Cognitive Services

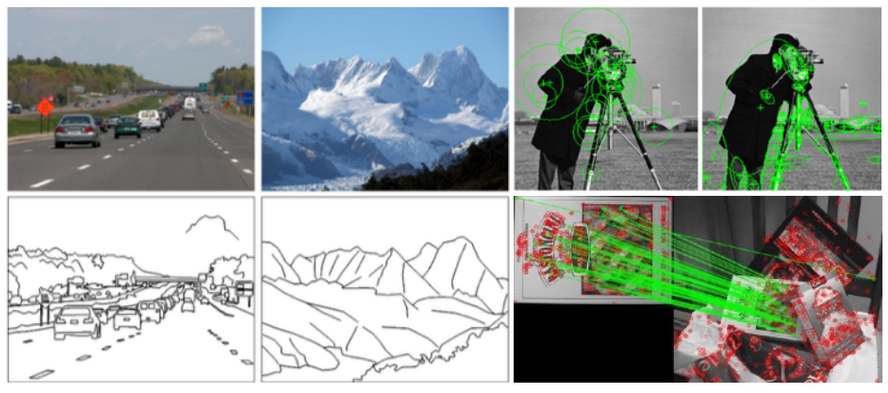
LM Informatica & Data Science - 2nd semester - 6 CFU

Foundations: edges, local invariant features, descriptors

Lamberto Ballan

What we will learn today?

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



Background reading:

Forsyth and Ponce, Computer Vision, Chapter 8

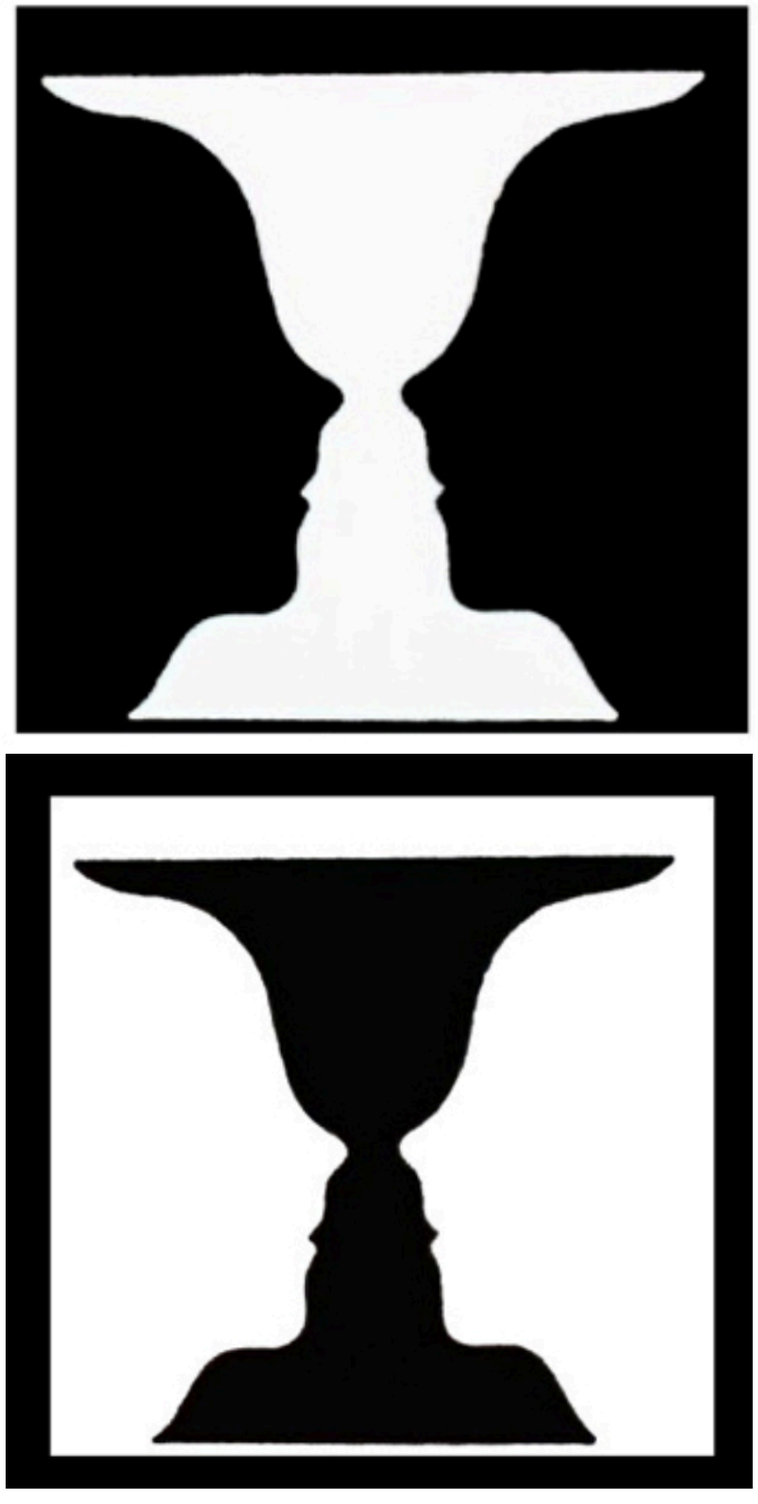
D.G. Lowe, SIFT paper, IJCV 2014

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

What’s an edge?

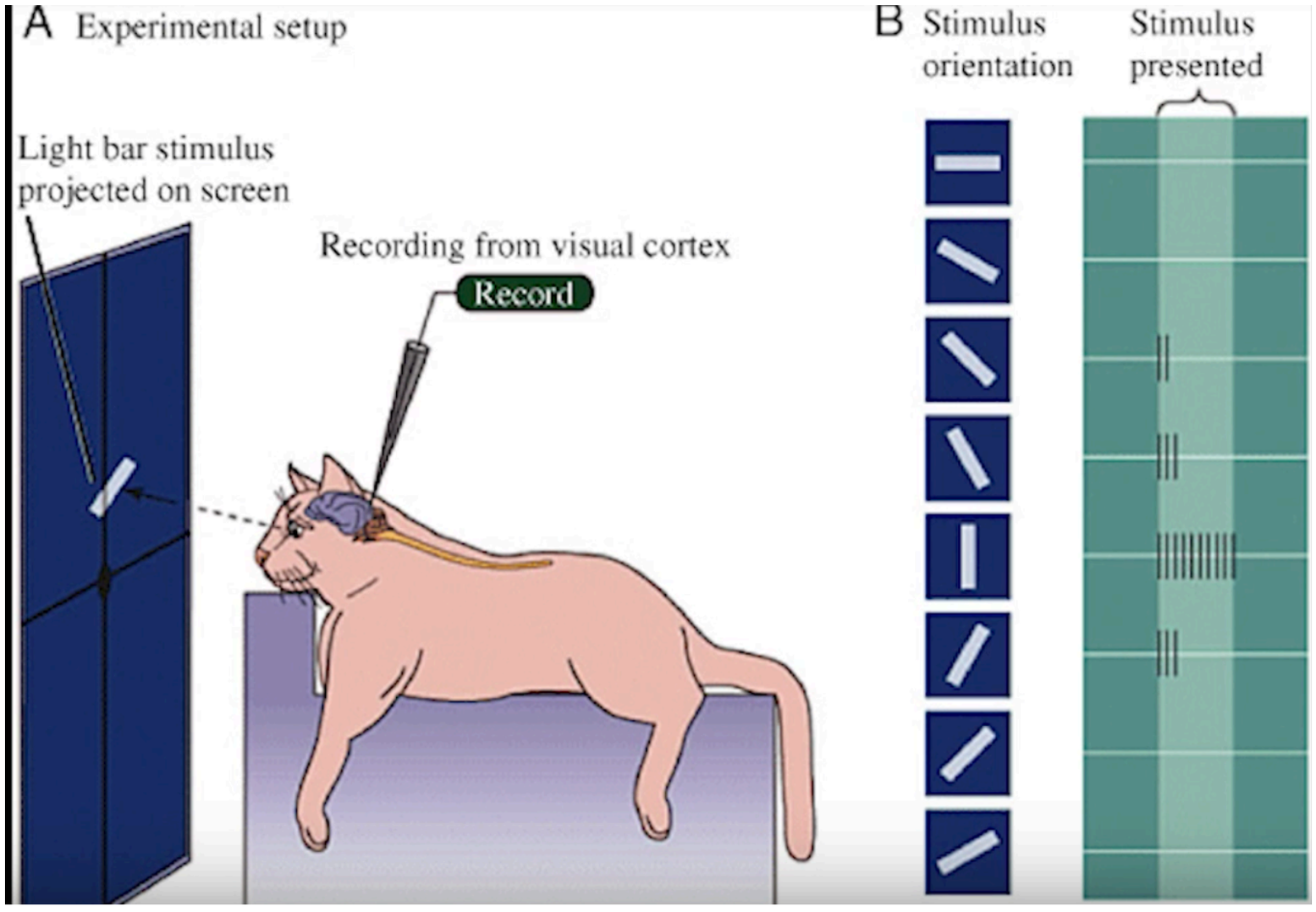


Cave painting (Chauvet, France, ~30,000 BC)

Shen Zhou’s “Poet on a mountain top” (China, ~1,500 AD)

What's an edge?

- We know edges are special from human (mammalian) vision studies

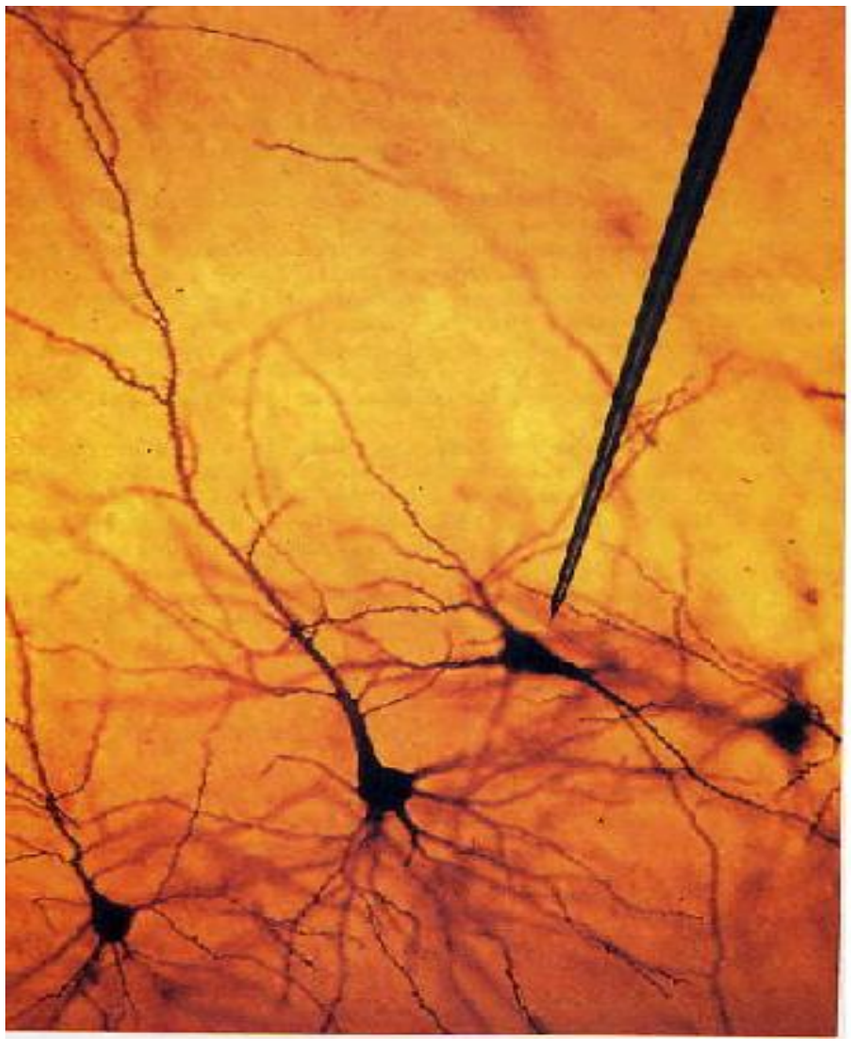


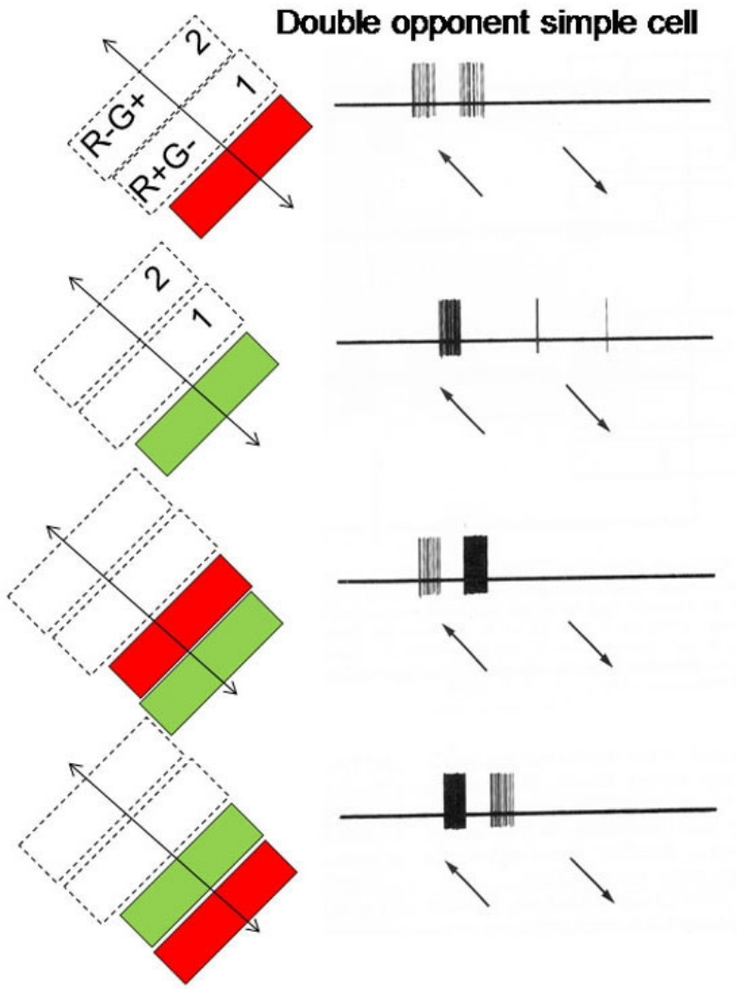
Hubel and Wiesel, 1960s

What's an edge?

- We know edges are special from human (mammalian) vision studies

Hubel and Wiesel, 1960s





https://www.youtube.com/watch?v=8VdFf3egwfg&t=18s

Edge detection

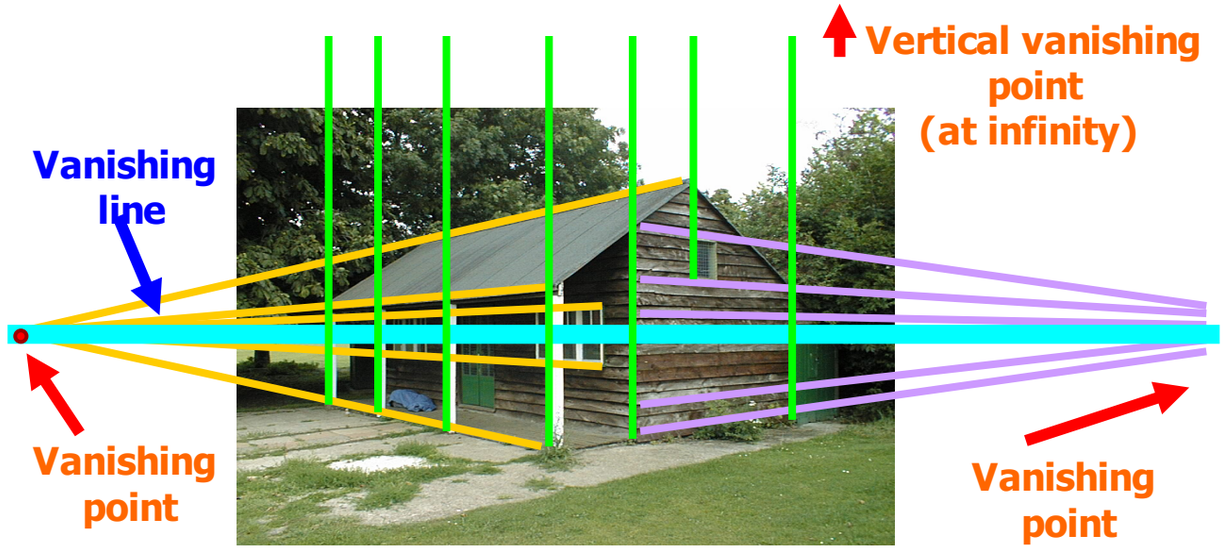
- **Goal:** Identify sudden changes (discontinuities) in an image
  - Intuitively, most semantic and shape information from the image can be encoded in the edges
  - More compact than pixels

- **Ideal:** artist's line drawing (but artist is also using object-level knowledge)

Why do we care about edges?

- Extract information, recognize objects

- Recover geometry and viewpoint

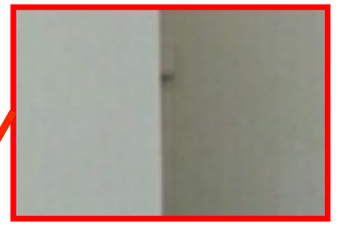


Slide credit: J. Hayes

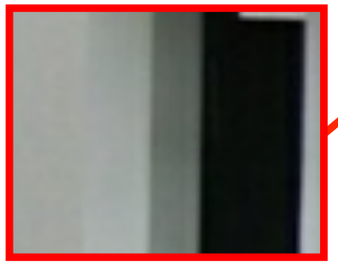
What causes an edge?

Illumination discontinuity:
cast shadows

Depth discontinuity:
object boundary



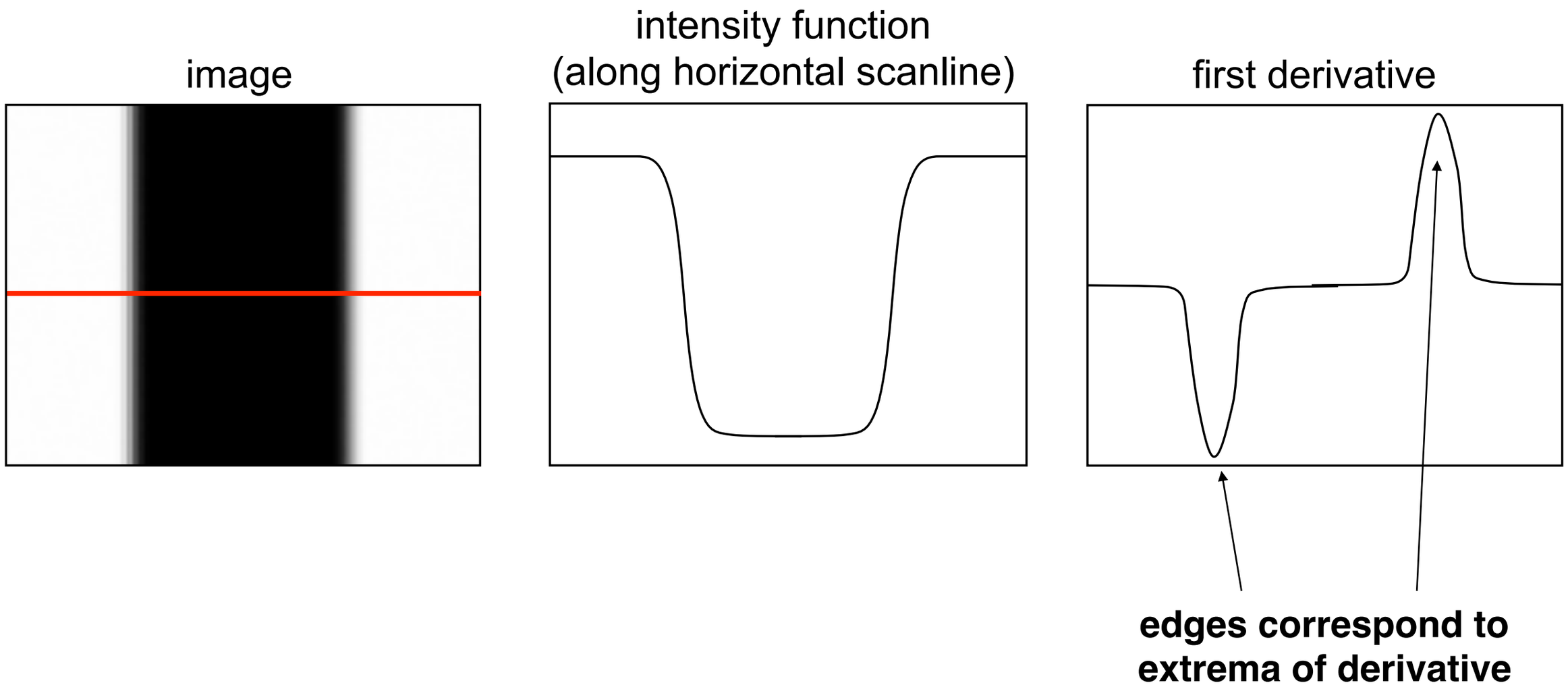
Change in surface
orientation: shape



Surface color
discontinuity

Edges and image gradients

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



Slide credit: S. Lazebnik

Derivatives: a quick recap

- Derivative in 1D:

$$\frac{df}{dx} = \lim_{\Delta x \to 0} \frac{f(x) - f(x - \Delta x)}{\Delta x} = f'(x) = f_x$$

A simple example:

$$y = x^2 + x^4$$

$$\frac{dy}{dx} = 2x + 4x^3$$

- Discrete derivative in 1D:

$$\frac{df}{dx} = \lim_{\Delta x \to 0} \frac{f(x) - f(x - \Delta x)}{\Delta x} = f'(x)$$

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

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

we can approximate using finite differences

Derivatives: a quick recap

- Types of Discrete derivative in 1D:

Filter

  - Backward: $\frac{df}{dx} = f(x) - f(x-1) = f'(x)$ `[0 1 -1]`
  - Forward: $\frac{df}{dx} = f(x) - f(x+1) = f'(x)$ `[-1 1 0]`
  - Central: $\frac{df}{dx} = f(x+1) - f(x-1) = f'(x)$ `[1 0 -1]`

Derivatives: a quick recap

- 1D discrete derivate example:

$$f(x) = 10 \quad 15 \quad 10 \quad 10 \quad 25 \quad 20 \quad 20 \quad 20$$

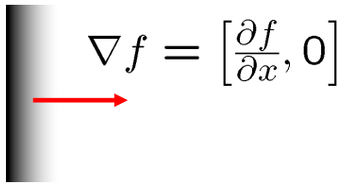
$$f'(x) = \quad 0 \quad 5 \quad -5 \quad 0 \quad 15 \quad -5 \quad 0 \quad 0$$

2D discrete derivative: image gradient

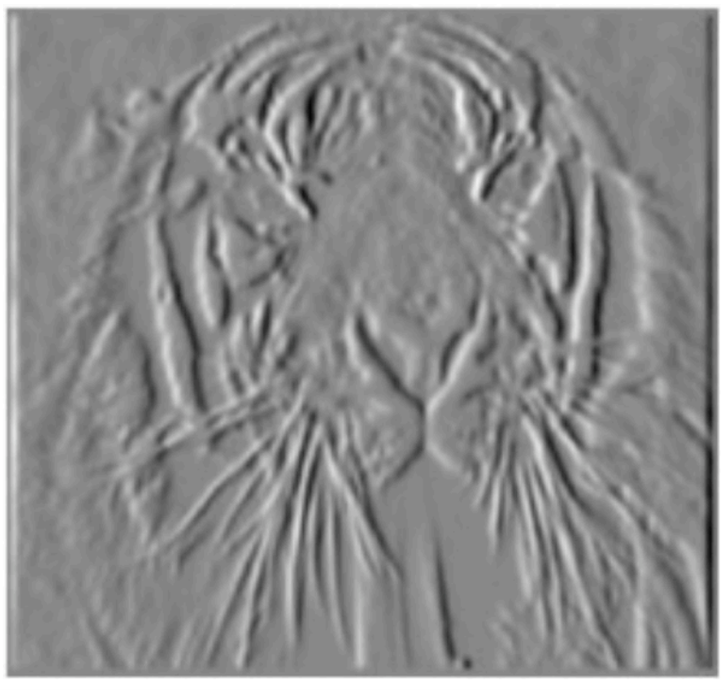
Given function $f(x, y)$

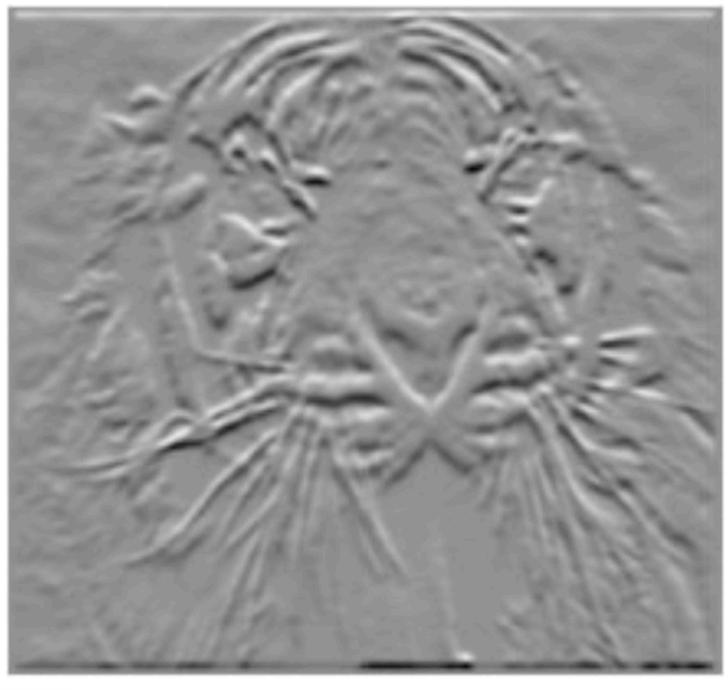
Gradient vector $$\nabla f(x, y) = \begin{bmatrix} \frac{\partial f(x, y)}{\partial x} \\ \frac{\partial f(x, y)}{\partial y} \end{bmatrix} = \begin{bmatrix} f_x \\ f_y \end{bmatrix}$$

Which one is the gradient in the x-direction? How about y-direction?



x-direction





y-direction

Slide credit: S. Seitz

2D discrete derivate example

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

Filter

What happens when we apply this filter?

2D discrete derivate example

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

$$= \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

2D discrete derivate example

$$I = \begin{bmatrix} 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \end{bmatrix} * \underbrace{\frac{1}{3}\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}}_{\textit{Filter}} = ?$$

What happens when we apply this filter?

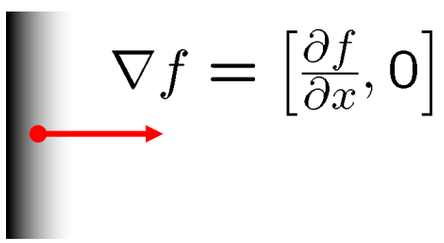
2D discrete derivate example

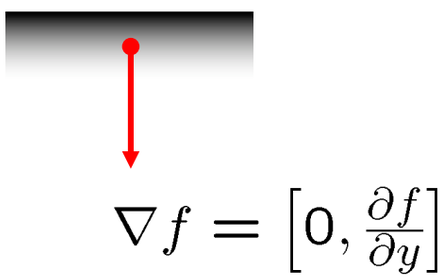
$$I = \begin{bmatrix} 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \end{bmatrix} * \underbrace{\frac{1}{3}\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}}_{\textit{Filter}} =$$

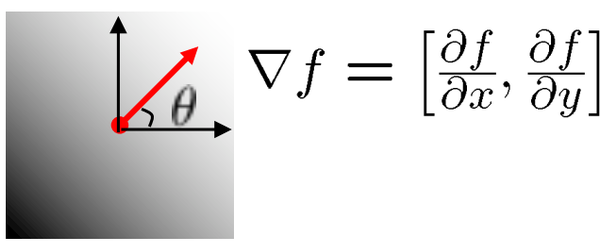
$$= \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 10 & 10 & 0 & 0 \\ 0 & 10 & 10 & 0 & 0 \\ 0 & 10 & 10 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Image gradient

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





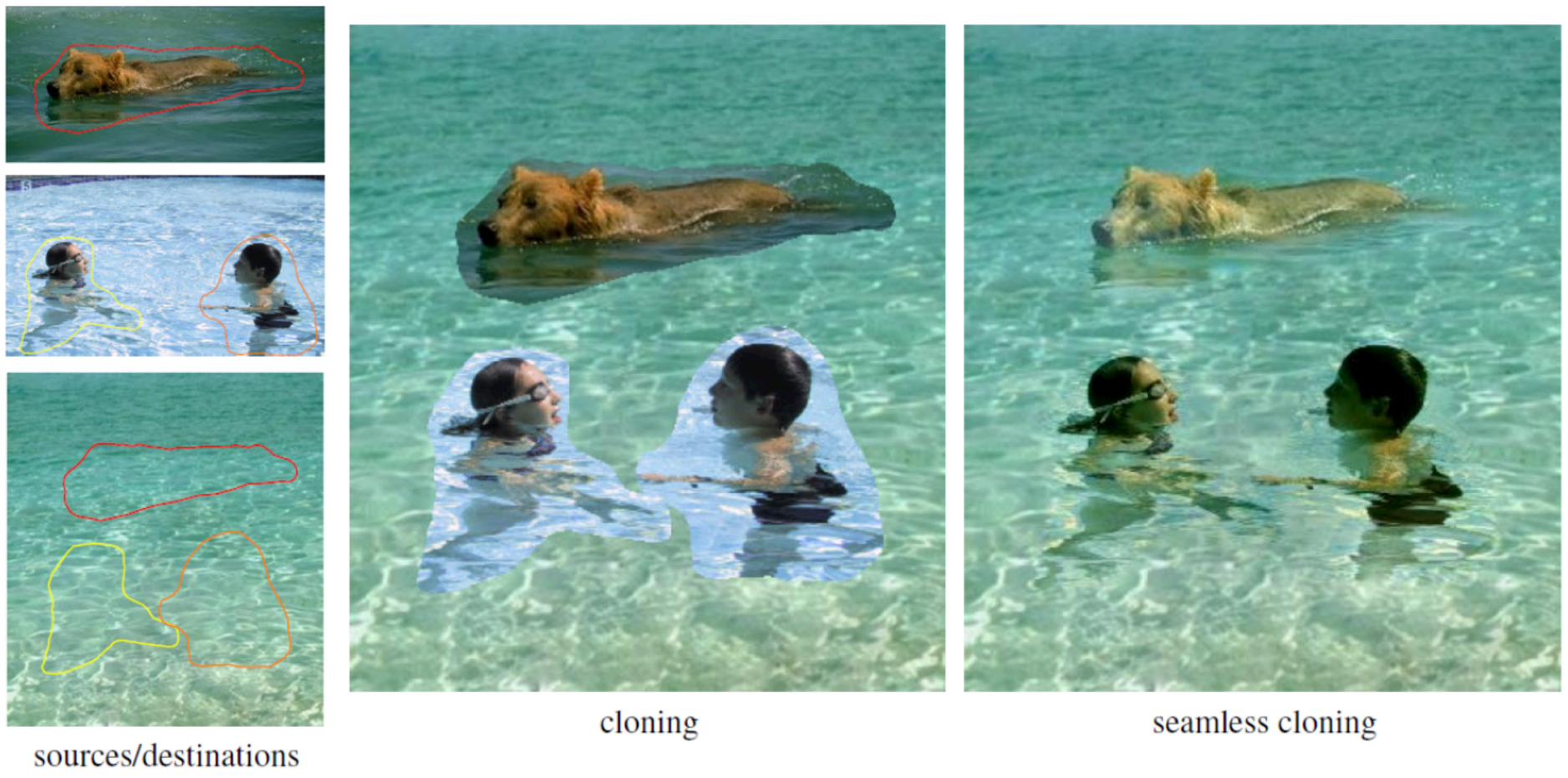


  - The gradient vector points in the direction of most rapid increase in intensity
  - The gradient direction is given by: $\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$
  - The *edge strength* is given by the gradient magnitude:

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

Applications: an example

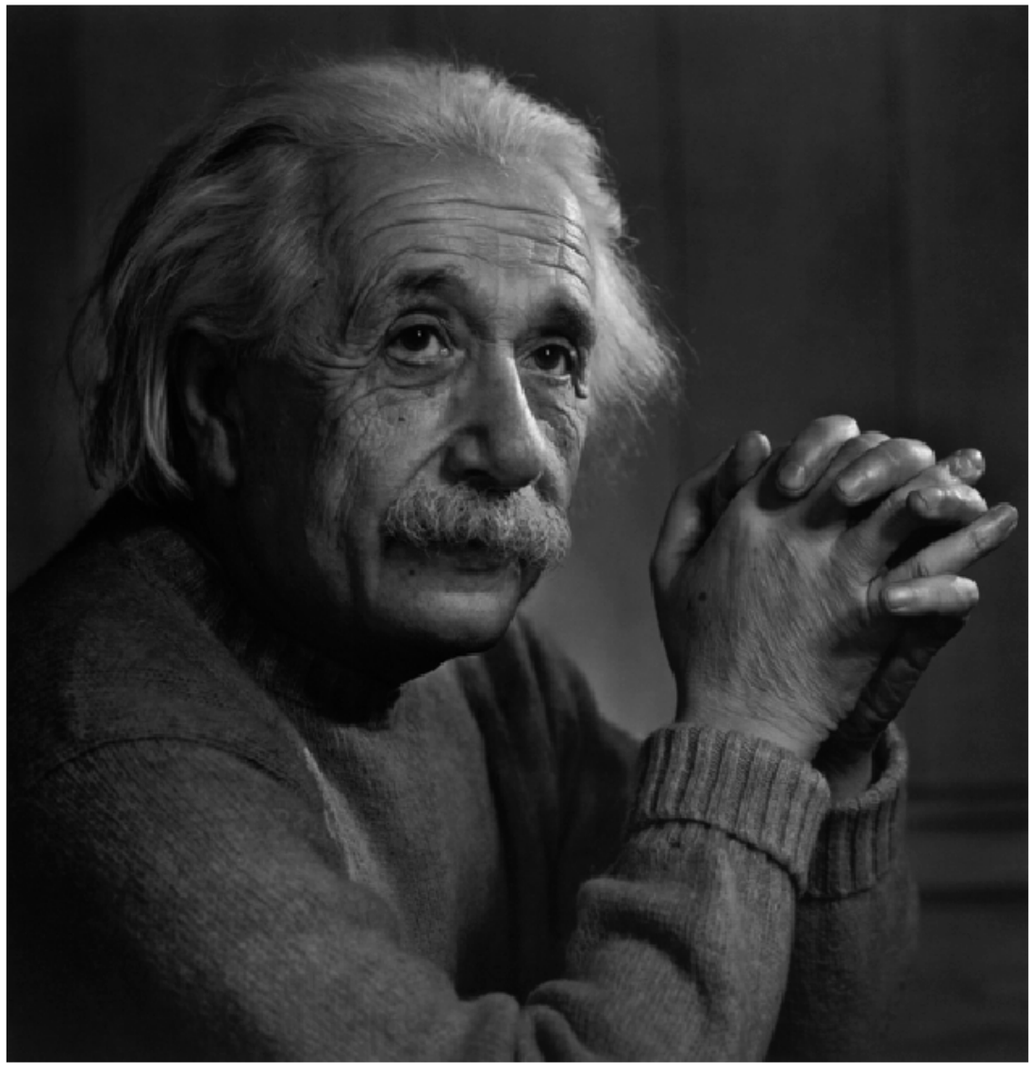
- Gradient-domain image editing
  - solve for pixel values in the target region to match gradients of the source region while keeping background pixels the same

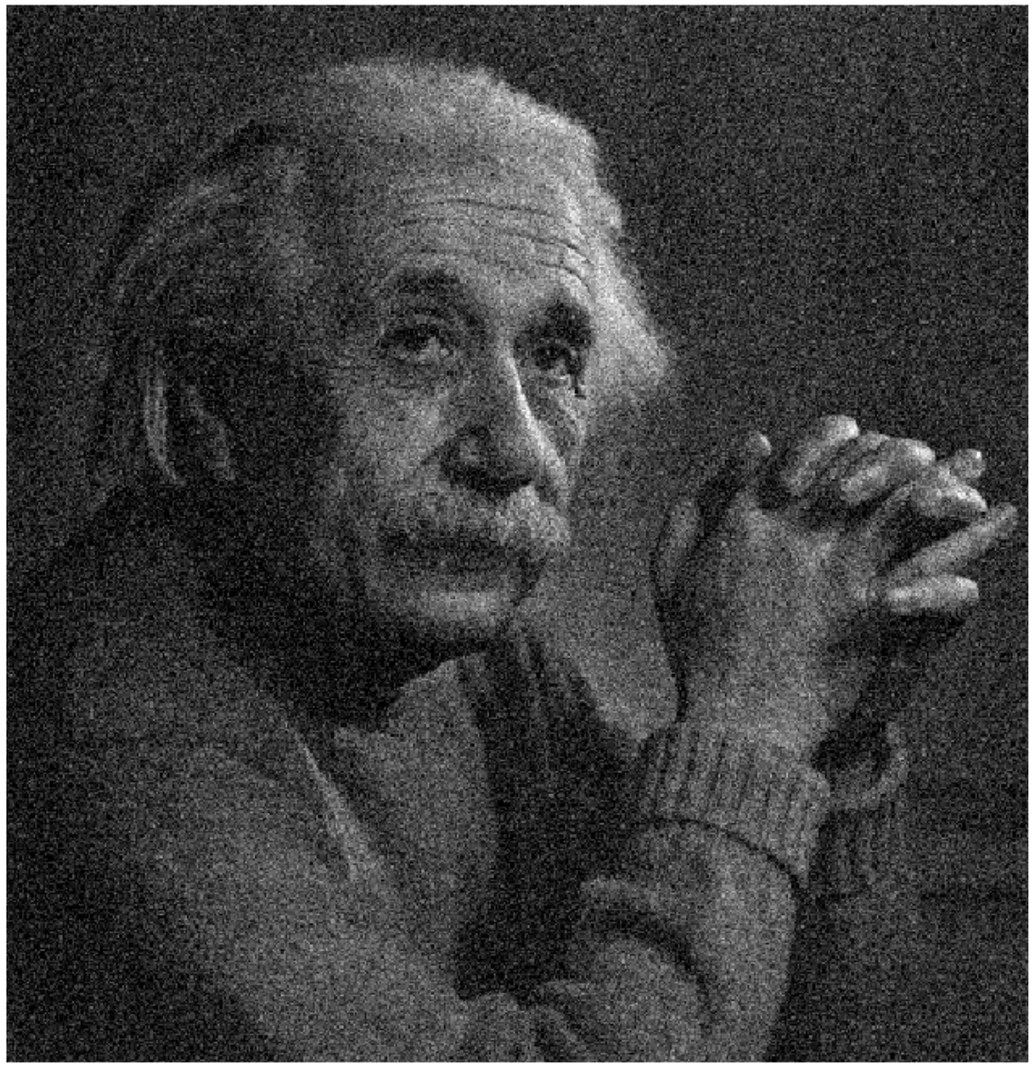


P. Perez, M. Gangnet, A. Blake, Poisson Image Editing, SIGGRAPH 2003

Slide credit: S. Lazebnik

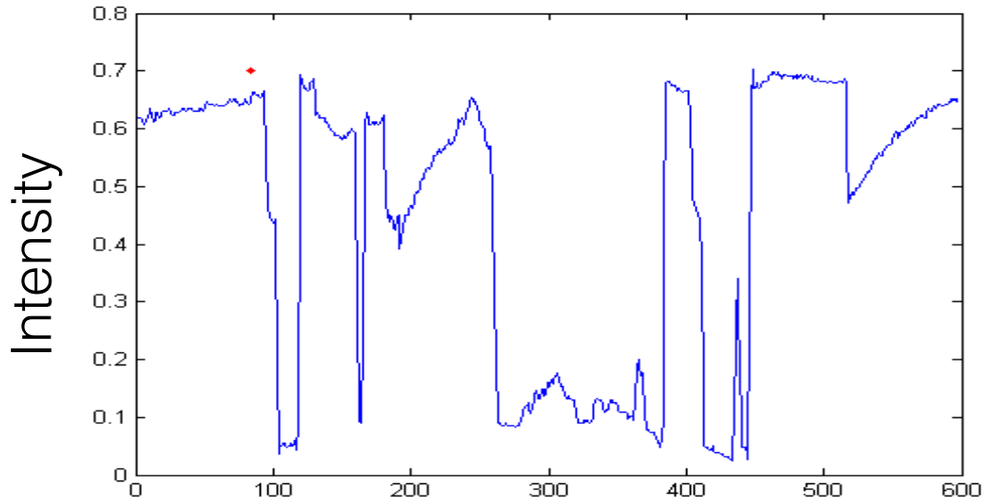
Effects of noise

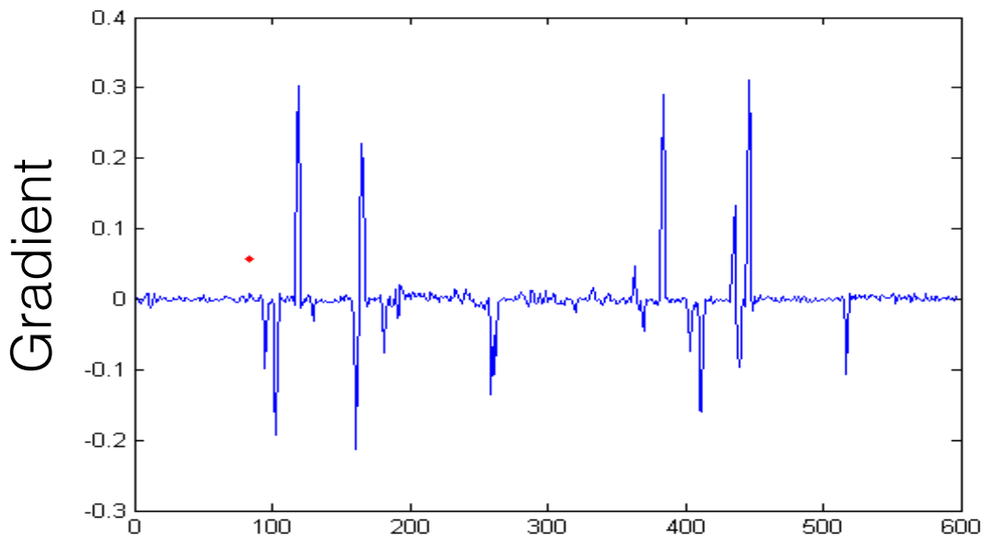




Intensity profile

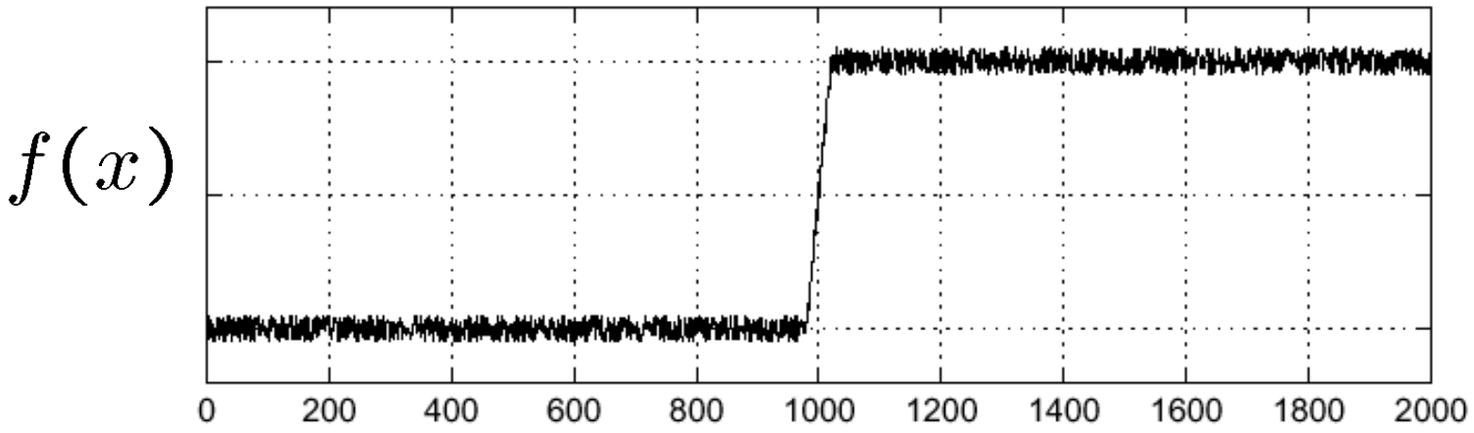
- Consider a single row or column of the image
  - Plotting intensity as a function of position gives a signal

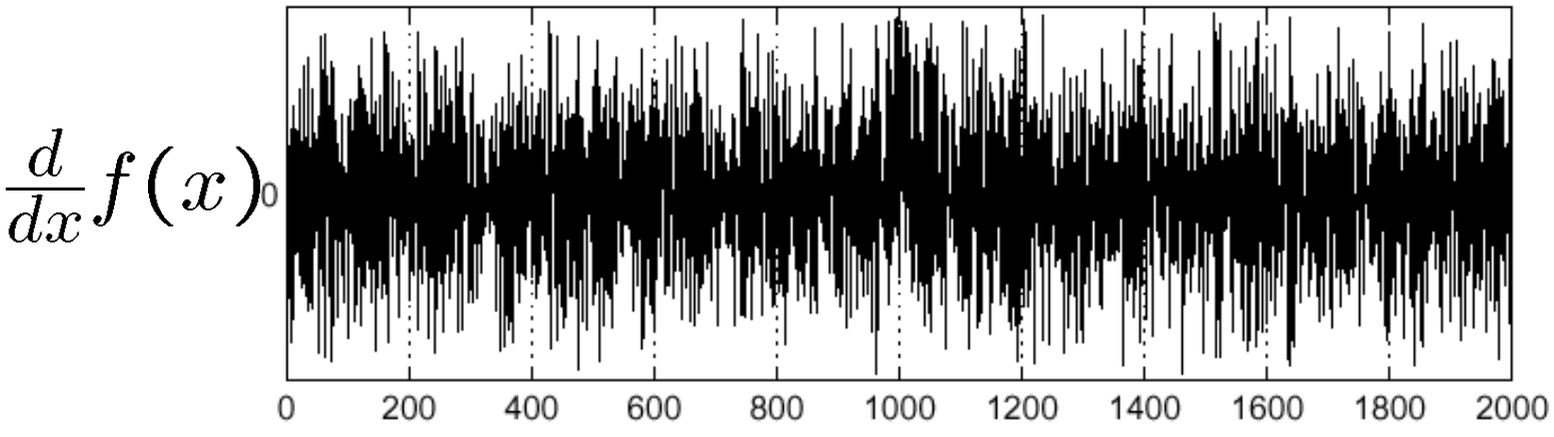




Effects of noise

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





Where is the edge?

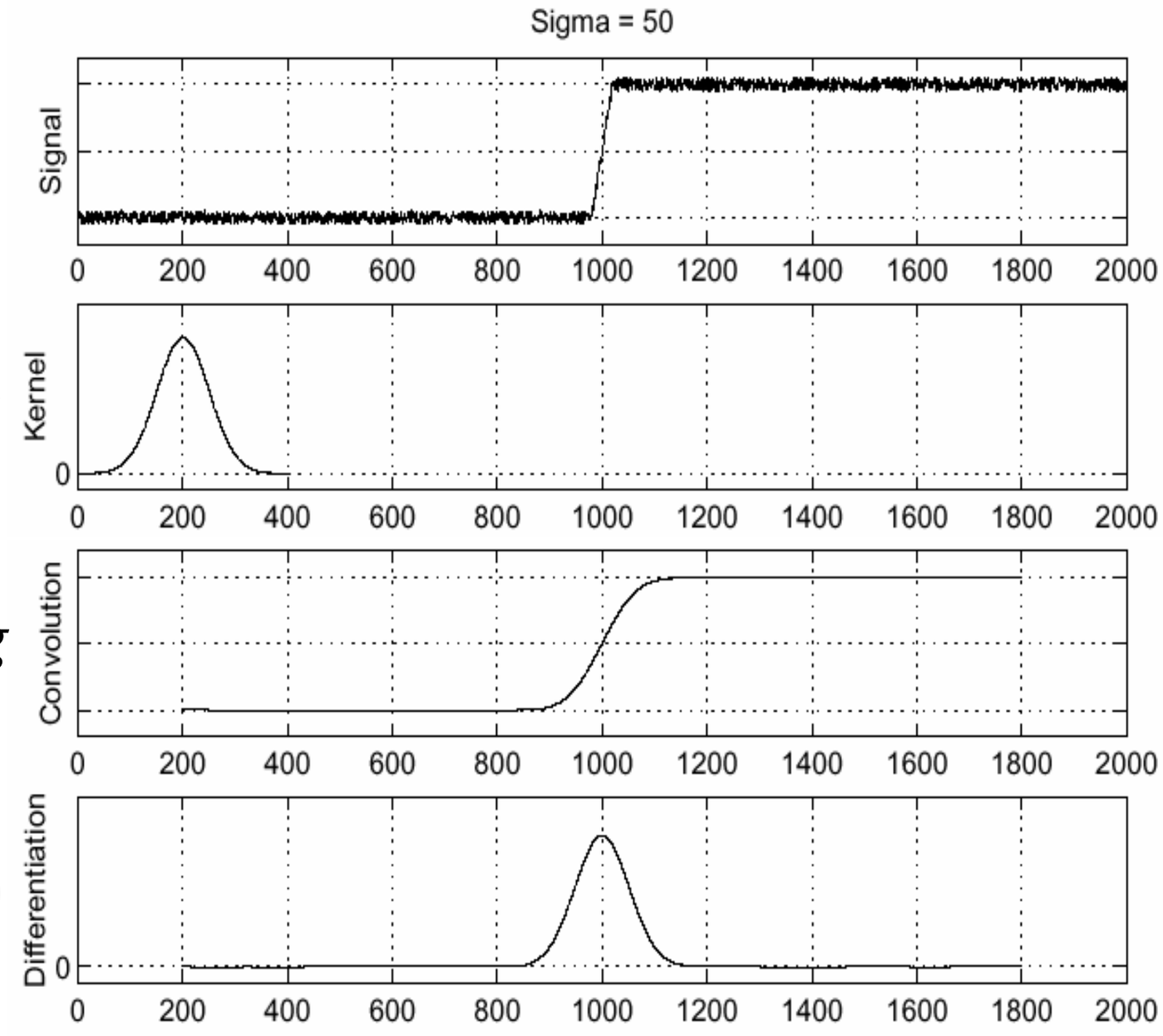
Slide credit: S. Seitz

Effects of noise

- Finite difference filters respond strongly to noise
  - Image noise results in pixels that look very different from their neighbors
  - Generally, 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

Slide credit: D. Forsyth

Solution: smooth first



f

g

$f * g$

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

Where is the edge?

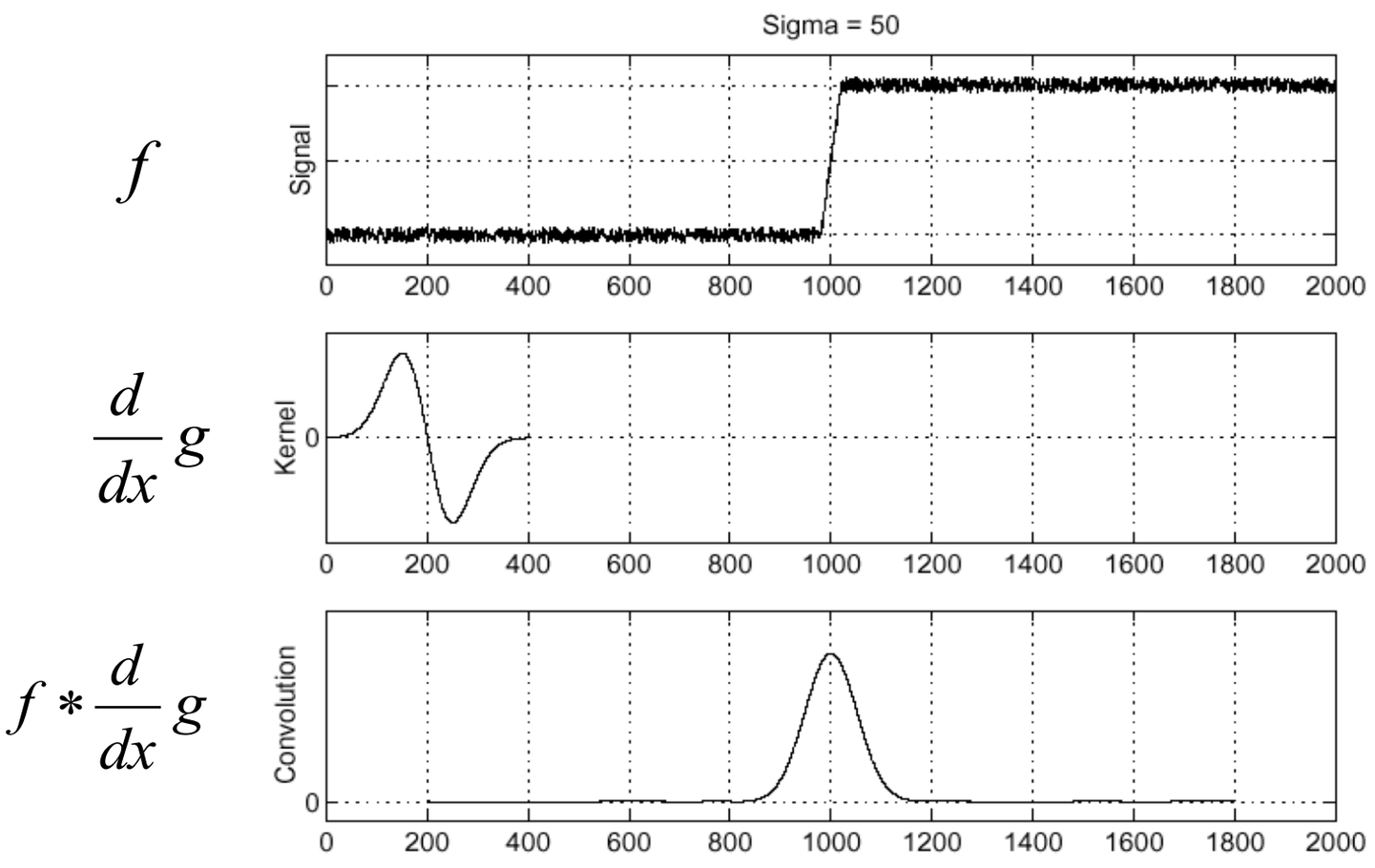
Look for peaks in $\frac{d}{dx}(f * g)$

Derivative theorem of convolution

- Differential property of convolution:

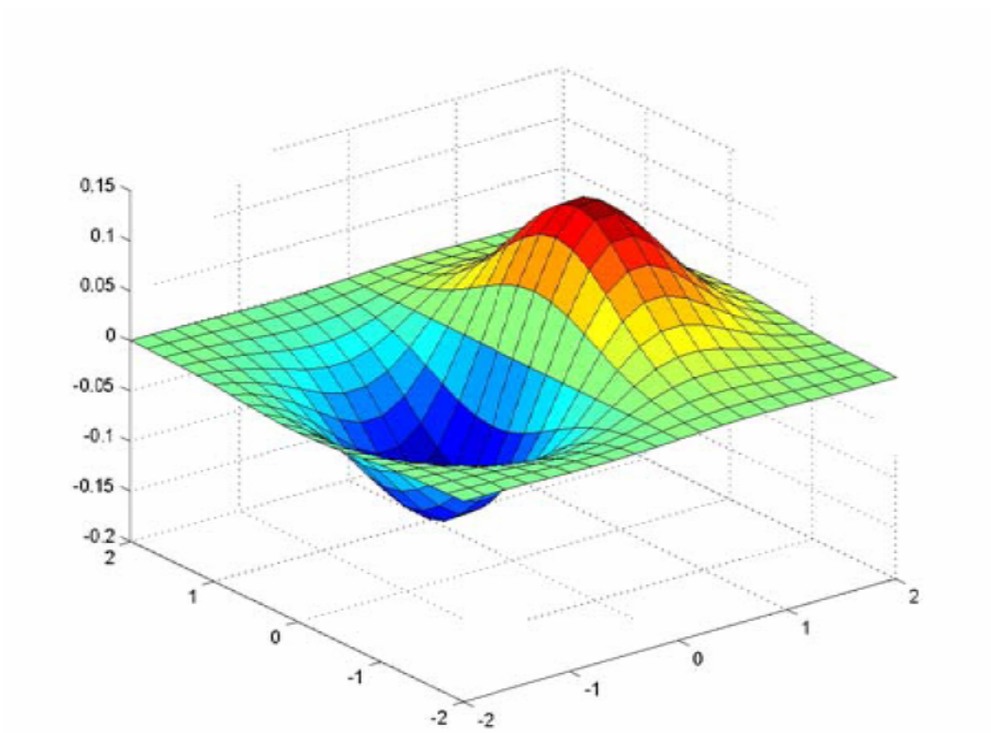
$$\frac{d}{dx}(f * g) = f * \frac{d}{dx} g$$

  - It gives us a useful property and save us one operation:

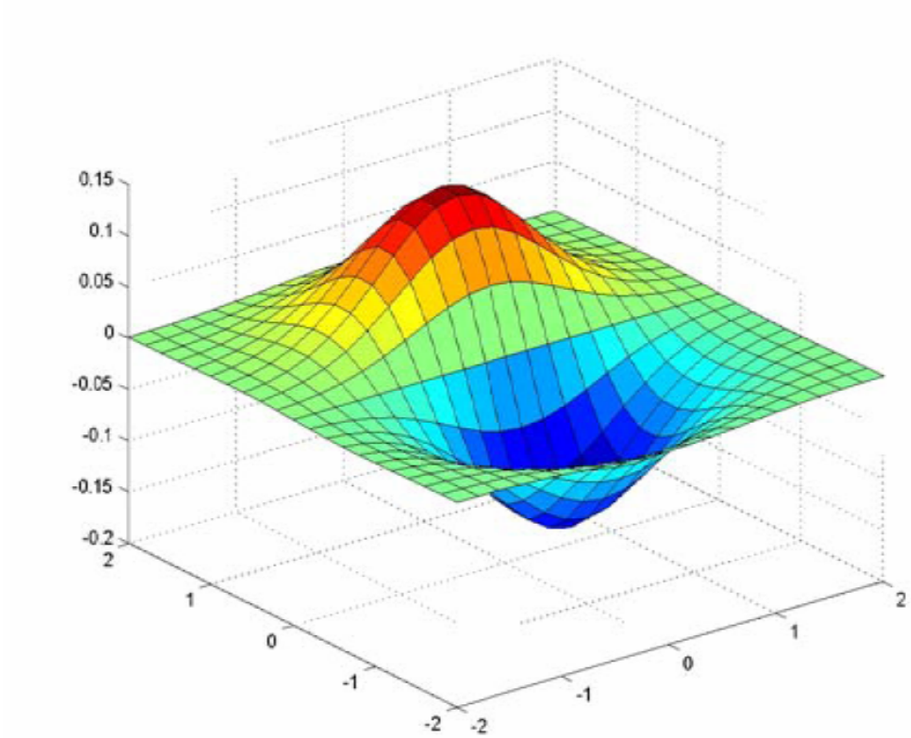


Slide credit: S. Seitz

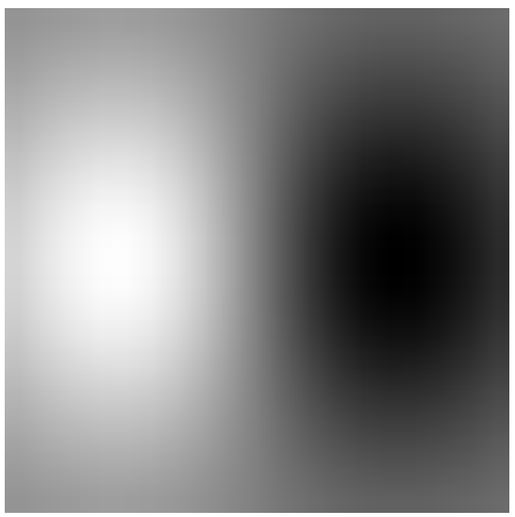
Derivative of Gaussian filter

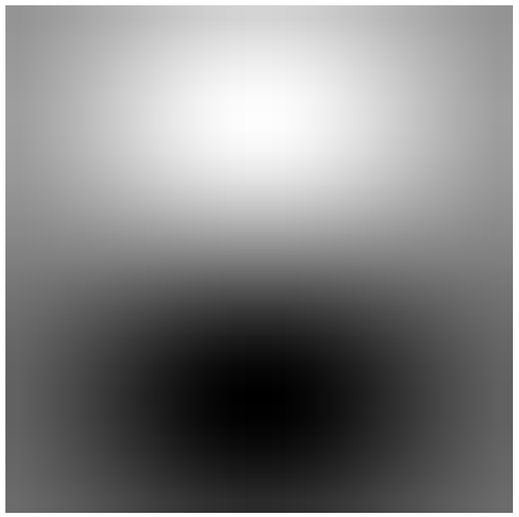


x-direction



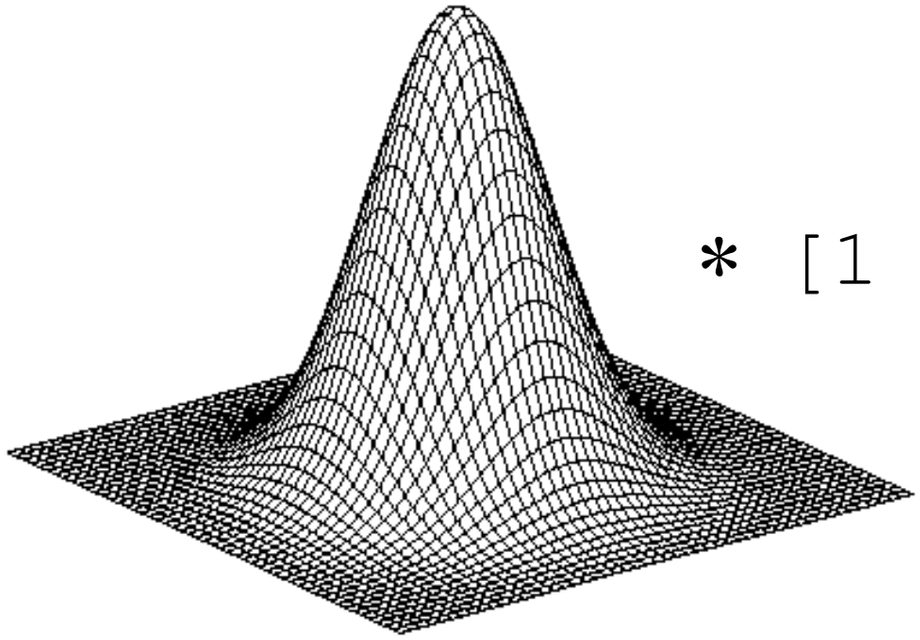
y-direction



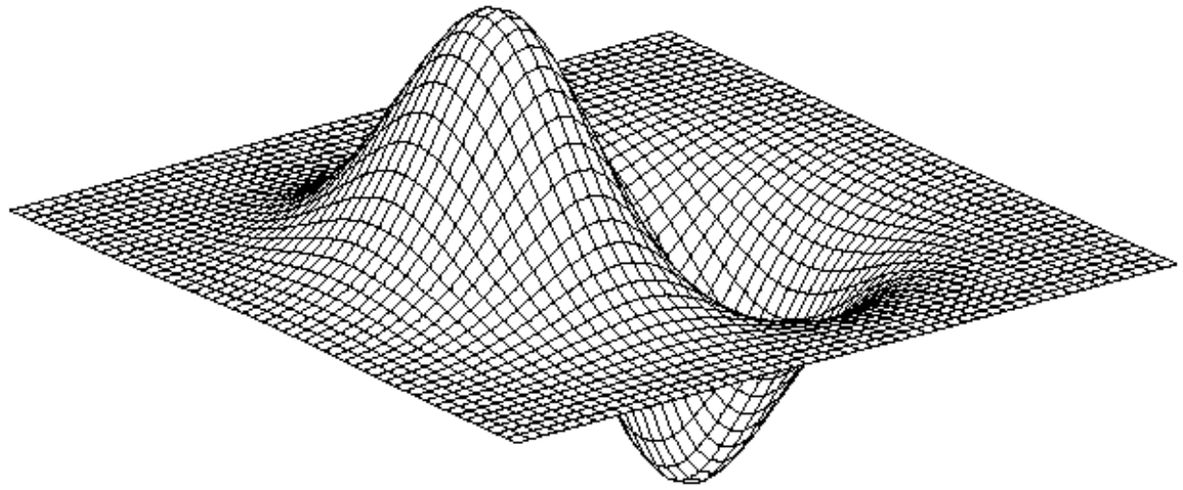


Slide credit: S. Lazebnik

Derivative of Gaussian filter



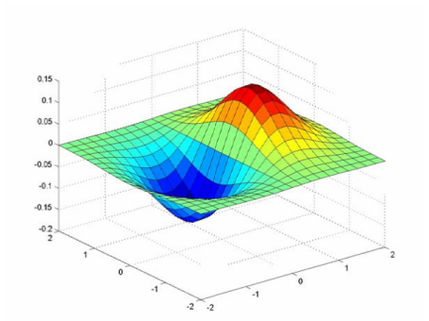
* [1 0 -1] =



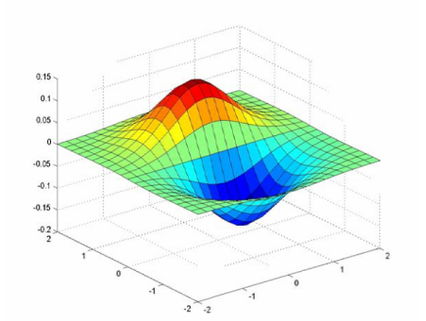
2D Gaussian

x-derivative

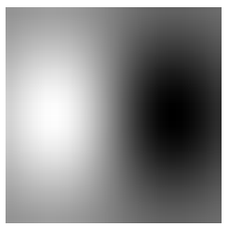
Derivative of Gaussian filter

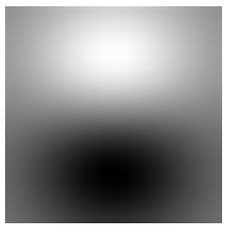


x-direction



y-direction





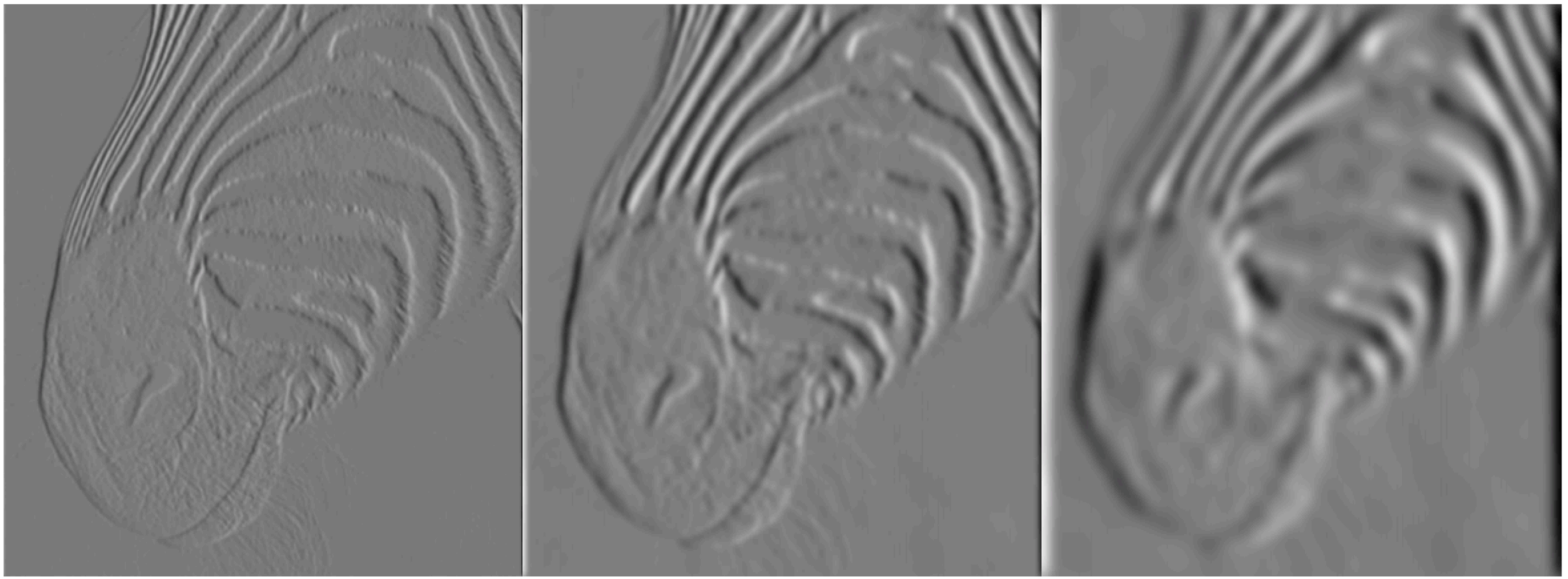
Note: separability of the Gaussian Filter

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

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

- 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

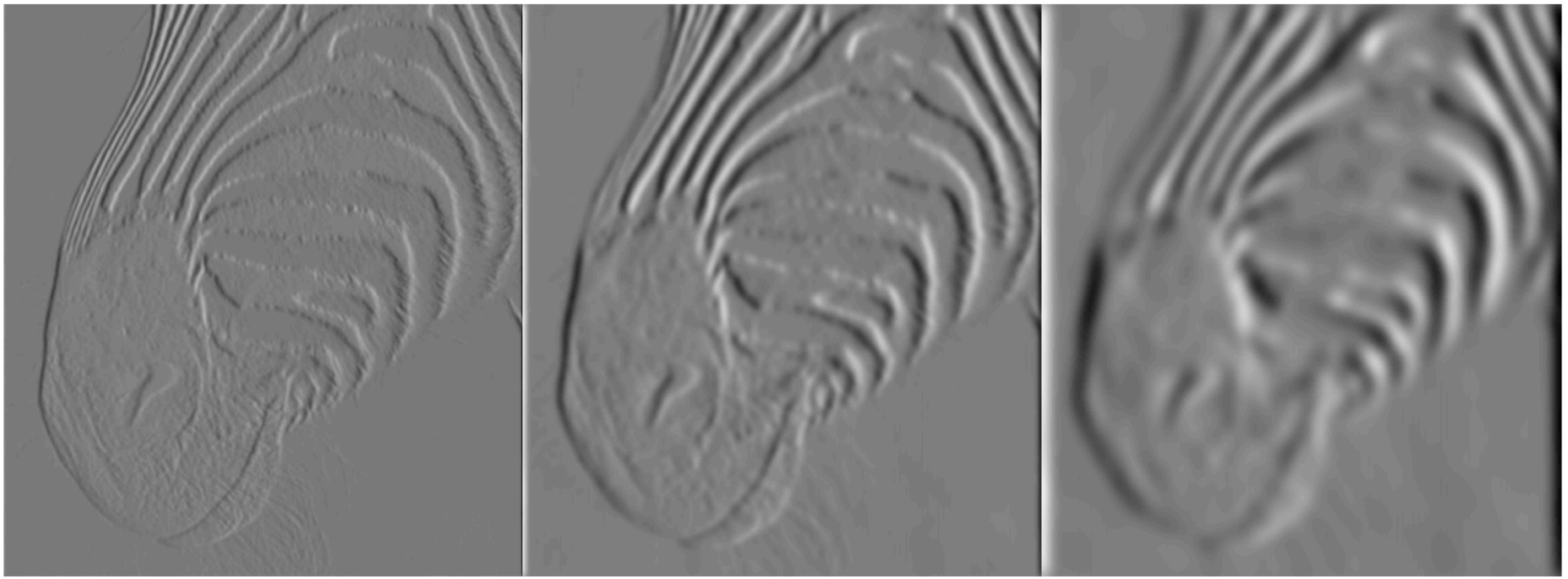


1 pixel 3 pixels 7 pixels

- 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



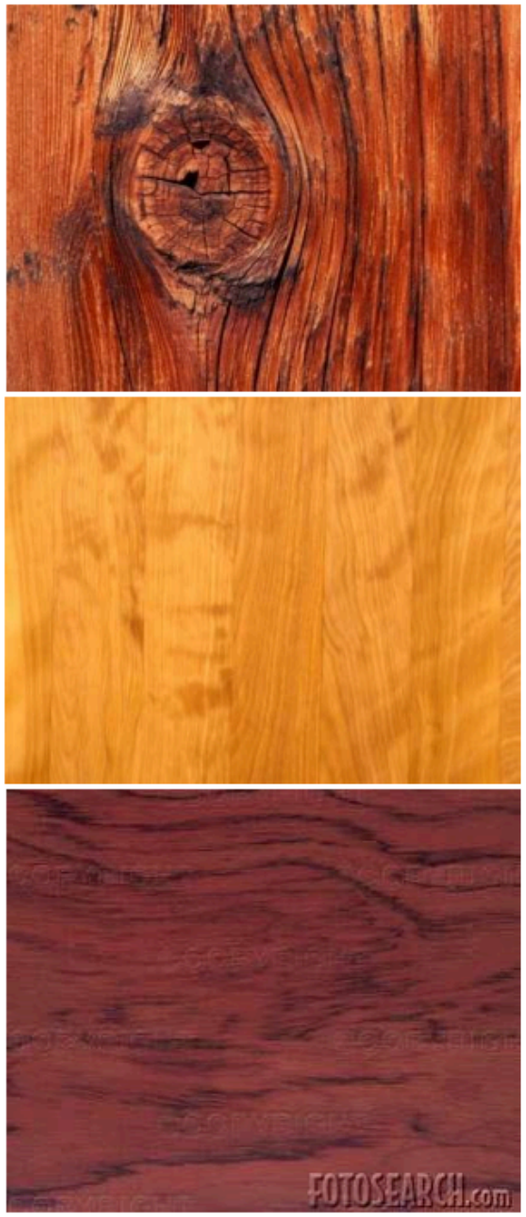
1 pixel 3 pixels 7 pixels

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

Image credit: D. Forsyth

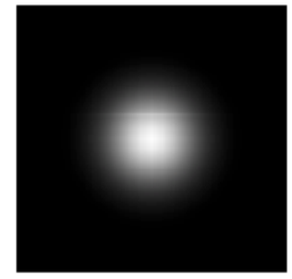
So, what scale to choose?

- It depends what we are looking for…

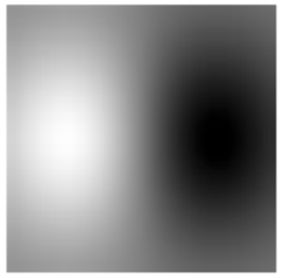


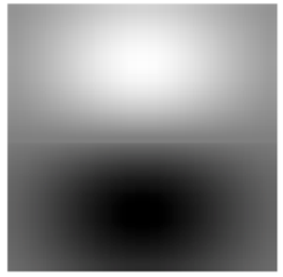
Slide credit: K. Grauman

Review: smoothing *vs* derivative filters



- Smoothing filters
  - Gaussian: remove high-frequency components
  - Can the values of a smoothing filter be negative?
  - What should the values sum to?
    - **One**: constant regions are not affected by the filter





- Derivative filters
  - Derivatives of Gaussian
  - Can the values of a derivative filter be negative?
  - What should the values sum to?
    - **Zero**: no response in constant regions
  - High absolute value at points of high contrast

Building an edge detector

Original (input) image

Output

Edge detection: Sobel filter

- A derivative filter + some smoothing

Input image

$*$

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

Left Sobel

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

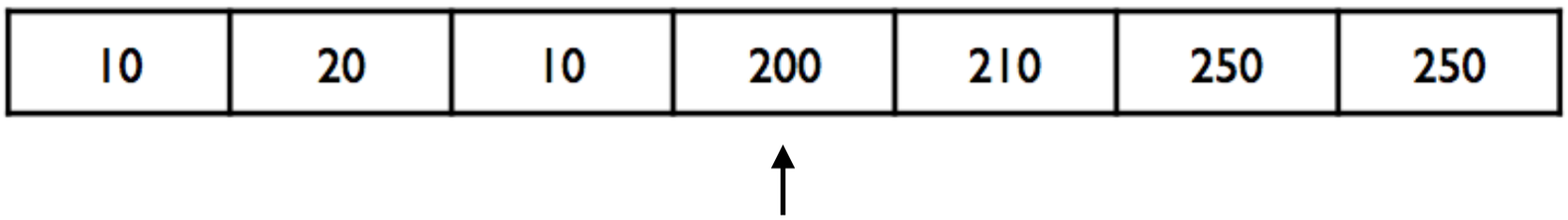
- Where does this filter come?
  - Remember: (central) discrete derivative in 1D

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

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

1D derivative filter

  - Example:



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

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

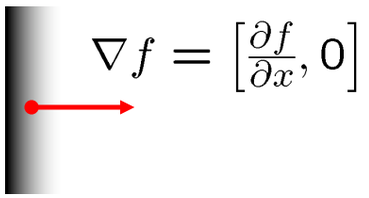
0	1
-1	0

1	0
0	-1

Building an edge detector

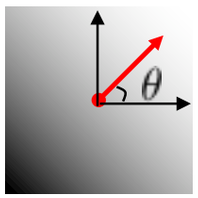
- The Canny Edge detector
  - Filter image with derivative of Gaussian
  - Find magnitude and orientation of gradient
  - …

Image gradient



$$\nabla f = \left[\frac{\partial f}{\partial x}, 0\right]$$

$$\nabla f = \left[0, \frac{\partial f}{\partial y}\right]$$



$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$

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

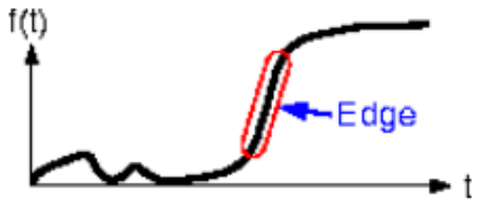
$$\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

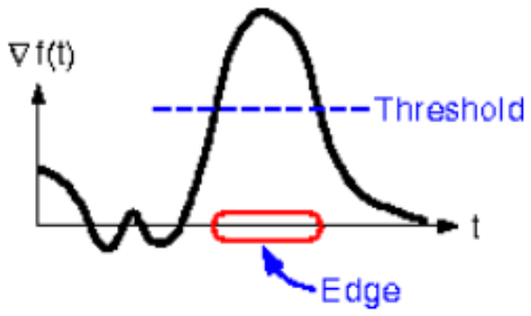
Building an edge detector: Canny

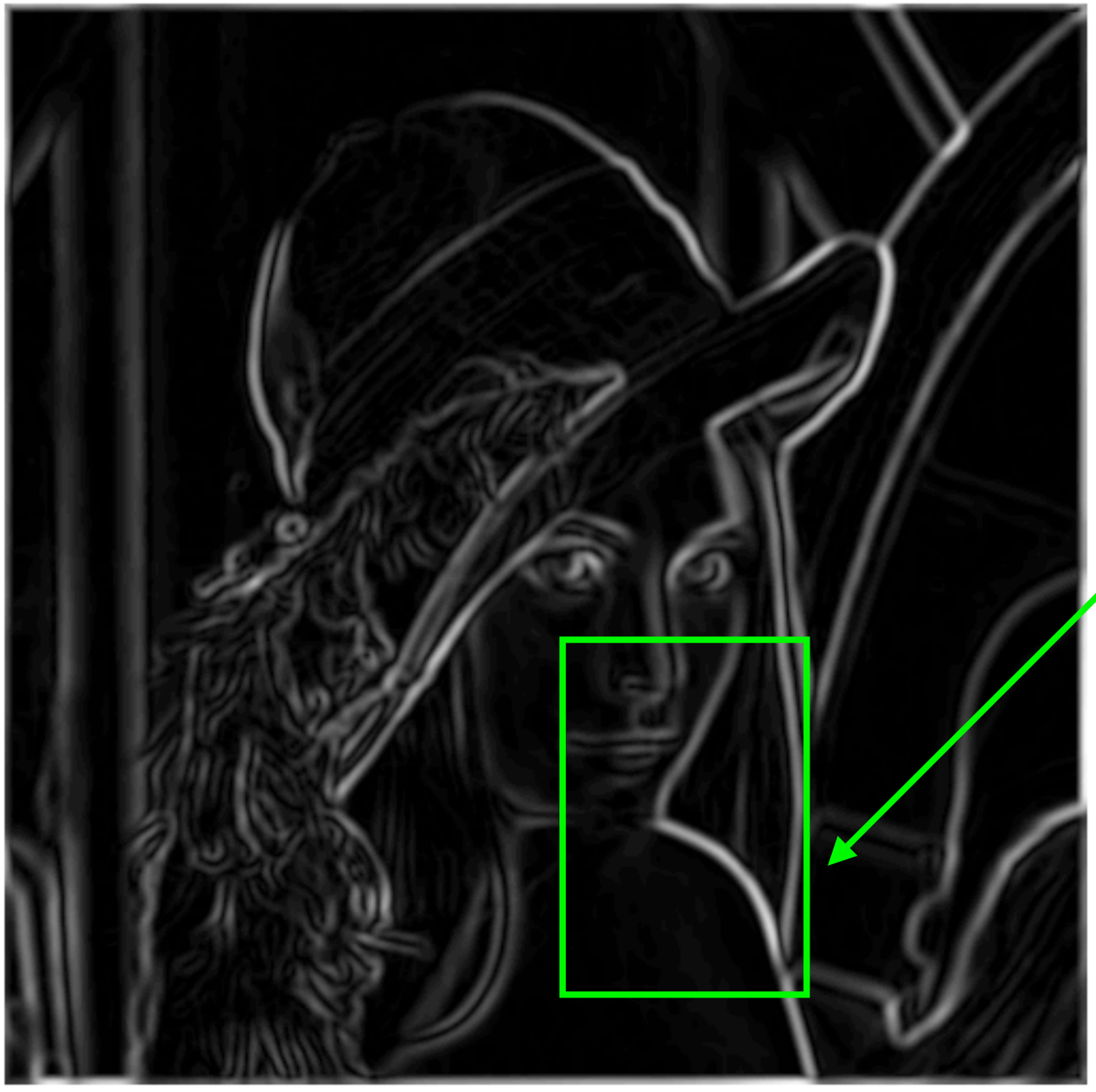
Gradient magnitude

Building an edge detector: Canny

Thresholding





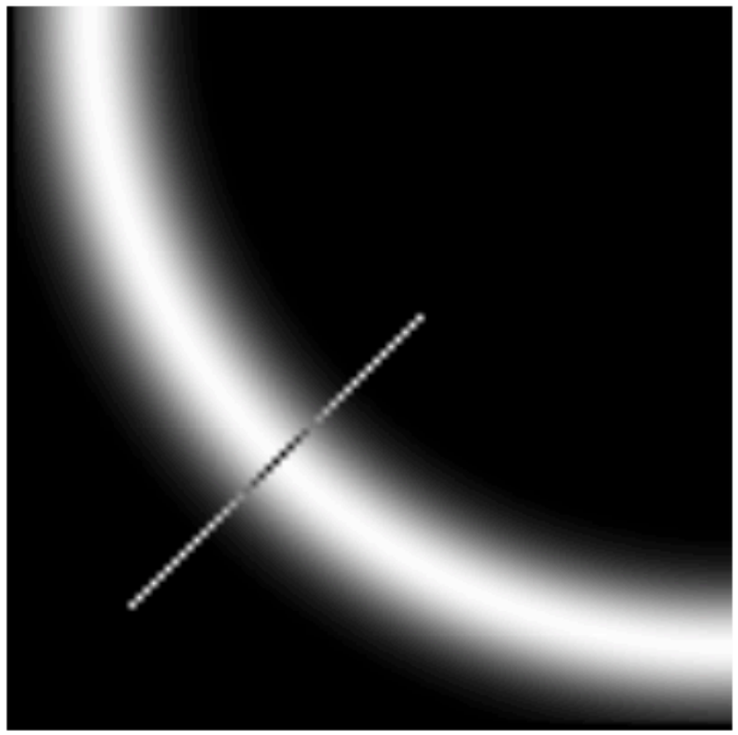


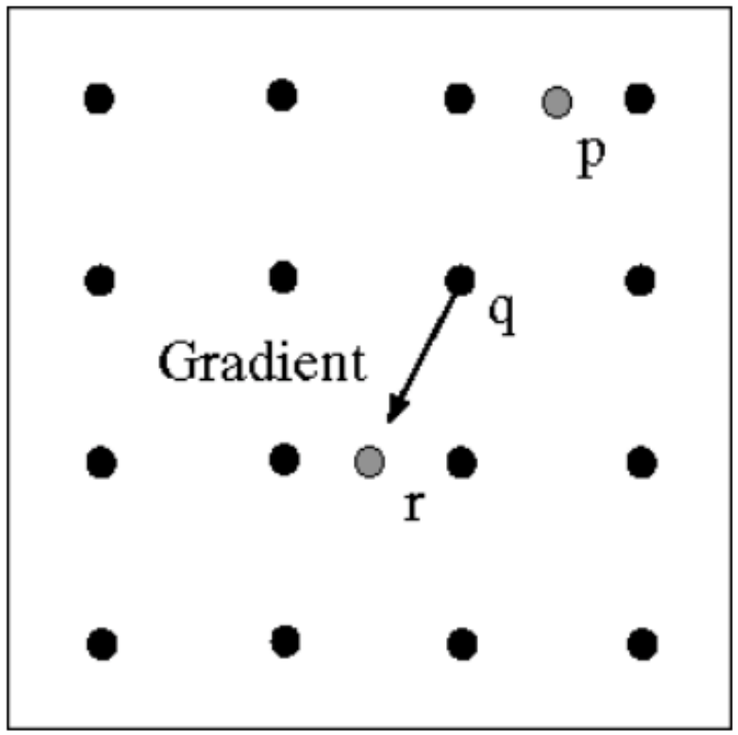
Gradient magnitude

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

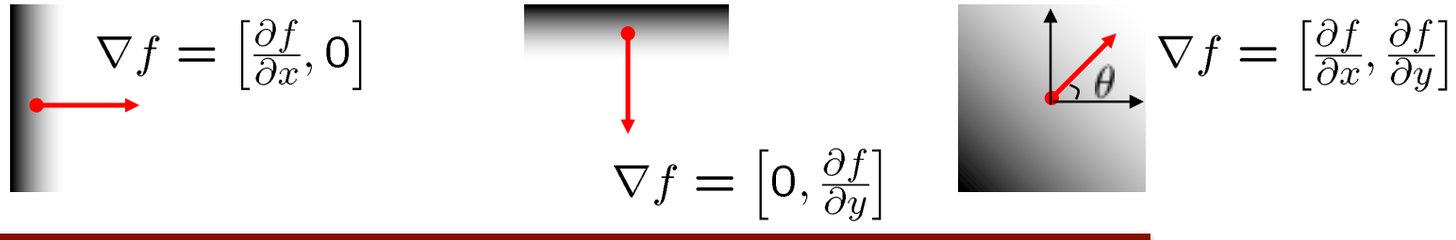
Non-maximum suppression: thinning

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

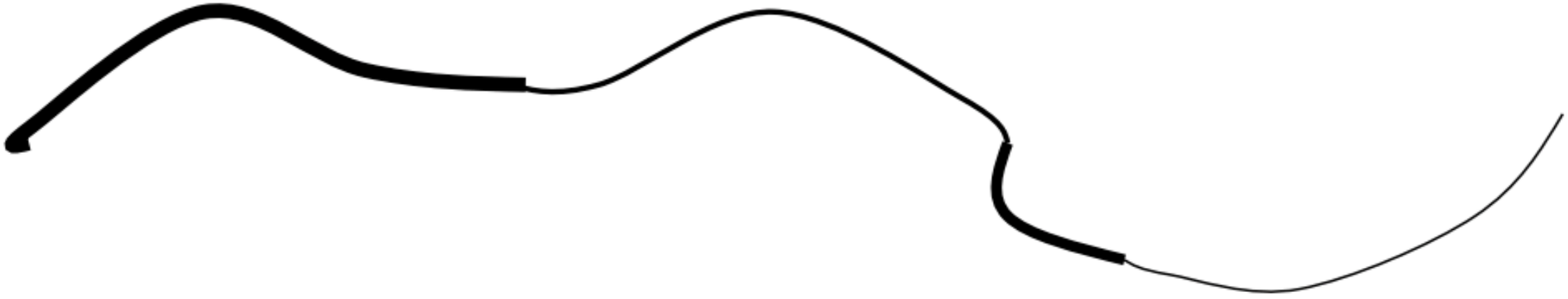


$$\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$$

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

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');`

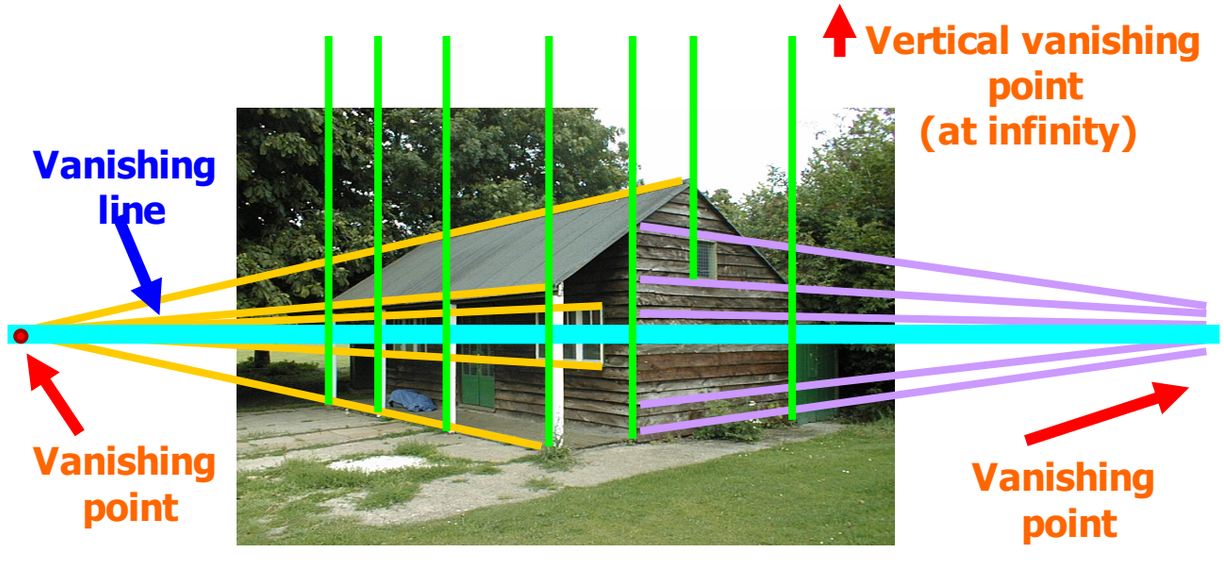
J. Canny, “A Computational Approach To Edge Detection”, TPAMI 1986

Recall: why do we care about edges?

- Extract information, recognize objects

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

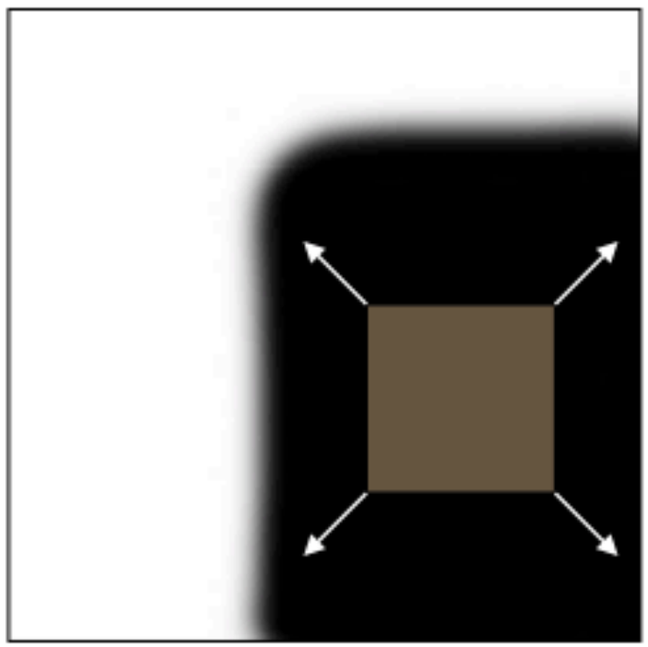
- Recover geometry and viewpoint



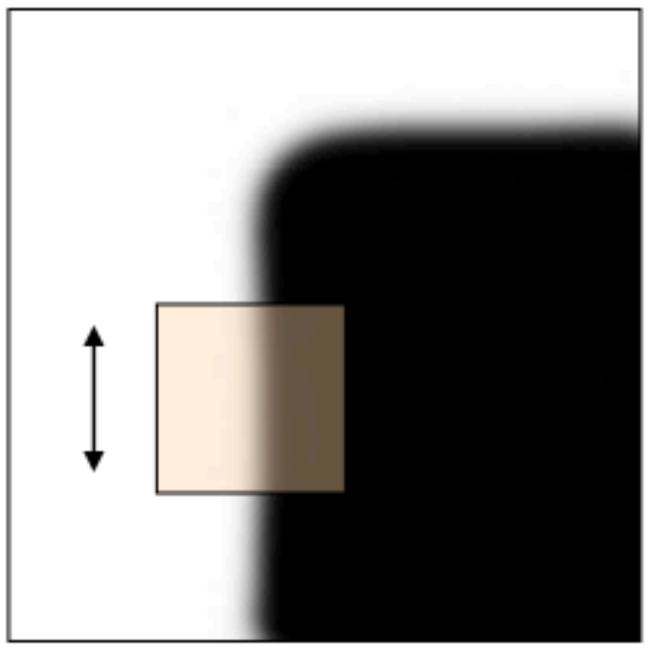
Slide credit: J. Hayes

Edges, corners and blobs (keypoints)

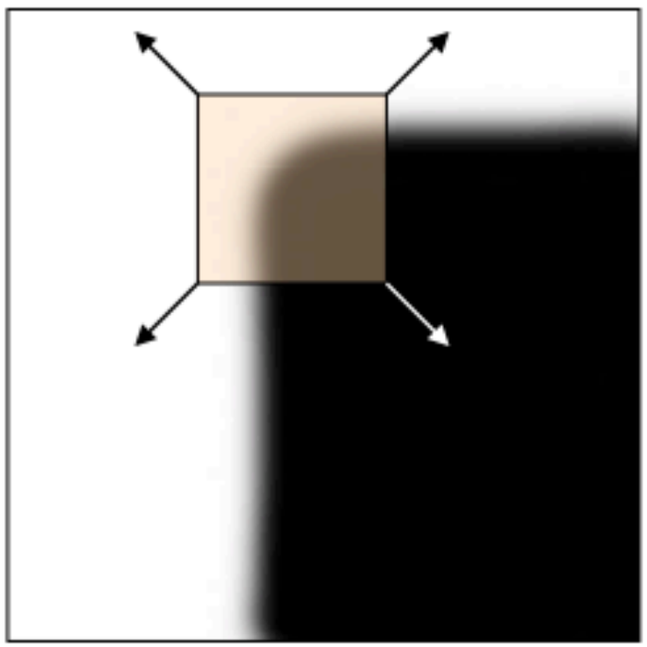
- Low-level image features: a basic classification



“flat” region:
no change in all directions



“edge”:
no change along the edge direction

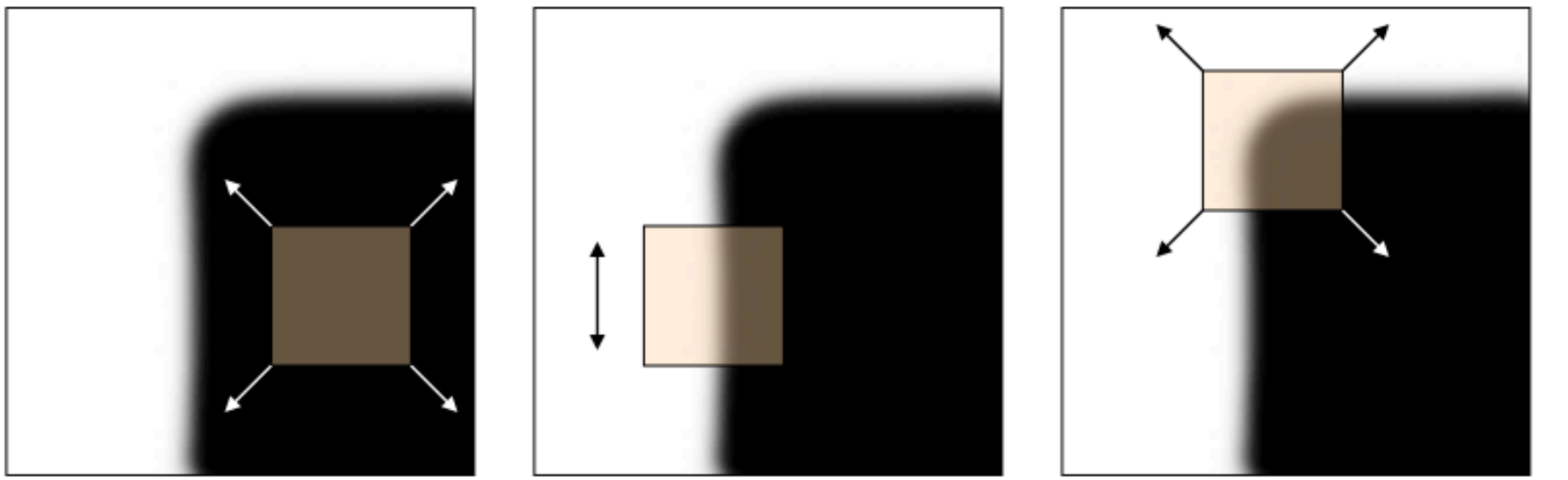


“corner”:
significant change in all directions

Slide credit: A. Efros

Corner detection: basic idea

- 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

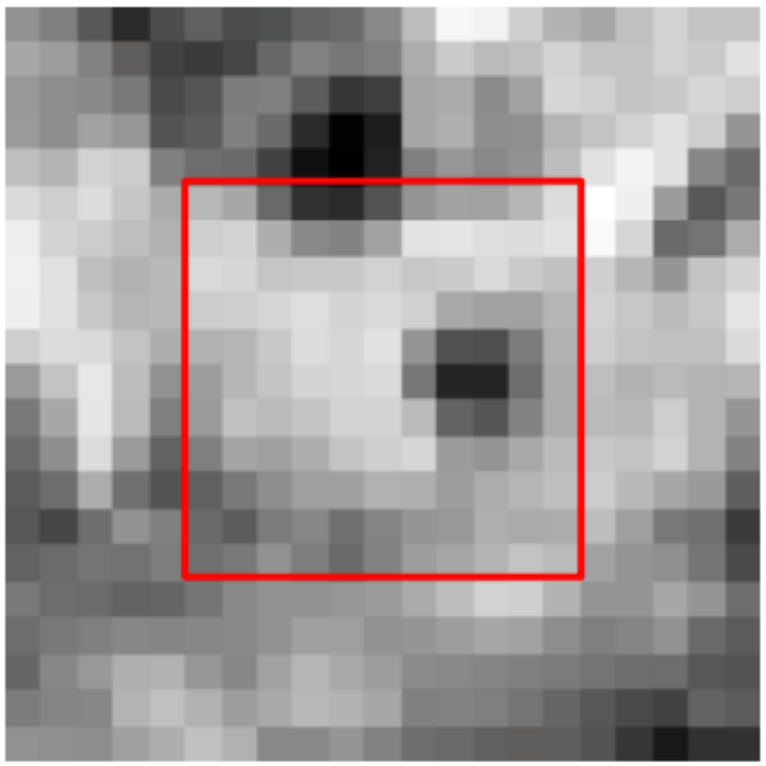
Slide credit: S. Lazebnik

Corner detection: basic idea

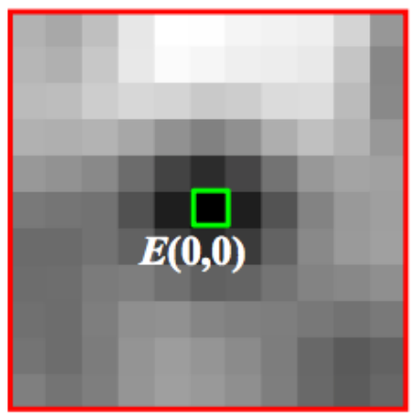
- Given an image I, Captures change in appearance of window W for the shift $[u, v]$:

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

$I(x, y)$

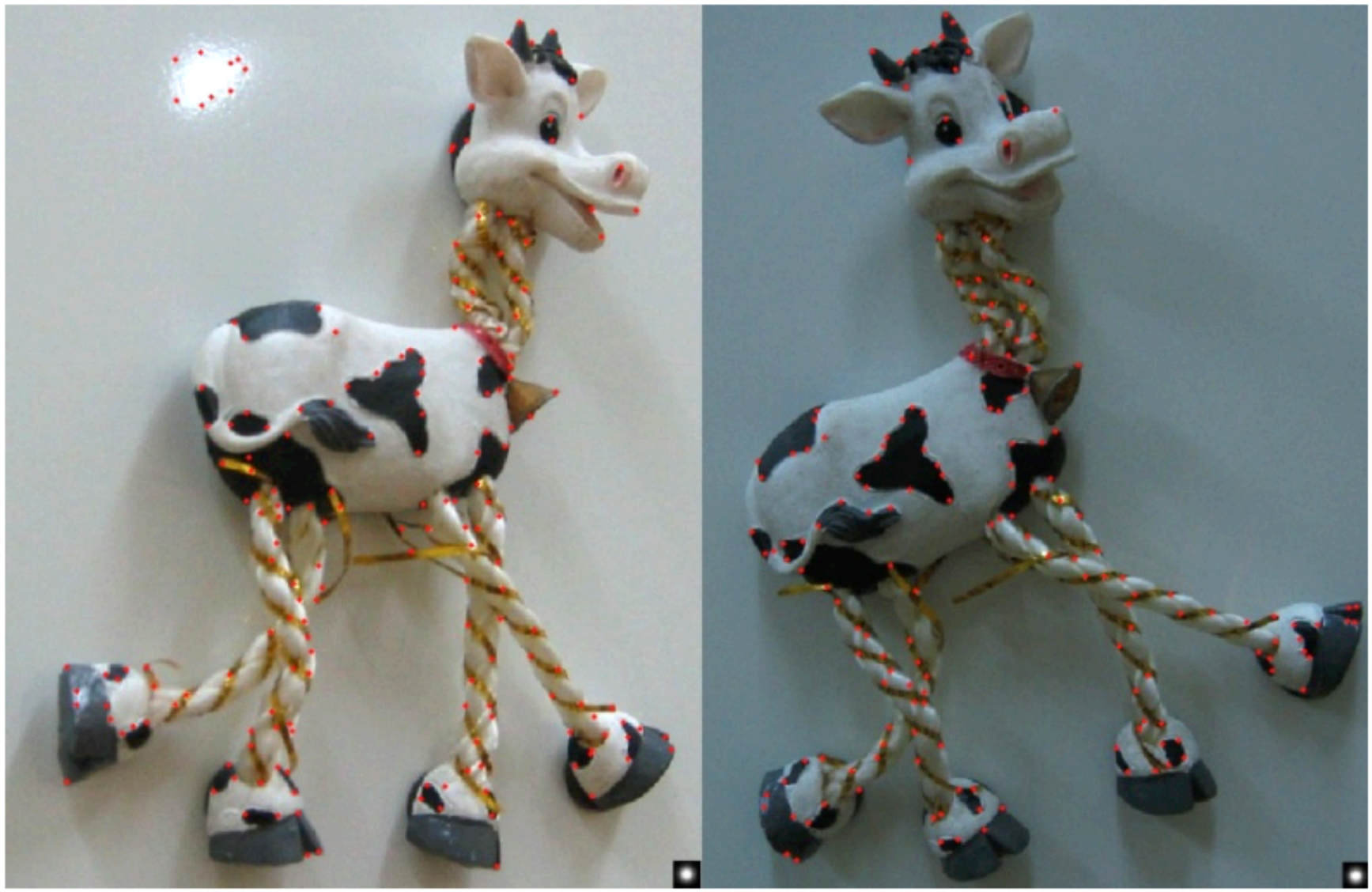


$E(u, v)$



Corners and keypoints

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



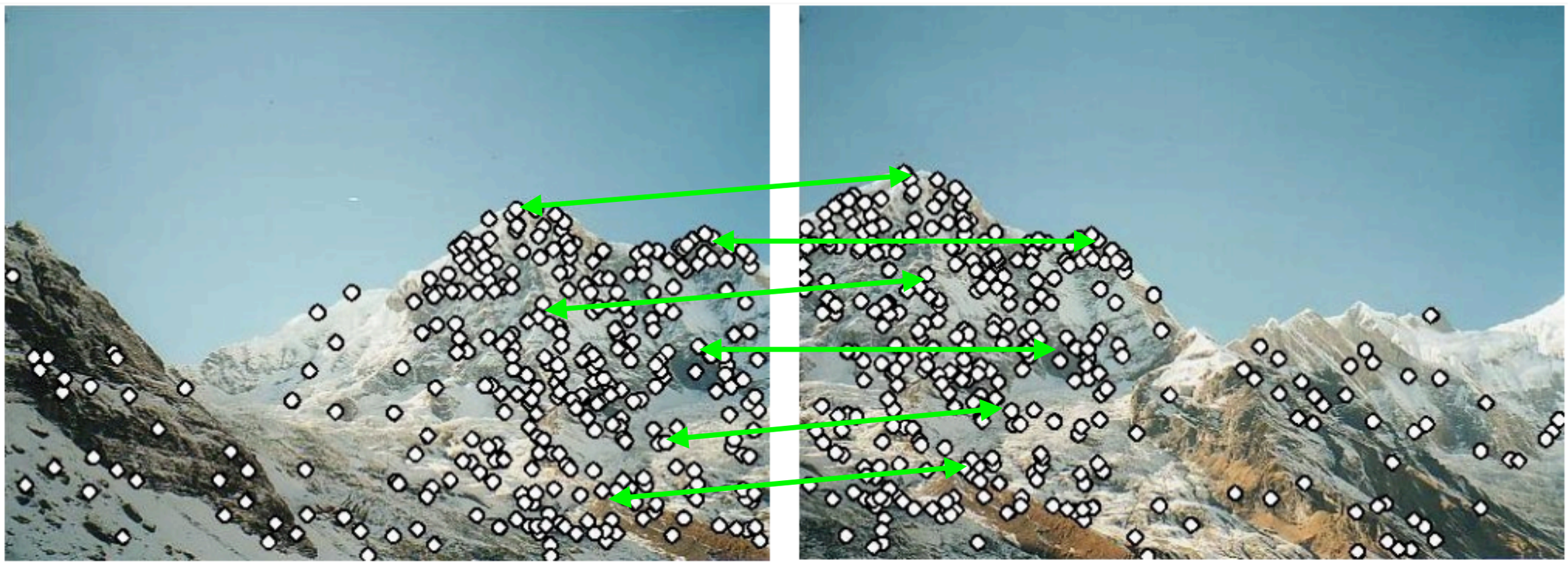
Harris corners example

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

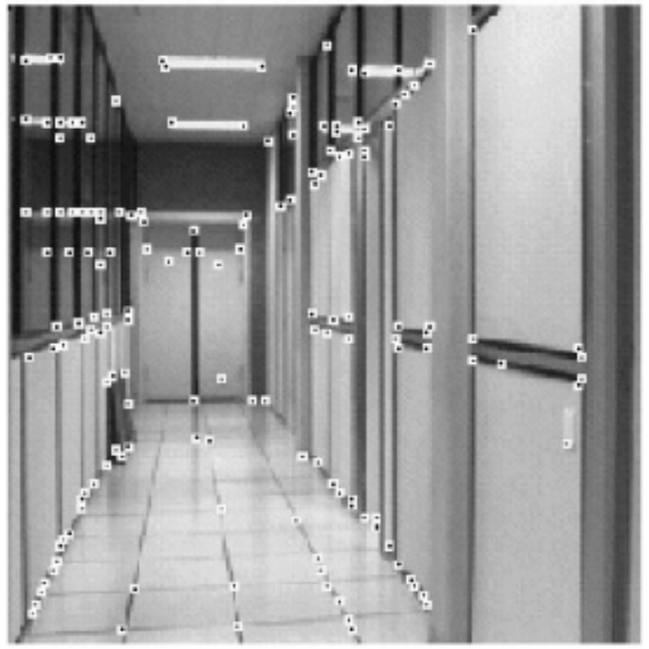
Why extract keypoints from an image?

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

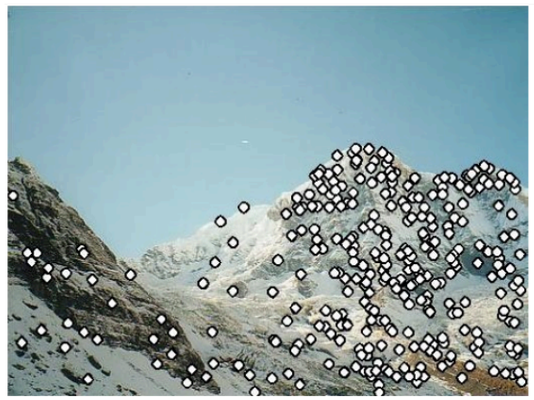
Step 1: extract keypoints **Step 2:** match keypoint features **Step 3:** align images

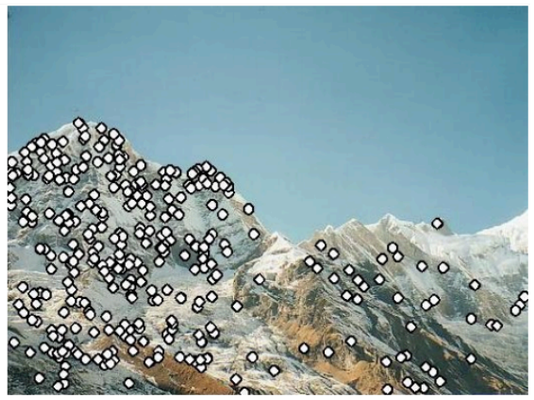
Keypoints: applications

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



Characteristics of Good Keypoints



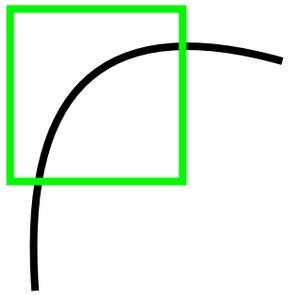


- Repeatability
  - Can be found despite geometric/photometric transformations
- Salience
  - Each keypoint is distinctive
- Compactness and efficiency
  - Many fewer keypoints than image pixels
- Locality
  - Occupies small area of the image: robust to clutter & occlusion

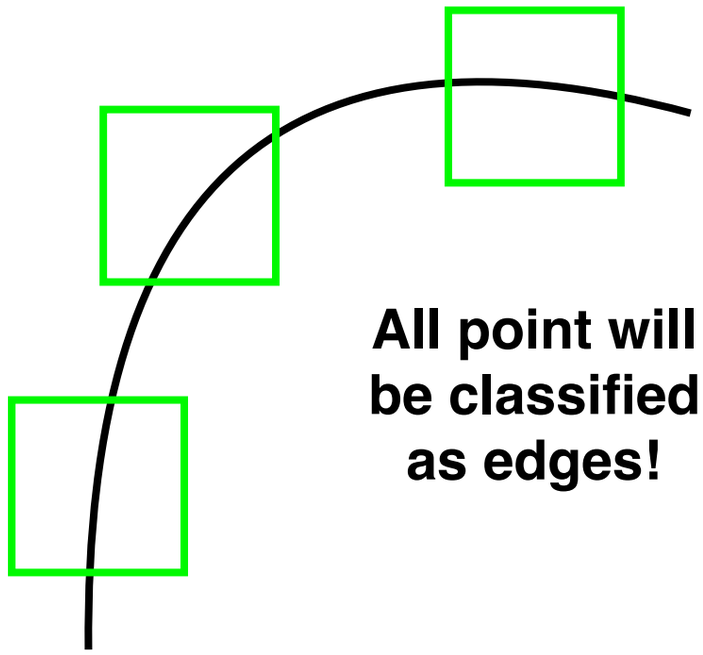
Slide credit: O. Russakovsky

An example: Harris corner detector

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



Corner

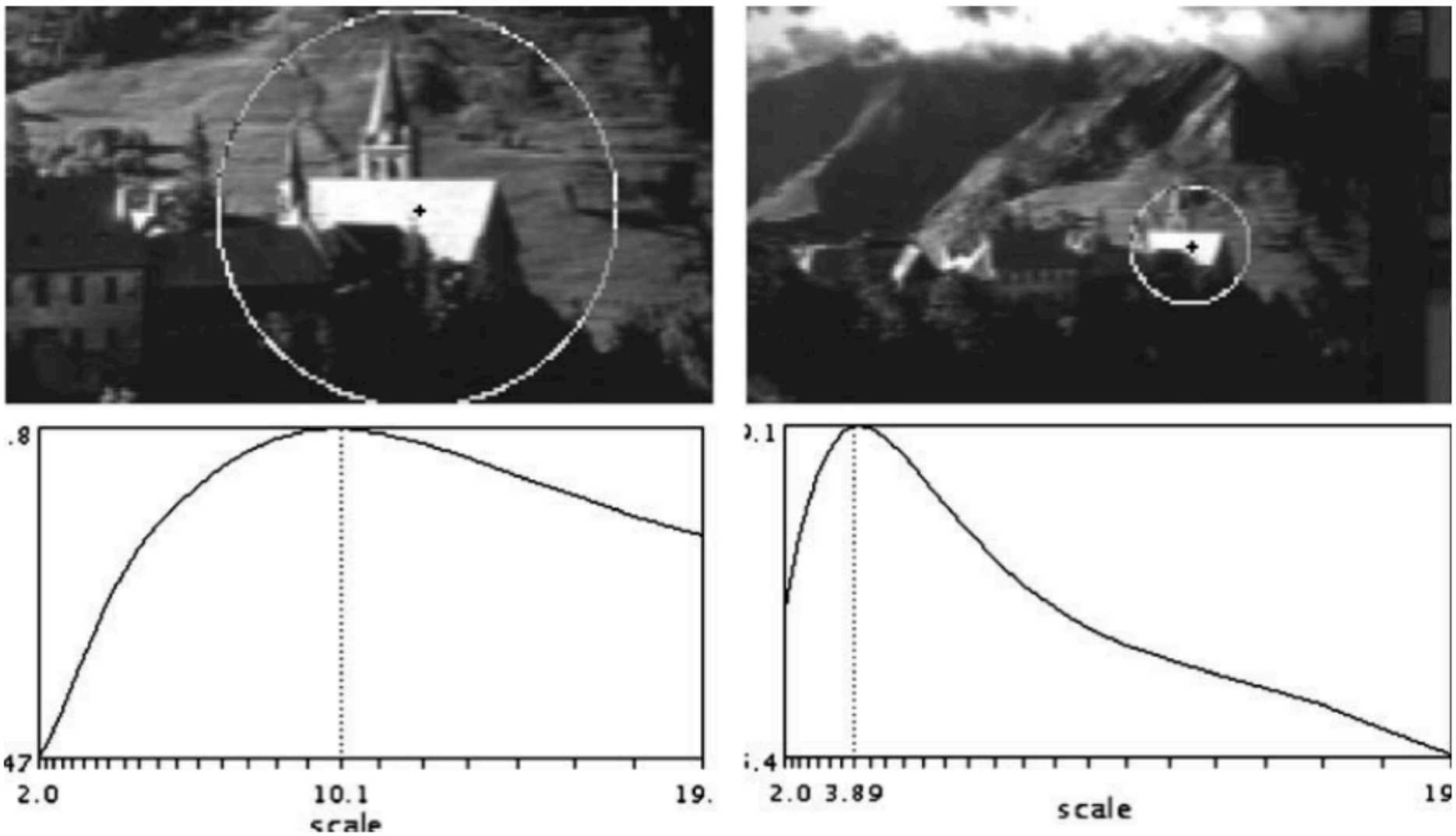


All point will be classified as edges!

Slide credit: K. Grauman

Feature detection with scale selection

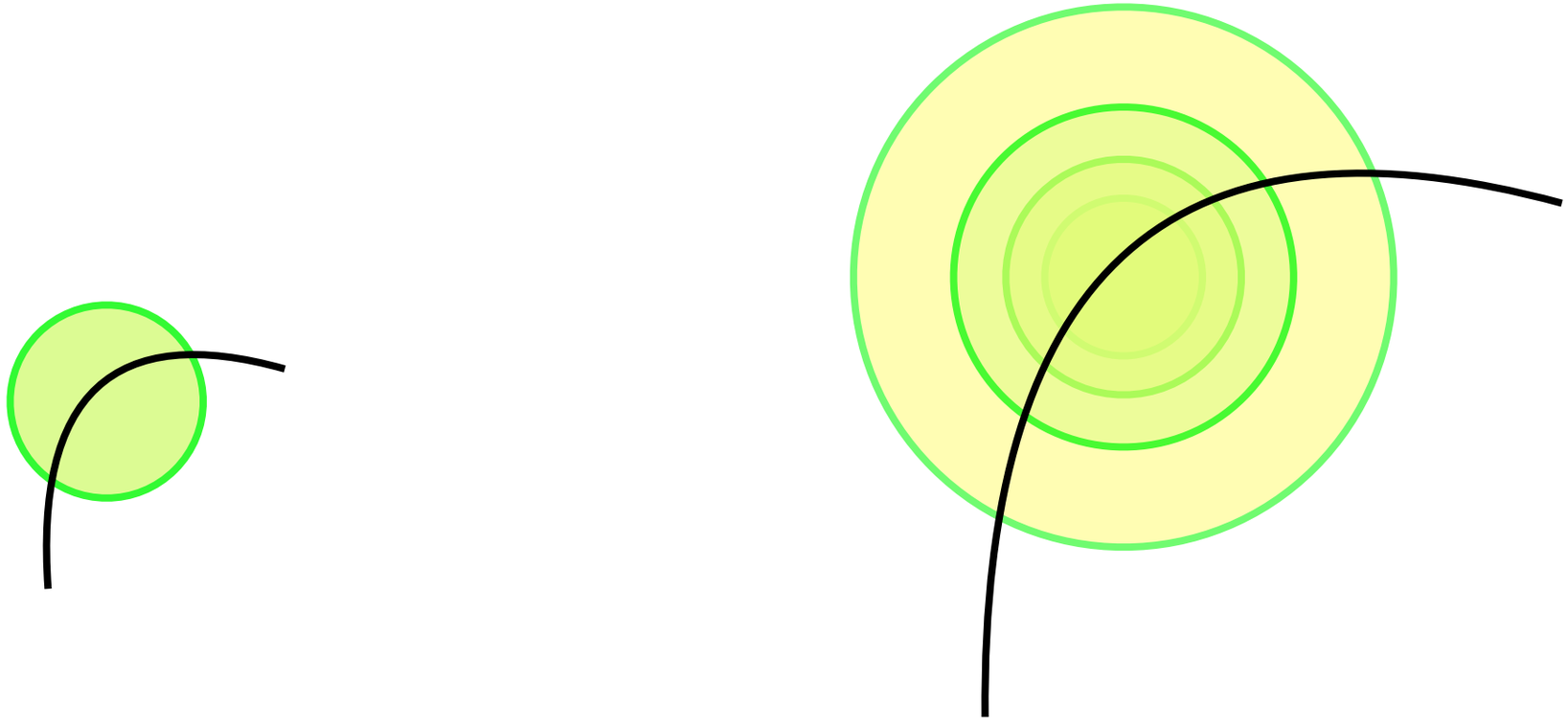
- We want to extract features with characteristic scale that is invariant (*covariant*) with the image transformation



Slide credit: S. Lazebnik

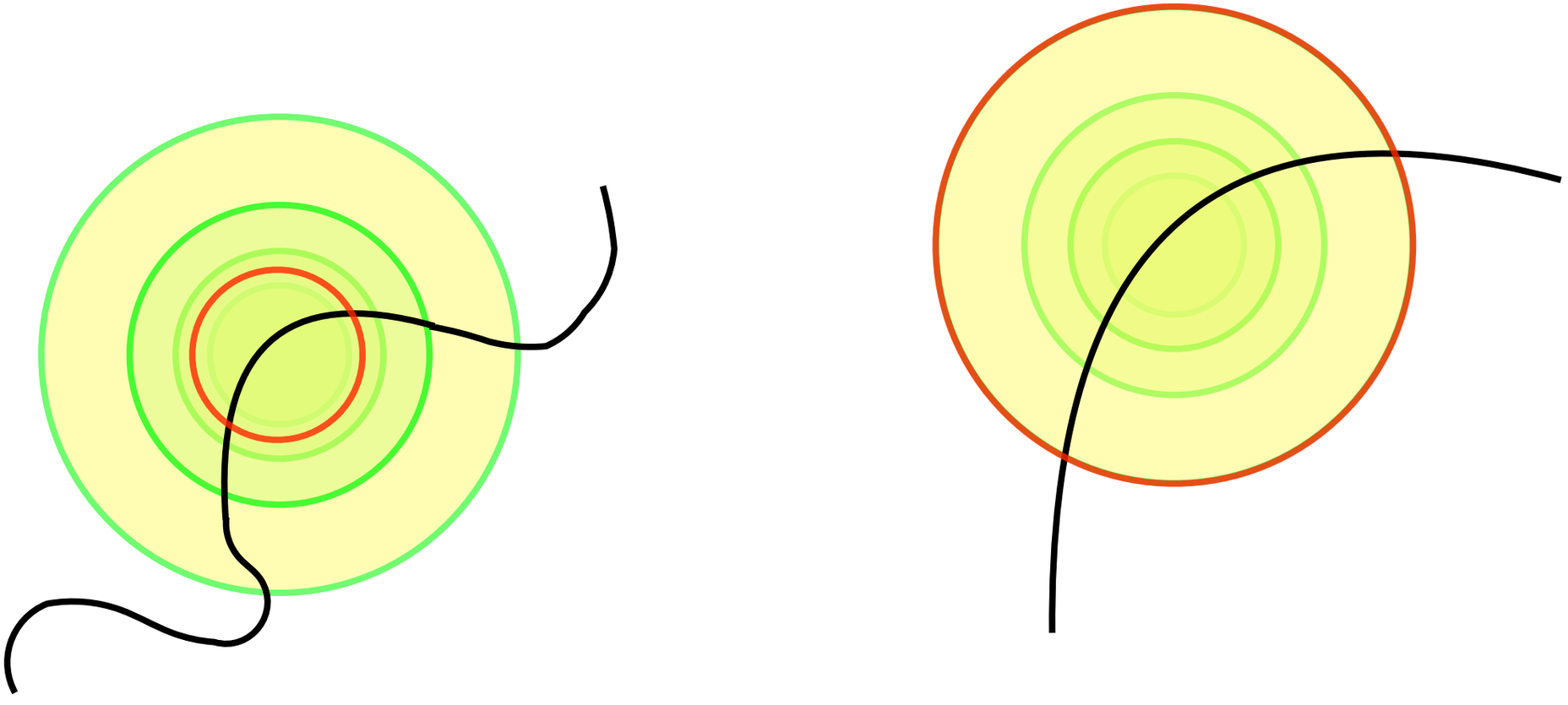
Scale invariant detection

- 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?



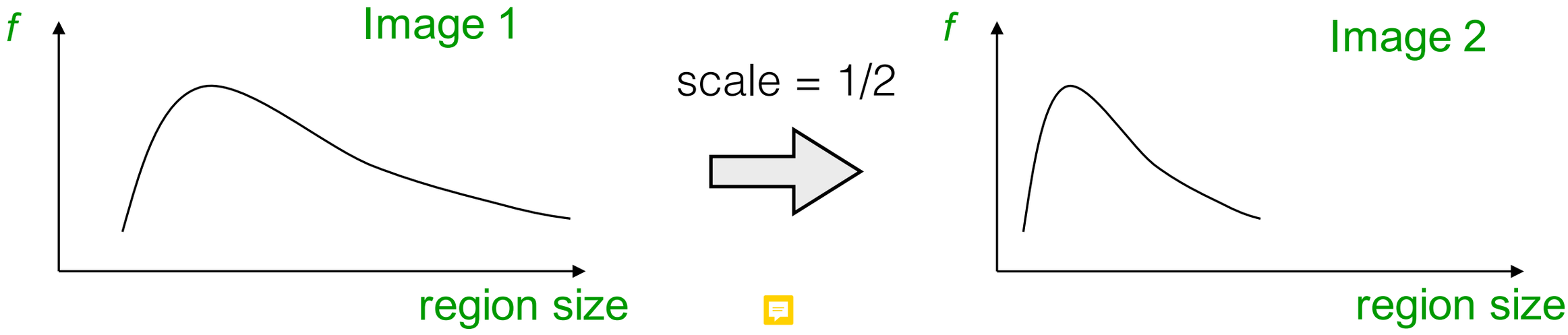
Scale invariant detection

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



Scale invariant detection

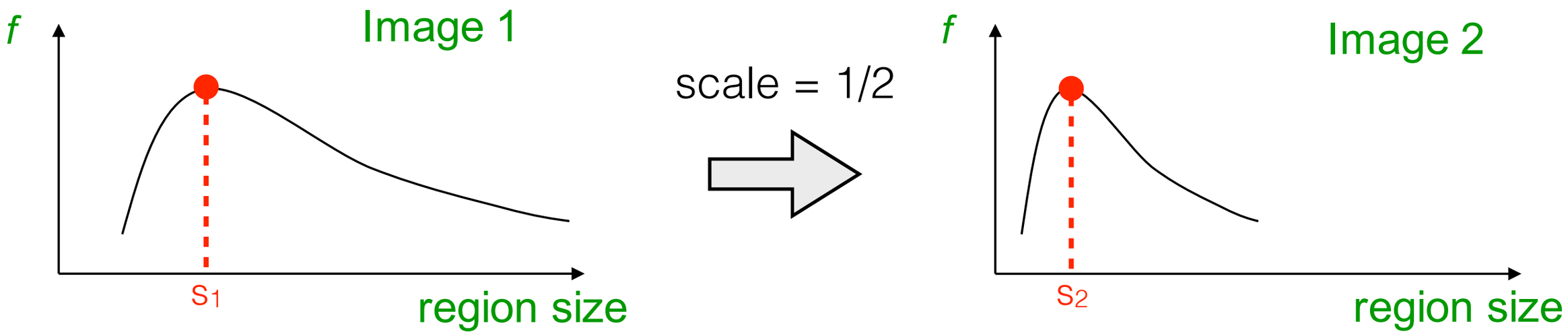
- Solution: design a function on the region (circle) which is scale invariant (*i.e.* the same for corresponding regions, even if they are at different scales)
  - e.g. average intensity (same even for different sizes)
- For a point in one image, we can consider it as a function of region size (circle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



Scale invariant detection

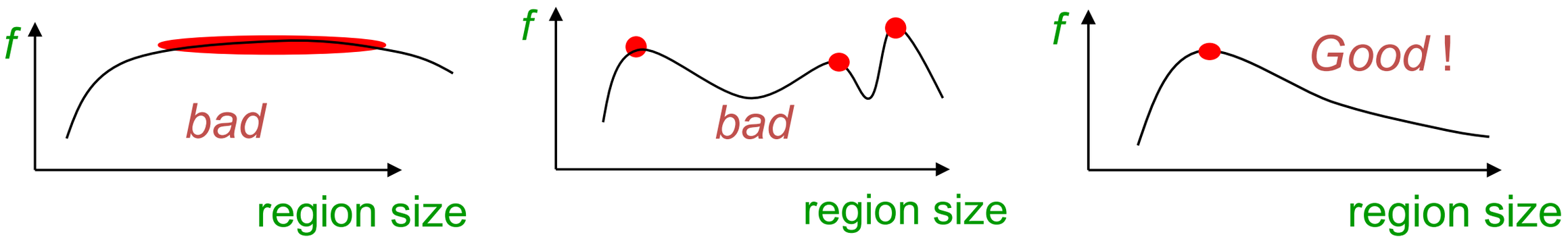
- Common approach: take a local max of this function
- Observation: region size for which the maximum is achieved should be *invariant* (*co-variant*) with scale

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



Scale invariant detection

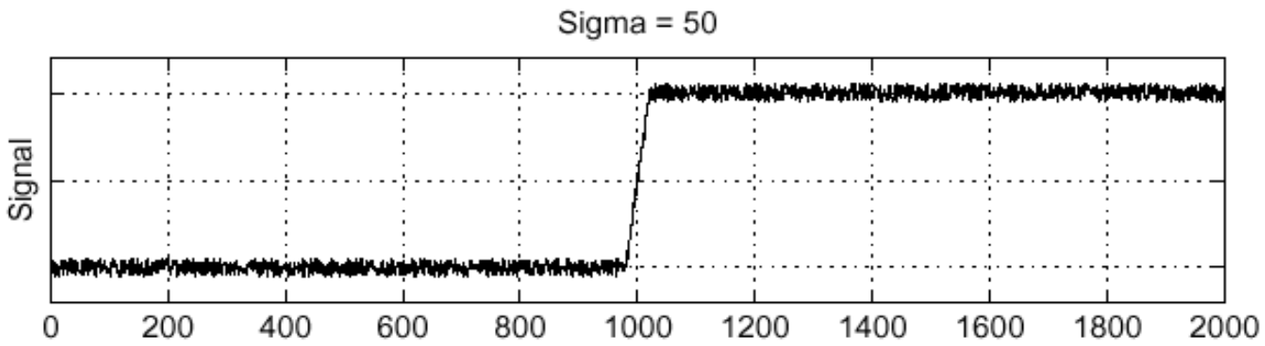
- A good function for scale detection has one stable sharp peak



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

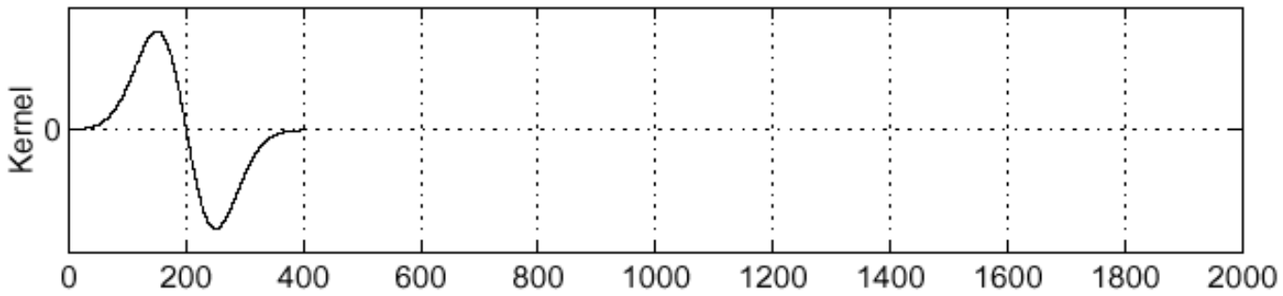
Recall: edge detection

f



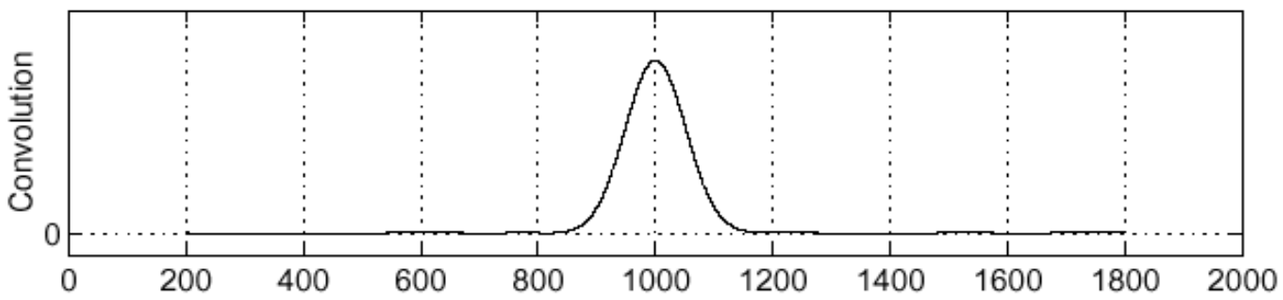
Edge

$\frac{d}{dx} g$



Derivative of Gaussian

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

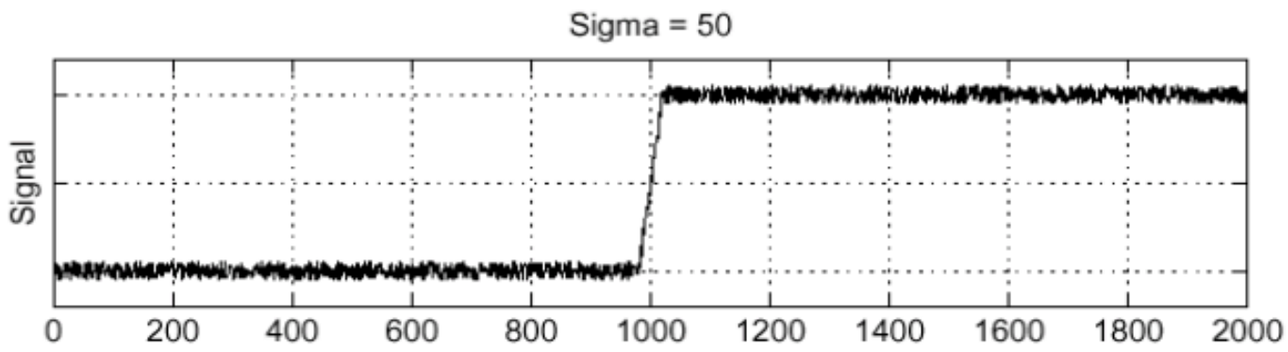


Edge = maximum of derivative

Slide credit: S. Seitz

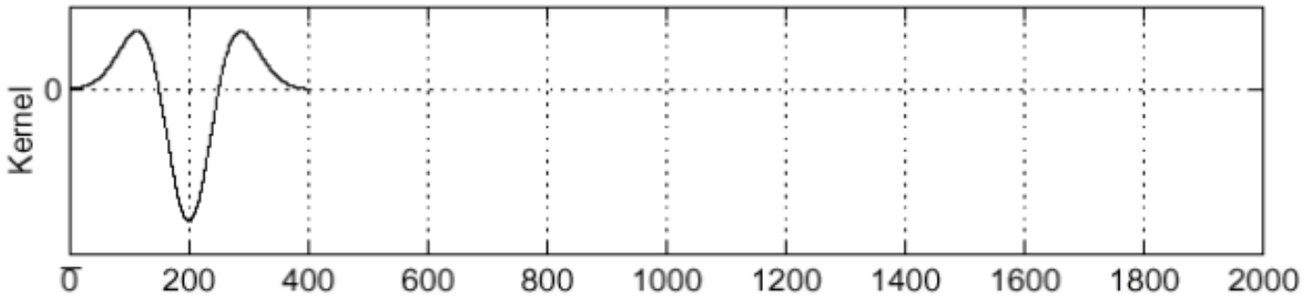
A bit more on edge detection

f



Edge

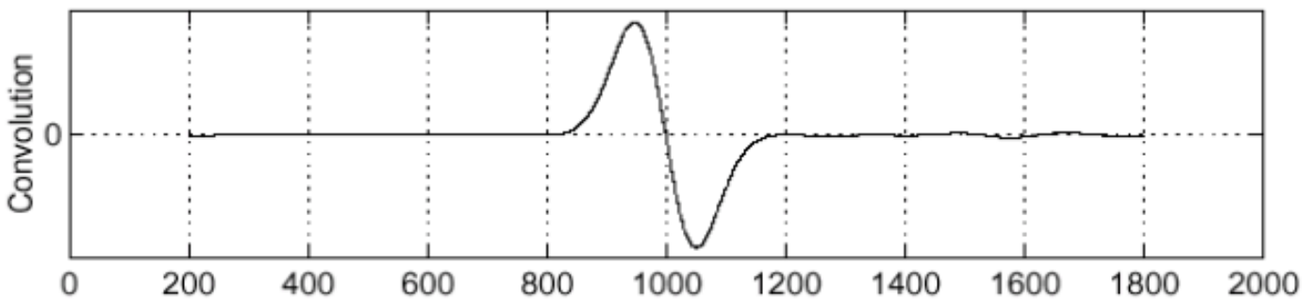
$\frac{d^2}{dx^2} g$



Second Derivative of Gaussian (Laplacian)

Mexican hat filter

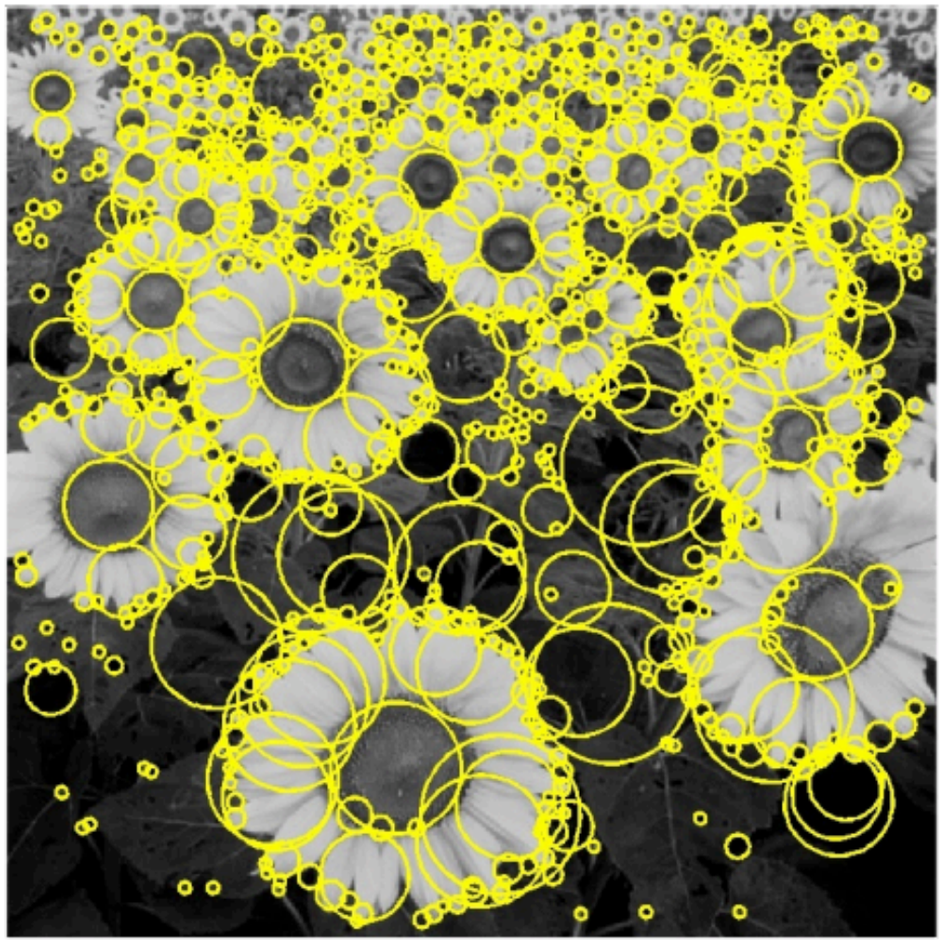
$f * \frac{d^2}{dx^2} g$

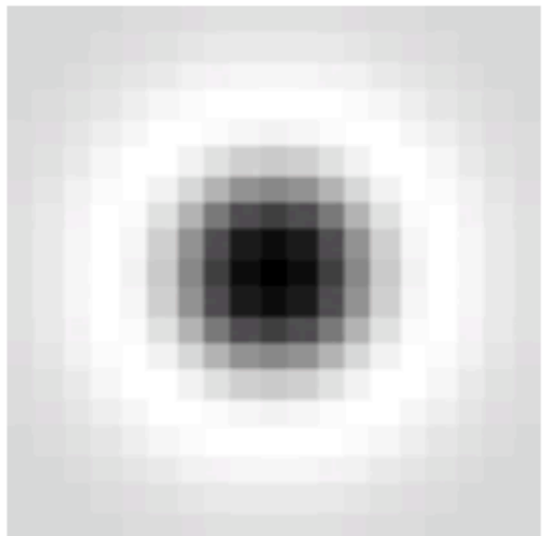


Edge = zero crossing of second derivative

Blob detection: basic idea

- To detect “blobs”, convolve the image with a blob filter at multiple scales and look for extrema of filter response in the resulting *scale space*

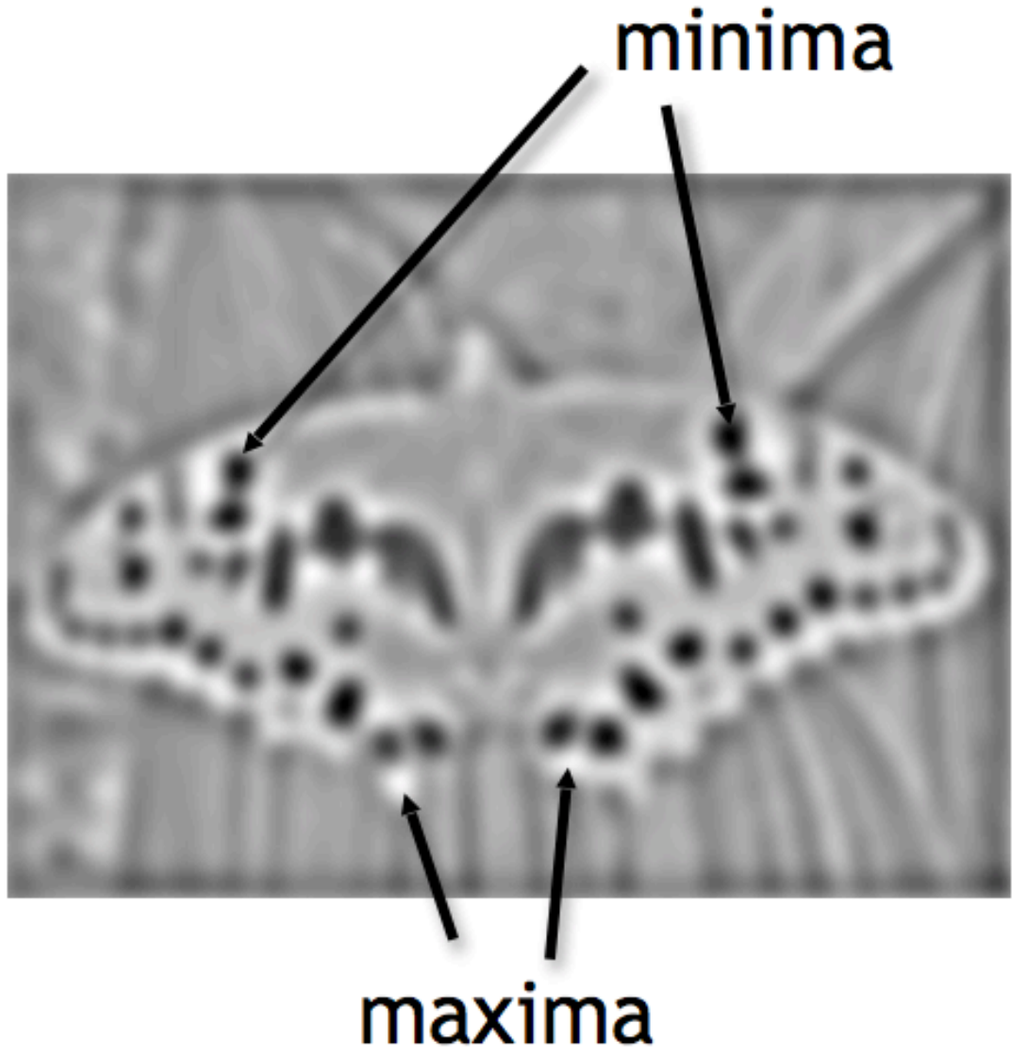




Blob filter

Blob detection: basic idea

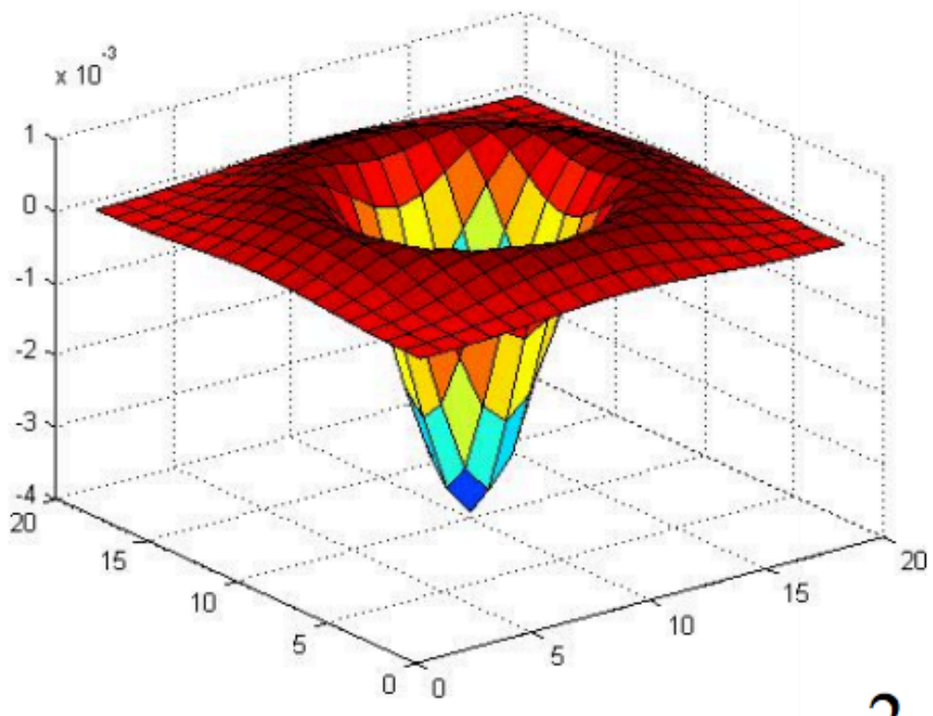
- Find maxima and minima of blob filter response in space and scale

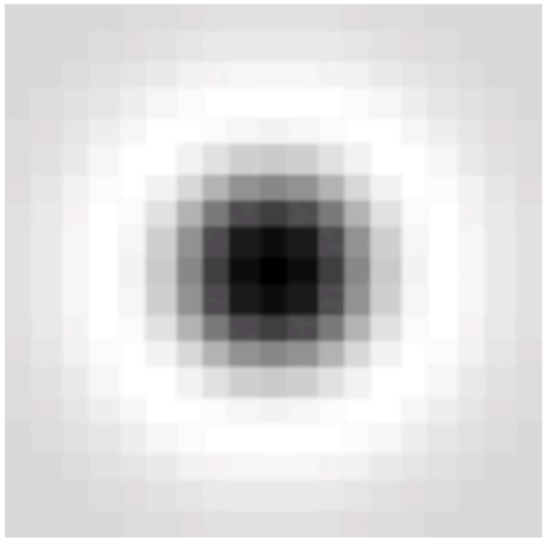


Slide credit: N. Snavely

Blob filter

- **Laplacian of Gaussian (LoG)**: circularly symmetric operator for blob detection in 2D





$$\nabla^2 g = \frac{\partial^2 g}{\partial x^2} + \frac{\partial^2 g}{\partial y^2}$$

Scale invariant detection

- Functions for determining scale: $f = \text{Kernel} * \text{Image}$

Kernels:

$$L = \sigma^2 \left(G_{xx}(x, y, \sigma) + G_{yy}(x, y, \sigma) \right)$$

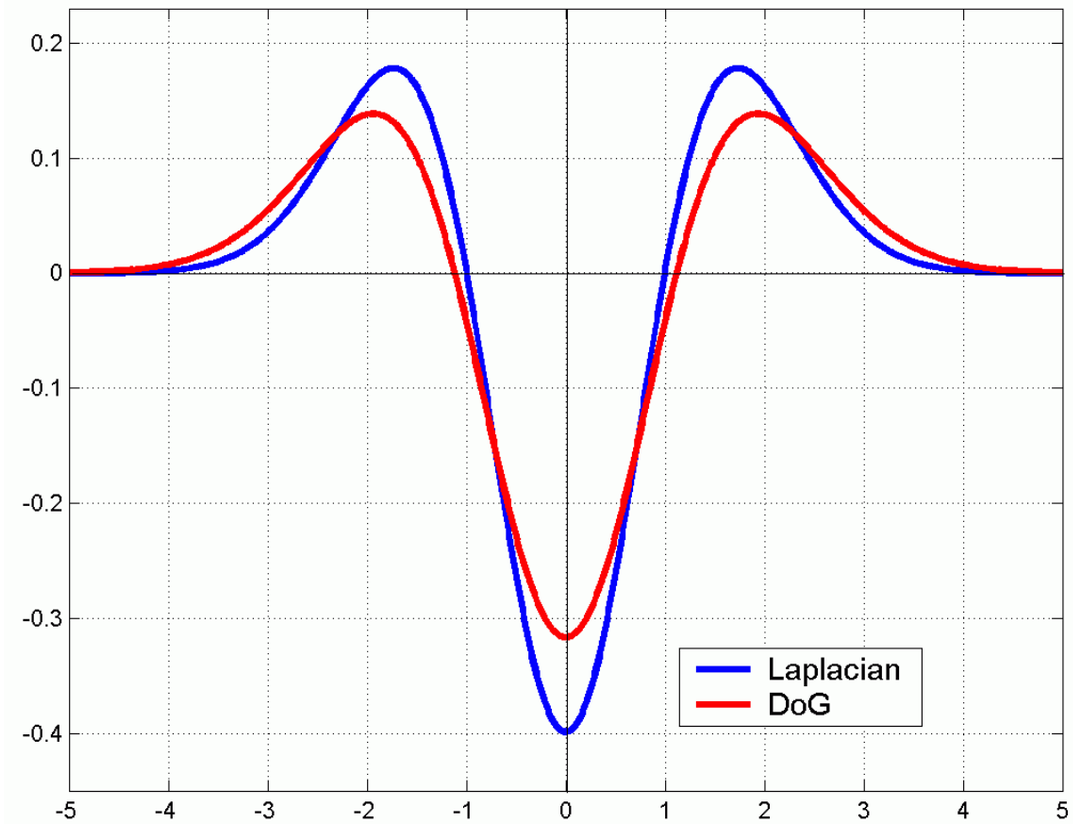
(Laplacian)

$$DoG = G(x, y, k\sigma) - G(x, y, \sigma)$$

(Difference of Gaussians)

where Gaussian

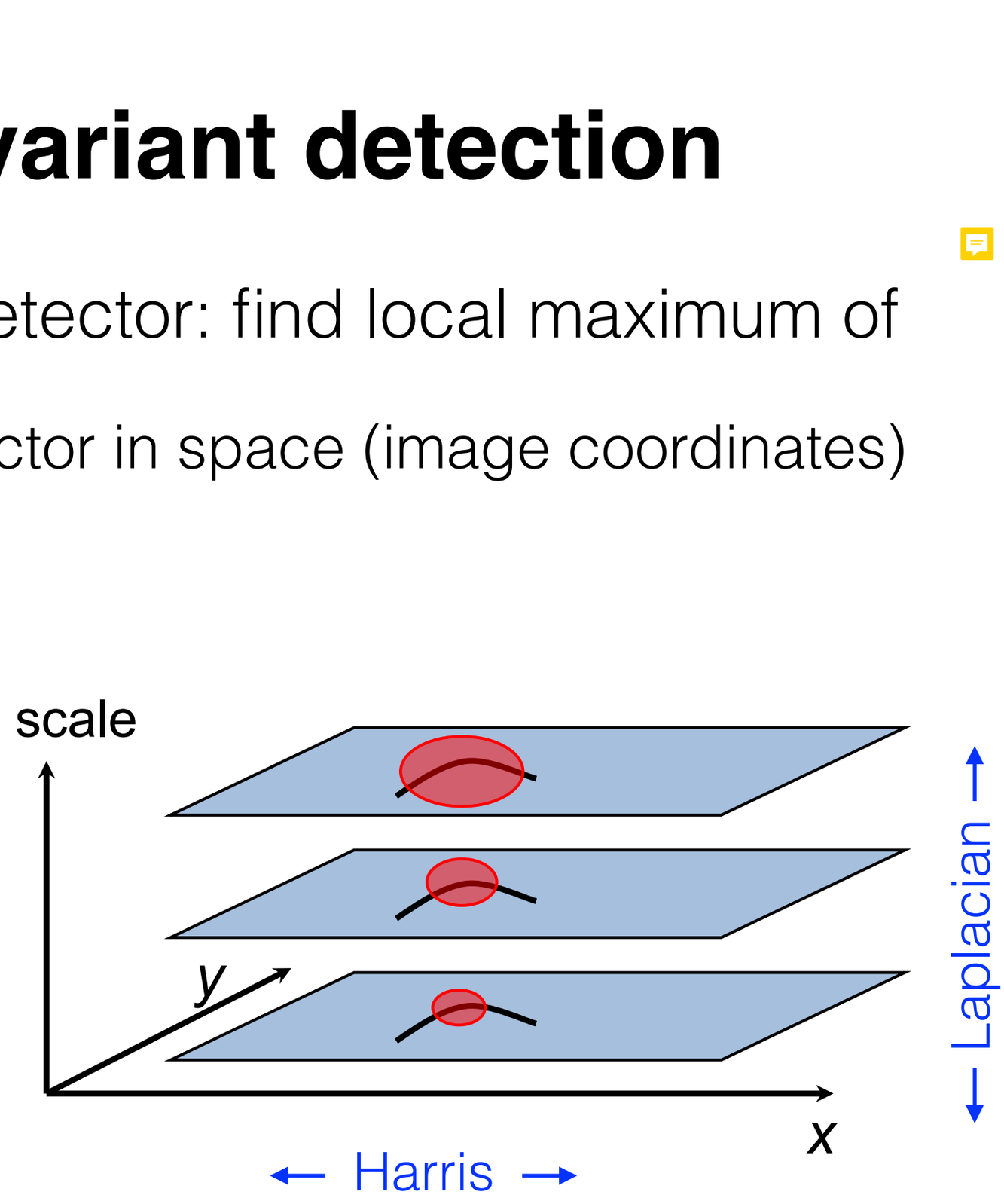
$$G(x, y, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2 + y^2}{2\sigma^2}}$$



Note: both kernels are invariant to scale and rotation

Scale invariant detection

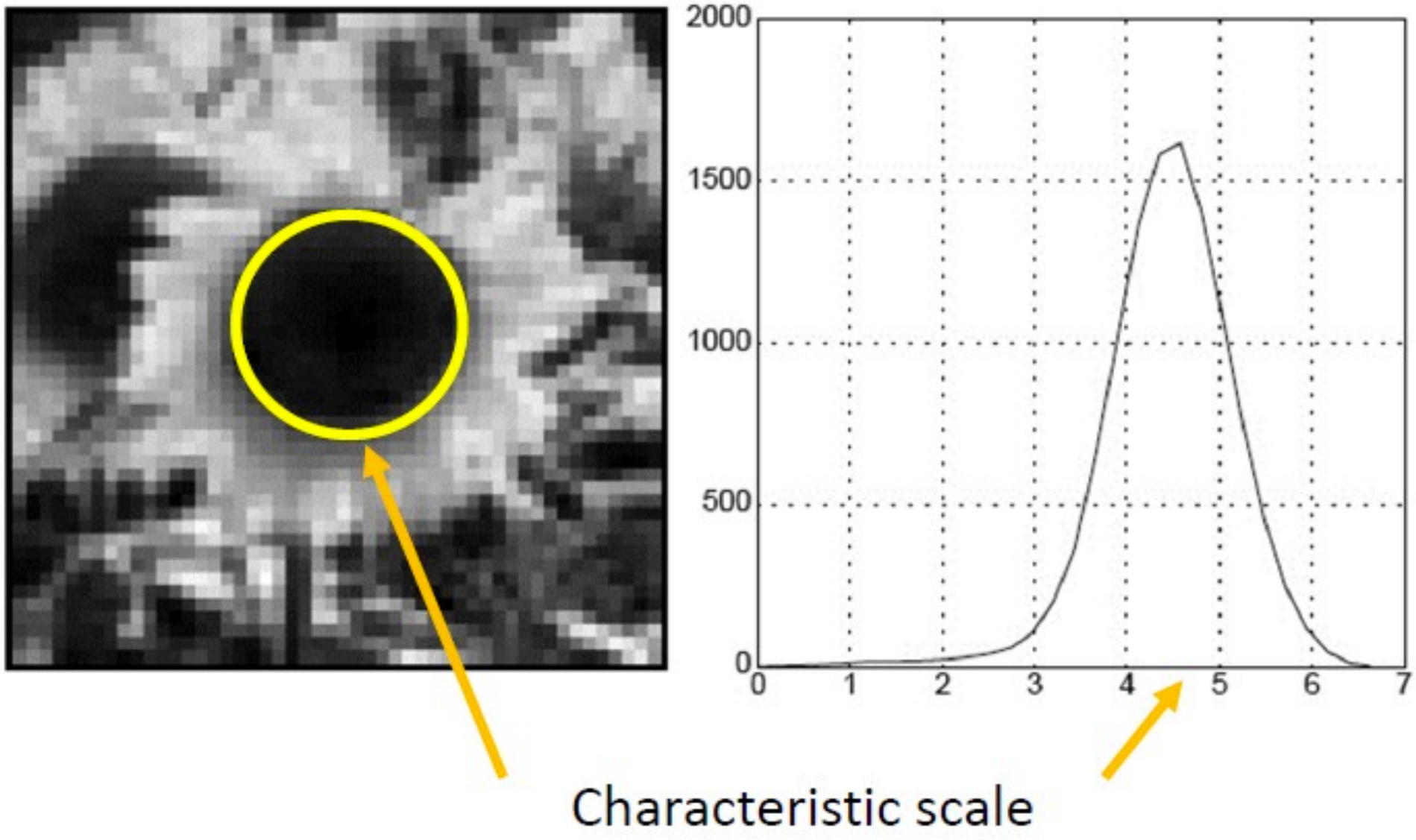
- **Harris-Laplacian** detector: find local maximum of
  - Harris corner detector in space (image coordinates)
  - Laplacian in scale



K.Mikolajczyk, C.Schmid, “Indexing Based on Scale Invariant Interest Points”, ICCV 2001

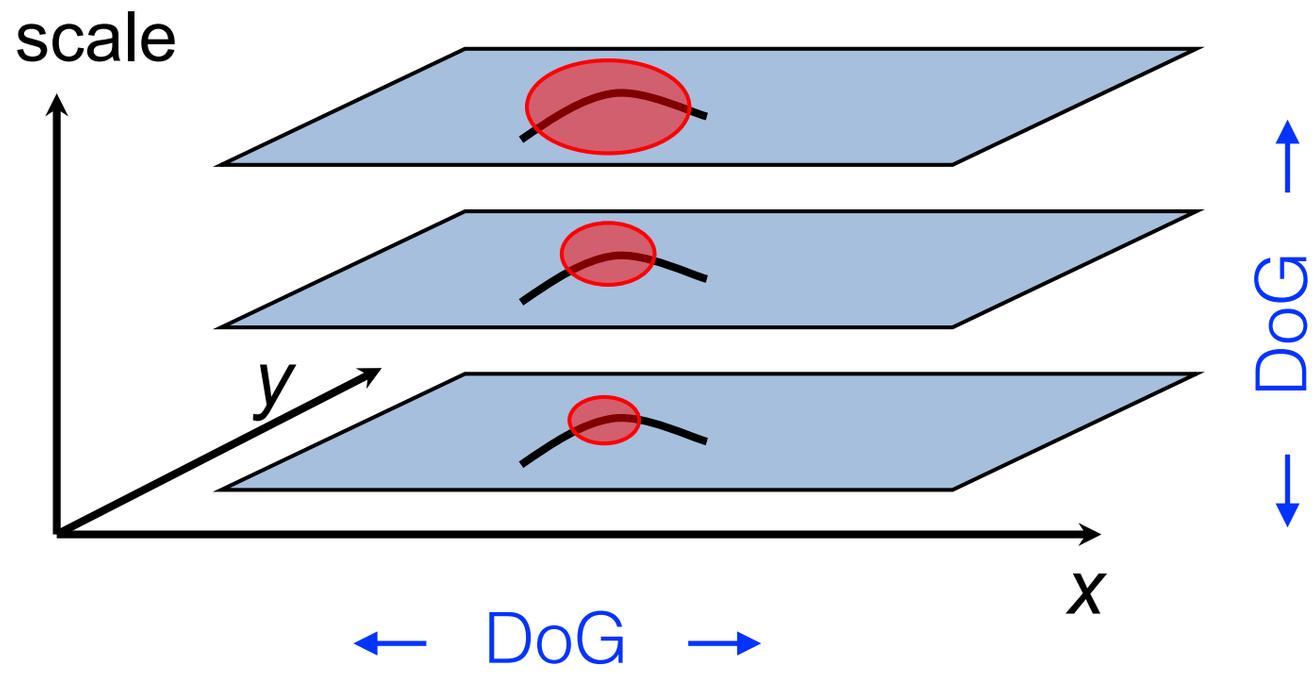
Scale invariant detection

- An example (Laplacian):



Scale invariant detection

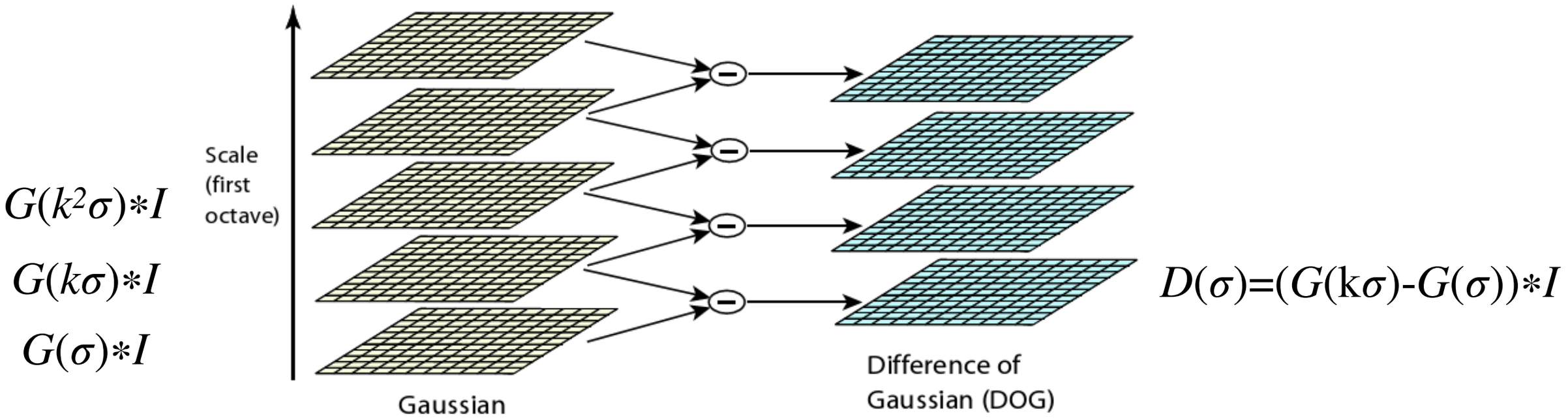
- **SIFT** detector: find local maximum of
  - Difference of Gaussians in space
  - Difference of Gaussians in scale

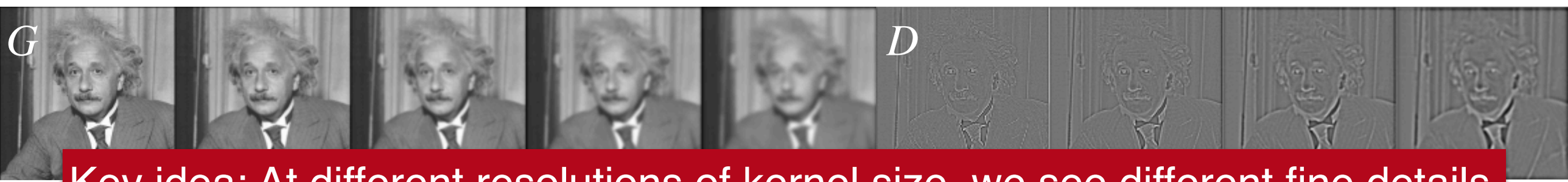


D. Lowe, “Distinctive image features from scale-invariant keypoints”, IJCV 2004

Difference of Gaussians (SIFT detector)

- Efficient approximation of LoG

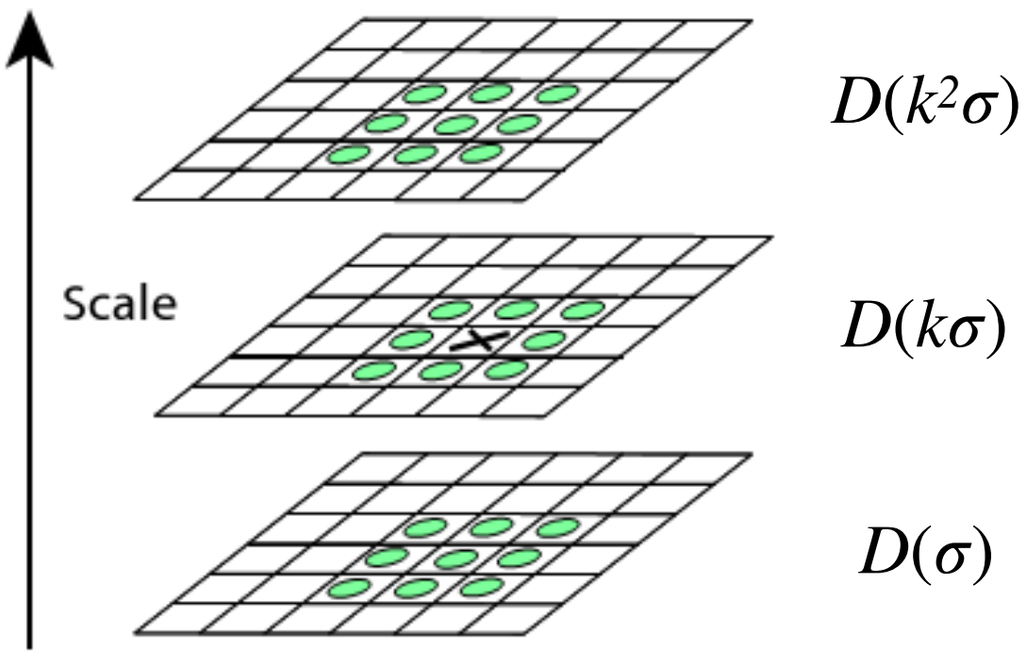




Key idea: At different resolutions of kernel size, we see different fine details of the image. In other words, we can capture keypoints at varying scales.

Difference of Gaussians (SIFT detector)

- Keypoint are selected as scale-space extrema
  - choose all extrema within 3x3x3 neighborhood



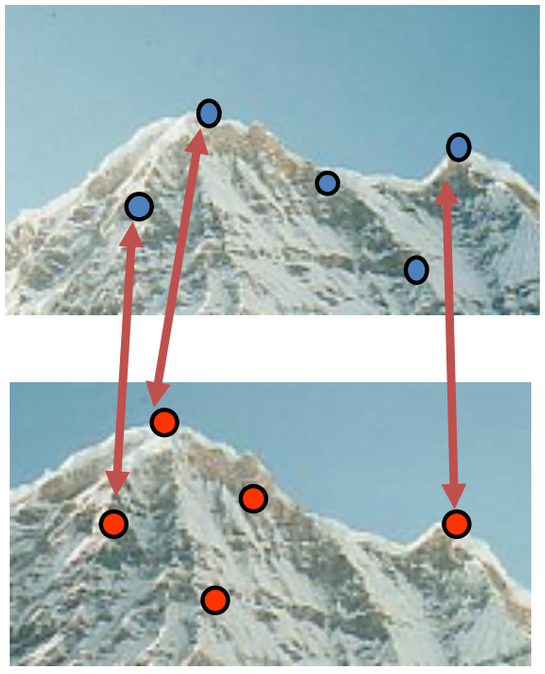
X is selected if it is larger or smaller than all 26 neighbors

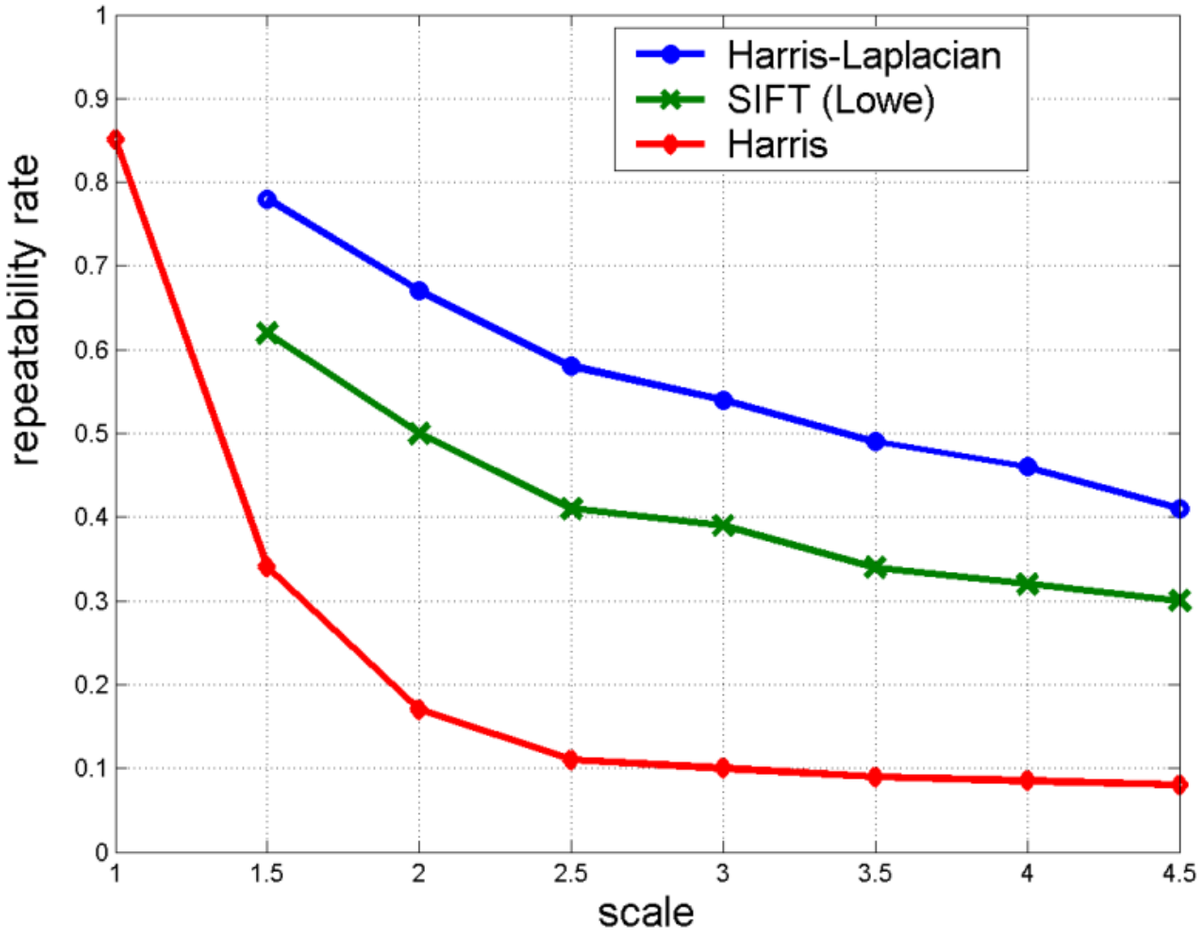
Scale Invariant Detectors

- Experimental evaluation of detectors w.r.t. scale change

Repeatability rate:

$$\frac{\#\text{ correspondences}}{\#\text{ possible correspondences}}$$





K. Mikolajczyk, C. Schmid, “Indexing Based on Scale Invariant Interest Points”, ICCV 2001

Scale Invariant Detectors

- Scale invariant detection summary:
  - **Given:** two images of the same scene with a large scale difference between them
  - **Goal:** find the same interest points independently in each image
  - **Solution:** search for maxima of suitable functions in scale and in space (over the image)

Popular methods:

1. Harris-Laplacian [Mikolajczyk, Schmid]: maximize Laplacian of Gaussian over scale, Harris' measure of corner response over the image
2. SIFT [Lowe]: maximize Difference of Gaussians over scale and space

Scale Invariant Detectors

- Actually there are many other options…
- List/comparison of keypoint detectors:

Feature Detector	Corner	Blob	Region	Rotation invariant	Scale invariant	Affine invariant	Repeatability	Localization accuracy	Robustness	Efficiency
Harris	√			√			+++	+++	+++	++
Hessian		√		√			++	++	++	+
SUSAN	√			√			++	++	++	+++
Harris-Laplace	√	(√)		√	√		+++	+++	++	+
Hessian-Laplace	(√)	√		√	√		+++	+++	+++	+
DoG	(√)	√		√	√		++	++	++	++
SURF	(√)	√		√	√		++	++	++	+++
Harris-Affine	√	(√)		√	√	√	+++	+++	++	++
Hessian-Affine	(√)	√		√	√	√	+++	+++	+++	++
Salient Regions	(√)	√		√	√	(√)	+	+	++	+
Edge-based	√			√	√	√	+++	+++	+	+
MSER			√	√	√	√	+++	+++	++	+++
Intensity-based			√	√	√	√	++	++	++	++
Superpixels			√	√	(√)	(√)	+	+	+	+

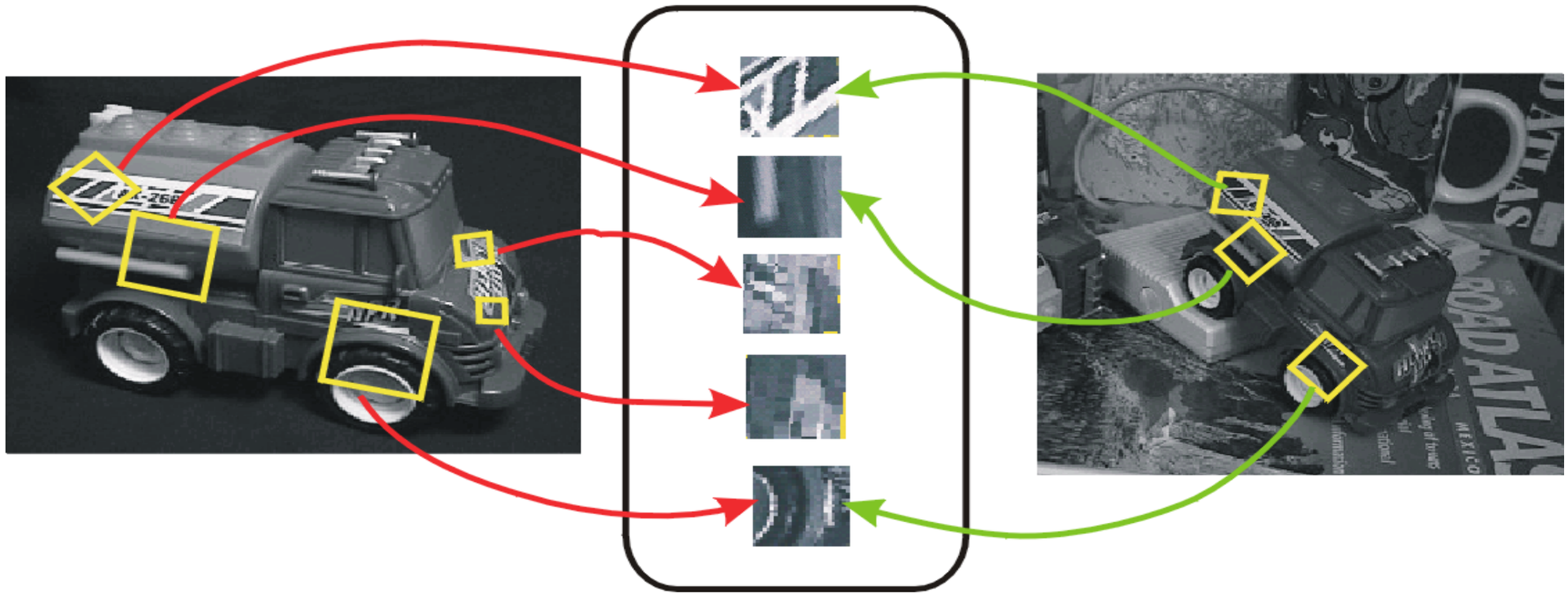
Source: T. Tuytelaars, K. Mikolajczyk

From feature detection to description

- Scaled and rotated versions of the same neighborhood will give rise to blobs that are related by the same transformation
- What to do if we want to compare the appearance of these image regions?
  - Normalization: transform these regions into same-size circles
  - Problem: **rotational ambiguity**

Invariant local features

- Image content is transformed into local feature that are invariant to *translation*, *rotation*, *scale*, and other imaging parameters



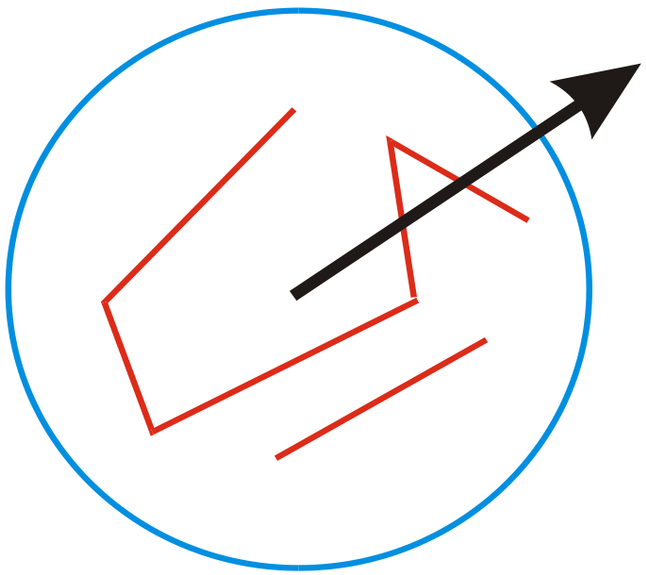
Reference: CVPR 2003 Tutorial on “Recognition and Matching Based on Local Invariant Features” by David Lowe

Desired properties of feature descriptors

- Easily compared (compact, fixed-dimensional)
- Invariant:
  - Translation
  - Rotation
  - Scale
  - Change in image brightness (illumination)
  - *Change in perspective?*

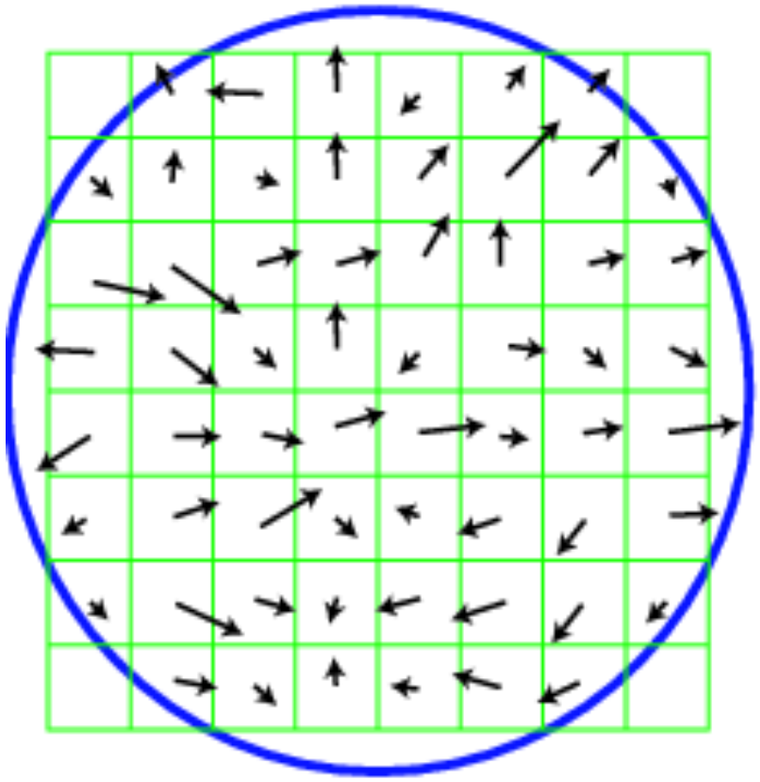
SIFT descriptors

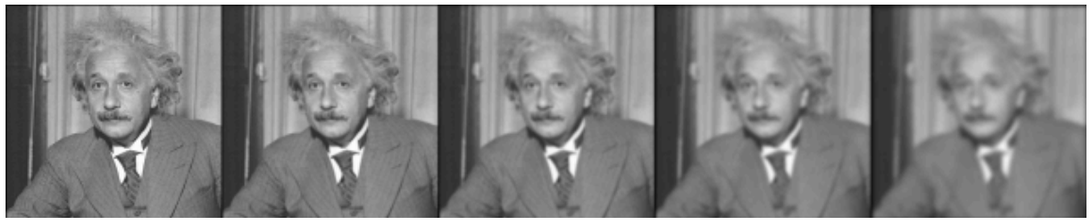
- We are given a keypoint and its scale from DoG
- To assign the orientation to circular image windows:
  - Create histogram of local gradient directions in the patch
  - Then we select a *characteristic orientation* for the keypoint based on the most prominent gradient (see next slides)
  - We will describe all features relative to this orientation



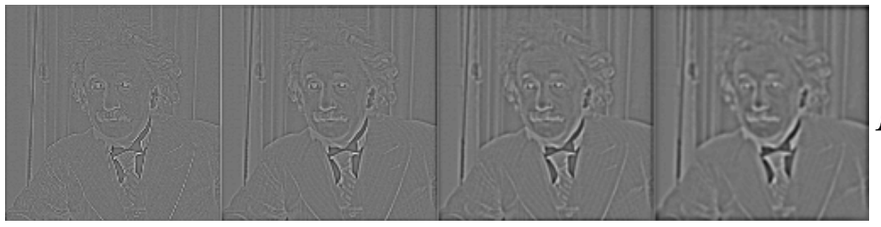
If the keypoint appears rotated in another image, the features will be the same, because they're relative to the characteristic orientation

SIFT descriptor formation





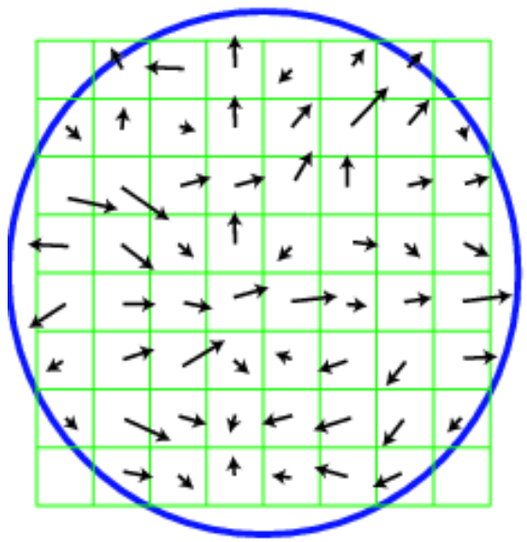
$L(x, y, \sigma)$

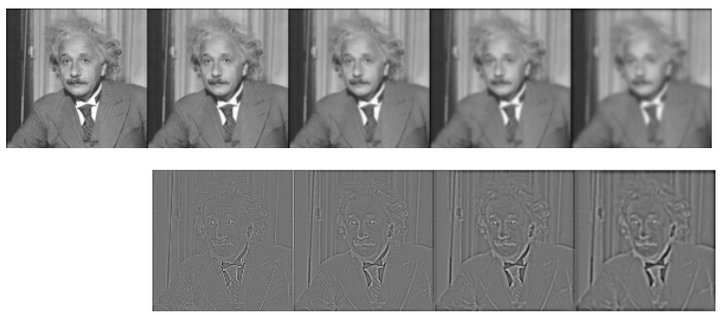


$D(x, y, \sigma)$

- Use the blurred image associated with the keypoint's scale
- Take image gradients over the keypoint neighborhood
- To become rotation invariant, rotate the gradient directions AND locations by keypoint orientation

SIFT descriptor formation





$L(x, y, \sigma)$

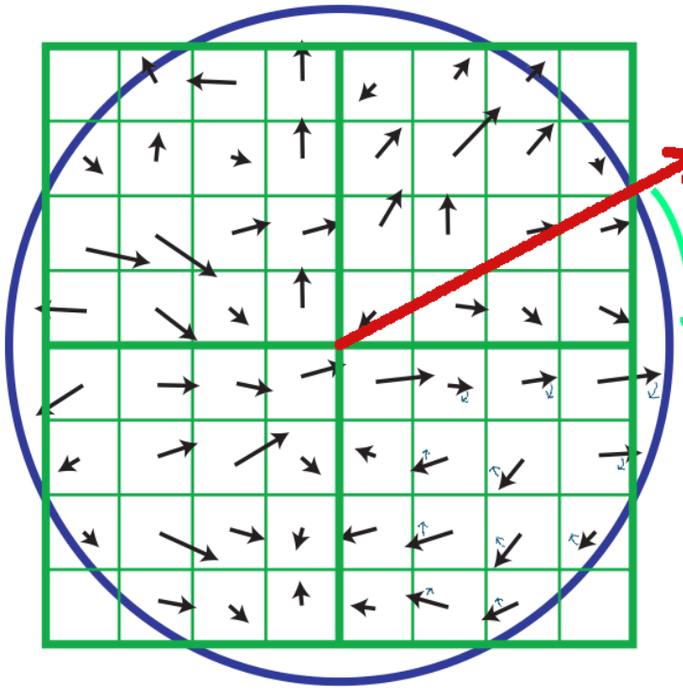
$D(x, y, \sigma)$

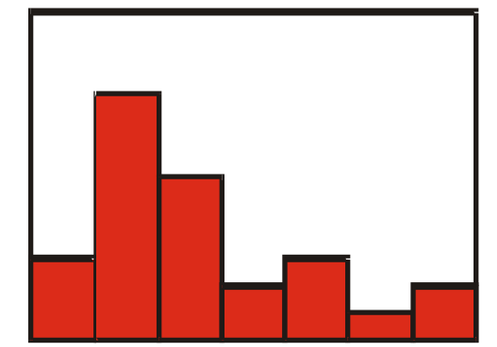
$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2}$$

$$\theta(x,y) = \text{atan2}(L(x,y+1) - L(x,y-1), L(x+1,y) - L(x-1,y))$$

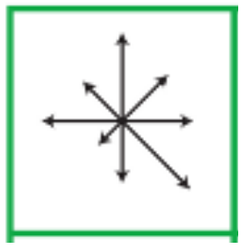
- Use the blurred image associated with the keypoint's scale
- Take image gradients over the keypoint neighborhood
- To become rotation invariant, rotate the gradient directions AND locations by keypoint orientation

SIFT descriptor formation





orientation
(peak in the histogram)



orientation histogram
(discretization: 8 bins)

- Use the blurred image associated with the keypoint's scale
- Take image gradients over the keypoint neighborhood
- To become rotation invariant, rotate the gradient directions AND locations by keypoint orientation

SIFT descriptor formation

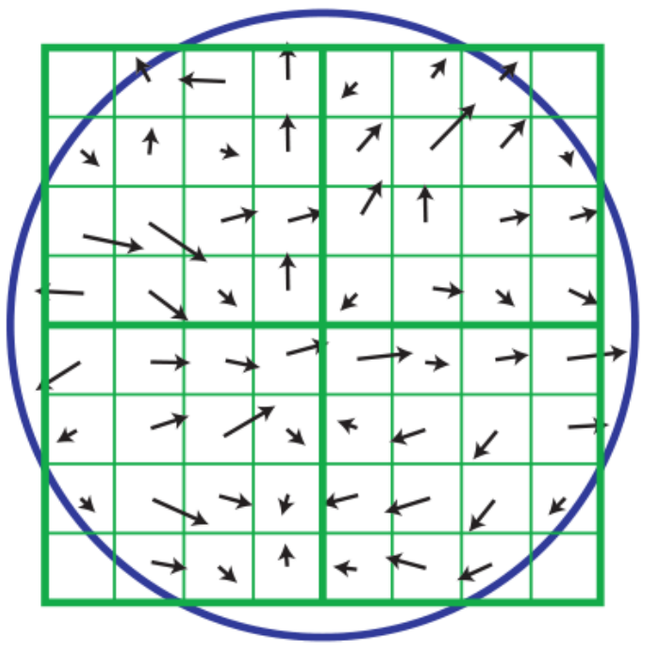
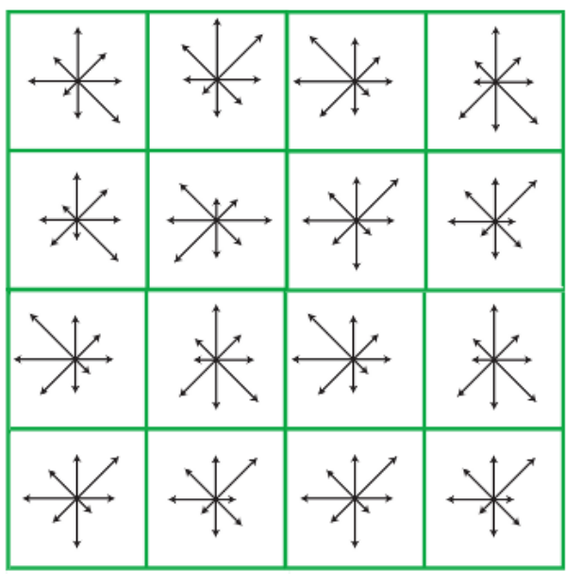


Image gradients

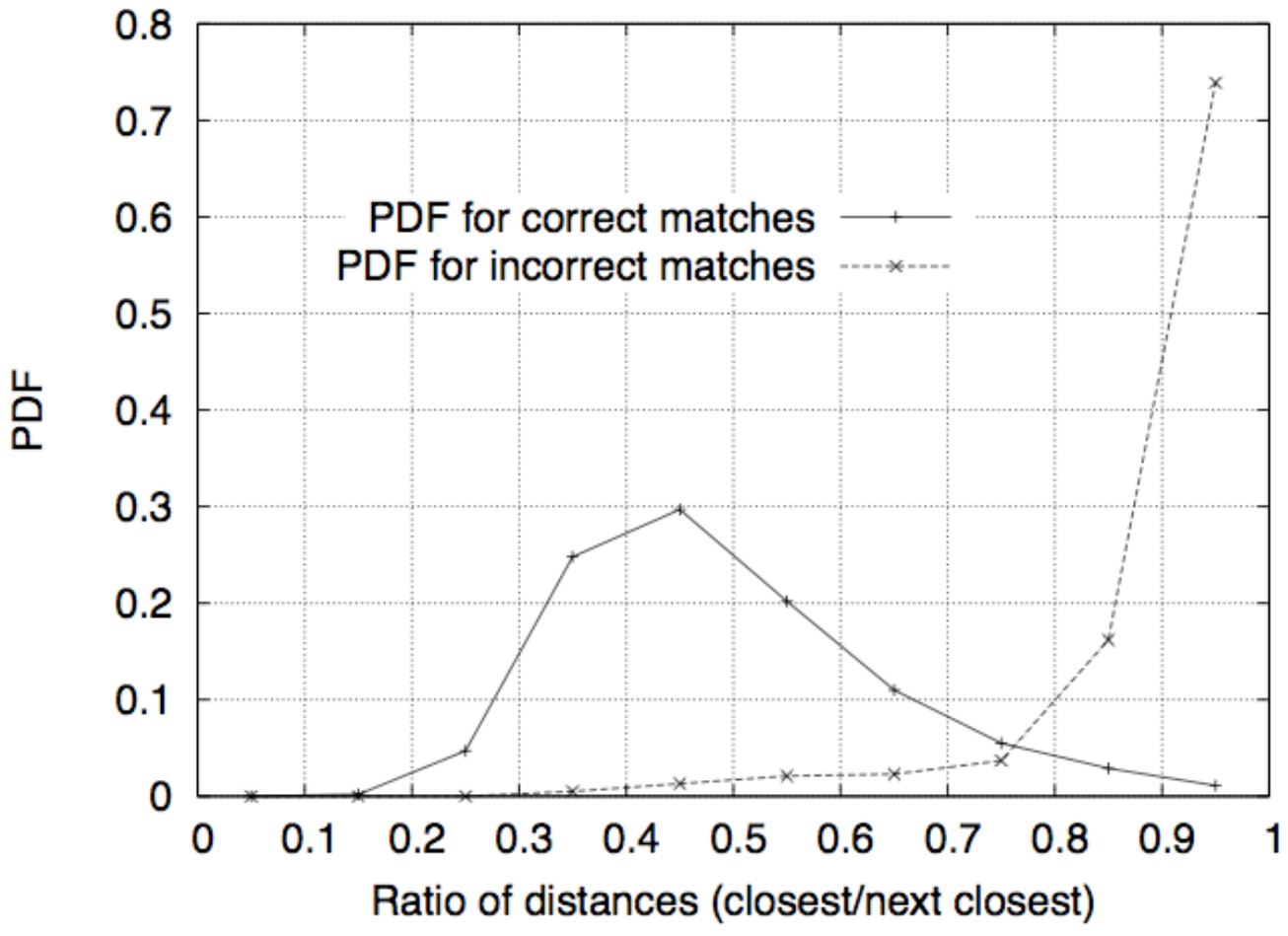


Keypoint descriptor

- 8 orientation bins per histogram and a 4x4 histogram array yields 8x4x4=128 numbers
- So a SIFT descriptor is a 128-d vector that is invariant to rotation (because we rotated the descriptor) and scale (because of DoG)
- We can now compare each vector from image *A* to each vector from image *B* to find matching keypoints

SIFT matching

- Euclidean distance between SIFT descriptor gives a good measure of keypoint similarity
- Ratio of distances is more reliable for matching



The probability that a match is correct can be determined by taking the ratio of distance from the closest neighbor to the distance of the second closest

(plot: PDF obtained using a database of approx. 40,000 keypoints)

Image credit: D. Lowe

SIFT matching

- An example:

Summary

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