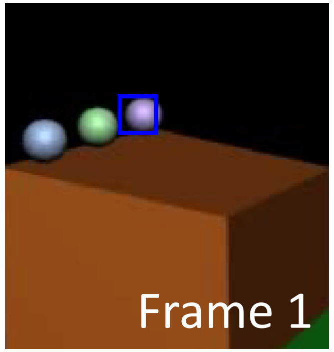
- For each pixel in the cut out image identify the histogram bin corresponding to its color.
- Set the intensity value of the pixel in backprojected image to value of the histogram V bin
- The backprojected image is same size as the cut out image



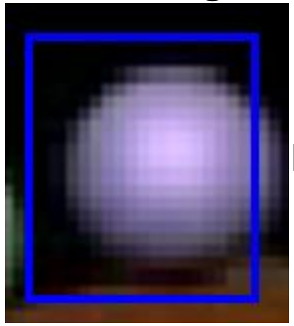
$$w_i = \sqrt{\frac{q_{b(x_i)}}{p_{b(x_i)}(x_0)}}$$

A single iteration within a time-step

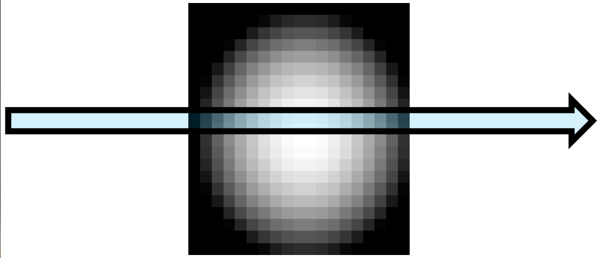
Tracking in Frame 1: *iteratively re-localize the target by MS (step 2c)*



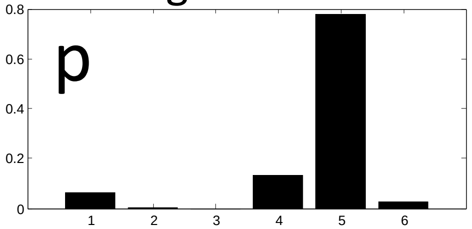
Cut out the target image



Kernel



Histogram extracted using the Kernel



$$x^{(k+1)} = \frac{\sum_{i=1}^{N} x_i w_i g\left(\left\| \frac{x^{(k)} - x_i}{h} \right\|^2\right)}{\sum_{i=1}^{N} w_i g\left(\left\| \frac{x^{(k)} - x_i}{h} \right\|^2\right)}$$

Multiply the backprojected image by the kernel $g(r)$:

- The kernel is derivative of the reparameterized Kernel w.r.t. parameter:

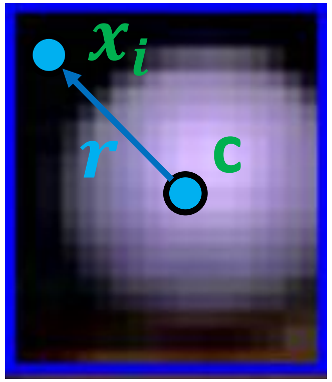
Epanechnikov kernel:

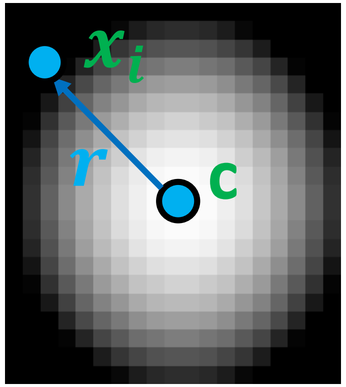
$$k(r) = \begin{cases} 1 - r & \text{if } \|\mathrm{r}\| \leq 1 \\ 0 & \text{otherwise} \end{cases}$$

$$r = \|\mathbf{c} - \mathbf{x}_i\| / h$$

center

pixel coordinate in the cutout window

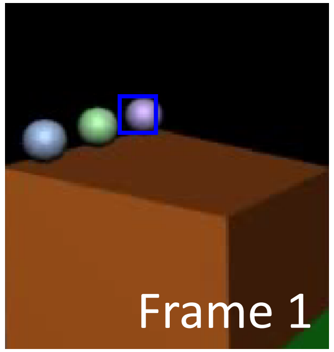




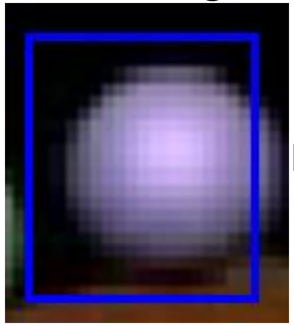
$k(r)$

A single iteration within a time-step

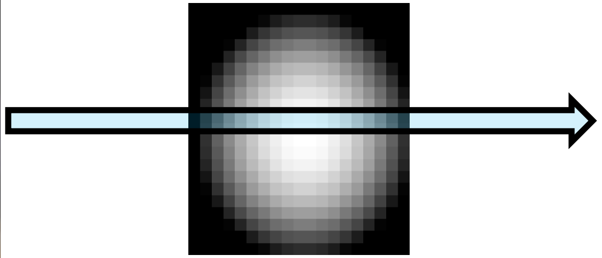
Tracking in Frame 1: *iteratively re-localize the target by MS (step 2c)*



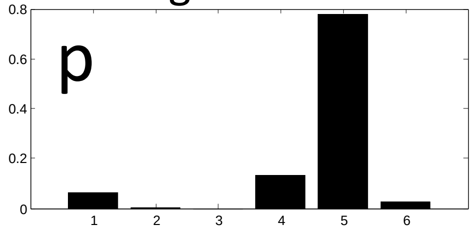
Cut out the target image



Kernel



Histogram extracted using the Kernel



Multiply the backprojected image by the kernel (derivative kernel):

- The kernel is derivative of the reparameterized Kernel w.r.t. parameter:

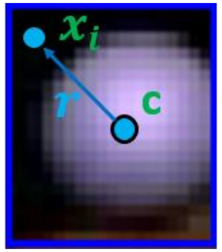
Epanechnikov kernel:

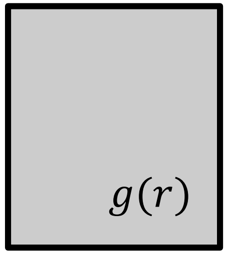
$$k(r) = \begin{cases} 1-r & \text{if } \|r\| \le 1 \\ 0 & \text{otherwise} \end{cases}$$

$$r = \|\mathbf{c} - \mathbf{x}_i\| / h$$

The "derivative" of the Epanechnikov is a Uniform kernel:

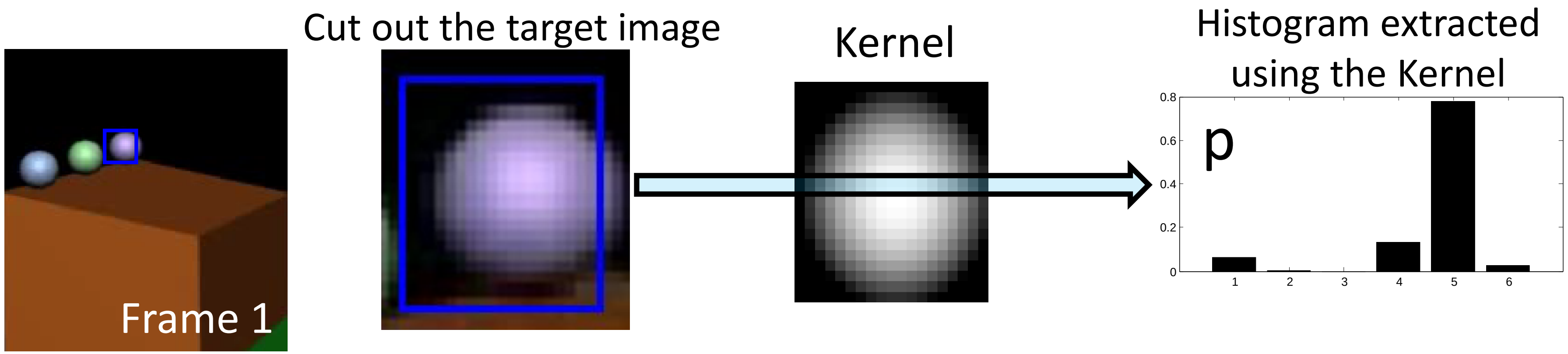
$$g(r) = -k'(r) = \begin{cases} 1 & \text{if } \|r\| \le 1 \\ 0 & \text{otherwise} \end{cases}$$





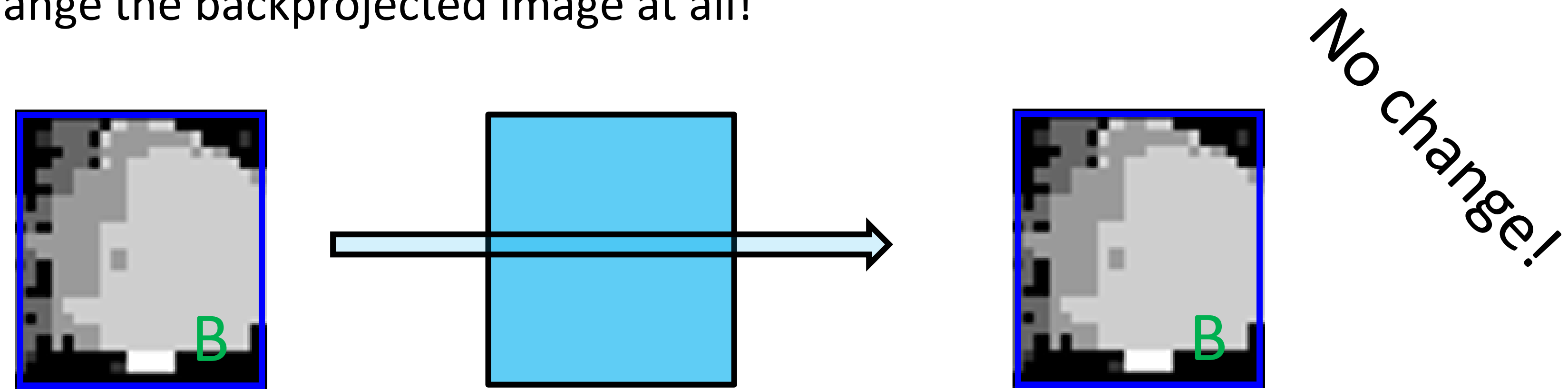
A single iteration within a time-step

Tracking in Frame 1: *iteratively re-localize the target by MS (step 2c)*



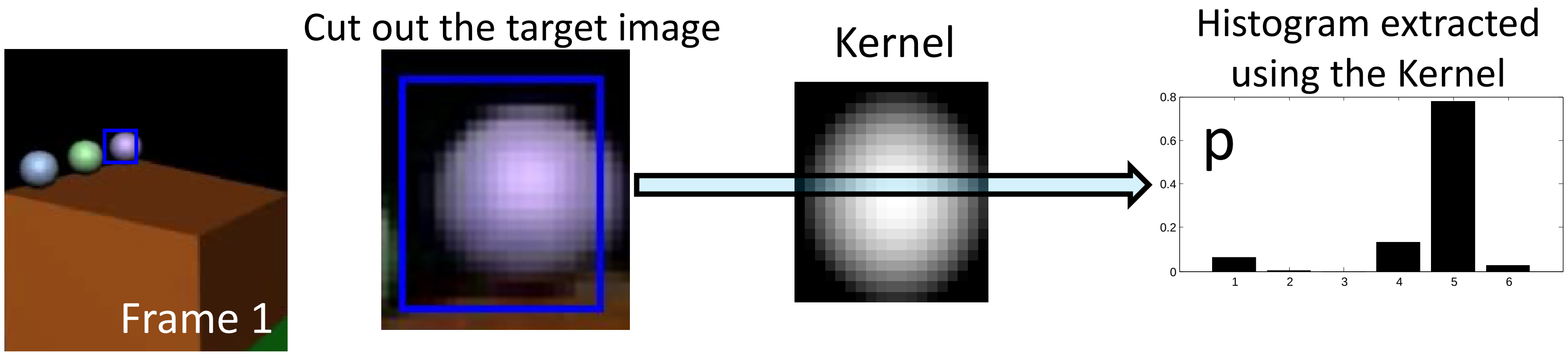
Multiply the backprojected image by the kernel:

- The kernel is derivative of the reparameterized Kernel w.r.t. parameter.
- In case of Epanechnikov kernel, the derivative is a Uniform kernel, which does not change the backprojected image at all!

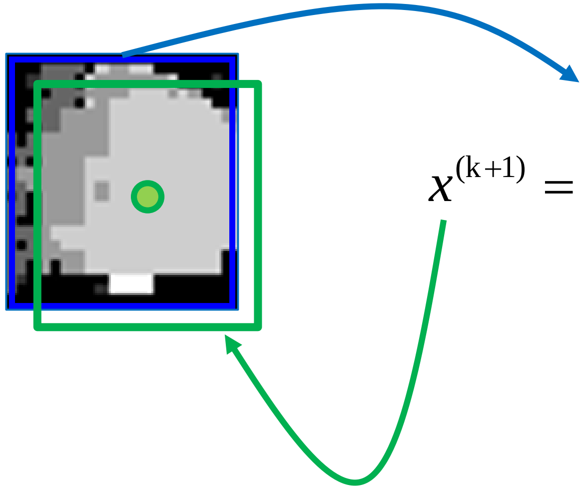


A single iteration within a time-step

Tracking in Frame 1: *iteratively re-localize the target by MS (step 3)*



Compute the weighted average position:



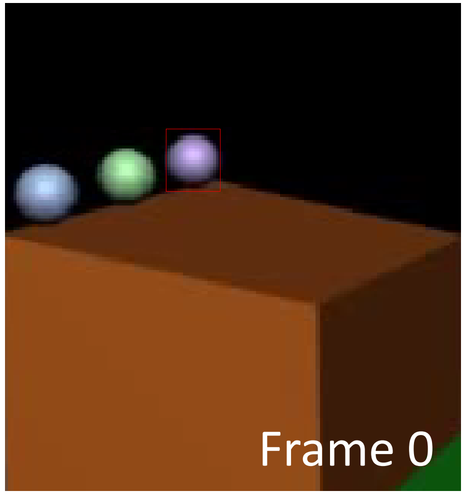
$$x^{(k+1)} = \frac{\sum_{i=1}^{n} x_i w_i}{\sum_{i=1}^{n} w_i}$$

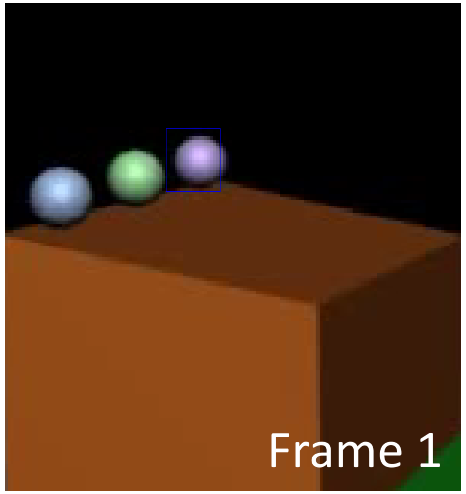
Repeat until convergence:

- *(1a) Cut out image at new position*
- *(1b) Compute* $\boldsymbol{p}$
- *(2a) Compute* $\boldsymbol{V}$
- *(2b) Compute back-projected image* ***W***
- *(2c) Multiply by derivative kernel*
- *(3) Calculate average position*

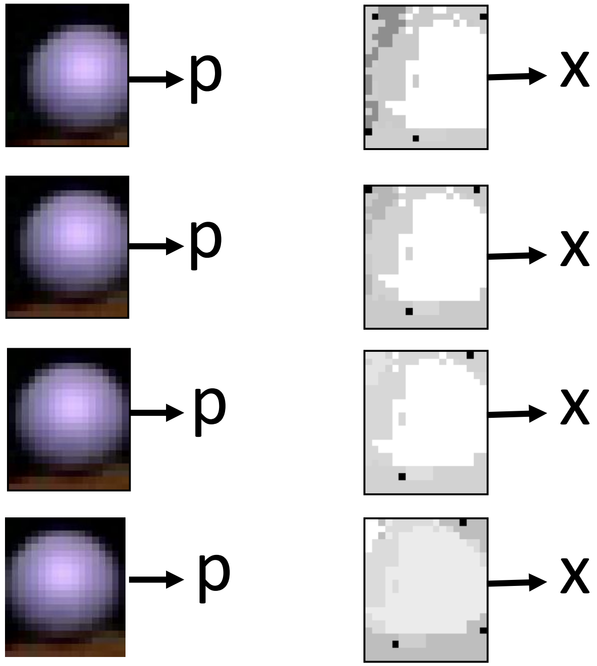
Apply iterations until covergence

Tracking in Frame 1: *iteratively re-localize the target by MS (all steps)*

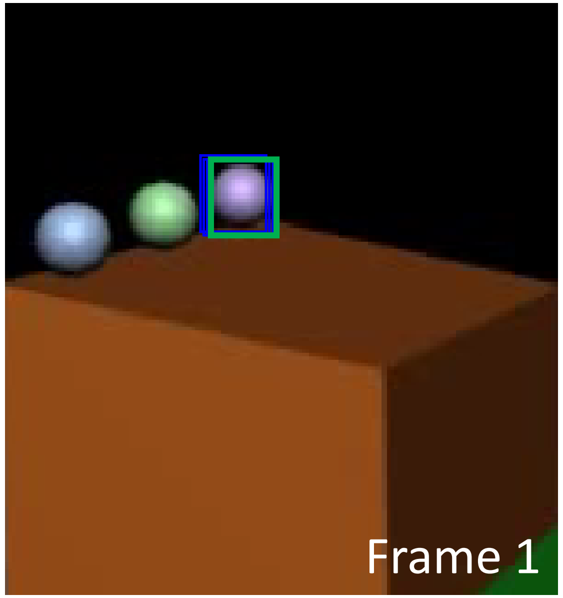




Cropped windows   Backprojections



Outputs of
MS iterations:



Converged: more MS iterations do not change *x.*

Implementation details

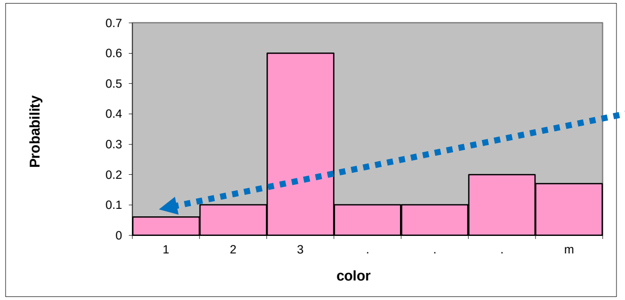
- Repeat MS iterations until the shift < 1 pixel
- Limit the number of iterations to N_{max}=20
- Kernels with Epanechnikov profile are preferred since the iteration becomes very simple.
  (but other kernels can be used as well)
- For speed: usually rescale the image such that the target is of size 50x50 pixels.
- Recommended using RGB histogram $16 \times 16 \times 16$ bins
- For further details see the paper[1].

[1]D. Comaniciu, V. Ramesh, P. Meer: Kernel-Based Object Tracking, TPAMI, 2003

Integrated feature selection

- Can search for the target by focusing on the features that discriminate the target from the background

Extract a histogram: $\hat{o} = \{\hat{o}_u\}_{u=1:m}$



Smallest nonzero entry

$$\left\{ c_u = \min\left(\frac{\hat{o}^*}{\hat{o}_u}, 1 \right) \right\}_{u=1:m}$$

Correct target and candidate model:

$$q_u^{(\text{corrected})} = c_u q_u^{(original)}$$

$$p_u^{(\text{corrected})} = c_u p_u^{(original)}$$

D. Comaniciu, V. Ramesh, P. Meer: Kernel-Based Object Tracking, TPAMI, 2003 (Sec. 6.1)

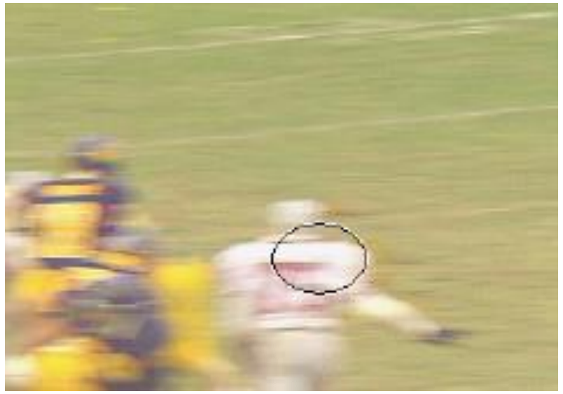
Mean Shift tracking example

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

Mean Shift tracking example

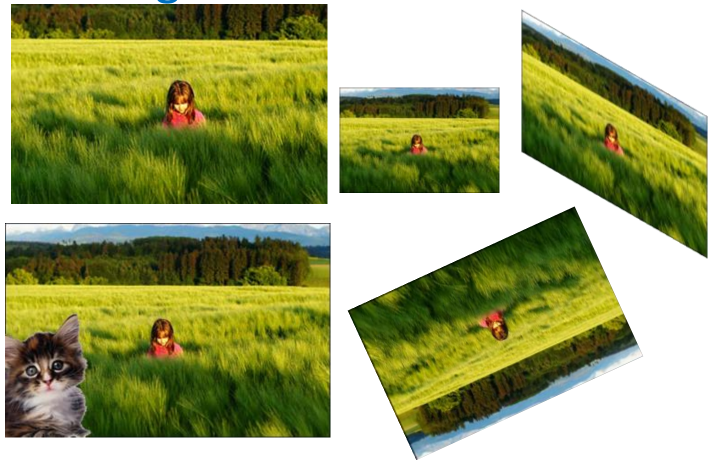
Partial occlusion

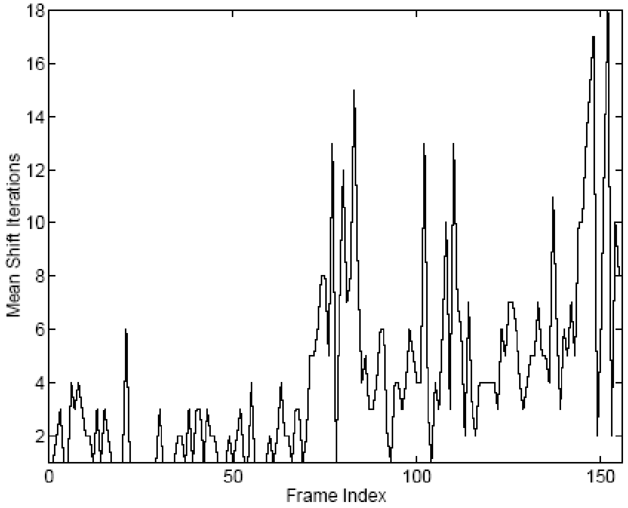
Distraction



Motion blur

Recall geometric transform invariance…





D. Comaniciu, V. Ramesh, P. Meer: *Kernel-Based Object Tracking* TPAMI, 2003

Mean Shift tracking example

D. Comaniciu, V. Ramesh, P. Meer: *Kernel-Based Object Tracking* TPAMI, 2003

Drawback: scale estimation

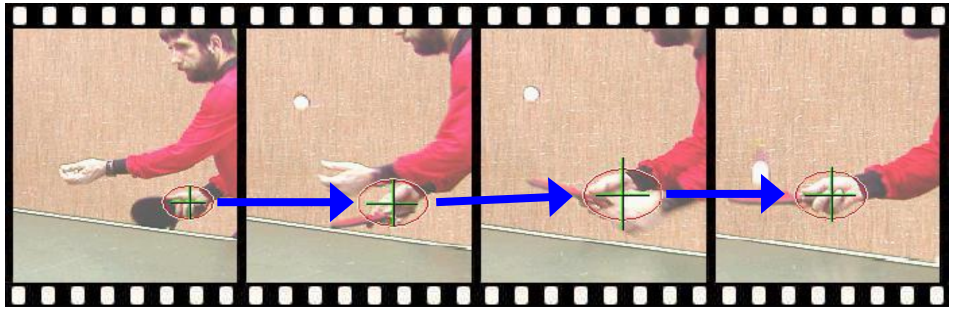
http://www.youtube.com/watch?v=RG5uV_h50b0

Scale changes

- The basic MS does not adapt to scale

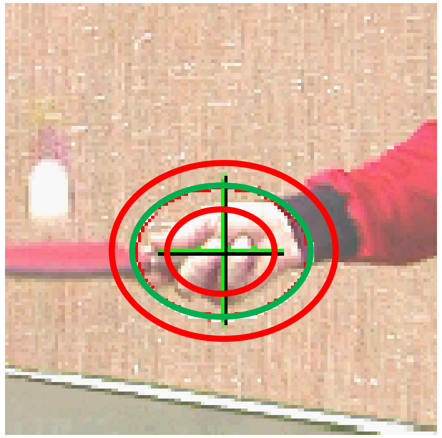
Problem:

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



Solution:

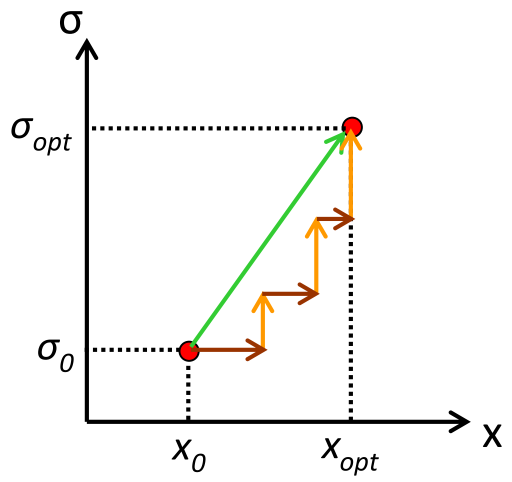
Run localization 3 times with different size → Choose *the size* that achieves maximum similarity



D. Comaniciu, V. Ramesh, P. Meer: *Kernel-Based Object Tracking* TPAMI, 2003

Alternating scale-shift estimation

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

Fixed-scale

± 10% scale

Tracking through scale space

Some recent scale-space advances

Robust Scale-Adaptive Mean-Shift for Tracking

Vojir, Noskova, Matas, SCIA, 2013

MS tracking by information fusion

D. Comaniciu: [Nonparametric Information Fusion for Motion Estimation](), CVPR, 2003

References

You should read to properly implement MS tracker:

- D. Comaniciu, V. Ramesh, P. Meer: *Kernel-Based Object Tracking*, TPAMI, Vol. 25, No. 5, 564-575, 2003
    - Read at least Sections 2-4.

If you want to learn more:

- Collins , Yanxi, Online Selection of Discriminative Tracking Features, TPAMI 2005 (code and videos)
- Collins, Mean-shift blob tracking through scale space, CVPR, 2003
- Tomaš Vojir, Jana Noskova, Jiri Matas, Robust Scale-Adaptive Mean-Shift for Tracking, SCIA, 2013
- D. Comaniciu: *Nonparametric Information Fusion for Motion Estimation*, CVPR, 2003

Acknowledgment

- Some parts of images and slides have been taken from the following presentation: Yaron Ukrainitz & Bernard Sarel, Mean Shift – Theory and applications
  - Check it out, it’s a nice presentation