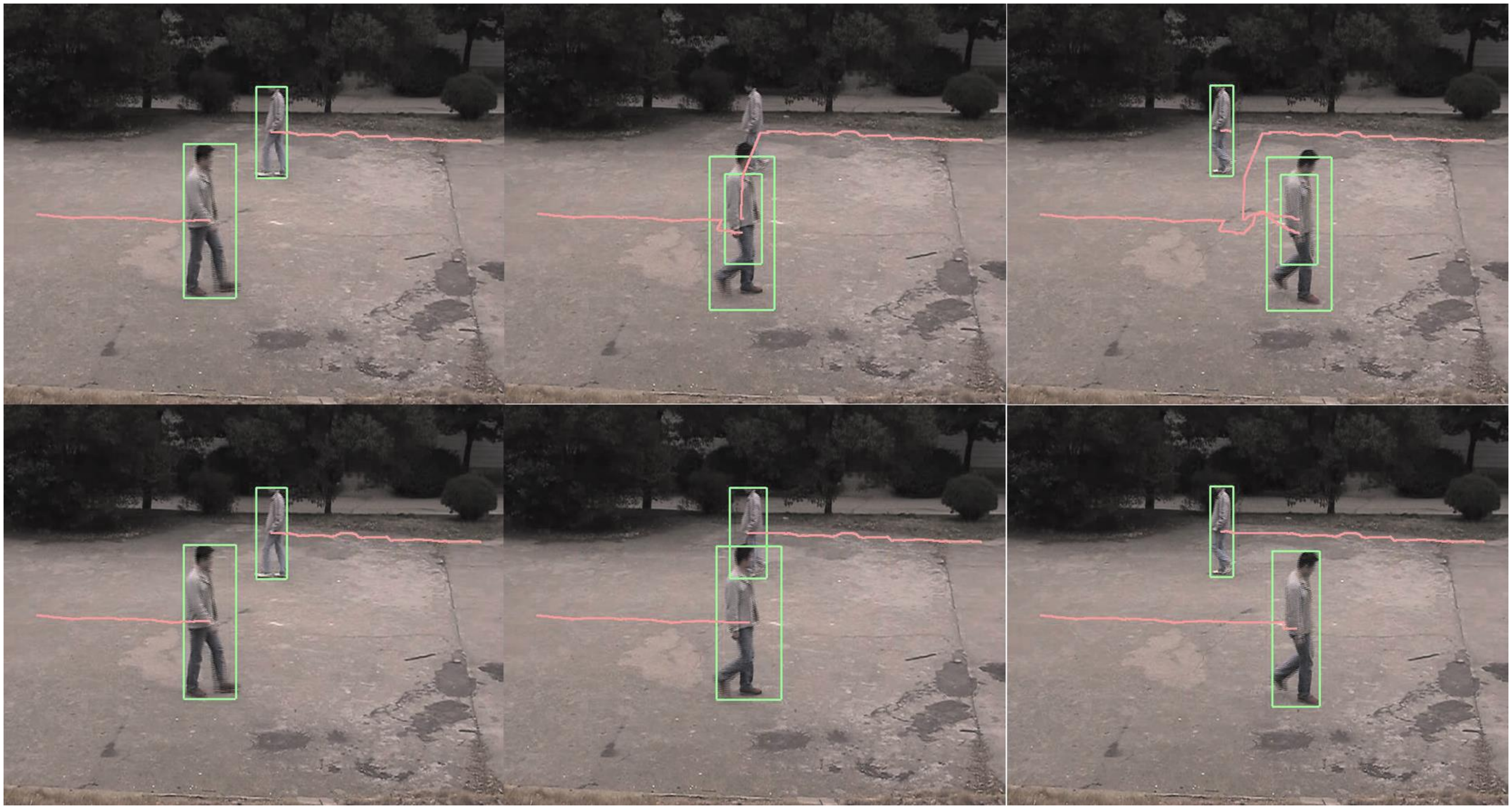
Tracking



16-385 Computer Vision
Spring 2018, Lecture 24

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

Course announcements

- Homework 6 has been posted and is due on April 20th.
  - Any questions about the homework?
  - How many of you have looked at/started/finished homework 6?

- This week Yannis' office hours will be on Wednesday 3-6 pm.
  - Added an extra hour to make up for change.

Overview of today’s lecture

- KLT tracker.
- Mean-shift algorithm.
- Kernel density estimation.
- Mean-shift tracker.
- Modern trackers.

Slide credits

Most of these slides were adapted from:

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

International Conference on Computational Photography

Carnegie Mellon University, Pittsburgh, PA

May 4 - 6, 2018

http://iccp2018.ece.cmu.edu/


International Conference on Computational Photography
Carnegie Mellon University, Pittsburgh, PA
May 4 - 6, 2018
http://iccp2018.ece.cmu.edu/


International Conference on Computational Photography
Carnegie Mellon University, Pittsburgh, PA
May 4 - 6, 2018
http://iccp2018.ece.cmu.edu/

Friday May 04	Morning session -- McConomy Auditorium
8:00am - 9:00am	Registration
9:00am - 9:15am	Welcome
9:15am - 10:30am	Session 1
10:30am - 11:00am	Coffee break
11:00am - 12pm	Keynote 1: Kyros Kutulakos
12:00pm - 1:30pm	Lunch in Wiegand gym
1:30pm - 2:45pm	Session 2
2:45pm - 3:15pm	Coffee break
3:15pm - 4:30pm	Session 3
4:30pm - 6:30pm	Optional lab tours (Details TBD)
6:30pm - 9:30pm	Reception in Phipps Outdoor garden

Saturday May 05	Morning session -- McConomy Auditorium
8:30am - 9:15am	Registration
9:15am - 10:30am	Session 4
10:30am - 11:00am	Coffee break
11:00am - 12:00pm	Keynote 2: Quyen Nguyen
12:00pm - 1:30pm	Lunch in Wiegand gym
1:30pm - 3:10pm	Session 5
3:10pm - 6:30pm	Poster session in Wiegand gym

Sunday May 06	Morning session -- McConomy Auditorium
8:15am - 9:15am	Registration
9:15am - 10:30am	Session 6
10:30am - 11:00am	Coffee break
11:00am - 12:00pm	Keynote 3: Sönke Johnsen
12:00pm - 1:30pm	Lunch in Wiegand gym
1:30pm - 3:10pm	Session 7
3:10pm - 4:00pm	Concluding remarks and awards ceremony

All lectures and talks are free for CMU students!

(You only need to pay for registration to attend the free food events.)

Friday May 04	Morning session -- McConomy Auditorium
8:00am - 9:00am	Registration
9:00am - 9:15am	Welcome
9:15am - 10:30am	Session 1
10:30am - 11:00am	Coffee break
11:00am - 12pm	Keynote 1: Kyros Kutulakos
12:00pm - 1:30pm	Lunch in Wiegand gym
1:30pm - 2:45pm	Session 2
2:45pm - 3:15pm	Coffee break
3:15pm - 4:30pm	Session 3
4:30pm - 6:30pm	Optional lab tours (Details TBD)
6:30pm - 9:30pm	Reception in Phipps Outdoor garden

Saturday May 05	Morning session -- McConomy Auditorium
8:30am - 9:15am	Registration
9:15am - 10:30am	Session 4
10:30am - 11:00am	Coffee break
11:00am - 12:00pm	Keynote 2: Quyen Nguyen
12:00pm - 1:30pm	Lunch in Wiegand gym
1:30pm - 3:10pm	Session 5
3:10pm - 6:30pm	Poster session in Wiegand gym

Sunday May 06	Morning session -- McConomy Auditorium
8:15am - 9:15am	Registration
9:15am - 10:30am	Session 6
10:30am - 11:00am	Coffee break
11:00am - 12:00pm	Keynote 3: Sönke Johnsen
12:00pm - 1:30pm	Lunch in Wiegand gym
1:30pm - 3:10pm	Session 7
3:10pm - 4:00pm	Concluding remarks and awards ceremony

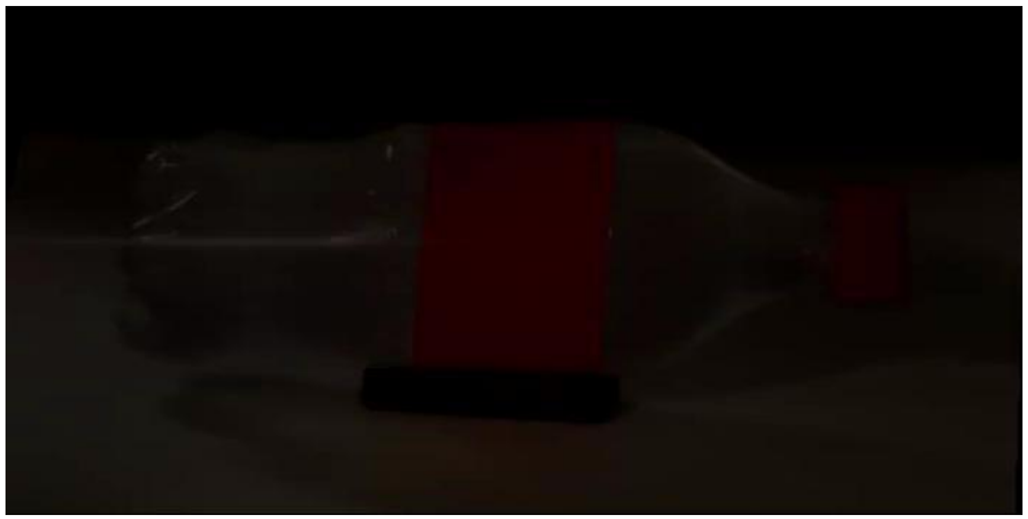
All lectures and talks are free for CMU students!

(You only need to pay for registration to attend the free food events.)

Everything in blue is free!

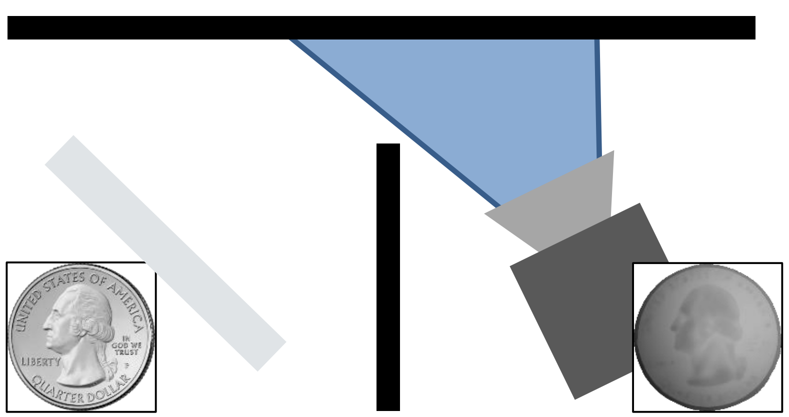
15-463/15-663/15-862 Computational Photography

Learn about this and other unconventional cameras – and build some on your own!



cameras that take video at the speed of light

cameras that measure depth in real time



cameras that see around corners

cameras that capture entire focal stacks

http://graphics.cs.cmu.edu/courses/15-463/

ICCP covers all of these

Kanade-Lucas-Tomasi (KLT) tracker

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

Feature-based tracking

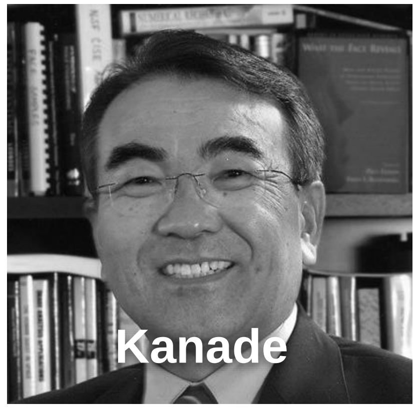
Up to now, we've been aligning entire images
but we can also track just small image regions too!
(sometimes called sparse tracking or sparse alignment)

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

How should we track them from frame to frame?

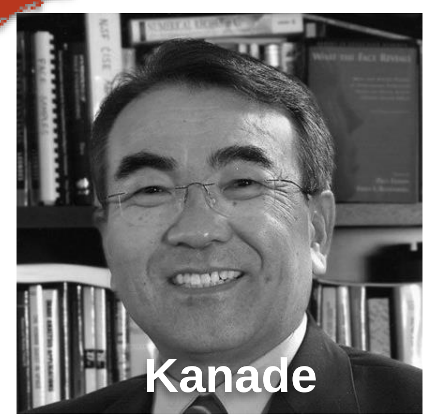
History of the

Kanade-Lucas-Tomasi (KLT) Tracker



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

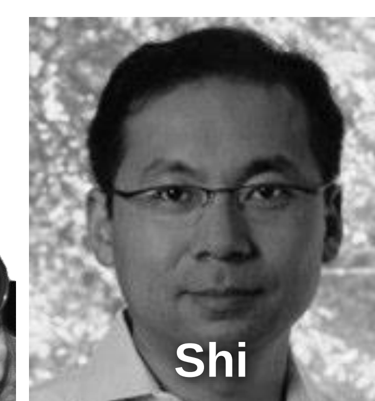
1981



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 …

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

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 …

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

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

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

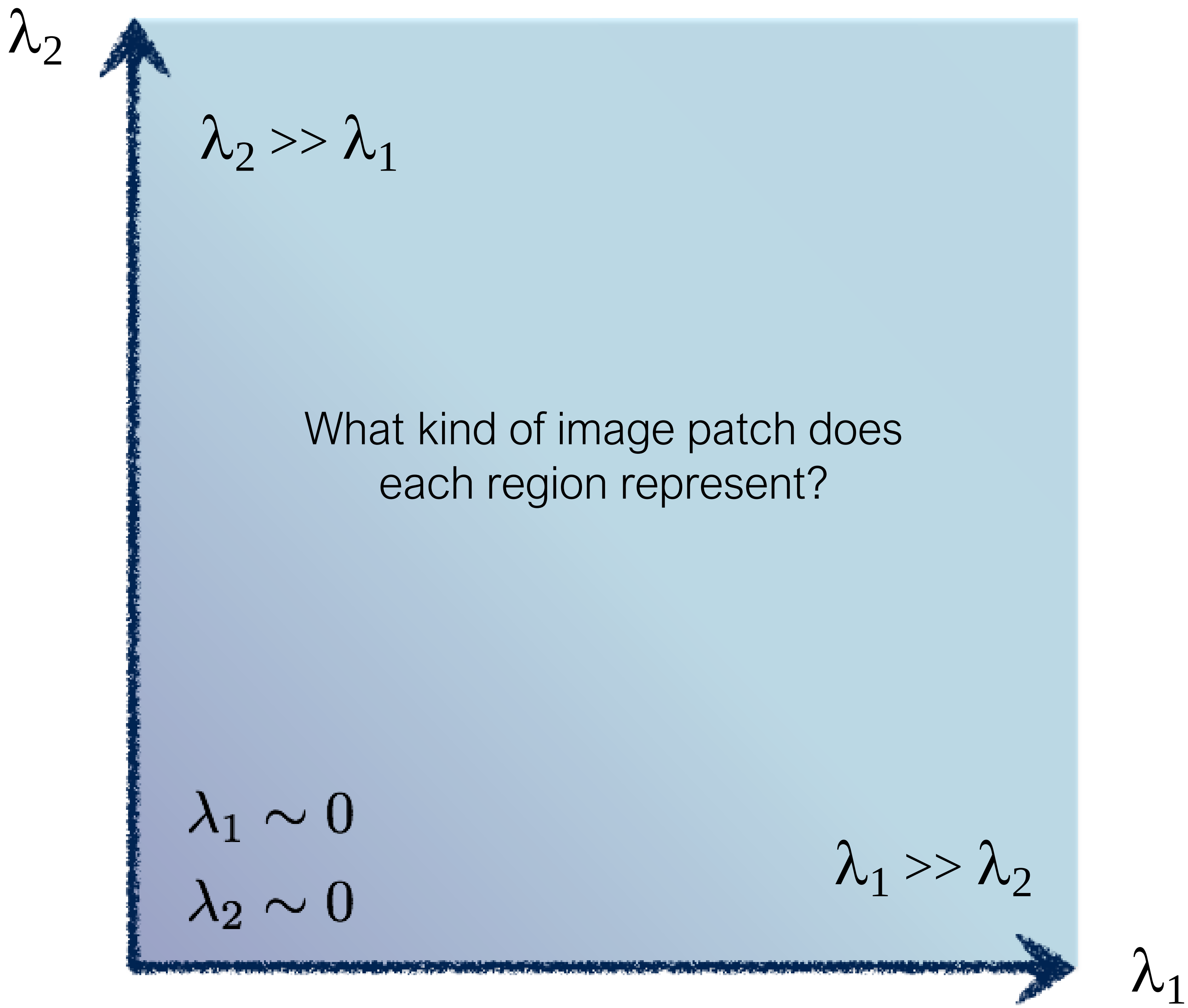
$$= \sum_{\boldsymbol{x}} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} I_x \\ I_y \end{bmatrix} \begin{bmatrix} I_x & I_y \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

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

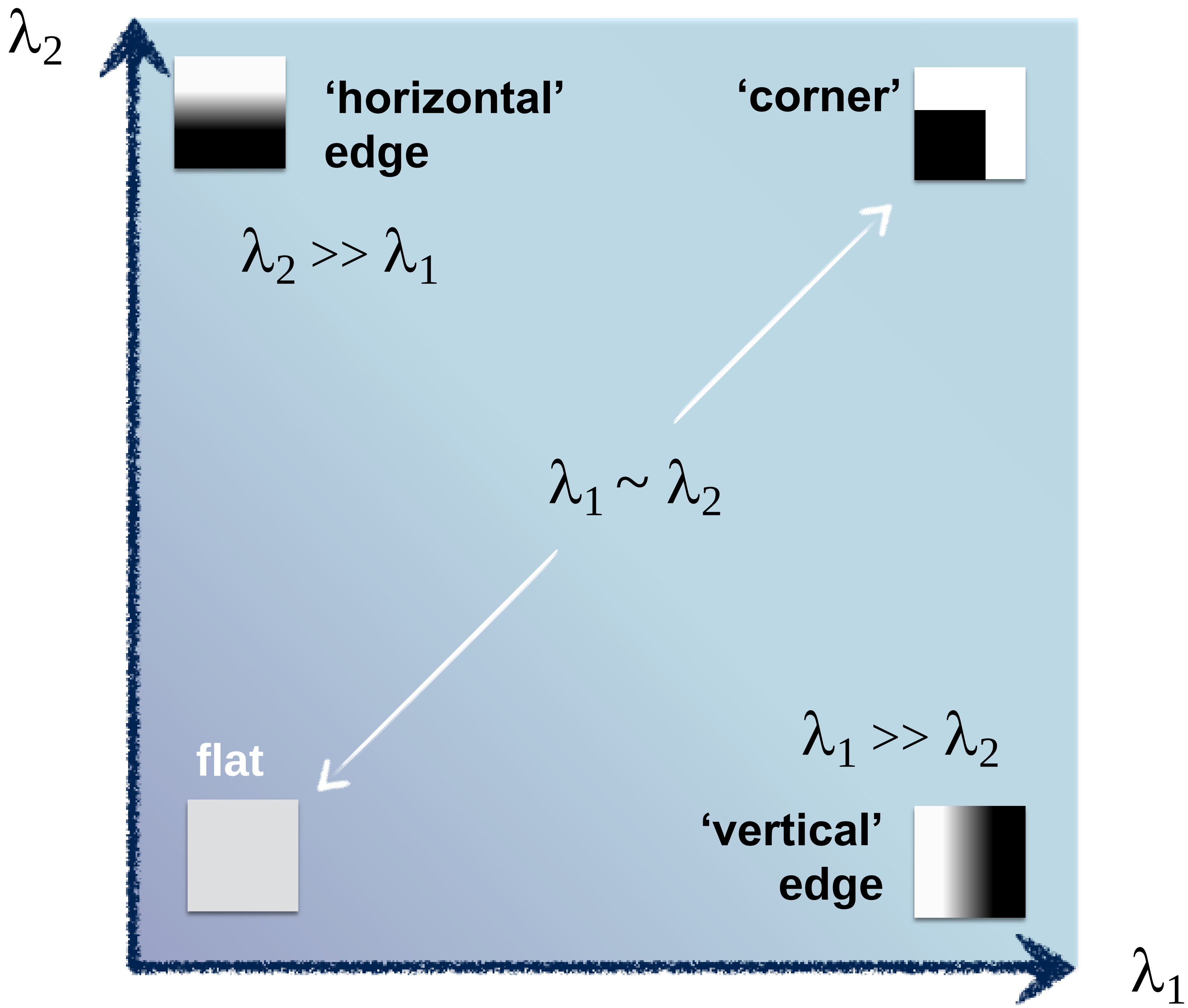
←when is this singular?

How are the eigenvalues related to image content?

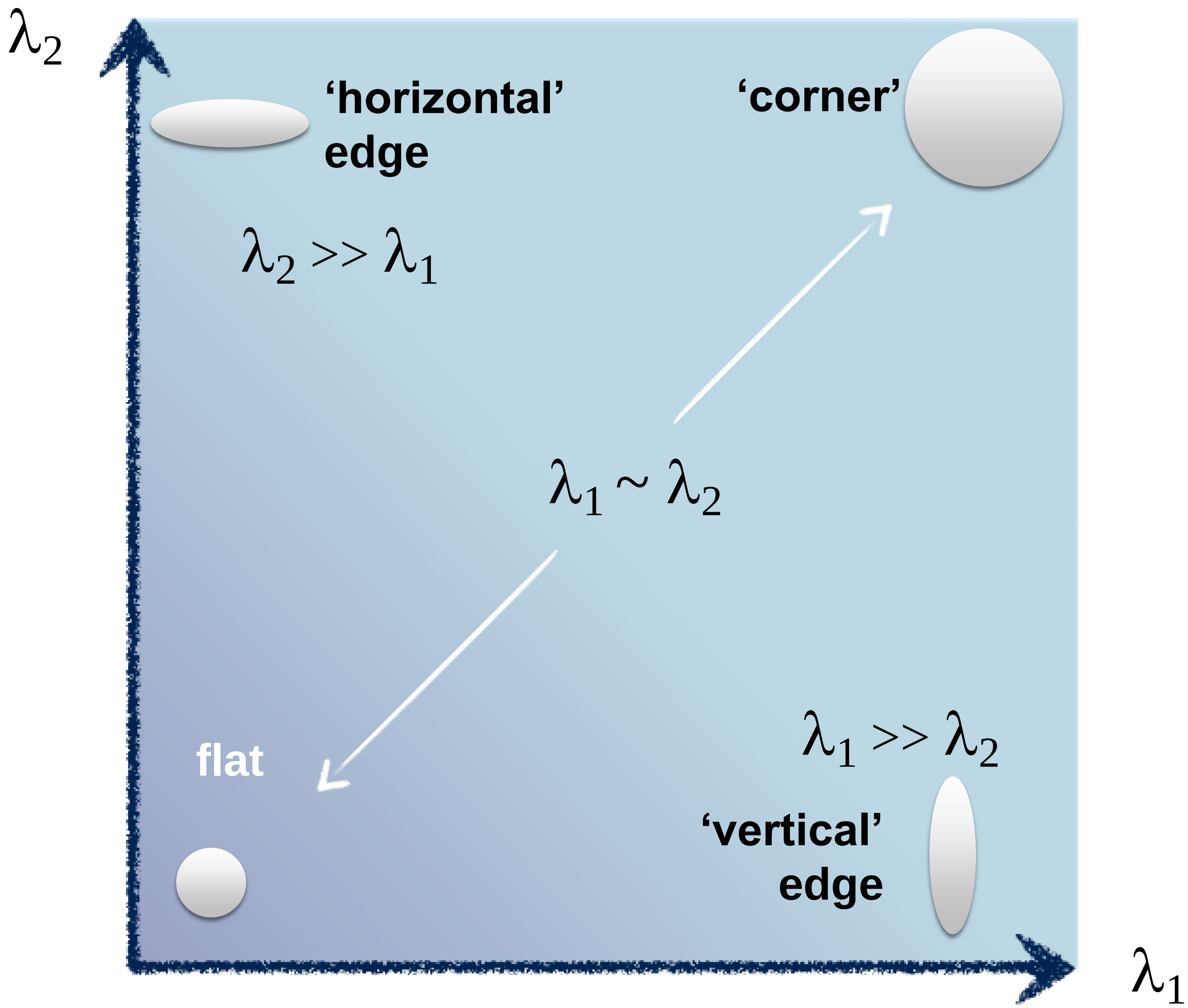
interpreting eigenvalues



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 milj.
ING

Mean Shift Algorithm

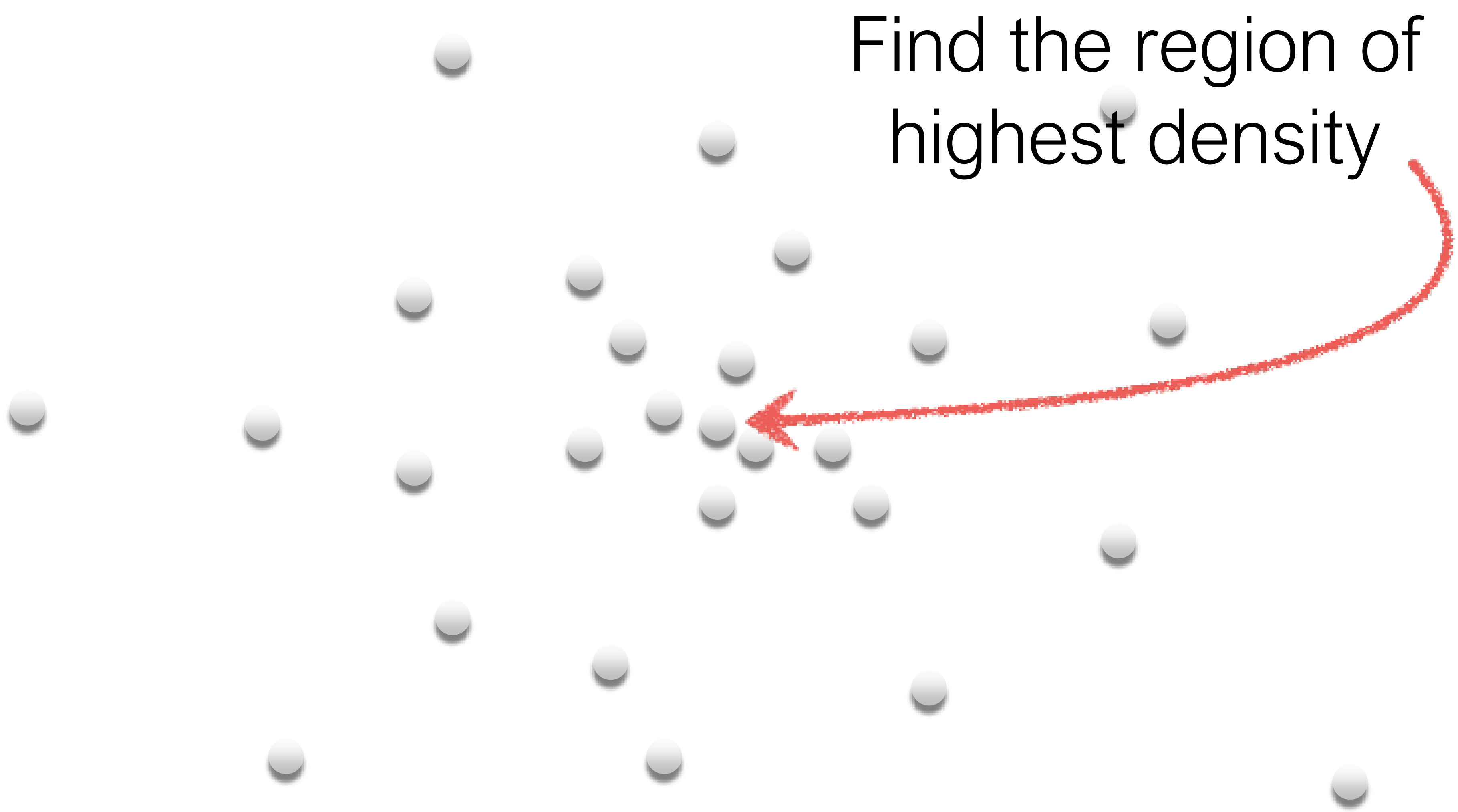
A ‘mode seeking’ algorithm

Fukunaga & Hostetler (1975)

Mean Shift Algorithm

A ‘mode seeking’ algorithm

Fukunaga & Hostetler (1975)



Mean Shift Algorithm

A ‘mode seeking’ algorithm

Fukunaga & Hostetler (1975)

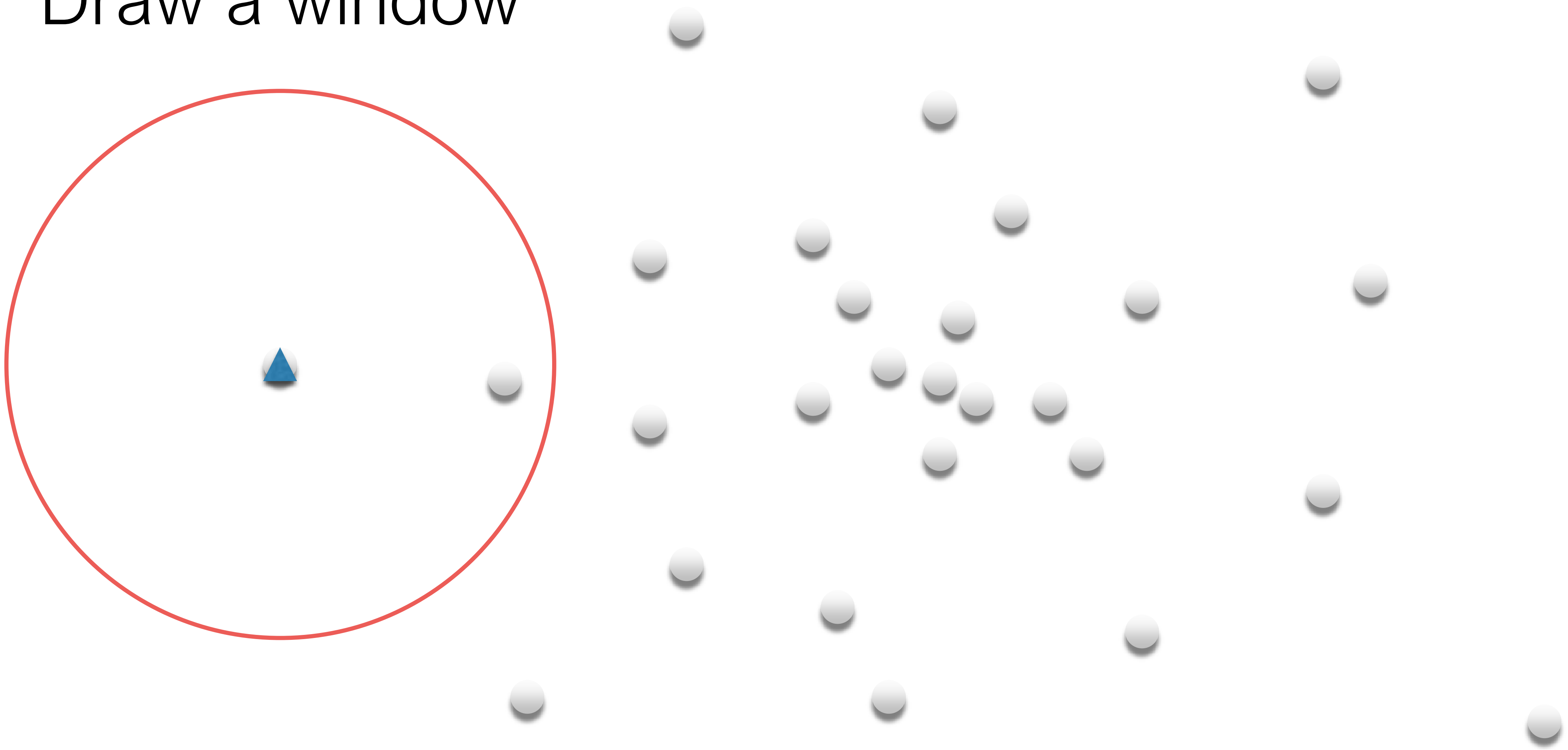
Pick a point

Mean Shift Algorithm

A ‘mode seeking’ algorithm

Fukunaga & Hostetler (1975)

Draw a window

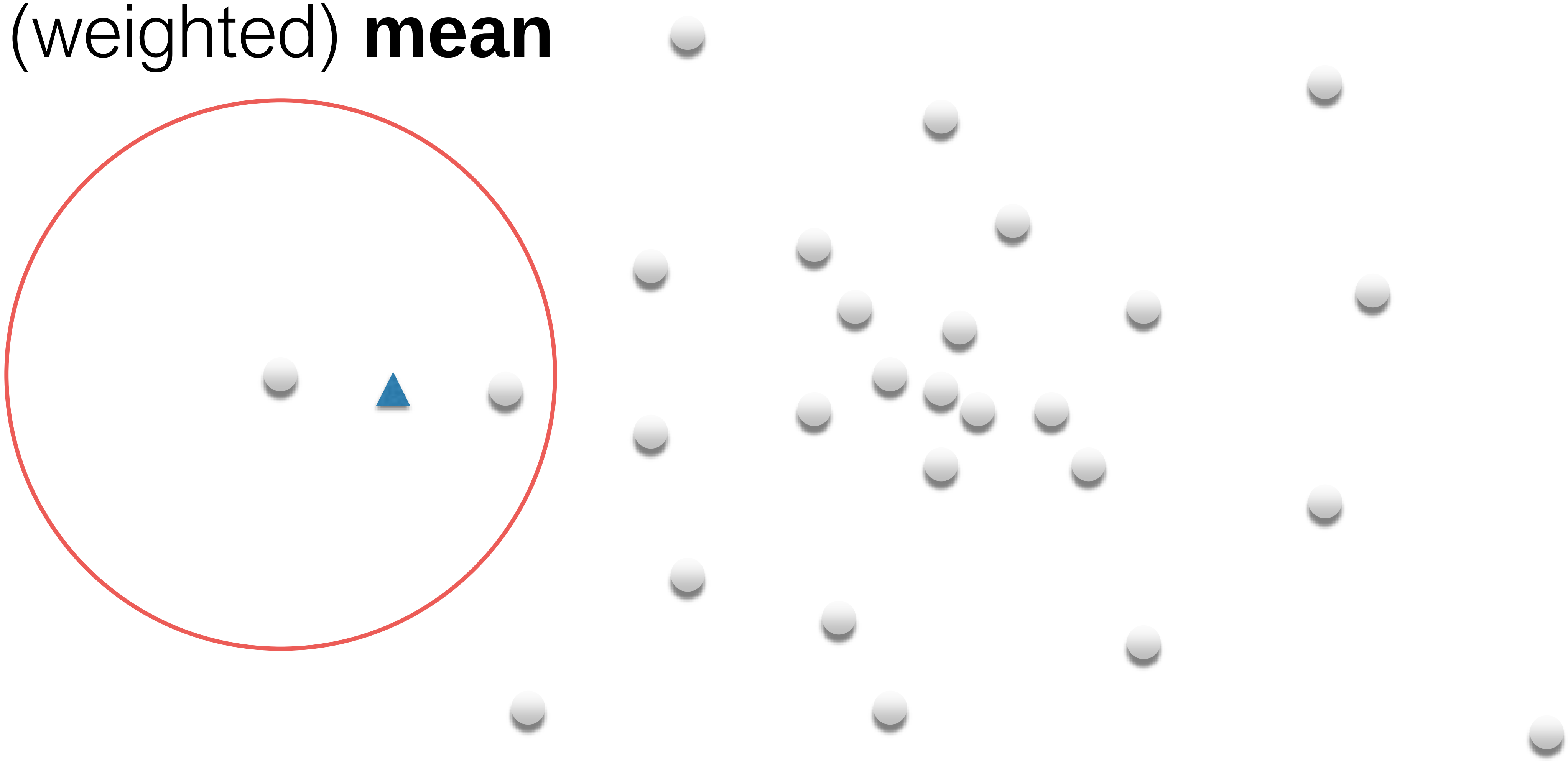


Mean Shift Algorithm

A ‘mode seeking’ algorithm

Fukunaga & Hostetler (1975)

Compute the
(weighted) **mean**

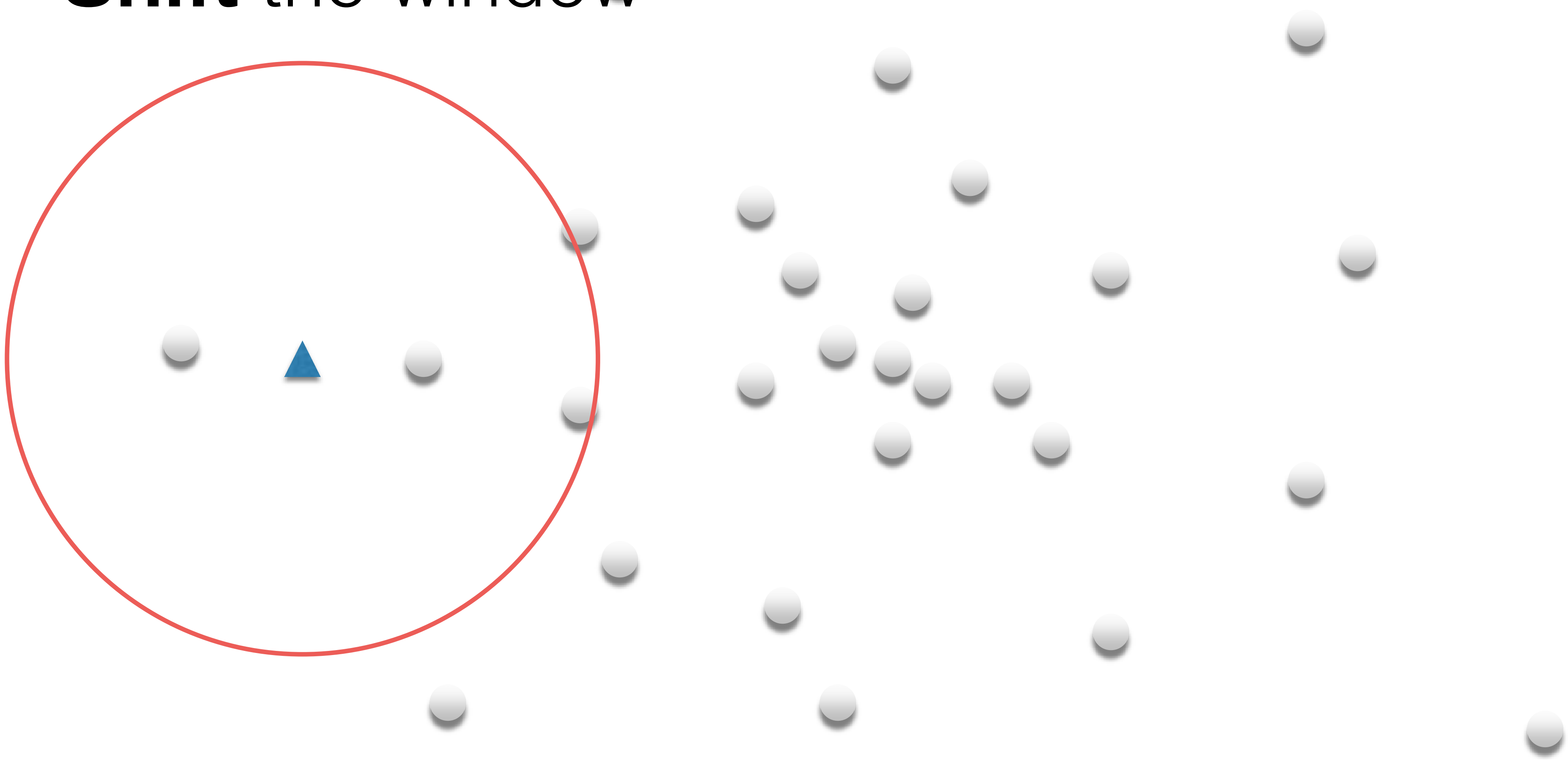


Mean Shift Algorithm

A ‘mode seeking’ algorithm

Fukunaga & Hostetler (1975)

Shift the window

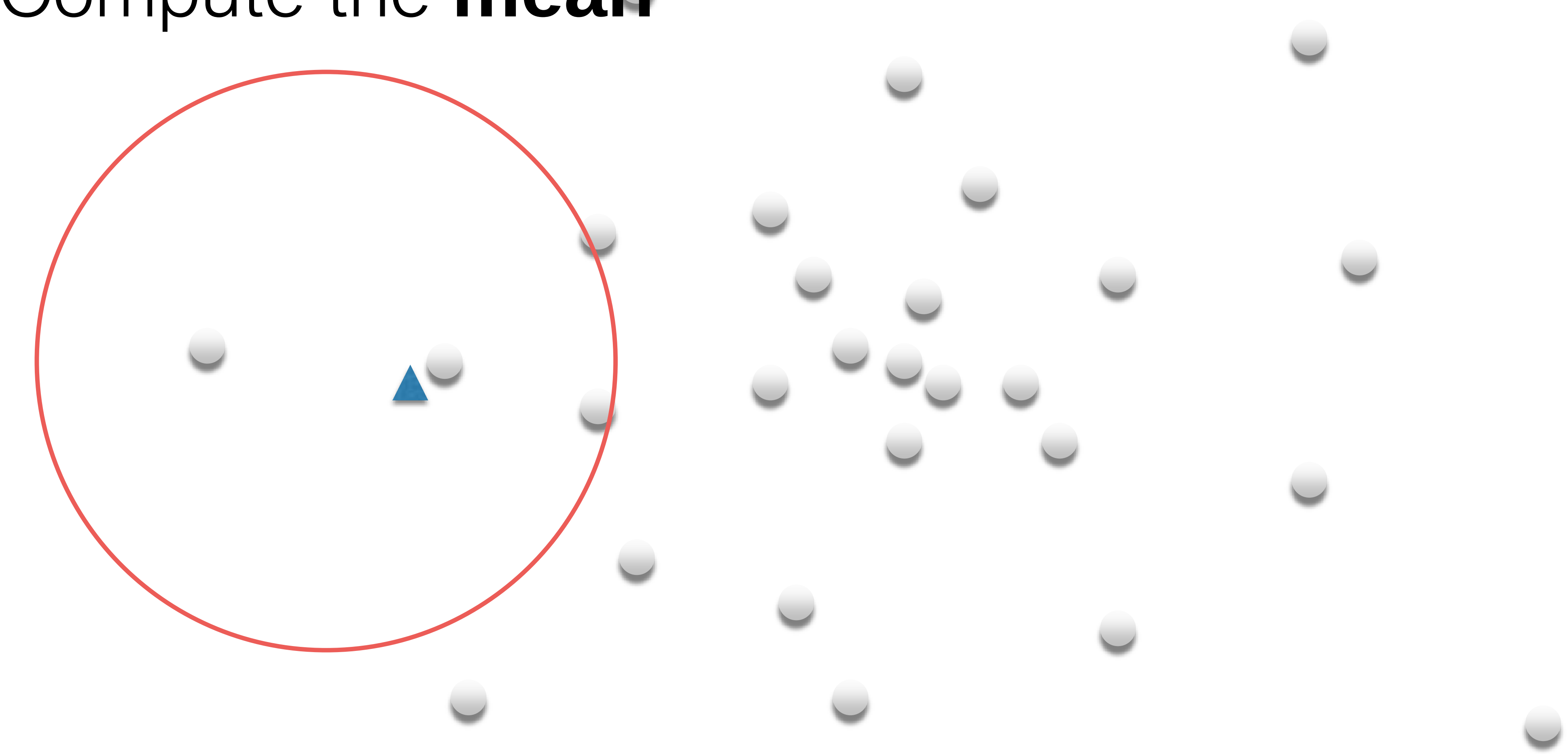


Mean Shift Algorithm

A ‘mode seeking’ algorithm

Fukunaga & Hostetler (1975)

Compute the **mean**

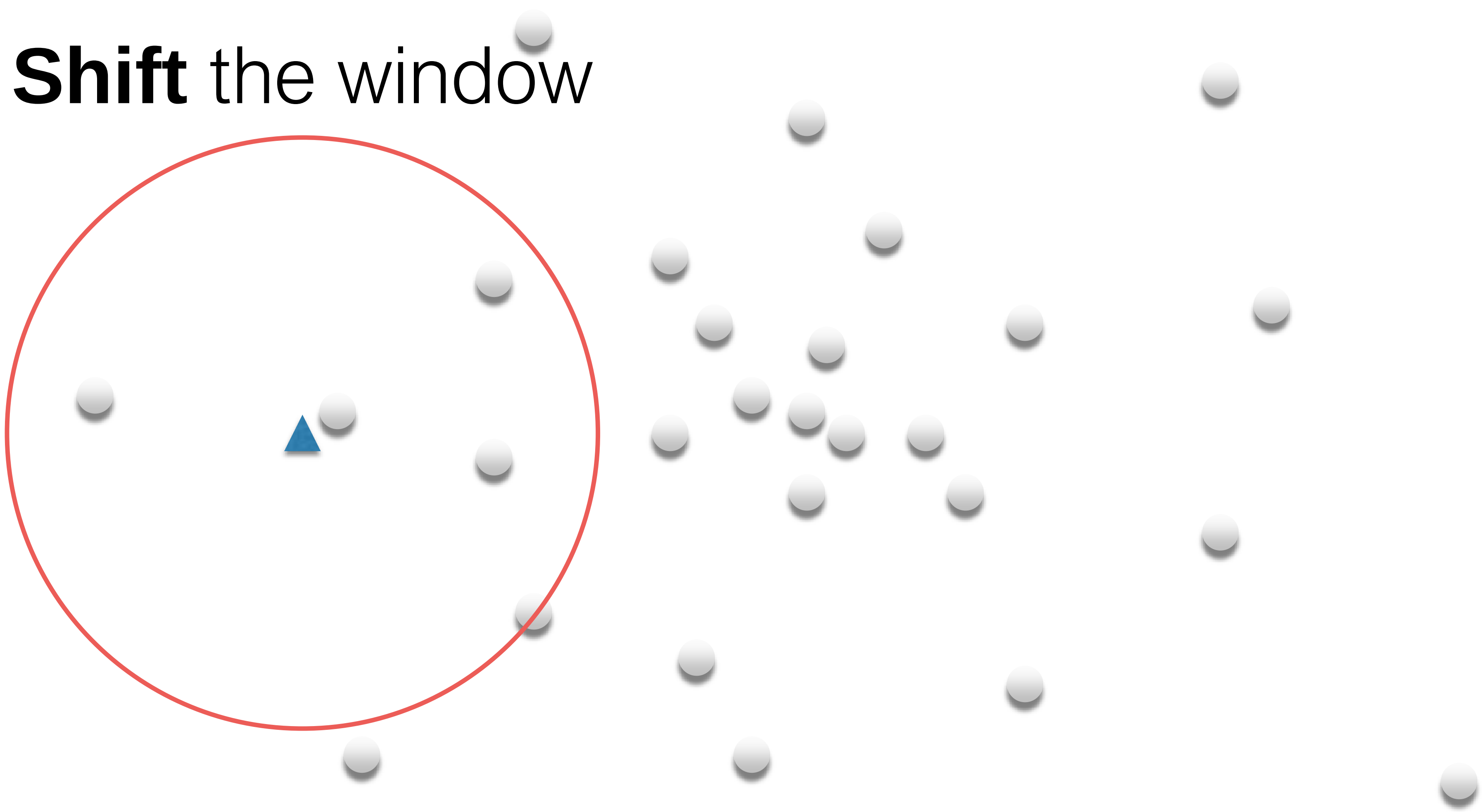


Mean Shift Algorithm

A ‘mode seeking’ algorithm

Fukunaga & Hostetler (1975)

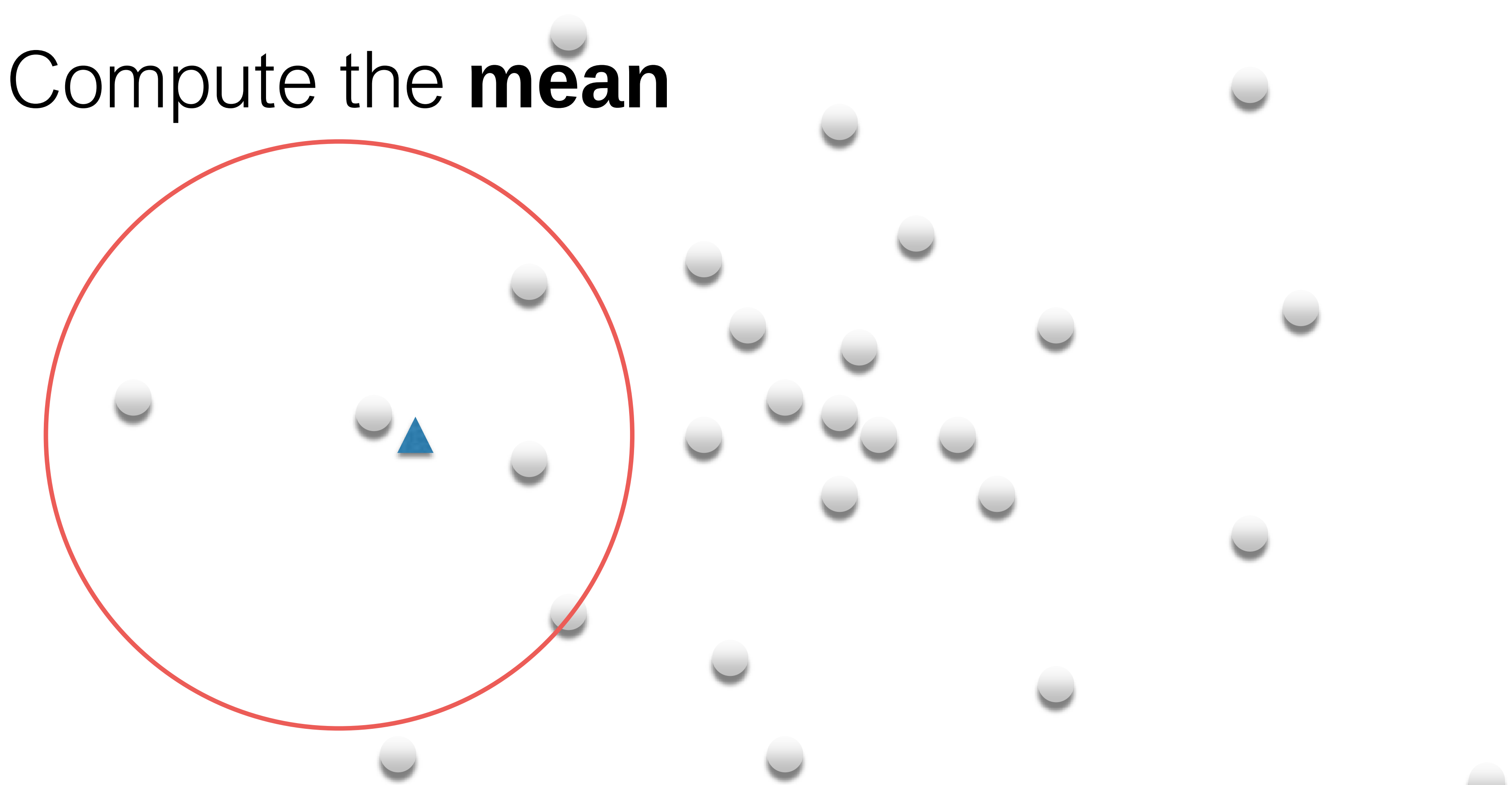
Shift the window



Mean Shift Algorithm

A ‘mode seeking’ algorithm

Fukunaga & Hostetler (1975)

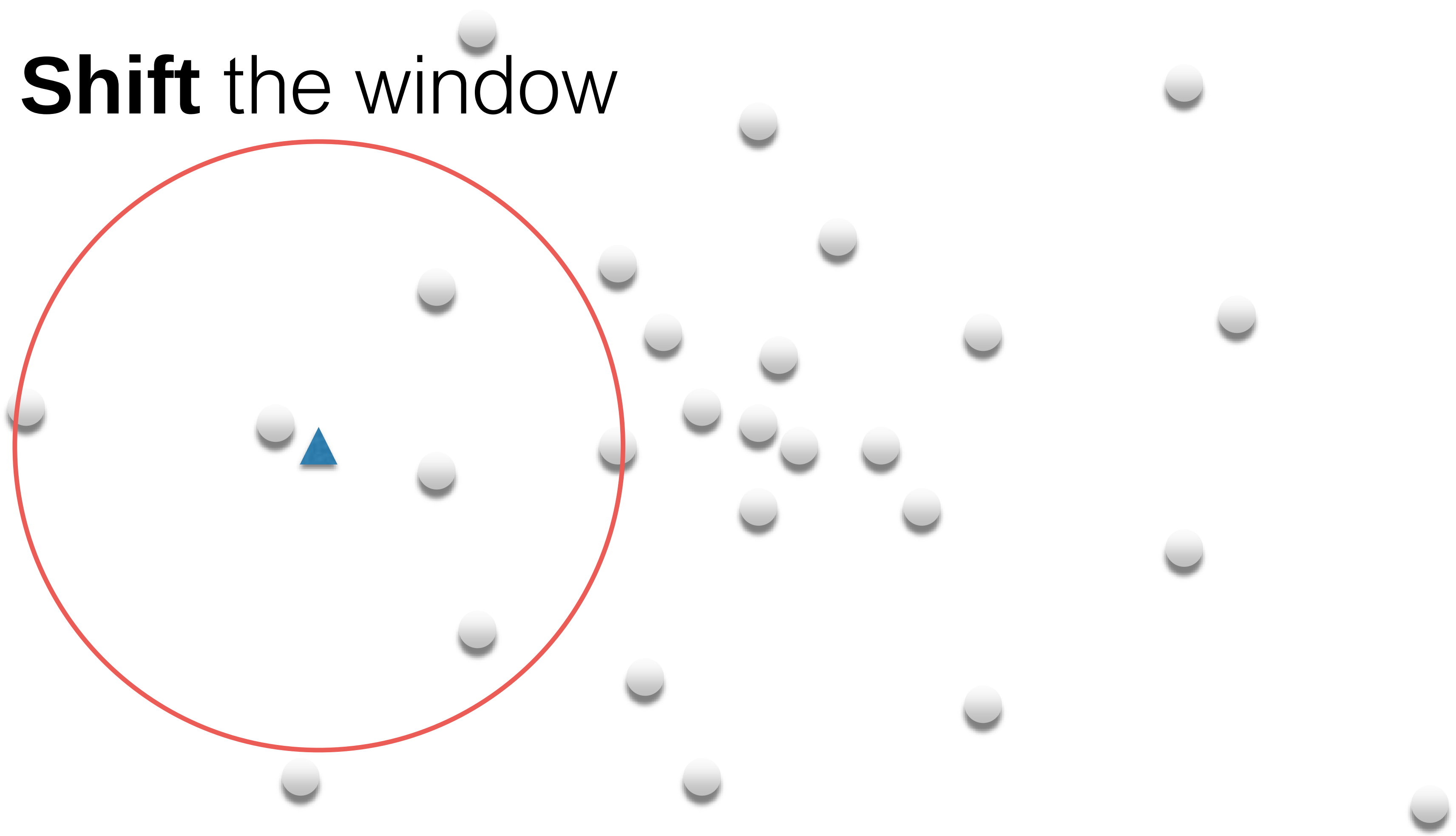


Mean Shift Algorithm

A 'mode seeking' algorithm

Fukunaga & Hostetler (1975)

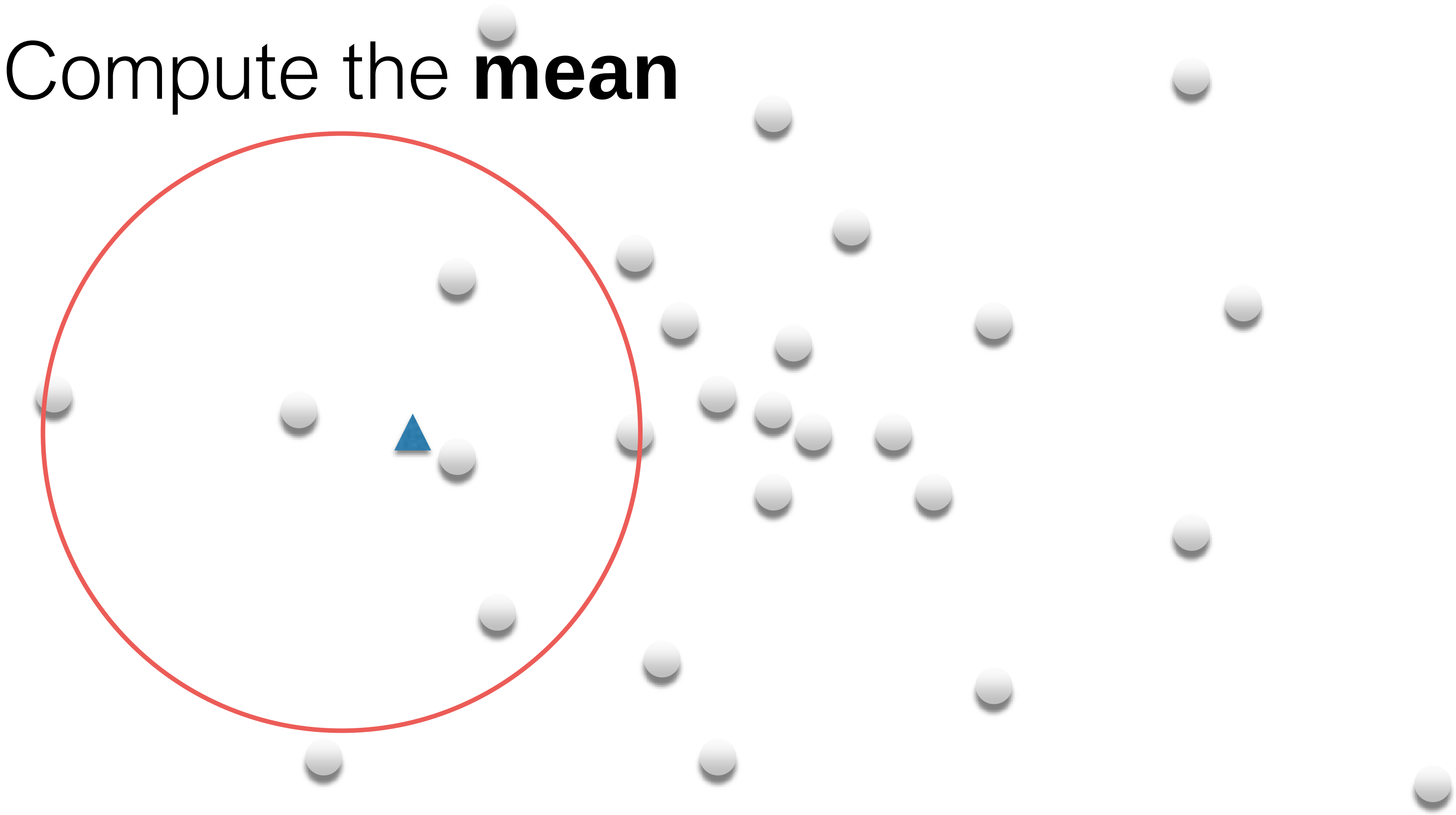
Shift the window



Mean Shift Algorithm

A ‘mode seeking’ algorithm

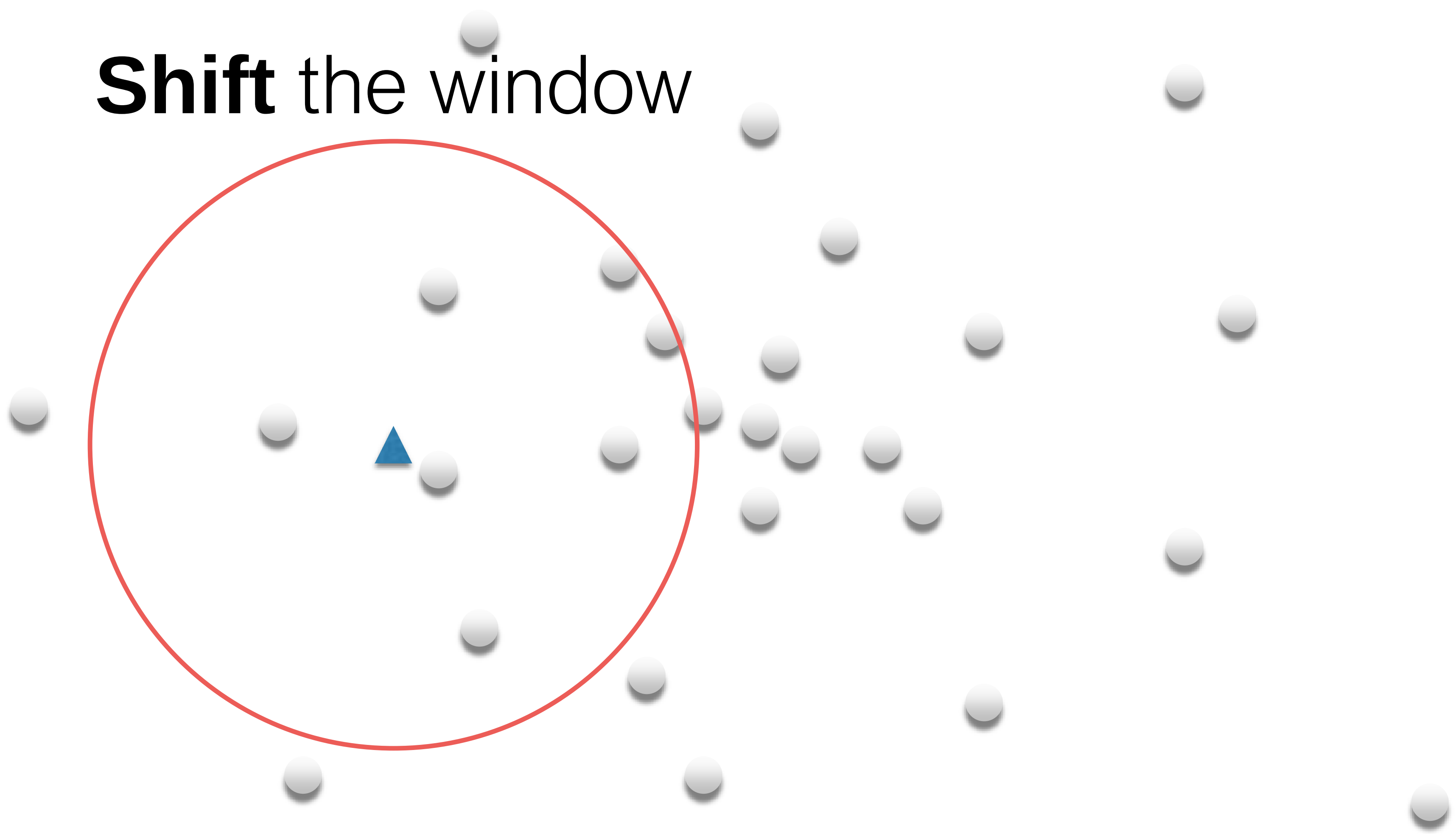
Fukunaga & Hostetler (1975)



Mean Shift Algorithm

A ‘mode seeking’ algorithm

Fukunaga & Hostetler (1975)

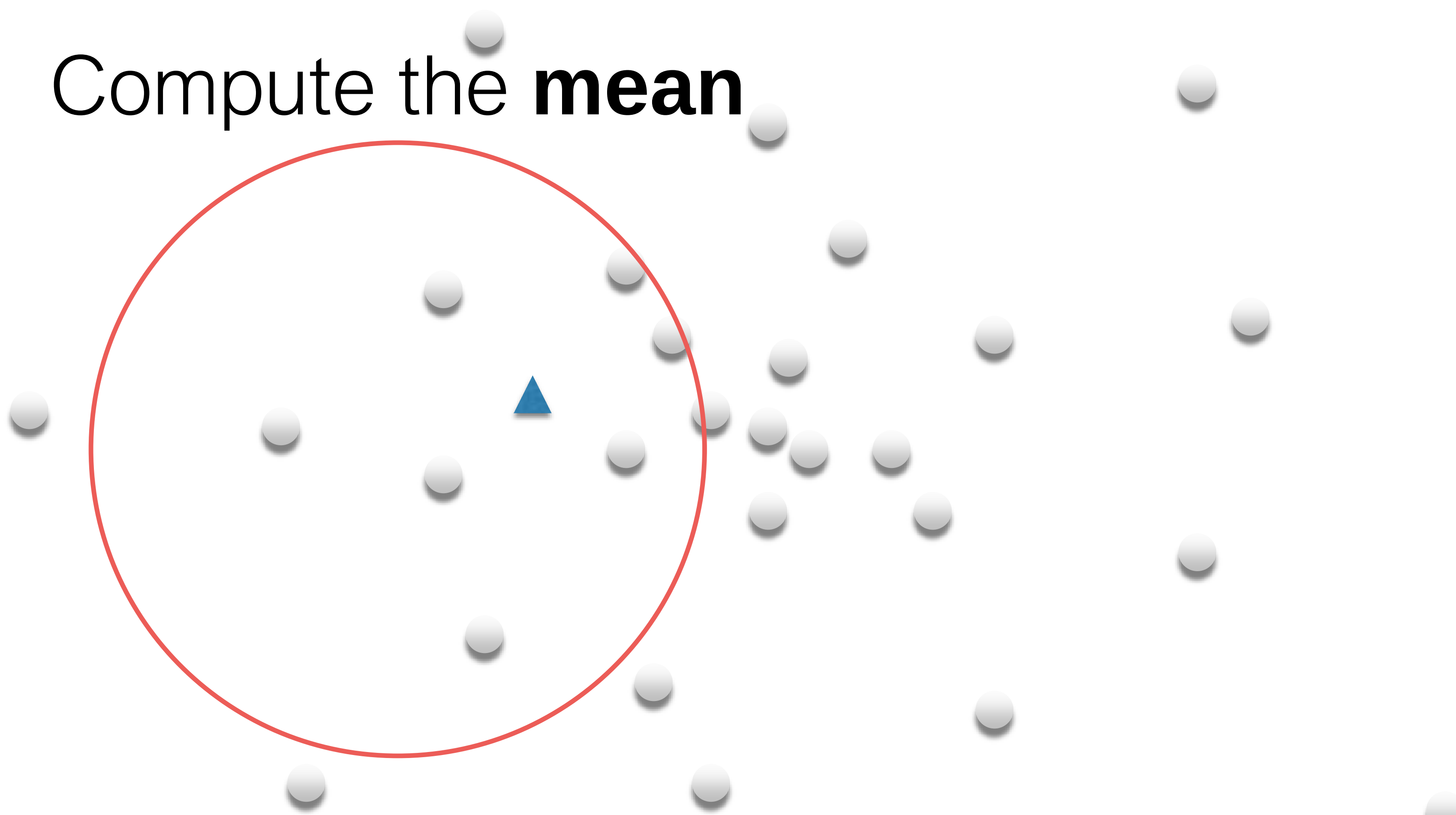


Mean Shift Algorithm

A ‘mode seeking’ algorithm

Fukunaga & Hostetler (1975)

Compute the **mean**

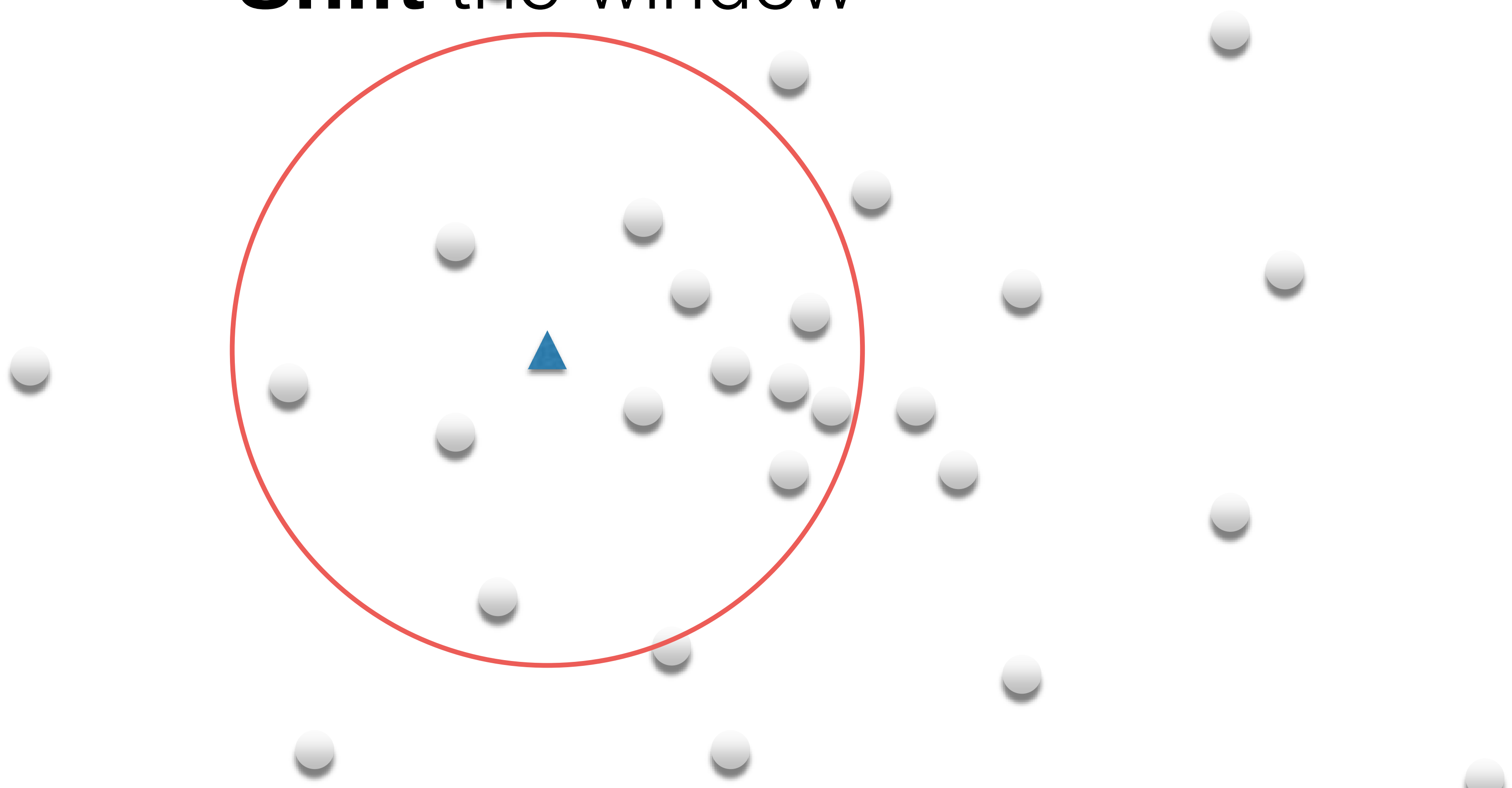


Mean Shift Algorithm

A ‘mode seeking’ algorithm

Fukunaga & Hostetler (1975)

Shift the window

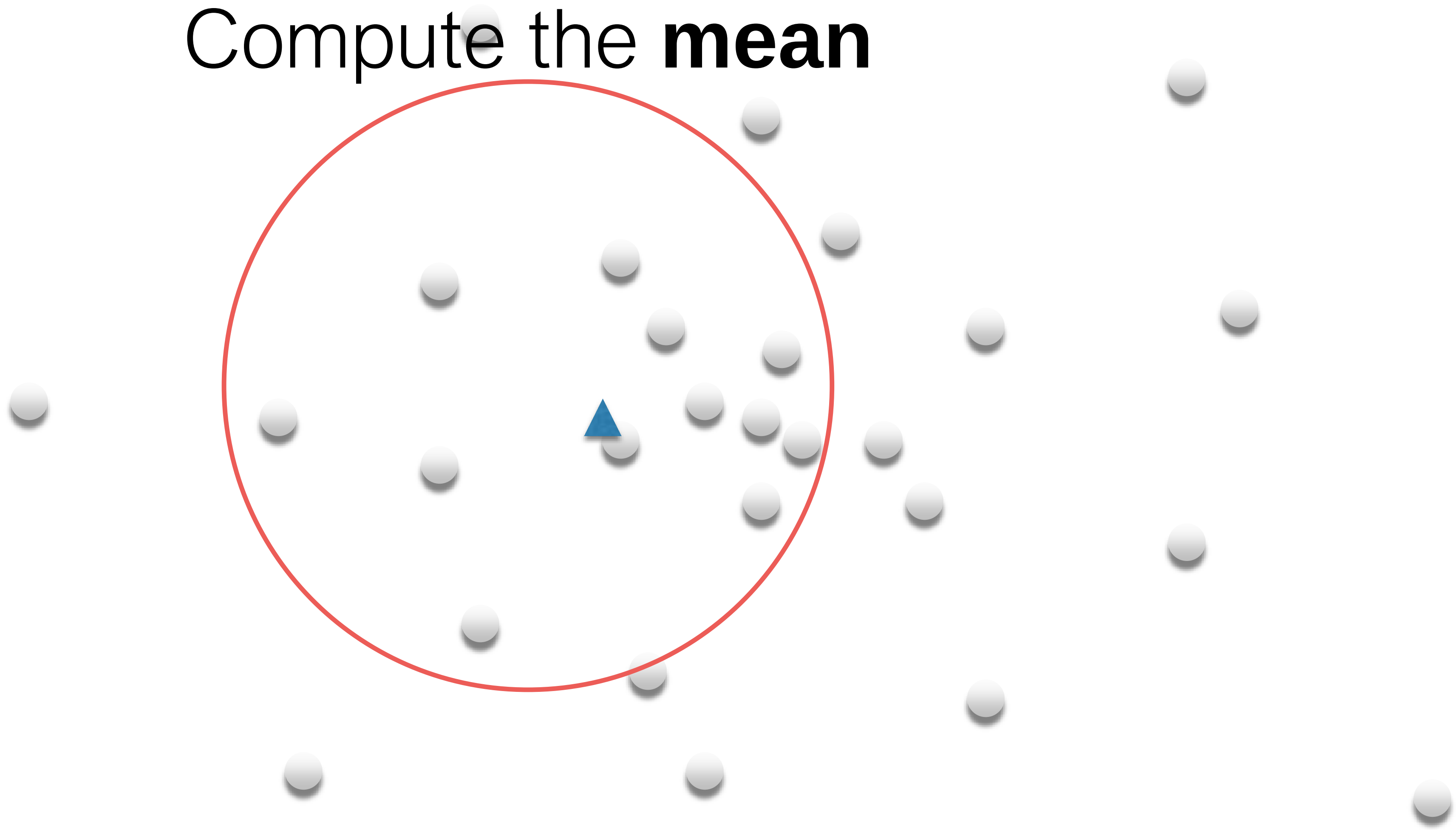


Mean Shift Algorithm

A ‘mode seeking’ algorithm

Fukunaga & Hostetler (1975)

Compute the **mean**

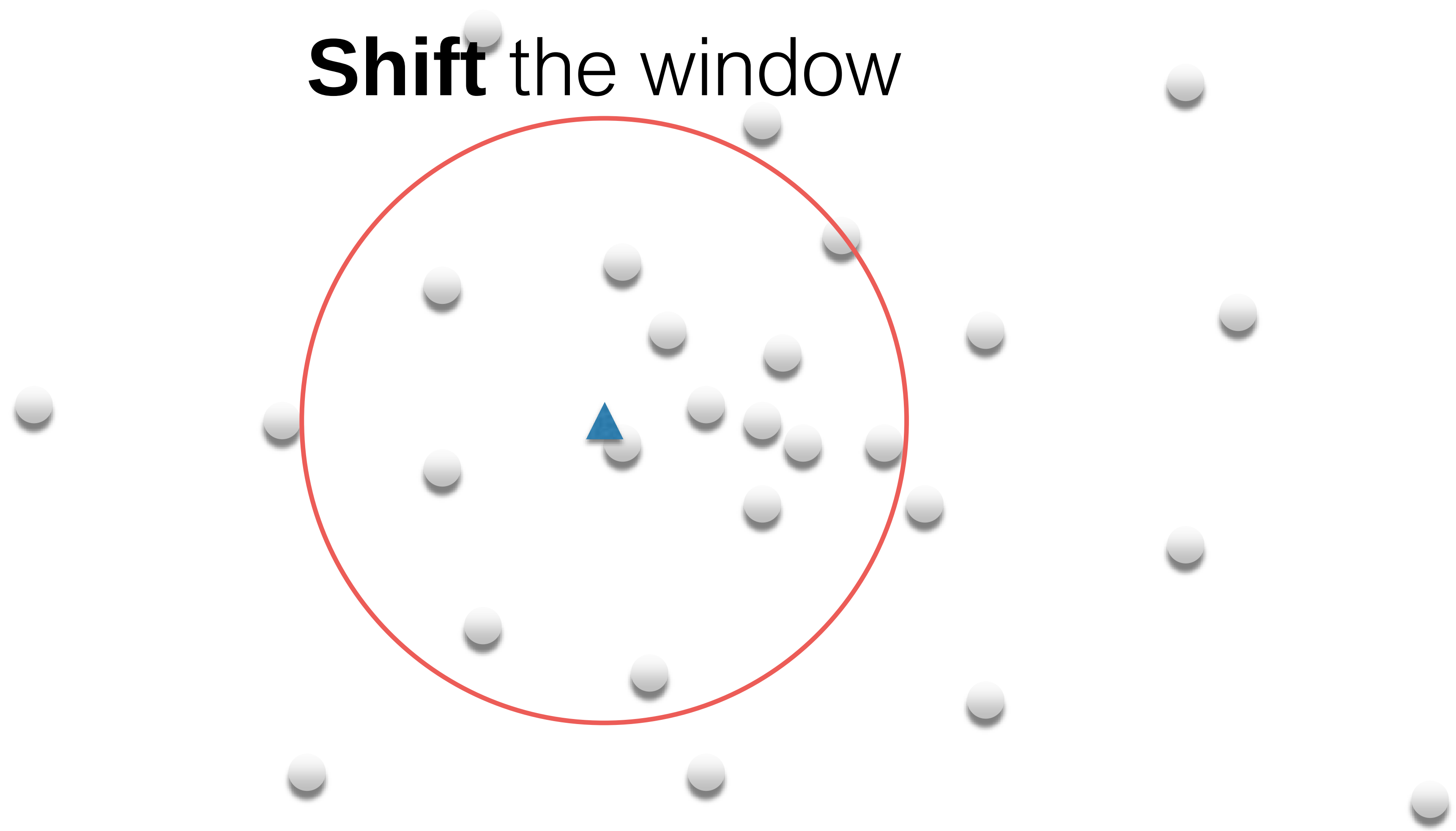


Mean Shift Algorithm

A ‘mode seeking’ algorithm

Fukunaga & Hostetler (1975)

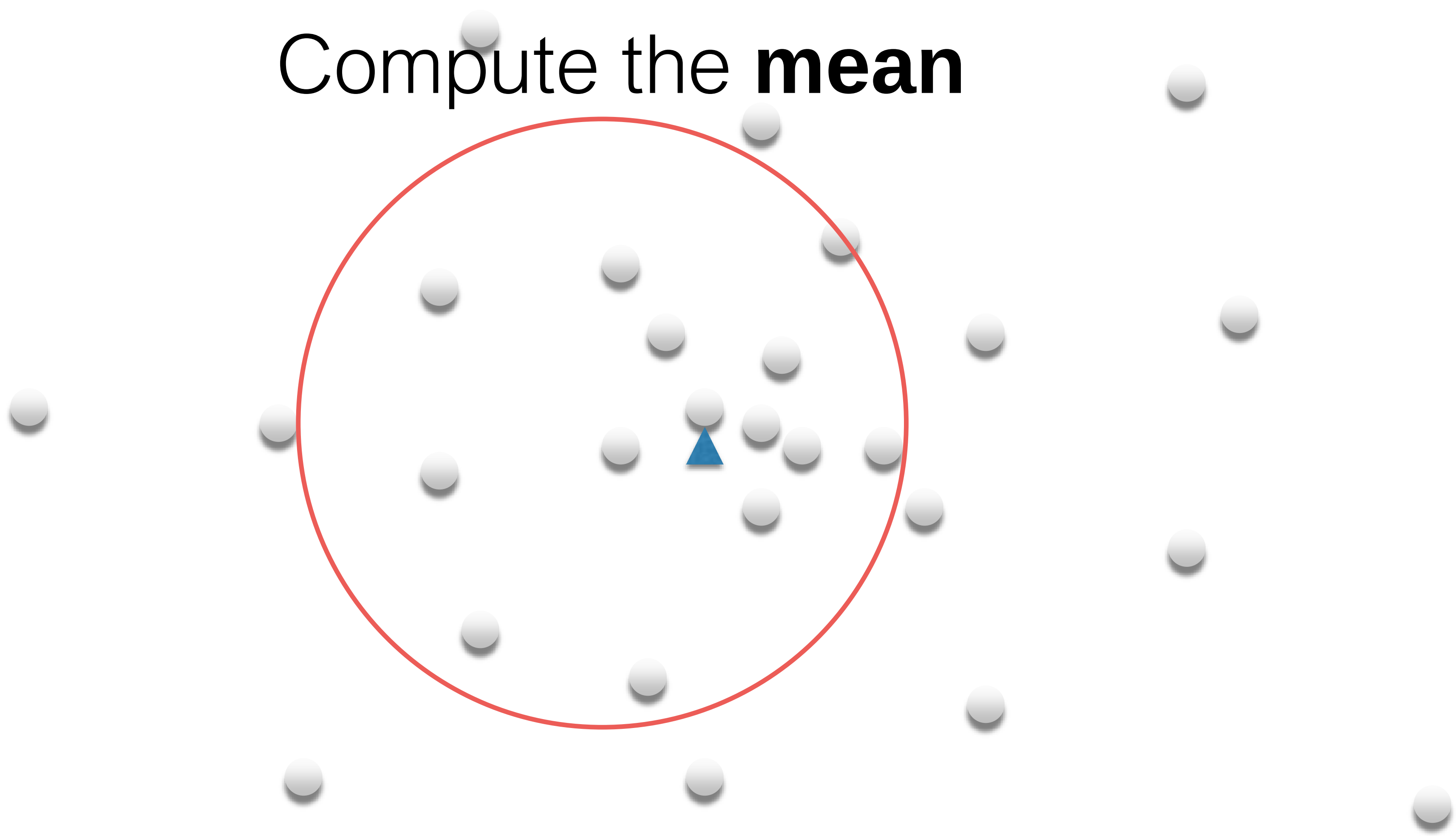
Shift the window



Mean Shift Algorithm

A ‘mode seeking’ algorithm

Fukunaga & Hostetler (1975)

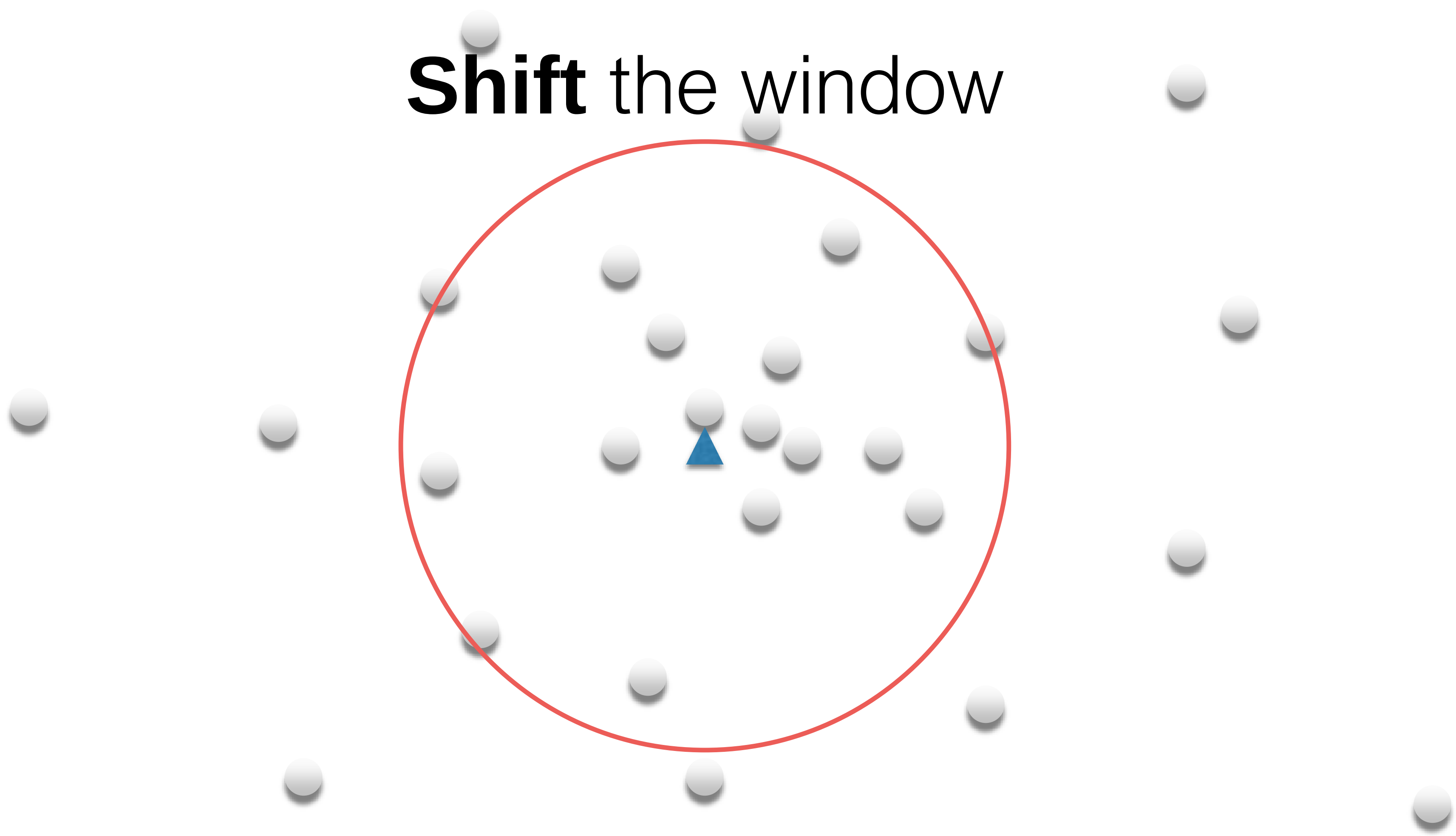


Mean Shift Algorithm

A 'mode seeking' algorithm

Fukunaga & Hostetler (1975)

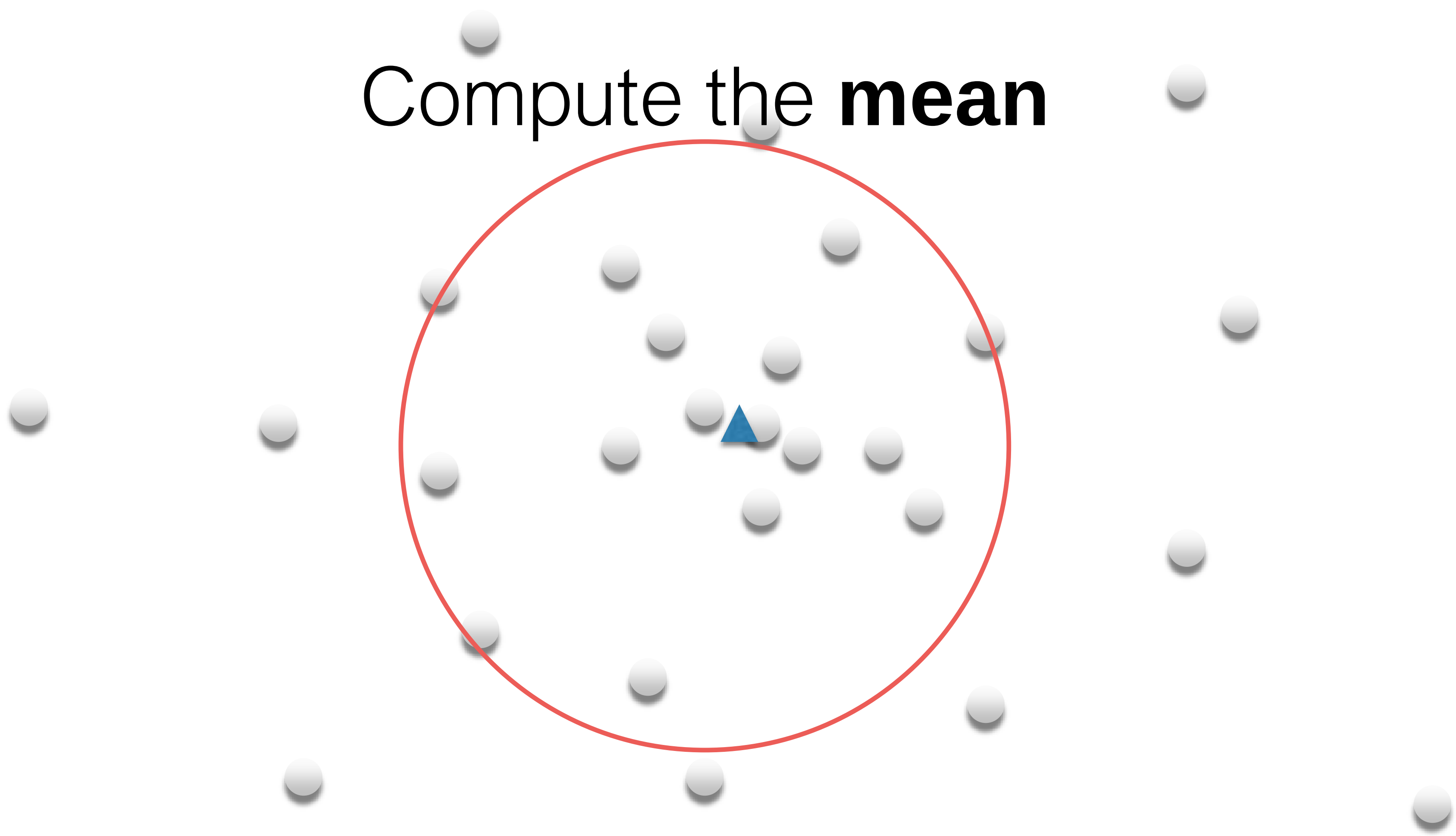
Shift the window



Mean Shift Algorithm

A ‘mode seeking’ algorithm

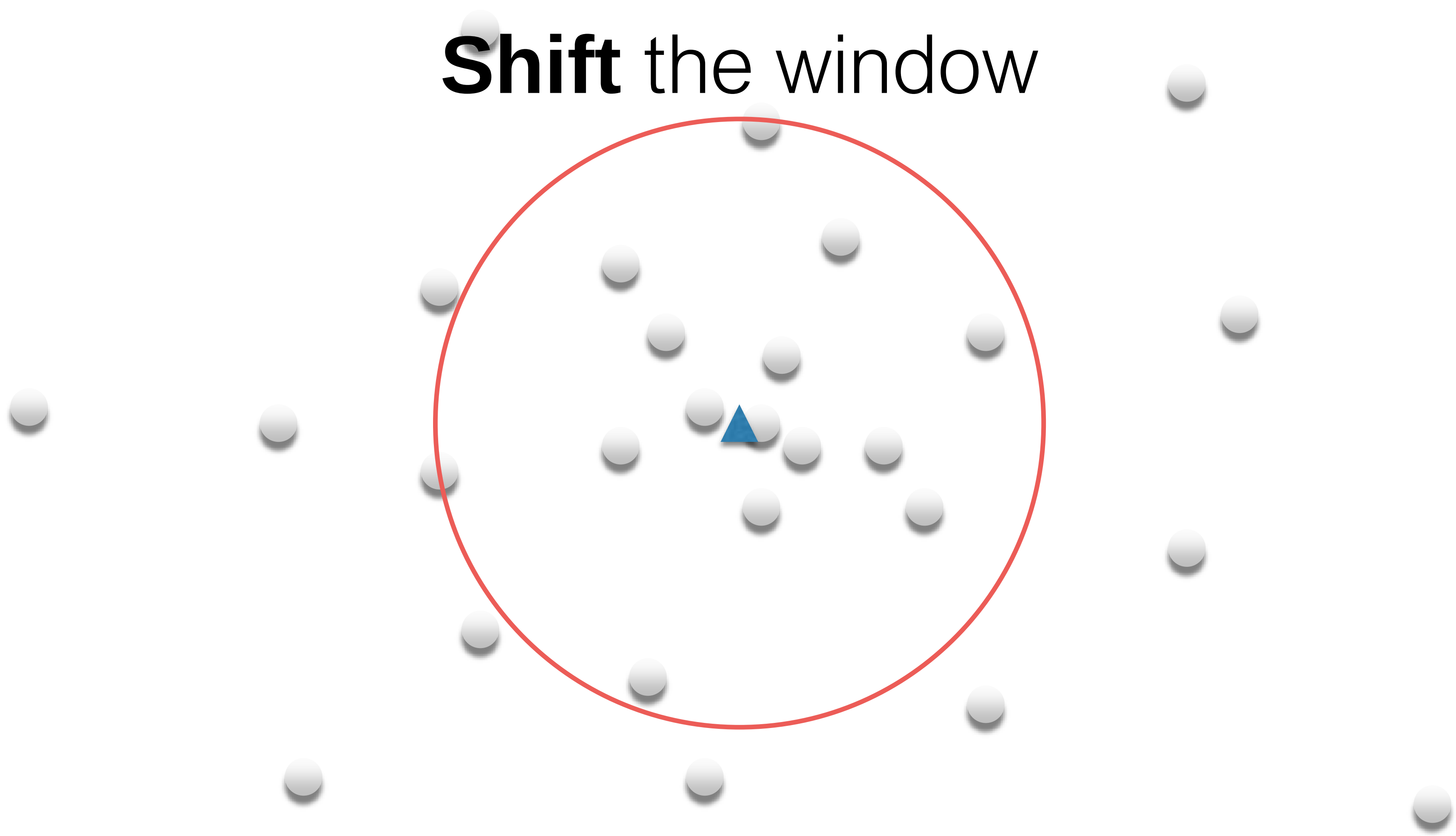
Fukunaga & Hostetler (1975)



Mean Shift Algorithm

A ‘mode seeking’ algorithm

Fukunaga & Hostetler (1975)

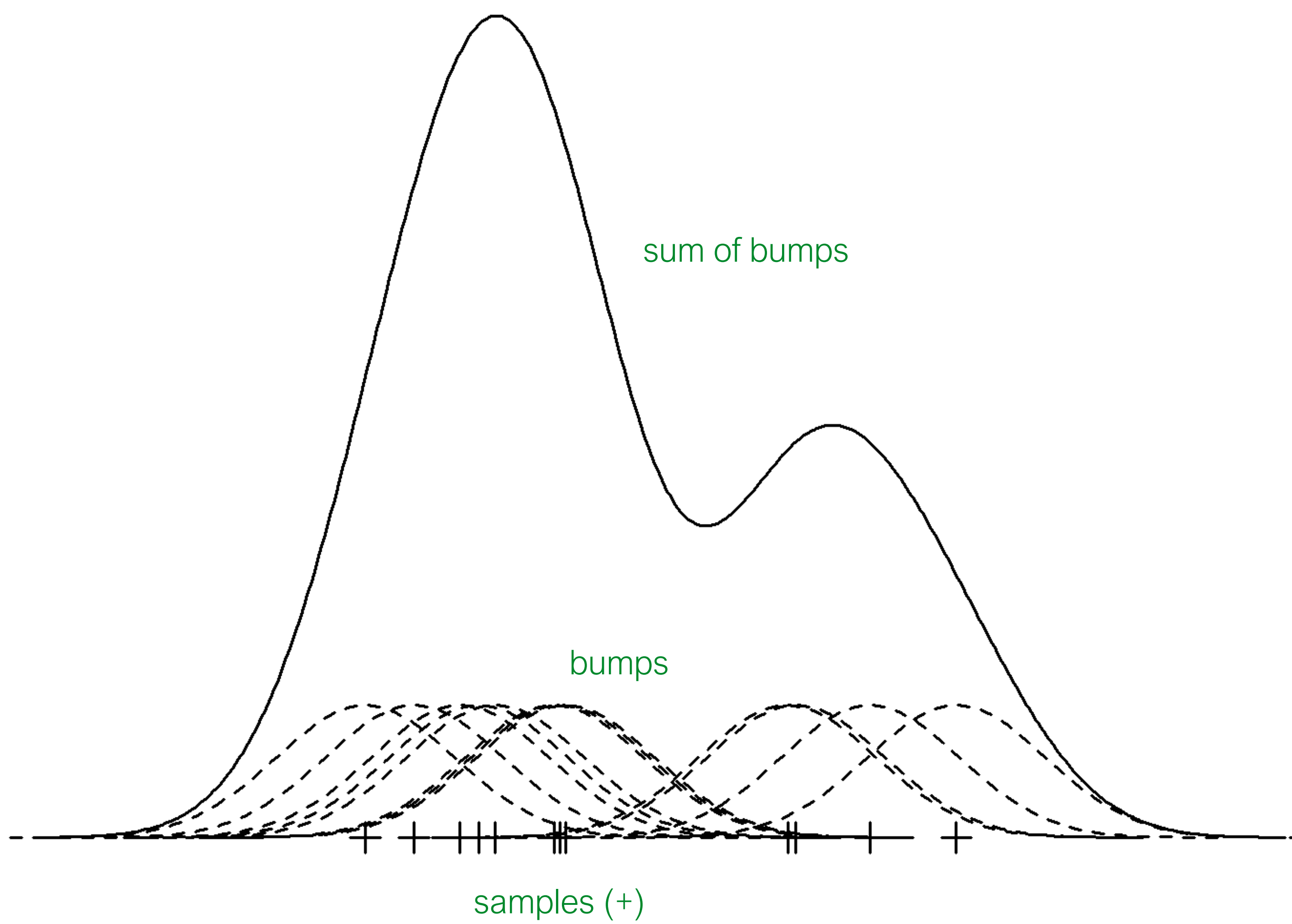


To understand the theory behind this we need to understand…

Kernel density estimation

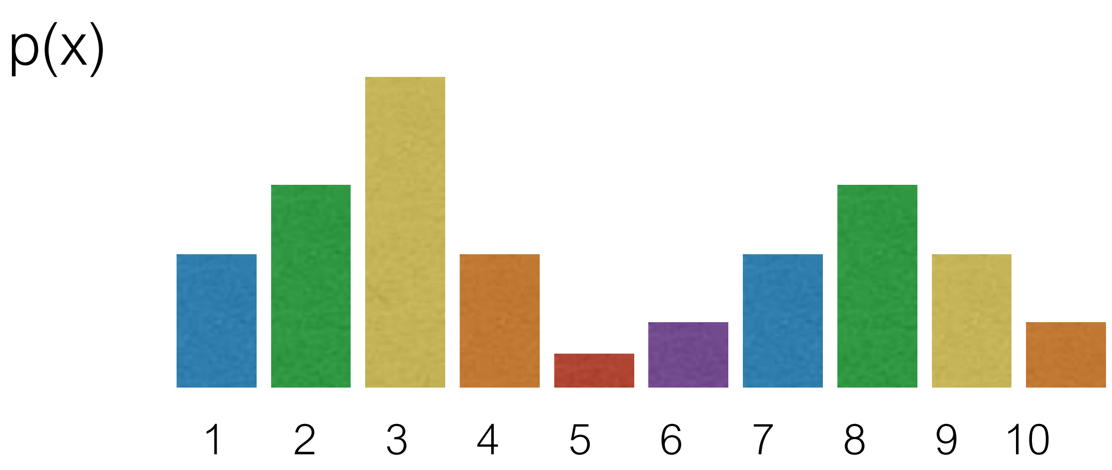
Kernel Density Estimation

A method to approximate an underlying PDF from samples

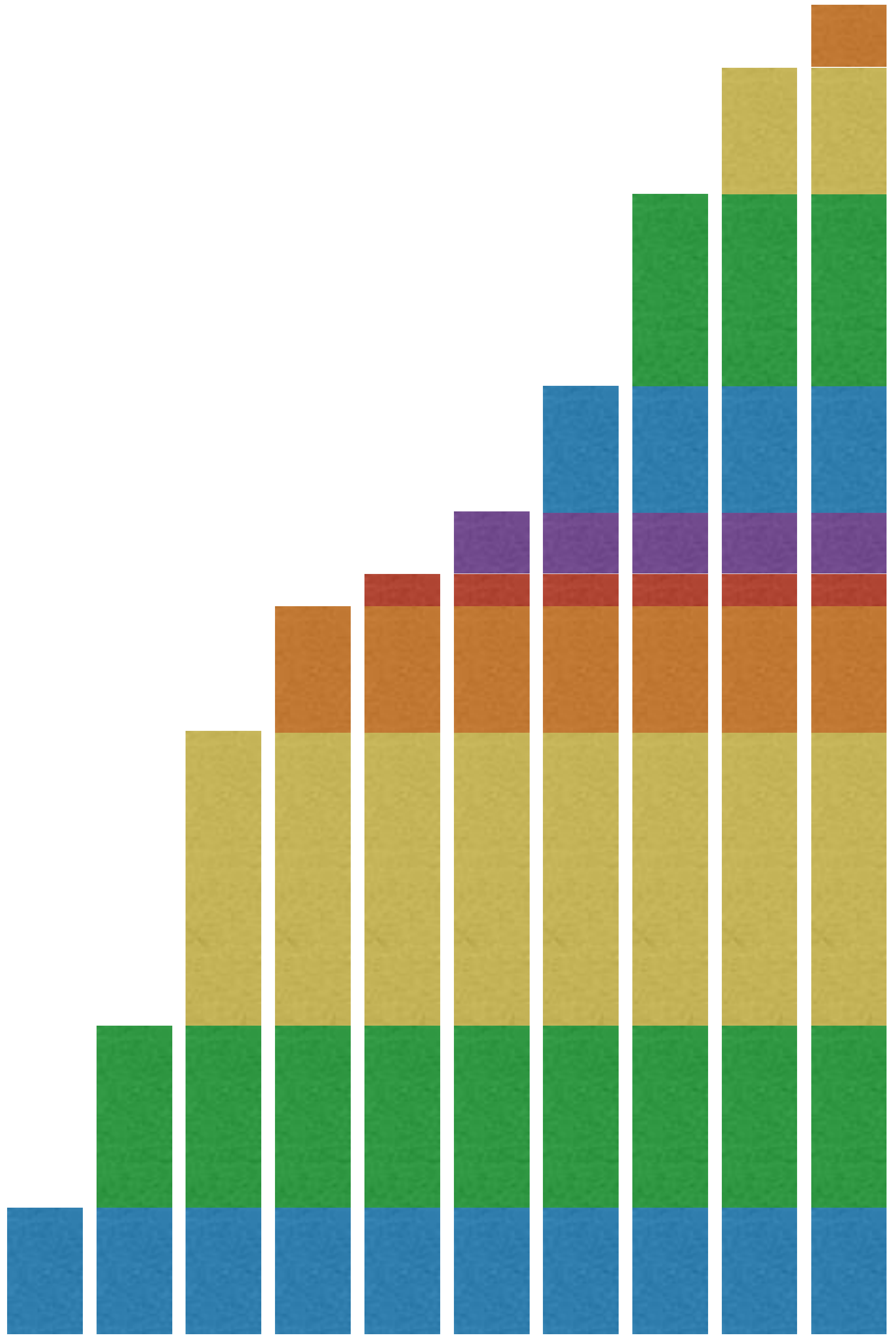


Put ‘bump’ on every sample to approximate the PDF

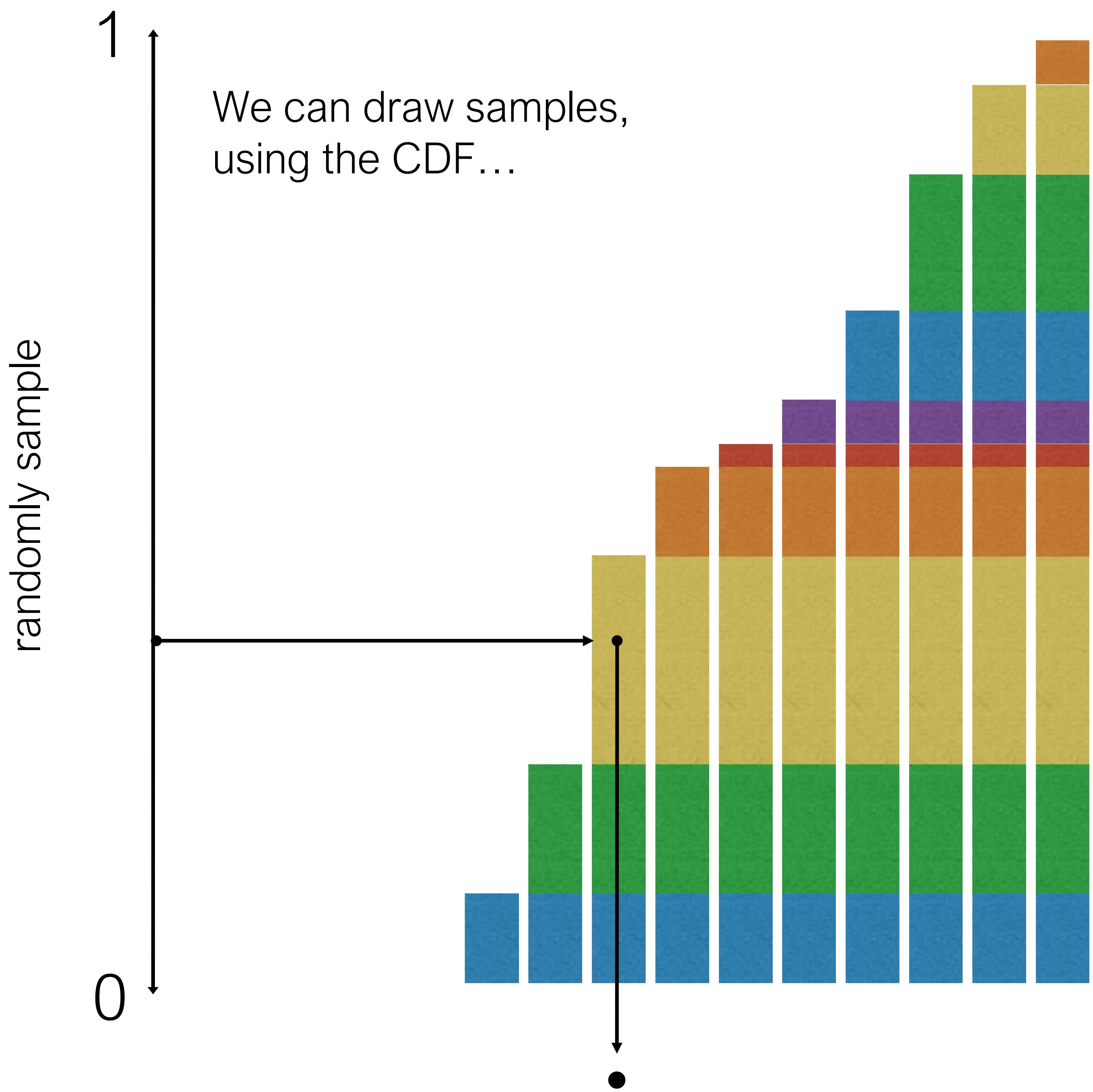
Say we have some hidden PDF…



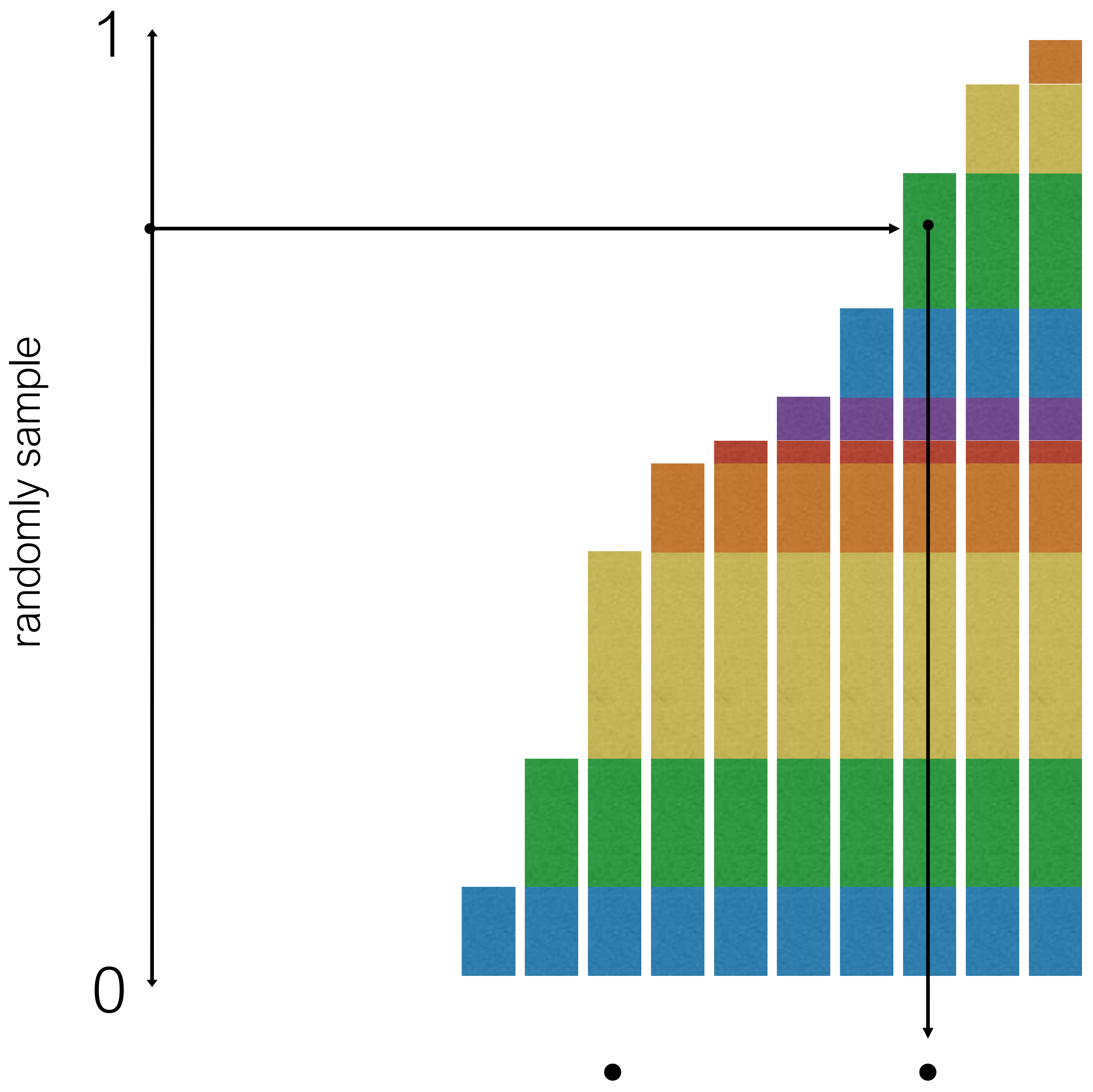
probability density function

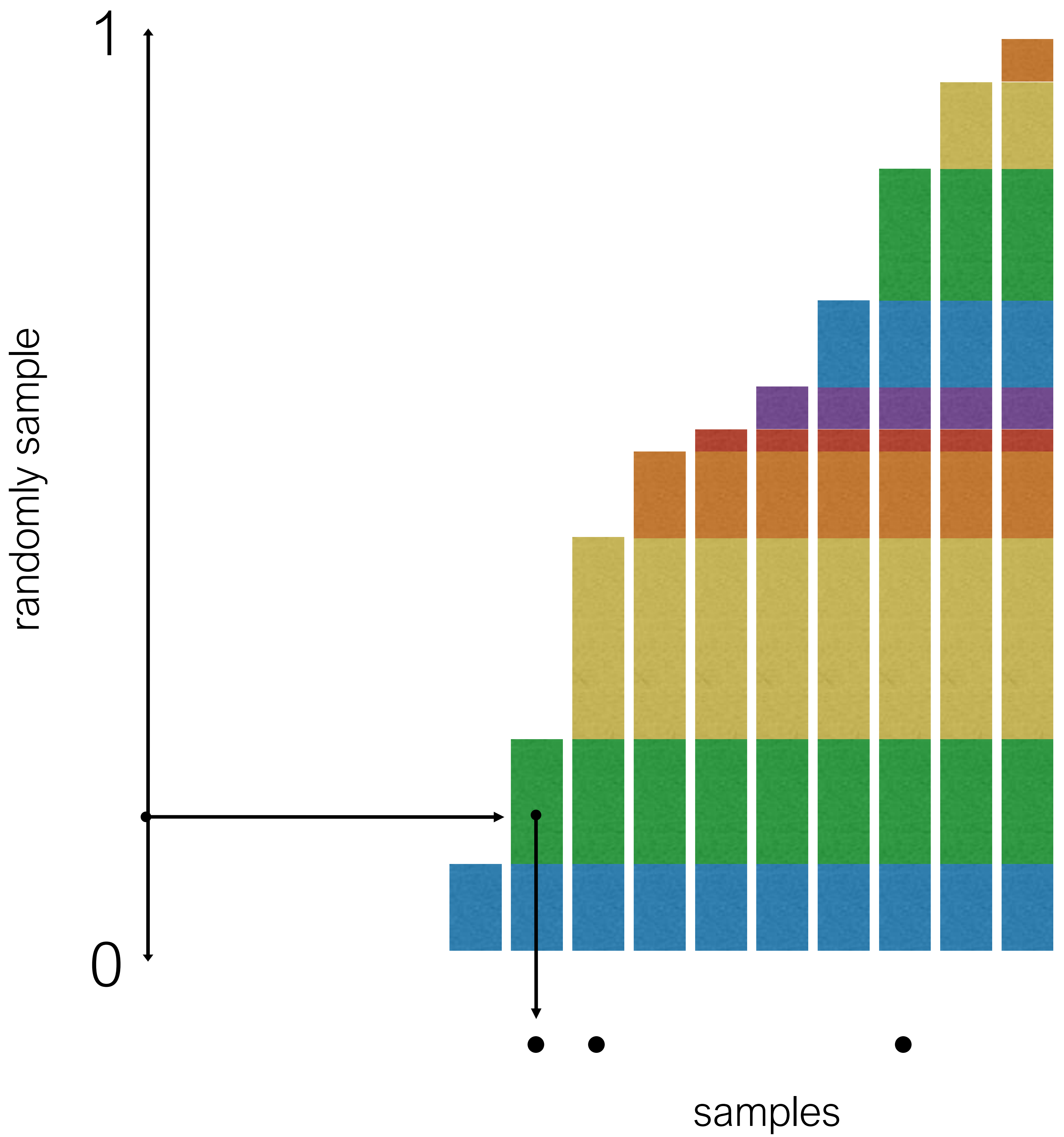


cumulative density function

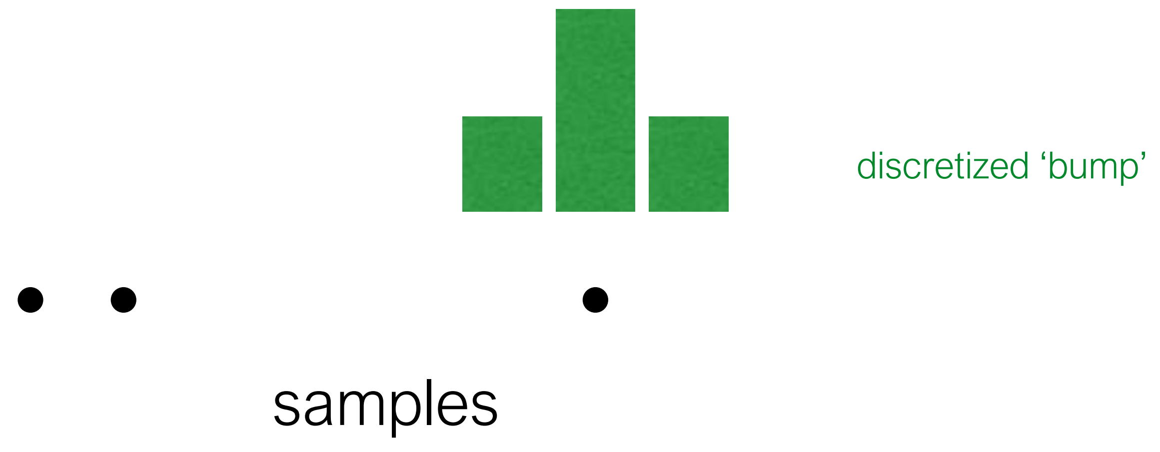


1
0
randomly sample
We can draw samples,
using the CDF…

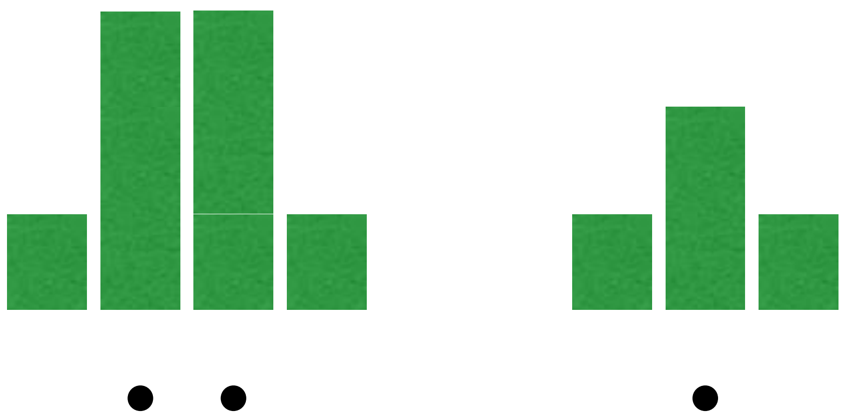

1
randomly sample
0


1
randomly sample
0
samples

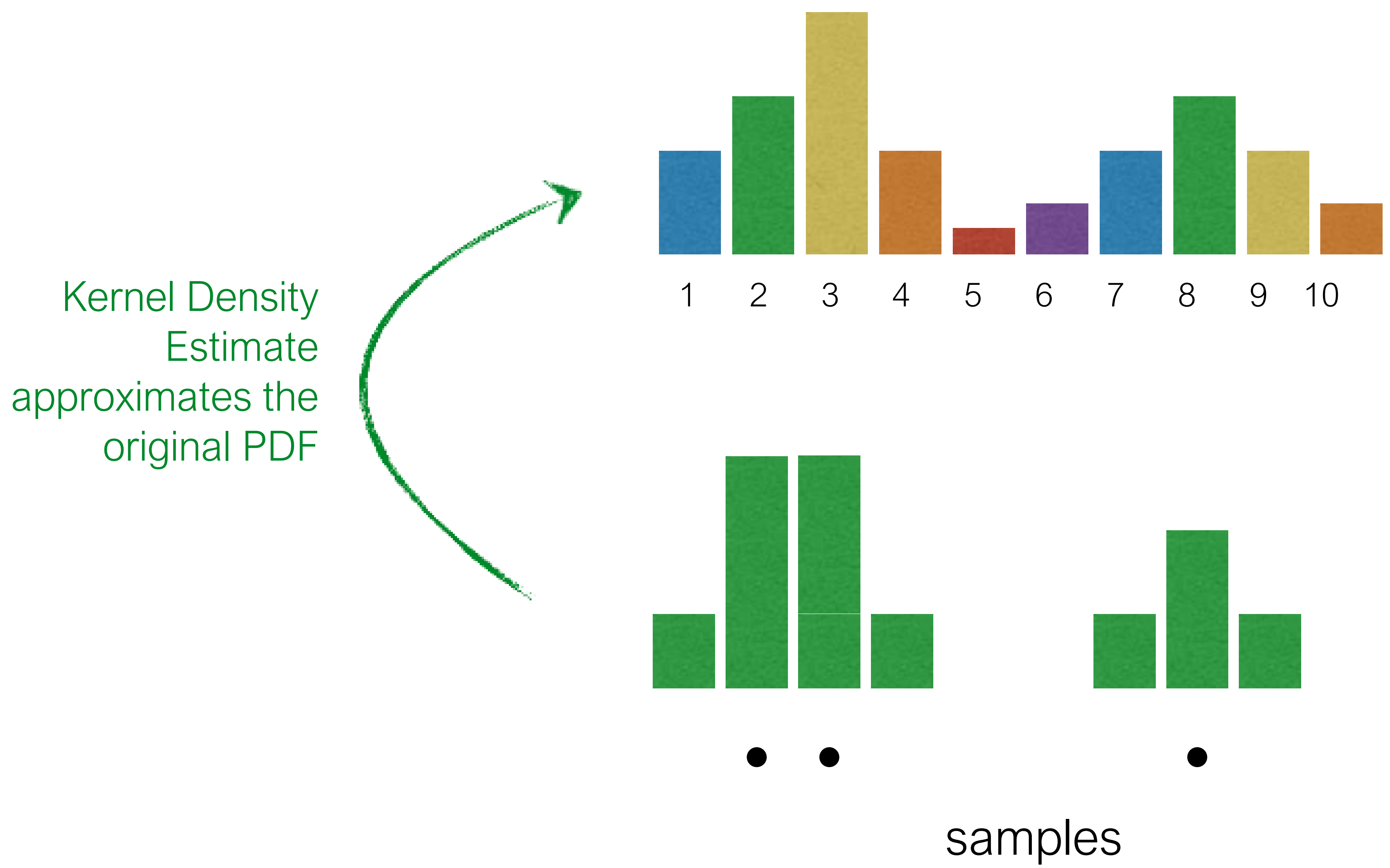
Now to estimate the ‘hidden’ PDF
place Gaussian bumps on the samples…


discretized ‘bump’
samples

samples

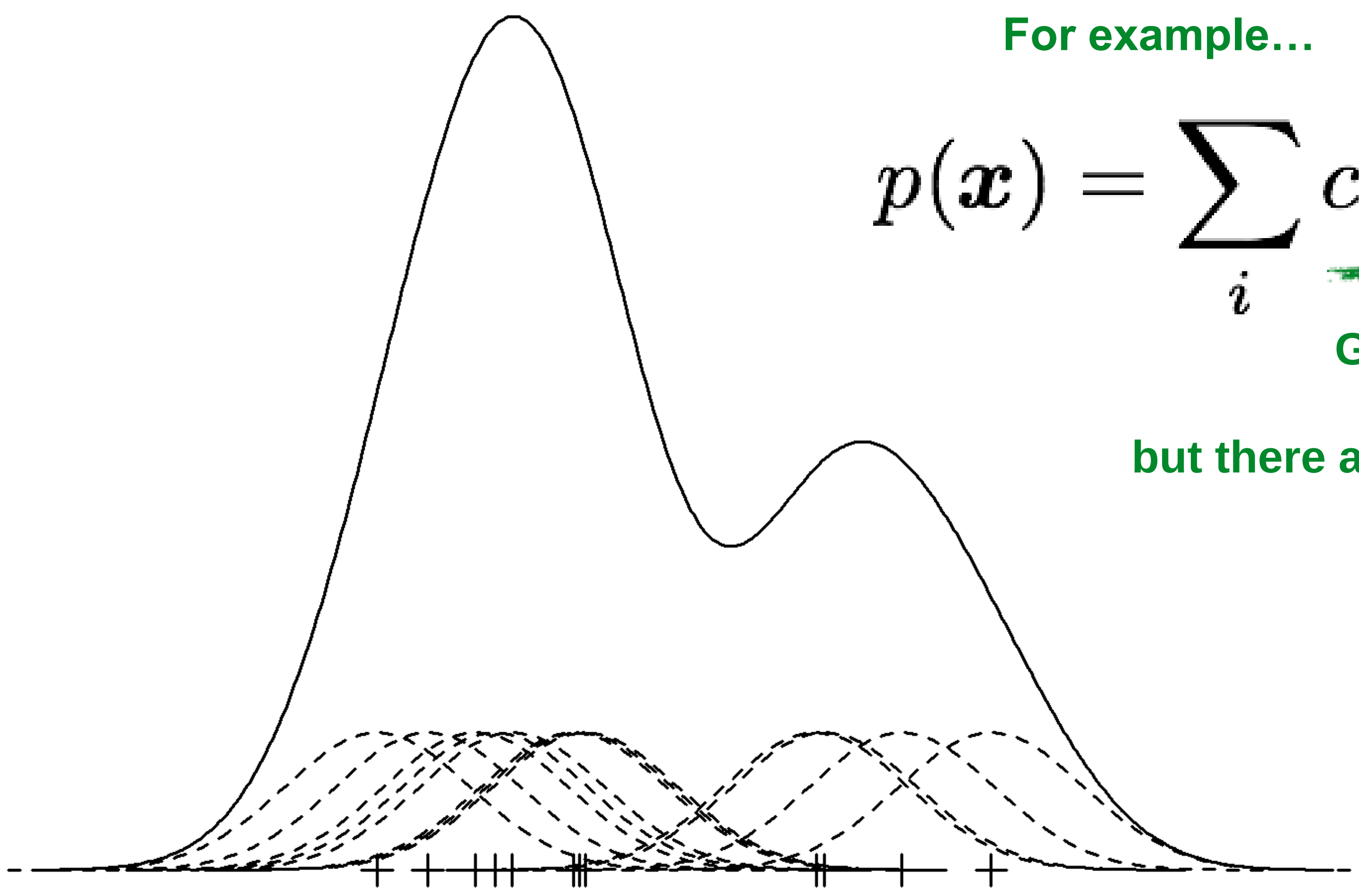


samples


Kernel Density
Estimate
approximates the
original PDF
1
2
3
4
5
6
7
8
9
10
samples

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!

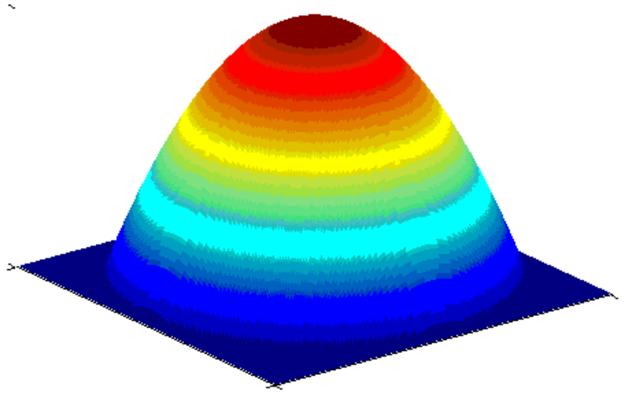
Put 'bump' on every sample to approximate the PDF

Kernel Function

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

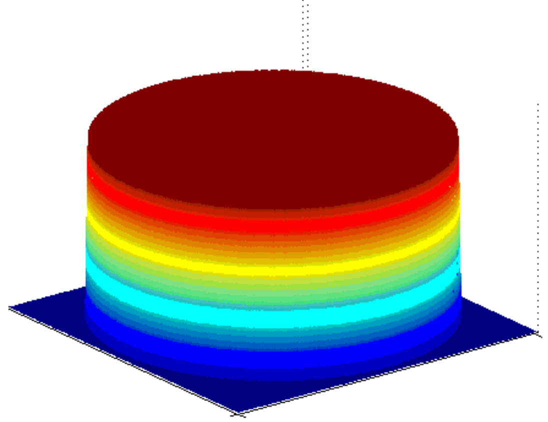
returns the 'distance' between two points

Epanechnikov kernel



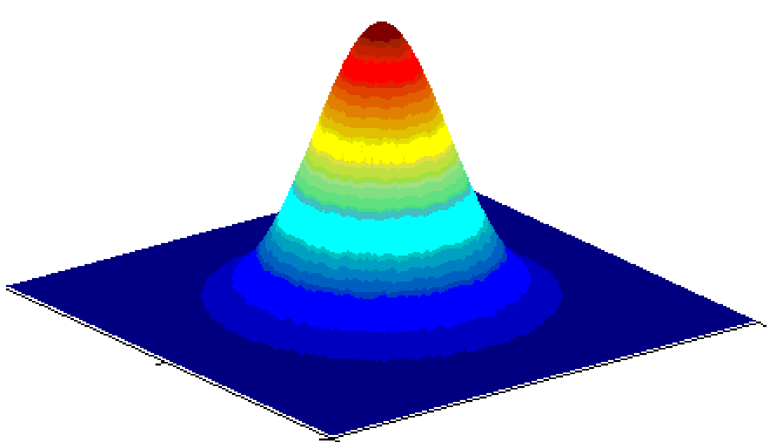
$$K(\boldsymbol{x}, \boldsymbol{x}') = \begin{cases} c(1 - \|\boldsymbol{x} - \boldsymbol{x}'\|^2) & \|\boldsymbol{x} - \boldsymbol{x}'\|^2 \leq 1 \\ 0 & \text{otherwise} \end{cases}$$

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

shift values becomes really small

1. Compute mean-shift

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

compute the 'mean'

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

compute the 'shift'

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

update the point

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:

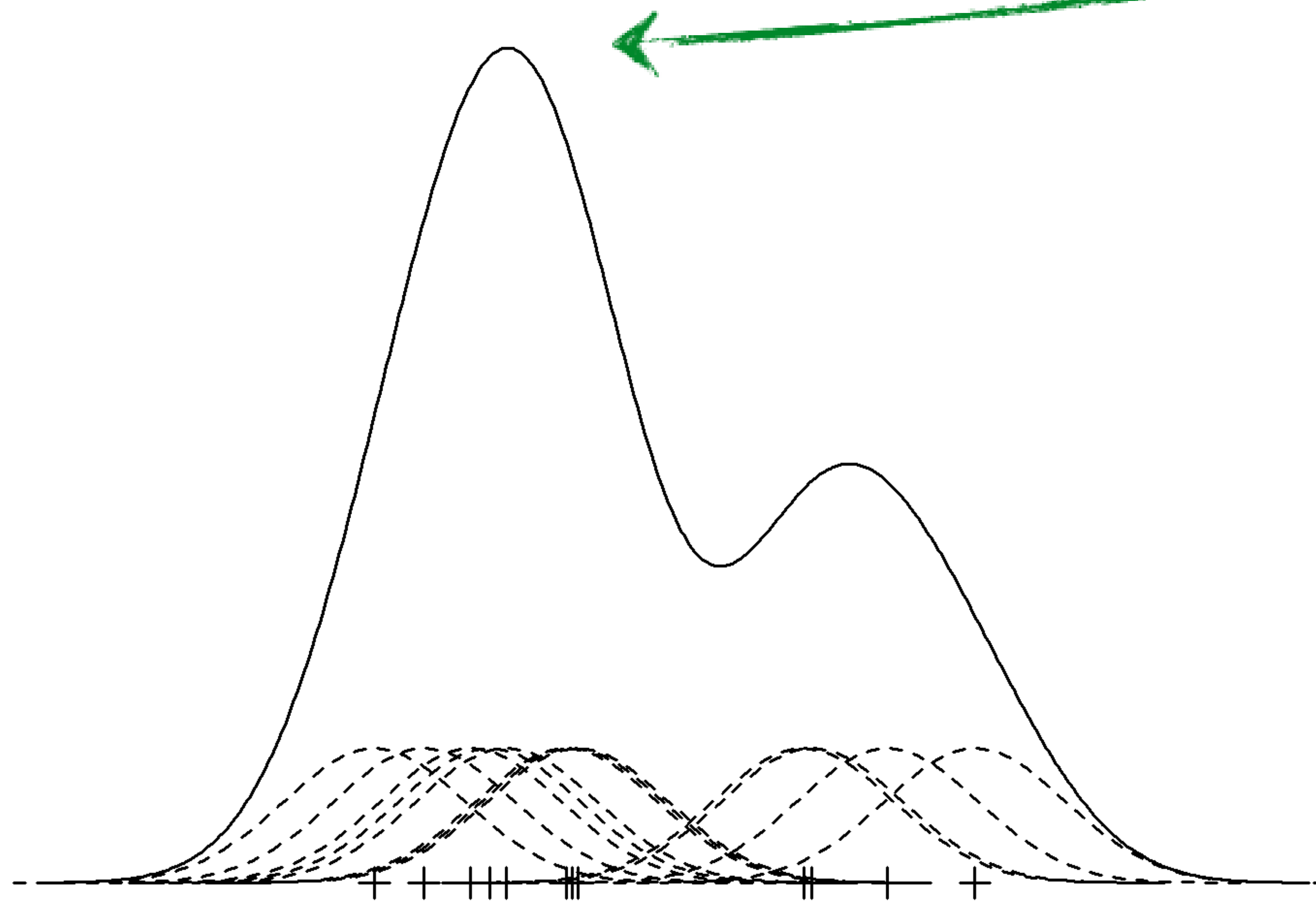
Gradient of the PDF is related to the mean shift vector

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

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

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

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

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

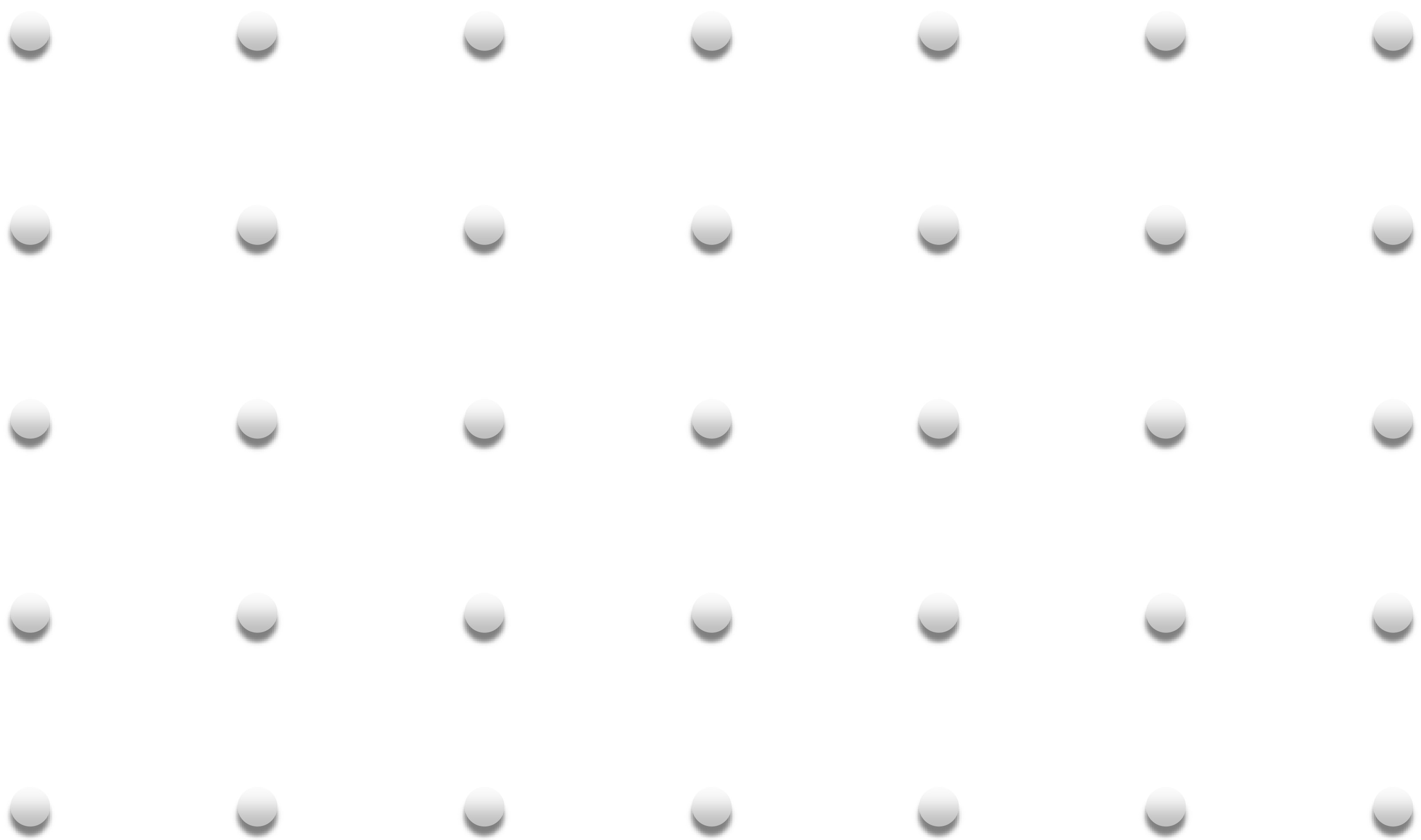
Just 5 lines of code!

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

same

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

Mean-Shift Algorithm

(for images)

Initialize $\boldsymbol{x}$

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

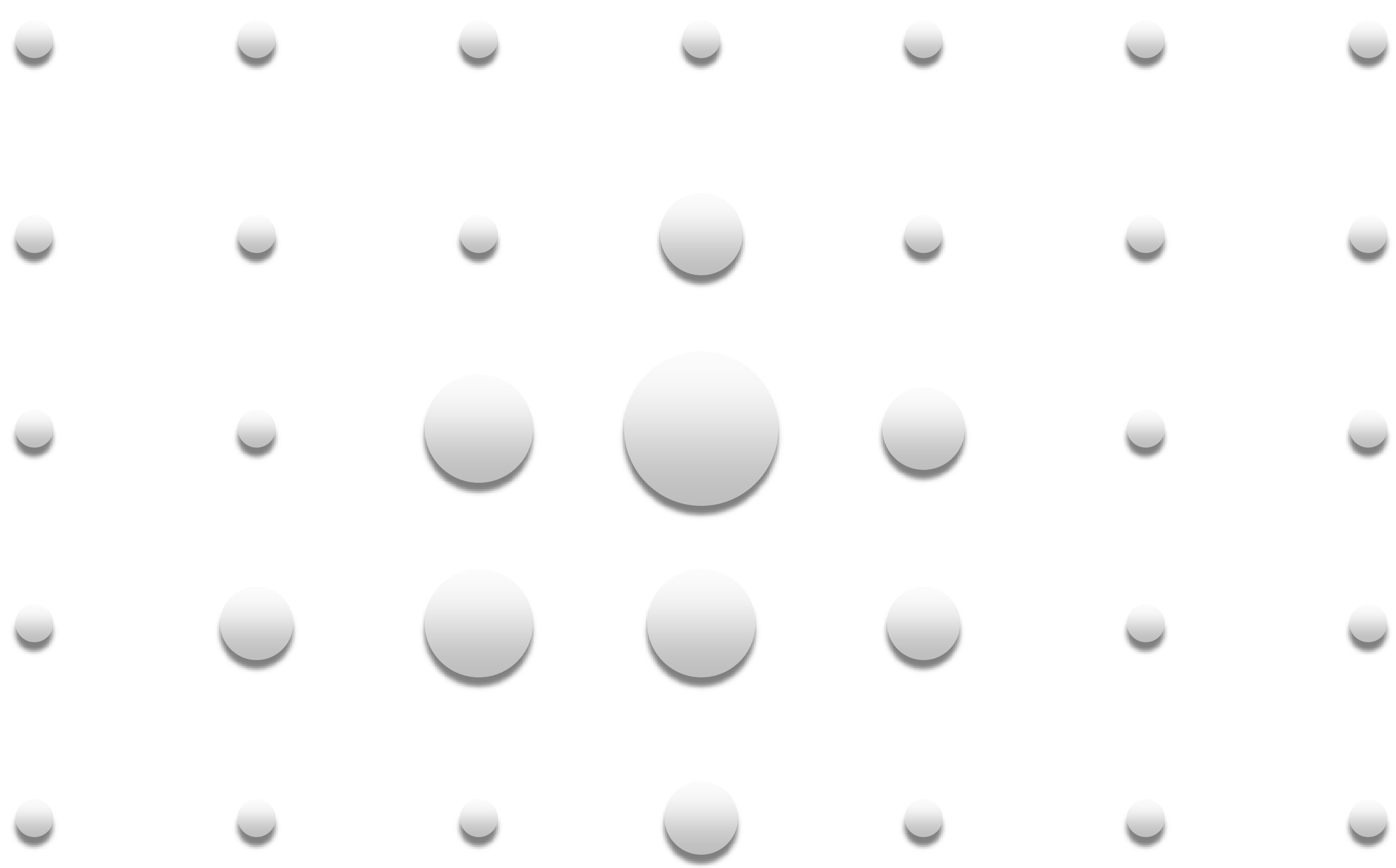
1. Compute mean-shift

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

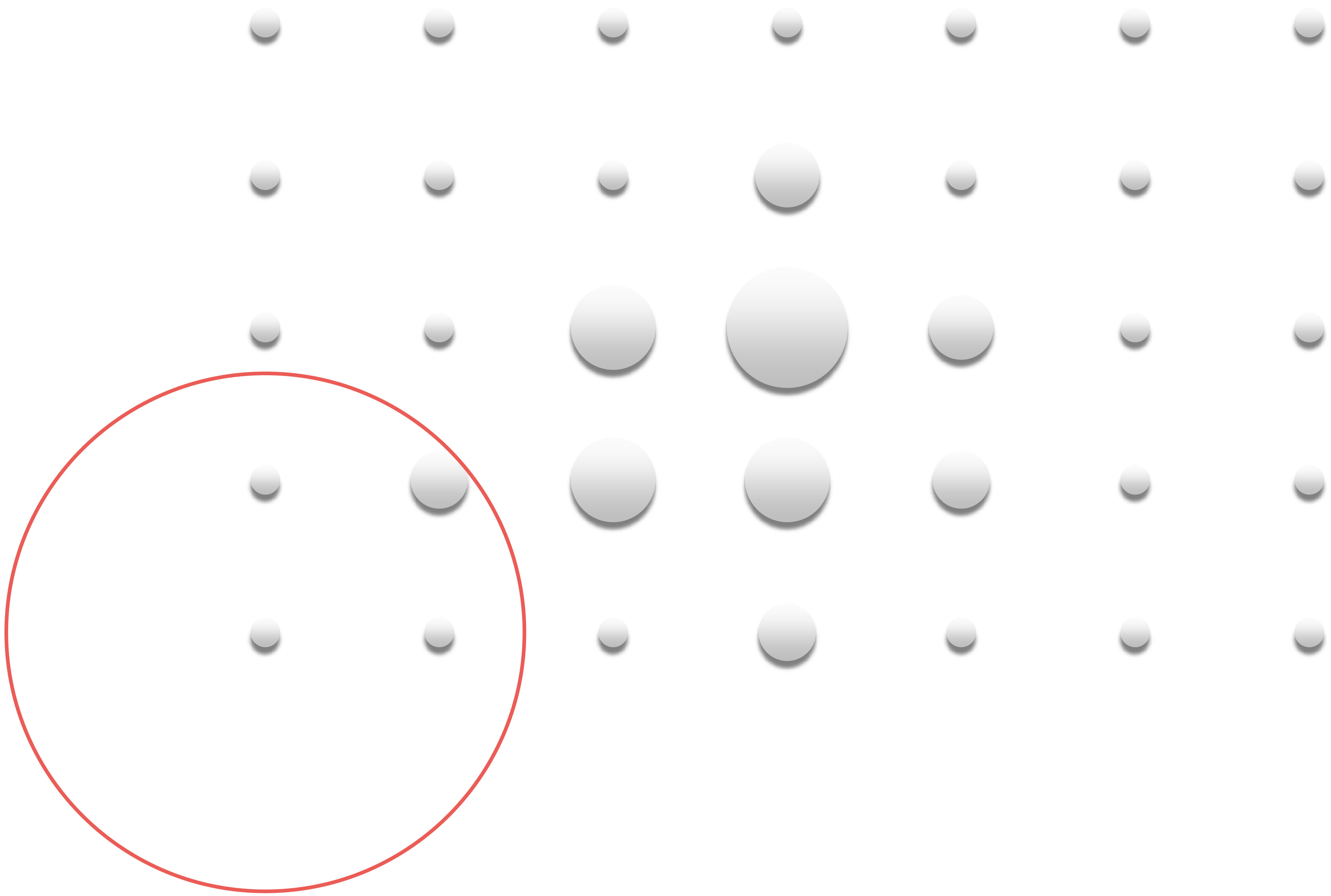
$$v(\boldsymbol{x}) = m(\boldsymbol{x}) - \boldsymbol{x}$$

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

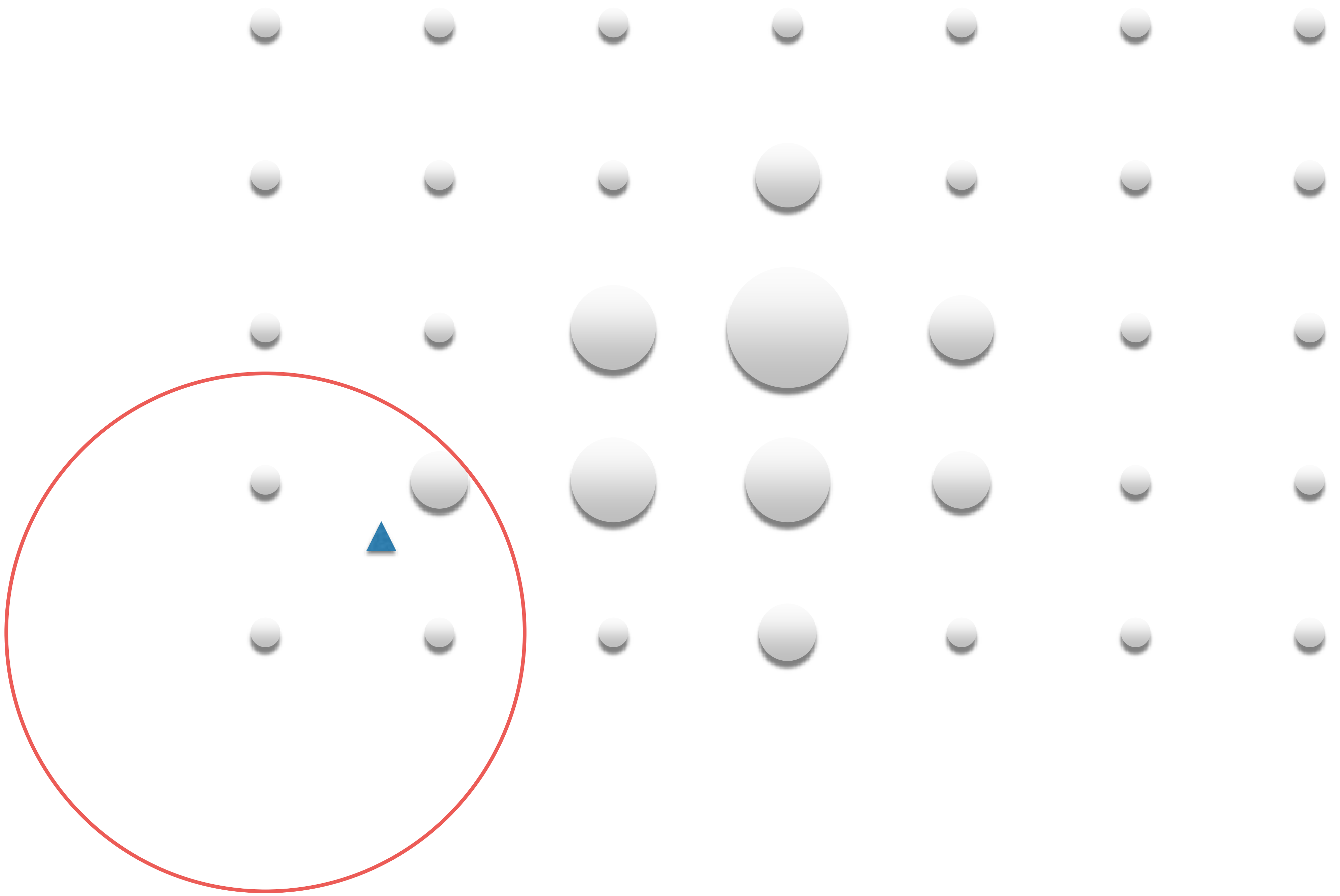
For images, each pixel is point with a weight



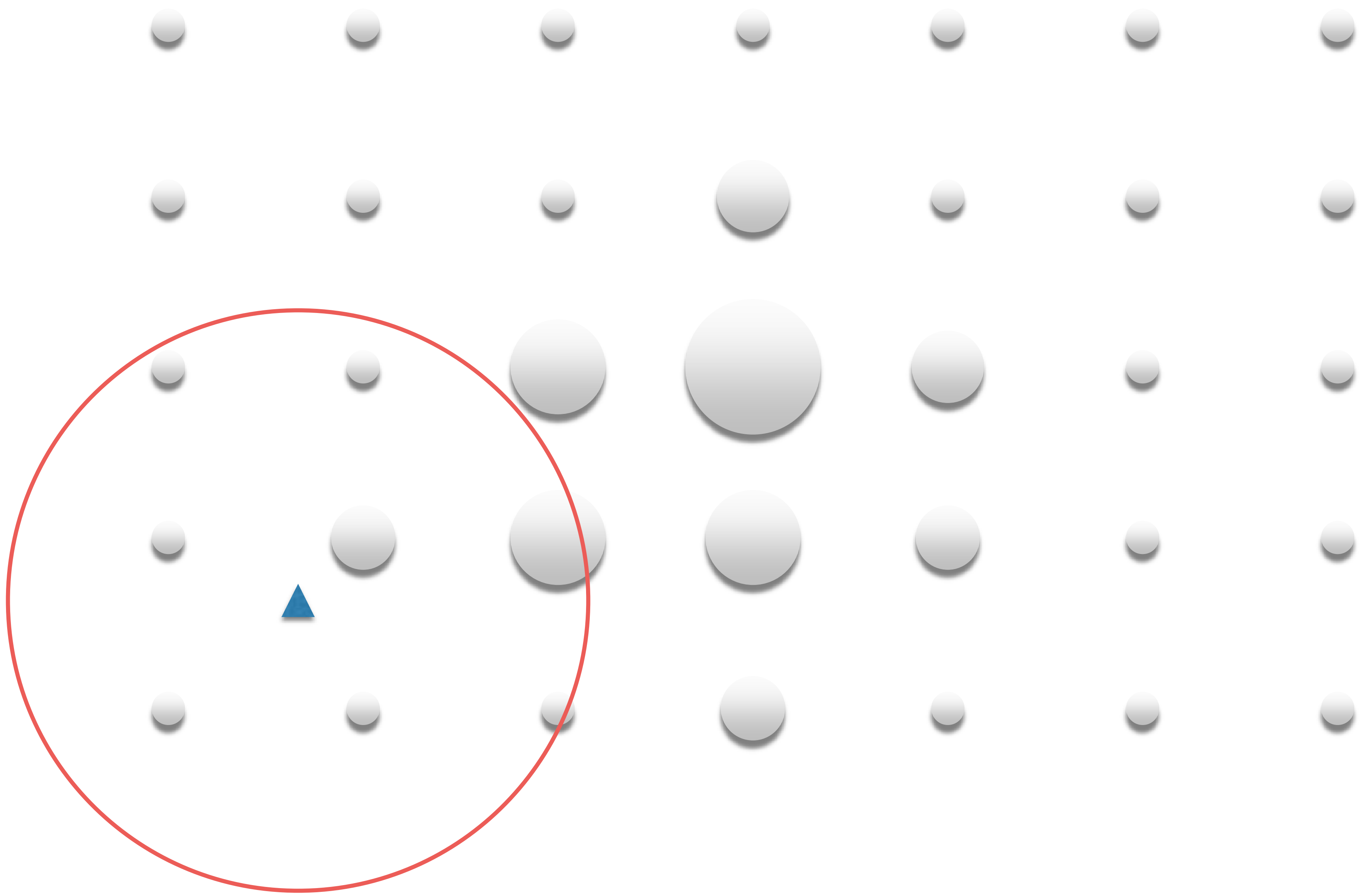
For images, each pixel is point with a weight



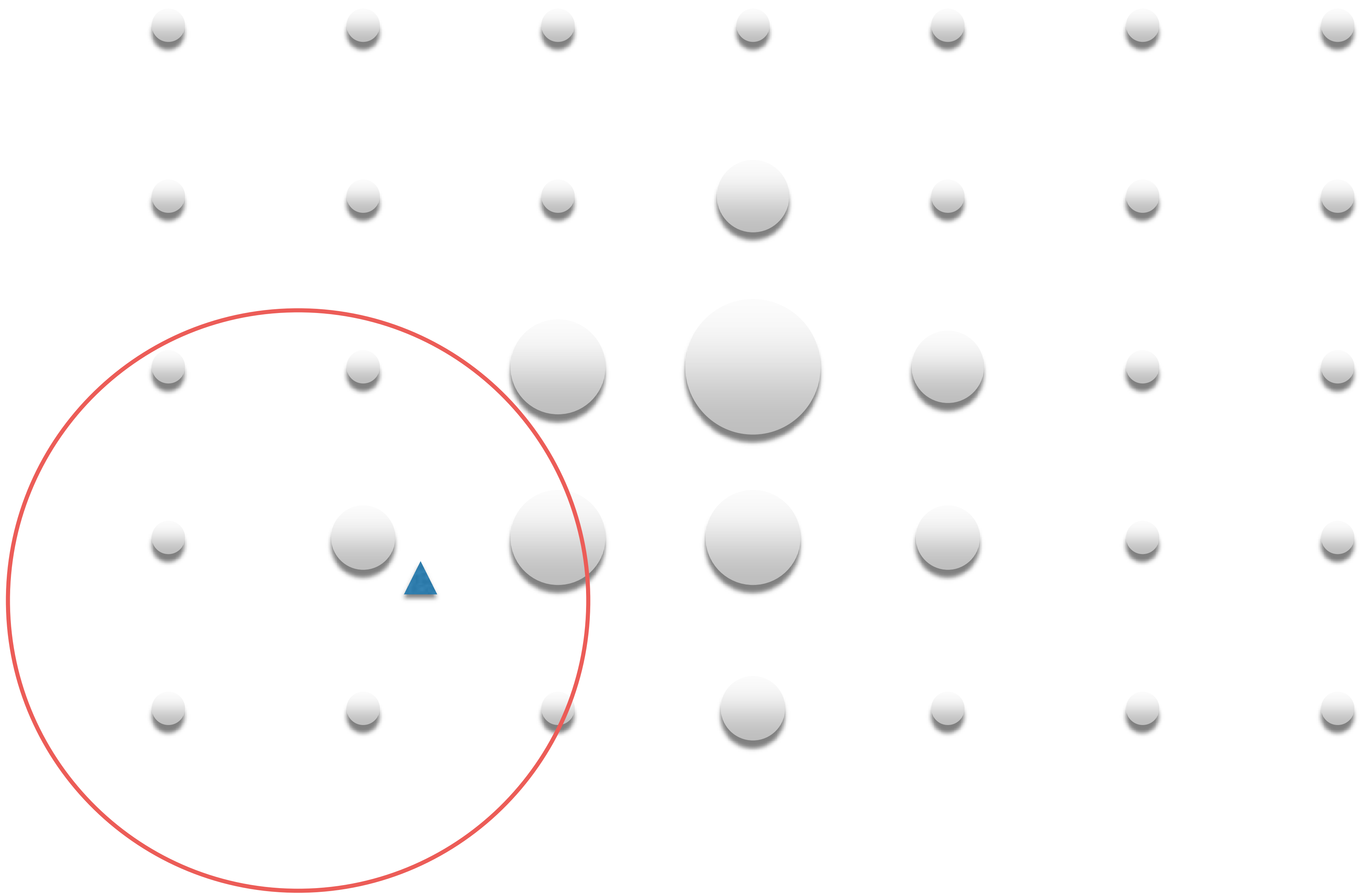
For images, each pixel is point with a weight



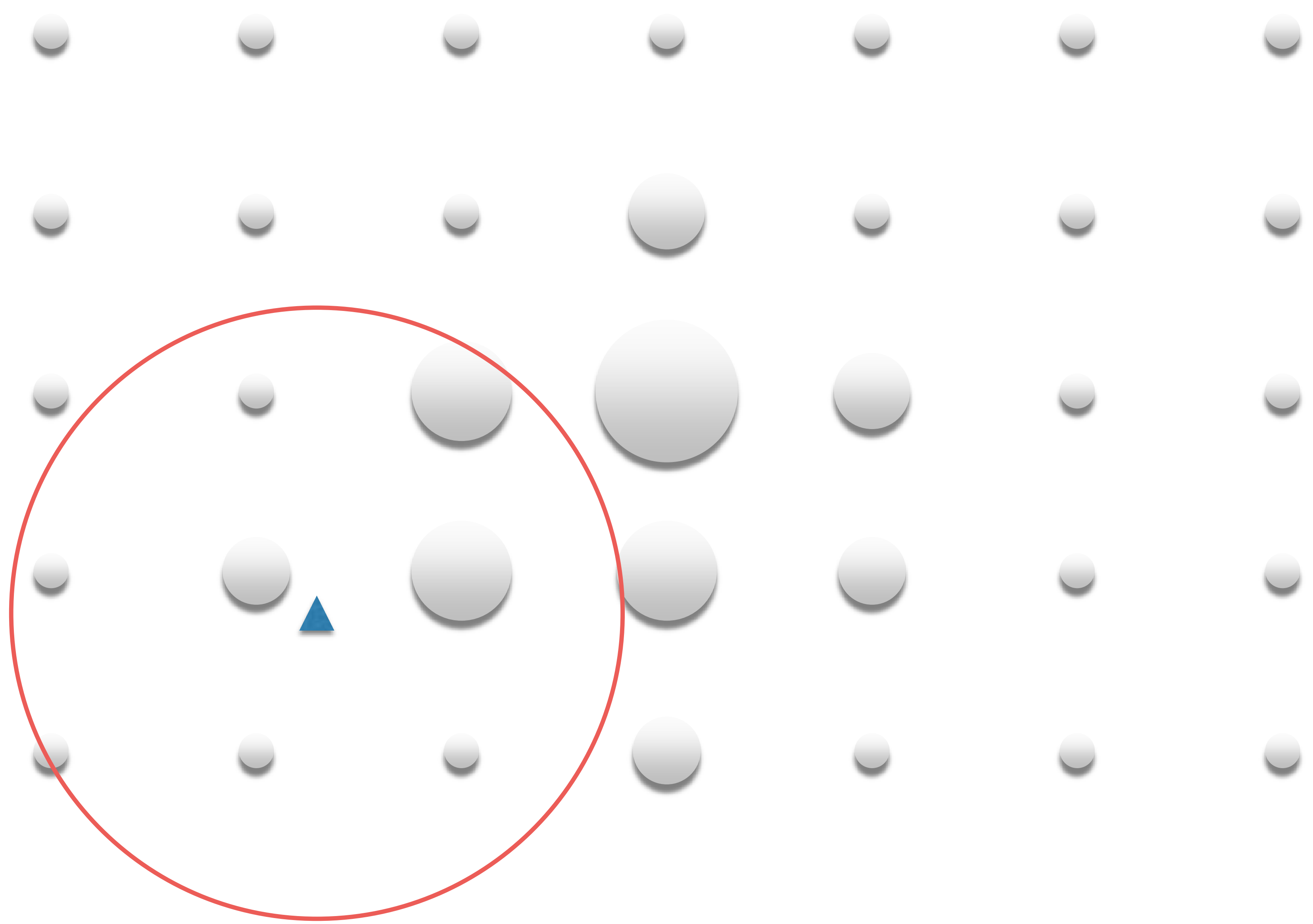
For images, each pixel is point with a weight



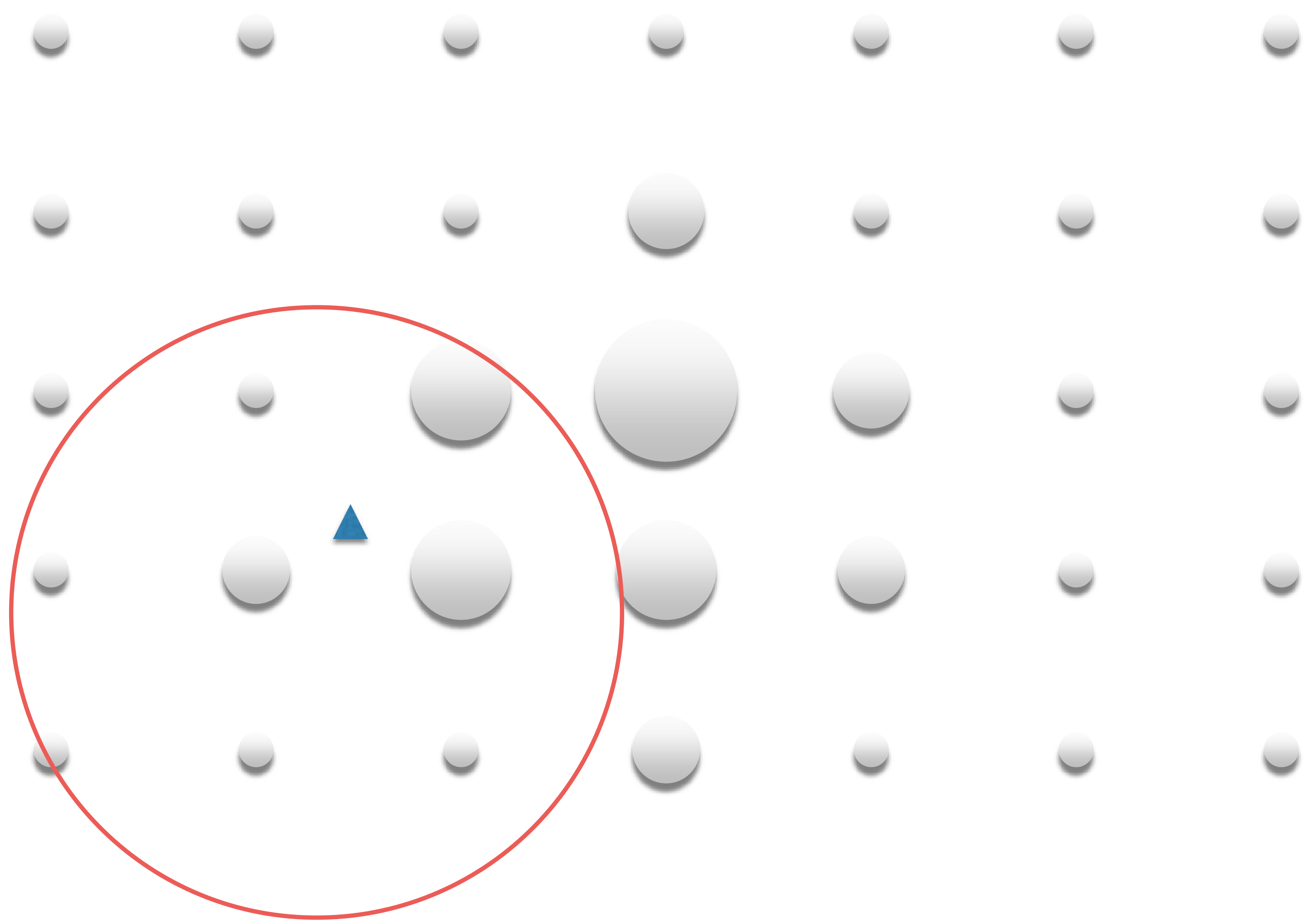
For images, each pixel is point with a weight



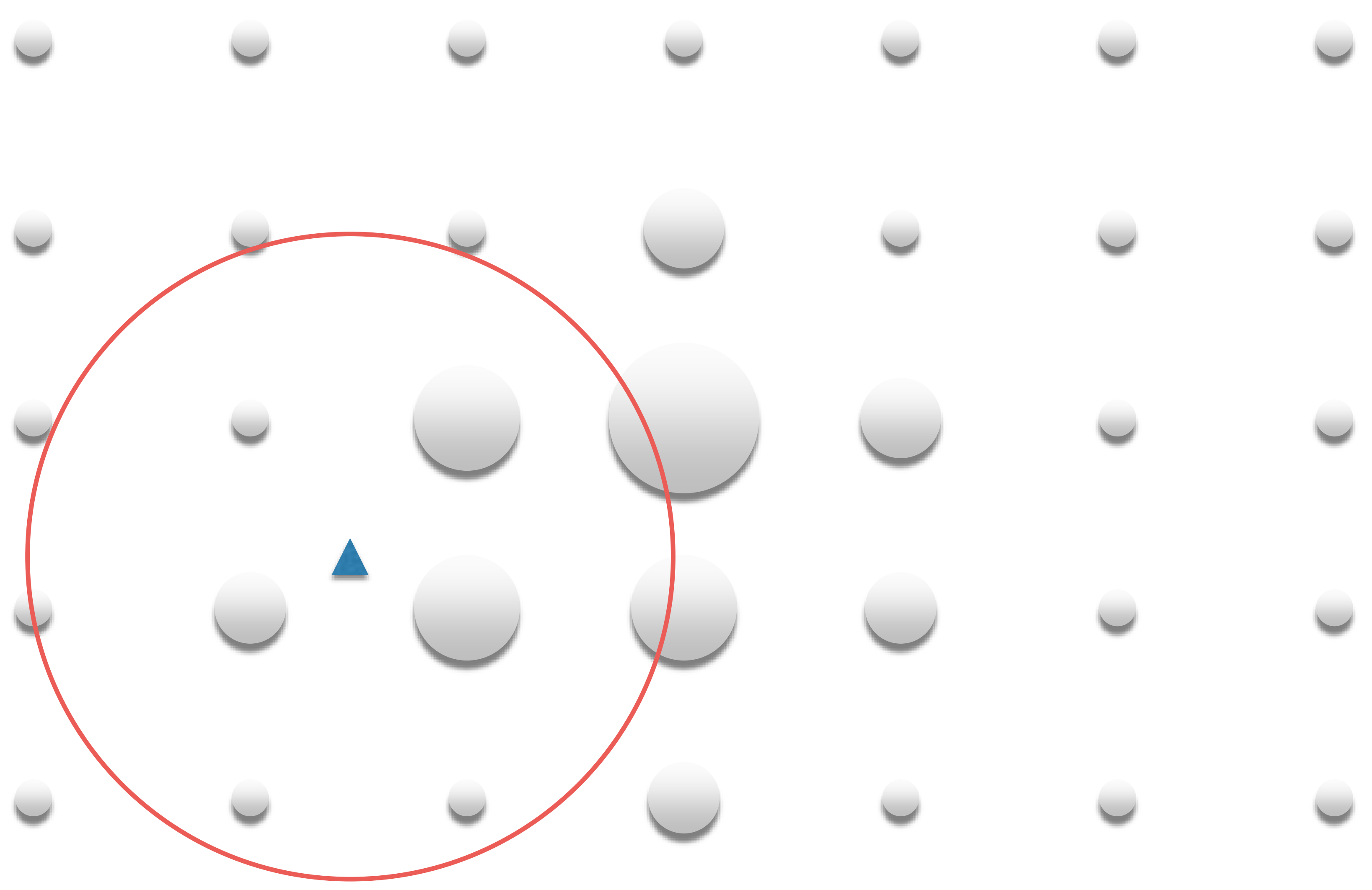
For images, each pixel is point with a weight



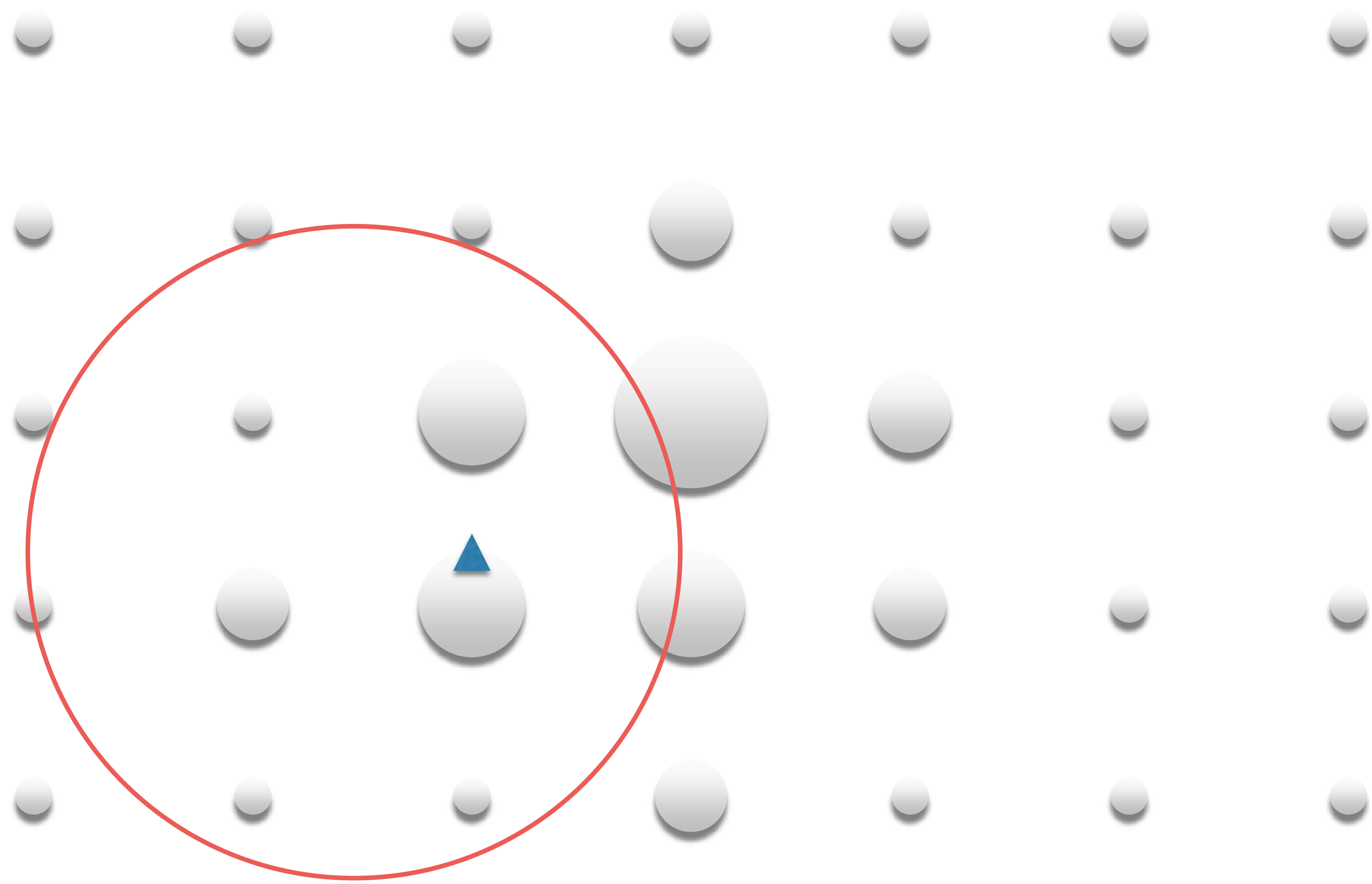
For images, each pixel is point with a weight



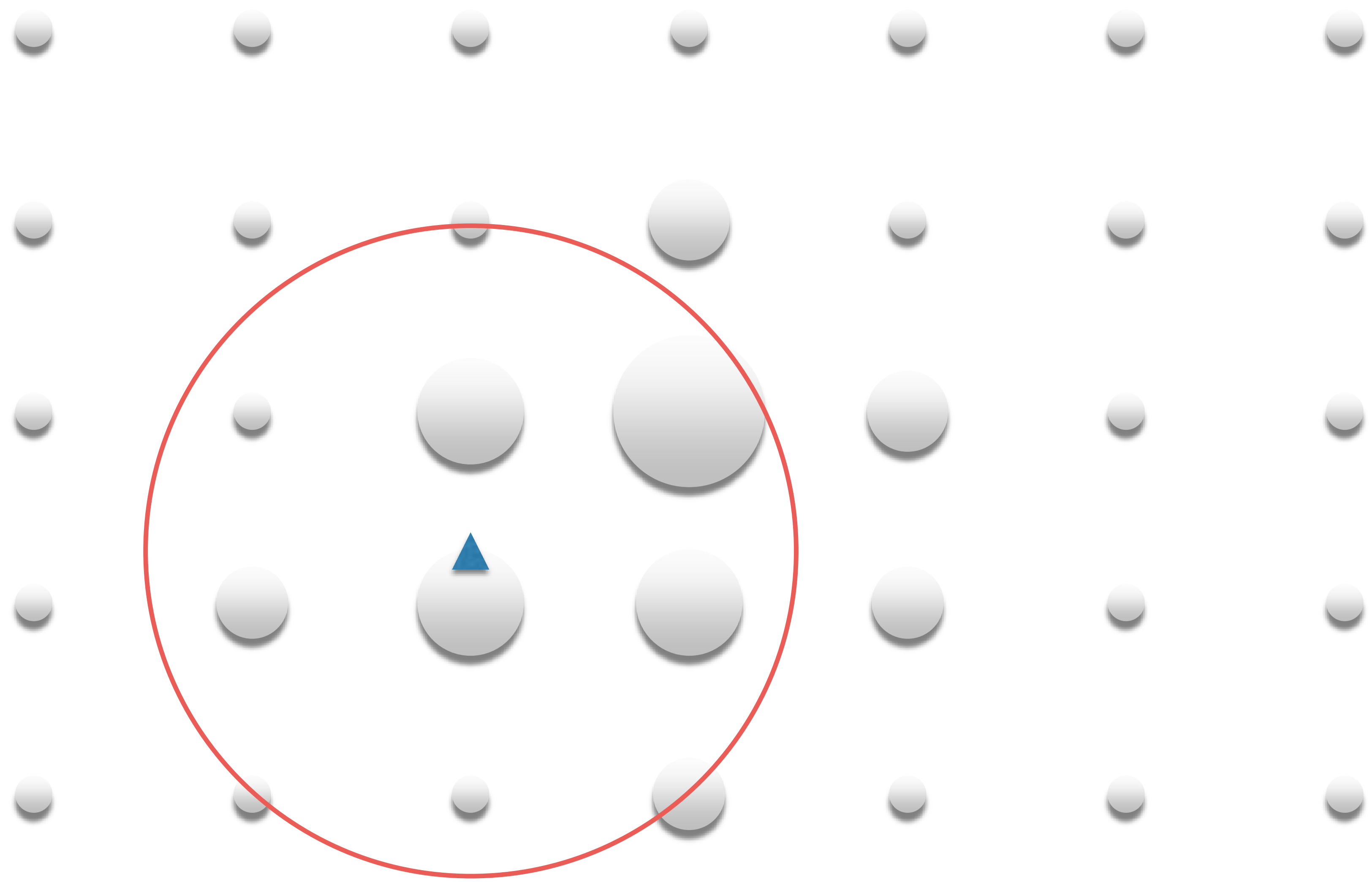
For images, each pixel is point with a weight



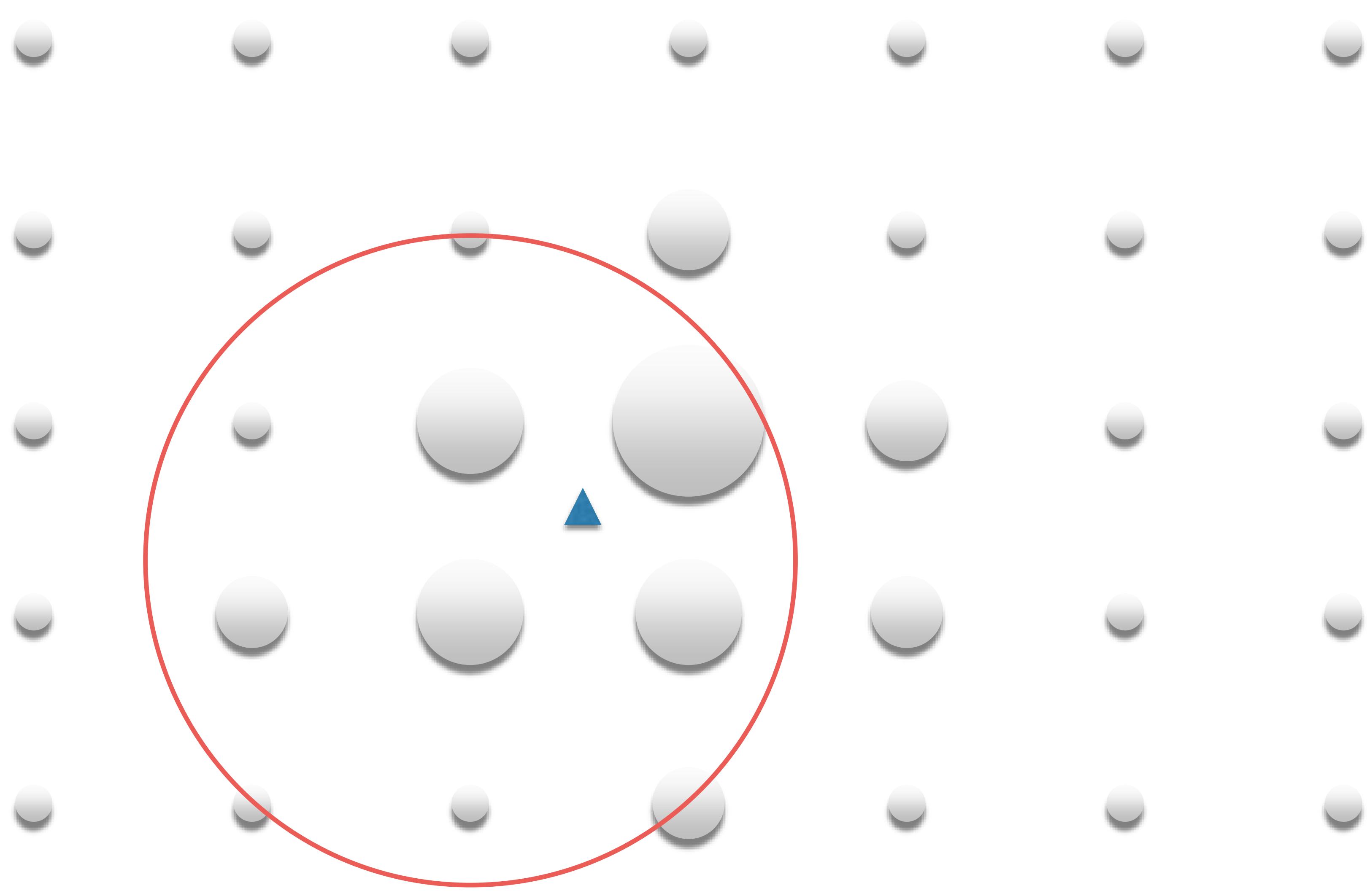
For images, each pixel is point with a weight



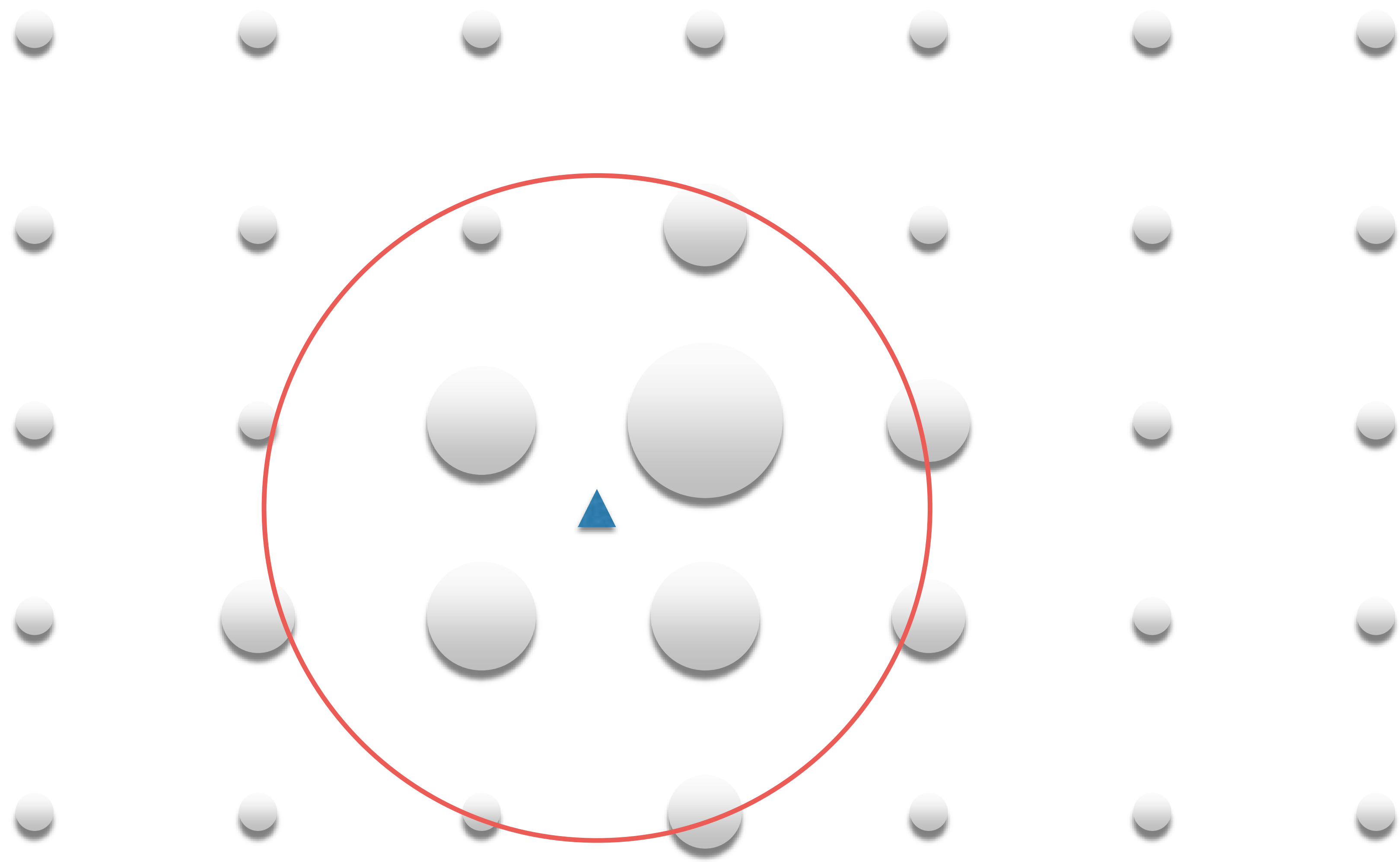
For images, each pixel is point with a weight



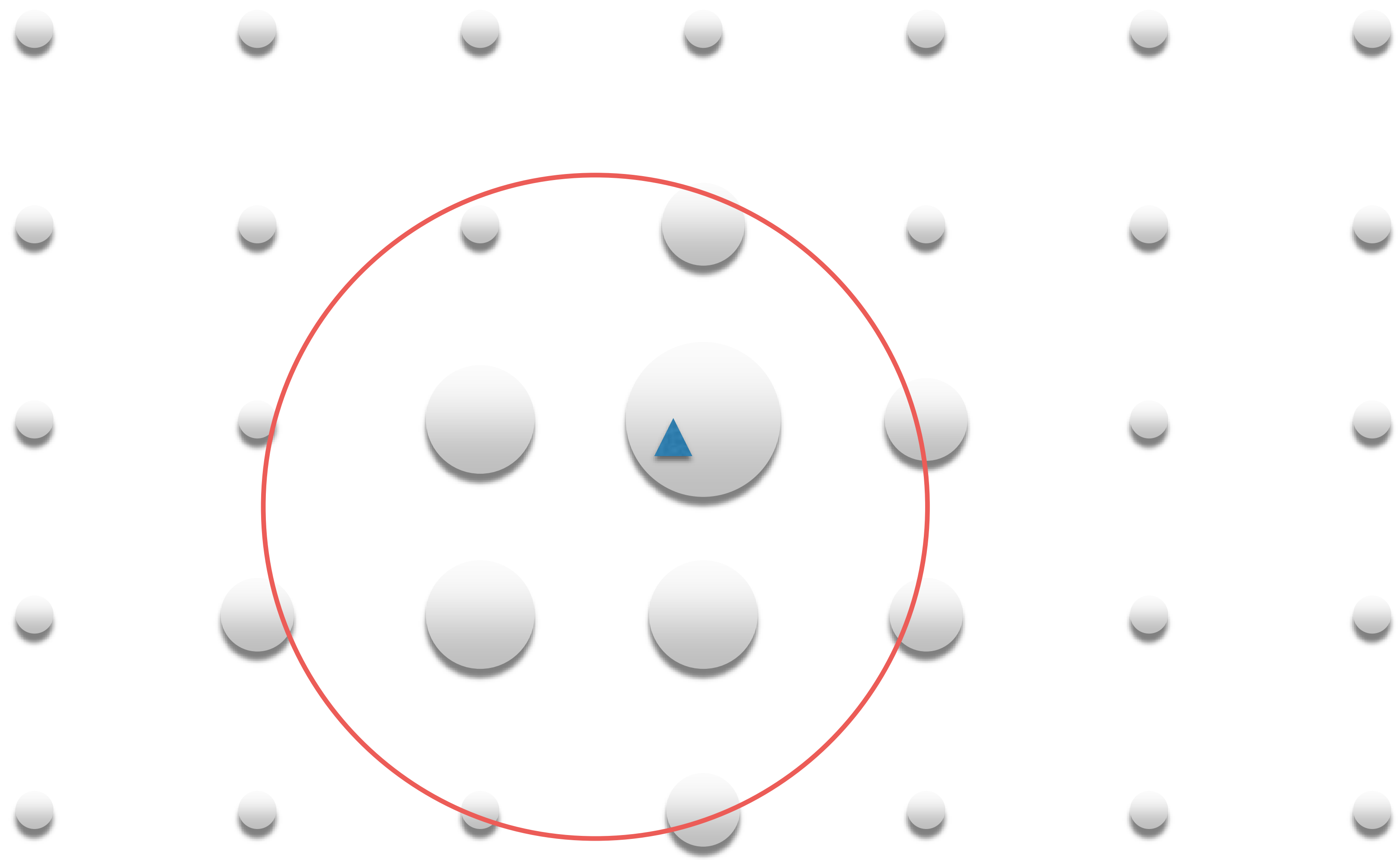
For images, each pixel is point with a weight



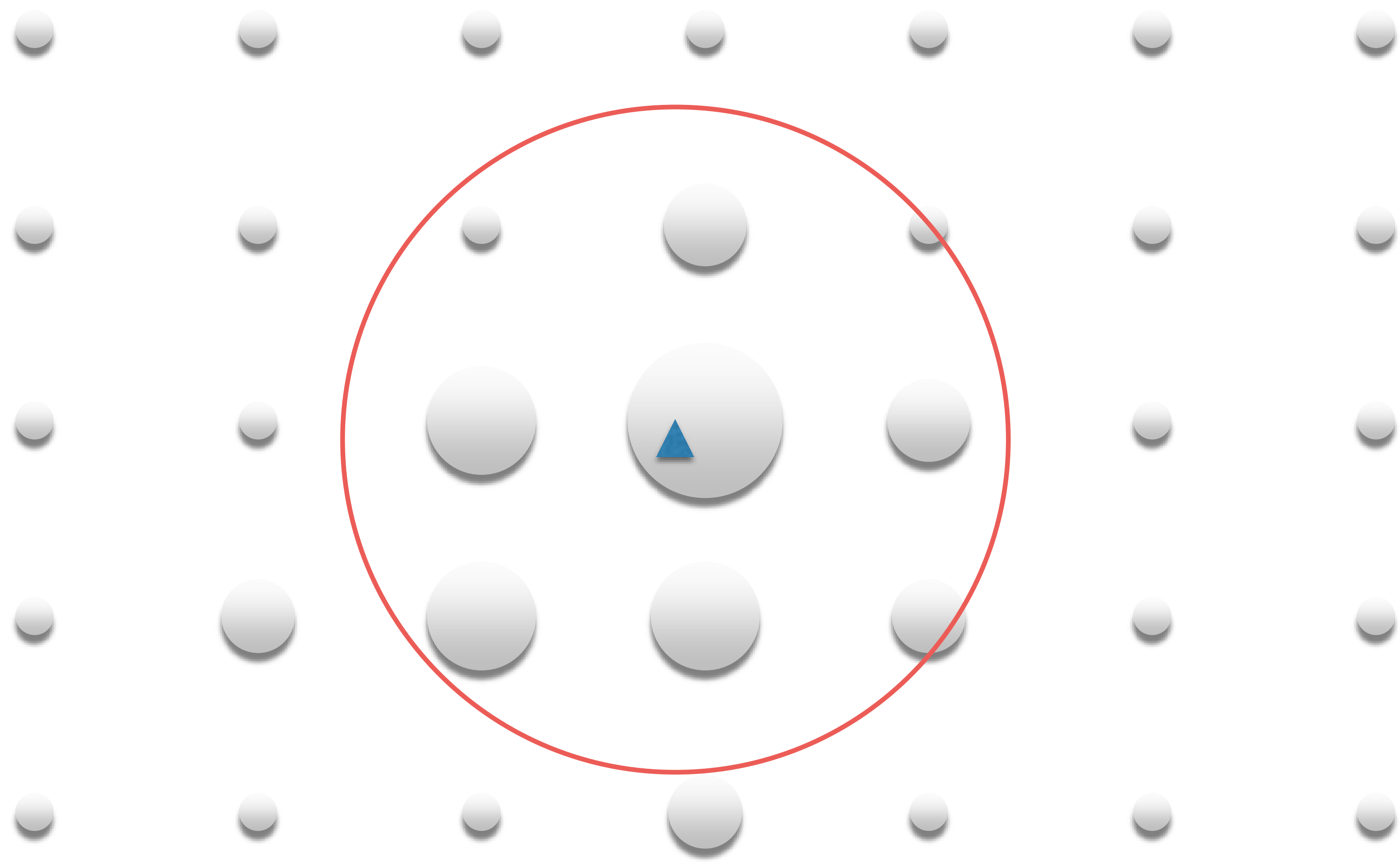
For images, each pixel is point with a weight



For images, each pixel is point with a weight

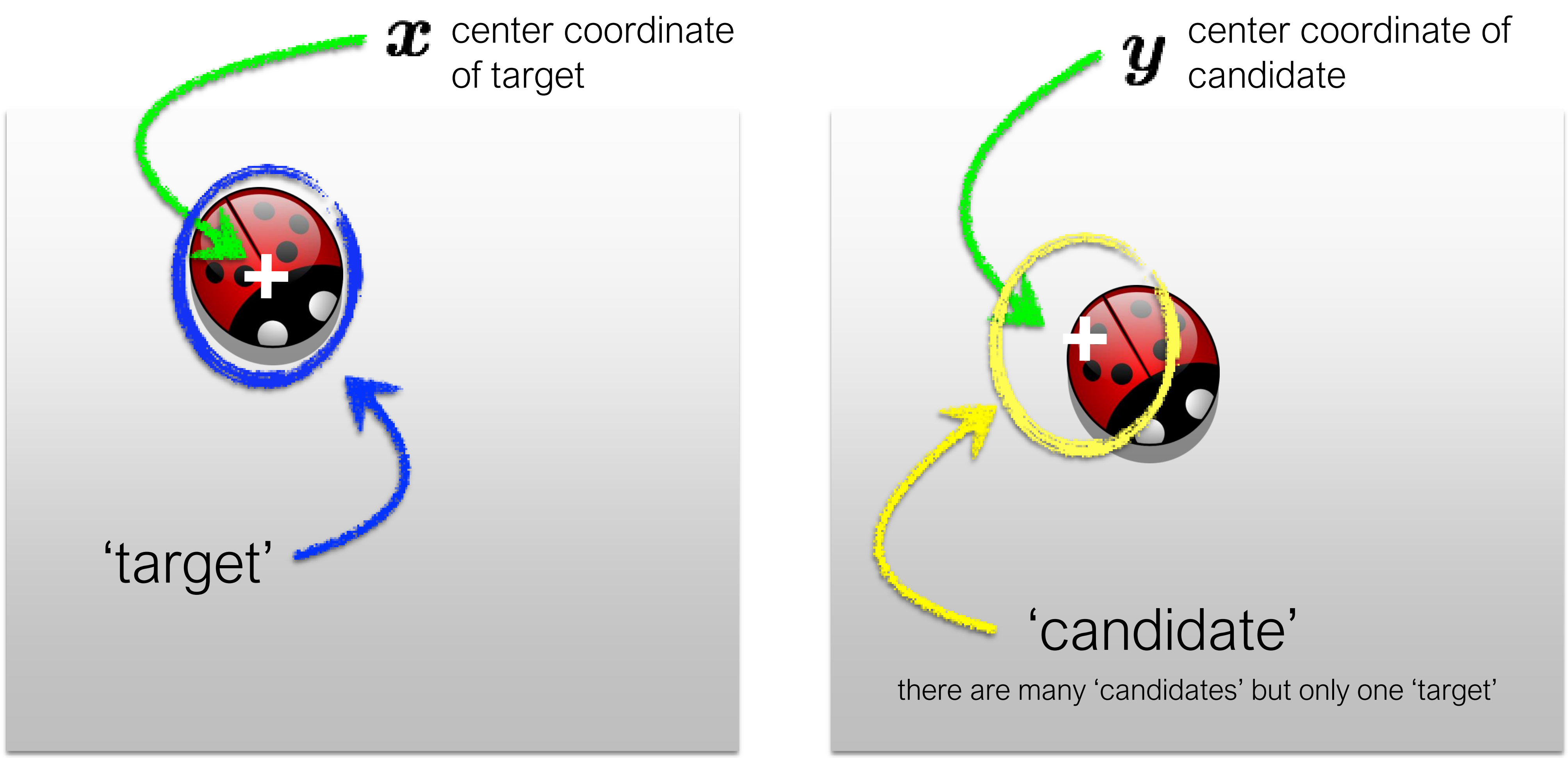


For images, each pixel is point with a weight



Finally… mean shift tracking in video!

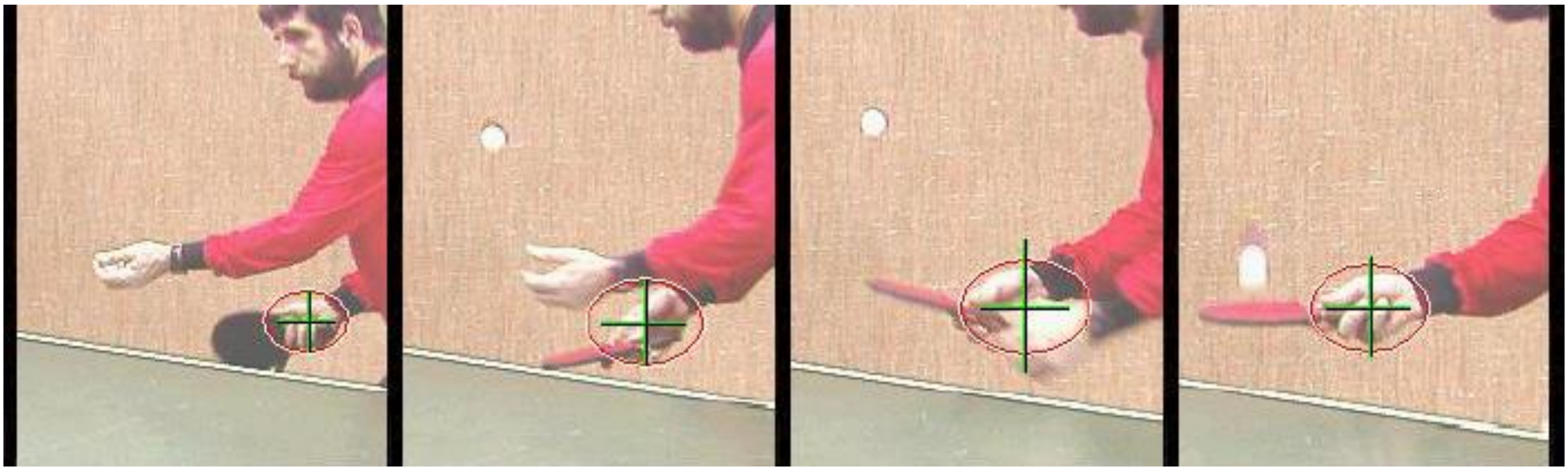
Goal: find the best candidate location in frame 2



Frame 1 Frame 2

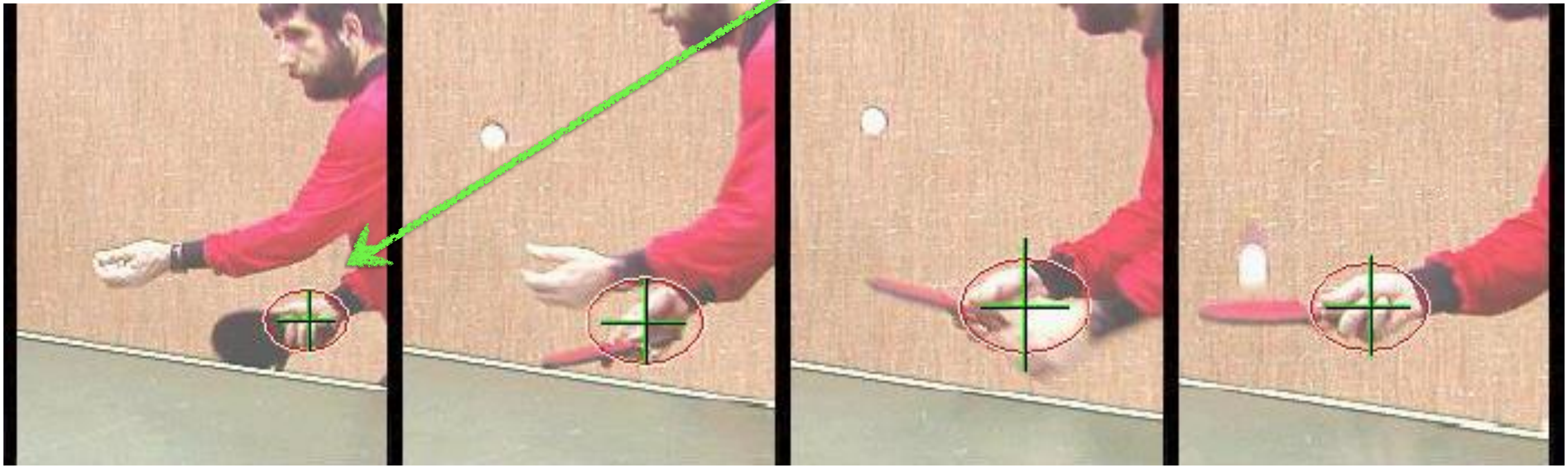
Use the mean shift algorithm
to find the best candidate location

Non-rigid object tracking



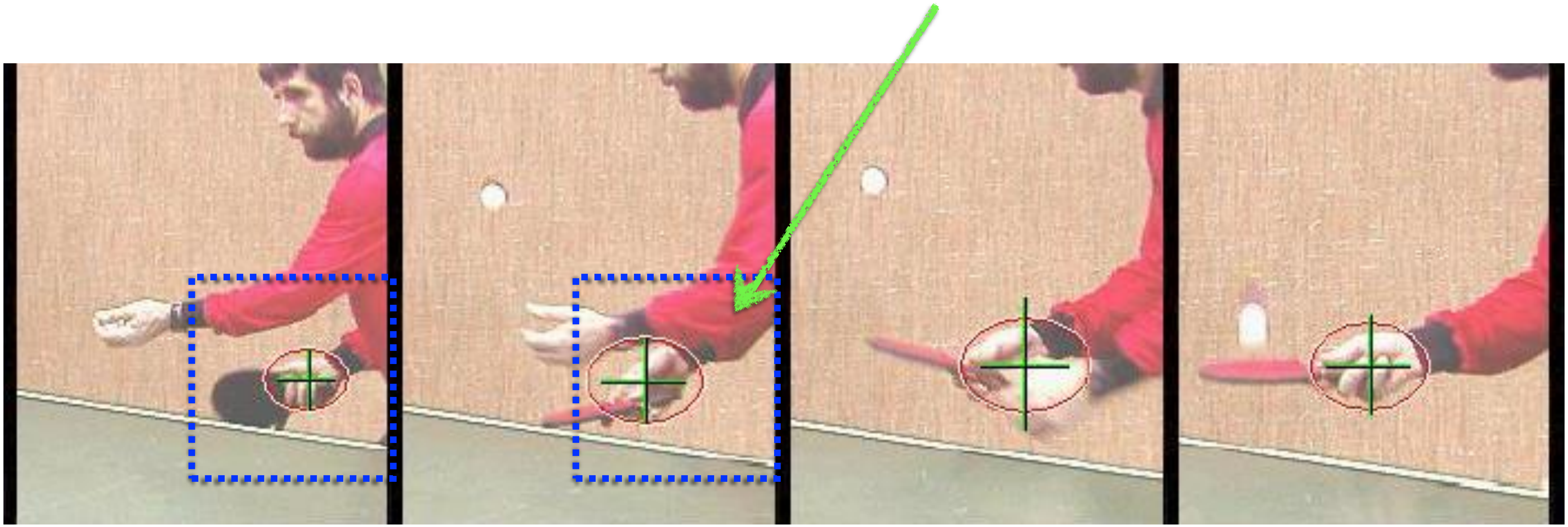
hand tracking

Compute a descriptor for the target



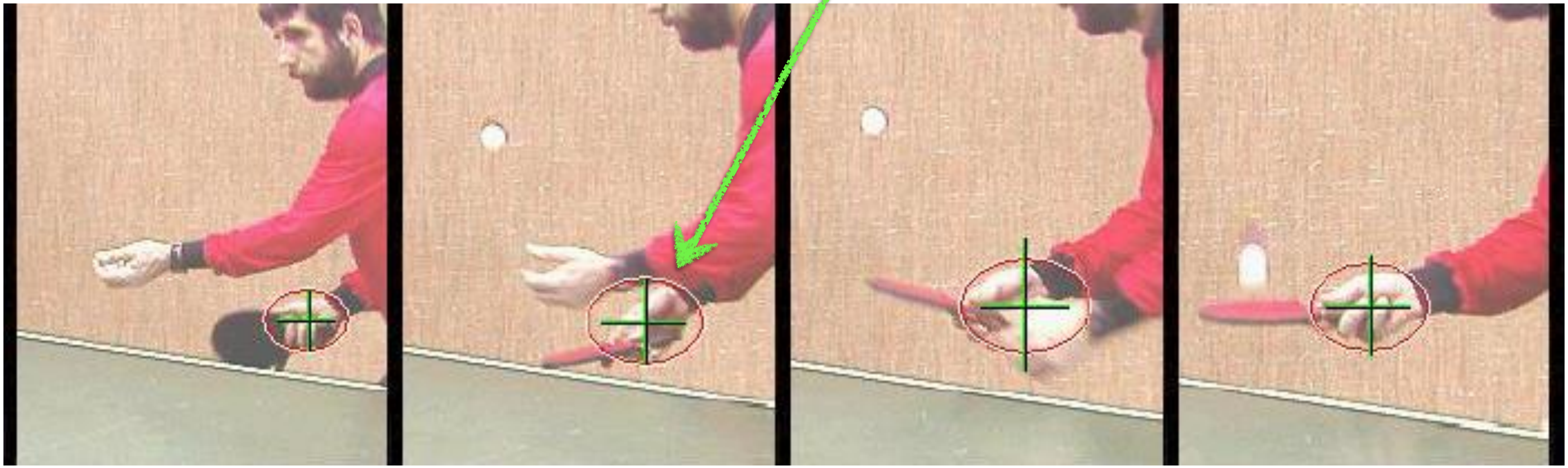
Target

Search for similar descriptor in neighborhood in next frame



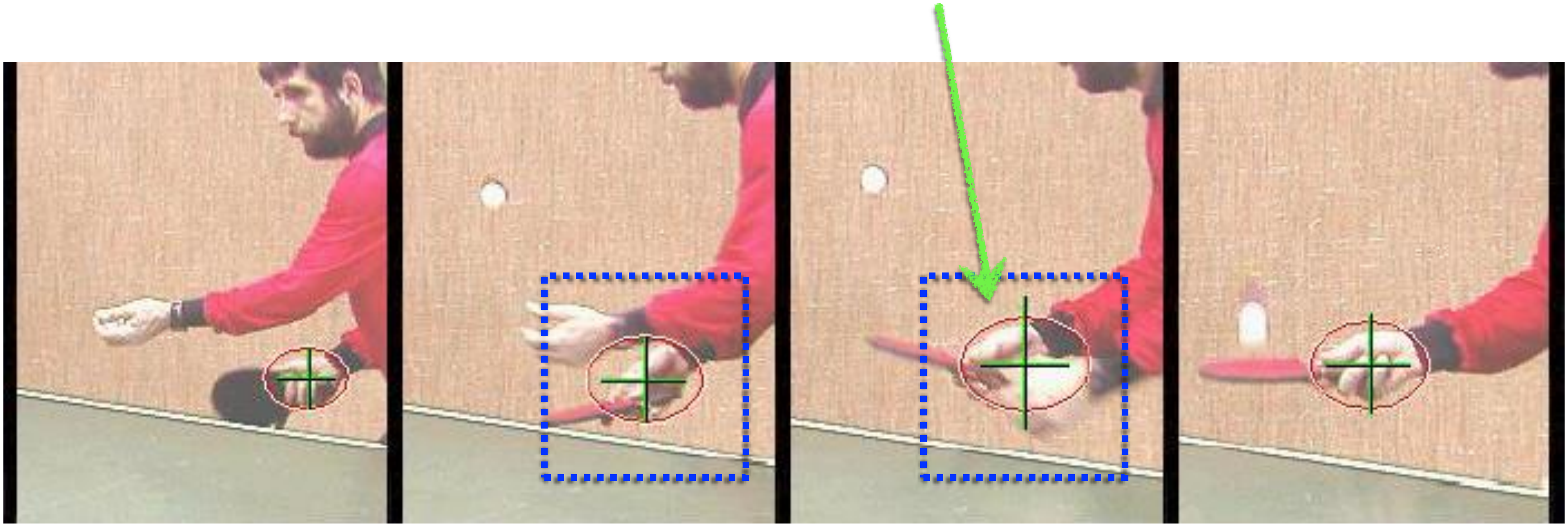
Target Candidate

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

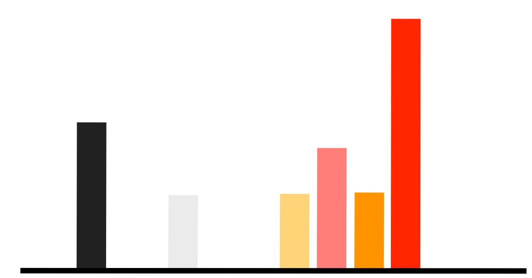
sum over all pixels

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

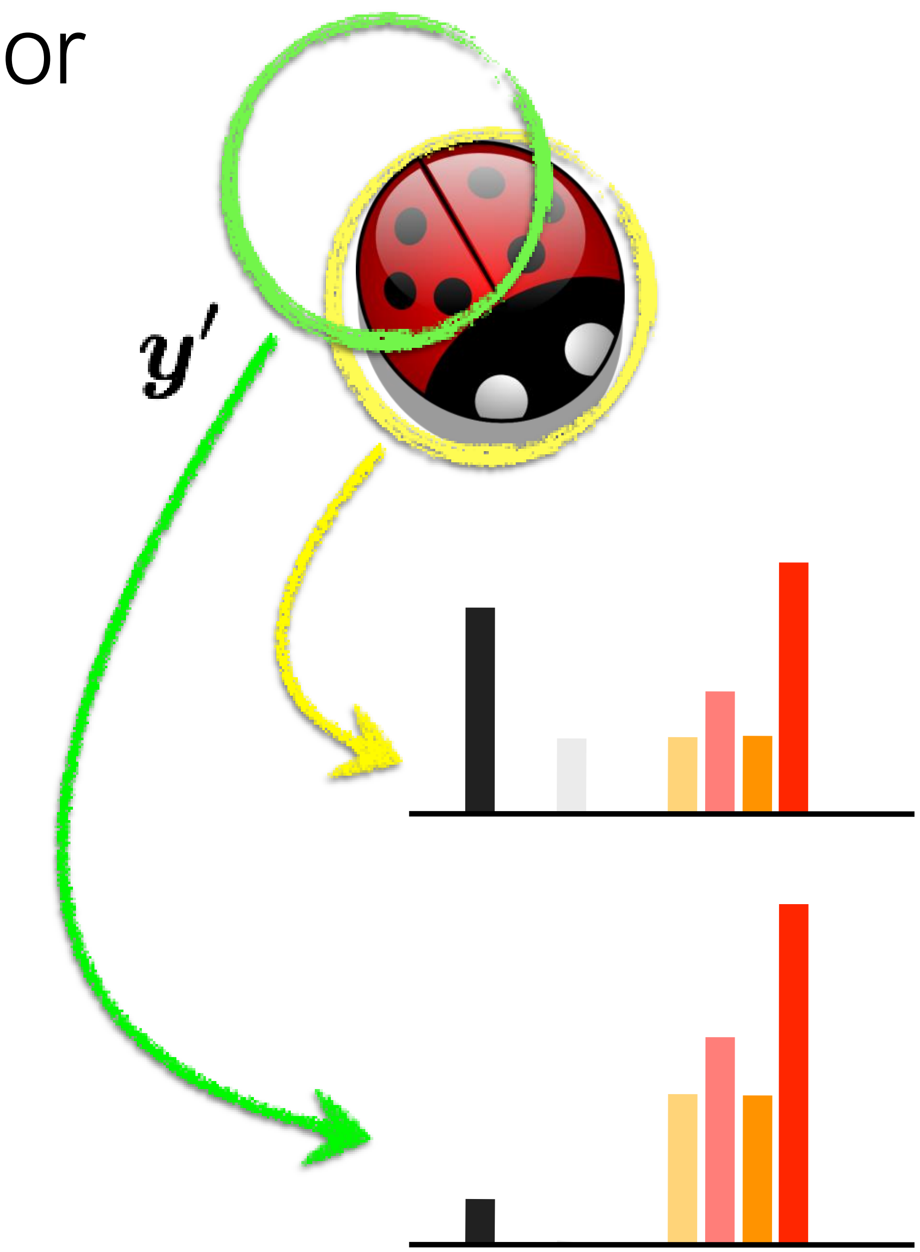
$$\boldsymbol{p}(\boldsymbol{y}) = \{p_1(\boldsymbol{y}), \ldots, p_M(\boldsymbol{y})\}$$

(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

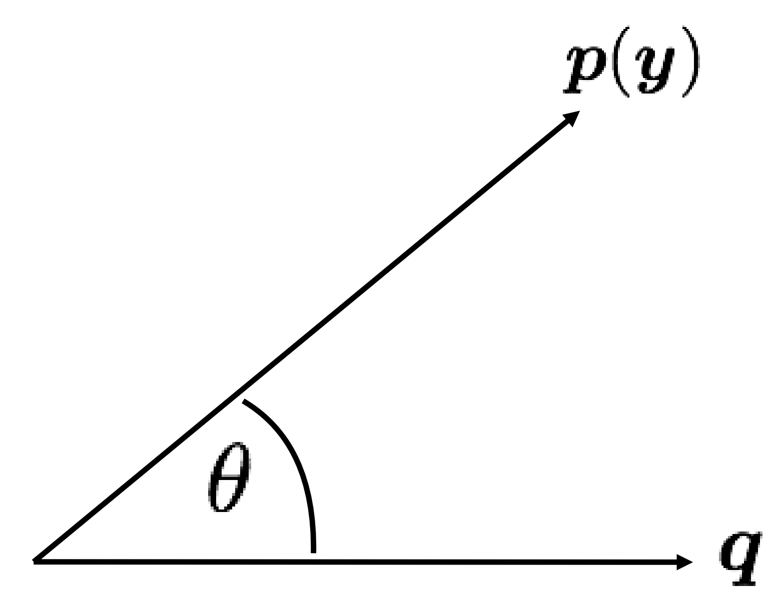
$$d(\boldsymbol{y}) = \sqrt{1 - \rho[\boldsymbol{p}(\boldsymbol{y}), \boldsymbol{q}]}$$

Bhattacharyya Coefficient

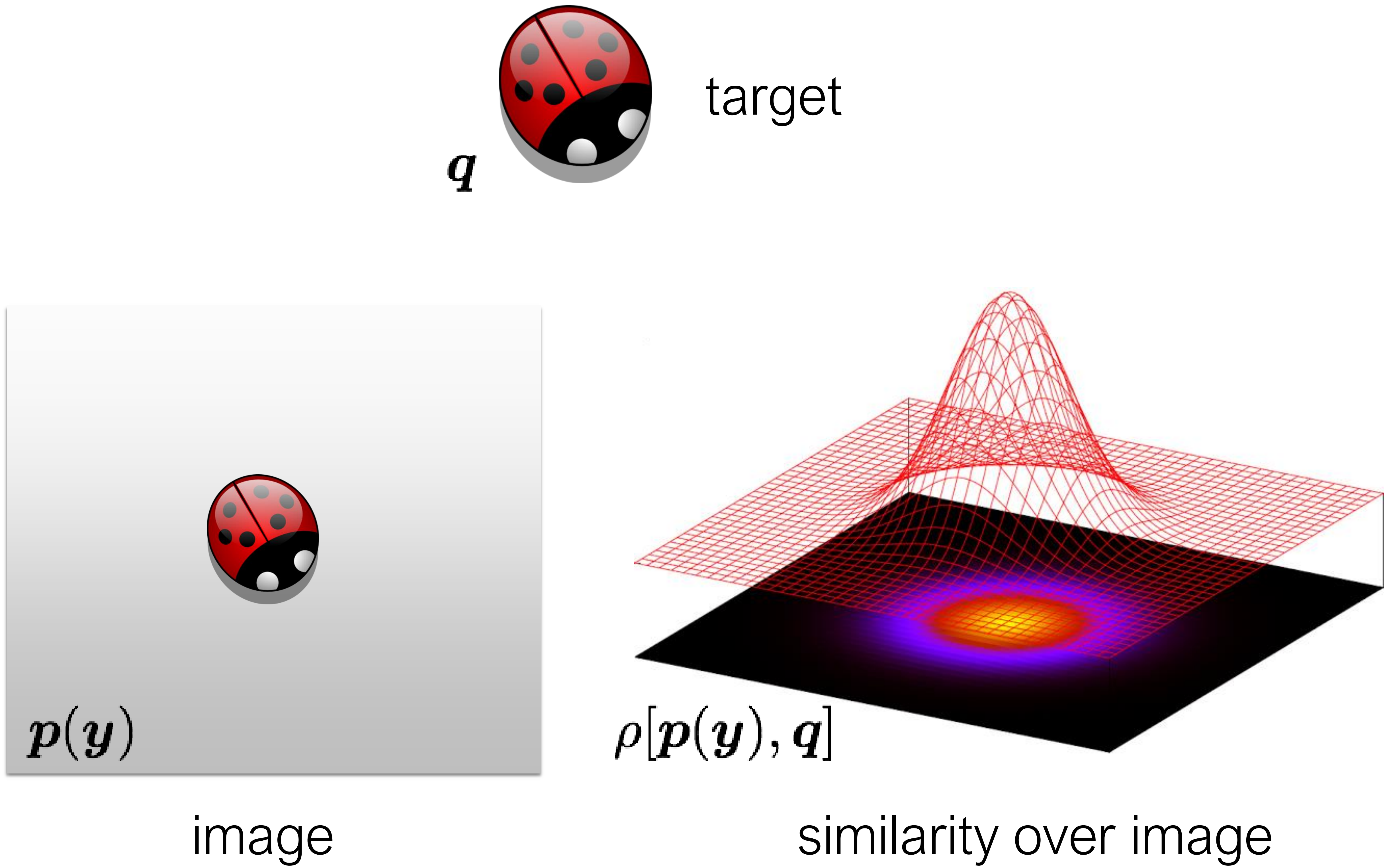
$$\rho(y) \equiv \rho[\boldsymbol{p}(\boldsymbol{y}), \boldsymbol{q}] = \sum_m \sqrt{p_m(\boldsymbol{y}) q_u}$$

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

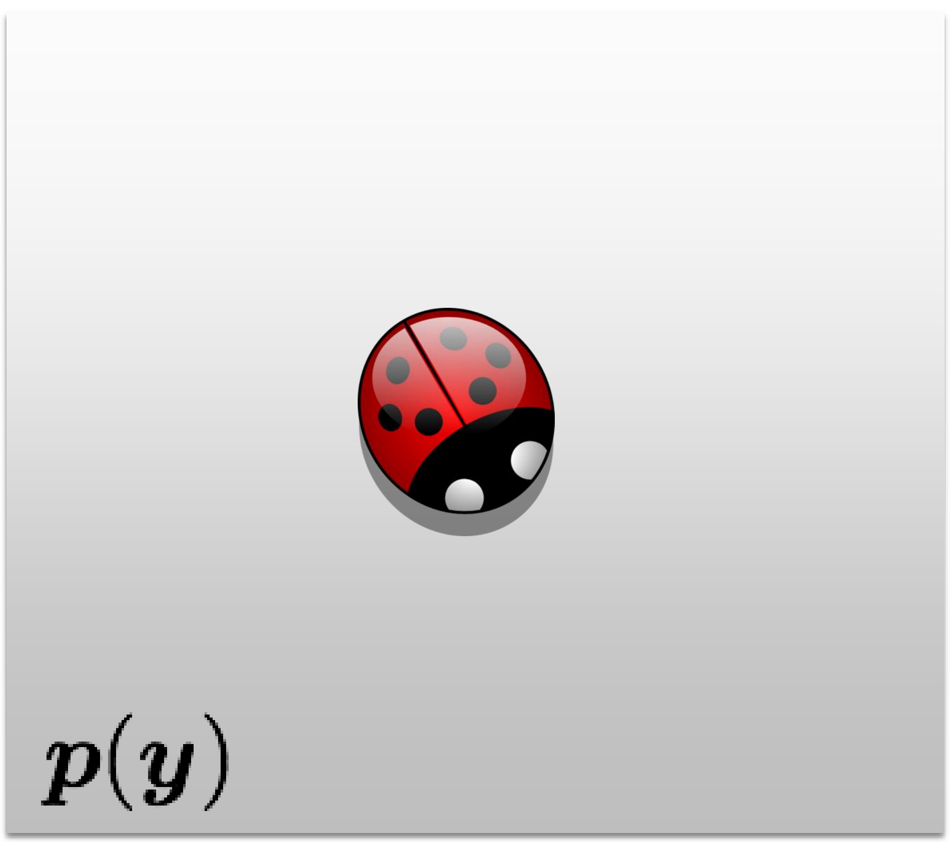


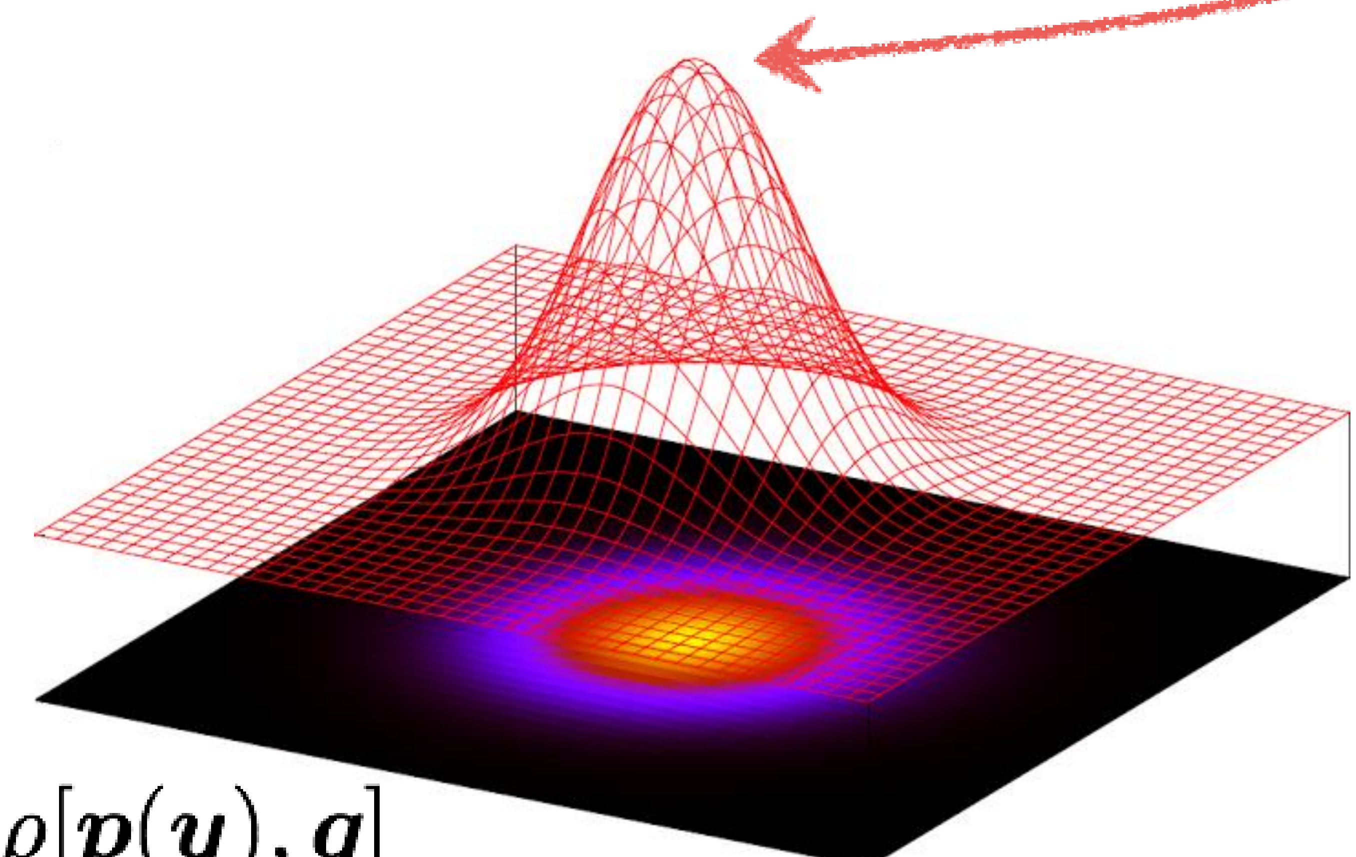
$\boldsymbol{q}$
target
$\boldsymbol{p}(\boldsymbol{y})$
$\rho[\boldsymbol{p}(\boldsymbol{y}), \boldsymbol{q}]$
image
similarity over image

q

target

we want to find this peak





$\rho[p(y), q]$

image

similarity over image

Objective function

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

Assuming a good initial guess

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

Linearize around the initial guess (Taylor series expansion)

$$\rho[\boldsymbol{p}(\boldsymbol{y}), \boldsymbol{q}] \approx \underbrace{\frac{1}{2} \sum_m \sqrt{p_m(\boldsymbol{y}_0) q_m}}_{\text{function at specified value}} + \frac{1}{2} \sum_m p_m(\boldsymbol{y}) \underbrace{\sqrt{\frac{q_m}{p_m(\boldsymbol{y}_0)}}}_{\text{derivative}}$$

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…

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

Does not depend on unknown **y**

Weighted kernel density estimate

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

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

OK, why are we doing all this math?

We want to maximize this

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

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

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

doesn't depend on unknown **y**

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

We want to maximize this

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

only need to maximize this!

Fully expanded linearized objective

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

doesn't depend on unknown **y**

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

We want to maximize this

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

Fully expanded linearized objective

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

doesn't depend on unknown **y**

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

what can we use to solve this weighted KDE?

Mean Shift Algorithm!

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

the new sample of mean of this KDE is

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

(new candidate location)

(this was derived earlier)

Mean-Shift Object Tracking

For each frame:

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

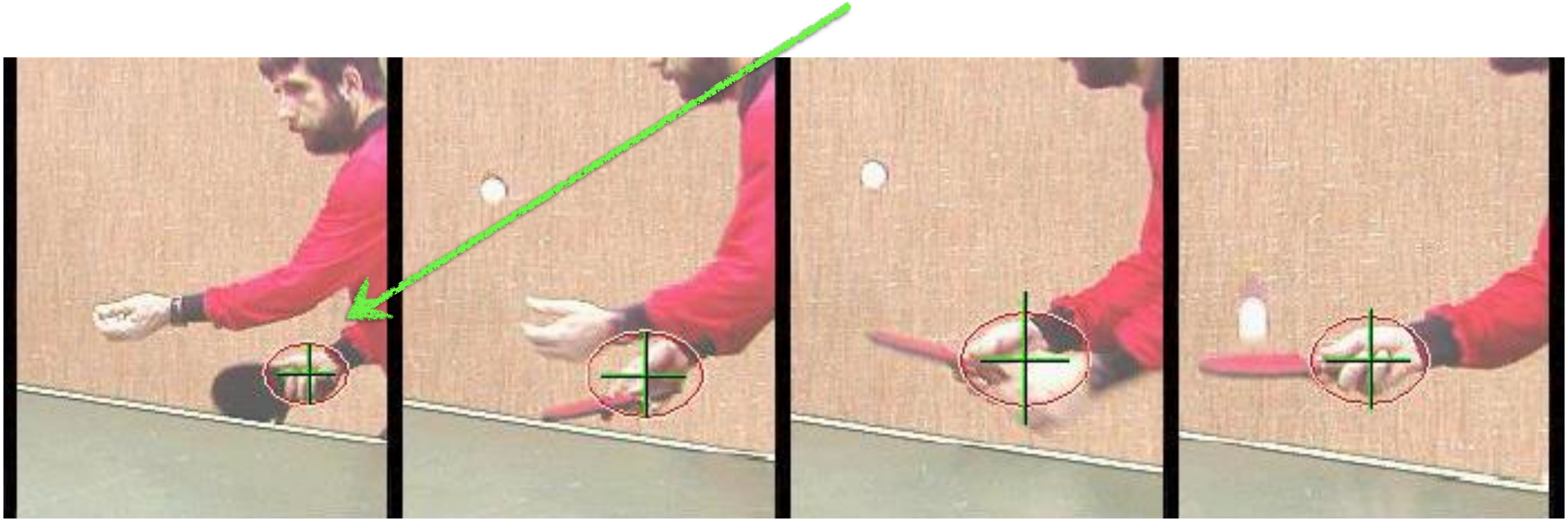
2. Derive weights w_n

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

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

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

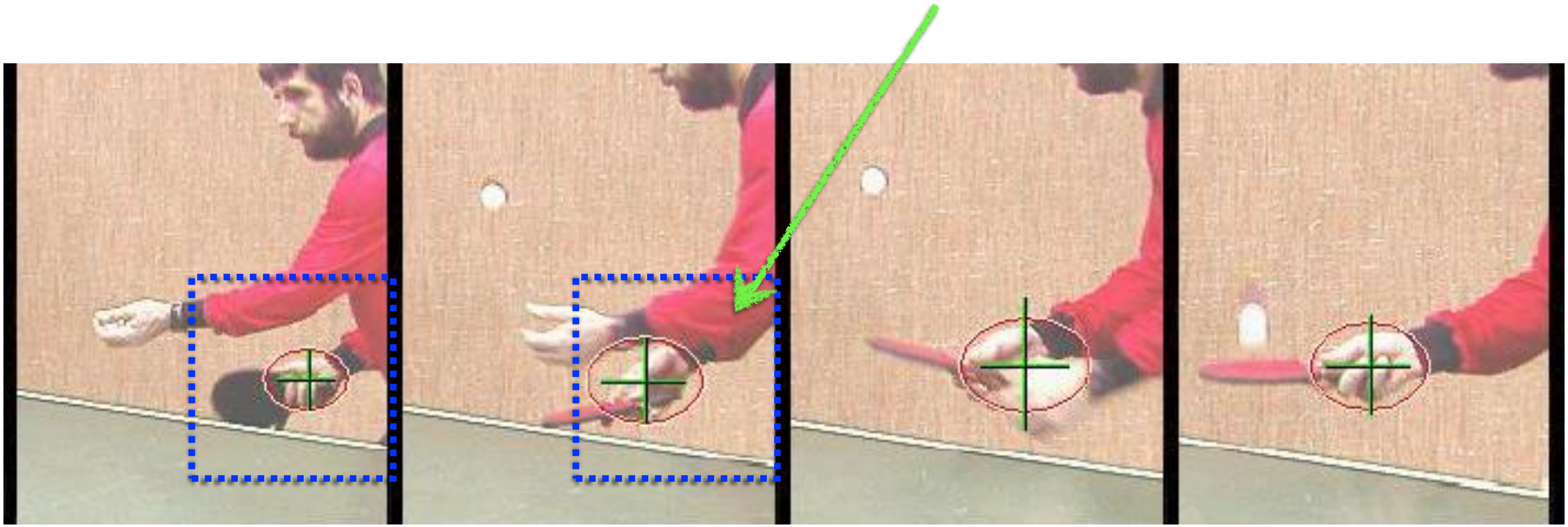
Compute a descriptor for the target



Target

$\boldsymbol{q}$

Search for similar descriptor in neighborhood in next frame

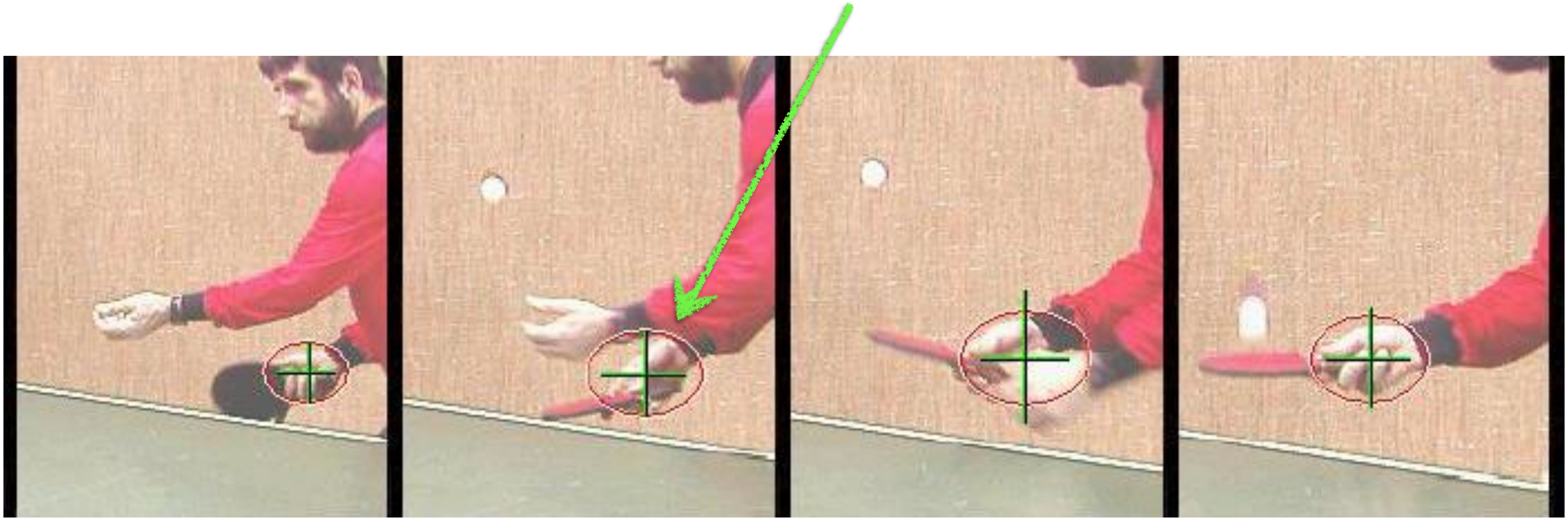


Target

Candidate

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

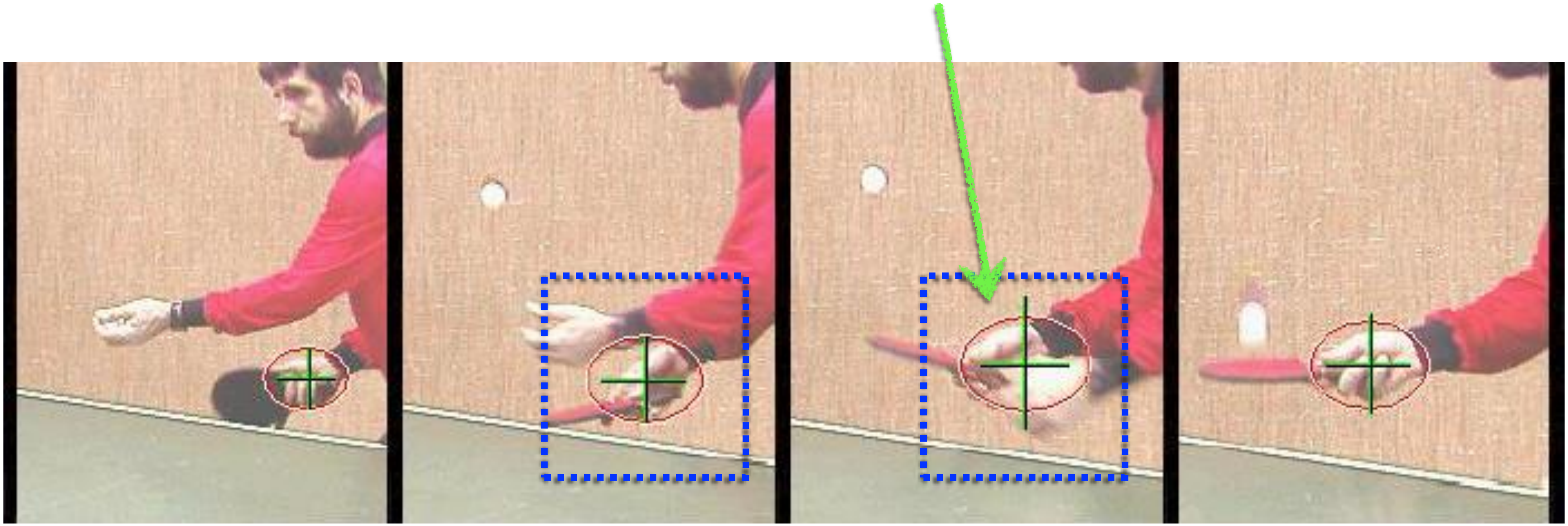
Compute a descriptor for the new target



Target

$\boldsymbol{q}$

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.