Vision and Cognitive Systems

SCQ1097939 - LM CS,DS,CYB,PD

Foundations: edges, keypoints and local features

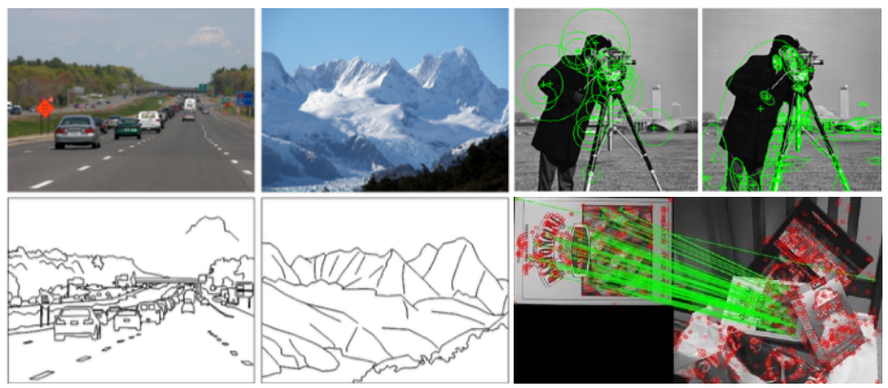
Prof. Lamberto Ballan

University of Padova, A.Y. 2023/24 November 6, 2023

DIPARTIMENTO MATEMATICA
Department of Mathematics “Tullio Levi-Civita”

What we will learn this week?

- Edge detection
- Image gradients
- Local invariant features
- DoG and SIFT descriptors



Chapter 4

Image filtering recap

- Compute a function of the local neighborhood at each pixel in the image
  - The function is specified by a “filter” saying how to combine values from neighbors

- Applications of image filtering:
  - extract information (edges, corners, blobs, …)
  - detect patterns (template matching)
  - de-noising, super-resolution, in-painting

Recap: derivate filter (Prewitt)

$$I = \begin{bmatrix} 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \end{bmatrix} * \frac{1}{3}\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

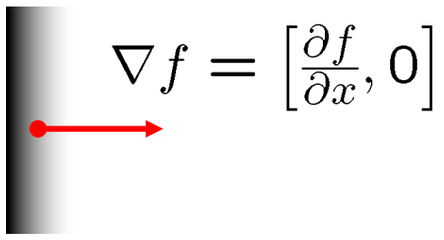
Vertical Derivatives

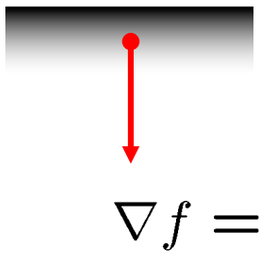
Horizontal Derivatives

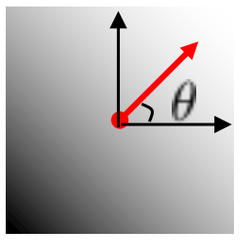
$$* \frac{1}{3}\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 10 & 10 & 0 & 0 \\ 0 & 10 & 10 & 0 & 0 \\ 0 & 10 & 10 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Recap: image gradient

- The gradient of an image: $\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$





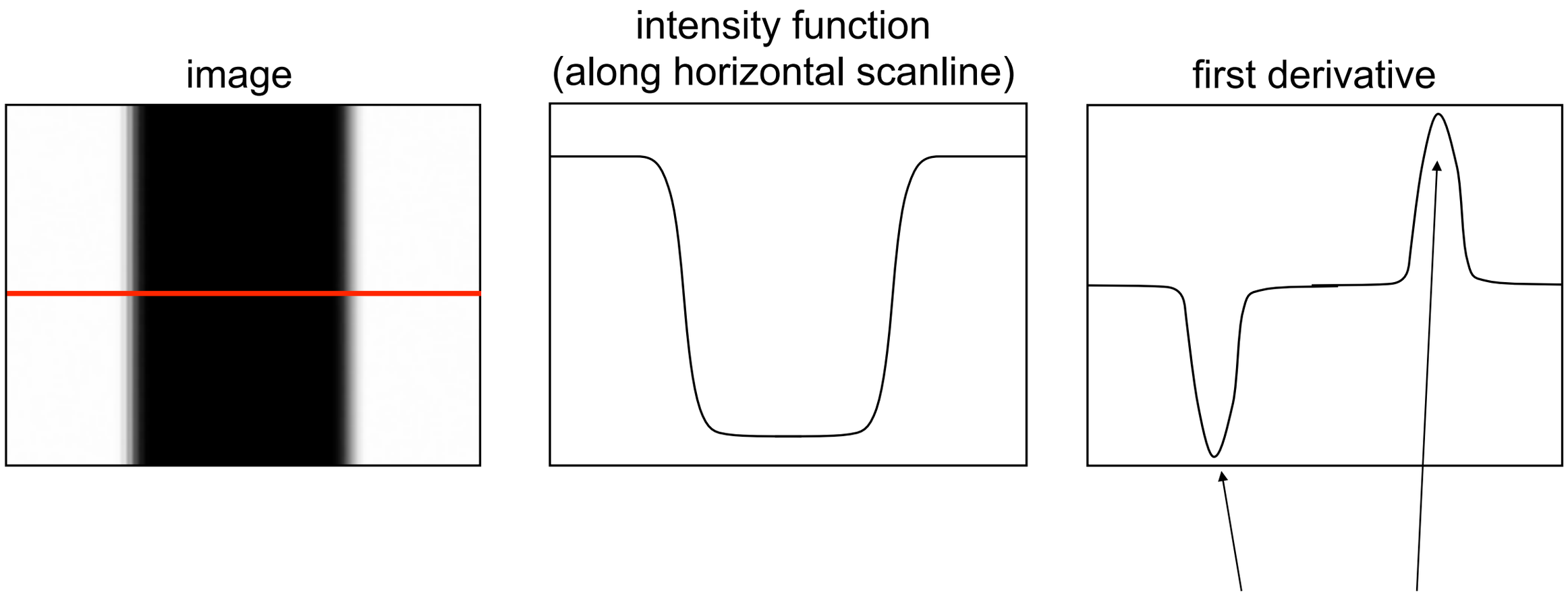


  - The gradient vector points in the direction of most rapid increase in intensity
  - The gradient direction is given by: $\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$
  - The *edge strength* is given by the gradient magnitude:

$$\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

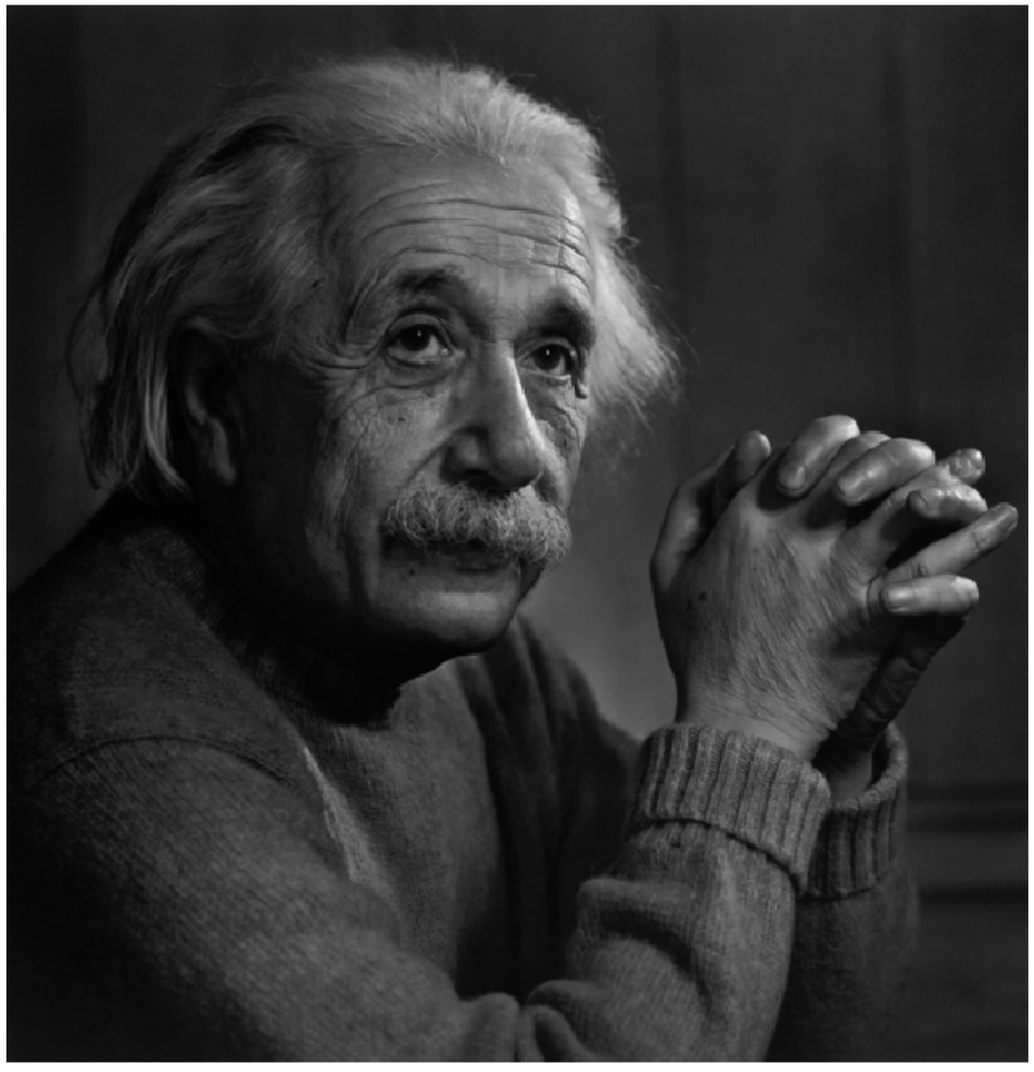
Recap: edges and image gradients

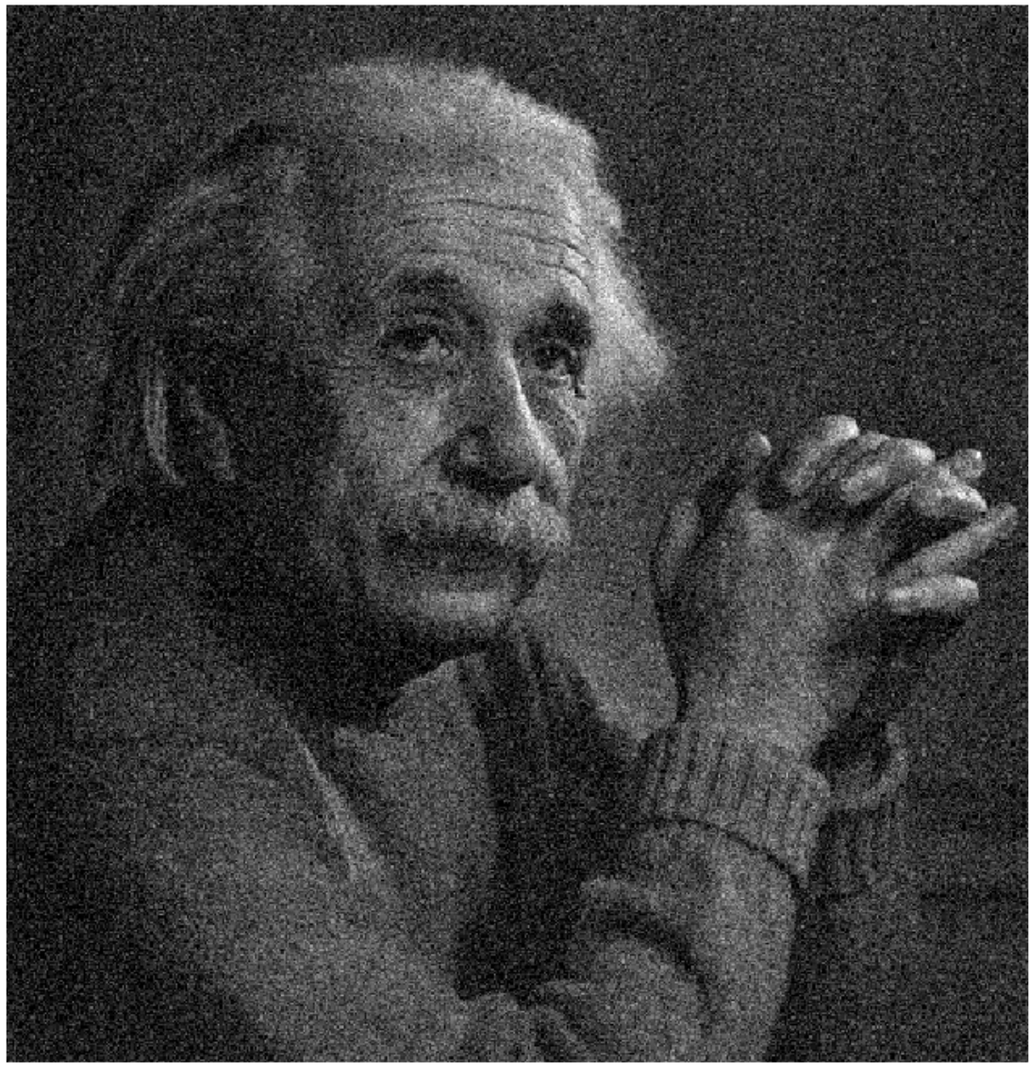
- An edge is a place of rapid change in the image intensity function



This could be the sketch of an edge detection algorithm

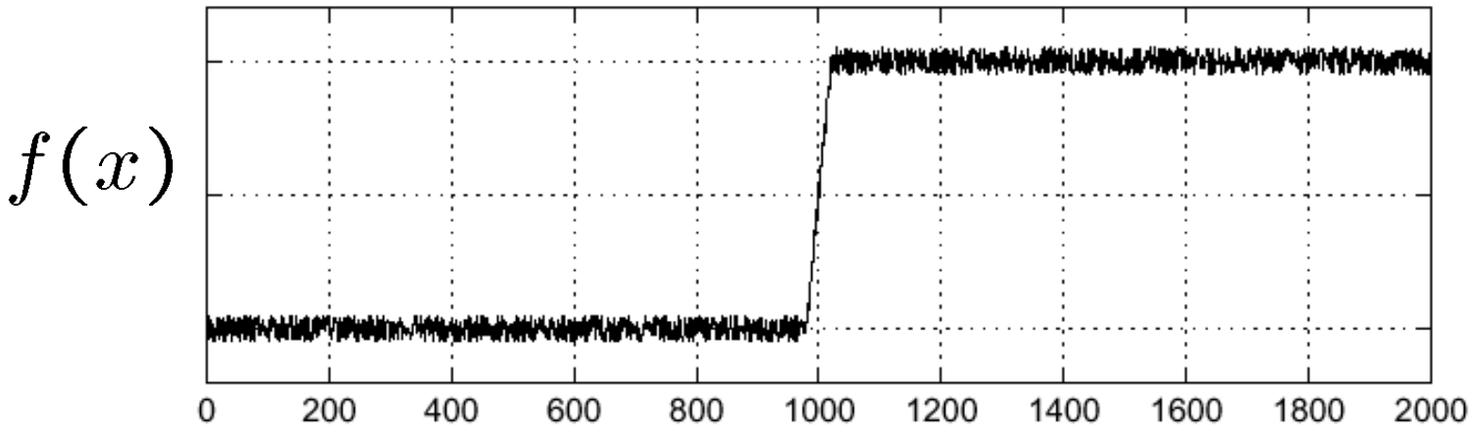
Effects of noise

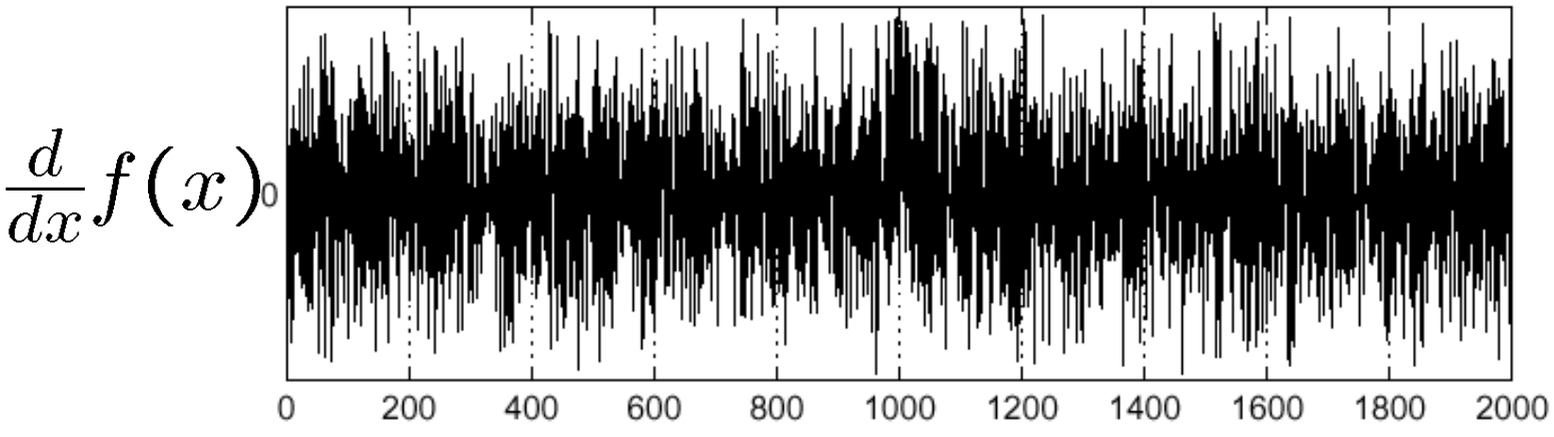




Effects of noise

- Consider a single row or column of the image
  - Plotting intensity as a function of position gives a signal





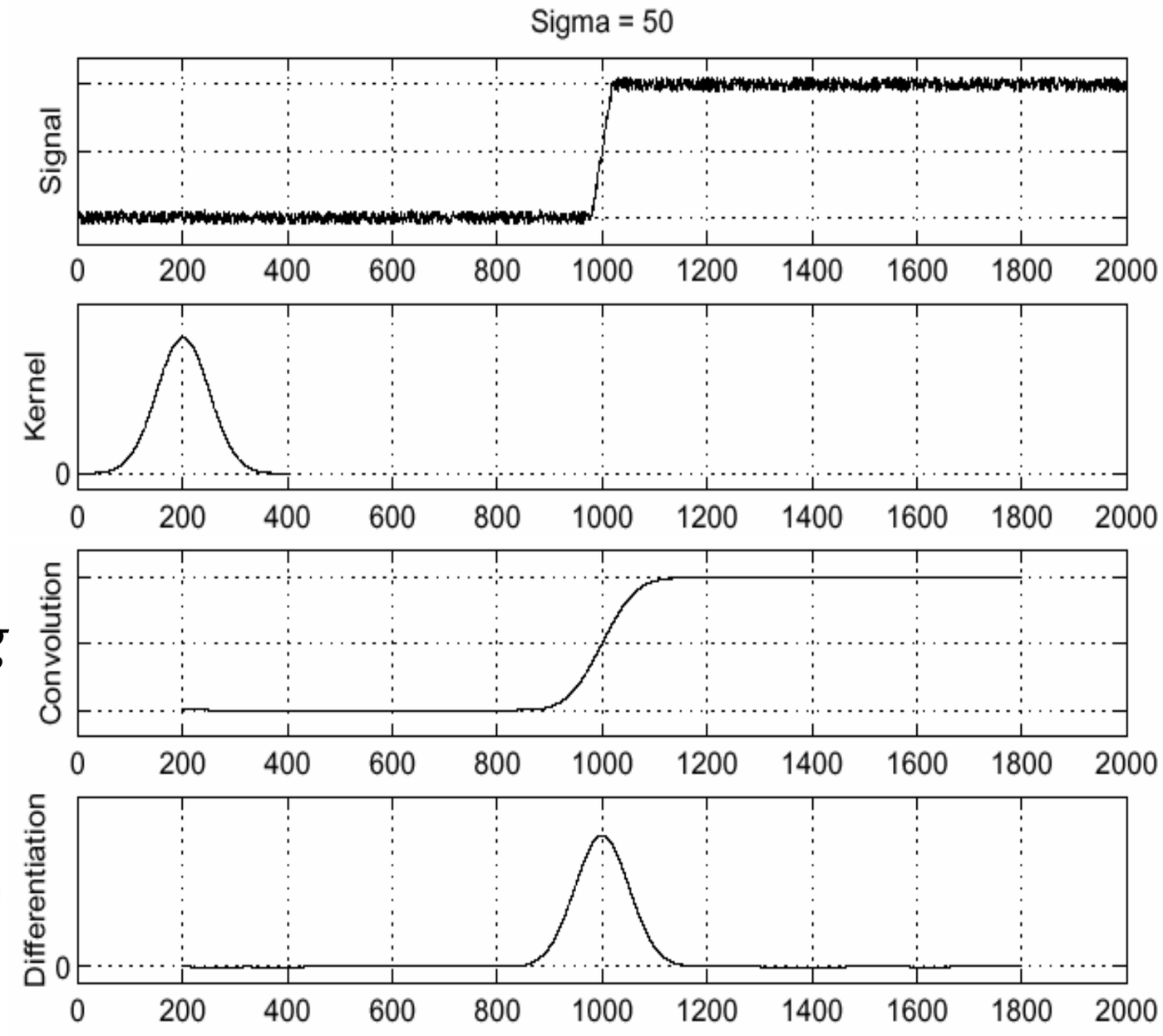
Where is the edge?

Effects of noise

- Finite difference filters respond strongly to noise
  - Image noise results in pixels that look very different from their neighbors
  - The larger the noise, the stronger the response

- What is to be done?
  - Smoothing the image should help, by forcing pixels different to their neighbors (*=noise pixels?*) to look more like neighbors

Solution: smooth first



f

g

$f * g$

$\frac{d}{dx}(f * g)$

Where is the edge?

Look for peaks in $\frac{d}{dx}(f * g)$

Derivative theorem of convolution

- Differential property of convolution:

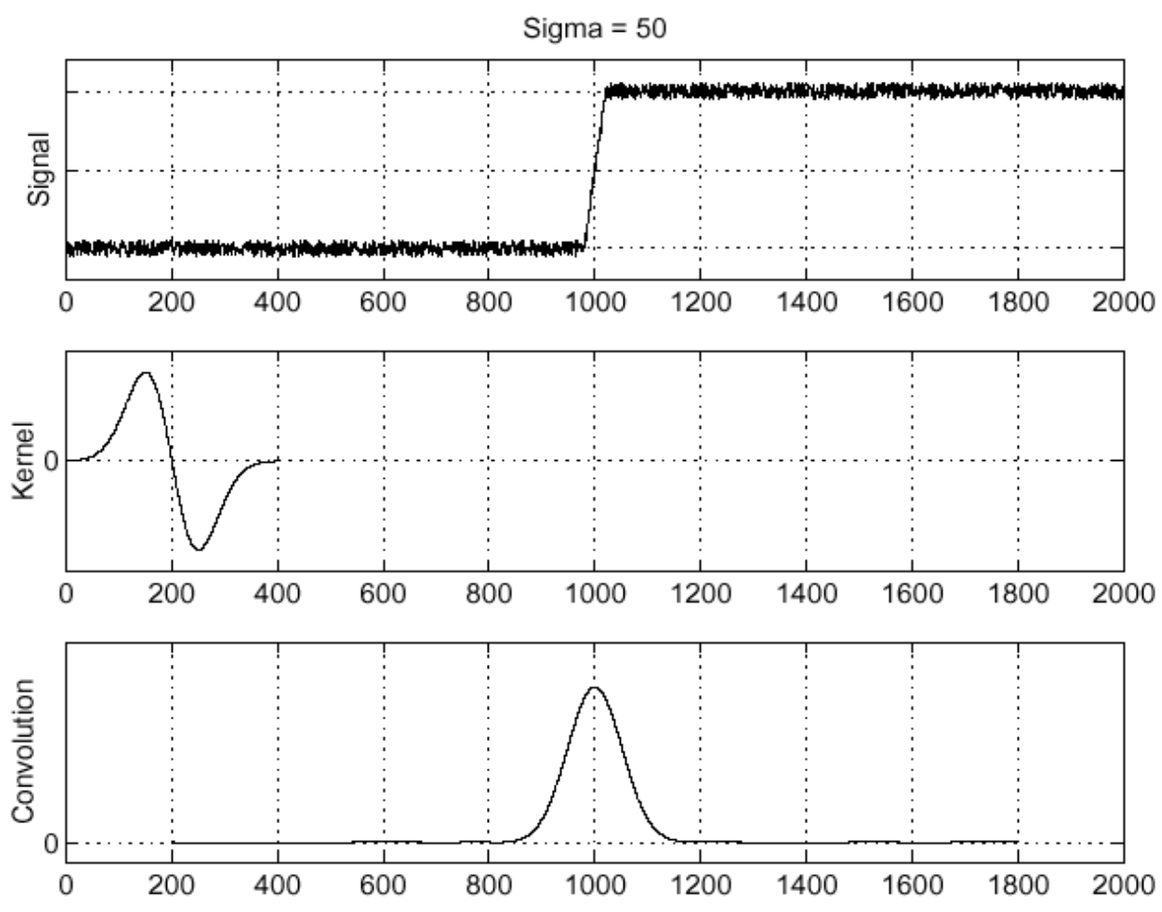
$$\frac{d}{dx}(f * g) = f * \frac{d}{dx} g$$

  - This is a useful property and save us one operation:

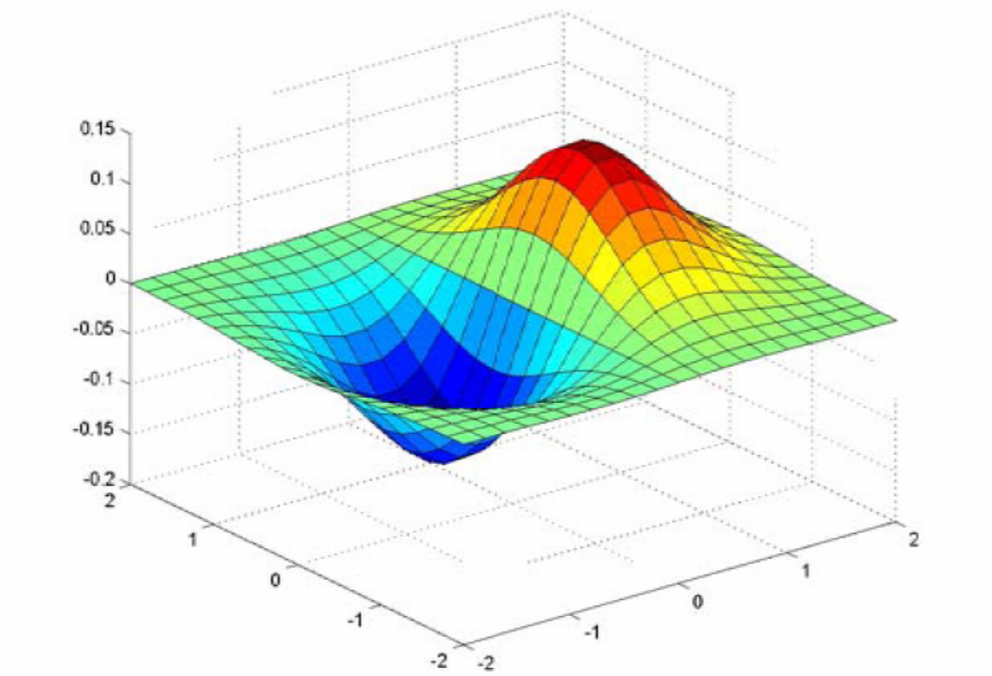
f

$\frac{d}{dx} g$

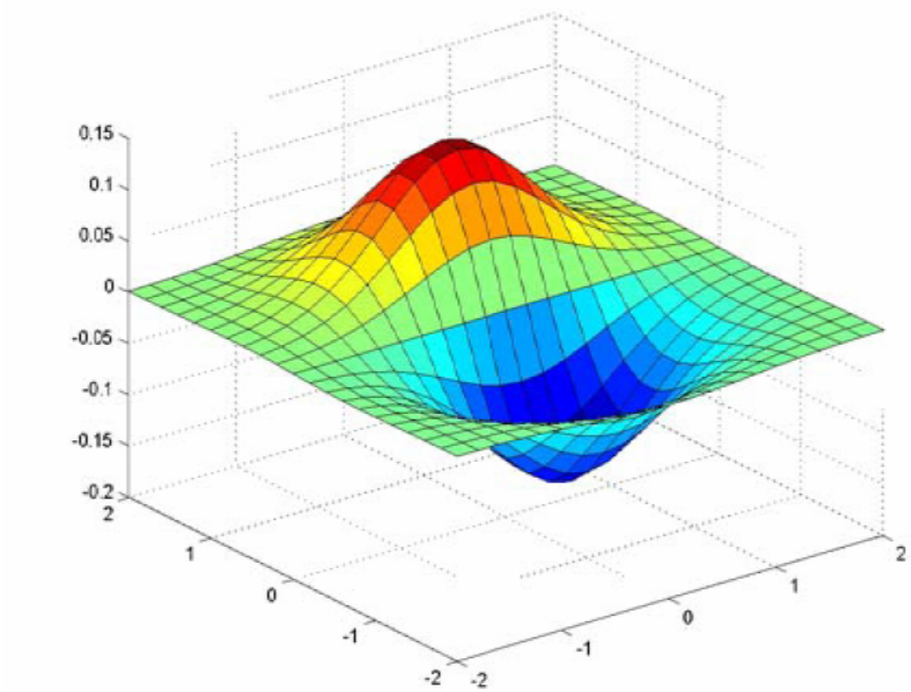
$f * \frac{d}{dx} g$



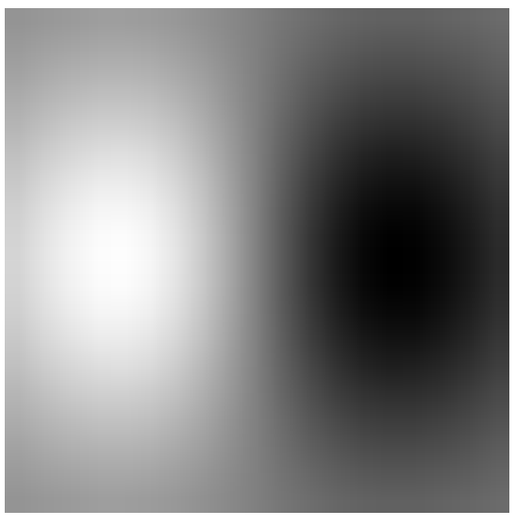
Derivative of Gaussian filter

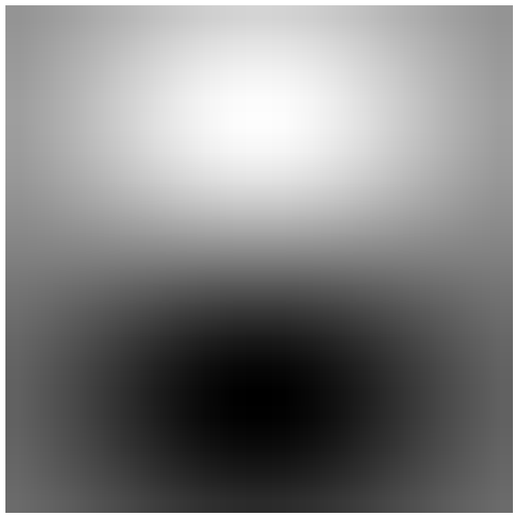


x-direction

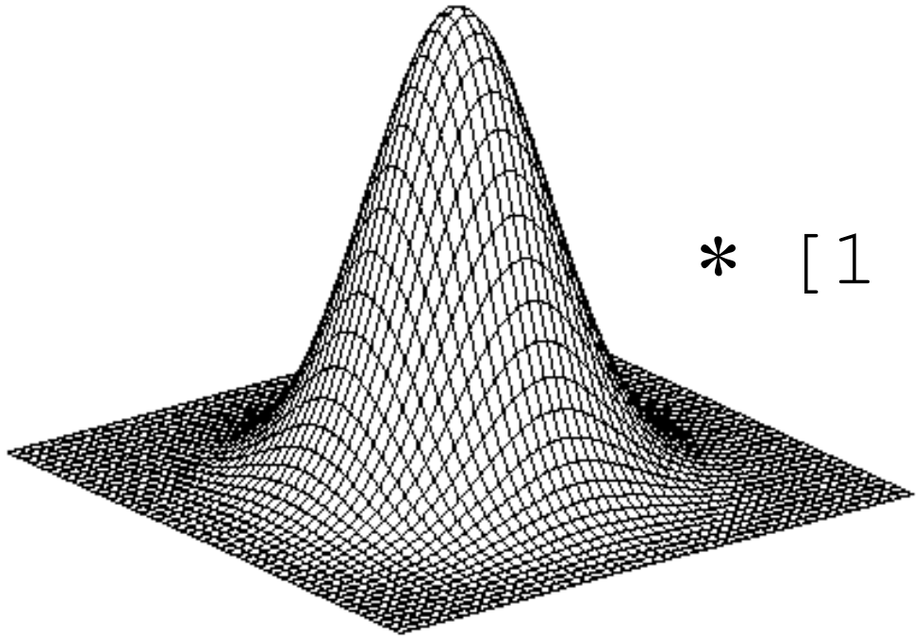


y-direction

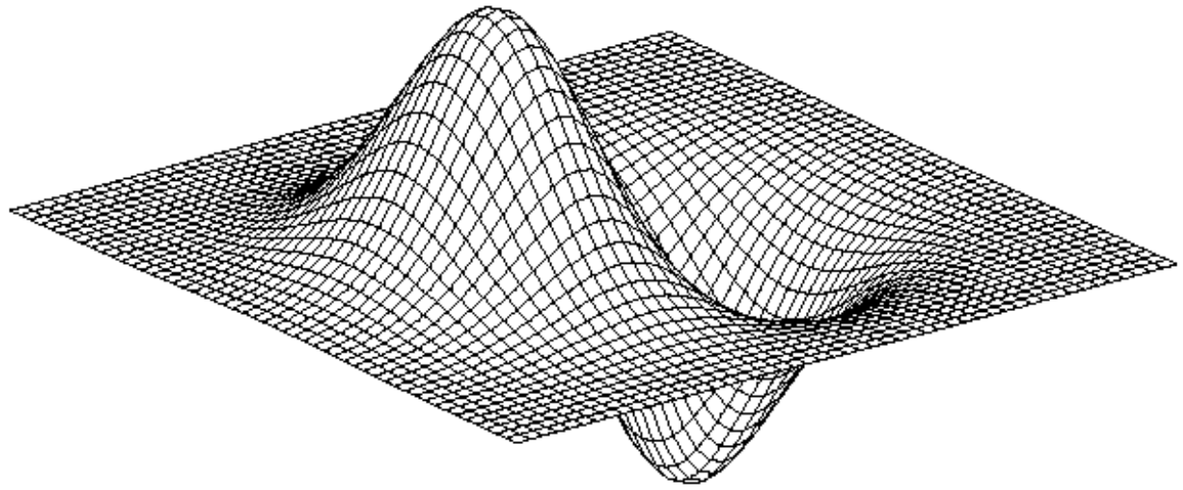




Derivative of Gaussian filter



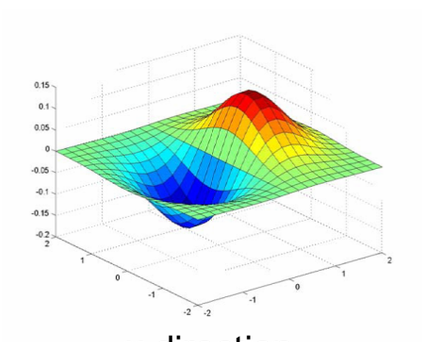
* [1 0 -1] =



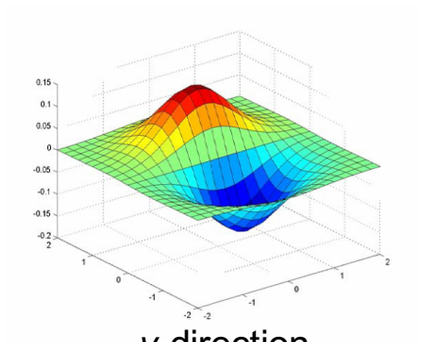
2D Gaussian

x-derivative

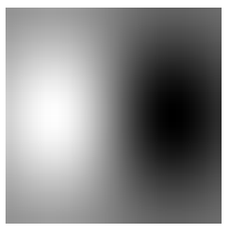
Derivative of Gaussian filter

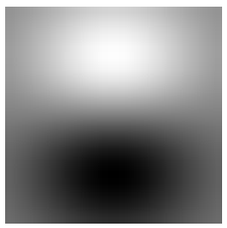


x-direction



y-direction





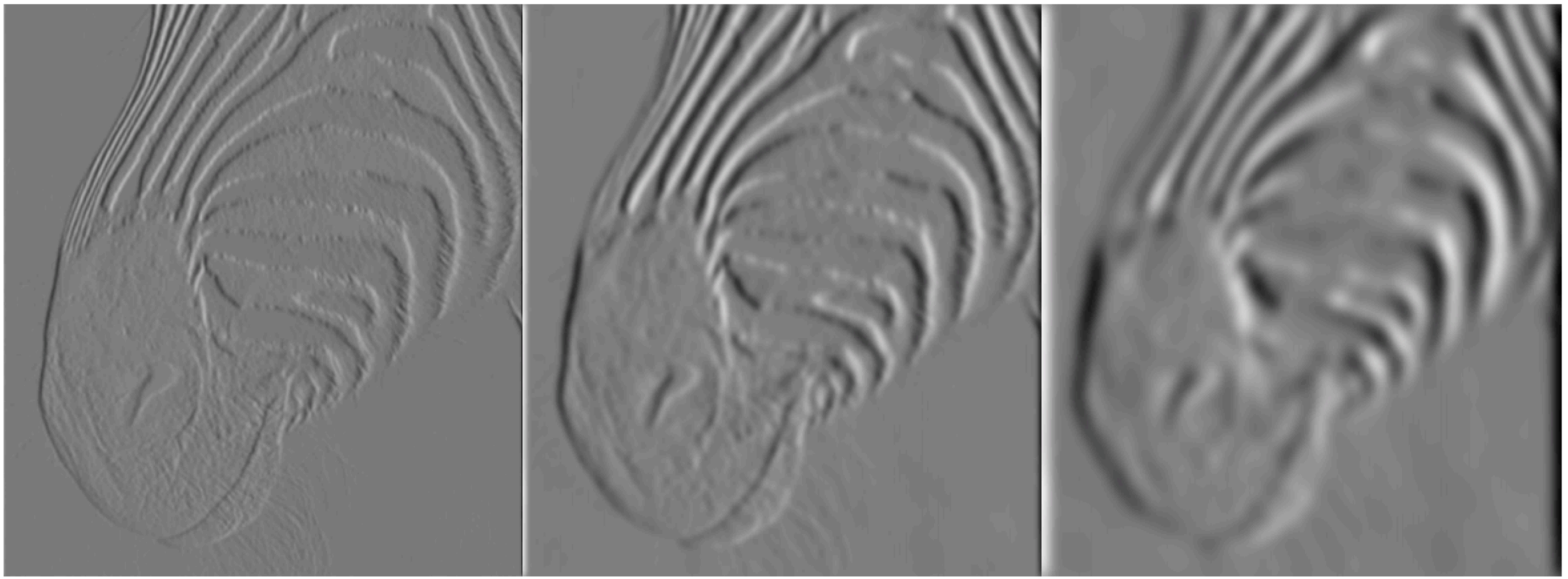
Note: separability of the Gaussian Filter

$$G_\sigma(x,y) = \frac{1}{2\pi\sigma^2}\exp^{-\frac{x^2+y^2}{2\sigma^2}}$$

$$= \left(\frac{1}{\sqrt{2\pi}\sigma}\exp^{-\frac{x^2}{2\sigma^2}}\right)\left(\frac{1}{\sqrt{2\pi}\sigma}\exp^{-\frac{y^2}{2\sigma^2}}\right)$$

- The 2D Gaussian can be expressed as the product of 2 functions (one a function of *x* and the other a function of *y*)
- In this case the two functions are the (identical) 1D Gaussian

Scale of Gaussian derivative filter

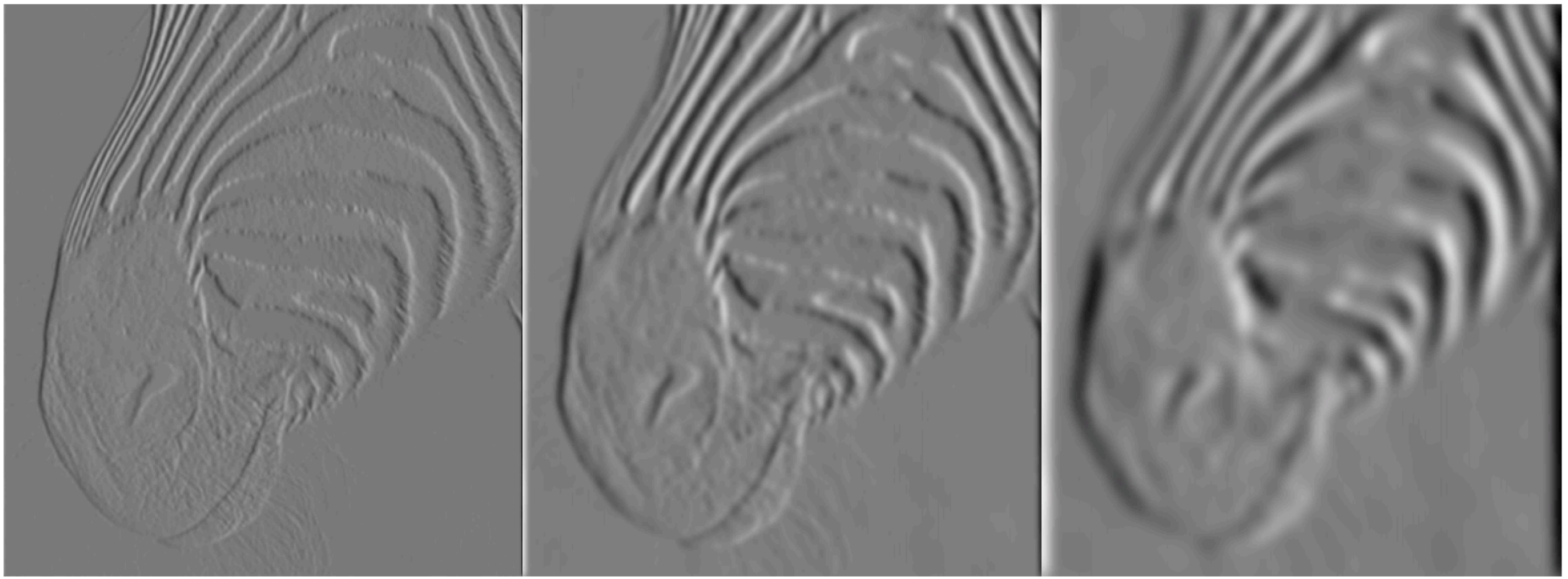


1 pixel 3 pixels 7 pixels

- Note: σ is the *scale* (*width*/*spread*) of the Gaussian kernel
- Smoothed derivative remove noise, but blurs edges

Image credit: D. Forsyth

Scale of Gaussian derivative filter



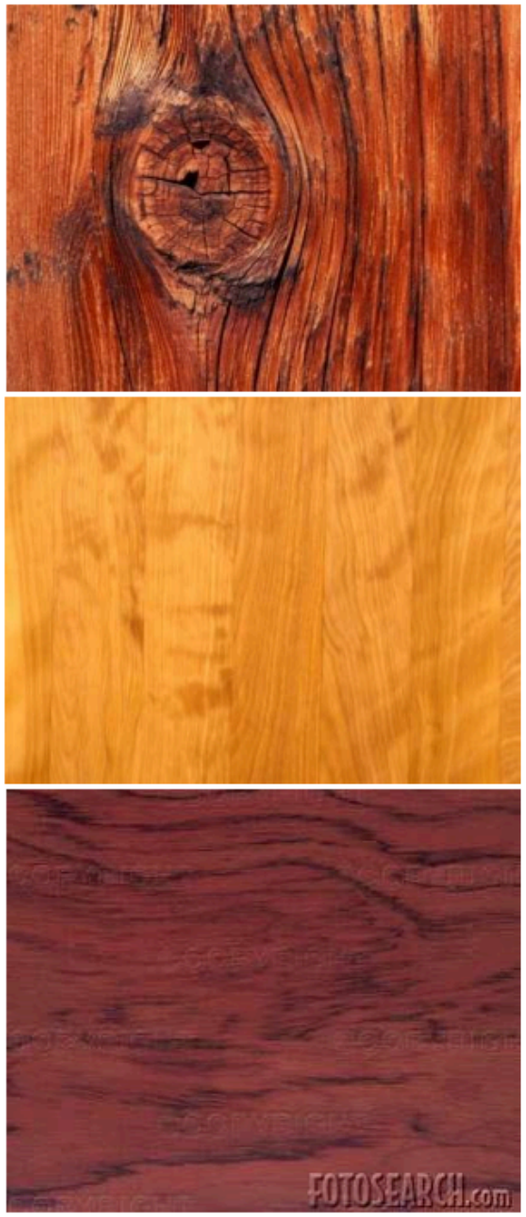
1 pixel 3 pixels 7 pixels

- The apparent structures differ depending on scale σ
  - Larger values: larger scale edges detected
  - Smaller values: finer features detected

Image credit: D. Forsyth

So, what scale to choose?

- It depends what we are looking for…



Edge detection: Sobel filter

- A derivative filter + some smoothing

Input image

$*$

1	0	-1
2	0	-2
1	0	-1

Left Sobel
(Horizontal Derivatives)

Filter returns large response on vertical or horizontal lines?

A: Responds to vertical lines

Is the output always positive?

A: Output can be positive or negative

Edge detection: Sobel filter

- Where does this filter come?
  - Remember: (central) discrete derivative in 1D

$$\frac{df}{dx} = f(x+1) - f(x-1) = f'(x)$$

```
[1  0  -1]
```

1D derivative filter

  - Example:

10	20	10	200	210	250	250

↑

$$f'(x) = f(x+1) - f(x-1) = 210 - 10 = 200$$

Edge detection: Sobel filter

- Decomposing the Sobel Filter

Input image

$*$

1	0	-1
2	0	-2
1	0	-1

Left Sobel

1
2
1

1	0	-1

1D x-derivative filter

What's this? *A: Weighted average and scaling*

Edge detection: derivative filters

- Common derivative filters:

Sobel

1	0	-1
2	0	-2
1	0	-1

1	2	1
0	0	0
-1	-2	-1

Scharr

3	0	-3
10	0	-10
3	0	-3

3	10	3
0	0	0
-3	-10	-3

Prewitt

1	0	-1
1	0	-1
1	0	-1

1	1	1
0	0	0
-1	-1	-1

Roberts

0	1
-1	0

1	0
0	-1

Building a “robust” edge detector

Original (input) image

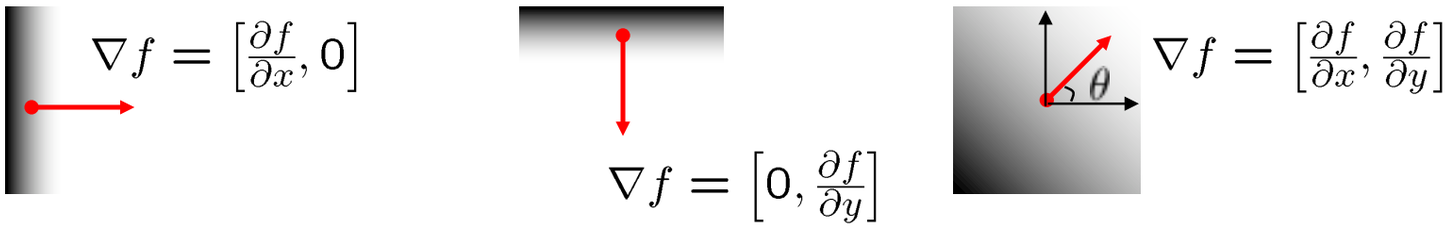
“Expected” Output

Building a “robust” edge detector

- The Canny Edge detector
  - Filter image with Derivative of Gaussian
  - Find magnitude and orientation of gradient
  - …

J. Canny, “ A Computational Approach To Edge Detection”, TPAMI 1986

Image gradient



$$\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$$

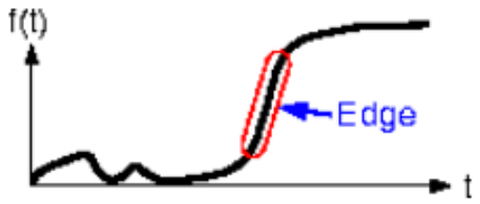
$$\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

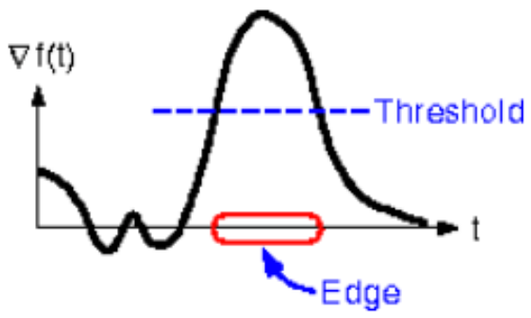
Building an edge detector: Canny

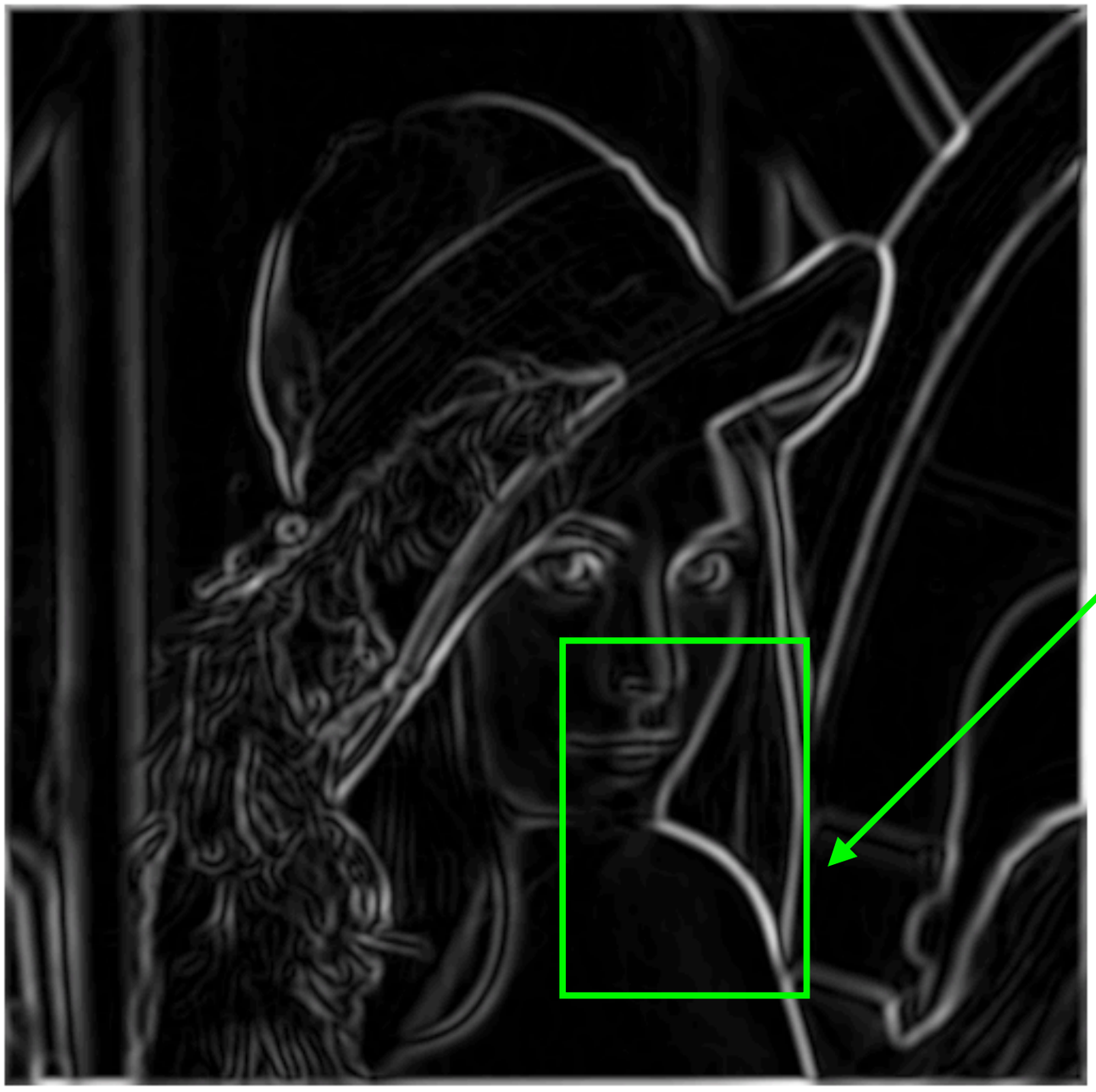
Gradient magnitude

Building an edge detector: Canny

Thresholding





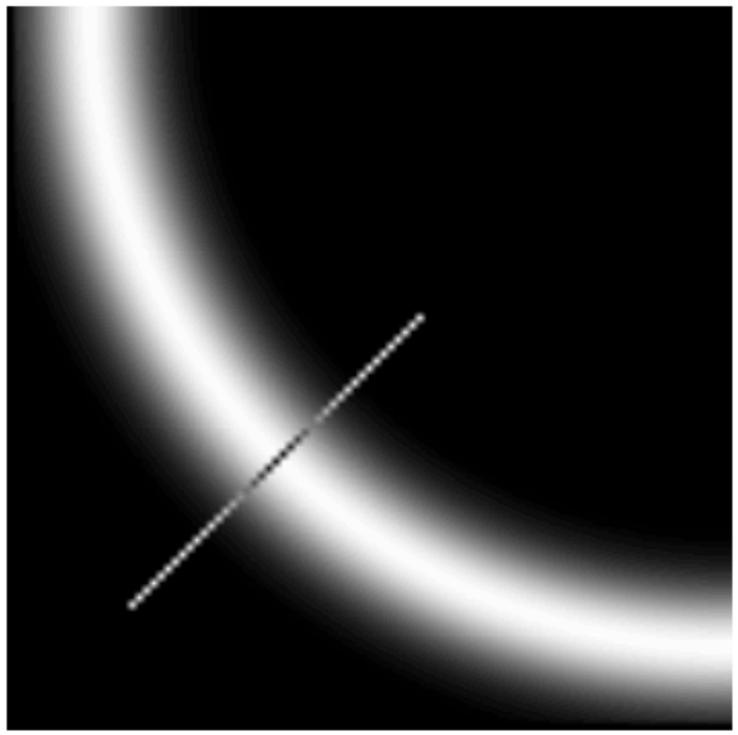


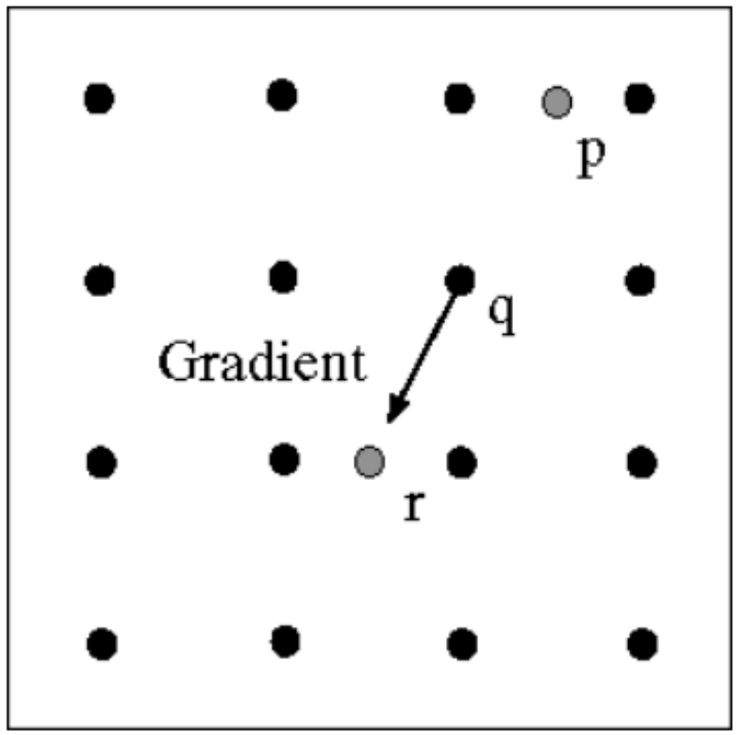
Gradient magnitude

How to turn these thick regions of the gradient into curves?

Building an edge detector: Canny

- Non-maximum suppression
  - Check if pixel is local max along gradient direction
  - Select single maximum across width of the edge (requires checking interpolated pixels $\mathbf{p}$ and $\mathbf{r}$)





Slide credit: S. Seitz

Building an edge detector: Canny

Problem: pixels along this edge didn't survive the thresholding

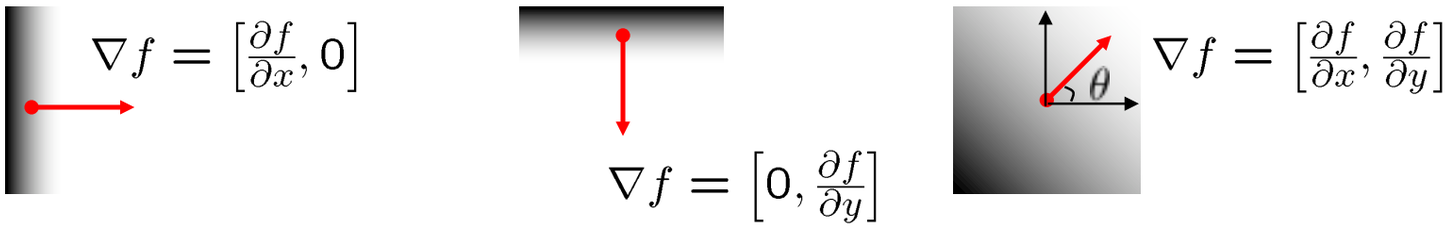
Non-maximum suppression: thinning

Slide credit: S. Seitz

Building an edge detector: Canny

- The Canny Edge detector
  - Filter image with derivative of Gaussian
  - Find magnitude and orientation of gradient
  - Non-maximum suppression
    - Thin wide “ridges” down to single pixel width

Image gradient

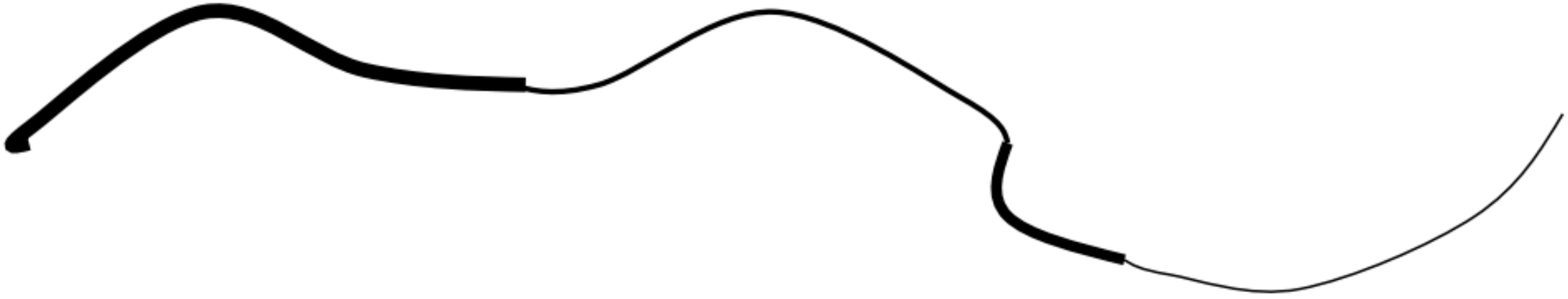


$$\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$$

$$\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

Building an edge detector: Canny

- Hysteresis thresholding: use a high threshold to start edge curves, and a low threshold to continue them



Slide credit: S. Seitz

Building an edge detector: Canny

- Hysteresis thresholding

Original image

high threshold (strong edges)

low threshold (weak edges)

Hysteresis threshold

Slide credit: L. Fei-Fei

Building an edge detector: Canny

- The Canny Edge detector
  - Filter image with derivative of Gaussian
  - Find magnitude and orientation of gradient
  - Non-maximum suppression
    - Thin wide “ridges” down to single pixel width
  - Linking and thresholding (hysteresis)
    - Two thresholds: use the high threshold to start edge curves and the low threshold to continue them

Matlab: `edge(image, 'canny');`

J. Canny, “ A Computational Approach To Edge Detection”, TPAMI 1986

Recall: a *(rough)* computer vision timeline

1966: Marvin Minsky assigns computer vision as an undergrad summer project

1970s: interpretation of synthetic worlds and carefully selected images

1980s: shift towards geometry and increased mathematical rigor

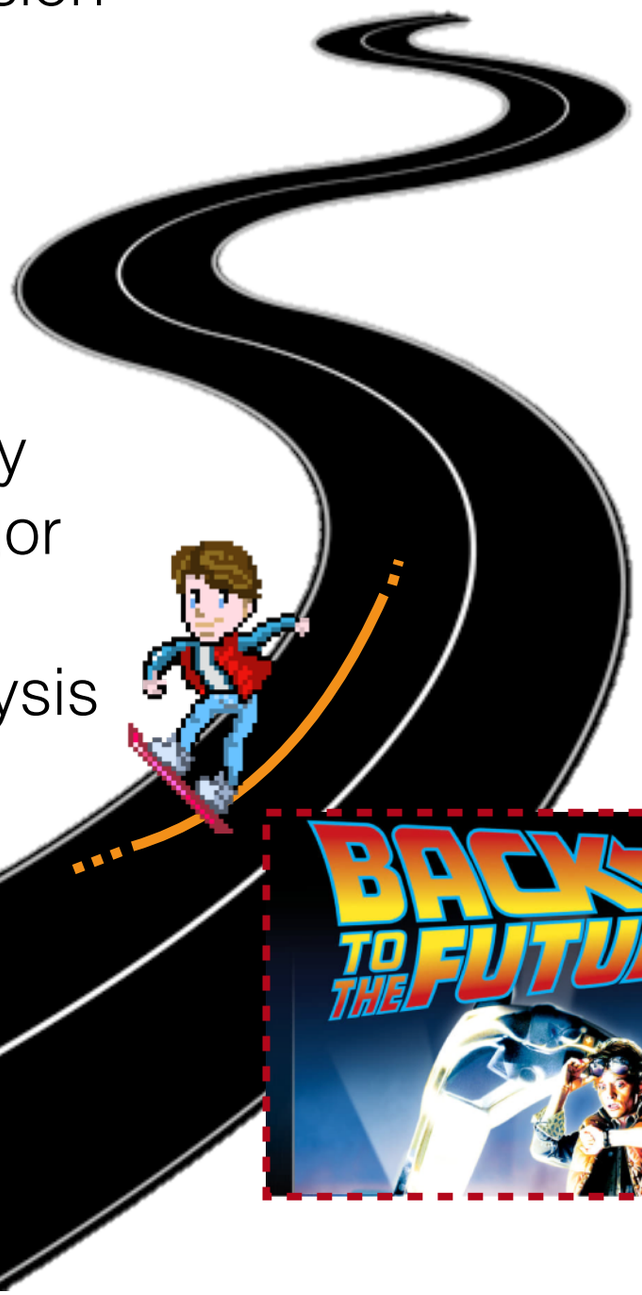
1990s: face recognition, statistical analysis

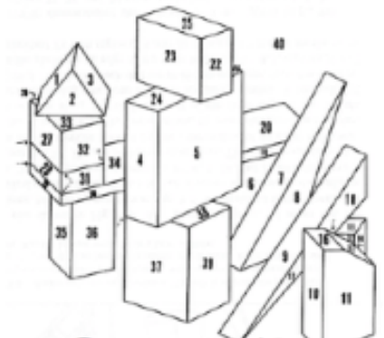
2000s: object recognition, categorization, annotated datasets available

2010s: large-scale visual recognition, visual intelligence

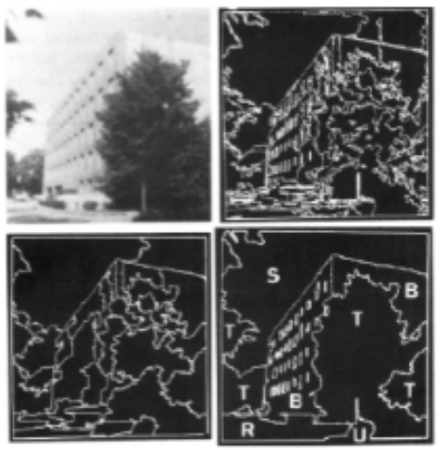
2020s: ???

(big/"foundation" models, …)





Guzman '68



Ohta Kanade '78

Turk and Pentland '91

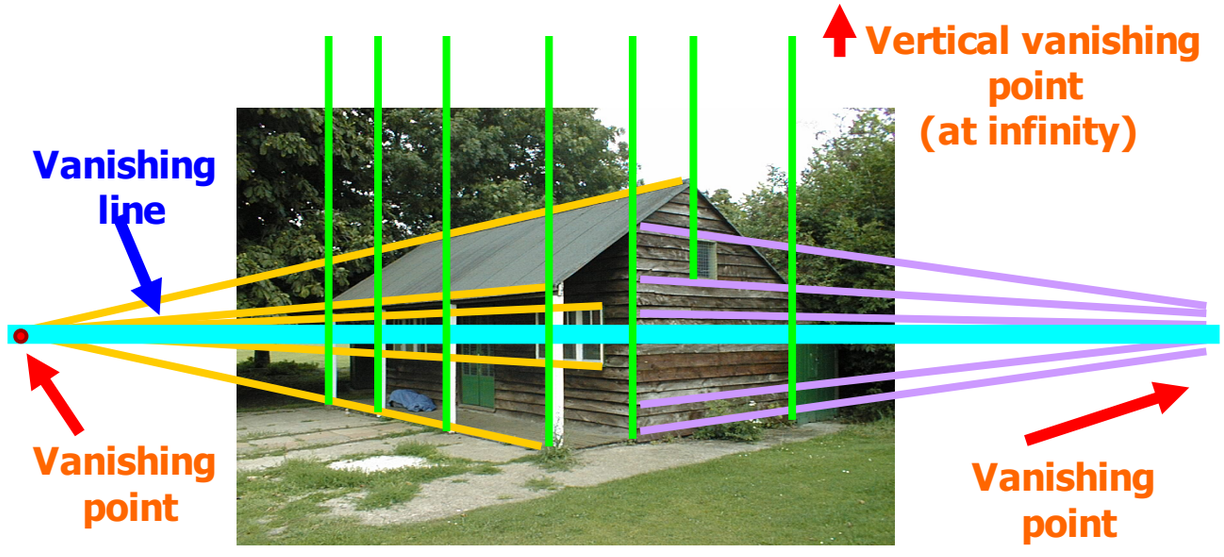
Adapted from slide by O. Russakovsky

Recall: why do we care about edges?

- Extract information, recognize objects

Edges are great, but they are not very robust to a number of transformations…

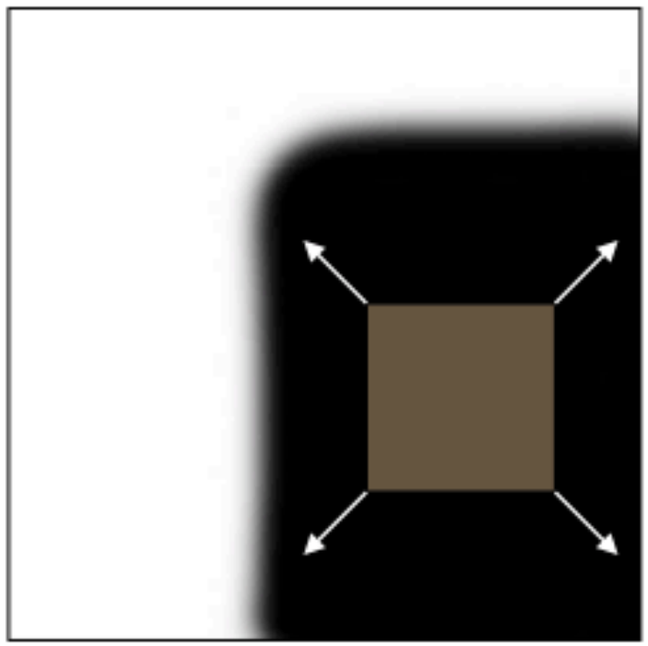
- Recover geometry and viewpoint



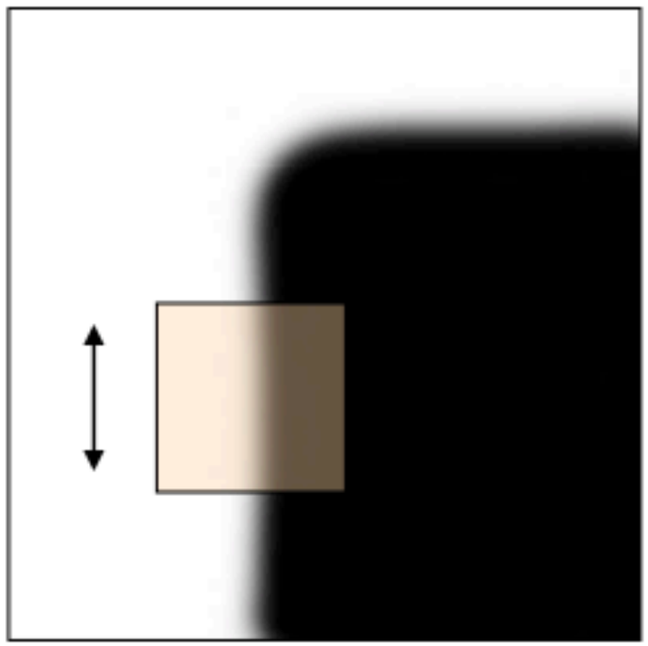
Slide credit: J. Hayes

Edges, corners and blobs (keypoints)

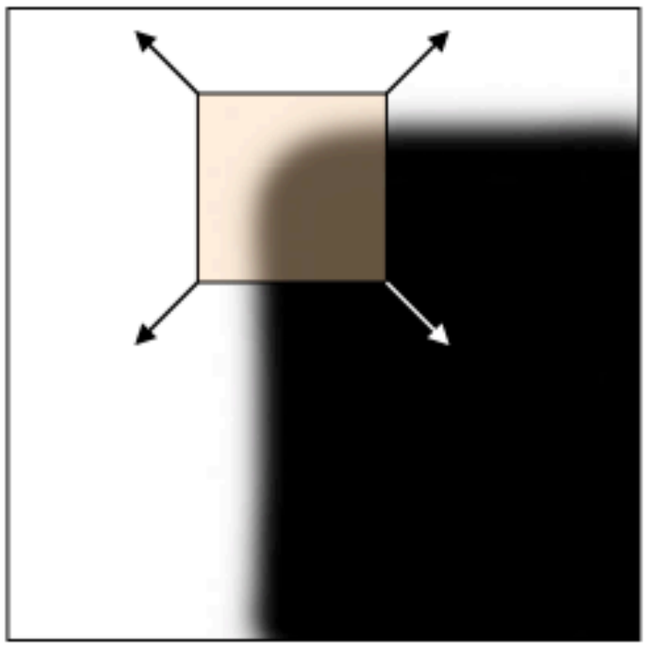
- Low-level image features: a basic classification



“flat” region:
no change in all directions



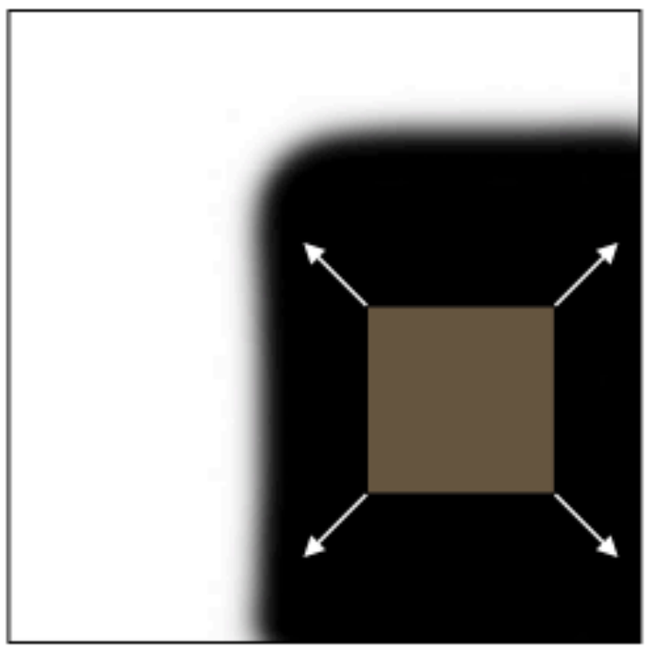
“edge”:
no change along the edge direction



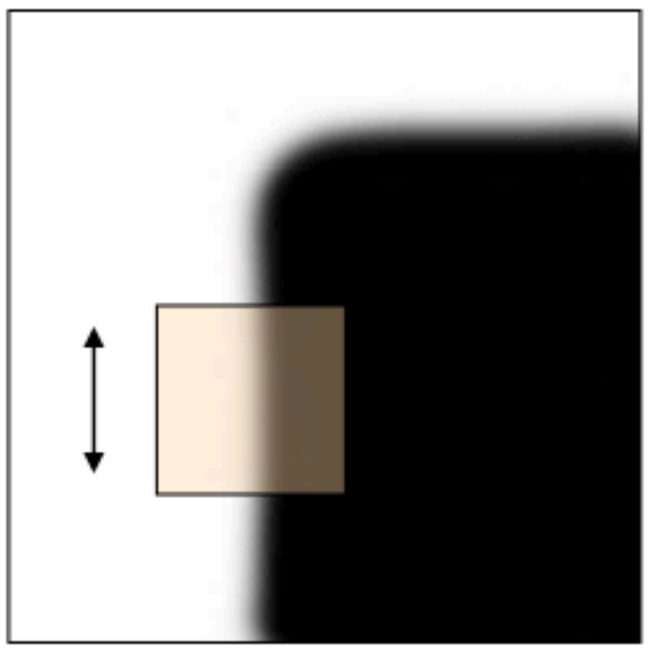
“corner”:
significant change in all directions

Corner detection: intuition

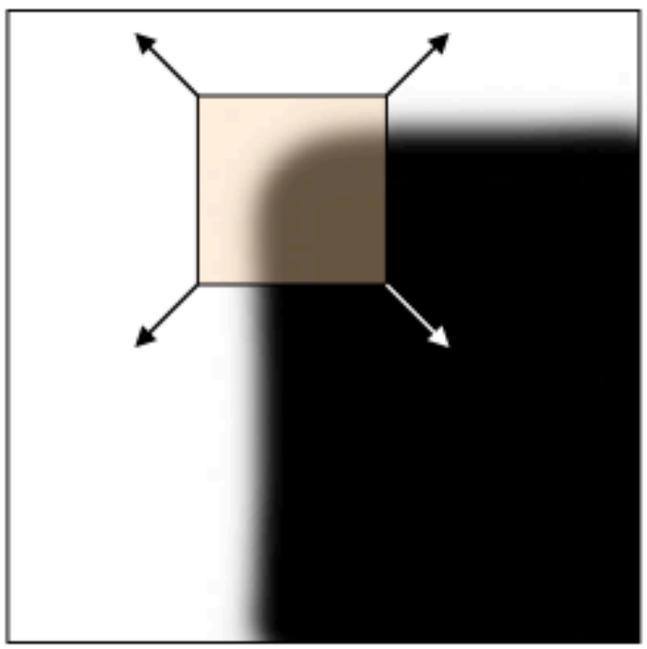
- We should easily recognize the point by looking through a small window (locally)
  - Shifting a window in any direction should give a large change in intensity



"flat" region: no change in all directions



"edge": no change along the edge direction

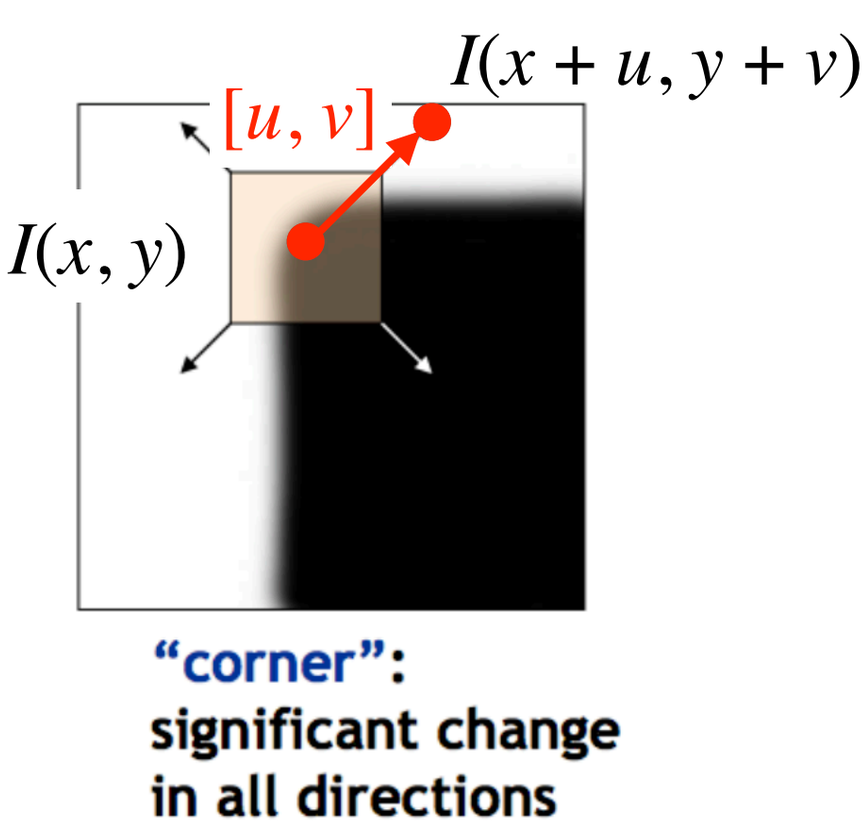


"corner": significant change in all directions

Corner detection: basic idea

- Given an image I, captures change in appearance of window W for the shift $[u, v]$:

$$E(u, v) = \sum_{(x,y) \in W} [I(x + u, y + v) - I(x, y)]^2$$



It measures the intensity change at the center pixel (x, y) when we shift by $[u, v]$

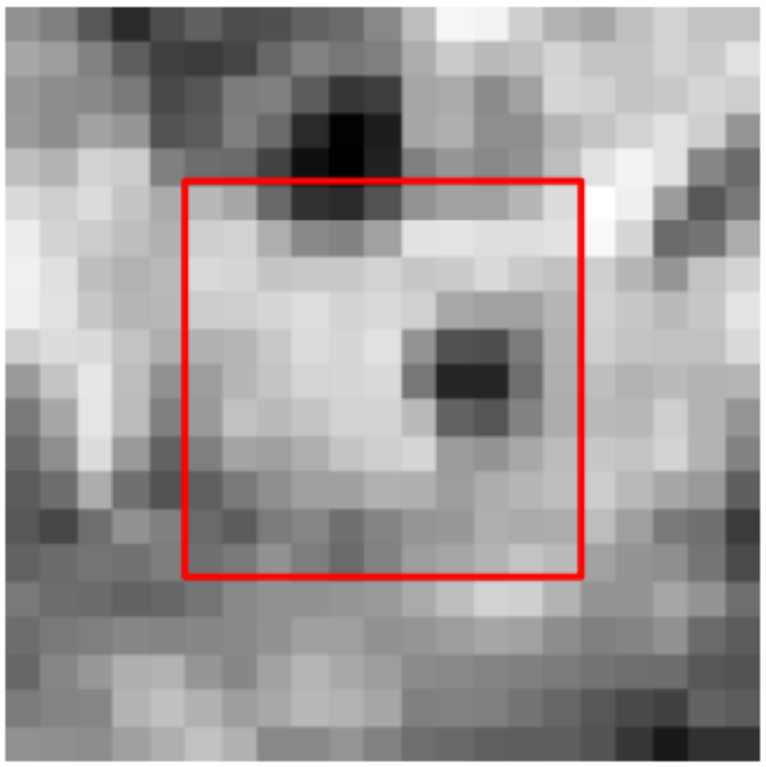
That's for a single point, $E(u, v)$ accumulates over the patch (small window) around the center pixel

Corner detection: basic idea

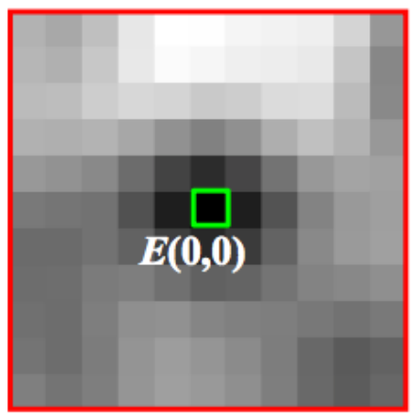
- Given an image I, captures change in appearance of window W for the shift $[u, v]$:

$$E(u, v) = \sum_{(x,y) \in W} [\, I(x + u, y + v) - I(x, y) \,]^2$$

$I(x, y)$



$E(u, v)$



Corner detection: basic idea

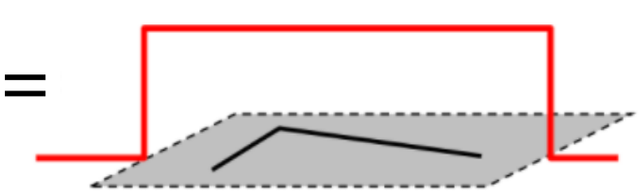
- Given an image I, captures change in appearance of window W for the shift $[u, v]$:

$$E(u,v) = \sum_{(x,y)} W(x,y) \, [\, I(x+u, y+v) - I(x,y) \,]^2$$

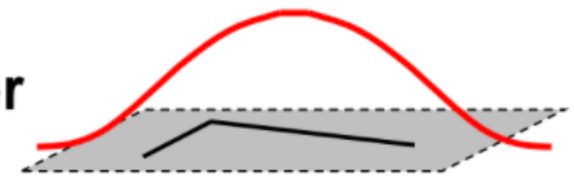
Window function — **Intensity change**

Window function

$$W(x,y) =$$



or

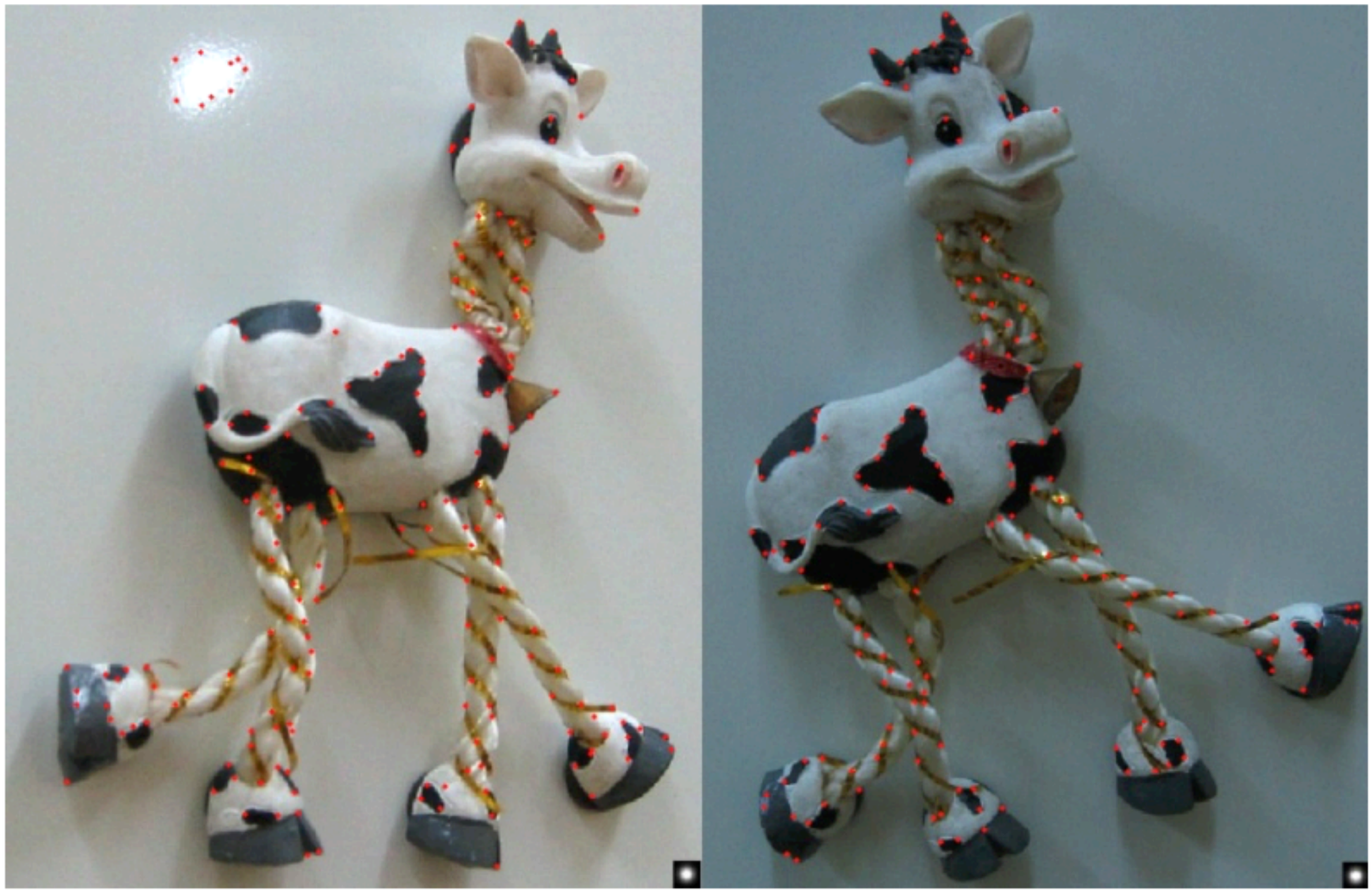


1 in window, 0 outside

Gaussian

Corners and keypoints

- Corners (e.g. Harris corner detector) are a good example of keypoints / interest points



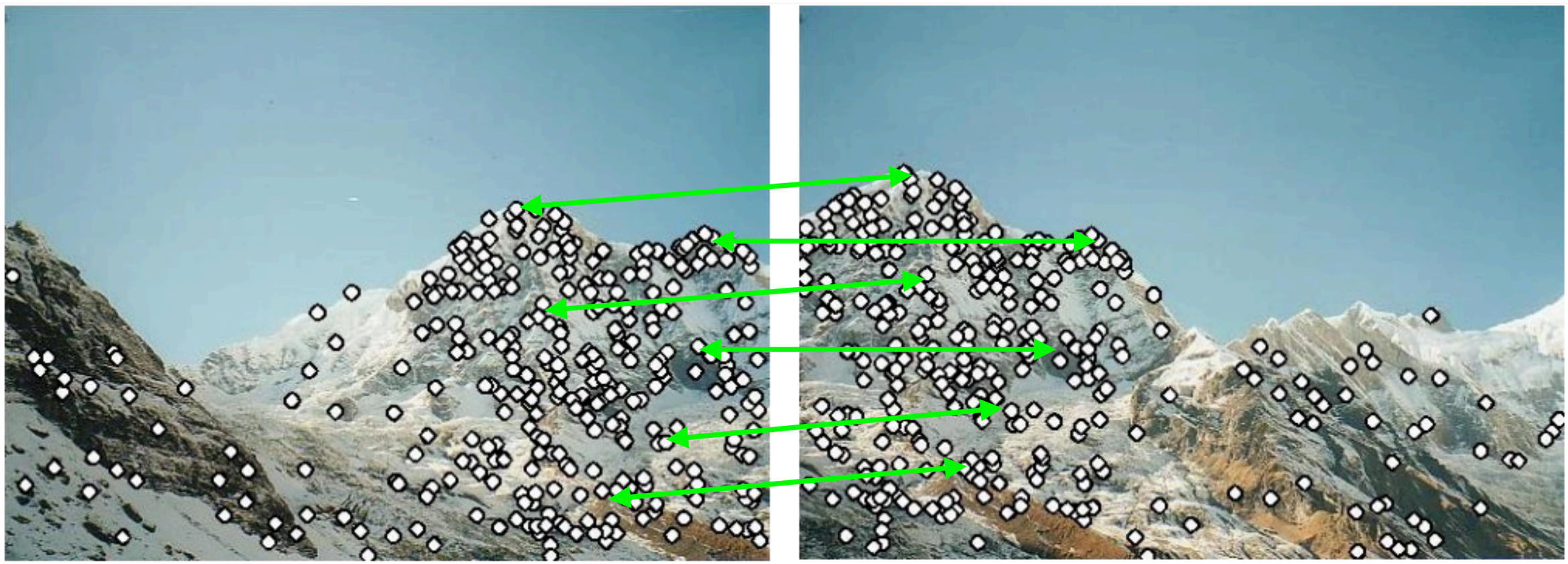
Harris corners example

Why extract keypoints from an image?

- Motivation: a key example is panorama stitching
  - i.e. we have two images, how to combine them?

Why extract keypoints from an image?

- Motivation: a key example is panorama stitching
  - i.e. we have two images, how to combine them?



Step 1: extract keypoints

Step 2: match keypoint features

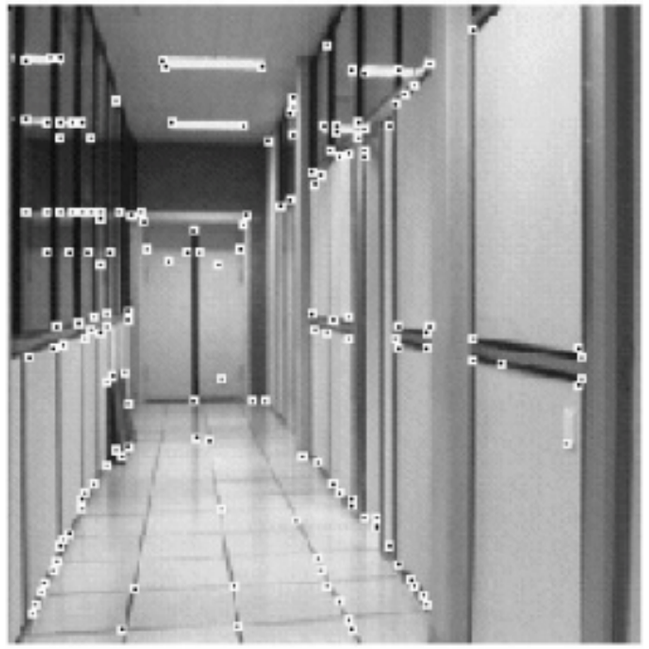
Why extract keypoints from an image?

- Motivation: a key example is panorama stitching
  - i.e. we have two images, how to combine them?

Step 1: extract keypoints **Step 2:** match keypoint features **Step 3:** align images

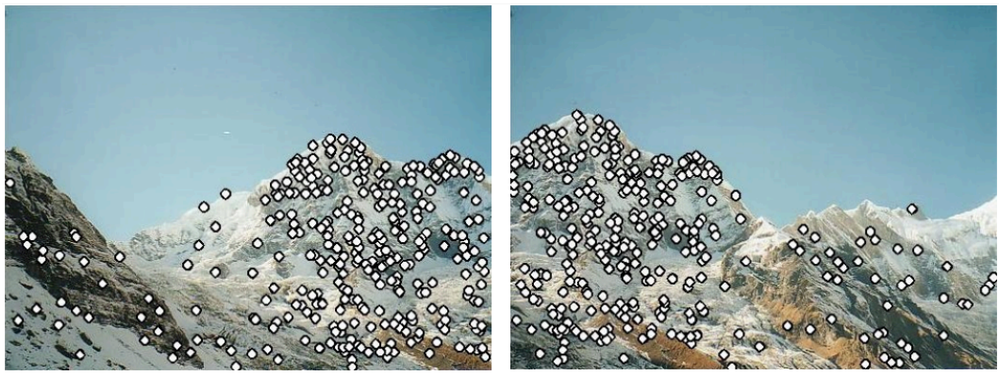
Keypoints: applications

- Keypoints are used for:
  - Image alignment
  - 3D reconstruction
  - Motion tracking
  - Robot navigation
  - Indexing and database retrieval
  - Object recognition



Characteristics of Good Keypoints

- Repeatability
  - Can be found despite geometric and photometric transformations



- Salience
  - Each keypoint is distinctive

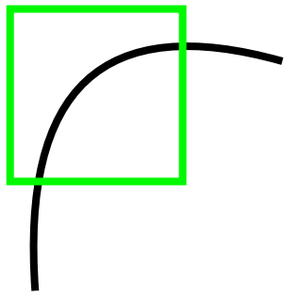
- Compactness and efficiency
  - Many fewer keypoints than image pixels

- Locality
  - Occupies small area of the image: therefore more robust to clutter and occlusion

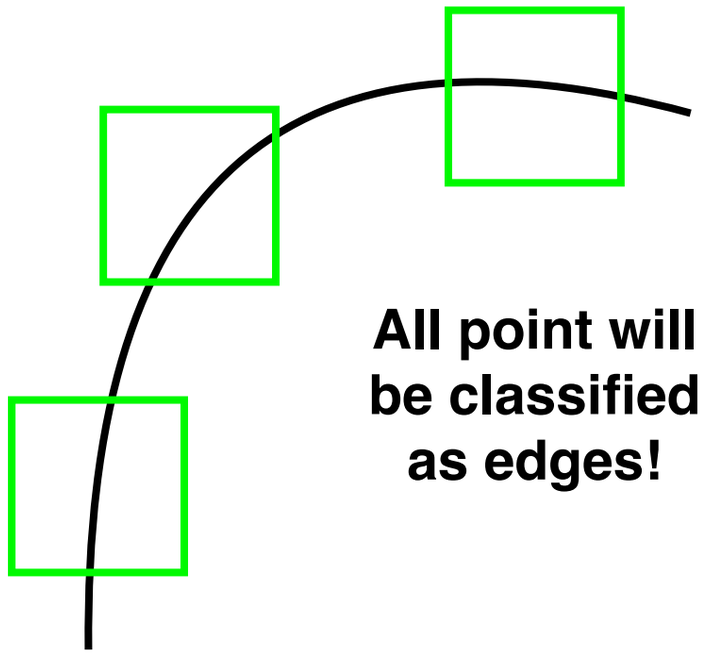
Adapted from slide by O. Russakovsky

An example: Harris corner detector

- Properties (i.e. geometric invariances):
  - Translation invariance? **Yes**
  - Rotation invariance? **Yes**
  - Scale invariance? **No**



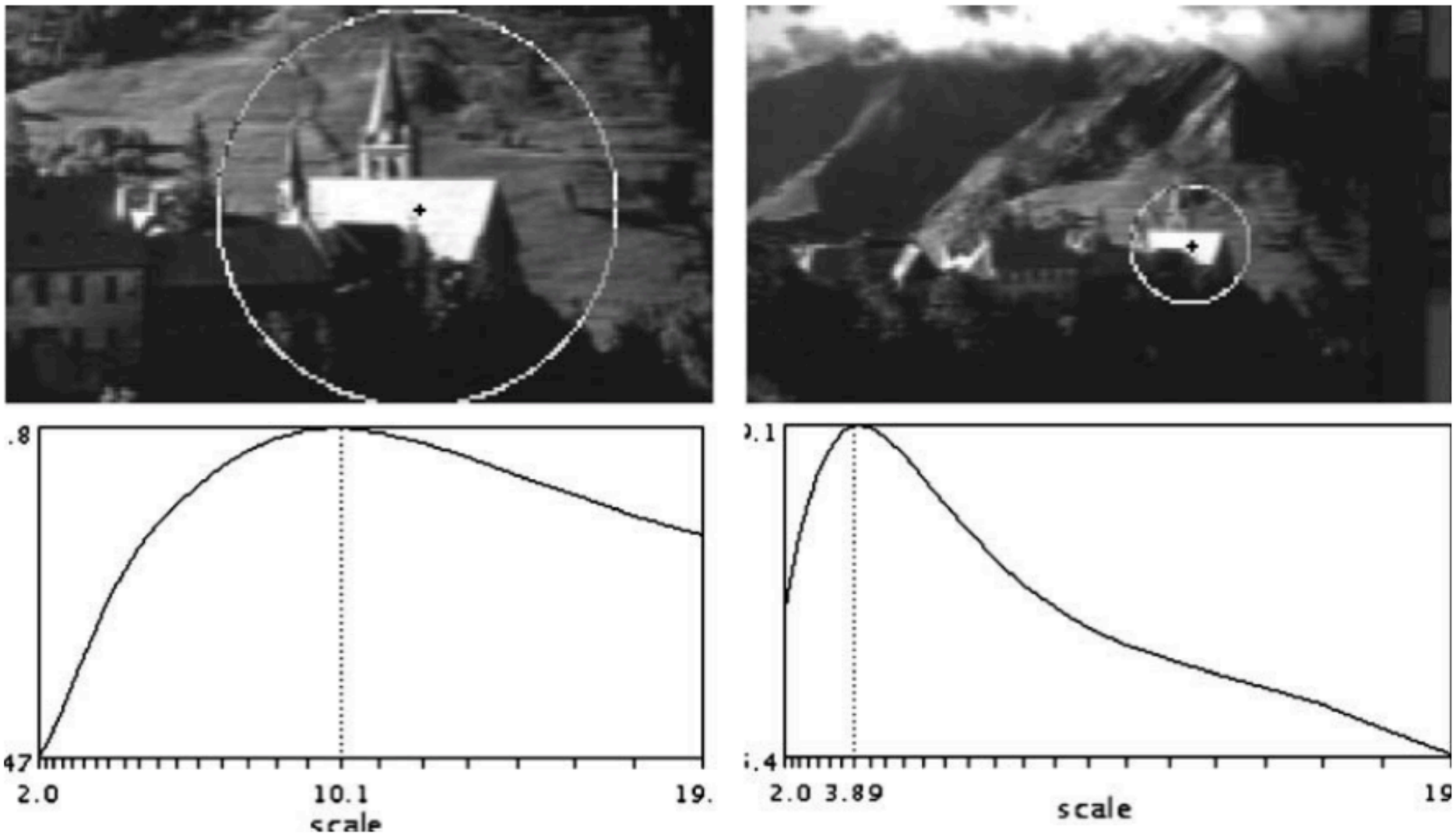
Corner



All point will be classified as edges!

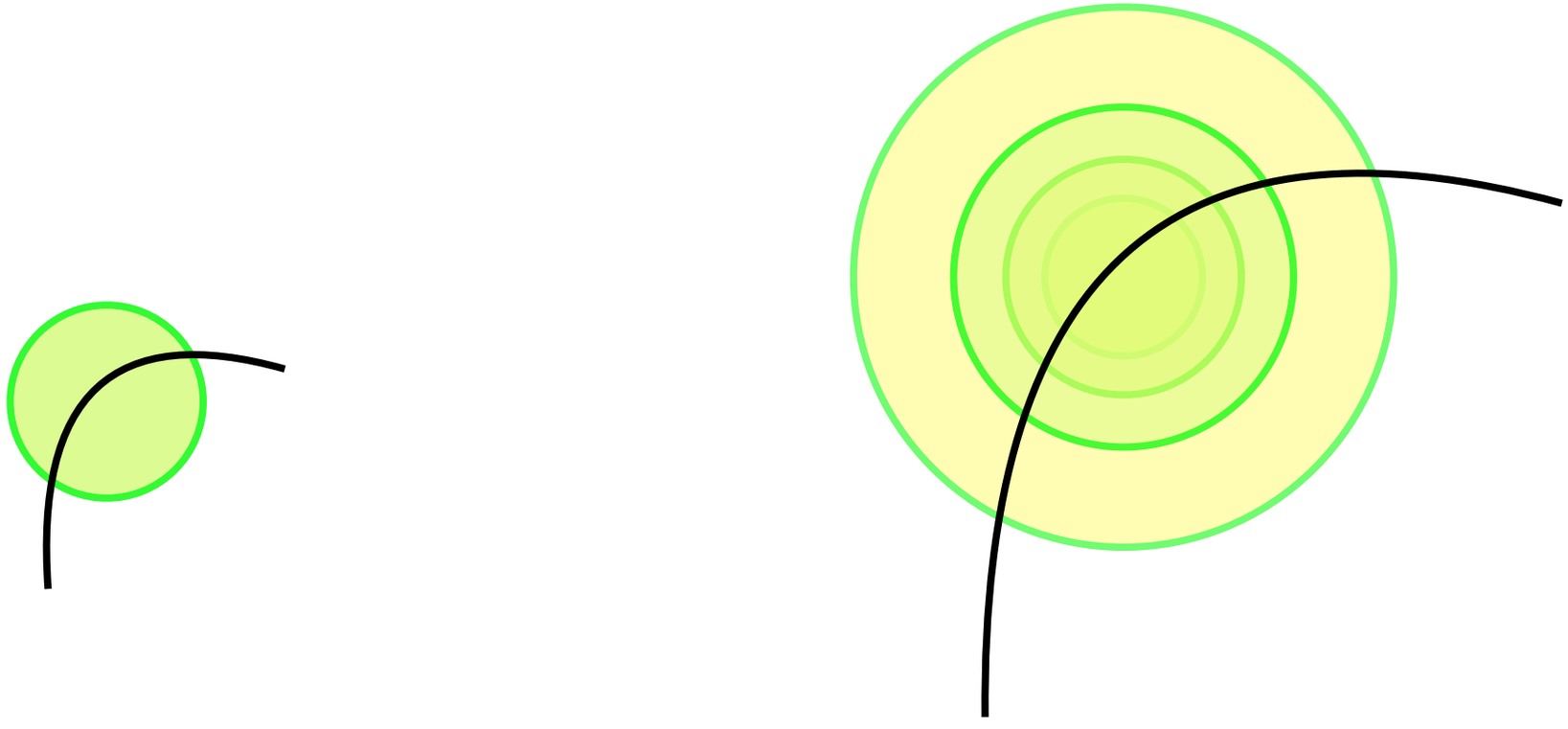
Feature detection with scale selection

- We want to extract features with characteristic scale that is invariant (*covariant*) with the image transformation



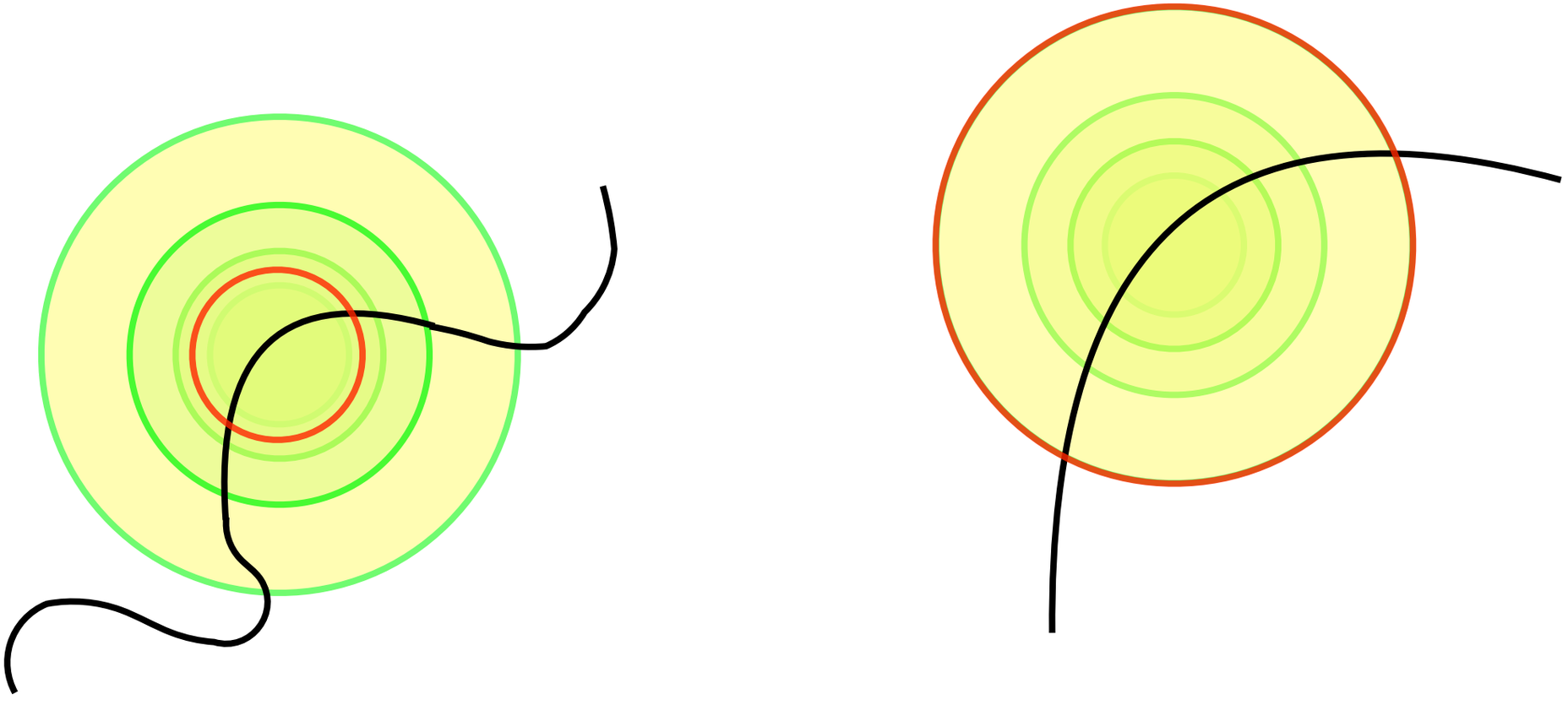
Scale invariant detection

- Consider regions (e.g. circles) of different sizes around a point
- How to find regions of corresponding sizes that will look the same in both images?



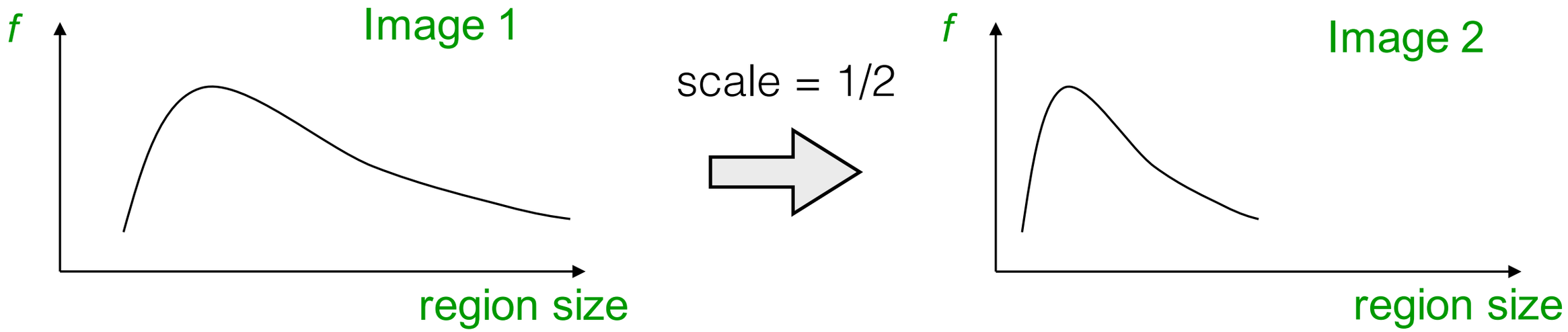
Scale invariant detection

- The problem: how do we choose corresponding circles **independently** in each image?



Scale invariant detection

- Solution: design a function on the circular region which is scale invariant *
  - e.g. average intensity (same even for different sizes)
- For a point in one image, we can consider it as a function of region size (circle radius)

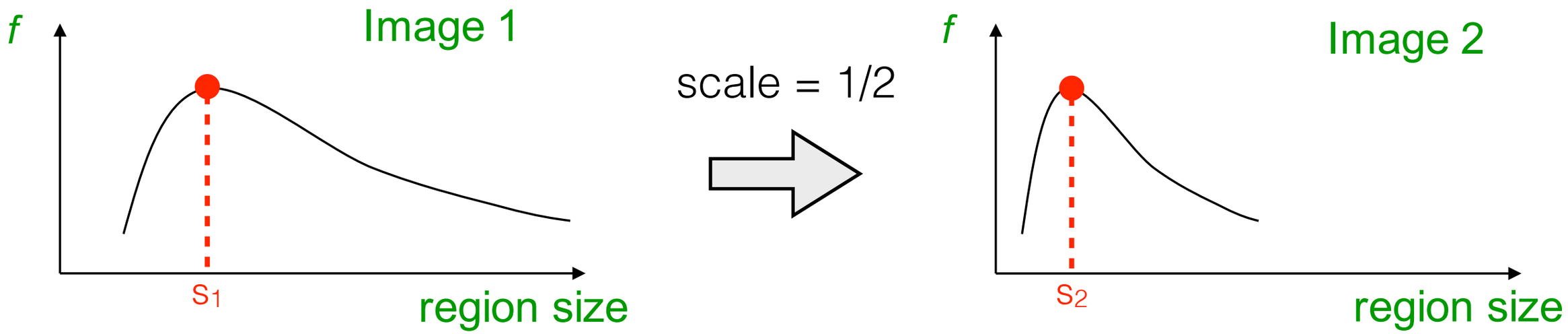


* *i.e. the same for corresponding regions, even if they are at different scales*

Scale invariant detection

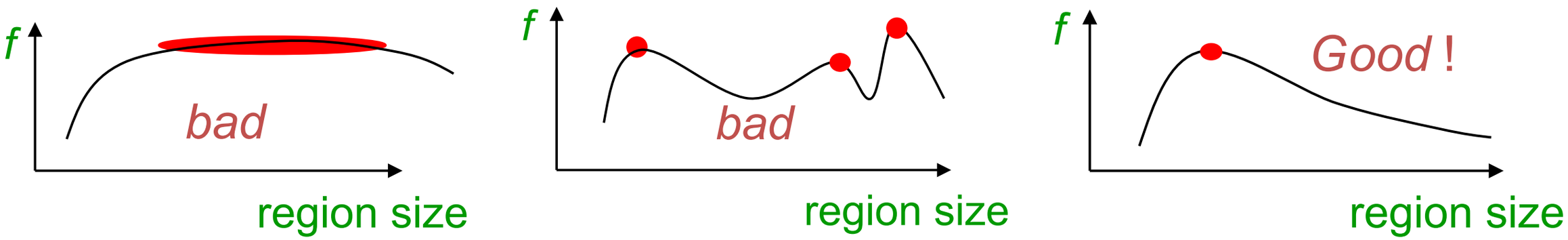
- Common approach: take a local max of this function
- Observation: region size for which the maximum is achieved should be *invariant* (*co-variant*) with scale

Important: this scale invariant region size is found in each image **independently**!



Scale invariant detection

- A good function for scale detection has one stable sharp peak



- So a good function would be one which responds to contrast (sharp local intensity change)

Contact

- **Office:** Torre Archimede, room 6CD3
- **Office hours** (ricevimento)**:** Friday 9:00-11:00

✉ lamberto.ballan@unipd.it

🏠 http://www.lambertoballan.net

🏠 http://vimp.math.unipd.it

@ twitter.com/lambertoballan