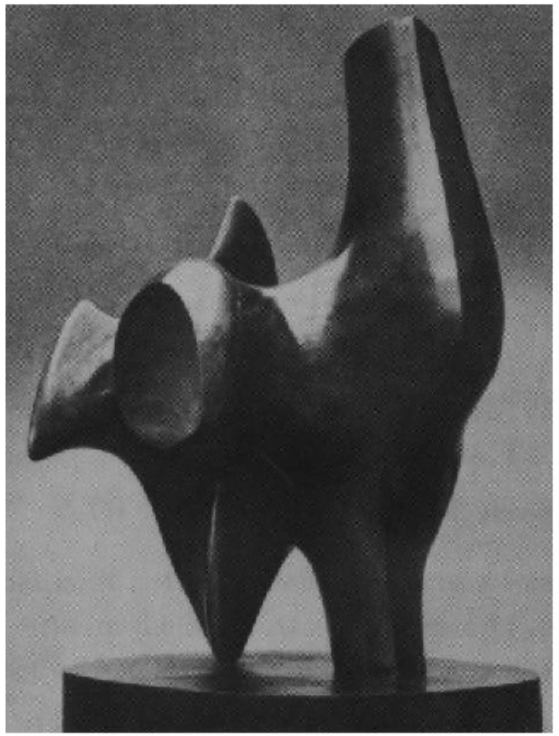
WPI

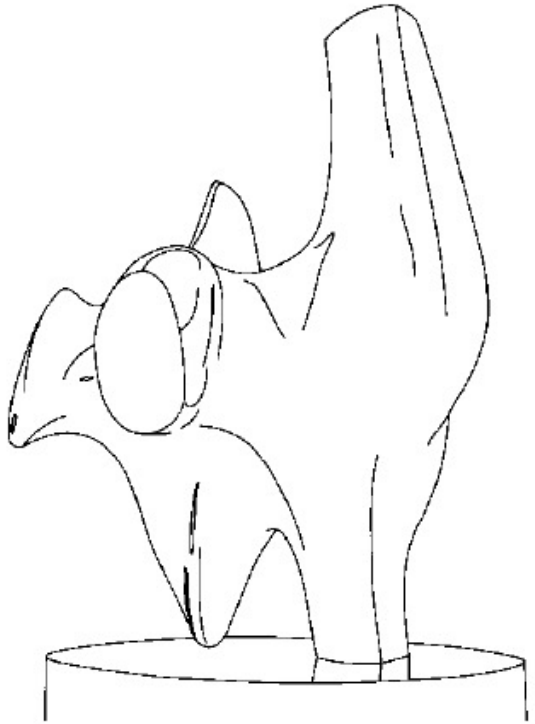
Lecture 12 - Vision

Image gradients and edges
Feature Extraction and Detection

Alexandros Lioulemes, PhD

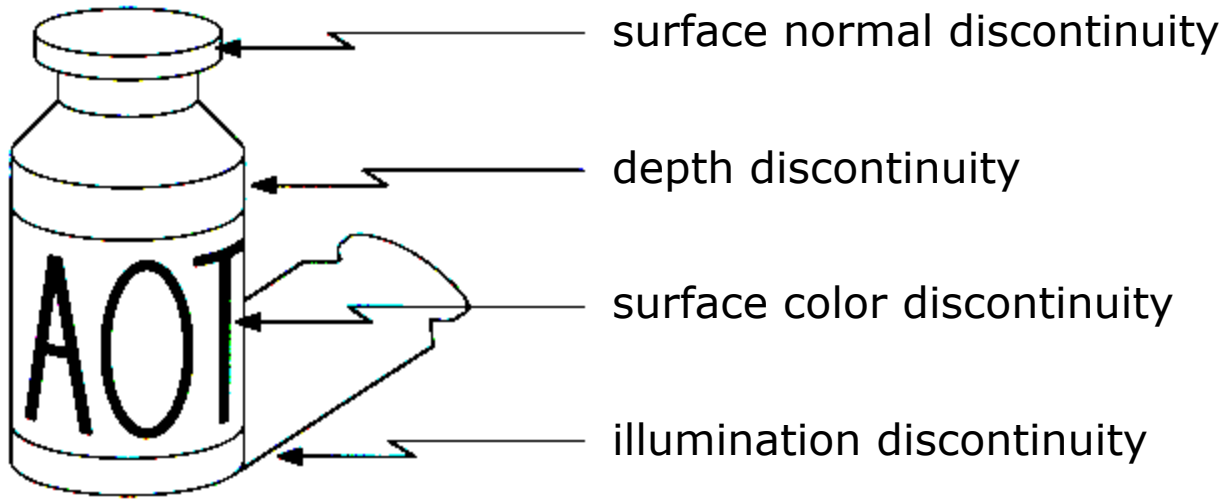
Edge detection





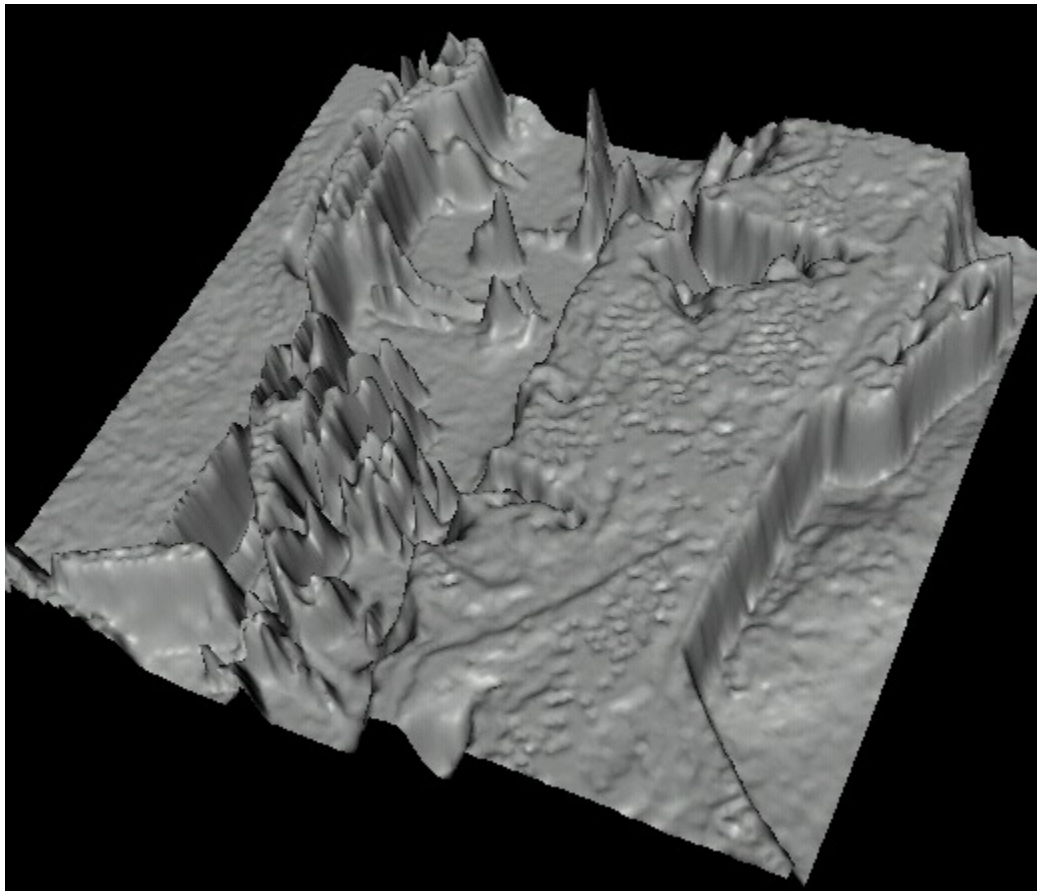
- Convert a 2D image into a set of curves
  - Extracts salient features of the scene
  - More compact than pixels

Origin of edges



- Edges are caused by a variety of factors

Images as function



- Edges look like steep cliffs

Characterizing edges

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

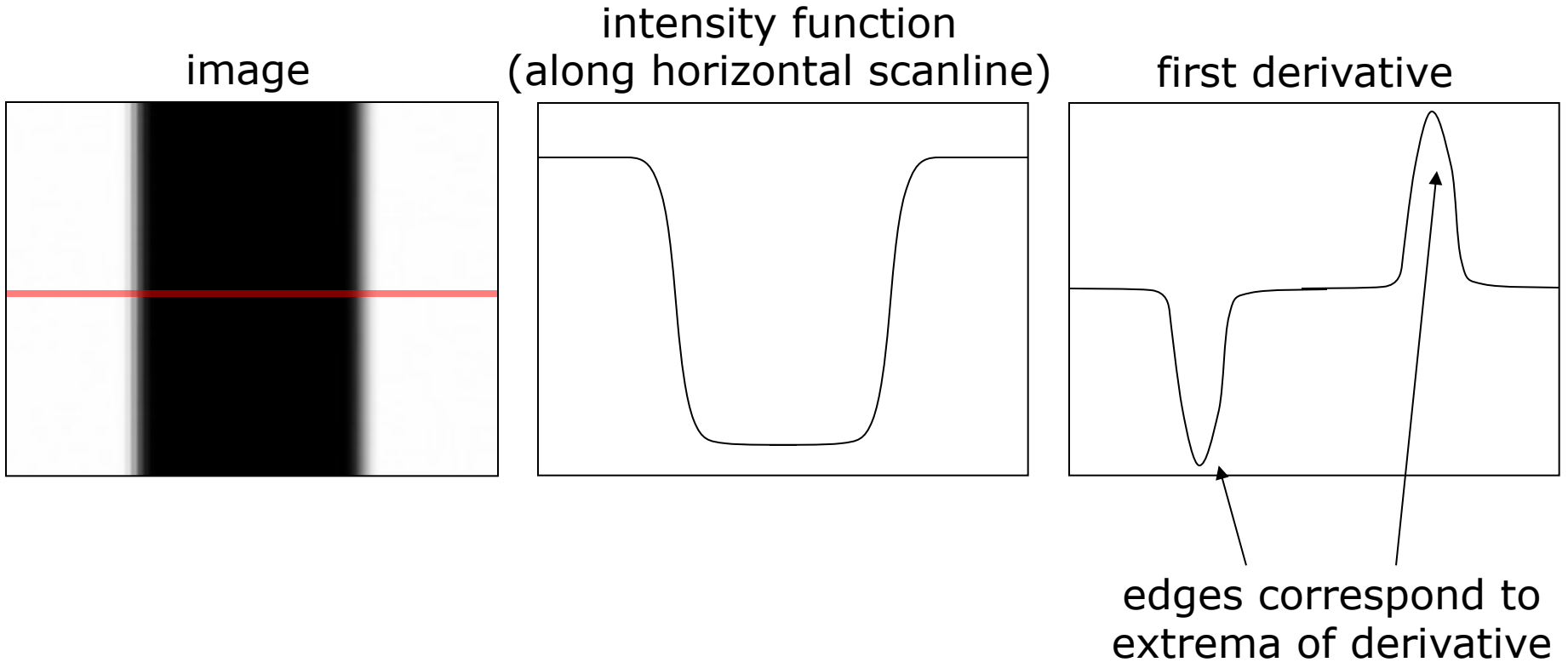


Image derivatives

- How can we differentiate a *digital* image F[x,y]?
  - Option 1: reconstruct a continuous image, *f*, then compute the derivative
  - Option 2: take discrete derivative (finite difference)

$$\frac{\partial f}{\partial x}[x,y] \approx F[x+1,y] - F[x,y]$$

How would you implement this as a linear filter?

$\frac{\partial f}{\partial x}$:

1	-1	

H_x

$\frac{\partial f}{\partial y}$:

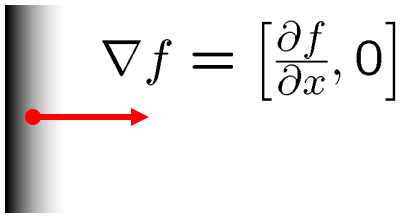
	-1	
	1	

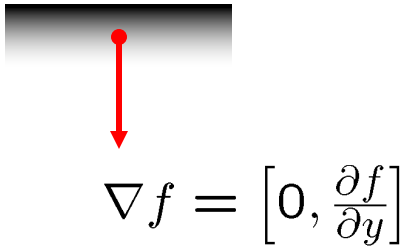
H_y

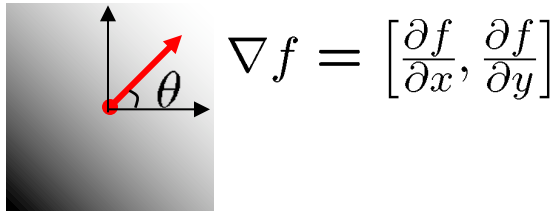
Image gradient

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

The gradient points in the direction of most rapid increase in intensity







The *edge strength* is given by the gradient magnitude:

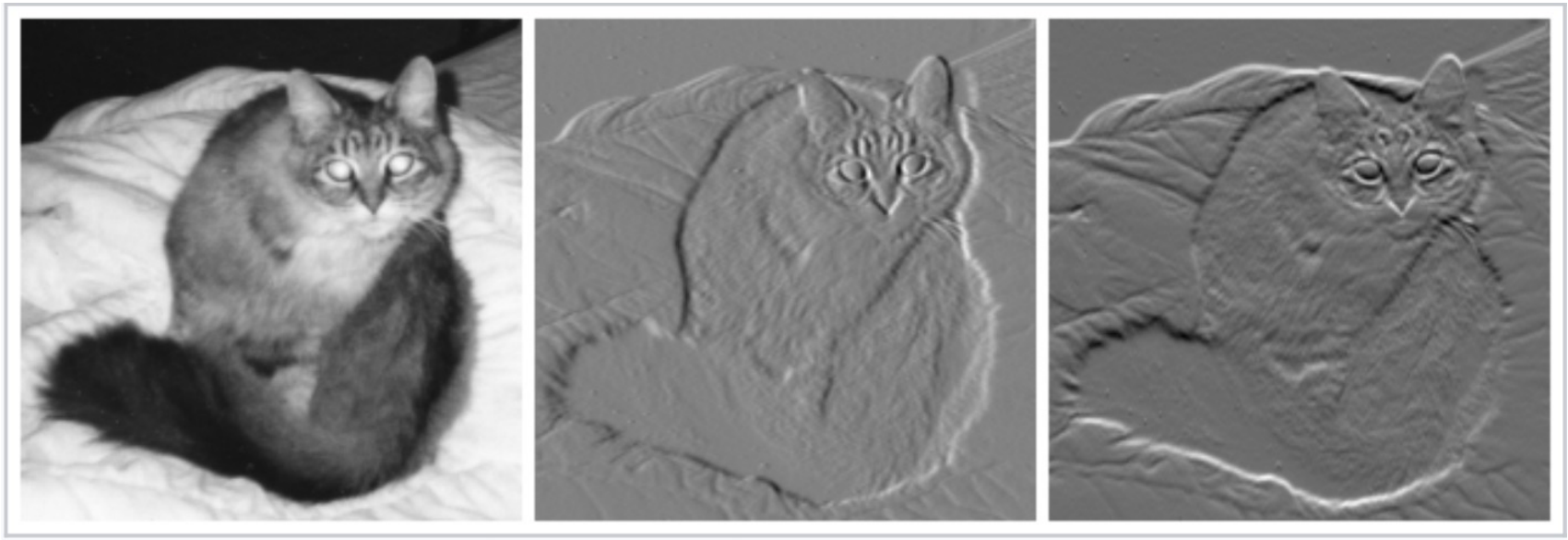
$$\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

The gradient direction is given by:

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

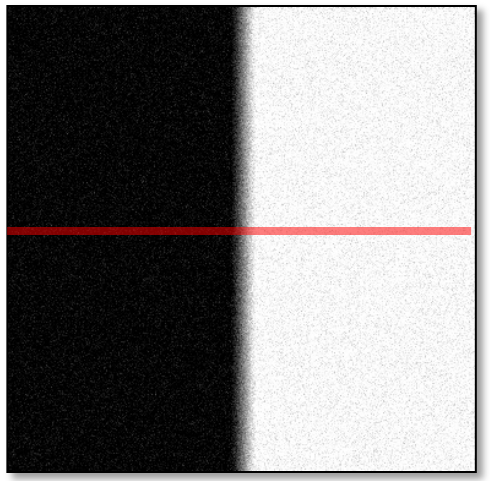
- how does this relate to the direction of the edge?

Image gradient



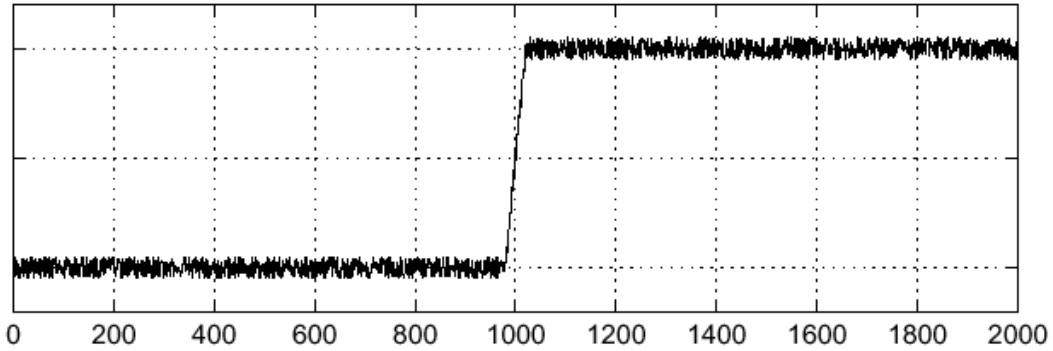
https://en.wikipedia.org/wiki/Image_gradient

Effects of noise

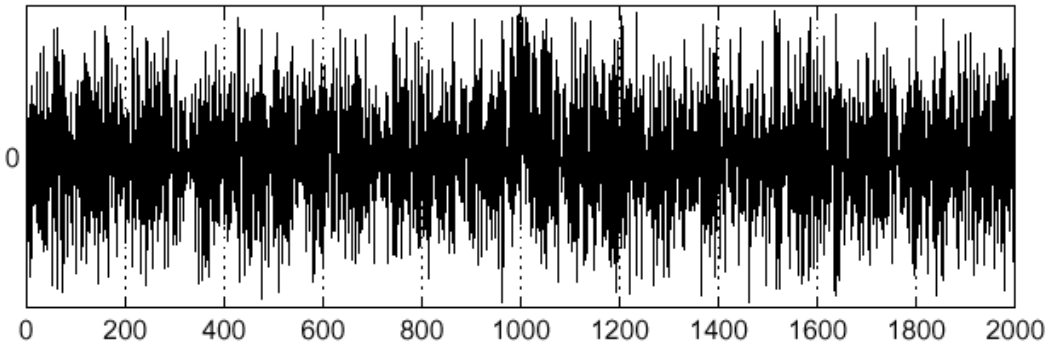


Noisy input image

$f(x)$

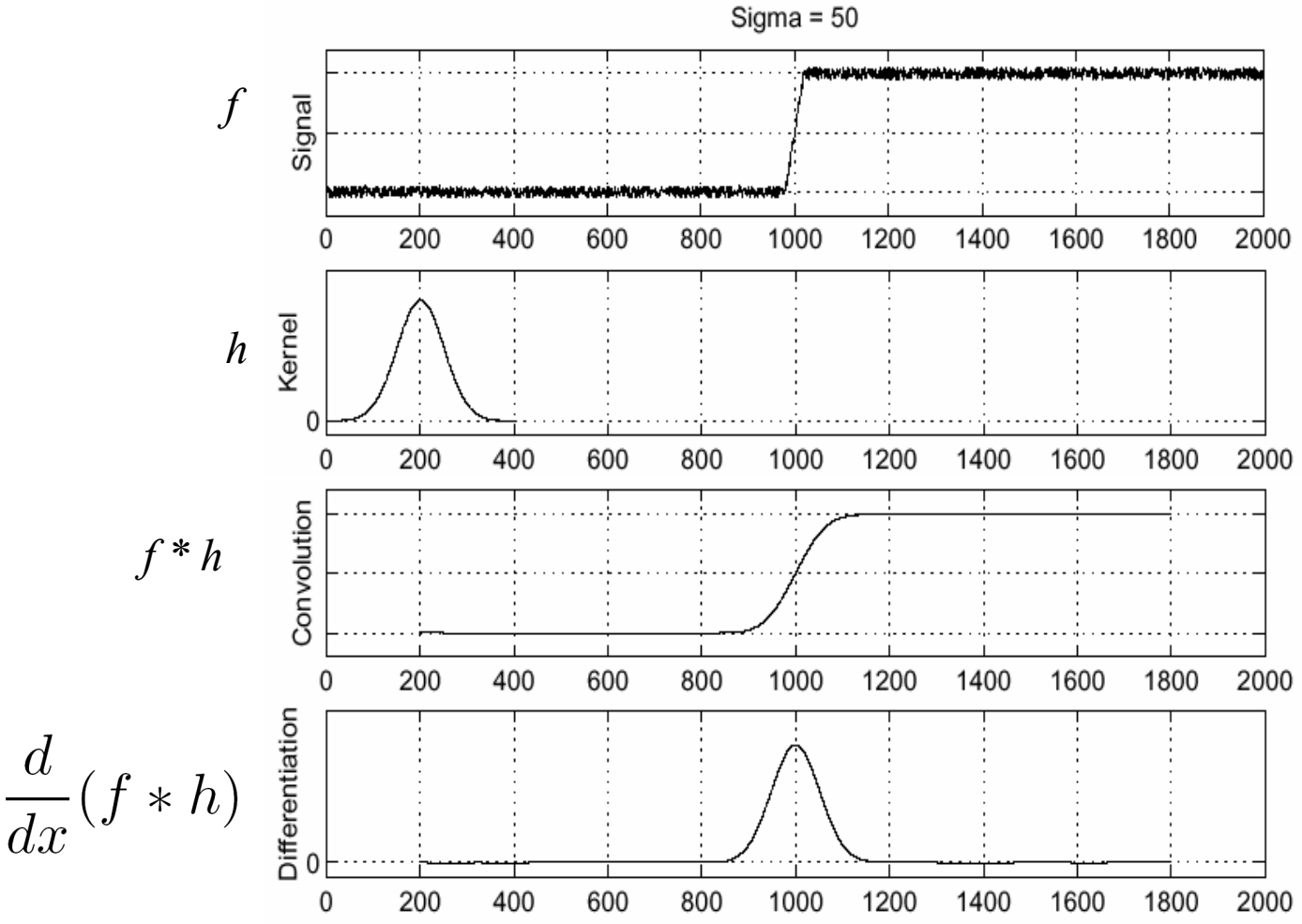


$\frac{d}{dx}f(x)$



Where is the edge?

Solution: smooth first



To find edges, look for peaks in $\frac{d}{dx}(f * h)$

Associative property of convolution

- Differentiation is convolution, and convolution is associative:

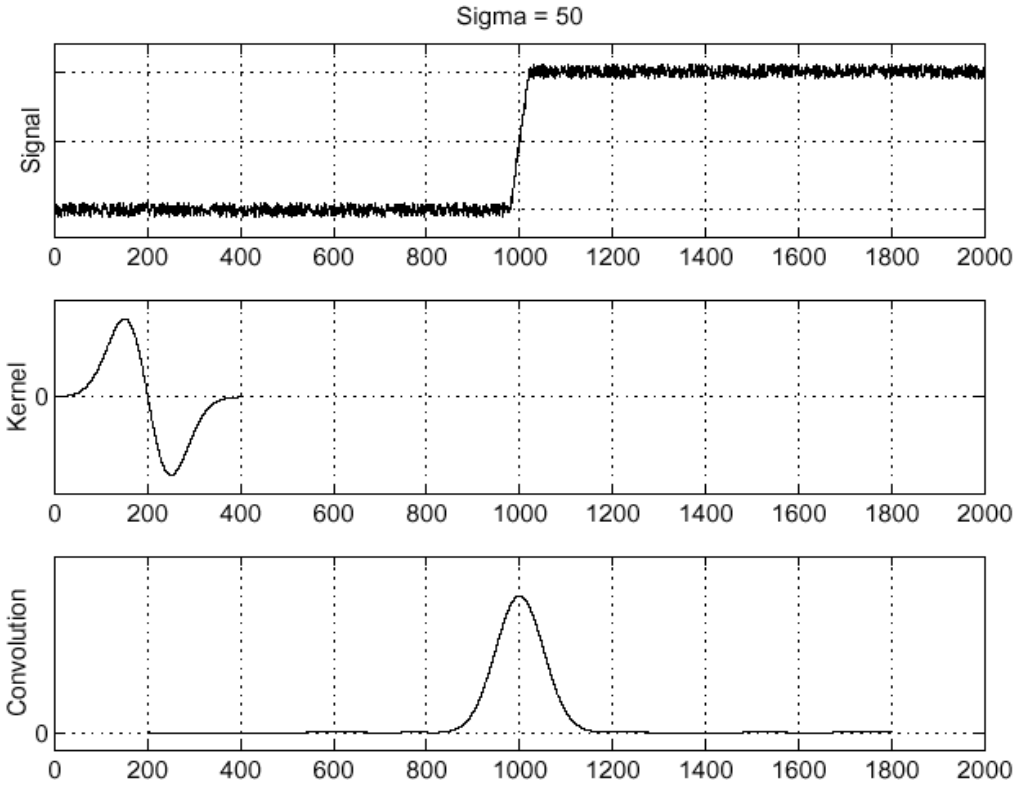
$$\frac{d}{dx}(f * h) = f * \frac{d}{dx}h$$

- This saves us one operation:

f

$\frac{d}{dx}h$

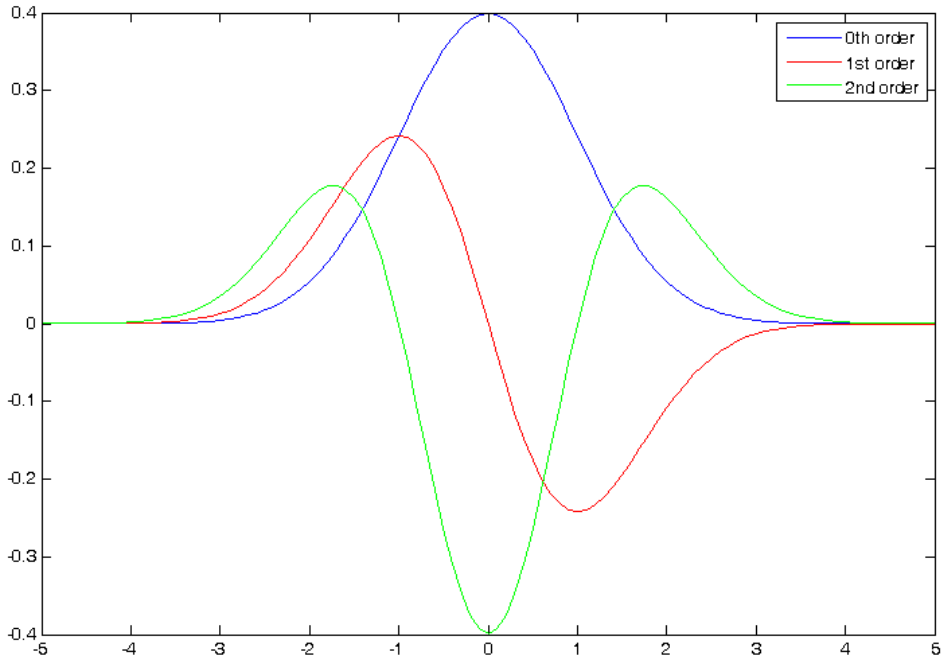
$f * \frac{d}{dx}h$



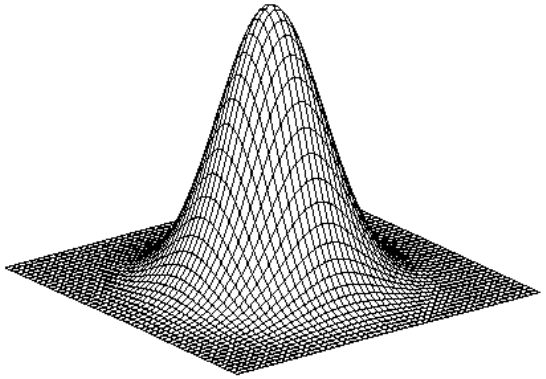
The 1D Gaussian and its derivatives

$$G_\sigma(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}}$$

$$G'_\sigma(x) = \frac{d}{dx} G_\sigma(x) = -\frac{1}{\sigma}\left(\frac{x}{\sigma}\right) G_\sigma(x)$$

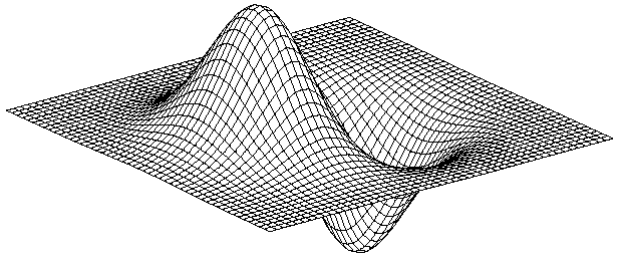


2D edge detection filters



Gaussian

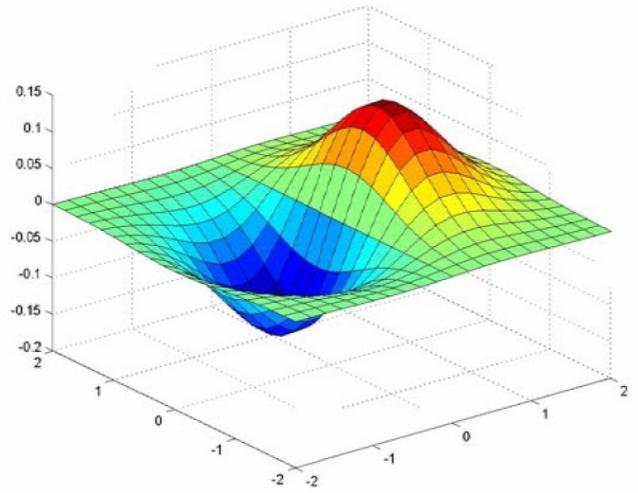
$$h_\sigma(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$



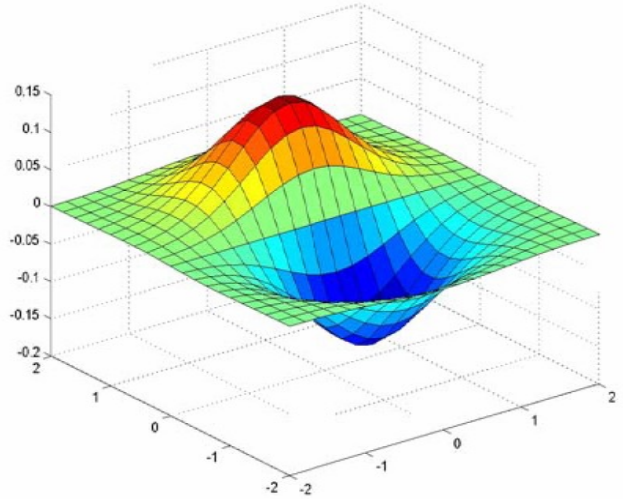
derivative of Gaussian (x)

$$\frac{\partial}{\partial x} h_\sigma(u,v)$$

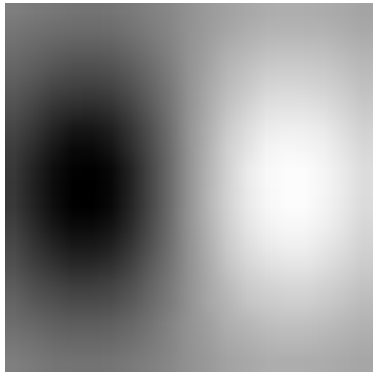
Derivative of Gaussian filter

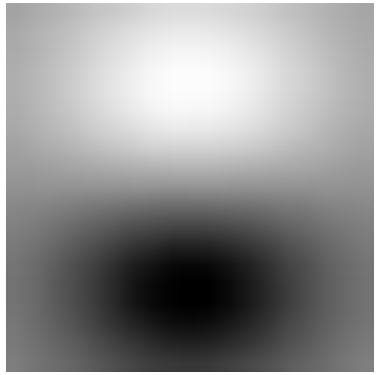


x-direction



y-direction





The Sobel operator

- Common approximation of derivative of Gaussian

$$\frac{1}{8}\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad s_x$$

$$\frac{1}{8}\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \qquad s_y$$

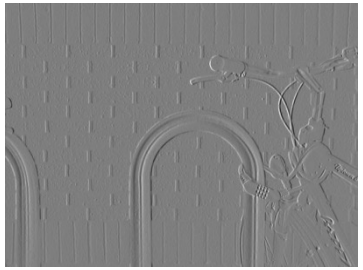
- The standard definition of the Sobel operator omits the 1/8 term
  - doesn't make a difference for edge detection
  - the 1/8 term **is** needed to get the right gradient magnitude

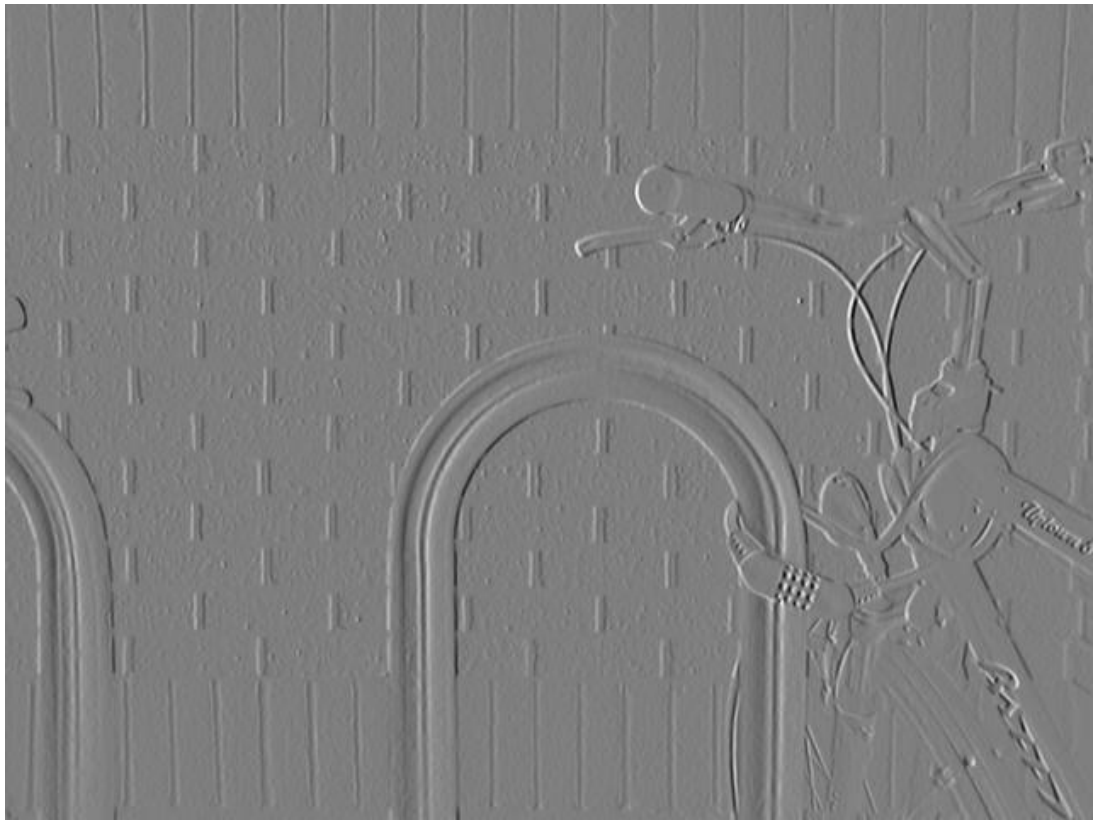
Sobel operator: example

https://en.wikipedia.org/wiki/Sobel_operator

Sobel operator: example

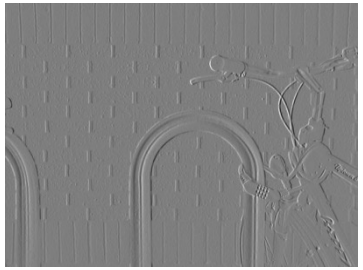
https://en.wikipedia.org/wiki/Sobel_operator

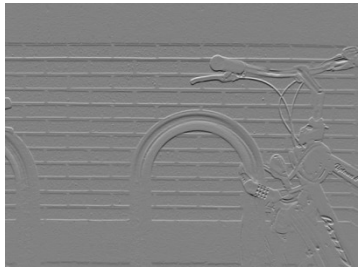


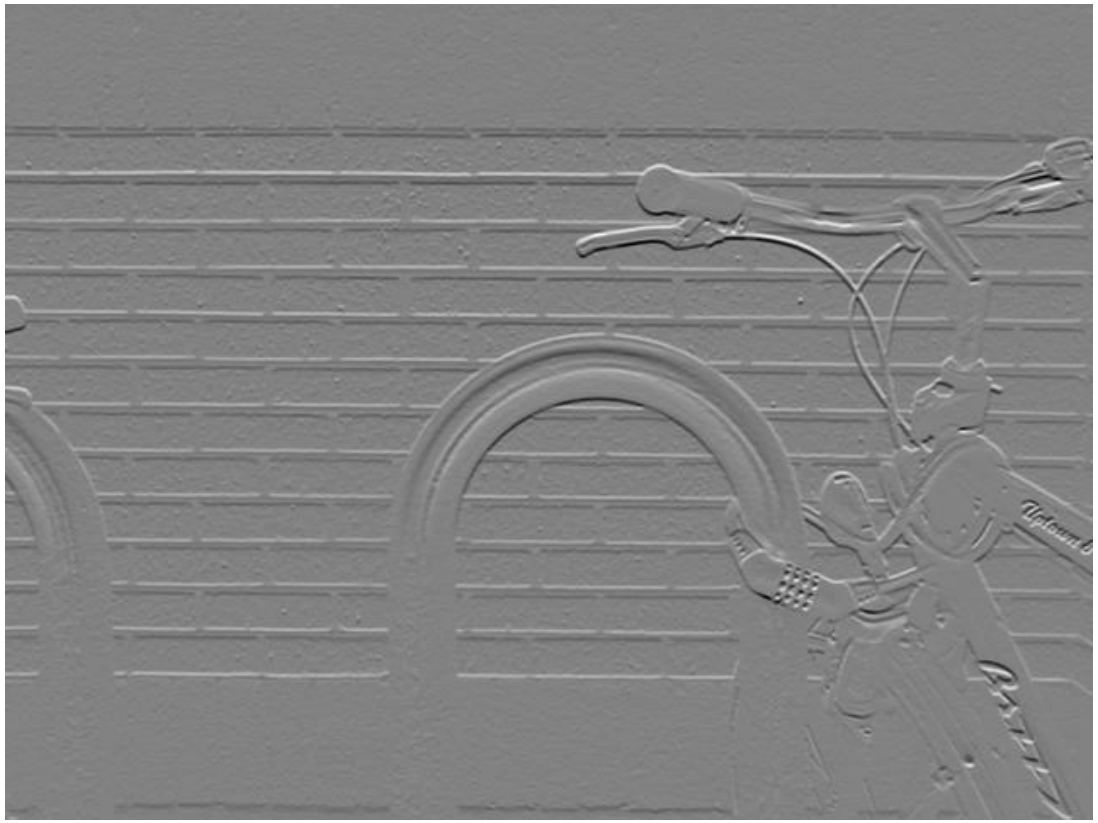


Sobel operator: example

https://en.wikipedia.org/wiki/Sobel_operator

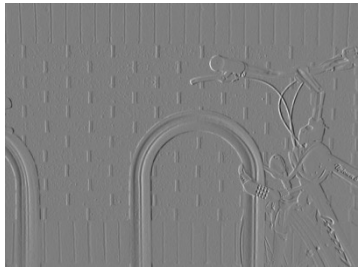
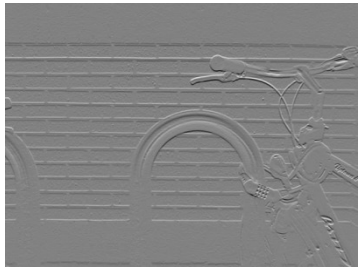






Sobel operator: example

https://en.wikipedia.org/wiki/Sobel_operator

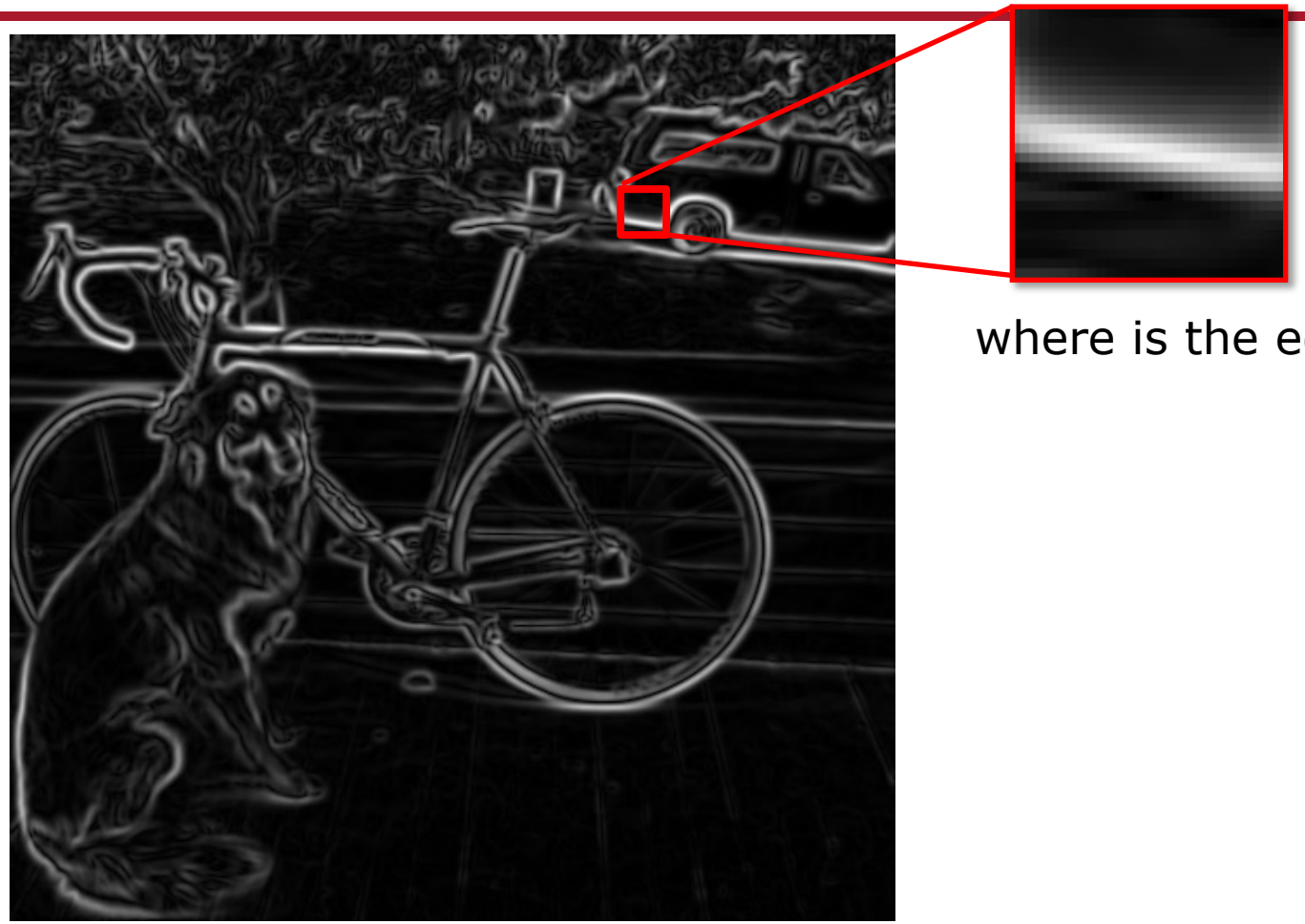

Canny Edge detection

original image

Finding edges

smoothed gradient magnitude

Finding edges



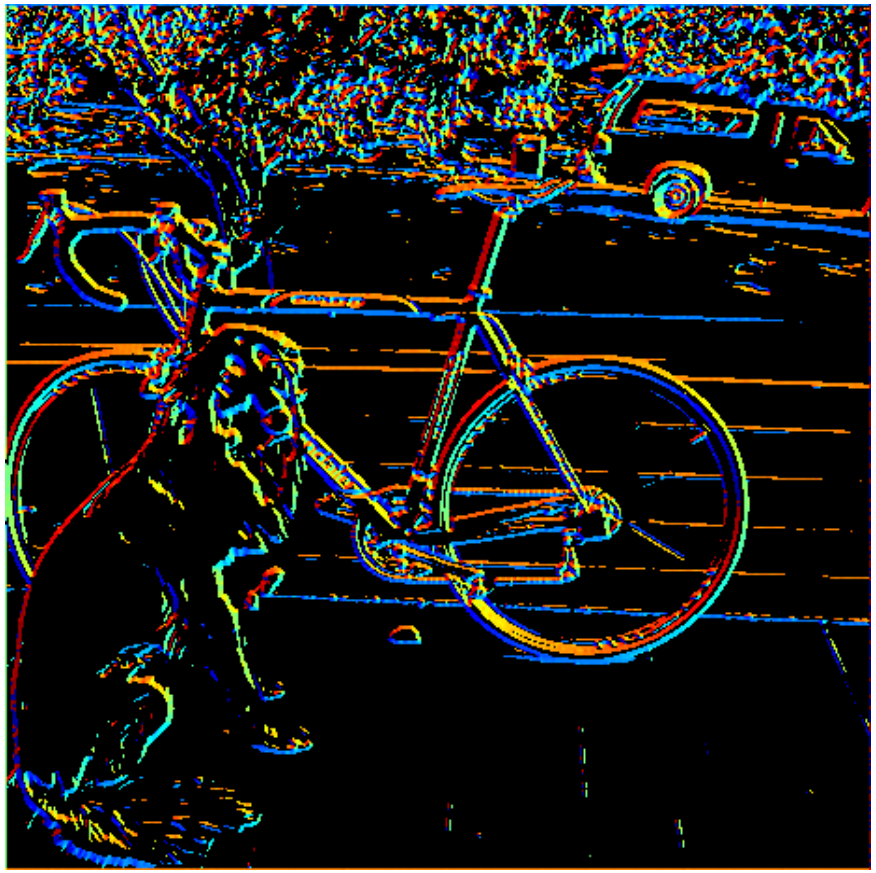
where is the edge?

smoothed gradient magnitude

Get Orientation at Each Pixel

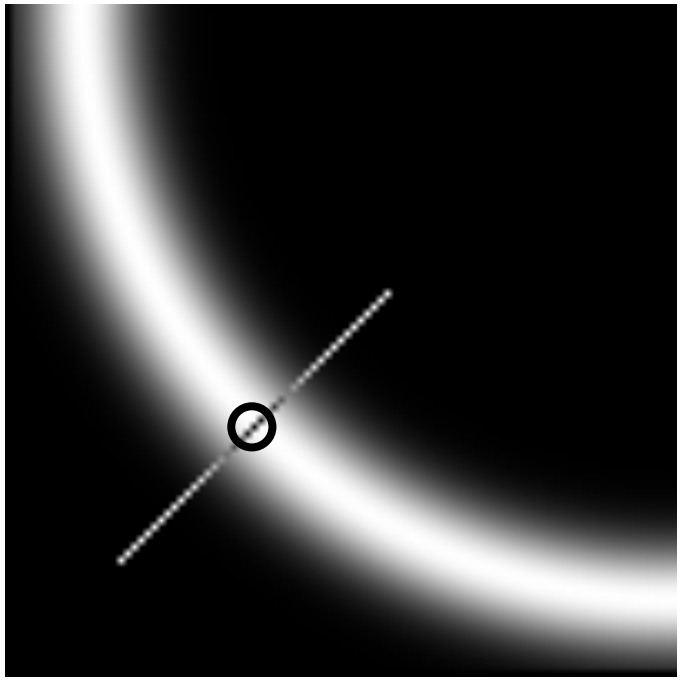
- Get orientation (below, threshold at minimum gradient magnitude)

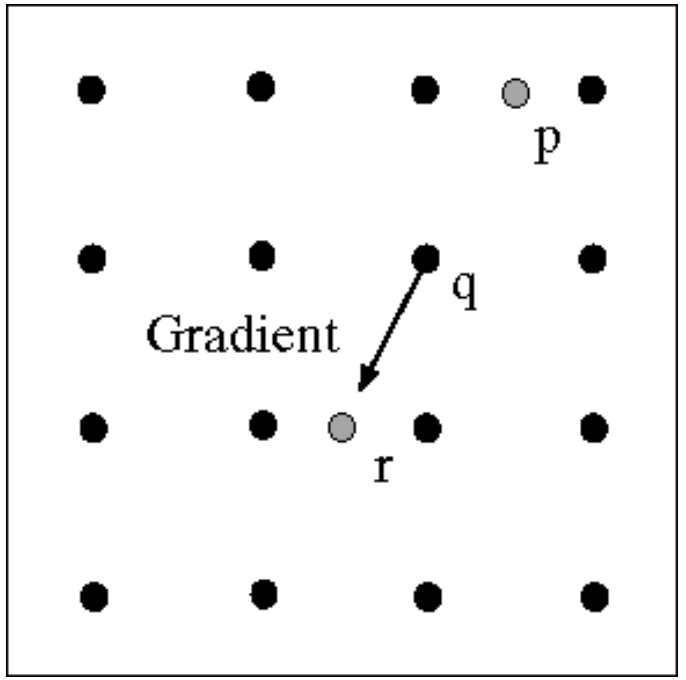
theta = atan2(gy, gx)

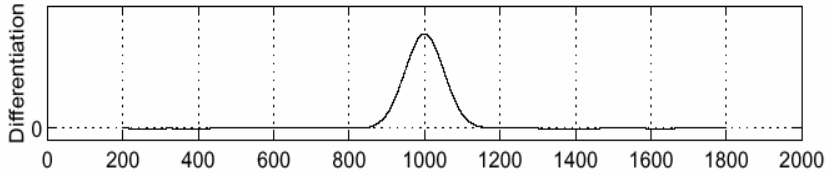


Gradient orientation angle

Non-maximum suppression







- Check if pixel is local maximum along gradient direction
  - requires *interpolating* pixels p and r

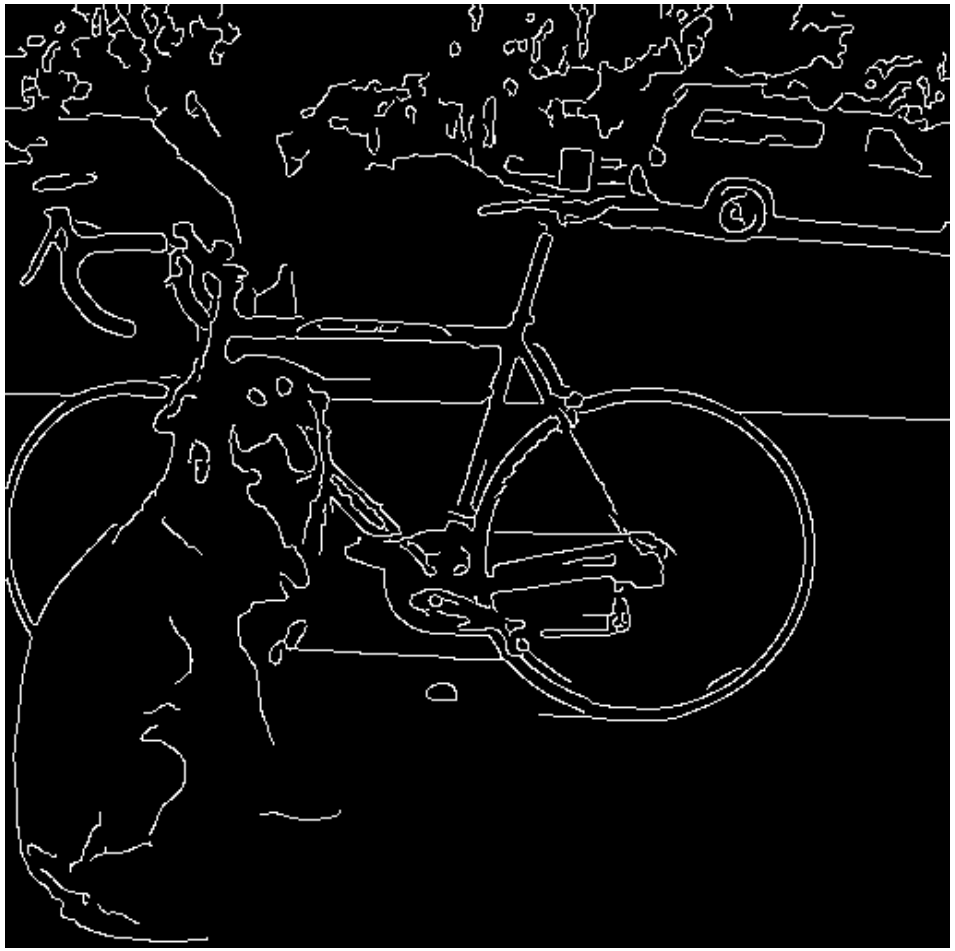
Non-maximum suppression - Before

Non-maximum suppression - After

Thresholding edges

- Still some noise
- Only want strong edges
- 2 thresholds, 3 cases
  - R > T: strong edge
  - R < T but R > t: weak edge
  - R < t: no edge
- Why two thresholds?

Connecting edges



- Strong edges are edges!
- Weak edges are edges if they connect to strong
- Look in some neighborhood (usually 8 closest)

Canny edge detector

original

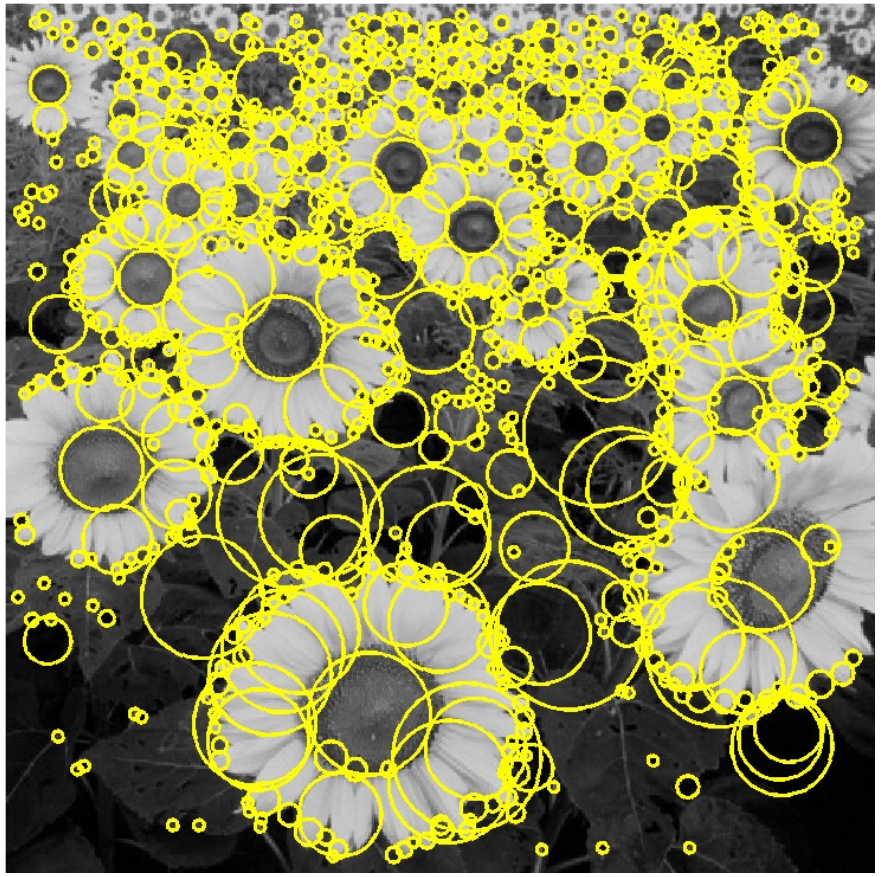
Canny with $\sigma = 1$

Canny with $\sigma = 2$

- The choice of σ depends on desired behavior
  - large σ detects "large-scale" edges
  - small σ detects fine edges

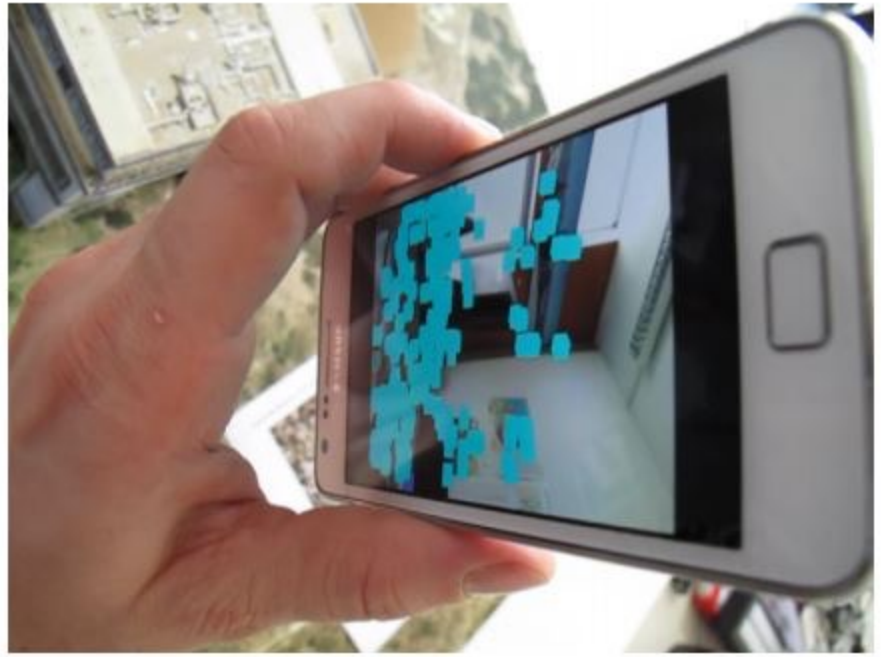
https://docs.opencv.org/4.x/da/d22/tutorial_py_canny.html

Feature extraction - Corners and blobs

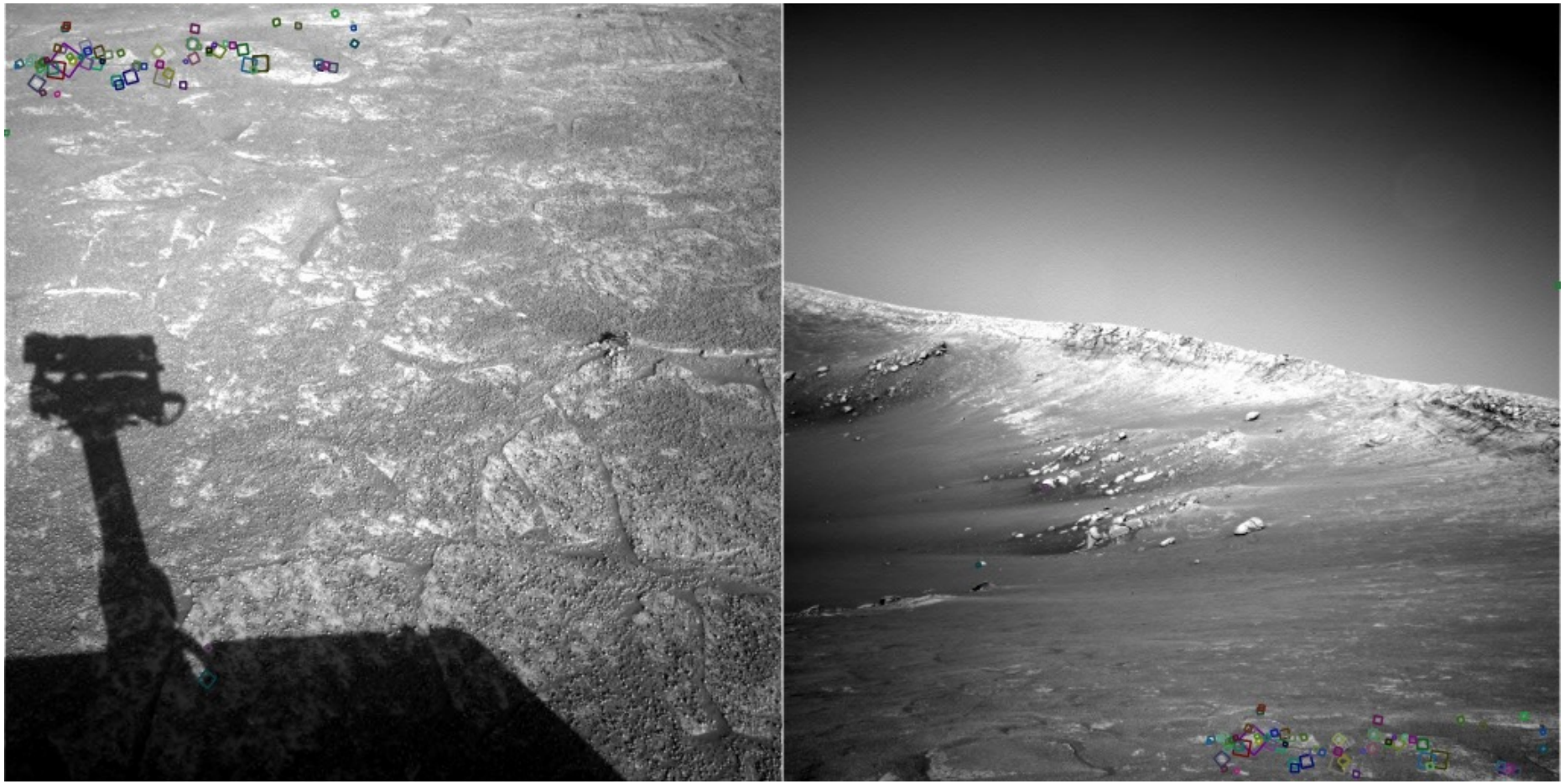


Application - Visual SLAM

- (aka Simultaneous Localization and Mapping)

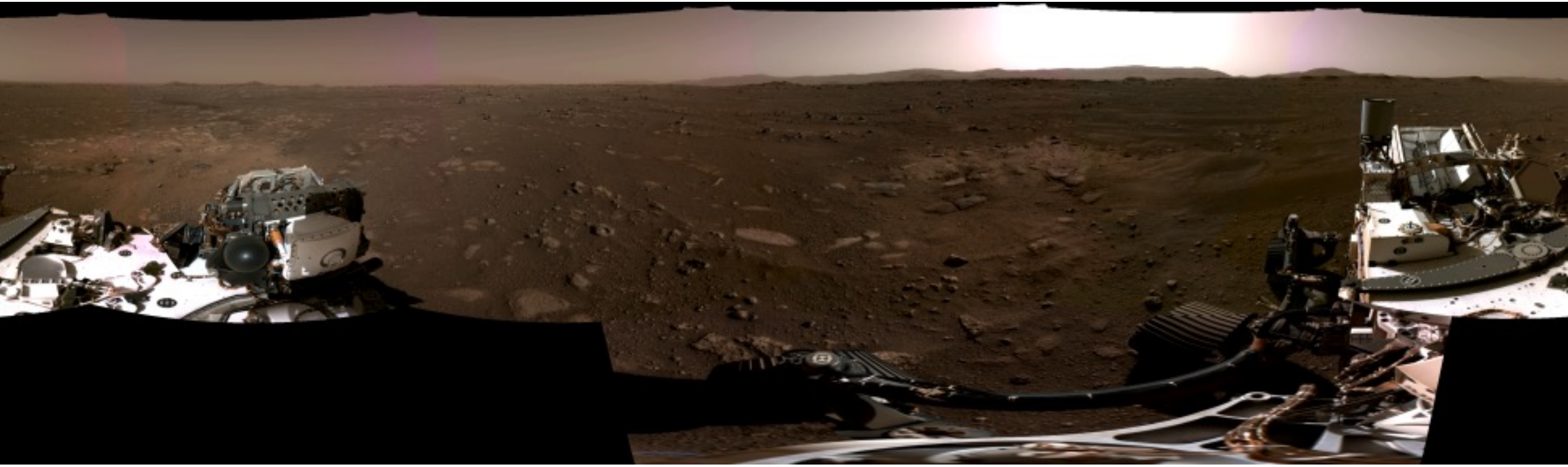


Application - Mars exploration



NASA Mars Rover images
with SIFT feature matches

Application - Panorama stitching



Panorama captured by Perseverance Rover, Feb. 20, 2021

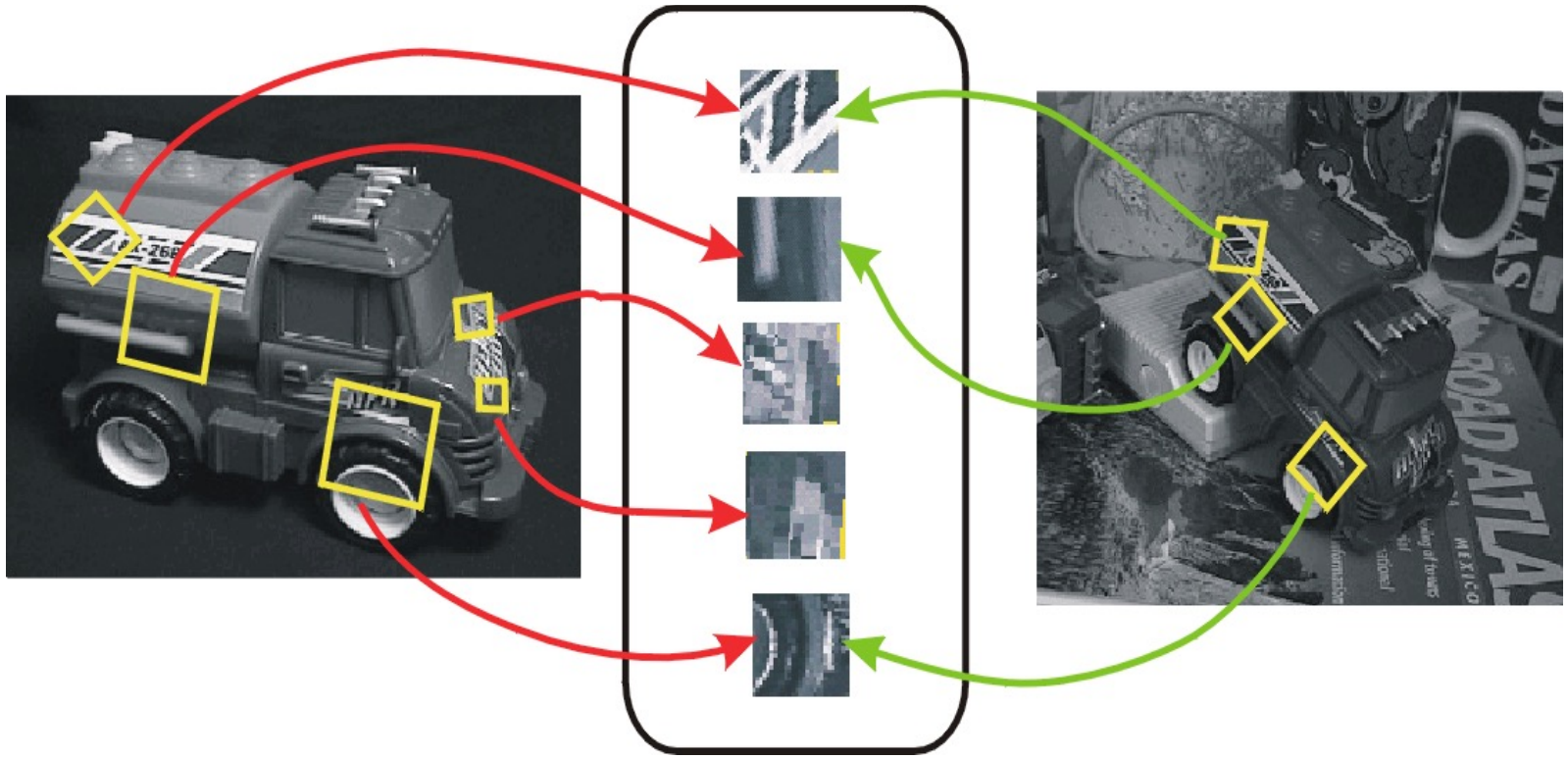
https://www.space.com/nasa-perseverance-rover-first-panorama-mars

Application - Object search

Invariant local features

Find features that are invariant to transformations

- geometric invariance: translation, rotation, scale
- photometric invariance: brightness, exposure, …



Feature Descriptors

More motivation …

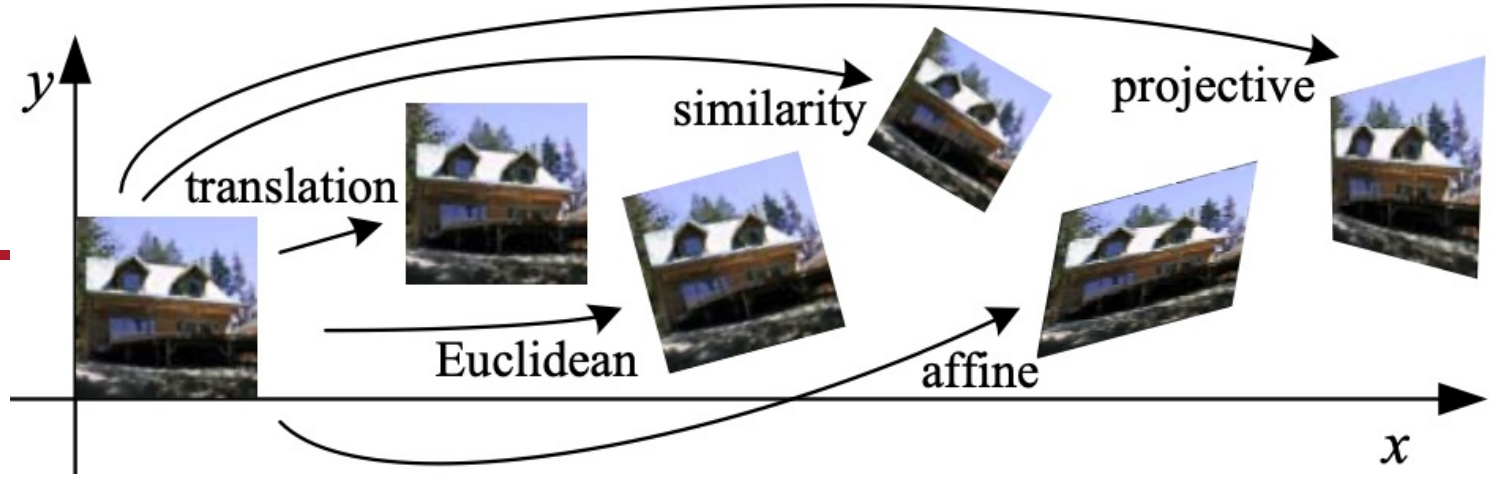
Feature points are used for:

- Image alignment (e.g., mosaics)
- 3D reconstruction
- Motion tracking (e.g., for AR)
- Object recognition
- Image retrieval
- Robot/car navigation

Feature extraction algorithms

- Canny Edge Detector
- Harris Corner Detector
- Shi-Tomasi Corner Detector
- SIFT (Scale-Invariant Feature Transform)
- SURF (Speed-Up Robust Features)
- ORB (Oriented FAST and Rotated BRIEF)
- HOG (Histogram of Oriented Gradients)
- LBP (Local Binary Pattern)
- ORB-SLAM
- …

Image transformations



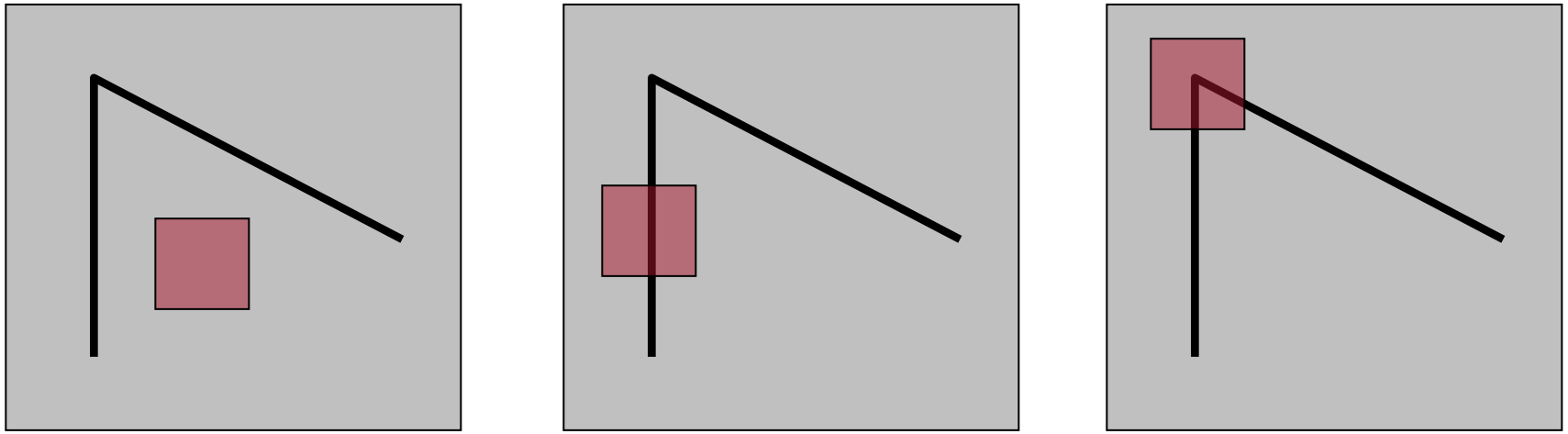
Transformation	Matrix	# DoF	Preserves	Icon
translation	$\begin{bmatrix} \mathbf{I} & \mathbf{t} \end{bmatrix}_{2\times 3}$	2	orientation	
rigid (Euclidean)	$\begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix}_{2\times 3}$	3	lengths	
similarity	$\begin{bmatrix} s\mathbf{R} & \mathbf{t} \end{bmatrix}_{2\times 3}$	4	angles	
affine	$\begin{bmatrix} \mathbf{A} \end{bmatrix}_{2\times 3}$	6	parallelism	
projective	$\begin{bmatrix} \tilde{\mathbf{H}} \end{bmatrix}_{3\times 3}$	8	straight lines	

Feature extraction - Corners

Local measures of uniqueness

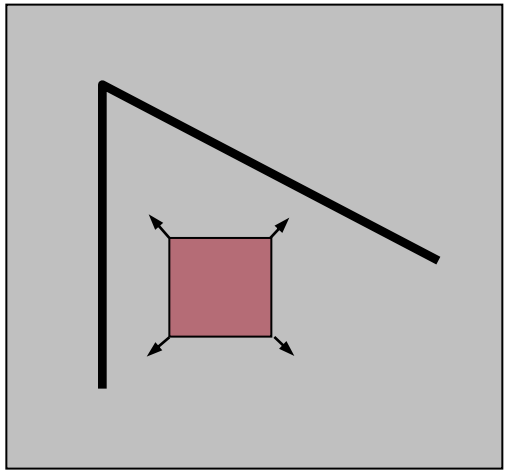
Suppose we only consider a small window of pixels

- What defines whether a feature is a good or bad candidate?

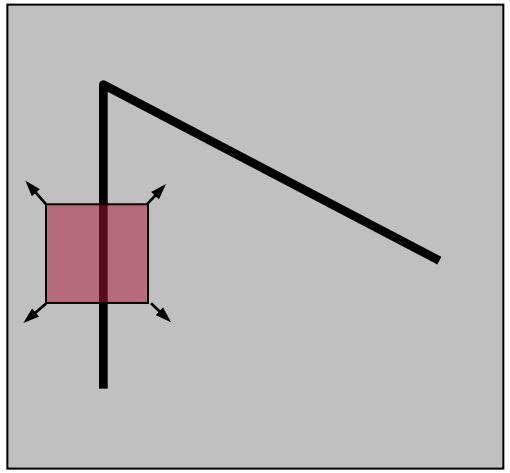


Corner

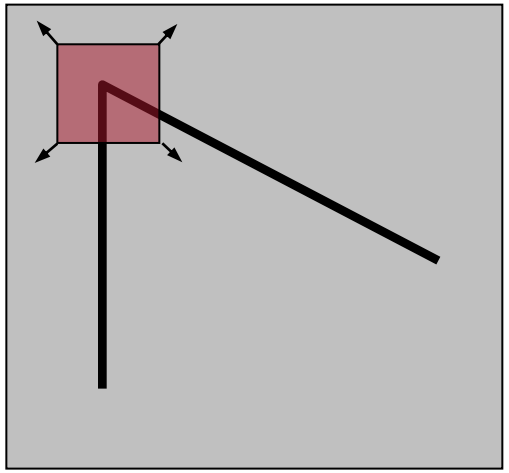
- How does the window change when you shift it?
- Shifting the window in any direction causes a big change



"flat" region: no change in all directions



"edge": no change along the edge direction

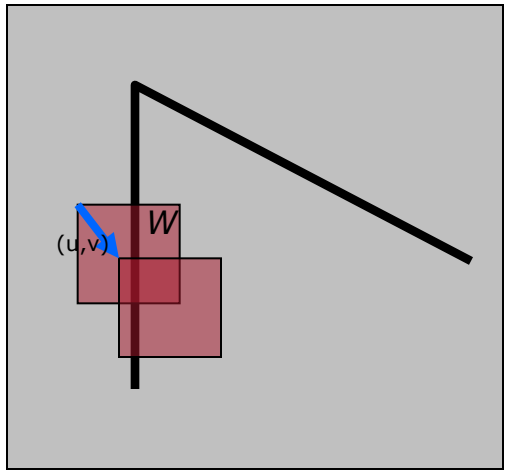


"corner": significant change in all directions

Corner Detection

- Consider shifting the window *W* by (*u,v*)
- Compare each pixel before and after by summing up the squared differences (SSD)
- This defines an SSD "error" *E*(*u,v*):

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



Taylor Series / Expansion - Reminder

Taylor Series expansion of I:

$$I(x+u, y+v) = I(x,y) + \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v + \text{higher order terms}$$

If the motion (u, v) is small, then first order approximation is good

$$I(x+u, y+v) \approx I(x,y) + \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v$$

$$\approx I(x,y) + \begin{bmatrix} I_x & I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}$$

shorthand: $I_x = \frac{\partial I}{\partial x}$

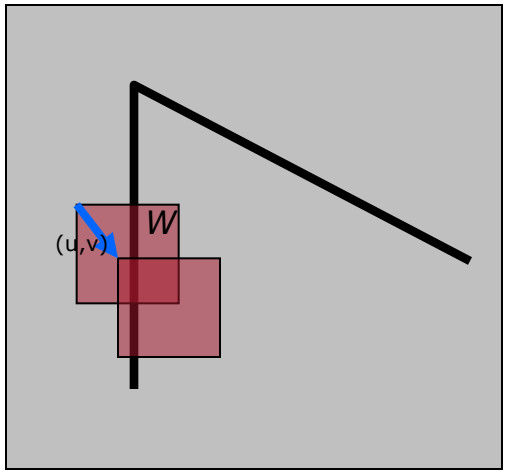
Plugging this into the formula on the previous slide…

https://en.wikipedia.org/wiki/Taylor_series

Corner Detection

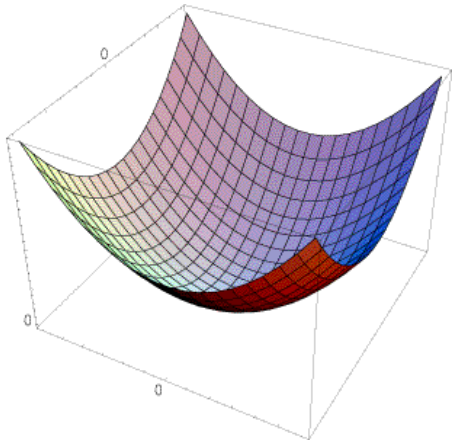
- Consider shifting the window *W* by (*u*,*v*)
- Compare each pixel before and after by summing up the squared differences (SSD)
- This defines an SSD "error" *E*(*u*,*v*):

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



- **Apply the Taylor Expansion:**

$$E(u,v) \approx [u \quad v]\left(\sum \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}\right)\begin{bmatrix} u \\ v \end{bmatrix}$$



Corner Detection

$$M = \sum w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

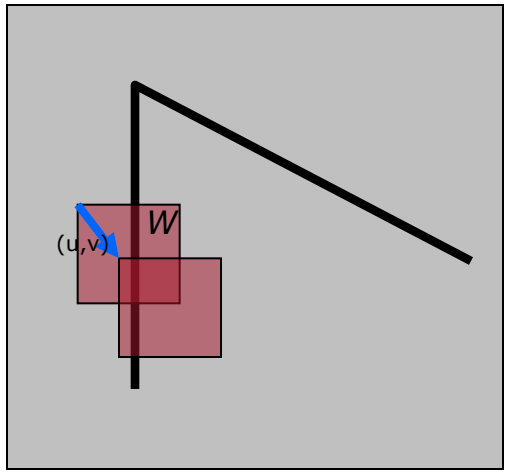
$$R = Det(M) - kTrace^2(M)$$

k is an empirically determined constant; k∈[0.04,0.06]

Where

$$Det(M) = \lambda_1 \lambda_2$$

$$Trace(M) = \lambda_1 + \lambda_2$$

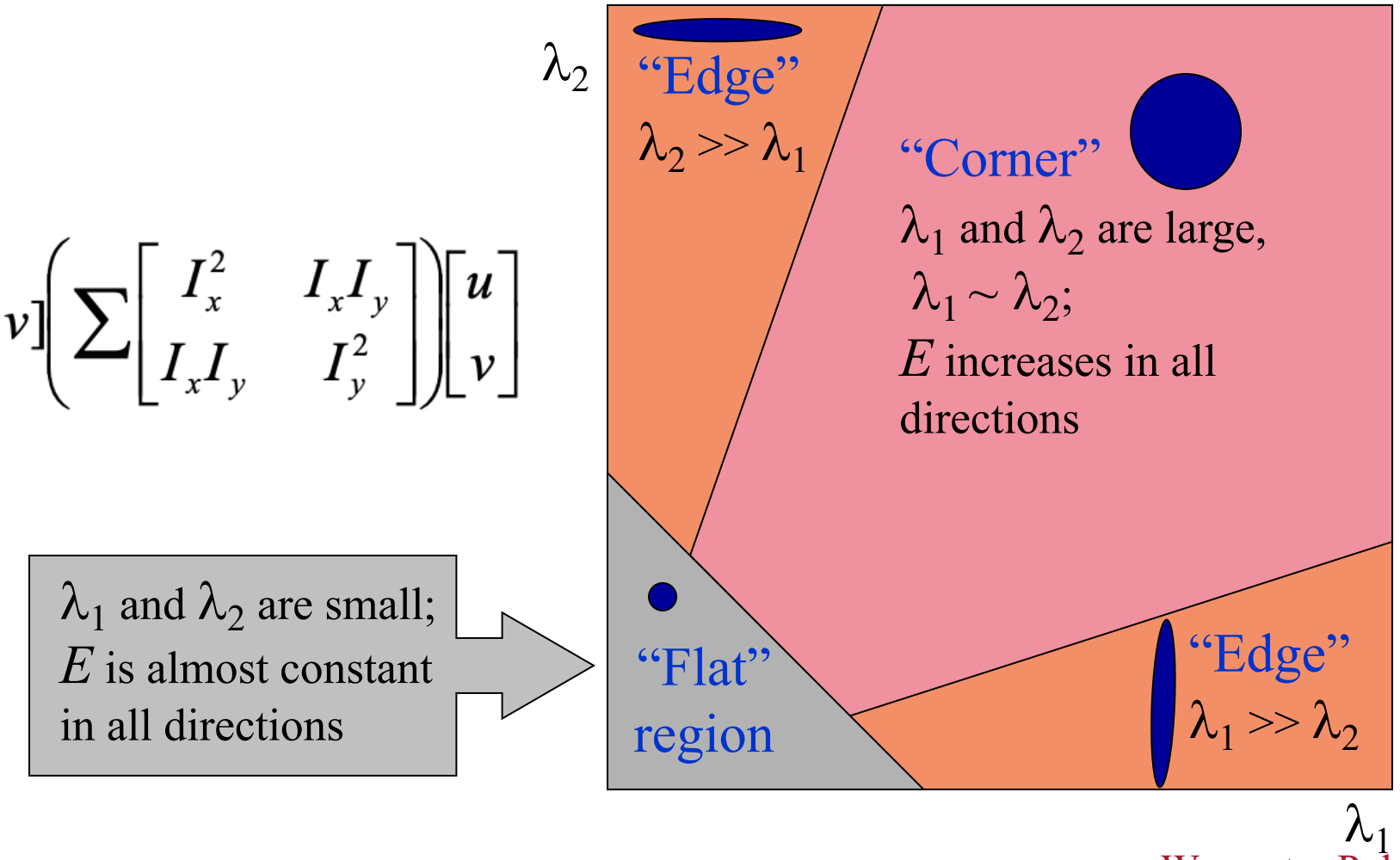


https://en.wikipedia.org/wiki/Structure_tensor

Interpreting the eigenvalues

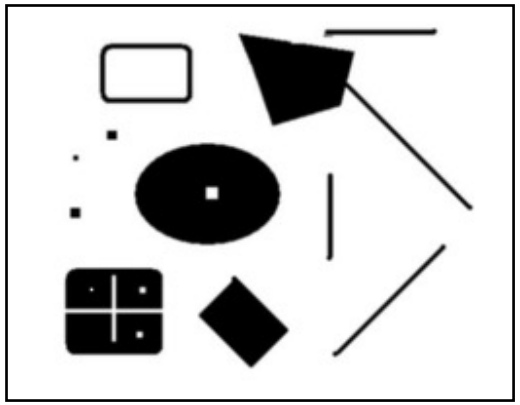
Classification of image points using eigenvalues of M:

$$E(u,v) \approx [u \quad v]\left(\sum \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}\right)\begin{bmatrix} u \\ v \end{bmatrix}$$

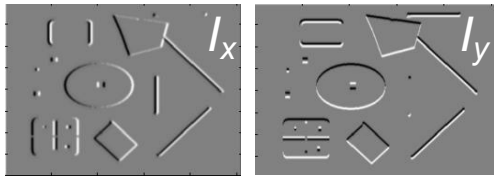


Harris Detector [Harris88] - Steps

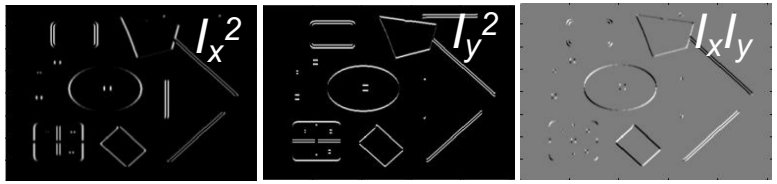
0. Input image



1. Image derivatives

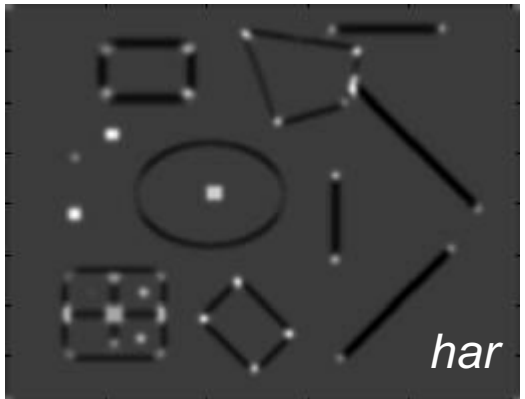


2. Square of derivatives



3. Gaussian filter $G(s_I)$

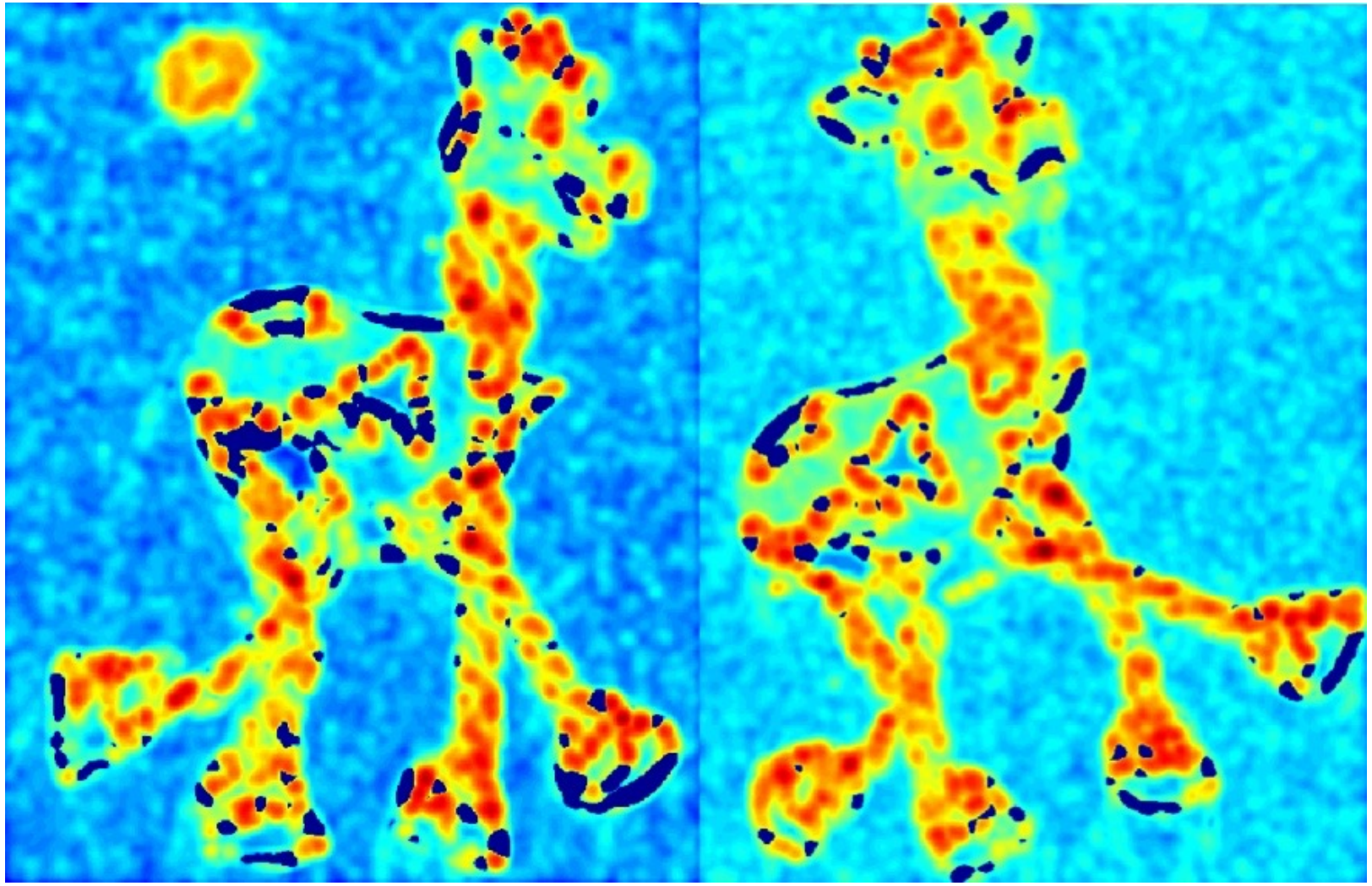
4. Cornerness function - both eigenvalues are strong



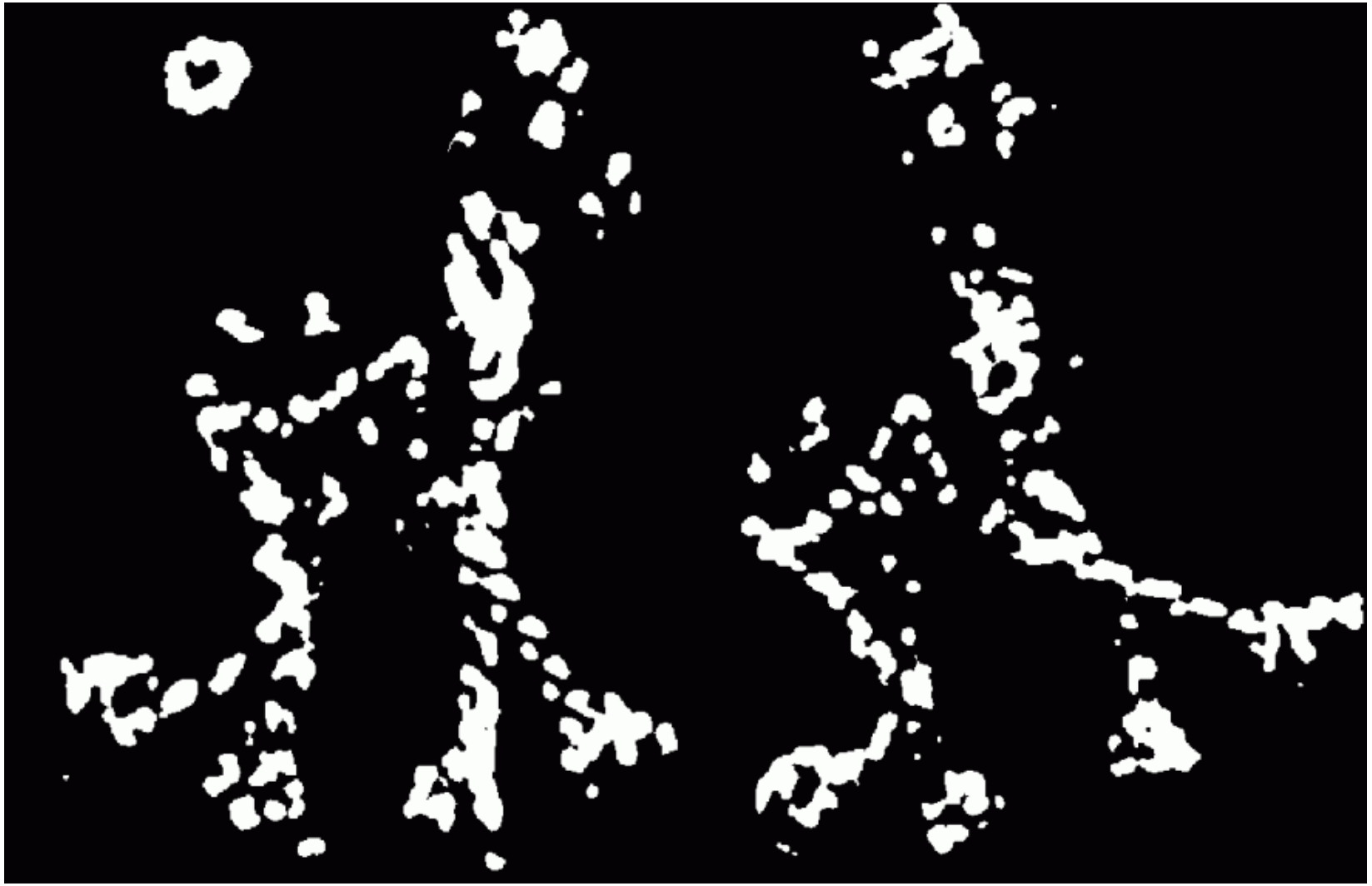
5. Non-maxima suppression

Harris detector example

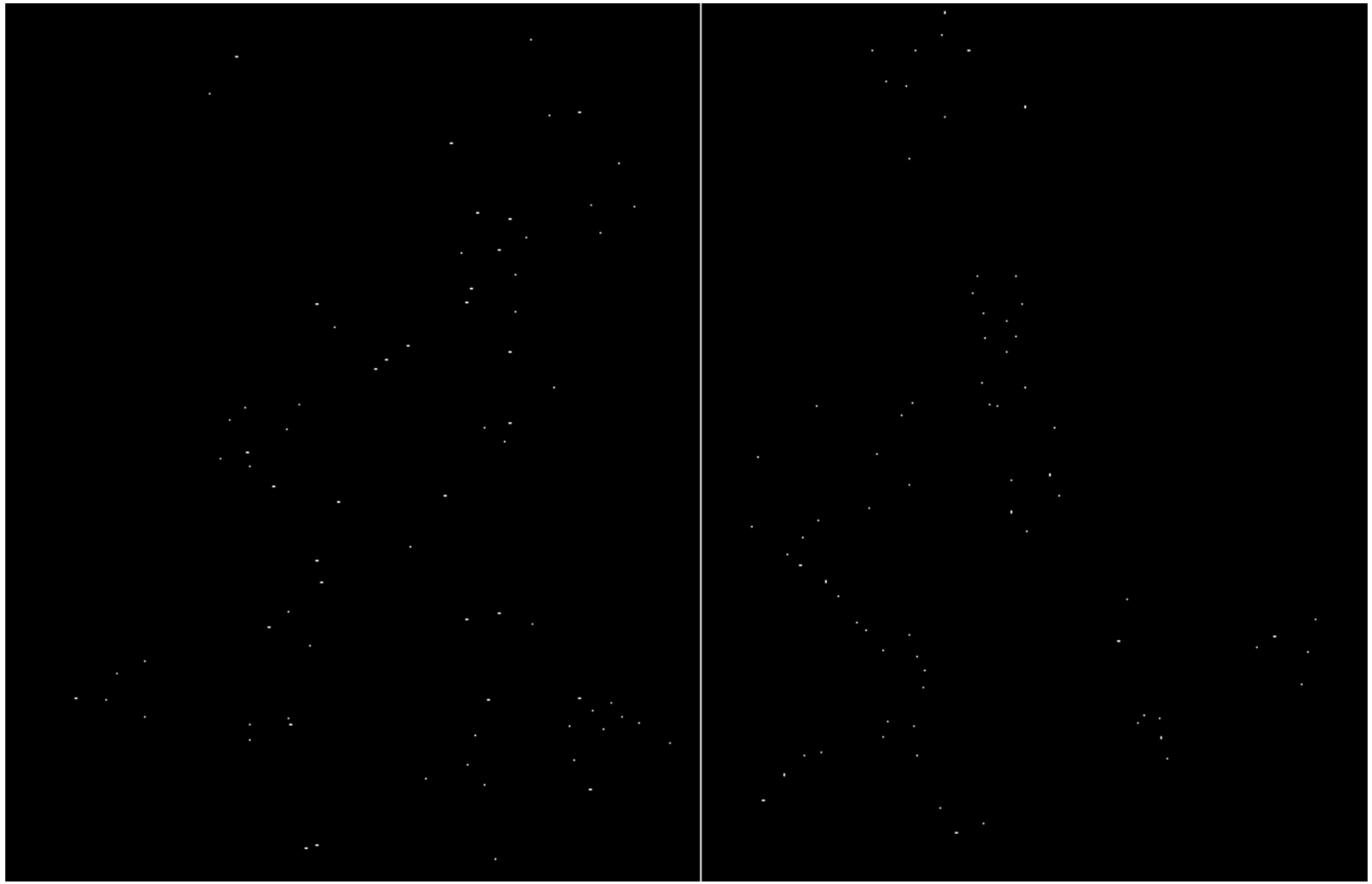
R value (red high, blue low)



Threshold (R > value)

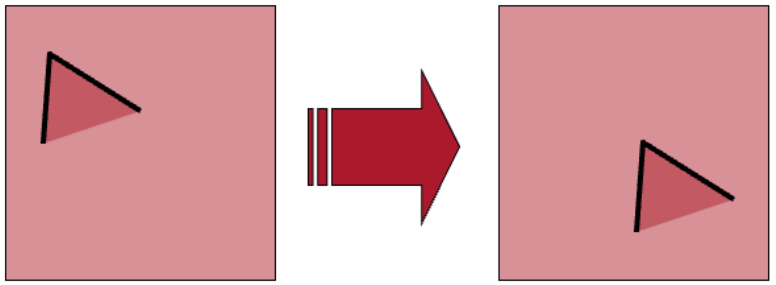


Find local maxima of R (non-max suppression)

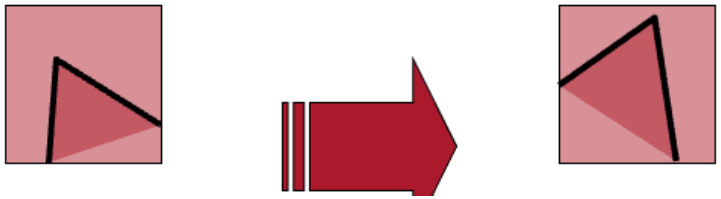


Harris features

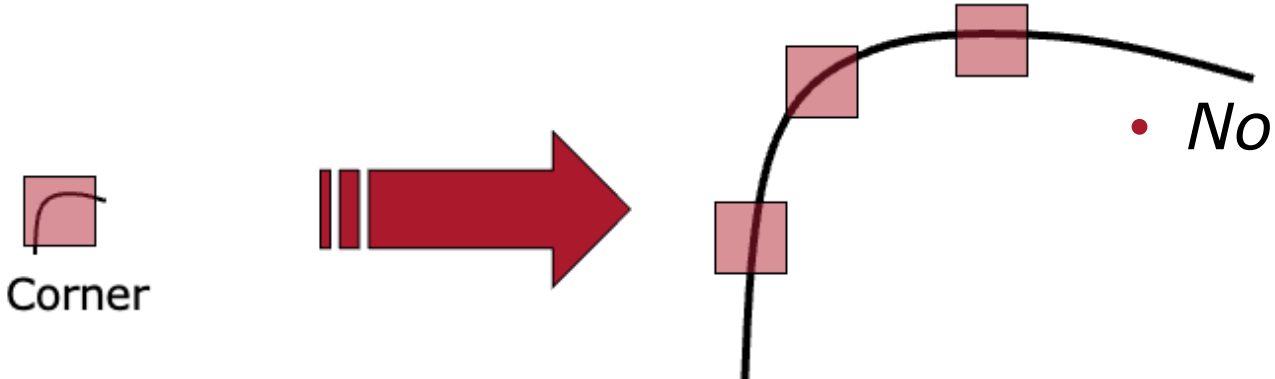
Harris corners features - Properties



- Corner location is equivariant w.r.t. translation



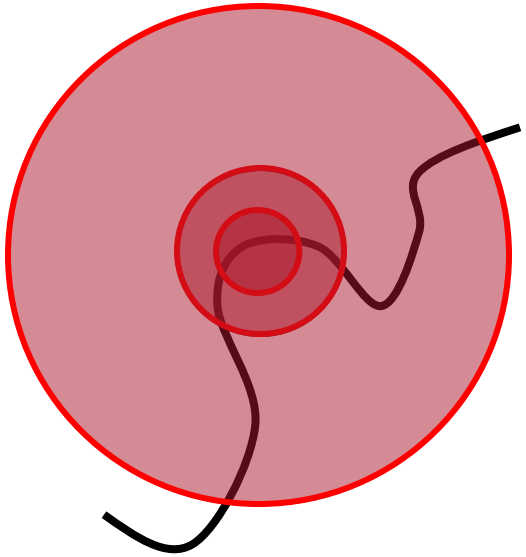
- Corner location is equivariant w.r.t. image rotation



- *No invariant* to scaling

Scale invariant detection

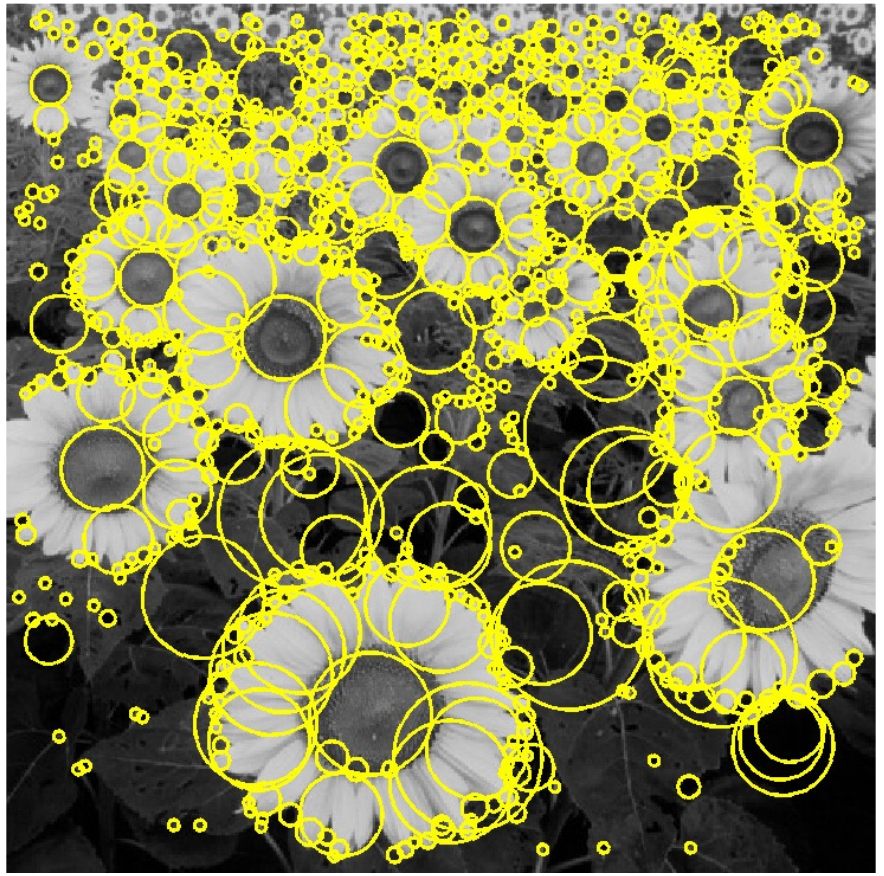
Suppose you're looking for corners



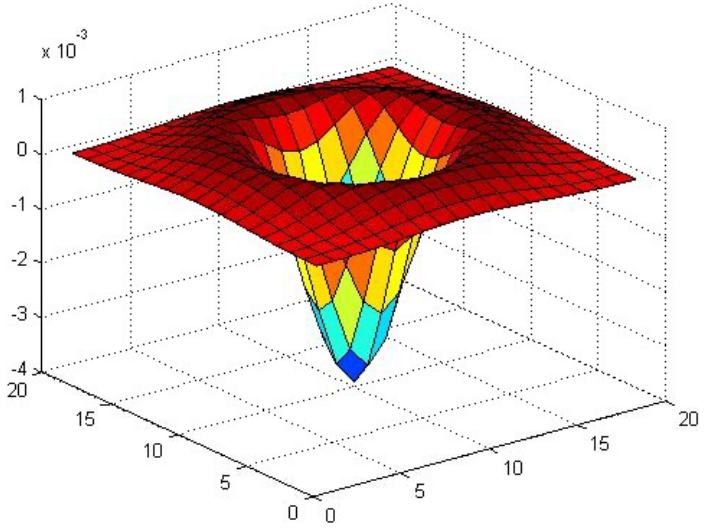
Key idea: find scale that gives local maximum of *f*

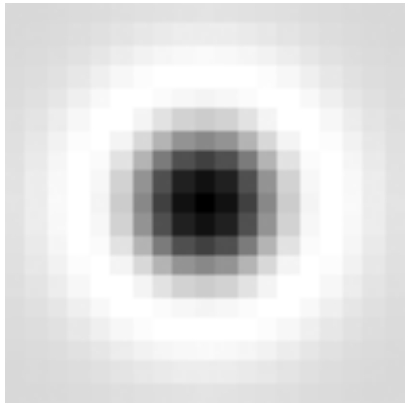
- in both position and scale
- One definition of *f*: the Harris operator

Feature extraction - Blobs



Laplacian of Gaussian (LoG)



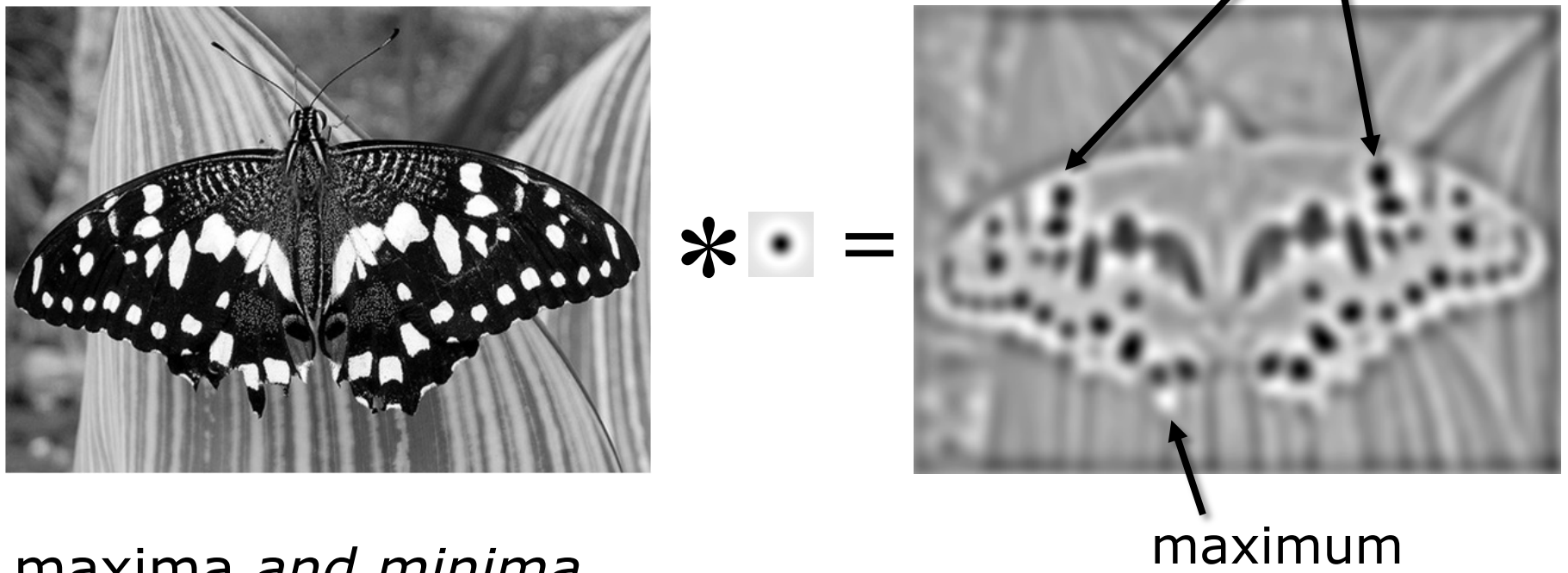


$$\nabla^2 g = \frac{\partial^2 g}{\partial x^2} + \frac{\partial^2 g}{\partial y^2}$$

(very similar to a Difference of Gaussians (DoG) – i.e., a Gaussian minus a slightly smaller Gaussian)

Laplacian of Gaussian (LoG)

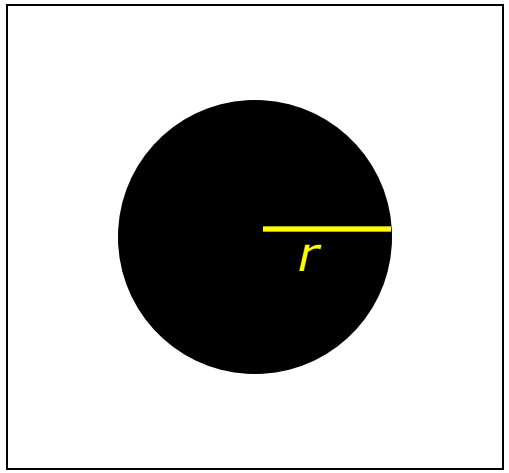
- "Blob" detector



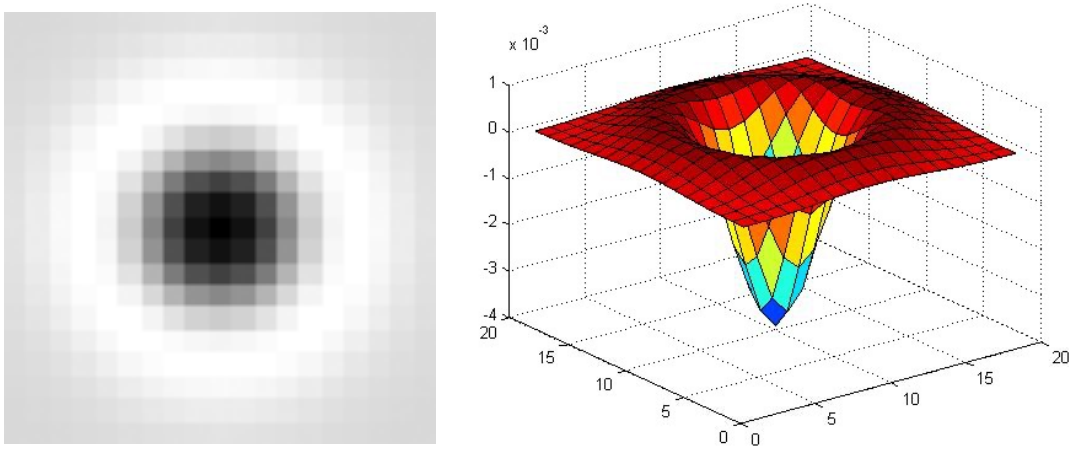
- Find maxima *and minima* of LoG operator in space and scale

Scale selection

- At what scale does the Laplacian achieve a maximum response for a binary circle of radius r?



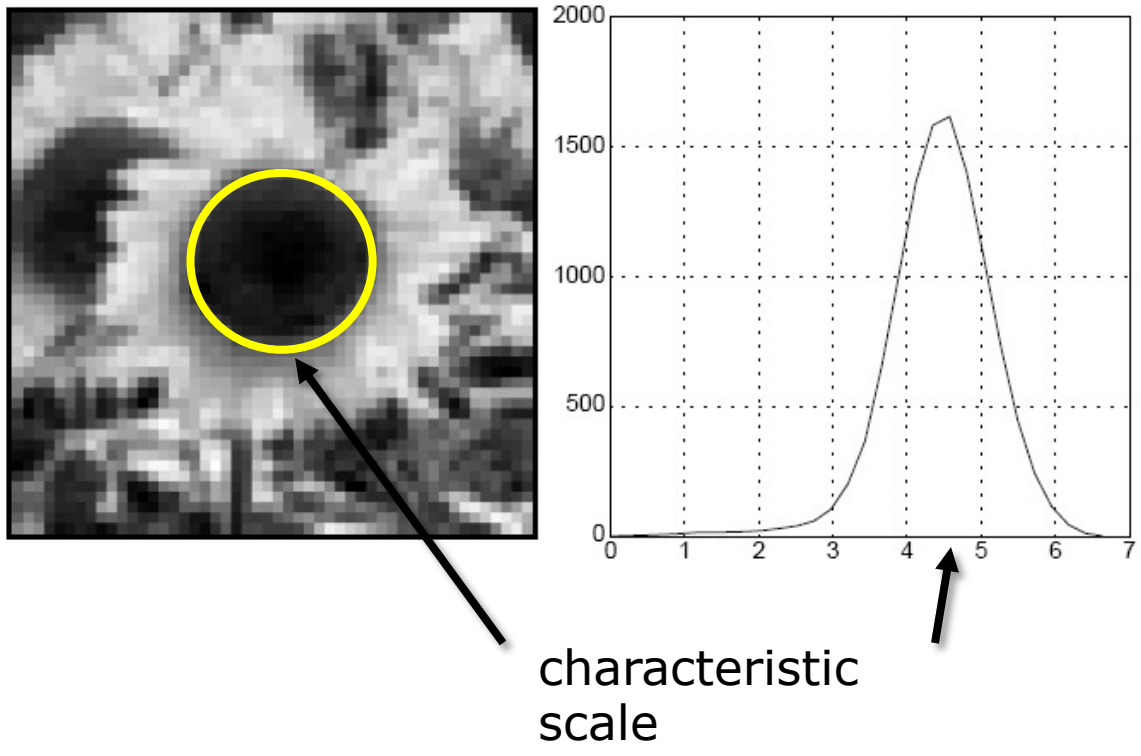
image



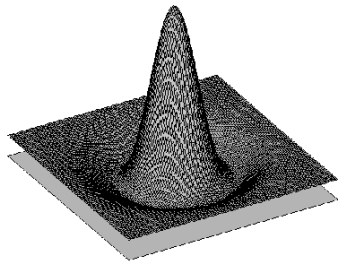
Laplacian

Characteristic scale

- We define the characteristic scale as the scale that produces peak of Laplacian response



Laplacian of Gaussian (LoG) Detector



$$L_{xx}(\sigma) + L_{yy}(\sigma)$$

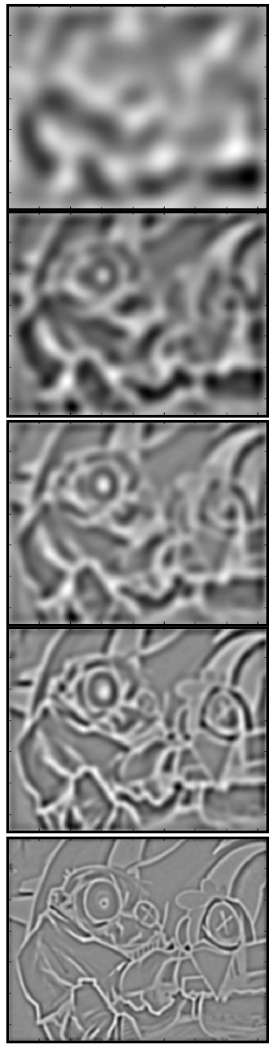
σ^5

σ^4

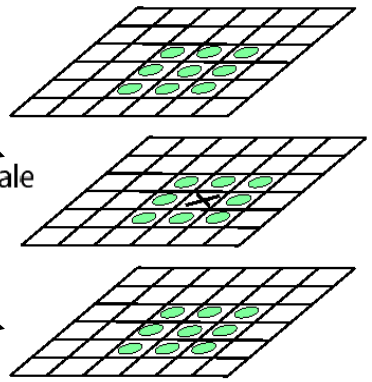
σ^3

σ^2

σ



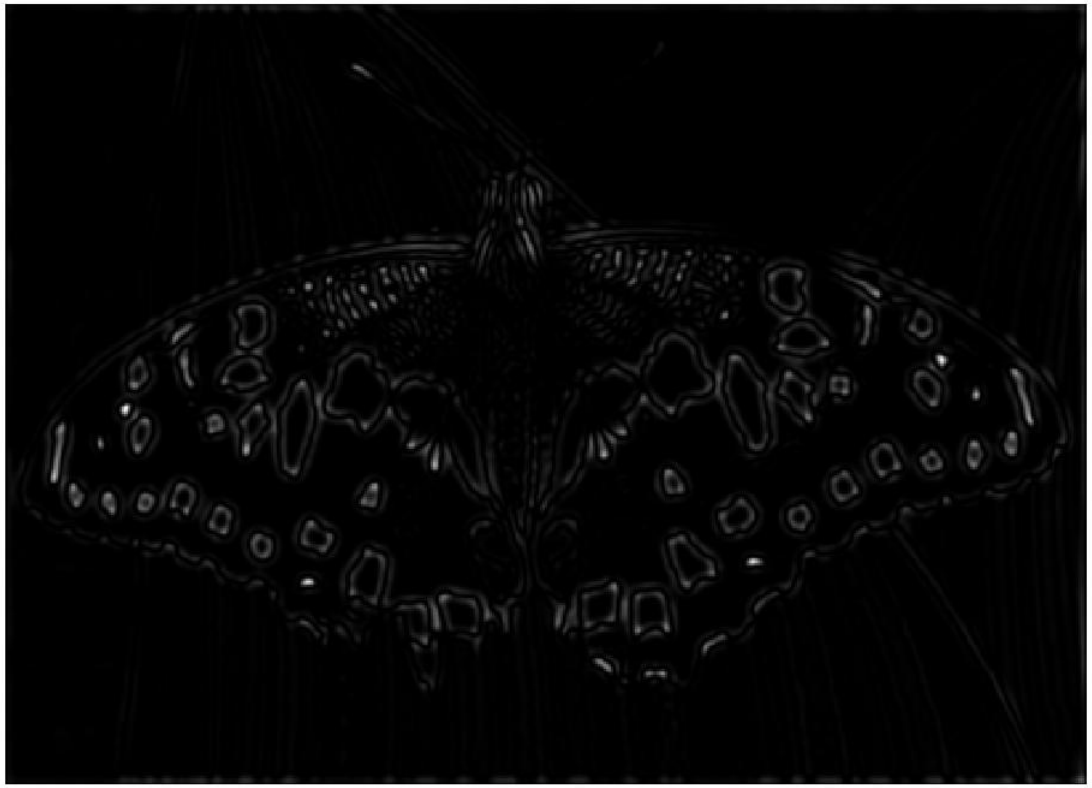
Scale



List of (x, y, σ)

Scale-space blob detector: Example

Scale-space blob detector: Example

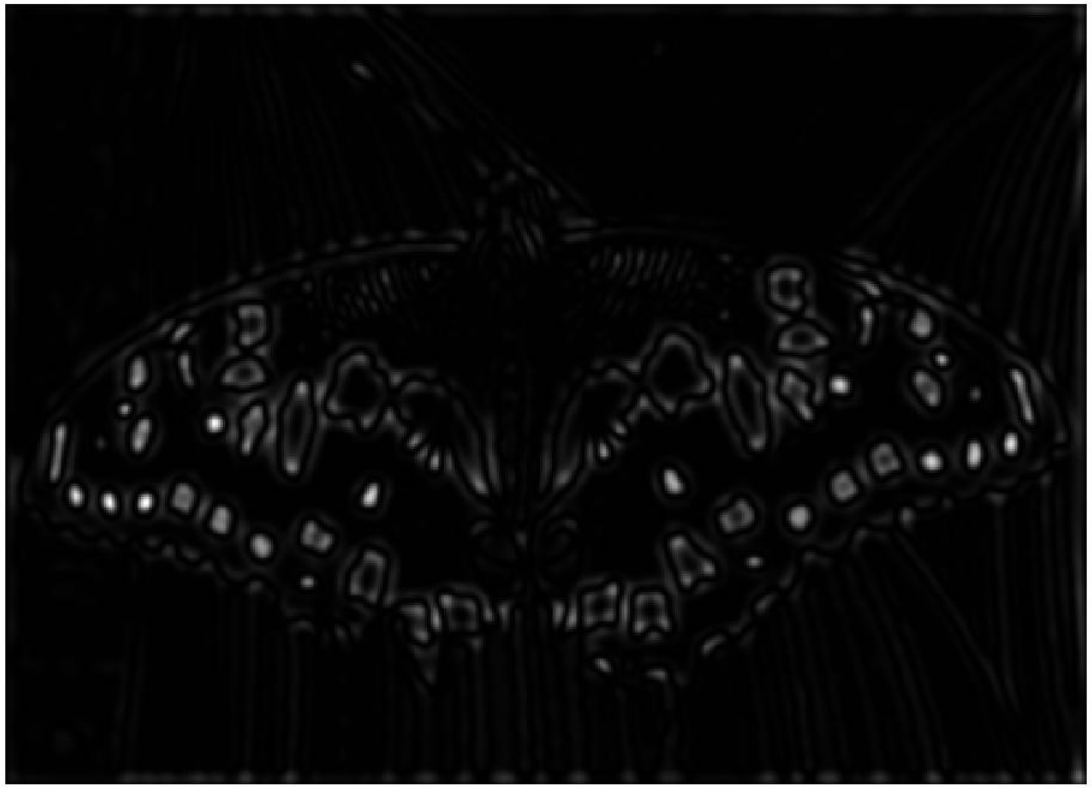


sigma = 2

Scale-space blob detector: Example

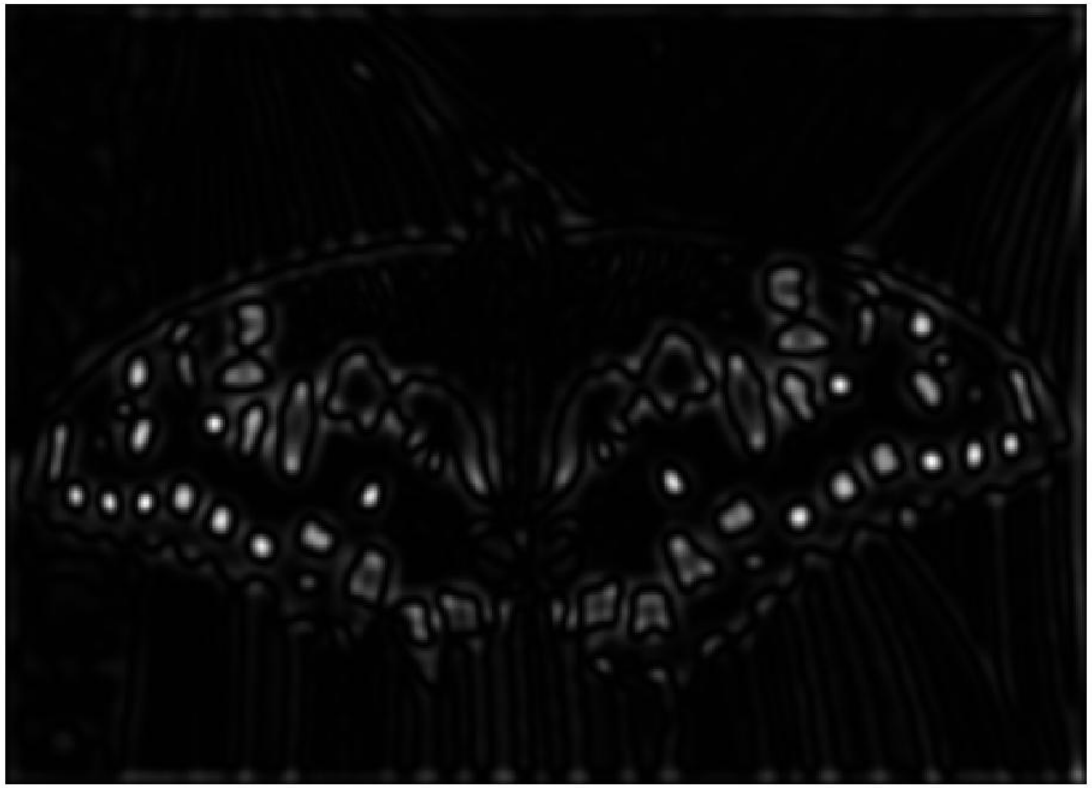
sigma = 2.5018

Scale-space blob detector: Example



sigma = 3.1296

Scale-space blob detector: Example

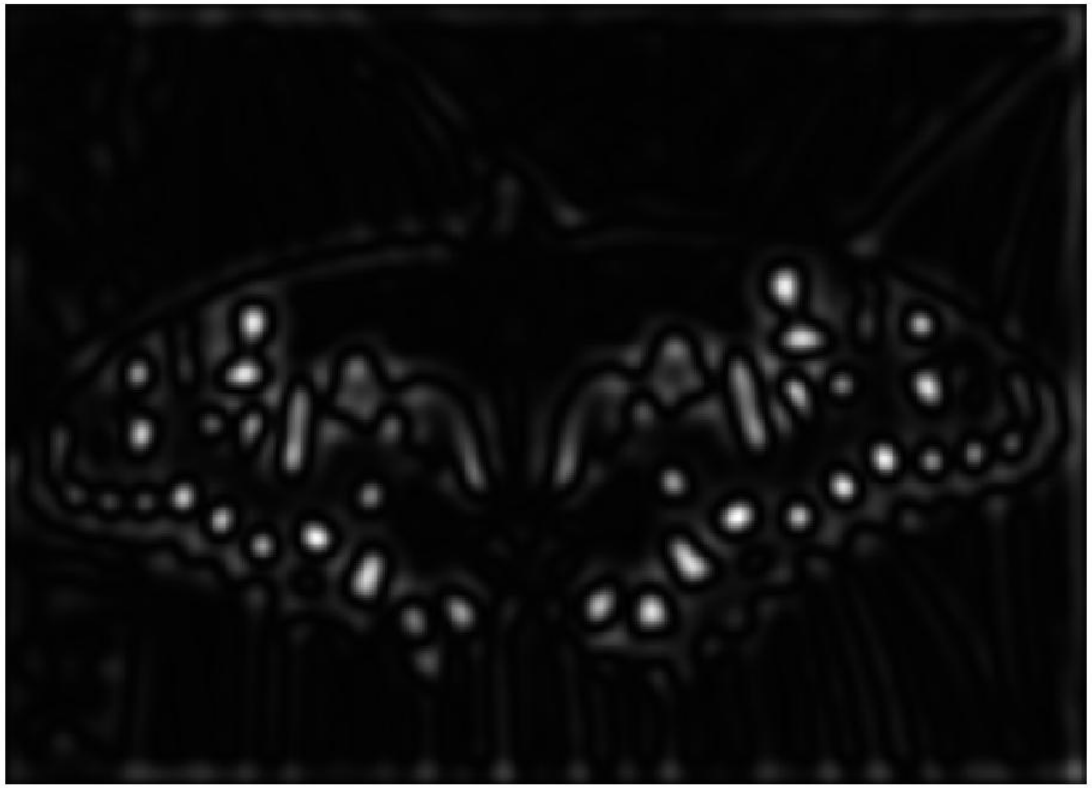


sigma = 3.9149

Scale-space blob detector: Example

sigma = 4.8972

Scale-space blob detector: Example



sigma = 6.126

Scale-space blob detector: Example

sigma = 7.6631

Scale-space blob detector: Example

sigma = 9.5859

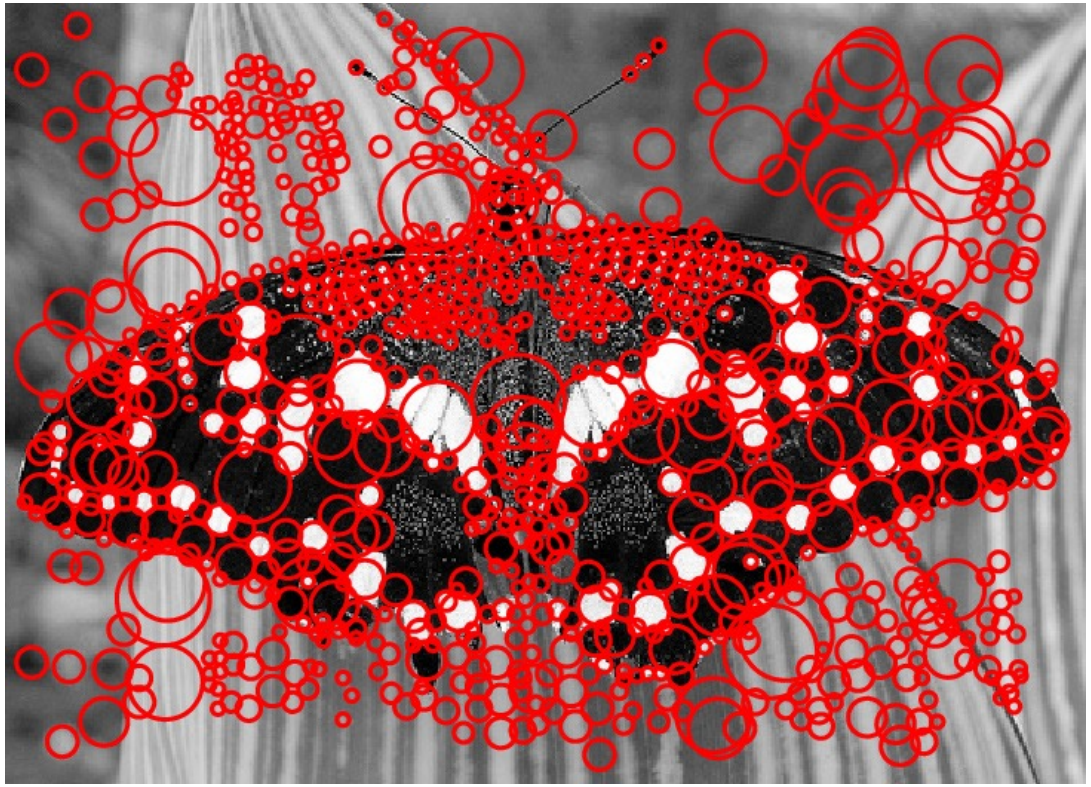
Scale-space blob detector: Example

sigma = 11.9912

Scale-space blob detector: Example

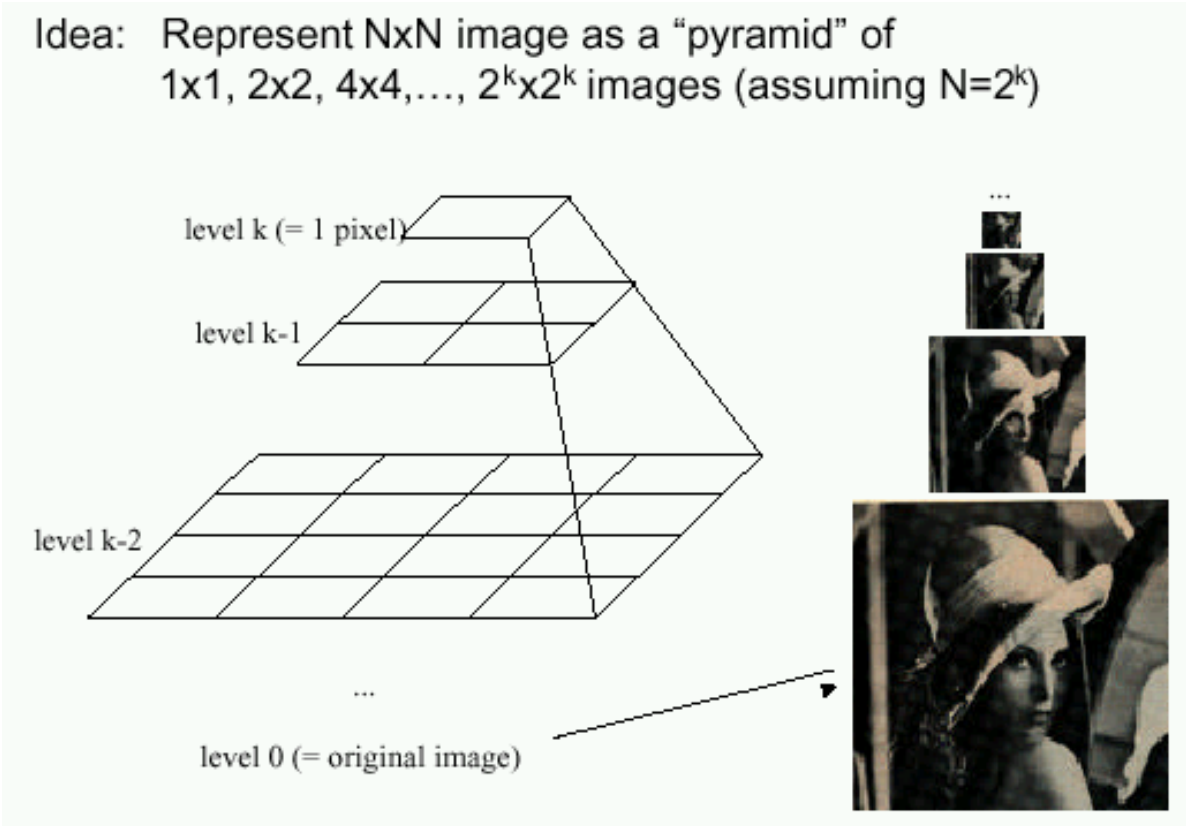
sigma = 15

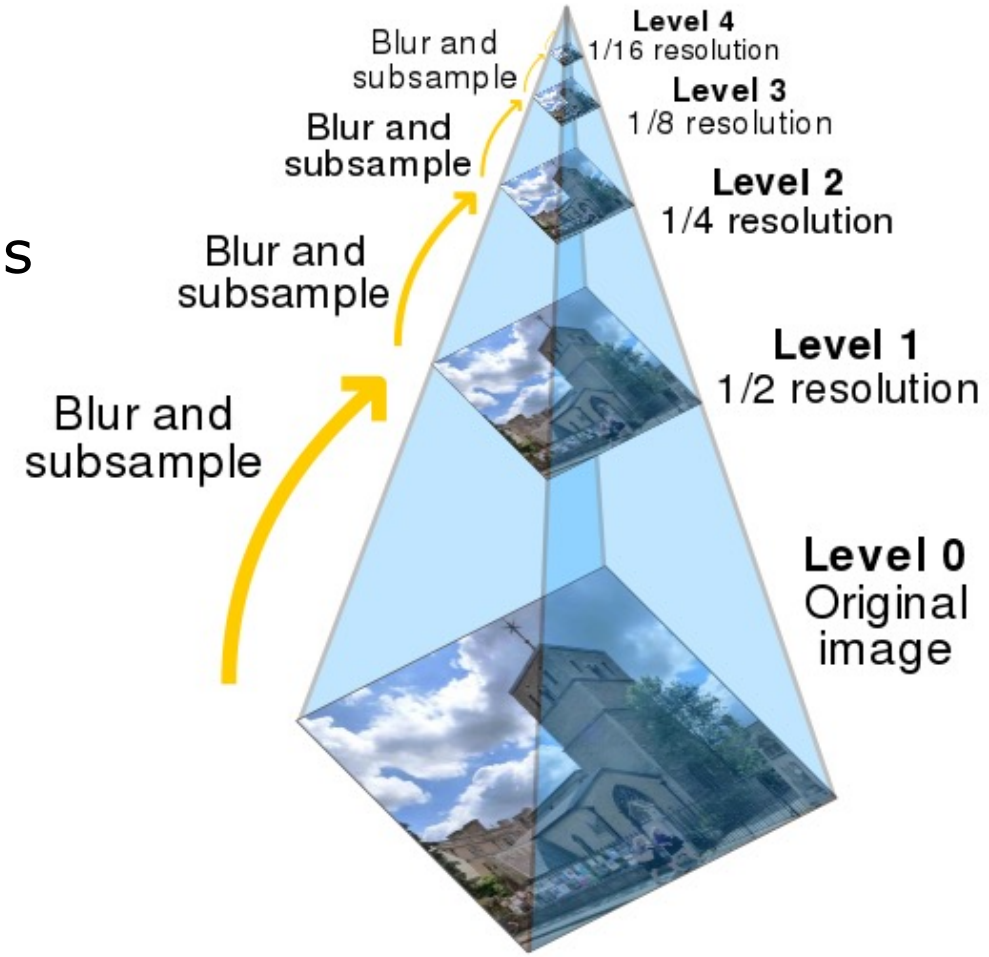
Scale-space blob detector: Example



Gaussian pyramids [Burt and Adelson, 1983]

- Multi-scale representation of an image
- Gaussian smoothing and down-sampling
- Series of progressively lower-resolution images

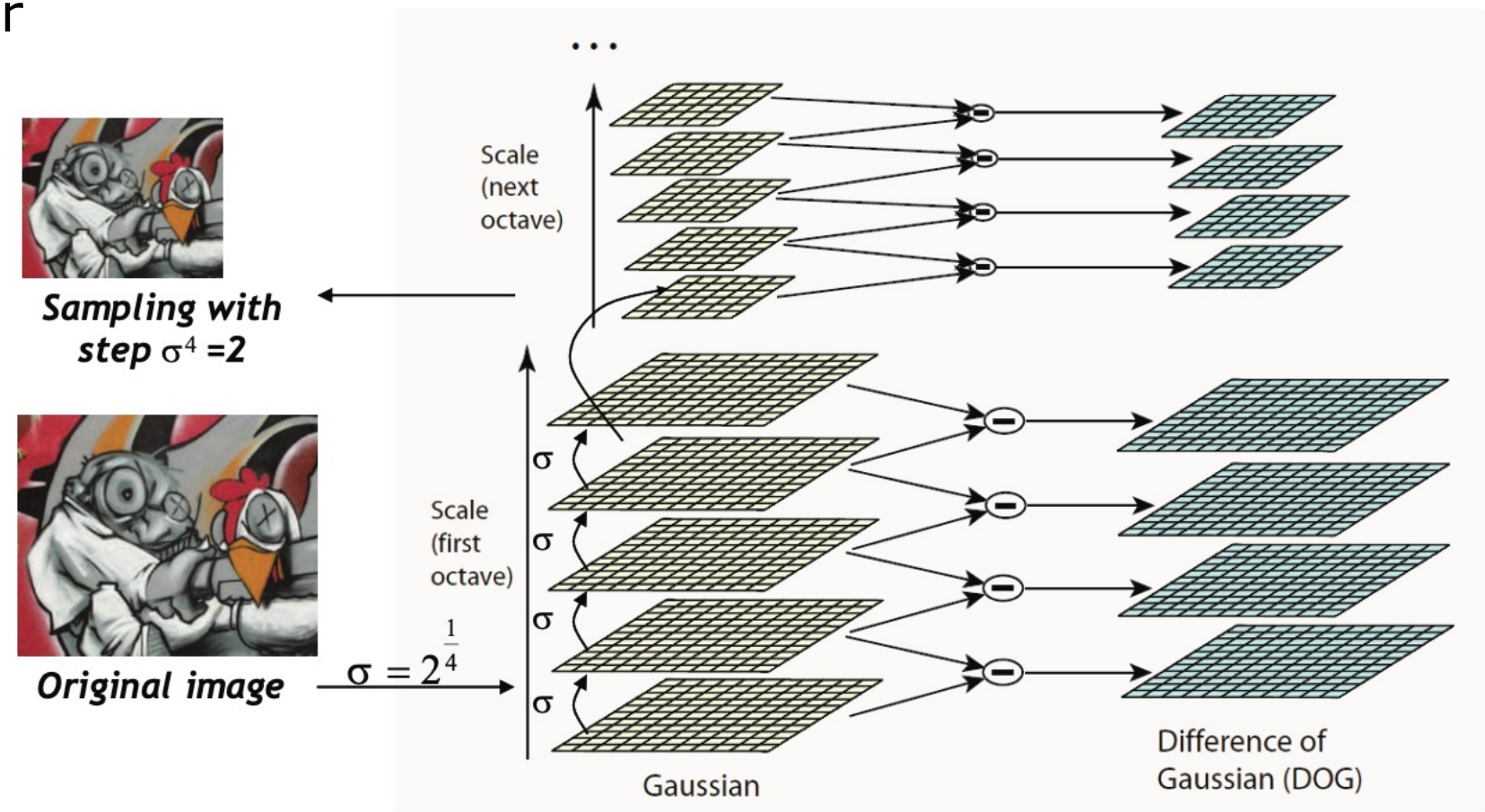




https://en.wikipedia.org/wiki/Pyramid_(image_processing)

Difference of Gaussian (DoG) Detector

- DoG good approximator for LoG
- Subtracting Adjacent scale of a Gaussian pyramid
- Search for 3D scale space extrema



Scale Invariant Feature Transform (SIFT)

- Extract keypoints and compute its descriptors
- LoG for the detection of blobs
- Find local maxima across the scale and space → List of (x, y, σ)
- **LoG** costly → SIFT ← **DoG**
  - two different σ, let it be σ and kσ
- Different octaves of images in Gaussian pyramid
- Local extrema over scale and space
- One pixel is compared with its 8 neighbors and 9 pixels in previous scales

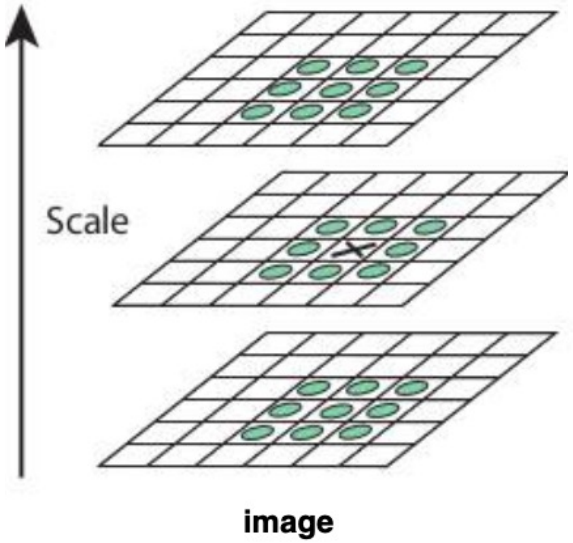
David Lowe

E-mail: lowe@cs.ubc.ca

Distinctive Image Features from Scale-Invariant Keypoints

David G. Lowe
Computer Science Department
University of British Columbia
Vancouver, B.C., Canada
lowe@cs.ubc.ca

January 5, 2004



https://www.cs.ubc.ca/~lowe/papers/ijcv04.pdf

… end of Lecture 12 - Vision