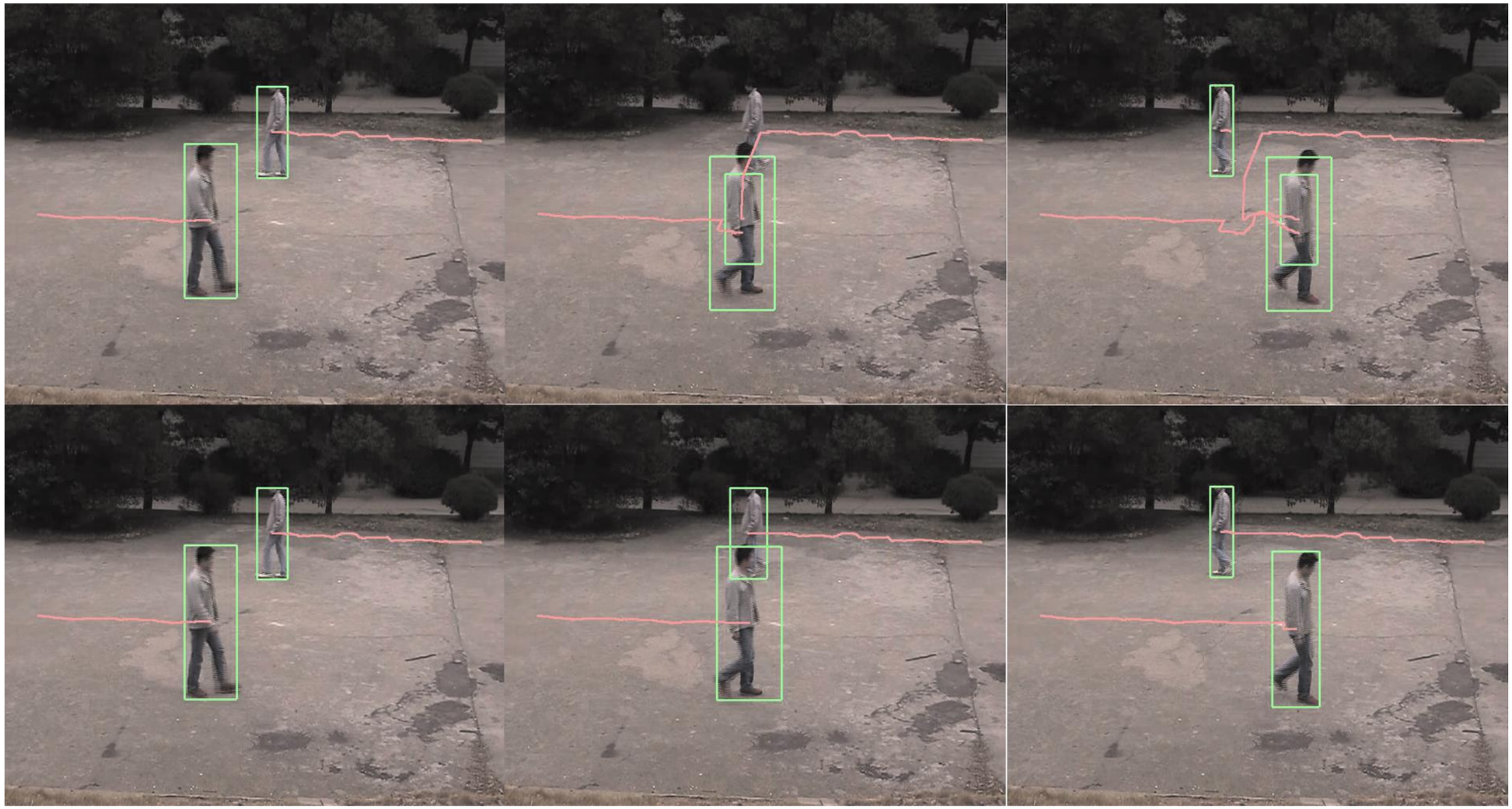
# Alignment and tracking



http://www.cs.cmu.edu/~16385/

16-385 Computer Vision
Spring 2020, Lecture 25

# Course announcements

- Homework 6 has been posted and is due on April 22nd.
  - Any questions about the homework?
  - How many of you have looked at/started/finished homework 6?

- Take-home quiz 10 has been posted and is due on April 19th.

# Overview of today's lecture

- Finish optical flow lecture.
- Motion magnification using optical flow.
- Image alignment.
- Lucas-Kanade alignment.
- Baker-Matthews alignment.
- Inverse alignment.
- KLT tracking.
- Mean-shift tracking.
- Modern trackers.

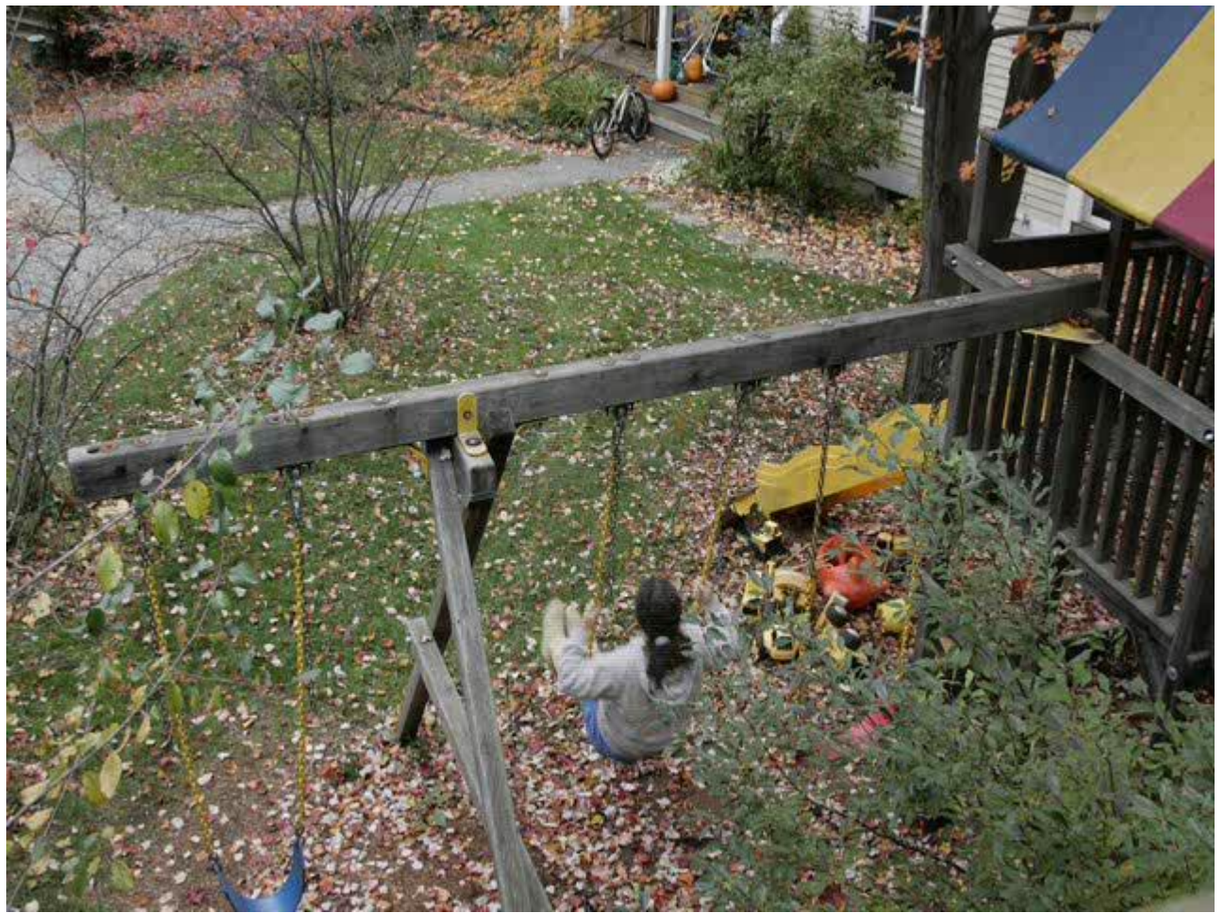
# Slide credits

Most of these slides were adapted from:

- Kris Kitani (16-385, Spring 2017).

# Motion magnification using optical flow

# How would you achieve this effect?



original

motion-magnified

- Compute optical flow from frame to frame.
- Magnify optical flow velocities.
- Appropriately warp image intensities.

# How would you achieve this effect?

naïvely motion-magnified

motion-magnified

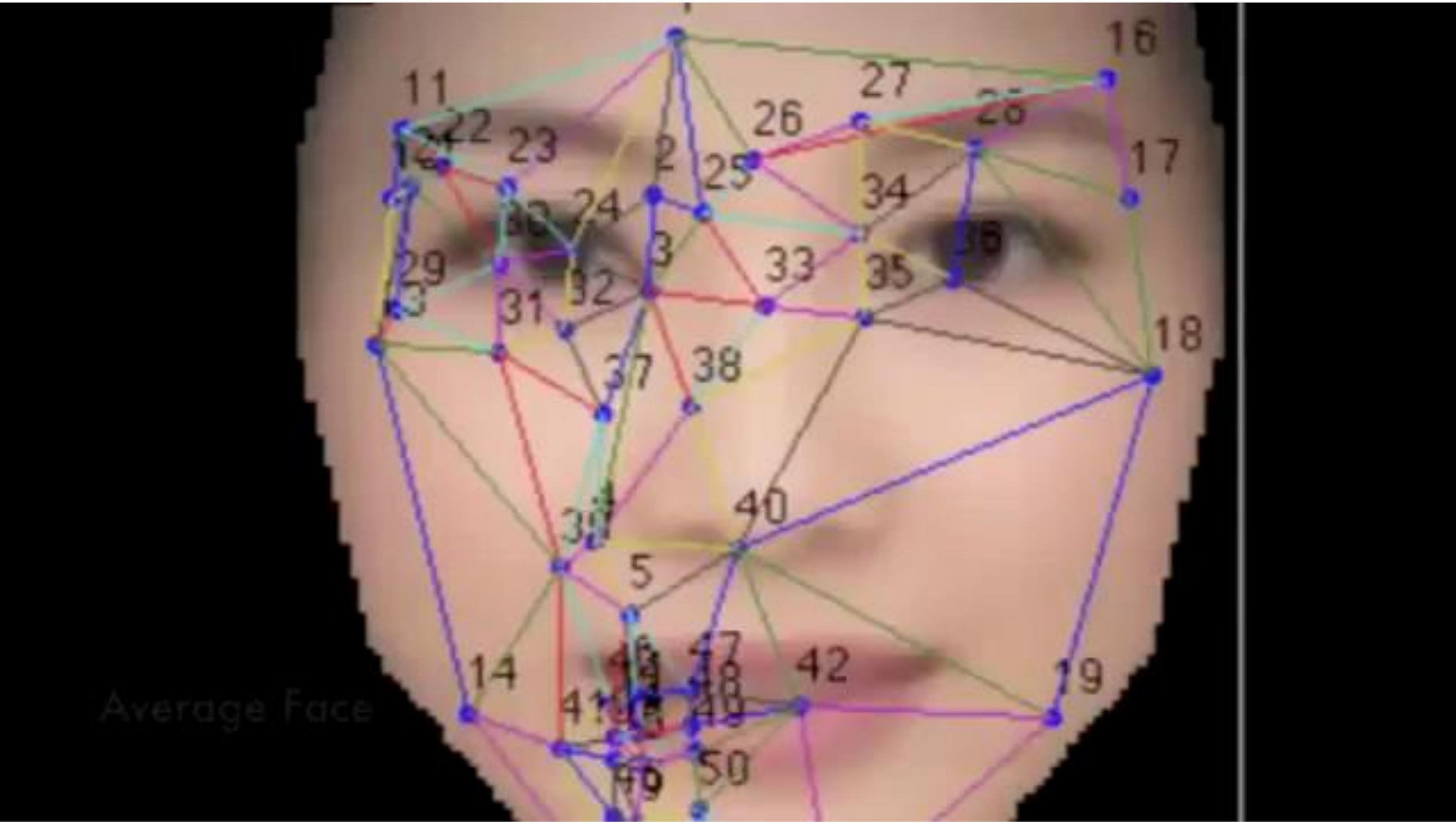
- Compute optical flow from frame to frame.
- Magnify optical flow velocities.
- Appropriately warp image intensities.

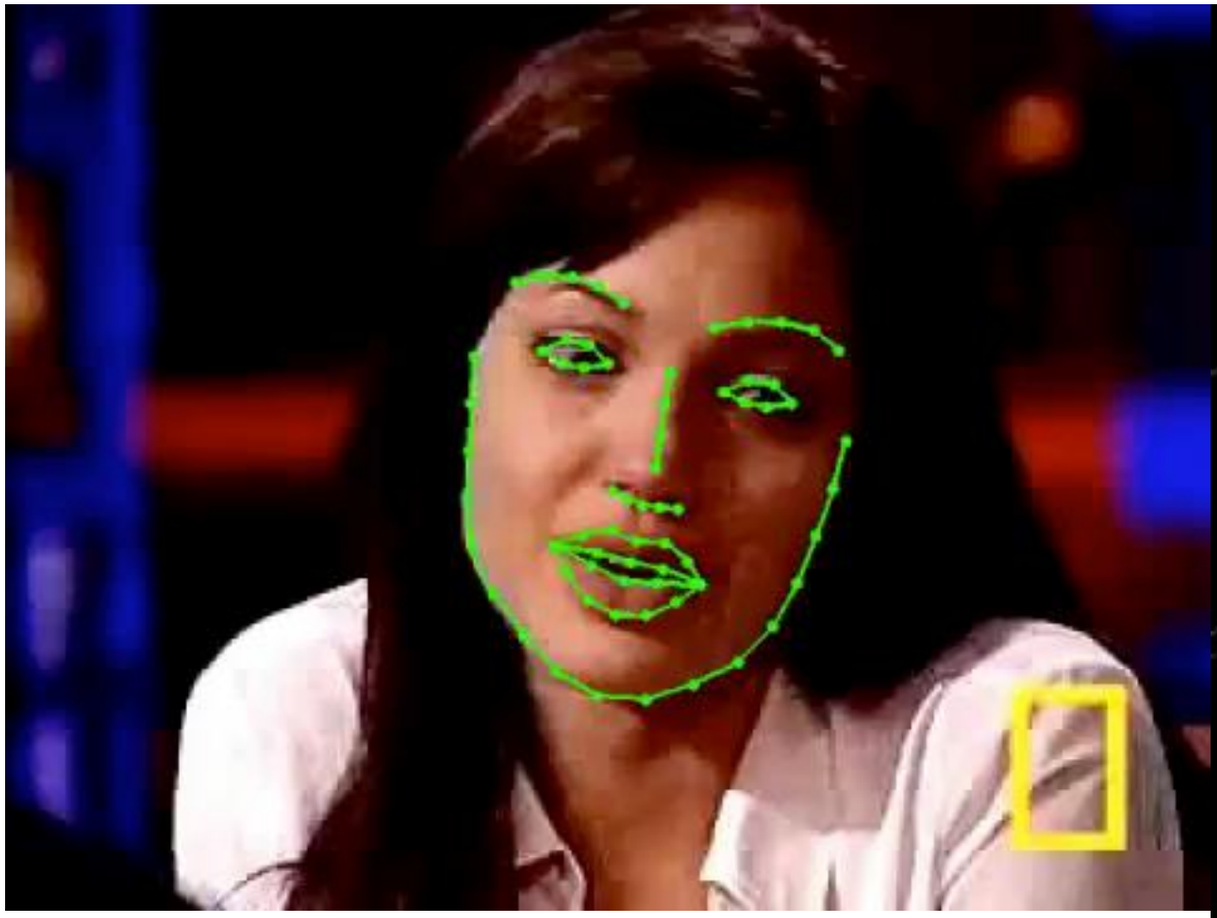
In practice, many additional steps are required for a good result.

# Some more examples

# Some more examples

# Image alignment


11
22
23
26
27
16
17
24
25
34
29
31
32
33
35
18
37
38
40
5
14
42
19
Average Face




# IntraFace

http://www.humansensing.cs.cmu.edu/intraface/


31 fps

How can I find  in the image?

# Idea #1: Template Matching

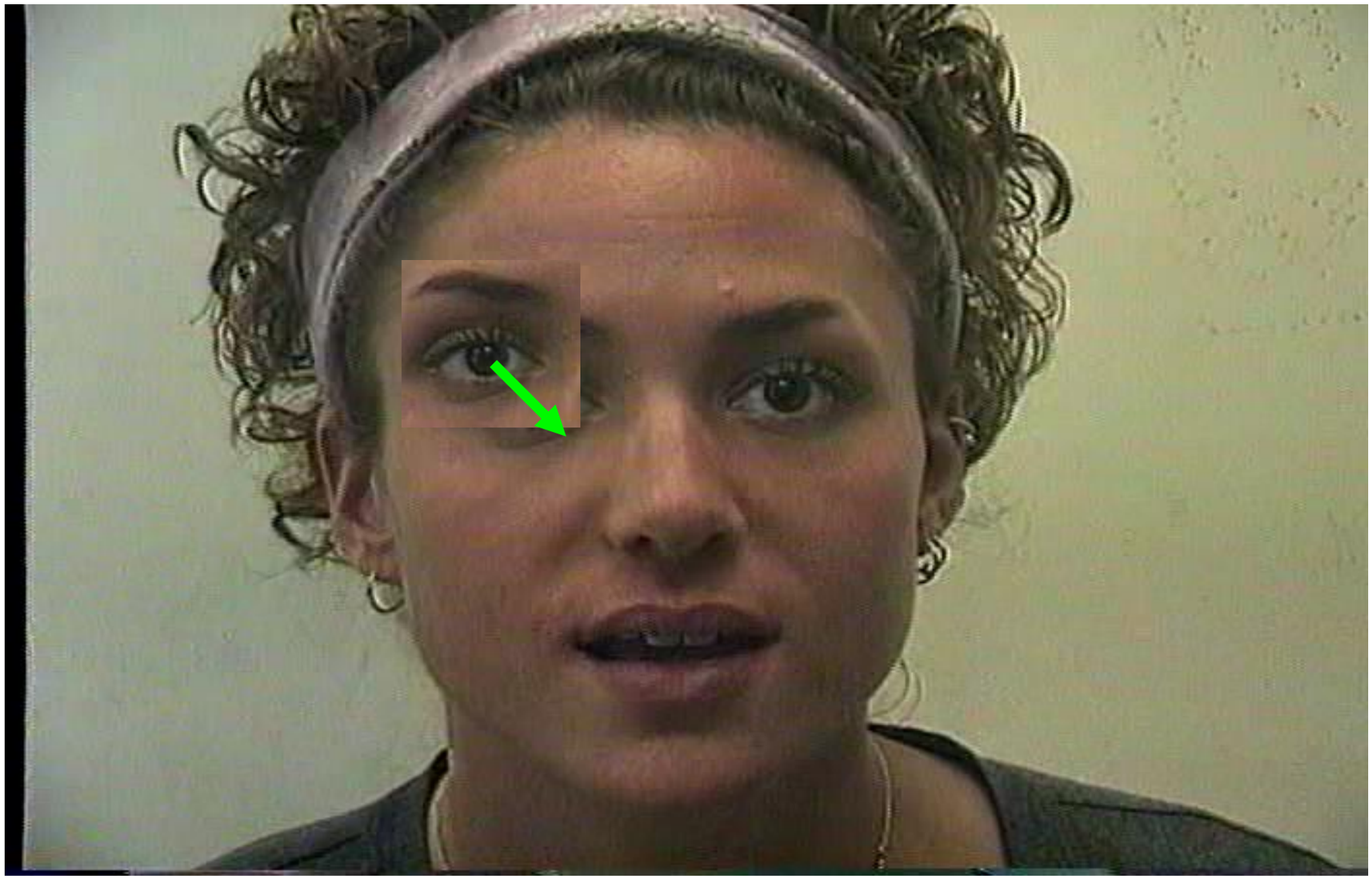
Slow, combinatory, global solution

# Idea #2: Pyramid Template Matching

Faster, combinatory, 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

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

Pixel value at a coordinate

**Translation**

**Affine**

# 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

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

Pixel value at a coordinate

**Translation**

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

$$= \begin{bmatrix} 1 & 0 & p_1 \\ 0 & 1 & p_2 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

transform coordinate

**Affine**

# 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

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

Pixel value at a coordinate

**Translation**

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

$$= \begin{bmatrix} 1 & 0 & p_1 \\ 0 & 1 & p_2 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

transform coordinate

**Affine**

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

$$= \begin{bmatrix} p_1 & p_2 & p_3 \\ p_4 & p_5 & p_6 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

affine transform coordinate

can be written in matrix form when linear

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

$\mathbf{W}(\boldsymbol{x};\boldsymbol{p})$ takes a _____ as input and returns a _____

$\mathbf{W}(\boldsymbol{x};\boldsymbol{p})$ is a function of ____ variables

$\mathbf{W}(\boldsymbol{x};\boldsymbol{p})$ returns a _____ of dimension ___ x ___

$\boldsymbol{p} = \{p_1, \ldots, p_N\}$ where N is _____ for an affine model

$I(\boldsymbol{x}') = I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p}))$ this warp changes pixel values?

# Image alignment

(problem definition)

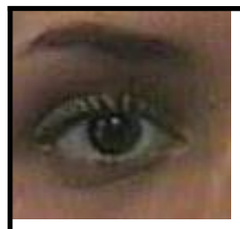
$$\min_{\boldsymbol{p}} \sum_{\boldsymbol{x}} \left[ I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p})) - T(\boldsymbol{x}) \right]^2$$

warped image template image

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

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

$T(\boldsymbol{x})$



$I(\boldsymbol{x})$

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

# 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?*

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

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

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

Hard to optimize

*What can you do to make it easier to solve?*

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

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

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

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 $\boldsymbol{p}$...

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

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)

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 $\boldsymbol{I}$ for a really small perturbation of **p**?*

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 $\boldsymbol{I}$ for a really small perturbation of **p**?*

Taylor series approximation!

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

Multivariable Taylor Series Expansion
(First order approximation)

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

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

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

Multivariable Taylor Series Expansion
(First order approximation)

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

Linear approximation

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

*What are the unknowns here?*

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

Multivariable Taylor Series Expansion
(First order approximation)

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

## 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

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

$\boldsymbol{x}$ is a _____ of dimension ___ x ___

output of $\mathbf{W}$ is a _____ of dimension ___ x ___

$\boldsymbol{p}$ is a _____ of dimension ___ x ___

$I(\cdot)$ is a function of _____ variables

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

(A matrix of partial derivatives)

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

$$\mathbf{W} = \begin{bmatrix} W_x(x,y) \\ W_y(x,y) \end{bmatrix}$$

$$\frac{\partial \mathbf{W}}{\partial \boldsymbol{p}} = \begin{bmatrix} \frac{\partial W_x}{\partial p_1} & \frac{\partial W_x}{\partial p_2} & \cdots & \frac{\partial W_x}{\partial p_N} \\ \\ \frac{\partial W_y}{\partial p_1} & \frac{\partial W_y}{\partial p_2} & \cdots & \frac{\partial W_y}{\partial p_N} \end{bmatrix}$$

**Rate of change of the warp**

Affine transform

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

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

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

$$\frac{\partial \mathbf{W}}{\partial \boldsymbol{p}} = \begin{bmatrix} x & 0 & y & 0 & 1 & 0 \\ 0 & x & 0 & y & 0 & 1 \end{bmatrix}$$

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

$\nabla I$ is a _____ of dimension ___ x ___

$\frac{\partial \mathbf{W}}{\partial \boldsymbol{p}}$ is a _____ of dimension ___ x ___

$\Delta \boldsymbol{p}$ is a _____ of dimension ___ x ___

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

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

pixel coordinate
(2 x 1)

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

pixel coordinate
(2 x 1)

image intensity
(scalar)

warp function
(2 x 1)

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

pixel coordinate
(2 x 1)

image intensity
(scalar)

warp function
(2 x 1)

warp parameters
(6 for affine)

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

pixel coordinate
(2 x 1)

image intensity
(scalar)

warp function
(2 x 1)

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)

pixel coordinate
(2 x 1)

image intensity
(scalar)

warp function
(2 x 1)

warp parameters
(6 for affine)

Partial derivatives of warp function
(2 x 6)

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

pixel coordinate
(2 x 1)

image intensity
(scalar)

image gradient
(1 x 2)

warp function
(2 x 1)

warp parameters
(6 for affine)

Partial derivatives of warp function
(2 x 6)

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

pixel coordinate
(2 x 1)

image intensity
(scalar)

image gradient
(1 x 2)

incremental warp
(6 x 1)

warp function
(2 x 1)

warp parameters
(6 for affine)

Partial derivatives of warp function
(2 x 6)

template image intensity
(scalar)

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

pixel coordinate
(2 x 1)

image intensity
(scalar)

image gradient
(1 x 2)

incremental warp
(6 x 1)

When you implement this, you will compute everything in parallel and store as matrix … don't loop over x!

# Summary

(of Lucas-Kanade Image Alignment)

Problem:

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

warped image

template image

Difficult non-linear optimization problem

Strategy:

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

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 $\Delta\boldsymbol{p}$

OK, so how do we solve this?

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

Another way to look at it…

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

(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?*

Another way to look at it…

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

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

constant variable constant

Looks like $\mathbf{A}\mathbf{x} - \mathbf{b}$

*How do you solve this?*

Least squares approximation

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

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

$A^\top A$

## 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:

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

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

## Solution:

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

Solution to least squares 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

This is called…

# Gauss-Newton gradient decent non-linear optimization!

# Lucas Kanade (Additive alignment)

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 $\nabla I(\boldsymbol{x}')$

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

**Just 8 lines of code!**

# Baker-Matthews alignment

# Image Alignment

(start with an initial solution, match the image and template)

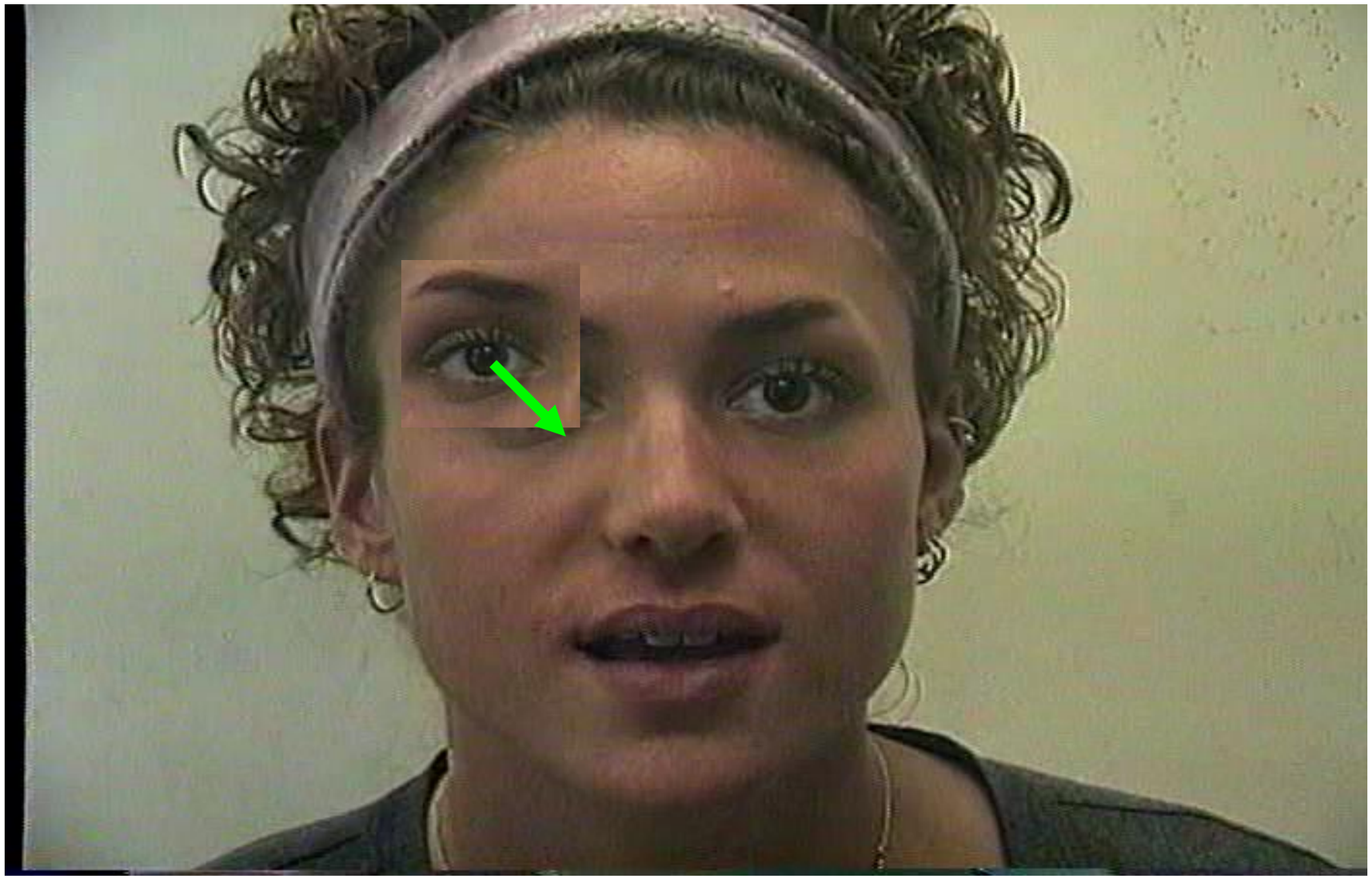


Image Alignment Objective Function

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

Given an initial solution…several possible formulations

## Additive Alignment

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

incremental perturbation of parameters

## Image Alignment Objective Function

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

Given an initial solution…several possible formulations

### Additive Alignment

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

incremental perturbation of parameters

### Compositional Alignment

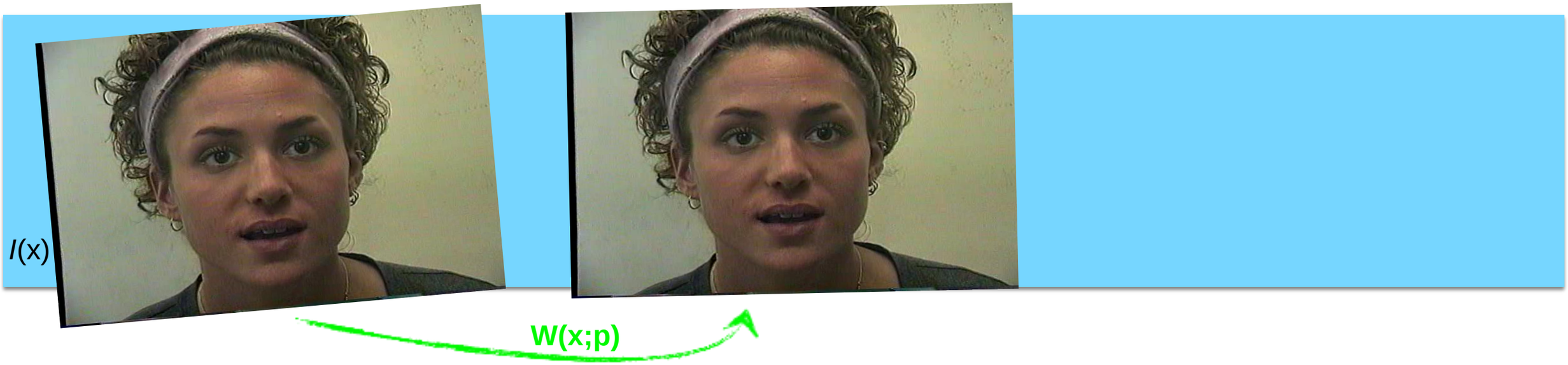
$$\sum_{\boldsymbol{x}} \left[ I(\mathbf{W}(\ \mathbf{W}(\boldsymbol{x};\Delta\boldsymbol{p});\boldsymbol{p}\ ) - T(\boldsymbol{x}) \right]^2$$

incremental warps of image

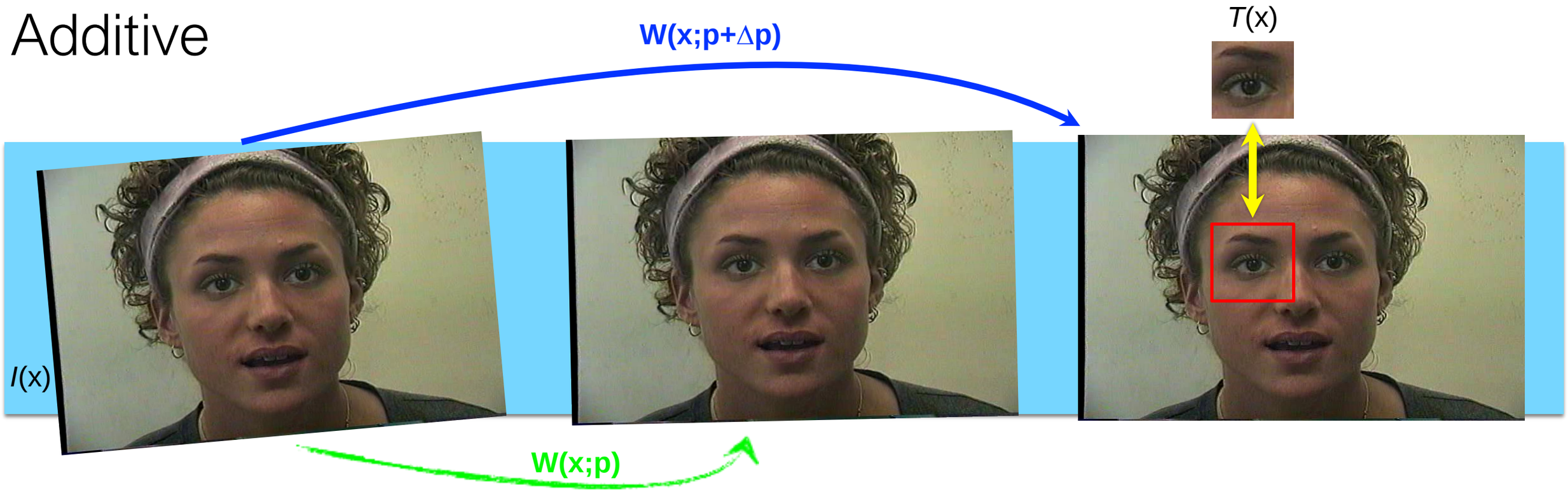
# Additive strategy

# Compositional strategy

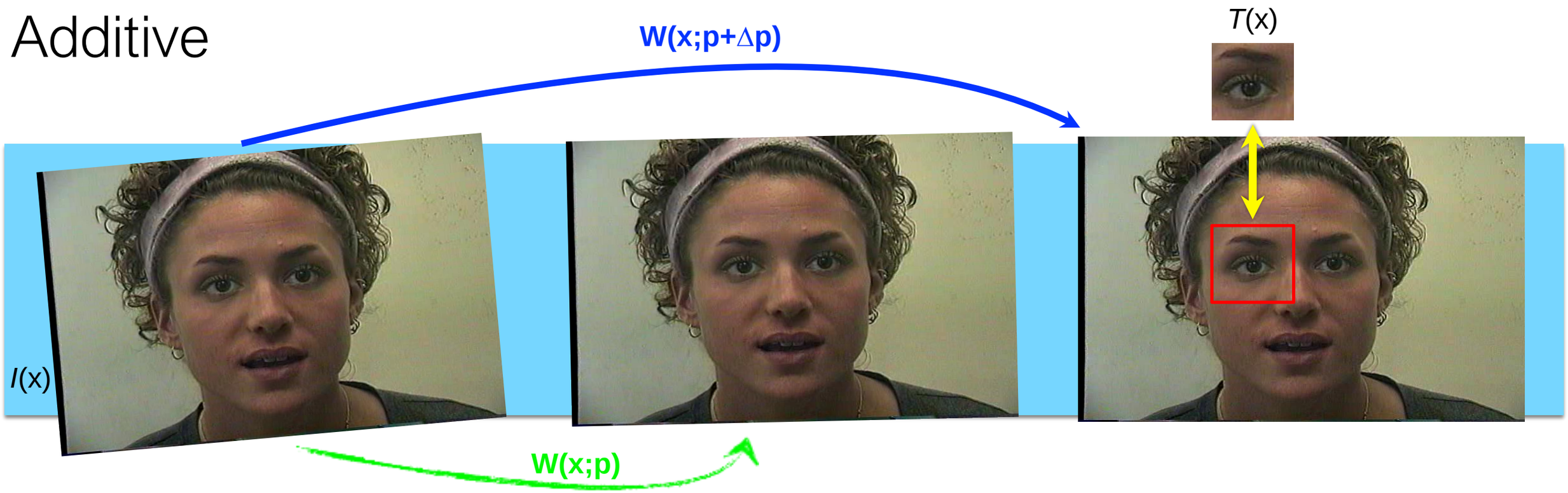
# Additive



# Additive

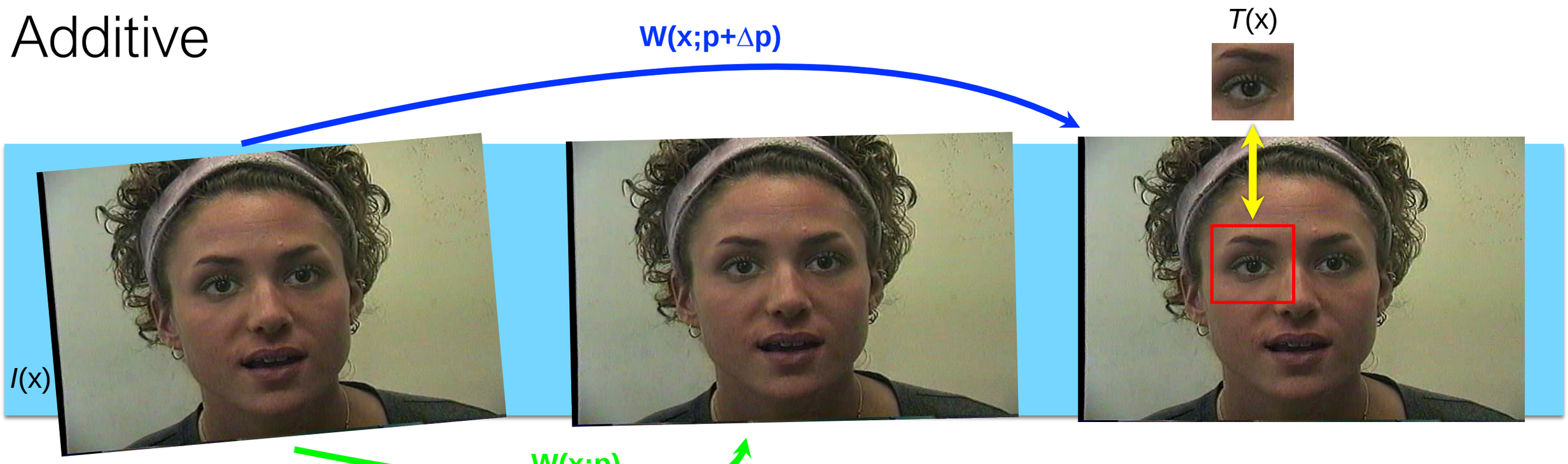


# Additive

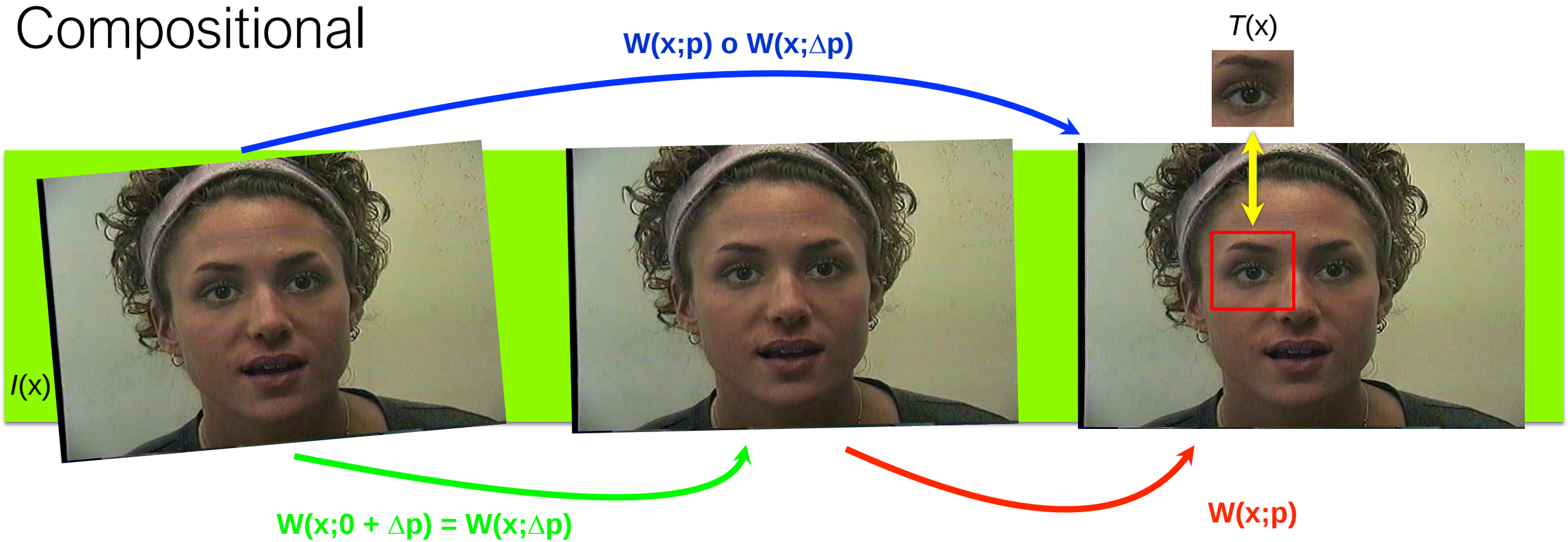



# Compositional

# Additive




# Compositional



# Compositional Alignment

Original objective function (SSD)

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

Assuming an initial solution **p** and a compositional warp increment

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

# Compositional Alignment

Original objective function (SSD)

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

Assuming an initial solution **p** and a compositional warp increment

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

Another way to write the composition

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

Identity warp

$$\mathbf{W}(\boldsymbol{x};\mathbf{0})$$

# Compositional Alignment

Original objective function (SSD)

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

Assuming an initial solution **p** and a compositional warp increment

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

Another way to write the composition

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

Identity warp

$$\mathbf{W}(\boldsymbol{x};\mathbf{0})$$

Skipping over the derivation…the new update rule is

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

So what’s so great about this compositional form?

## Additive Alignment

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

linearized form

$$\sum_{\boldsymbol{x}} \left[ I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p})) + \nabla I(\boldsymbol{x}') \frac{\partial \mathbf{W}}{\partial \boldsymbol{p}} \Delta\boldsymbol{p} - T(\boldsymbol{x}) \right]^2$$

## Compositional Alignment

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

linearized form

$$\sum_{\boldsymbol{x}} \left[ I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p})) + \nabla I(\boldsymbol{x}') \frac{\partial \mathbf{W}(\boldsymbol{x};\mathbf{0})}{\partial \boldsymbol{p}} \Delta\boldsymbol{p} - T(\boldsymbol{x}) \right]^2$$

## Additive Alignment

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

linearized form

$$\sum_{\boldsymbol{x}} \left[ I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p})) + \nabla I(\boldsymbol{x}') \frac{\partial \mathbf{W}}{\partial \boldsymbol{p}} \Delta\boldsymbol{p} - T(\boldsymbol{x}) \right]^2$$

Jacobian of W(x;p)

## Compositional Alignment

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

linearized form

$$\sum_{\boldsymbol{x}} \left[ I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p})) + \nabla I(\boldsymbol{x}') \frac{\partial \mathbf{W}(\boldsymbol{x};\mathbf{0})}{\partial \boldsymbol{p}} \Delta\boldsymbol{p} - T(\boldsymbol{x}) \right]^2$$

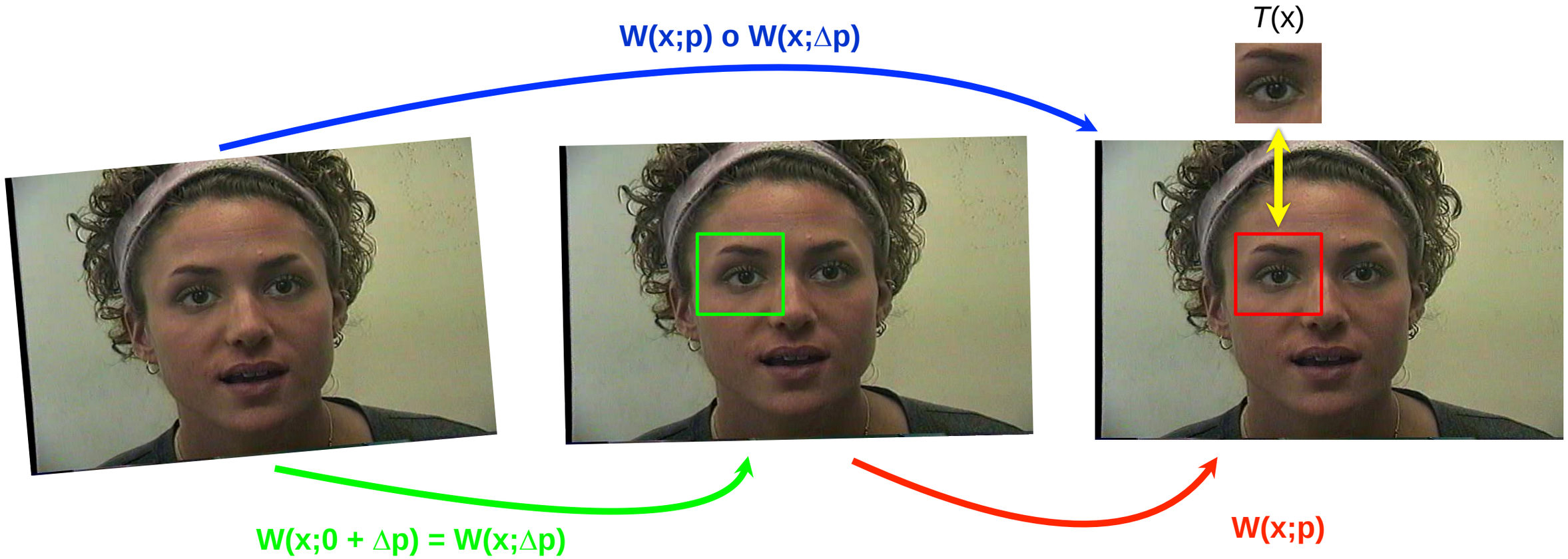
Jacobian of W(x;0)

**The Jacobian is constant.**
**Jacobian can be precomputed!**

# Compositional Image Alignment

Minimize

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




Jacobian is simple and can be precomputed

# Lucas Kanade (Additive alignment)

1. Warp image $I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p}))$
2. Compute error image $[T(\boldsymbol{x}) - I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p}))]^2$
3. Compute gradient $\nabla I(\boldsymbol{x}')$
4. Evaluate Jacobian $\frac{\partial \mathbf{W}}{\partial \boldsymbol{p}}$
5. Compute Hessian $H$
6. Compute $\Delta \boldsymbol{p}$
7. Update parameters $\boldsymbol{p} \leftarrow \boldsymbol{p} + \Delta \boldsymbol{p}$

# Shum-Szeliski (Compositional alignment)

1. Warp image $I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p}))$
2. Compute error image $[T(\boldsymbol{x}) - I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p}))]^2$
3. Compute gradient $\nabla I(\boldsymbol{x}')$
4. Evaluate Jacobian $\frac{\partial \mathbf{W}(\boldsymbol{x};\mathbf{0})}{\partial \boldsymbol{p}}$
5. Compute Hessian $H$
6. Compute $\Delta \boldsymbol{p}$
7. Update parameters $\mathbf{W}(\boldsymbol{x};\boldsymbol{p}) \leftarrow \mathbf{W}(\boldsymbol{x};\boldsymbol{p}) \circ \mathbf{W}(\boldsymbol{x};\Delta \boldsymbol{p})$

Any other speed up techniques?

# Inverse alignment

# Why not compute warp updates on the template?

Additive Alignment

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

Compositional Alignment

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

Why not compute warp updates on the template?

Additive Alignment

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

Compositional Alignment

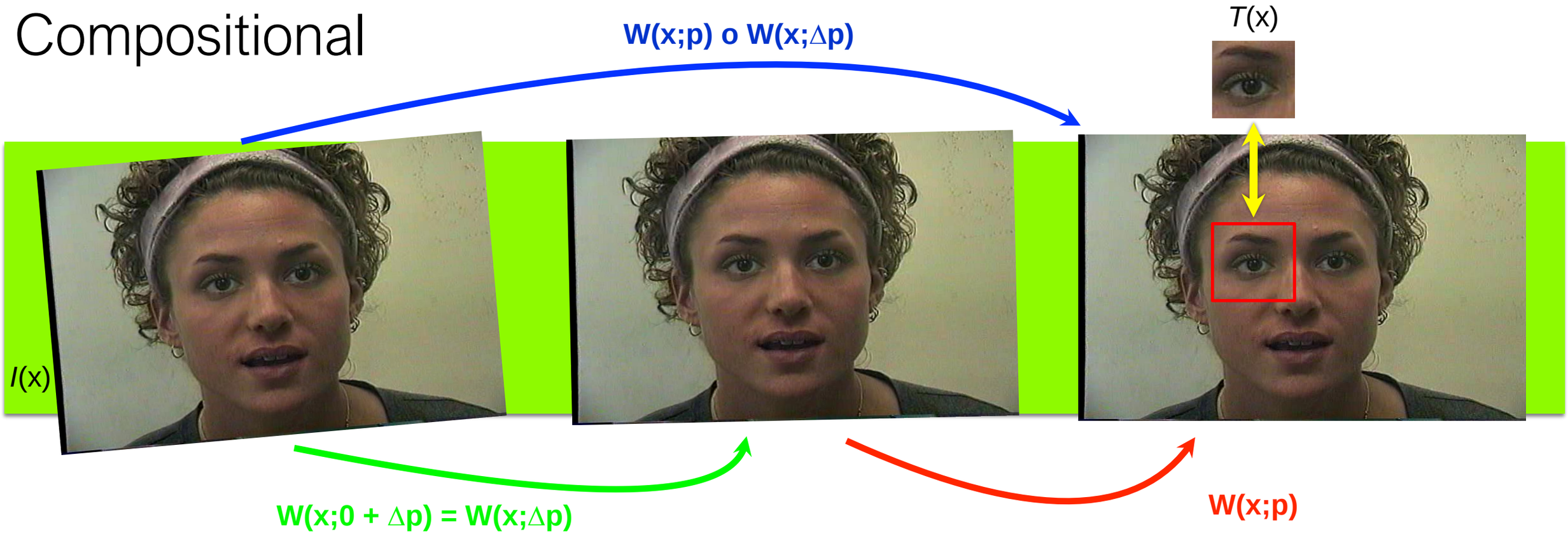
$$\sum_{\boldsymbol{x}} [I(\mathbf{W}(\ \mathbf{W}(\boldsymbol{x};\Delta\boldsymbol{p});\boldsymbol{p}\ ) - T(\boldsymbol{x})]^2$$

What happens if you let the template be warped too?

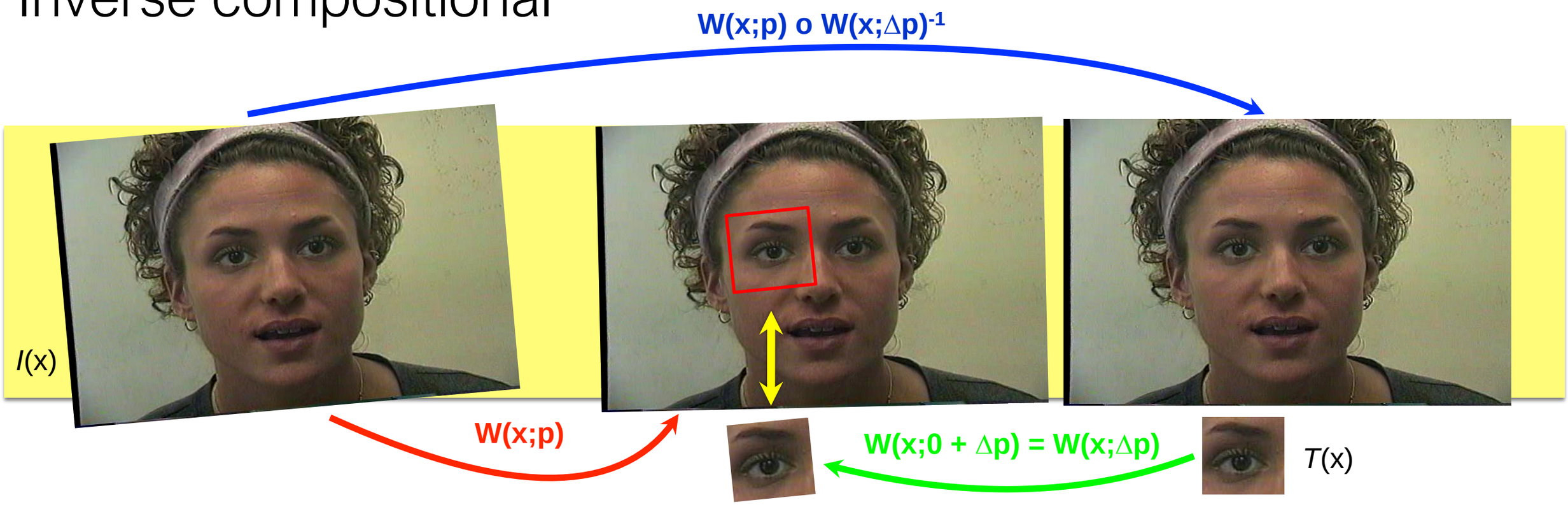
Inverse Compositional Alignment

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

# Compositional




# Inverse compositional



# Compositional strategy

# Inverse Compositional strategy

So what's so great about this inverse compositional form?

# Inverse Compositional Alignment

**Minimize**

$$\sum_{\boldsymbol{x}} [T(\mathbf{W}(\boldsymbol{x};\Delta\boldsymbol{p}) - I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p}))]^2 \approx \sum \boldsymbol{x} \left[ T(\mathbf{W}(\boldsymbol{x};\mathbf{0})) + \nabla T \frac{\partial \mathbf{W}}{\partial \boldsymbol{p}} \Delta \boldsymbol{p} - I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p}) \right]^2$$

**Solution**

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

can be precomputed from template!

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

**Update**

$$\mathbf{W}(\boldsymbol{x};\boldsymbol{p}) \leftarrow \mathbf{W}(\boldsymbol{x};\boldsymbol{p}) \circ \mathbf{W}(\boldsymbol{x};\Delta\boldsymbol{p})^{-1}$$

# Properties of inverse compositional alignment

**Jacobian** can be precomputed

It is constant - evaluated at W(x;0)

**Gradient of template** can be precomputed

It is constant

**Hessian** can be precomputed

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

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

(main term that needs to be computed)

**Warp** must be invertible

# Lucas Kanade (Additive alignment)

1. Warp image $I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p}))$
2. Compute error image $[T(\boldsymbol{x}) - I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p}))]^2$
3. Compute gradient $\nabla I(\mathbf{W})$
4. Evaluate Jacobian $\frac{\partial \mathbf{W}}{\partial \boldsymbol{p}}$
5. Compute Hessian $H$
6. Compute $\Delta \boldsymbol{p}$
7. Update parameters $\boldsymbol{p} \leftarrow \boldsymbol{p} + \Delta \boldsymbol{p}$

## Shum-Szeliski (Compositional alignment)

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 $\frac{\partial \mathbf{W}(\boldsymbol{x};\mathbf{0})}{\partial \boldsymbol{p}}$
5. Compute Hessian $H$
6. Compute $\Delta \boldsymbol{p}$
7. Update parameters $\mathbf{W}(\boldsymbol{x};\boldsymbol{p}) \leftarrow \mathbf{W}(\boldsymbol{x};\boldsymbol{p}) \circ \mathbf{W}(\boldsymbol{x};\Delta \boldsymbol{p})$

## Baker-Matthews (Inverse Compositional alignment)

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 $\nabla T(\mathbf{W})$

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

5. Compute Hessian $H$ $H = \sum_{\boldsymbol{x}} \left[\nabla T \frac{\partial \mathbf{W}}{\partial \boldsymbol{p}}\right]^{\top} \left[\nabla T \frac{\partial \mathbf{W}}{\partial \boldsymbol{p}}\right]$

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

7. Update parameters $\mathbf{W}(\boldsymbol{x};\boldsymbol{p}) \leftarrow \mathbf{W}(\boldsymbol{x};\boldsymbol{p}) \circ \mathbf{W}(\boldsymbol{x};\Delta \boldsymbol{p})^{-1}$

| Algorithm | Efficient | Authors |
|---|---|---|
| **Forwards Additive** | No | Lucas, Kanade |
| **Forwards compositional** | No | Shum, Szeliski |
| **Inverse Additive** | Yes | Hager, Belhumeur |
| **Inverse Compositional** | Yes | Baker, Matthews |

# Kanade-Lucas-Tomasi (KLT) tracker

https://www.youtube.com/watch?v=rwIjkECpY0M

# Feature-based tracking

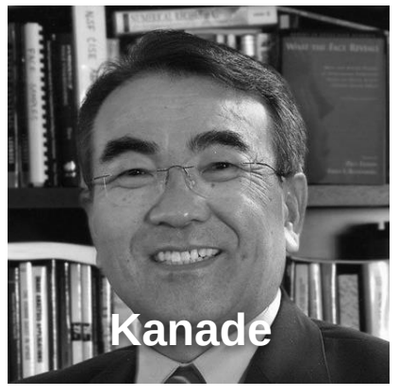
Up to now, we've been aligning entire images
but we can also track just small image regions too!
(sometimes called sparse tracking or sparse alignment)

How should we select the 'small images' (features)?

How should we track them from frame to frame?

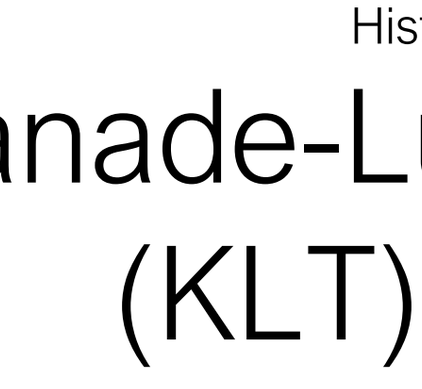
History of the

# Kanade-Lucas-Tomasi (KLT) Tracker


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**

# 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?*

## *What are good features for tracking?*

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

*What are good features for tracking?*

Can be derived from the tracking algorithm

## *What are good features for tracking?*

Can be derived from the tracking algorithm

*‘A feature is good if it can be tracked well’*

Recall the Lucas-Kanade image alignment method:

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

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

Recall the Lucas-Kanade image alignment method:

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

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

Recall the Lucas-Kanade image alignment method:

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

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

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

Recall the Lucas-Kanade image alignment method:

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

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

Stability of gradient decent iterations depends on …

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

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?*

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?

## Above the noise level

$$\lambda_1 \gg 0$$

$$\lambda_2 \gg 0$$

both Eigenvalues are large

## Well-conditioned

both Eigenvalues have similar magnitude

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

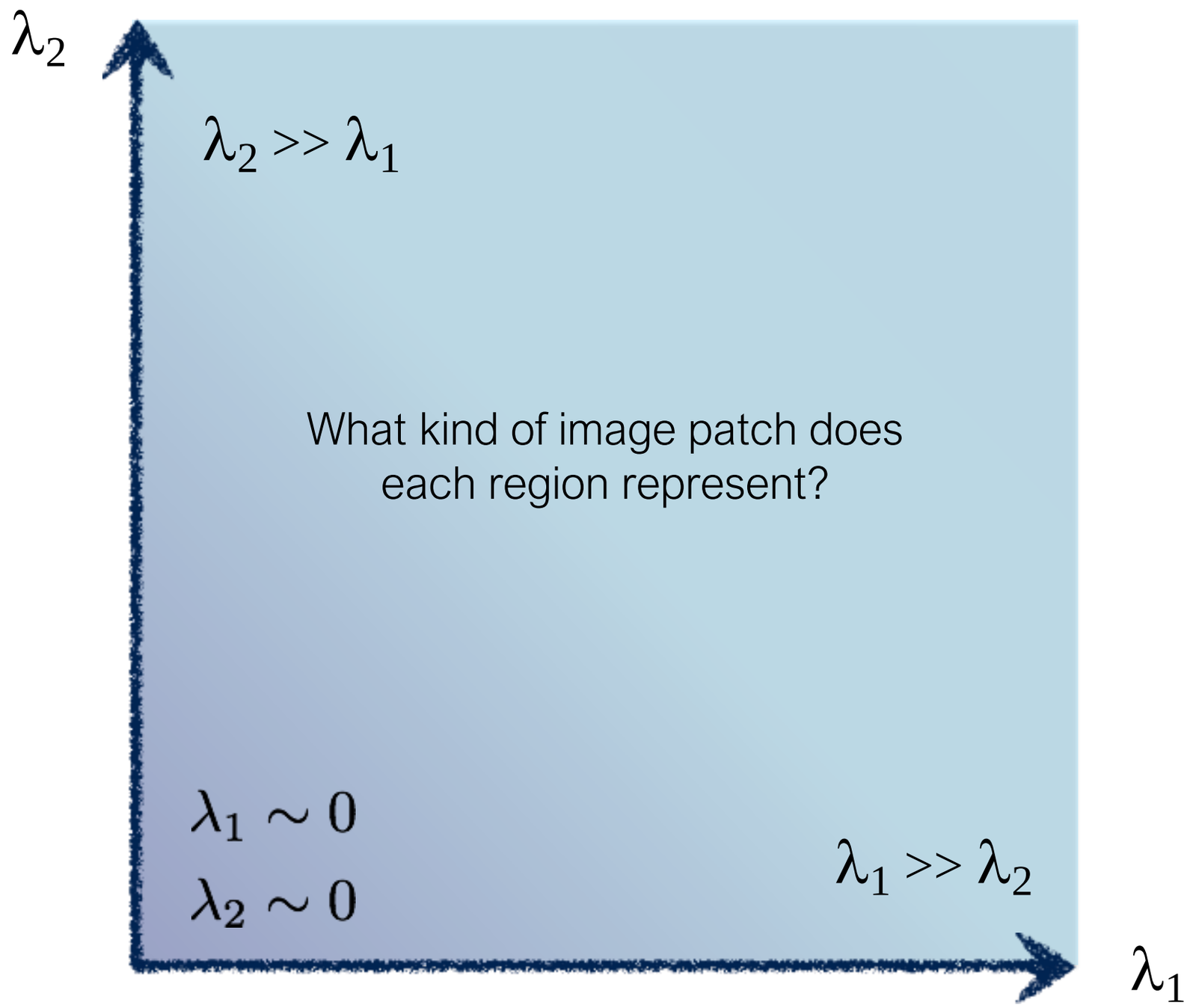
$$H = \sum_{\boldsymbol{x}} \left[\nabla I \frac{\partial \mathbf{W}}{\partial \boldsymbol{p}}\right]^\top \left[\nabla I \frac{\partial \mathbf{W}}{\partial \boldsymbol{p}}\right]$$

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

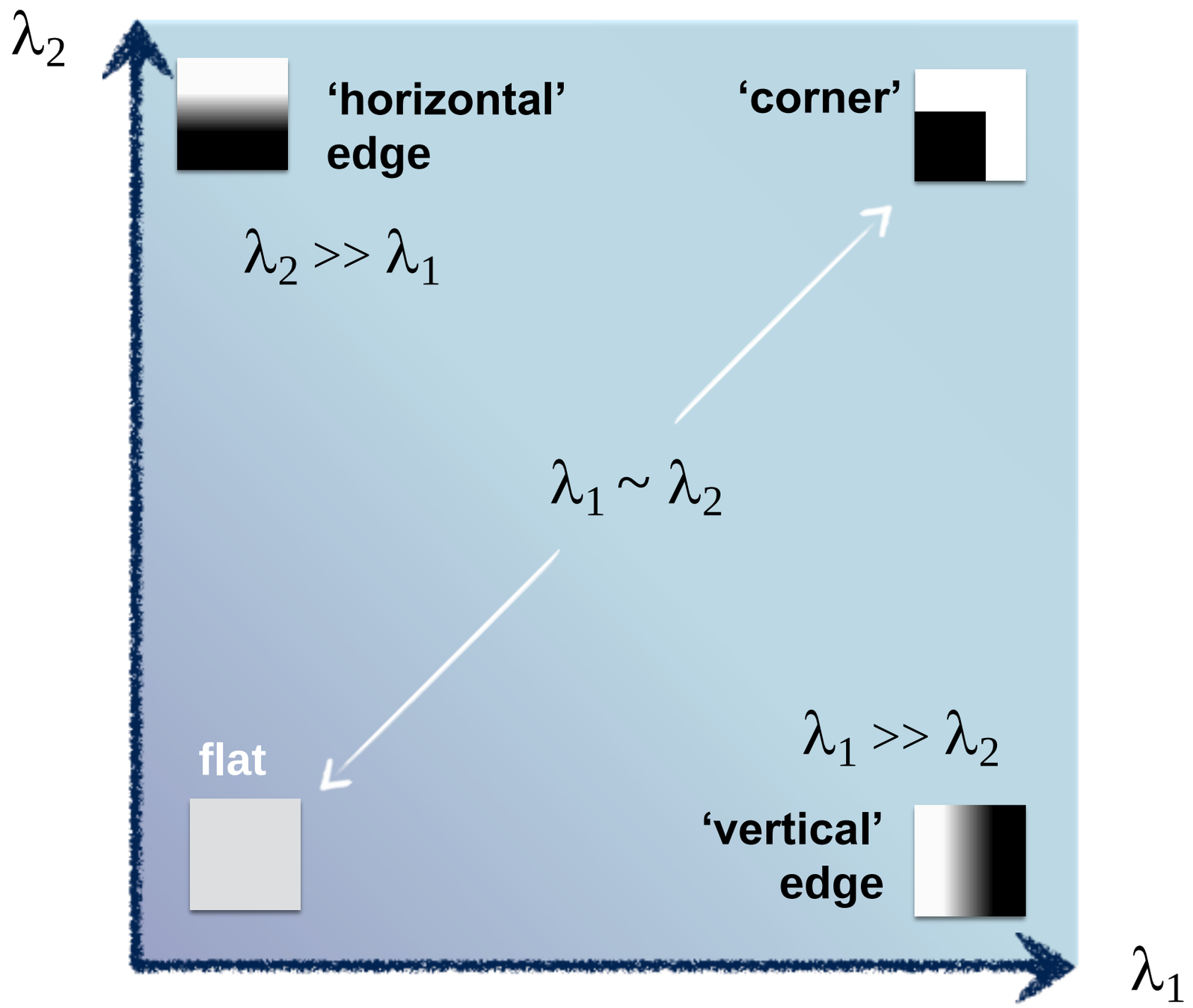
$$= \begin{bmatrix} \sum_{\boldsymbol{x}} I_x I_x & \sum_{\boldsymbol{x}} I_y I_x \\ \sum_{\boldsymbol{x}} I_x I_y & \sum_{\boldsymbol{x}} I_y I_y \end{bmatrix}$$ ←*when is this singular?*

*How are the eigenvalues related to image content?*

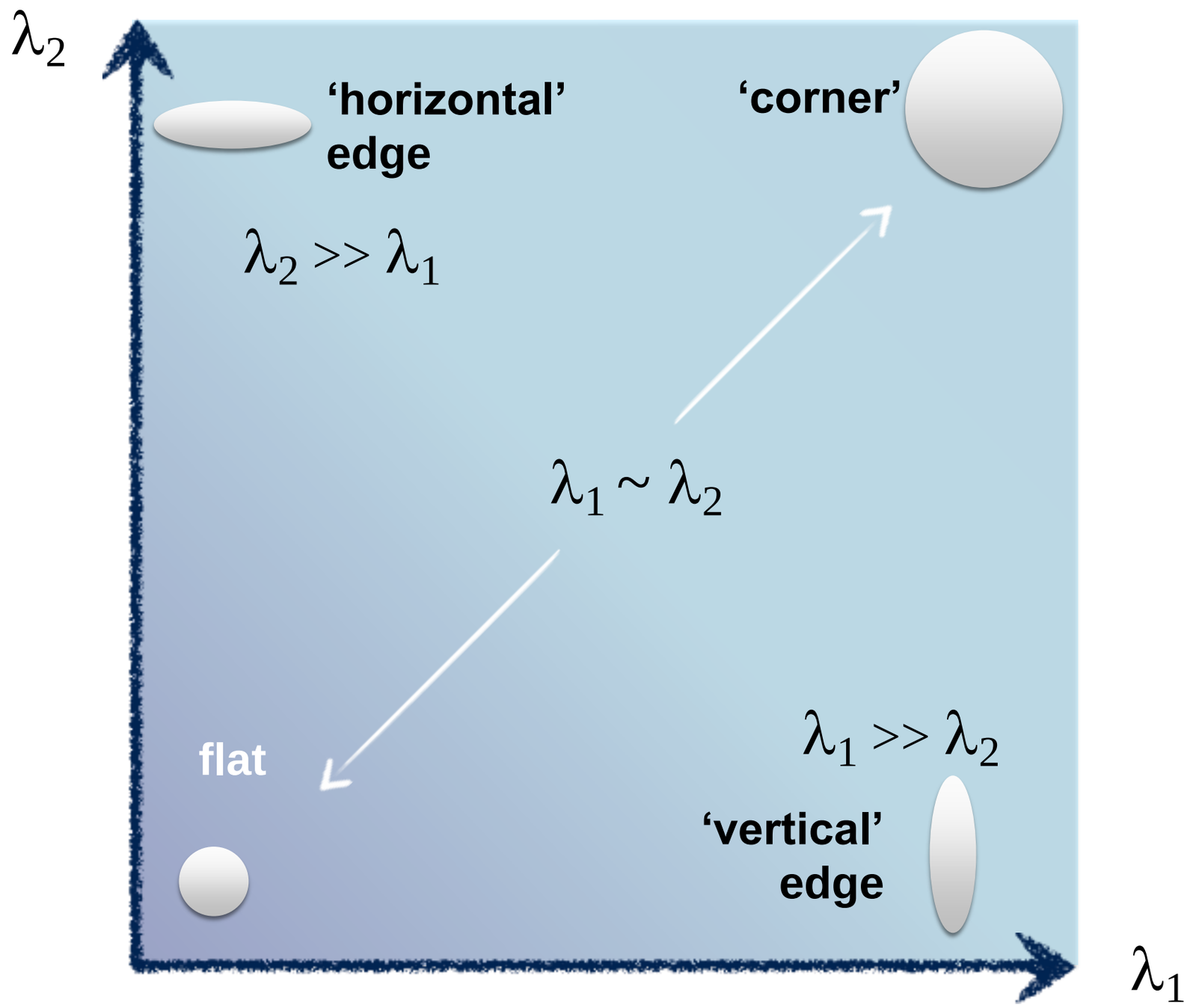
# interpreting eigenvalues



# interpreting eigenvalues



# interpreting eigenvalues



*What are good features for tracking?*

*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

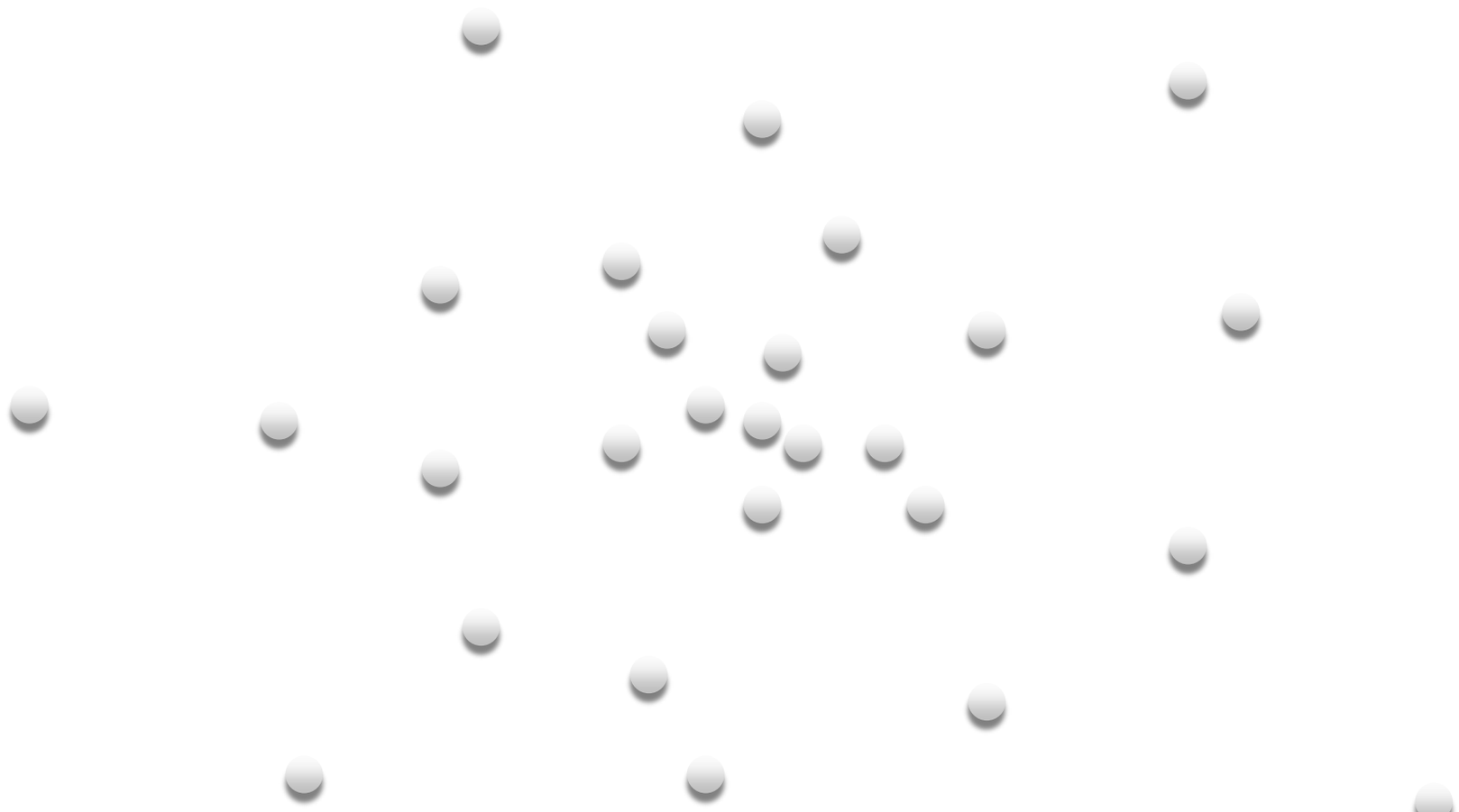
# Mean-shift algorithm


Nationale-Nederlanden
jackpot €5 mil
ING

# Mean Shift Algorithm

## A ‘mode seeking’ algorithm

Fukunaga & Hostetler (1975)

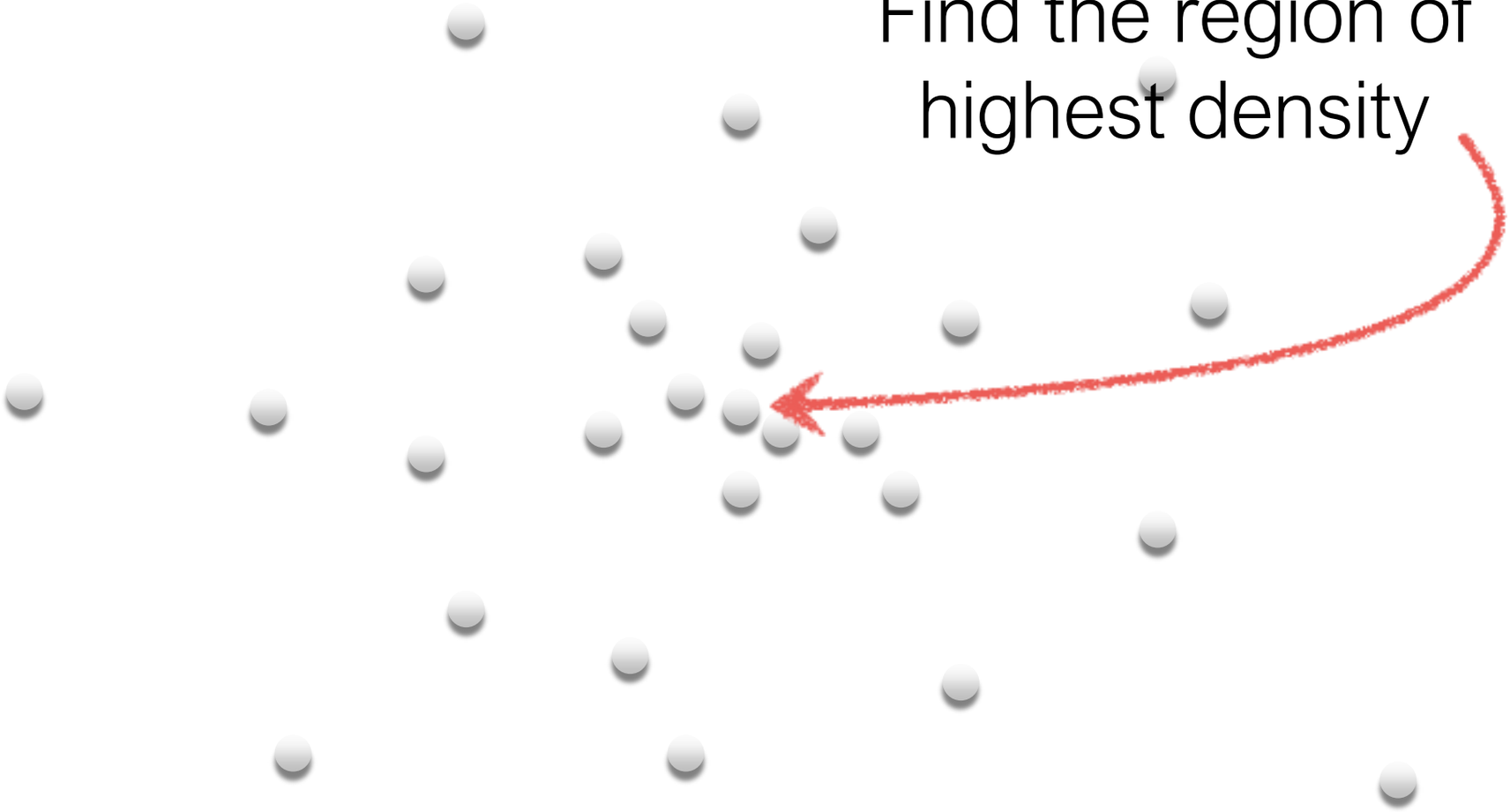


# Mean Shift Algorithm

## A ‘mode seeking’ algorithm

Fukunaga & Hostetler (1975)

Find the region of highest density



# Mean Shift Algorithm

## A ‘mode seeking’ algorithm

Fukunaga & Hostetler (1975)

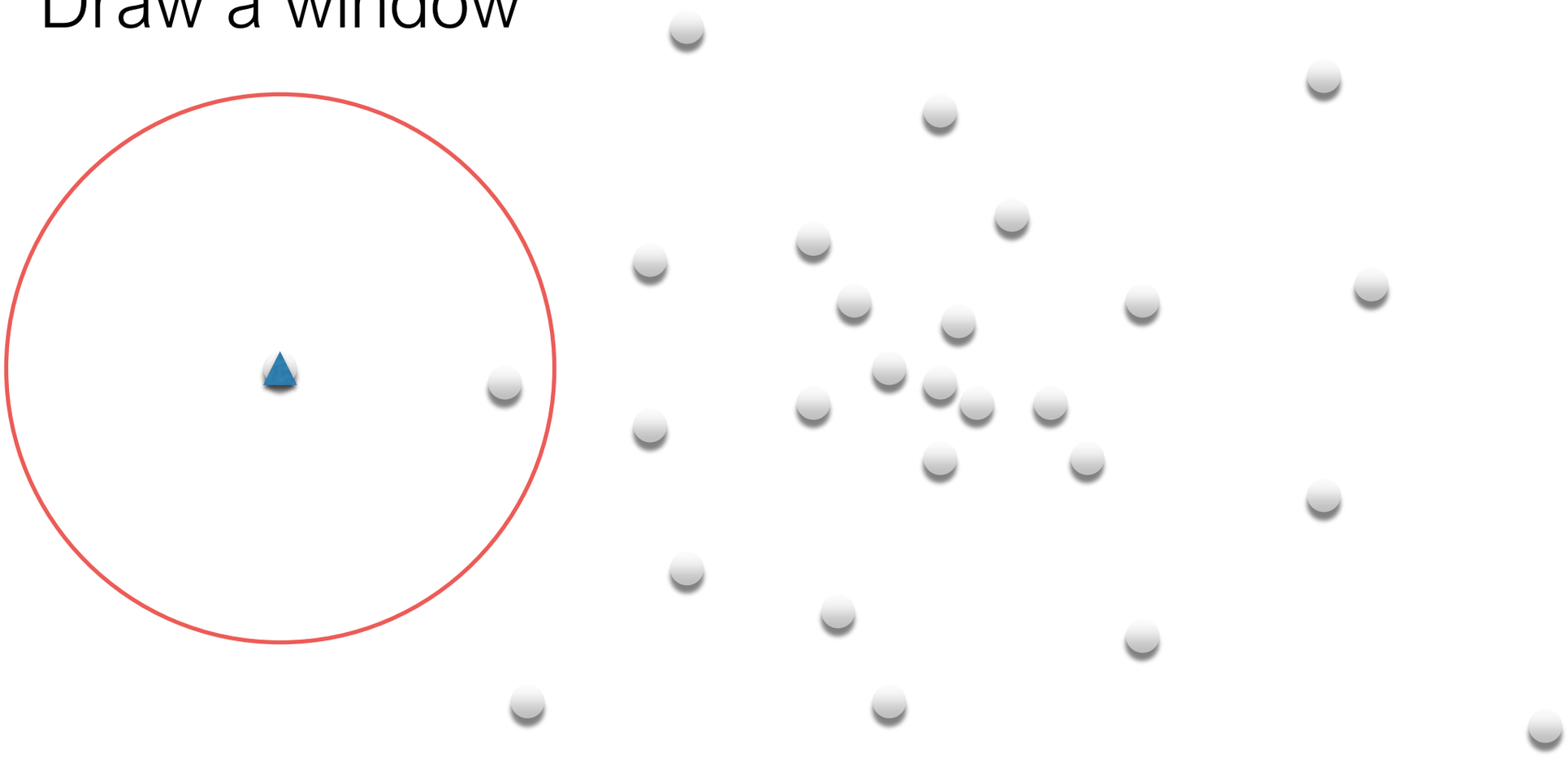
Pick a point

# Mean Shift Algorithm

## A ‘mode seeking’ algorithm

Fukunaga & Hostetler (1975)

Draw a window

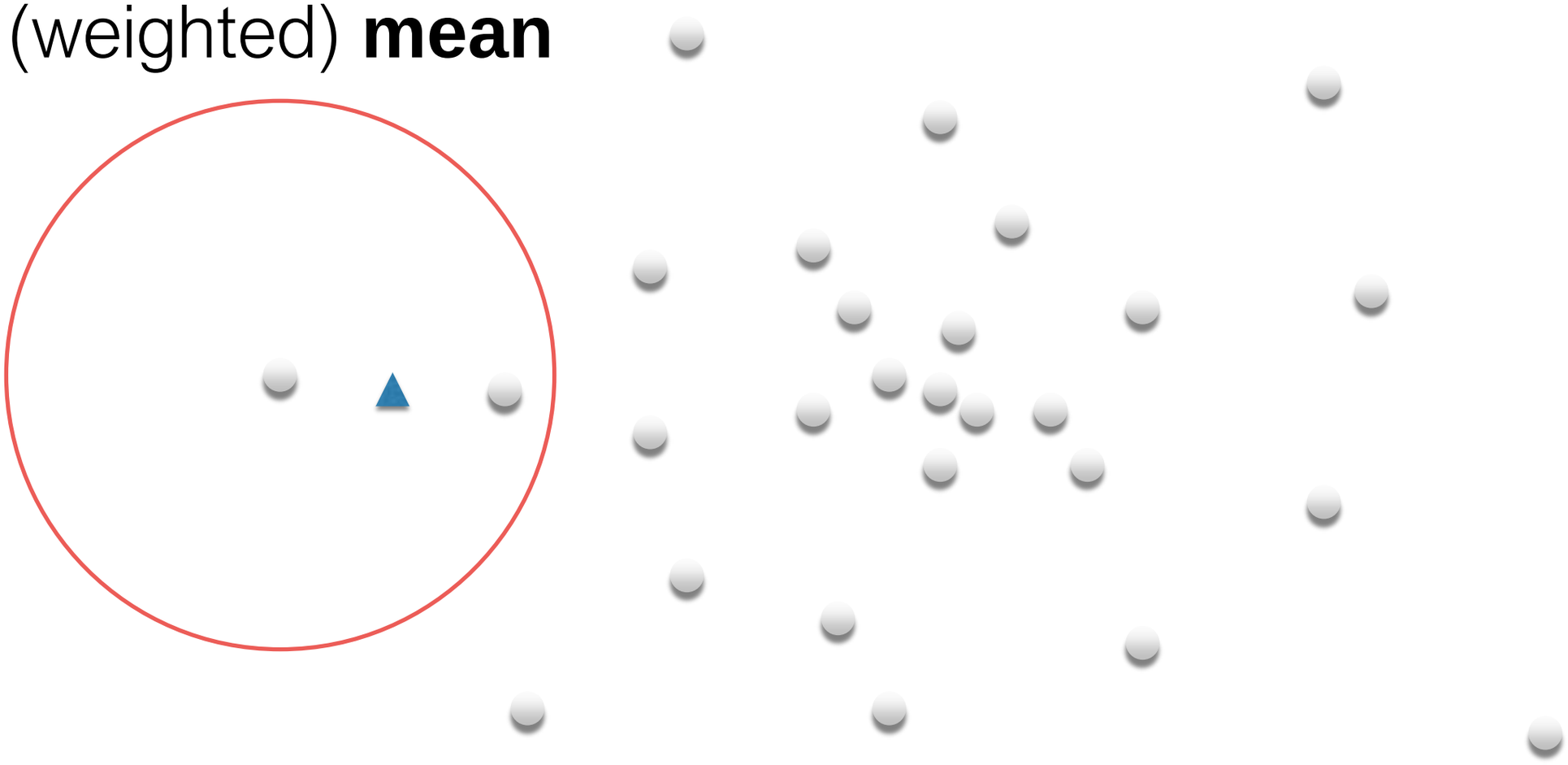


# Mean Shift Algorithm

## A ‘mode seeking’ algorithm

Fukunaga & Hostetler (1975)

Compute the
(weighted) **mean**

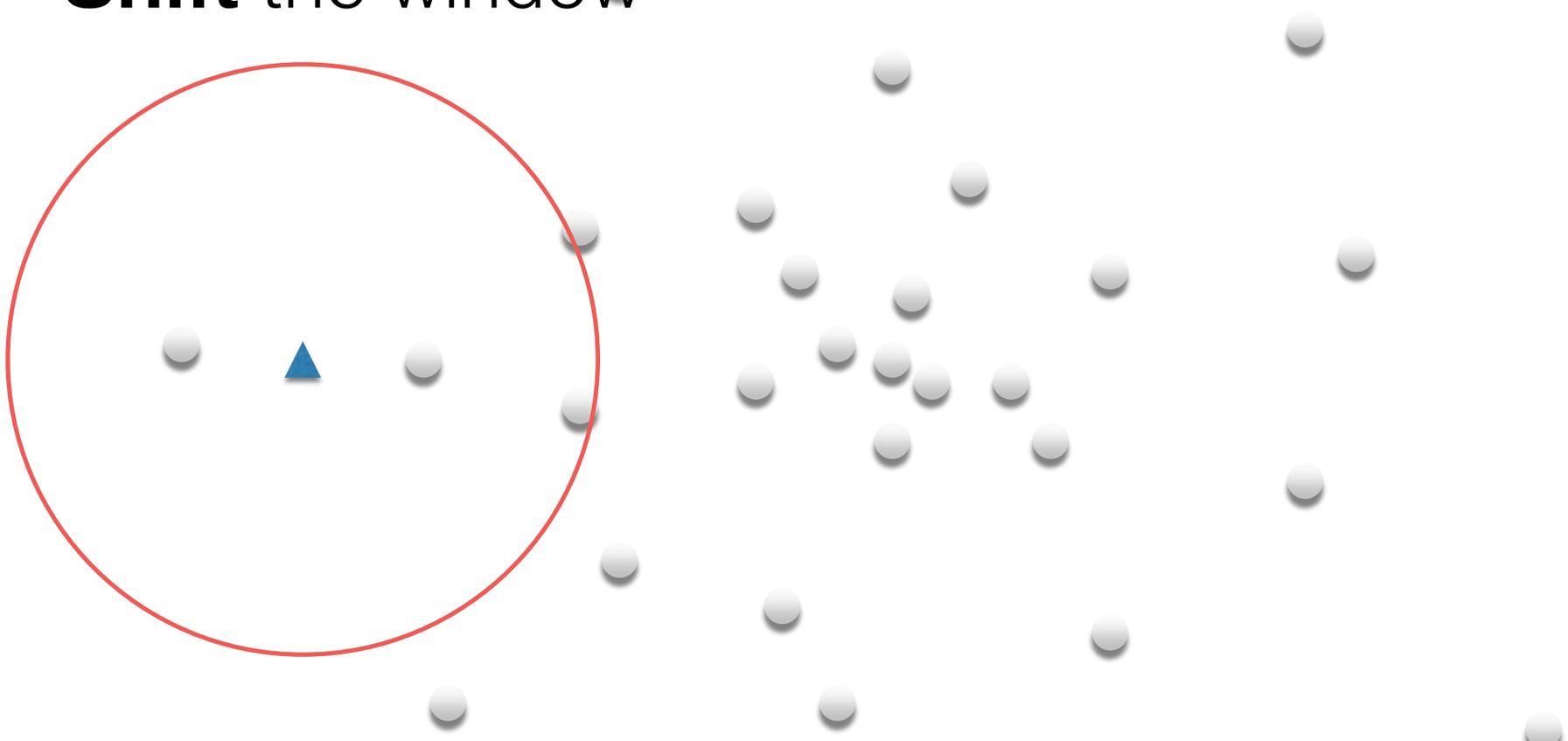


# Mean Shift Algorithm

## A ‘mode seeking’ algorithm

Fukunaga & Hostetler (1975)

**Shift** the window

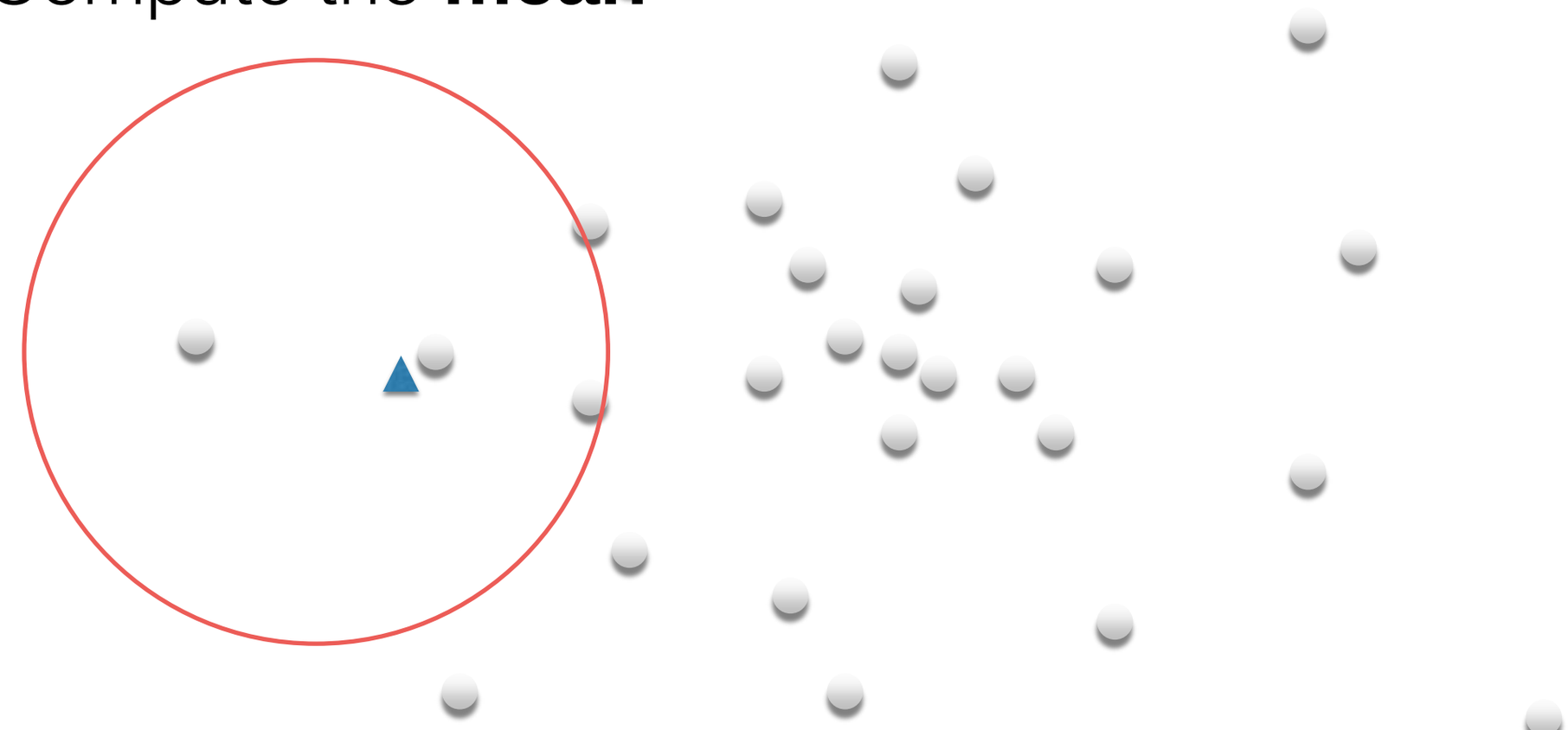


# Mean Shift Algorithm

## A ‘mode seeking’ algorithm

Fukunaga & Hostetler (1975)

Compute the **mean**

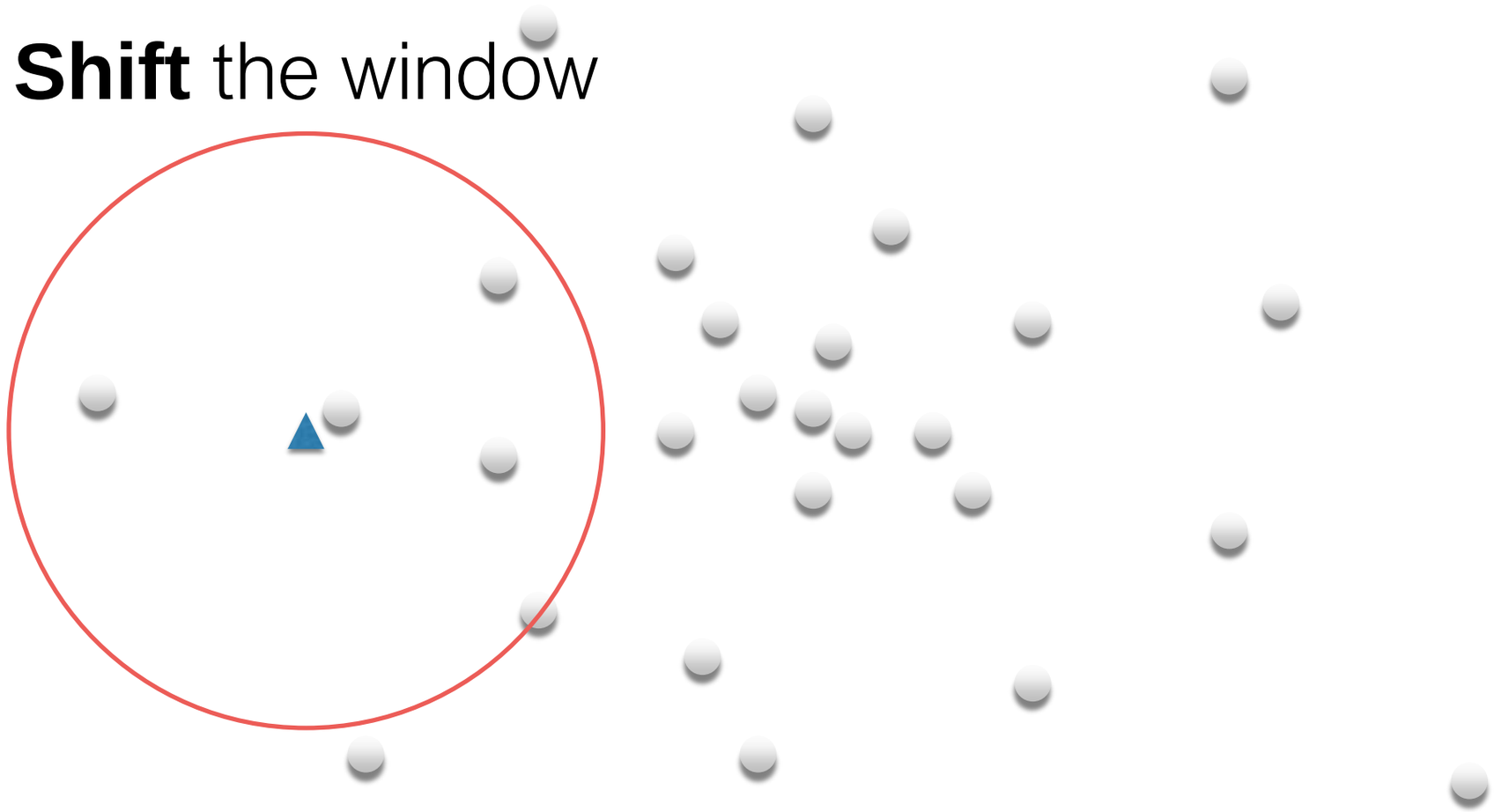


# Mean Shift Algorithm

## A ‘mode seeking’ algorithm

Fukunaga & Hostetler (1975)

**Shift** the window

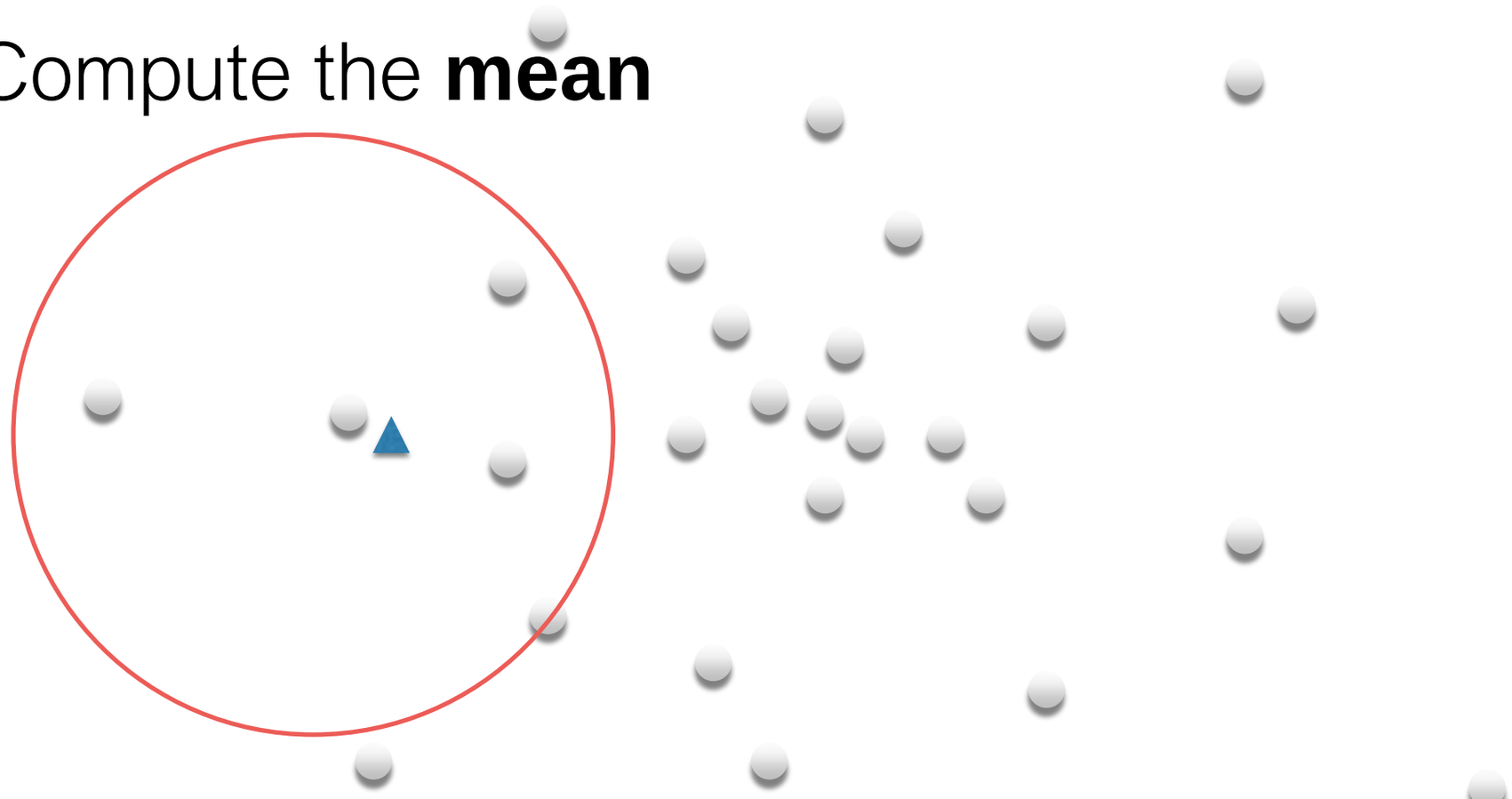


# Mean Shift Algorithm

## A ‘mode seeking’ algorithm

Fukunaga & Hostetler (1975)

Compute the **mean**

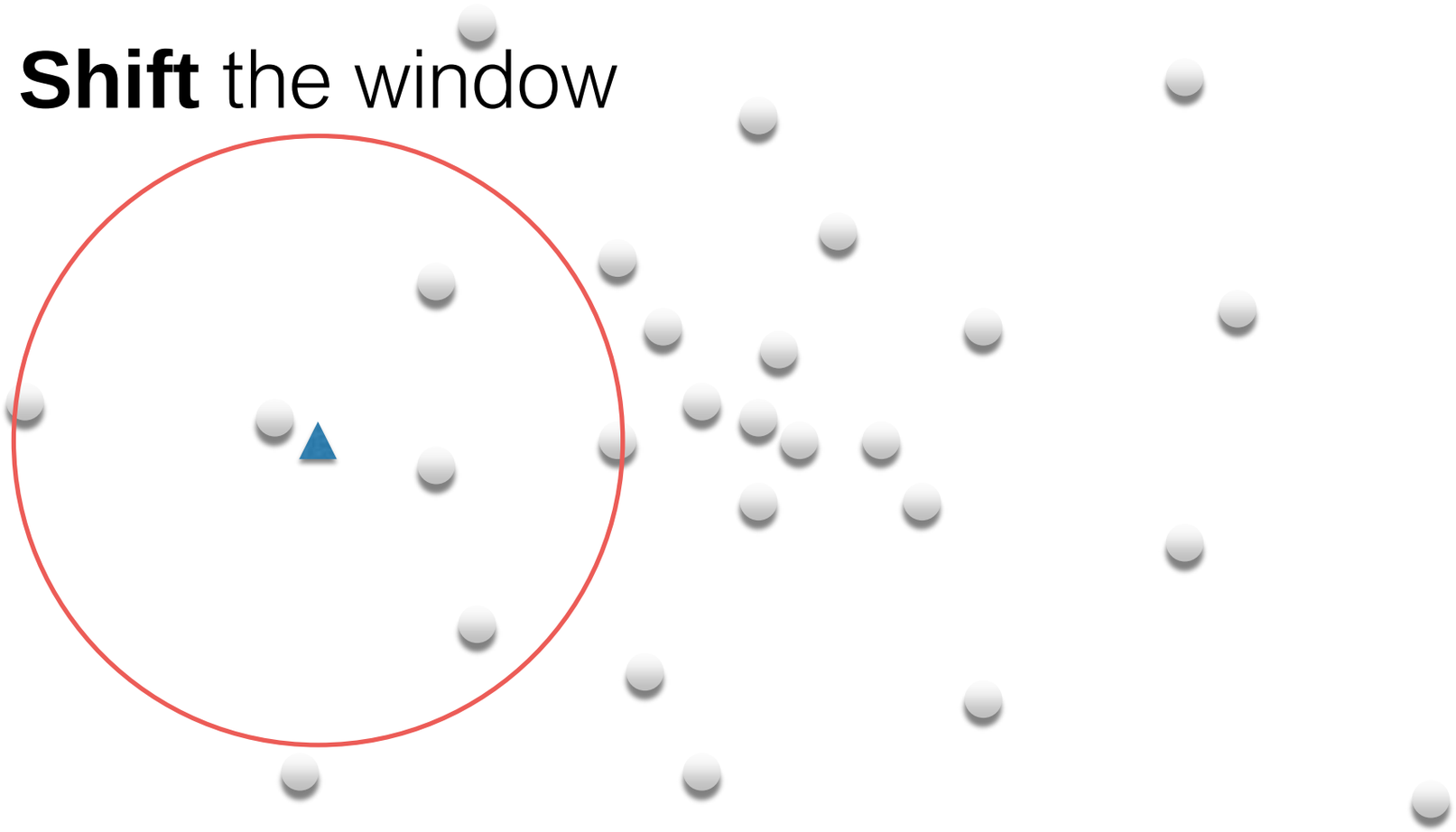


# Mean Shift Algorithm

## A ‘mode seeking’ algorithm

Fukunaga & Hostetler (1975)

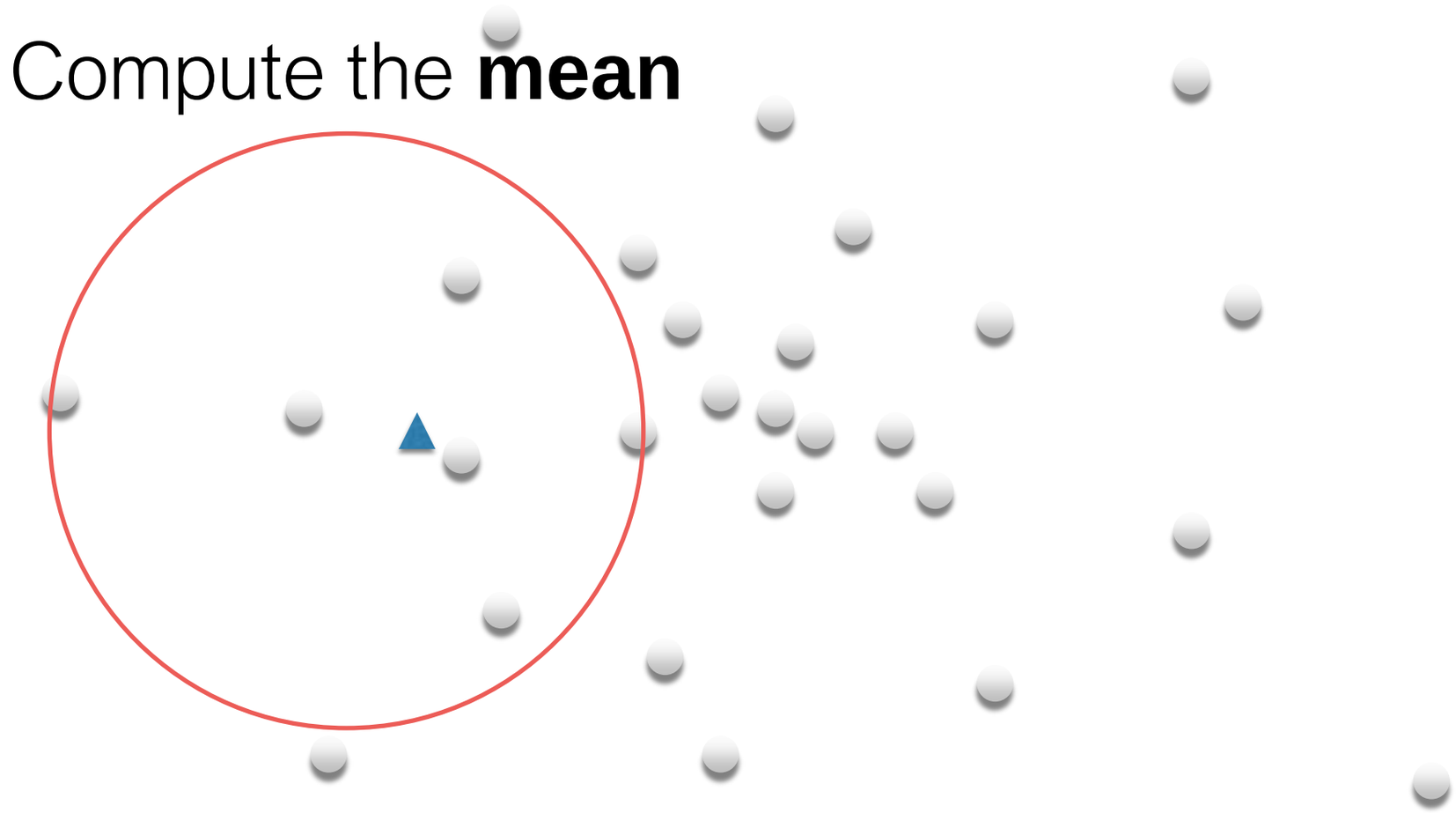
**Shift** the window



# Mean Shift Algorithm

## A ‘mode seeking’ algorithm

Fukunaga & Hostetler (1975)

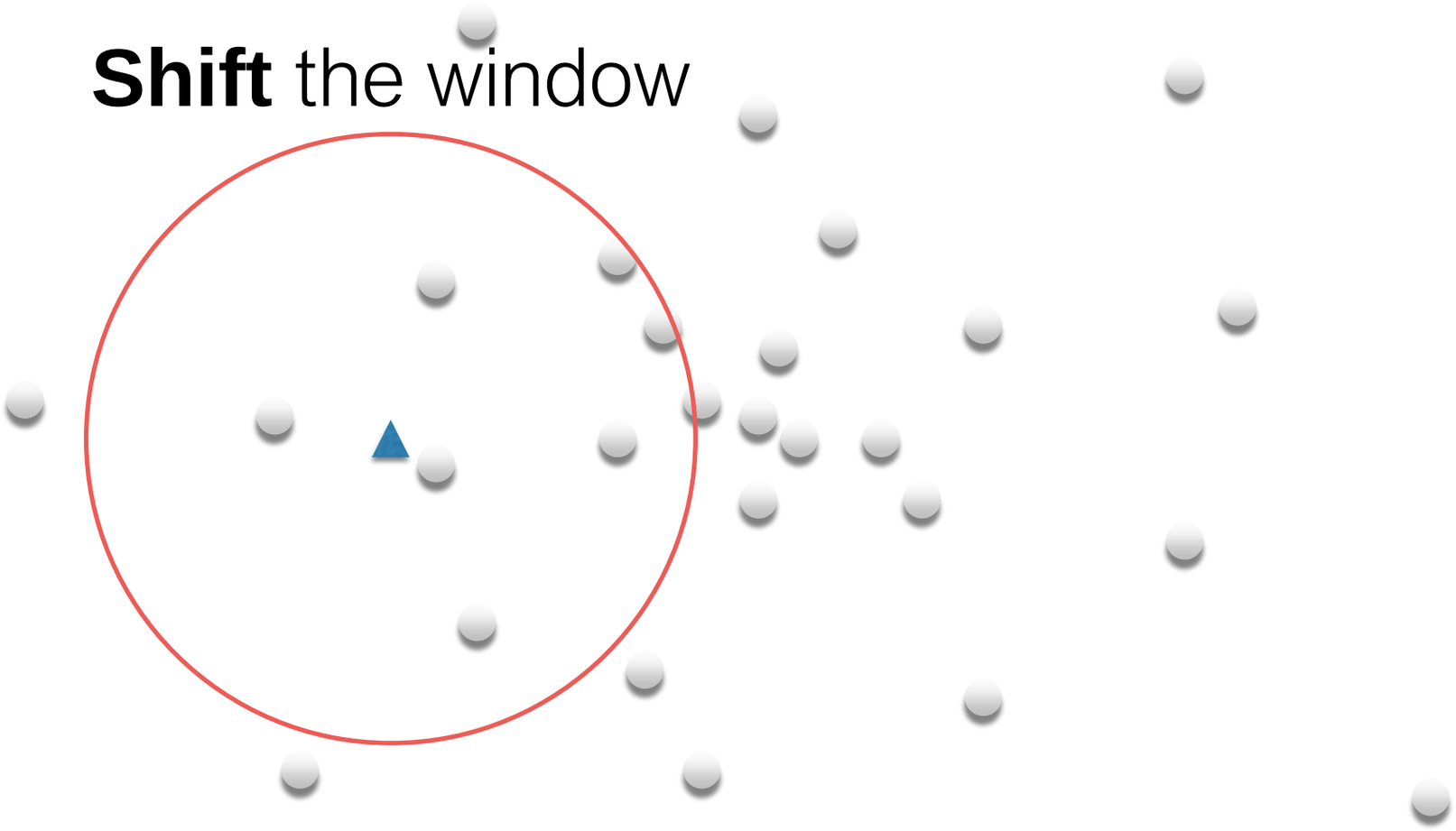


# Mean Shift Algorithm

## A ‘mode seeking’ algorithm

Fukunaga & Hostetler (1975)

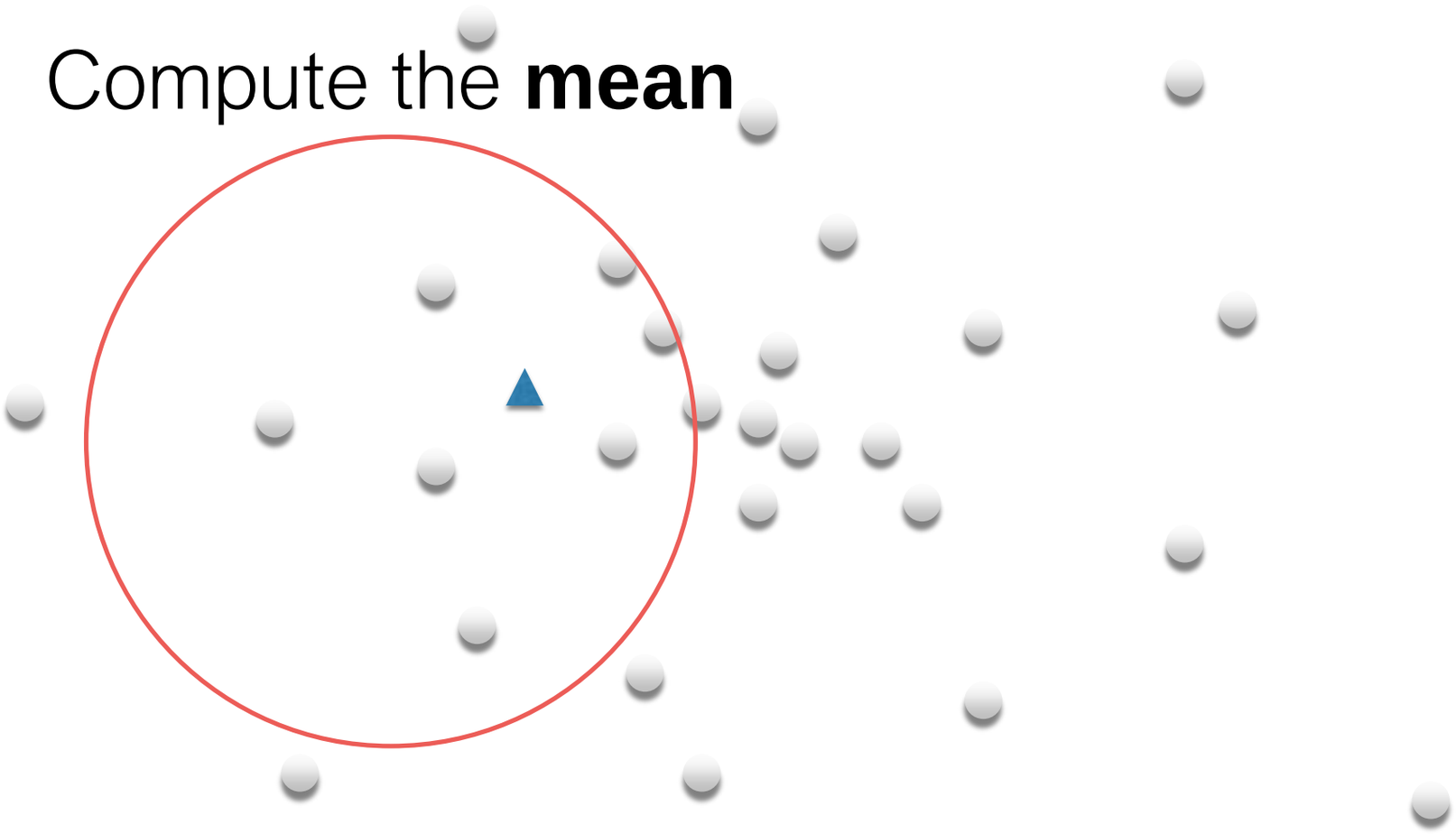
**Shift** the window



# Mean Shift Algorithm

## A ‘mode seeking’ algorithm

Fukunaga & Hostetler (1975)

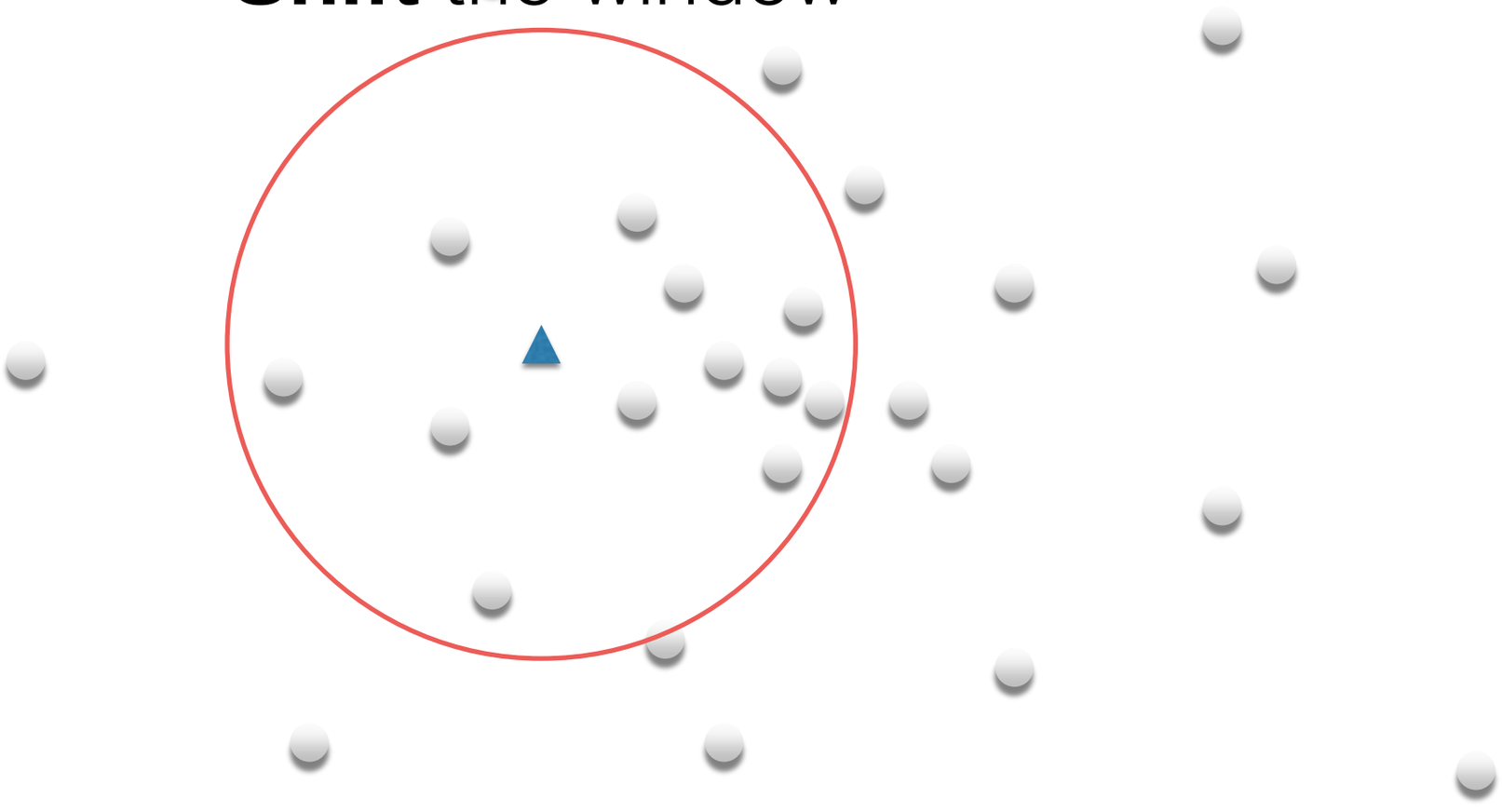


# Mean Shift Algorithm

## A ‘mode seeking’ algorithm

Fukunaga & Hostetler (1975)

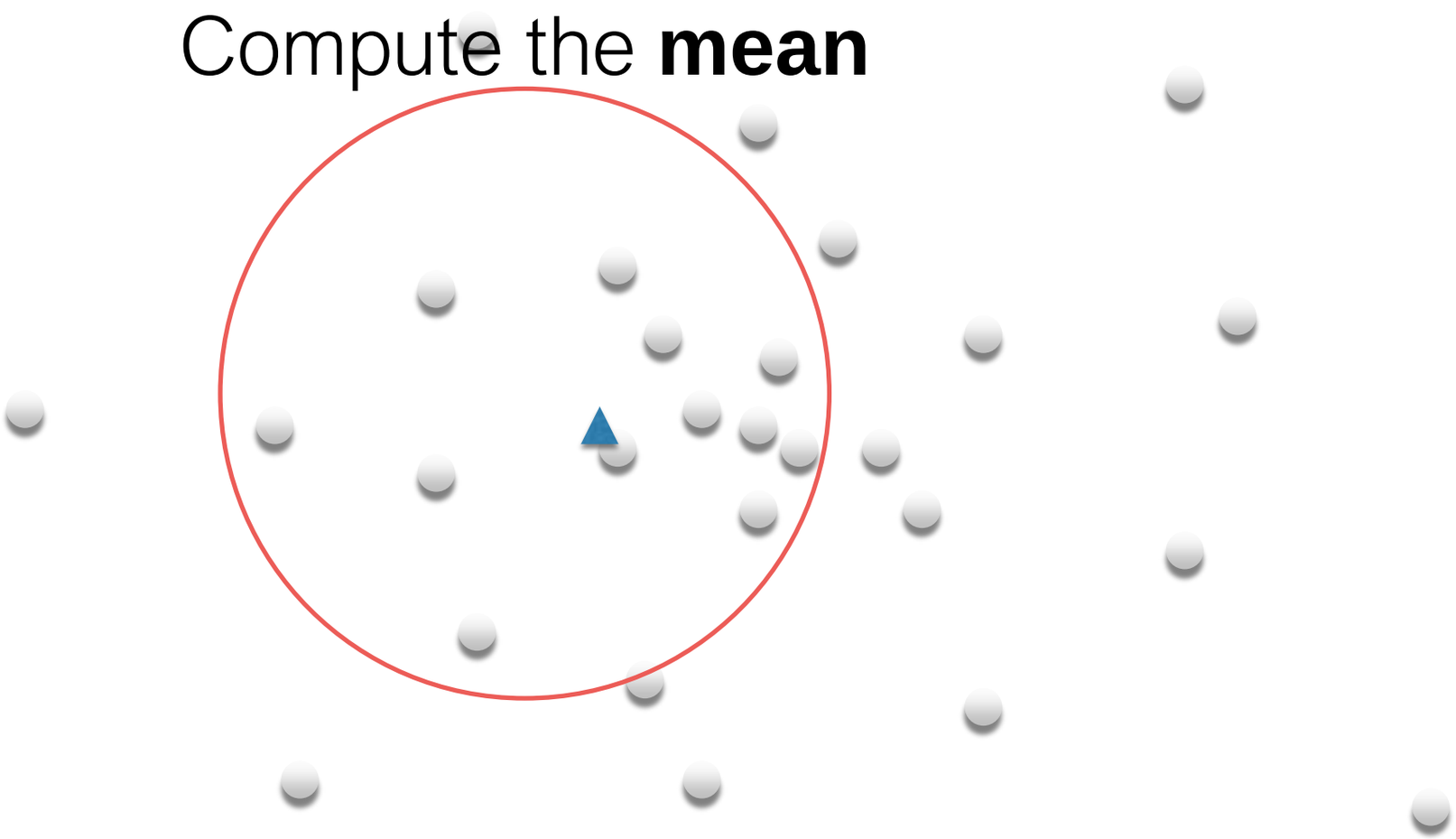
**Shift** the window



# Mean Shift Algorithm

## A ‘mode seeking’ algorithm

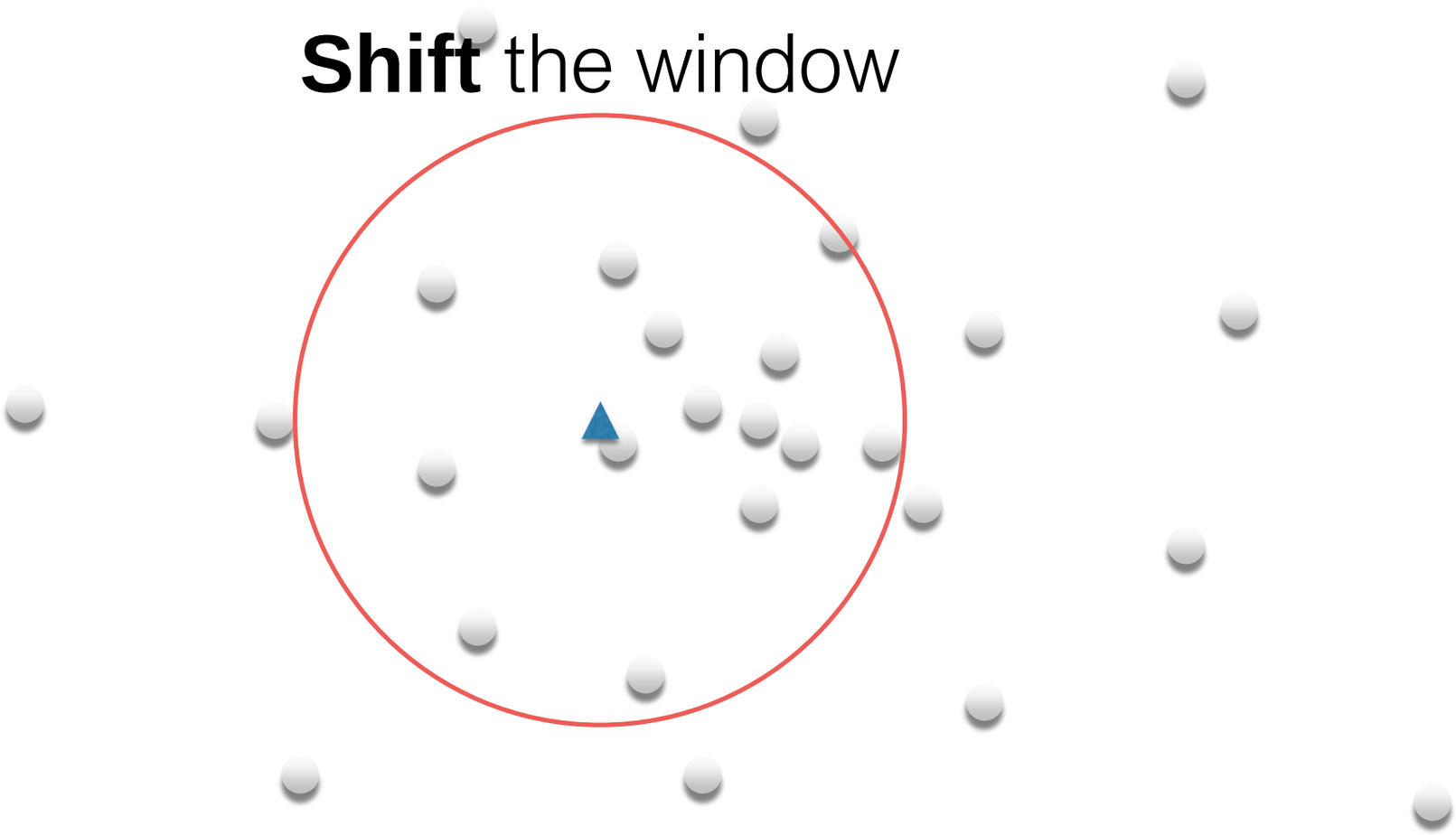
Fukunaga & Hostetler (1975)



# Mean Shift Algorithm

## A ‘mode seeking’ algorithm

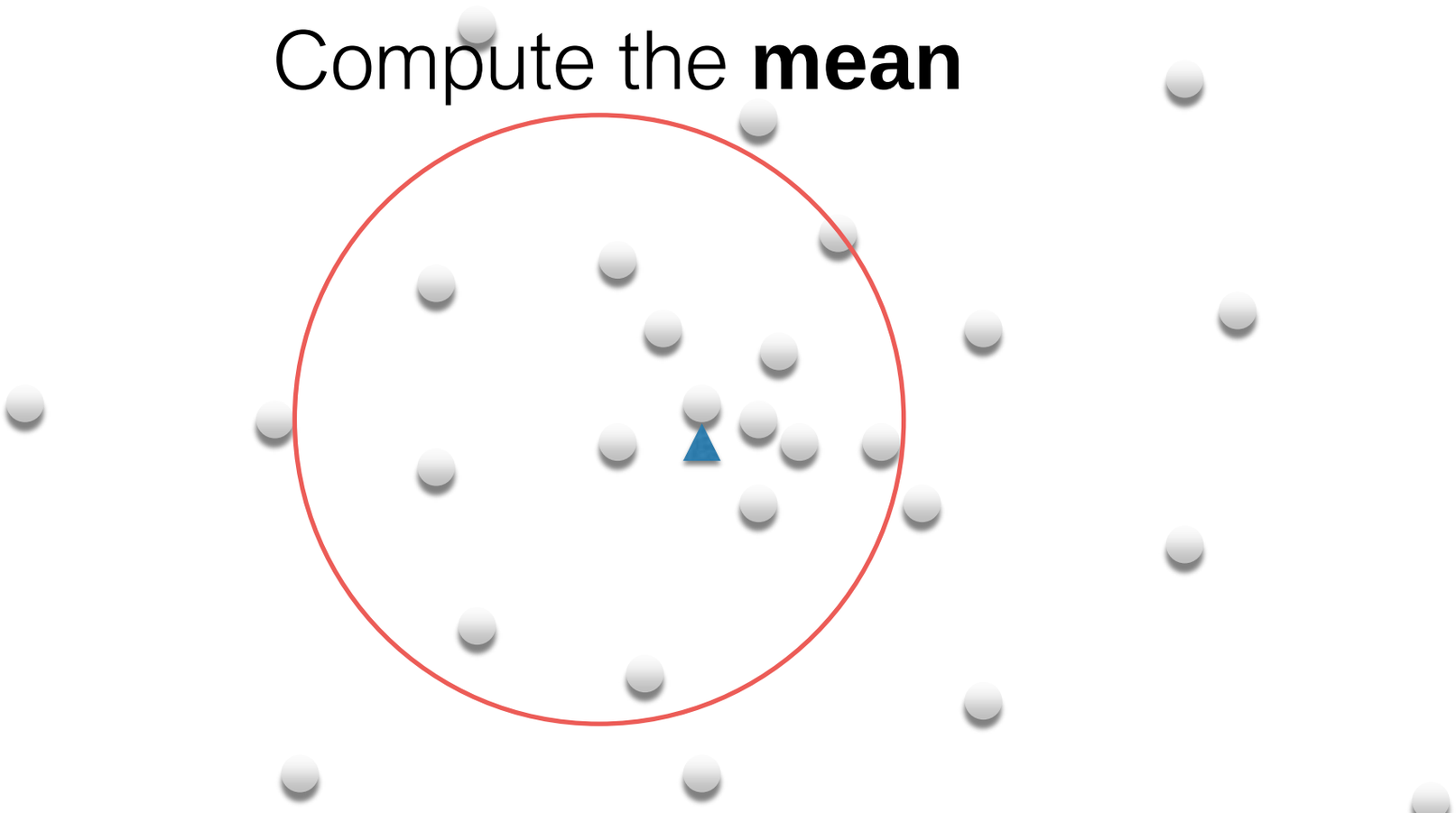
Fukunaga & Hostetler (1975)



# Mean Shift Algorithm

## A ‘mode seeking’ algorithm

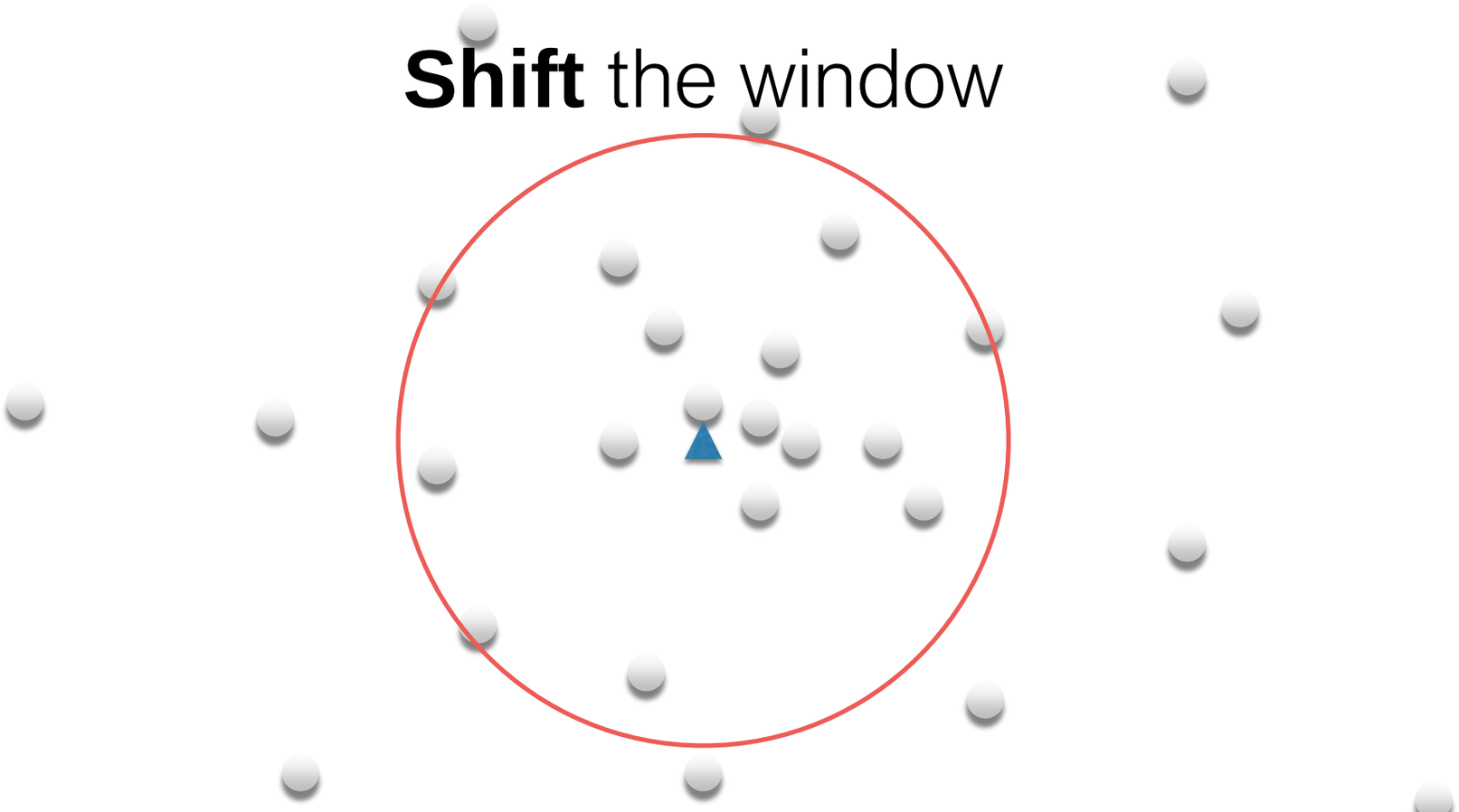
Fukunaga & Hostetler (1975)



# Mean Shift Algorithm

## A ‘mode seeking’ algorithm

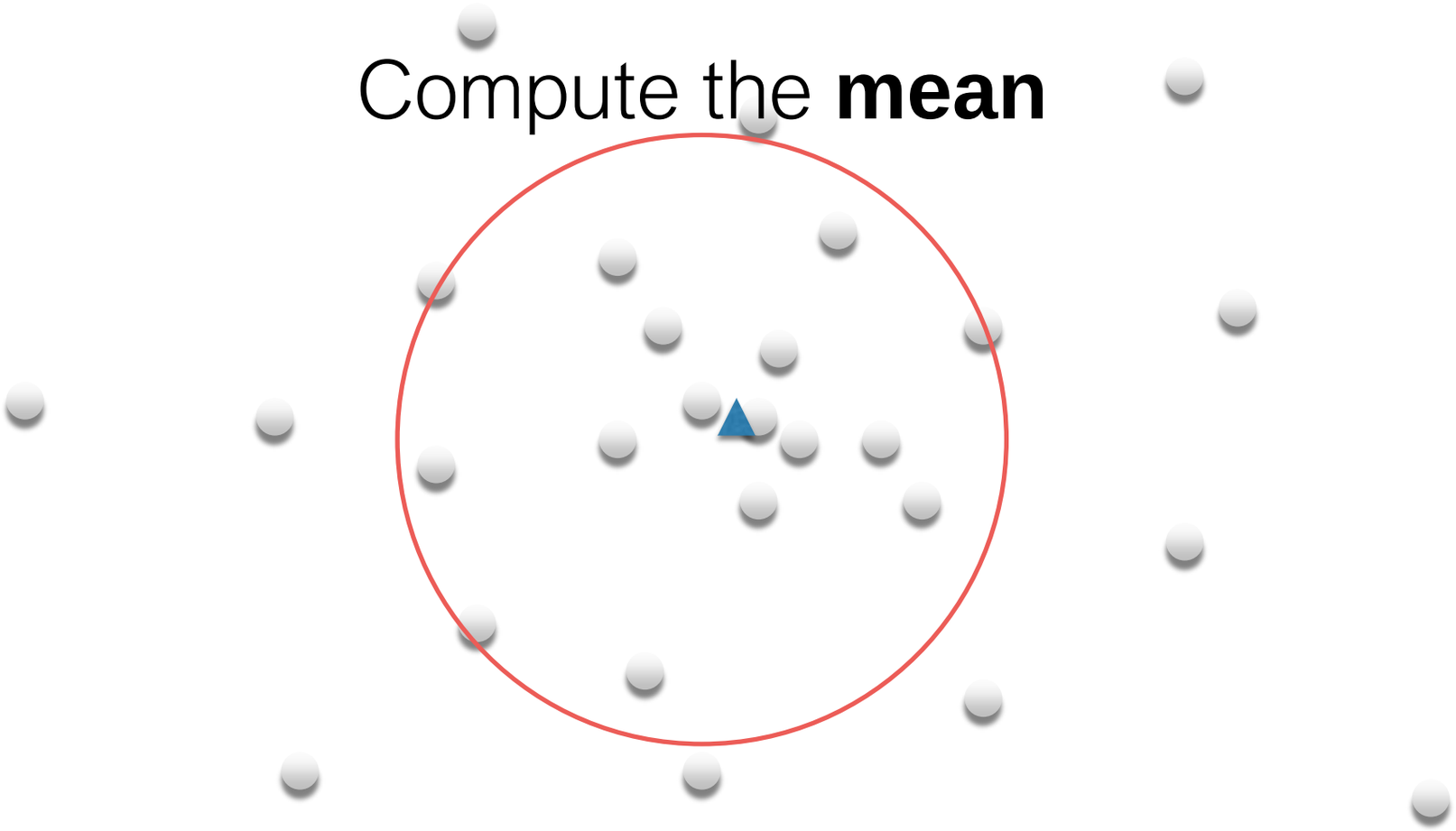
Fukunaga & Hostetler (1975)



# Mean Shift Algorithm

## A ‘mode seeking’ algorithm

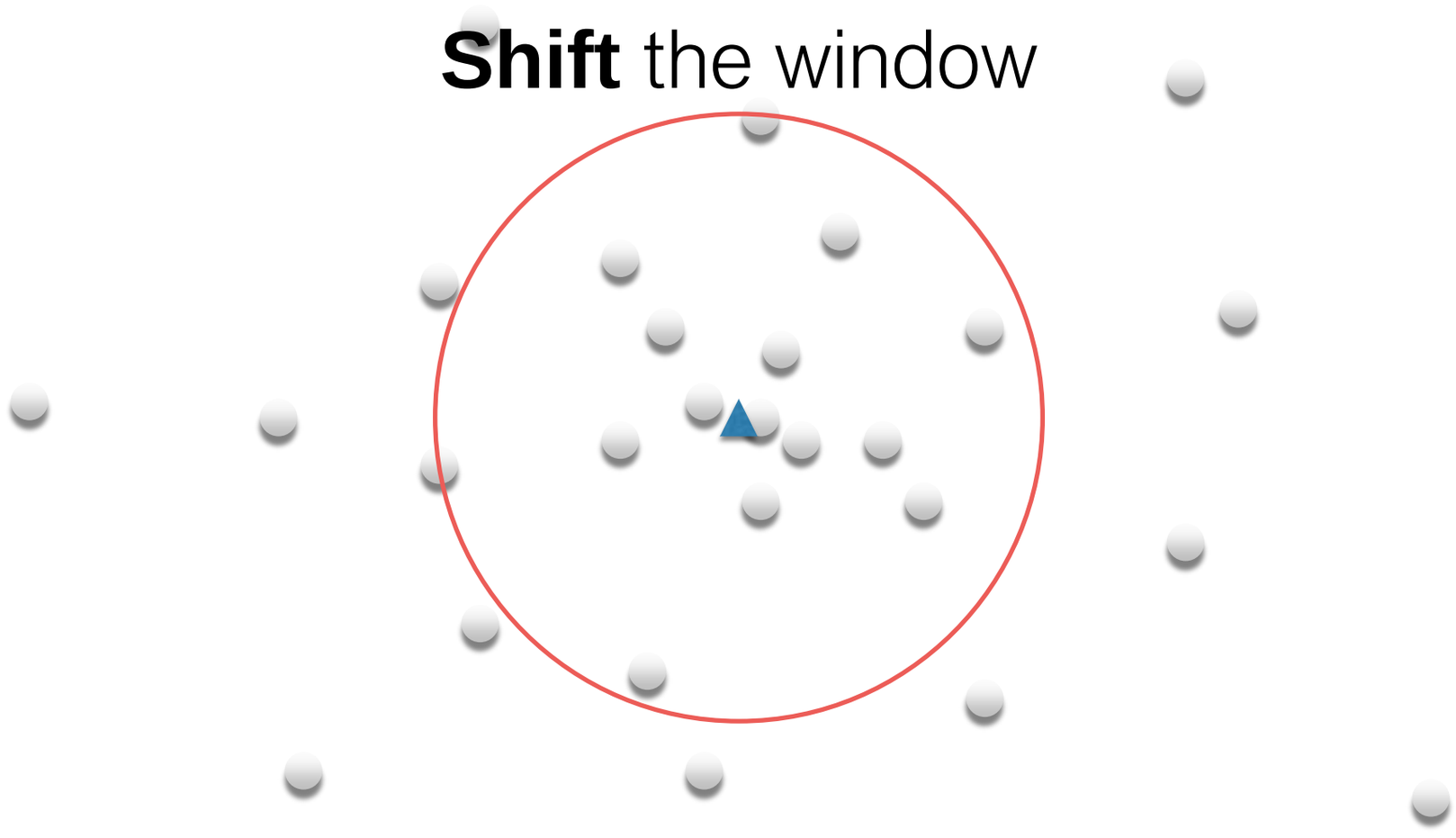
Fukunaga & Hostetler (1975)



# Mean Shift Algorithm

## A ‘mode seeking’ algorithm

Fukunaga & Hostetler (1975)



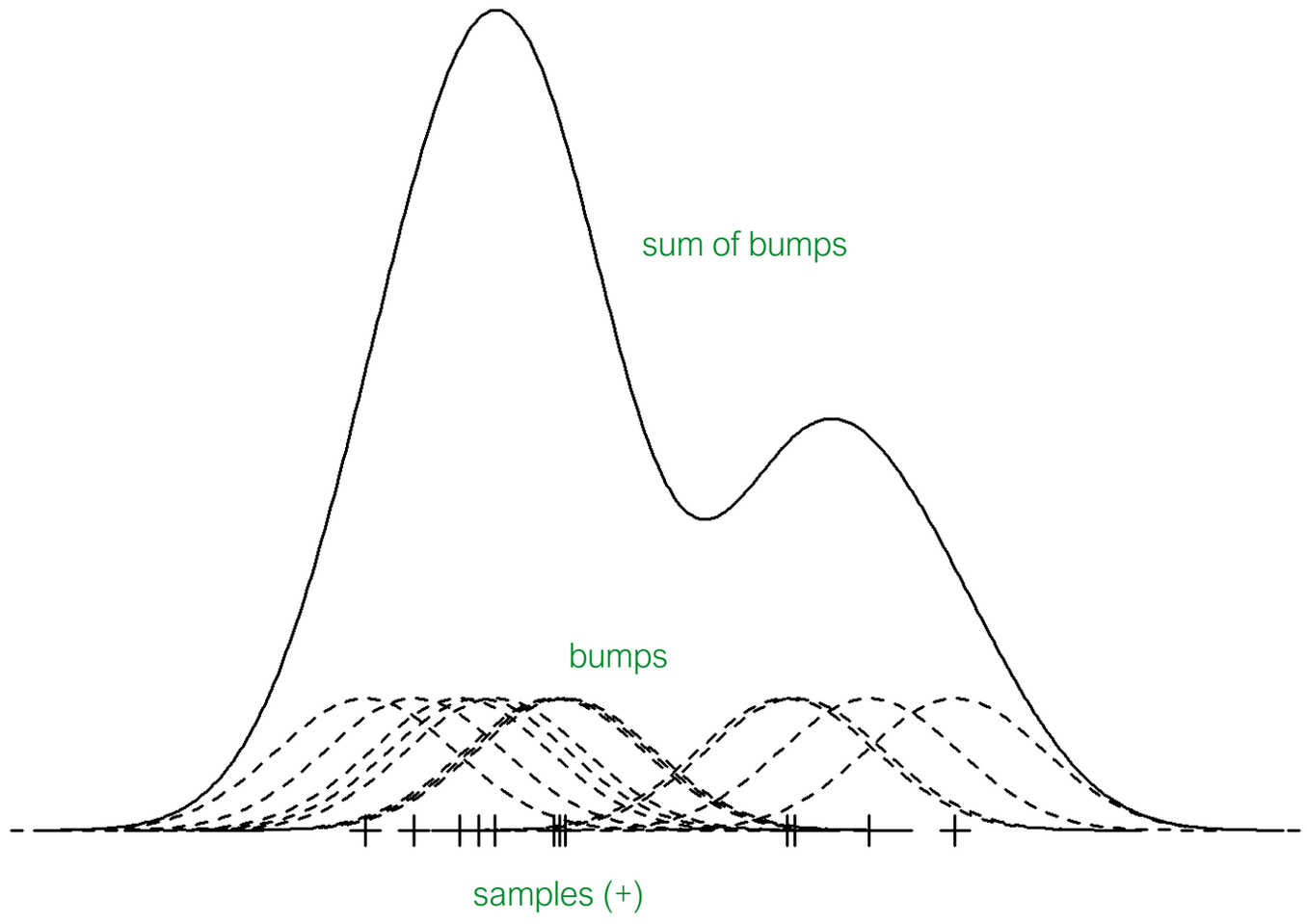

To understand the theory behind this we need to understand…

# Kernel density estimation

To understand the mean shift algorithm …

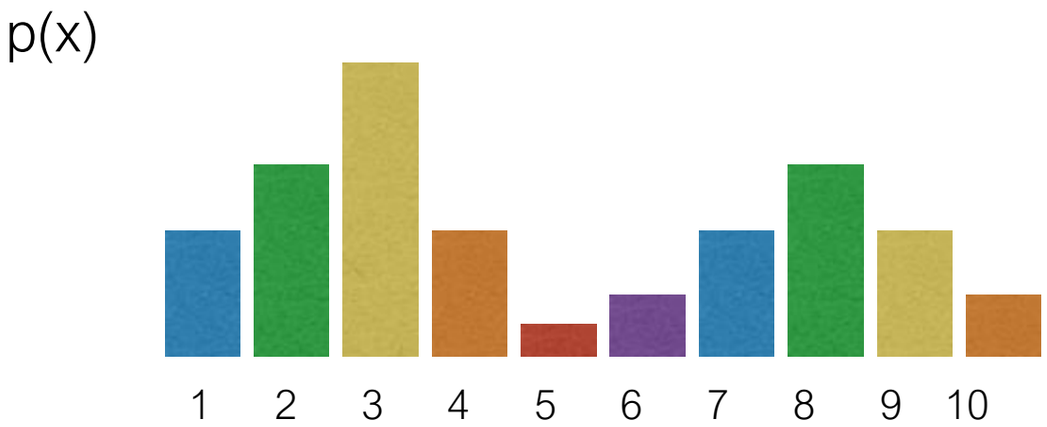
# Kernel Density Estimation

## A method to approximate an underlying PDF from samples

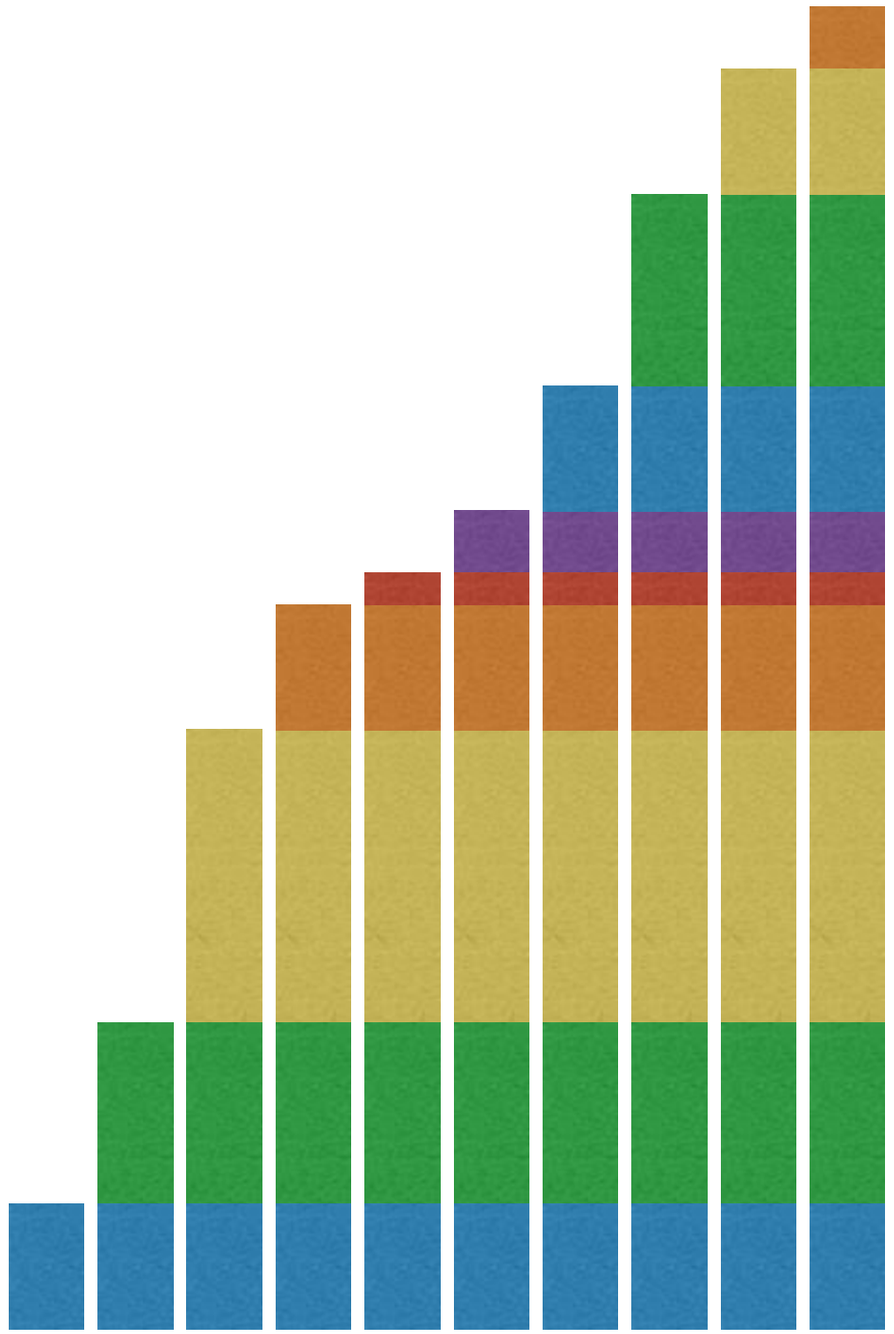



## Put ‘bump’ on every sample to approximate the PDF

Say we have some hidden PDF…

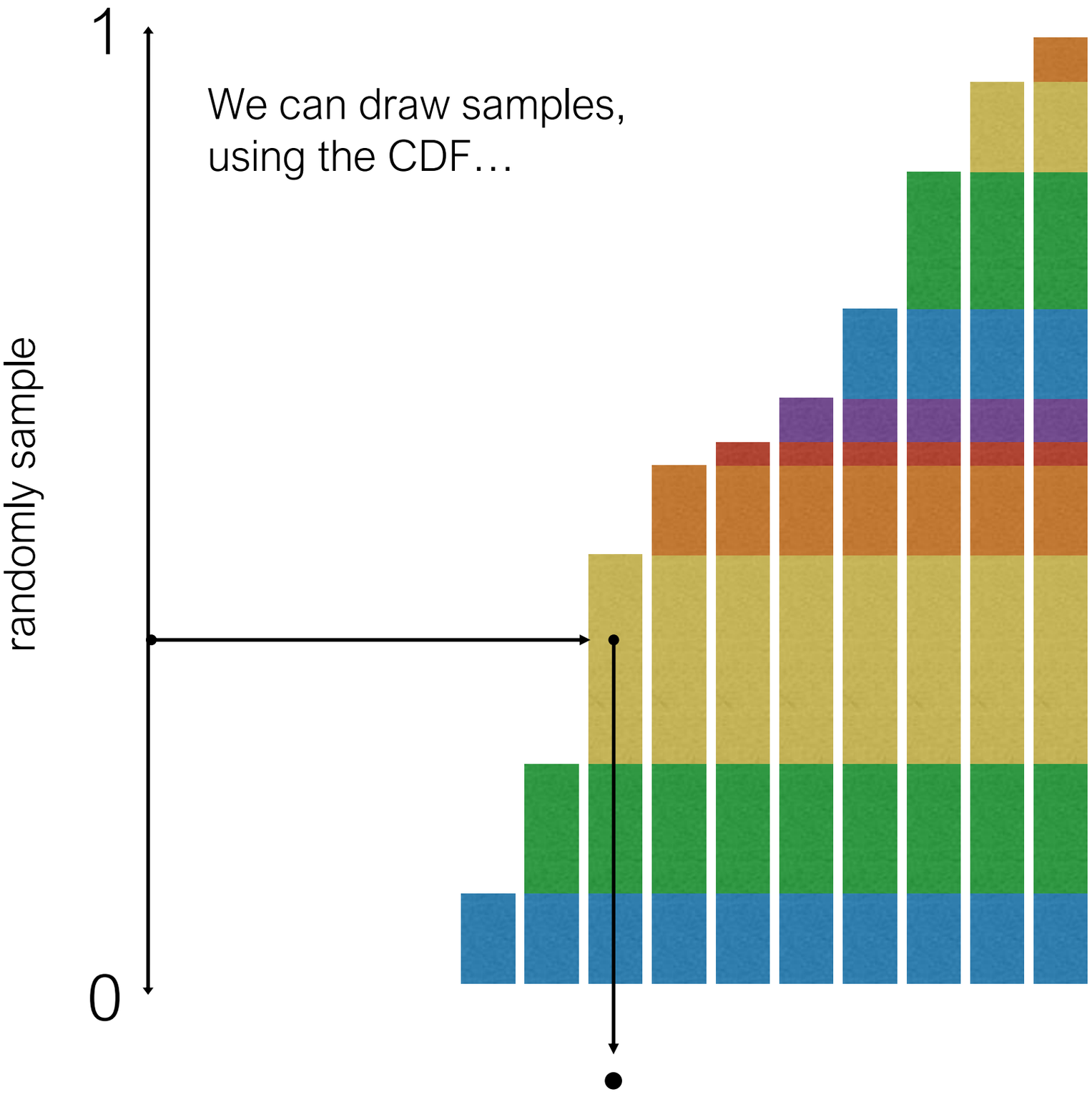


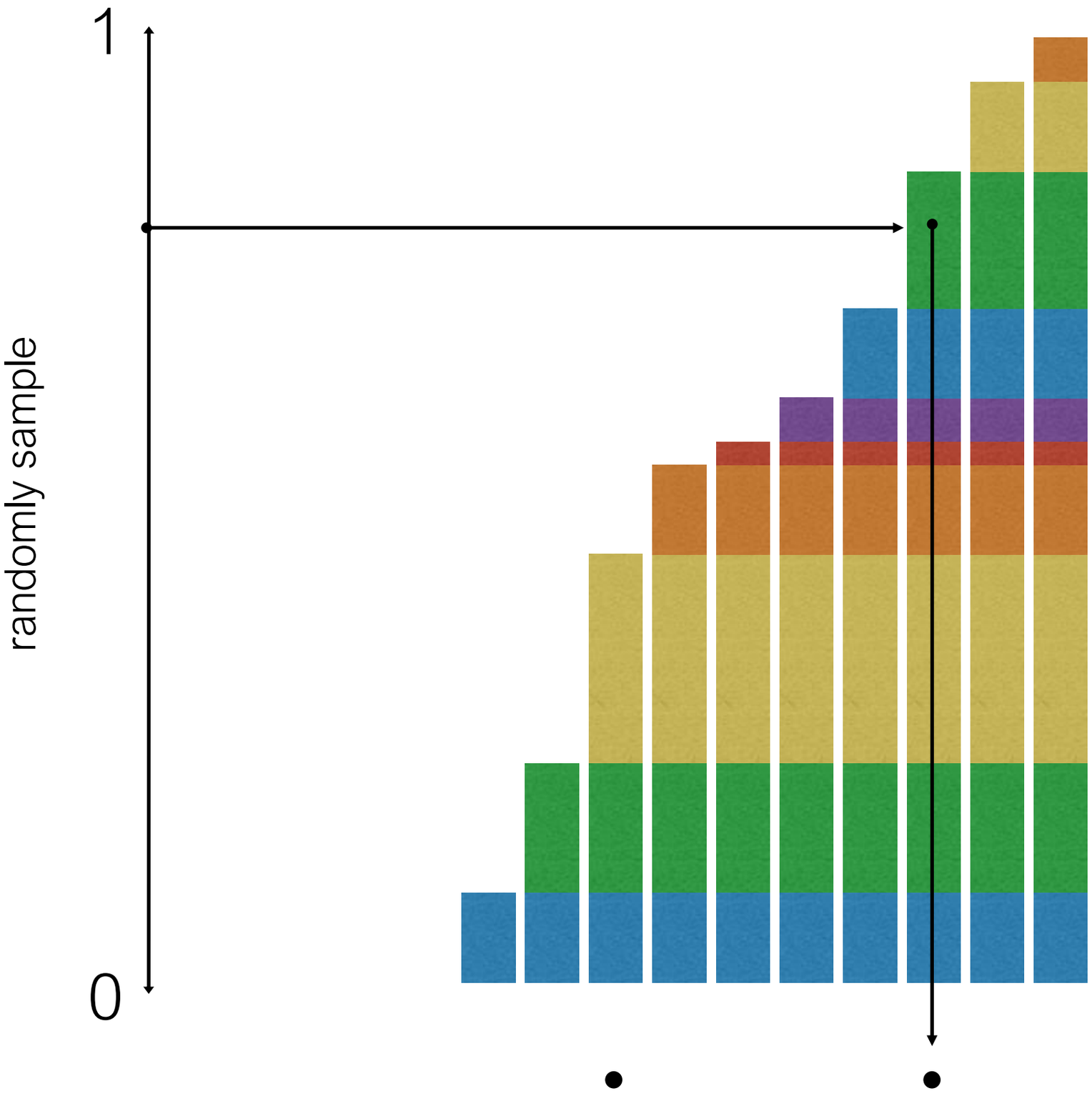

probability density function

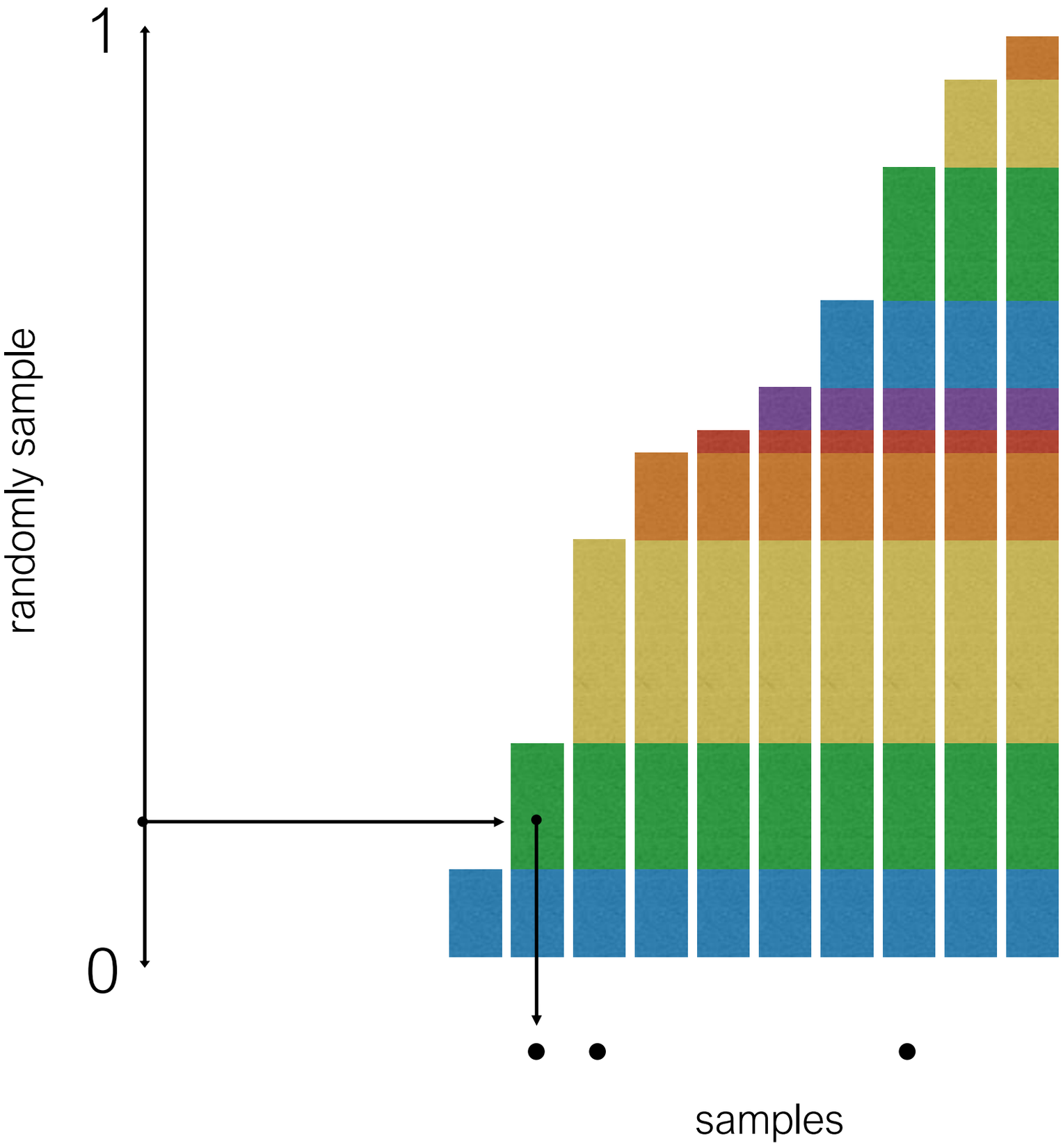


cumulative density function

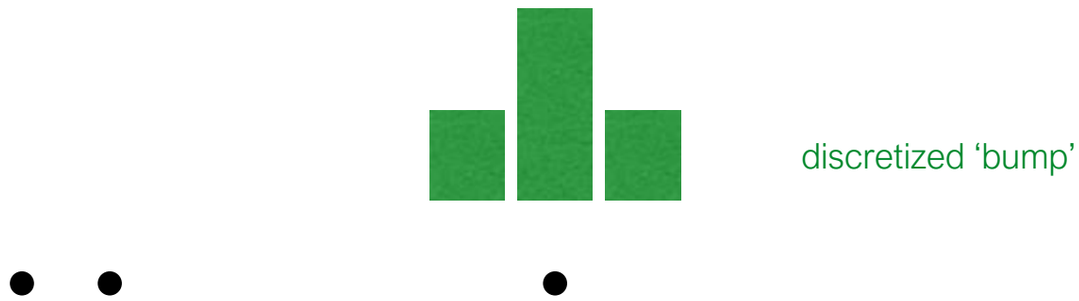
We can draw samples,
using the CDF…



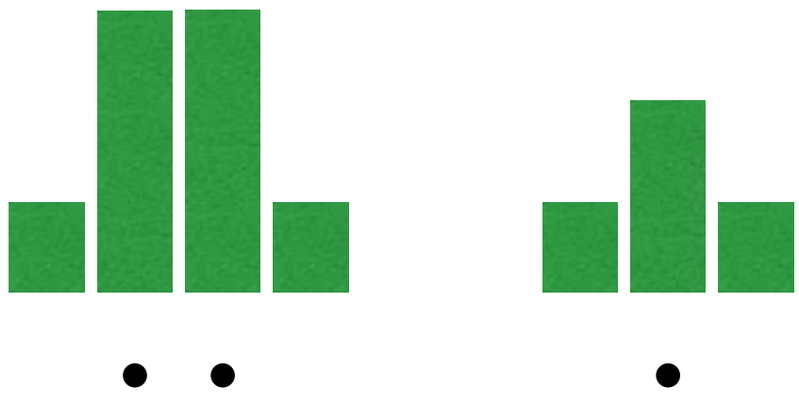

1
randomly sample
0


1
0
random sample
samples

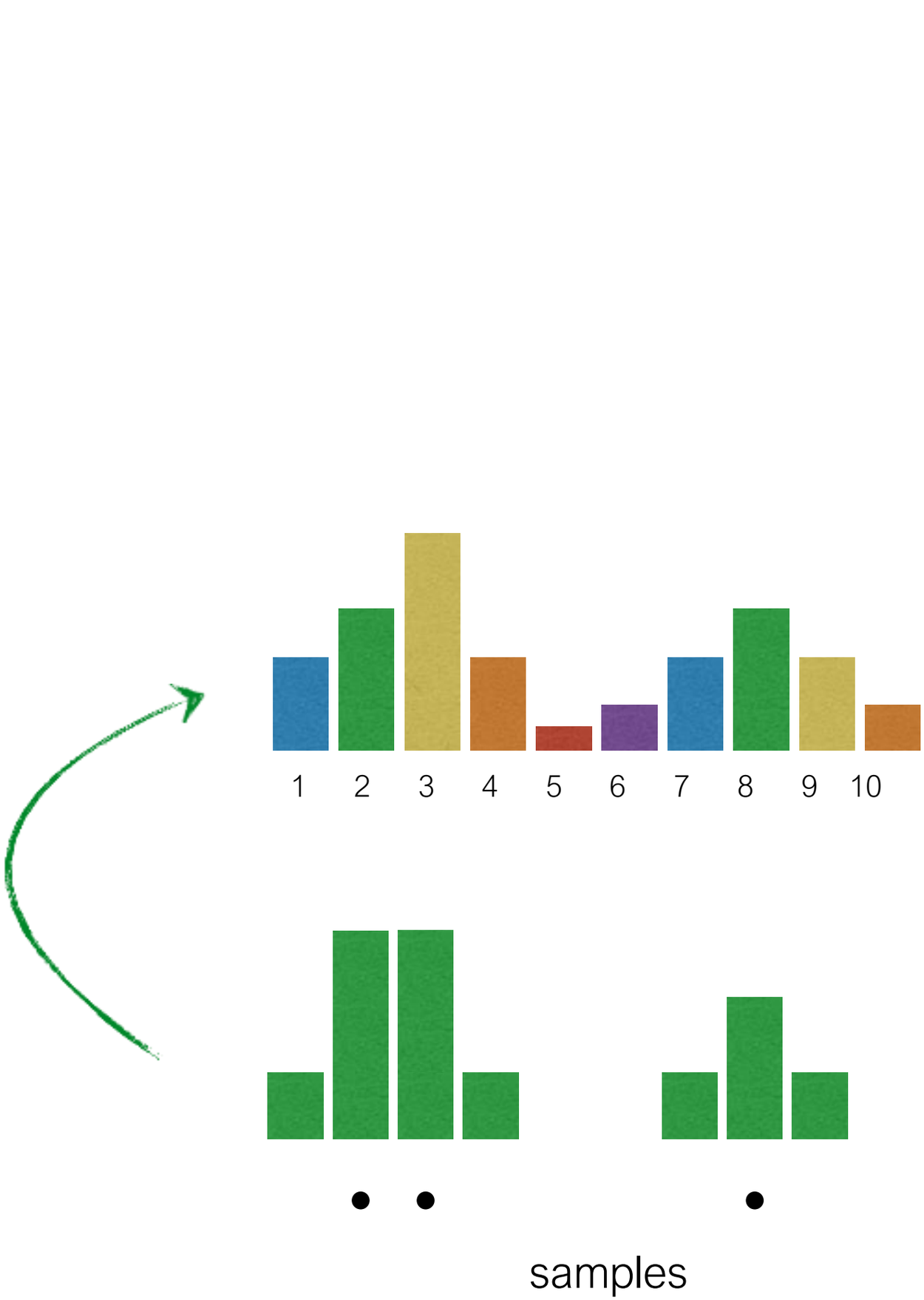
Now to estimate the ‘hidden’ PDF place Gaussian bumps on the samples…


discretized ‘bump’
samples


samples

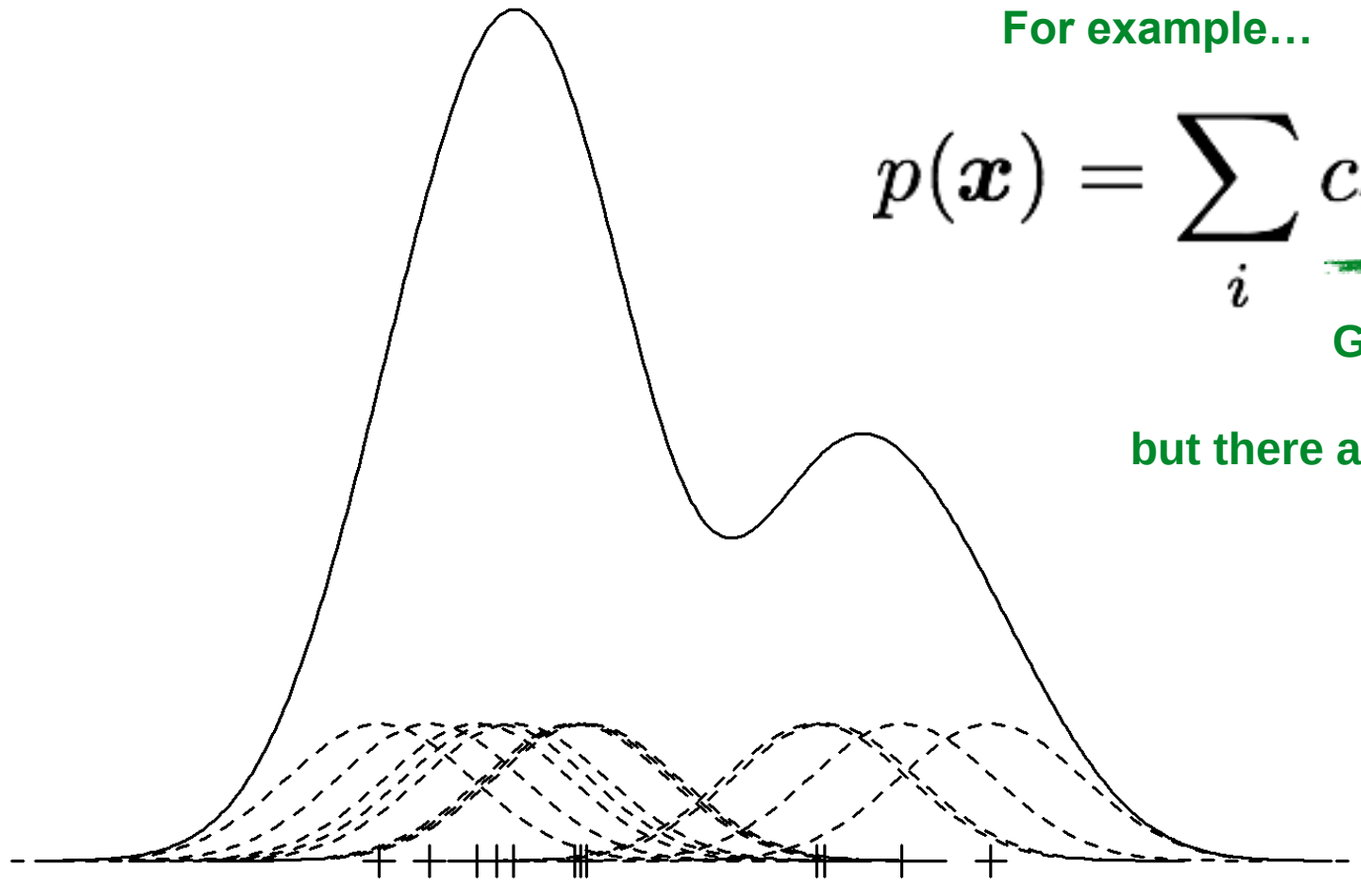

samples

Kernel Density Estimate approximates the original PDF



# Kernel Density Estimation

Approximate the underlying PDF from samples from it



**For example…**

$$p(\boldsymbol{x}) = \sum_i c_i e^{-\frac{(\boldsymbol{x}-\boldsymbol{x}_i)^2}{2\sigma^2}}$$

**Gaussian** ‘bump’ aka ‘kernel’

**but there are many types of kernels!**

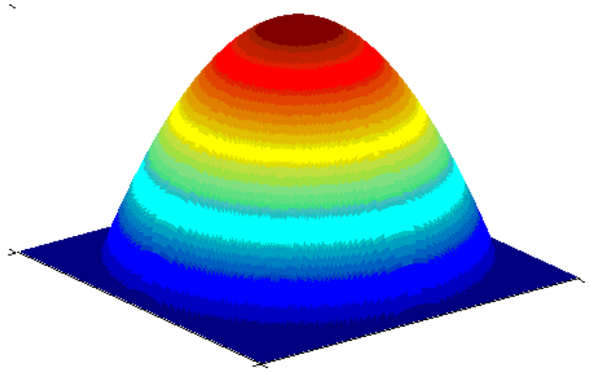
Put ‘bump’ on every sample to approximate the PDF

# Kernel Function

$$K(\boldsymbol{x}, \boldsymbol{x}')$$

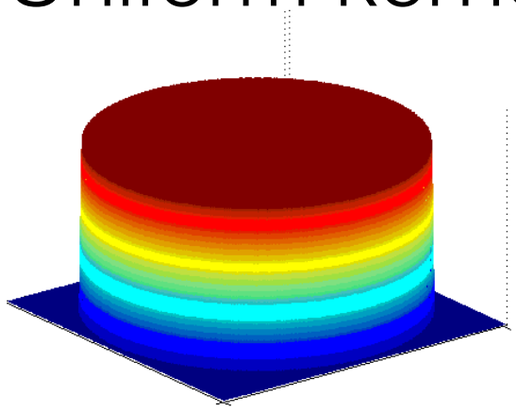
returns the ‘distance’ between two points

## Epanechnikov kernel



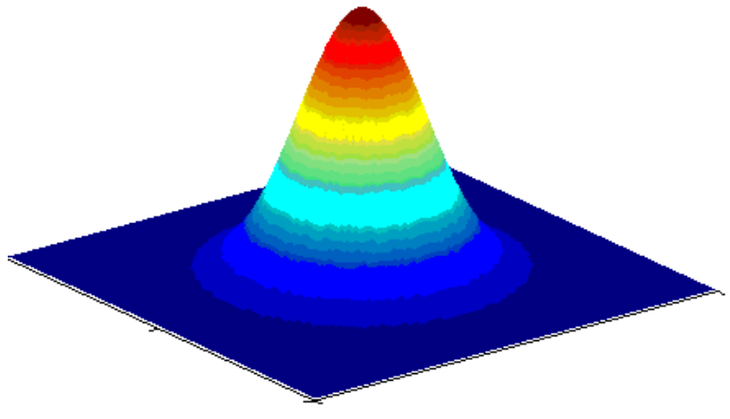
$$K(\boldsymbol{x}, \boldsymbol{x}') = \begin{cases} c(1 - \|\boldsymbol{x} - \boldsymbol{x}'\|^2) & \|\boldsymbol{x} - \boldsymbol{x}'\|^2 \leq 1 \\ 0 & \text{otherwise} \end{cases}$$

## Uniform kernel



$$K(\boldsymbol{x}, \boldsymbol{x}') = \begin{cases} c & \|\boldsymbol{x} - \boldsymbol{x}'\|^2 \leq 1 \\ 0 & \text{otherwise} \end{cases}$$

## Normal kernel



$$K(\boldsymbol{x}, \boldsymbol{x}') = c \exp\left(\frac{1}{2}\|\boldsymbol{x} - \boldsymbol{x}'\|^2\right)$$

These are all radially symmetric kernels

# Radially symmetric kernels

…can be written in terms of its *profile*

$$K(\boldsymbol{x}, \boldsymbol{x}') = c \cdot k(\|\boldsymbol{x} - \boldsymbol{x}'\|^2)$$

profile

# Connecting KDE and the Mean Shift Algorithm

# 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:

$$K(\boldsymbol{x}, \boldsymbol{x}')$$

Find the mean sample point:

$$\boldsymbol{x}$$

# Mean-Shift Algorithm

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

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

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

compute the ‘shift’

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

*Where does this algorithm come from?*

# Mean-Shift Algorithm

Initialize $\boldsymbol{x}$

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

1. Compute mean-shift

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

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

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

*Where does this come from?*

*Where does this algorithm come from?*

# *How is the KDE related to the mean shift algorithm?*

**Recall:**

Kernel density estimate

(radially symmetric kernels)

$$P(\boldsymbol{x}) = \frac{1}{N} c \sum_{n} k(\|\boldsymbol{x} - \boldsymbol{x}_n\|^2)$$

can compute probability for any point using the KDE!

**We can show that:**

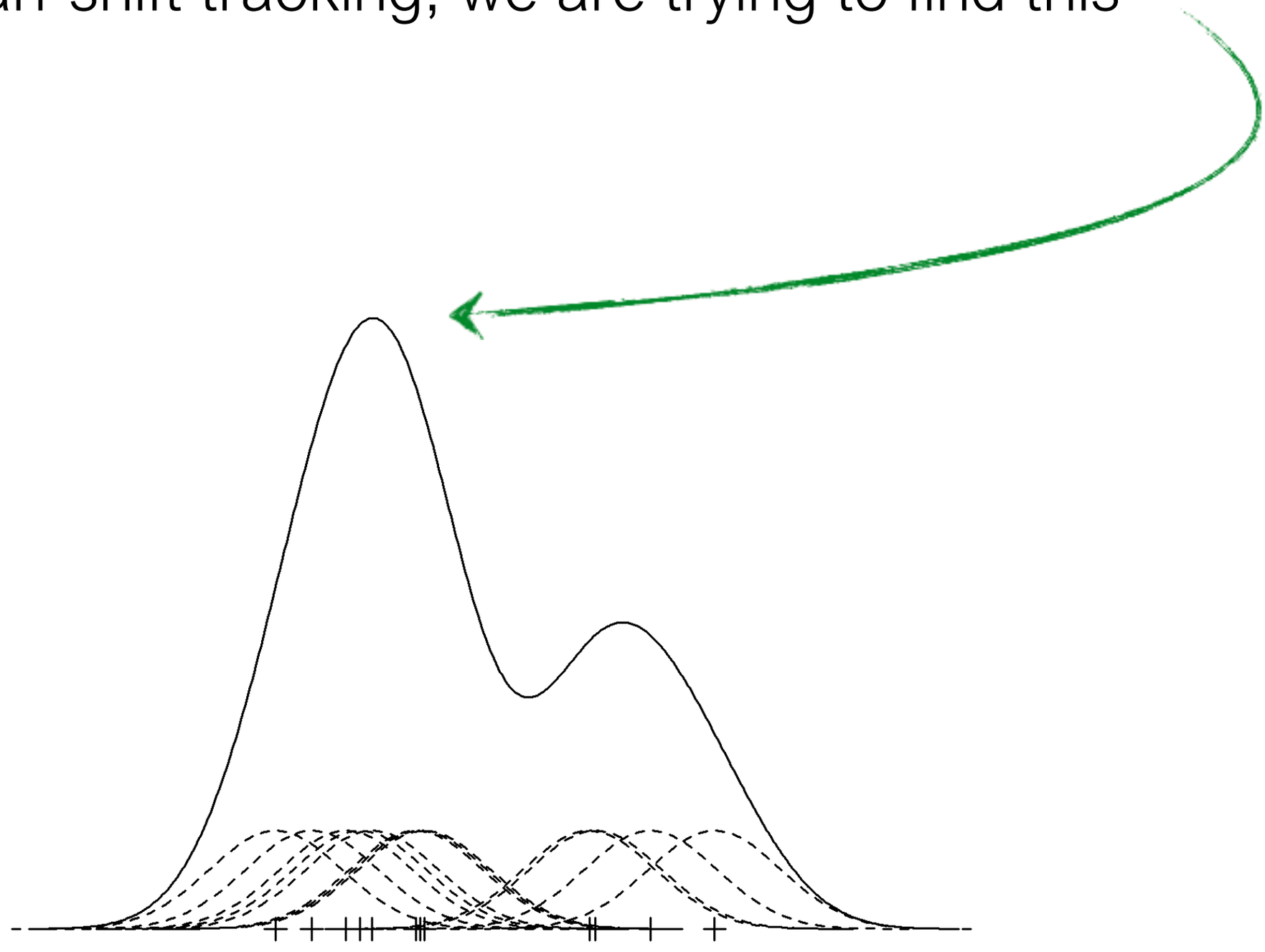
Gradient of the PDF is related to the mean shift vector

$$\nabla P(\boldsymbol{x}) \propto m(\boldsymbol{x})$$

The mean shift vector is a ‘step’ in the direction of the gradient of the KDE

mean-shift algorithm is maximizing the objective function

In mean-shift tracking, we are trying to find this



which  means we are trying to…

We are trying to optimize this:

$$\boldsymbol{x} = \arg\max_{\boldsymbol{x}} P(\boldsymbol{x})$$

find the solution that has the highest probability

$$= \arg\max_{\boldsymbol{x}} \frac{1}{N} c \sum_{n} k(||\boldsymbol{x} - \boldsymbol{x}_n||^2)$$

usually non-linear

non-parametric

*How do we optimize this non-linear function?*

We are trying to optimize this:

$$\boldsymbol{x} = \arg\max_{\boldsymbol{x}} P(\boldsymbol{x})$$

$$= \arg\max_{\boldsymbol{x}} \frac{1}{N} c \sum_{n} k(||\boldsymbol{x} - \boldsymbol{x}_n||^2)$$

usually non-linear

non-parametric

*How do we optimize this non-linear function?*

compute partial derivatives … **gradient descent!**

$$P(\boldsymbol{x}) = \frac{1}{N} c \sum_{n} k(\|\boldsymbol{x} - \boldsymbol{x}_n\|^2)$$

Compute the gradient

$$P(\boldsymbol{x}) = \frac{1}{N} c \sum_n k(\|\boldsymbol{x} - \boldsymbol{x}_n\|^2)$$

Gradient

$$\nabla P(\boldsymbol{x}) = \frac{1}{N} c \sum_n \nabla k(\|\boldsymbol{x} - \boldsymbol{x}_n\|^2)$$

Expand the gradient (algebra)

$$P(\boldsymbol{x}) = \frac{1}{N} c \sum_n k(\|\boldsymbol{x} - \boldsymbol{x}_n\|^2)$$

Gradient

$$\nabla P(\boldsymbol{x}) = \frac{1}{N} c \sum_n \nabla k(\|\boldsymbol{x} - \boldsymbol{x}_n\|^2)$$

Expand gradient

$$\nabla P(\boldsymbol{x}) = \frac{1}{N} 2c \sum_n (\boldsymbol{x} - \boldsymbol{x}_n) k'(\|\boldsymbol{x} - \boldsymbol{x}_n\|^2)$$

$$P(\boldsymbol{x}) = \frac{1}{N} c \sum_n k(\|\boldsymbol{x} - \boldsymbol{x}_n\|^2)$$

Gradient

$$\nabla P(\boldsymbol{x}) = \frac{1}{N} c \sum_n \nabla k(\|\boldsymbol{x} - \boldsymbol{x}_n\|^2)$$

Expand gradient

$$\nabla P(\boldsymbol{x}) = \frac{1}{N} 2c \sum_n (\boldsymbol{x} - \boldsymbol{x}_n) k'(\|\boldsymbol{x} - \boldsymbol{x}_n\|^2)$$

Call the gradient of the kernel function g

$$k'(\cdot) = -g(\cdot)$$

$$P(\boldsymbol{x}) = \frac{1}{N} c \sum_n k(\|\boldsymbol{x} - \boldsymbol{x}_n\|^2)$$

Gradient

$$\nabla P(\boldsymbol{x}) = \frac{1}{N} c \sum_n \nabla k(\|\boldsymbol{x} - \boldsymbol{x}_n\|^2)$$

Expand gradient

$$\nabla P(\boldsymbol{x}) = \frac{1}{N} 2c \sum_n (\boldsymbol{x} - \boldsymbol{x}_n) k'(\|\boldsymbol{x} - \boldsymbol{x}_n\|^2)$$

change of notation
(kernel-shadow pairs)

$$\nabla P(\boldsymbol{x}) = \frac{1}{N} 2c \sum_n (\boldsymbol{x}_n - \boldsymbol{x}) g(\|\boldsymbol{x} - \boldsymbol{x}_n\|^2)$$

keep this in memory: $k'(\cdot) = -g(\cdot)$

$$\nabla P(\boldsymbol{x}) = \frac{1}{N} 2c \sum_n (\boldsymbol{x}_n - \boldsymbol{x}) g(\|\boldsymbol{x} - \boldsymbol{x}_n\|^2)$$

multiply it out

$$\nabla P(\boldsymbol{x}) = \frac{1}{N} 2c \sum \boldsymbol{x}_n g(\|\boldsymbol{x} - \boldsymbol{x}_n\|^2) - \frac{1}{N} 2c \sum \boldsymbol{x} g(\|\boldsymbol{x} - \boldsymbol{x}_n\|^2)$$

too long!
(use short hand notation)

$$\nabla P(\boldsymbol{x}) = \frac{1}{N} 2c \sum_n \boldsymbol{x}_n g_n - \frac{1}{N} 2c \sum_n \boldsymbol{x} g_n$$

$$\nabla P(\boldsymbol{x}) = \frac{1}{N} 2c \sum_n \boldsymbol{x}_n g_n - \frac{1}{N} 2c \sum_n \boldsymbol{x} g_n$$

multiply by one!

$$\nabla P(\boldsymbol{x}) = \frac{1}{N} 2c \sum_n \boldsymbol{x}_n g_n \left( \frac{\sum_n g_n}{\sum_n g_n} \right) - \frac{1}{N} 2c \sum_n \boldsymbol{x} g_n$$

collecting like terms…

$$\nabla P(\boldsymbol{x}) = \frac{1}{N} 2c \sum_n g_n \left( \frac{\sum_n \boldsymbol{x}_n g_n}{\sum_n g_n} - \boldsymbol{x} \right)$$

*What's happening here?*

mean

shift

$$\nabla P(\boldsymbol{x}) = \underbrace{\frac{1}{N} 2c \sum_n g_n}_{\text{constant}} \underbrace{\left( \frac{\sum_n \boldsymbol{x}_n g_n}{\sum_n g_n} - \boldsymbol{x} \right)}_{\text{mean shift!}}$$

The **mean shift** is a ‘step’ in the direction of the gradient of the KDE

Let $$\boldsymbol{v}(\boldsymbol{x}) = \left( \frac{\sum_n \boldsymbol{x}_n g_n}{\sum_n g_n} - \boldsymbol{x} \right) = \frac{\nabla P(\boldsymbol{x})}{\frac{1}{N} 2c \sum_n g_n}$$

Can interpret this to be gradient ascent with data dependent step size

# Mean-Shift Algorithm

Initialize $\boldsymbol{x}$

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

1. Compute mean-shift

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

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

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

gradient with adaptive step size

$$\frac{\nabla P(\boldsymbol{x})}{\frac{1}{N} 2c \sum_n g_n}$$

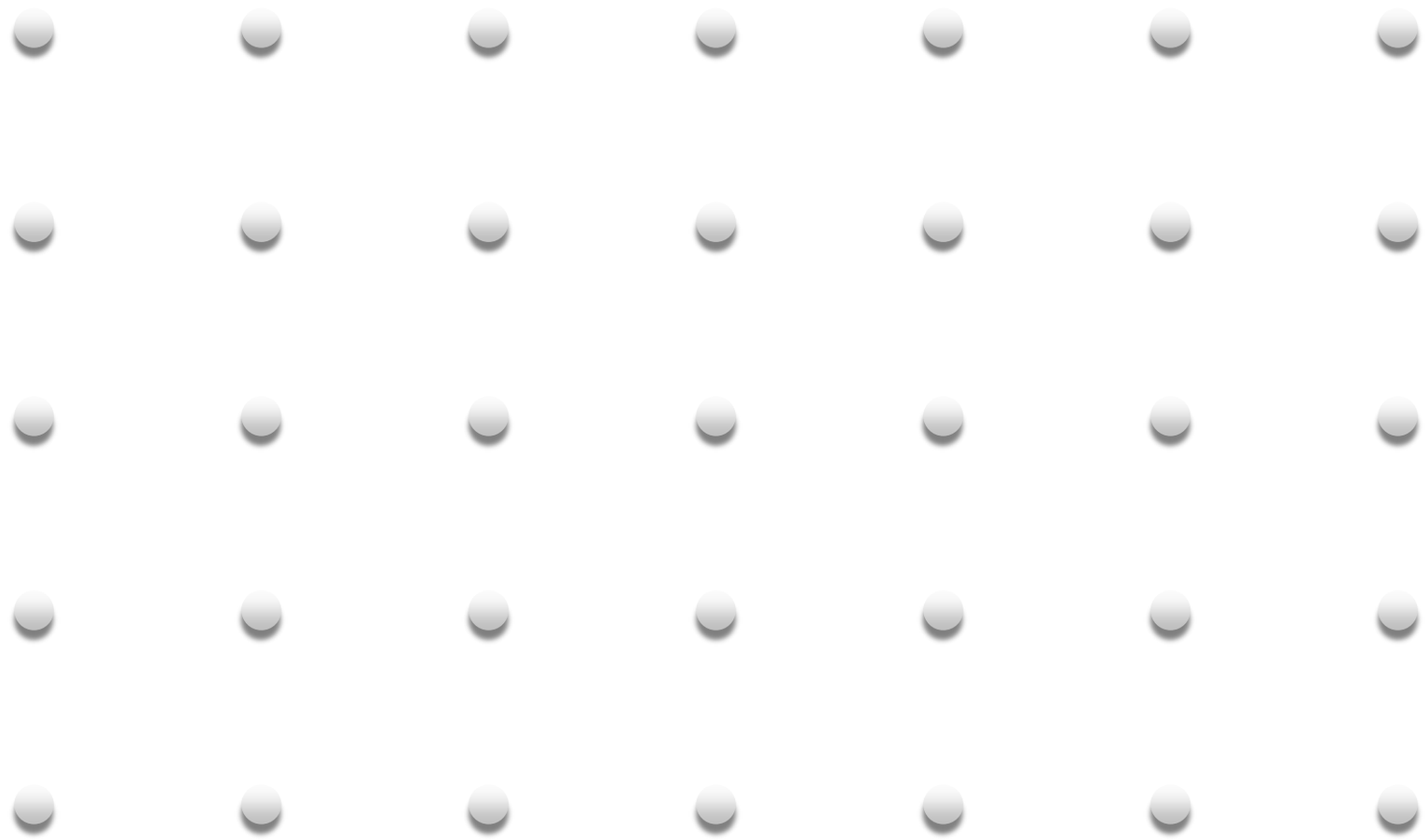
Just 5 lines of code!

Everything up to now has been about distributions over samples…

# Mean-shift tracker

# Dealing with images

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



*What can we do?*

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

same

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

# Mean-Shift Algorithm

(for images)

Initialize $\boldsymbol{x}$

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

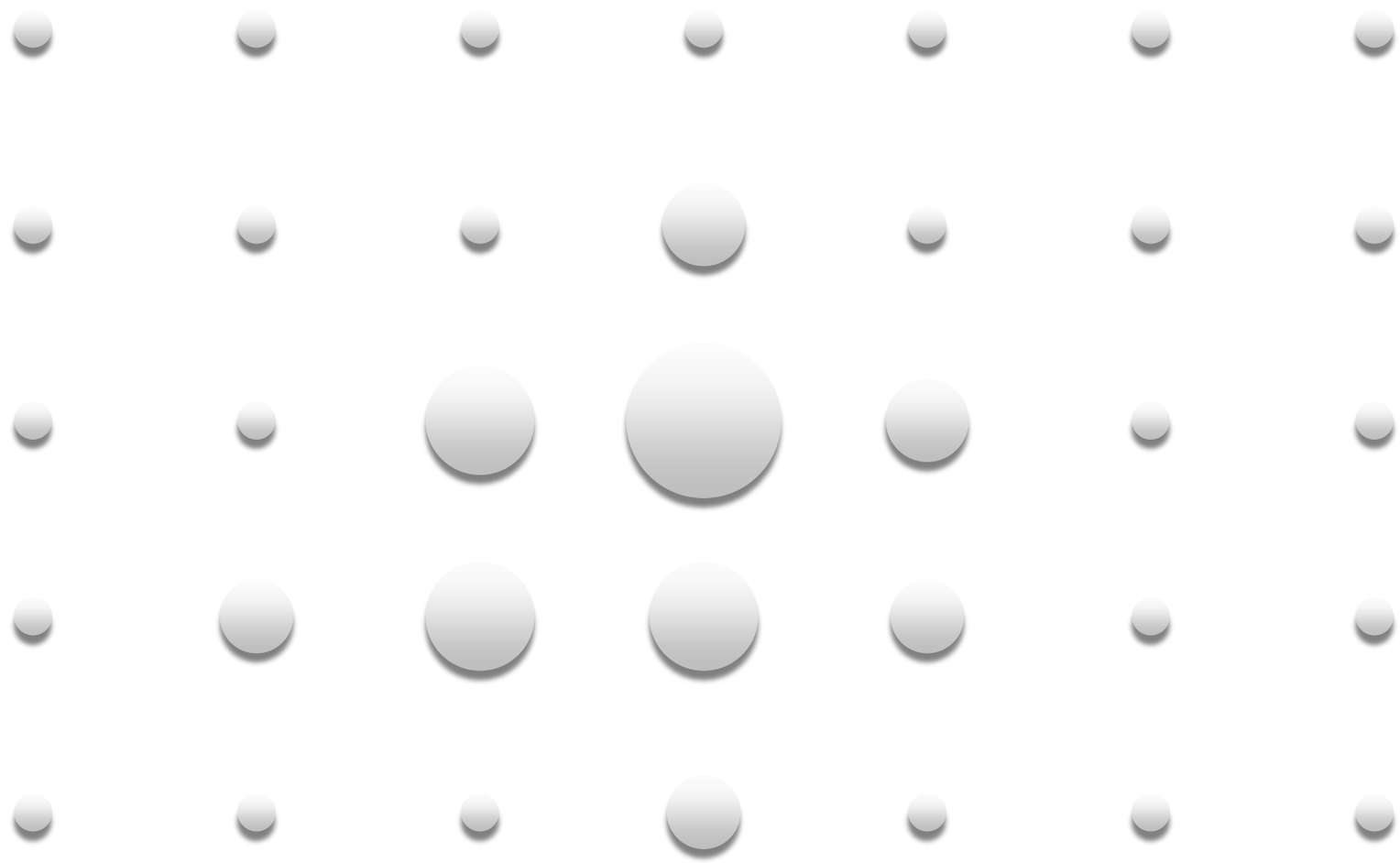
1. Compute mean-shift

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

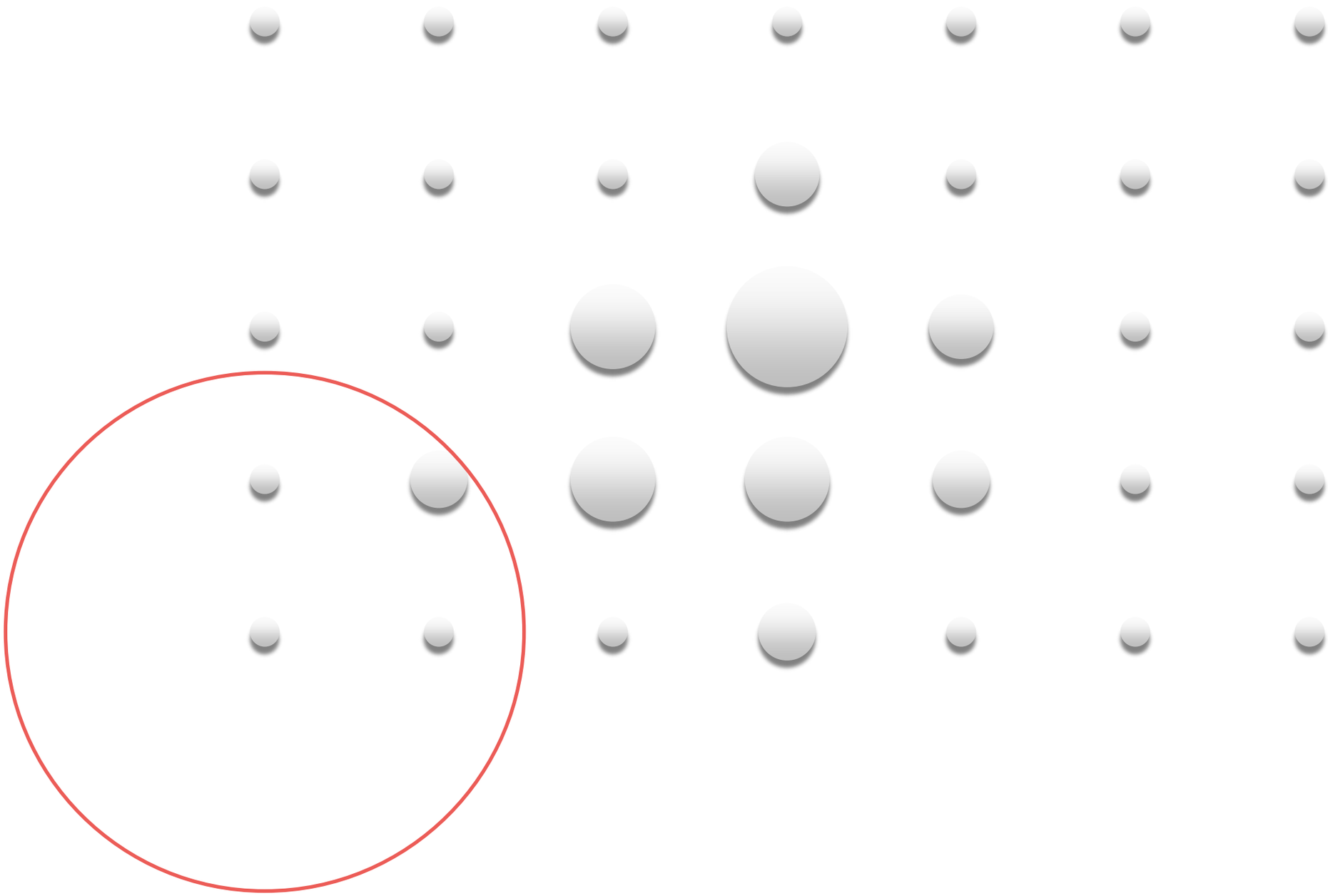
$$v(\boldsymbol{x}) = m(\boldsymbol{x}) - \boldsymbol{x}$$

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

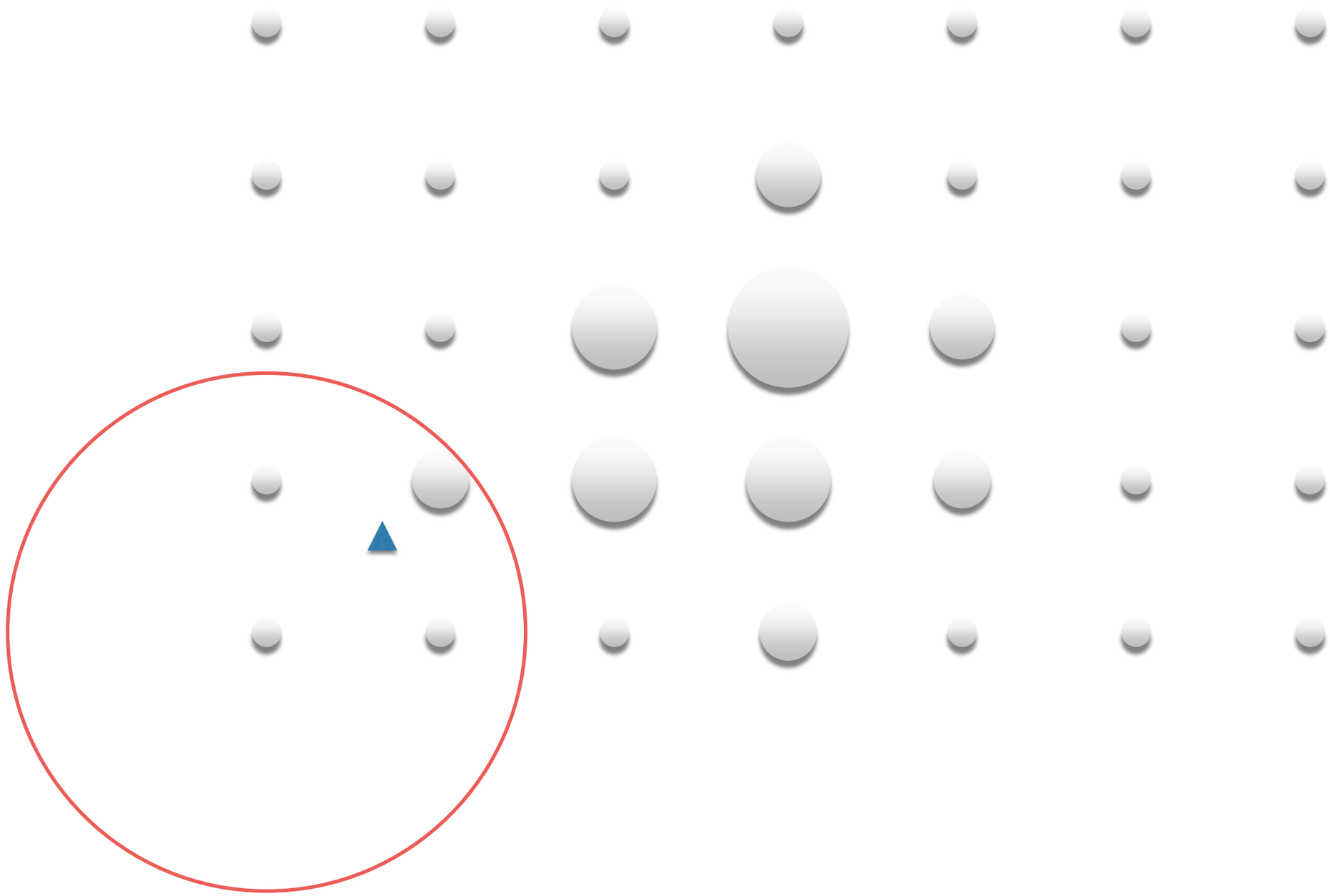
For images, each pixel is point with a weight



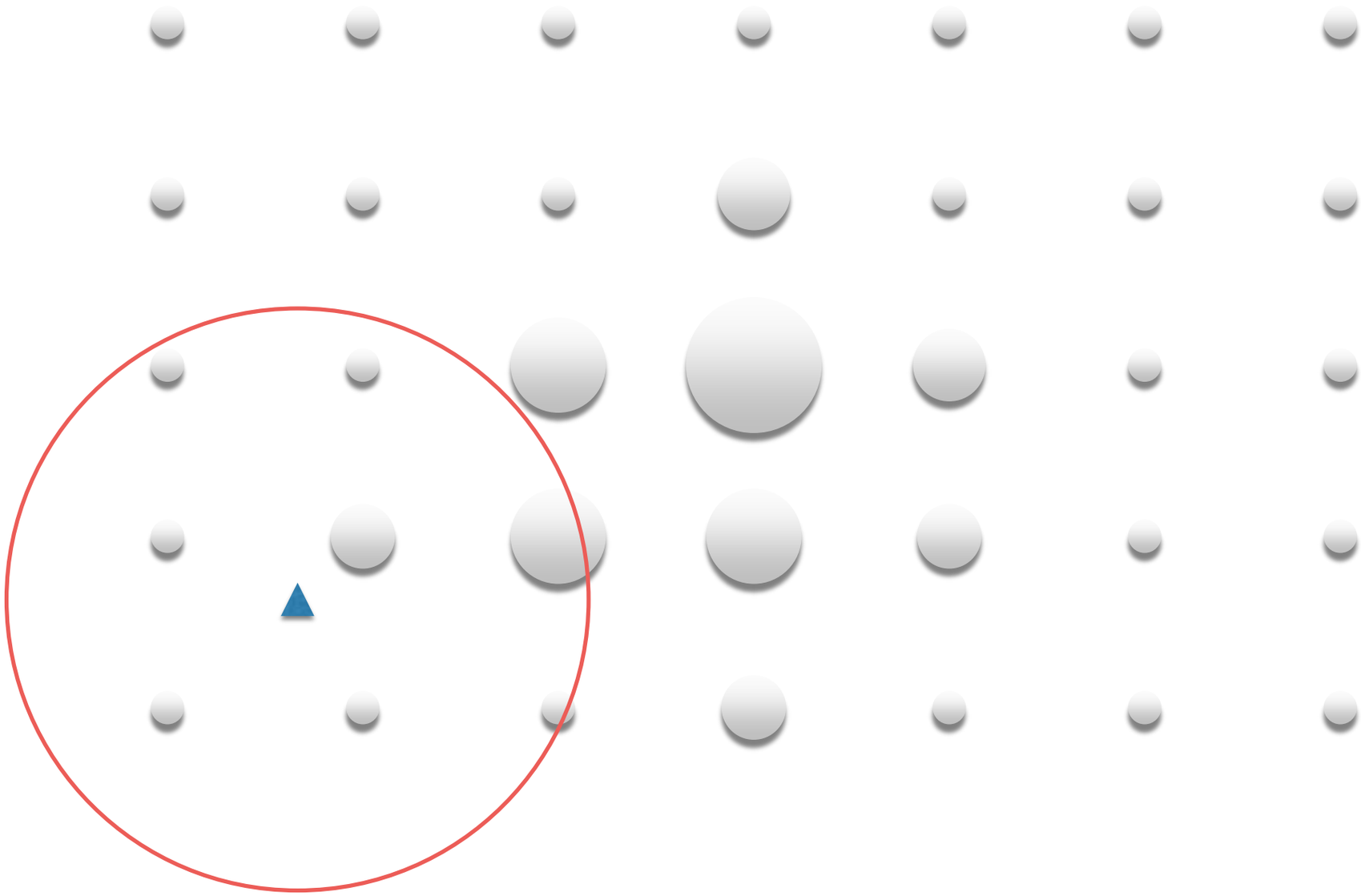
For images, each pixel is point with a weight



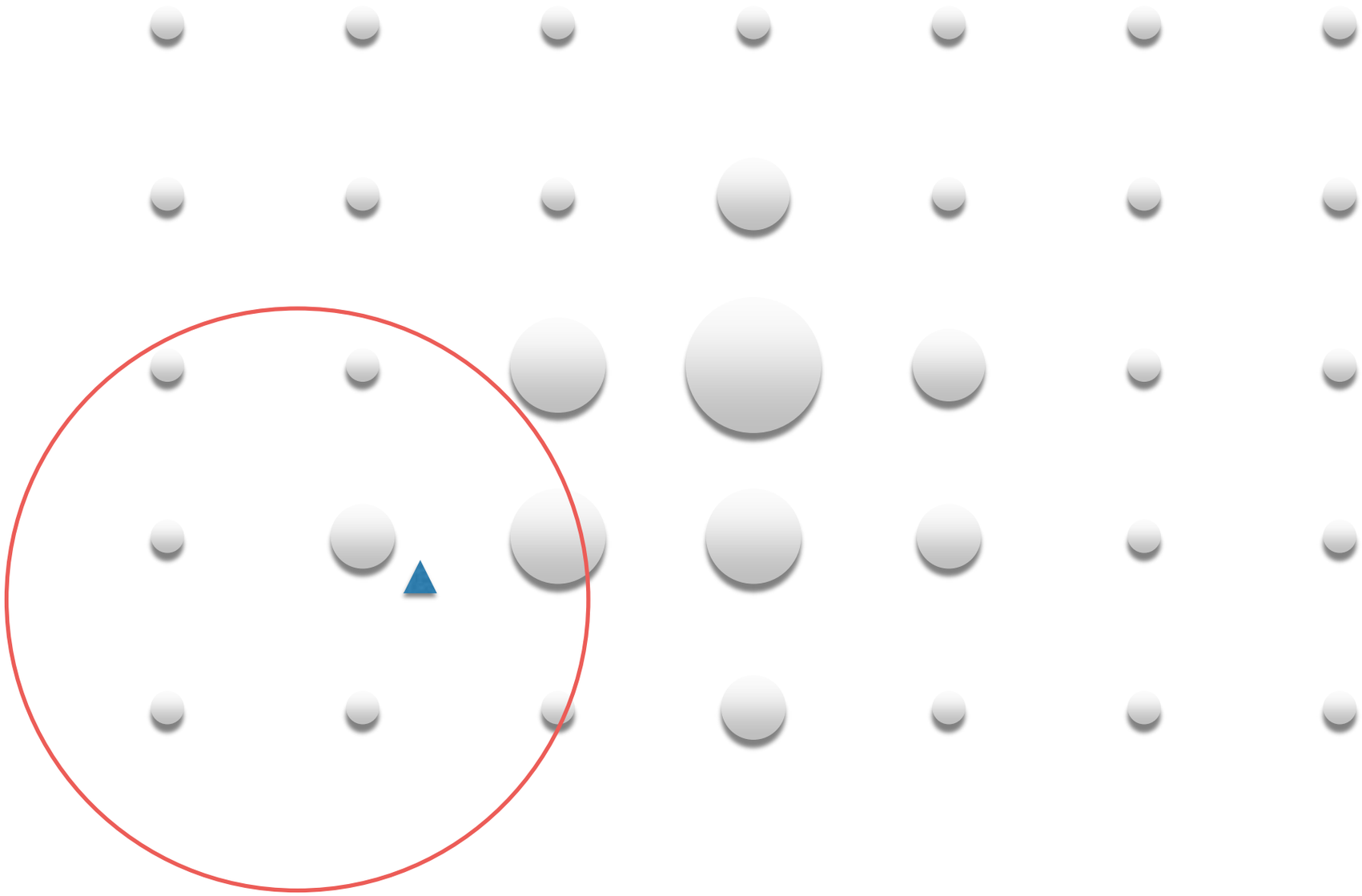
For images, each pixel is point with a weight



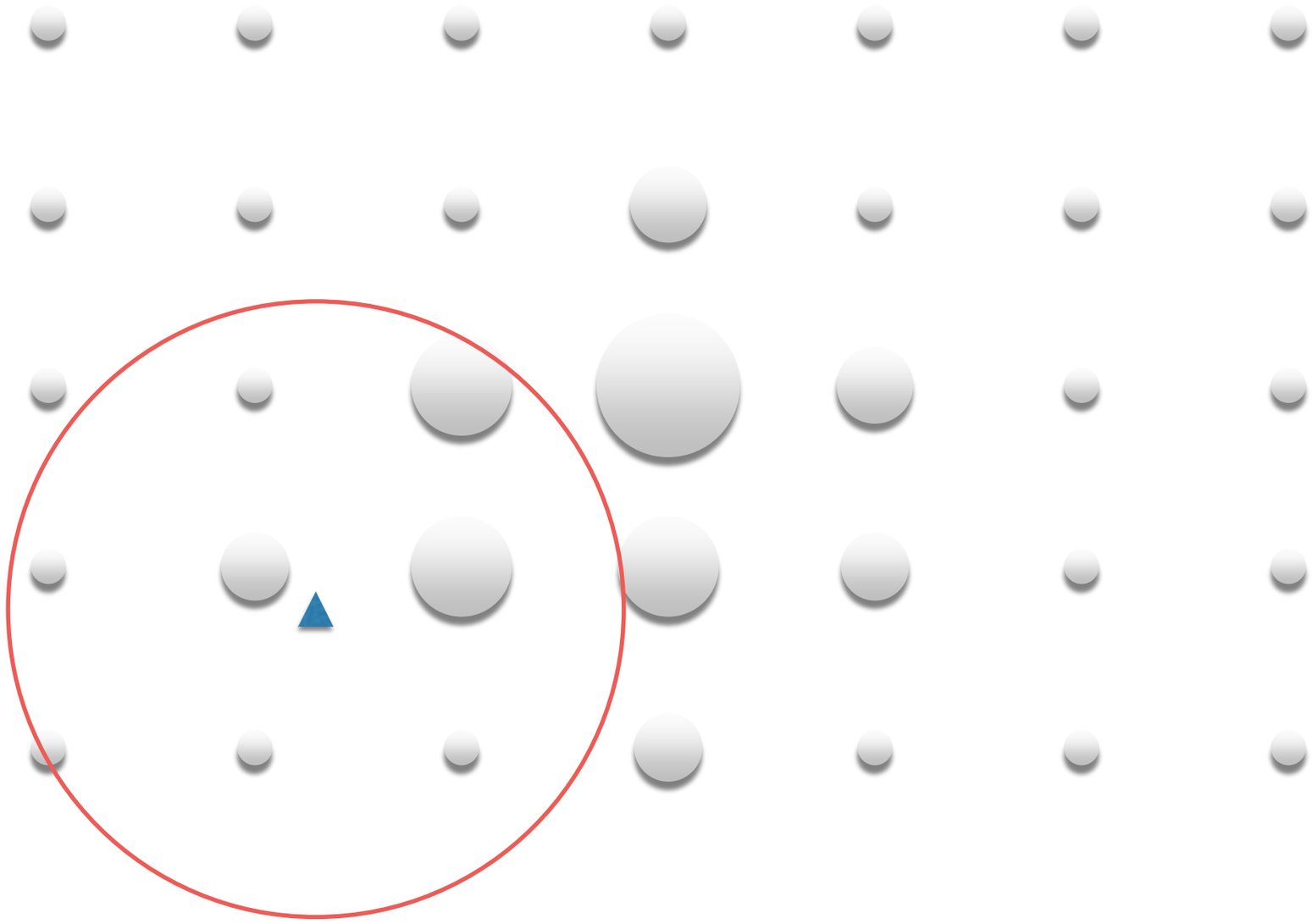
For images, each pixel is point with a weight



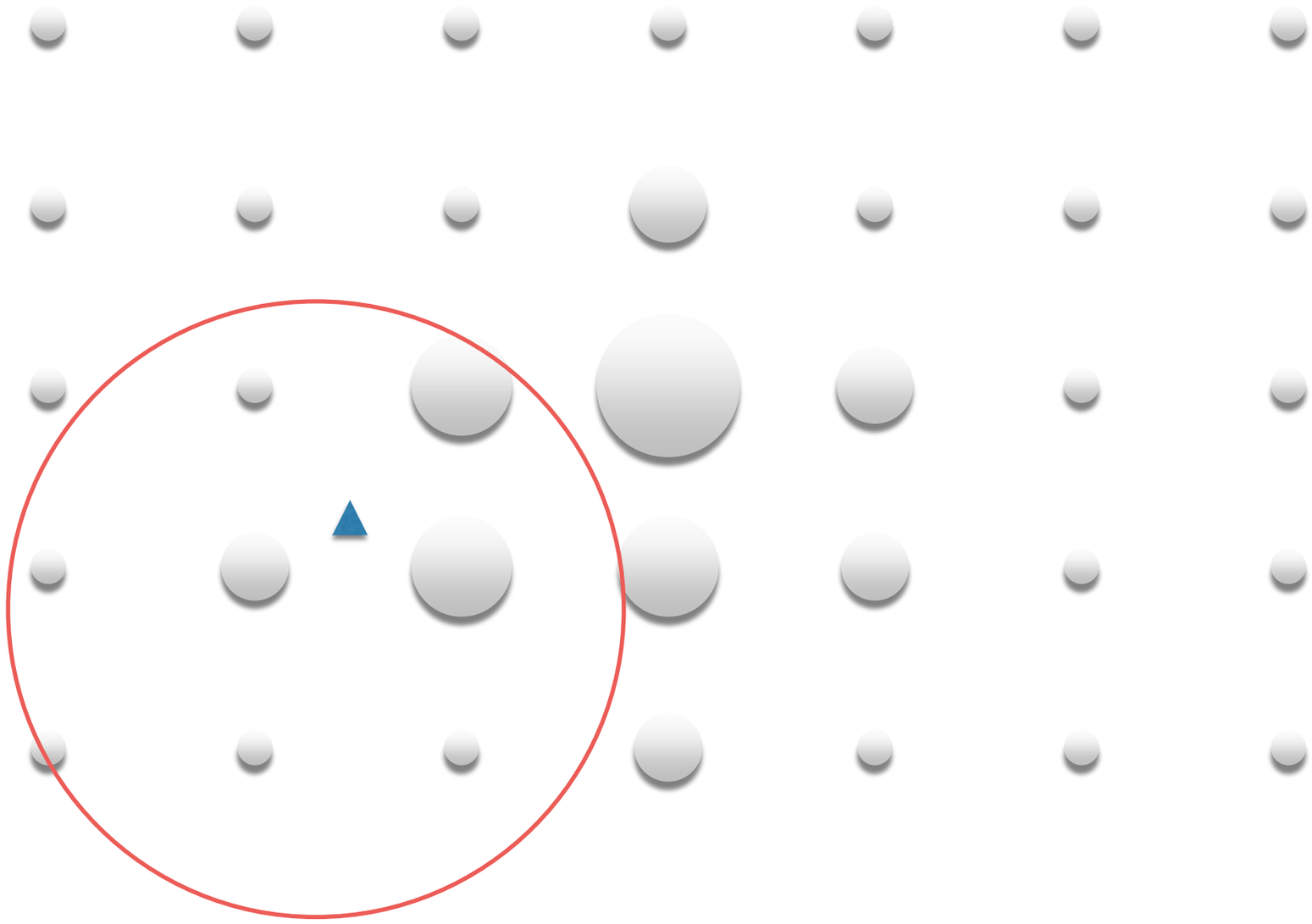
For images, each pixel is point with a weight



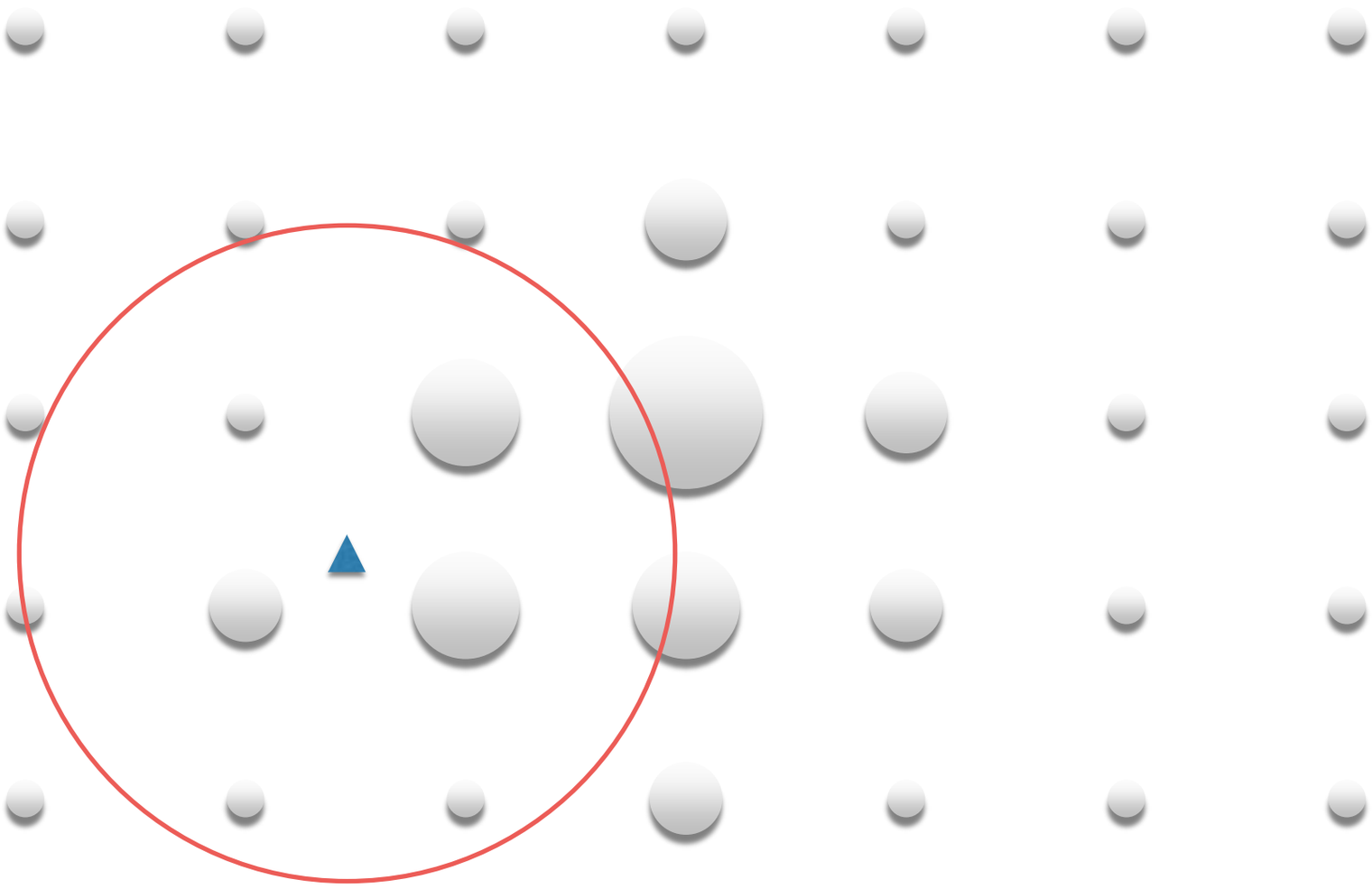
For images, each pixel is point with a weight



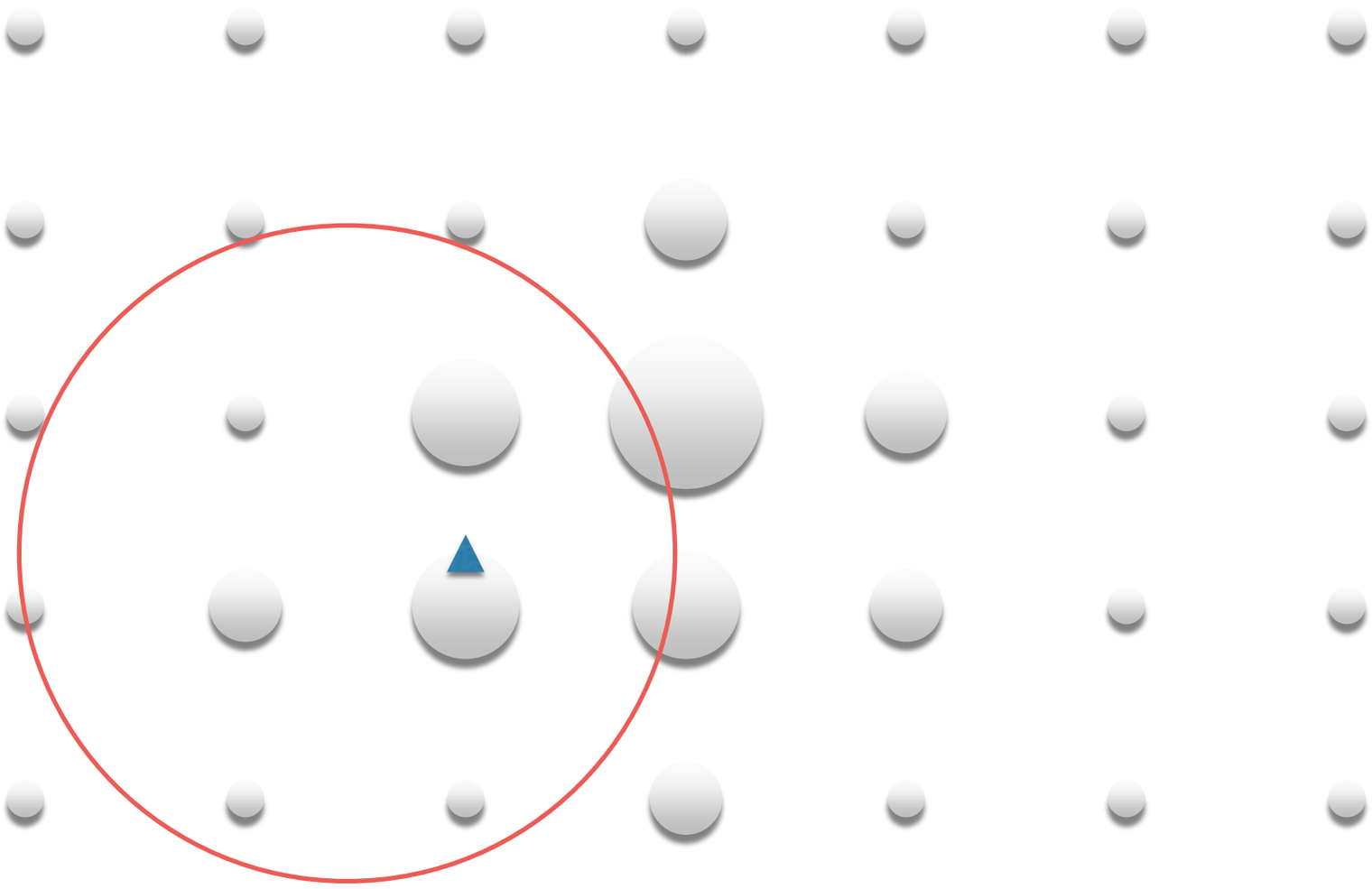
For images, each pixel is point with a weight



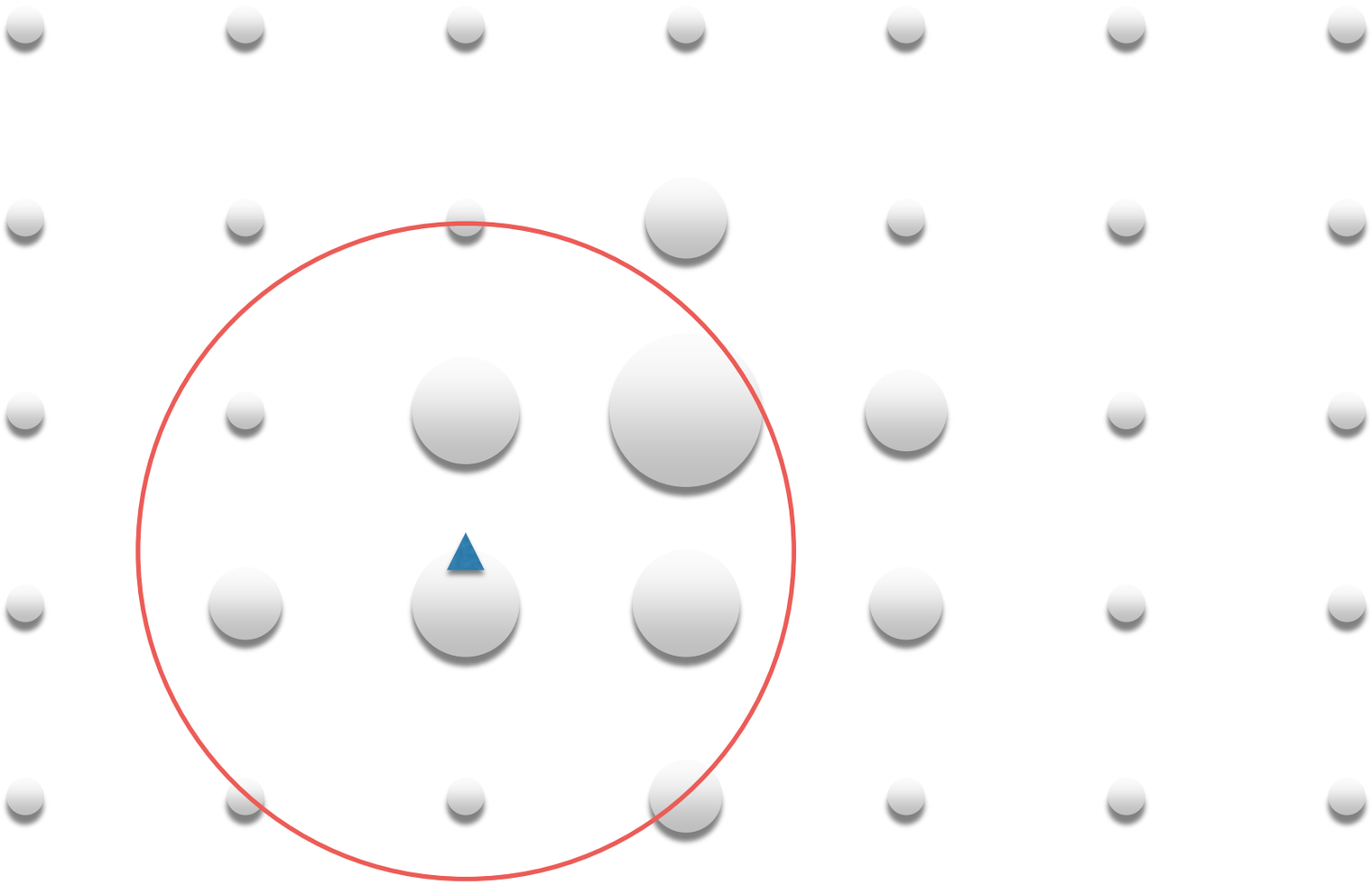
For images, each pixel is point with a weight



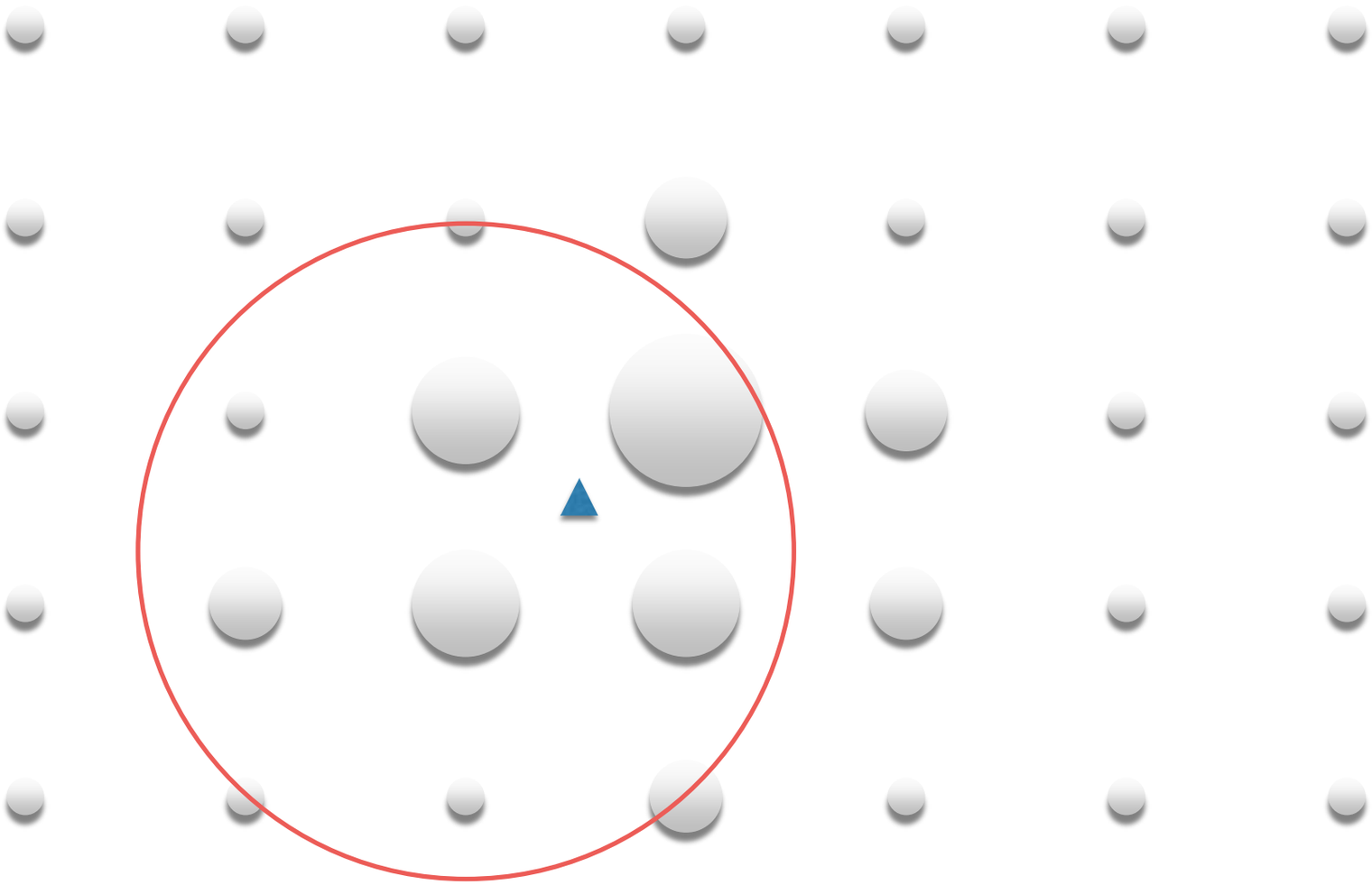
For images, each pixel is point with a weight



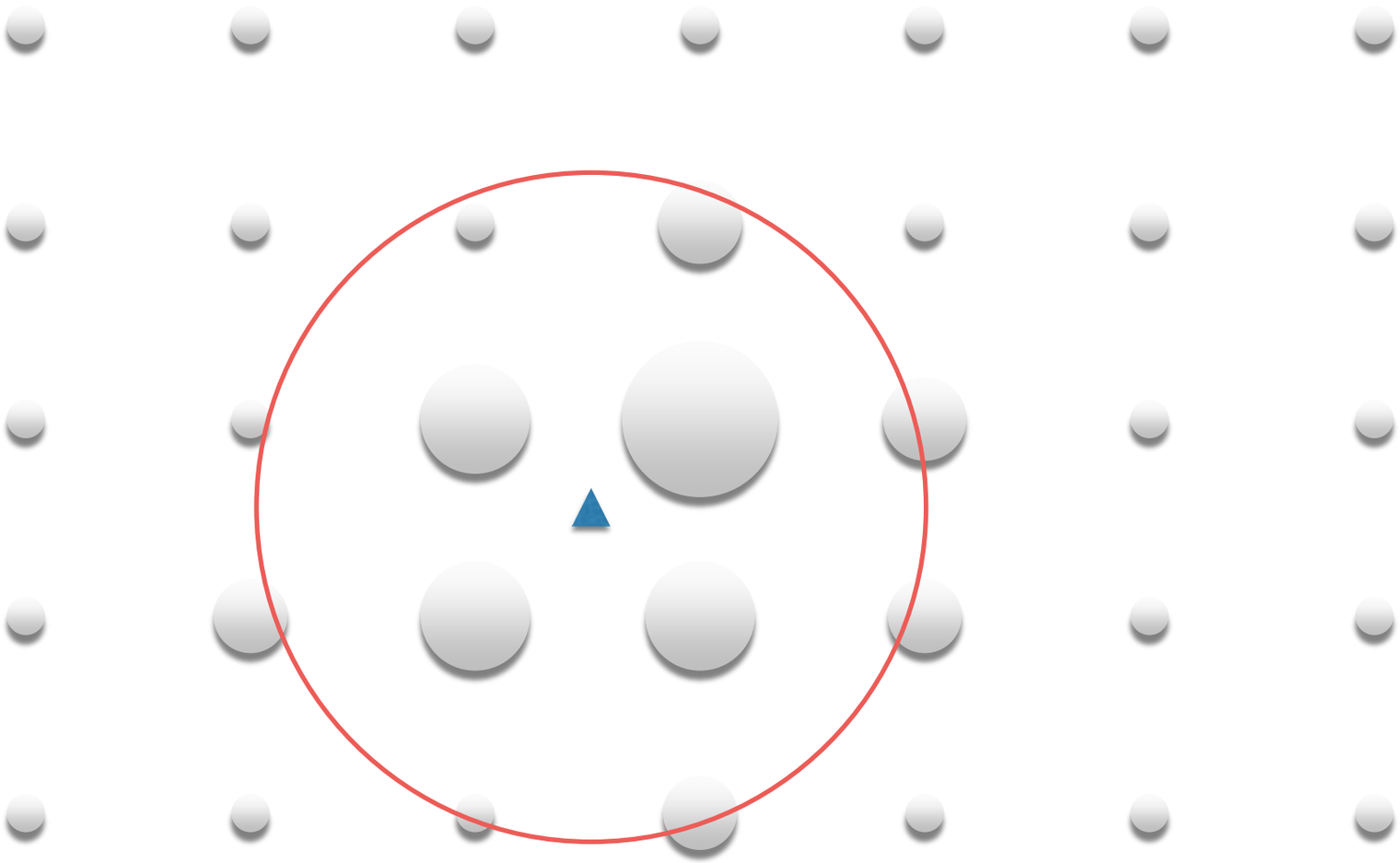
For images, each pixel is point with a weight



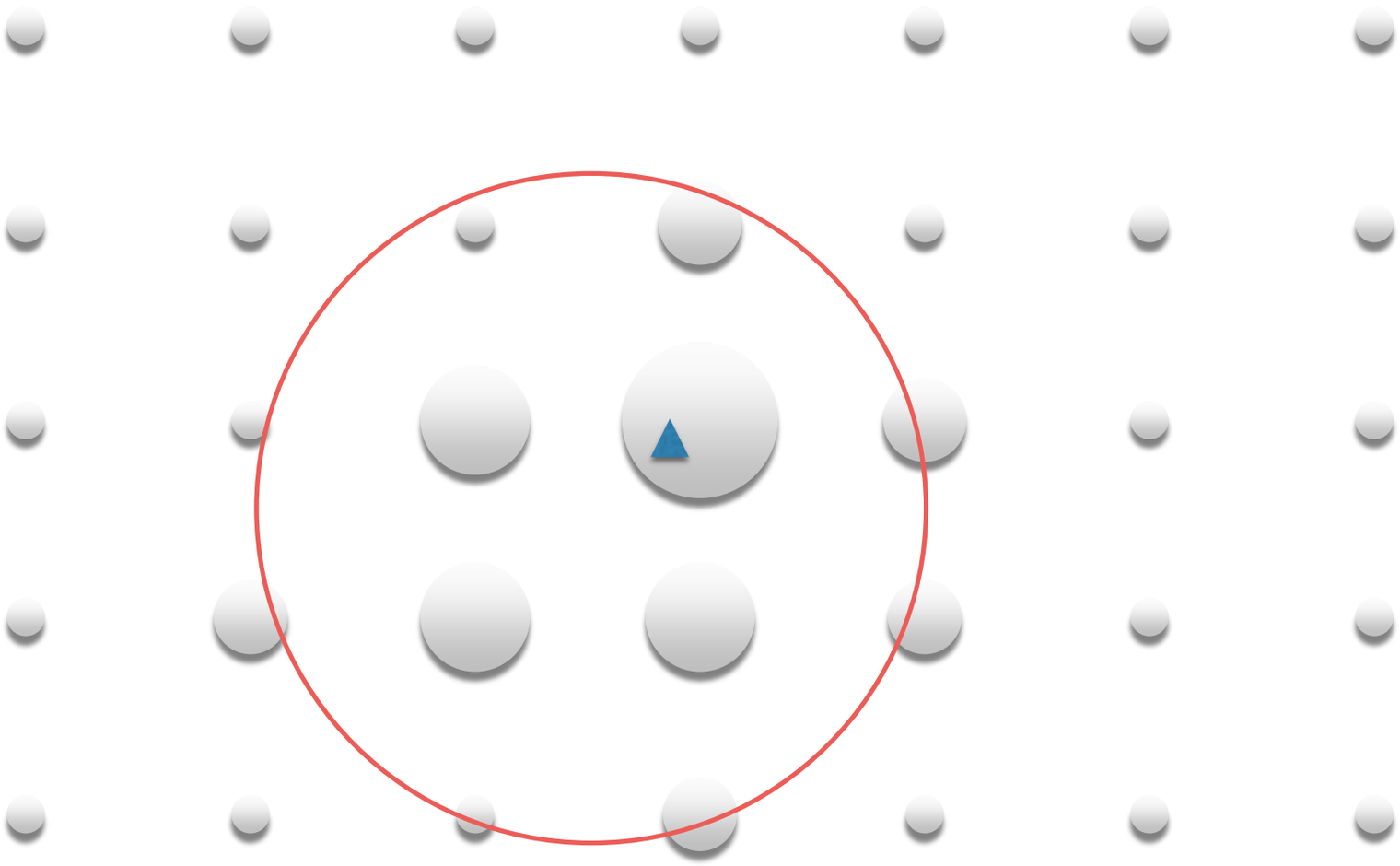
For images, each pixel is point with a weight



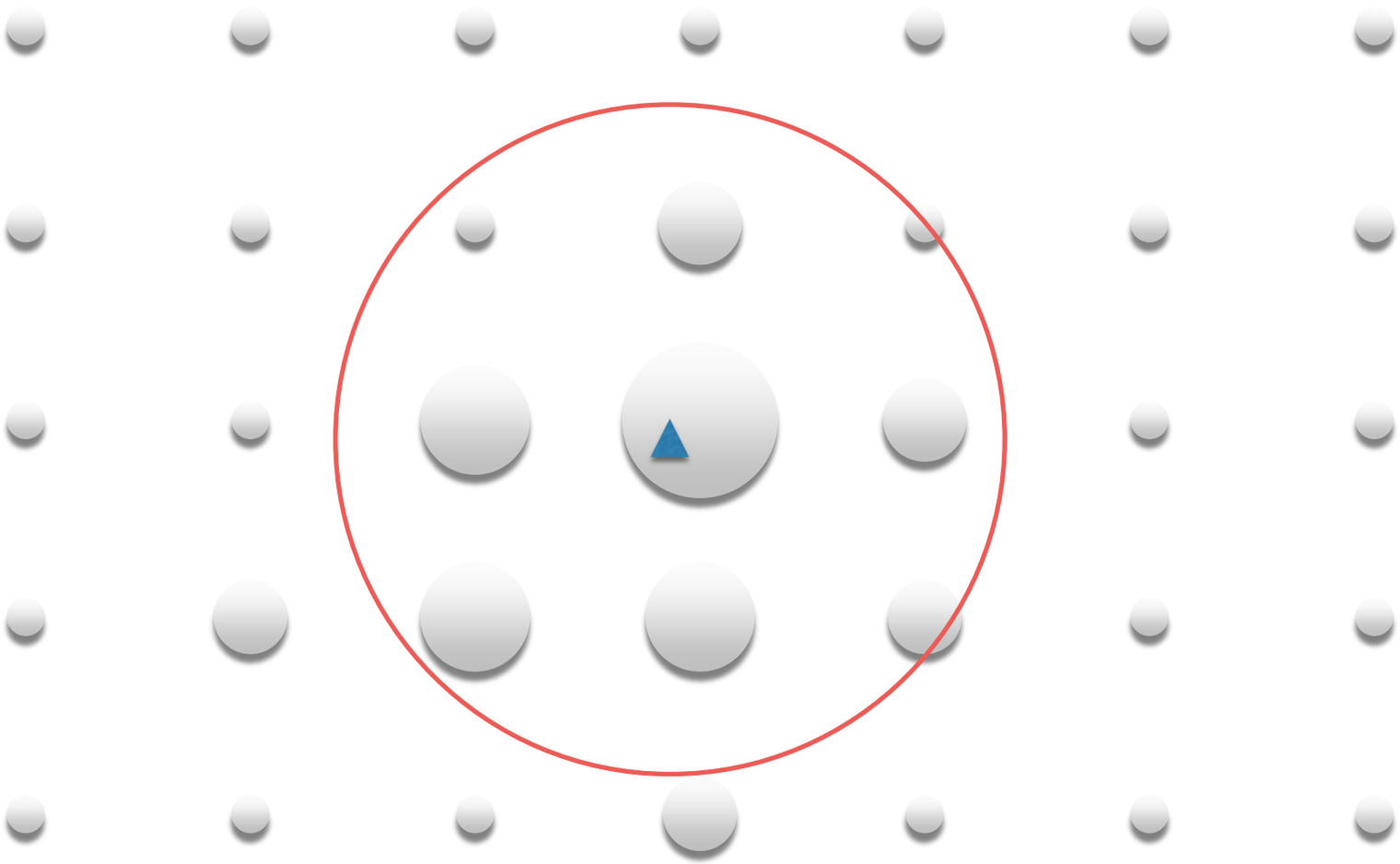
For images, each pixel is point with a weight



For images, each pixel is point with a weight

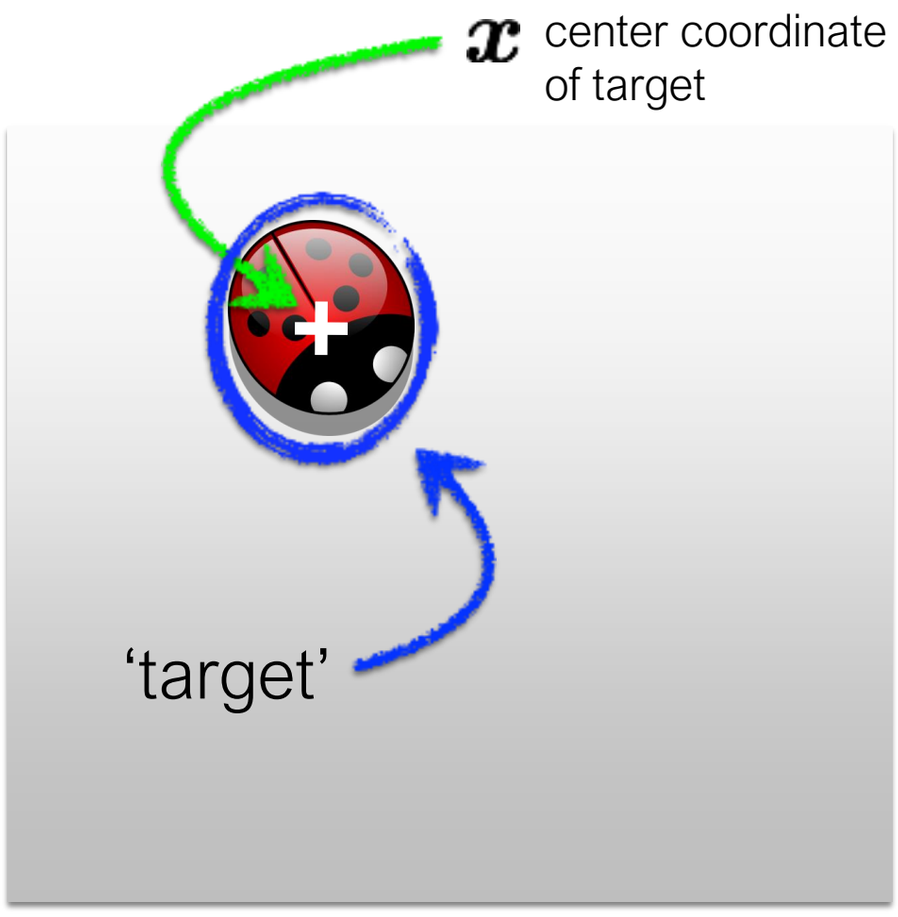


For images, each pixel is point with a weight

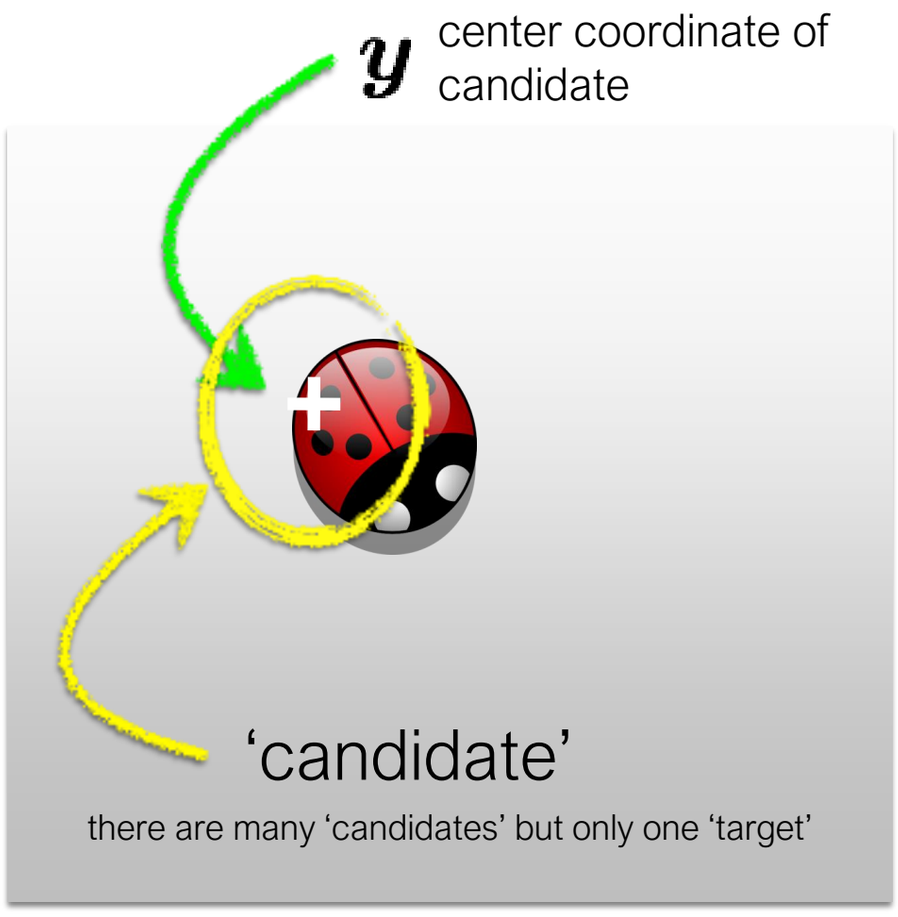


Finally… mean shift tracking in video!

# **Goal:** find the best candidate location in frame 2



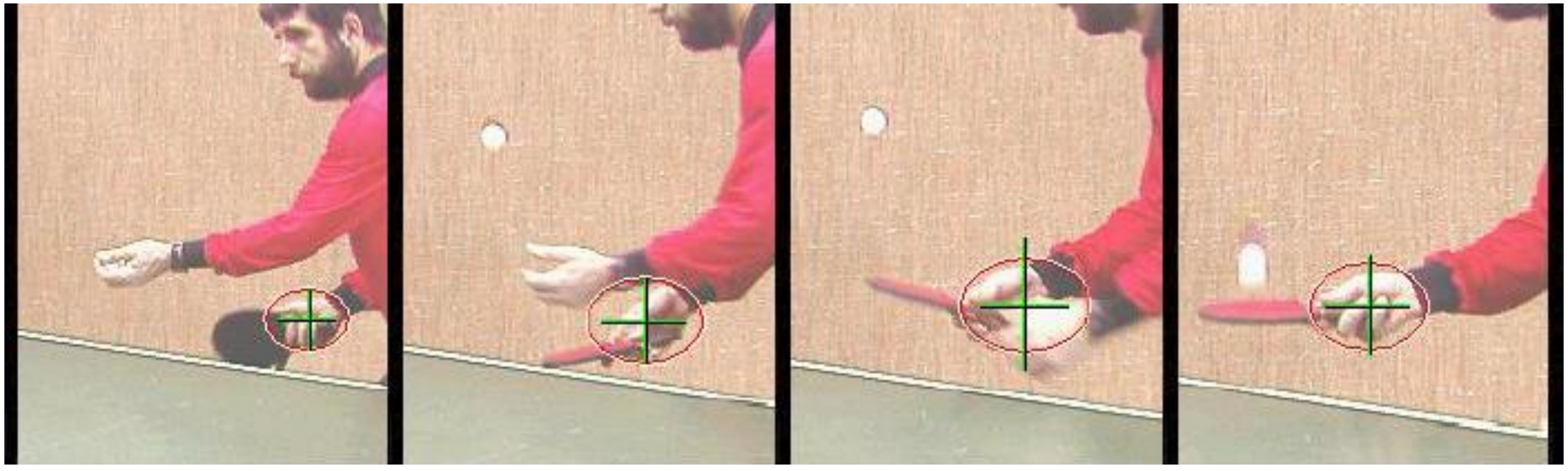

Frame 1




Frame 2

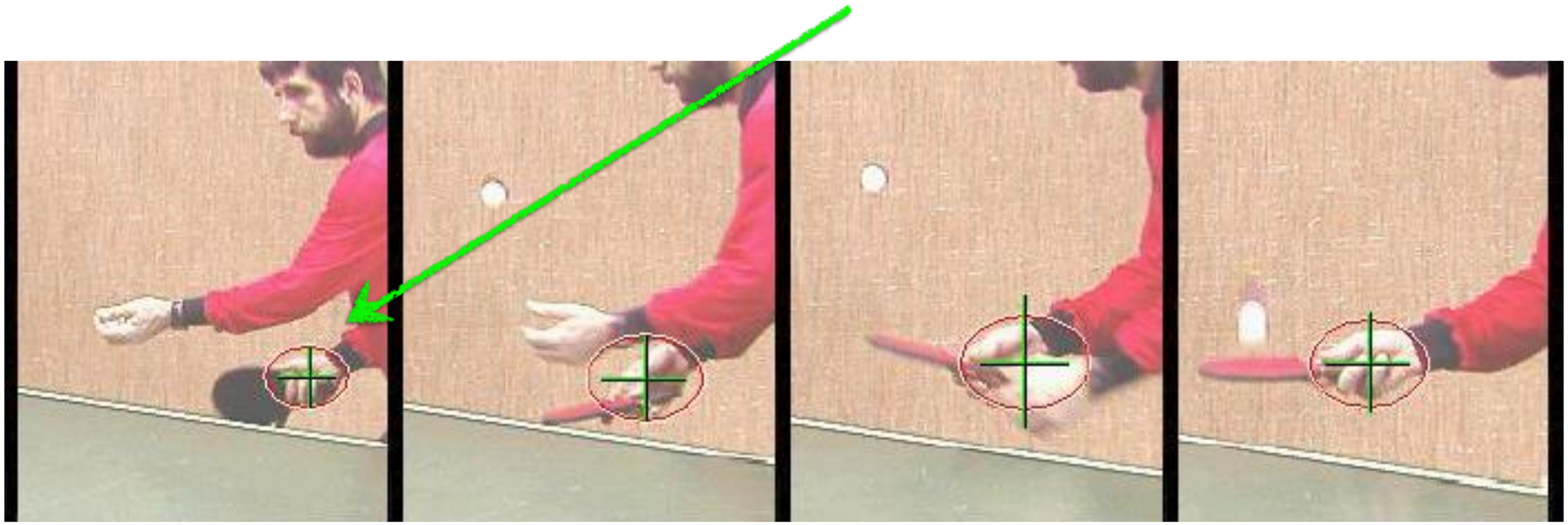
Use the mean shift algorithm
to find the best candidate location

# Non-rigid object tracking



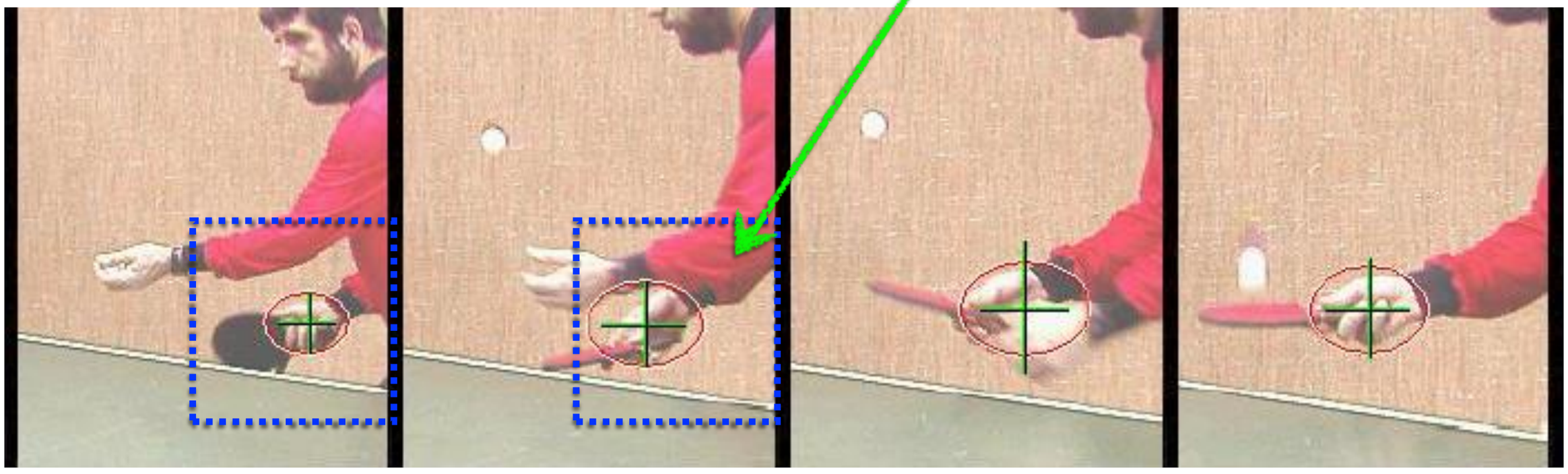
hand tracking

Compute a descriptor for the target



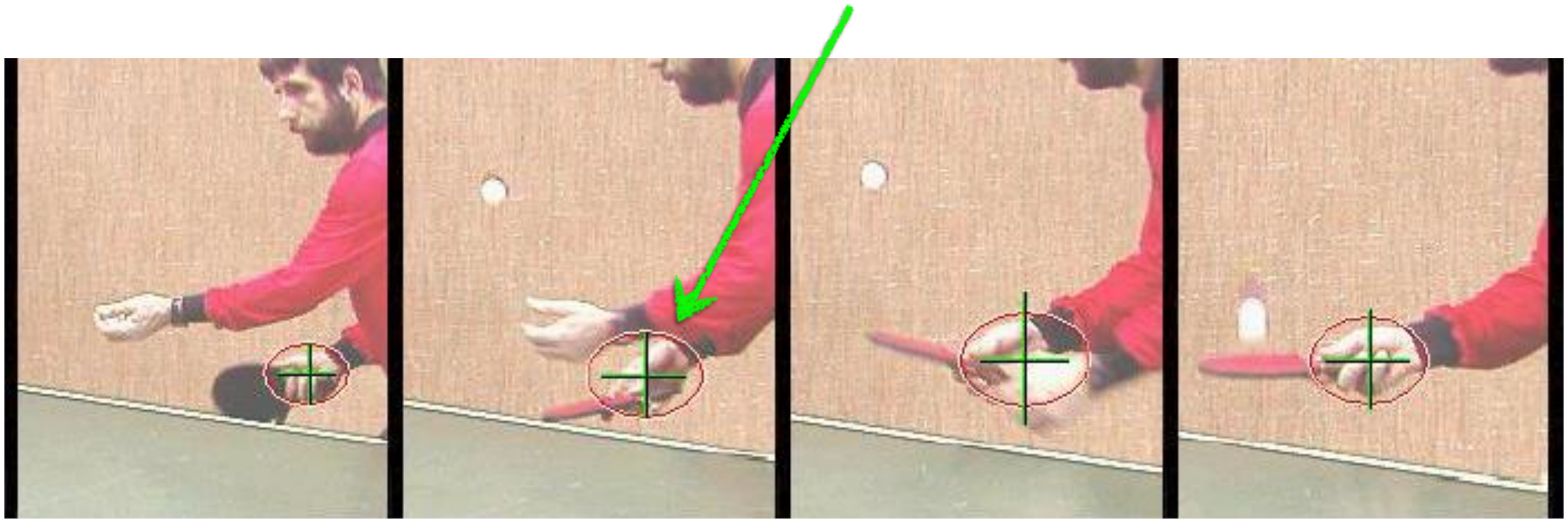
Target

Search for similar descriptor in neighborhood in next frame



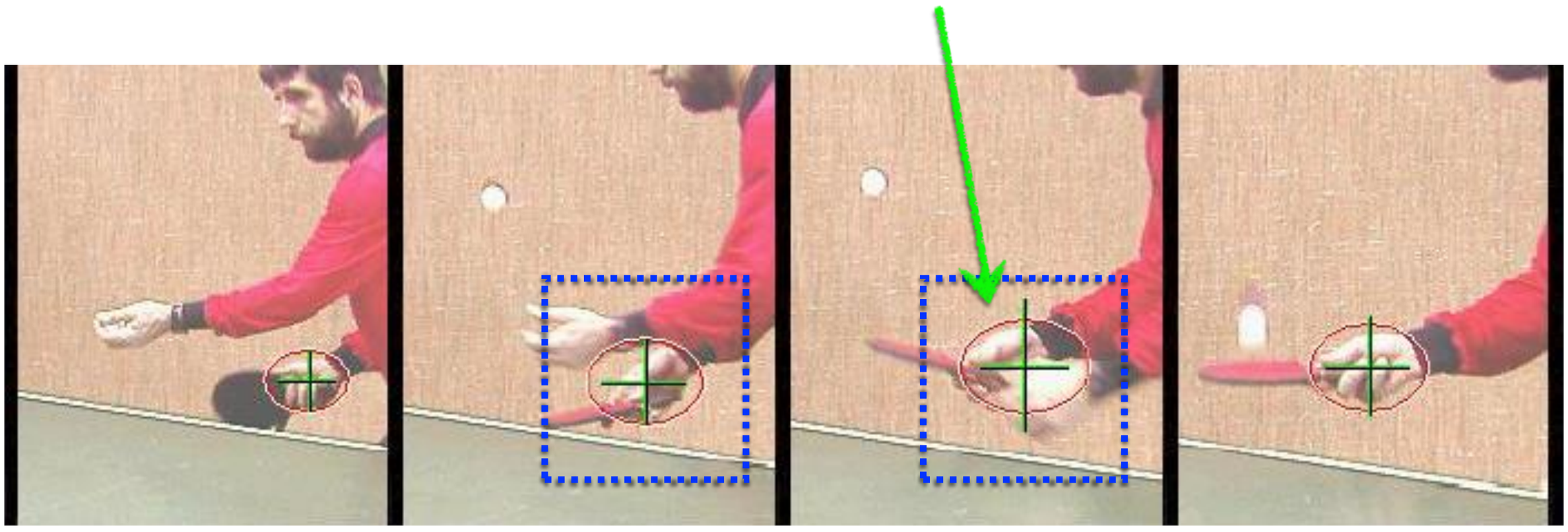
Target Candidate

Compute a descriptor for the new target



Target

Search for similar descriptor in neighborhood in next frame



Target Candidate

# How do we model the target and candidate regions?

# Modeling 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

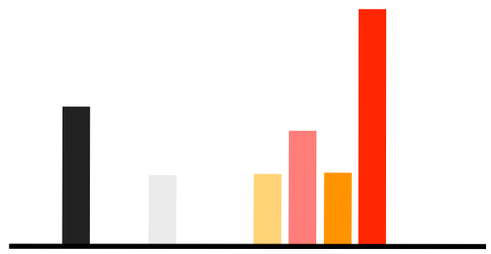
sum over all pixels

function of inverse distance (weight)

Kronecker delta function

quantization function

bin ID



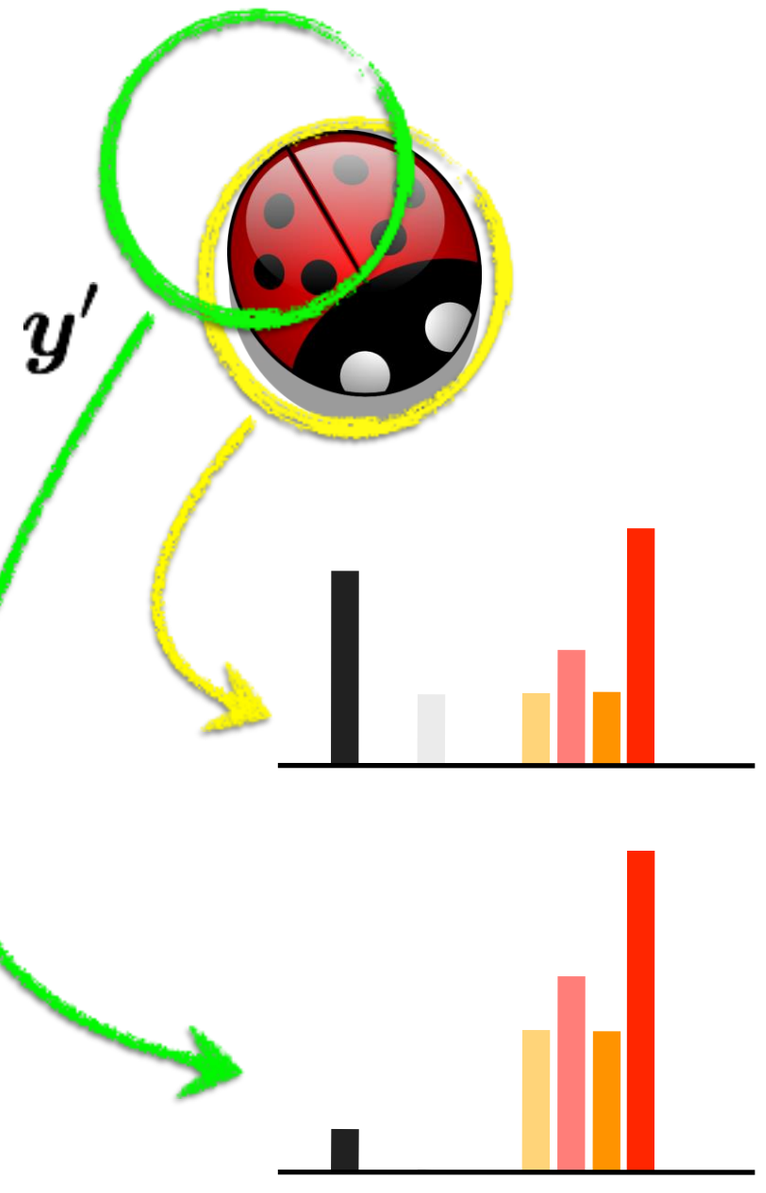
A normalized color histogram (weighted by distance)

# Modeling 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

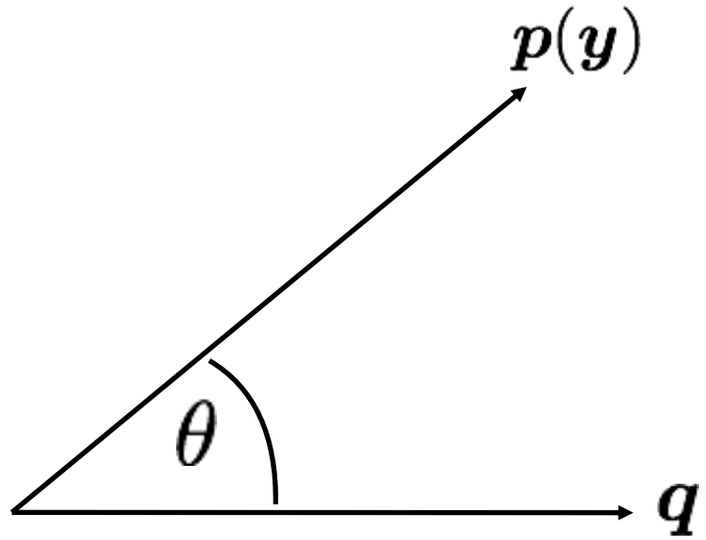
$$d(\boldsymbol{y}) = \sqrt{1 - \rho[\boldsymbol{p}(\boldsymbol{y}), \boldsymbol{q}]}$$

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

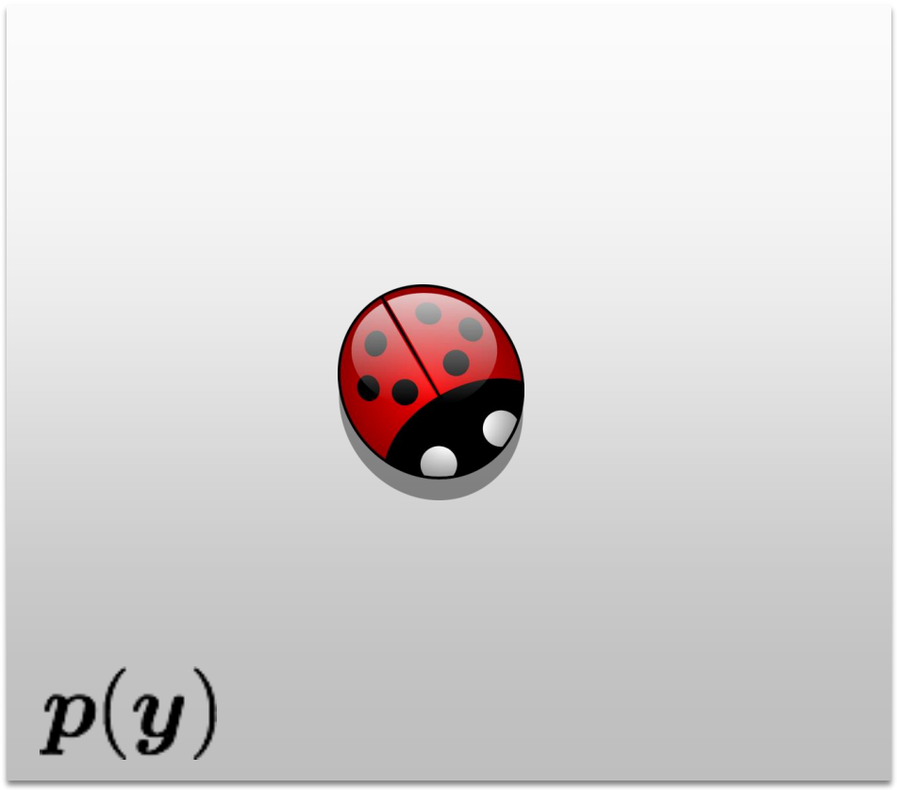
$$\rho(\boldsymbol{y}) = \cos\theta_{\boldsymbol{y}} = \frac{\boldsymbol{p}(\boldsymbol{y})^\top \boldsymbol{q}}{\|\boldsymbol{p}\|\|\boldsymbol{q}\|} = \sum_m \sqrt{p_m(\boldsymbol{y}) q_m}$$



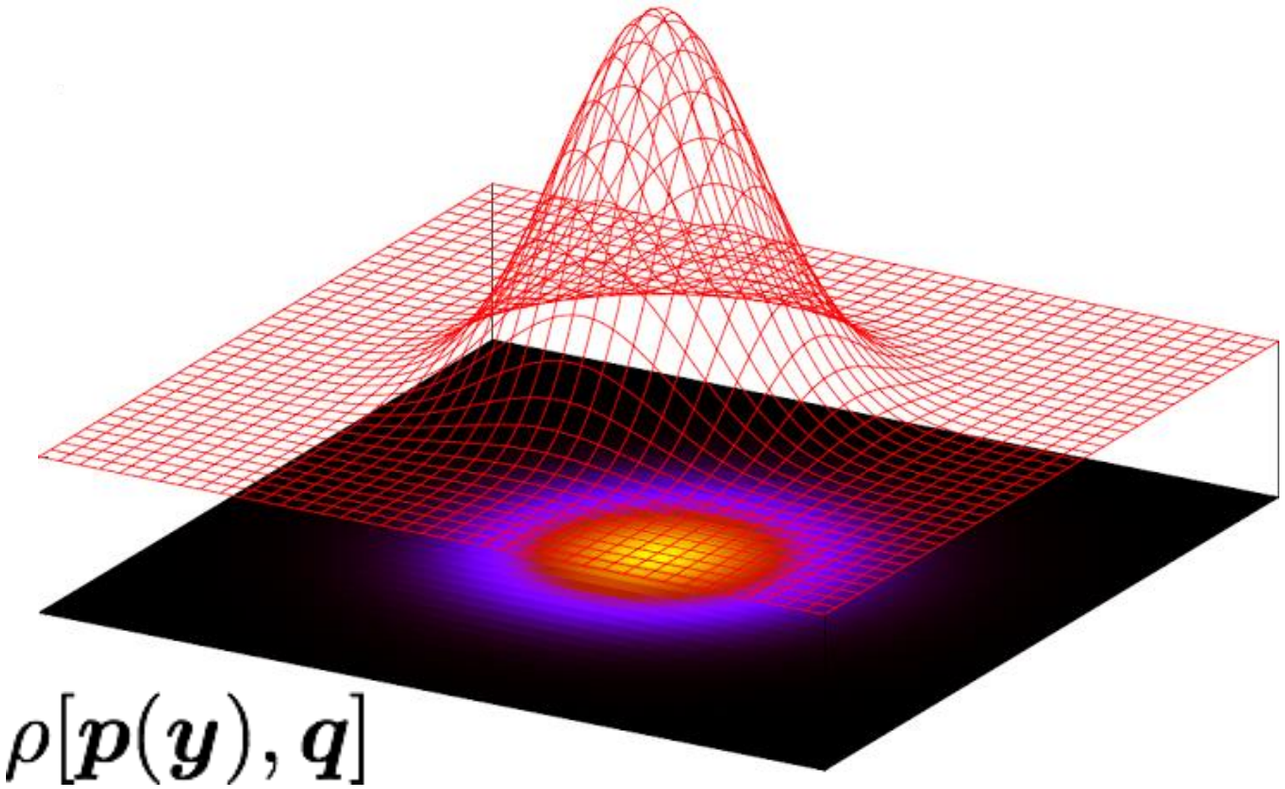
Now we can compute the similarity between a target and multiple candidate regions

$q$

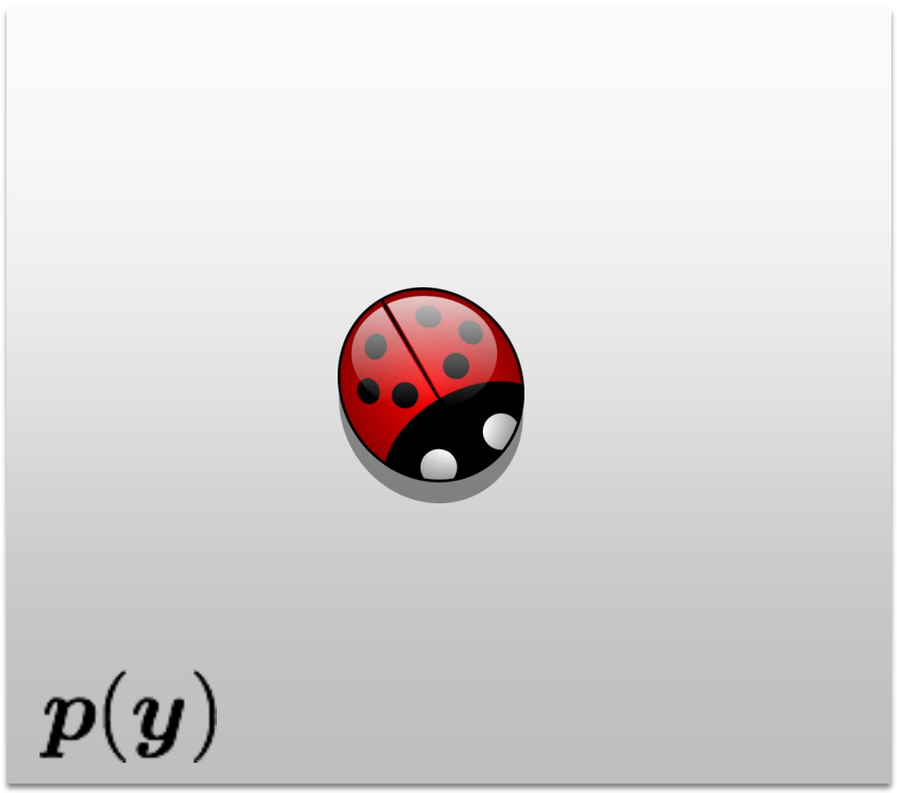
target




image



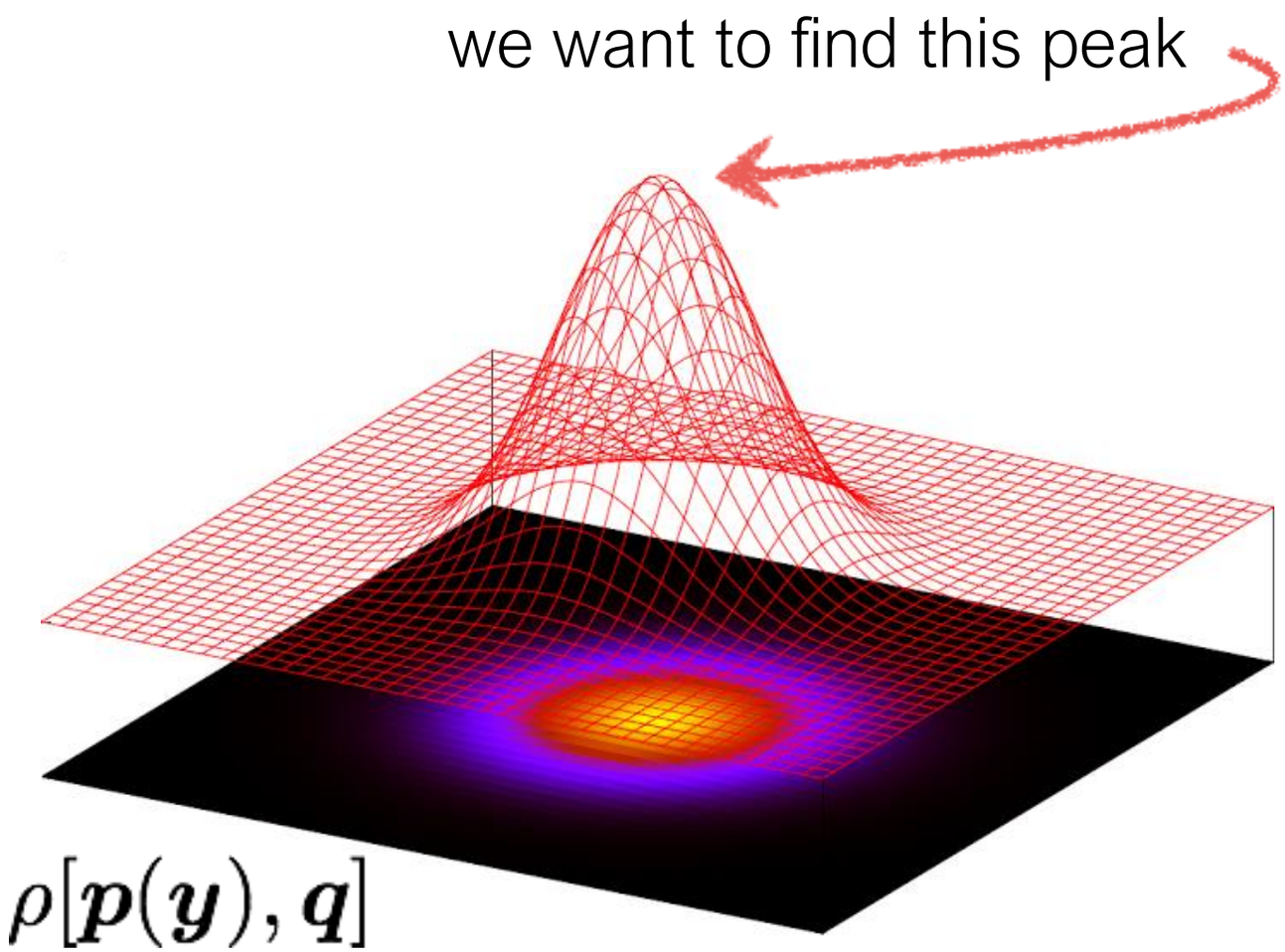

similarity over image




image


target

we want to find this peak




similarity over image

# Objective function

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

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

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

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

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…

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

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

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

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

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

Fully expanded linearized objective

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

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

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

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

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

Fully expanded linearized objective

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

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 $\boldsymbol{p}(\boldsymbol{y}_0)$

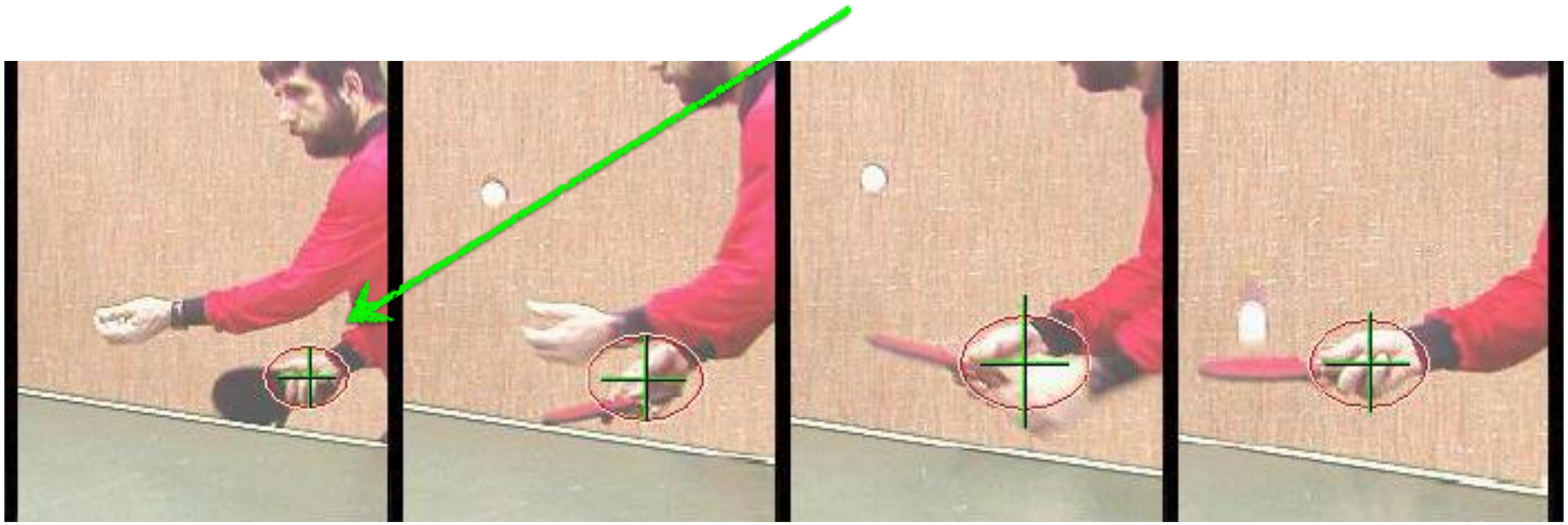
2. Derive weights $w_n$

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

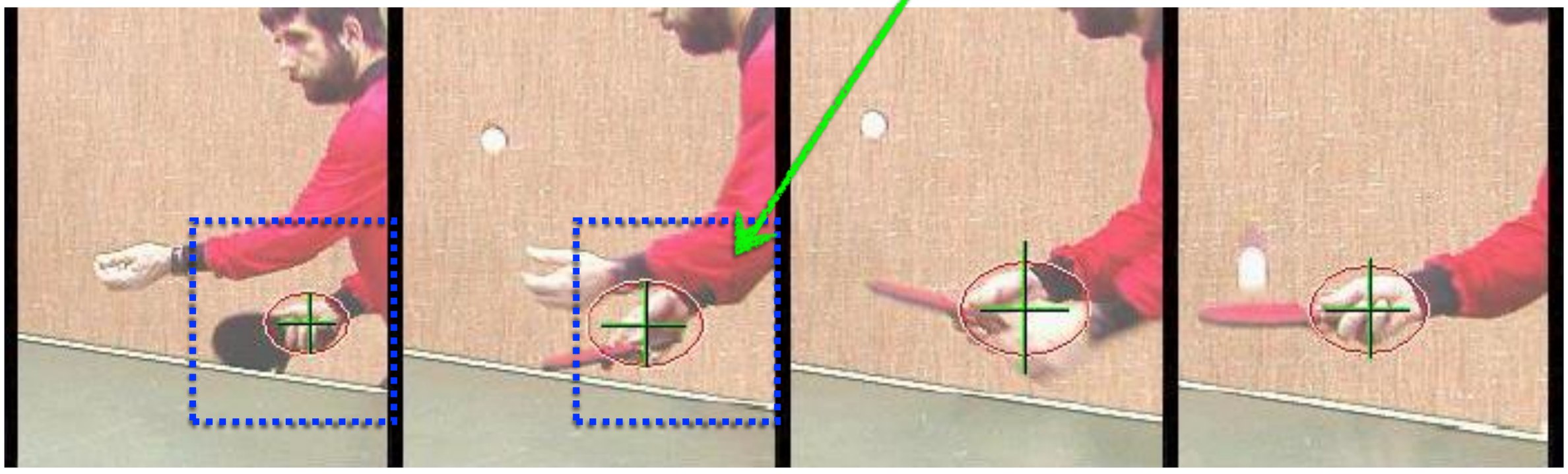
Compute a descriptor for the target



Target

$q$

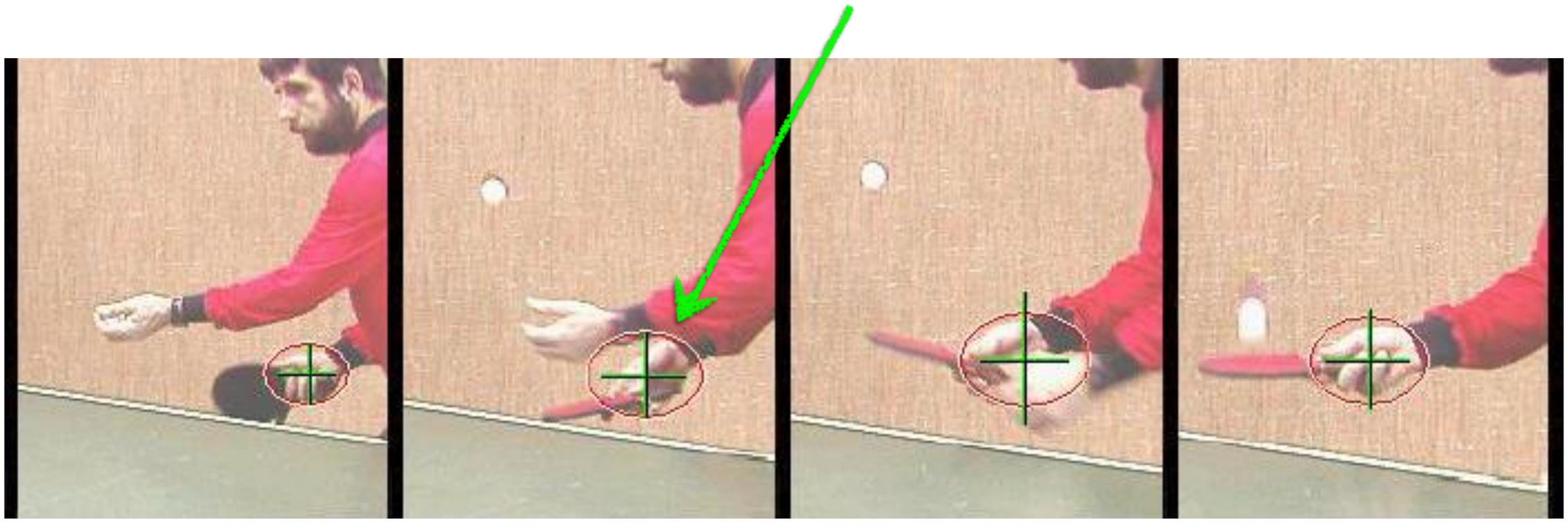
Search for similar descriptor in neighborhood in next frame



Target Candidate

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

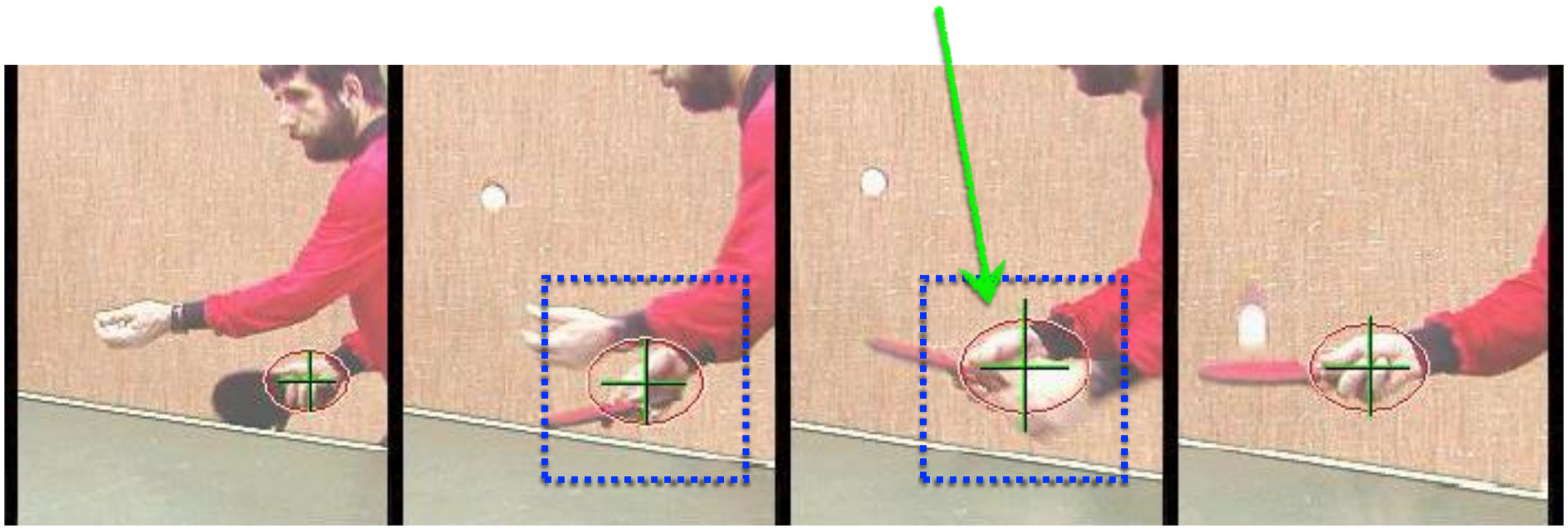
Compute a descriptor for the new target



Target

$q$

Search for similar descriptor in neighborhood in next frame



Target Candidate

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

# Modern trackers

# Learning Multi-Domain Convolutional Neural Networks for Visual Tracking

Hyeonseob Nam and Bohyung Han

# References

Basic reading:

- Szeliski, Sections 4.1.4, 5.3, 8.1.