EBU7240
Computer Vision
- Tracking: Image Alignment -

Semester 1, 2021

Changjae Oh

Content

- Motion Estimation (Review of EBU6230 content)
- Image Alignment
- Kanade-Lucas-Tomasi (KLT) Tracking
- Mean-shift Tracking

Objectives

- To review **Lucas-Kanade optical flow** in EBU6230
- To understand **Lucas-Kanade image alignment**
- To understand the **relationship** between Lucas-Kanade optical flow and im age alignment
- To understand **Kanade-Lucas-Tomasi tracker**

Content

- **Motion Estimation (Review of EBU6230 content)**
- Image Alignment
- Kanade-Lucas-Tomasi (KLT) Tracking
- Mean-shift Tracking

Motion Estimation: Gradient method

- Brightness consistency constraint

$$H(x, y, t) = I(x + \Delta x, y + \Delta y, t + \Delta t)$$

- small motion: (Δx and Δy are less than 1 pixel)
  - suppose we take the Taylor series expansion of I:

$$I(x + \Delta, y + \Delta y, t + \Delta t) = I(x, y, t) + \frac{\partial I}{\partial x}\Delta x + \frac{\partial I}{\partial y}\Delta y + \frac{\partial I}{\partial t}\Delta t$$
$$+ \text{higher order terms}$$

$$I(x + \Delta, y + \Delta y, t + \Delta t) \approx I(x, y, t) + \frac{\partial I}{\partial x}\Delta x + \frac{\partial I}{\partial y}\Delta y + \frac{\partial I}{\partial t}\Delta t$$

Gradient method

- Spatio-temporal constraint

$$\boxed{\frac{\partial I}{\partial x} V_x + \frac{\partial I}{\partial y} V_y + \frac{\partial I}{\partial t} = 0}$$

- This equation introduces one constraint only
  - Where the motion vector of a pixel has 2 components (parameters)
  - A second constraints is necessary to solve the system

Aperture problem

- The aperture problem
  - stems from the need to solve one equation with two unknowns, which are the two components of optical flow
  - it is not possible to estimate both components of the optical flow from the local spatial and temporal derivatives

- By applying a constraint
  - the optical flow field changes smoothly in a small neighborhood
    it is possible to estimate both components of the optical flow
    if the spatial and temporal derivatives of the image
    intensity are available

Solving the aperture problem

- How to get more equations for a pixel?

- By applying a constraint
  - the optical flow field changes smoothly in a small neighborhood
    it is possible to estimate both components of the optical flow
    if the spatial and temporal derivatives of the image
    intensity are available

- Lucas–Kanade method

Gradient method

- The Lucas–Kanade method assumes that the displacement of the image contents between two nearby instants (frames) is small

$$
\begin{aligned}
& I_x(q_1)V_x + I_y(q_1)V_y = -I_t(q_1) \\
& I_x(q_2)V_x + I_y(q_2)V_y = -I_t(q_2) \\
& \dots \\
& I_x(q_n)V_x + I_y(q_n)V_y = -I_t(q_n)
\end{aligned}
$$

- Matrix form

$$
A = \begin{bmatrix} I_x(q_1) & I_y(q_1) \\ I_x(q_2) & I_y(q_2) \\ \dots & \dots \\ I_x(q_n) & I_y(q_n) \end{bmatrix} \qquad v = \begin{bmatrix} V_x \\ V_y \end{bmatrix} \qquad b = \begin{bmatrix} -I_t(q_1) \\ -I_t(q_2) \\ \dots \\ -I_t(q_n) \end{bmatrix}
$$

Gradient method

- Prob: we have more equations than unknowns

$$Av = b \longrightarrow \text{minimize } \|Av - b\|^2$$

- Solution: solve least squares problem

$$(A^T A)v = A^T b$$

- minimum least squares solution given by solution of:

$$\begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} \begin{bmatrix} V_x \\ V_y \end{bmatrix} = -\begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}$$

- The summations are over all n pixels in the K x K window
- This technique was first proposed by Lukas & Kanade (1981)

Lucas-Kanade flow

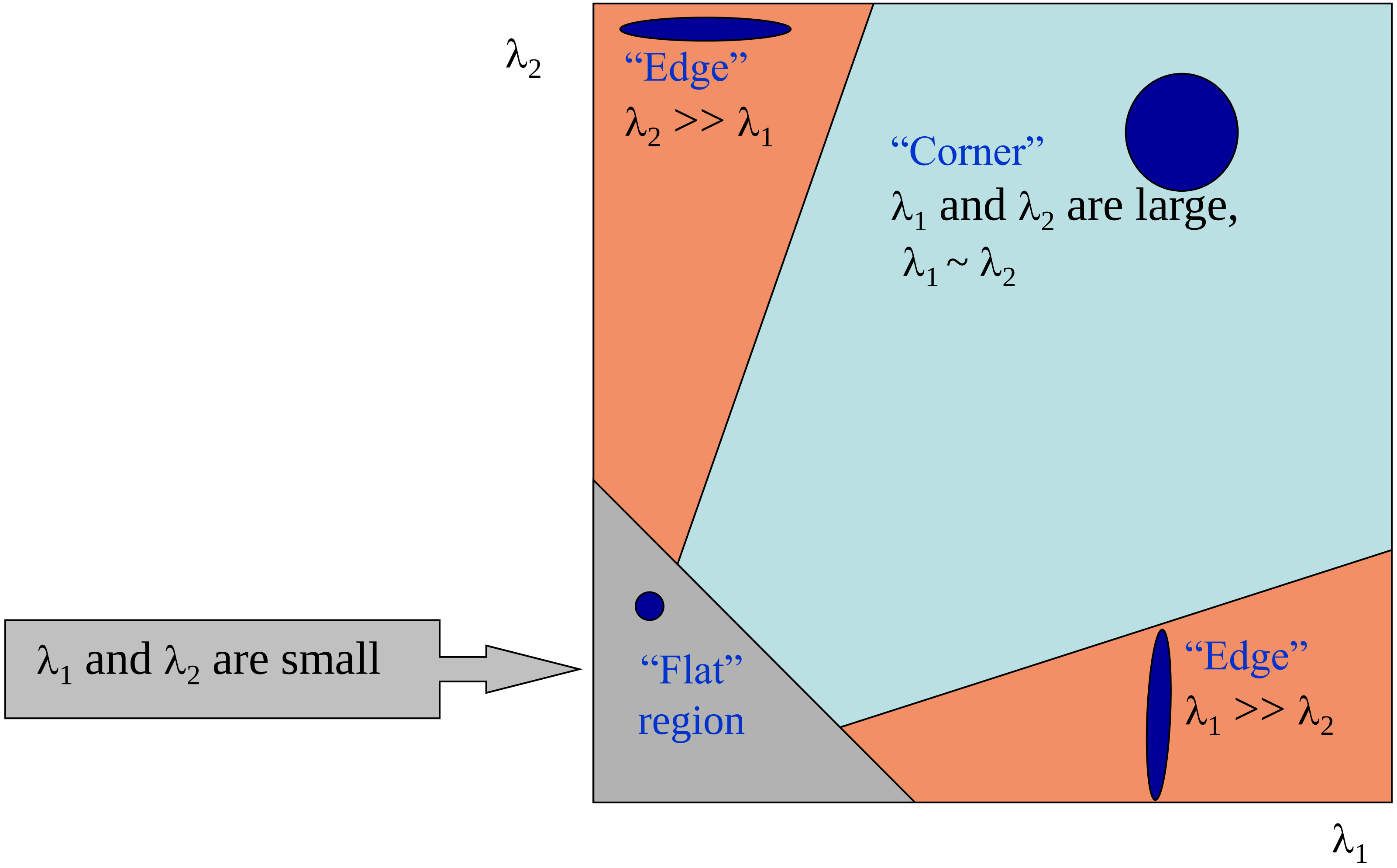
$$\underbrace{\begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix}}_{A^T A} \begin{bmatrix} V_x \\ V_y \end{bmatrix} = - \underbrace{\begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}}_{A^T b}$$

- When is this solvable?
  - **A^T A** should be invertible
  - **A^T A** should not be too small due to noise
    – eigenvalues λ_1 and λ_2 of **A^T A** should not be too small
  - **A^T A** should be well-conditioned
    – λ_1 / λ_2 should not be too large (λ_1 = larger eigenvalue)

– Recall the Harris corner detector: $M = A^T A$ is the *second moment matrix*

Interpreting the eigenvalues

Classification of image points using eigenvalues of the second moment matrix:



Uniform region

– gradients have small magnitude
– small λ_1, small λ_2
– system is ill-conditioned

Edge

– gradients have one dominant direction
– large λ_1, small λ_2
– system is ill-conditioned

High-texture or corner region

– gradients have different directions, large magnitudes
– large λ_1, large λ_2
– system is well-conditioned

Content

- Motion Estimation (Review of EBU6230 content)
- Image Alignment
- Kanade-Lucas-Tomasi (KLT) Tracking
- Mean-shift Tracking


31 fps

How can I find  in the image?

Idea #1: Template Matching

Slow, global solution

Idea #2: Pyramid Template Matching

Faster, locally optimal

Idea #3: Model refinement

(when you have a good initial solution)

Fastest, locally optimal

Some notation before we get into the math...

2D image transformation

$$\mathbf{W}(\boldsymbol{x};\boldsymbol{p})$$

2D image coordinate

$$\boldsymbol{x} = \begin{bmatrix} x \\ y \end{bmatrix}$$

Parameters of the transformation

$$\boldsymbol{p} = \{p_1, \ldots, p_N\}$$

Warped image

$$I(\boldsymbol{x}') = \underbrace{I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p}))}_{\text{Pixel value at a coordinate}}$$

Translation

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

$$= \underbrace{\begin{bmatrix} 1 & 0 & p_1 \\ 0 & 1 & p_2 \end{bmatrix}}_{\text{transform}} \underbrace{\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}}_{\text{coordinate}}$$

Affine

$$\mathbf{W}(\boldsymbol{x};\boldsymbol{p}) = \begin{bmatrix} p_1 x + p_2 y + p_3 \\ p_4 x + p_5 y + p_6 \end{bmatrix}$$

$$= \underbrace{\begin{bmatrix} p_1 & p_2 & p_3 \\ p_4 & p_5 & p_6 \end{bmatrix}}_{\text{affine transform}} \underbrace{\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}}_{\text{coordinate}}$$

can be written in matrix form when linear
affine warp matrix can also be 3x3 when last row is [0 0 1]

Image alignment

- **Problem definition**

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

warped image　　　　　template image

Find the warp parameters **p** such that the SSD is minimized

Image alignment

Find the warp parameters **p** such that the SSD is minimized

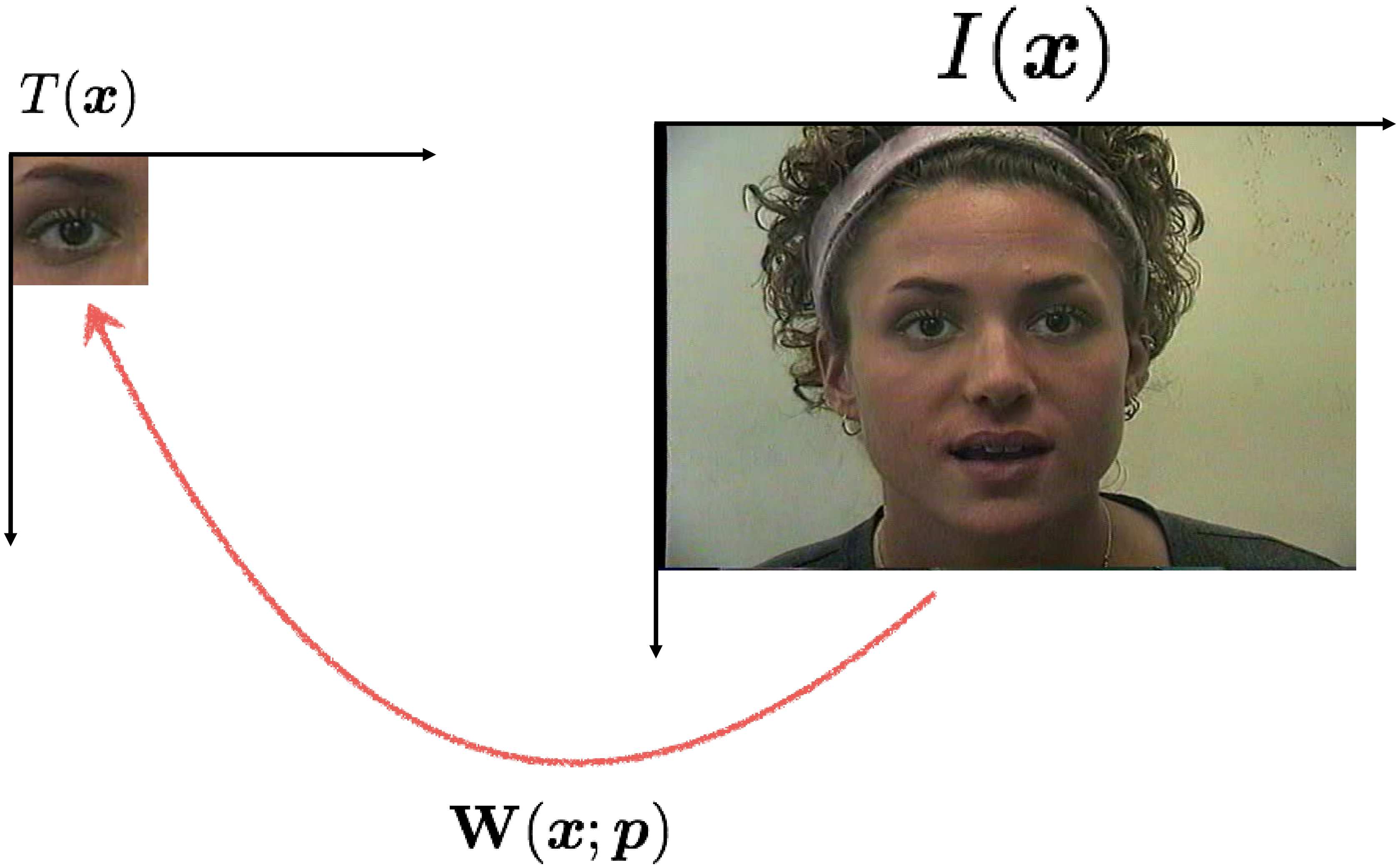


Image alignment

- **Problem definition**

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

warped image   template image

Find the warp parameters **p** such that the SSD is minimized

How could you find a solution to this problem?

Image alignment

This is a non-linear (quadratic) function of a non-parametric function!

(Function $\boldsymbol{I}$ is non-parametric)

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

Hard to optimize

What can you do to make it easier to solve?

assume good initialization,
linearized objective and update incrementally

Lucas-Kanade alignment

(pretty strong assumption)

If you have a good initial guess ***p***...

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

can be written as ...

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

(a small incremental adjustment)
(this is what we are solving for now)

Lucas-Kanade alignment

This is **still** a non-linear (quadratic) function of a non-parametric function!

(Function $\boldsymbol{I}$ is non-parametric)

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

*How can we linearize the function **I** for a really small perturbation of **p**?*

Taylor series approximation!

Lucas-Kanade alignment

Multivariable Taylor Series Expansion
(First order approximation)

$$f(x,y) \approx f(a,b) + f_x(a,b)(x-a) + f_y(a,b)(y-b)$$

Recall: $\boldsymbol{x}' = \mathbf{W}(\boldsymbol{x};\boldsymbol{p})$

$$I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p}+\Delta\boldsymbol{p})) \approx I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p})) + \frac{\partial I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p})}{\partial \boldsymbol{p}}\Delta\boldsymbol{p}$$

chain rule $= I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p})) + \frac{\partial I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p})}{\partial \boldsymbol{x}'}\frac{\partial \mathbf{W}(\boldsymbol{x};\boldsymbol{p})}{\partial \boldsymbol{p}}\Delta\boldsymbol{p}$

short-hand $= I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p})) + \nabla I \frac{\partial \mathbf{W}}{\partial \boldsymbol{p}}\Delta\boldsymbol{p}$

short-hand

Lucas-Kanade alignment

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

By linear approximation,

$$\sum_{\boldsymbol{x}} \left[I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p})) + \nabla I \frac{\partial \mathbf{W}}{\partial \boldsymbol{p}} \Delta\boldsymbol{p} - T(\boldsymbol{x}) \right]^2$$

Now, the function is a linear function of the unknowns

Lucas-Kanade alignment

The Jacobian $\frac{\partial \mathbf{W}}{\partial \boldsymbol{p}}$

(A matrix of partial derivatives)

$$\boldsymbol{x} = \left[\begin{array}{c} x \\ y \end{array} \right]$$

$$\partial \mathbf{W} = \left[\begin{array}{c} W_x(x,y) \\ W_y(x,y) \end{array} \right]$$

$$\frac{\partial \mathbf{W}}{\partial \boldsymbol{p}} = \left[\begin{array}{cccc} \frac{\partial W_x}{\partial p_1} & \frac{\partial W_x}{\partial p_2} & \cdots & \frac{\partial W_x}{\partial p_N} \\ \frac{\partial W_y}{\partial p_1} & \frac{\partial W_y}{\partial p_2} & \cdots & \frac{\partial W_y}{\partial p_N} \end{array} \right]$$

Rate of change of the warp

Affine transform

$$\mathbf{W}(\boldsymbol{x};\boldsymbol{p}) = \left[\begin{array}{c} p_1 x + p_3 y + p_5 \\ p_2 x + p_4 y + p_6 \end{array} \right]$$

$$\frac{\partial W_x}{\partial p_1} = x \qquad \frac{\partial W_x}{\partial p_2} = 0 \qquad \cdots$$

$$\frac{\partial W_y}{\partial p_1} = 0 \qquad \cdots$$

$$\frac{\partial \mathbf{W}}{\partial \boldsymbol{p}} = \left[\begin{array}{cccccc} x & 0 & y & 0 & 1 & 0 \\ 0 & x & 0 & y & 0 & 1 \end{array} \right]$$

Lucas-Kanade alignment

Partial derivatives of warp function
(2 x 6)

warp function
(2 x 1)

template image intensity
(scalar)

warp parameters
(6 for affine)

$$\sum_{\boldsymbol{x}} \left[I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p})) + \nabla I \frac{\partial \mathbf{W}}{\partial \boldsymbol{p}} \Delta \boldsymbol{p} - T(\boldsymbol{x}) \right]^2$$

image gradient
(1 x 2)

incremental warp
(6 x 1)

pixel coordinate
(2 x 1)

image intensity
(scalar)

Summary: Lucas-Kanade alignment

Problem:

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

warped image template image

Difficult non-linear optimization problem

Strategy:

$$\sum_{\boldsymbol{x}} [I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p} + \Delta\boldsymbol{p})) - T(\boldsymbol{x})]^2$$

Assume known approximate solution
Solve for increment

$$\sum_{\boldsymbol{x}} \left[I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p})) + \nabla I \frac{\partial \mathbf{W}}{\partial \boldsymbol{p}} \Delta\boldsymbol{p} - T(\boldsymbol{x})\right]^2$$

Taylor series approximation
Linearize

then solve for $\Delta\boldsymbol{p}$

Lucas-Kanade alignment - Solver

OK, so how do we solve this?

$$\min_{\Delta \boldsymbol{p}} \sum_{\boldsymbol{x}} \left[I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p})) + \nabla I \frac{\partial \mathbf{W}}{\partial \boldsymbol{p}} \Delta \boldsymbol{p} - T(\boldsymbol{x}) \right]^2$$

Lucas-Kanade alignment - Solver

$$\min_{\Delta \boldsymbol{p}} \sum_{\boldsymbol{x}} \left[I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p})) + \nabla I \frac{\partial \mathbf{W}}{\partial \boldsymbol{p}} \Delta \boldsymbol{p} - T(\boldsymbol{x}) \right]^2$$

(moving terms around)

$$\min_{\Delta \boldsymbol{p}} \sum_{\boldsymbol{x}} \left[\nabla I \frac{\partial \mathbf{W}}{\partial \boldsymbol{p}} \Delta \boldsymbol{p} - \{ T(\boldsymbol{x}) - I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p})) \} \right]^2$$

vector of constants

vector of variables

constant

Have you seen this form of optimization problem before?

Lucas-Kanade alignment - Solver

$$\min_{\Delta \boldsymbol{p}} \sum_{\boldsymbol{x}} \left[I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p})) + \nabla I \frac{\partial \mathbf{W}}{\partial \boldsymbol{p}} \Delta \boldsymbol{p} - T(\boldsymbol{x}) \right]^2$$

$$\min_{\Delta \boldsymbol{p}} \sum_{\boldsymbol{x}} \left[\nabla I \frac{\partial \mathbf{W}}{\partial \boldsymbol{p}} \Delta \boldsymbol{p} - \{ T(\boldsymbol{x}) - I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p})) \} \right]^2$$

constant variable constant

Looks like $\mathbf{Ax} - \mathbf{b}$

How do you solve this?

Lucas-Kanade alignment - Solver

Least squares approximation

$$\hat{x} = \arg\min_x ||Ax - b||^2$$ is solved by $$x = (A^\top A)^{-1} A^\top b$$

Applied to our tasks:

$$\min_{\Delta \boldsymbol{p}} \sum_{\boldsymbol{x}} \left[\nabla I \frac{\partial \mathbf{W}}{\partial \boldsymbol{p}} \Delta \boldsymbol{p} - \{ T(\boldsymbol{x}) - I(\mathbf{W}(\boldsymbol{x}; \boldsymbol{p})) \} \right]^2$$

is optimized when

$$\Delta \boldsymbol{p} = H^{-1} \sum_{\boldsymbol{x}} \left[\nabla I \frac{\partial \mathbf{W}}{\partial \boldsymbol{p}} \right]^\top [T(\boldsymbol{x}) - I(\mathbf{W}(\boldsymbol{x}; \boldsymbol{p}))]$$

after applying

$$x = (A^\top A)^{-1} A^\top b$$

where $$H = \sum_{\boldsymbol{x}} \left[\nabla I \frac{\partial \mathbf{W}}{\partial \boldsymbol{p}} \right]^\top \left[\nabla I \frac{\partial \mathbf{W}}{\partial \boldsymbol{p}} \right]$$

$$A^\top A$$

Lucas-Kanade alignment - Solver

Solve:

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

warped image template image

Difficult non-linear optimization problem

Strategy:

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

<u>Assume</u> known approximate solution
Solve for increment

$$\sum_{\boldsymbol{x}} \left[I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p})) + \nabla I \frac{\partial \mathbf{W}}{\partial \boldsymbol{p}} \Delta\boldsymbol{p} - T(\boldsymbol{x}) \right]^2$$

Taylor series approximation
Linearize

Solution:

$$\Delta\boldsymbol{p} = H^{-1} \sum_{\boldsymbol{x}} \left[\nabla I \frac{\partial \mathbf{W}}{\partial \boldsymbol{p}} \right]^\top \left[T(\boldsymbol{x}) - I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p})) \right]$$

Solution to least square
s approximation

$$H = \sum_{\boldsymbol{x}} \left[\nabla I \frac{\partial \mathbf{W}}{\partial \boldsymbol{p}} \right]^\top \left[\nabla I \frac{\partial \mathbf{W}}{\partial \boldsymbol{p}} \right]$$

Hessian

Called Gauss-Newton gradient decent non-linear optimization!

Lucas-Kanade alignment - Algorithm

1. Warp image $I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p}))$

2. Compute error image $[T(\boldsymbol{x}) - I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p}))]$

3. Compute gradient $\nabla I(\boldsymbol{x}')$

   $\boldsymbol{x}'$: coordinates of the warped image (gradients of the warped image)

4. Evaluate Jacobian $\frac{\partial \mathbf{W}}{\partial \boldsymbol{p}}$

5. Compute Hessian H

$$H = \sum_{\boldsymbol{x}} \left[\nabla I \frac{\partial \mathbf{W}}{\partial \boldsymbol{p}}\right]^\top \left[\nabla I \frac{\partial \mathbf{W}}{\partial \boldsymbol{p}}\right]$$

6. Compute $\Delta \boldsymbol{p}$

$$\Delta \boldsymbol{p} = H^{-1} \sum_{\boldsymbol{x}} \left[\nabla I \frac{\partial \mathbf{W}}{\partial \boldsymbol{p}}\right]^\top [T(\boldsymbol{x}) - I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p}))]$$

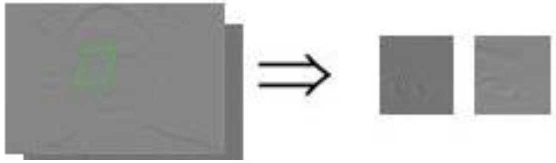
7. Update parameters $\boldsymbol{p} \leftarrow \boldsymbol{p} + \Delta \boldsymbol{p}$

Lucas-Kanade alignment - Algorithm

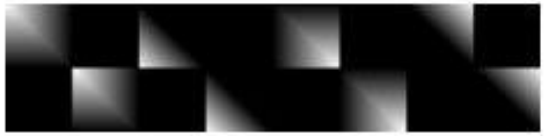
1. Warp image $I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p}))$

2. Compute error image $[T(\boldsymbol{x}) - I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p}))]$

3. Compute gradient $\nabla I(\boldsymbol{x}')$



4. Evaluate Jacobian $\frac{\partial \mathbf{W}}{\partial \boldsymbol{p}}$



5. Compute Hessian H

$$H = \sum_{\boldsymbol{x}} \left[\nabla I \frac{\partial \mathbf{W}}{\partial \boldsymbol{p}}\right]^\top \left[\nabla I \frac{\partial \mathbf{W}}{\partial \boldsymbol{p}}\right]$$

6. Compute $\Delta \boldsymbol{p}$

$$\Delta \boldsymbol{p} = H^{-1} \sum_{\boldsymbol{x}} \left[\nabla I \frac{\partial \mathbf{W}}{\partial \boldsymbol{p}}\right]^\top [T(\boldsymbol{x}) - I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p}))]$$

7. Update parameters $\boldsymbol{p} \leftarrow \boldsymbol{p} + \Delta \boldsymbol{p}$

L-K motion estimation *vs* L-K image alignment?

- **Relationships**

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

  - Lucas-Kanade motion estimation (what we learned in EBU6230) can be seen as a spec ial case of the Lucas-Kanade image alignment with a translational warp model

Translation

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

$$= \underbrace{\begin{bmatrix} 1 & 0 & p_1 \\ 0 & 1 & p_2 \end{bmatrix}}_{\text{transform}} \underbrace{\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}}_{\text{coordinate}}$$

Affine

$$\mathbf{W}(\boldsymbol{x};\boldsymbol{p}) = \begin{bmatrix} p_1 x + p_2 y + p_3 \\ p_4 x + p_5 y + p_6 \end{bmatrix}$$

$$= \underbrace{\begin{bmatrix} p_1 & p_2 & p_3 \\ p_4 & p_5 & p_6 \end{bmatrix}}_{\text{affine transform}} \underbrace{\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}}_{\text{coordinate}}$$

Content

- Motion Estimation (Review of EBU6230 content)
- Image Alignment
- Kanade-Lucas-Tomasi (KLT) Tracking
- Mean-shift Tracking

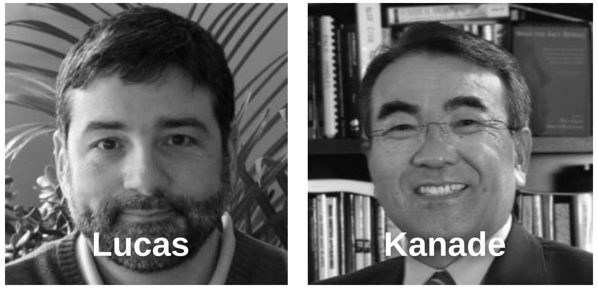
https://www.youtube.com/watch?v=rwljkECpY0M

Feature-based tracking

- **Up to now, we've been aligning entire images**
  - but we can also track just small image regions too

- **Questions to solve tracking**
  - How should we select features?
  - How should we track them from frame to frame?

KLT-tracker: history



An Iterative Image Registration Technique
with an Application to Stereo Vision.

1981

Detection and Tracking of Feature Points.

1991

The original KLT algorithm

Good Features to Track.

1994

KLT-tracker: history

Kanade-Lucas-Tomasi

How should we track them from frame to frame?

Lucas-Kanade

Method for aligning (tracking) an image patch

How should we select features?

Tomasi-Kanade

Method for choosing the best feature (image patch) for tracking

What are good features for tracking?

Intuitively, we want to avoid smooth regions and edges.
But is there a more principled way to define good features?

What are good features for tracking?

Can be derived from the tracking algorithm

'A feature is good if it can be tracked well'

Recall: Lucas-Kanade image alignment

error function (SSD)

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

incremental update

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

linearize

$$\sum_{\boldsymbol{x}} \left[I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p})) + \nabla I \frac{\partial \mathbf{W}}{\partial \boldsymbol{p}} \Delta\boldsymbol{p} - T(\boldsymbol{x}) \right]^2$$

Gradient update

$$\Delta\boldsymbol{p} = H^{-1} \sum_{\boldsymbol{x}} \left[\nabla I \frac{\partial \mathbf{W}}{\partial \boldsymbol{p}} \right]^\top \left[T(\boldsymbol{x}) - I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p})) \right]$$

$$H = \sum_{\boldsymbol{x}} \left[\nabla I \frac{\partial \mathbf{W}}{\partial \boldsymbol{p}} \right]^\top \left[\nabla I \frac{\partial \mathbf{W}}{\partial \boldsymbol{p}} \right]$$

Update

$$\boldsymbol{p} \leftarrow \boldsymbol{p} + \Delta\boldsymbol{p}$$

Hessian matrix

Stability of gradient decent iterations depends on …

$$\Delta \boldsymbol{p} = H^{-1} \sum_{\boldsymbol{x}} \left[\nabla I \frac{\partial \mathbf{W}}{\partial \boldsymbol{p}} \right]^{\top} [T(\boldsymbol{x}) - I(\mathbf{W}(\boldsymbol{x}; \boldsymbol{p}))]$$

Inverting the Hessian

$$H = \sum_{\boldsymbol{x}} \left[\nabla I \frac{\partial \mathbf{W}}{\partial \boldsymbol{p}} \right]^{\top} \left[\nabla I \frac{\partial \mathbf{W}}{\partial \boldsymbol{p}} \right]$$

When does the inversion fail?

H is singular. But what does that mean?

Hessian matrix

Above the noise level

$$\lambda_1 \gg 0$$
$$\lambda_2 \gg 0$$

both Eigenvalues are large

Well-conditioned

both Eigenvalues have similar magnitude

Hessian matrix

Concrete example: Consider translation model

$$\mathbf{W}(\boldsymbol{x};\boldsymbol{p}) = \begin{bmatrix} x + p_1 \\ y + p_2 \end{bmatrix} \qquad \frac{\mathbf{W}}{\partial \boldsymbol{p}} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Hessian

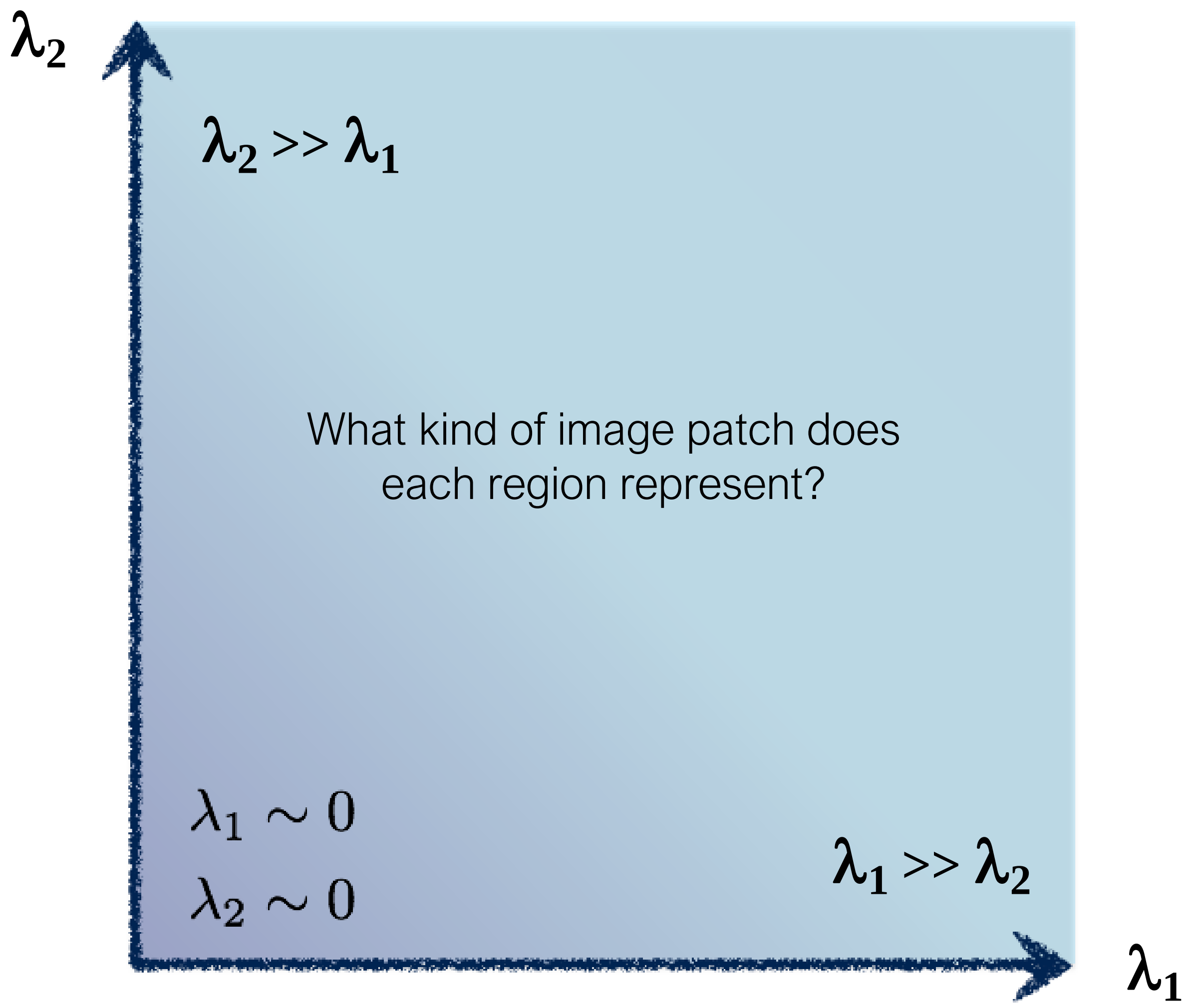
$$H = \sum_{\boldsymbol{x}} \left[\nabla I \frac{\partial \mathbf{W}}{\partial \boldsymbol{p}} \right]^\top \left[\nabla I \frac{\partial \mathbf{W}}{\partial \boldsymbol{p}} \right]$$

$$= \sum_{\boldsymbol{x}} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} I_x \\ I_y \end{bmatrix} \begin{bmatrix} I_x & I_y \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

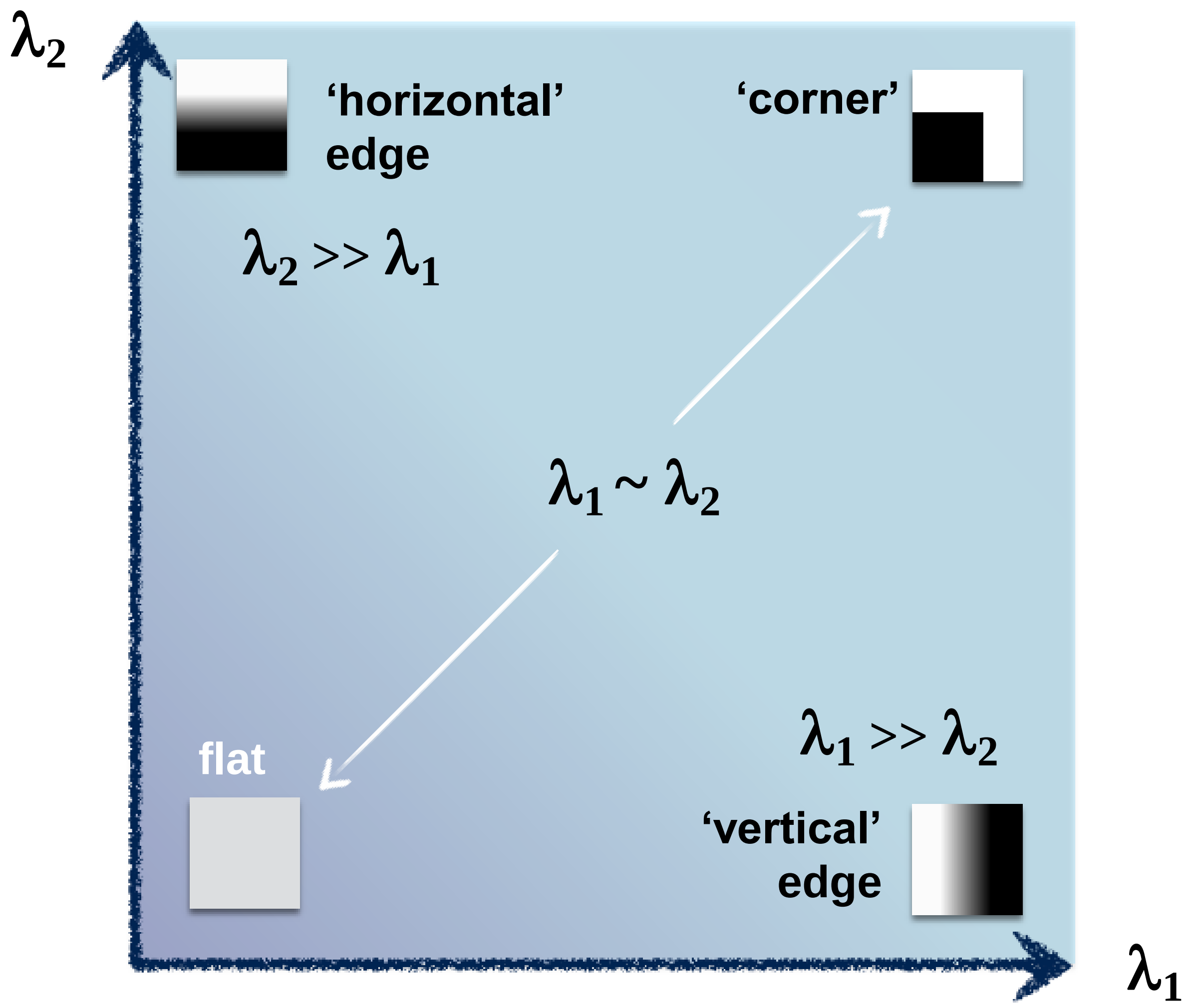
$$= \begin{bmatrix} \sum_{\boldsymbol{x}} I_x I_x & \sum_{\boldsymbol{x}} I_y I_x \\ \sum_{\boldsymbol{x}} I_x I_y & \sum_{\boldsymbol{x}} I_y I_y \end{bmatrix}$$ ←*when is this singular?*

How are the eigenvalues related to image content?

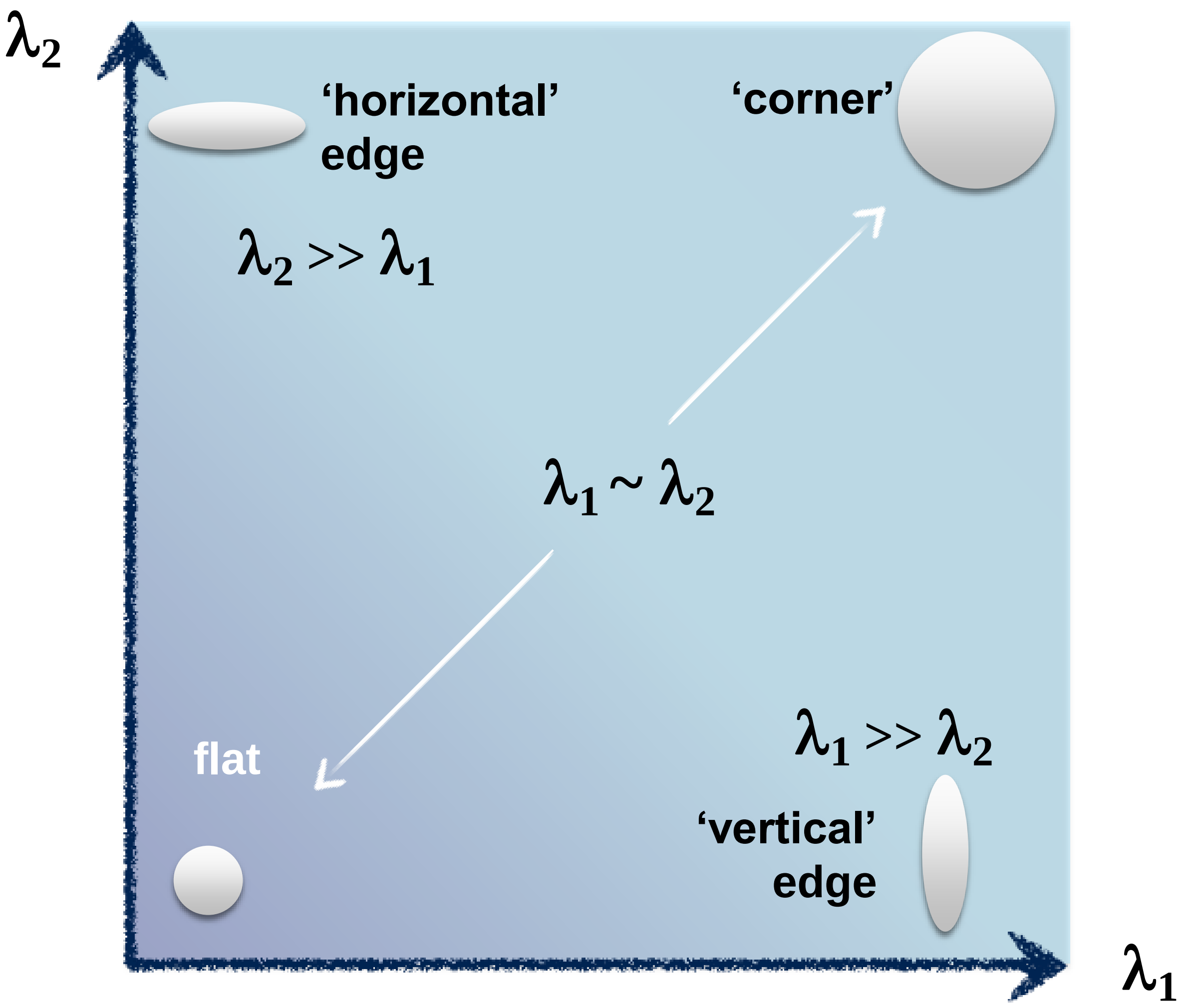
Interpreting eigenvalues



Interpreting eigenvalues



Interpreting eigenvalues



What are good features for tracking?

$$\min(\lambda_1, \lambda_2) > \lambda$$

'big Eigenvalues means good for tracking'

KLT algorithm

1. Find corners satisfying $\min(\lambda_1, \lambda_2) > \lambda$
2. For each corner compute displacement to next frame using the Lucas-Kanade method
3. Store displacement of each corner, update corner position
4. (optional) Add more corner points every M frames using 1
5. Repeat 2 to 3 (4)
6. Returns long trajectories for each corner point

EBU7240
Computer Vision
- Mean-shift tracking -

Semester 1, 2021

Changjae Oh

Content

- Motion Estimation (Review of EBU6230 content)
- Image Alignment
- Kanade-Lucas-Tomasi (KLT) Tracking
- **Mean-shift Tracking**


Nationale-Nederlanden
jackpot €5 milj.
ING

Mean Shift Algorithm

A ‘mode seeking’ algorithm

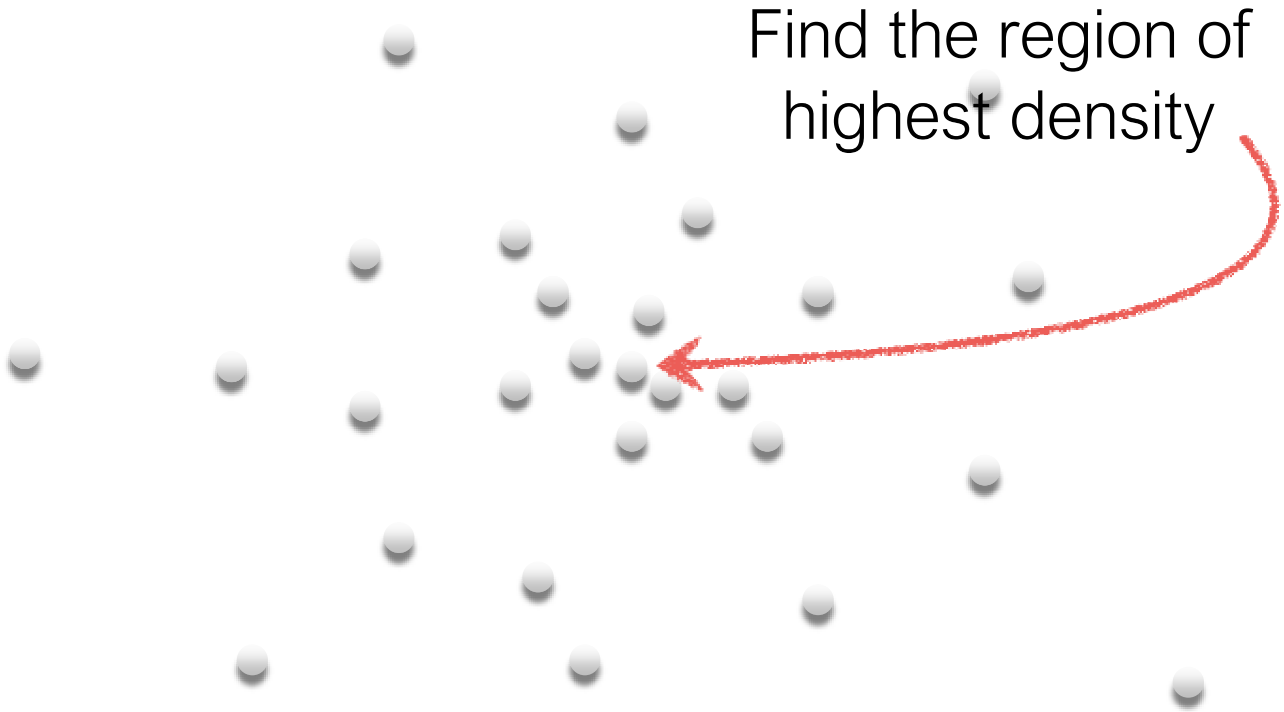
Fukunaga & Hostetler (1975)

Slides credit: K. Kitani

Mean Shift Algorithm

A ‘mode seeking’ algorithm

Fukunaga & Hostetler (1975)



Mean Shift Algorithm

A ‘mode seeking’ algorithm

Fukunaga & Hostetler (1975)

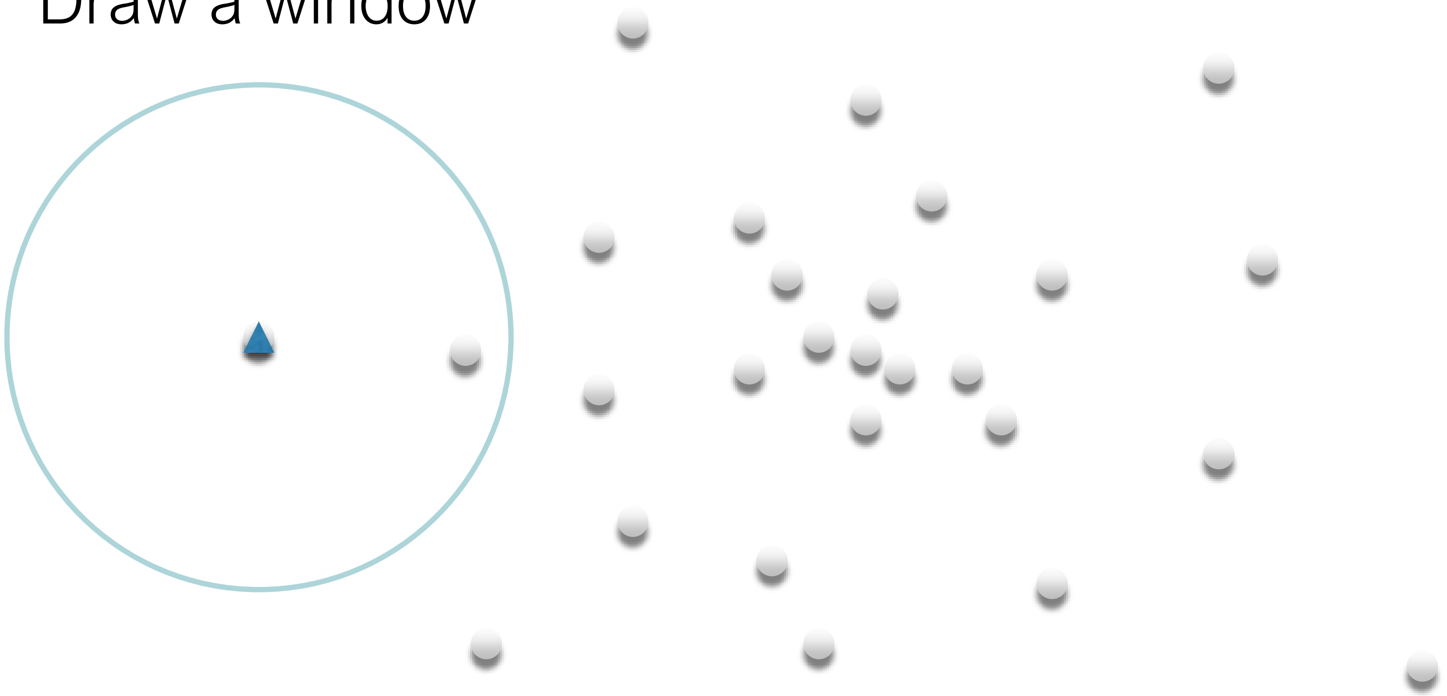
Pick a point

Mean Shift Algorithm

A ‘mode seeking’ algorithm

Fukunaga & Hostetler (1975)

Draw a window

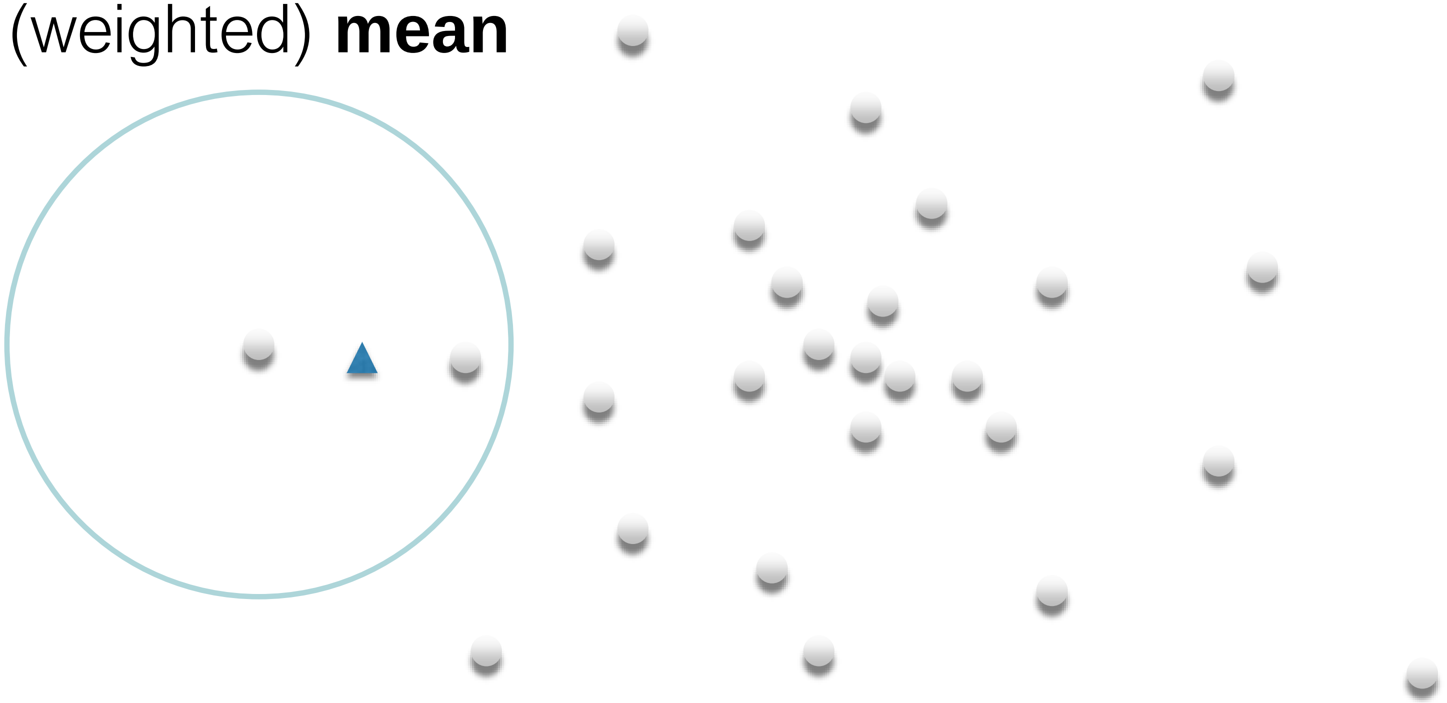


Mean Shift Algorithm

A ‘mode seeking’ algorithm

Fukunaga & Hostetler (1975)

Compute the (weighted) **mean**

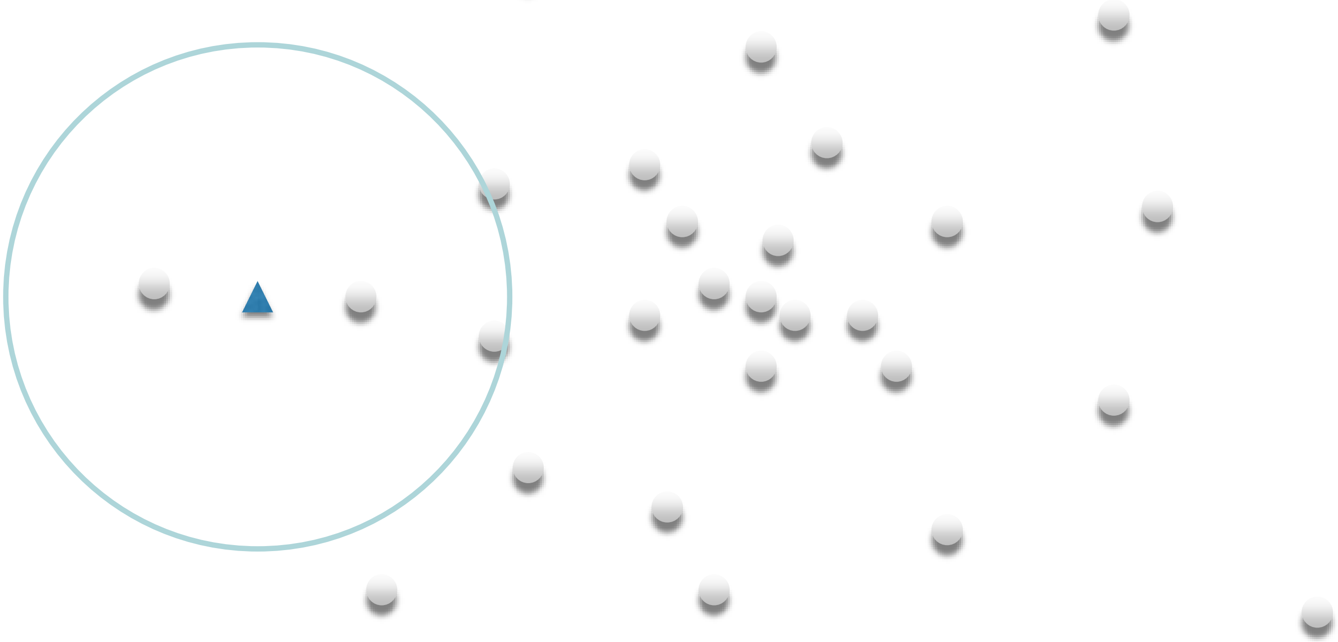


Mean Shift Algorithm

A ‘mode seeking’ algorithm

Fukunaga & Hostetler (1975)

Shift the window

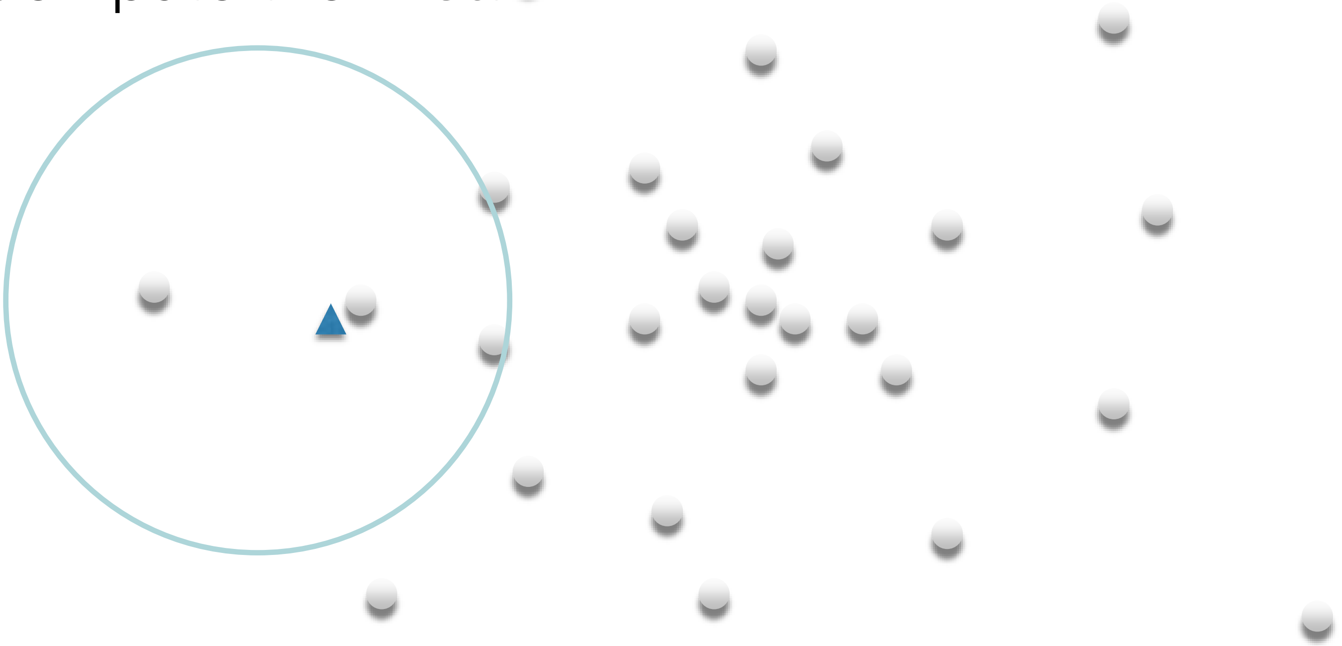


Mean Shift Algorithm

A ‘mode seeking’ algorithm

Fukunaga & Hostetler (1975)

Compute the **mean**

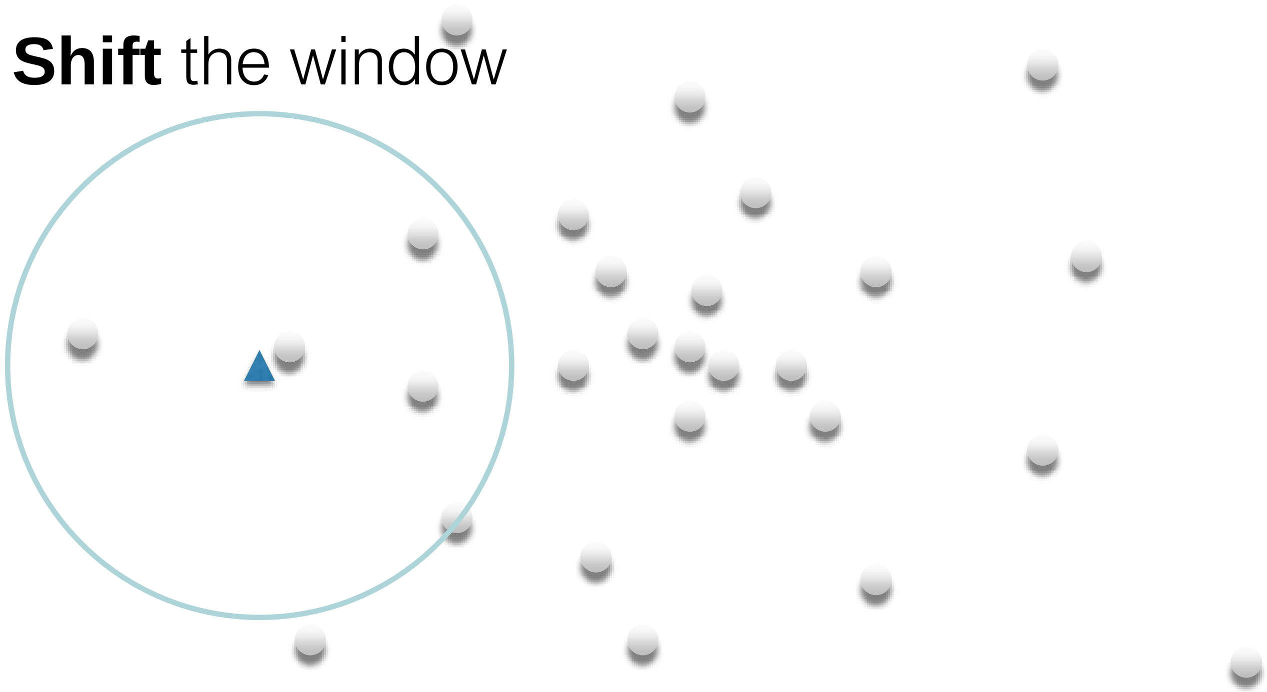


Mean Shift Algorithm

A ‘mode seeking’ algorithm

Fukunaga & Hostetler (1975)

Shift the window

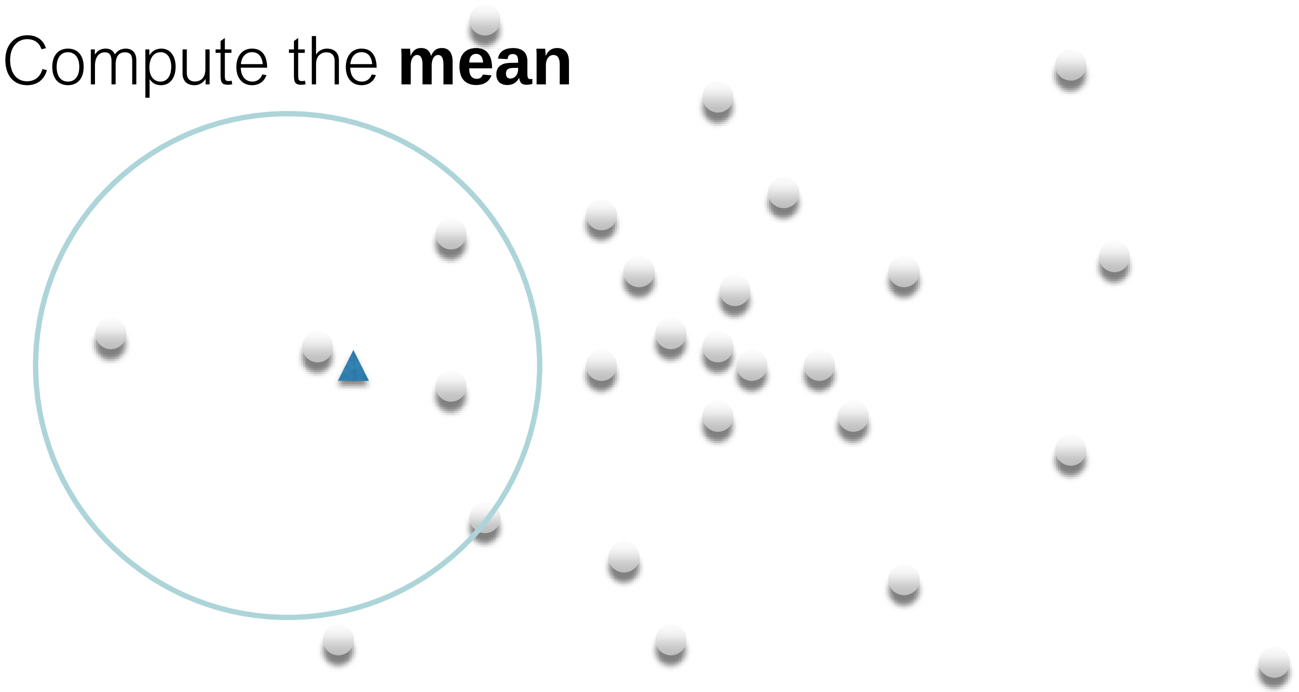


Mean Shift Algorithm

A ‘mode seeking’ algorithm

Fukunaga & Hostetler (1975)

Compute the **mean**

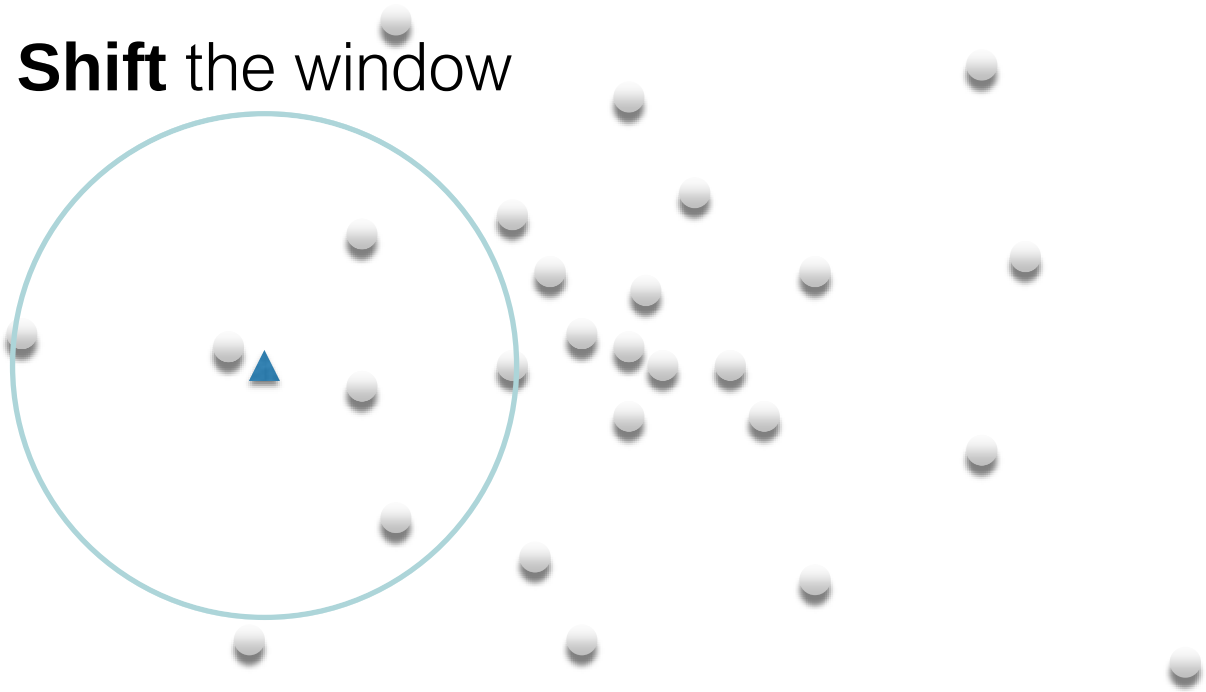


Mean Shift Algorithm

A ‘mode seeking’ algorithm

Fukunaga & Hostetler (1975)

Shift the window

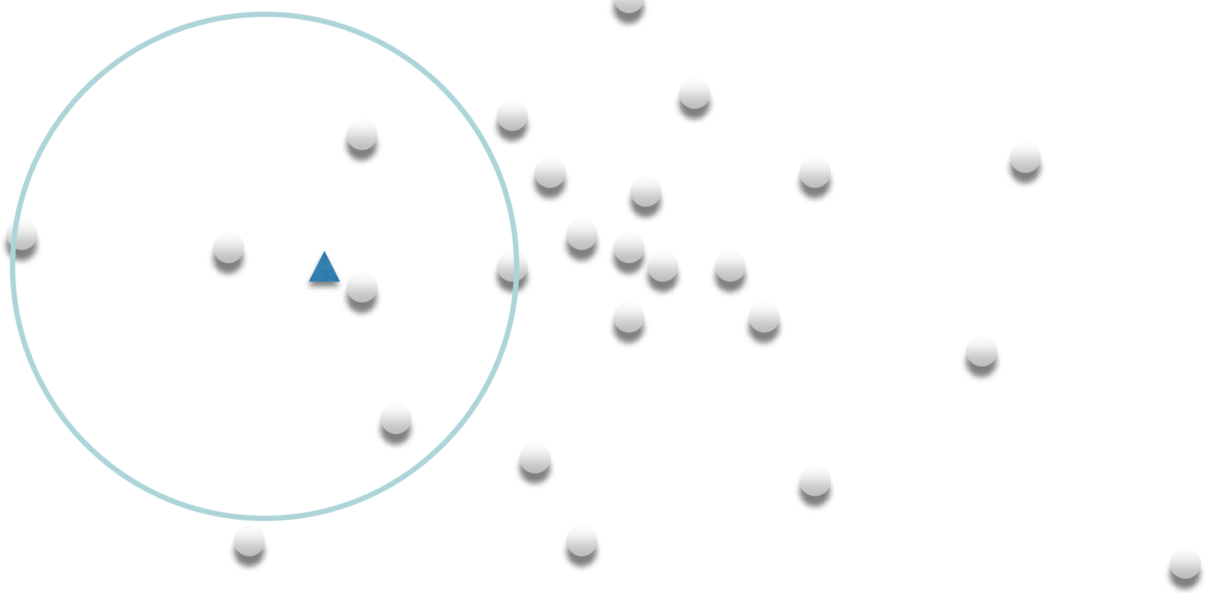


Mean Shift Algorithm

A ‘mode seeking’ algorithm

Fukunaga & Hostetler (1975)

Compute the **mean**

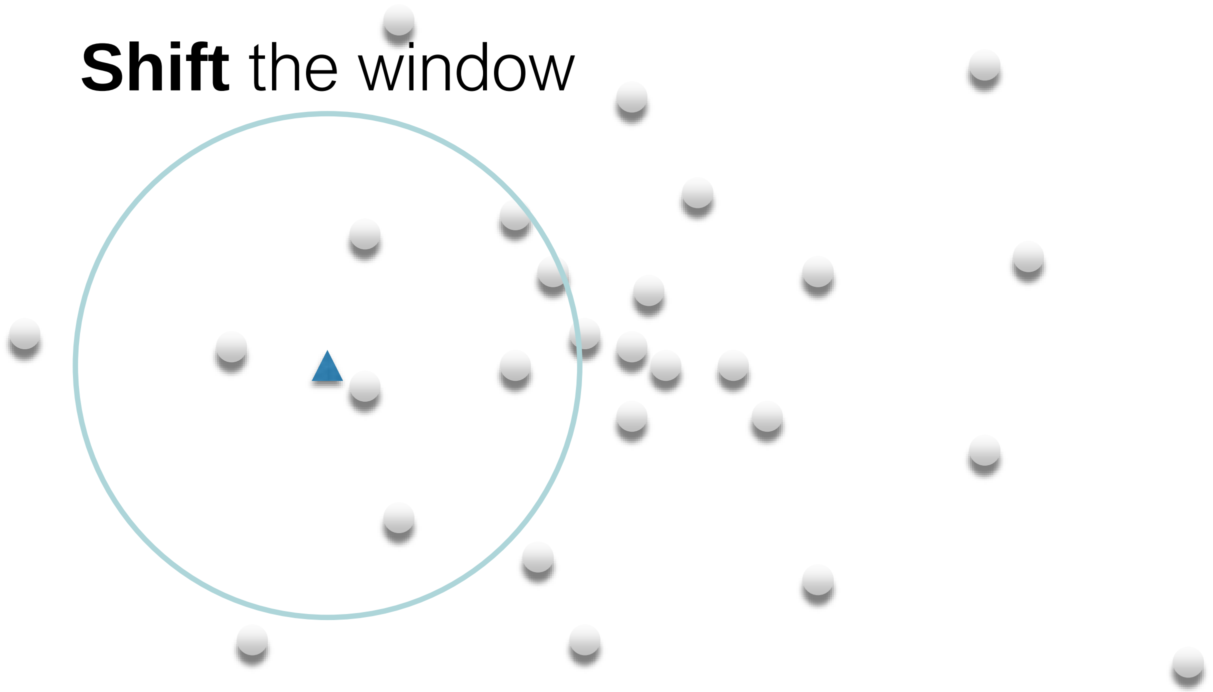


Mean Shift Algorithm

A ‘mode seeking’ algorithm

Fukunaga & Hostetler (1975)

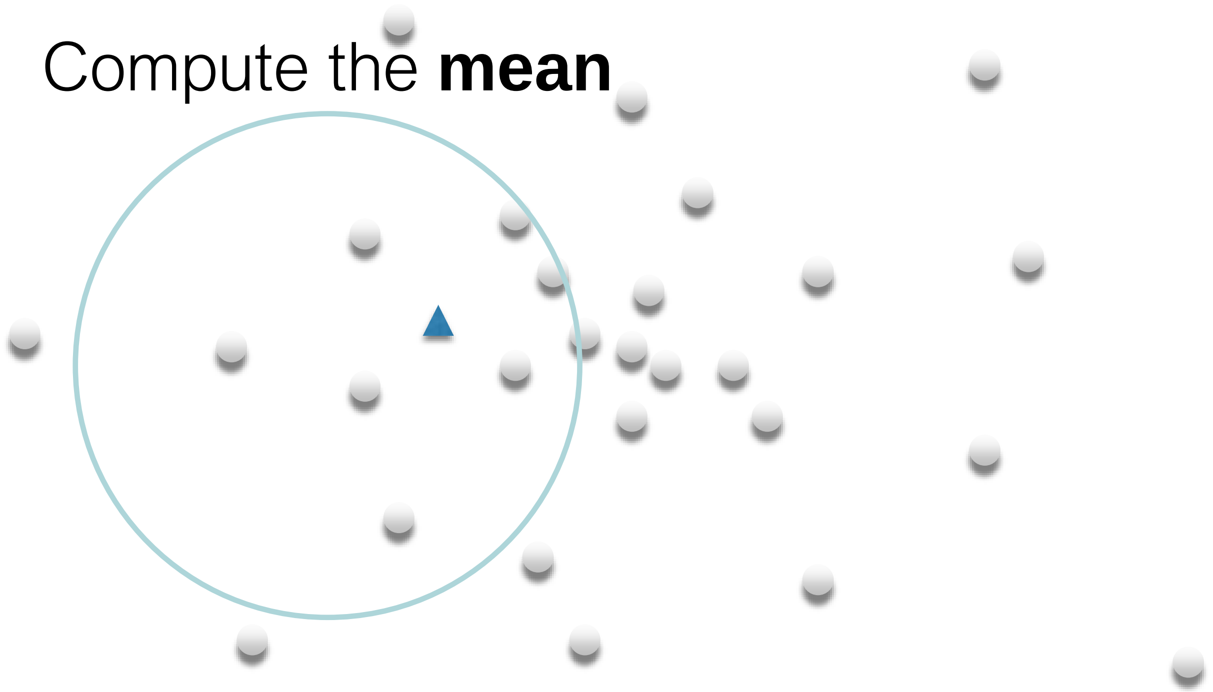
Shift the window



Mean Shift Algorithm

A ‘mode seeking’ algorithm

Fukunaga & Hostetler (1975)

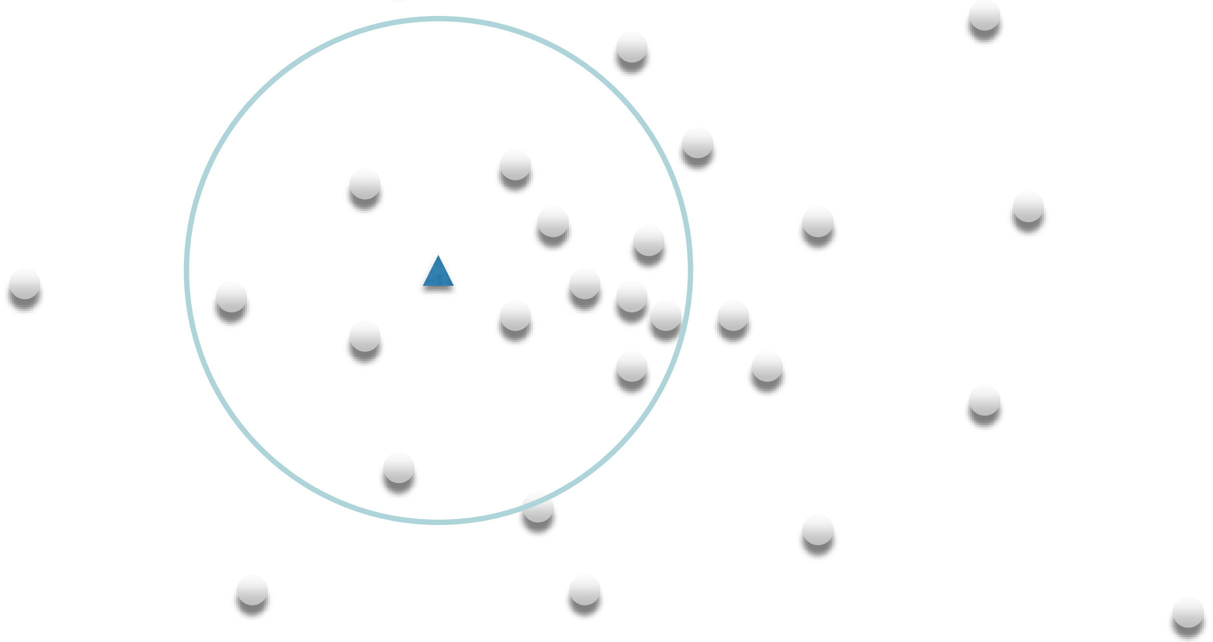


Mean Shift Algorithm

A 'mode seeking' algorithm

Fukunaga & Hostetler (1975)

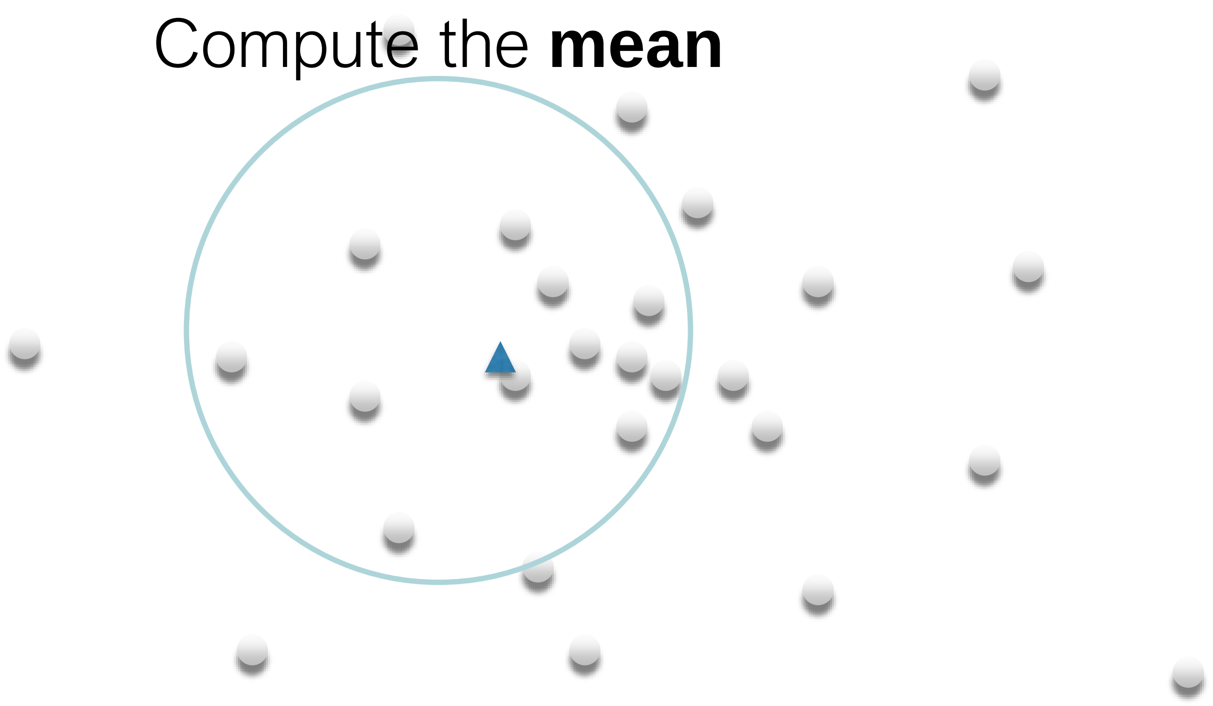
Shift the window



Mean Shift Algorithm

A 'mode seeking' algorithm

Fukunaga & Hostetler (1975)

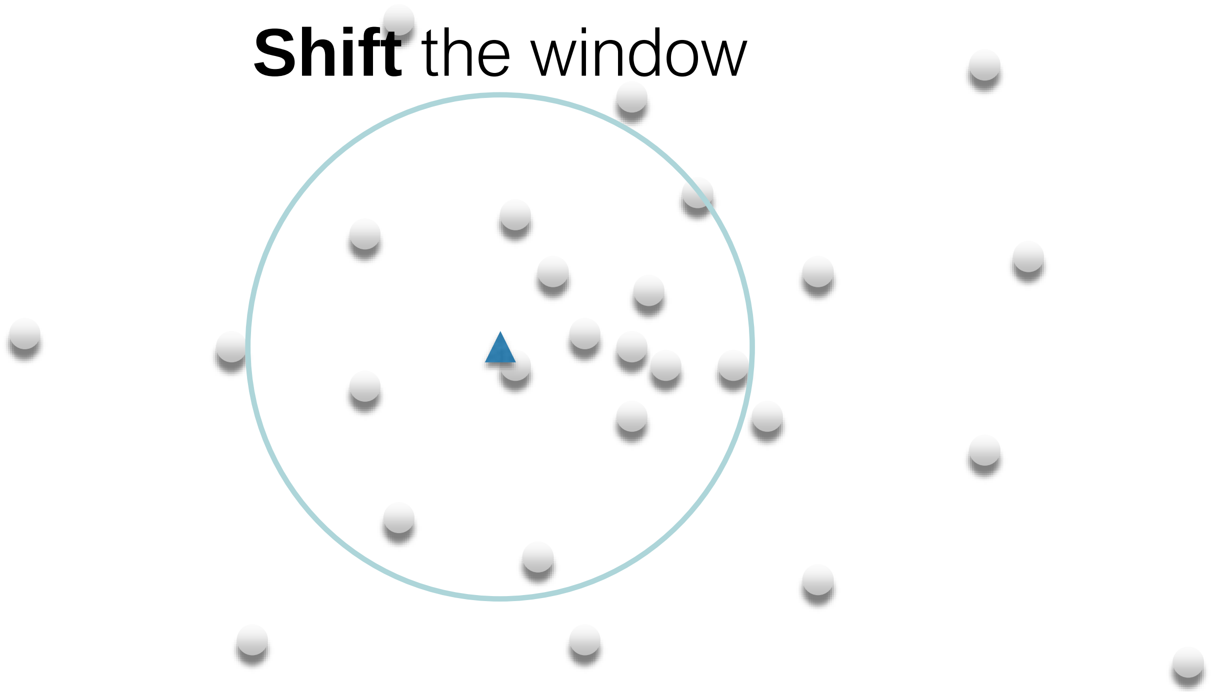


Mean Shift Algorithm

A ‘mode seeking’ algorithm

Fukunaga & Hostetler (1975)

Shift the window

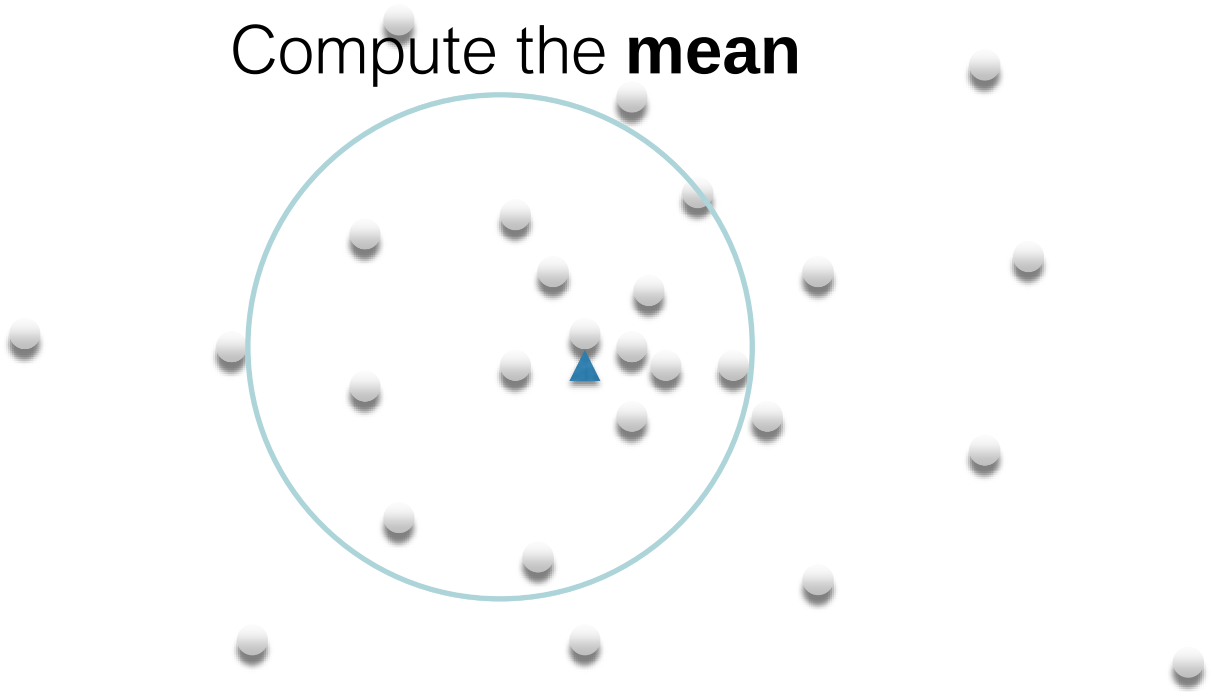


Mean Shift Algorithm

A ‘mode seeking’ algorithm

Fukunaga & Hostetler (1975)

Compute the **mean**

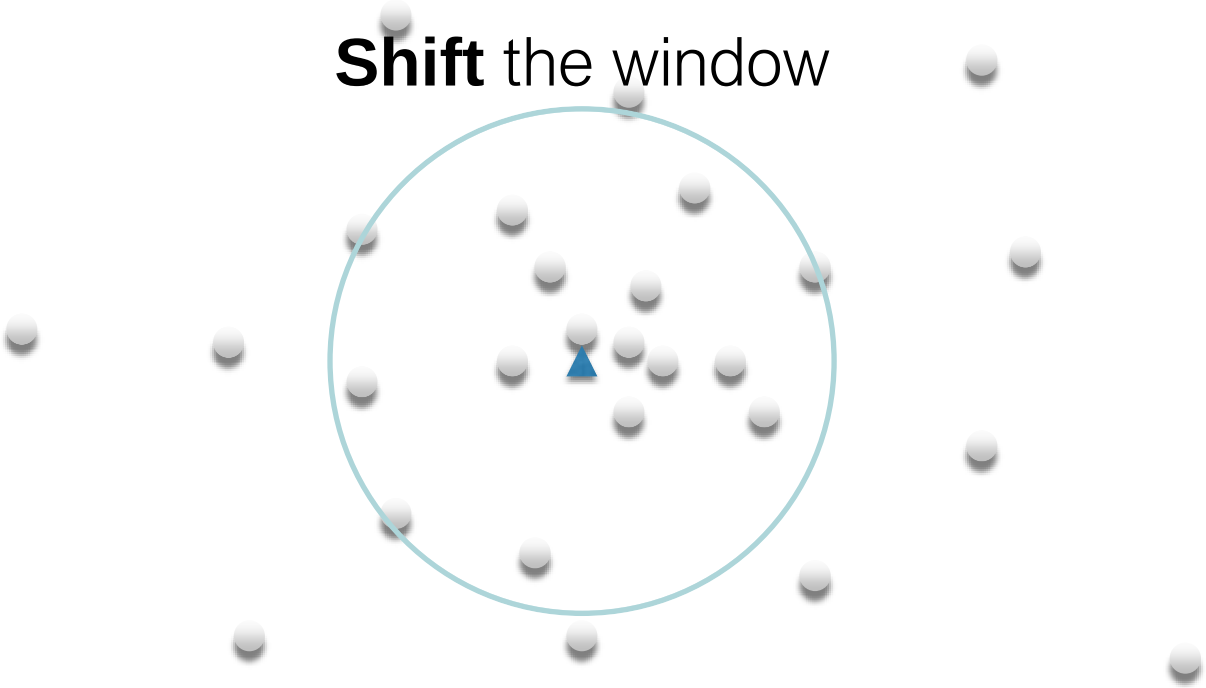


Mean Shift Algorithm

A ‘mode seeking’ algorithm

Fukunaga & Hostetler (1975)

Shift the window

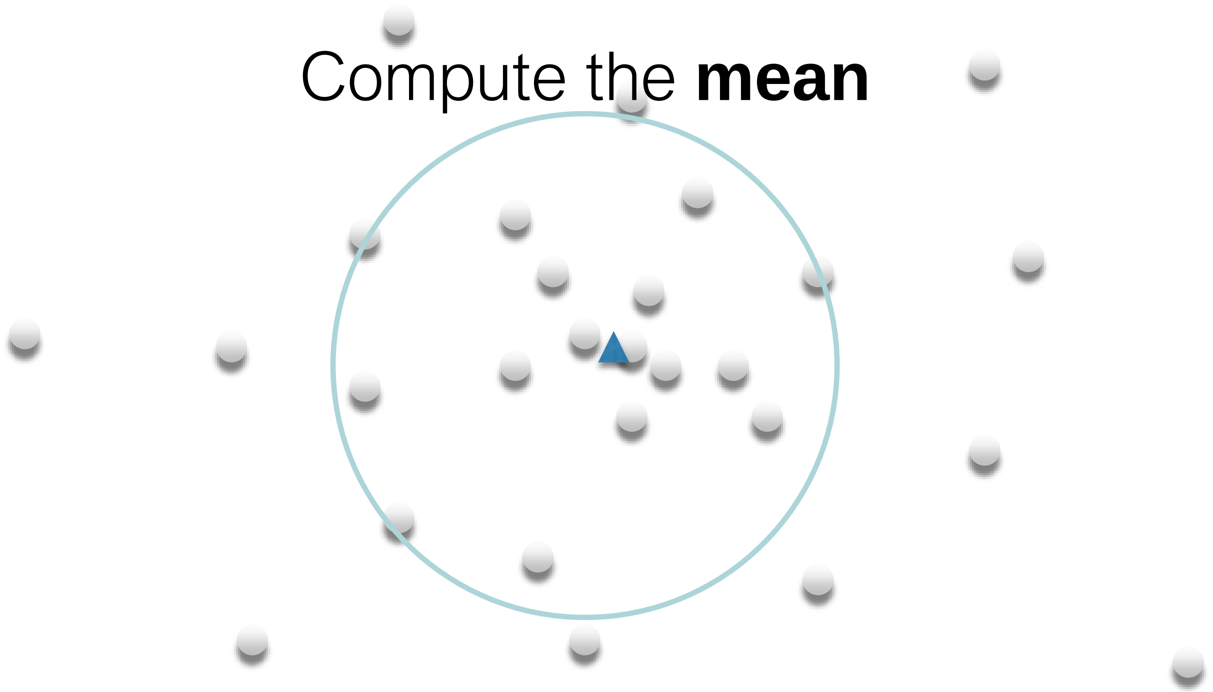


Mean Shift Algorithm

A ‘mode seeking’ algorithm

Fukunaga & Hostetler (1975)

Compute the **mean**

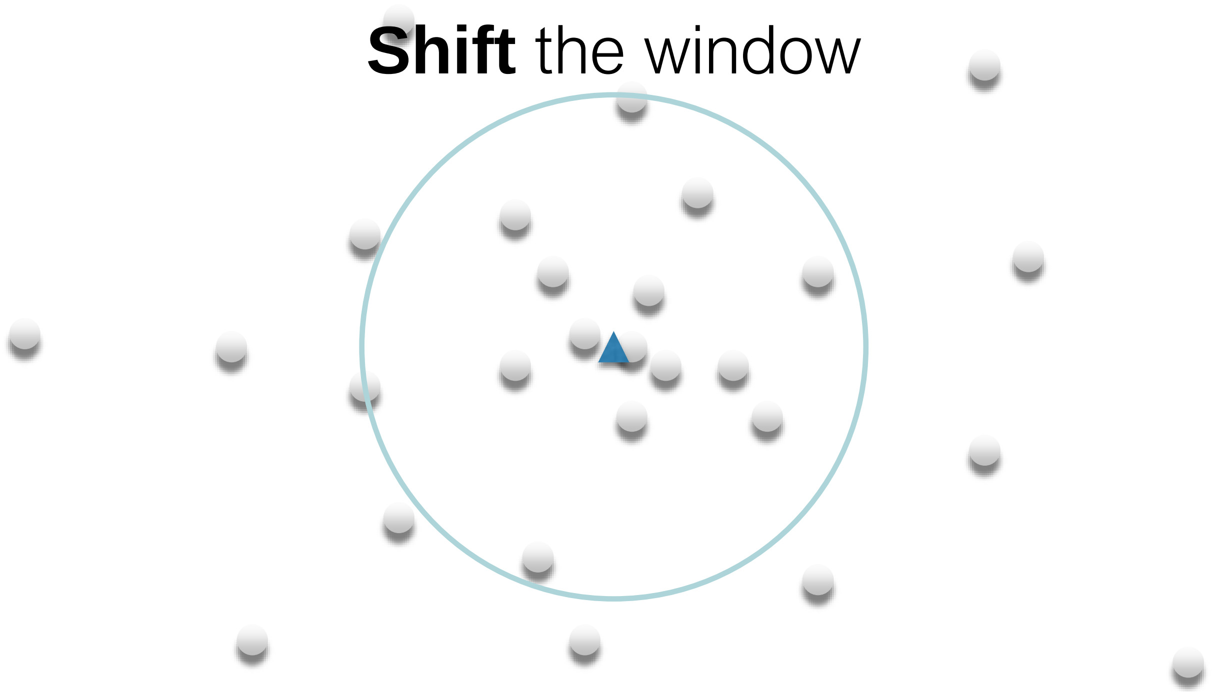


Mean Shift Algorithm

A ‘mode seeking’ algorithm

Fukunaga & Hostetler (1975)

Shift the window



Mean-Shift Algorithm

Initialize $\boldsymbol{x}$ place we start

While $v(\boldsymbol{x}) > \epsilon$ shift values becomes really small

1. Compute mean-shift

$$m(\boldsymbol{x}) = \frac{\sum_s K(\boldsymbol{x}, \boldsymbol{x}_s)\boldsymbol{x}_s}{\sum_s K(\boldsymbol{x}, \boldsymbol{x}_s)}$$

compute the 'mean'

$$v(\boldsymbol{x}) = m(\boldsymbol{x}) - \boldsymbol{x}$$

compute the 'shift'

2. Update $\boldsymbol{x} \leftarrow \boldsymbol{x} + \boldsymbol{v}(\boldsymbol{x})$ update the point

Mean-Shift Tracking

Given a set of points:

$$\{\boldsymbol{x}_s\}_{s=1}^{S} \qquad \boldsymbol{x}_s \in \mathcal{R}^d$$

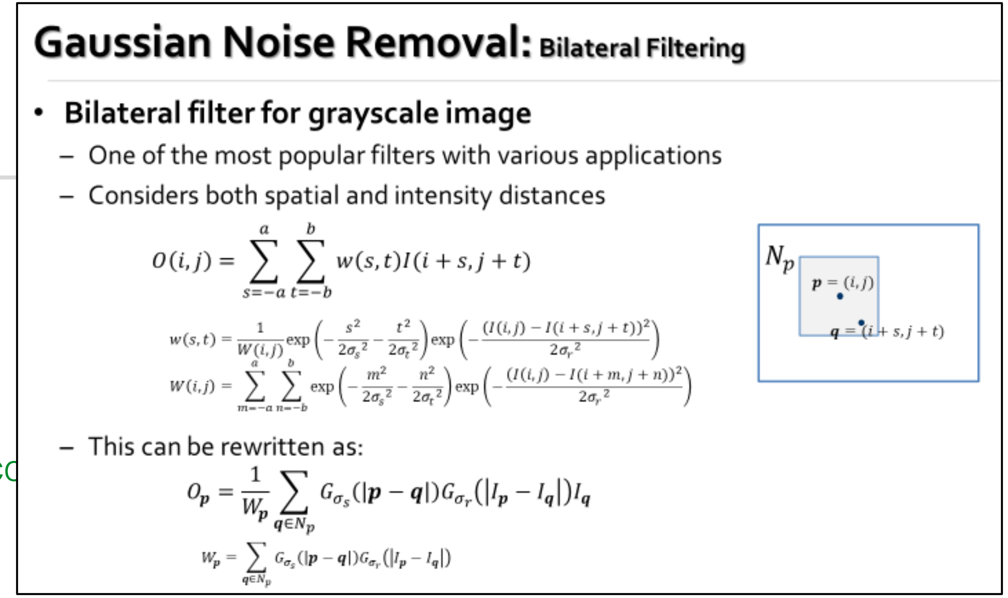
and a kernel:

$$K(\boldsymbol{x}, \boldsymbol{x}_s) = g\left(\frac{\|\boldsymbol{x} - \boldsymbol{x}_s\|^2}{h}\right)$$

Find the mean sample point:

$$\boldsymbol{x}$$

Mean-Shift Algorithm



Gaussian Noise Removal: Bilateral Filtering

- **Bilateral filter for grayscale image**
  - One of the most popular filters with various applications
  - Considers both spatial and intensity distances

$$O(i,j) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) I(i+s, j+t)$$

$$w(s,t) = \frac{1}{W(i,j)} \exp\left(-\frac{s^2}{2\sigma_s^2} - \frac{t^2}{2\sigma_t^2}\right) \exp\left(-\frac{(I(i,j) - I(i+s, j+t))^2}{2\sigma_r^2}\right)$$

$$W(i,j) = \sum_{m=-a}^{a} \sum_{n=-b}^{b} \exp\left(-\frac{m^2}{2\sigma_s^2} - \frac{n^2}{2\sigma_t^2}\right) \exp\left(-\frac{(I(i,j) - I(i+m, j+n))^2}{2\sigma_r^2}\right)$$

  - This can be rewritten as:

$$O_{\boldsymbol{p}} = \frac{1}{W_{\boldsymbol{p}}} \sum_{\boldsymbol{q} \in N_p} G_{\sigma_s}(|\boldsymbol{p} - \boldsymbol{q}|) G_{\sigma_r}(|I_{\boldsymbol{p}} - I_{\boldsymbol{q}}|) I_{\boldsymbol{q}}$$

$$W_{\boldsymbol{p}} = \sum_{\boldsymbol{q} \in N_p} G_{\sigma_s}(|\boldsymbol{p} - \boldsymbol{q}|) G_{\sigma_r}(|I_{\boldsymbol{p}} - I_{\boldsymbol{q}}|)$$

Initialize $\boldsymbol{x}$ — place we start

While $v(\boldsymbol{x}) > \epsilon$ — shift values bec

1. Compute mean-shift

$$m(\boldsymbol{x}) = \frac{\sum_s K(\boldsymbol{x}, \boldsymbol{x}_s)\boldsymbol{x}_s}{\sum_s K(\boldsymbol{x}, \boldsymbol{x}_s)}$$ compute the 'mean'

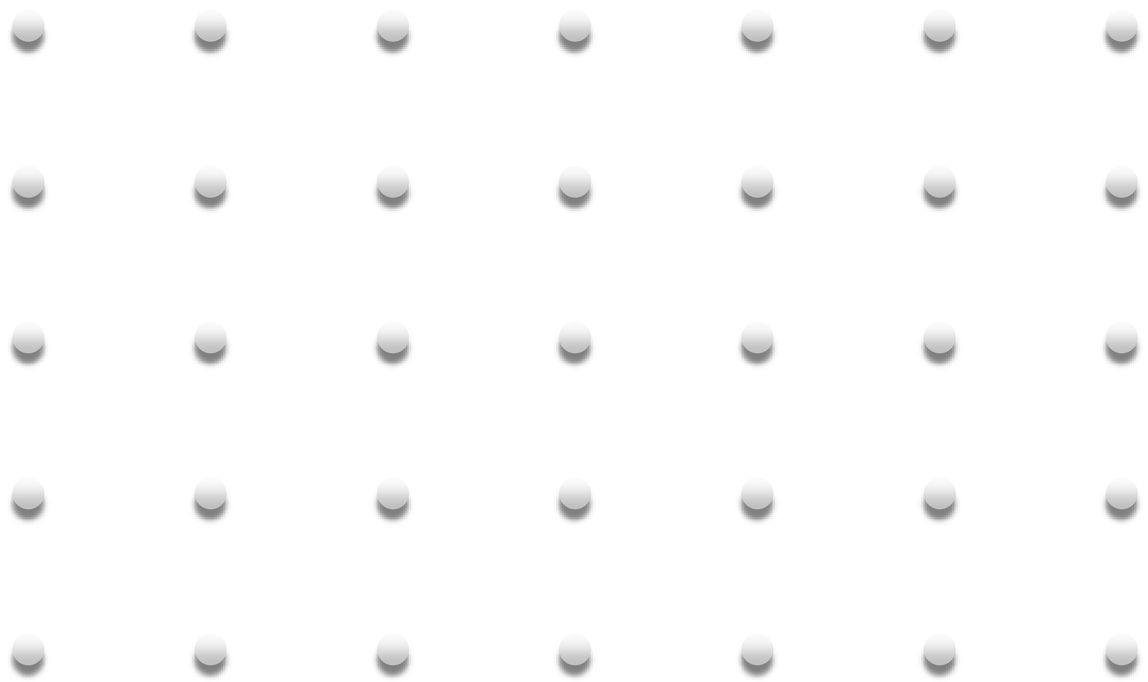
$$v(\boldsymbol{x}) = m(\boldsymbol{x}) - \boldsymbol{x}$$ compute the 'shift'

2. Update $\boldsymbol{x} \leftarrow \boldsymbol{x} + \boldsymbol{v}(\boldsymbol{x})$ — update the point

Everything up to now has been about distributions over samples…

Dealing with Images

Pixels for a lattice, spatial density is the same everywhere!



What can we do?

Dealing with Images

same

Consider a set of points: $\{\boldsymbol{x}_s\}_{s=1}^{S}$ $\boldsymbol{x}_s \in \mathcal{R}^d$

Associated weights: $w(\boldsymbol{x}_s)$

Sample mean: $m(\boldsymbol{x}) = \dfrac{\sum_s K(\boldsymbol{x}, \boldsymbol{x}_s) w(\boldsymbol{x}_s) \boldsymbol{x}_s}{\sum_s K(\boldsymbol{x}, \boldsymbol{x}_s) w(\boldsymbol{x}_s)}$

same

Mean shift: $m(\boldsymbol{x}) - \boldsymbol{x}$

Mean-Shift Algorithm (for images)

Initialize $\boldsymbol{x}$

While $v(\boldsymbol{x}) > \epsilon$

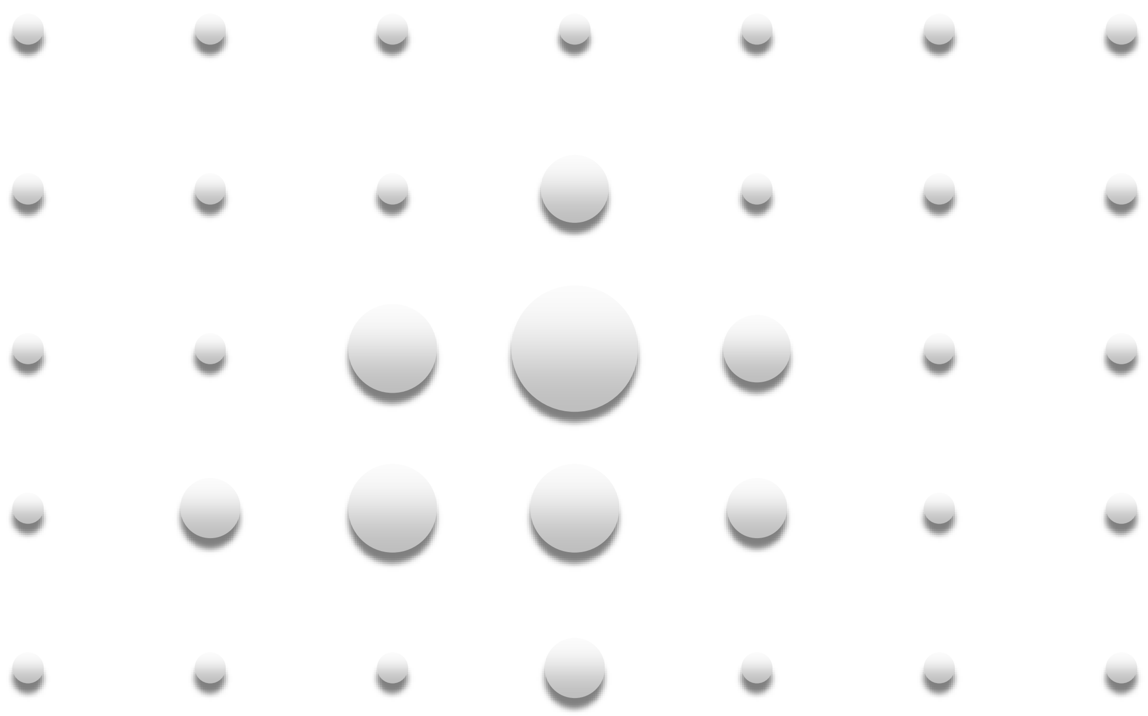
1. Compute mean-shift

$$m(\boldsymbol{x}) = \frac{\sum_s K(\boldsymbol{x}, \boldsymbol{x}_s) w(\boldsymbol{x}_s) \boldsymbol{x}_s}{\sum_s K(\boldsymbol{x}, \boldsymbol{x}_s) w(\boldsymbol{x}_s)}$$

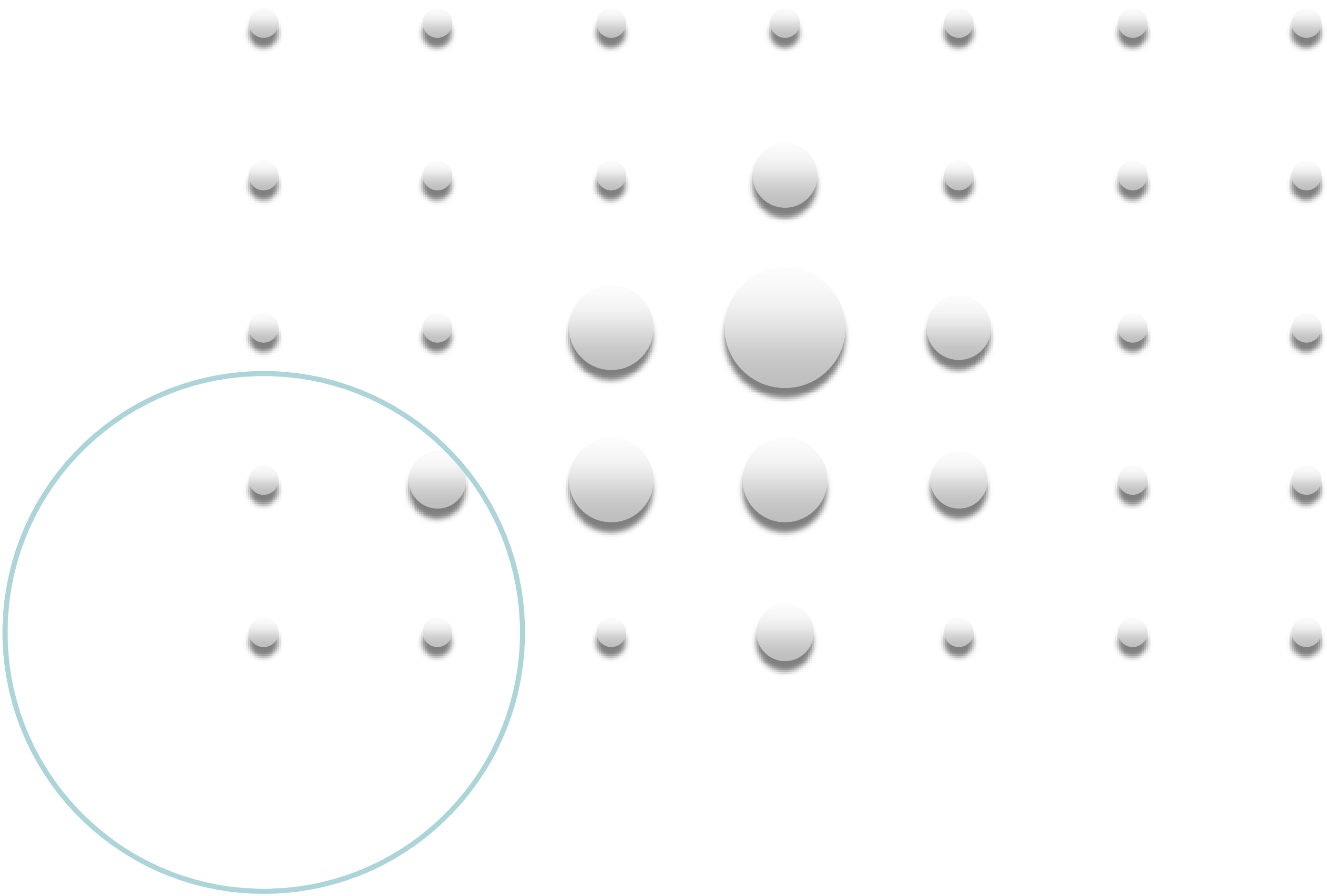
$$v(\boldsymbol{x}) = m(\boldsymbol{x}) - \boldsymbol{x}$$

2. Update $\boldsymbol{x} \leftarrow \boldsymbol{x} + \boldsymbol{v}(\boldsymbol{x})$

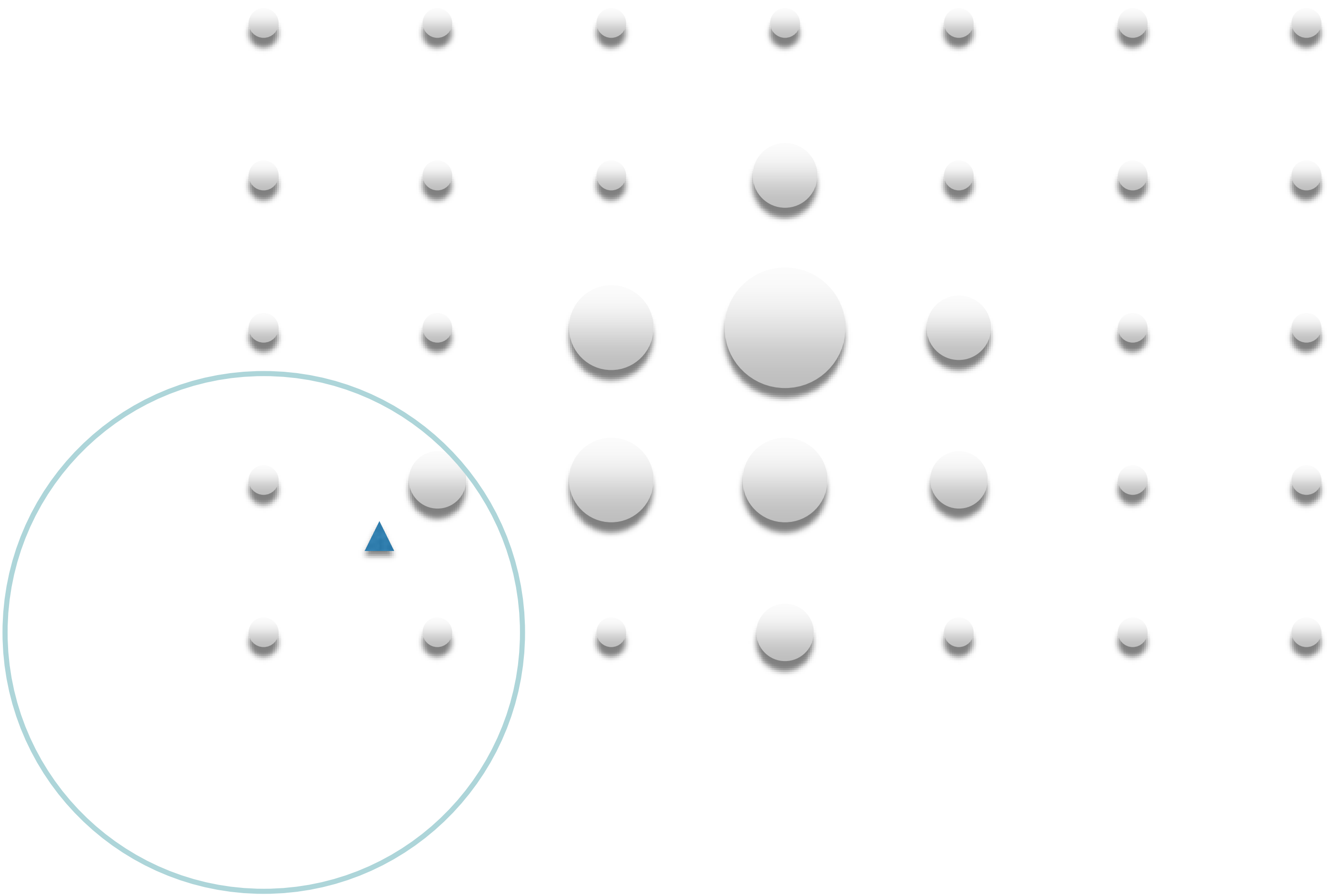
For images, each pixel is point with a weight



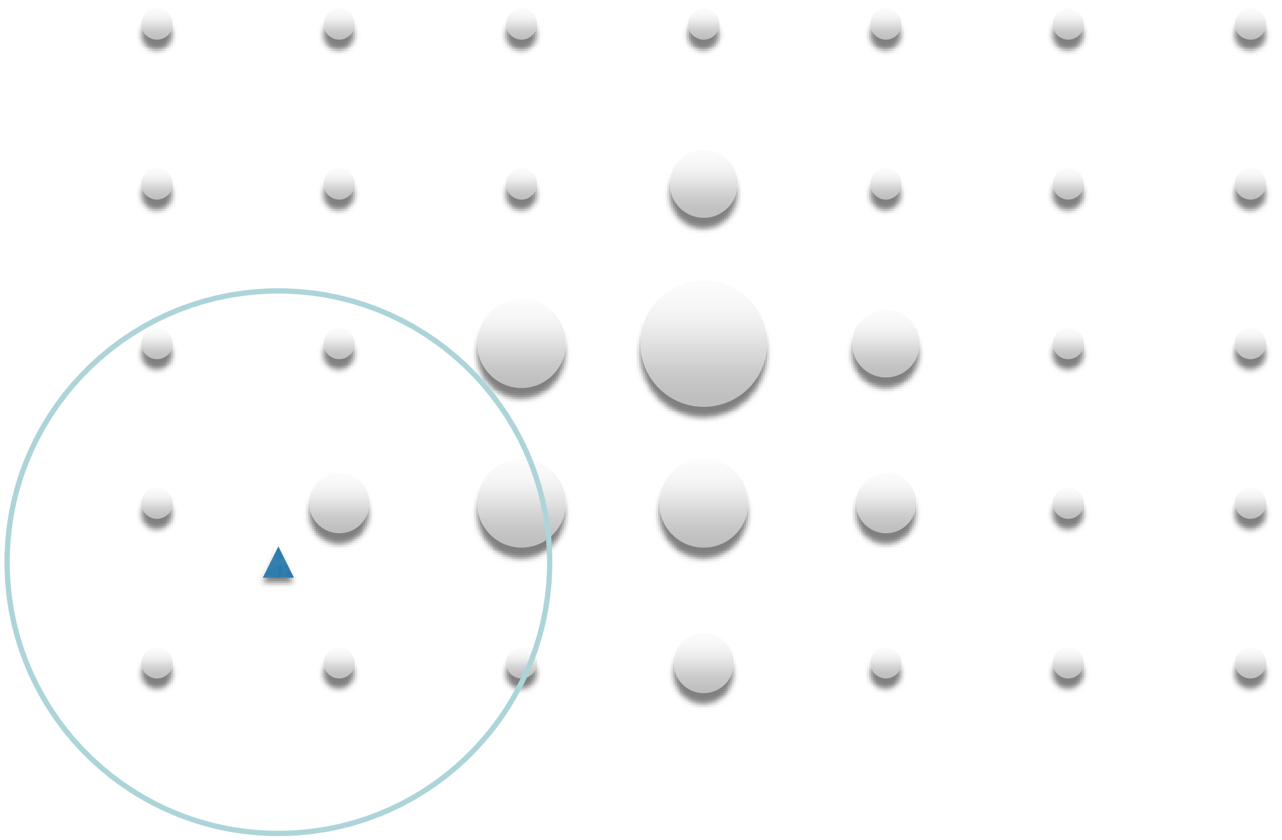
For images, each pixel is point with a weight



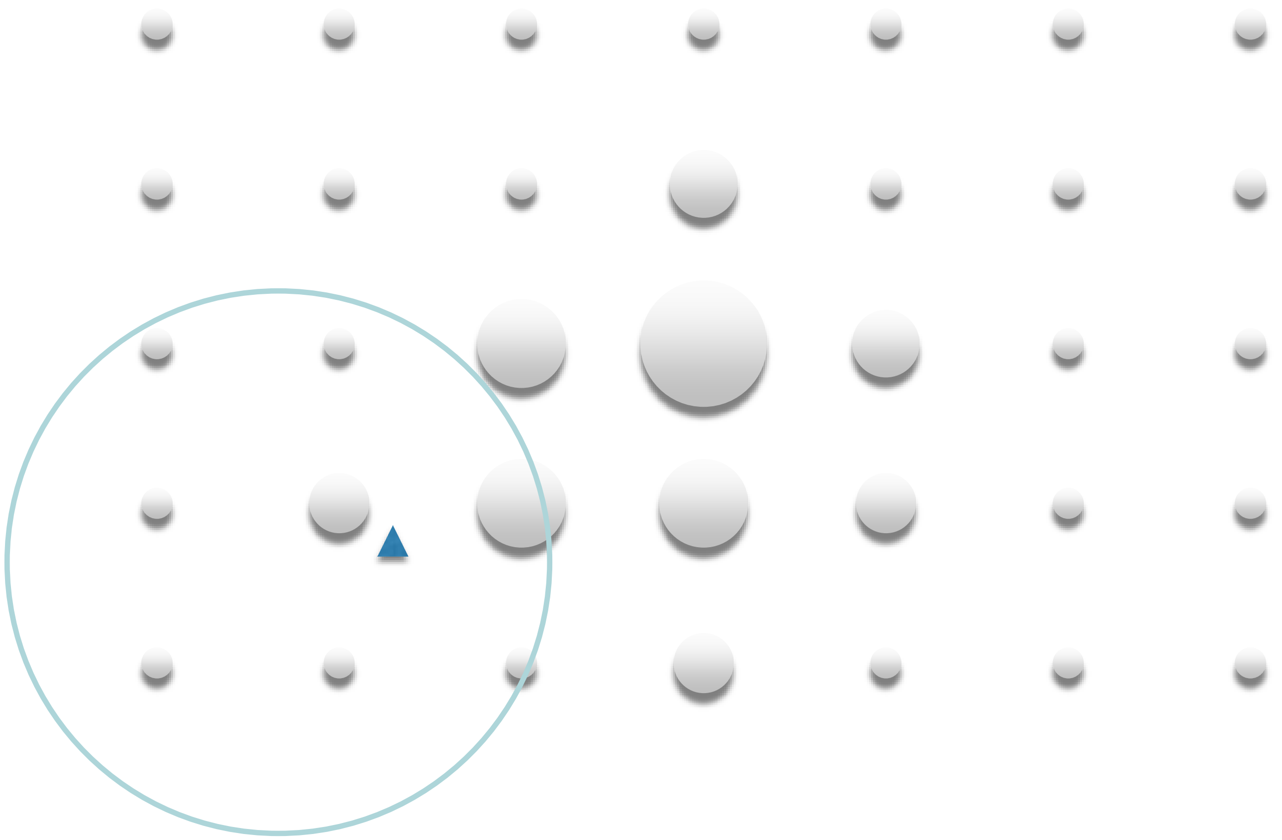
For images, each pixel is point with a weight



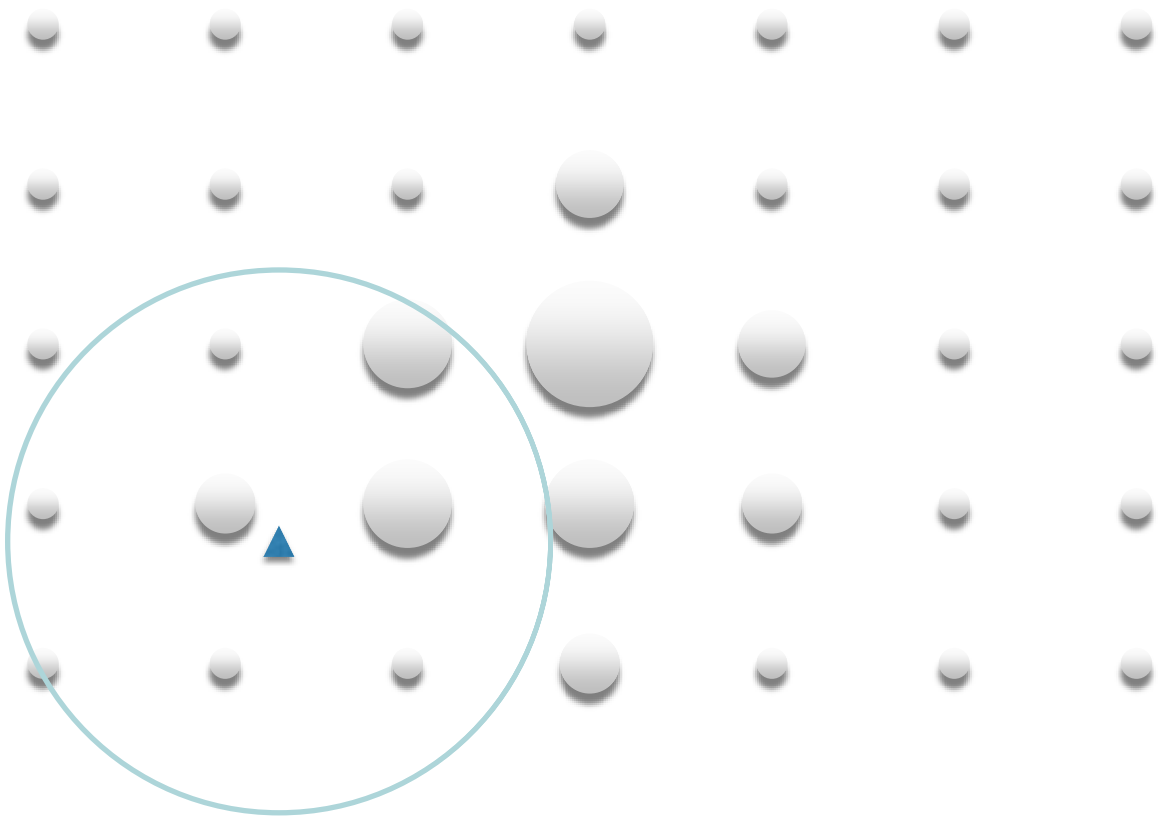
For images, each pixel is point with a weight



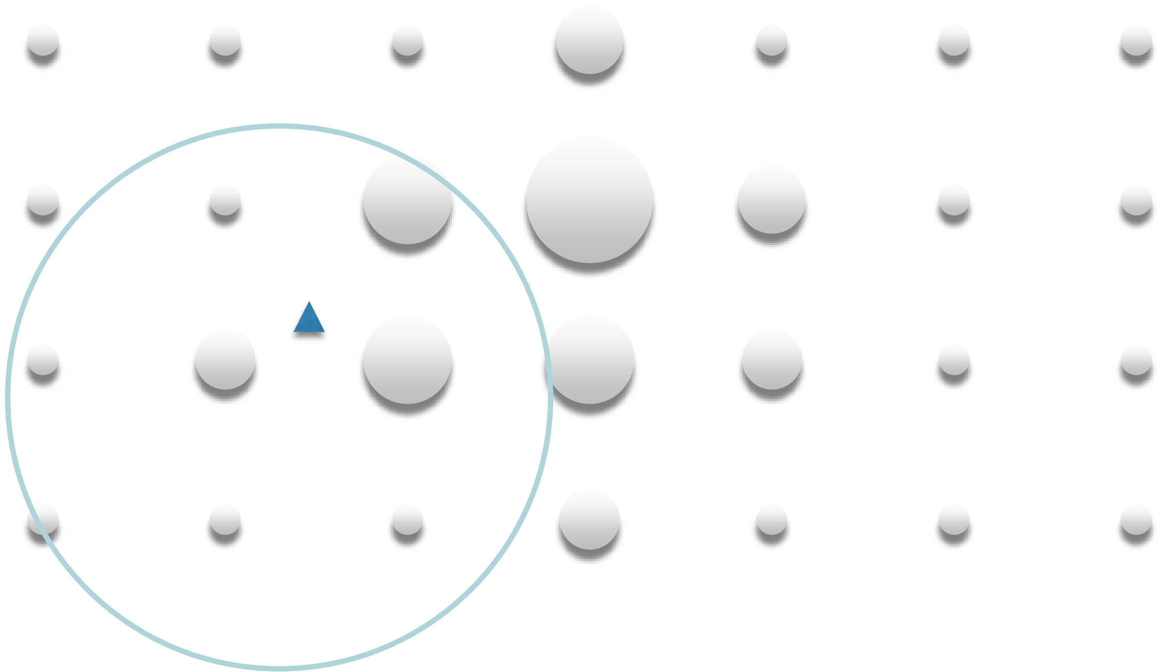
For images, each pixel is point with a weight



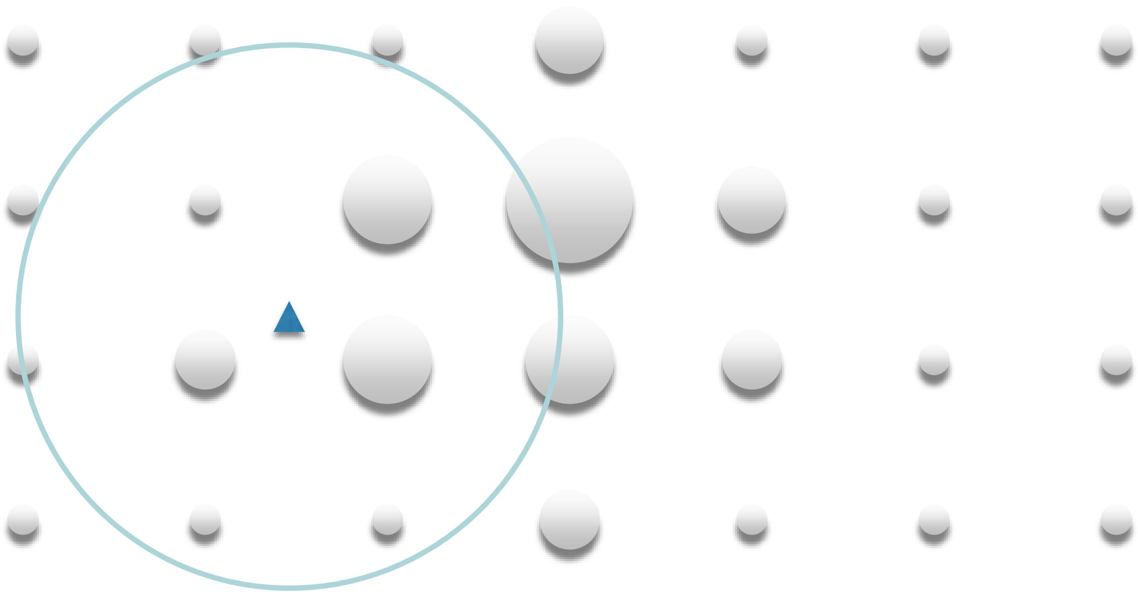
For images, each pixel is point with a weight



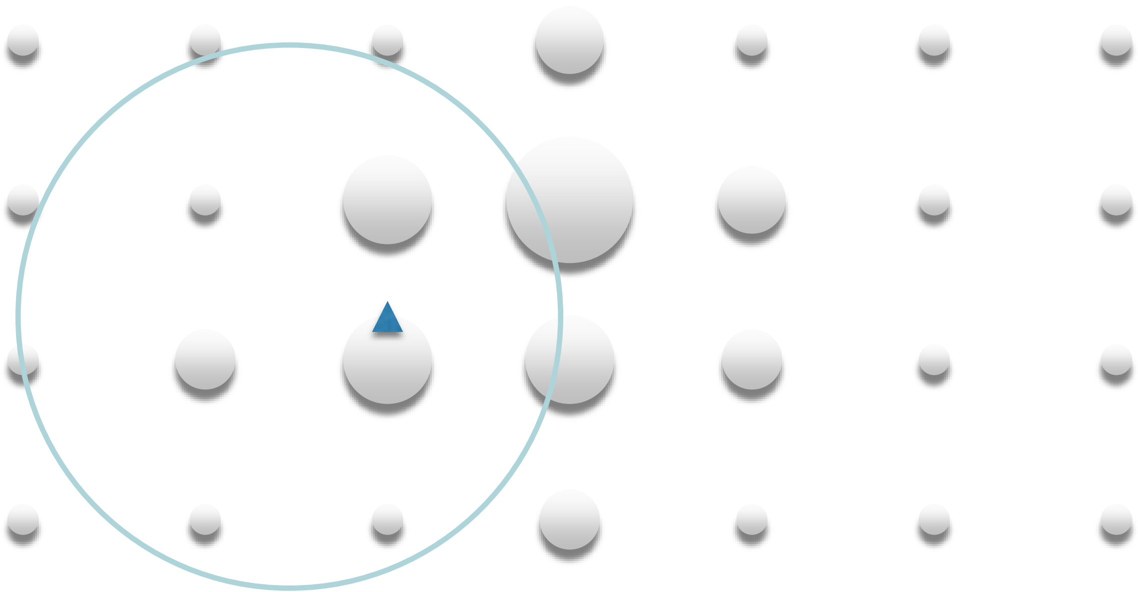
For images, each pixel is point with a weight



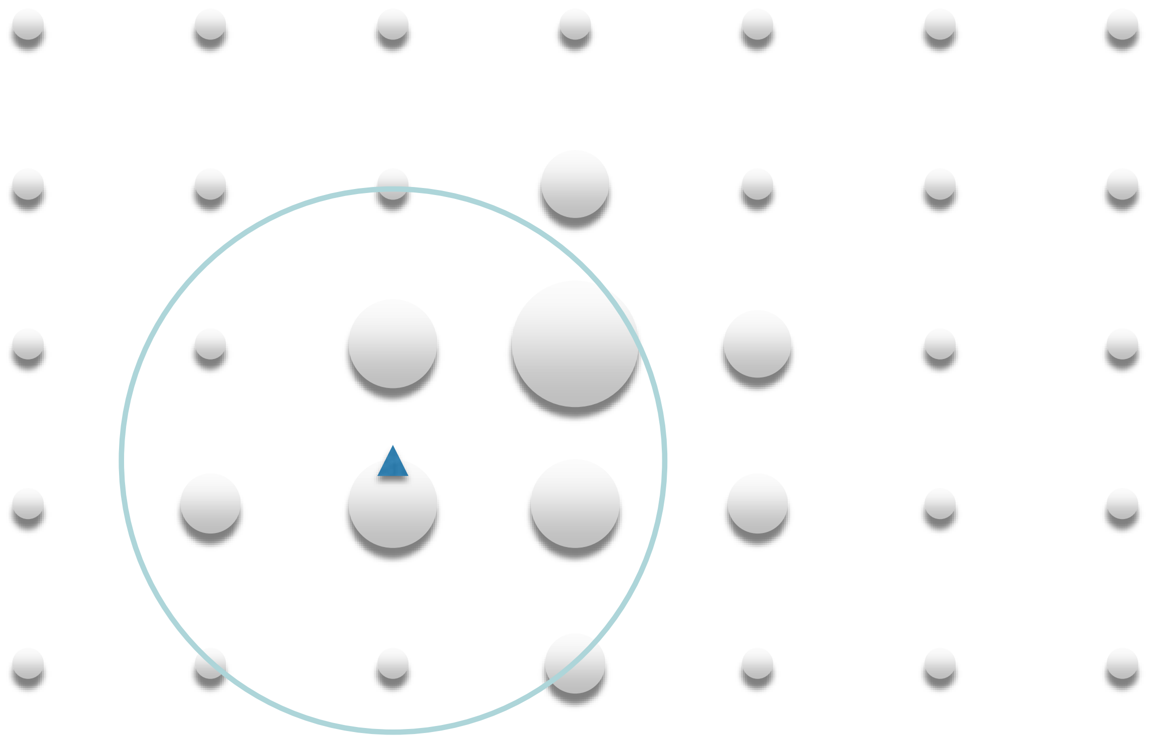
For images, each pixel is point with a weight



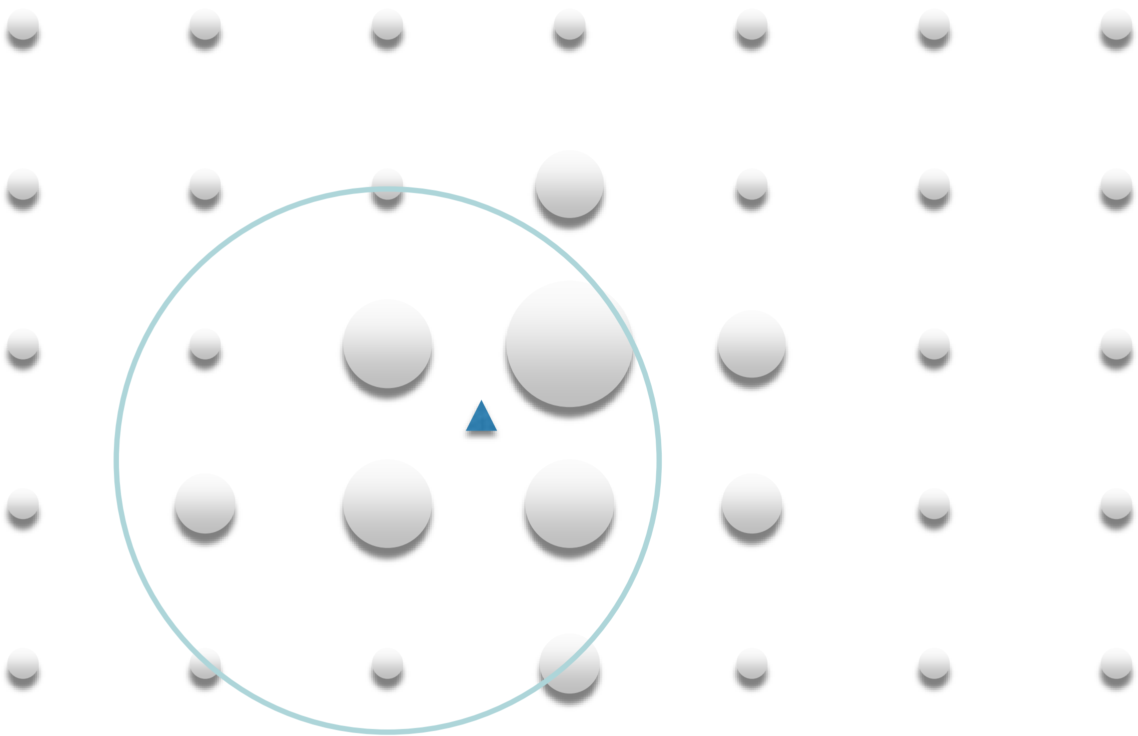
For images, each pixel is point with a weight



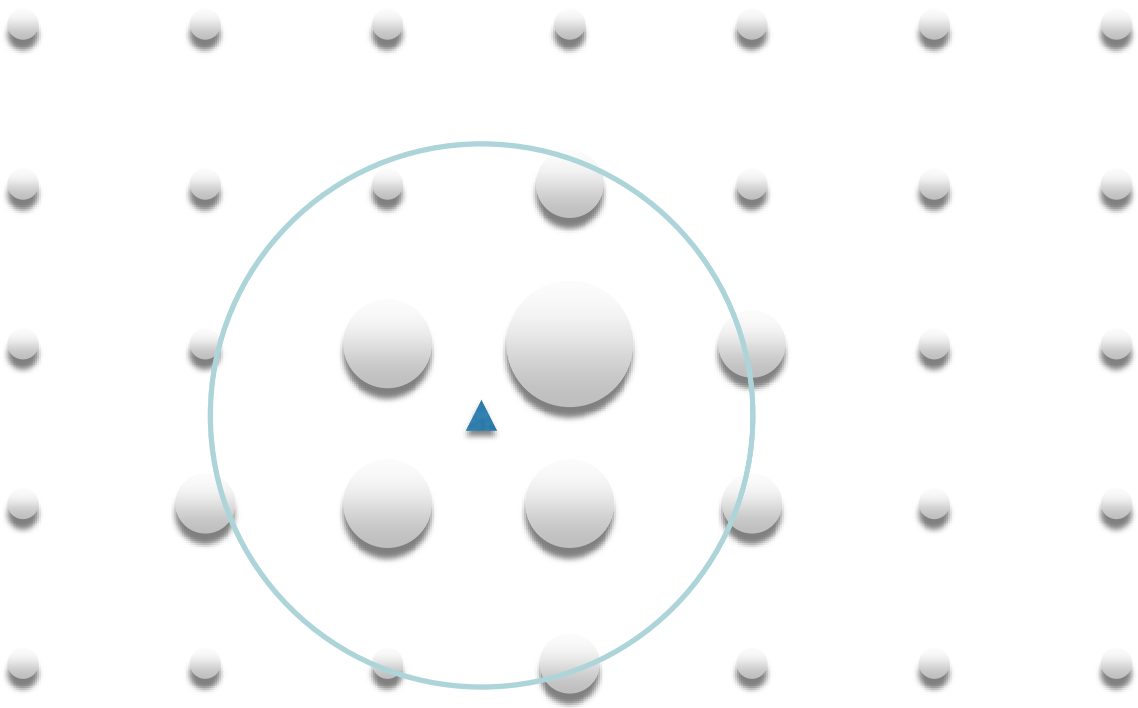
For images, each pixel is point with a weight



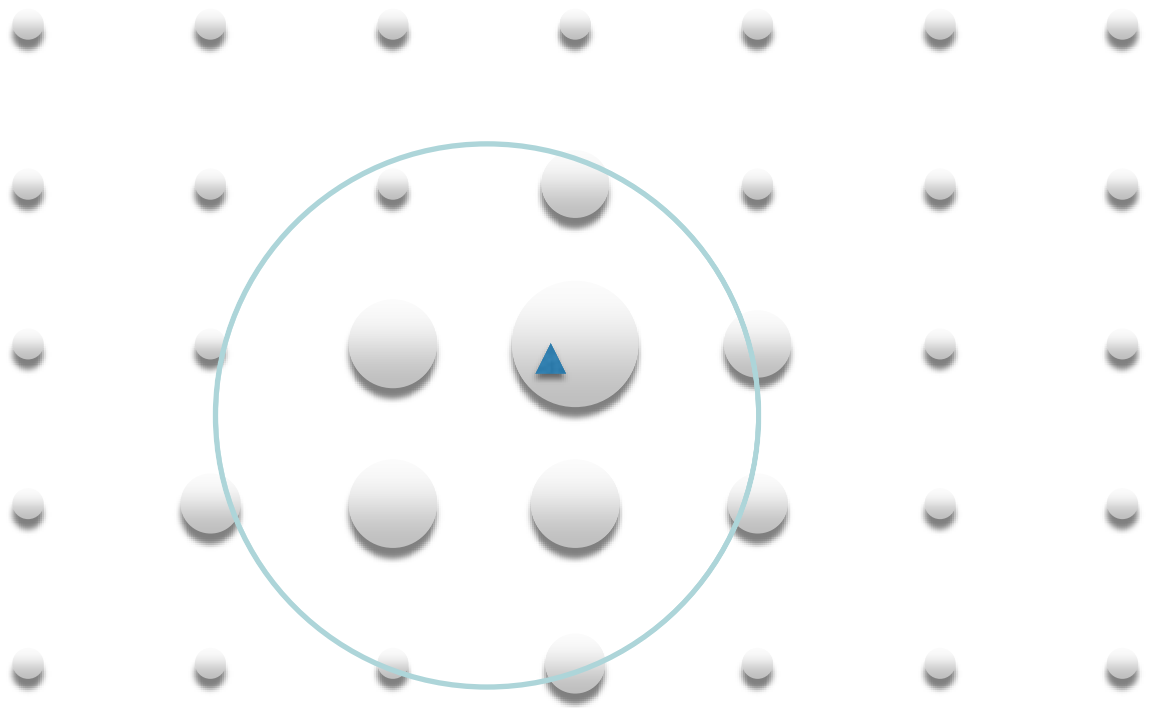
For images, each pixel is point with a weight



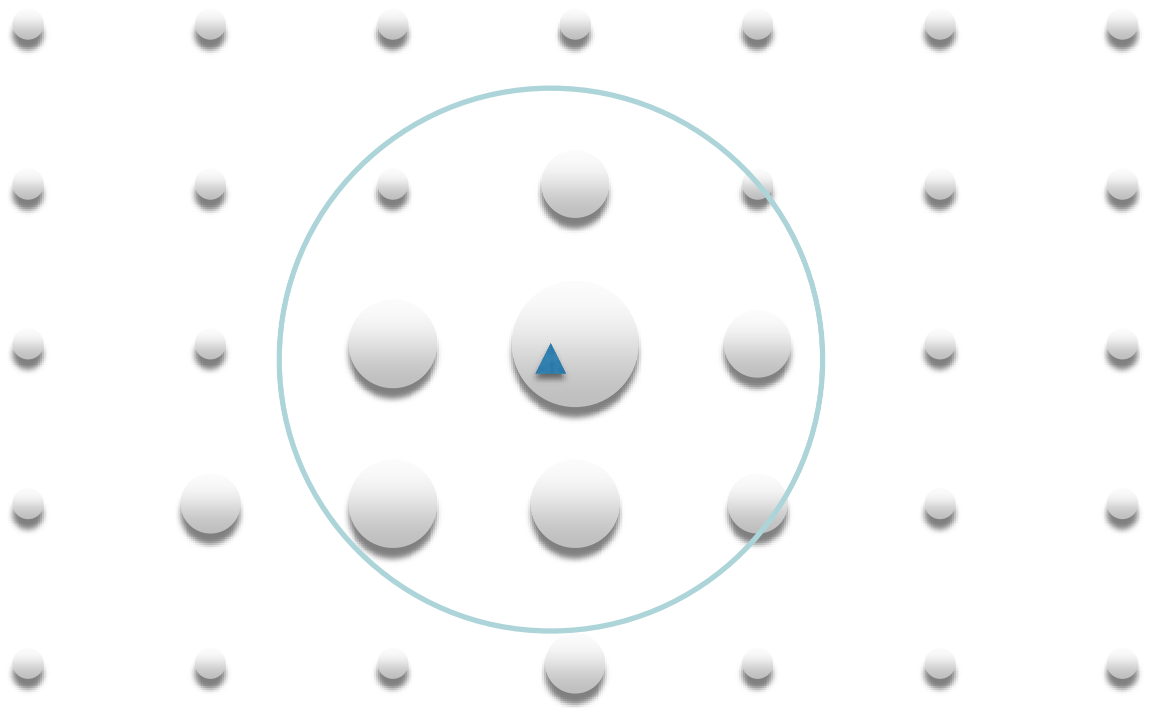
For images, each pixel is point with a weight



For images, each pixel is point with a weight



For images, each pixel is point with a weight



Mean-Shift procedure

- **Simple Mean Shift procedure:**
  - Compute mean shift vector
  - Translate the Kernel window by m(x)

Initialize $\boldsymbol{x}$

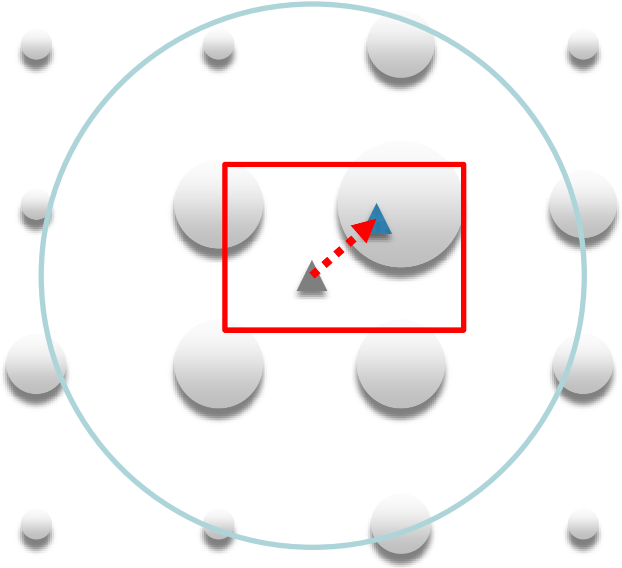
While $v(\boldsymbol{x}) > \epsilon$

1. Compute mean-shift

$$m(\boldsymbol{x}) = \frac{\sum_s K(\boldsymbol{x}, \boldsymbol{x}_s) w(\boldsymbol{x}_s) \boldsymbol{x}_s}{\sum_s K(\boldsymbol{x}, \boldsymbol{x}_s) w(\boldsymbol{x}_s)}$$

$$\boxed{v(\boldsymbol{x}) = m(\boldsymbol{x}) - \boldsymbol{x}}$$

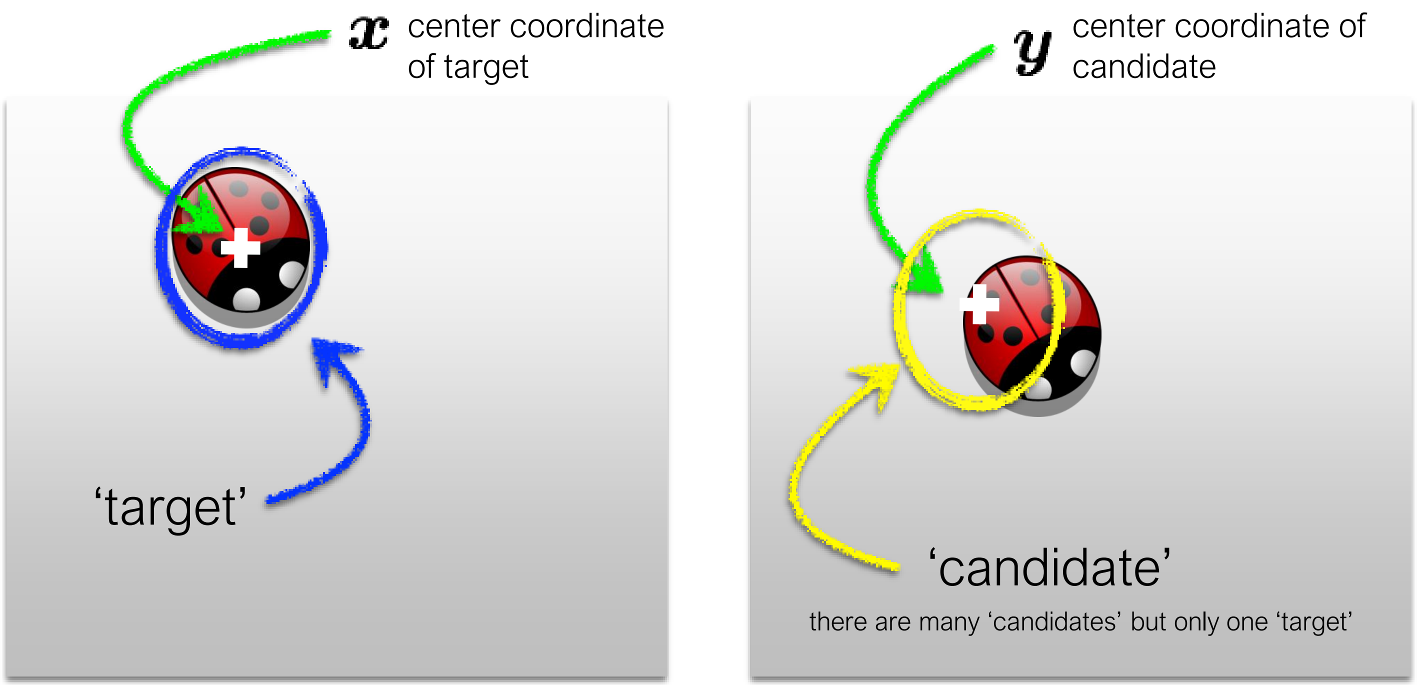
2. Update $\boldsymbol{x} \leftarrow \boldsymbol{x} + v(\boldsymbol{x})$



Finally… mean shift tracking in video!

Mean shift tracking in video

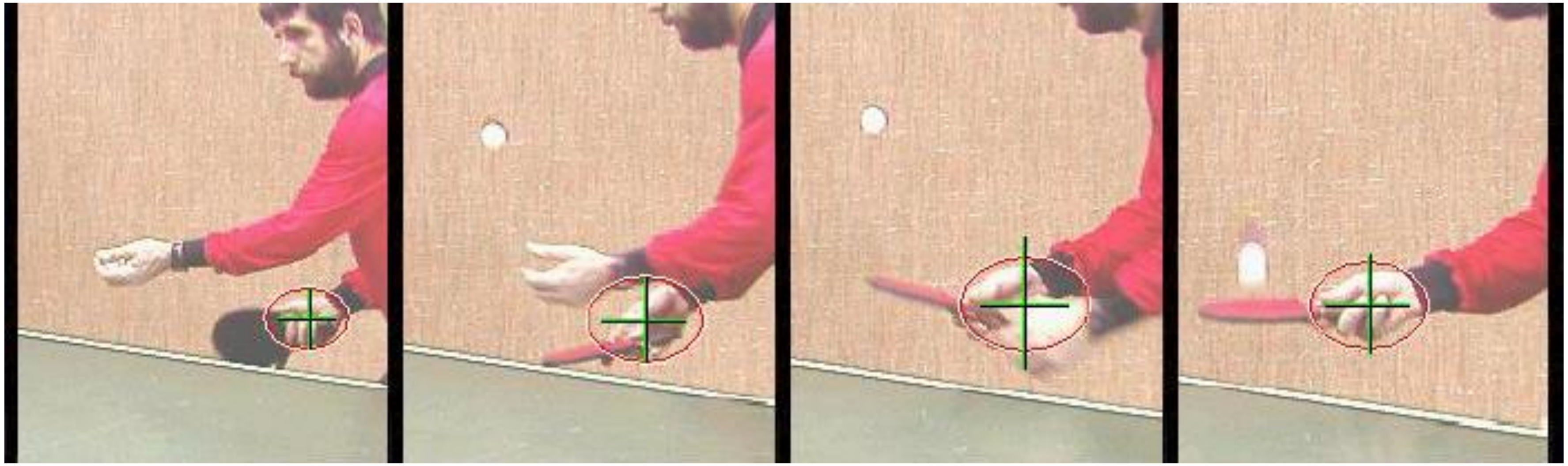
Goal: find the best candidate location in frame 2



Frame 1 Frame 2

Use the mean shift algorithm
to find the best candidate location

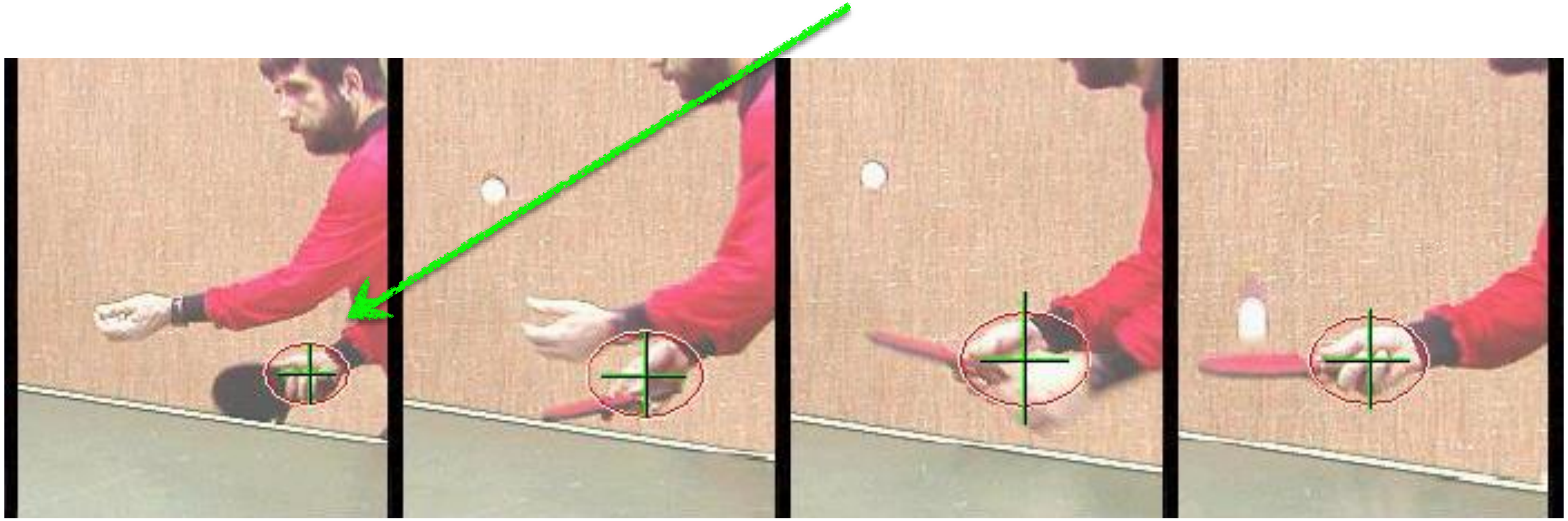
Non-rigid object tracking



hand tracking

Non-rigid object tracking

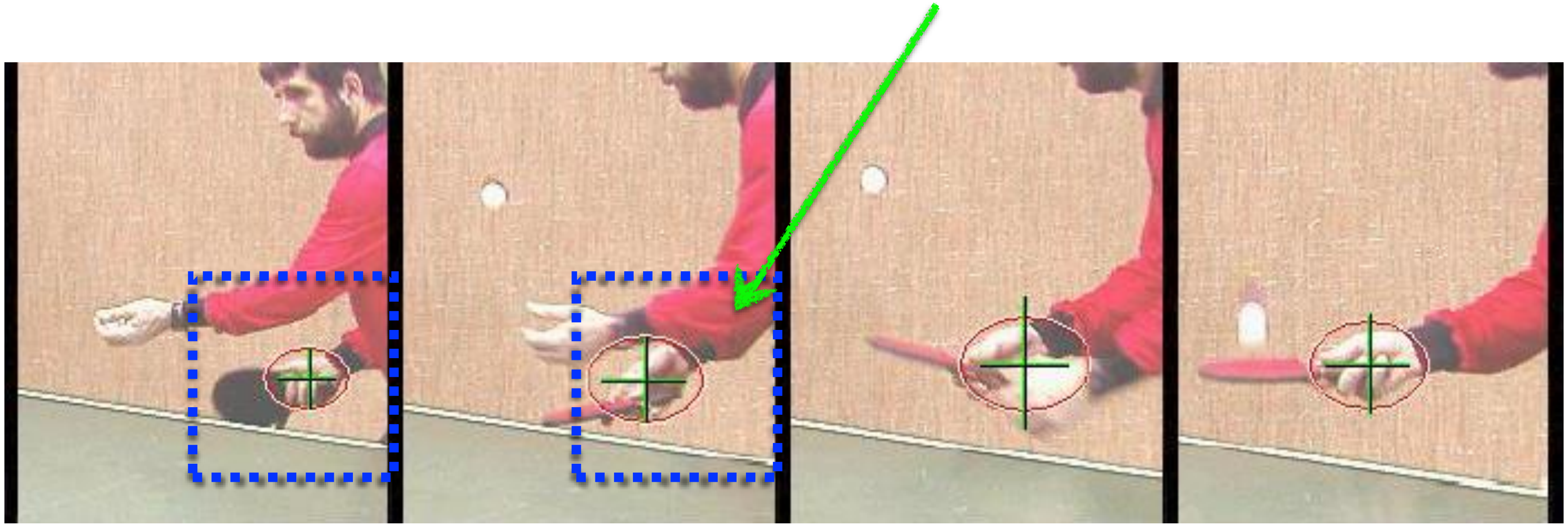
Compute a descriptor for the target



Target

Non-rigid object tracking

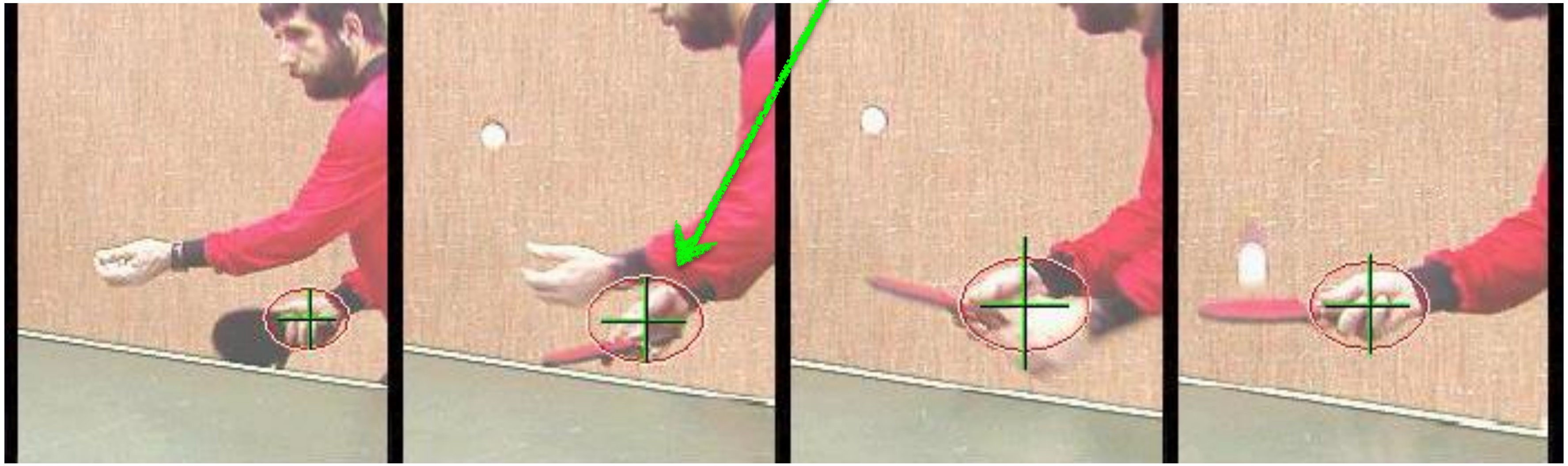
Search for similar descriptor in neighborhood in next frame



Target Candidate

Non-rigid object tracking

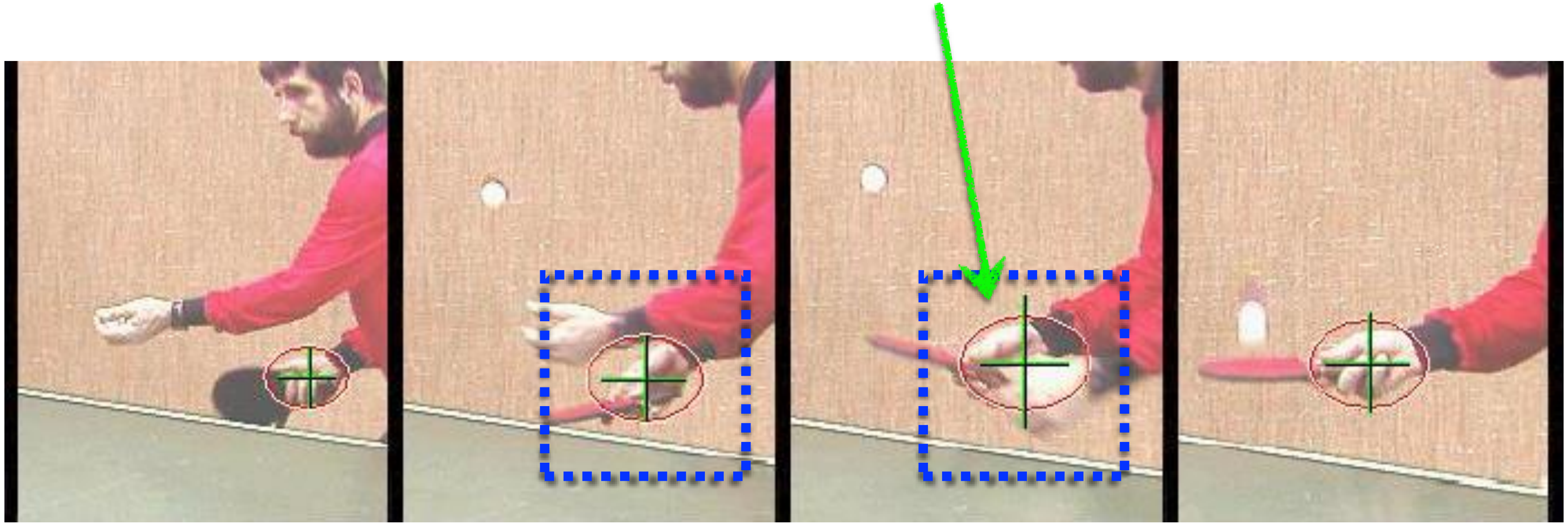
Compute a descriptor for the new target



Target

Non-rigid object tracking

Search for similar descriptor in neighborhood in next frame



Target Candidate

How do we model the target and candidate regions?

Modelling the target

M-dimensional **target** descriptor

$$\boldsymbol{q} = \{q_1, \ldots, q_M\}$$

(centered at target center)

a 'fancy' (confusing) way to write a weighted histogram

$$q_m = C \sum_n k(||\boldsymbol{x}_n||^2)\delta[b(\boldsymbol{x}_n) - m]$$

Normalization factor

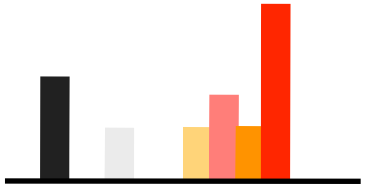
sum over all pixels

function of inverse distance (weight)

Kronecker delta function

quantization function

bin ID



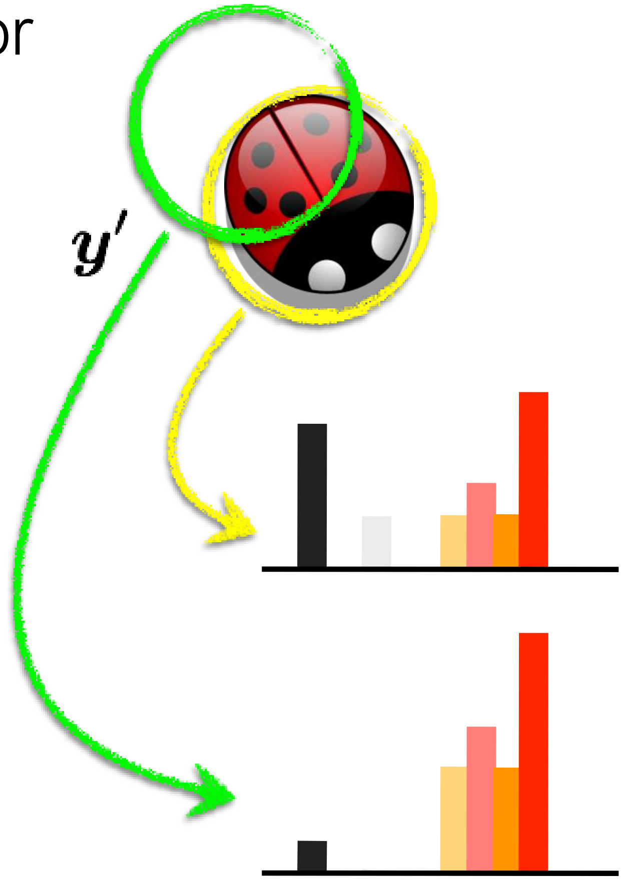
A normalized color histogram (weighted by distance)

Modelling the candidate

M-dimensional **candidate** descriptor

$$\boldsymbol{p}(\boldsymbol{y}) = \{p_1(\boldsymbol{y}), \ldots, p_M(\boldsymbol{y})\}$$

(centered at location **y**)



a weighted histogram at y

$$p_m = C_h \sum_n k\left(\left\|\frac{\boldsymbol{y}-\boldsymbol{x}_n}{h}\right\|^2\right)\delta[b(\boldsymbol{x}_n)-m]$$

bandwidth

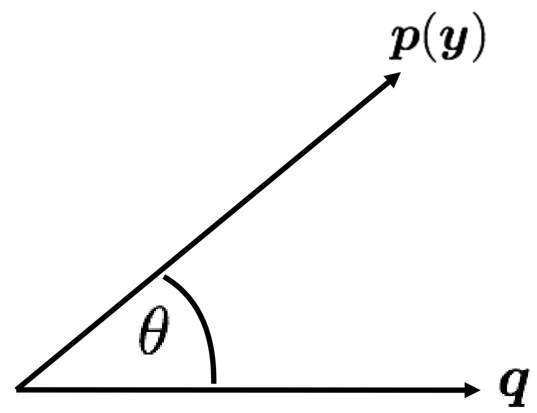
Similarity between the target and candidate

Distance function $$d(\boldsymbol{y}) = \sqrt{1 - \rho[\boldsymbol{p}(\boldsymbol{y}), \boldsymbol{q}]}$$

Bhattacharyya Coefficient $$\rho(y) \equiv \rho[\boldsymbol{p}(\boldsymbol{y}), \boldsymbol{q}] = \sum_m \sqrt{p_m(\boldsymbol{y}) q_u}$$

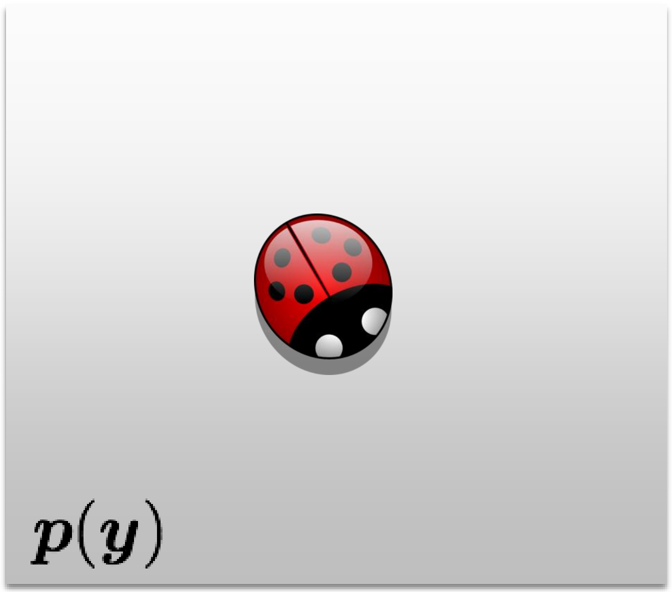
Just the Cosine distance between two unit vectors

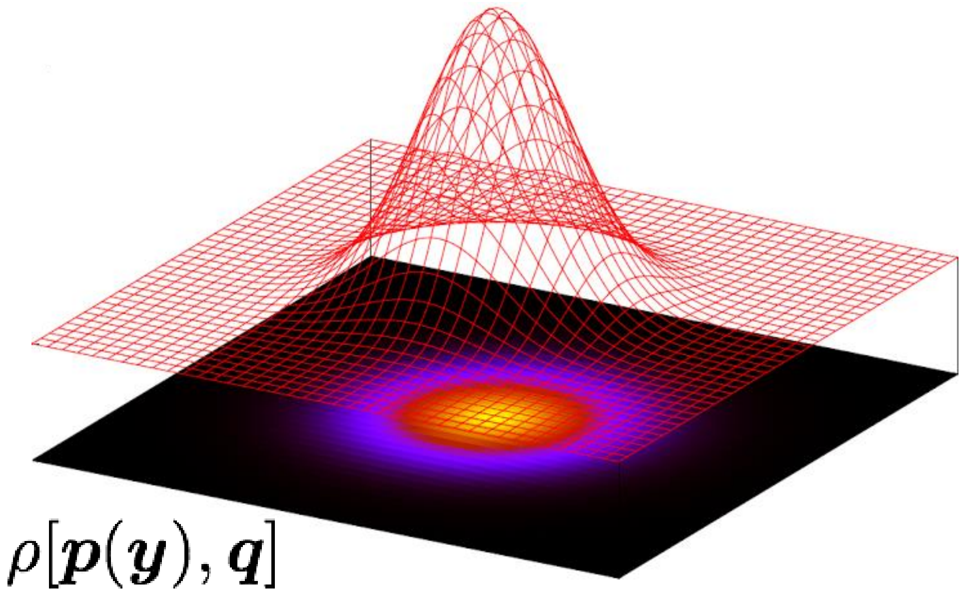
$$\rho(\boldsymbol{y}) = \cos\theta_{\boldsymbol{y}} = \frac{\boldsymbol{p}(\boldsymbol{y})^\top \boldsymbol{q}}{\|\boldsymbol{p}\| \|\boldsymbol{q}\|} = \sum_m \sqrt{p_m(\boldsymbol{y}) q_m}$$



Now we can compute the similarity between a target and multiple candidate regions

target





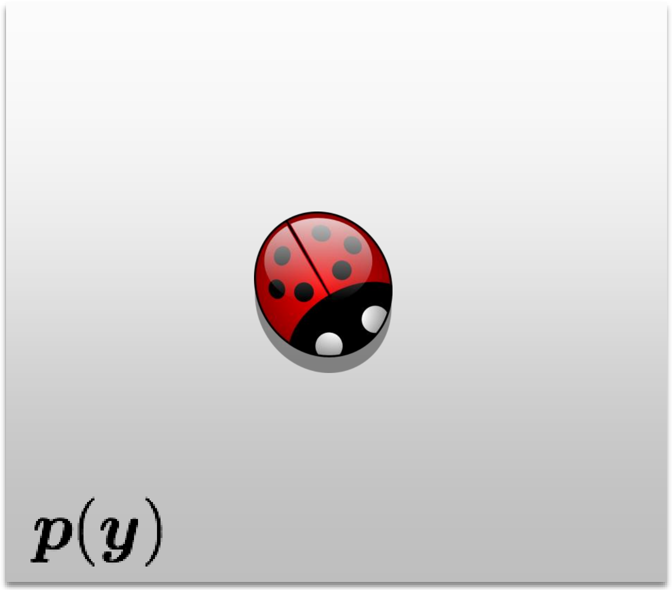
image

similarity over image

$\boldsymbol{q}$

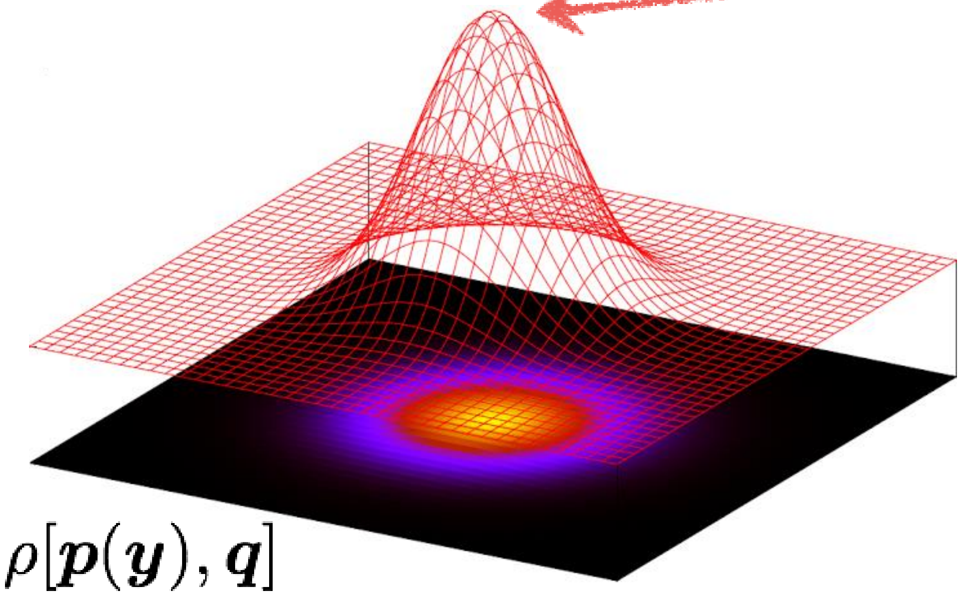
target

we want to find this peak



$\boldsymbol{p}(\boldsymbol{y})$

image



$\rho[\boldsymbol{p}(\boldsymbol{y}), \boldsymbol{q}]$

similarity over image

Objective function

$$\min_{\boldsymbol{y}} d(\boldsymbol{y})$$ same as $$\max_{\boldsymbol{y}} \rho[\boldsymbol{p}(\boldsymbol{y}), \boldsymbol{q}]$$

Assuming a good initial guess

$$\rho[\boldsymbol{p}(\boldsymbol{y}_0 + \boldsymbol{y}), \boldsymbol{q}]$$

Linearize around the initial guess (Taylor series expansion)

$$\rho[\boldsymbol{p}(\boldsymbol{y}), \boldsymbol{q}] \approx \frac{1}{2} \sum_m \sqrt{p_m(\boldsymbol{y}_0) q_m} + \frac{1}{2} \sum_m p_m(\boldsymbol{y}) \sqrt{\frac{q_m}{p_m(\boldsymbol{y}_0)}}$$

function at specified value

derivative

Objective function

Linearized objective

$$\rho[\boldsymbol{p}(\boldsymbol{y}), \boldsymbol{q}] \approx \frac{1}{2} \sum_m \sqrt{p_m(\boldsymbol{y}_0) q_m} + \frac{1}{2} \sum_m p_m(\boldsymbol{y}) \sqrt{\frac{q_m}{p_m(\boldsymbol{y}_0)}}$$

$$p_m = C_h \sum_n k\left(\left\|\frac{\boldsymbol{y} - \boldsymbol{x}_n}{h}\right\|^2\right) \delta[b(\boldsymbol{x}_n) - m]$$

Remember definition of this?

Fully expanded

$$\rho[\boldsymbol{p}(\boldsymbol{y}), \boldsymbol{q}] \approx \frac{1}{2} \sum_m \sqrt{p_m(\boldsymbol{y}_0) q_m} + \frac{1}{2} \sum_m \left\{ C_h \sum_n k\left(\left\|\frac{\boldsymbol{y} - \boldsymbol{x}_n}{h}\right\|^2\right) \delta[b(\boldsymbol{x}_n) - m] \right\} \sqrt{\frac{q_m}{p_m(\boldsymbol{y}_0)}}$$

Objective function

Fully expanded linearized objective

$$\rho[\boldsymbol{p}(\boldsymbol{y}), \boldsymbol{q}] \approx \frac{1}{2}\sum_m \sqrt{p_m(\boldsymbol{y}_0)q_m} + \frac{1}{2}\sum_m \left\{ C_h \sum_n k\left(\left\|\frac{\boldsymbol{y}-\boldsymbol{x}_n}{h}\right\|^2\right)\delta[b(\boldsymbol{x}_n)-m]\right\}\sqrt{\frac{q_m}{p_m(\boldsymbol{y}_0)}}$$

Moving terms around…

$$\rho[\boldsymbol{p}(\boldsymbol{y}), \boldsymbol{q}] \approx \boxed{\frac{1}{2}\sum_m \sqrt{p_m(\boldsymbol{y}_0)q_m}} + \boxed{\frac{C_h}{2}\sum_n w_n k\left(\left\|\frac{\boldsymbol{y}-\boldsymbol{x}_n}{h}\right\|^2\right)}$$

Does not depend on unknown **y**

Weighted kernel density estimate

where $$w_n = \sum_m \sqrt{\frac{q_m}{p_m(\boldsymbol{y}_0)}}\delta[b(\boldsymbol{x}_n)-m]$$

Weight is bigger when $q_m > p_m(\boldsymbol{y}_0)$

OK, why are we doing all this math?

We want to maximize this

$$\max_{\boldsymbol{y}} \rho[\boldsymbol{p}(\boldsymbol{y}), \boldsymbol{q}]$$

Fully expanded linearized objective

$$\rho[\boldsymbol{p}(\boldsymbol{y}), \boldsymbol{q}] \approx \frac{1}{2} \sum_m \sqrt{p_m(\boldsymbol{y}_0) q_m} + \frac{C_h}{2} \sum_n w_n k \left(\left\| \frac{\boldsymbol{y} - \boldsymbol{x}_n}{h} \right\|^2 \right)$$

where $\quad w_n = \sum_m \sqrt{\frac{q_m}{p_m(\boldsymbol{y}_0)}} \delta[b(\boldsymbol{x}_n) - m]$

We want to maximize this

$$\max_{\boldsymbol{y}} \rho[\boldsymbol{p}(\boldsymbol{y}), \boldsymbol{q}]$$

only need to maximize this!

Fully expanded linearized objective

$$\rho[\boldsymbol{p}(\boldsymbol{y}), \boldsymbol{q}] \approx \cancel{\frac{1}{2} \sum_m \sqrt{p_m(\boldsymbol{y}_0) q_m}} + \frac{C_h}{2} \sum_n w_n k \left(\left\| \frac{\boldsymbol{y} - \boldsymbol{x}_n}{h} \right\|^2 \right)$$

doesn't depend on unknown **y**

where $$w_n = \sum_m \sqrt{\frac{q_m}{p_m(\boldsymbol{y}_0)}} \delta[b(\boldsymbol{x}_n) - m]$$

We want to maximize this

$$\max_{\boldsymbol{y}} \rho[\boldsymbol{p}(\boldsymbol{y}), \boldsymbol{q}]$$

only need to maximize this!

Fully expanded linearized objective

$$\rho[\boldsymbol{p}(\boldsymbol{y}), \boldsymbol{q}] \approx \cancel{\frac{1}{2}\sum_m \sqrt{p_m(\boldsymbol{y}_0) q_m}} + \frac{C_h}{2}\sum_n w_n k\left(\left\|\frac{\boldsymbol{y}-\boldsymbol{x}_n}{h}\right\|^2\right)$$

doesn't depend on unknown **y**

where $$w_n = \sum_m \sqrt{\frac{q_m}{p_m(\boldsymbol{y}_0)}}\delta[b(\boldsymbol{x}_n) - m]$$

what can we use to solve this weighted KDE?

Mean Shift Algorithm!

$$\frac{C_h}{2}\sum_n w_n k\left(\left\|\frac{\boldsymbol{y}-\boldsymbol{x}_n}{h}\right\|^2\right)$$

the new sample of mean of this KDE is

$$\boldsymbol{y}_1 = \frac{\sum_n \boldsymbol{x}_n w_n g\left(\left\|\frac{\boldsymbol{y}_0-\boldsymbol{x}_n}{h}\right\|^2\right)}{\sum_n w_n g\left(\left\|\frac{\boldsymbol{y}_0-\boldsymbol{x}_n}{h}\right\|^2\right)}$$

(new candidate location)

(this was derived earlier)

Mean-Shift Object Tracking

For each frame:

1. **Initialize location** $\boldsymbol{y}_0$
   Compute $\boldsymbol{q}$
   Compute $\boldsymbol{p}(\boldsymbol{y}_0)$

2. **Derive weights** w_n

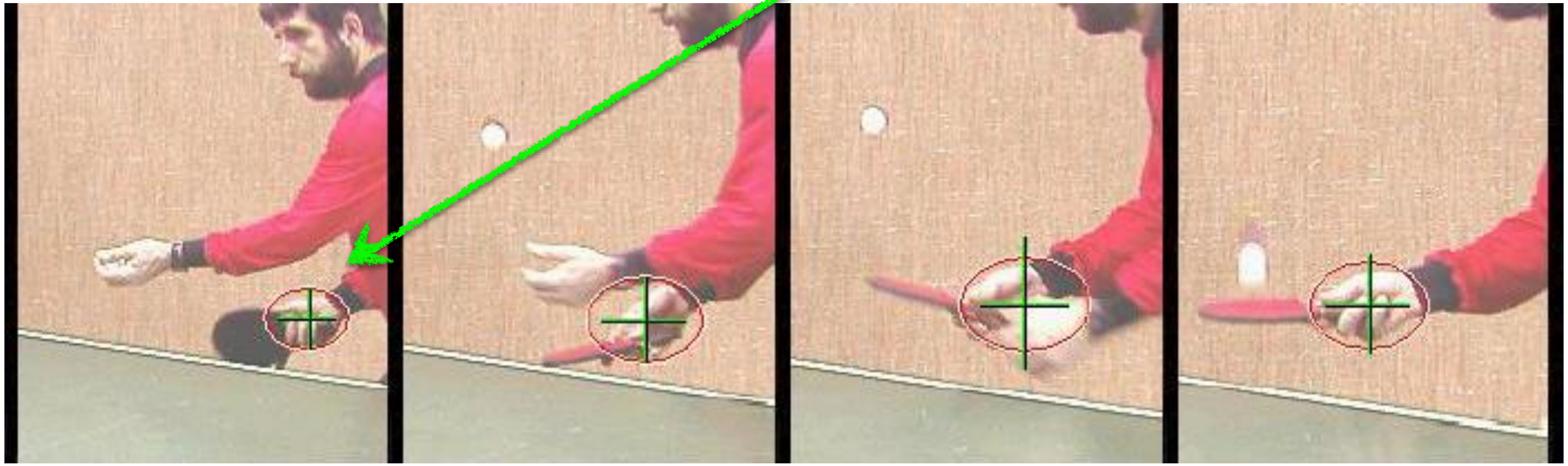
3. **Shift to new candidate location (mean shift)** $\boldsymbol{y}_1$

4. **Compute** $\boldsymbol{p}(\boldsymbol{y}_1)$

5. **If** $\|\boldsymbol{y}_0 - \boldsymbol{y}_1\| < \epsilon$ **return**
   Otherwise $\boldsymbol{y}_0 \leftarrow \boldsymbol{y}_1$ **and go back to 2**

Mean-Shift Object Tracking: example

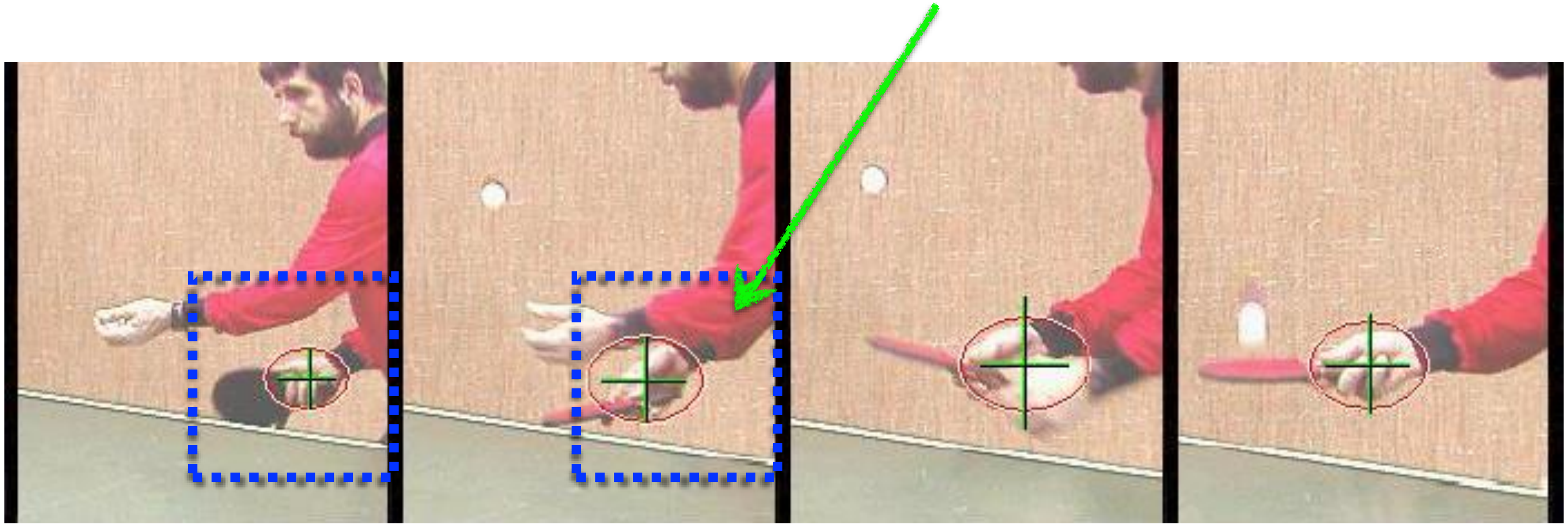
Compute a descriptor for the target



Target
$\boldsymbol{q}$

Mean-Shift Object Tracking: example

Search for similar descriptor in neighborhood in next frame

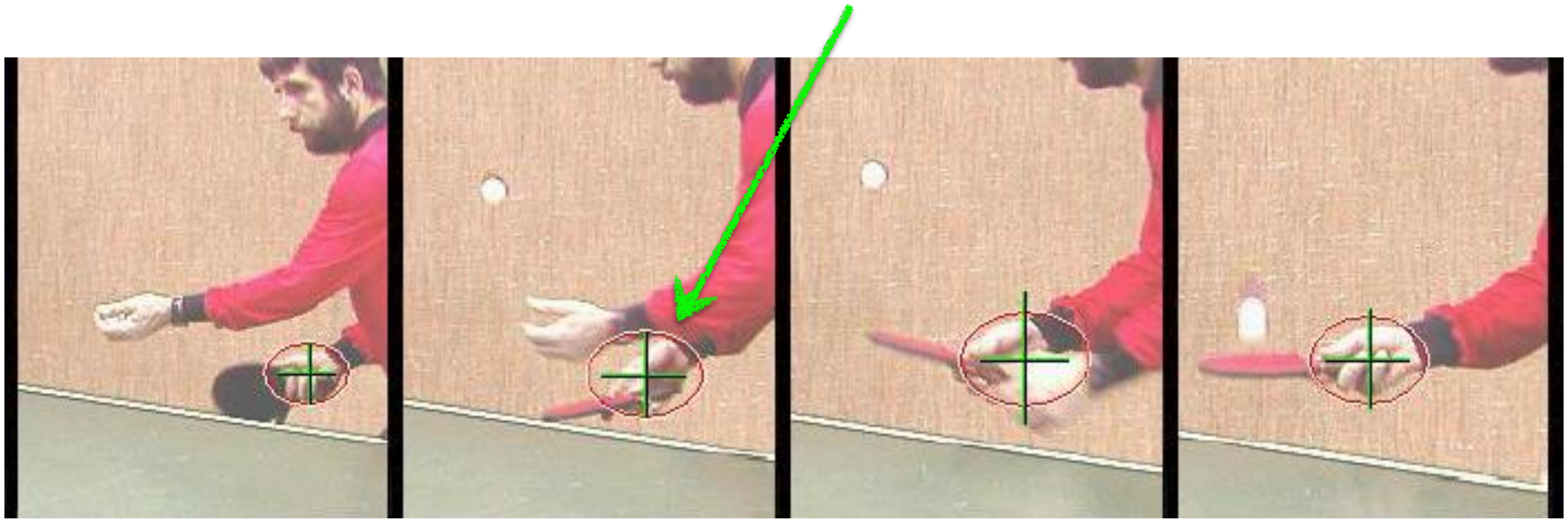


Target   Candidate

$$\max_{\boldsymbol{y}} \rho[\boldsymbol{p}(\boldsymbol{y}), \boldsymbol{q}]$$

Mean-Shift Object Tracking: example

Compute a descriptor for the new target

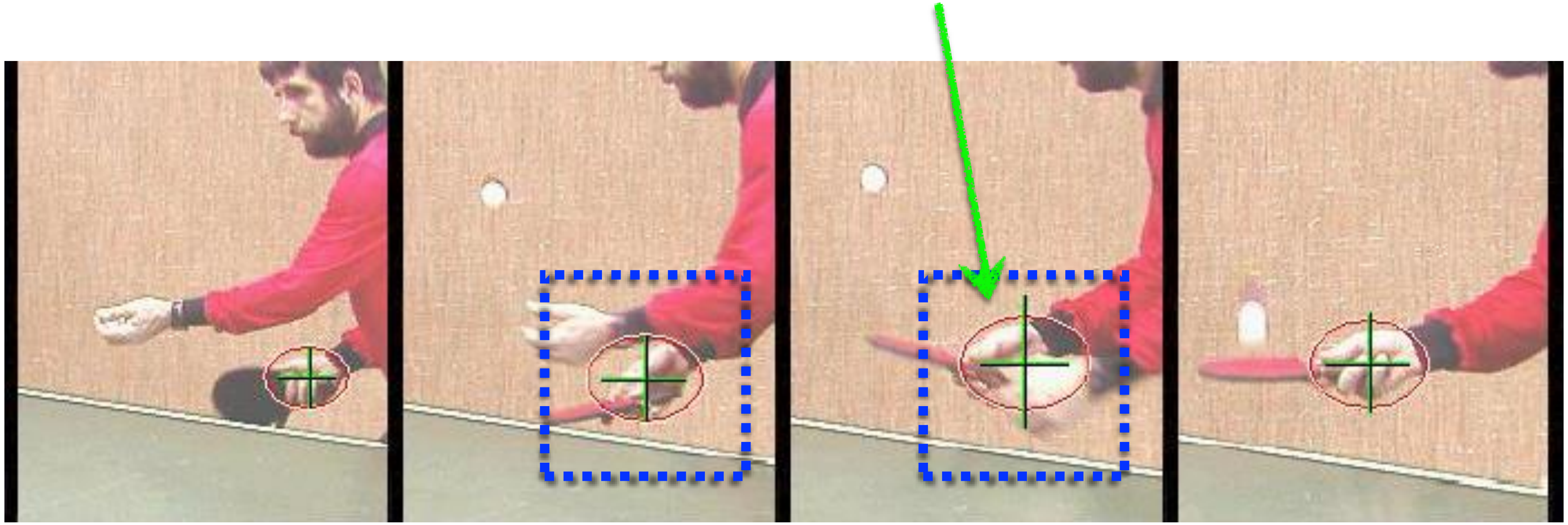


Target

$\boldsymbol{q}$

Mean-Shift Object Tracking: example

Search for similar descriptor in neighborhood in next frame



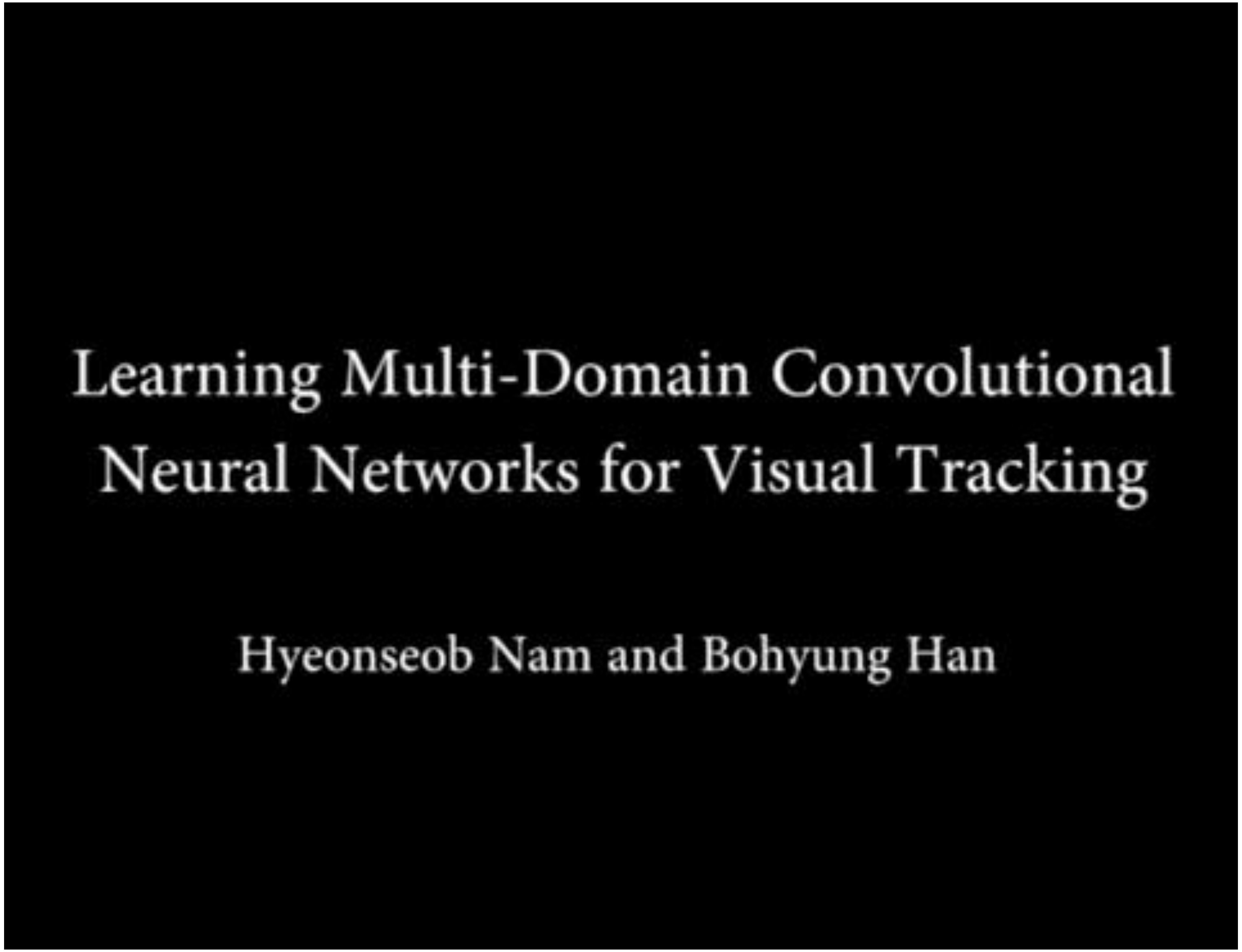
Target

Candidate

$$\max_{\boldsymbol{y}} \rho[\boldsymbol{p}(\boldsymbol{y}), \boldsymbol{q}]$$

Examples

Modern trackers



From Mid-level to High-level ?

https://youtu.be/9zoJP2FkpgU?t=7