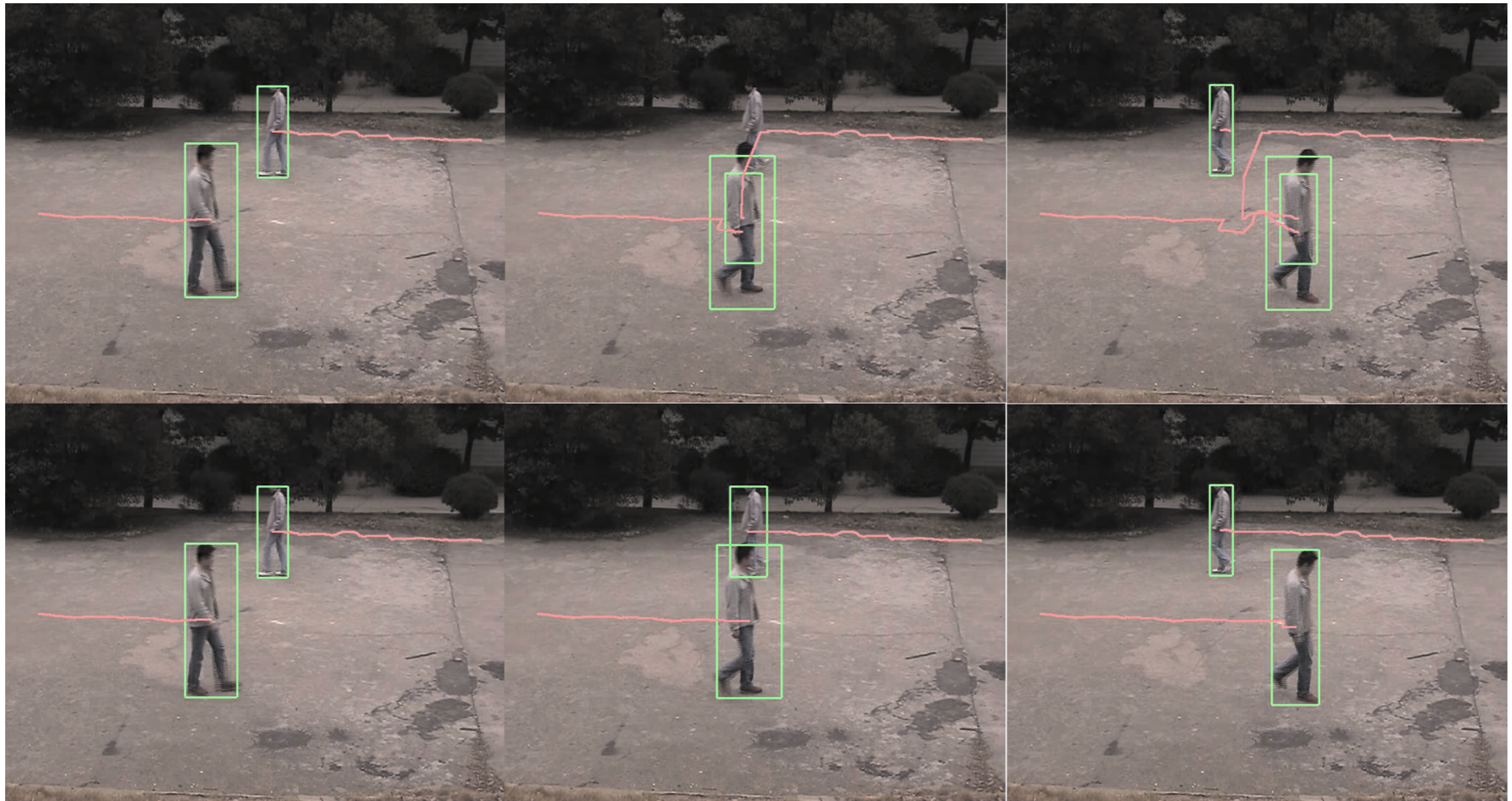


# Alignment and tracking



# Overview of today's 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

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



motion-magnified

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

# Some more examples

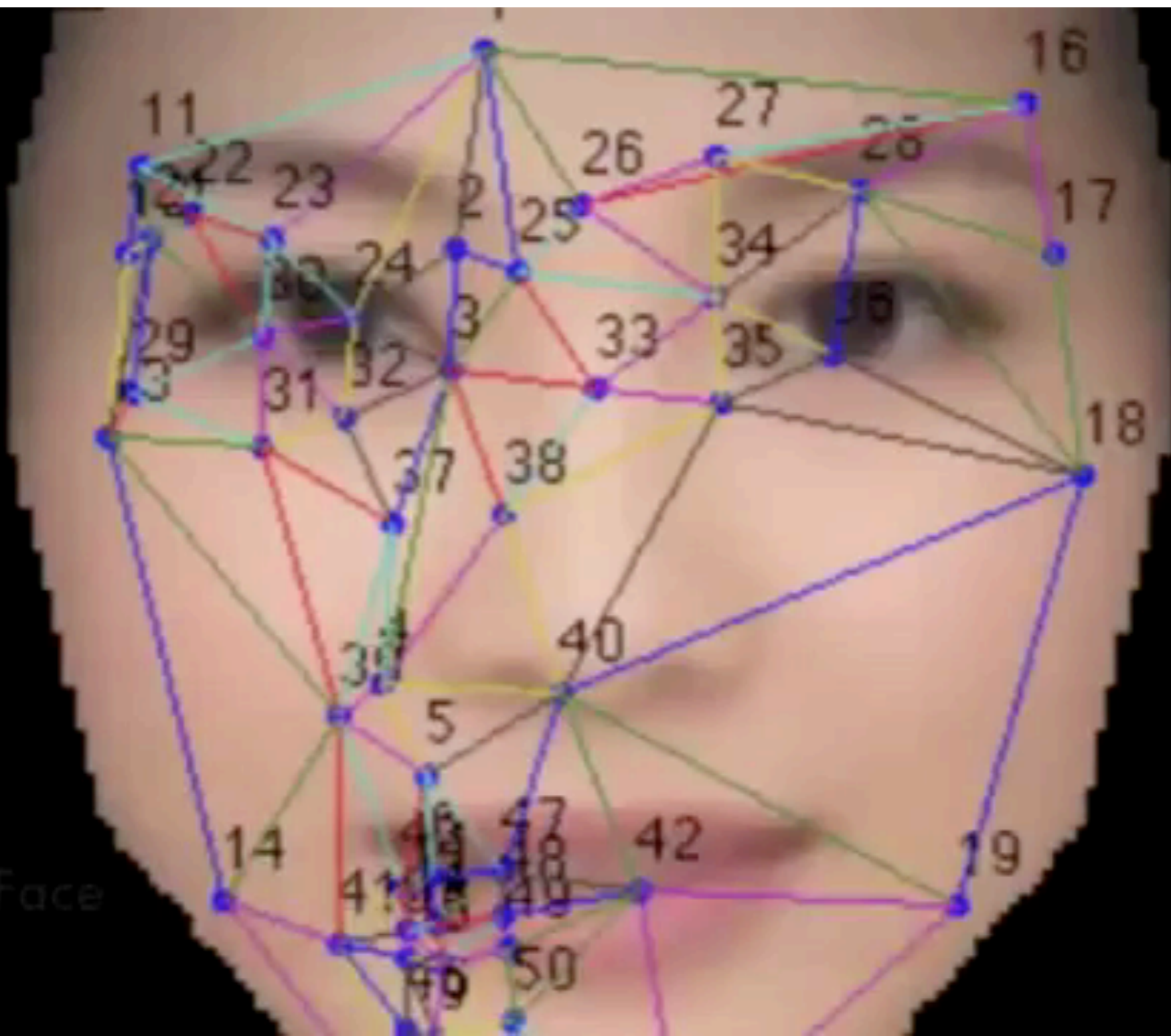


# Some more examples



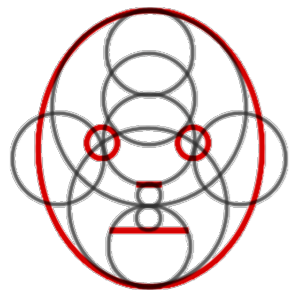
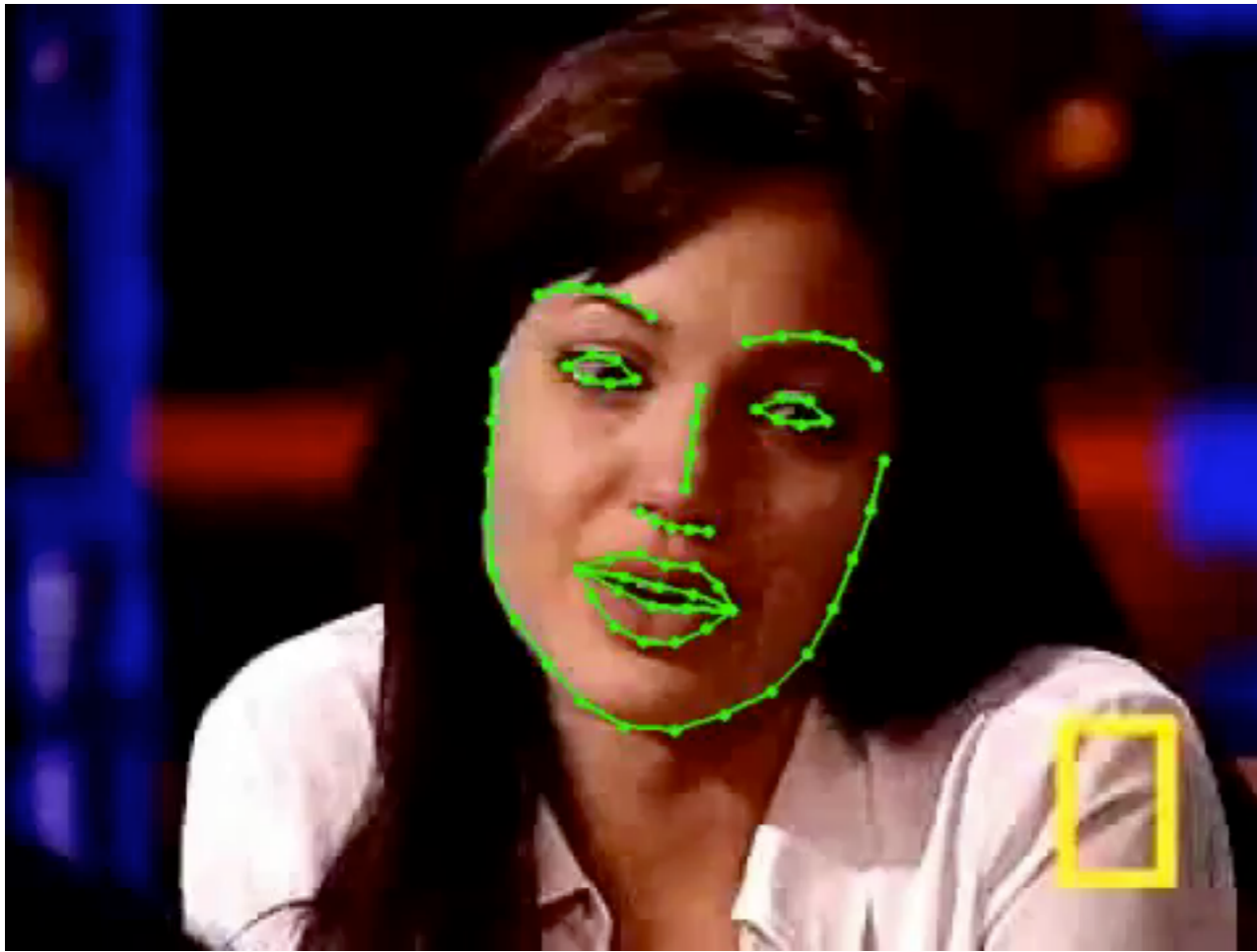


# Image alignment



Average Face





**IntraFace**



How can I find



in the image?



# Idea #1: Template Matching



Slow, combinatorial, global solution

# Idea #2: Pyramid Template Matching

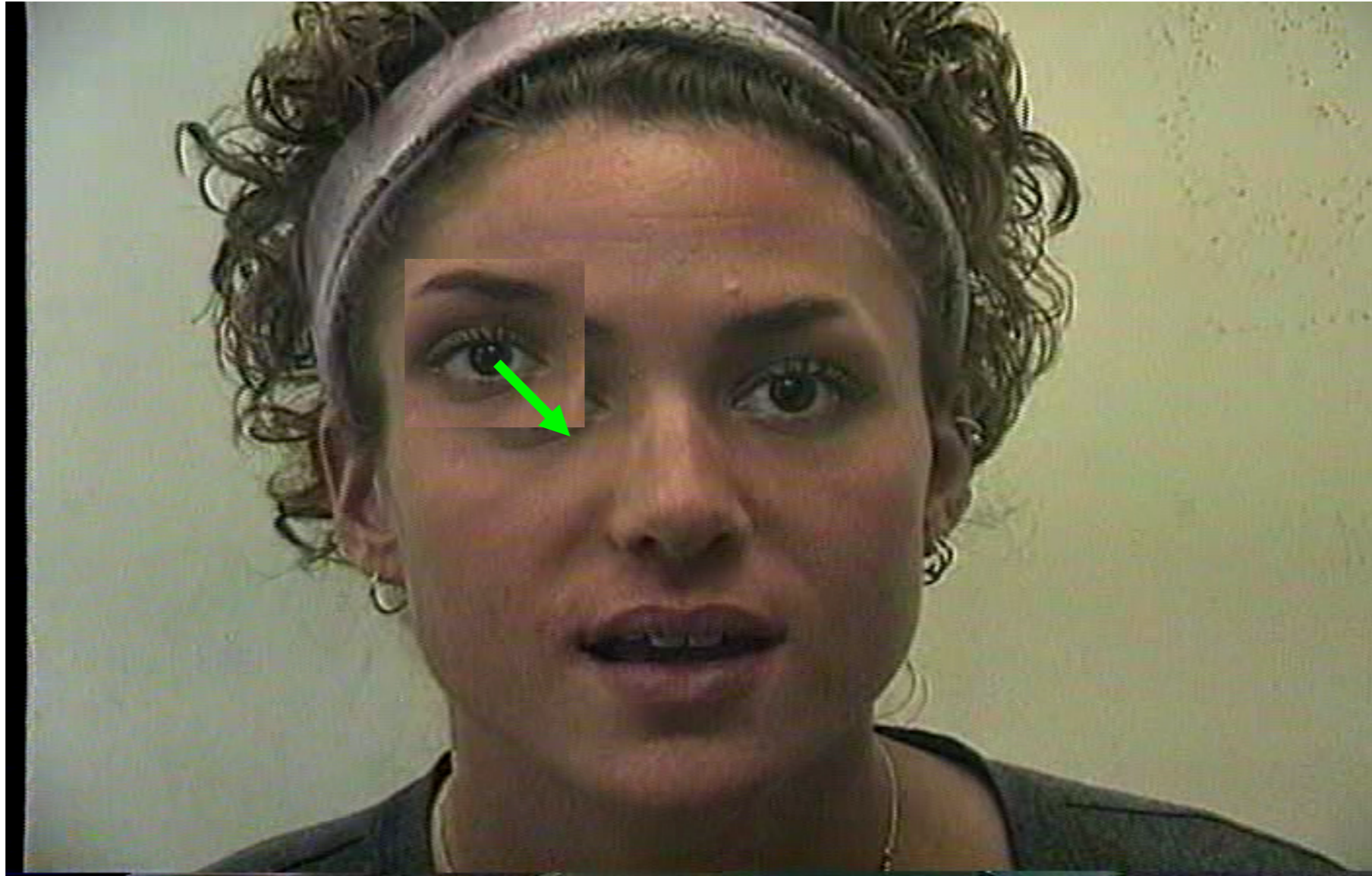


Faster, combinatorial, locally optimal



# 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

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

2D image coordinate

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

Parameters of the transformation

$$\mathbf{p} = \{p_1, \dots, p_N\}$$

Warped image

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

Pixel value at a coordinate

**Translation**

**Affine**

# Some notation before we get into the math...

2D image transformation

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

2D image coordinate

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

Parameters of the transformation

$$\mathbf{p} = \{p_1, \dots, p_N\}$$

Warped image

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

Pixel value at a coordinate

## Translation

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

transform                      coordinate

## Affine

# Some notation before we get into the math...

2D image transformation

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

2D image coordinate

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

Parameters of the transformation

$$\mathbf{p} = \{p_1, \dots, p_N\}$$

Warped image

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

Pixel value at a coordinate

## Translation

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

transform                      coordinate

## Affine

$$\begin{aligned} \mathbf{W}(\mathbf{x}; \mathbf{p}) &= \begin{bmatrix} p_1x + p_2y + p_3 \\ p_4x + p_5y + p_6 \end{bmatrix} \\ &= \begin{bmatrix} p_1 & p_2 & p_3 \\ p_4 & p_5 & p_6 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \end{aligned}$$

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]

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

$\mathbf{W}(\mathbf{x}; \mathbf{p})$  returns a \_\_\_\_\_ of dimension \_\_\_\_ x \_\_\_\_

$\mathbf{p} = \{p_1, \dots, p_N\}$  where N is \_\_\_\_\_ for an affine model

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

# Image alignment

(problem definition)

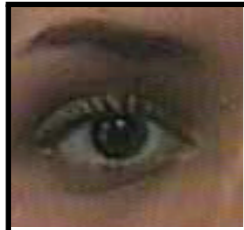
$$\min_{\mathbf{p}} \sum_{\mathbf{x}} [I(\mathbf{W}(\mathbf{x}; \mathbf{p})) - T(\mathbf{x})]^2$$

warped image                      template image

Find the warp parameters  $\mathbf{p}$  such that the SSD is minimized

Find the warp parameters  $\mathbf{p}$  such that the SSD is minimized

$T(\mathbf{x})$



$I(\mathbf{x})$



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



# Image alignment

(problem definition)

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

warped image                      template image

Find the warp parameters  $\mathbf{p}$  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  $I$  is non-parametric)

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

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  $I$  is non-parametric)

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

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  $\mathbf{p}$ ...

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

can be written as ...

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

(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  $I$  is non-parametric)

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

*How can we linearize the function  $I$  for a really small perturbation of  $\mathbf{p}$ ?*

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

(Function  $I$  is non-parametric)

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

*How can we linearize the function  $I$  for a really small perturbation of  $\mathbf{p}$ ?*

Taylor series approximation!

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

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

$$\begin{aligned} I(\mathbf{W}(\mathbf{x}; \mathbf{p} + \Delta \mathbf{p})) &\approx I(\mathbf{W}(\mathbf{x}; \mathbf{p})) + \frac{\partial I(\mathbf{W}(\mathbf{x}; \mathbf{p}))}{\partial \mathbf{p}} \Delta \mathbf{p} \\ \text{chain rule} \quad &= I(\mathbf{W}(\mathbf{x}; \mathbf{p})) + \frac{\partial I(\mathbf{W}(\mathbf{x}; \mathbf{p}))}{\partial \mathbf{x}'} \frac{\partial \mathbf{W}(\mathbf{x}; \mathbf{p})}{\partial \mathbf{p}} \Delta \mathbf{p} \\ \text{short-hand} \quad &= I(\mathbf{W}(\mathbf{x}; \mathbf{p})) + \nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \Delta \mathbf{p} \end{aligned}$$

short-hand



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

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

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

*What are the unknowns here?*

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

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

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

Now, the function is a linear function of the unknowns

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

$\mathbf{x}$  is a \_\_\_\_\_ of dimension \_\_\_\_ x \_\_\_\_

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

$\mathbf{p}$  is a \_\_\_\_\_ of dimension \_\_\_\_ x \_\_\_\_

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

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

(A matrix of partial derivatives)

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

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

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

Rate of change of the warp

Affine transform

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

$$\frac{\partial W_x}{\partial p_1} = x \quad \frac{\partial W_x}{\partial p_2} = 0 \quad \dots$$

$$\frac{\partial W_y}{\partial p_1} = 0 \quad \dots$$

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

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

$\nabla I$  is a \_\_\_\_\_ of dimension \_\_\_\_ x \_\_\_\_

$\frac{\partial \mathbf{W}}{\partial \mathbf{p}}$  is a \_\_\_\_\_ of dimension \_\_\_\_ x \_\_\_\_

$\Delta \mathbf{p}$  is a \_\_\_\_\_ of dimension \_\_\_\_ x \_\_\_\_

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



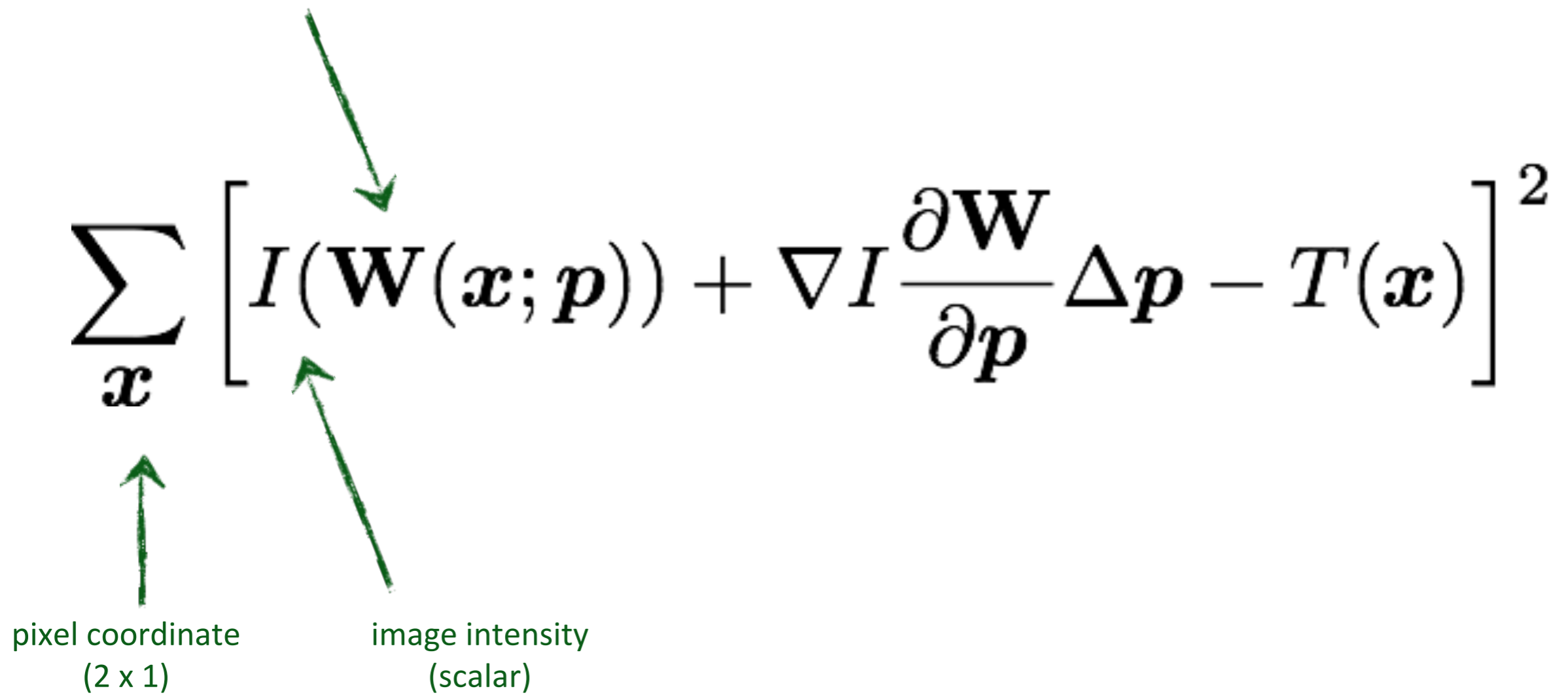
$$\sum_{\mathbf{x}} \left[ I(\mathbf{W}(\mathbf{x}; \mathbf{p})) + \nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \Delta \mathbf{p} - T(\mathbf{x}) \right]^2$$

↑  
pixel coordinate  
(2 x 1)

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

pixel coordinate  
(2 x 1)

image intensity  
(scalar)





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

warp function  
 (2 x 1)

pixel coordinate  
 (2 x 1)

image intensity  
 (scalar)

warp function  
(2 x 1)

warp parameters  
(6 for affine)

pixel coordinate  
(2 x 1)

image intensity  
(scalar)

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

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

warp function (2 x 1)   
 warp parameters (6 for affine)   
 pixel coordinate (2 x 1)   
 image intensity (scalar)   
 image gradient (1 x 2)

Partial derivatives of warp function  
(2 x 6)

warp function  
(2 x 1)

warp parameters  
(6 for affine)

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

image gradient  
(1 x 2)

image intensity  
(scalar)

pixel coordinate  
(2 x 1)

Partial derivatives of warp function  
(2 x 6)

warp function  
(2 x 1)

warp parameters  
(6 for affine)

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

image gradient  
(1 x 2)

incremental warp  
(6 x 1)

pixel coordinate  
(2 x 1)

image intensity  
(scalar)

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

Diagram illustrating the components of the image registration cost function:

- warp function** (2 x 1):  $\mathbf{W}(\mathbf{x}; \mathbf{p})$
- warp parameters** (6 for affine):  $\mathbf{p}$
- Partial derivatives of warp function** (2 x 6):  $\frac{\partial \mathbf{W}}{\partial \mathbf{p}}$
- template image intensity** (scalar):  $T(\mathbf{x})$
- image gradient** (1 x 2):  $\nabla I$
- incremental warp** (6 x 1):  $\Delta \mathbf{p}$
- pixel coordinate** (2 x 1):  $\mathbf{x}$
- image intensity** (scalar):  $I(\mathbf{W}(\mathbf{x}; \mathbf{p}))$

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:

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

warped image                      template image

Difficult non-linear optimization problem

Strategy:

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

Assume known approximate solution  
Solve for increment

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

Taylor series approximation  
Linearize

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

OK, so how do we solve this?

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



Another way to look at it...

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

(moving terms around)

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

vector of  
constants

vector of  
variables

constant

*Have you seen this form of optimization problem before?*

Another way to look at it...

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

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

Looks like

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

*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:

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

is optimized when

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

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

$$\text{where } H = \sum_x \left[ \nabla I \frac{\partial \mathbf{W}}{\partial p} \right]^\top \left[ \nabla I \frac{\partial \mathbf{W}}{\partial p} \right] \quad A^\top A$$

## Solve:

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

warped image                      template image

Difficult non-linear optimization problem

## Strategy:

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

Assume known approximate solution  
Solve for increment

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

Taylor series approximation  
Linearize

## Solution:

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

Solution to least squares  
approximation

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

Hessian

This is called...

**Gauss-Newton gradient decent  
non-linear optimization!**

# Lucas Kanade (Additive alignment)

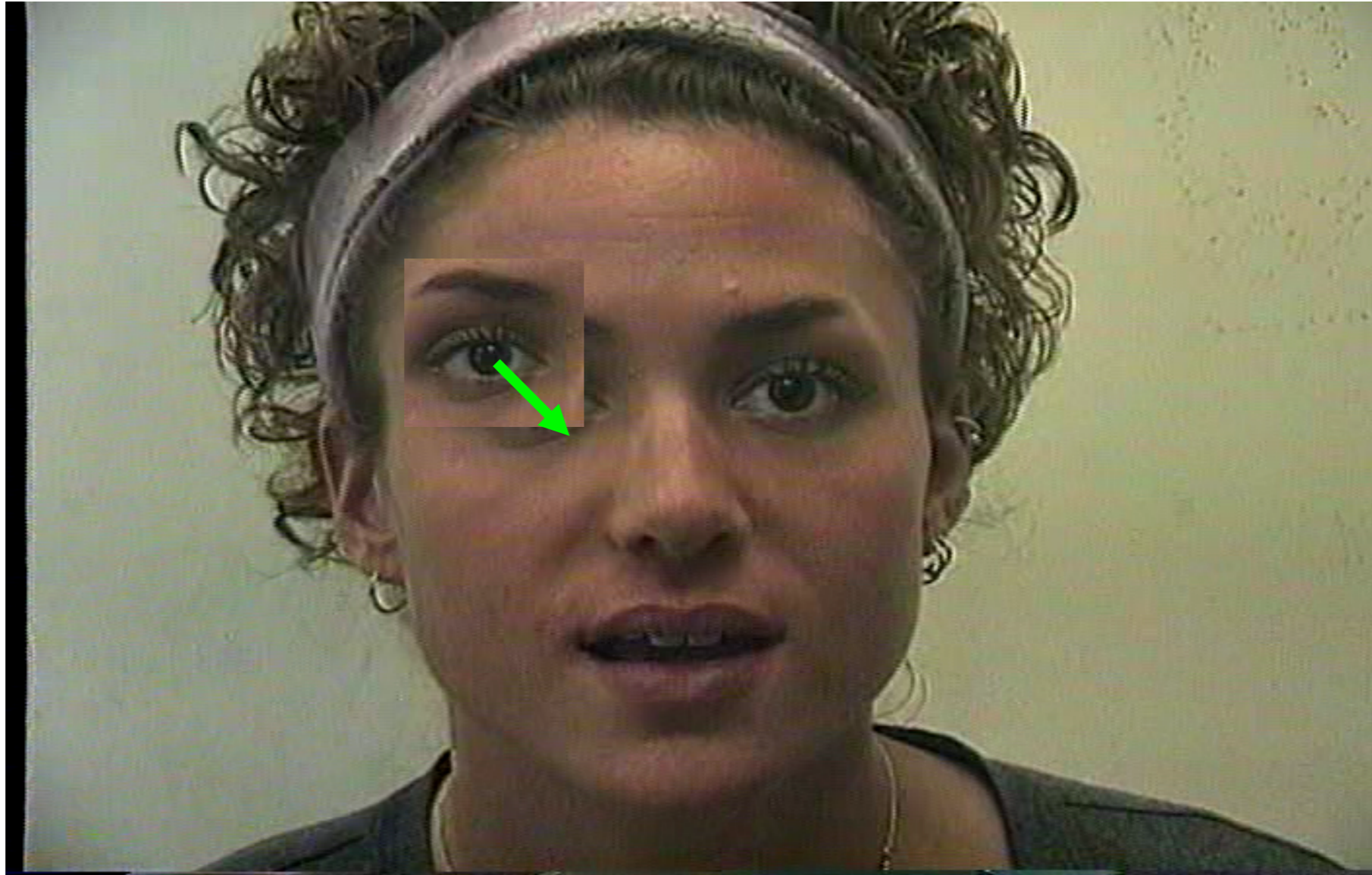
1. Warp image  $I(\mathbf{W}(\mathbf{x}; \mathbf{p}))$
2. Compute error image  $[T(\mathbf{x}) - I(\mathbf{W}(\mathbf{x}; \mathbf{p}))]$
3. Compute gradient  $\nabla I(\mathbf{x}')$   $x'$  coordinates of the warped image  
(gradients of the warped image)
4. Evaluate Jacobian  $\frac{\partial \mathbf{W}}{\partial \mathbf{p}}$
5. Compute Hessian  $H = \sum_{\mathbf{x}} \left[ \nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \right]^{\top} \left[ \nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \right]$
6. Compute  $\Delta \mathbf{p} = H^{-1} \sum_{\mathbf{x}} \left[ \nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \right]^{\top} [T(\mathbf{x}) - I(\mathbf{W}(\mathbf{x}; \mathbf{p}))]$
7. Update parameters  $\mathbf{p} \leftarrow \mathbf{p} + \Delta \mathbf{p}$

**Just 8 lines of code!**

# Baker-Matthews alignment

# Image Alignment

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





## Image Alignment Objective Function

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

Given an initial solution...several possible formulations

### Additive Alignment

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

incremental perturbation of parameters

## Image Alignment Objective Function

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

Given an initial solution...several possible formulations

### Additive Alignment

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

incremental perturbation of parameters

### Compositional Alignment

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

incremental warps of image

# Additive strategy



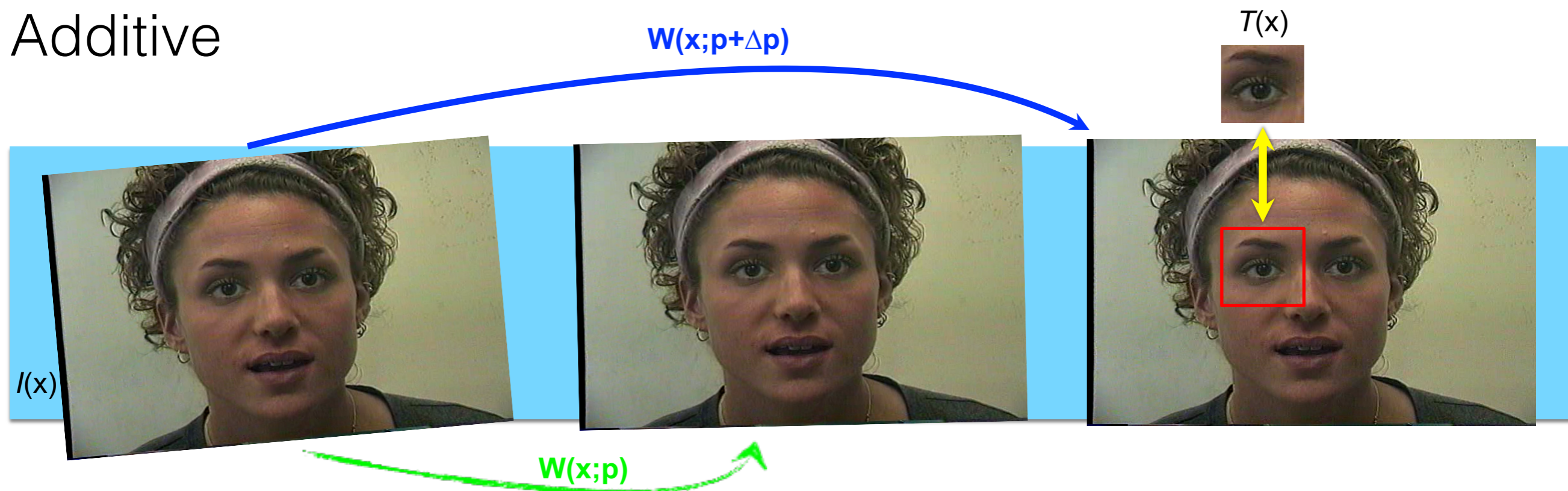
# Compositional strategy



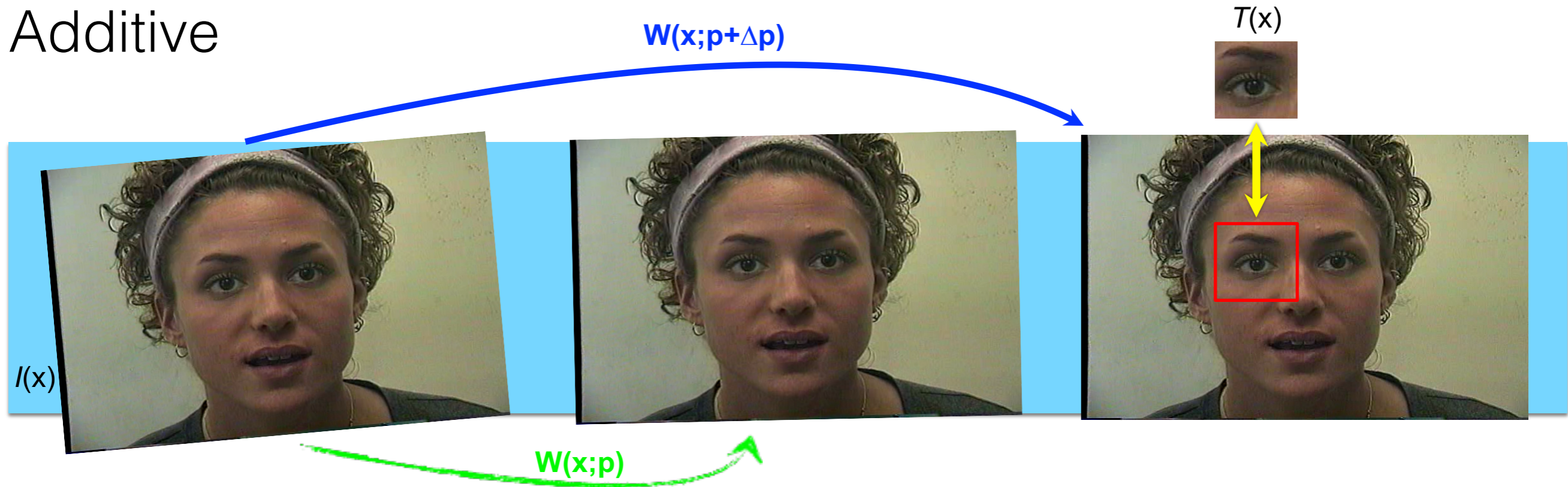
# Additive



Additive



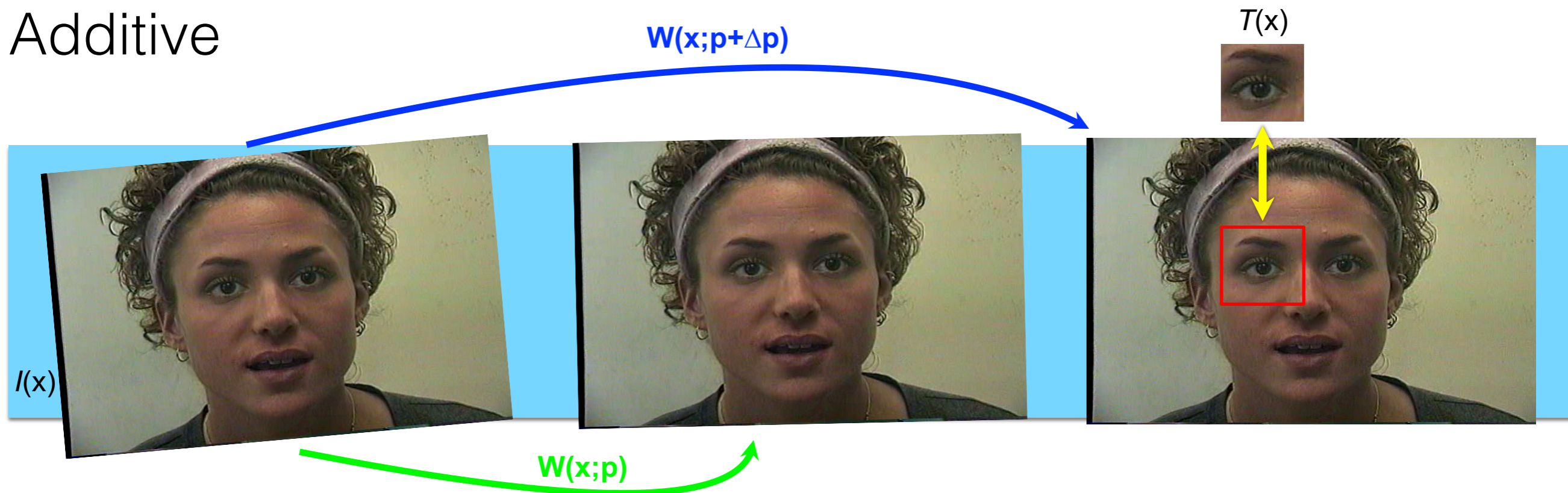
# Additive



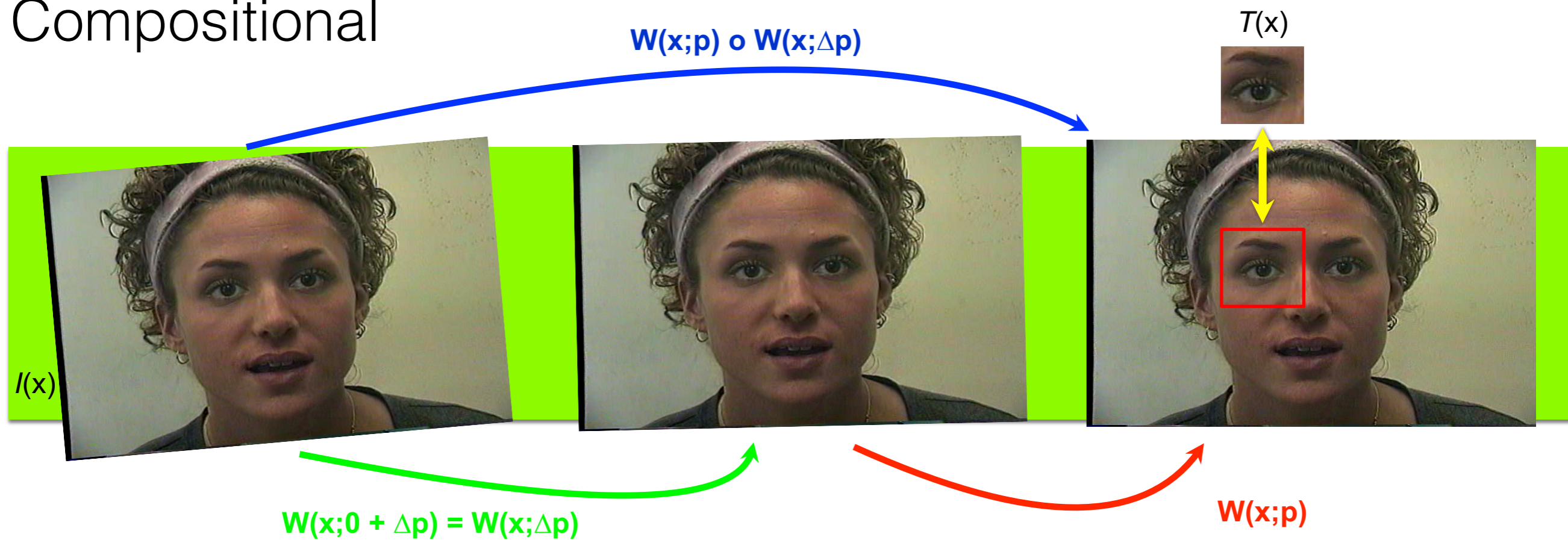
# Compositional



# Additive



# Compositional





# Compositional Alignment

Original objective function (SSD)

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

Assuming an initial solution  $\mathbf{p}$  and a compositional warp increment

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

# Compositional Alignment

Original objective function (SSD)

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

Assuming an initial solution  $\mathbf{p}$  and a compositional warp increment

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

Another way to write the composition

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

Identity warp

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

# Compositional Alignment

Original objective function (SSD)

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

Assuming an initial solution  $\mathbf{p}$  and a compositional warp increment

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

Another way to write the composition

Identity warp

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

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

Skipping over the derivation...the new update rule is

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

So what's so great about this compositional form?

## Additive Alignment

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

linearized form

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

## Compositional Alignment

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

linearized form

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

Additive Alignment

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

linearized form

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

Jacobian of  $W(\mathbf{x}; \mathbf{p})$

Compositional Alignment

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

linearized form

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

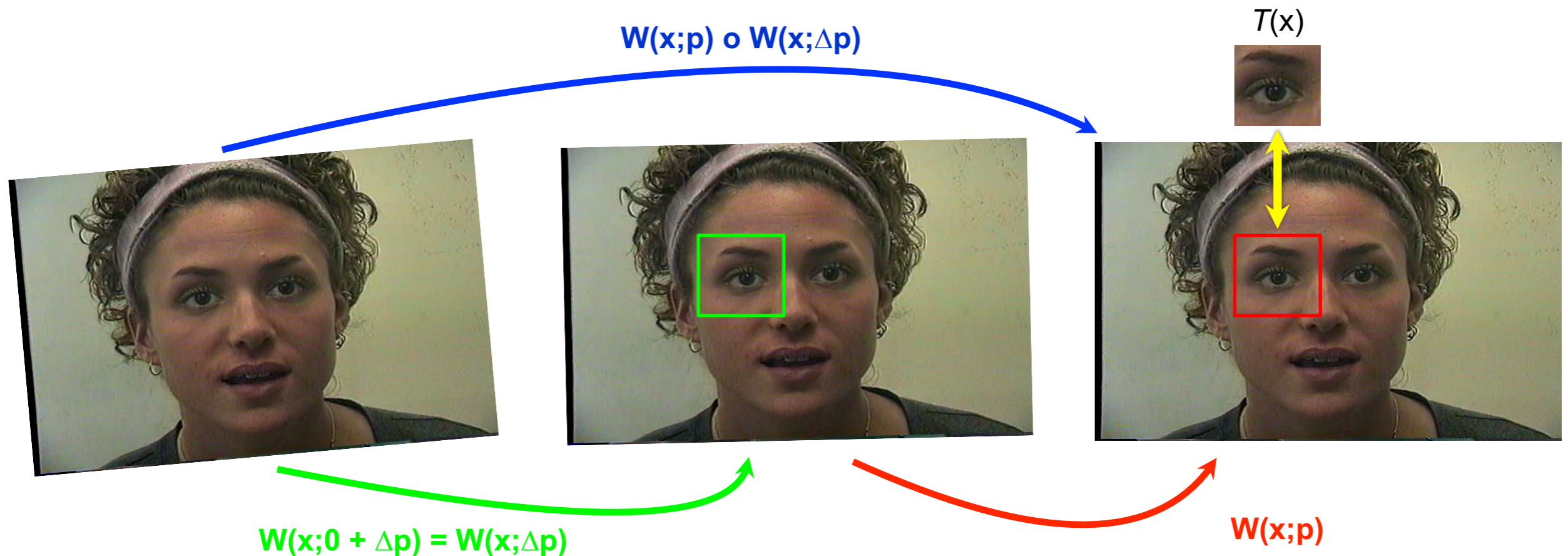
Jacobian of  $W(\mathbf{x}; \mathbf{0})$

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



# Shum-Szeliski (Compositional alignment)

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

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

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

4. Evaluate Jacobian  $\frac{\partial \mathbf{W}(\mathbf{x}; \mathbf{0})}{\partial \mathbf{p}}$

5. Compute Hessian  $H$

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

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

Any other speed up techniques?

Inverse alignment

Why not compute warp updates on the template?

Additive Alignment

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

Compositional Alignment

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

Why not compute warp updates on the template?

Additive Alignment

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

Compositional Alignment

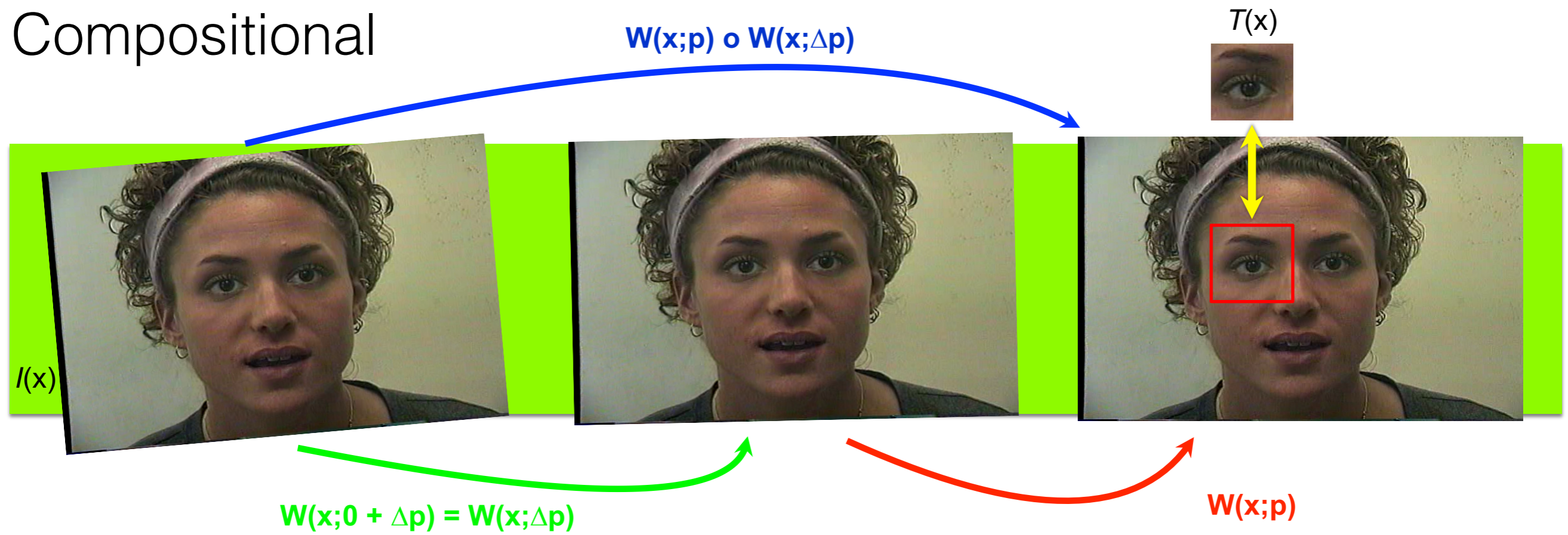
$$\sum_{\mathbf{x}} [I(\mathbf{W}(\mathbf{W}(\mathbf{x}; \Delta\mathbf{p}); \mathbf{p})) - T(\mathbf{x})]^2$$

What happens if you let the template be warped too?

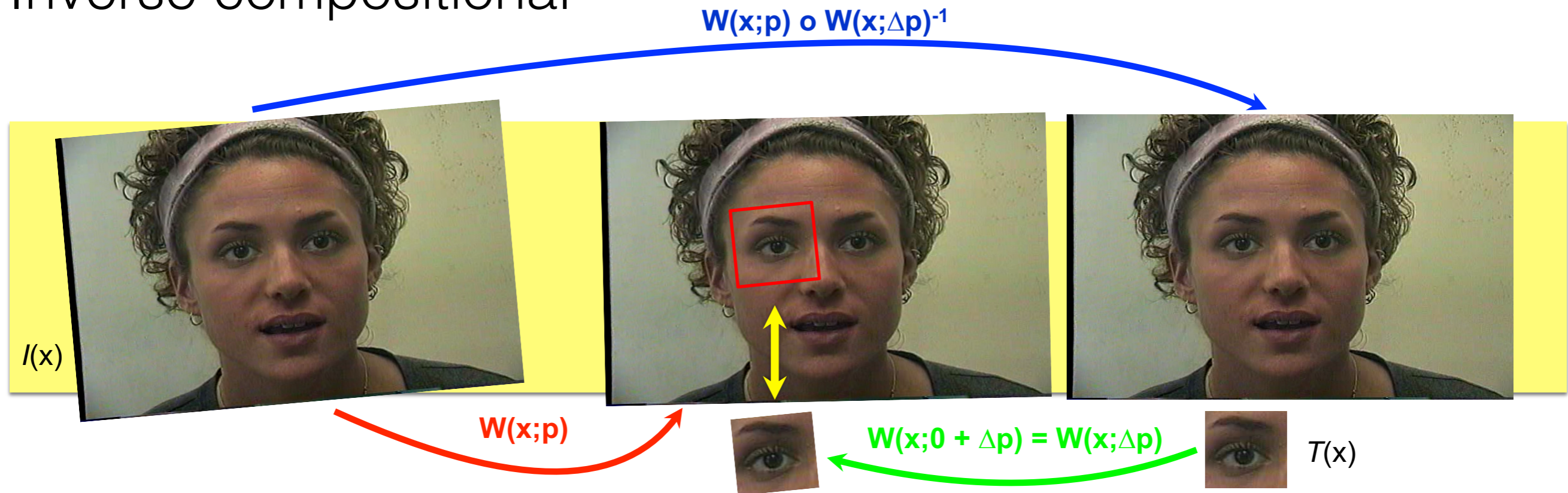
Inverse Compositional Alignment

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

# Compositional



# Inverse compositional



# Compositional strategy



# Inverse Compositional strategy





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

# Inverse Compositional Alignment

## Minimize

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

## Solution

$$H = \sum_{\mathbf{x}} \left[ \nabla T \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \right]^\top \left[ \nabla T \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \right] \quad \text{can be precomputed from template!}$$

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

## Update

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

# 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

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

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

(main term that needs to be computed)

**Warp** must be invertible

# Lucas Kanade (Additive alignment)

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

# Shum-Szeliski (Compositional alignment)

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

# Baker-Matthews (Inverse Compositional alignment)

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

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

3. Compute gradient  $\nabla T(\mathbf{W})$

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

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

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

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

<b>Algorithm</b>	<b>Efficient</b>	<b>Authors</b>
<b>Forwards Additive</b>	No	Lucas, Kanade
<b>Forwards compositional</b>	No	Shum, Szeliski
<b>Inverse Additive</b>	Yes	Hager, Belhumeur
<b>Inverse Compositional</b>	Yes	Baker, Matthews

# Kanade-Lucas-Tomasi (KLT) tracker





# 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)

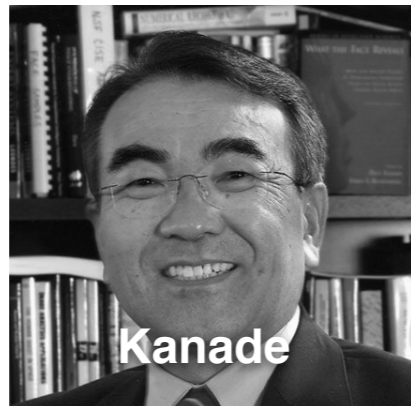
How should we select the 'small images' (features)?

How should we track them from frame to frame?

History of the  
**Kanade-Lucas-Tomasi  
(KLT) Tracker**



Lucas



Kanade

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

**1981**



Kanade

Tomasi

Detection and Tracking of Feature Points.

**1991**



The original KLT algorithm



Tomasi

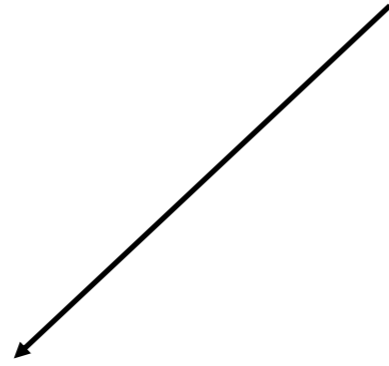


Shi

Good Features to Track.

**1994**

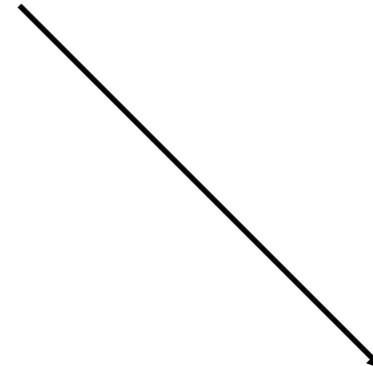
# Kanade-Lucas-Tomasi



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

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

Recall the Lucas-Kanade image alignment method:

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

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

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

Recall the Lucas-Kanade image alignment method:

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

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

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

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

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

Recall the Lucas-Kanade image alignment method:

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

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

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

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

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

Update  $\mathbf{p} \leftarrow \mathbf{p} + \Delta\mathbf{p}$

Stability of gradient decent iterations depends on ...

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

Stability of gradient decent iterations depends on ...

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

Inverting the Hessian

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

*When does the inversion fail?*

Stability of gradient decent iterations depends on ...

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

Inverting the Hessian

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

*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

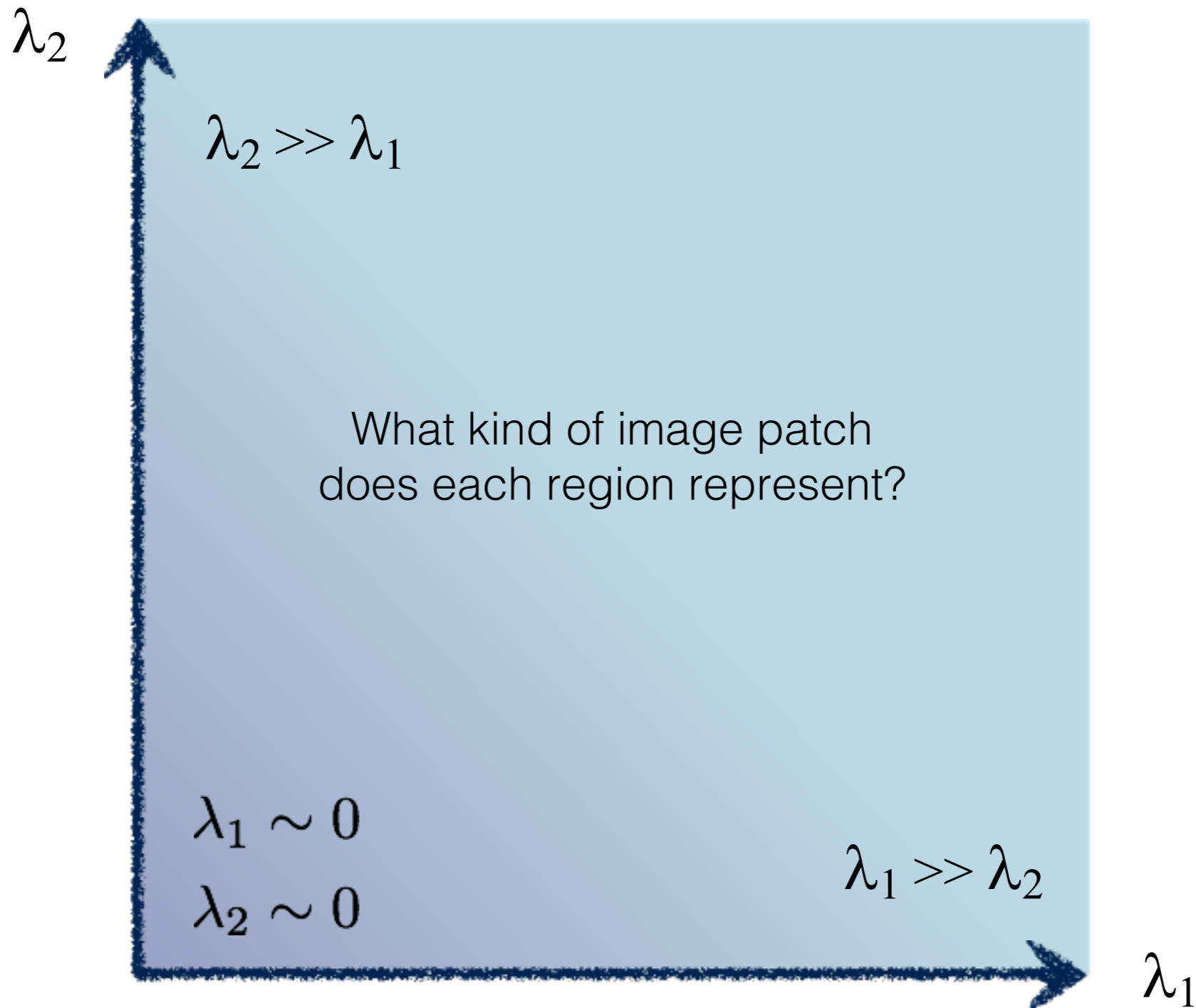
$$\mathbf{W}(\mathbf{x}; \mathbf{p}) = \begin{bmatrix} x + p_1 \\ y + p_2 \end{bmatrix} \quad \frac{\mathbf{W}}{\partial \mathbf{p}} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Hessian

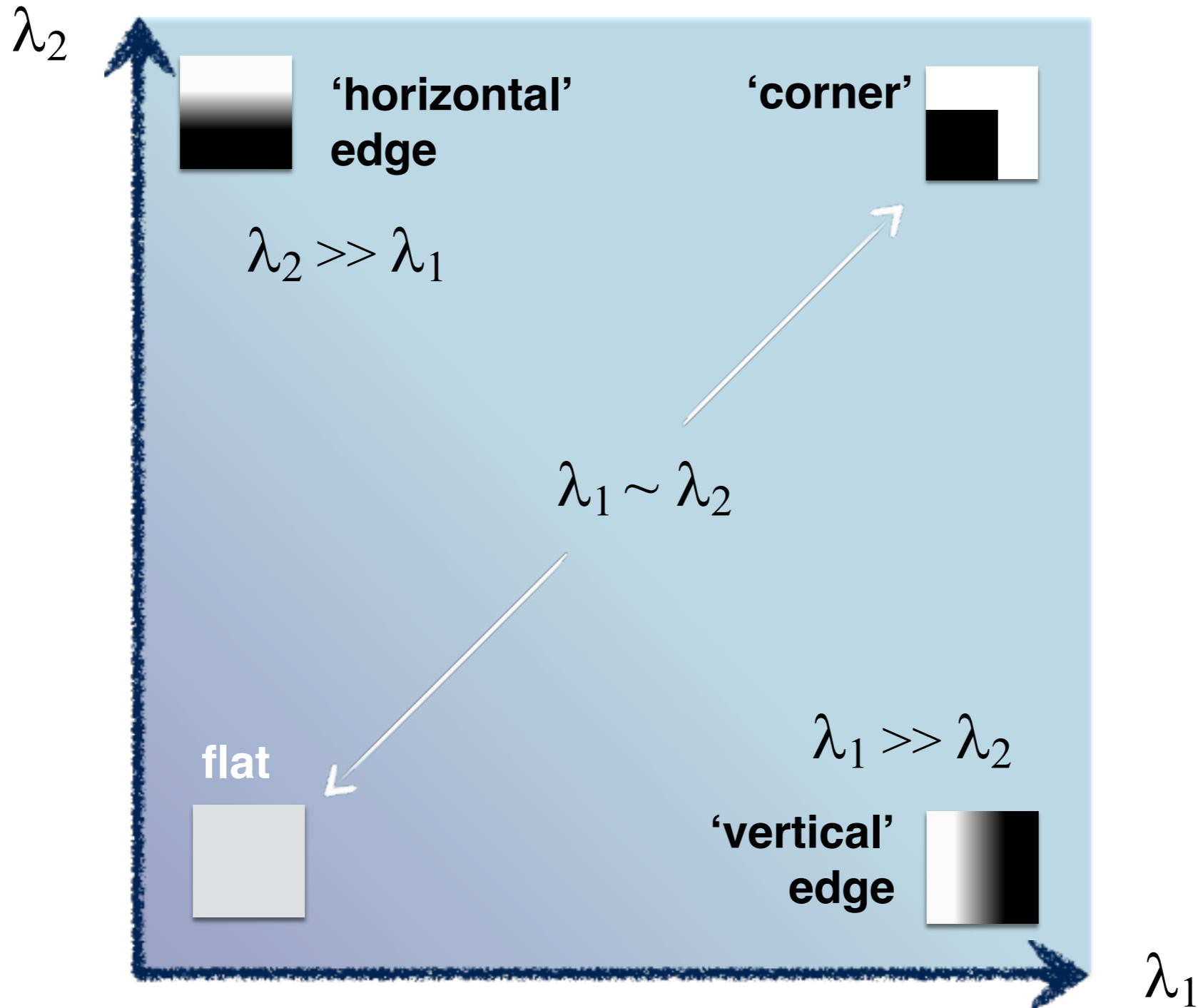
$$\begin{aligned} H &= \sum_{\mathbf{x}} \left[ \nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \right]^\top \left[ \nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \right] \\ &= \sum_{\mathbf{x}} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} I_x \\ I_y \end{bmatrix} \begin{bmatrix} I_x & I_y \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} \sum_{\mathbf{x}} I_x I_x & \sum_{\mathbf{x}} I_y I_x \\ \sum_{\mathbf{x}} I_x I_y & \sum_{\mathbf{x}} I_y I_y \end{bmatrix} \leftarrow \text{when is this singular?} \end{aligned}$$

*How are the eigenvalues related to image content?*

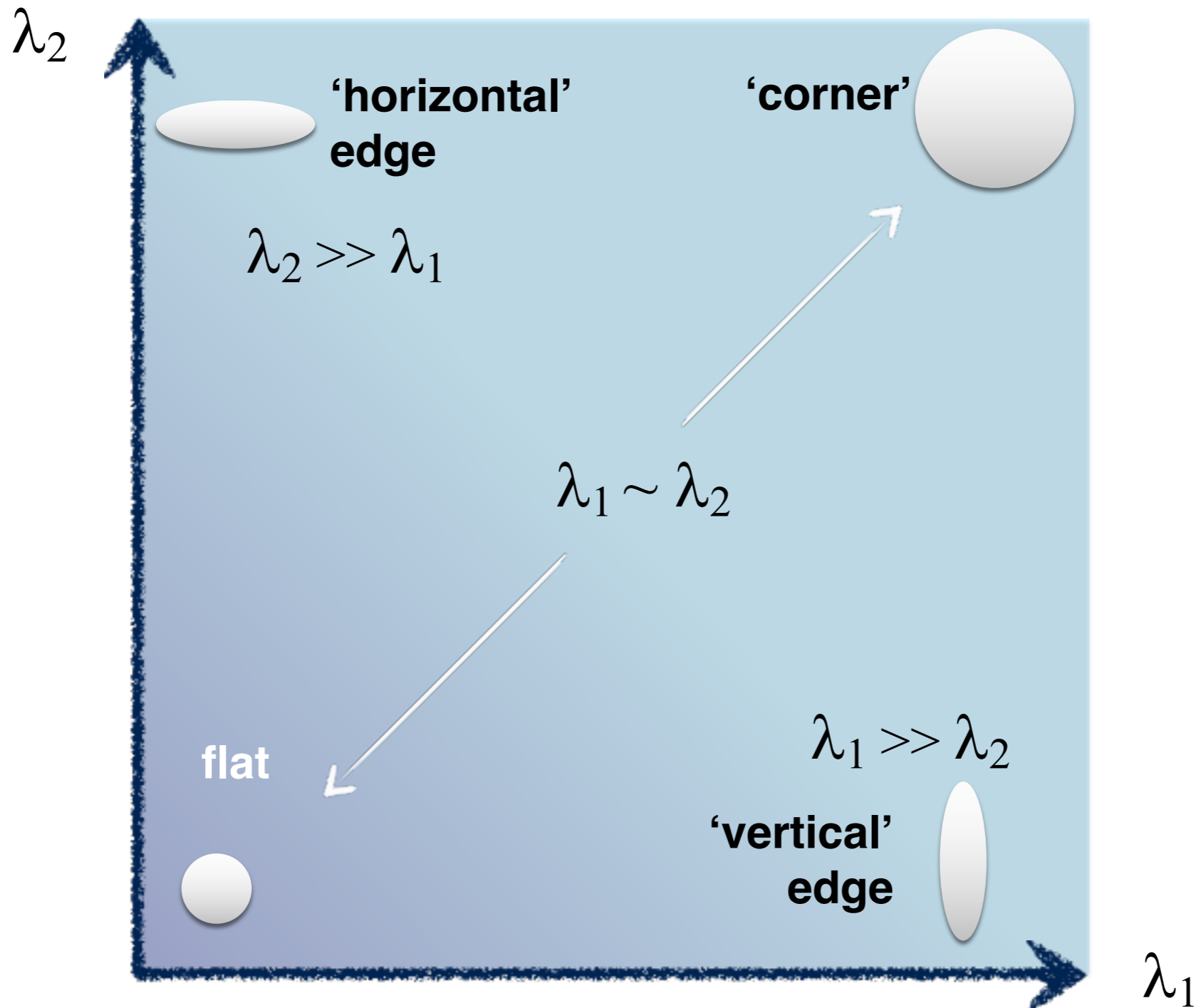
# interpreting eigenvalues



# 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





PORTUGOOL

Nationale-Nederlanden

jackpot €5 milj.

ING

# Mean Shift Algorithm

A 'mode seeking' algorithm

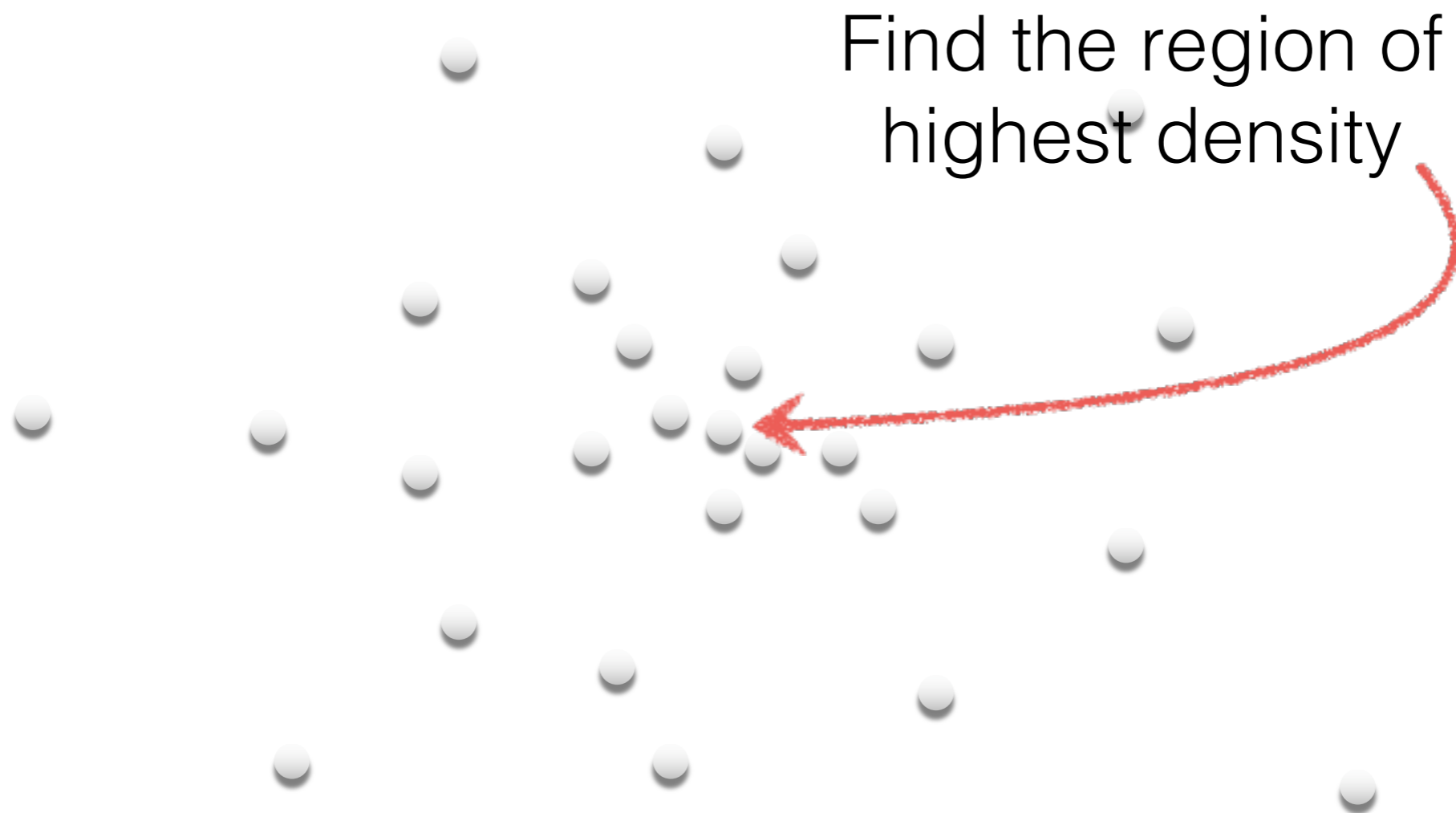
Fukunaga & Hostetler (1975)



# Mean Shift Algorithm

A 'mode seeking' algorithm

Fukunaga & Hostetler (1975)



# Mean Shift Algorithm

A 'mode seeking' algorithm

Fukunaga & Hostetler (1975)

Pick a point

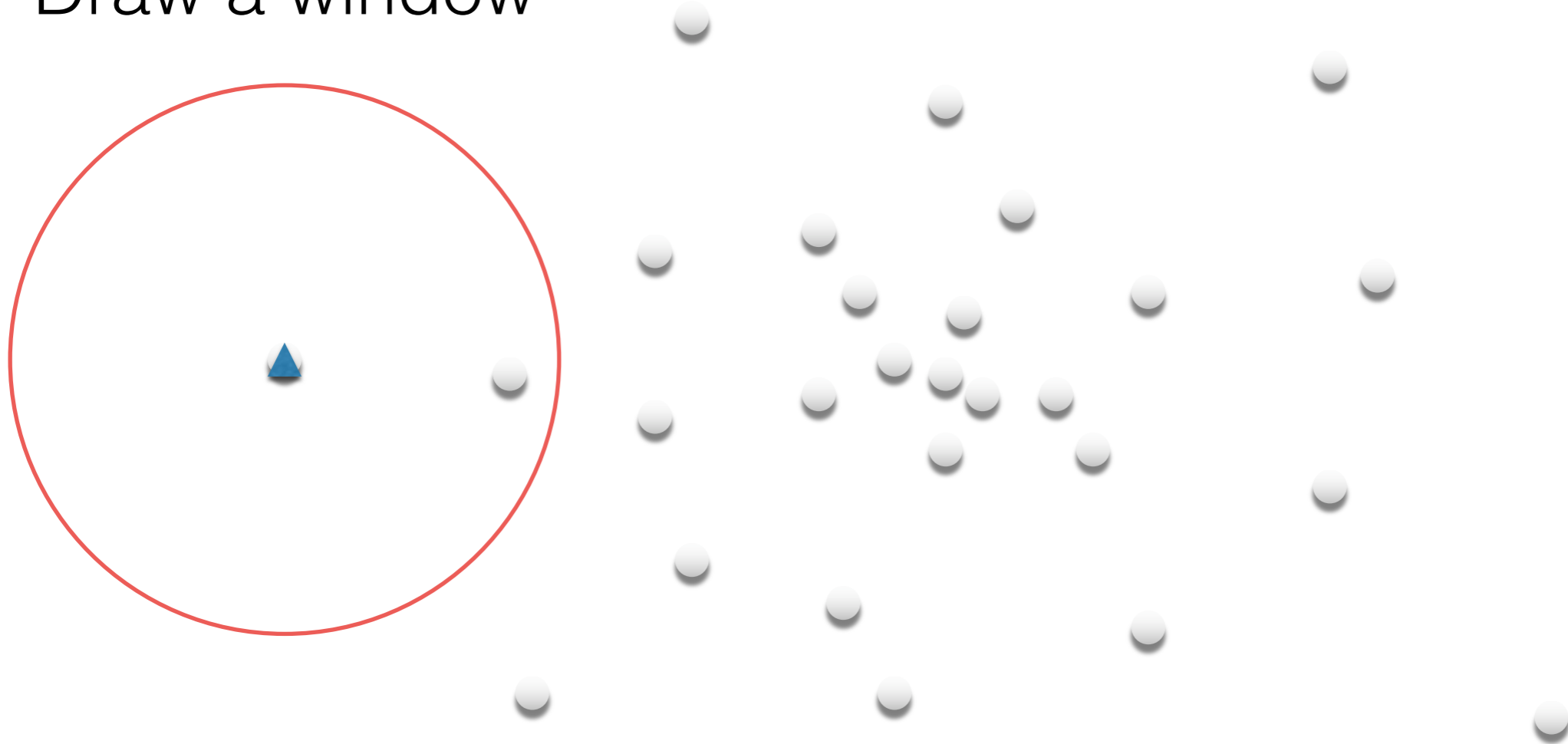


# Mean Shift Algorithm

A 'mode seeking' algorithm

Fukunaga & Hostetler (1975)

Draw a window

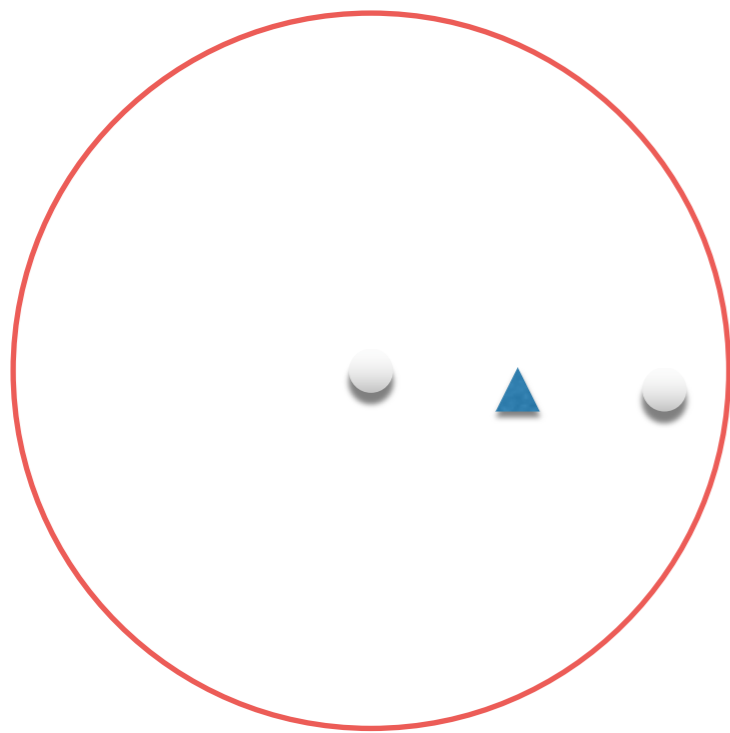


# Mean Shift Algorithm

A 'mode seeking' algorithm

Fukunaga & Hostetler (1975)

Compute the  
(weighted) **mean**

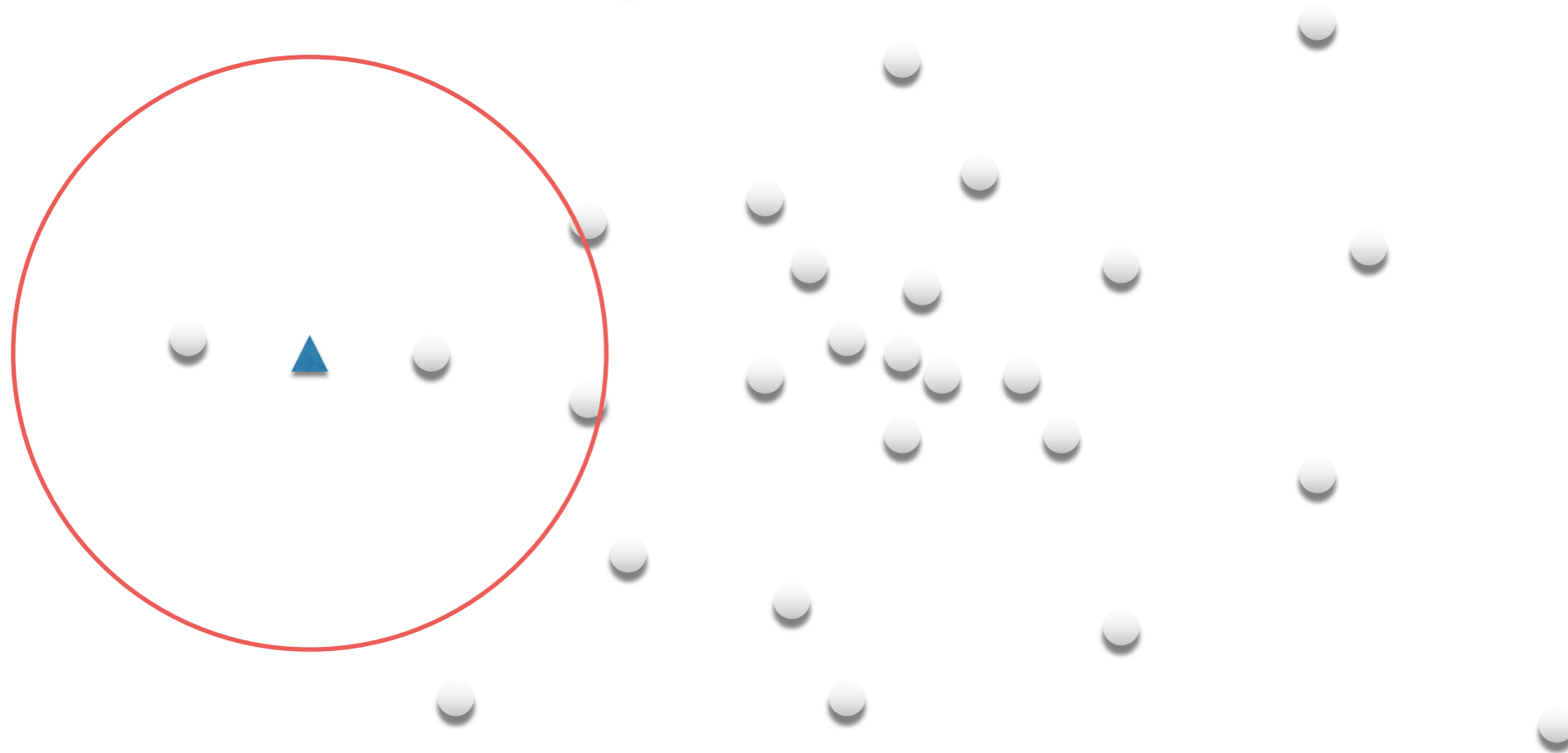


# Mean Shift Algorithm

A 'mode seeking' algorithm

Fukunaga & Hostetler (1975)

**Shift** the window

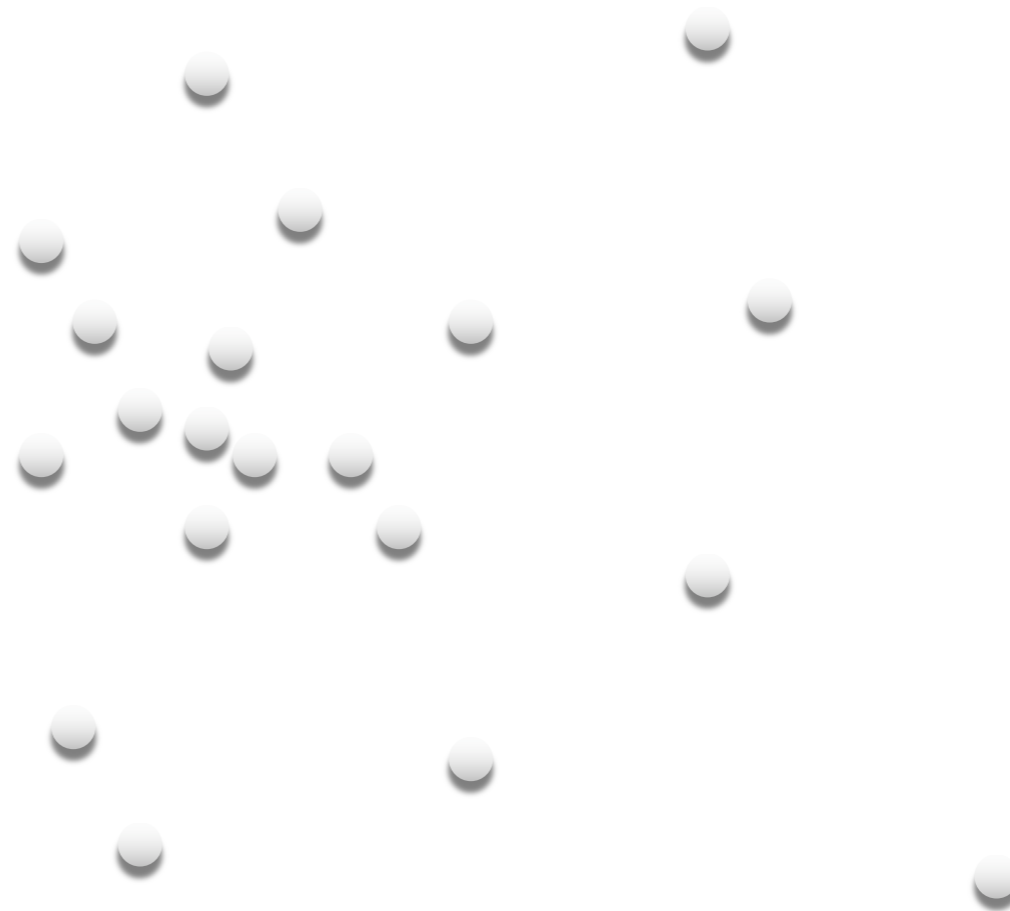
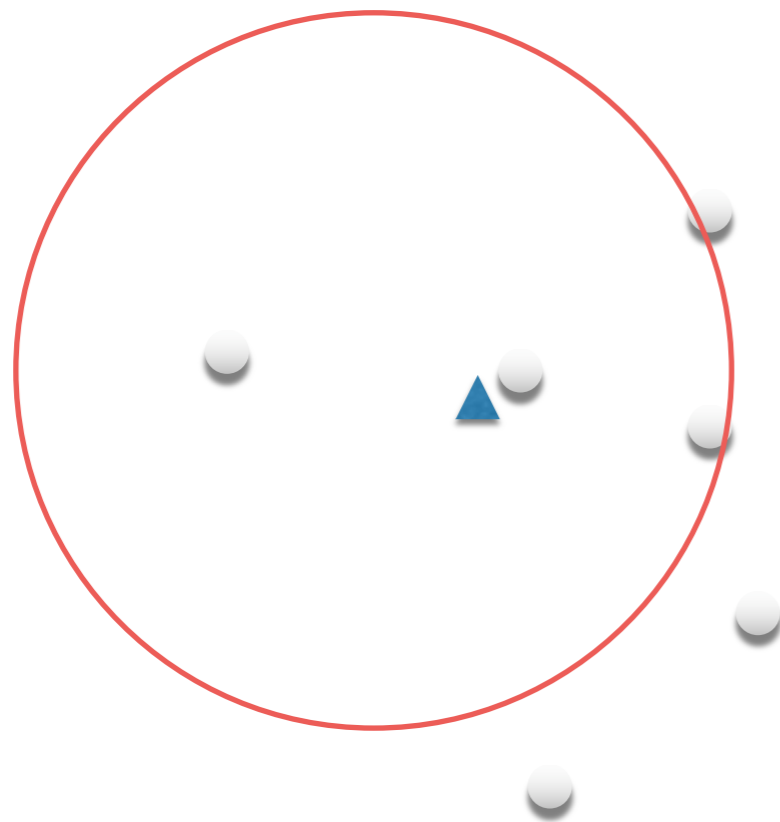


# Mean Shift Algorithm

A 'mode seeking' algorithm

Fukunaga & Hostetler (1975)

Compute the **mean**

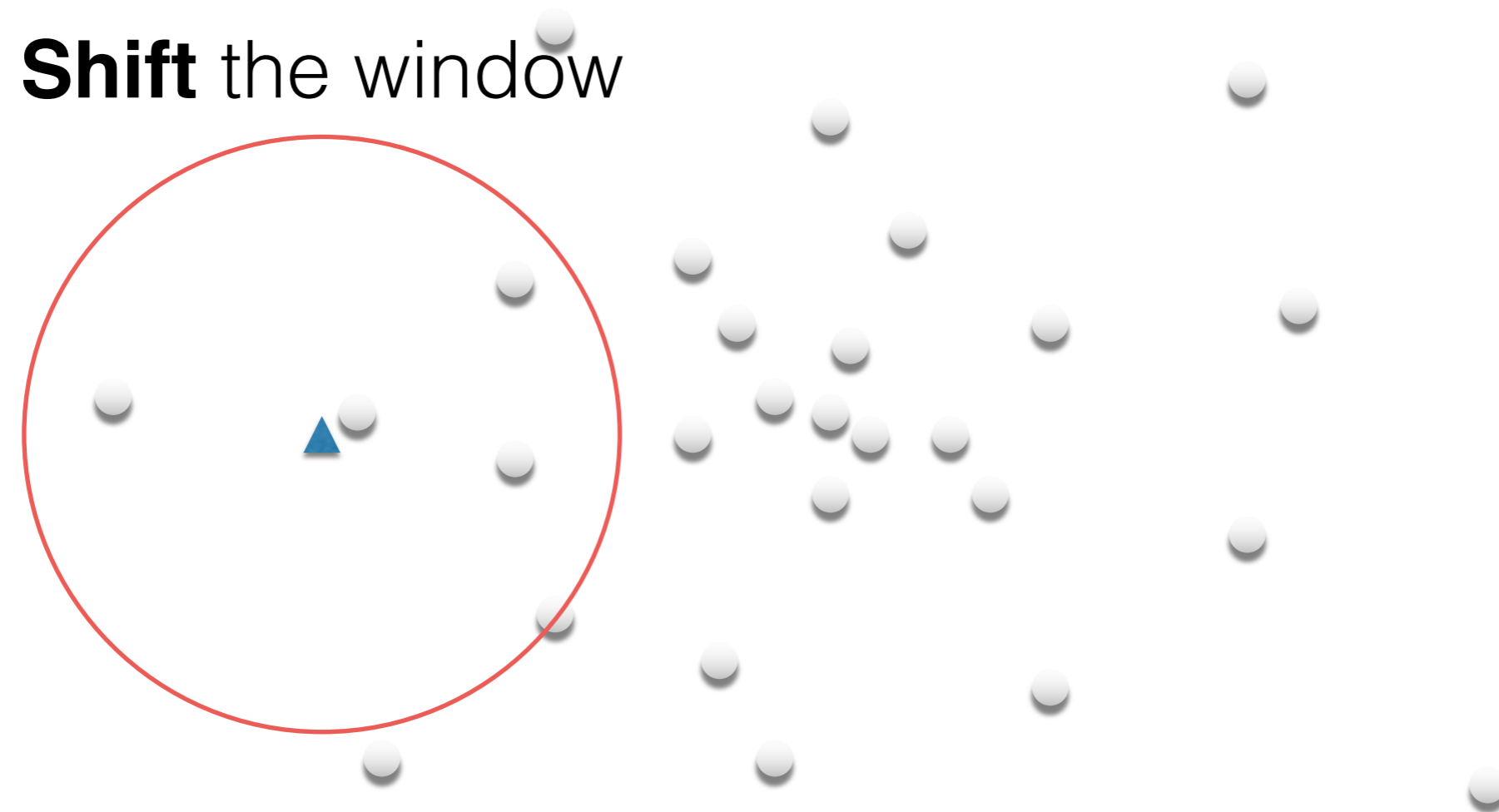




# Mean Shift Algorithm

A 'mode seeking' algorithm

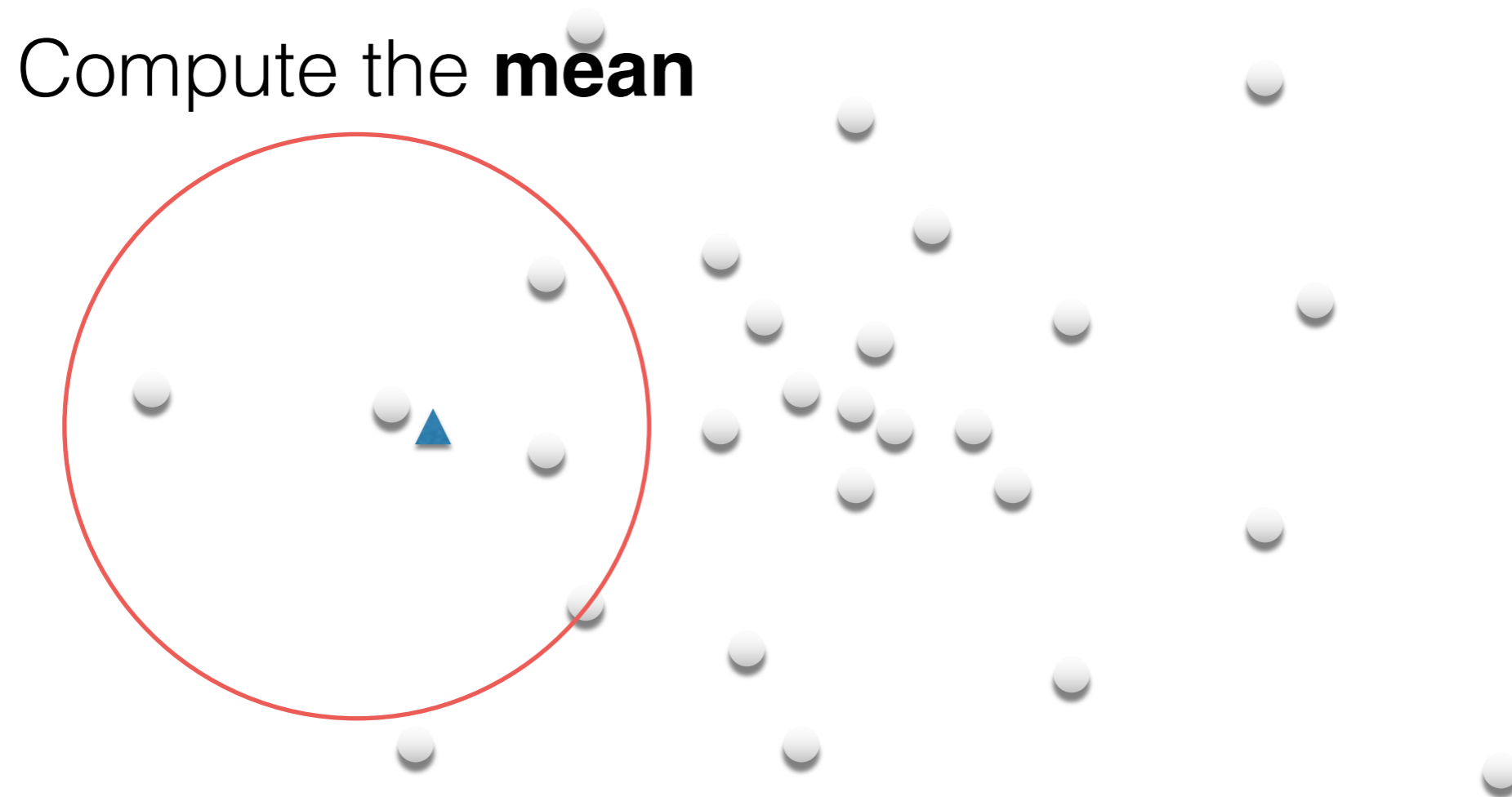
Fukunaga & Hostetler (1975)



# Mean Shift Algorithm

A 'mode seeking' algorithm

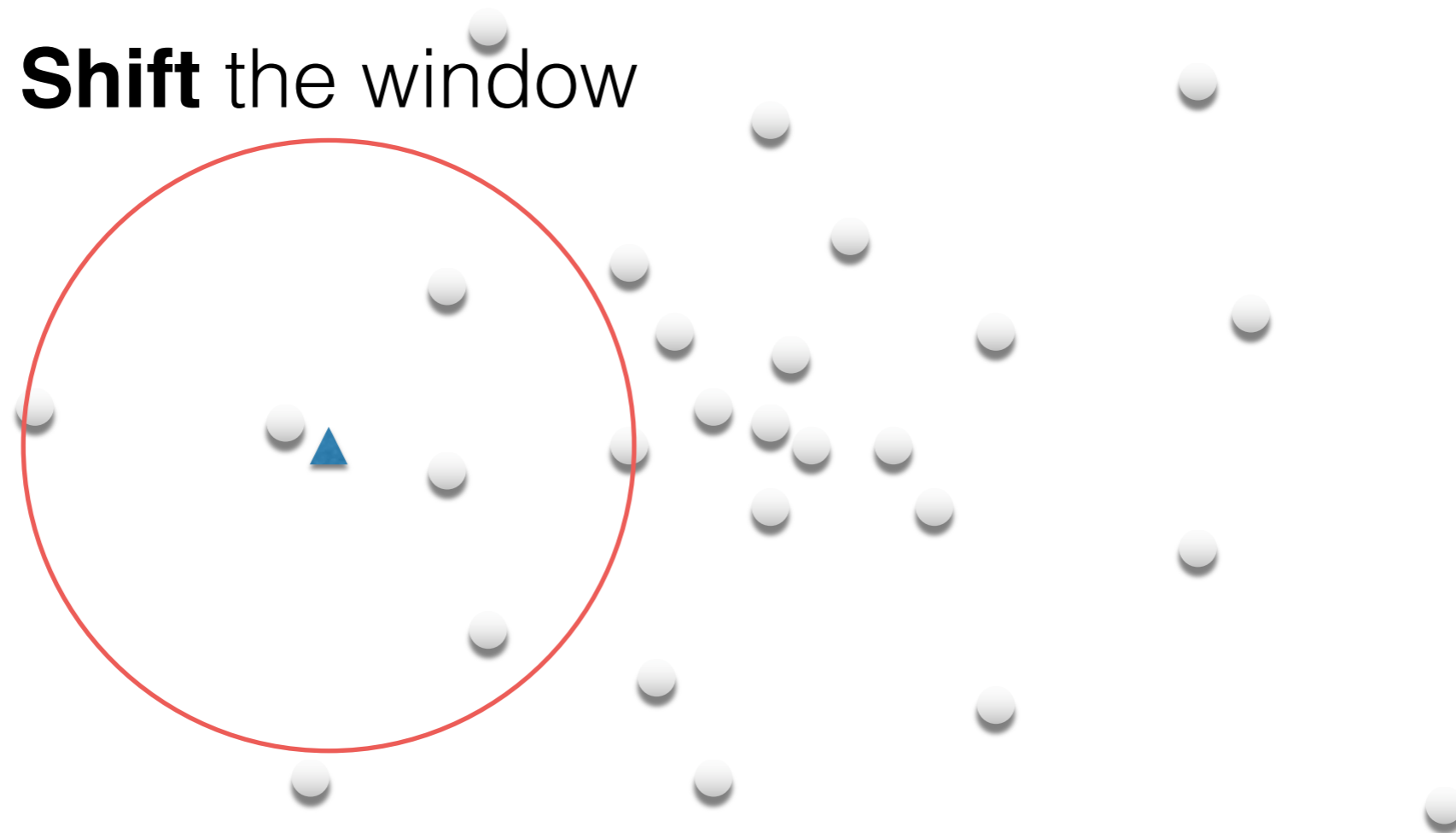
Fukunaga & Hostetler (1975)



# Mean Shift Algorithm

A 'mode seeking' algorithm

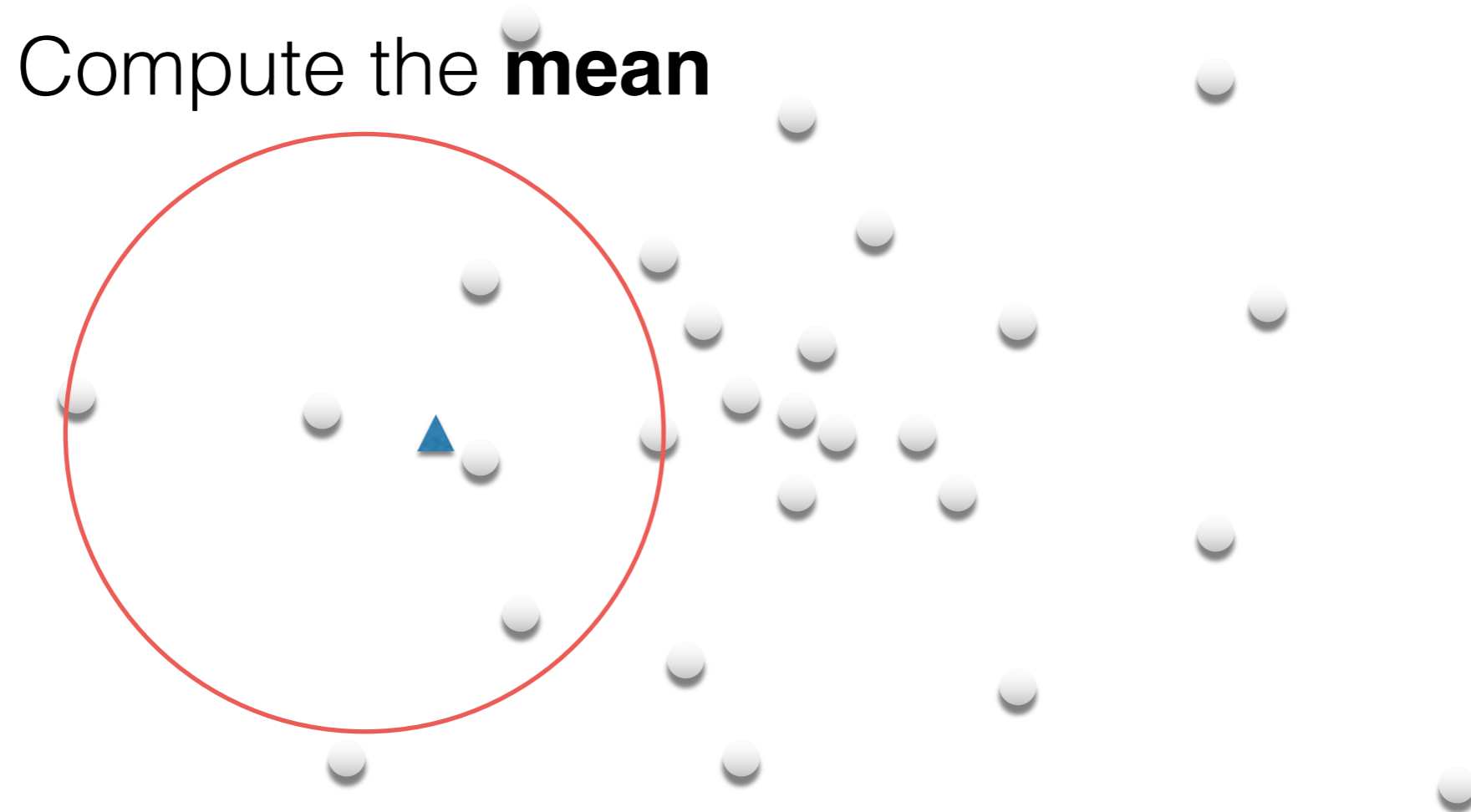
Fukunaga & Hostetler (1975)



# Mean Shift Algorithm

A 'mode seeking' algorithm

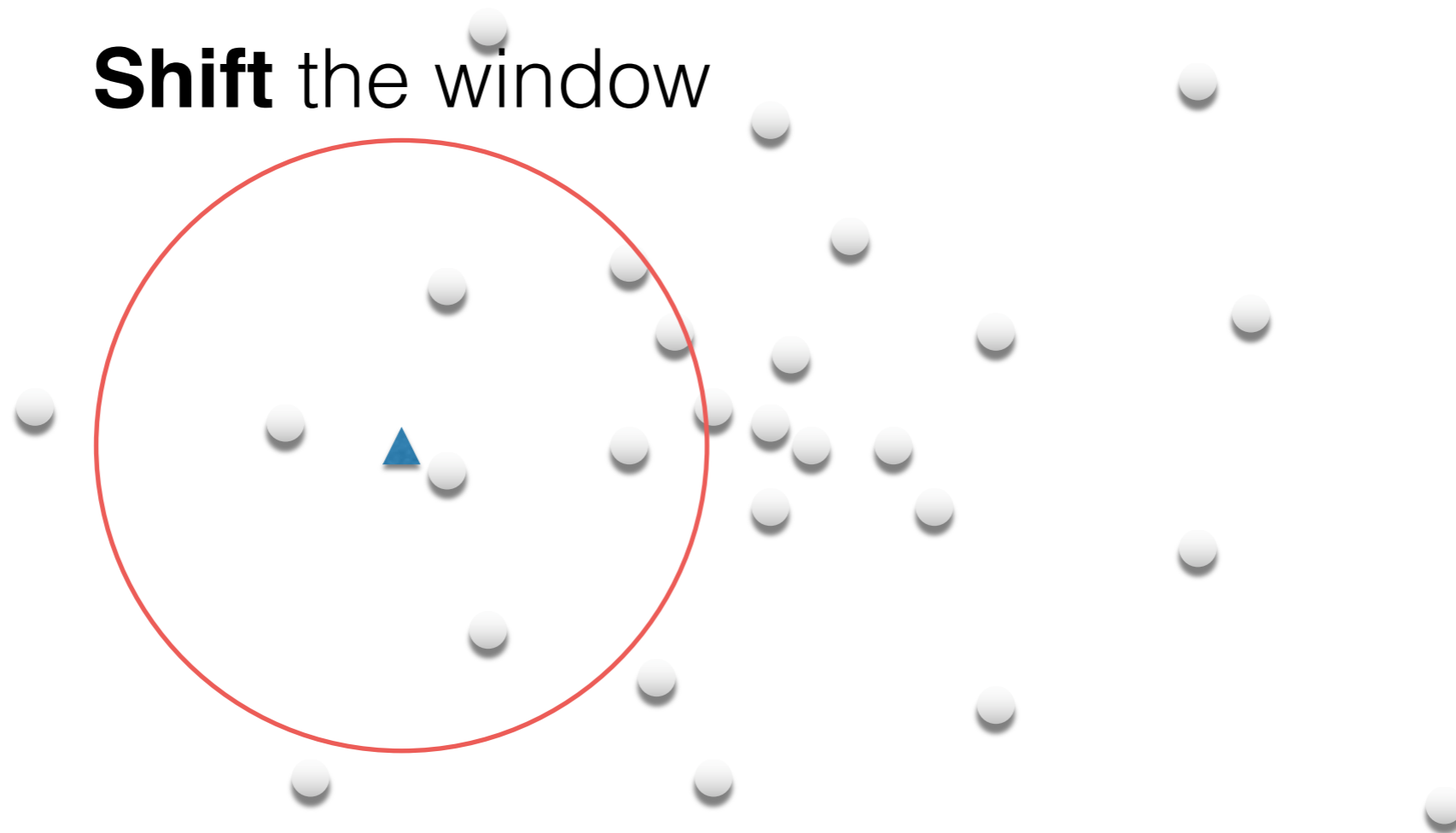
Fukunaga & Hostetler (1975)



# Mean Shift Algorithm

A 'mode seeking' algorithm

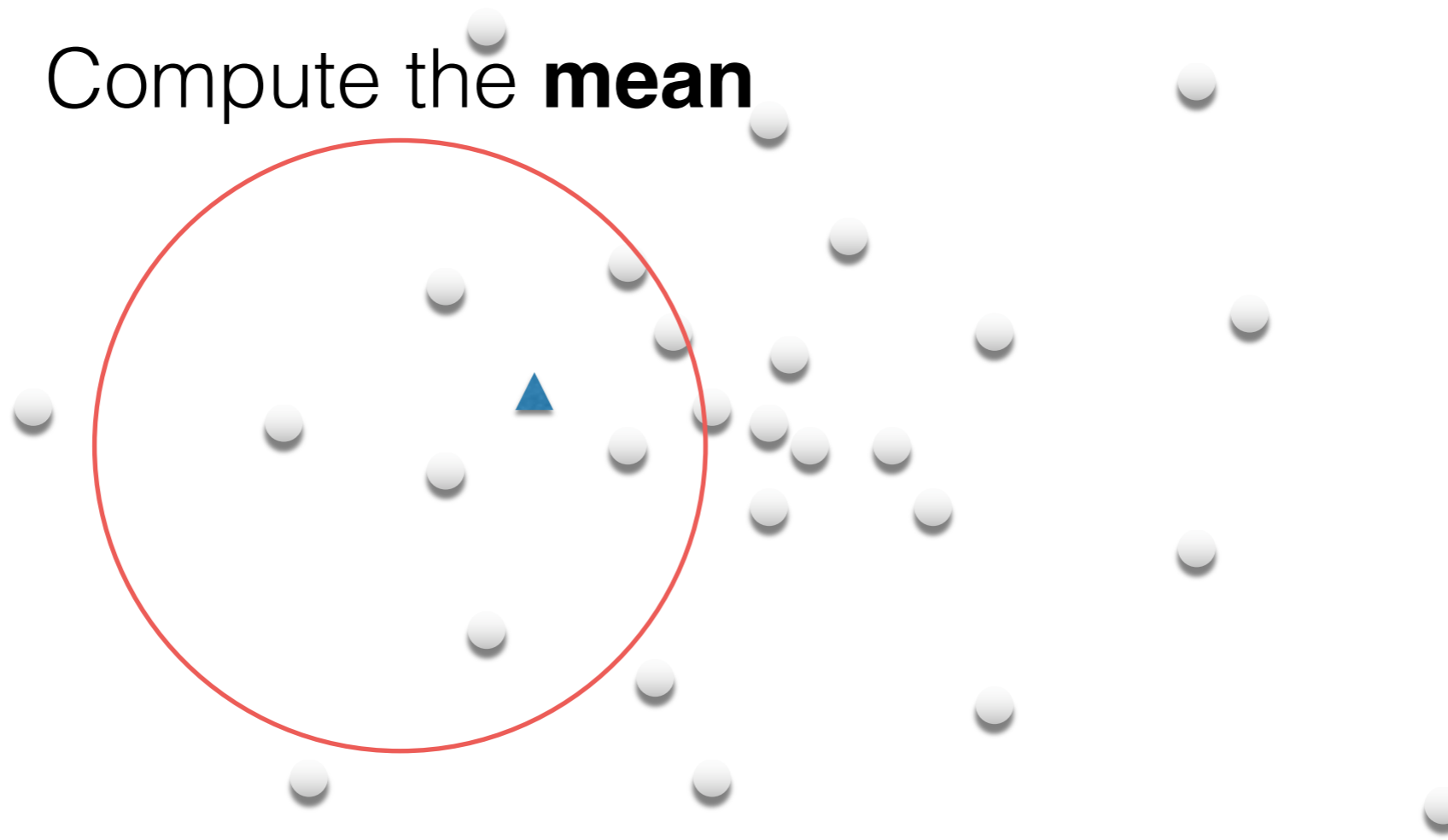
Fukunaga & Hostetler (1975)



# Mean Shift Algorithm

A 'mode seeking' algorithm

Fukunaga & Hostetler (1975)



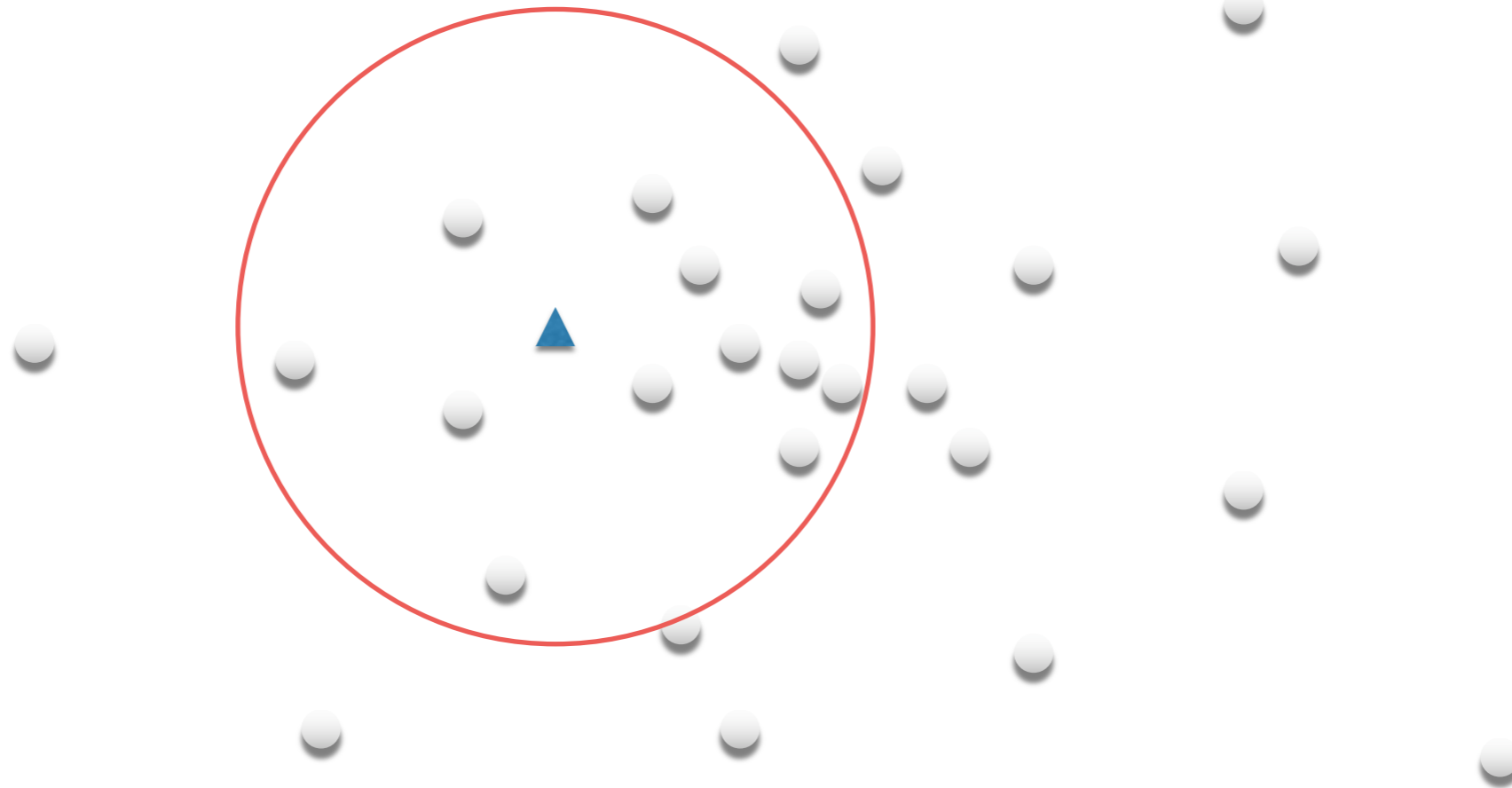
Compute the **mean**

# Mean Shift Algorithm

A 'mode seeking' algorithm

Fukunaga & Hostetler (1975)

**Shift** the window

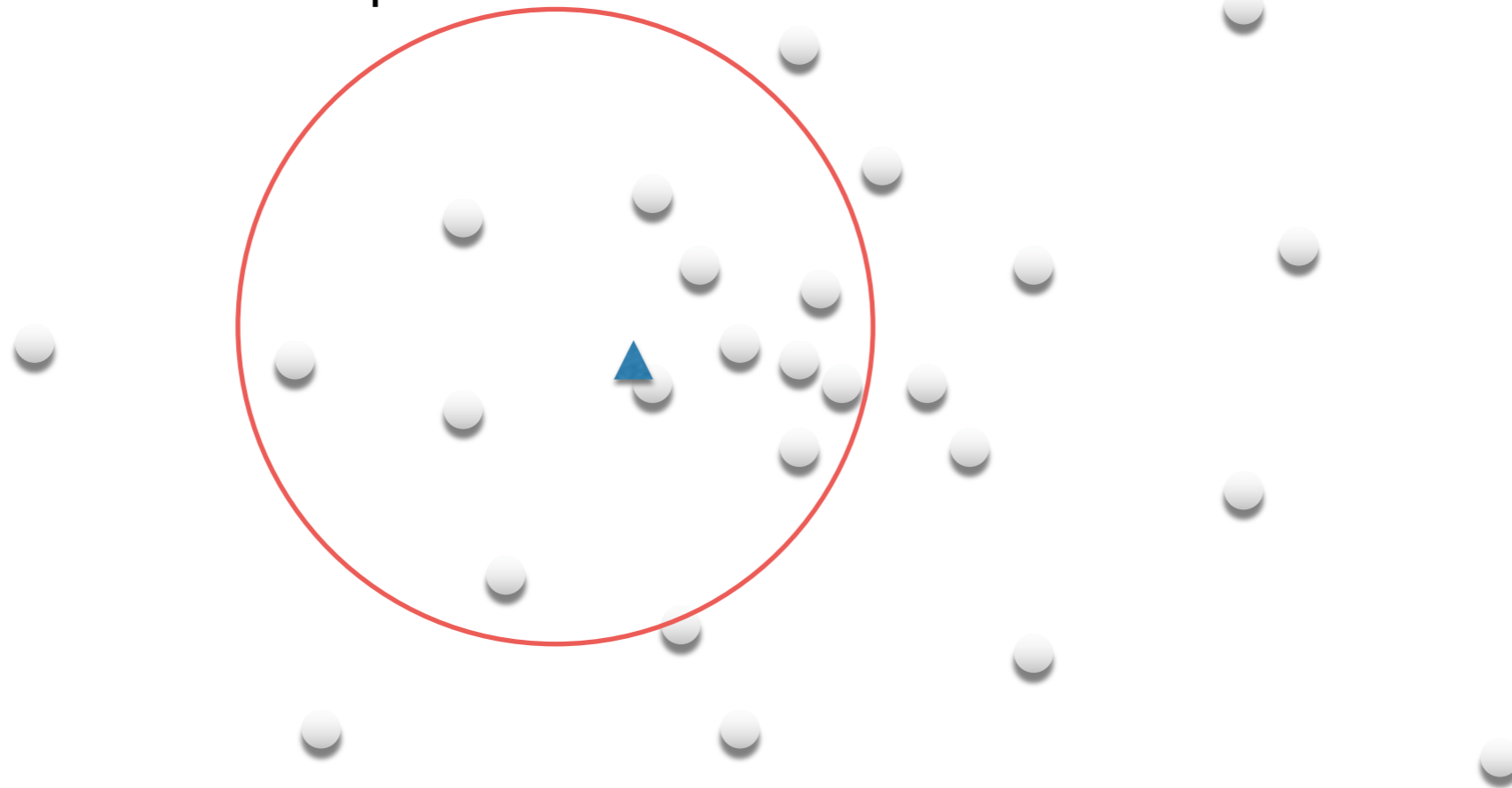


# Mean Shift Algorithm

A 'mode seeking' algorithm

Fukunaga & Hostetler (1975)

Compute the **mean**

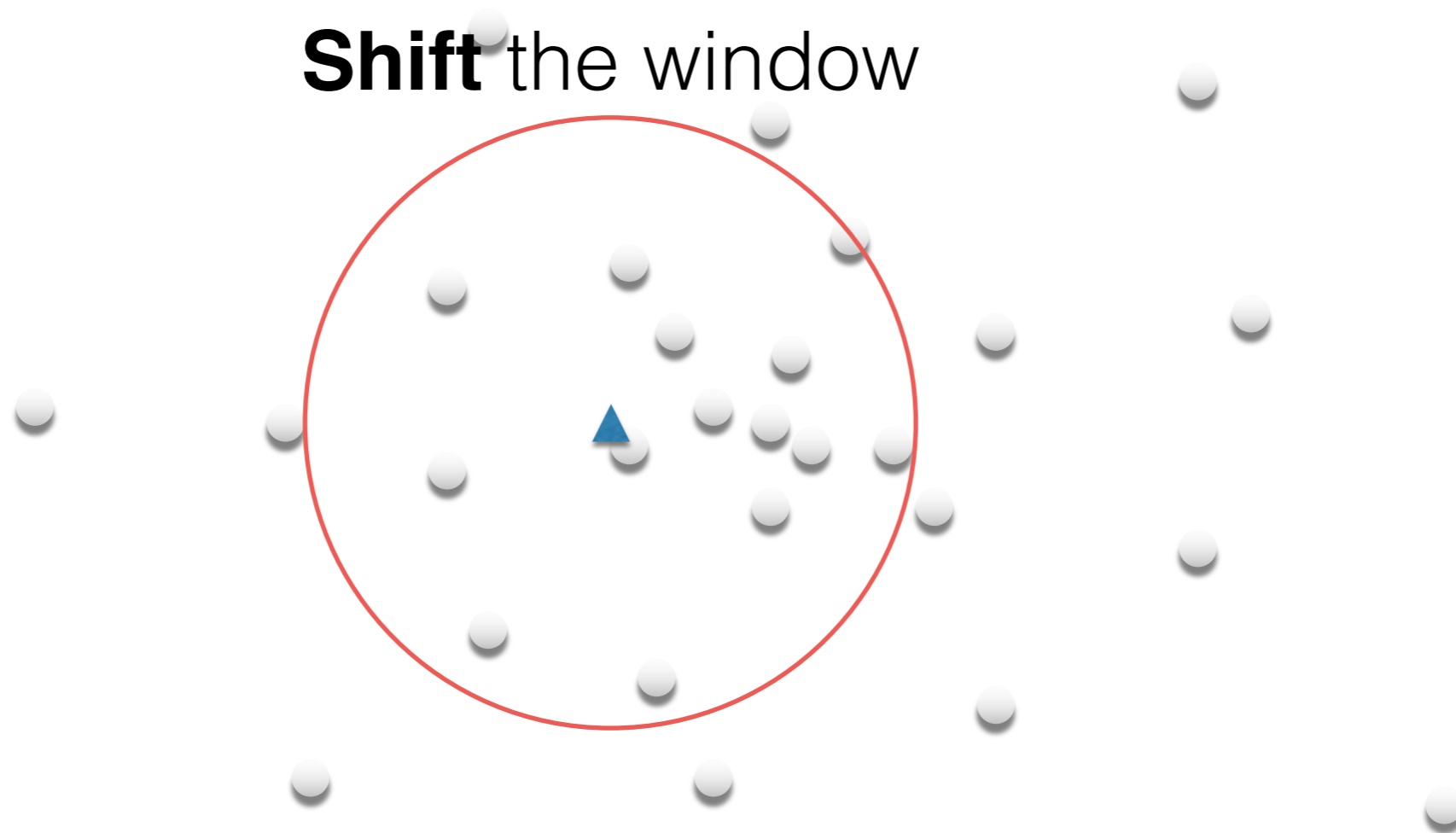




# Mean Shift Algorithm

A 'mode seeking' algorithm

Fukunaga & Hostetler (1975)

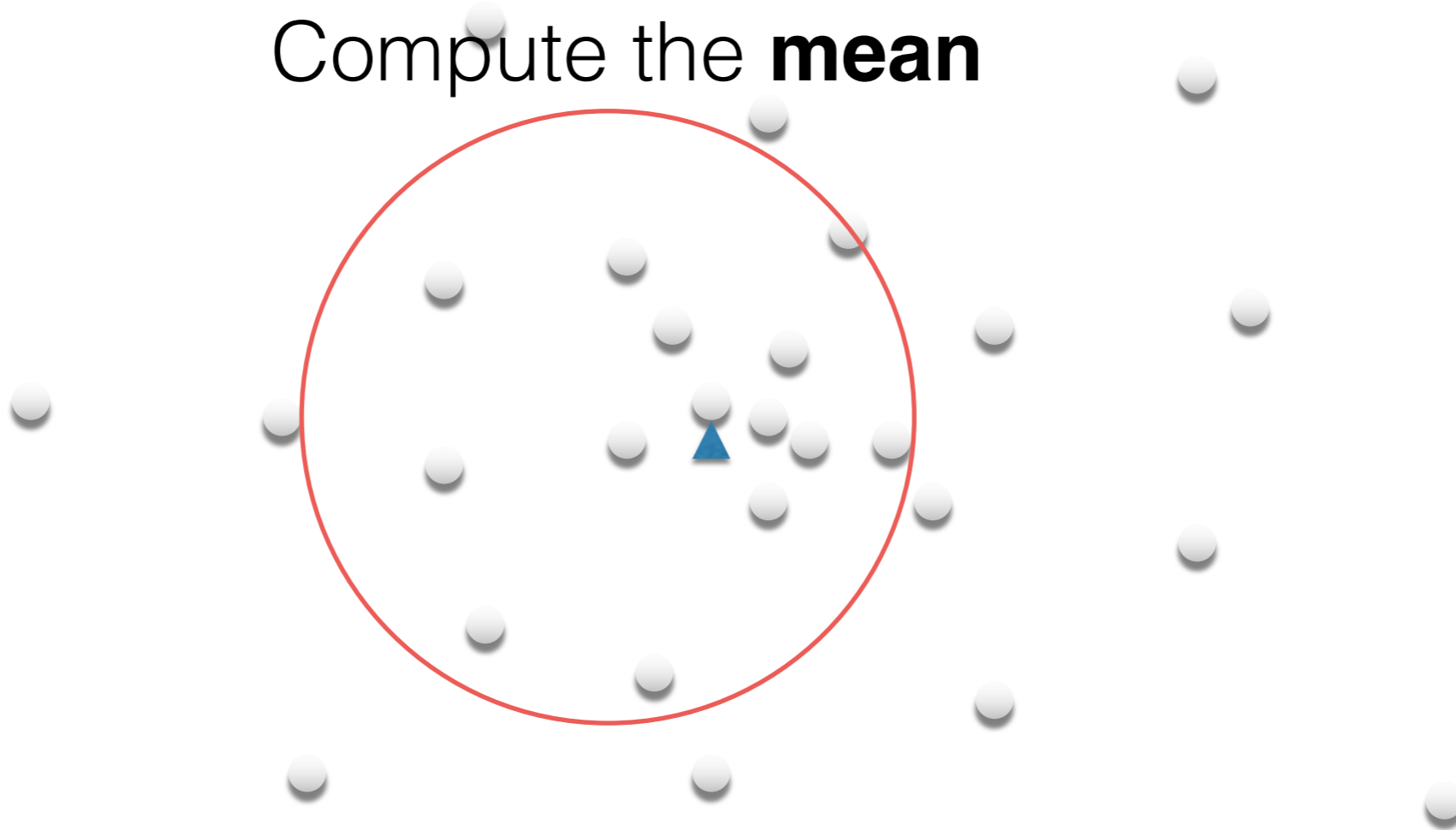


# Mean Shift Algorithm

A 'mode seeking' algorithm

Fukunaga & Hostetler (1975)

Compute the **mean**

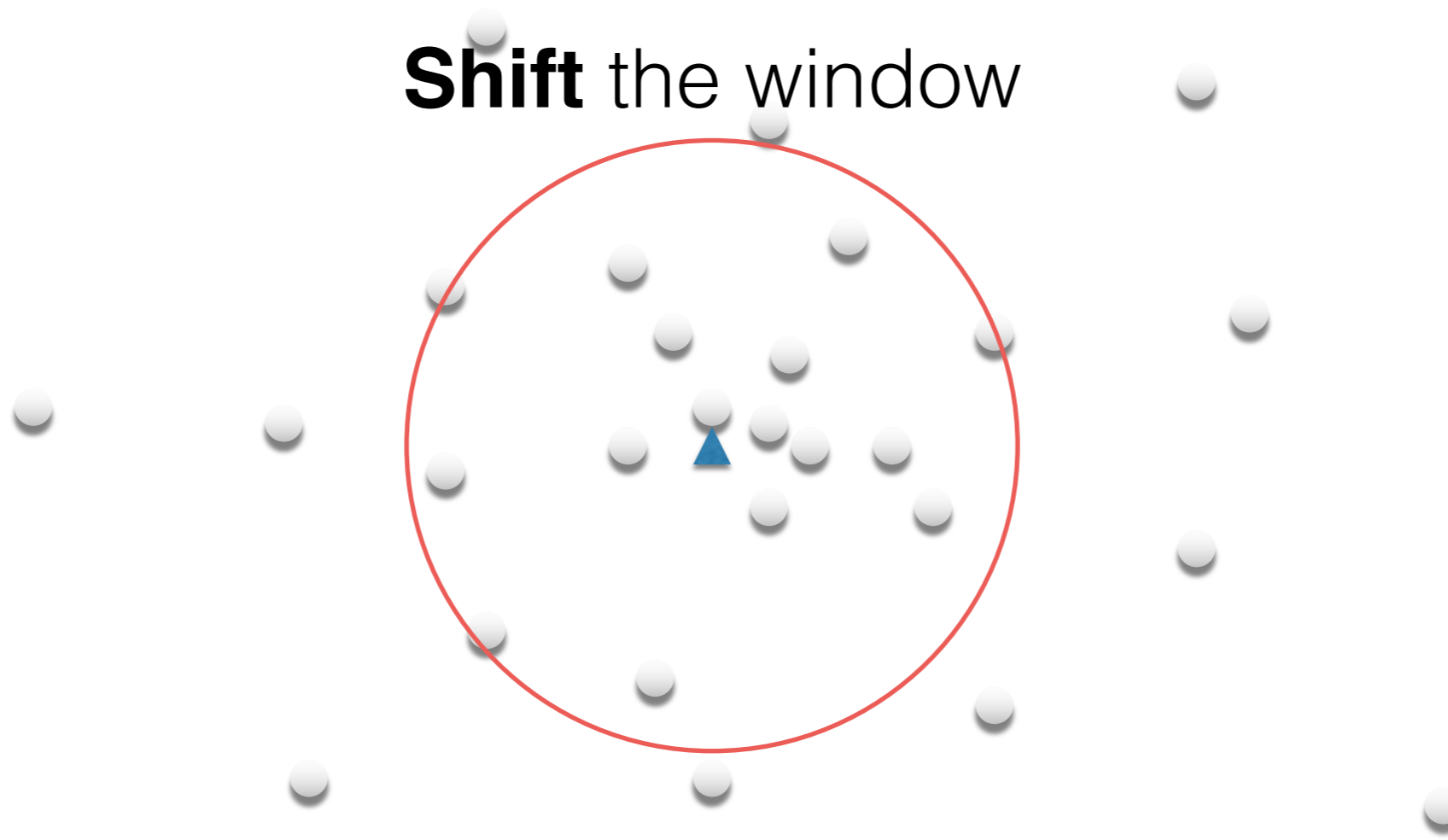


# Mean Shift Algorithm

A 'mode seeking' algorithm

Fukunaga & Hostetler (1975)

**Shift** the window

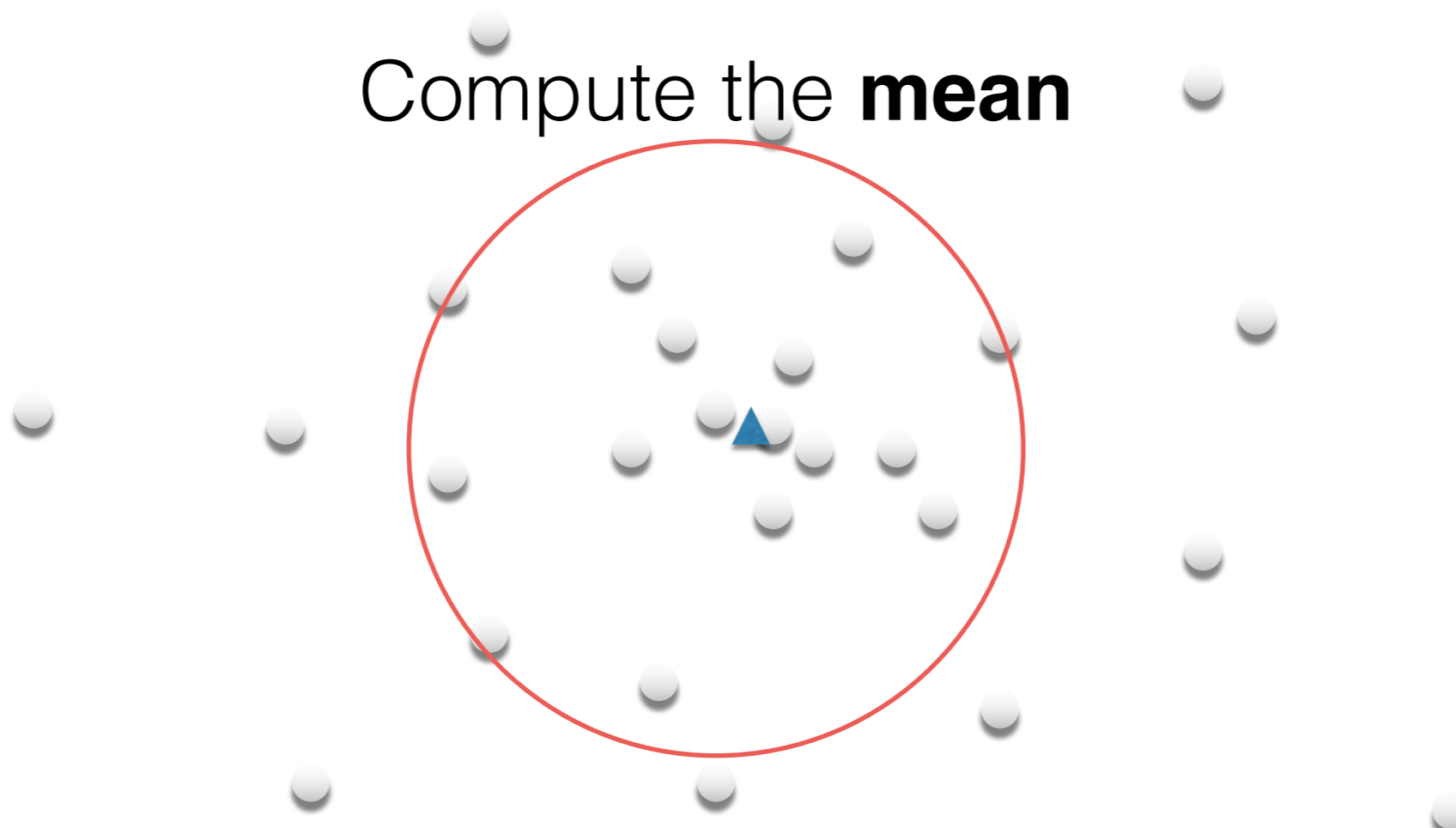


# Mean Shift Algorithm

A 'mode seeking' algorithm

Fukunaga & Hostetler (1975)

Compute the **mean**

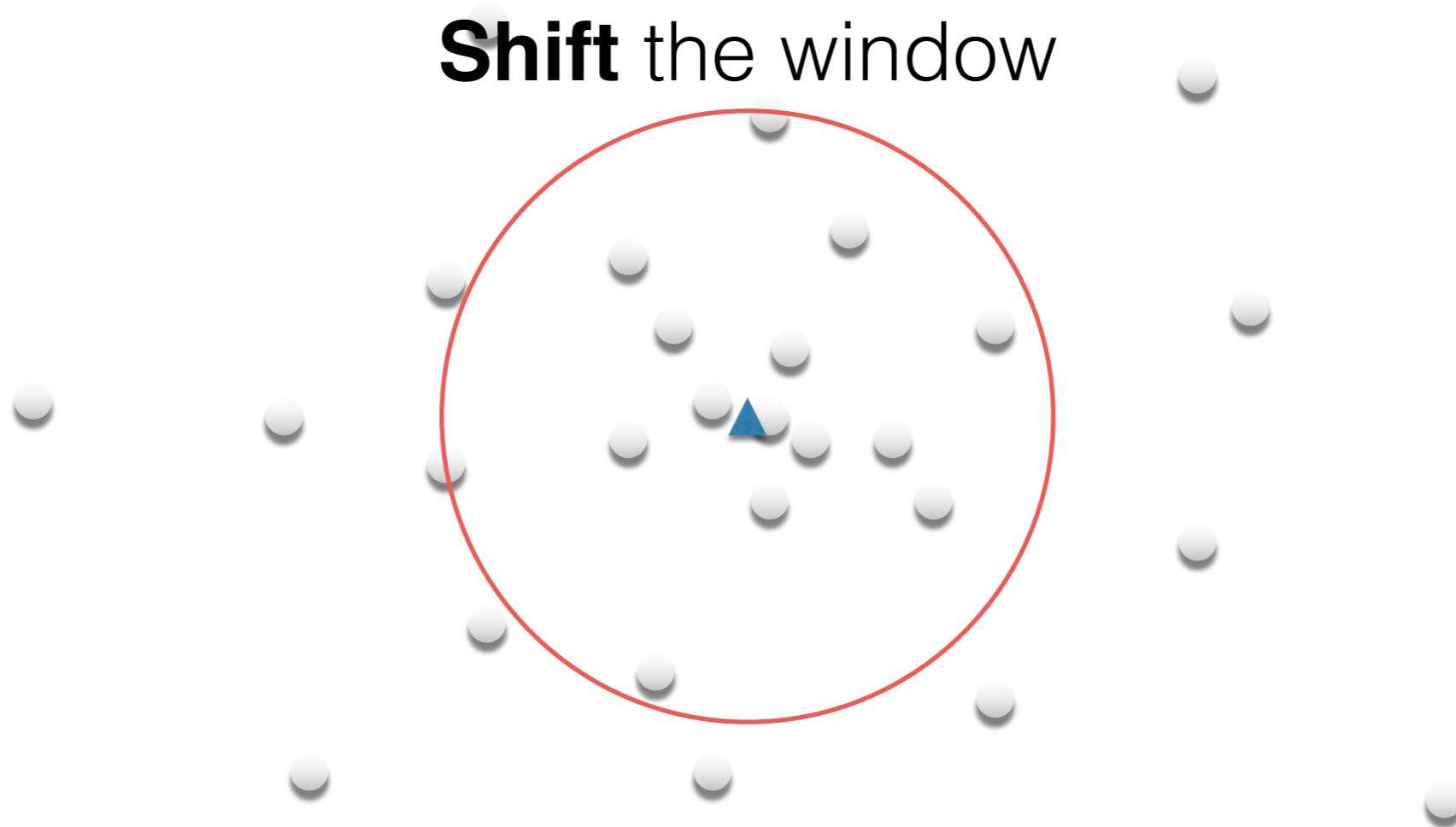


# Mean Shift Algorithm

A 'mode seeking' algorithm

Fukunaga & Hostetler (1975)

**Shift** the window



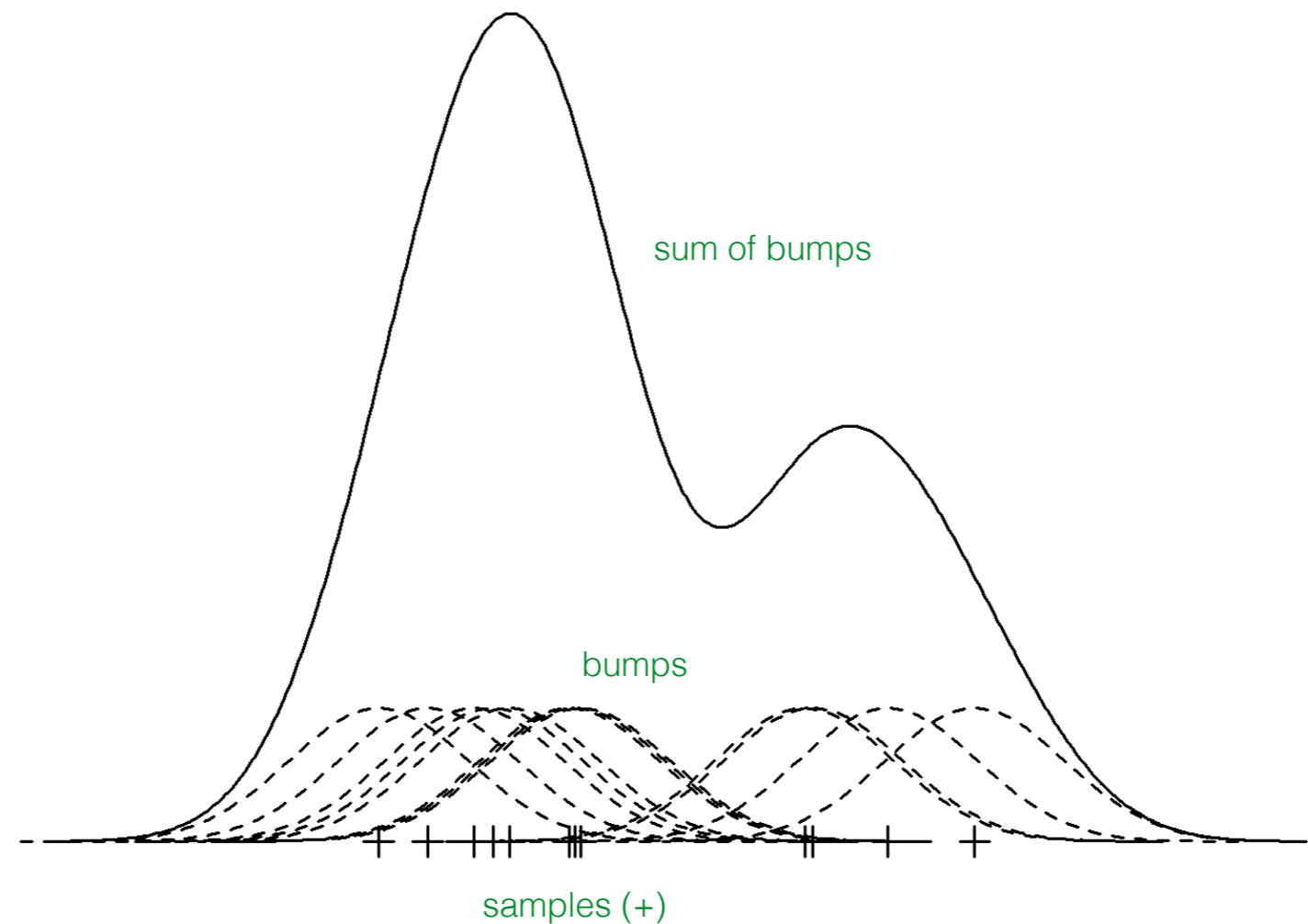
To understand the theory behind this we need to understand...

# 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

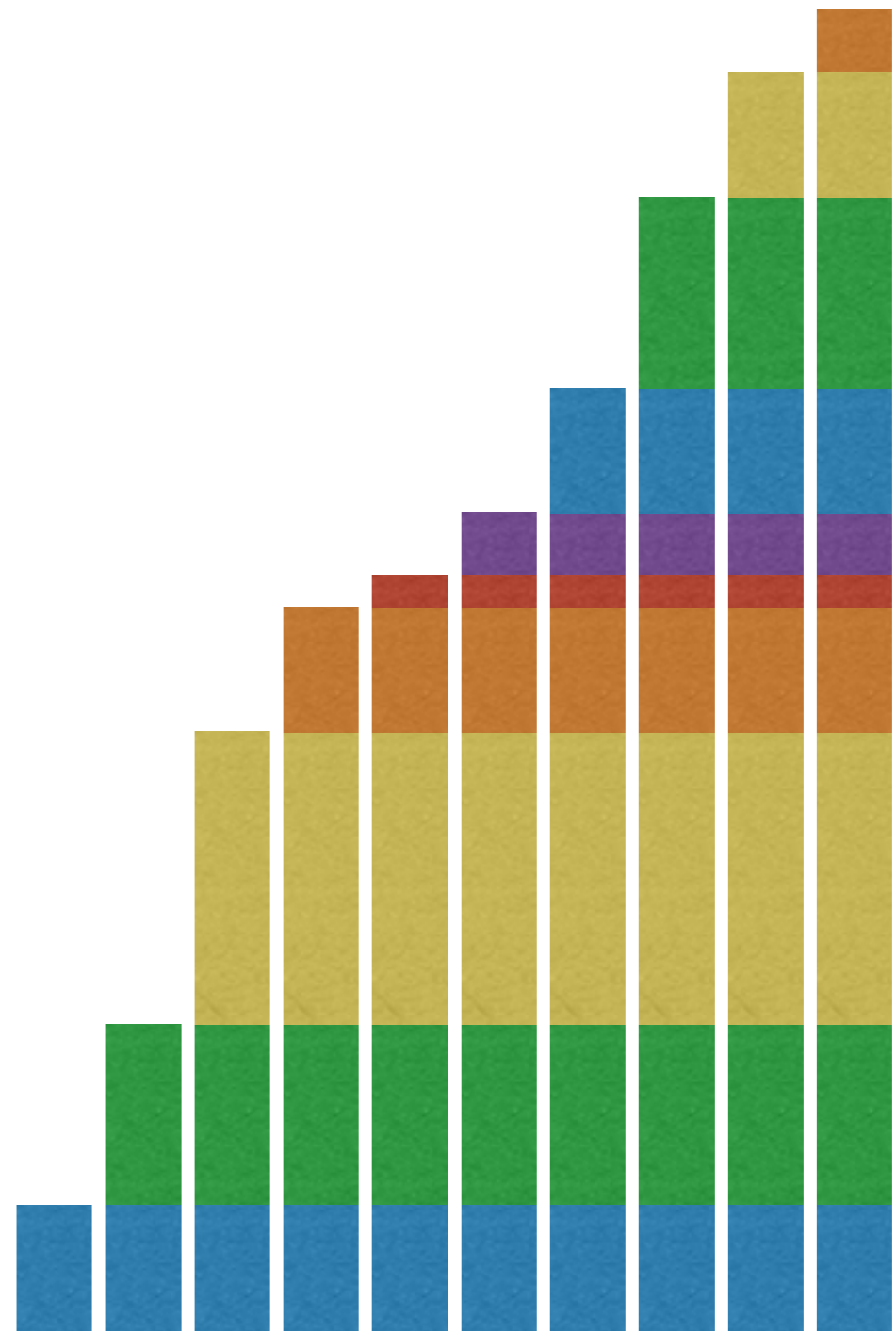
Say we have some hidden PDF...

$p(x)$



1 2 3 4 5 6 7 8 9 10

probability density function



cumulative density function

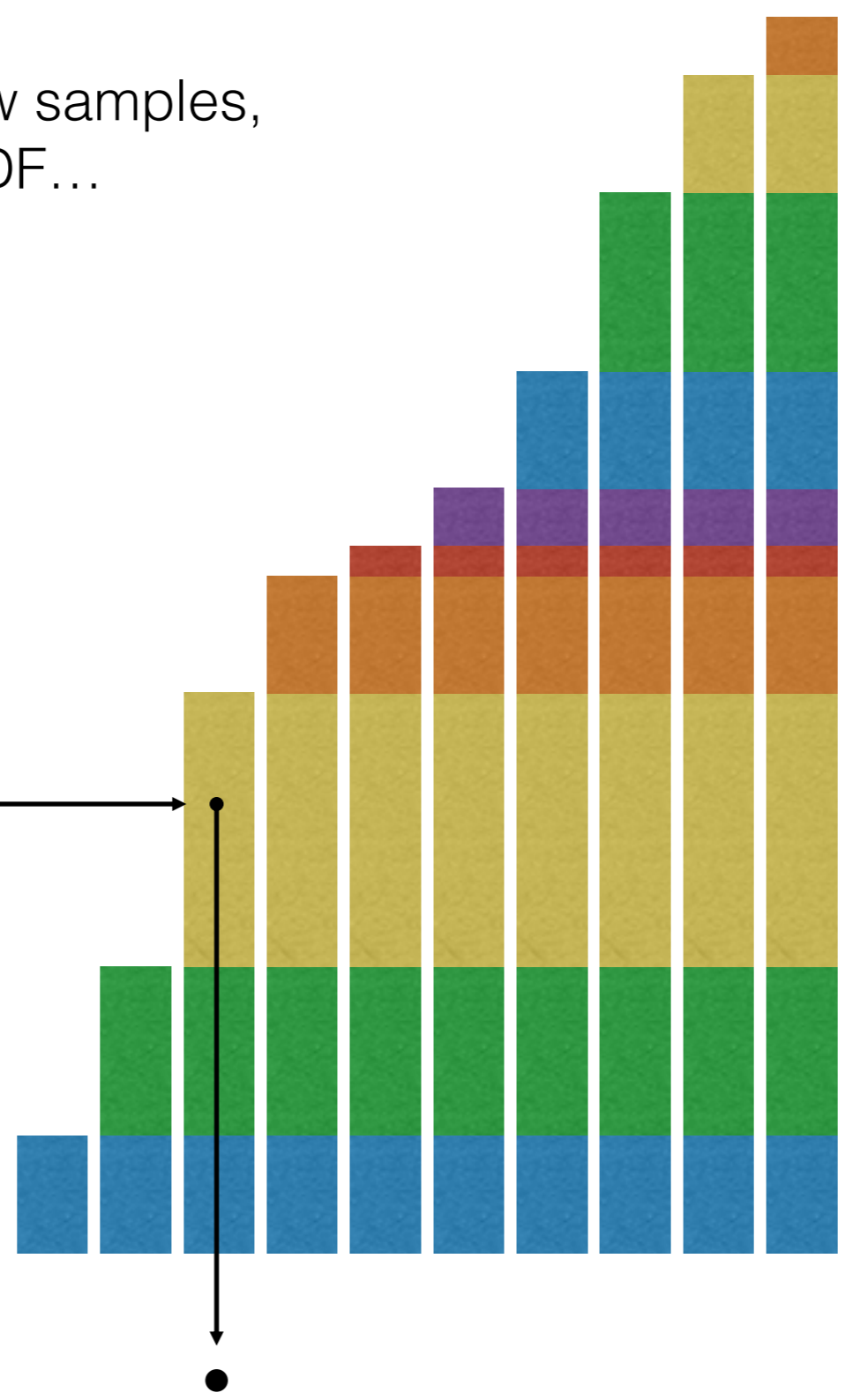


randomly sample

1

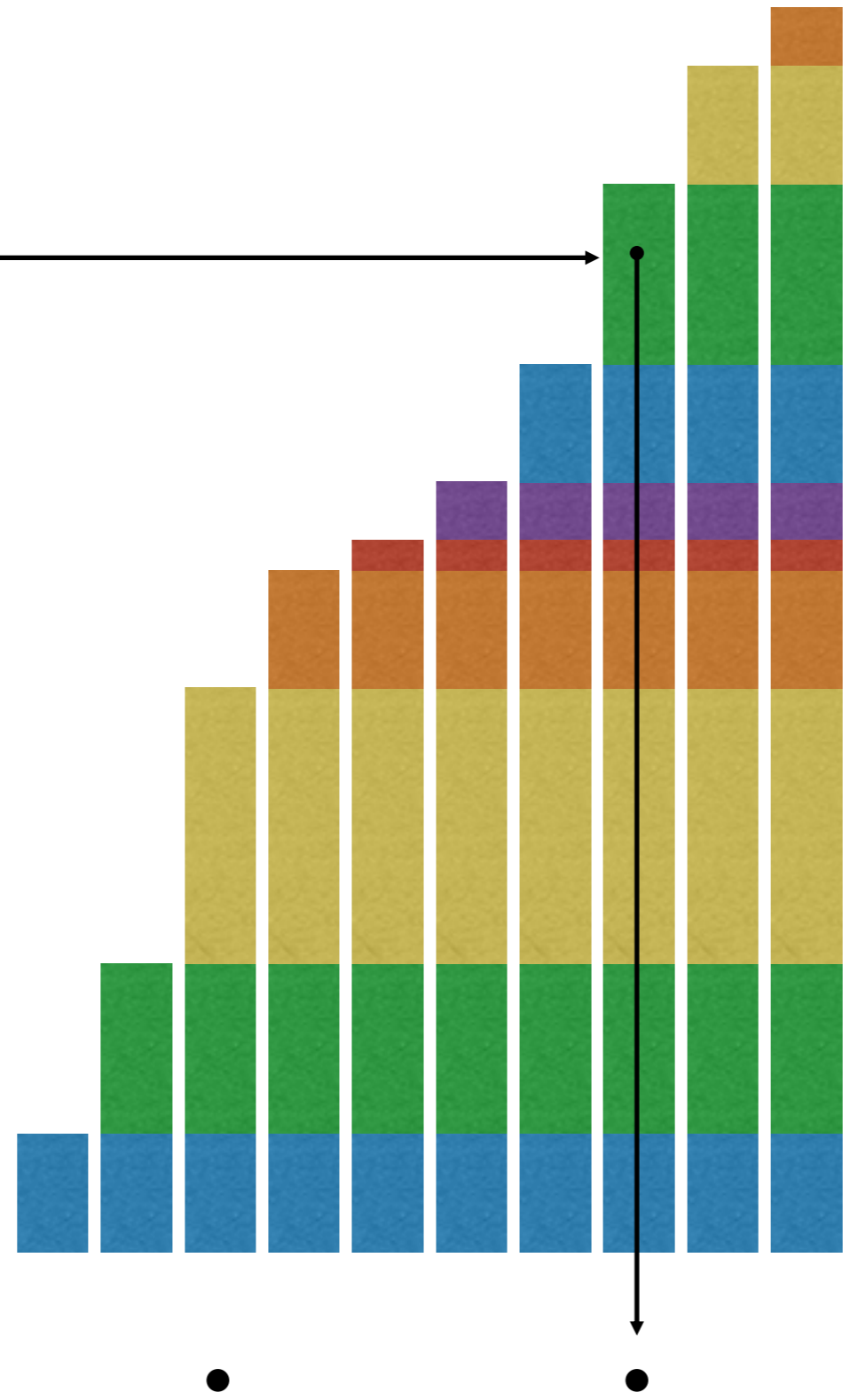
0

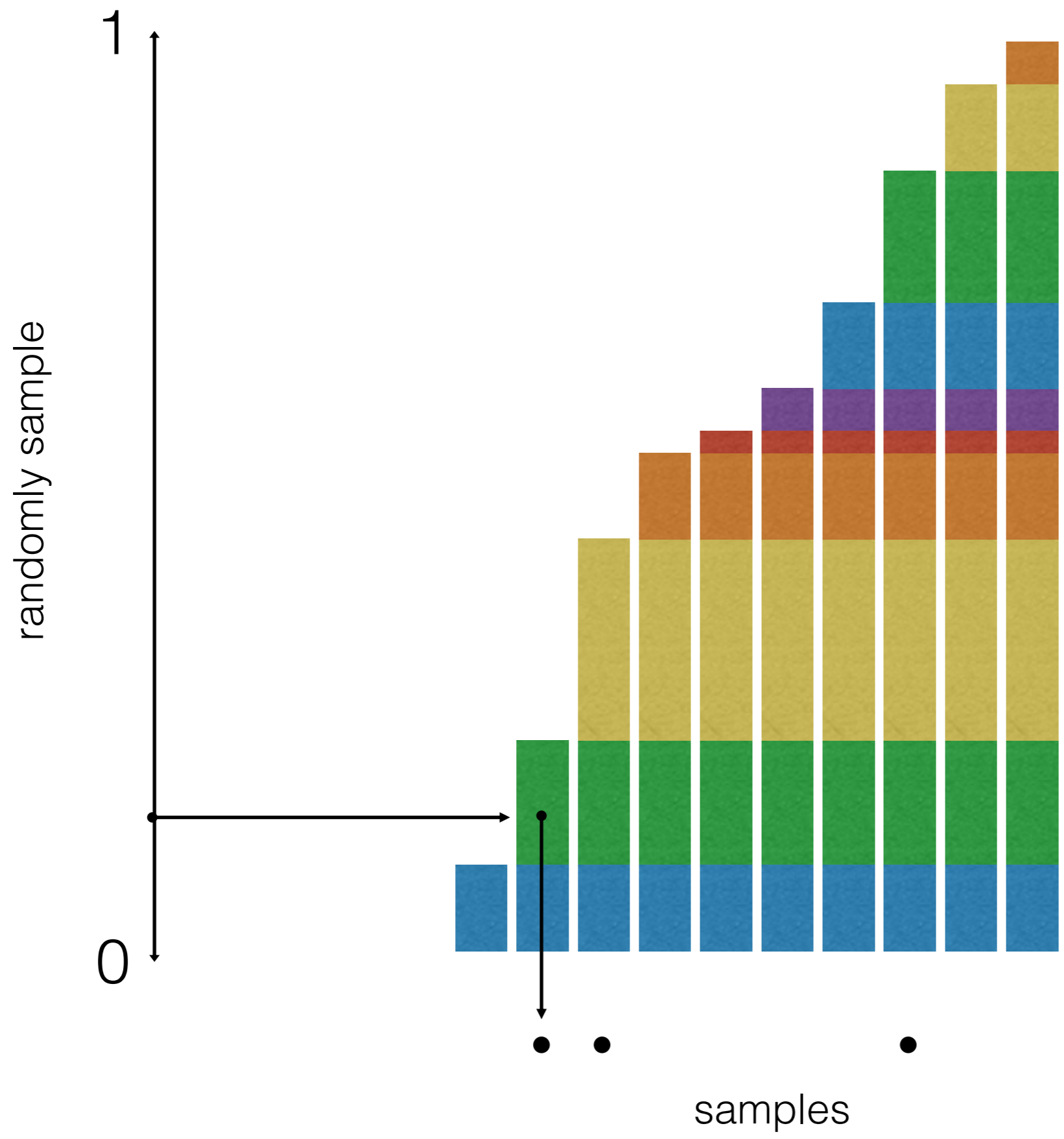
We can draw samples,  
using the CDF...



randomly sample

1  
0





Now to estimate the 'hidden' PDF  
place Gaussian bumps on the samples...



samples

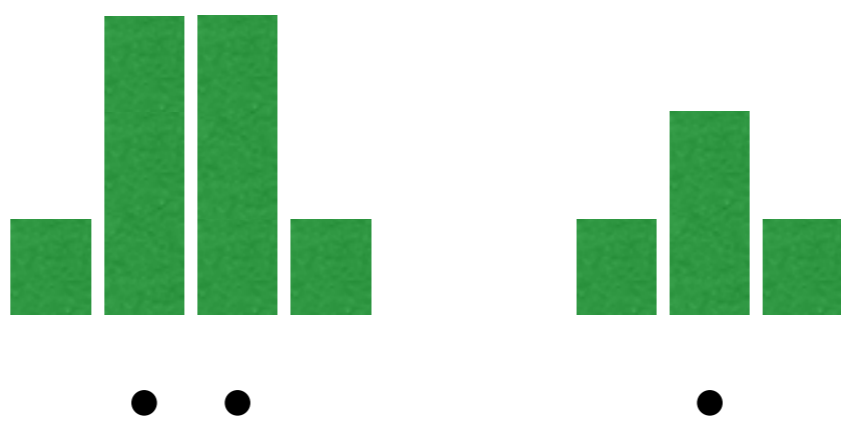


discretized 'bump'

samples

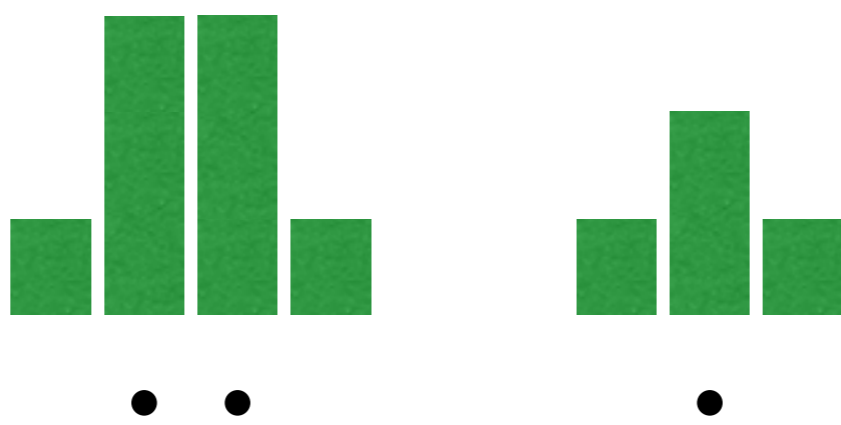
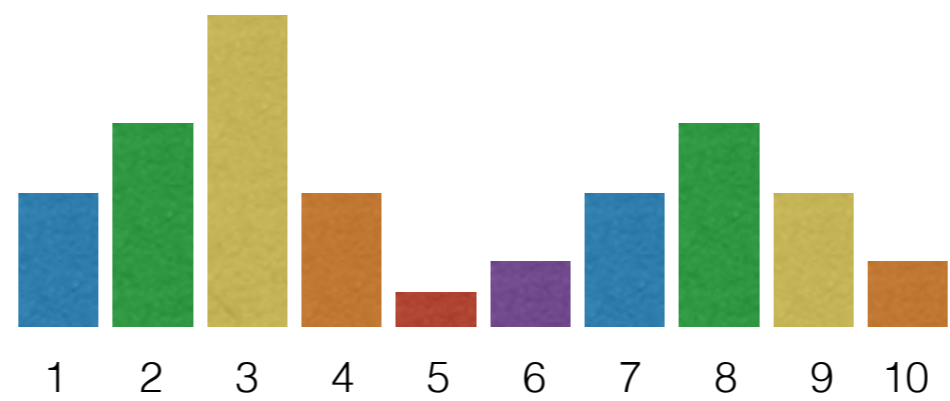


samples



samples

Kernel Density Estimate approximates the original PDF

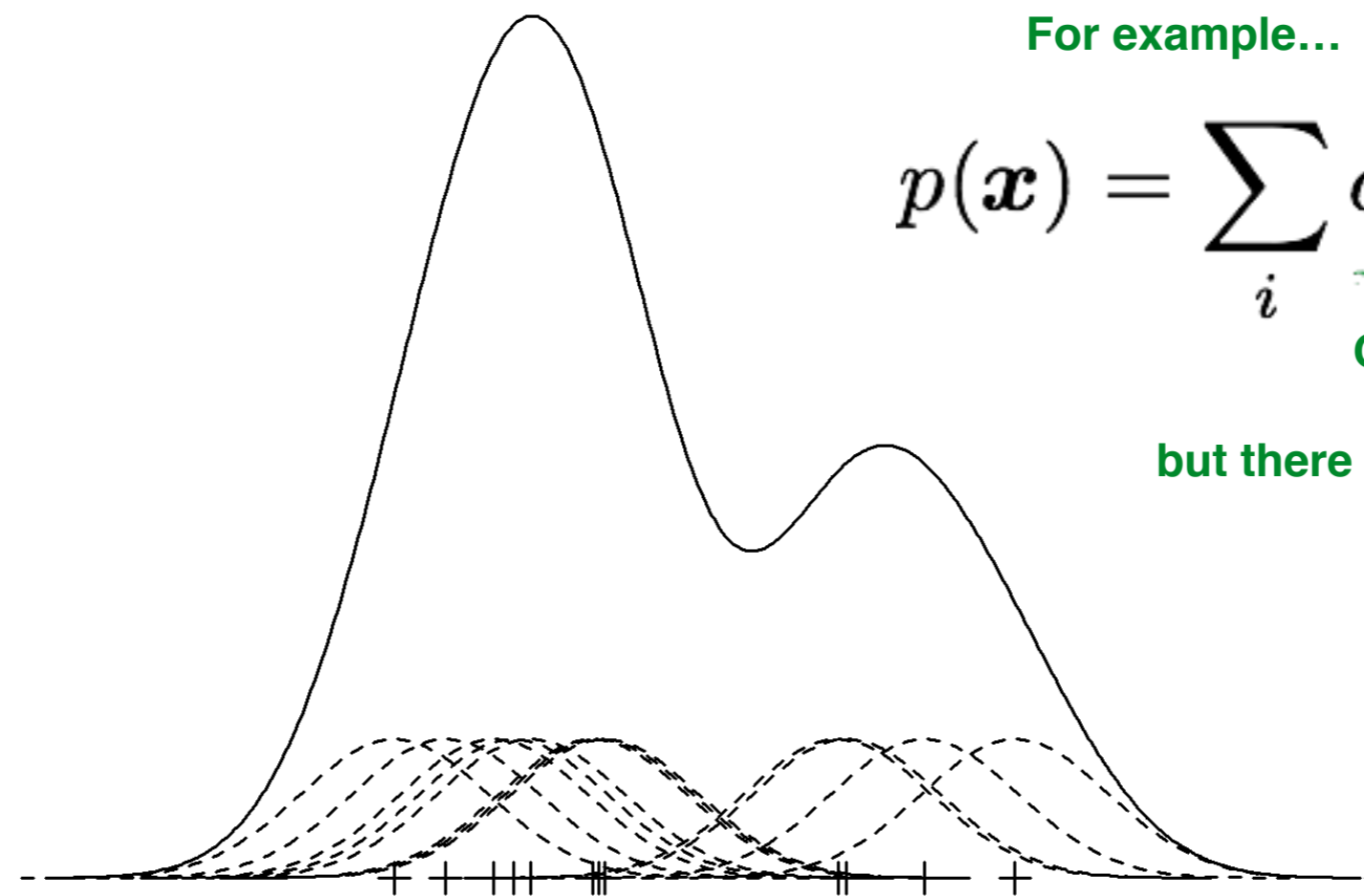


samples



# Kernel Density Estimation

Approximate the underlying PDF from samples from it



For example...

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

Gaussian 'bump' aka 'kernel'

but there are many types of kernels!

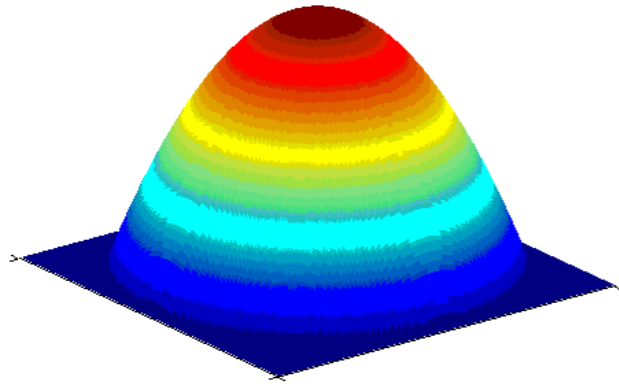
Put 'bump' on every sample to approximate the PDF

# Kernel Function

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

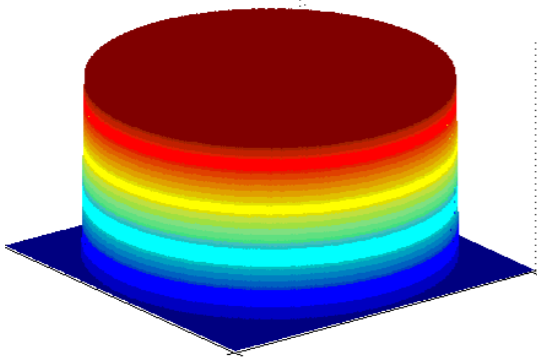
returns the 'distance' between two points

# Epanechnikov kernel



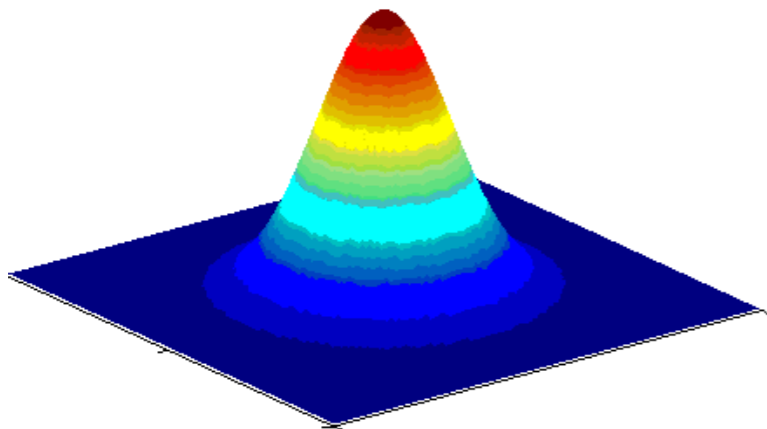
$$K(\mathbf{x}, \mathbf{x}') = \begin{cases} c(1 - \|\mathbf{x} - \mathbf{x}'\|^2) & \|\mathbf{x} - \mathbf{x}'\|^2 \leq 1 \\ 0 & \text{otherwise} \end{cases}$$

# Uniform kernel



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

# Normal kernel



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

These are all radially symmetric kernels

# Radially symmetric kernels

...can be written in terms of its *profile*

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



profile

# Connecting KDE and the Mean Shift Algorithm

# Mean-Shift Tracking

Given a set of points:

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

and a kernel:

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

Find the mean sample point:

$$\mathbf{x}$$

# Mean-Shift Algorithm

Initialize  $\mathbf{x}$  place we start

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

1. Compute mean-shift

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

compute the 'mean'

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

compute the 'shift'

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

update the point

*Where does this algorithm come from?*

# Mean-Shift Algorithm

Initialize  $\mathbf{x}$

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

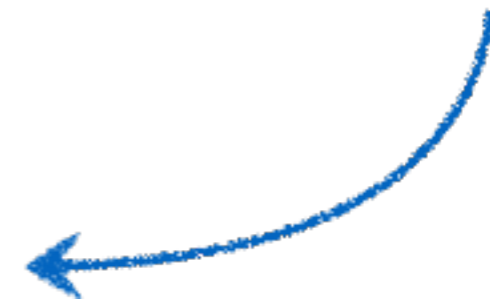
1. Compute mean-shift

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

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

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

*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)

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

can compute probability for any point using the KDE!

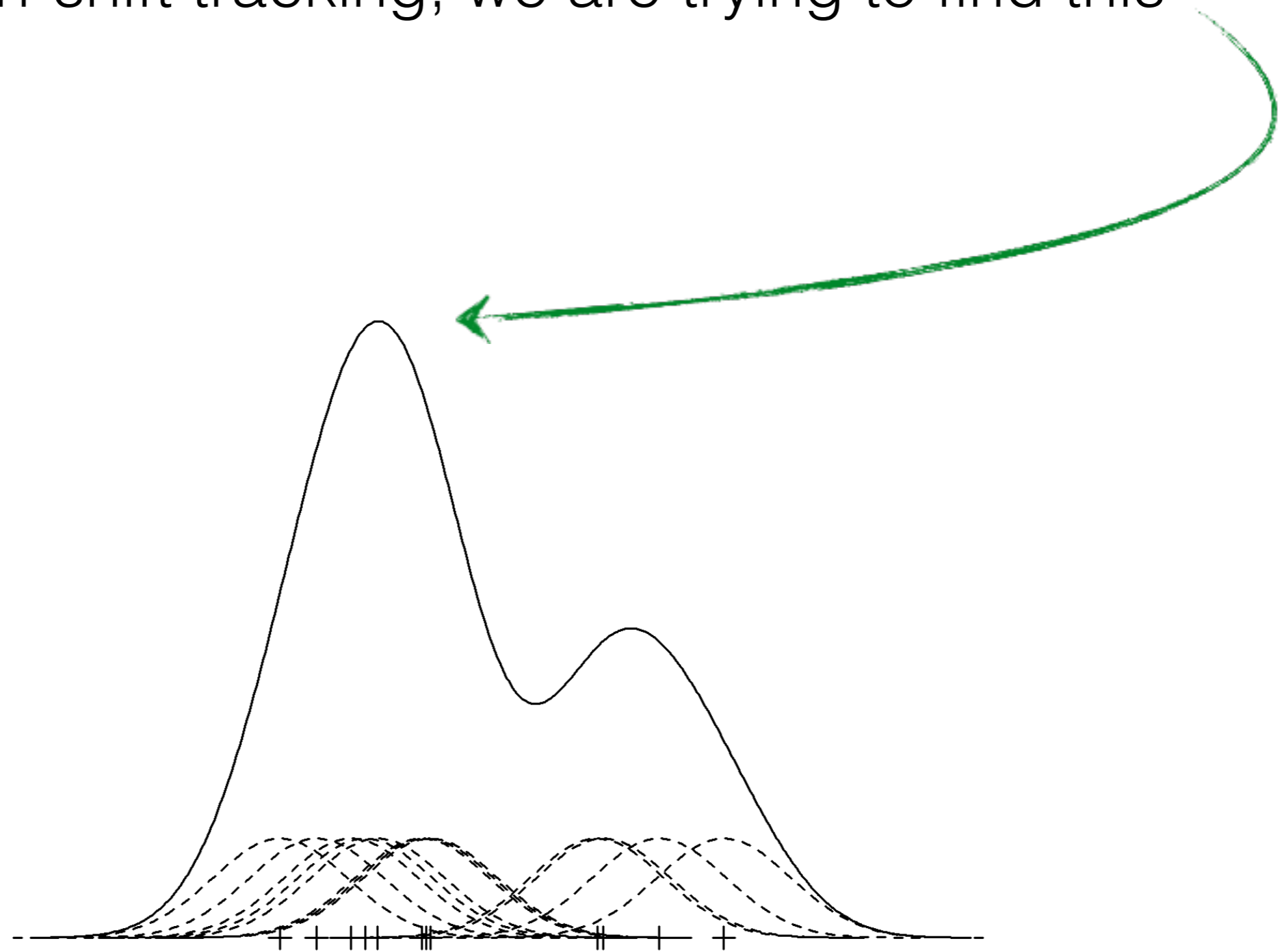
### We can show that:

Gradient of the PDF is related to the mean shift vector

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

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:

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

find the solution that has the highest probability

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

usually non-linear

non-parametric

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

We are trying to optimize this:

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

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

usually non-linear

non-parametric

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

compute partial derivatives ... **gradient descent!**

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

Compute the gradient

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

Gradient

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

Expand the gradient (algebra)

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

Gradient

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

Expand gradient

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

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

Gradient

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

Expand gradient

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

Call the gradient of the kernel function  $g$

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



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

Gradient

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

Expand gradient

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

change of notation  
(kernel-shadow pairs)

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

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

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

multiply it out

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

too long!

(use short hand notation)

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

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

multiply by one!

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

collecting like terms...

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

*What's happening here?*

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

mean
shift

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

Let

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

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

# Mean-Shift Algorithm

Initialize  $\mathbf{x}$

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

1. Compute mean-shift

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

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

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

gradient with  
adaptive step size

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

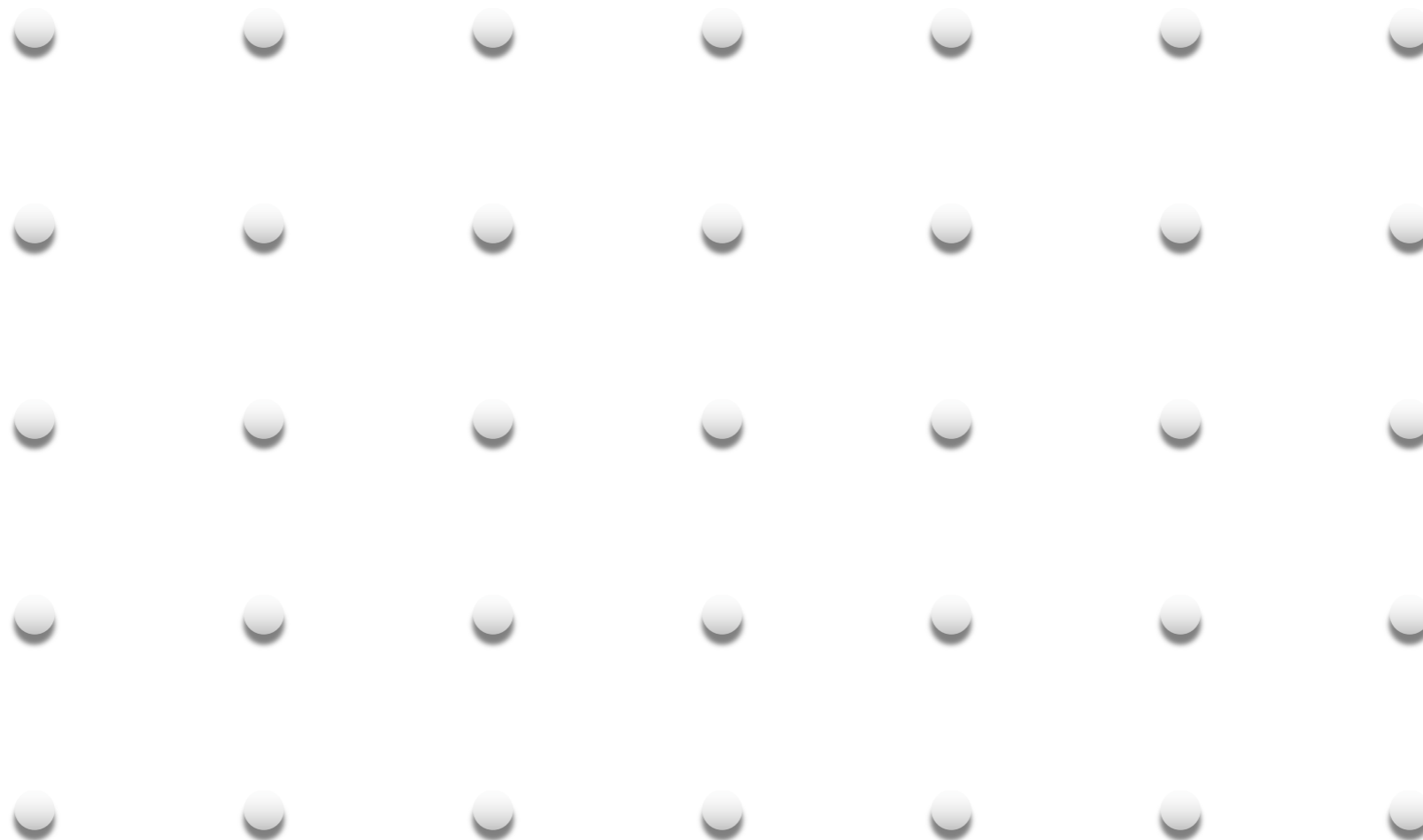
Just 5 lines of code!

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

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

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

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

# Mean-Shift Algorithm

(for images)

Initialize  $\mathbf{x}$

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

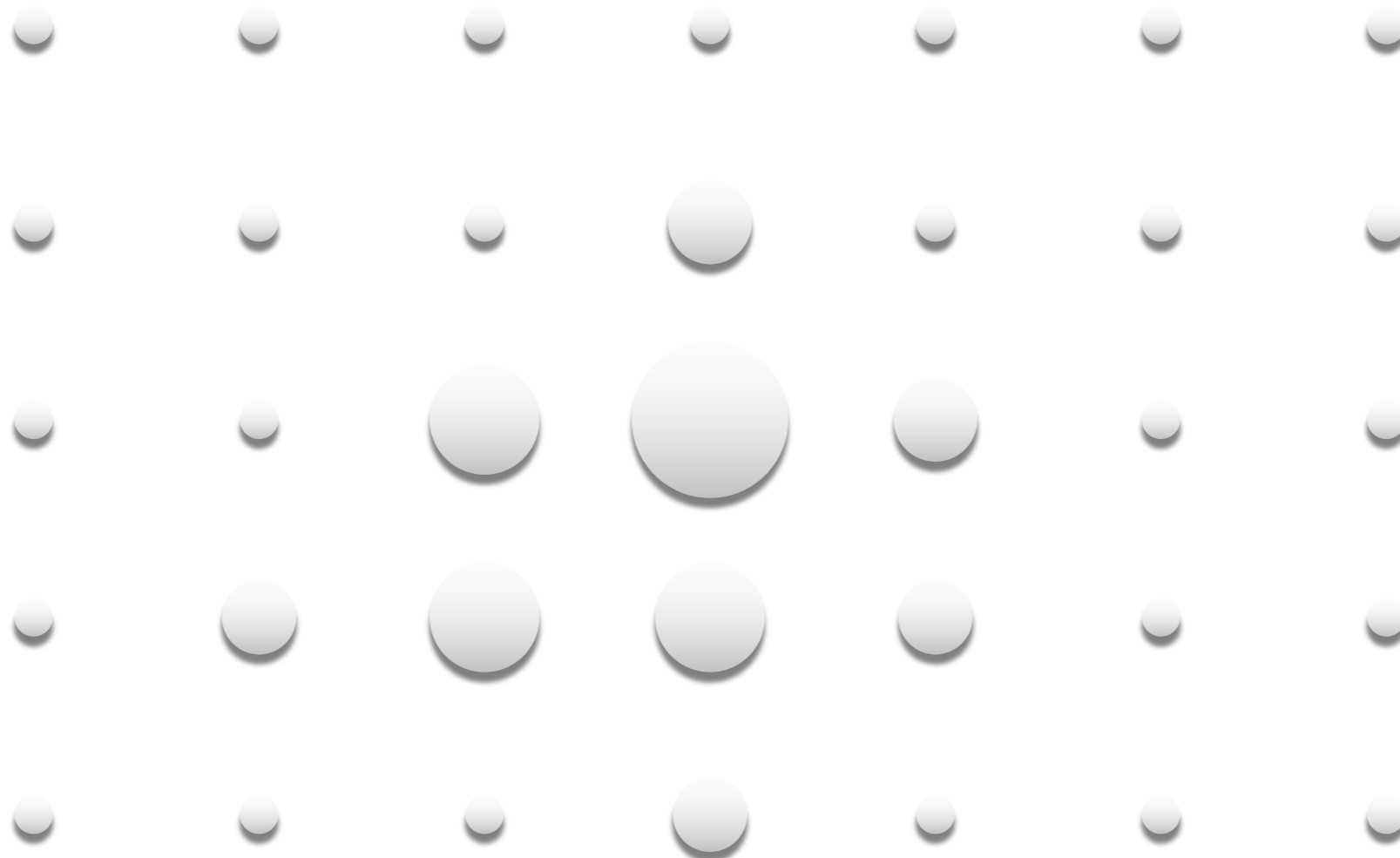
1. Compute mean-shift

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

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

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

For images, each pixel is point with a weight







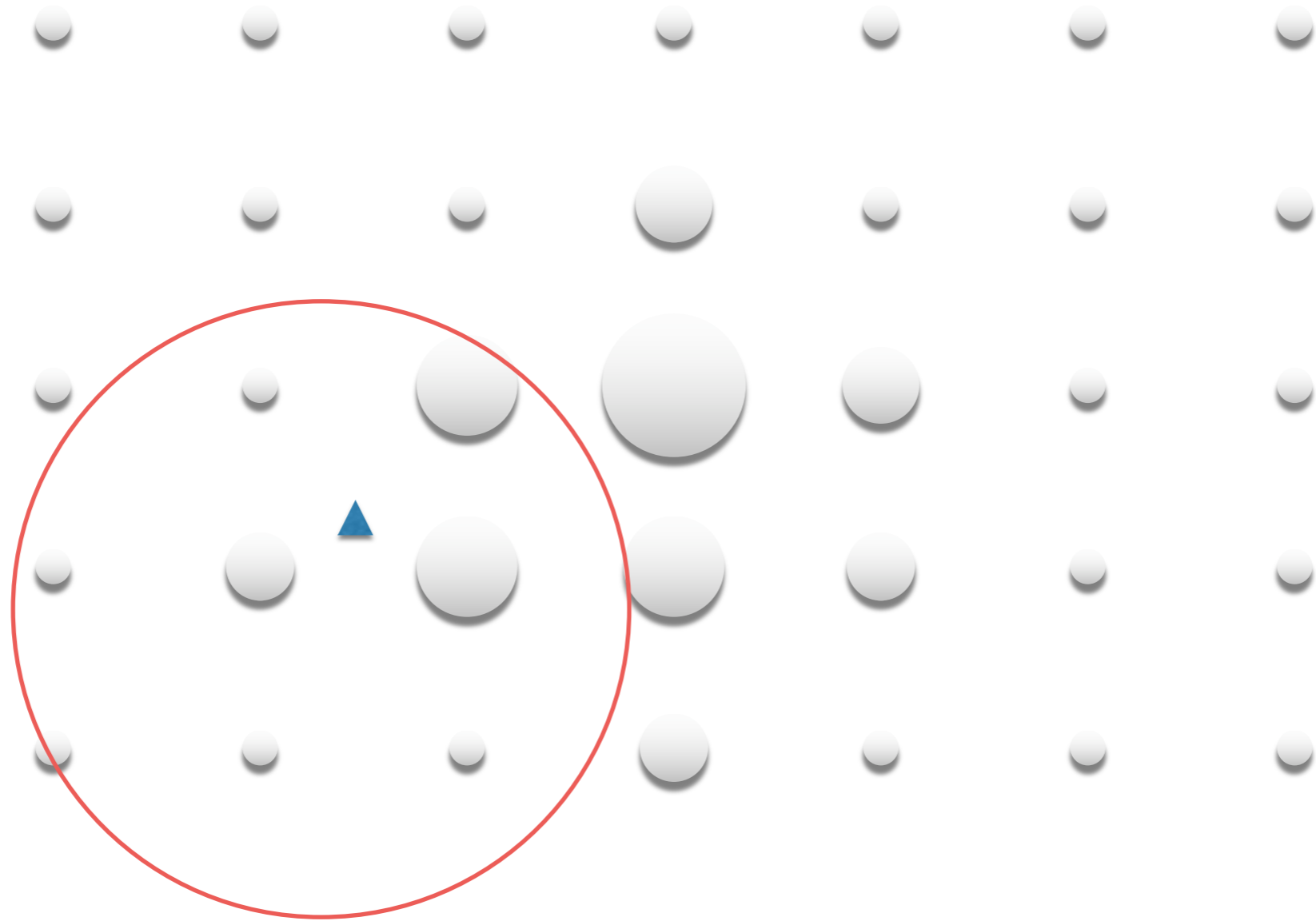






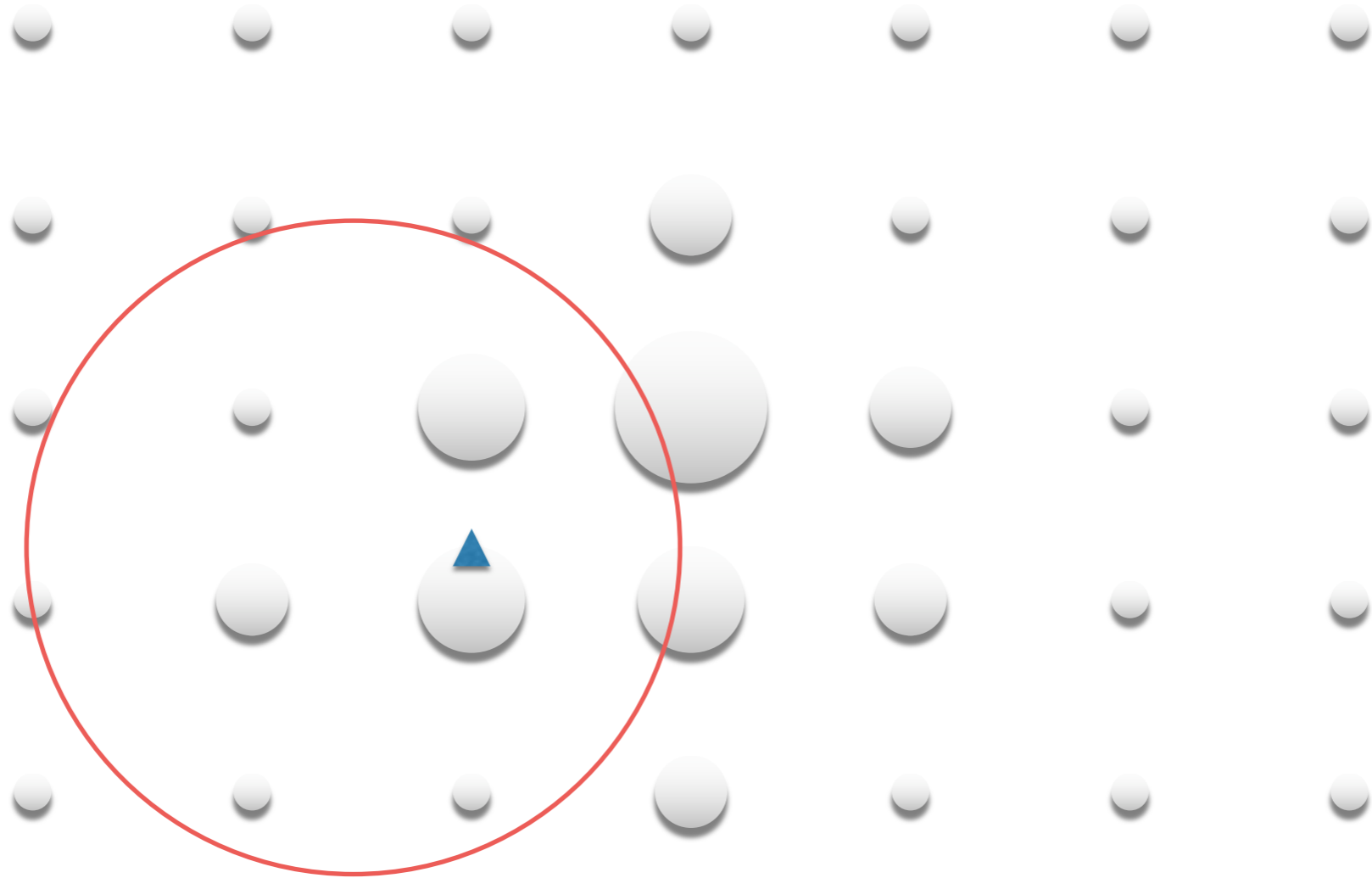


For images, each pixel is point with a weight

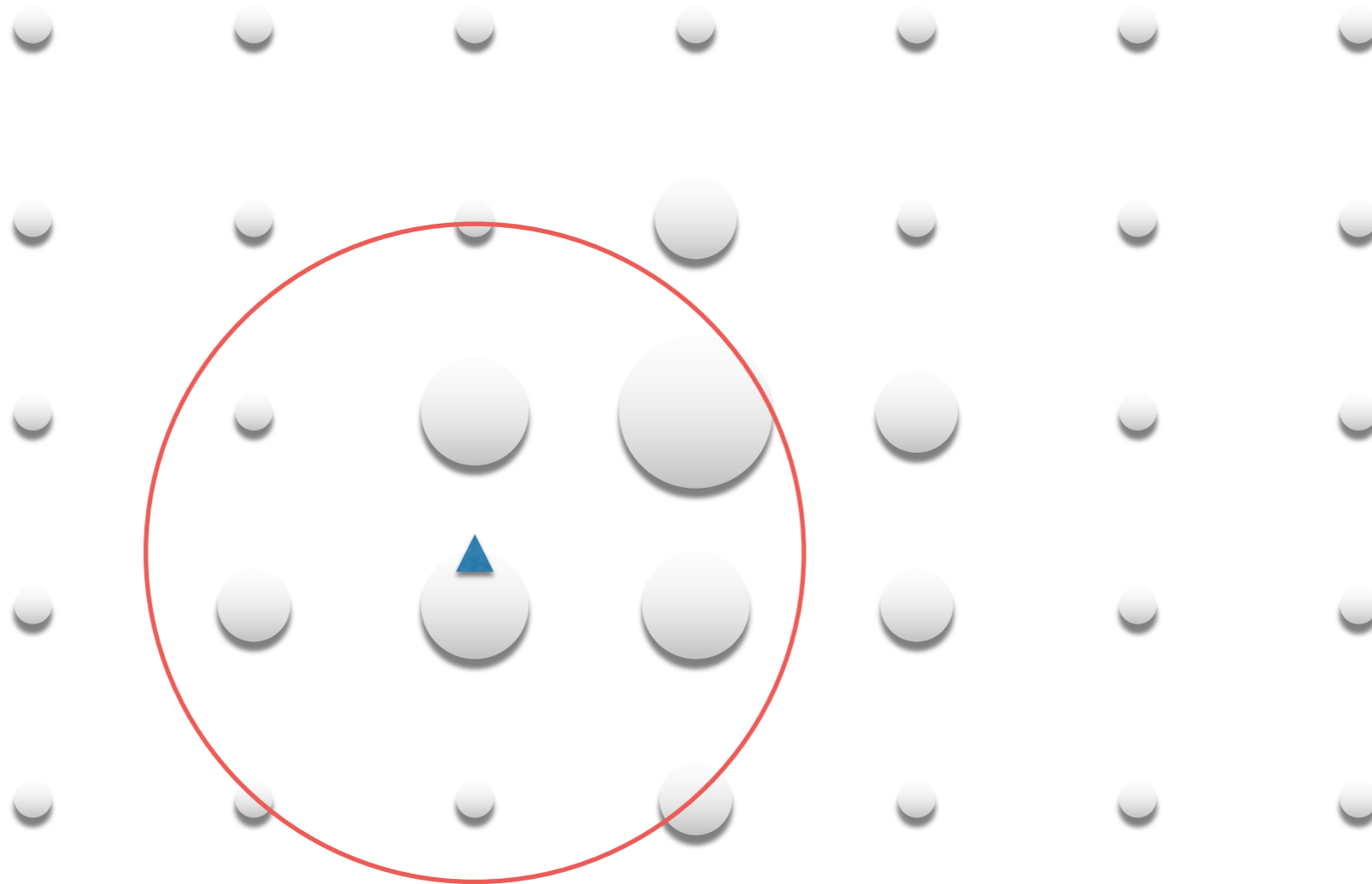




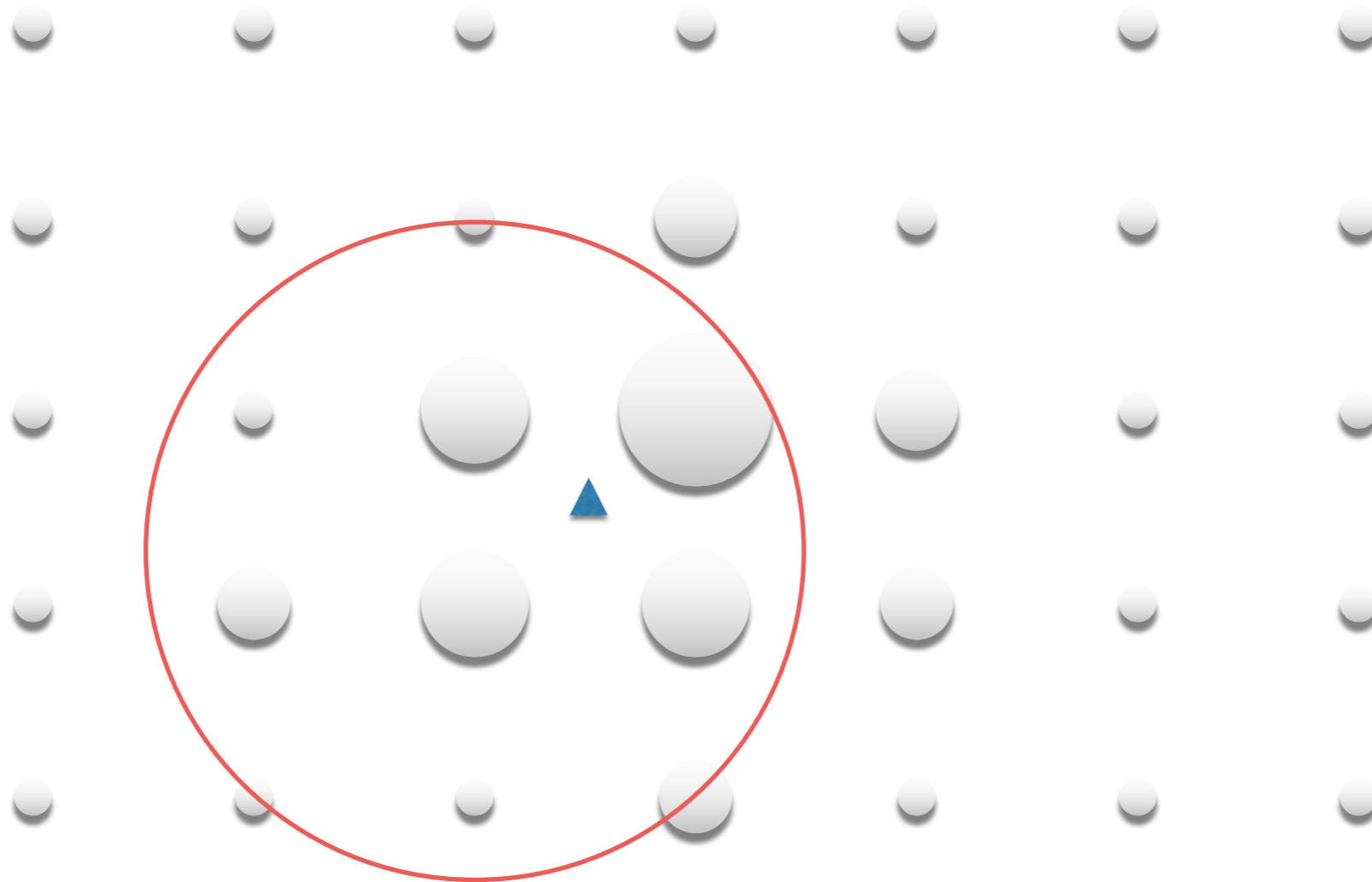
For images, each pixel is point with a weight



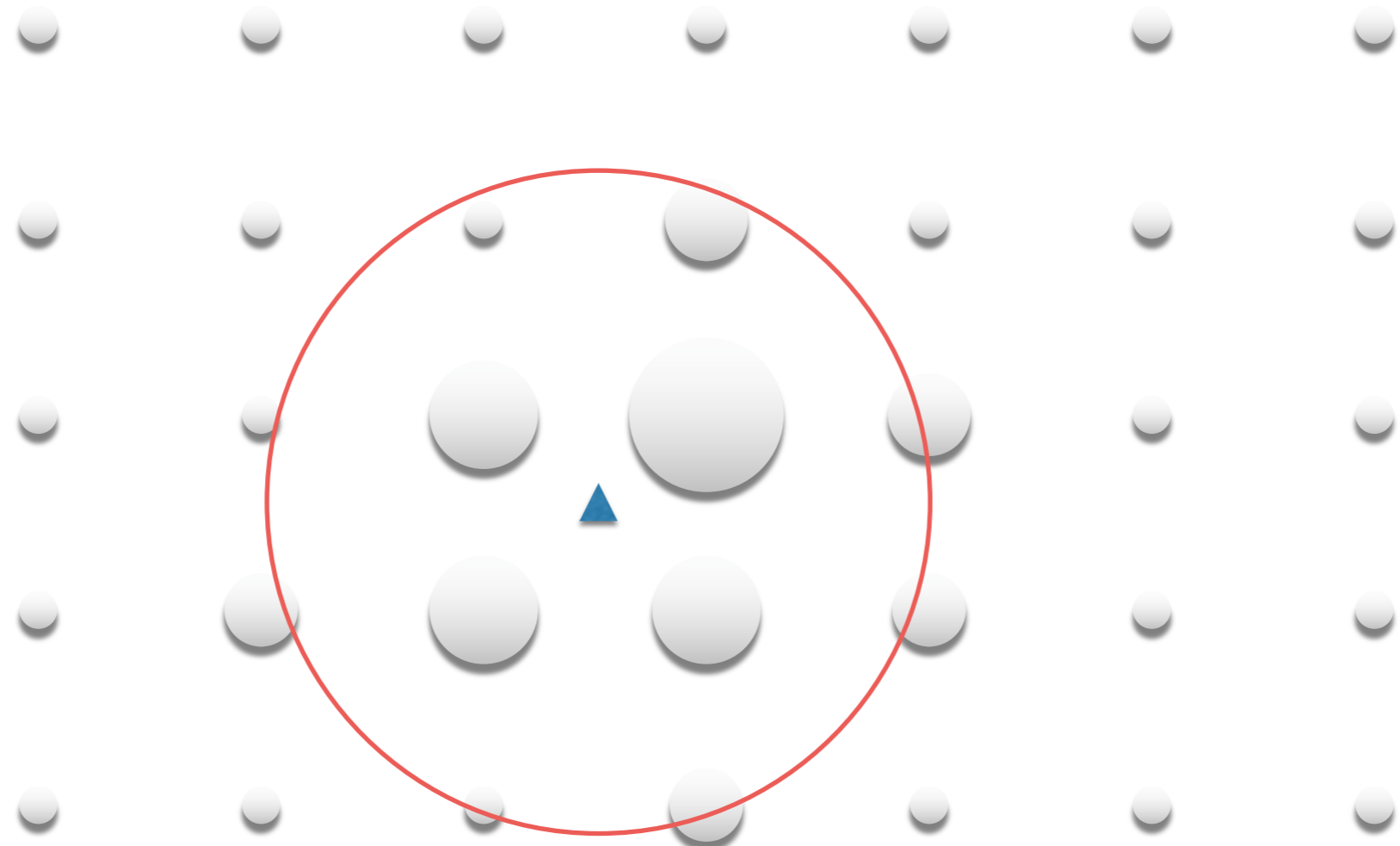
For images, each pixel is point with a weight



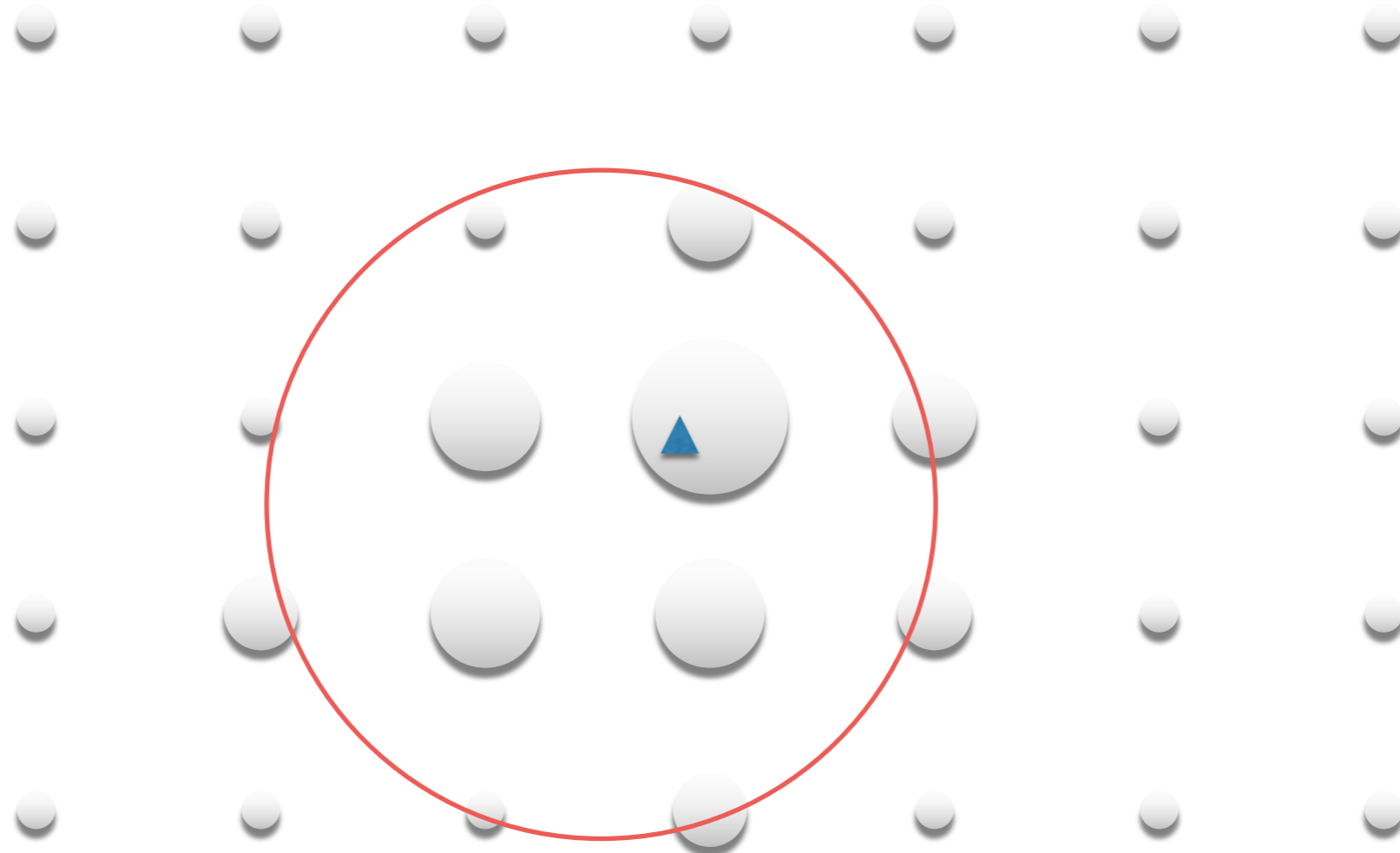
For images, each pixel is point with a weight



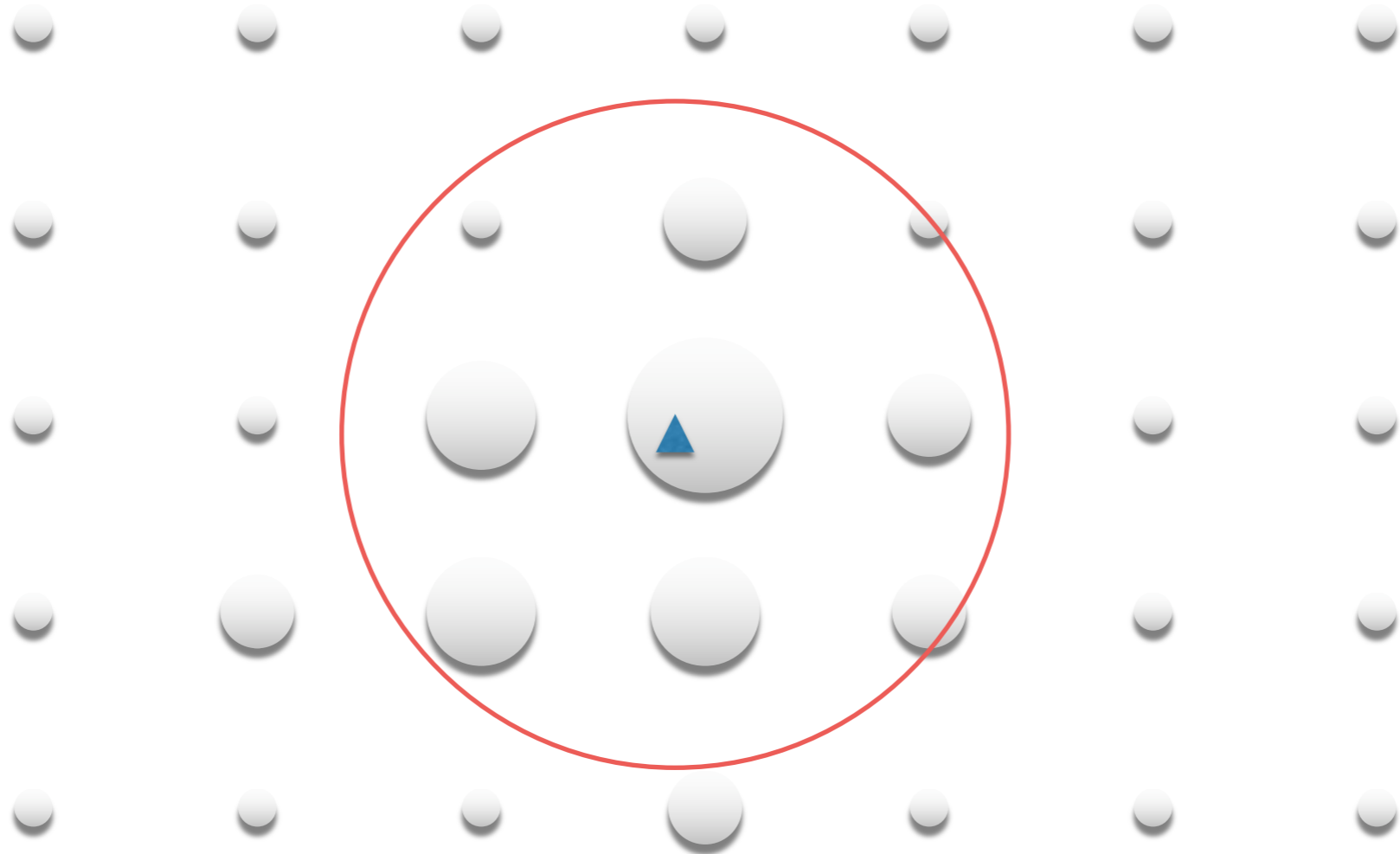
For images, each pixel is point with a weight



For images, each pixel is point with a weight



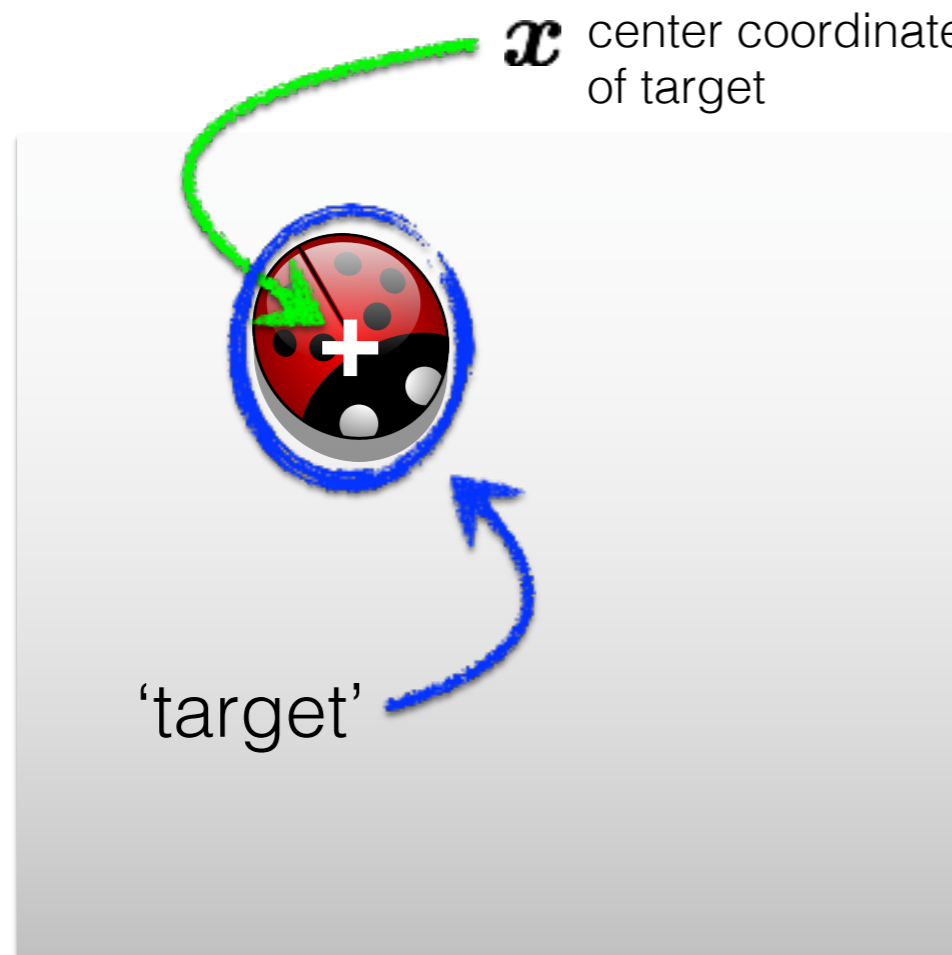
For images, each pixel is point with a weight





Finally... mean shift tracking in video!

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



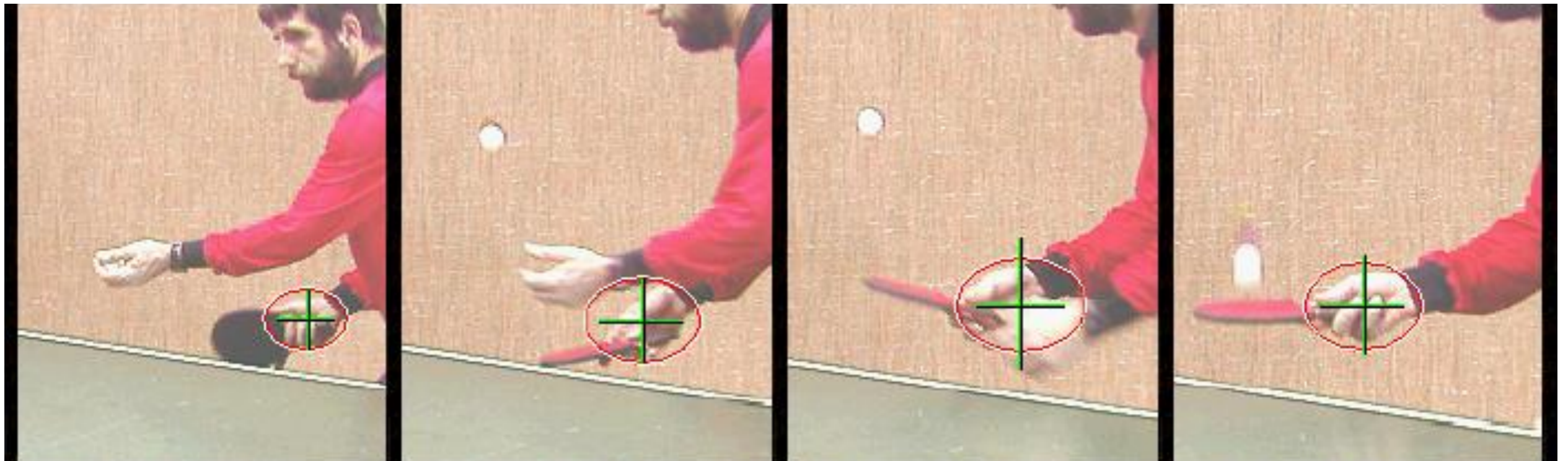
Frame 1



Frame 2

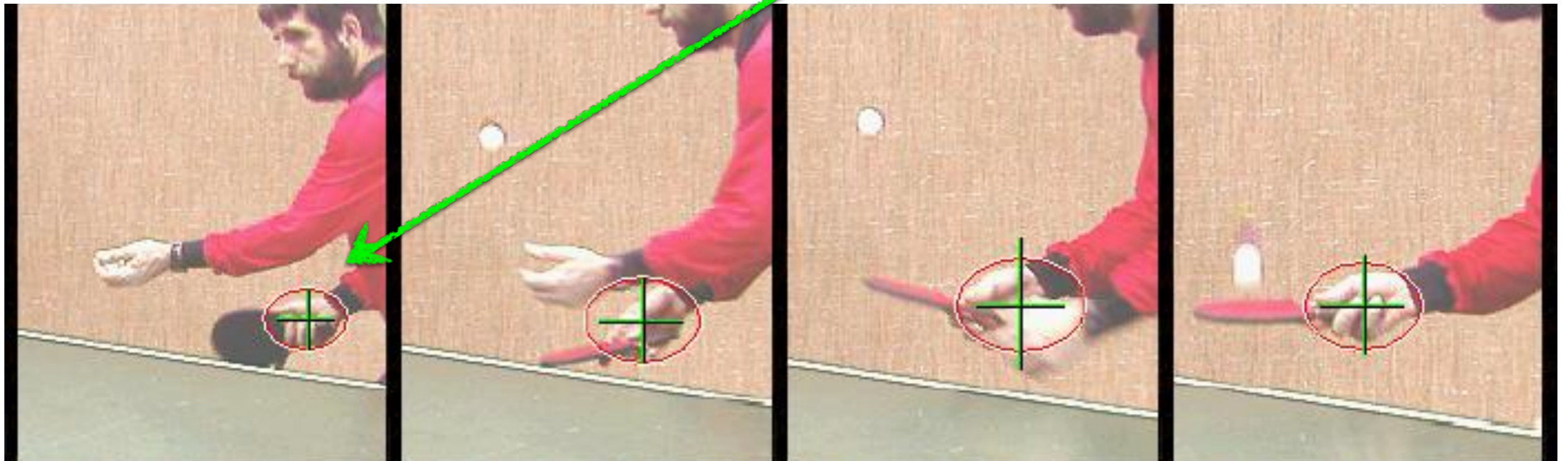
Use the mean shift algorithm  
to find the best candidate location

# Non-rigid object tracking



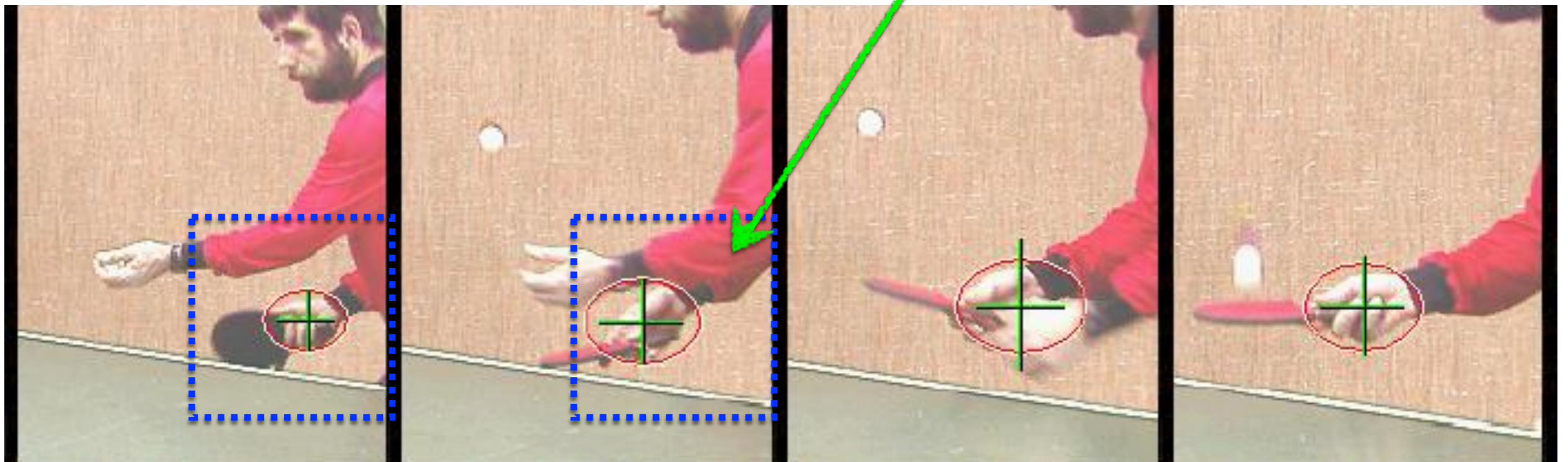
hand tracking

Compute a descriptor for the target



Target

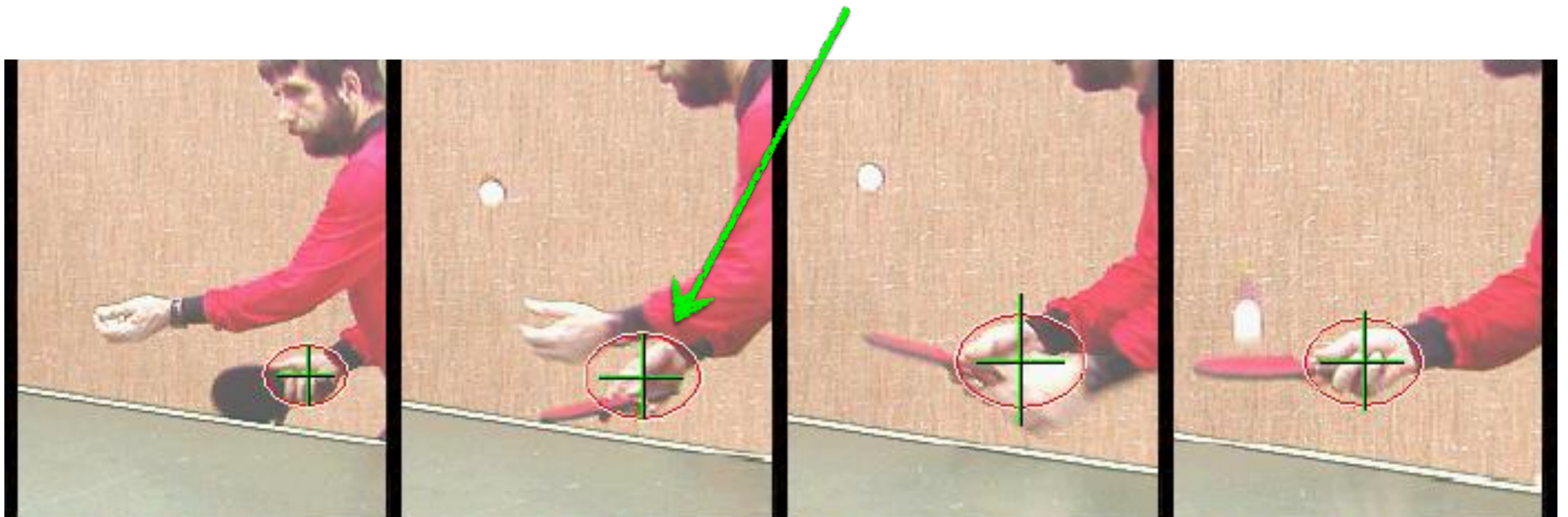
Search for similar descriptor in neighborhood in next frame



Target

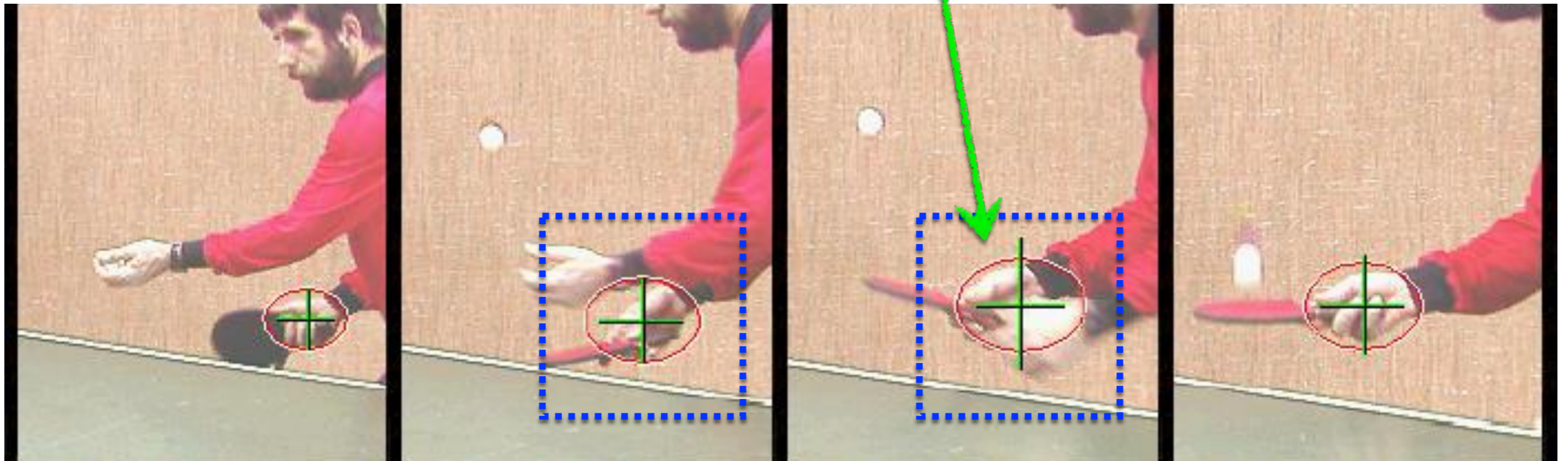
Candidate

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

$$\mathbf{q} = \{q_1, \dots, q_M\}$$

(centered at target center)

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

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

Normalization factor

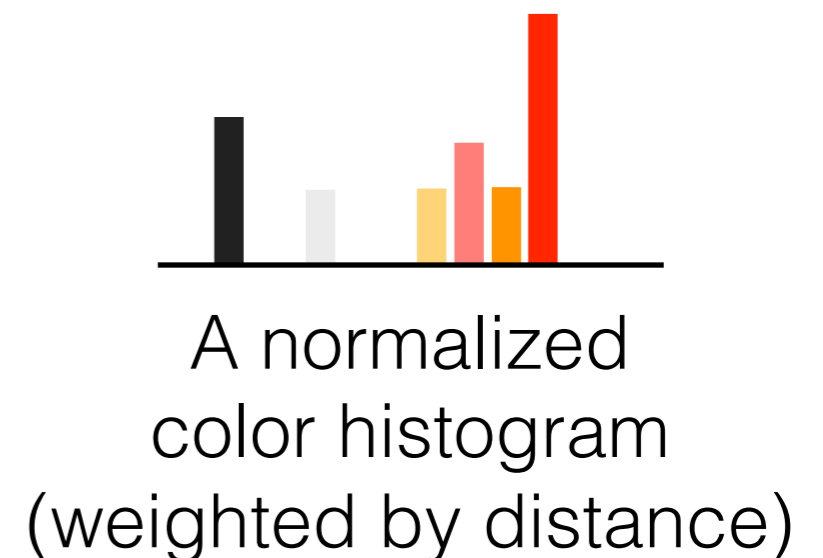
sum over all pixels

function of inverse distance (weight)

Kronecker delta function

quantization function

bin ID



# Modeling the candidate

M-dimensional **candidate** descriptor

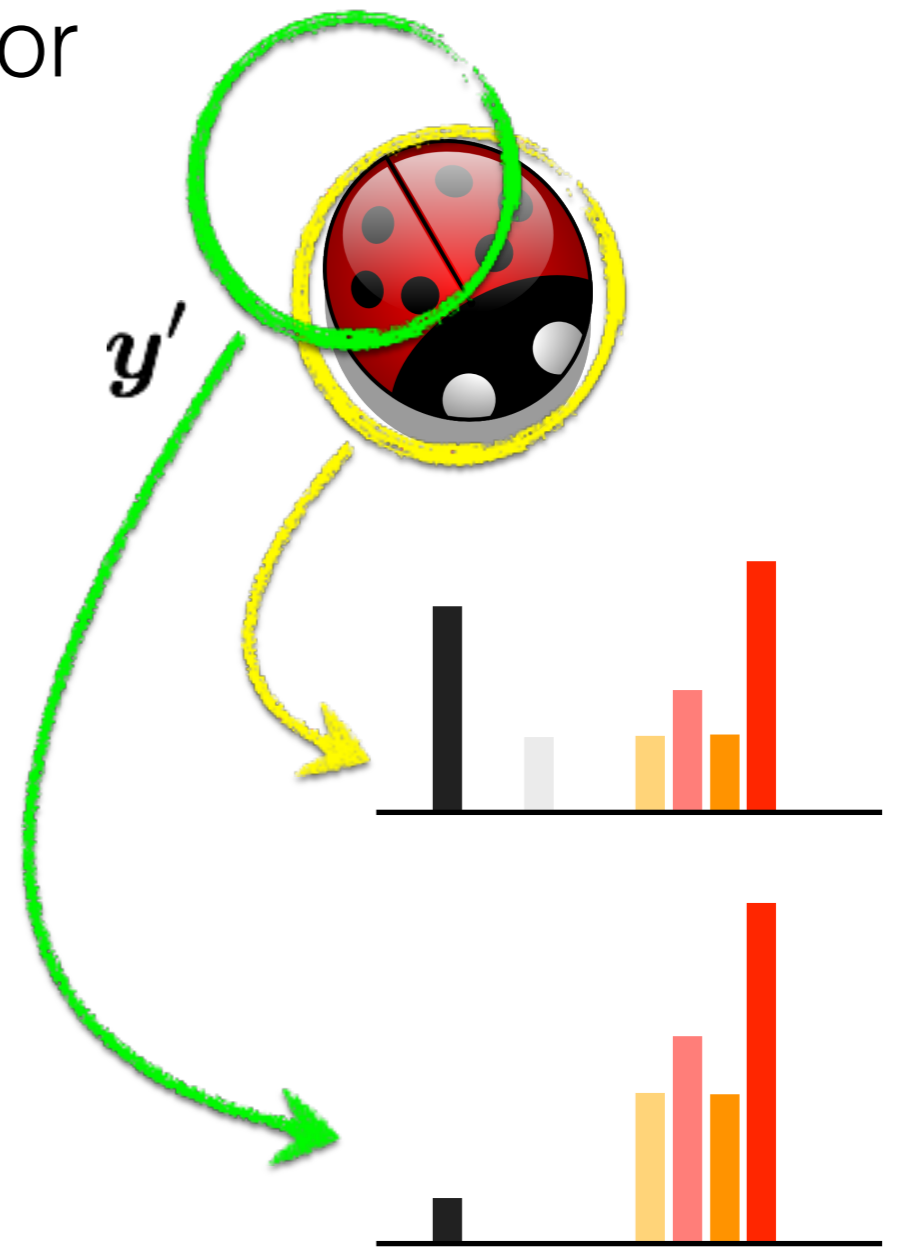
$$\mathbf{p}(\mathbf{y}) = \{p_1(\mathbf{y}), \dots, p_M(\mathbf{y})\}$$

(centered at location  $\mathbf{y}$ )

a weighted histogram at  $\mathbf{y}$

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

bandwidth



# Similarity between the target and candidate

Distance function

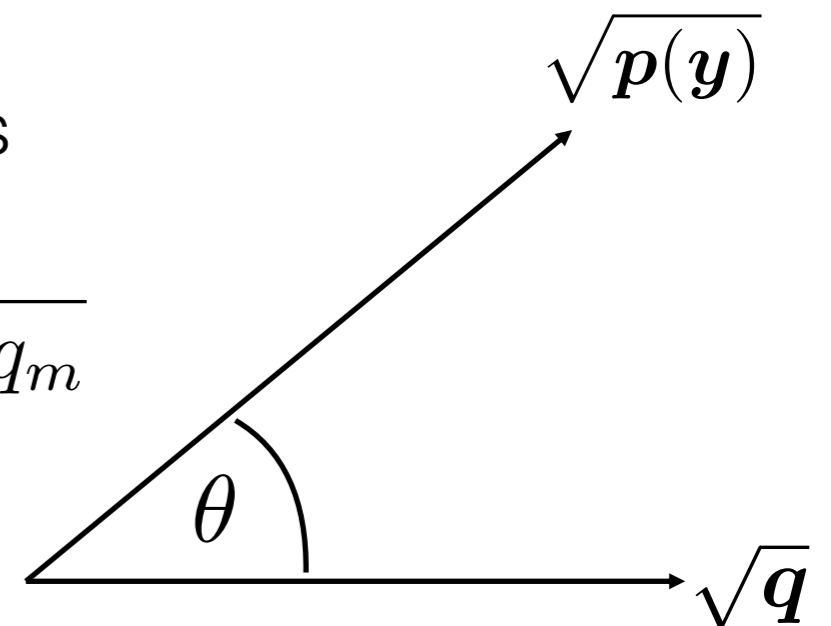
$$d(\mathbf{y}) = \sqrt{1 - \rho[\mathbf{p}(\mathbf{y}), \mathbf{q}]}$$

Bhattacharyya Coefficient

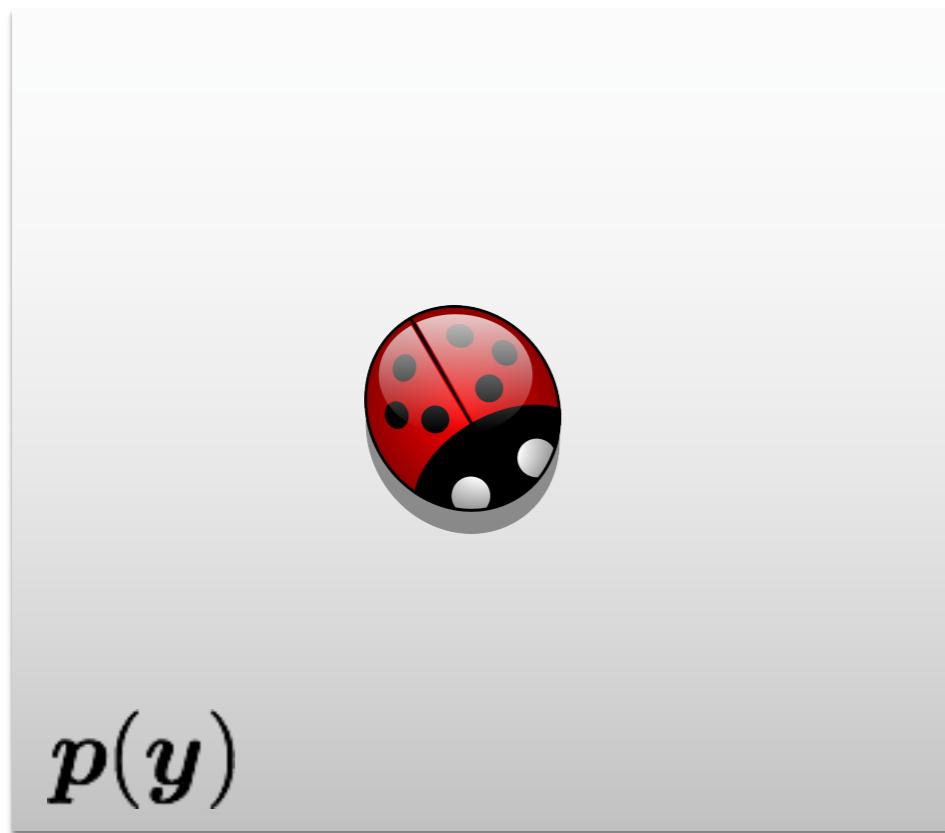
$$\rho(\mathbf{y}) \equiv \rho[\mathbf{p}(\mathbf{y}), \mathbf{q}] = \sum_m \sqrt{p_m(\mathbf{y})q_m}$$

Just the Cosine distance between two unit vectors

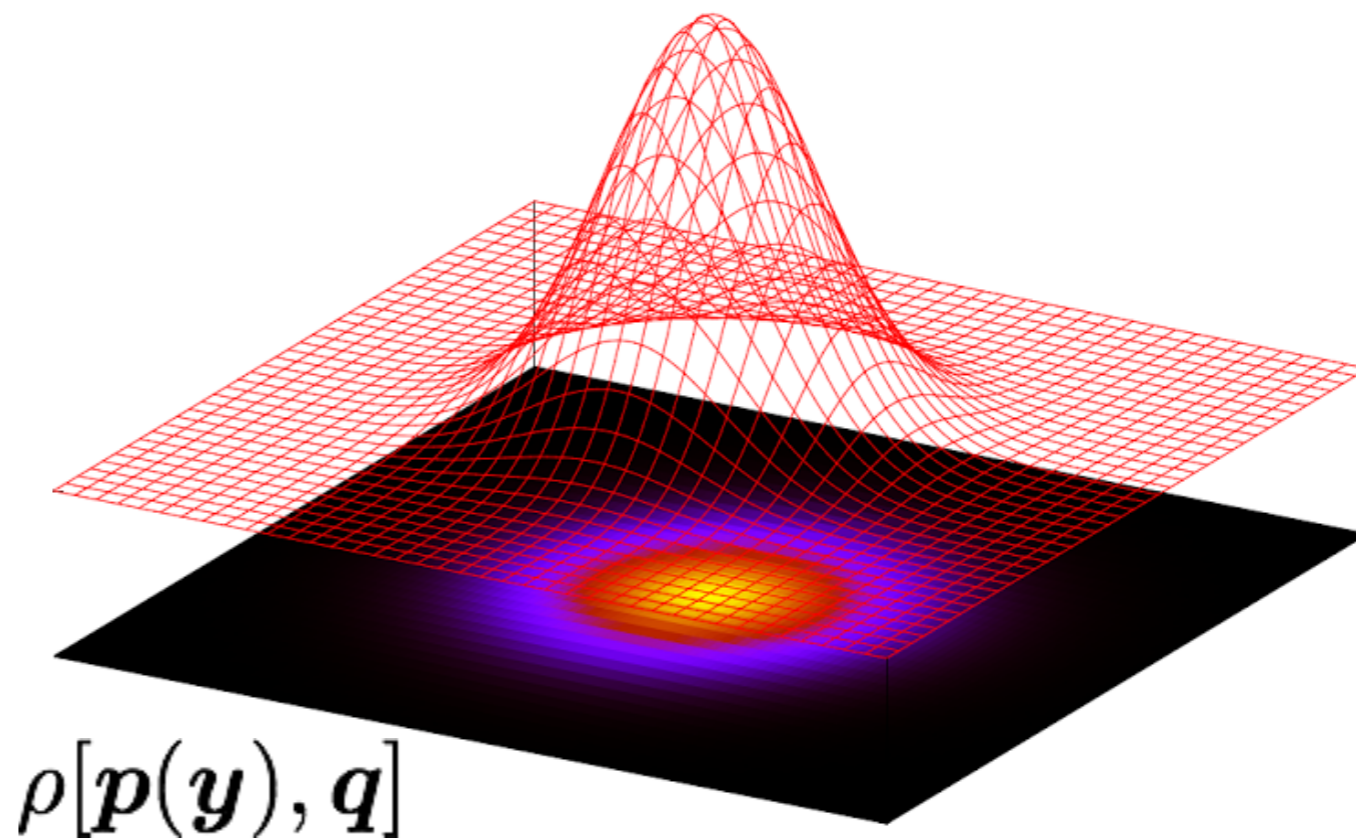
$$\rho(\mathbf{y}) = \cos \theta_{\mathbf{y}} = \frac{\sqrt{\mathbf{p}(\mathbf{y})}^T \sqrt{\mathbf{q}}}{\|\sqrt{\mathbf{p}(\mathbf{y})}\| \|\sqrt{\mathbf{q}}\|} = \sum_m \sqrt{p_m(\mathbf{y})q_m}$$



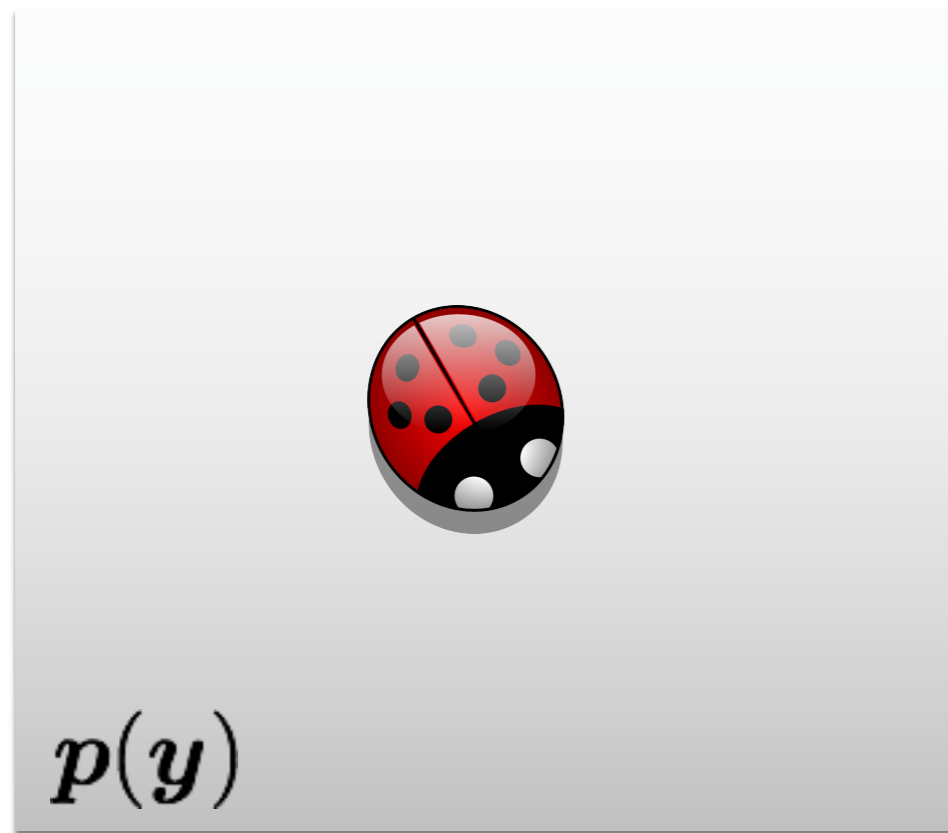
Now we can compute the similarity between  
a target and multiple candidate regions



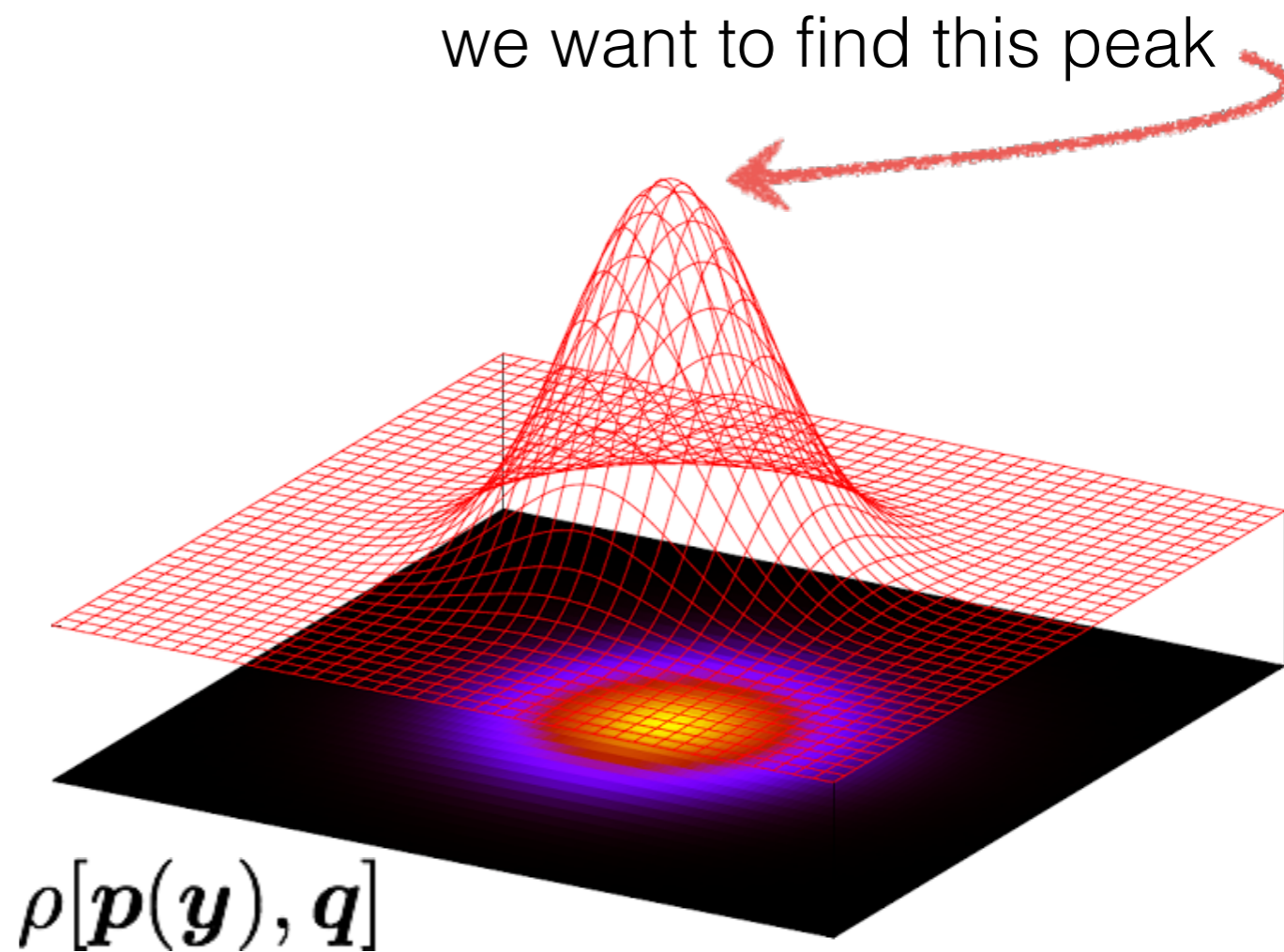
image



similarity over image



image



similarity over image

# Objective function

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

Assuming a good initial guess

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

Linearize around the initial guess (Taylor series expansion)

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

function at specified value derivative

Linearized objective

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

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

Remember  
definition of this?



Fully expanded

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



Fully expanded linearized objective

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

Moving terms around...

$$\rho[\mathbf{p}(\mathbf{y}), \mathbf{q}] \approx \underbrace{\frac{1}{2} \sum_m \sqrt{p_m(\mathbf{y}_0)q_m}}_{\text{Does not depend on unknown } \mathbf{y}} + \underbrace{\frac{C_h}{2} \sum_n w_n k \left( \left\| \frac{\mathbf{y} - \mathbf{x}_n}{h} \right\|^2 \right)}_{\text{Weighted kernel density estimate}}$$

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

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

OK, why are we doing all this math?

We want to maximize this

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

We want to maximize this

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

Fully expanded linearized objective

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

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

We want to maximize this

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

Fully expanded linearized objective

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

doesn't depend on unknown  $\mathbf{y}$

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

We want to maximize this

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

only need to  
maximize this!

Fully expanded linearized objective

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

doesn't depend on unknown  $\mathbf{y}$

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

We want to maximize this

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

Fully expanded linearized objective

$$\rho[\mathbf{p}(\mathbf{y}), \mathbf{q}] \approx \underbrace{\frac{1}{2} \sum_m \sqrt{p_m(\mathbf{y}_0)} q_m}_{\text{doesn't depend on unknown } \mathbf{y}} + \frac{C_h}{2} \sum_n w_n k \left( \left\| \frac{\mathbf{y} - \mathbf{x}_n}{h} \right\|^2 \right)$$

where

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

what can we use to solve this weighted KDE?

**Mean Shift Algorithm!**

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

the new sample of mean of this KDE is

$$\mathbf{y}_1 = \frac{\sum_n \mathbf{x}_n w_n g \left( \left\| \frac{\mathbf{y}_0 - \mathbf{x}_n}{h} \right\|^2 \right)}{\sum_n w_n g \left( \left\| \frac{\mathbf{y}_0 - \mathbf{x}_n}{h} \right\|^2 \right)} \quad \text{(this was derived earlier)}$$

(new candidate location)

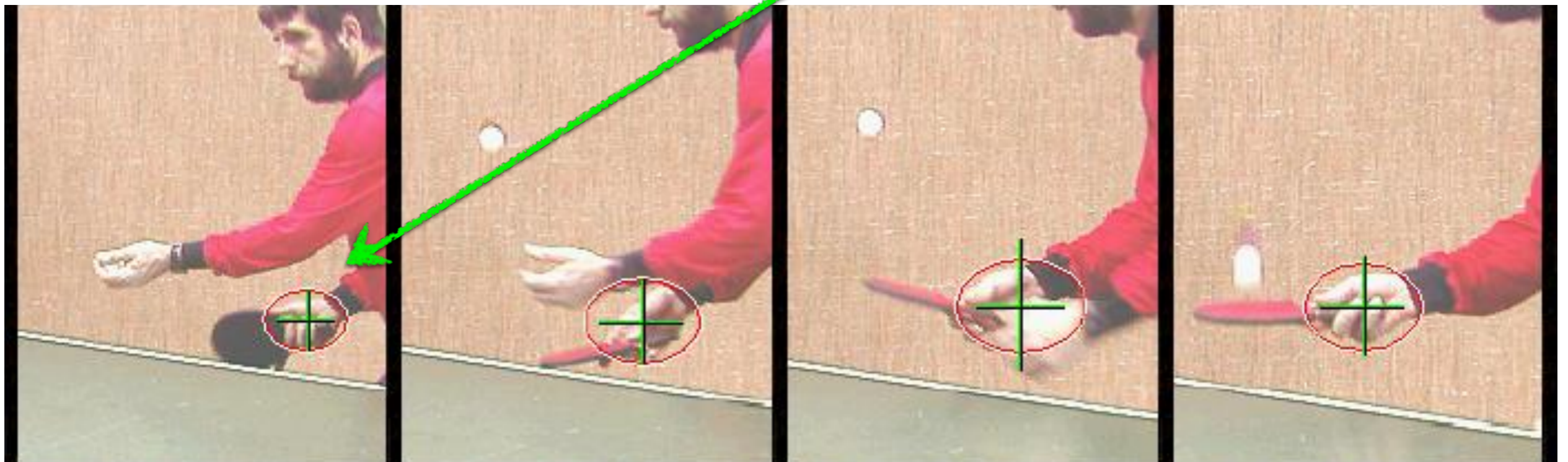


# Mean-Shift Object Tracking

For each frame:

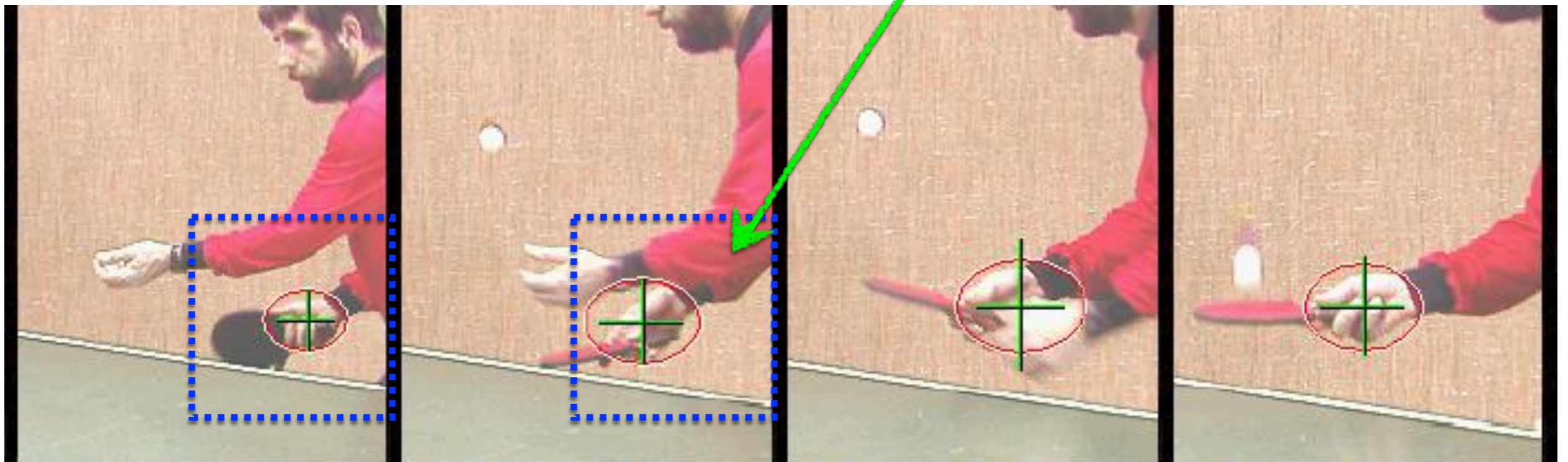
1. Initialize location  $\mathbf{y}_0$   
Compute  $\mathbf{q}$   
Compute  $p(\mathbf{y}_0)$
2. Derive weights  $w_n$
3. Shift to new candidate location (mean shift)  $\mathbf{y}_1$
4. Compute  $p(\mathbf{y}_1)$
5. If  $\|\mathbf{y}_0 - \mathbf{y}_1\| < \epsilon$  return  
Otherwise  $\mathbf{y}_0 \leftarrow \mathbf{y}_1$  and go back to 2

Compute a descriptor for the target



Target  
 $q$

Search for similar descriptor in neighborhood in next frame

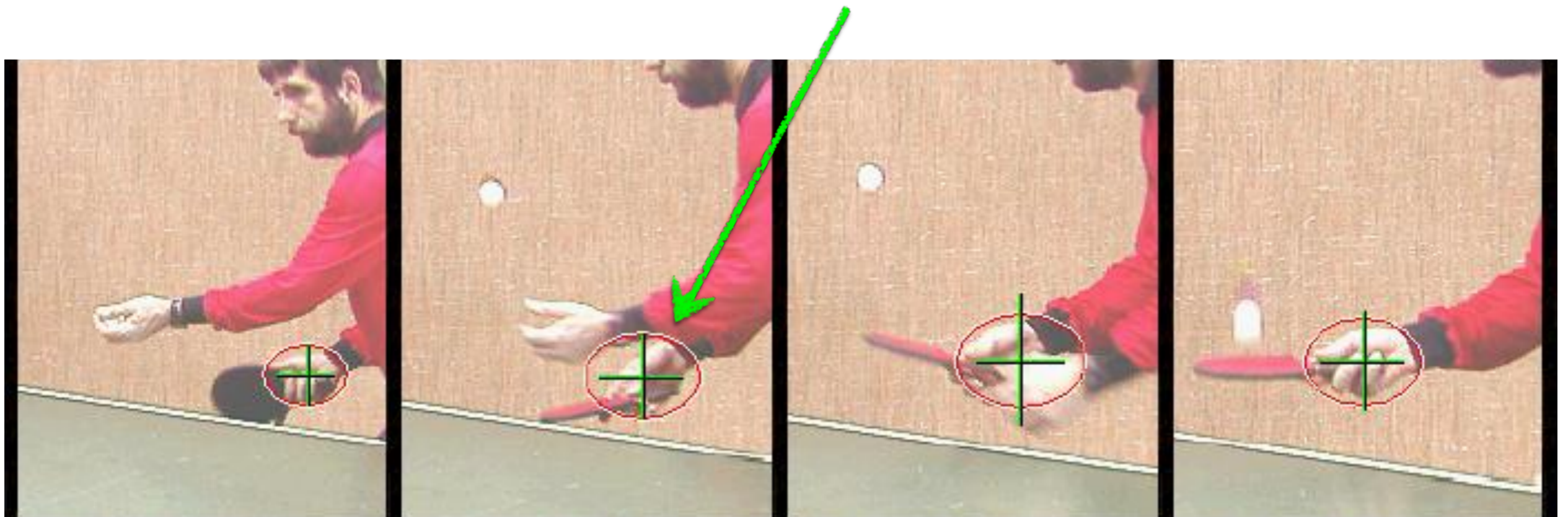


Target

Candidate

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

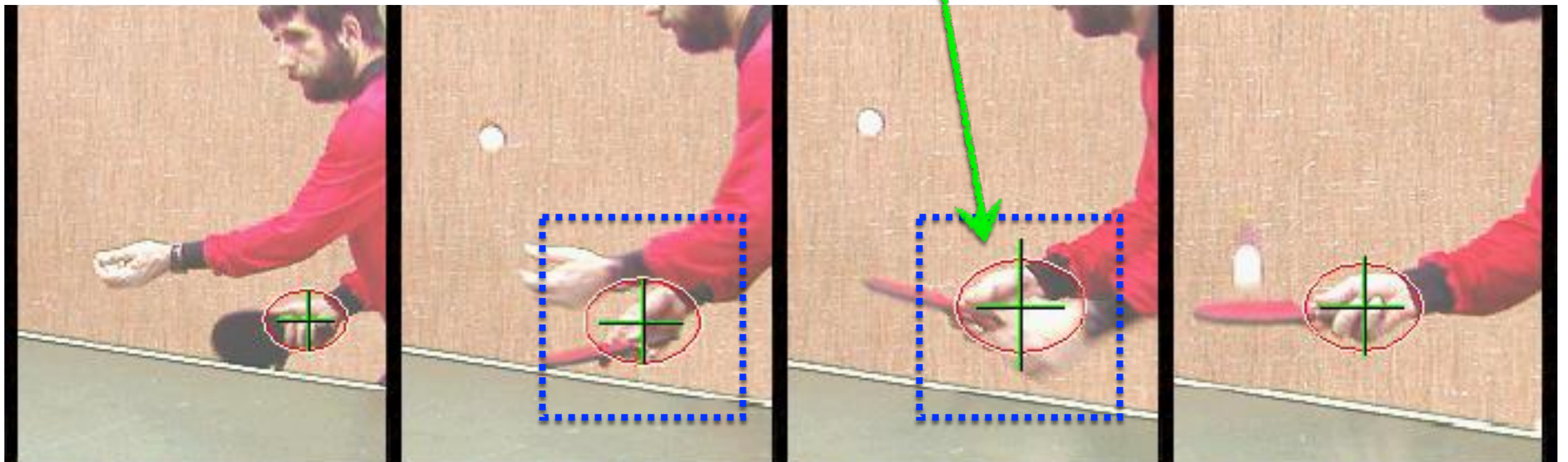
Compute a descriptor for the new target



Target

$q$

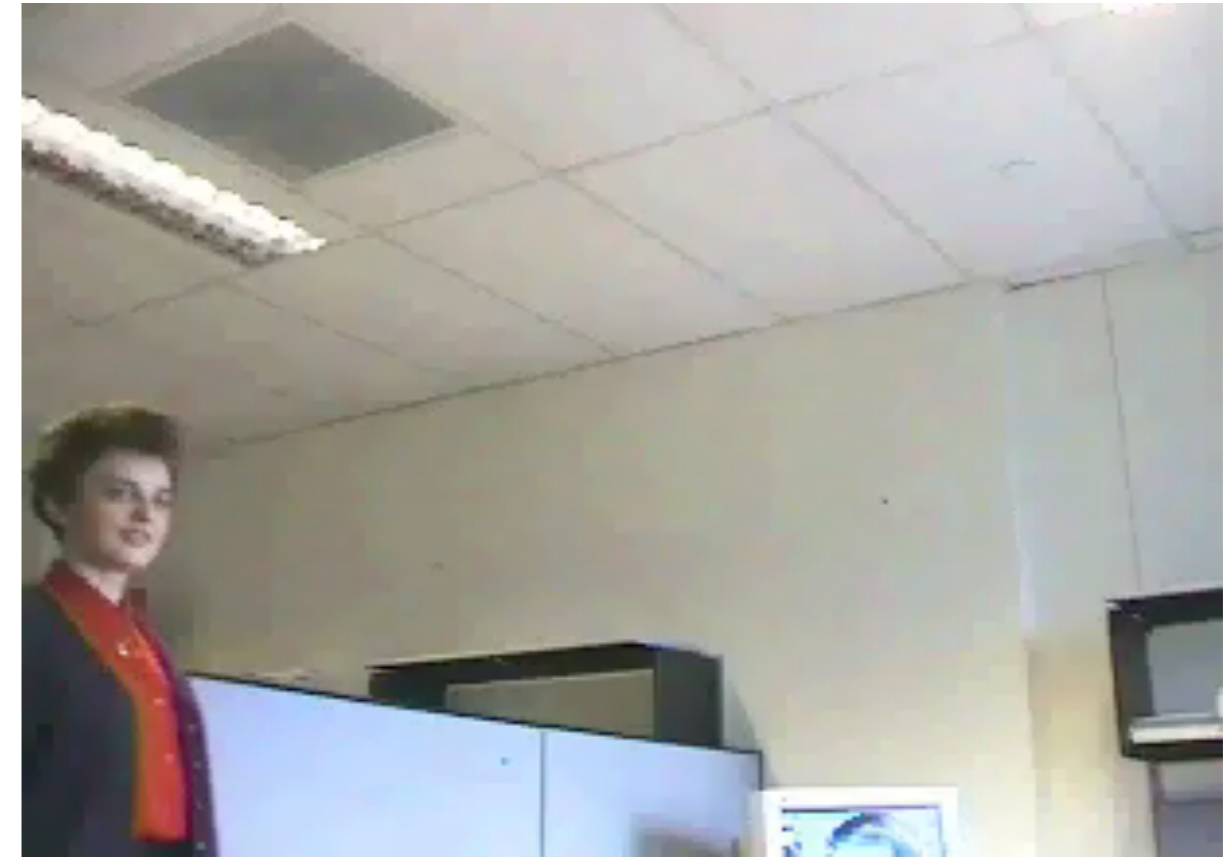
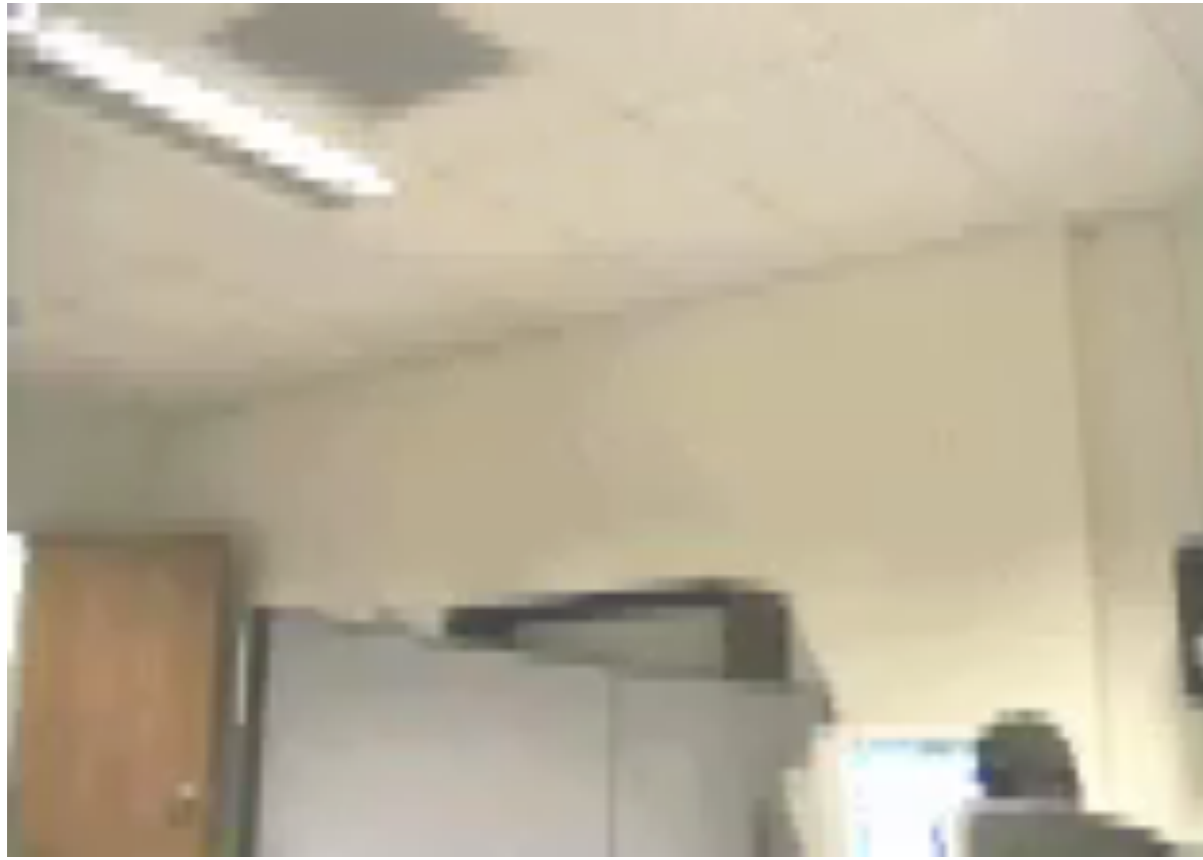
Search for similar descriptor in neighborhood in next frame



Target

Candidate

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



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.