Computer Vision

Bogdan Alexe

bogdan.alexe@fmi.unibuc.ro

University of Bucharest, 2nd semester, 2023-2024

Course structure

1. **Features and filters: low-level vision**

   Linear filters, color, texture, edge detection

2. **Grouping and fitting: mid-level vision**

   Fitting curves and lines, robust fitting, RANSAC, Hough transform, segmentation

3. **Multiple views**

   Local invariant feature and description, epipolar geometry and stereo, object instance recognition

4. **Object Recognition: high – level vision**

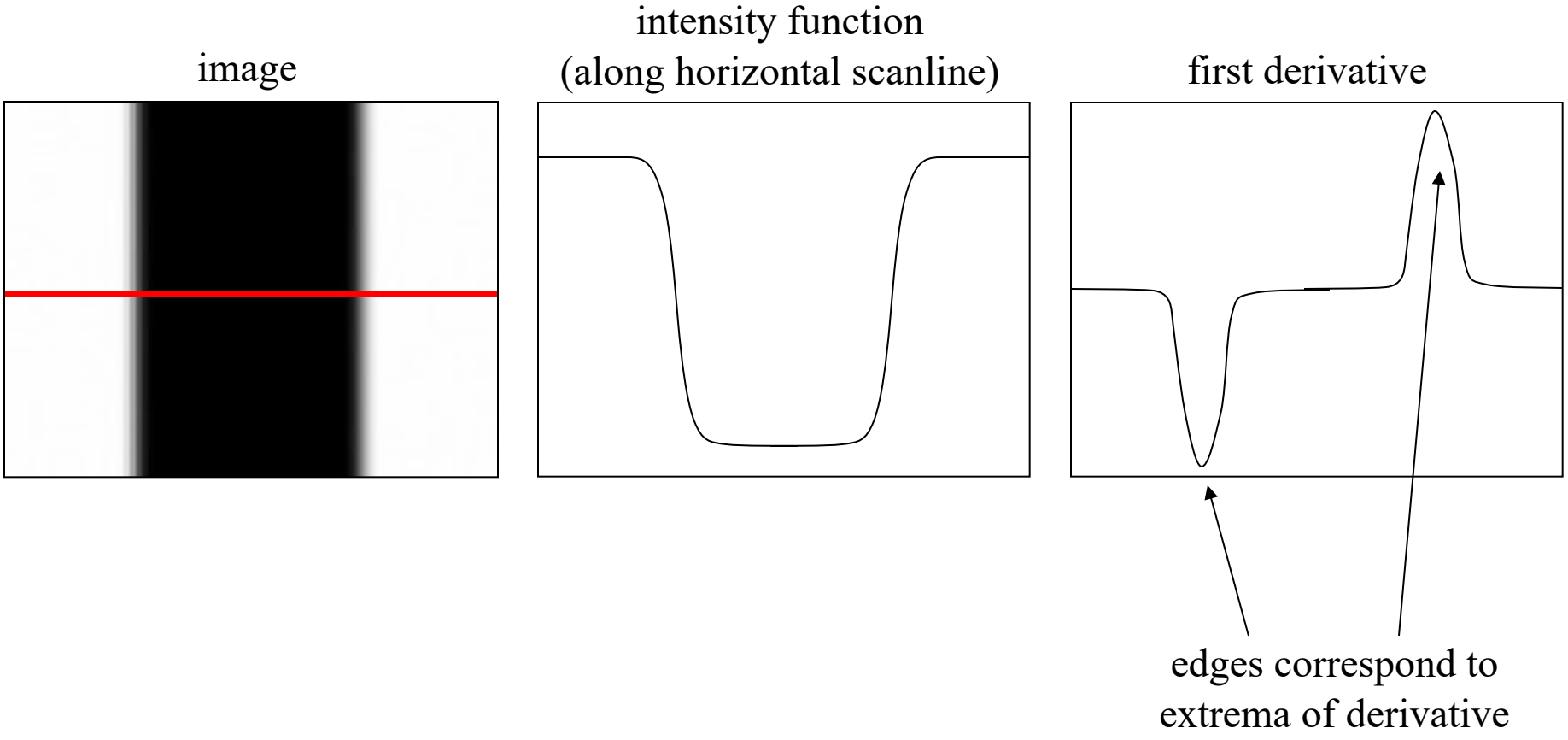
   Object classification, object detection, part based models, bovw models

5. **Video understanding**

   Object tracking, background subtraction, motion descriptors, optical flow

Characterizing edges

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



Derivatives with convolution

For 2D function, f(x,y), the partial derivative is:

$$\frac{\partial f(x,y)}{\partial x} = \lim_{\varepsilon \to 0} \frac{f(x+\varepsilon, y) - f(x,y)}{\varepsilon}$$

For discrete data, we can approximate using finite differences:

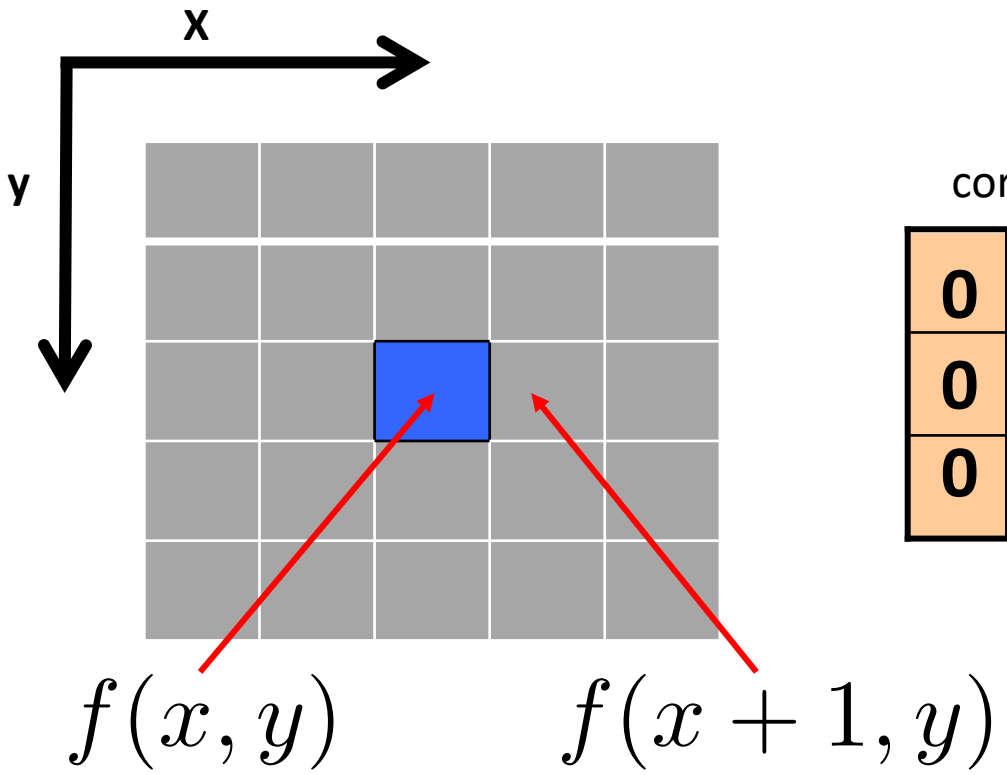
$$\frac{\partial f(x,y)}{\partial x} \approx \frac{f(x+1, y) - f(x,y)}{1}$$

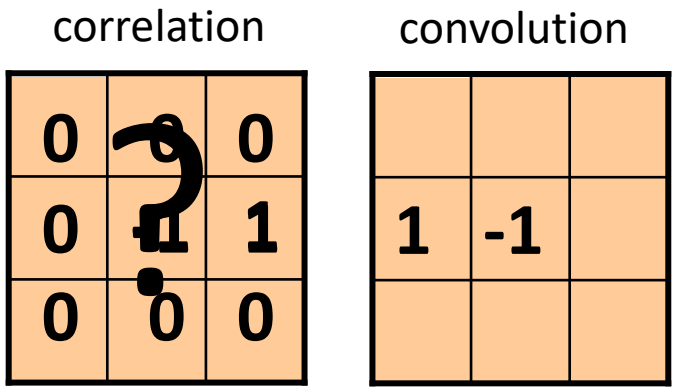
To implement above as convolution, what would be the associated filter?

Slide credit: Kristen Grauman

Derivatives with convolution

$$\frac{\partial f(x,y)}{\partial x} \approx \frac{f(x+1,y) - f(x,y)}{1}$$





Slide credit: Kristen Grauman

Other filters for finite diferences

Prewitt: $M_x =$

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

; $M_y =$

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

Sobel: $M_x =$

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

; $M_y =$

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

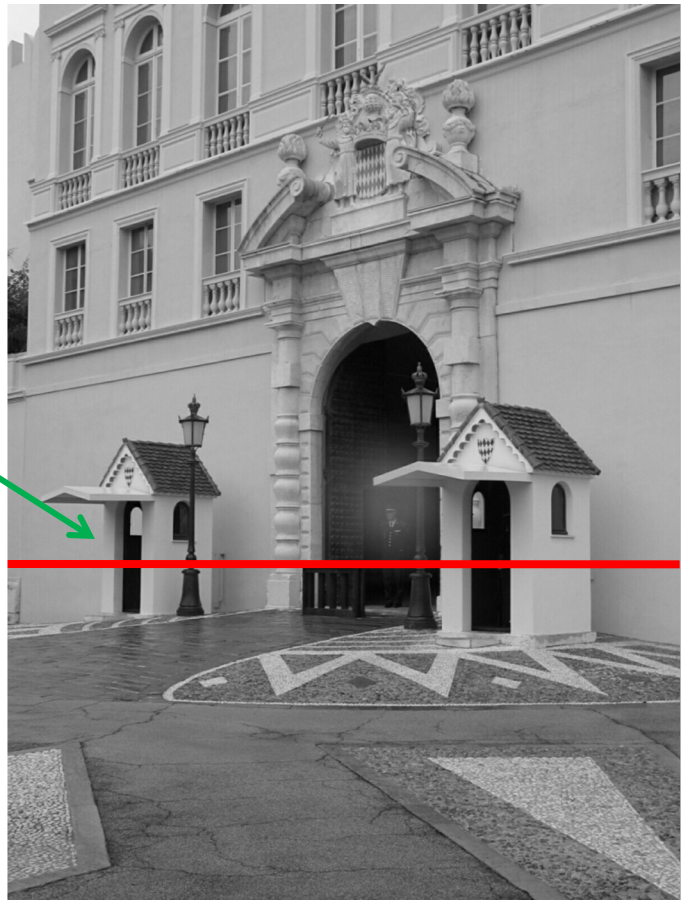
Roberts: $M_x =$

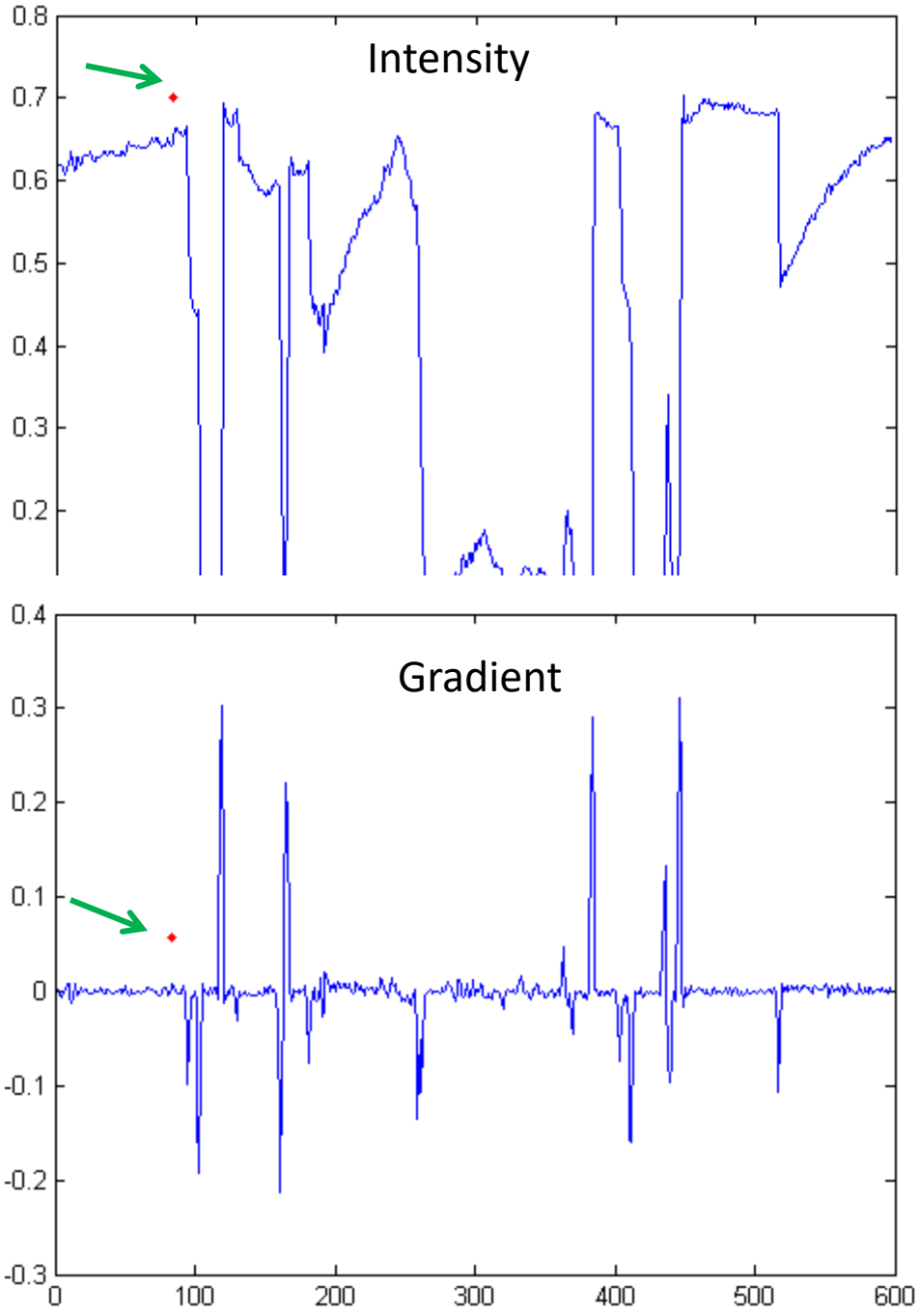
0	1
-1	0

; $M_y =$

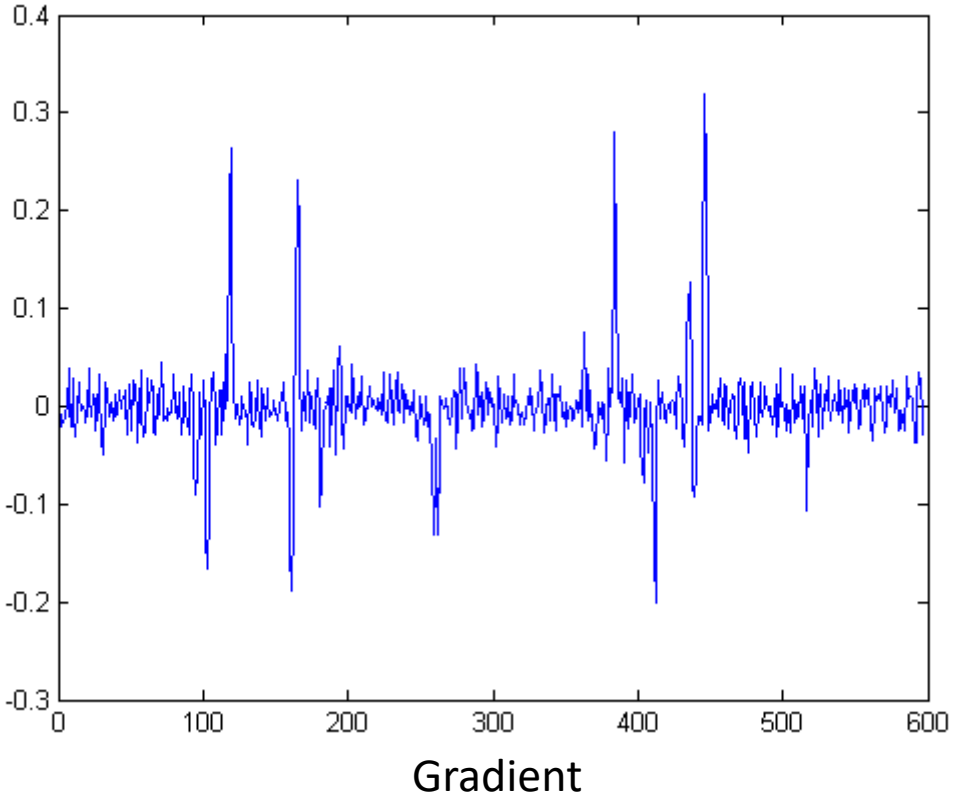
1	0
0	-1

Intensity profile



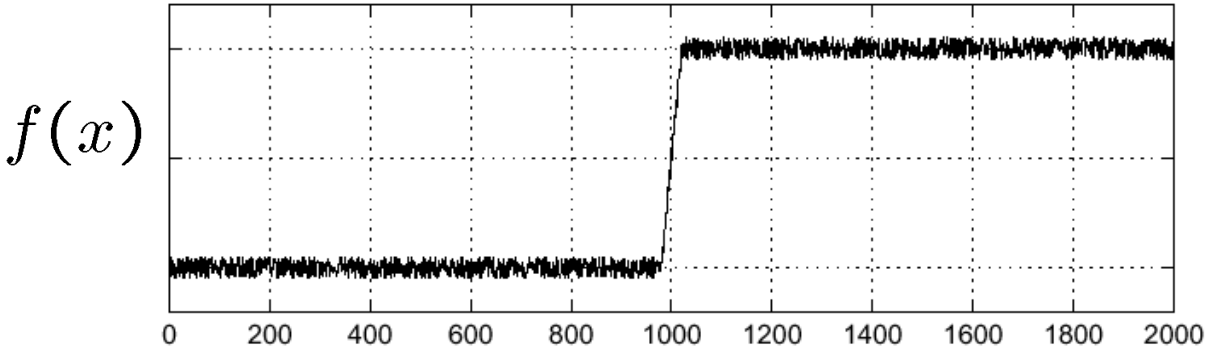


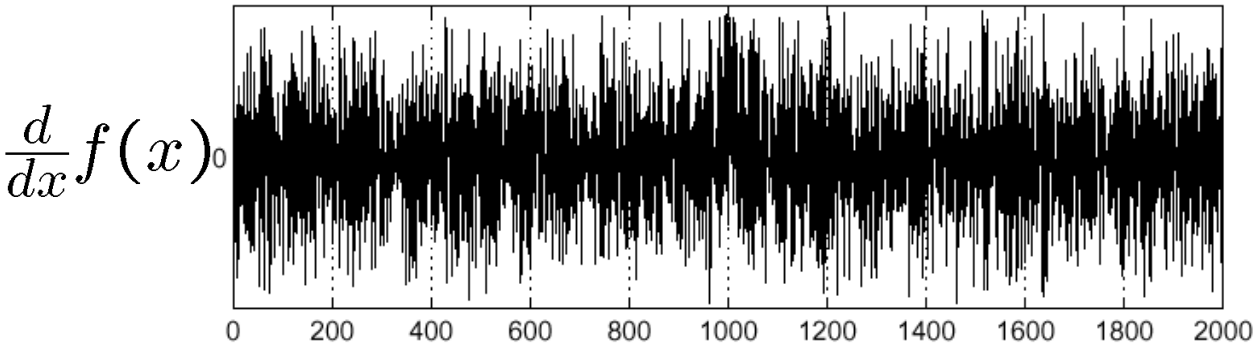
With a little Gaussian noise



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

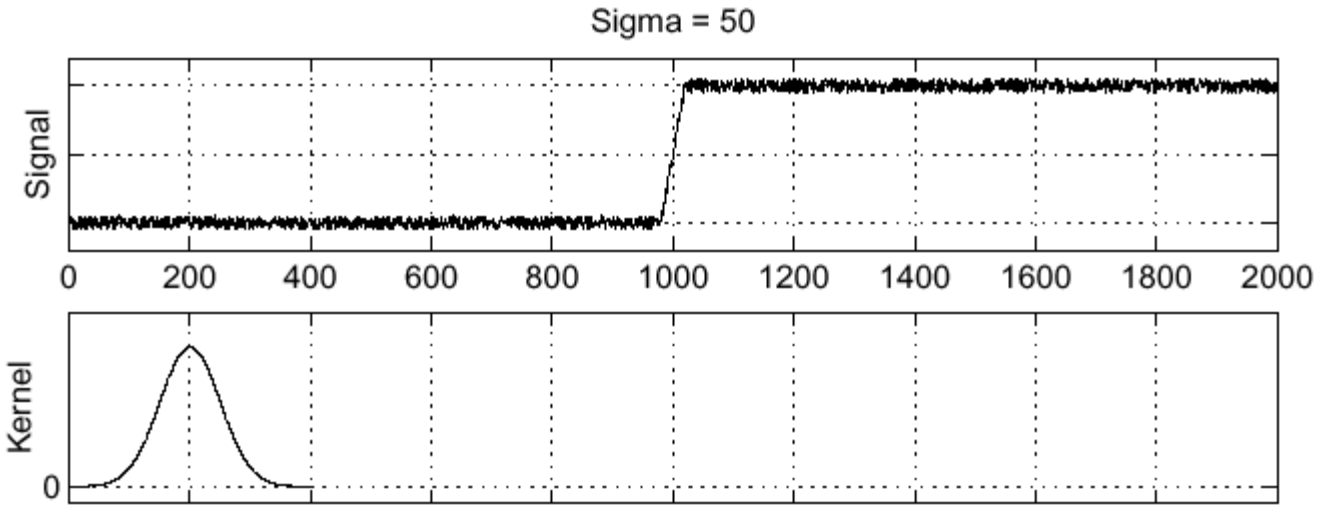
- Difference filters respond strongly to noise
  - Image noise results in pixels that look very different from their neighbors
  - Generally, the larger the noise the stronger the response
- What can we do about it?

Source: D. Forsyth

Solution: smooth first

f

h



Where is the edge?

Look for peaks in $\frac{\partial}{\partial x}(h \star f)$

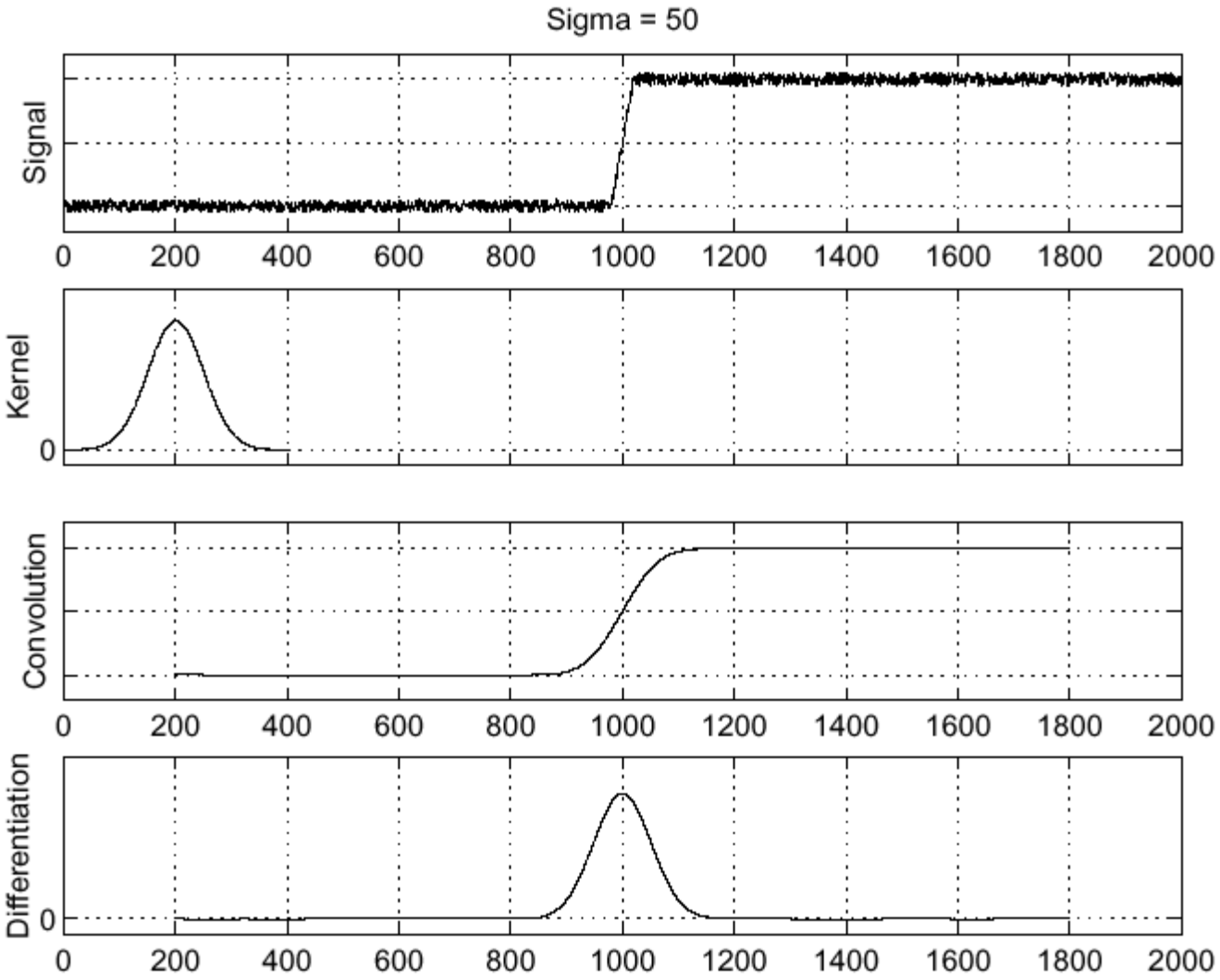
Solution: smooth first

f

h

$h \star f$

$\frac{\partial}{\partial x}(h \star f)$



Where is the edge?

Look for peaks in $\frac{\partial}{\partial x}(h \star f)$

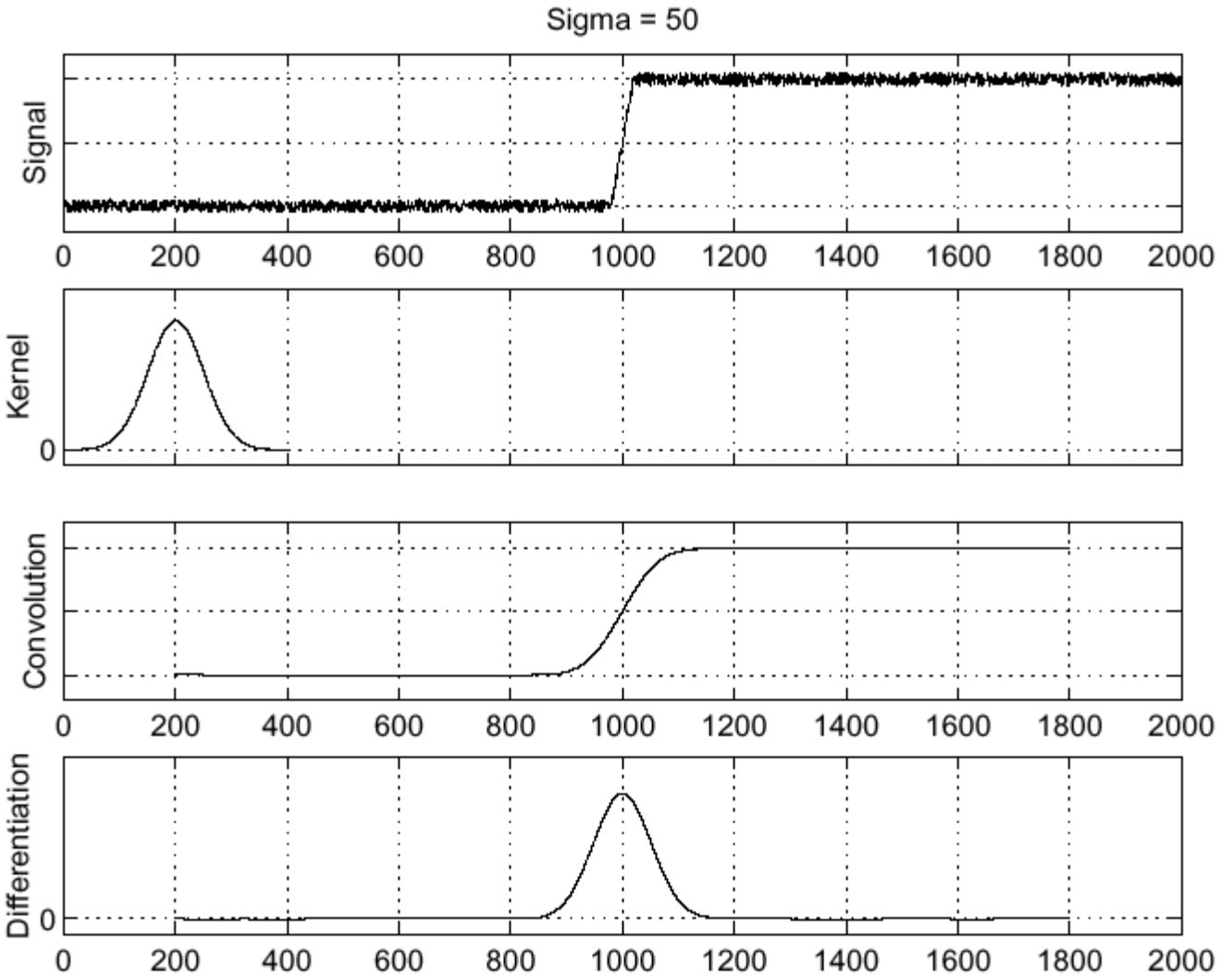
Solution: smooth first

f

h

$h \star f$

$\frac{\partial}{\partial x}(h \star f)$



Where is the edge?

Look for peaks in $\frac{\partial}{\partial x}(h \star f)$ = zero's of the function $\frac{\partial^2}{\partial x^2}(h \star f)$

Derivative theorem of convolution

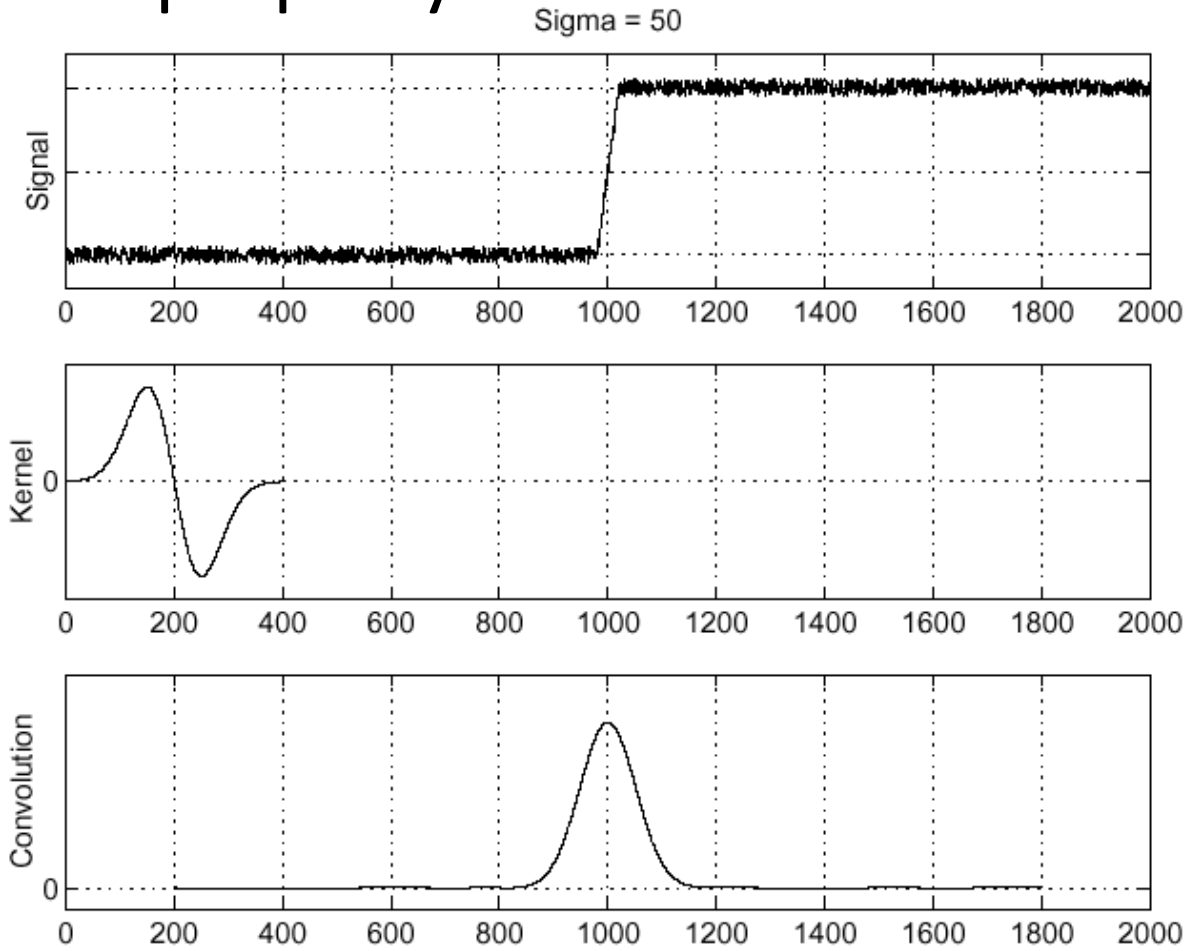
$$\frac{\partial}{\partial x}(h \star f) = (\frac{\partial}{\partial x}h) \star f$$

Differentiation property of convolution.

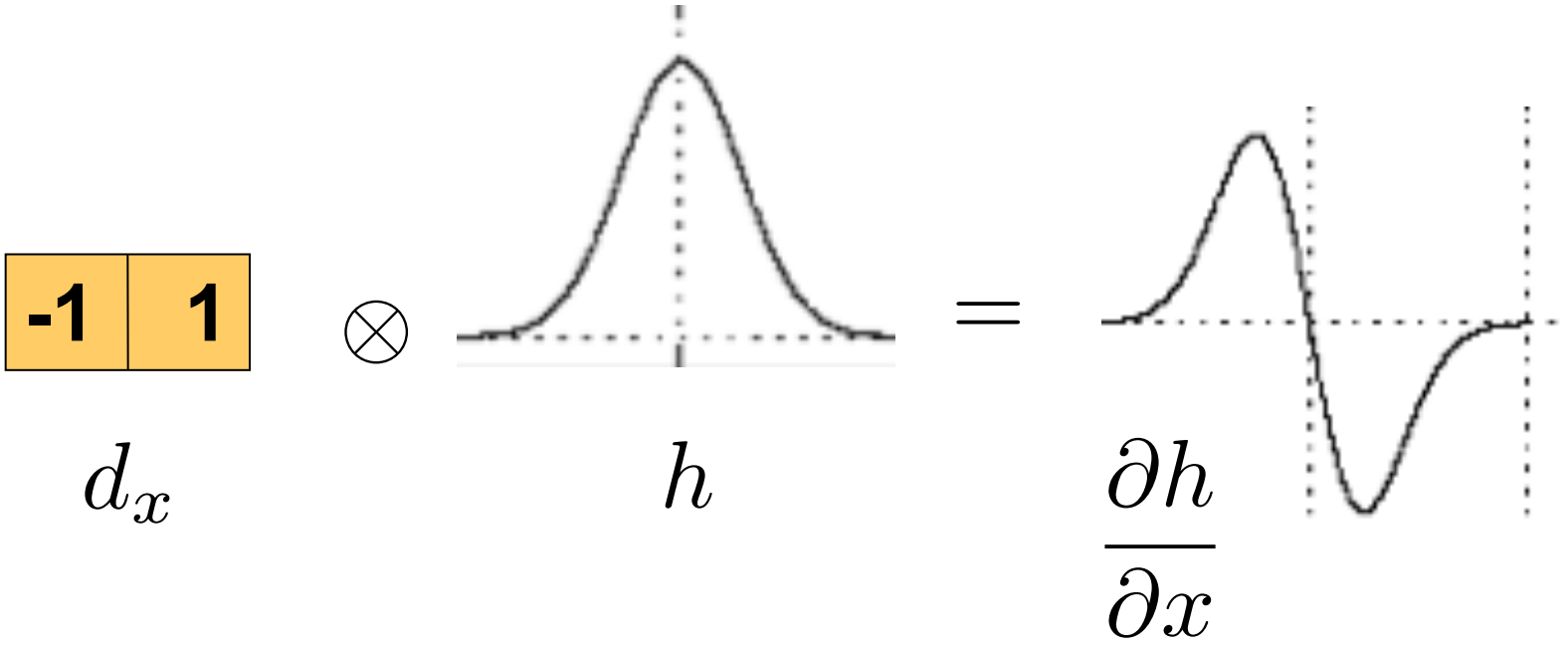
f

$\frac{\partial}{\partial x}h$

$(\frac{\partial}{\partial x}h) \star f$



1D



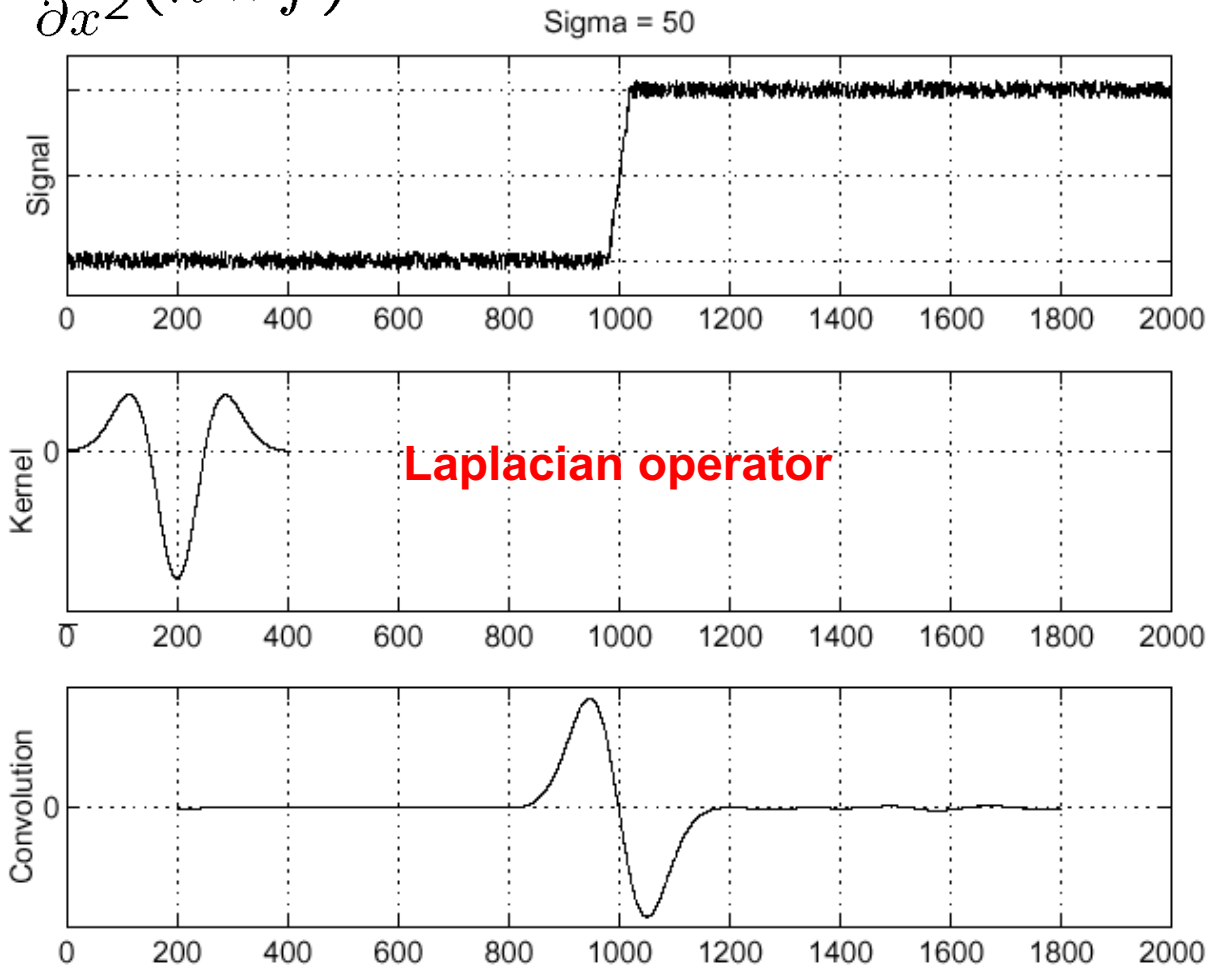
Laplacian of a Gaussian – 1D

Consider $\frac{\partial^2}{\partial x^2}(h \star f)$

f

$\frac{\partial^2}{\partial x^2}h$

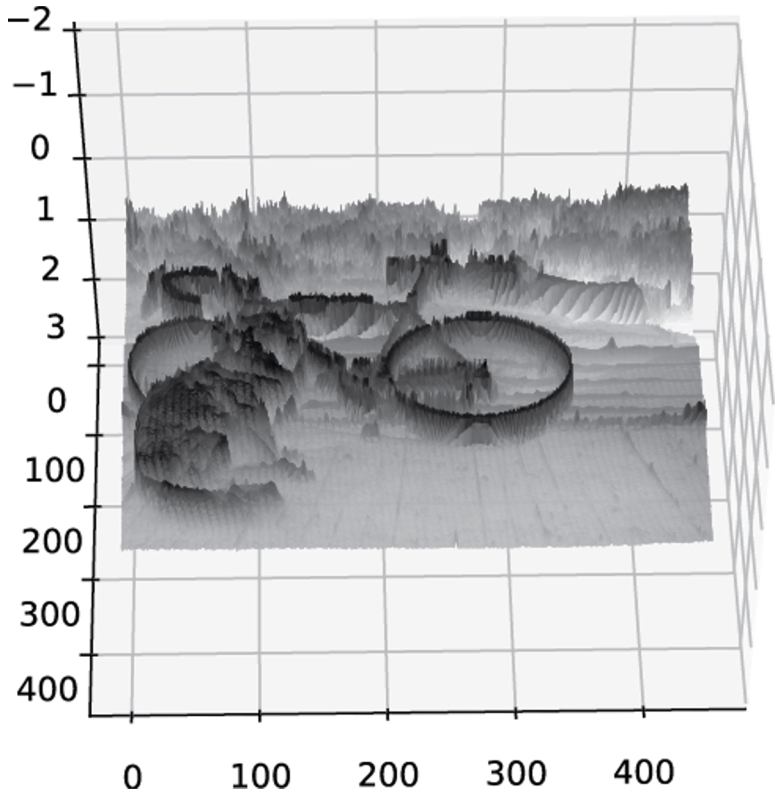
$(\frac{\partial^2}{\partial x^2}h) \star f$



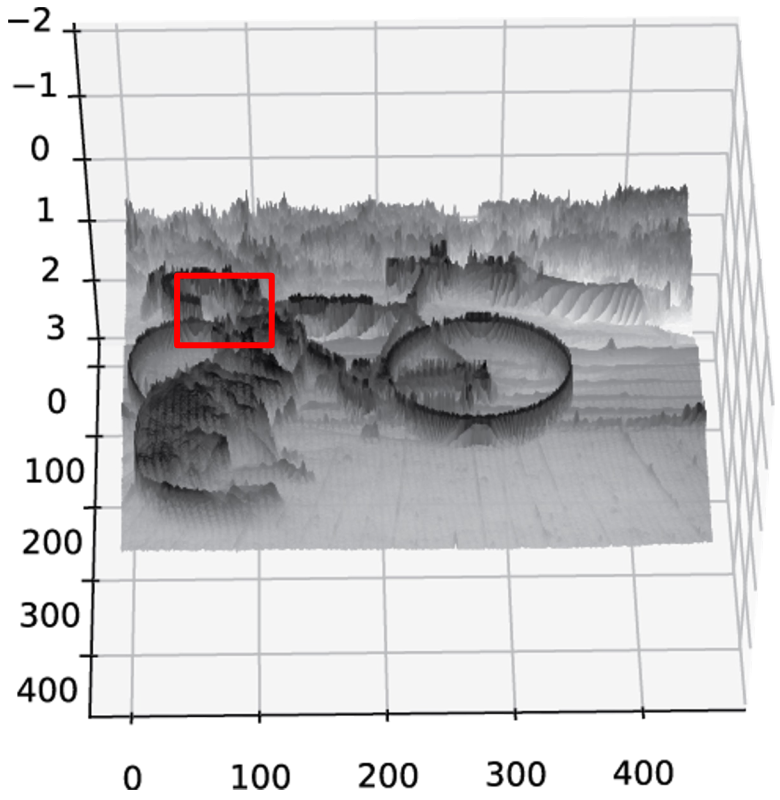
Where is the edge?

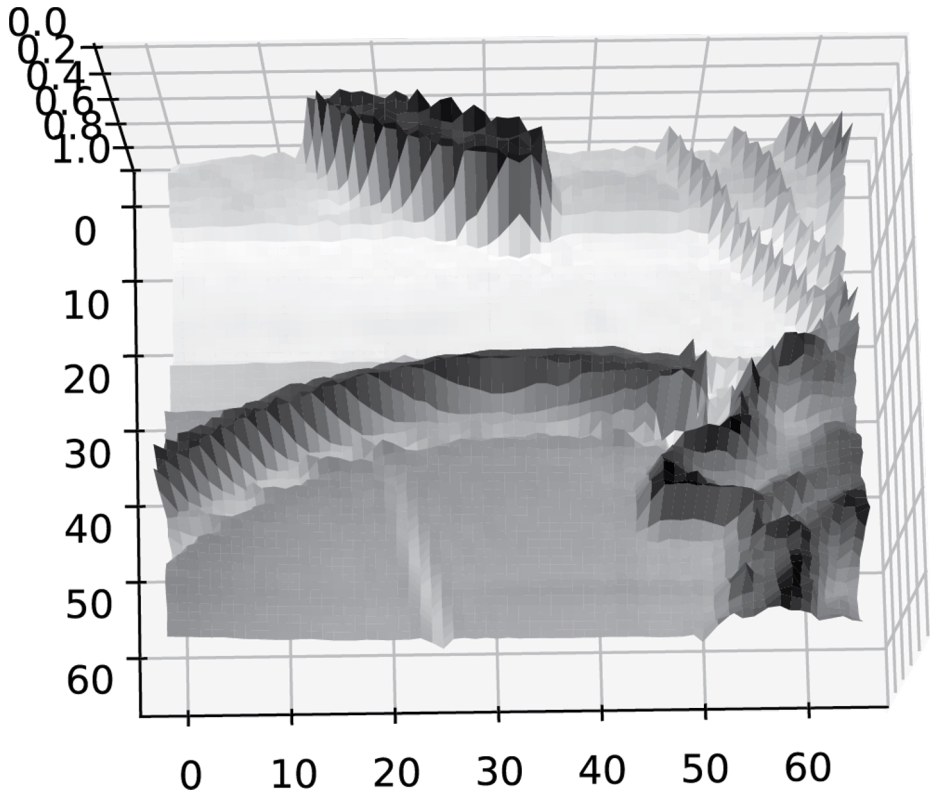
Zero-crossing of the function $(\frac{\partial^2}{\partial x^2}h) \star f$

2D: Images are noisy

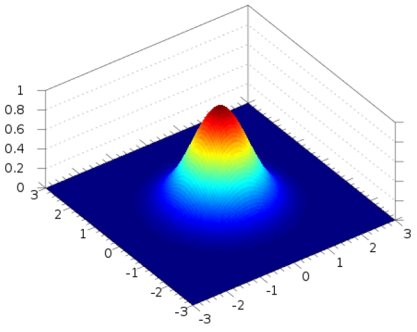


2D: Images are noisy

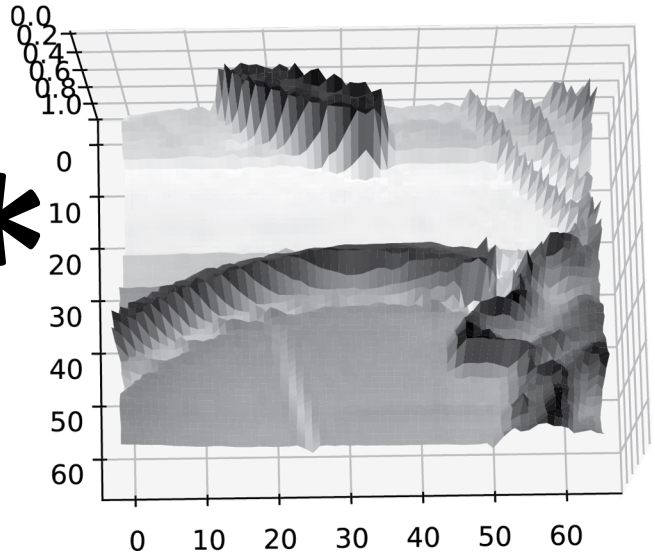




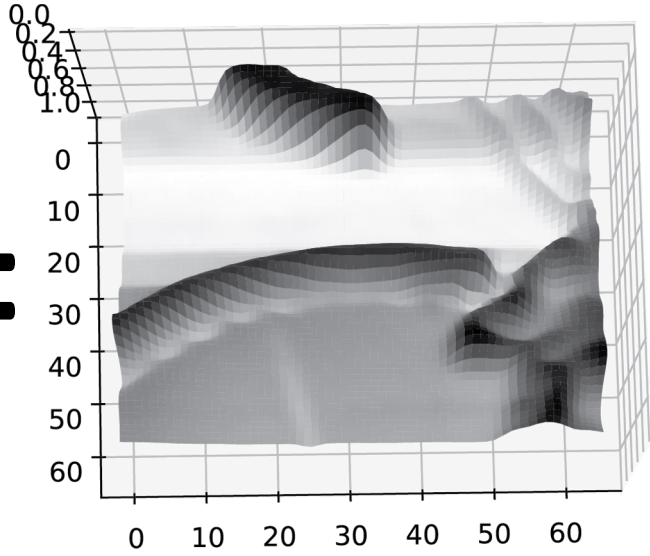
Solution: smooth first



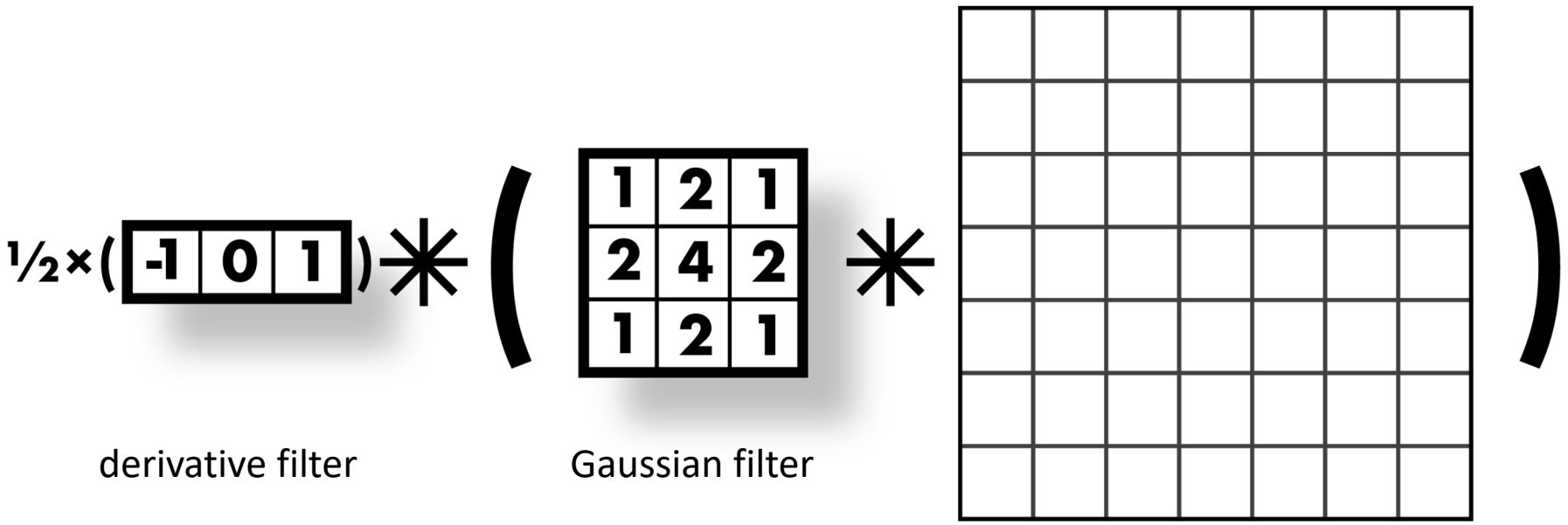
*



=



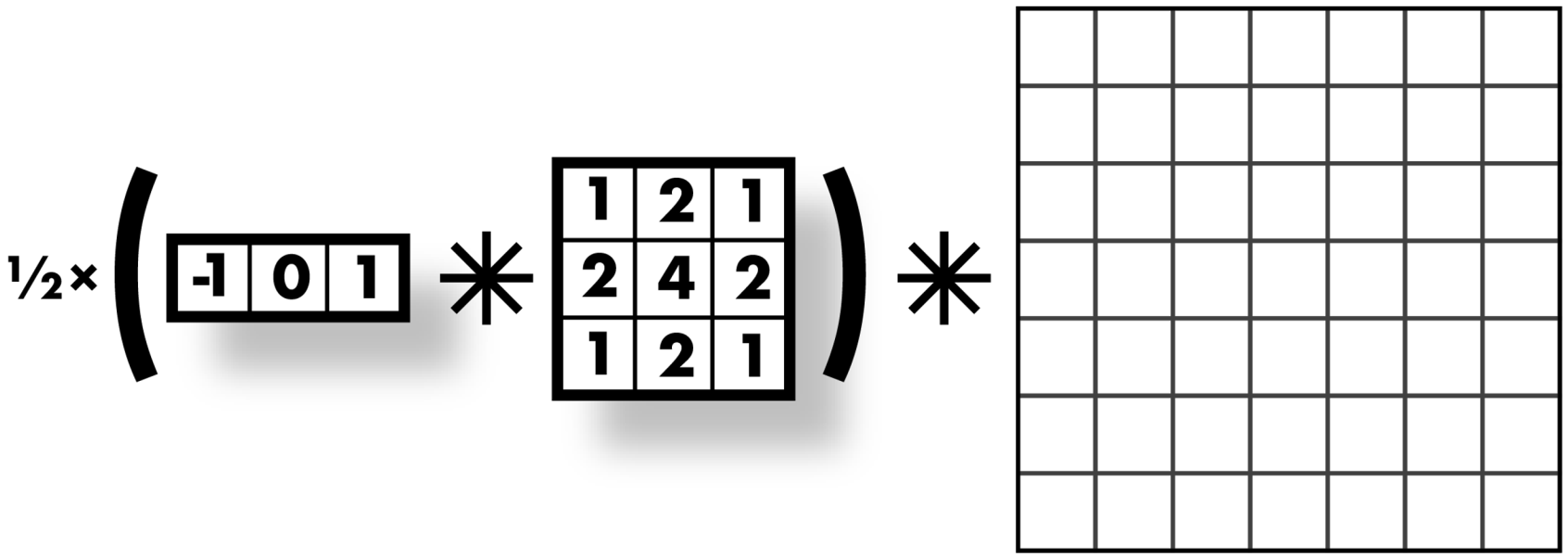
Smooth first, then derivative



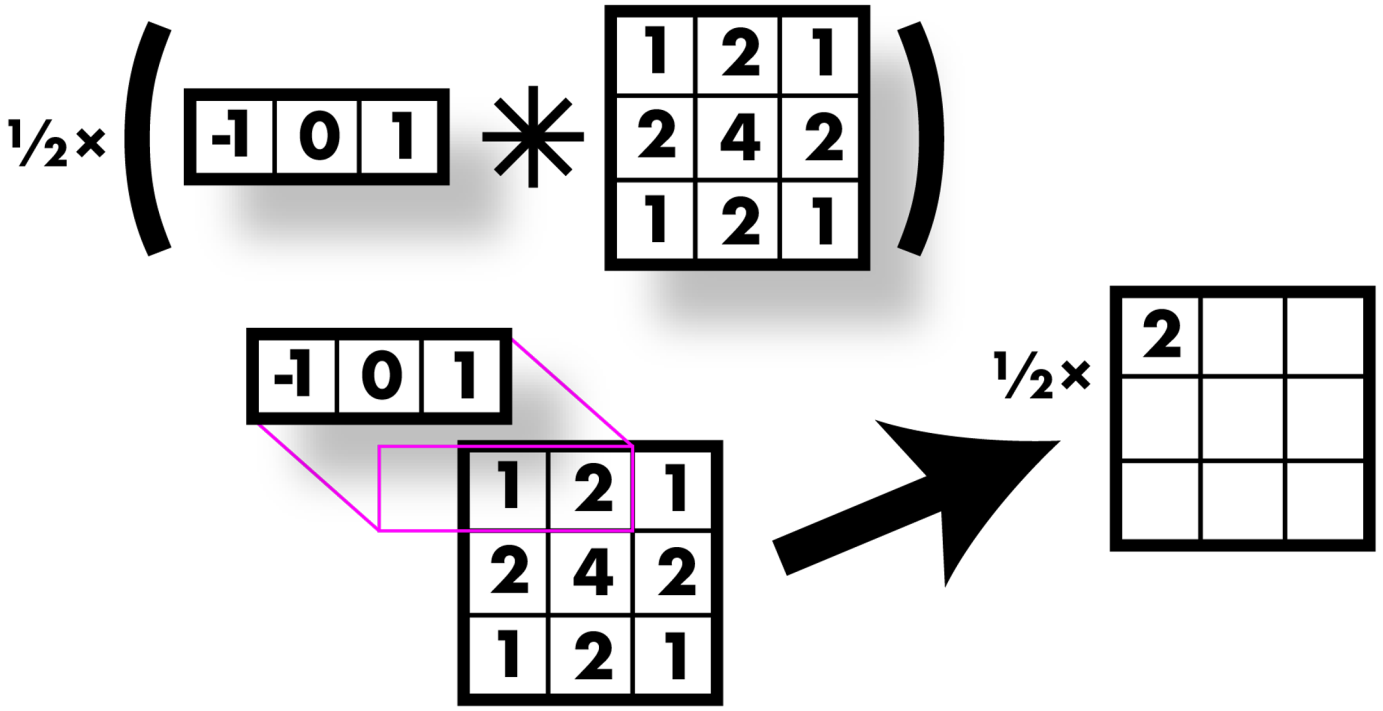
derivative filter

Gaussian filter

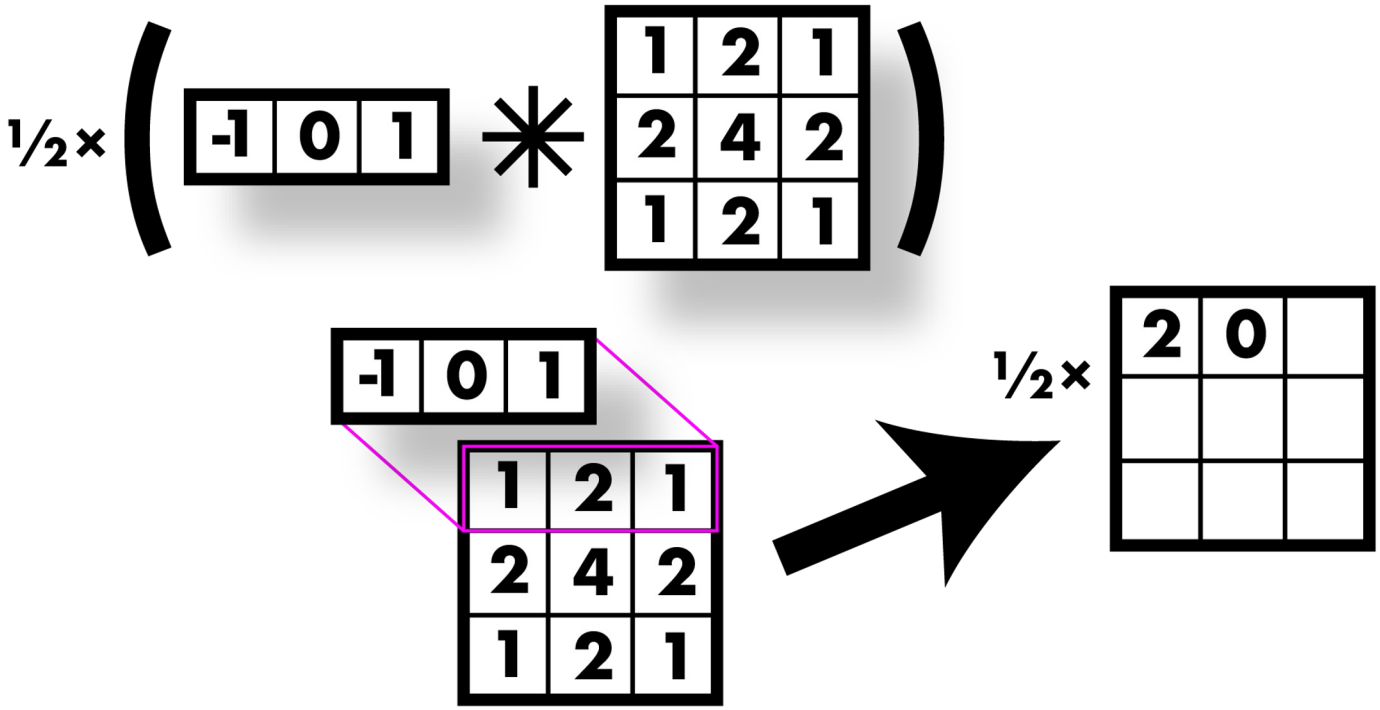
Smooth first, then derivative



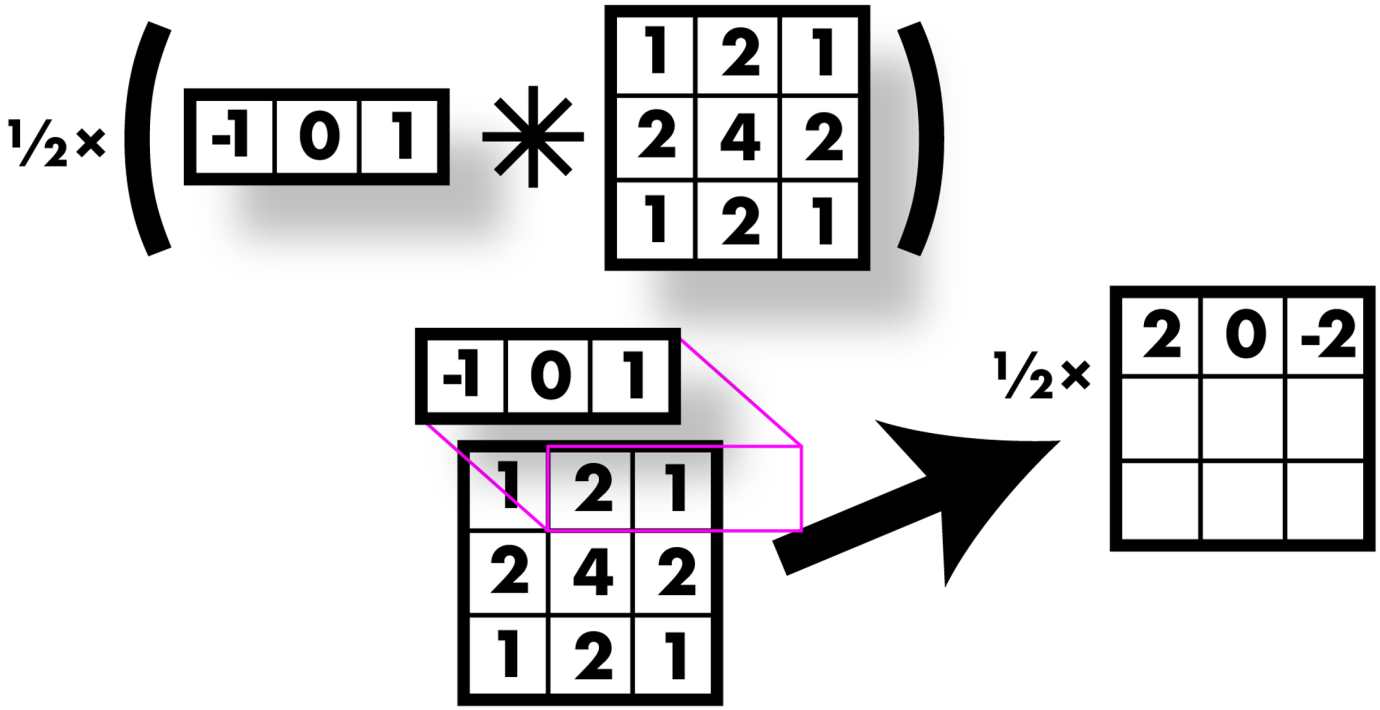
Smooth first, then derivative



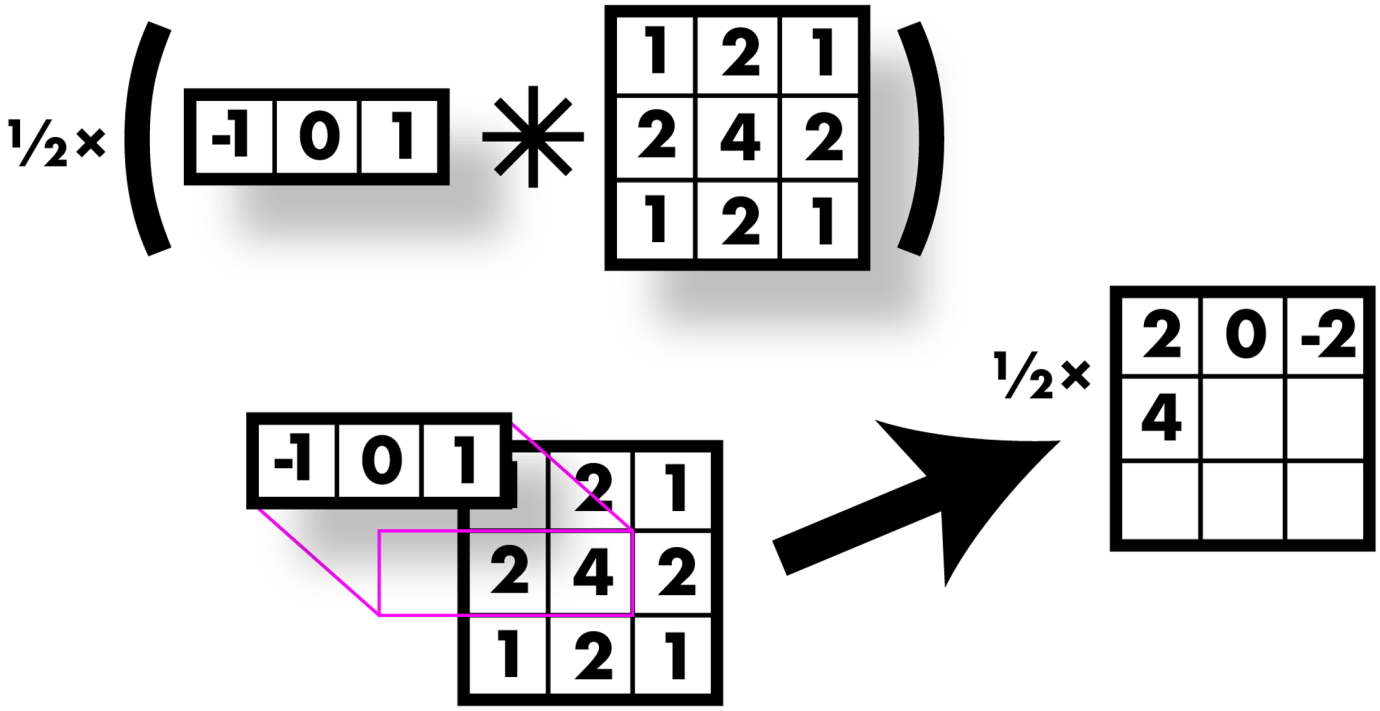
Smooth first, then derivative



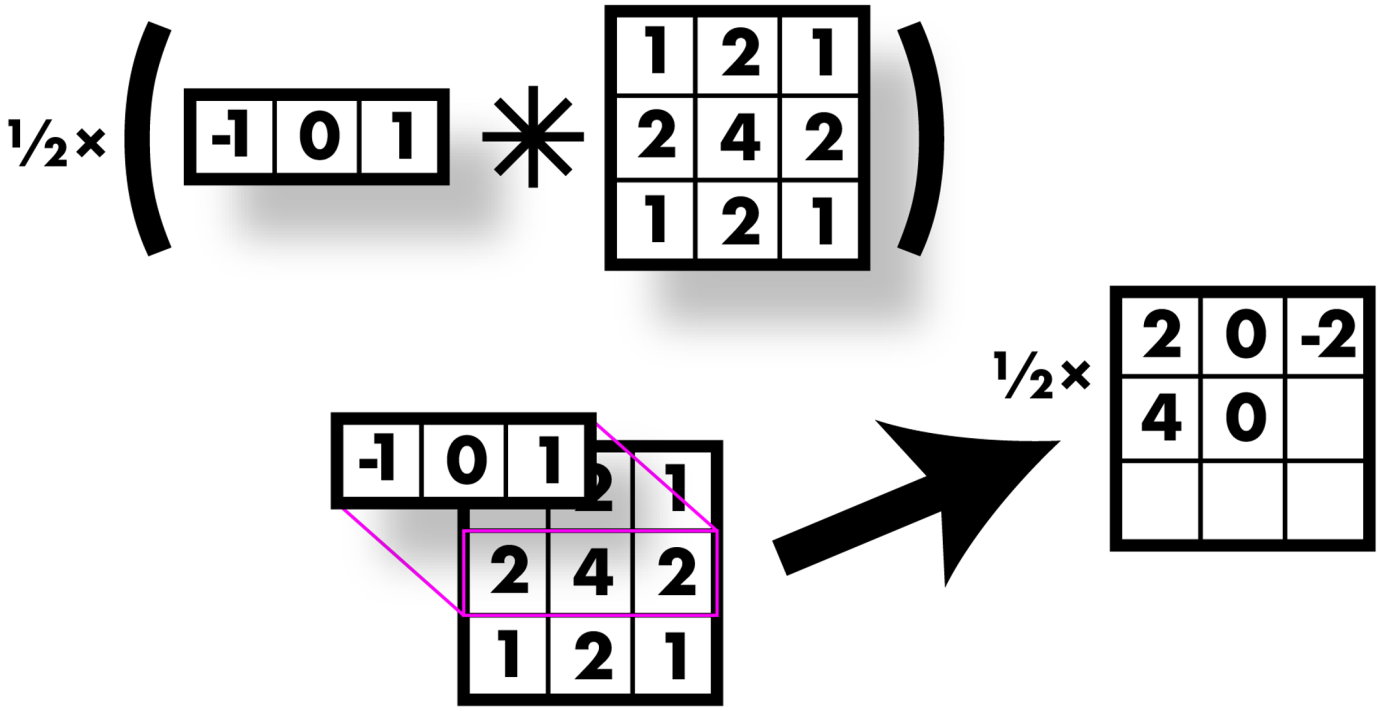
Smooth first, then derivative



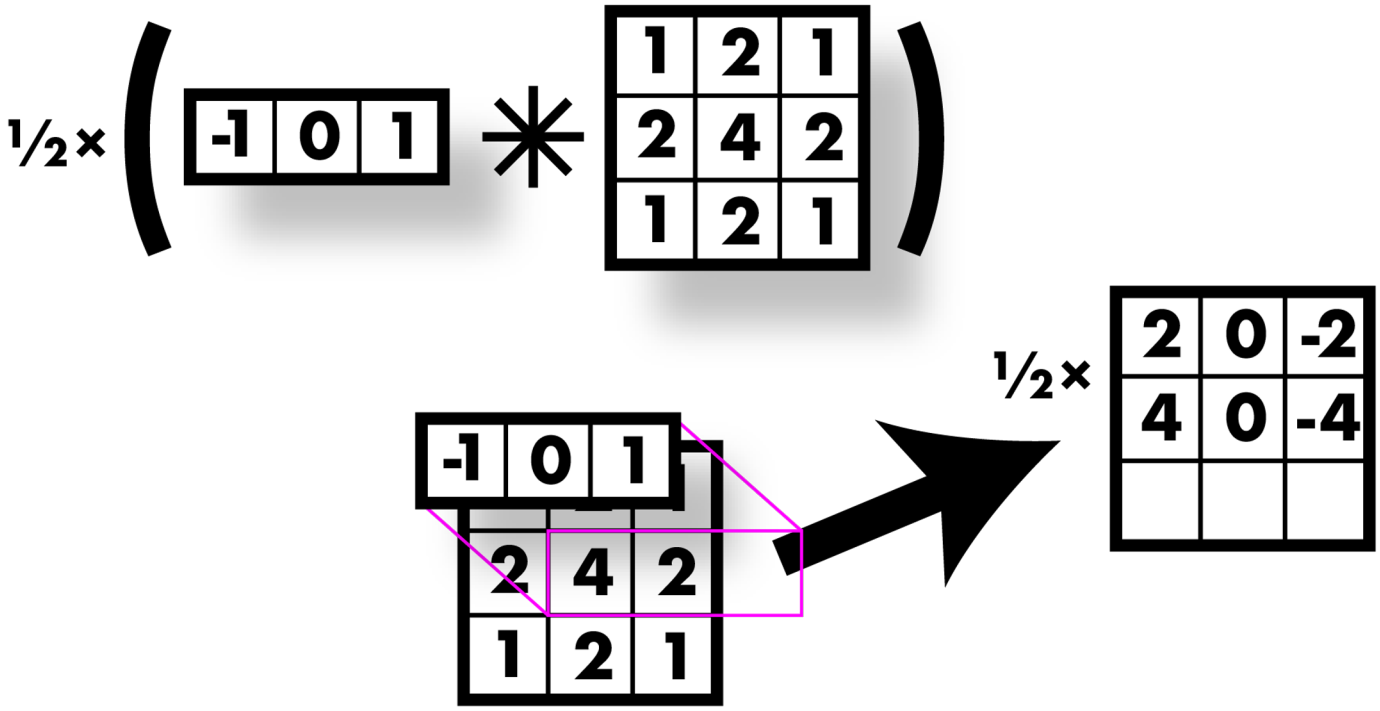
Smooth first, then derivative



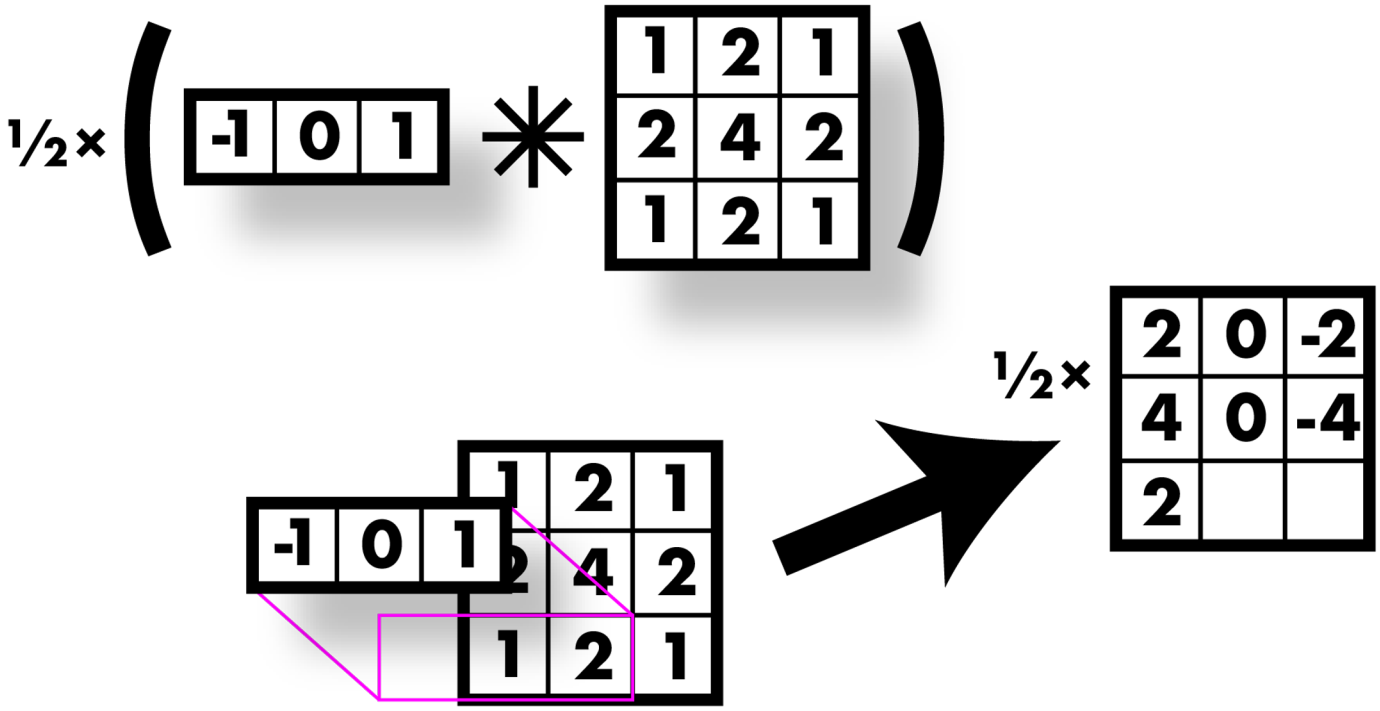
Smooth first, then derivative



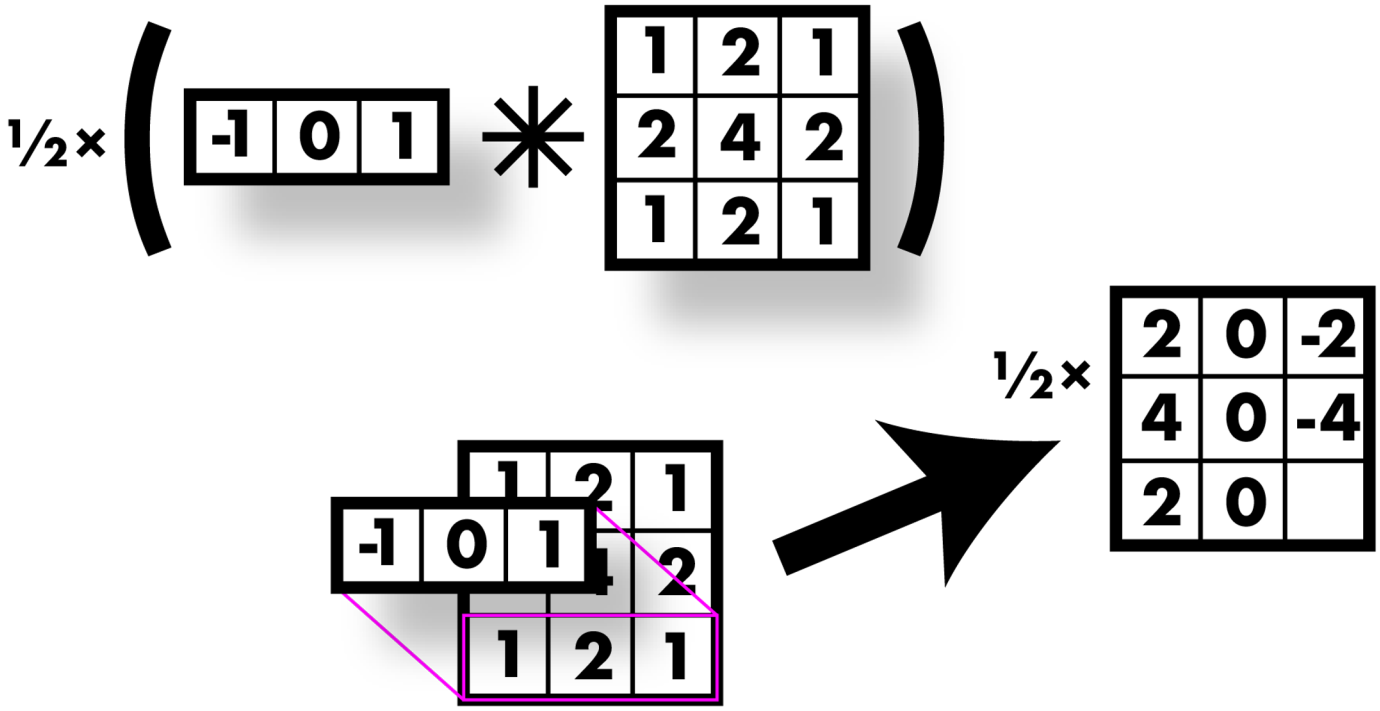
Smooth first, then derivative



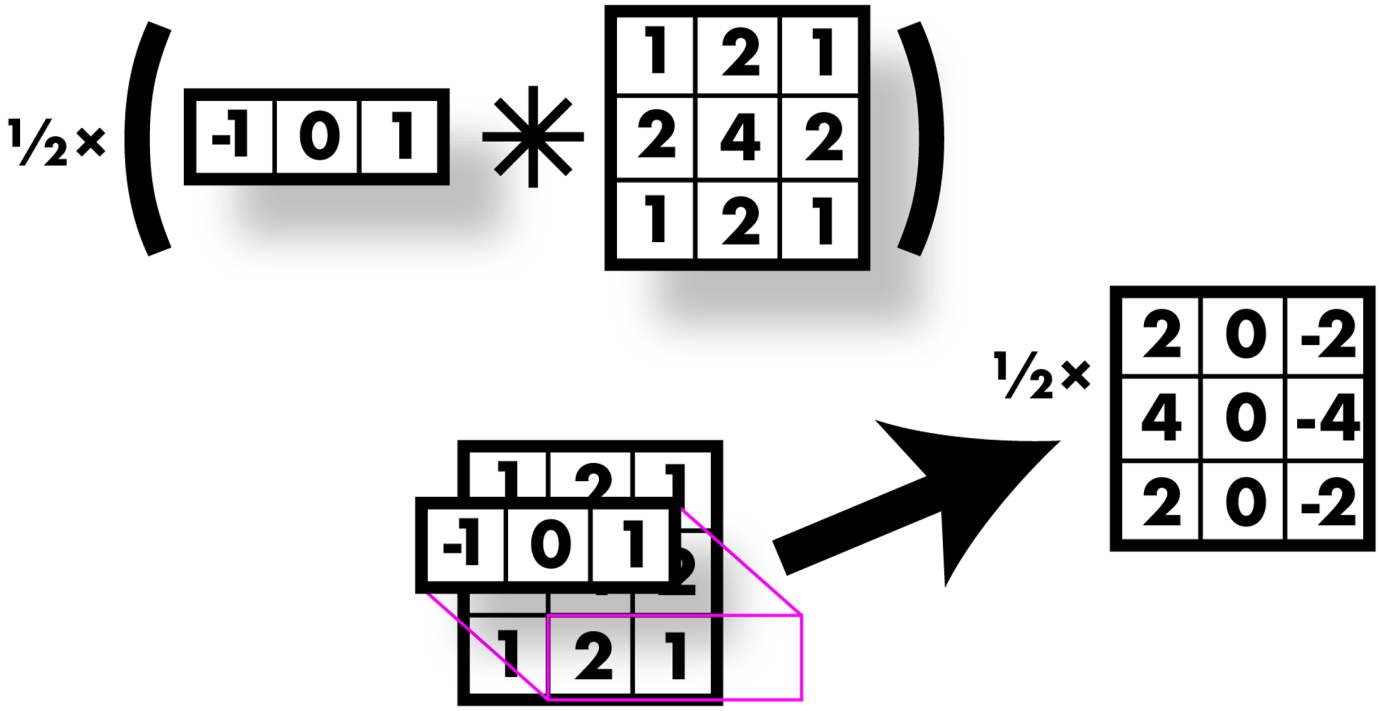
Smooth first, then derivative



Smooth first, then derivative

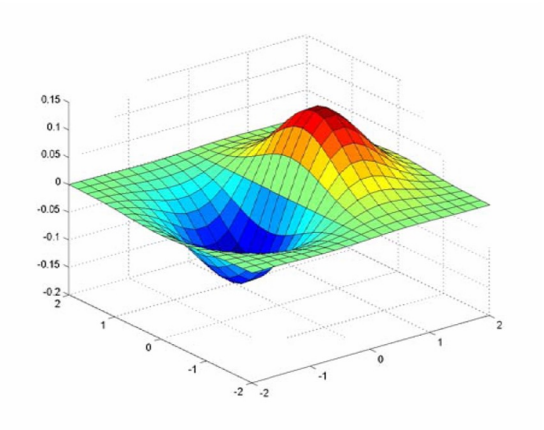


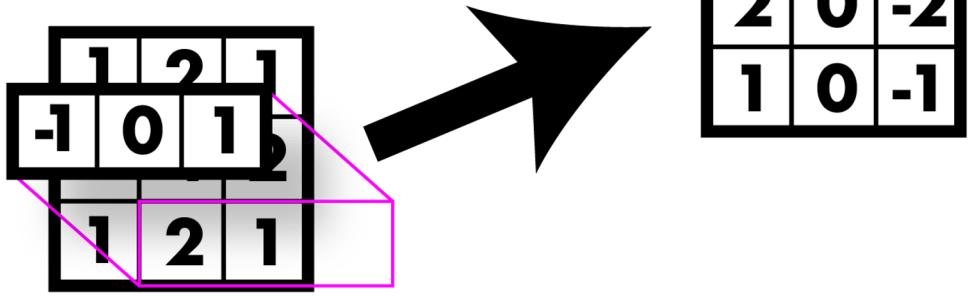
Smooth first, then derivative



Sobel filter: smooth & derivative

$$\tfrac{1}{2} \times \left(\begin{bmatrix} -1 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \right)$$





Derivative of Gaussian filters - 2D

$$d_x \otimes (h \otimes I) = (d_x \otimes h) \otimes I$$

derivation filter

Gaussian filter

image

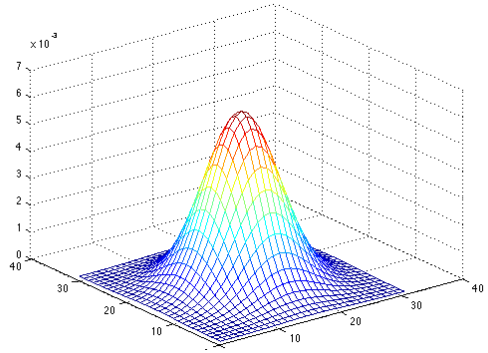
0	0	0
-1	0	1
0	0	0

$\otimes$

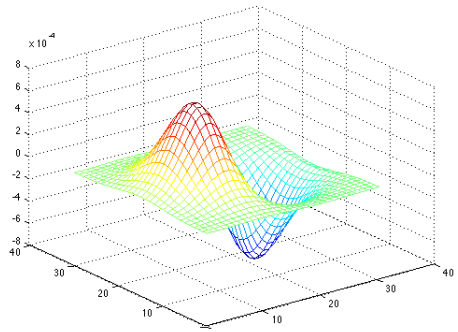
$$\begin{bmatrix} 0.0030 & 0.0133 & 0.0219 & 0.0133 & 0.0030 \\ 0.0133 & 0.0596 & 0.0983 & 0.0596 & 0.0133 \\ 0.0219 & 0.0983 & 0.1621 & 0.0983 & 0.0219 \\ 0.0133 & 0.0596 & 0.0983 & 0.0596 & 0.0133 \\ 0.0030 & 0.0133 & 0.0219 & 0.0133 & 0.0030 \end{bmatrix}$$

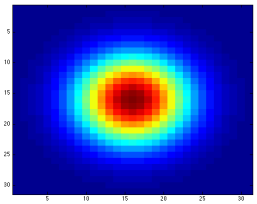
0	0	0
-1	0	1
0	0	0

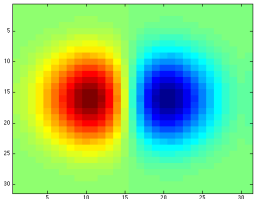
$\otimes$



$=$







Derivative of Gaussian filters - 2D

$$d_y \otimes (h \otimes I) = (d_y \otimes h) \otimes I$$

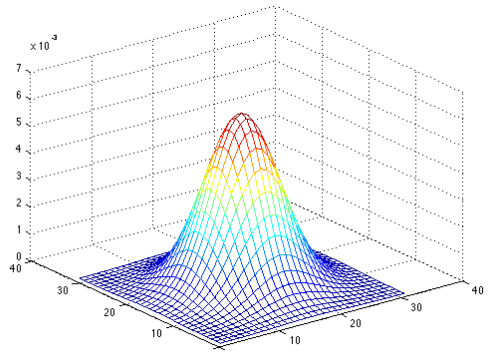
0	-1	0
0	0	0
0	1	0

$\otimes$

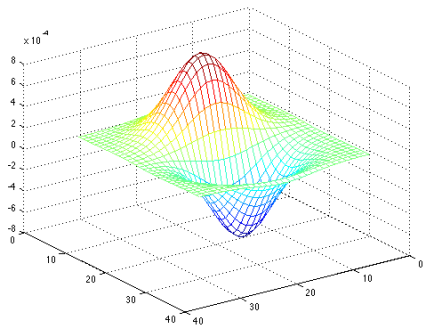
$$\begin{bmatrix} 0.0030 & 0.0133 & 0.0219 & 0.0133 & 0.0030 \\ 0.0133 & 0.0596 & 0.0983 & 0.0596 & 0.0133 \\ 0.0219 & 0.0983 & 0.1621 & 0.0983 & 0.0219 \\ 0.0133 & 0.0596 & 0.0983 & 0.0596 & 0.0133 \\ 0.0030 & 0.0133 & 0.0219 & 0.0133 & 0.0030 \end{bmatrix}$$

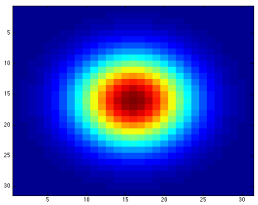
0	-1	0
0	0	0
0	1	0

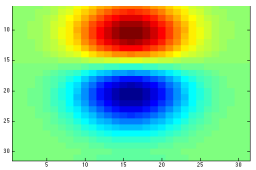
$\otimes$



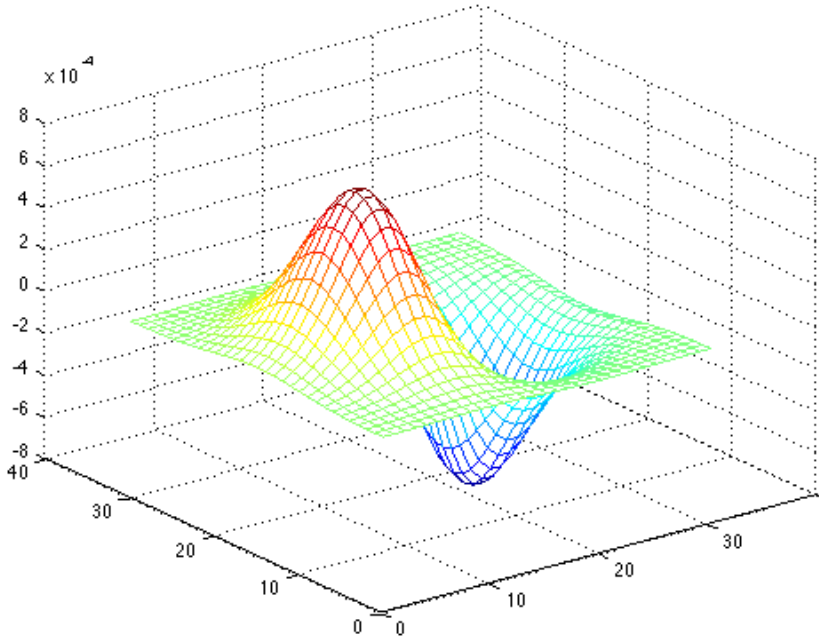
$=$

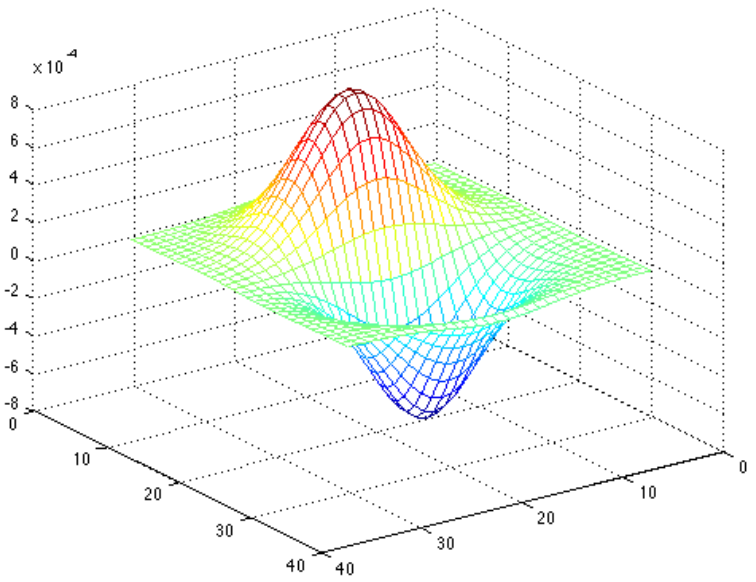






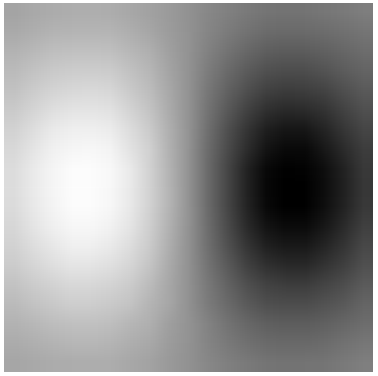
Derivative of Gaussian filters

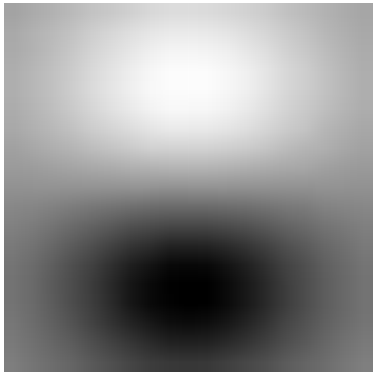




x - direction

y - direction



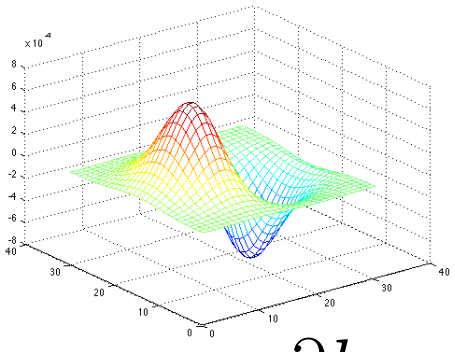


Derivative of Gaussian filters - 2D

0	0	0
-1	0	1
0	0	0

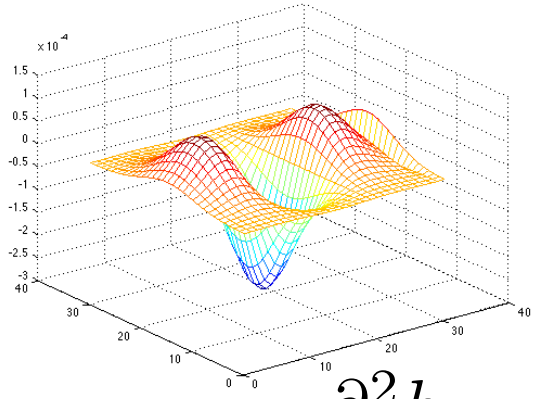
direction x

$\otimes$



$\frac{\partial h}{\partial x}$

$=$

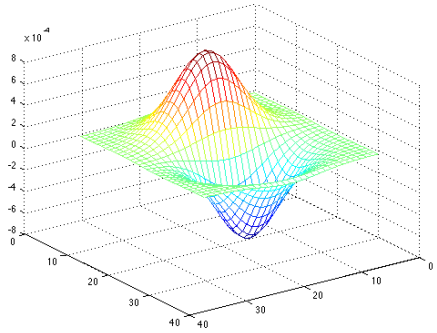


$\frac{\partial^2 h}{\partial x^2}$

0	-1	0
0	0	0
0	1	0

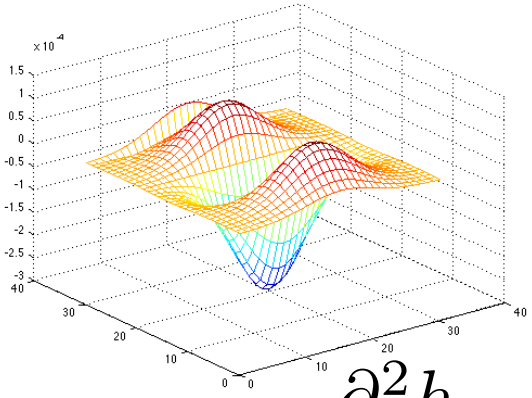
direction y

$\otimes$



$\frac{\partial h}{\partial y}$

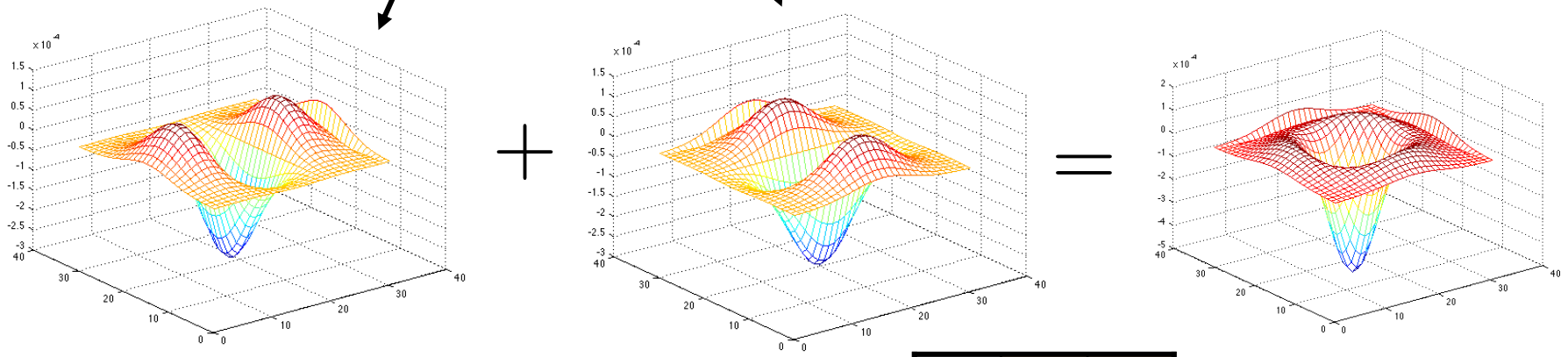
$=$



$\frac{\partial^2 h}{\partial y^2}$

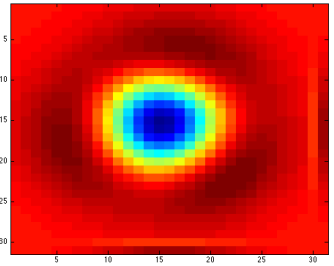
Laplacian of Gaussian

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



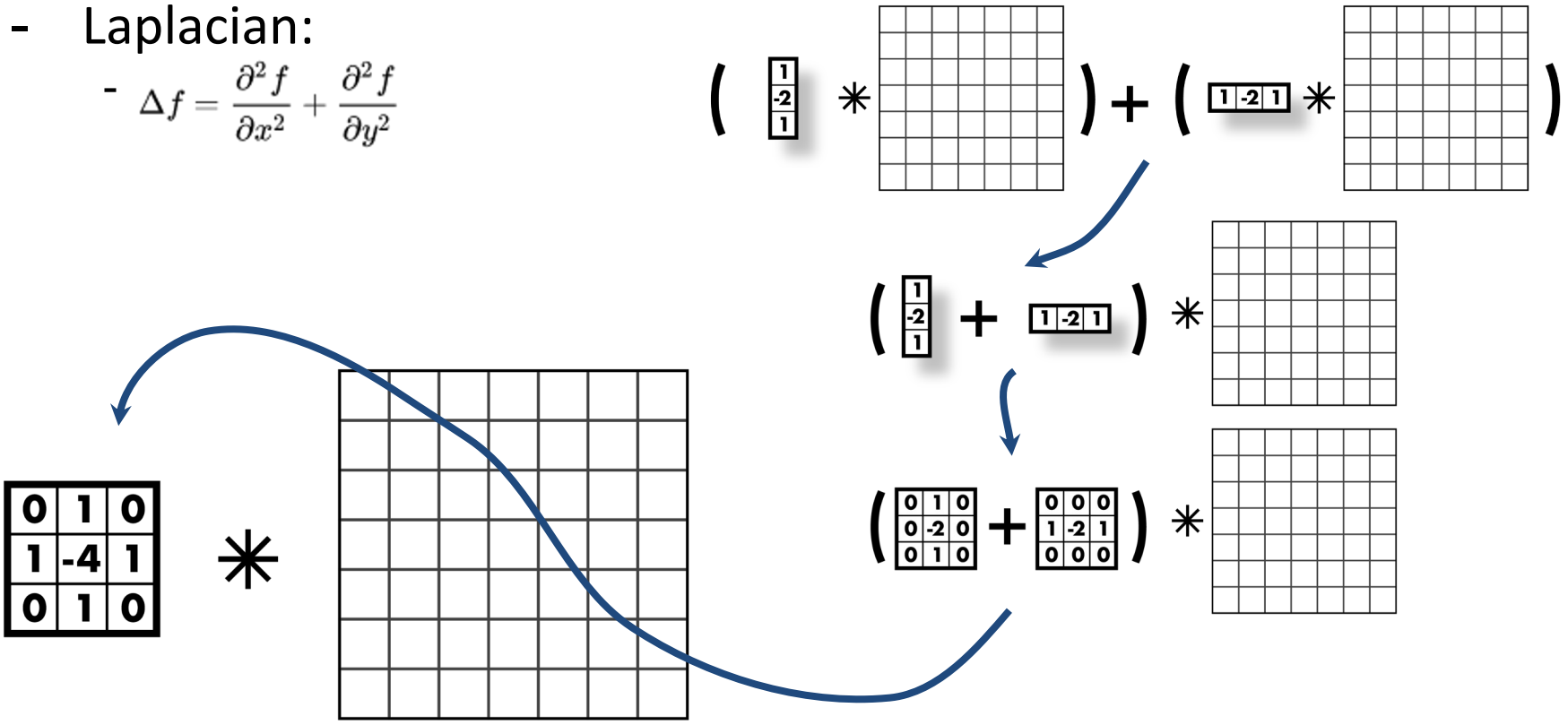
0	1	0
1	-4	1
0	1	0

Example of a Laplacian filter

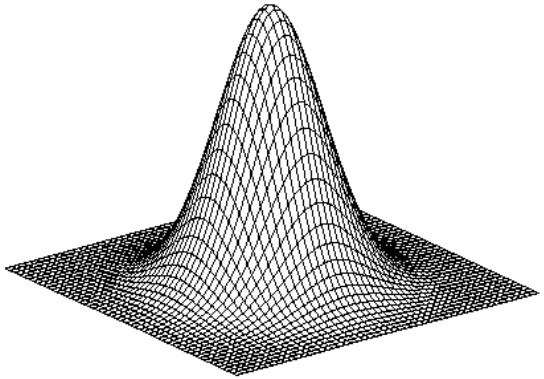


Laplacian of Gaussian

- Laplacian:
  - $\Delta f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$

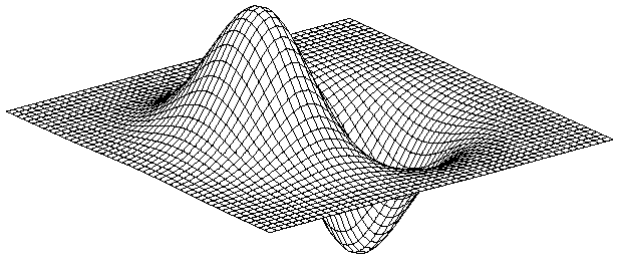


2D edge detection filters



Gaussian

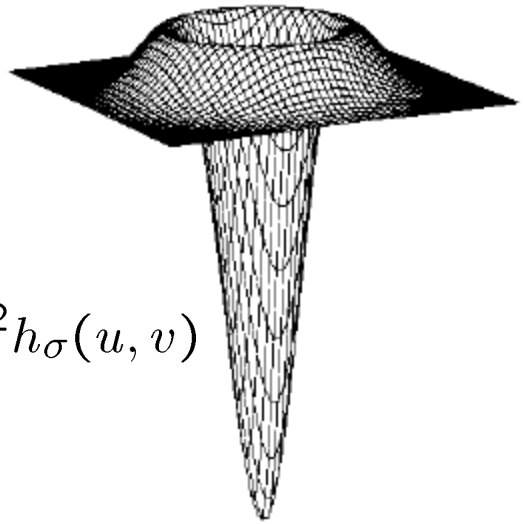
$$h_\sigma(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$



derivative of Gaussian

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

Laplacian of Gaussian



$$\nabla^2 h_\sigma(u,v)$$

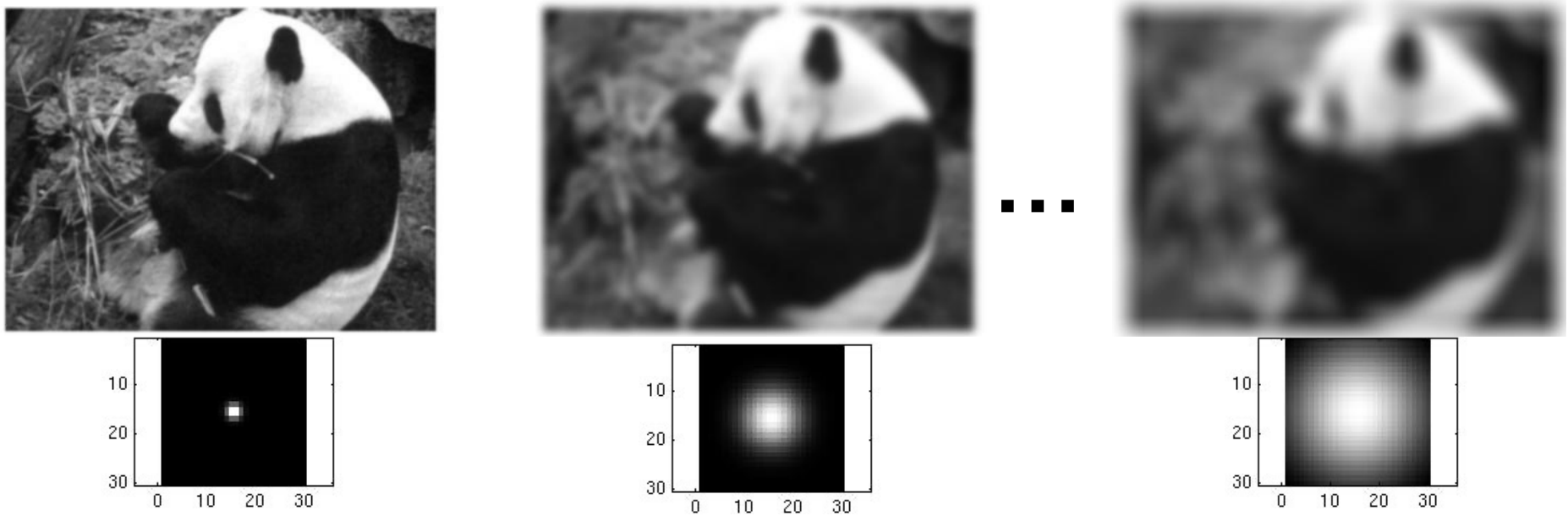
- ∇^2 is the Laplacian operator:

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

Slide credit: Steve Seitz

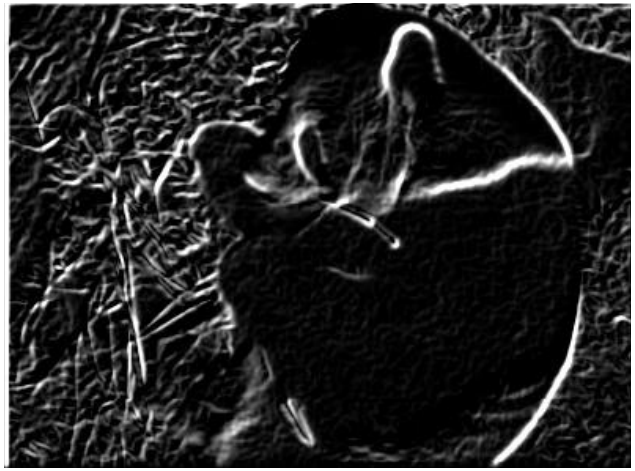
Smoothing with a Gaussian

Recall: parameter σ is the "scale" / "width" / "spread" of the Gaussian kernel, and controls the amount of smoothing.



Slide credit: Kristen Grauman

Effect of σ on derivatives



σ = 1 pixel

σ = 3 pixels

The apparent structures differ depending on Gaussian's scale parameter.

Larger values: larger scale edges detected
Smaller values: finer features detected

Slide credit: Kristen Grauman

Mask properties

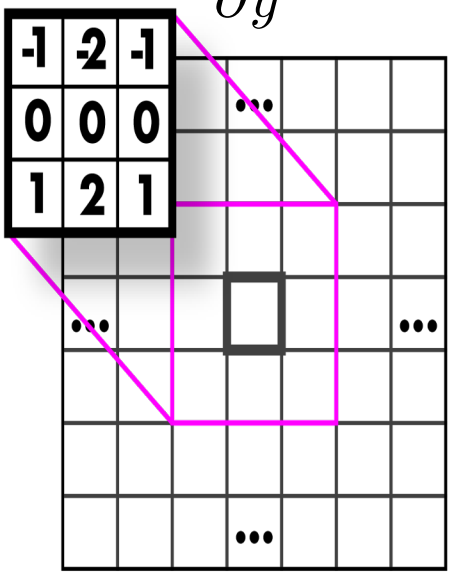
- Smoothing
  - Positive values
  - Sum to 1 → constant regions same as input
  - Amount of smoothing proportional to mask size
  - Remove “high-frequency” components; “low-pass” filter
- Derivatives
  - Opposite signs used to get high response in regions of high contrast
  - Sum to 0 → no response in constant regions
  - High absolute value at points of high contrast

Slide credit: Kristen Grauman

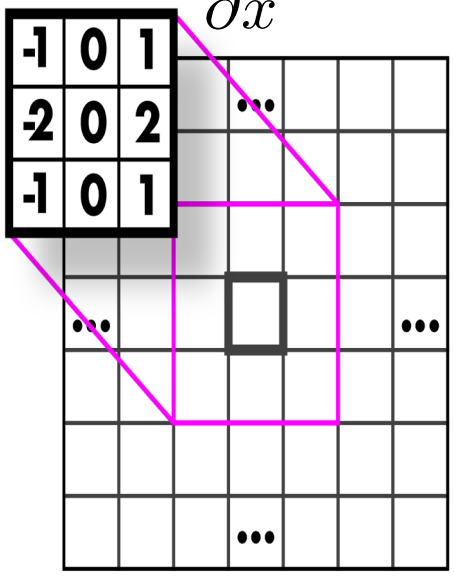
Another approach: gradient magnitude

- Don't need 2nd derivatives
- Just use magnitude of gradient
- Are we done? No!

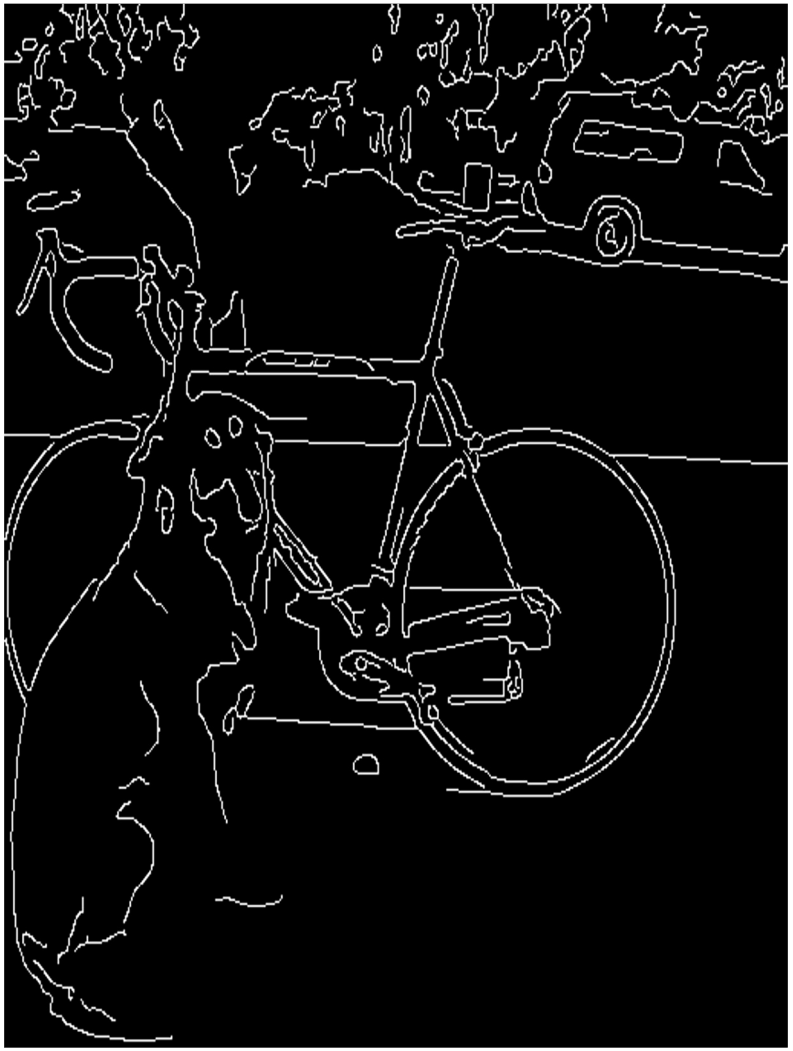
Horizontal Sobel filter for computing the partial derivative $\frac{\partial f(x,y)}{\partial y}$



Vertical Sobel filter for computing the partial derivative $\frac{\partial f(x,y)}{\partial x}$



What we really want: line drawing



Canny Edge Detection

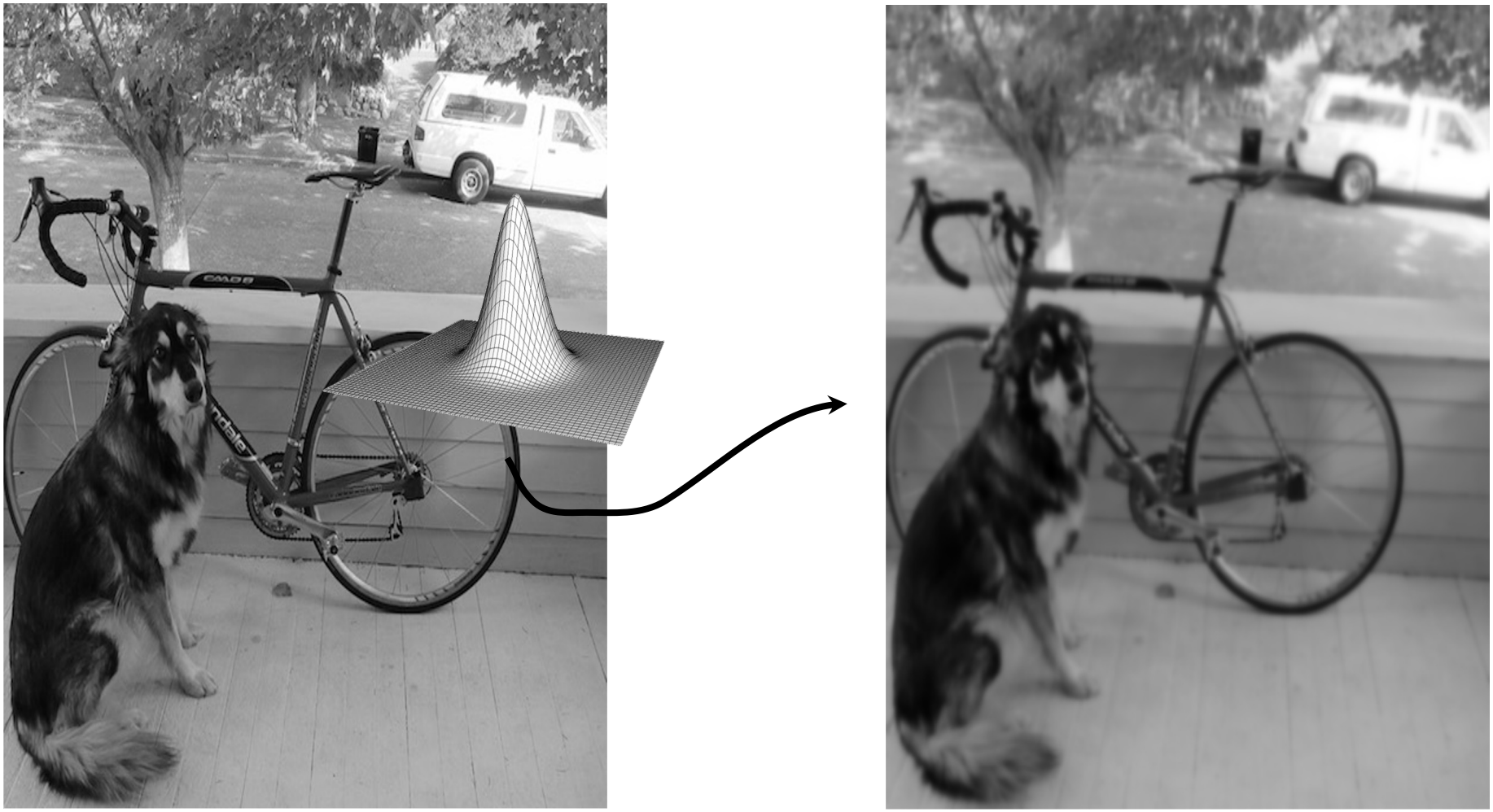
Algorithm:

- Smooth image (only want “real” edges, not noise)
- Calculate gradient direction and magnitude
- Non-maximum suppression perpendicular to edge
- Threshold into strong, weak, no edge
- Connect together components

http://bigwww.epfl.ch/demo/ip/demos/edgeDetector/

Smooth image

- Apply a Gaussian filter

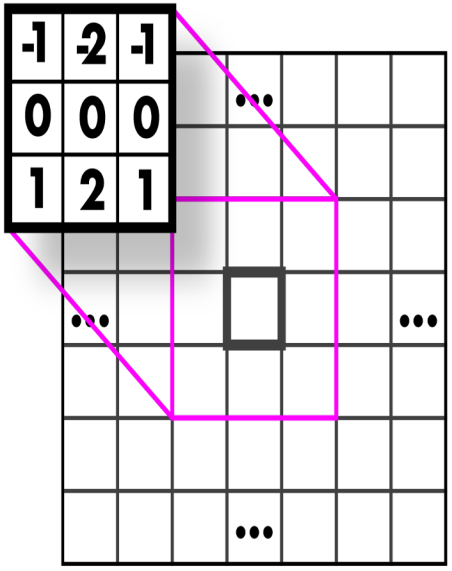


Gradient magnitude and direction

- Sobel filter

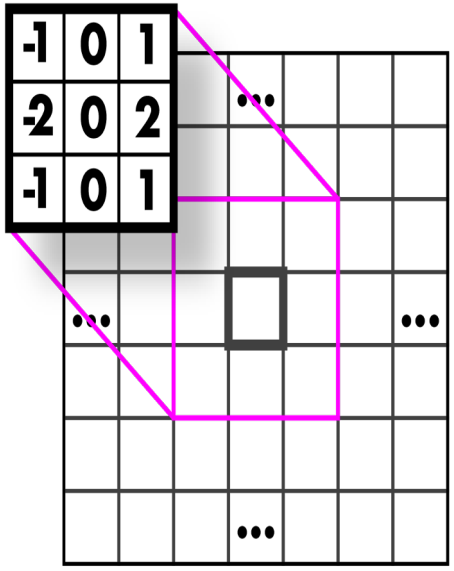
Horizontal Sobel filter for computing the partial derivative

$$\frac{\partial f(x,y)}{\partial y}$$



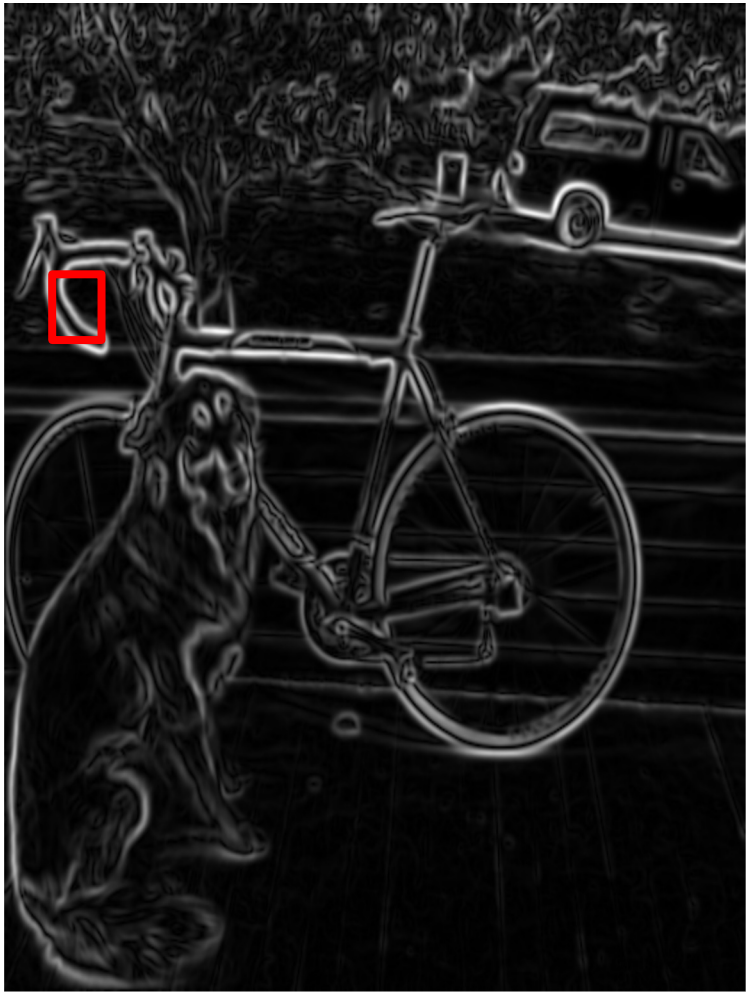
Vertical Sobel filter for computing the partial derivative

$$\frac{\partial f(x,y)}{\partial x}$$

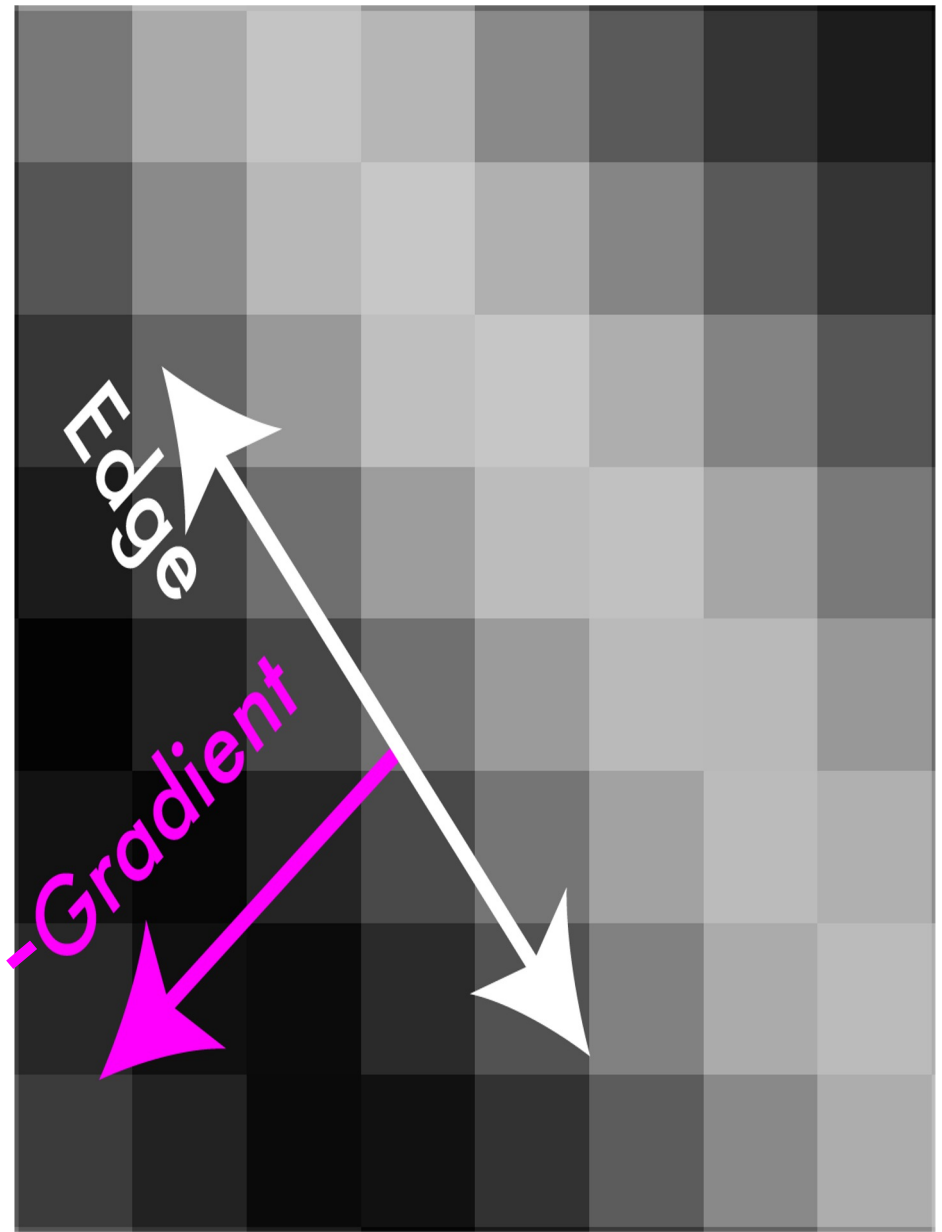


Non-maximum suppression

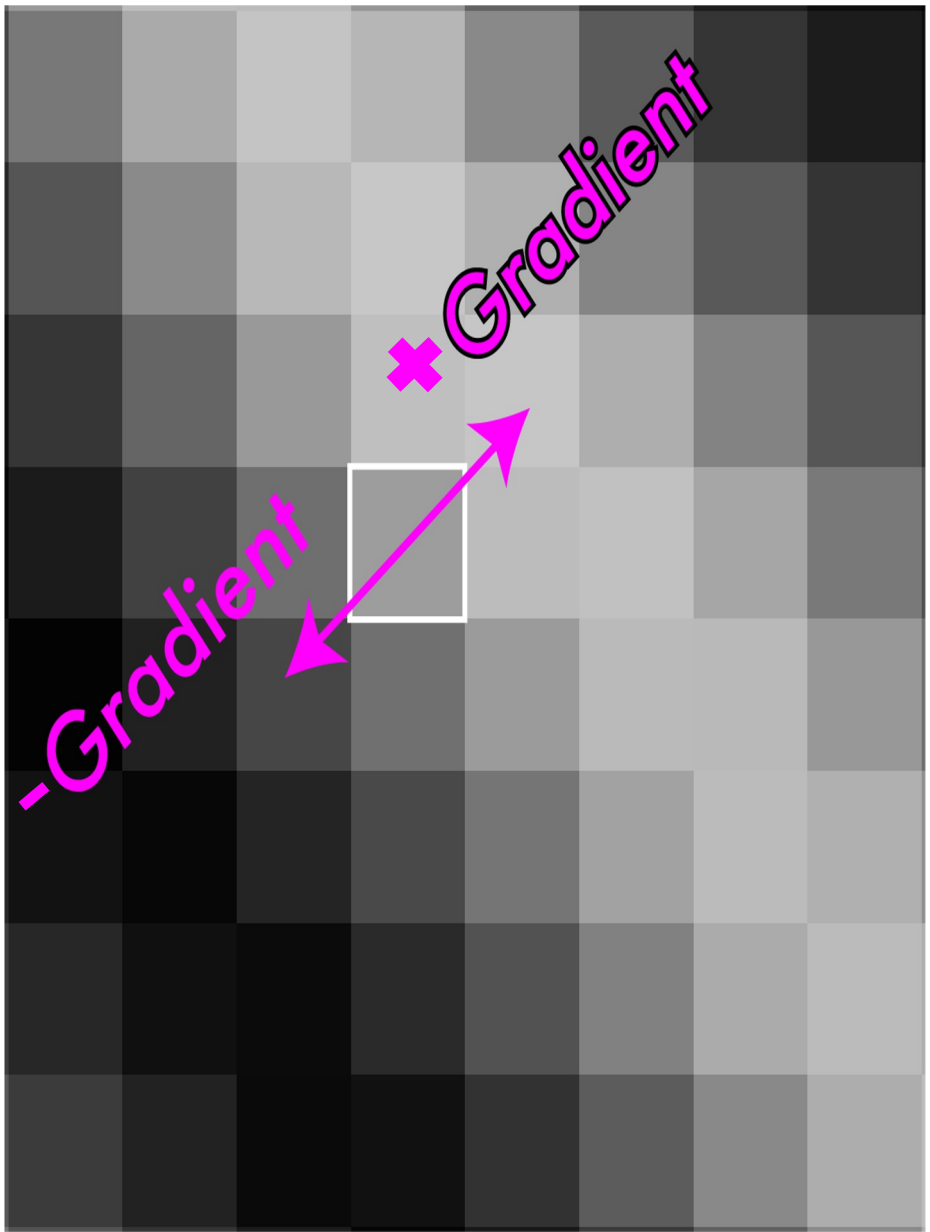
- Want single pixel edges, not thick blurry lines
- Need to check nearby pixels
- See if response is highest



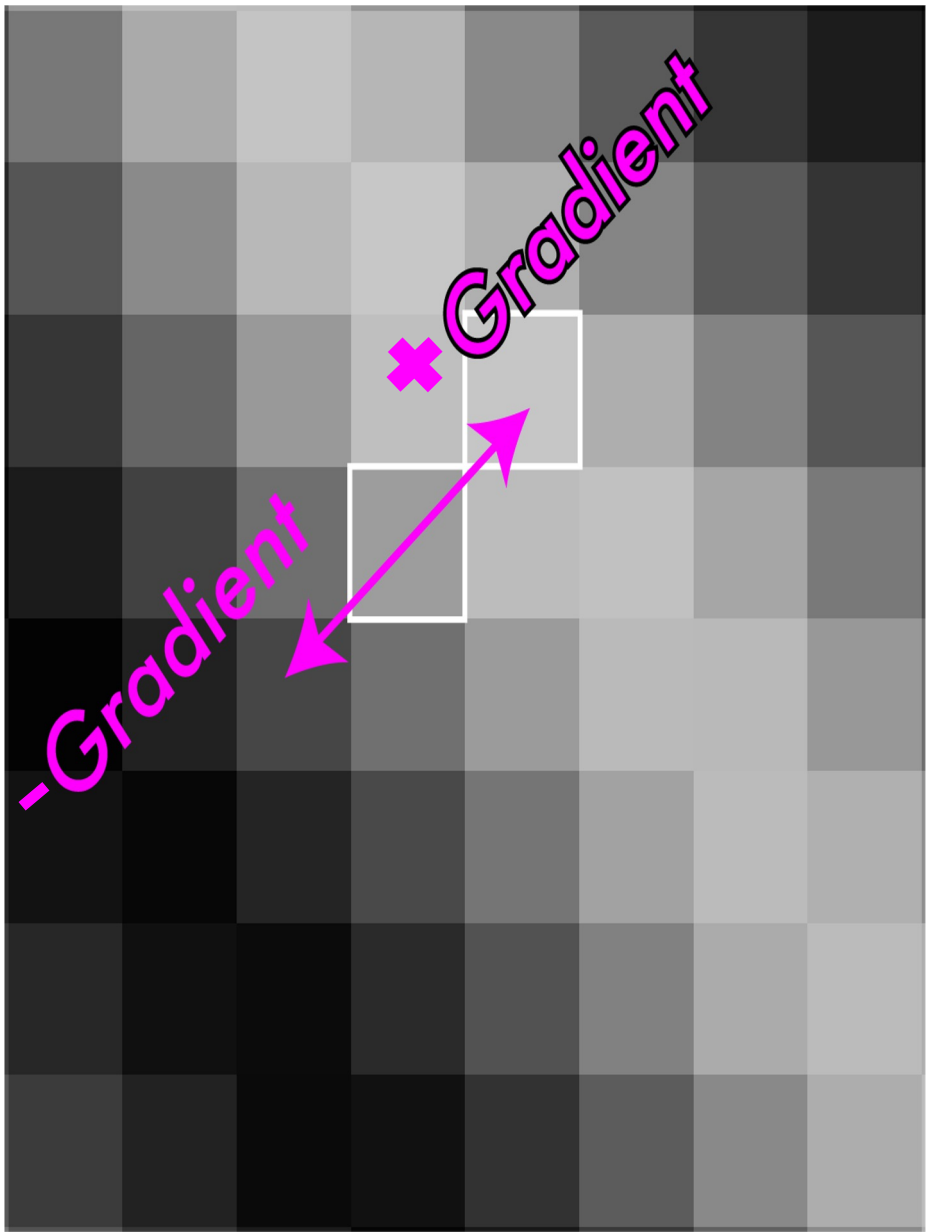
Non-maximum suppression



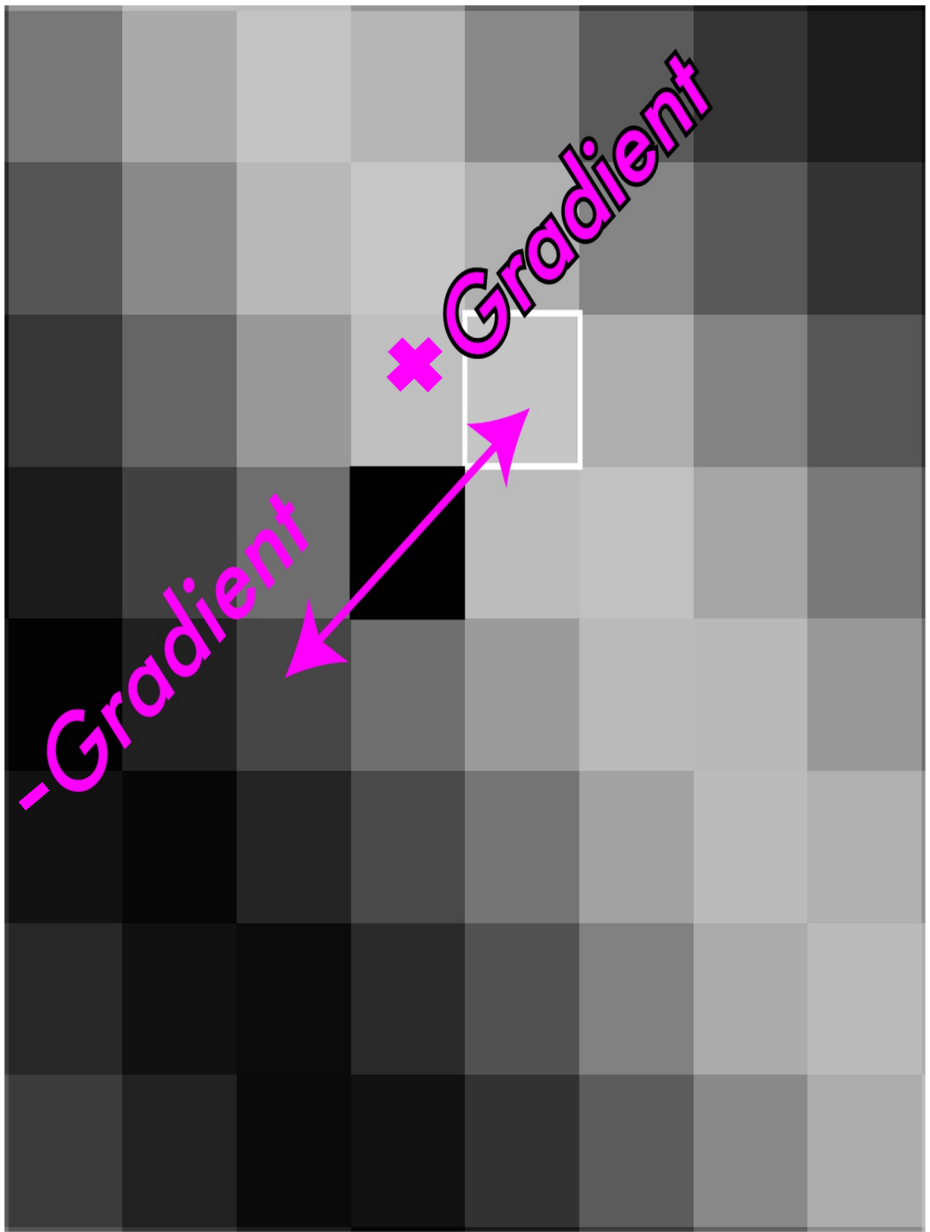
Non-maximum suppression



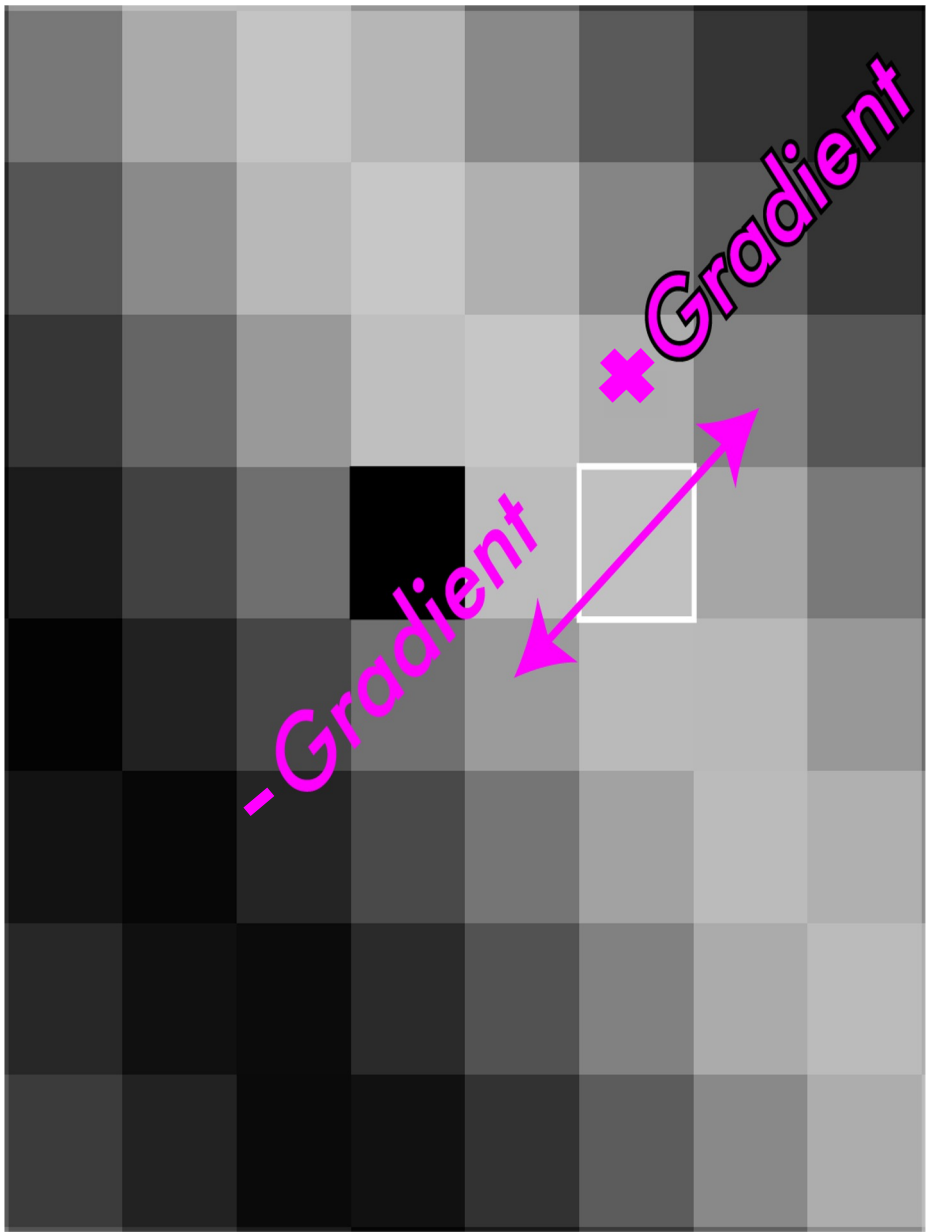
Non-maximum suppression



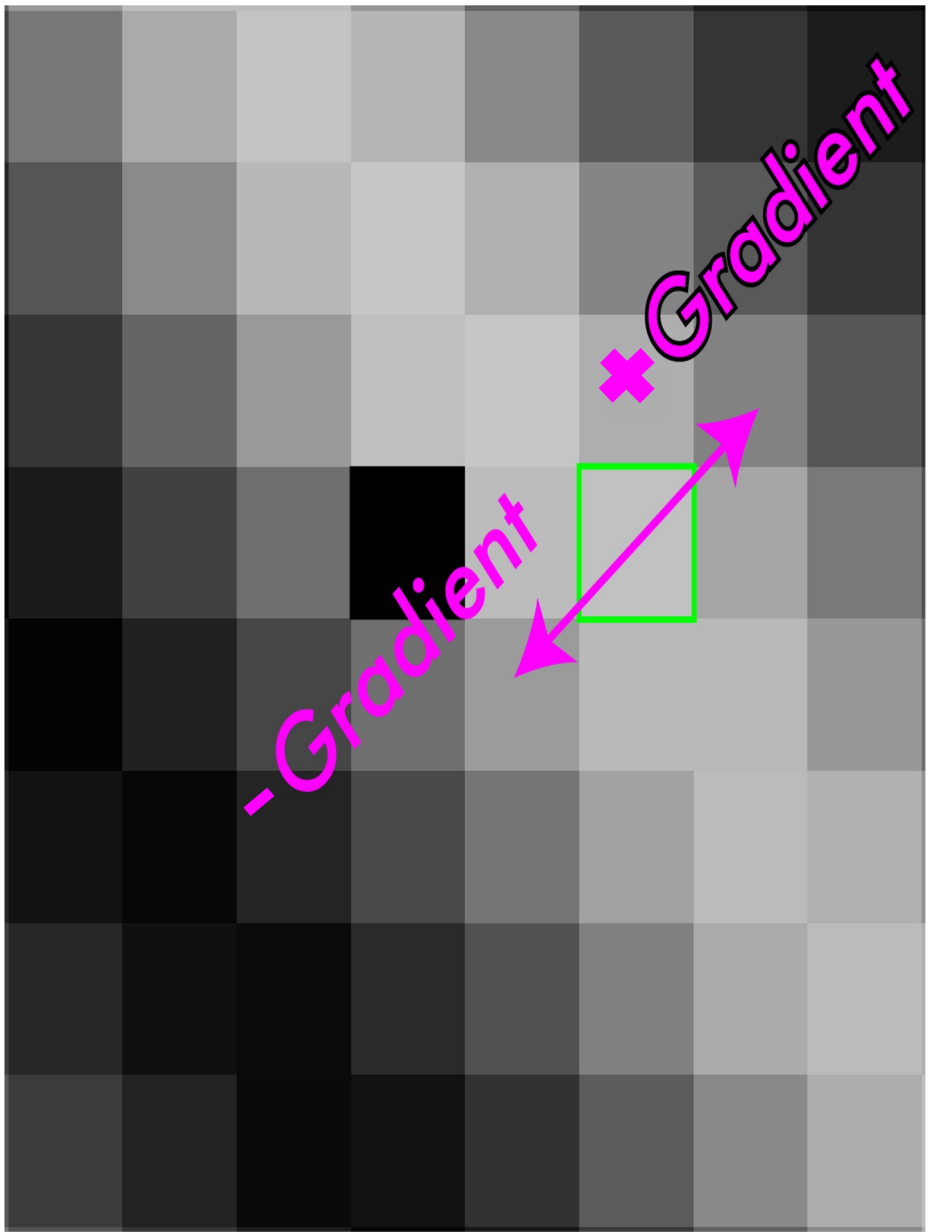
Non-maximum suppression



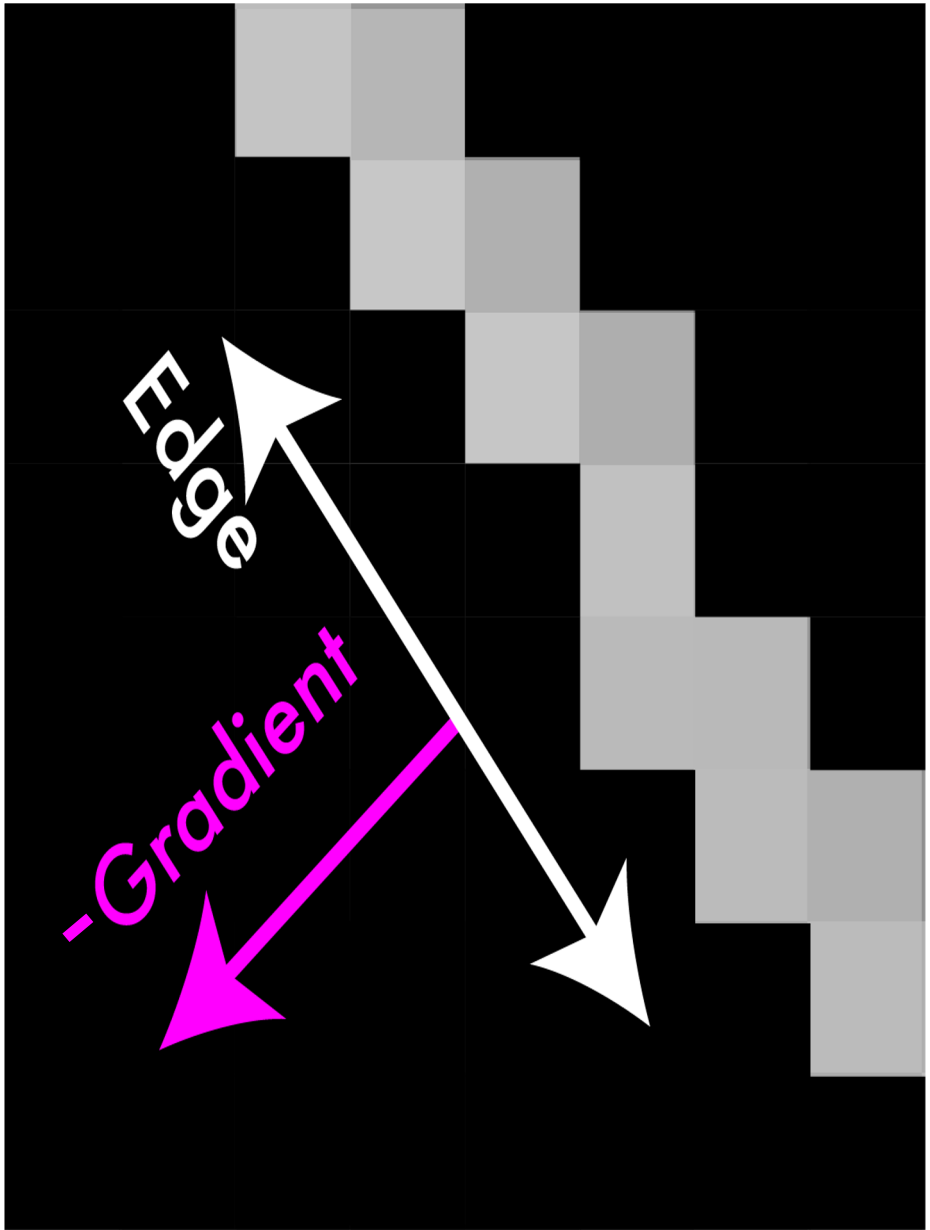
Non-maximum suppression



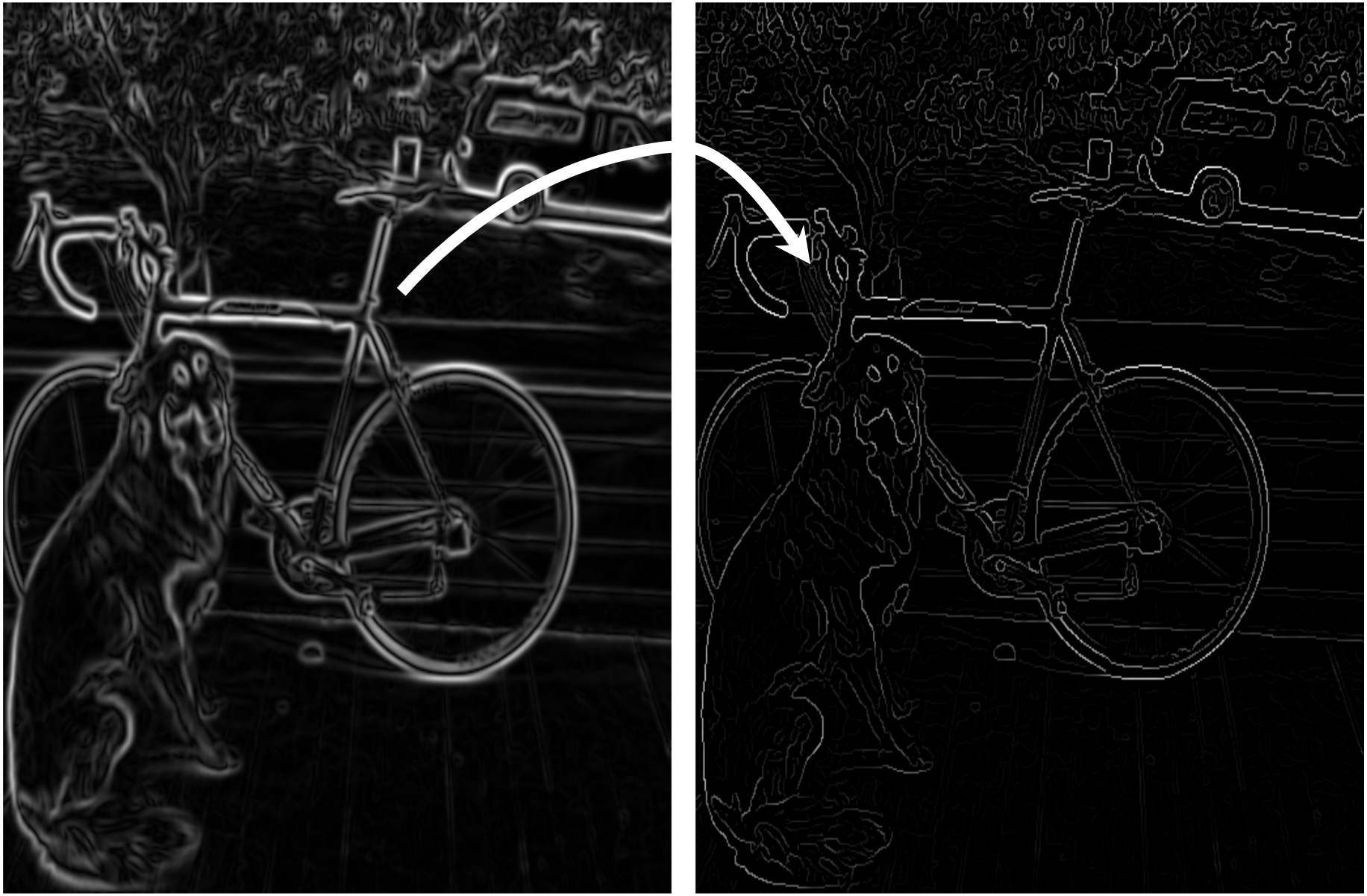
Non-maximum suppression



Non-maximum suppression

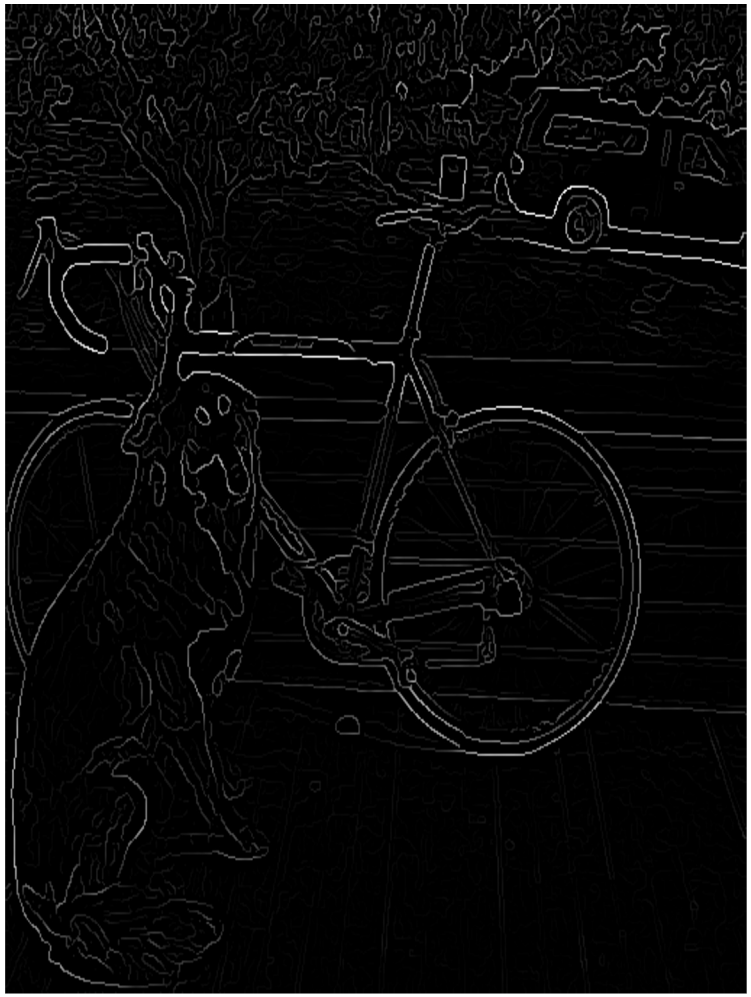


Non-maximum suppression



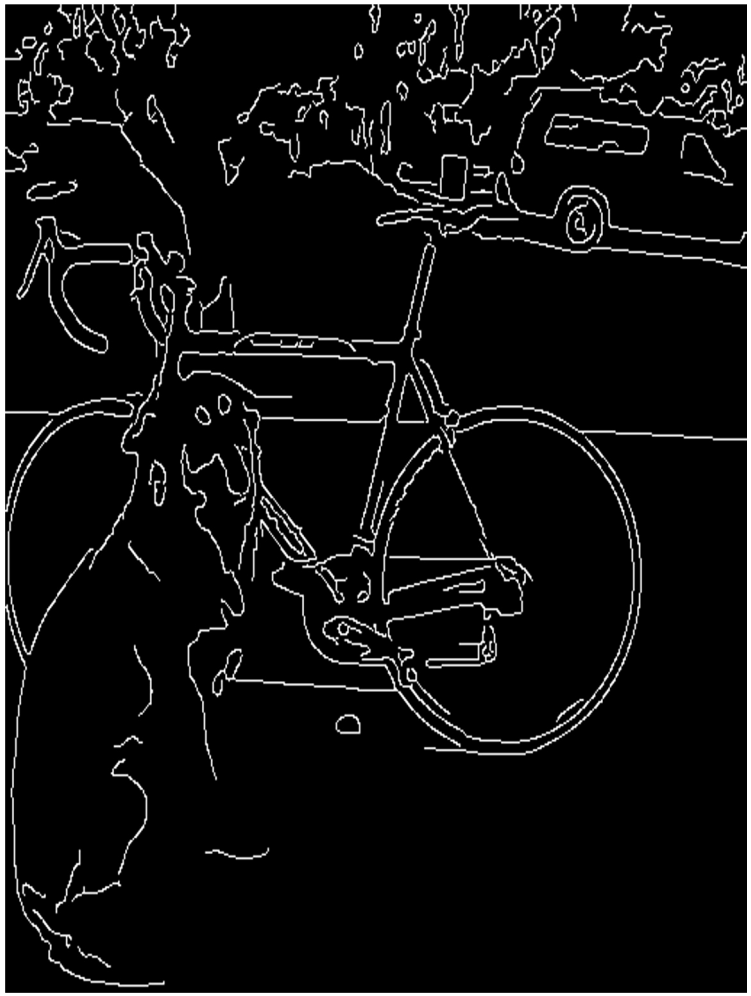
Threshold edges

- 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?



Connect them up!

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

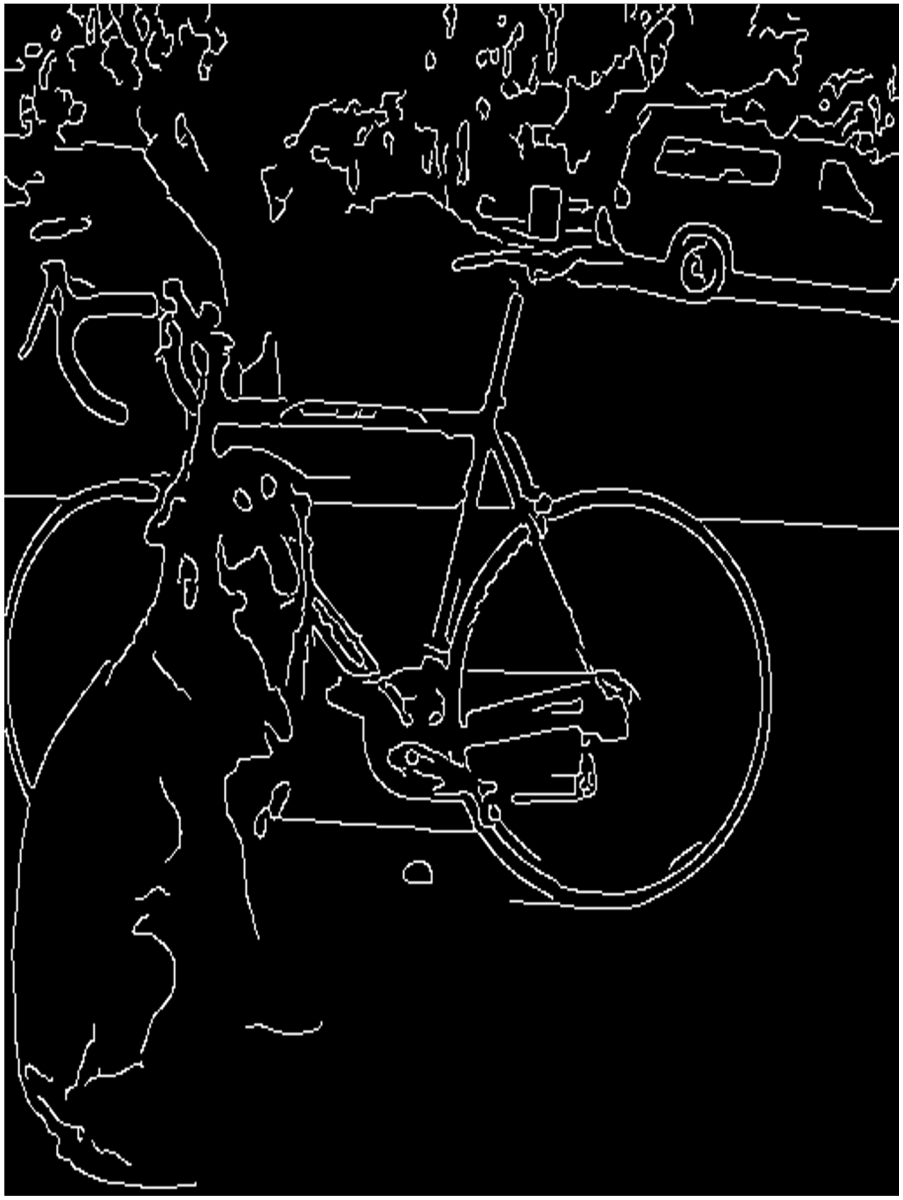


Canny Edge Detection

Algorithm:

- Smooth image (only want “real” edges, not noise)
- Calculate gradient direction and magnitude
- Non-maximum suppression perpendicular to edge
- Threshold into strong, weak, no edge
- Connect together components
- Tunable: Sigma, thresholds

Canny Edge Detection



Canny Edge Detection - demo

http://bigwww.epfl.ch/demo/ip/demos/edgeDetector/

Course structure

1. **Features and filters: low-level vision**

   Linear filters, color, texture, edge detection

2. **Grouping and fitting: mid-level vision**

   Fitting curves and lines, robust fitting, RANSAC, Hough transform, segmentation

3. **Multiple views**

   Local invariant feature and description, epipolar geometry and stereo, object instance recognition

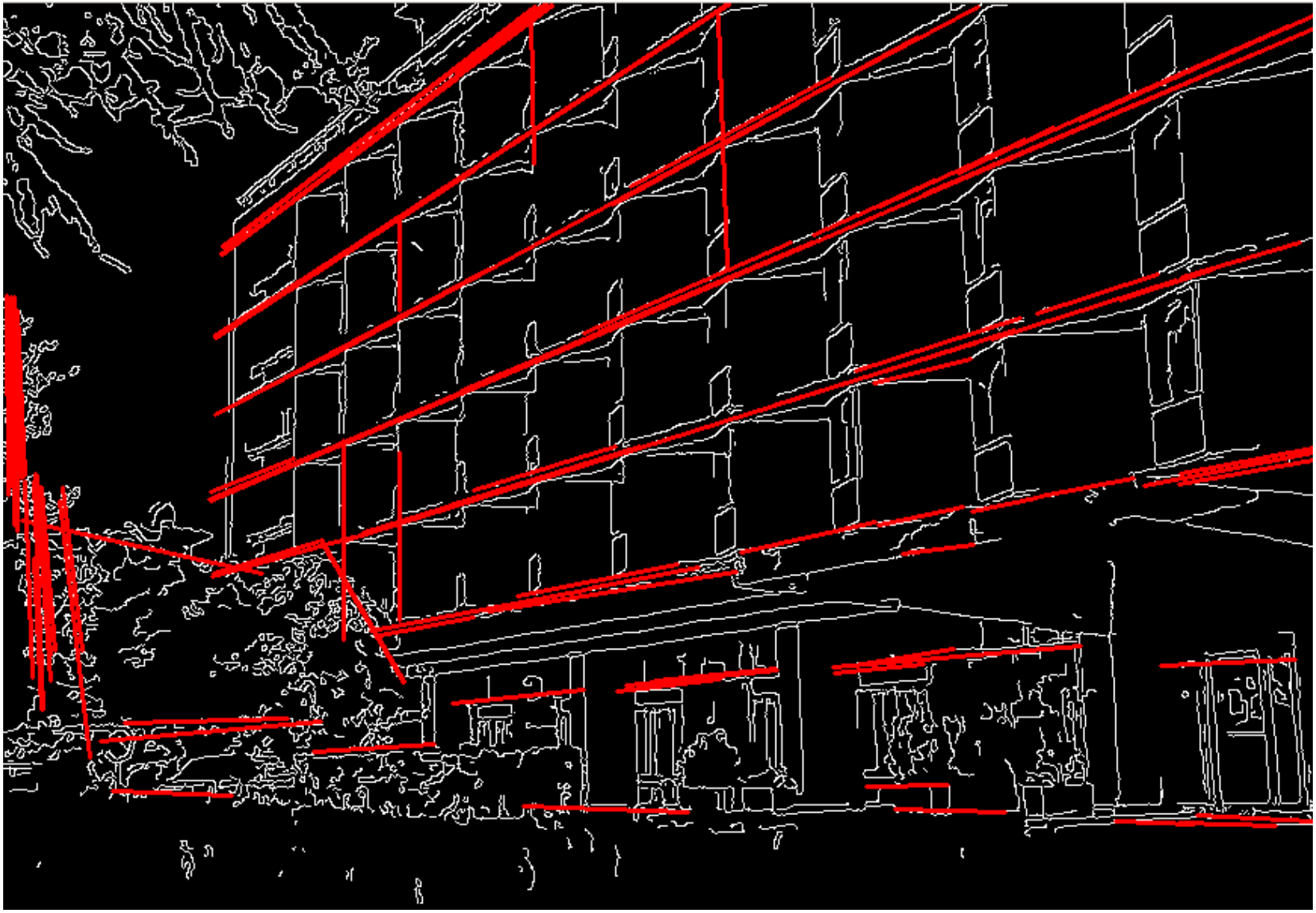
4. **Object Recognition: high – level vision**

   Object classification, object detection, part based models, bovw models

5. **Video understanding**

   Object tracking, background subtraction, motion descriptors, optical flow

Fitting

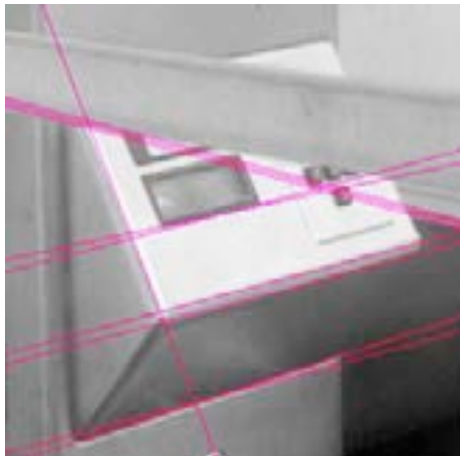


Fitting

- We have learned how to detect edges
- We would like to form a higher-level, more compact representation of the features in the image by grouping multiple features according to a simple model

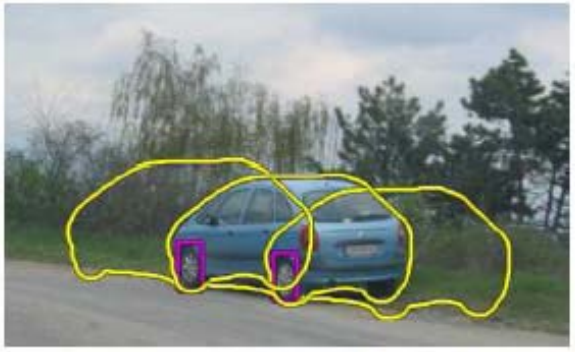
Fitting

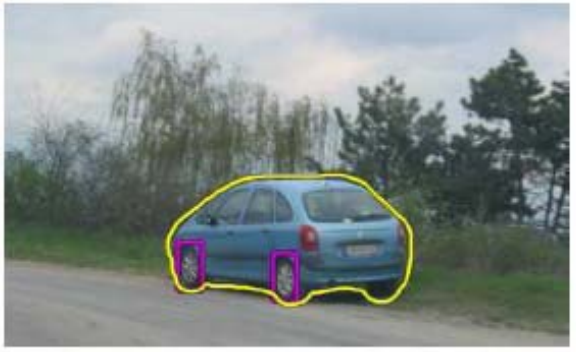
- Choose a *parametric model* to represent a set of features



simple model: lines

simple model: circles

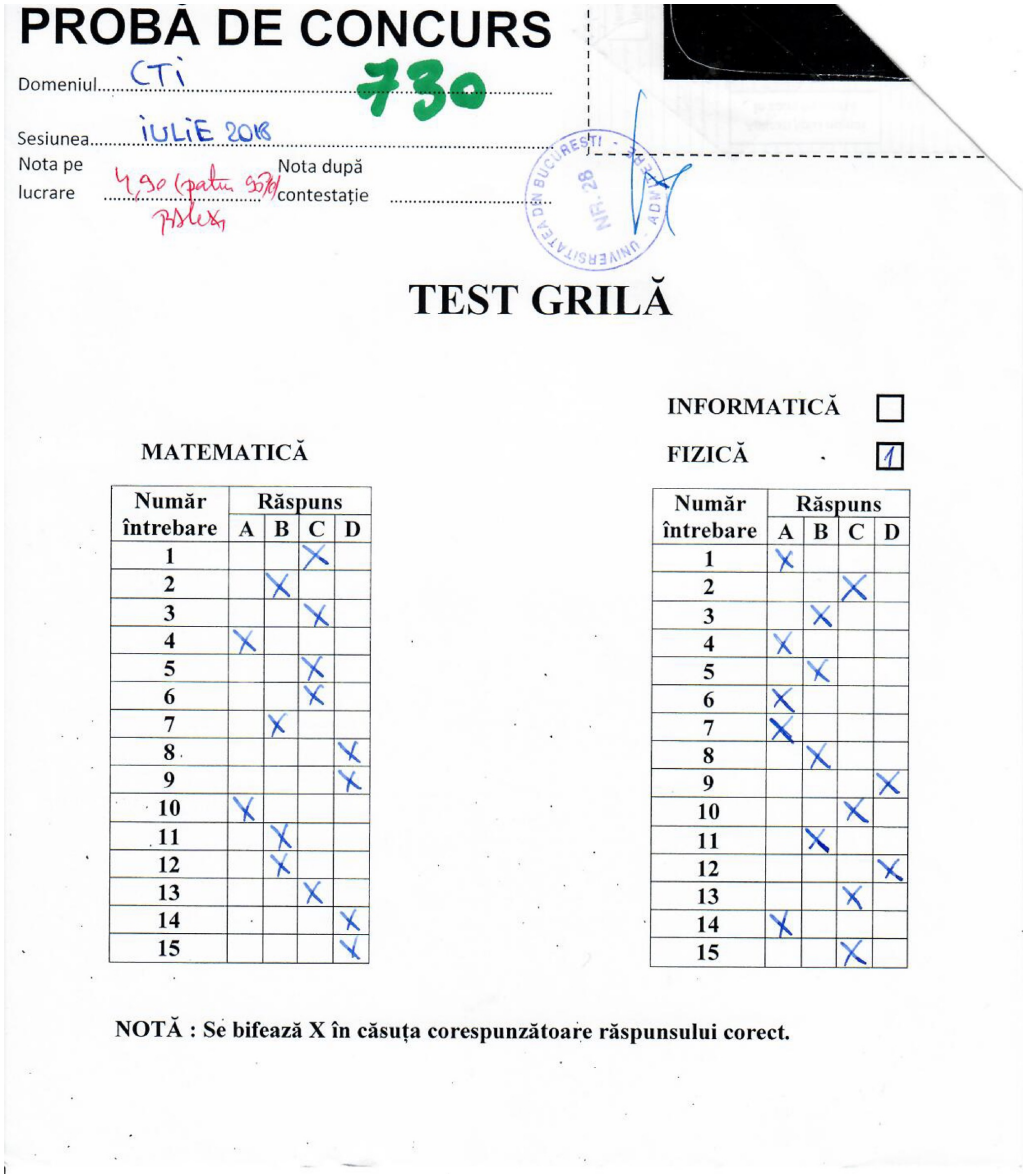




complex model: car

Slide credit : K. Grauman

Lab 2: Line detection - application



PROBĂ DE CONCURS

Domeniul CTI 730

Sesiunea IULIE 2018

Nota pe lucrare 4,90 (patru 90/) Nota după contestație

TEST GRILĂ

INFORMATICĂ ☐

FIZICĂ ☑

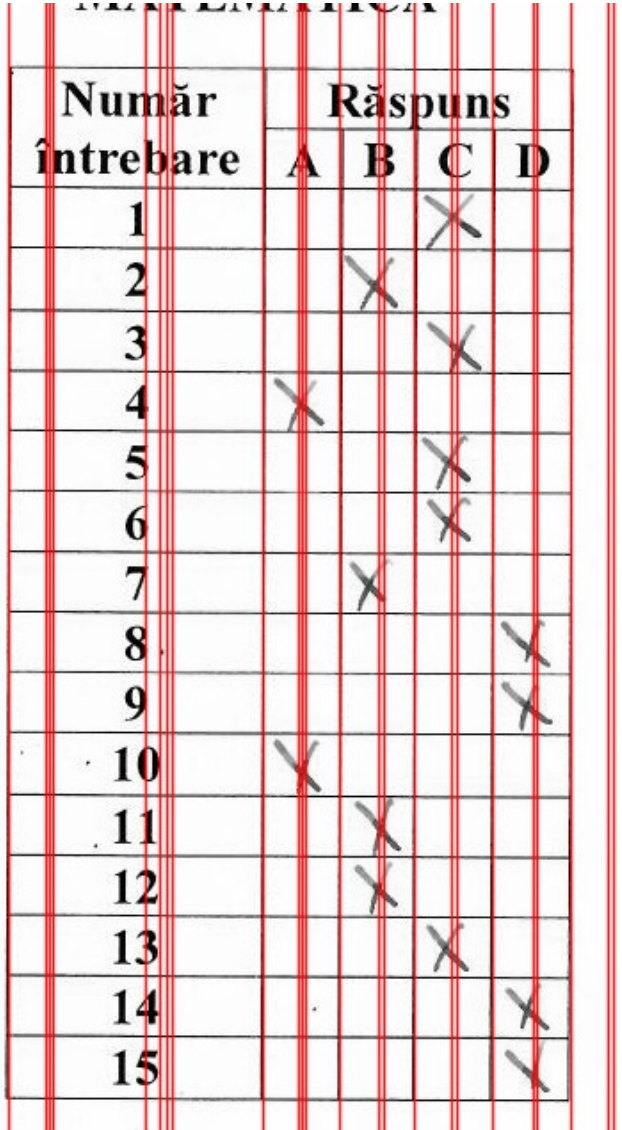
MATEMATICĂ

Număr întrebare	Răspuns A	B	C	D
1			X	
2		X		
3			X	
4	X			
5			X	
6			X	
7		X		
8				X
9				X
10	X			
11		X		
12		X		
13			X	
14				X
15				X

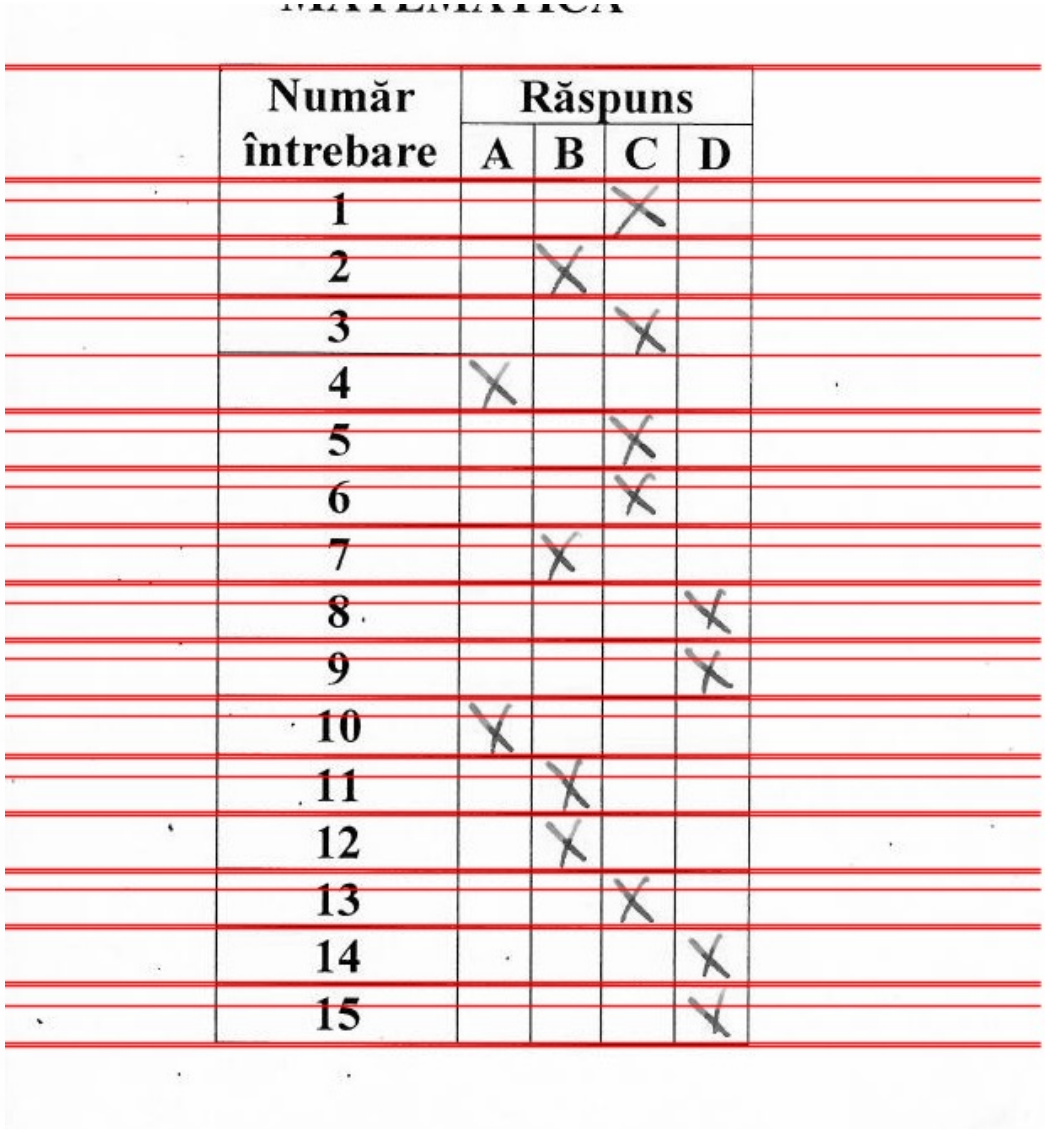
Număr întrebare	Răspuns A	B	C	D
1	X			
2			X	
3		X		
4	X			
5		X		
6	X			
7	X			
8		X		
9				X
10			X	
11		X		
12				X
13			X	
14	X			
15			X	

NOTĂ : Se bifează X în căsuța corespunzătoare răspunsului corect.

Lab 2: Line detection - application

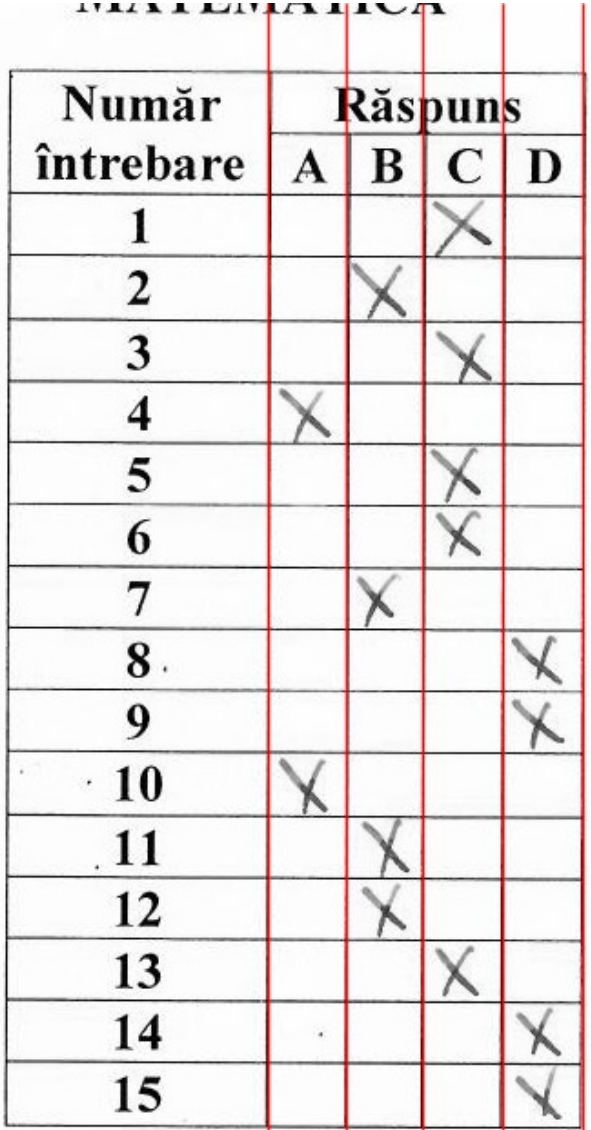


Număr întrebare	Răspuns A	B	C	D
1			X	
2		X		
3			X	
4	X			
5			X	
6			X	
7		X		
8				X
9				X
10	X			
11		X		
12		X		
13			X	
14				X
15				X

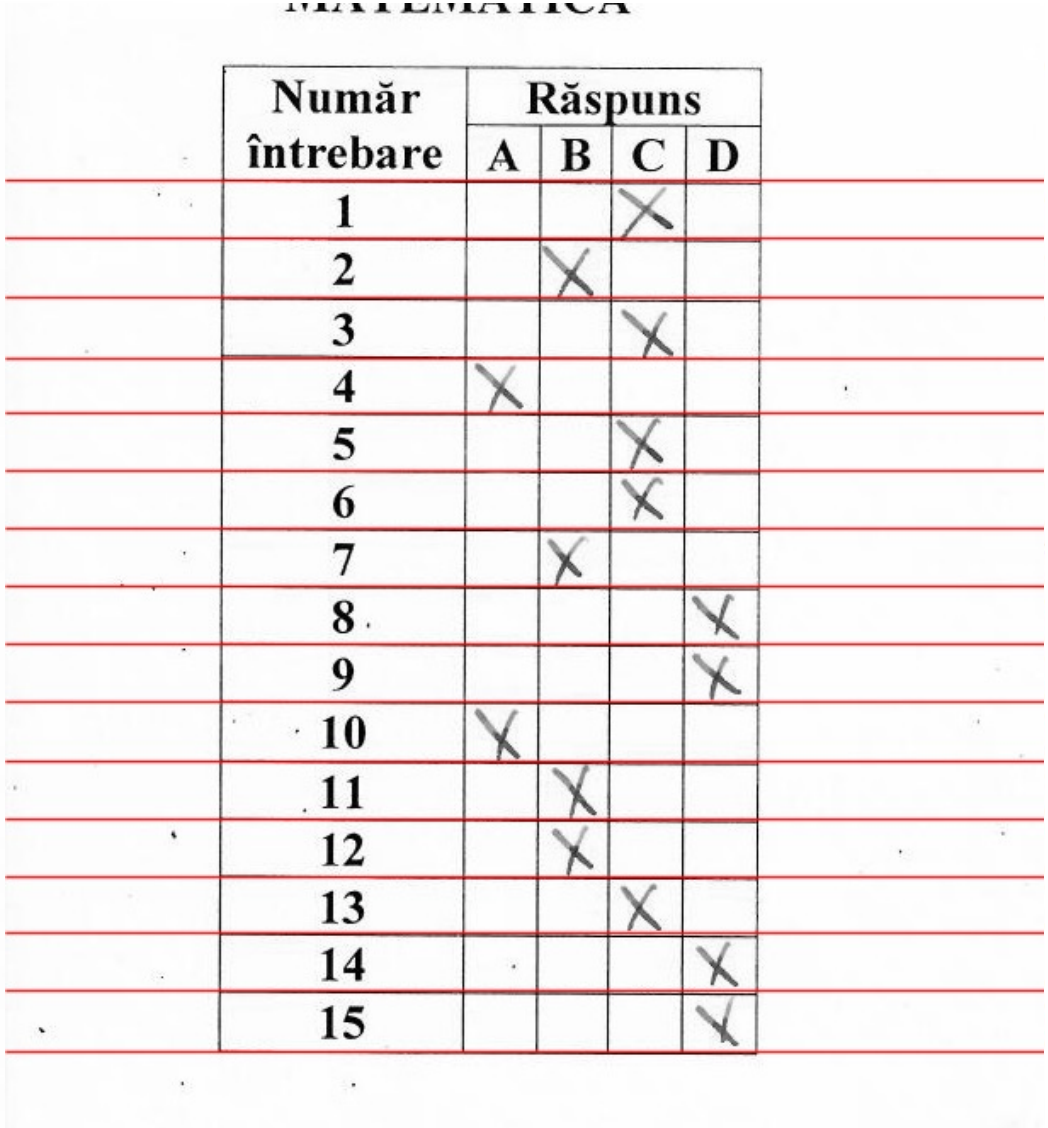


Număr întrebare	Răspuns A	B	C	D
1			X	
2		X		
3			X	
4	X			
5			X	
6			X	
7		X		
8				X
9				X
10	X			
11		X		
12		X		
13			X	
14				X
15				X

Lab 2: Line detection - application



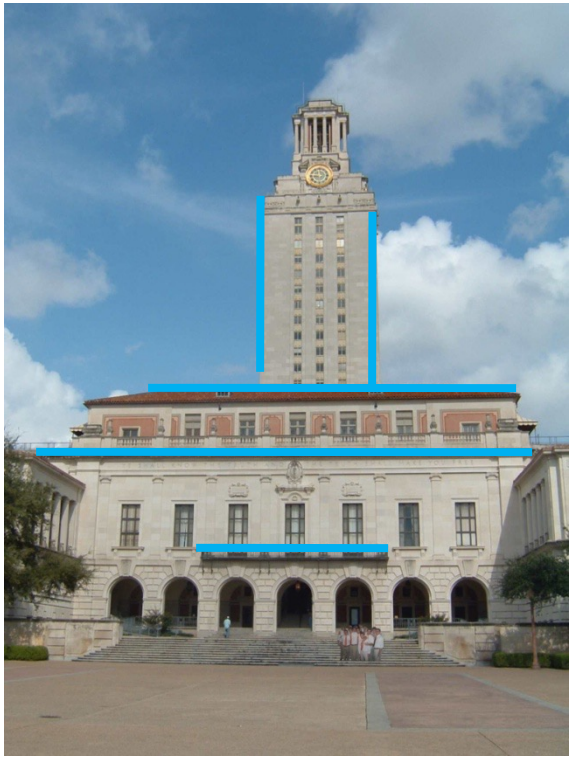
Număr întrebare	Răspuns			
	A	B	C	D
1			X	
2		X		
3			X	
4	X			
5			X	
6			X	
7		X		
8				X
9				X
10	X			
11		X		
12		X		
13			X	
14				X
15				X

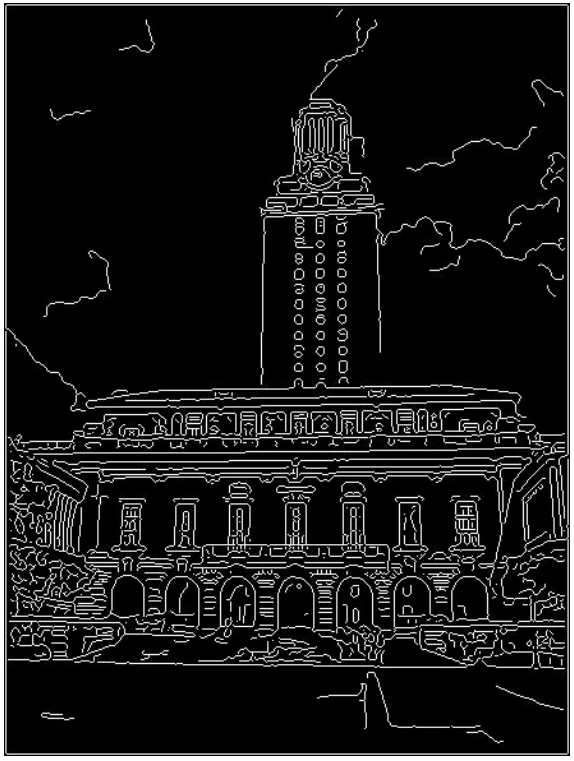


Număr întrebare	Răspuns			
	A	B	C	D
1			X	
2		X		
3			X	
4	X			
5			X	
6			X	
7		X		
8				X
9				X
10	X			
11		X		
12		X		
13			X	
14				X
15				X

Fitting: challenges

Case study: Line detection





- **Noise** in the measured feature locations
- **Extraneous data:** clutter (outliers), multiple lines
- **Missing data:** occlusions

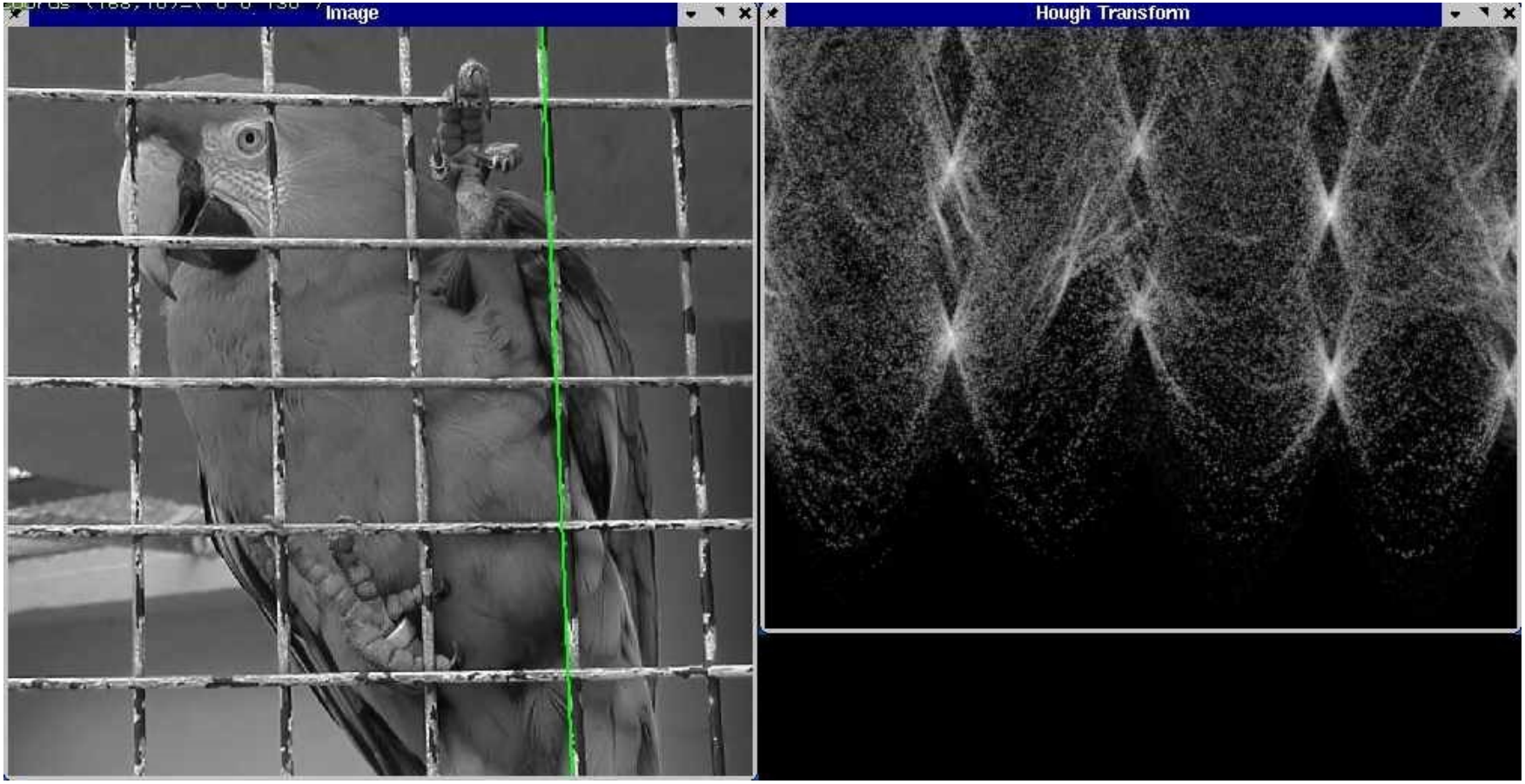
Slide credit : S. Lazebnik

Fitting: challenges

- If we know which points belong to the line, how do we find the "optimal" line parameters?
  - Least squares
- What if there are outliers?
  - Robust fitting, RANSAC
- What if there are many lines?
  - Voting methods: RANSAC, Hough transform

Slide credit : S. Lazebnik

Fitting: The Hough transform



Slide credit : S. Lazebnik

Voting schemes

- Let each feature vote for all the models that are compatible with it
  - Hopefully the noise features will not vote consistently for any single model
  - Missing data doesn't matter as long as there are enough features remaining to agree on a good model

Slide credit : S. Lazebnik

Hough transform

- An early type of voting scheme
  - Discretize *parameter space* into bins
  - For each feature point in the image, put a vote in every bin in the parameter space that could have generated this point
  - Find bins that have the most votes

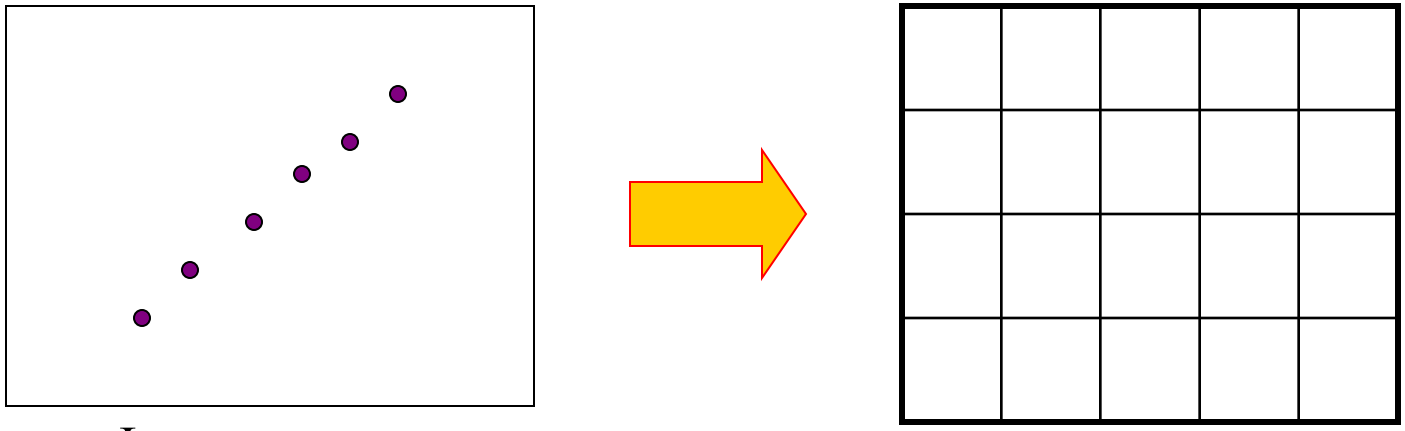


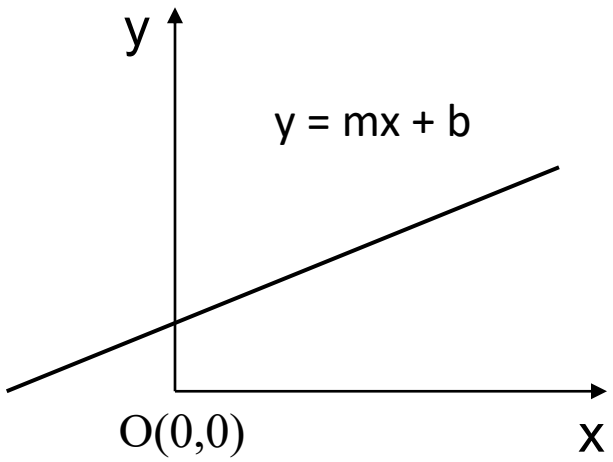
Image space

Hough parameter space

P.V.C. Hough, *Machine Analysis of Bubble Chamber Pictures*,
Proc. Int. Conf. High Energy Accelerators and Instrumentation, 1959

Slide credit : S. Lazebnik

Parameterization of a line



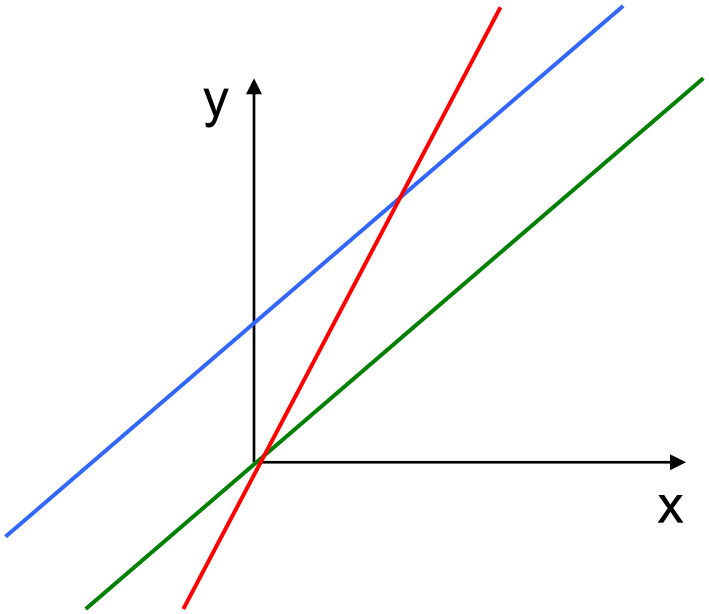
Parameterization of a line in image plane: $y = m * x + b$,

m – slope (defines the angle between the line and Ox axis),

b – intercept (bias to origin O(0,0))

(m,b) are line parameters

Parameterization of a line - examples



d_1: y = x, so m=1, b = 0

d_2: y = x + 2, so m=1, b = 2

d_3: y = 2x, so m=2, b = 0

Parameterization of a line in image plane: $y = m * x + b$,

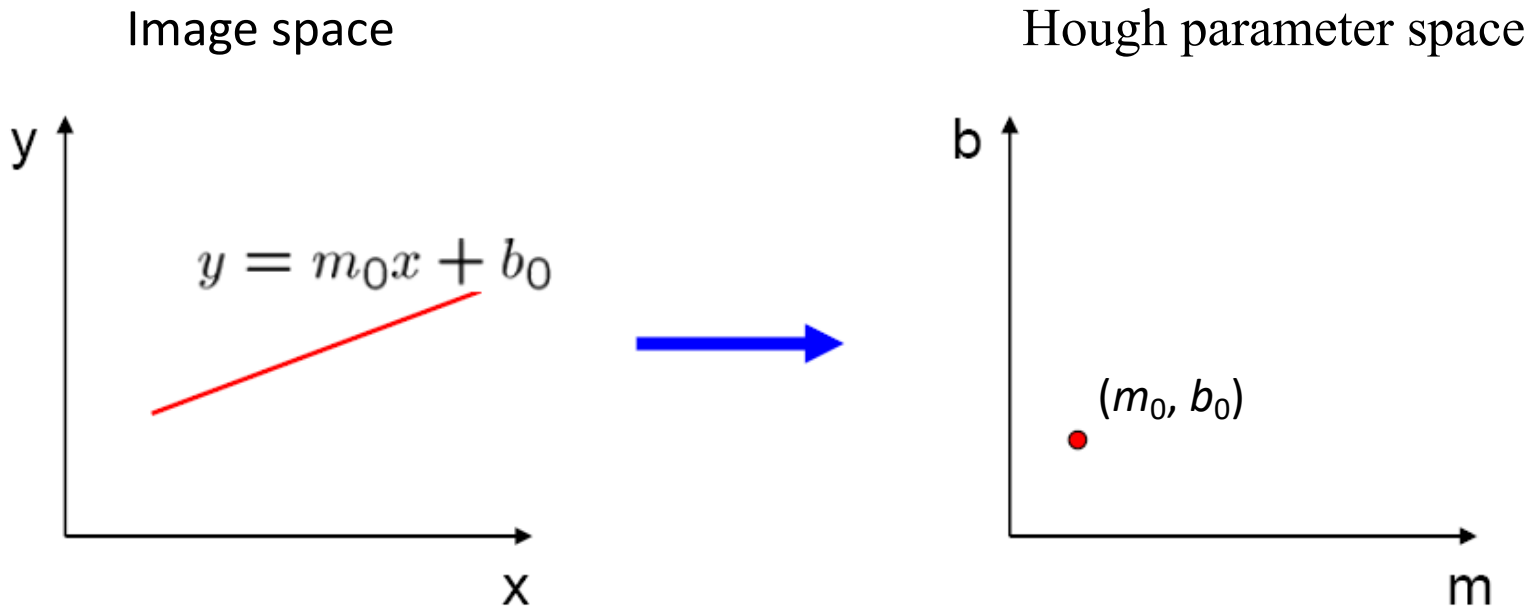
m – slope (defines the angle between the line and Ox axis),

b – intercept (bias to origin O(0,0))

(m,b) are line parameters

Parameter space representation

- A **line** in the image corresponds to a **point** in Hough space



Slide credit : S. Seitz

Hough parameter space

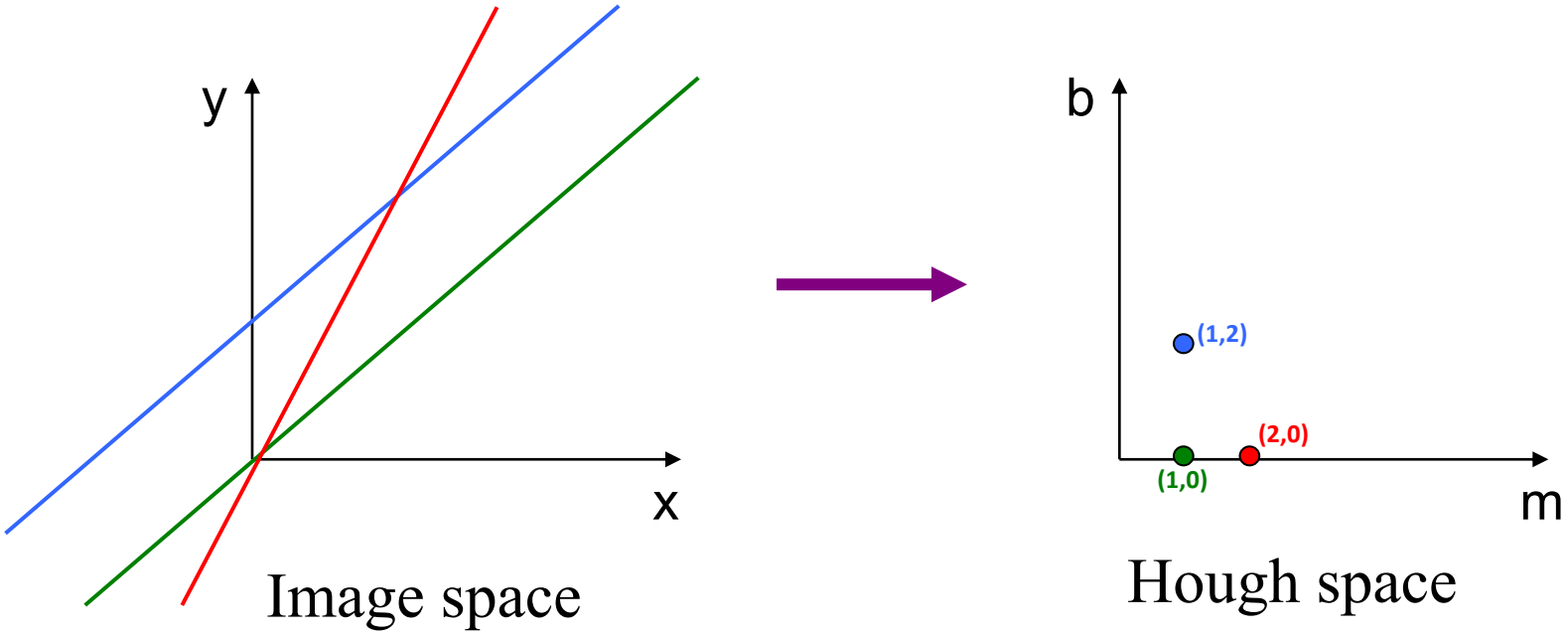


Image space

Hough space

d_1: y = x, so m=1, b = 0

d_1: m=1, b = 0 -> (1,0)

d_2: y = x + 2, so m=1, b = 2

d_2: m=1, b = 2 -> (1,2)

d_3: y = 2x, so m=2, b = 0

d_3: m=2, b = 0 -> (2,0)

A line in the image space corresponds to a point in the Hough space.

Correspondence between image and Hough spaces

A line in the image space corresponds to a point in the Hough space.

What does a **point** (x_0, y_0) in the image space map to in the Hough space?

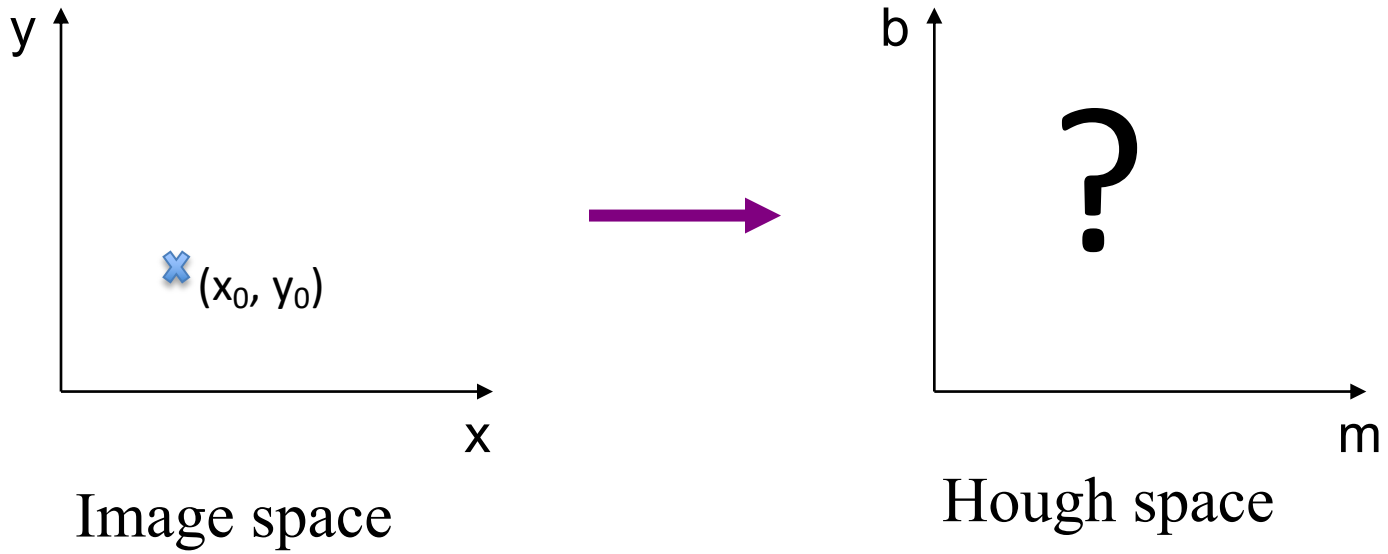


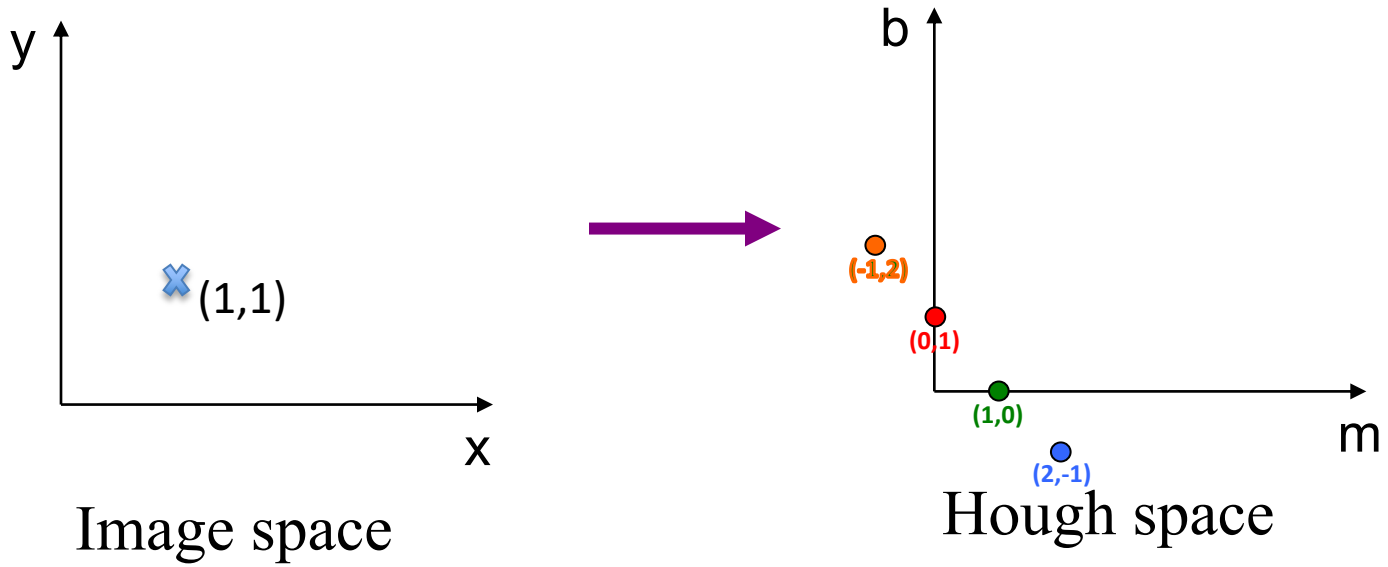
Image space

Hough space

Correspondence between image and Hough spaces

A line in the image space corresponds to a point in the Hough space.

What does a **point** (x_0, y_0) in the image space map to in the Hough space?



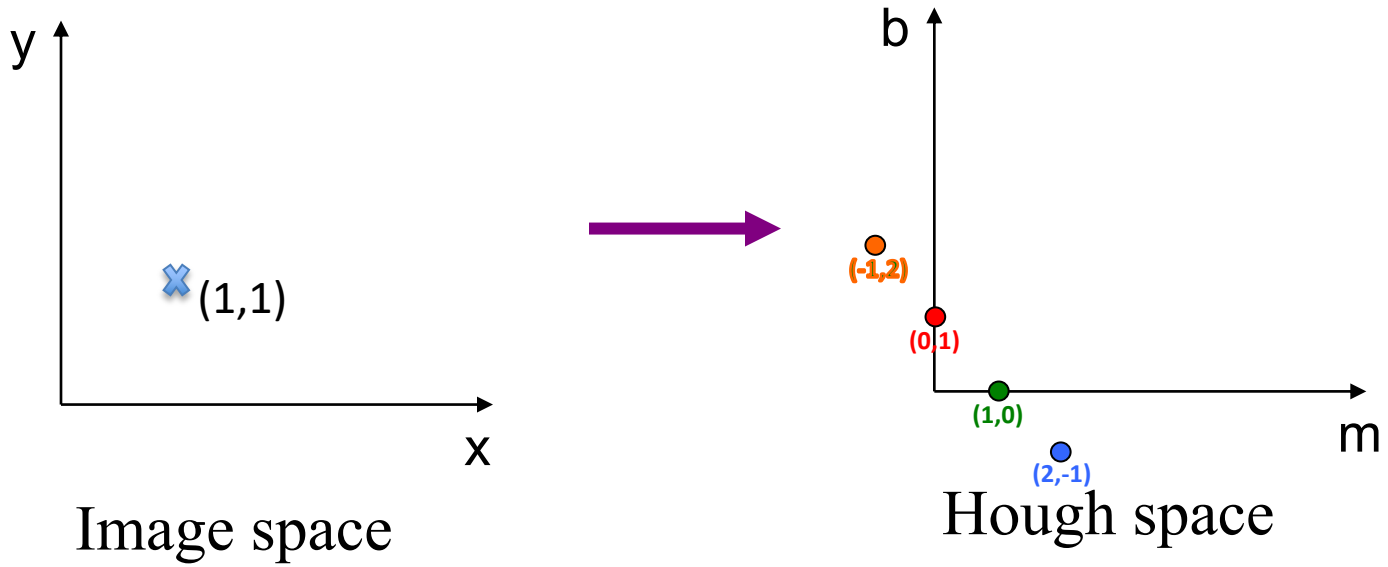
There is an infinity of lines of the form y = mx+b that contain point (1,1):

y = x (m=1, b=0); y = 2x-1 (m=2,b=-1), y = 1 (m=0, b = 1), y = -x + 2 (m=-1, b =2),

Correspondence between image and Hough spaces

A line in the image space corresponds to a point in the Hough space.

A **point** (x_0, y_0) in the image space maps to a line in the Hough space



There is an infinity of lines of the form y = mx+b that contain point (1,1):

y = x (m=1, b=0); y = 2x-1 (m=2,b=-1), y = 1 (m=0, b = 1), y = -x + 2 (m=-1, b =2),

All 4 points (1,0), (2,-1), (0,-1), (-1,2) are on the line b = -m+1

Correspondence between image and Hough spaces

A line in the image space corresponds to a point in the Hough space.

A **point** (x_0, y_0) in the image space maps to a line in the Hough space

Parameterization of a line in image plane: $y = m * x + b$,

All lines that contain point (x_0,y_0) can have any slope m and any intercept b, the only condition that should be satisfied is

$y_0 = m * x_0 + b$.

$y_0 = m * x_0 + b \Leftrightarrow b = -x_0 * m + y_0$ (parameterization of the line in the Hough space with slope $-x_0$ and intercept y_0)

$(x_0,y_0) = (1,1) \Rightarrow b = -m+1$

Parameter space representation

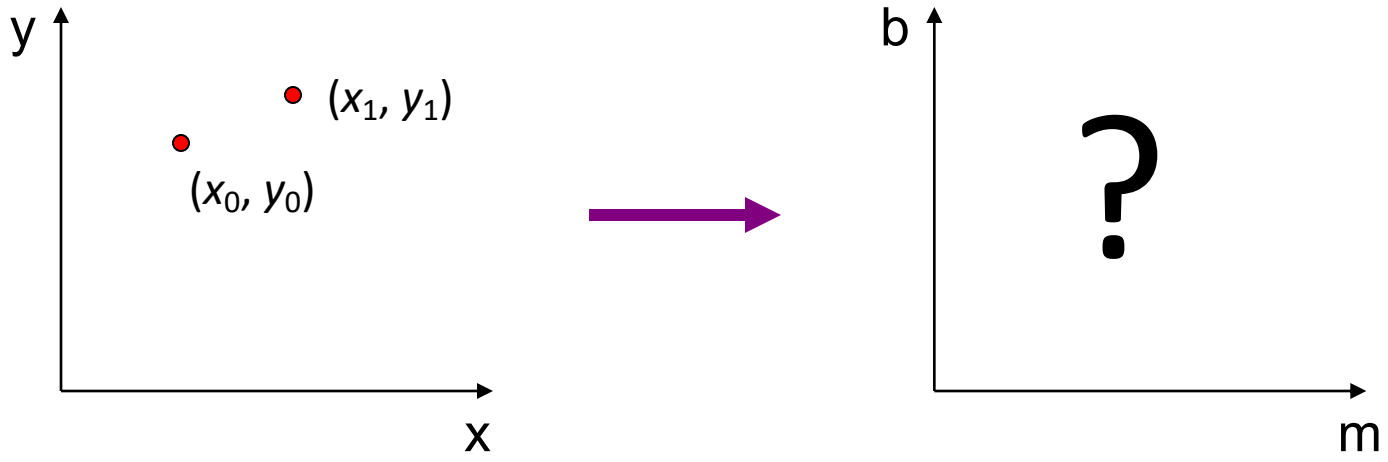


Image space

Hough space

Where is the line that contains both (x_0, y_0) and (x_1, y_1)?

Parameter space representation

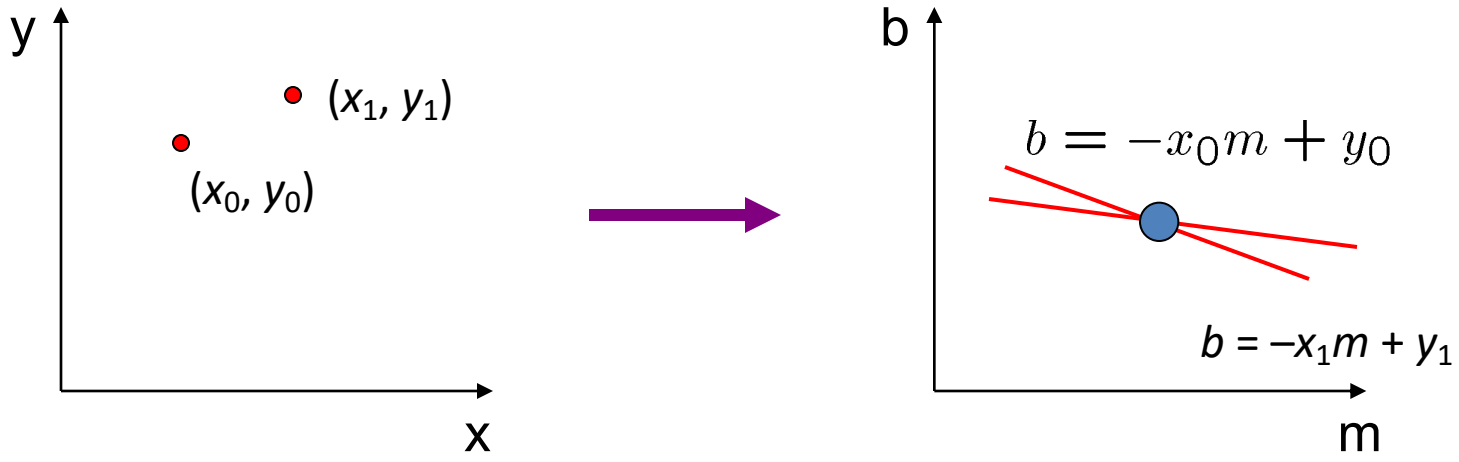


Image space

Hough space

Where is the line that contains both (x_0, y_0) and (x_1, y_1)?

Point (x_0,y_0) maps to the line in the Hough space: $b = -x_0 m + y_0$

Point (x_1,y_1) maps to the line in the Hough space: $b = -x_1 m + y_1$

The line that contains both (x_0, y_0) and (x_1, y_1) maps to a point in the Hough space. This point is the intersection of lines $b = -x_0 m + y_0$ și $b = -x_1 m + y_1$

Finding lines in an image with Hough

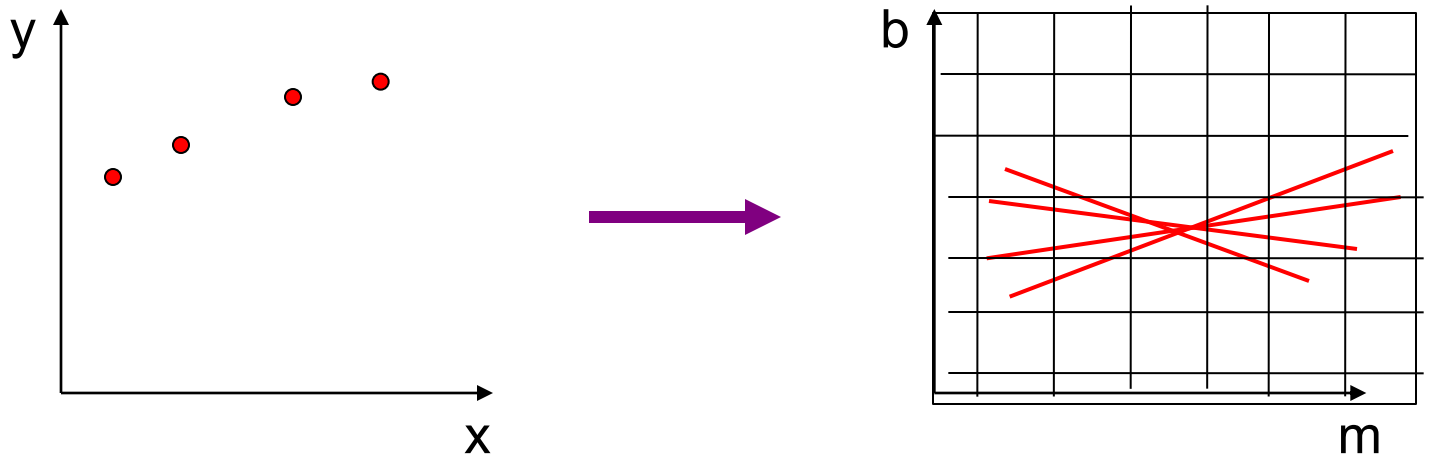


Image space

Hough space

How we can use the previous observations to find parameters (m,b) that define the most probable line in the image space?

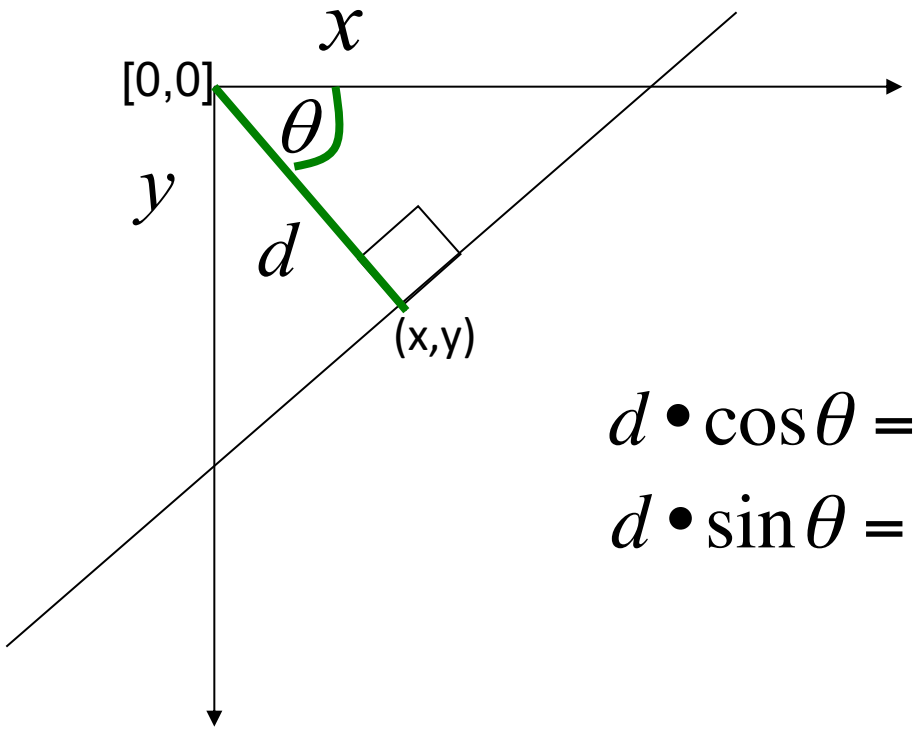
- each edgel point (part of an edge) from image space will vote (for an infinity of lines) in the Hough space (here a line is a point)
- discretize the Hough space in discrete intervals/cells; the cell with largest number of votes determines the parameters of the line

Polar representation

Problems with the (m,b) space:

- Unbounded parameter domains
- Vertical lines require infinite m

Alternative: polar representation



d: distance from origin to line (length of the perpendicular)

θ: angle between the perpendicular and Ox axis

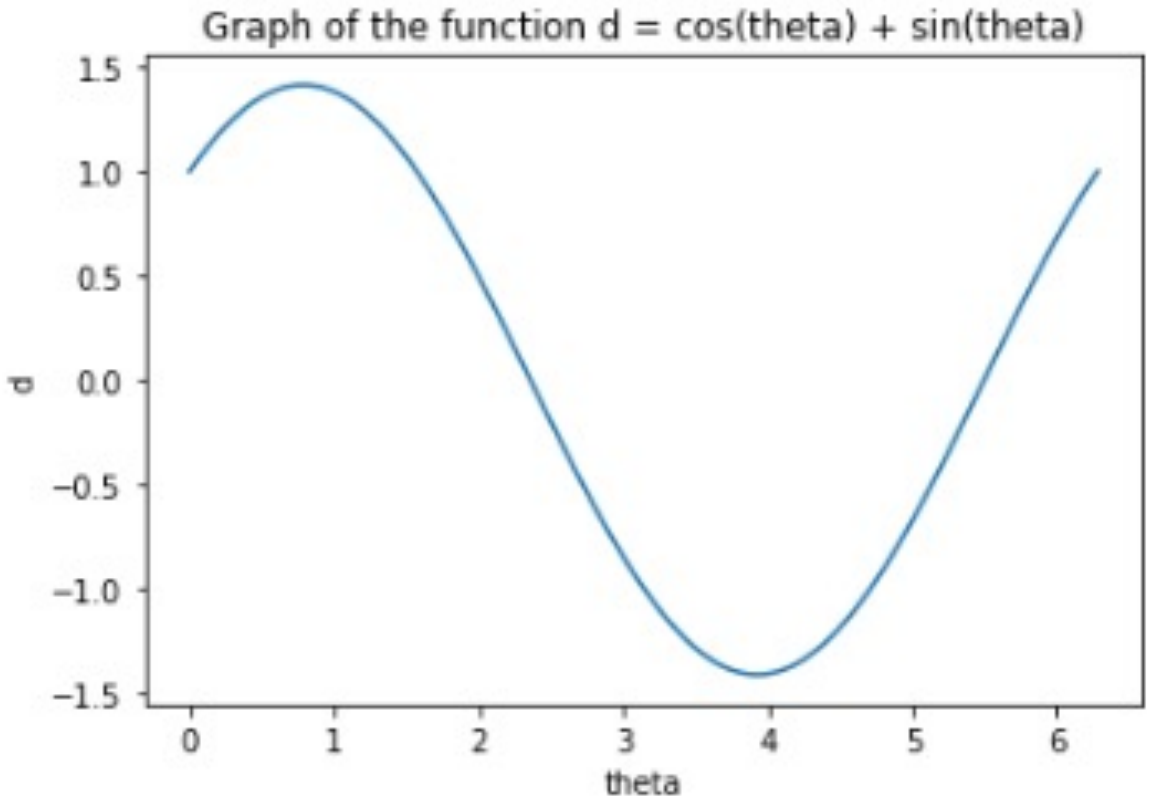
$$d \bullet \cos\theta = x$$
$$d \bullet \sin\theta = y$$

$$d \bullet (\cos\theta)^2 = x \bullet \cos\theta$$
$$d \bullet (\sin\theta)^2 = y \bullet \sin\theta$$

$$d = x \bullet \cos\theta + y \bullet \sin\theta$$

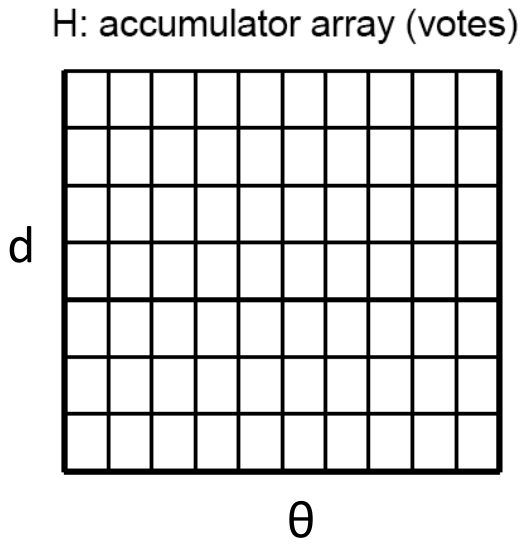
Point in the image space → sinusoid curve in the Hough space

```python
import numpy as np
import matplotlib.pyplot as plt
x, y = 1,1
theta = np.linspace(0,2*np.pi,100)
d = x * np.cos(theta) + y * np.sin(theta)
plt.figure,
plt.plot(theta, d)
plt.xlabel('theta')
plt.ylabel('d')
plt.title('Graph of the function d = cos(theta) + sin(theta)')
plt.show()
```



Hough transform-algorithm outline

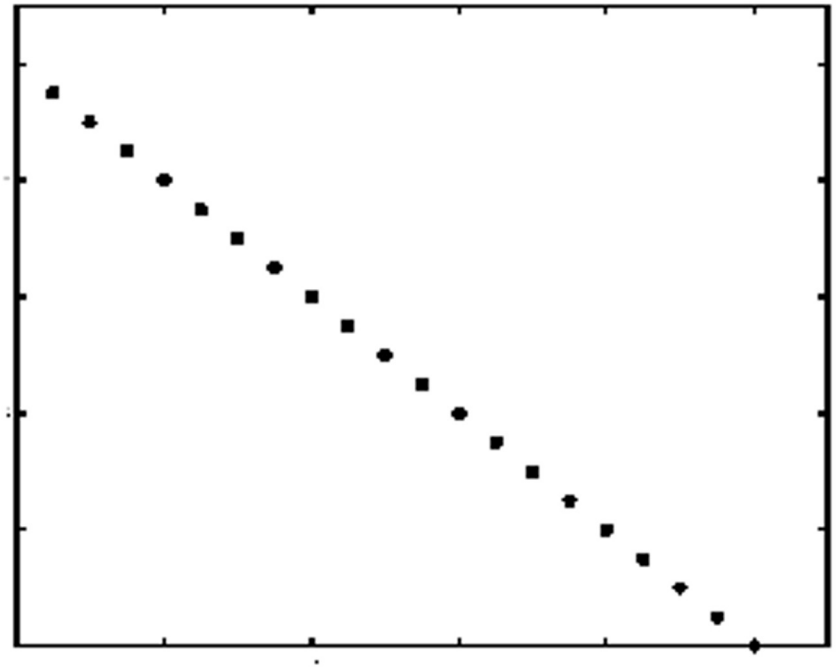
- Initialize accumulator H to all zeros
- For each feature point (x,y) in the image
  For θ = 0 to 180
    $d = x \cos \theta + y \sin \theta$
    $H(\theta, d) = H(\theta, d) + 1$
  end
end
- Find the value(s) of (θ, d) where H(θ, d) is a local maximum
  - The detected line in the image is given by
    $d = x \cos \theta + y \sin \theta$



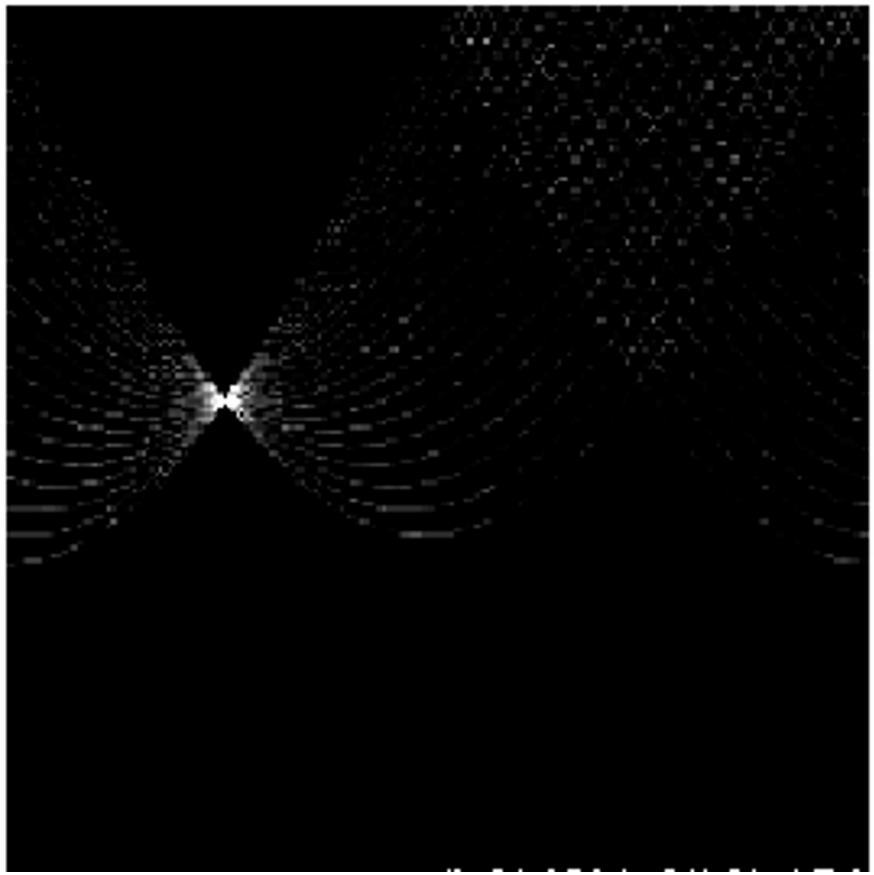
Hough transform - demo

https://www.aber.ac.uk/~dcswww/Dept/Teaching/CourseNotes/current/CS34110/hough.html

Basic illustration



features



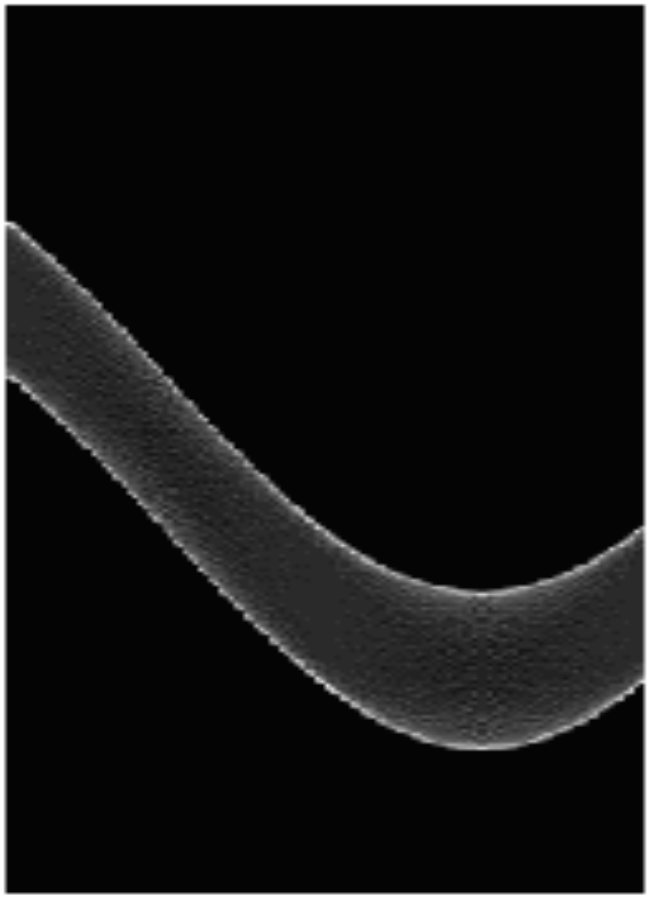
votes

Hough transform demo

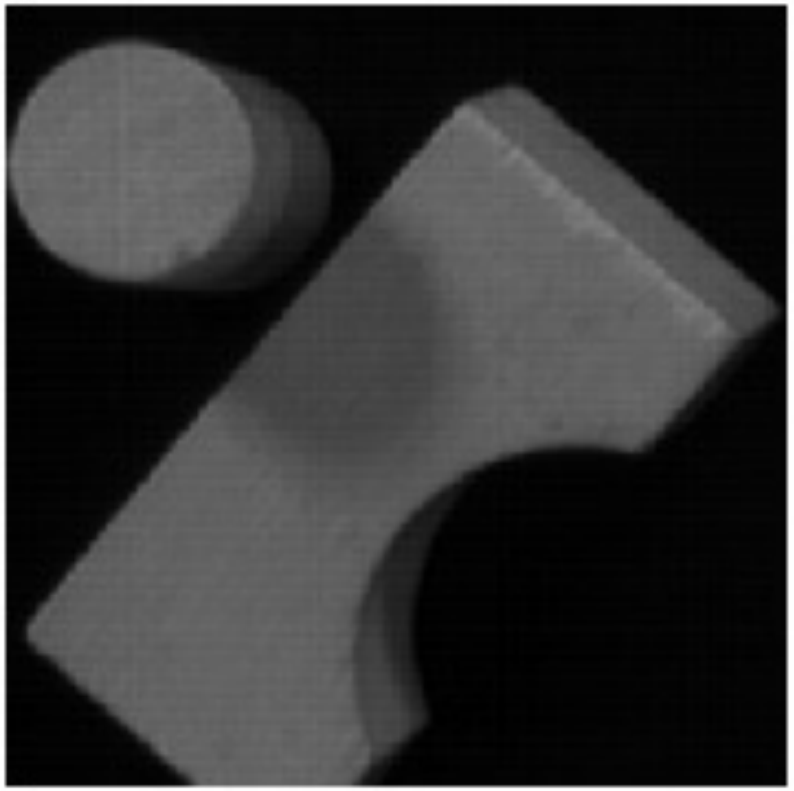
Other shapes

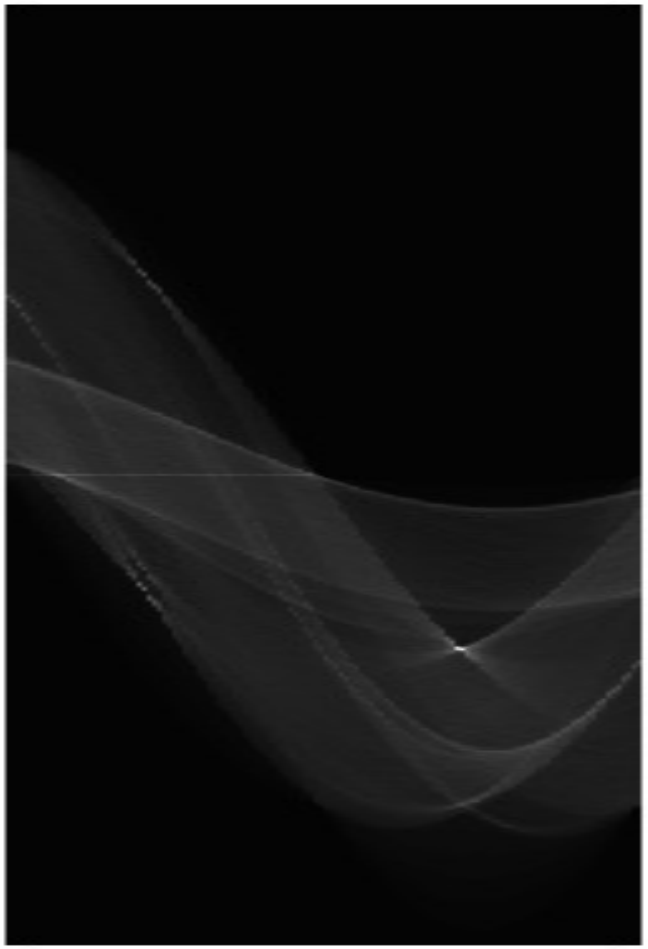
Square

Circle

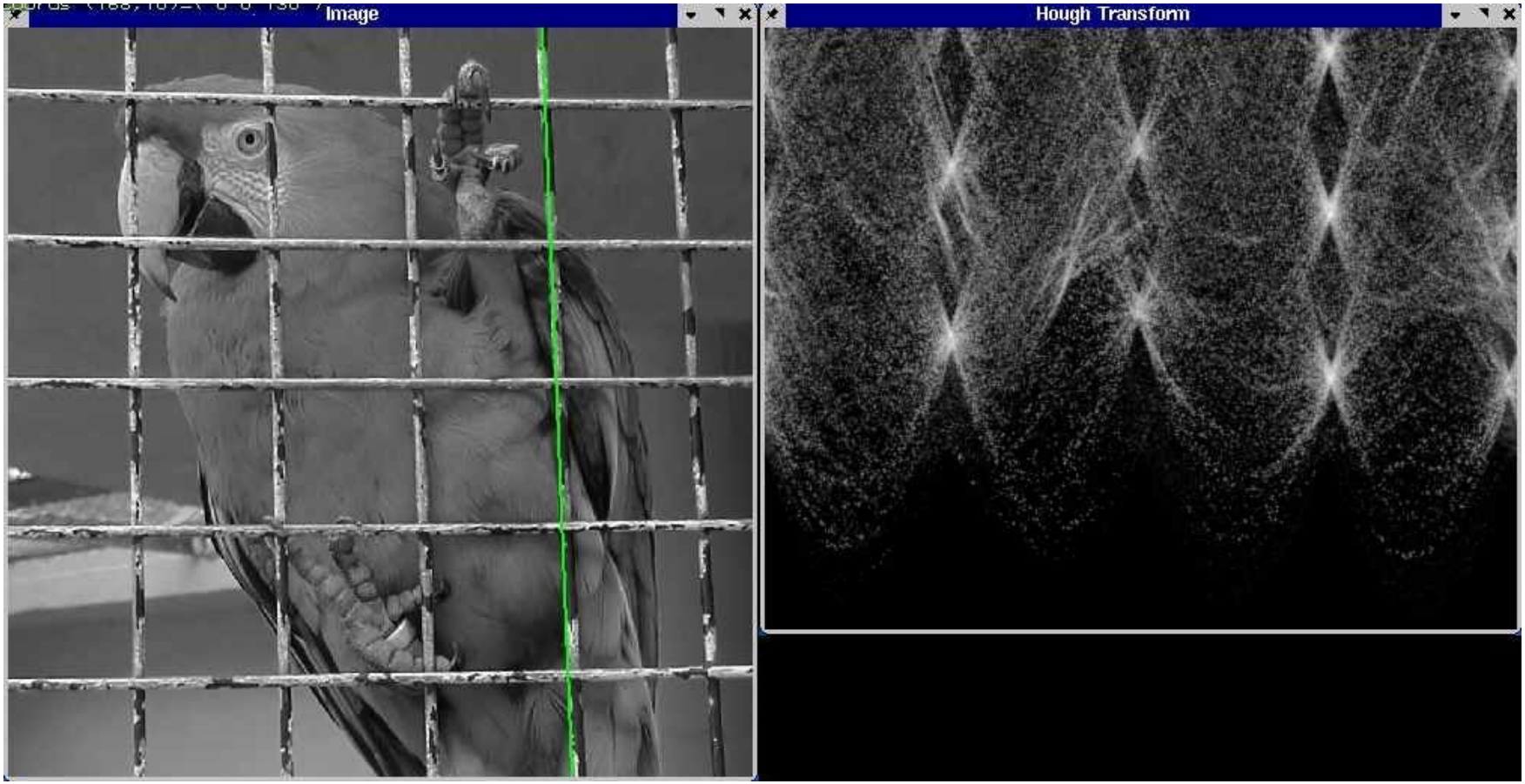


Several lines

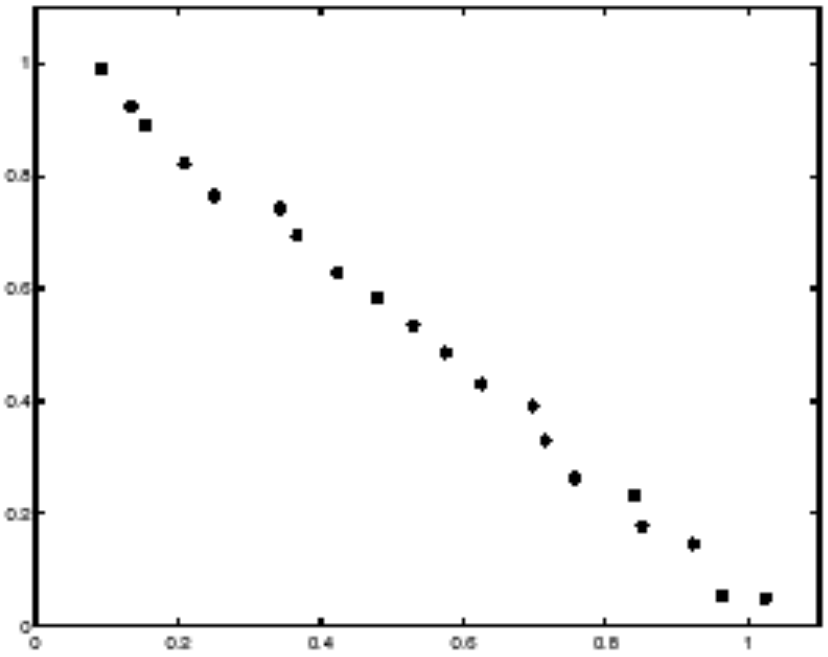




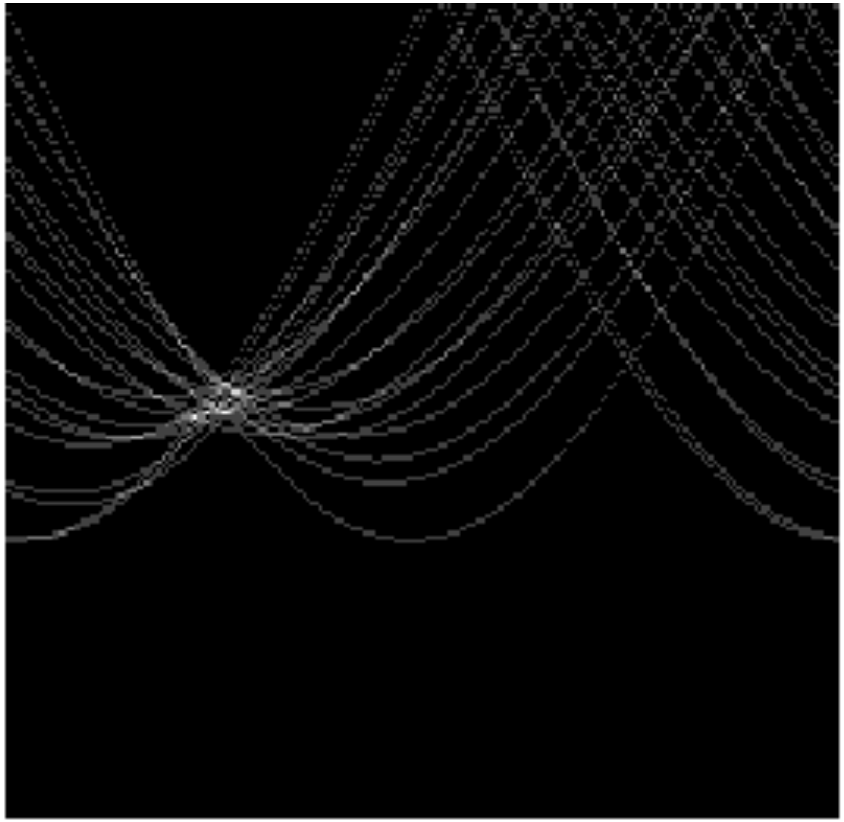
A more complex image



Effect of noise



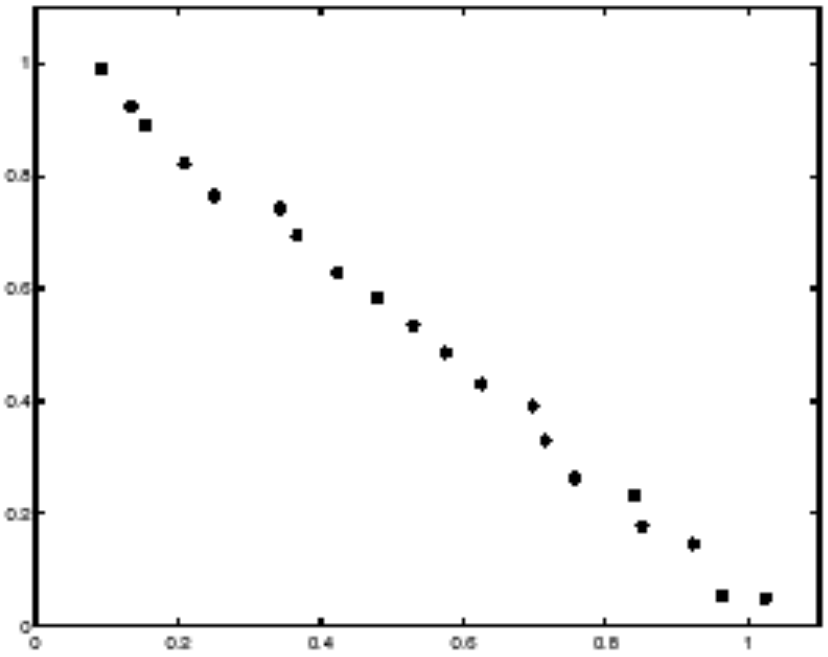
features



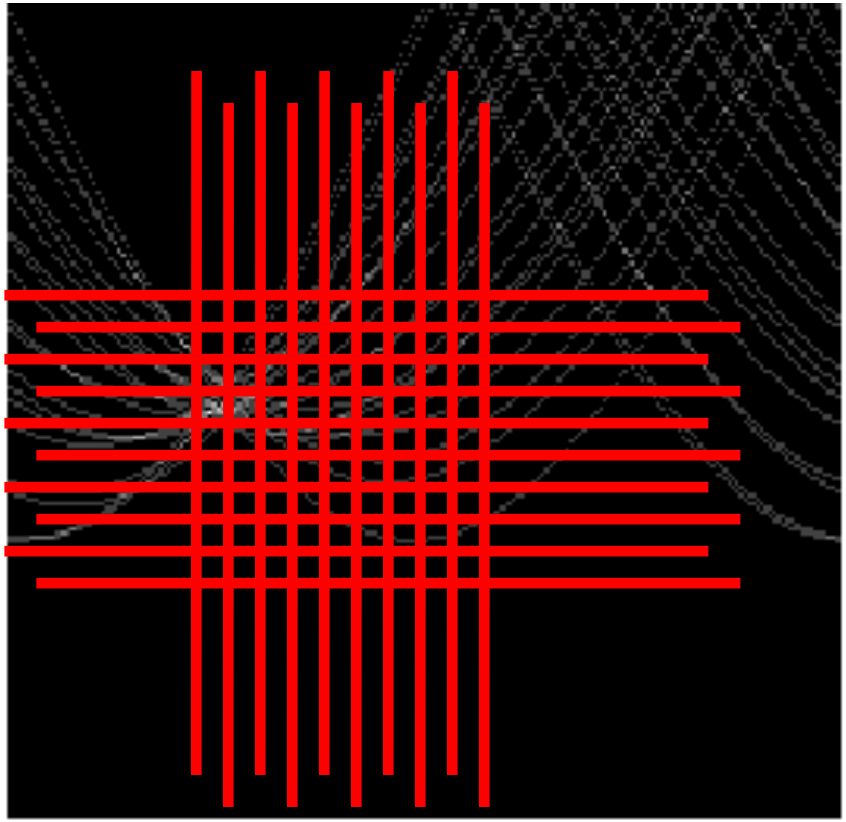
votes

- Peak gets fuzzy and hard to locate

Effect of noise



features

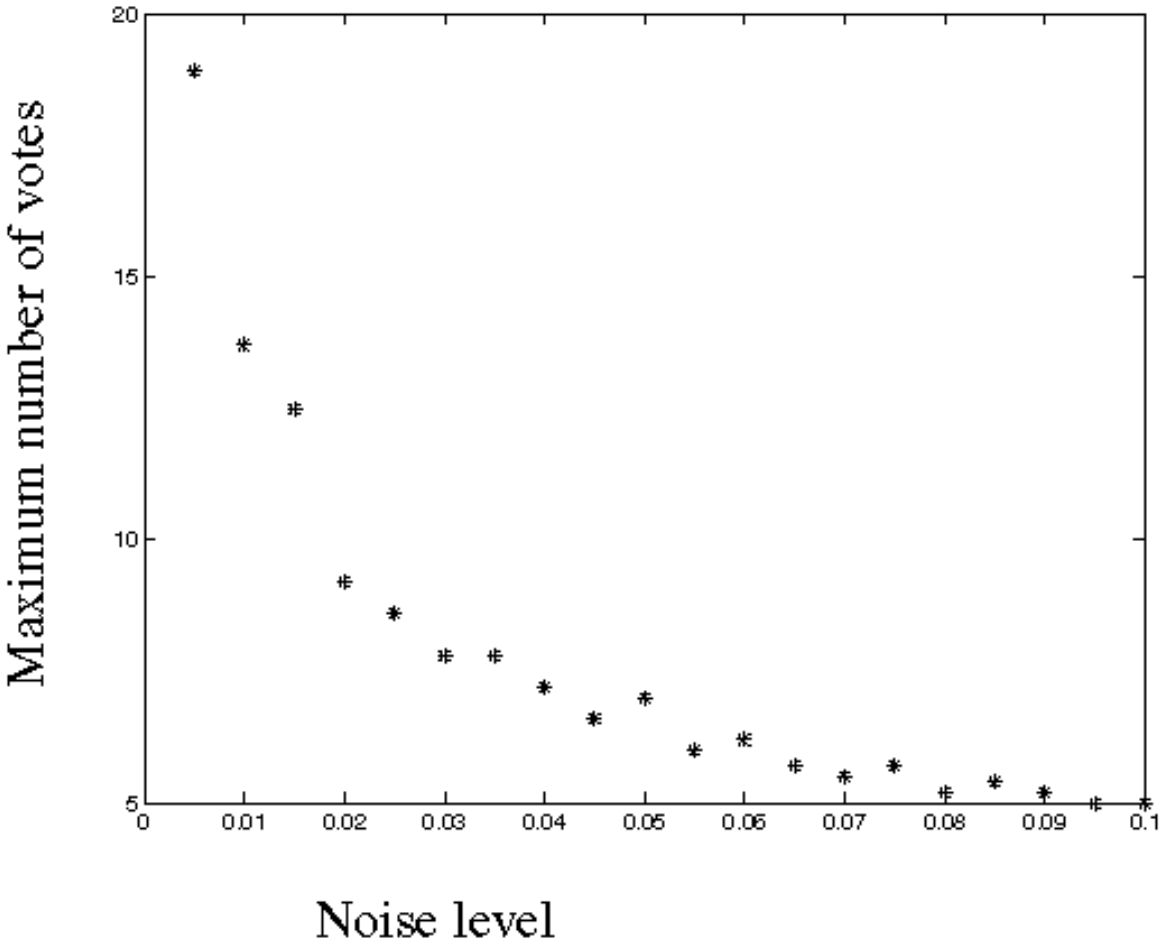


votes

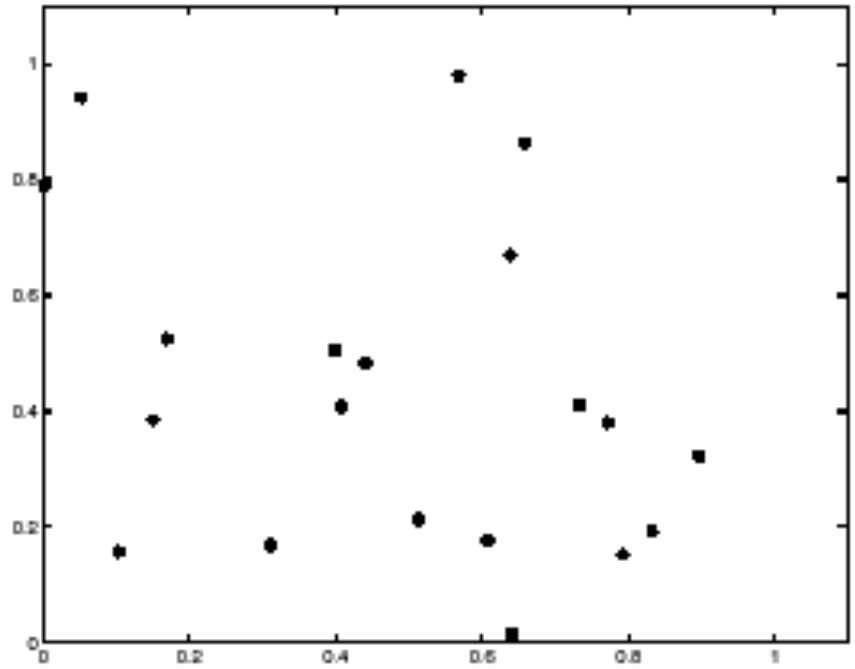
- Peak gets fuzzy and hard to locate

Effect of noise

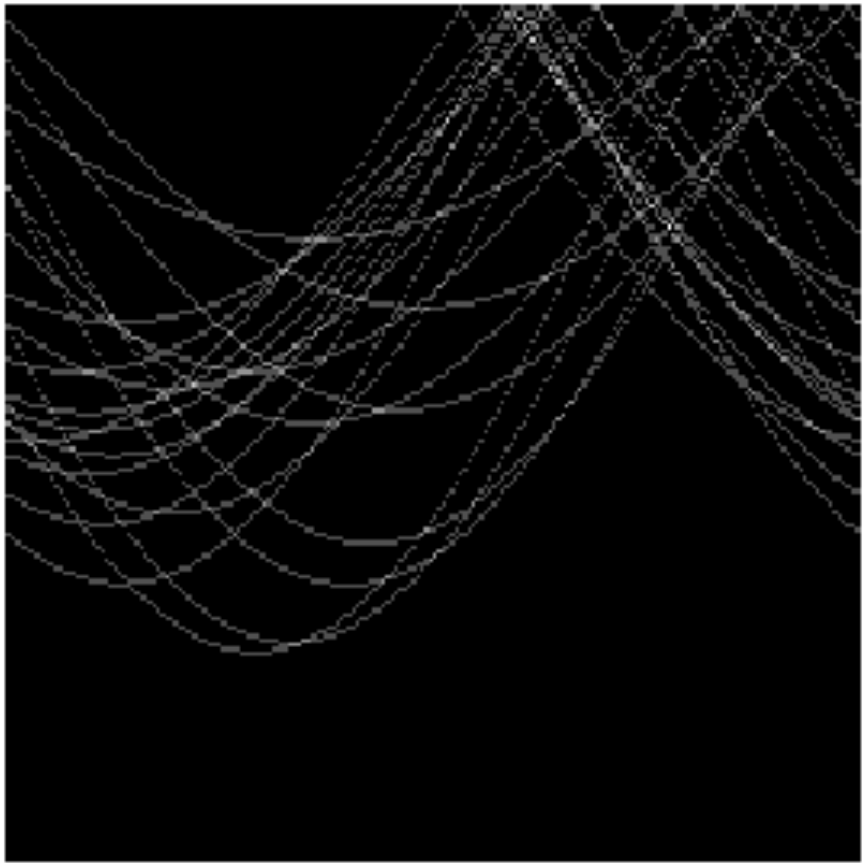
- Number of votes for a line of 20 points with increasing noise:



Random points



features

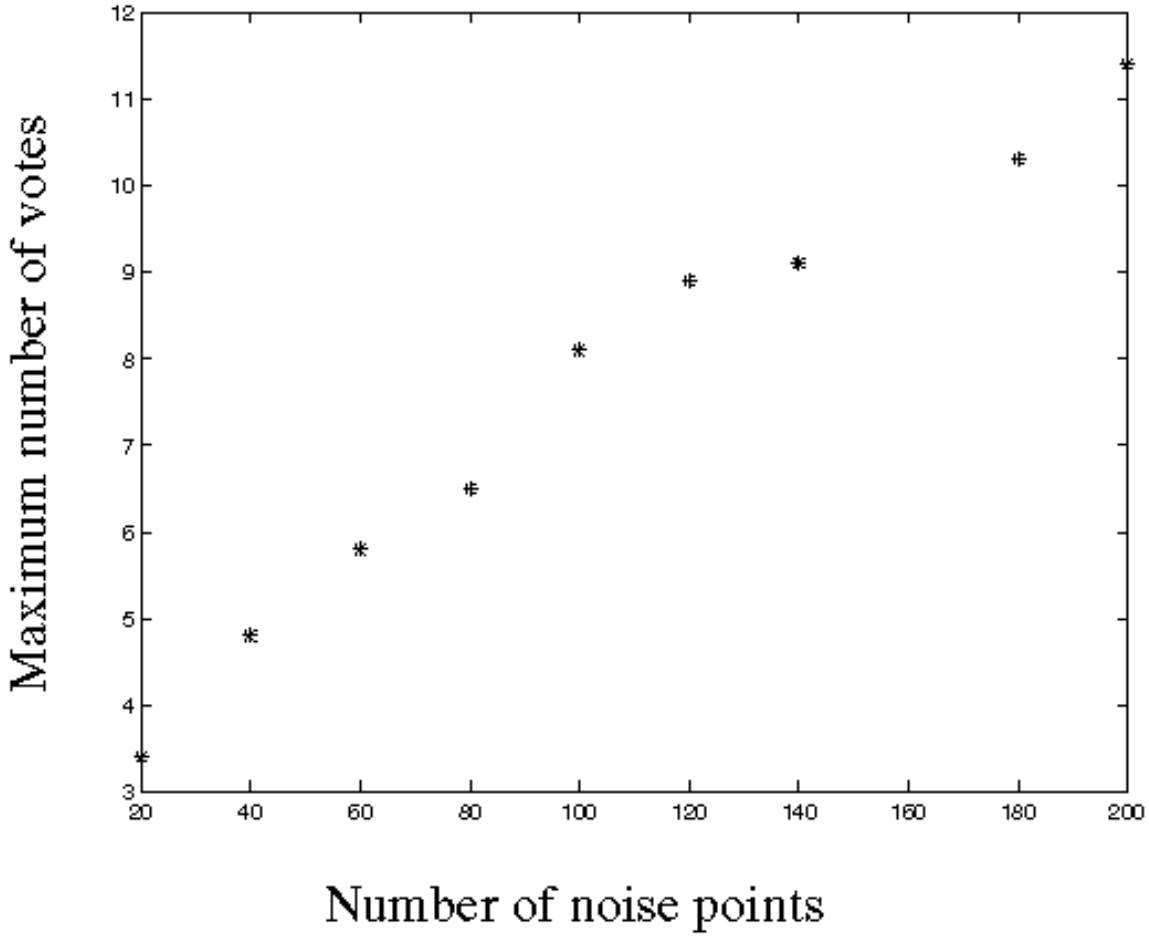


votes

- Uniform noise can lead to spurious peaks in the array

Random points

- As the level of uniform noise increases, the maximum number of votes increases too:

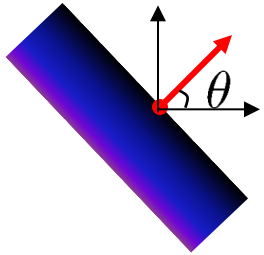


Dealing with noise

- Choose a good grid / discretization
  - **Too coarse:** large votes obtained when too many different lines correspond to a single bucket
  - **Too fine:** miss lines because some points that are not exactly collinear cast votes for different buckets
- Increment neighboring bins (smoothing in accumulator array)
- Try to get rid of irrelevant features
  - E.g., take only edge points with significant gradient magnitude

Incorporating image gradients

- When we detect an edge point, we also know its gradient orientation
- How does this constrain possible lines passing through the point?



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

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

- Modified Hough transform:

- For each edge point (x,y)
  θ = gradient orientation at (x,y)
  d = x cos θ + y sin θ
  H(θ, d) = H(θ, d) + 1

  end

Algorithm outline with image gradients

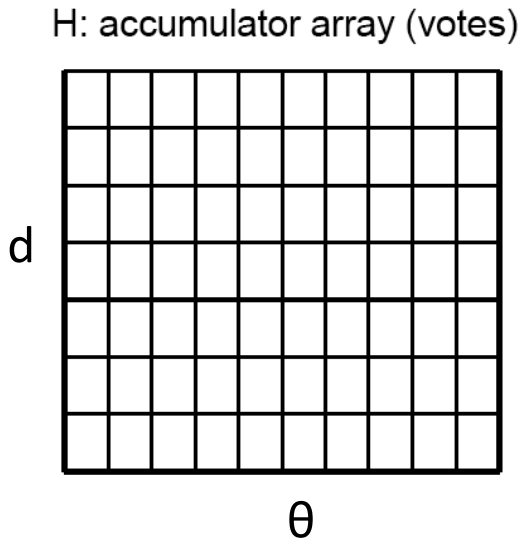
- Initialize accumulator H to all zeros
- For each feature point (x,y) in the image

  θ = gradient orientation at (x,y)

  $d = x \cos \theta + y \sin \theta$

  $H(\theta, d) = H(\theta, d) + 1$

  end



- Find the value(s) of (θ, d) where $H(\theta, d)$ is a local maximum
  - The detected line in the image is given by

    $d = x \cos \theta + y \sin \theta$

Algorithm outline with image gradients

- Initialize accumulator H to all zeros
- For each feature point (x,y) in the image

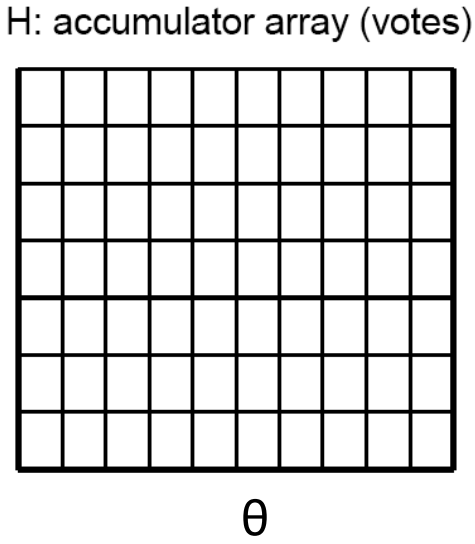
  θ = gradient orientation at (x,y)

  mag = gradient magnitude at (x,y)

  $d = x \cos\theta + y \sin\theta$

  $H(\theta, d) = H(\theta, d) + mag$

  end



- Find the value(s) of (θ, d) where $H(\theta, d)$ is a local maximum
  - The detected line in the image is given by

    $d = x \cos\theta + y \sin\theta$

Hough transform for circles

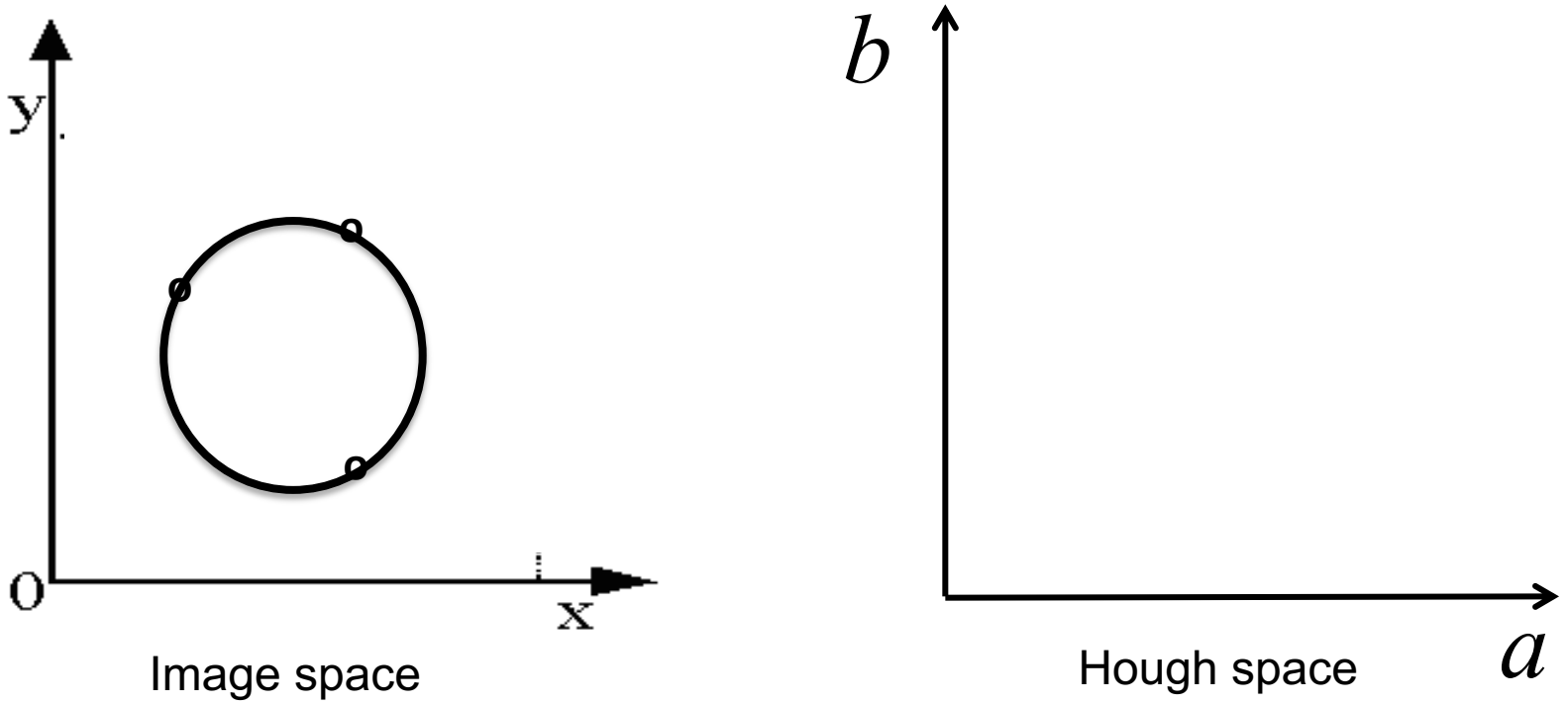
- How many dimensions will the parameter space have?
- Given an unoriented edge point, what are all possible bins that it can vote for?
- What about an *oriented* edge point?

Hough transform for circles

- Circle: center (a,b) and radius r. Equation?

$$(x_i - a)^2 + (y_i - b)^2 = r^2$$

- For fixed radius r

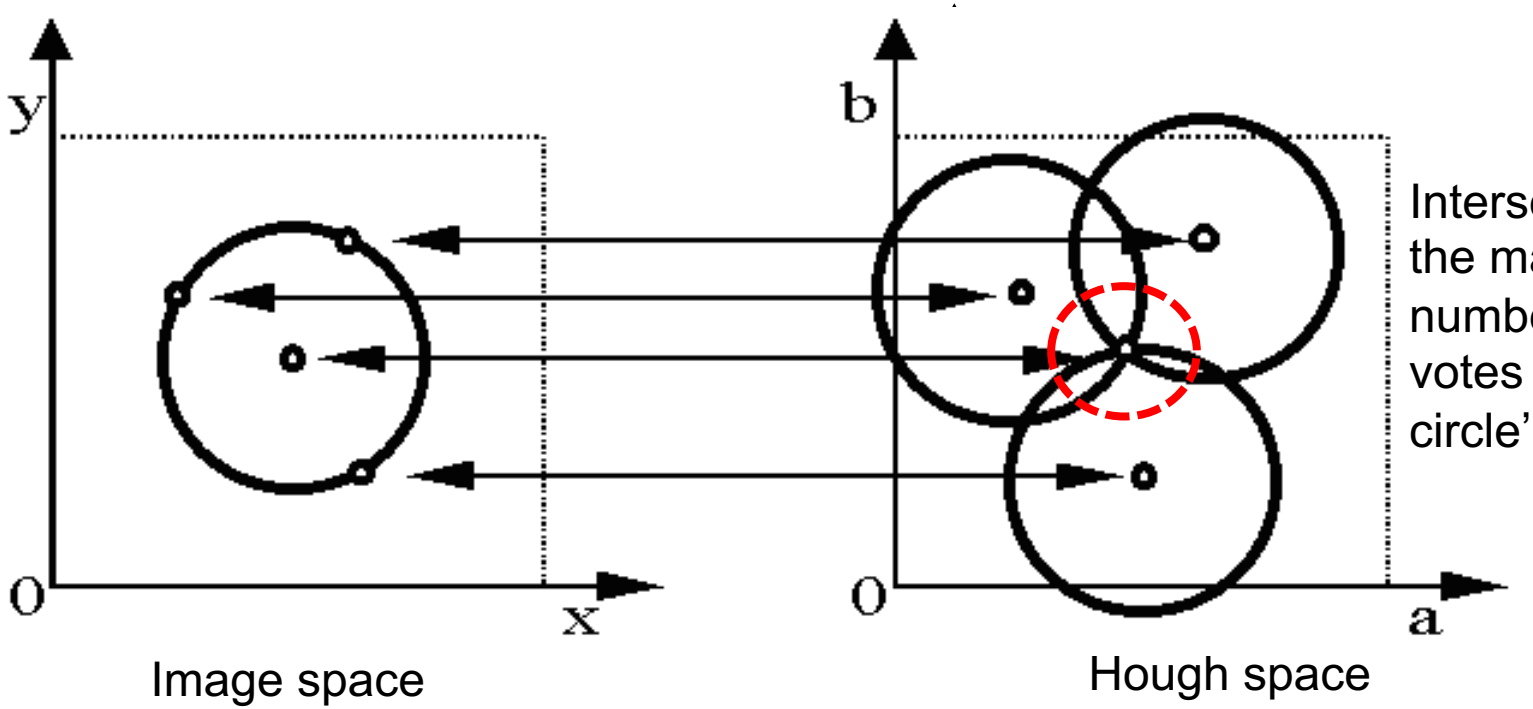


Hough transform for circles

- Circle: center (a,b) and radius r. Equation?

$$(x_i - a)^2 + (y_i - b)^2 = r^2$$

- For fixed radius r



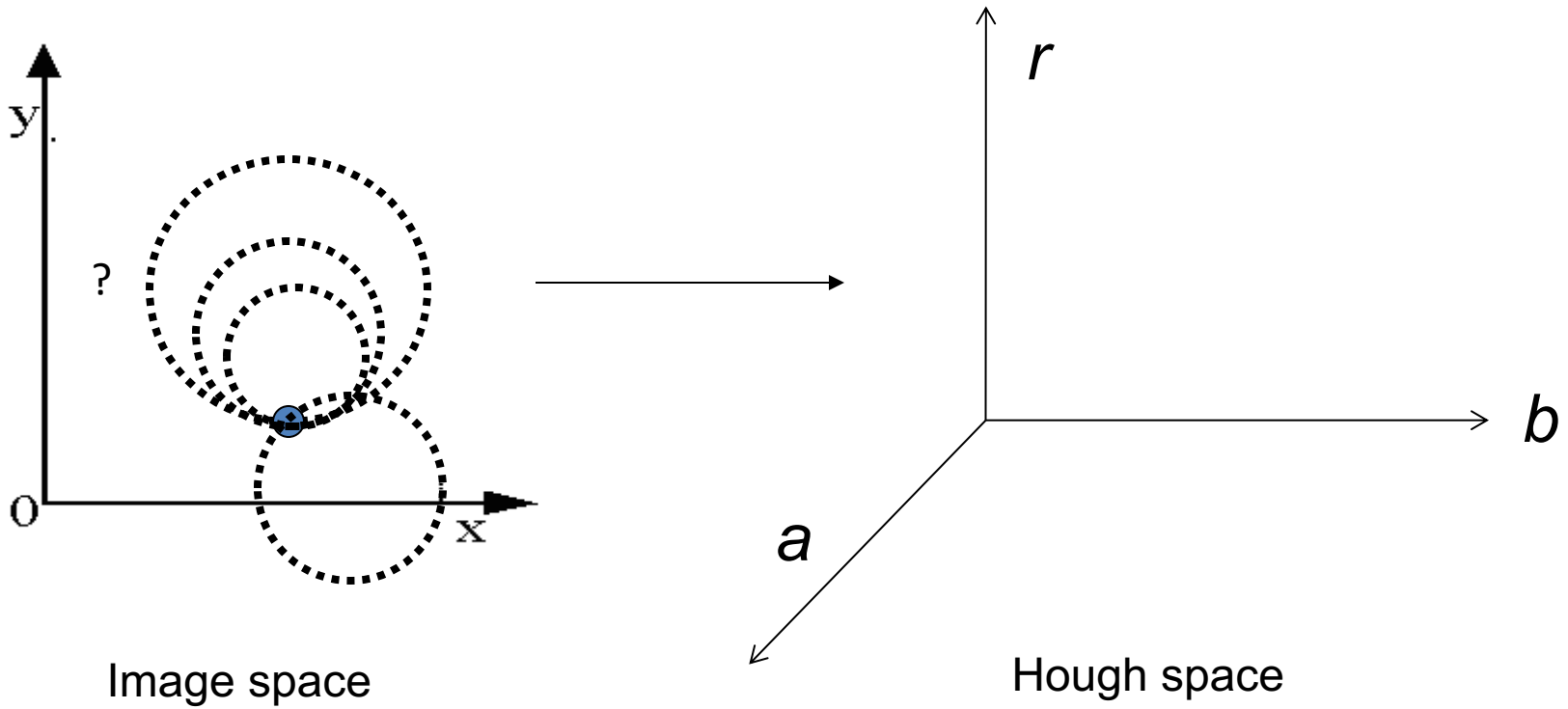
Slide credit: Kristen Grauman

Hough transform for circles

- Circle: center (a,b) and radius r. Equation?

$$(x_i - a)^2 + (y_i - b)^2 = r^2$$

- For unknown radius r



Slide credit: Kristen Grauman

Hough transform for circles

- Circle: center (a,b) and radius r. Equation?

$$(x_i - a)^2 + (y_i - b)^2 = r^2$$

- For unknown radius r

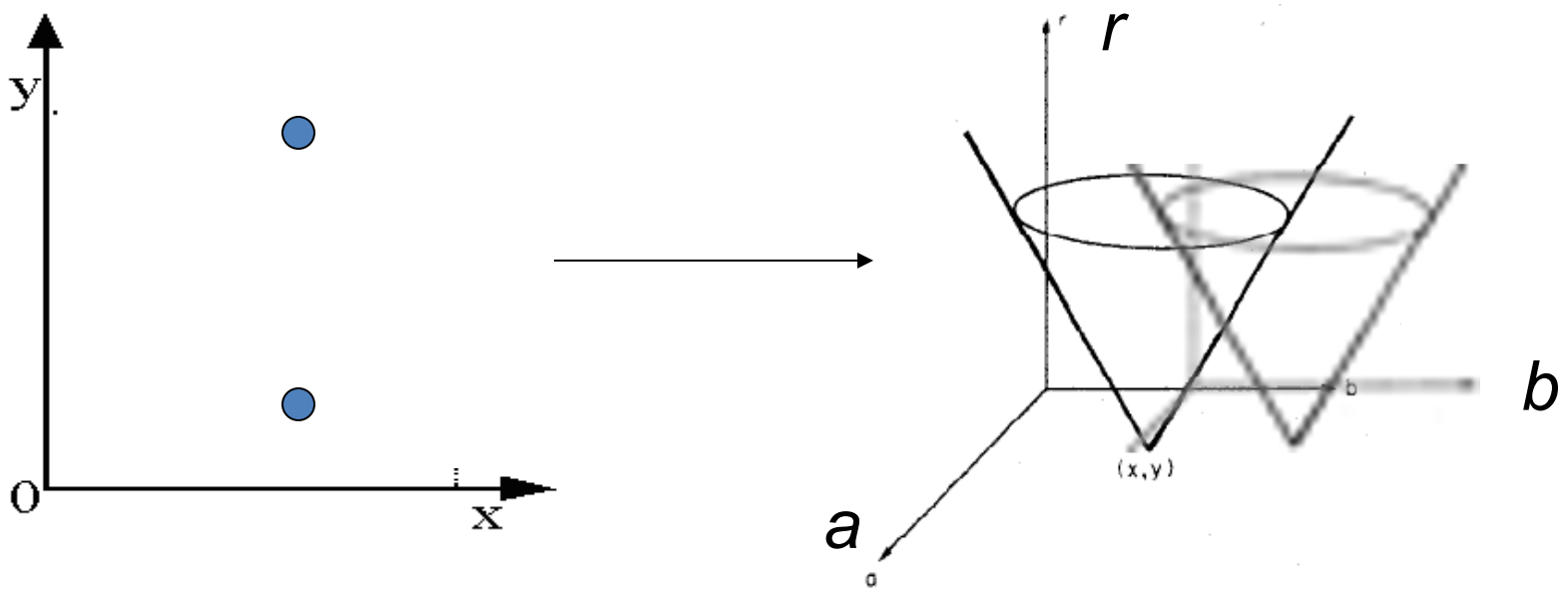
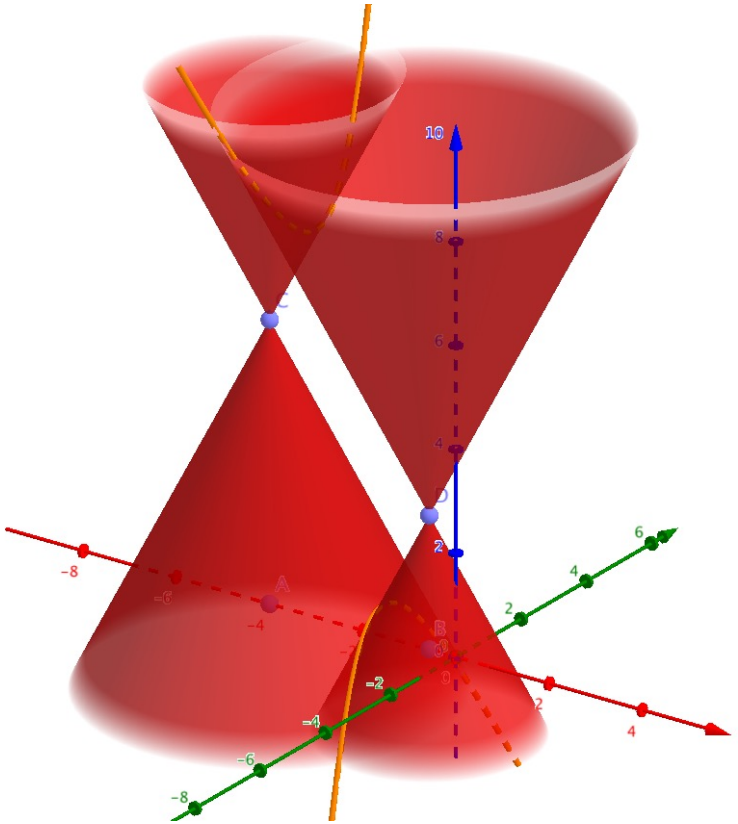


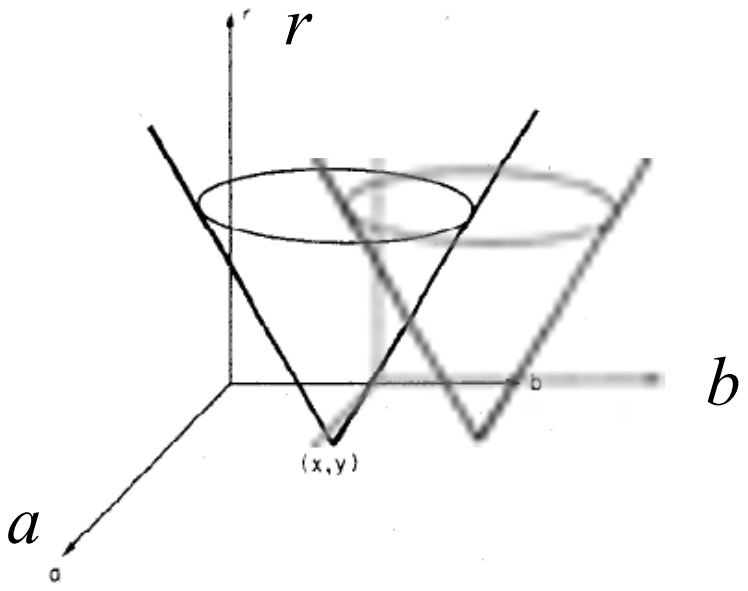
Image space

Hough space

Slide credit: Kristen Grauman

Intersection of two cones





Hough space

Hough transform for circles

- Circle: center (a,b) and radius r. Equation?

$$(x_i - a)^2 + (y_i - b)^2 = r^2$$

- For unknown radius r, with known gradient direction

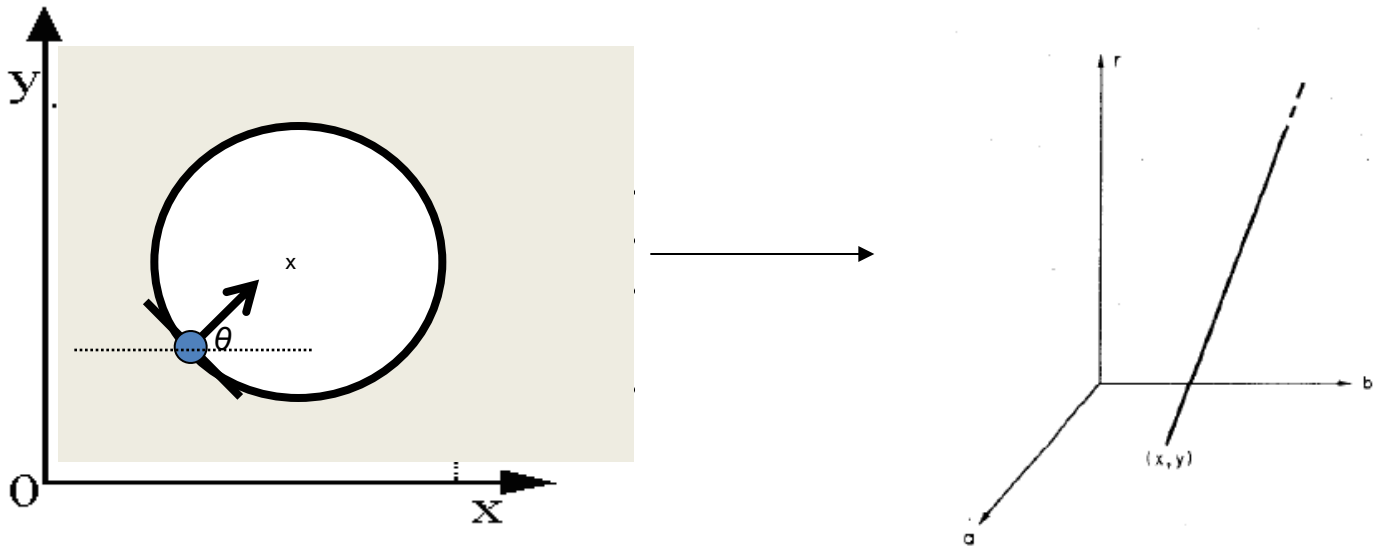


Image space

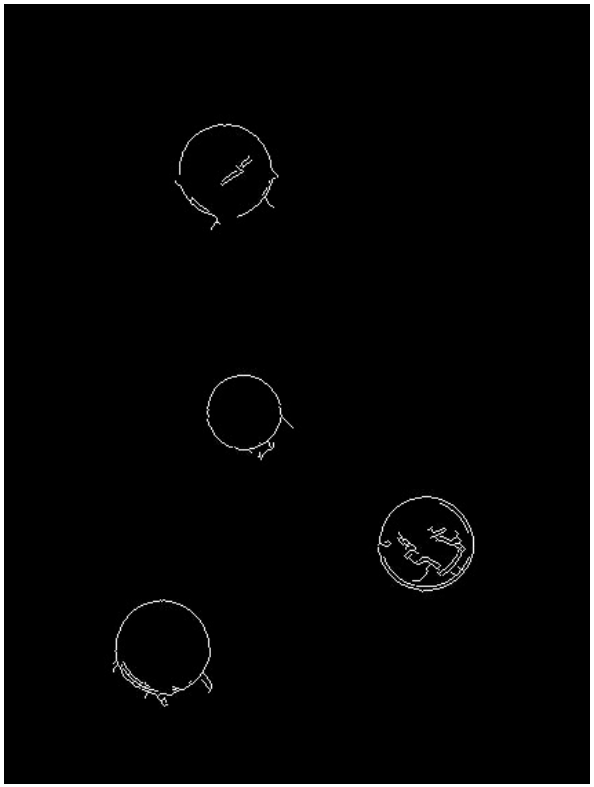
Hough space

Slide credit: Kristen Grauman

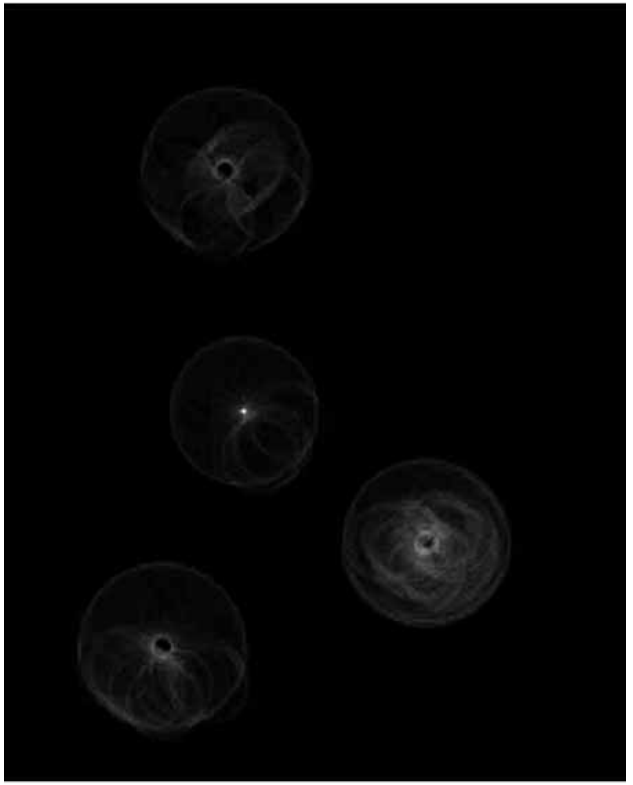
Circle detection using Hough transform

Original image

Edges



Votes ray 1



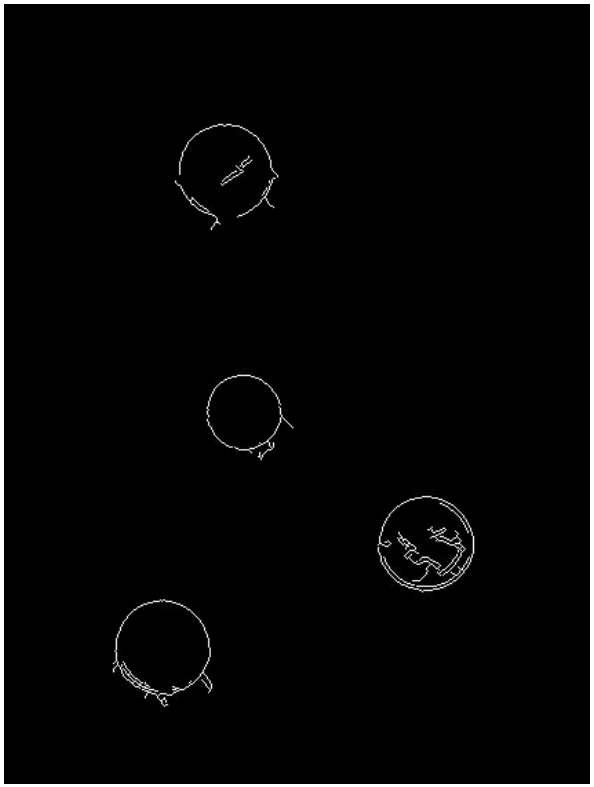
Use two Hough transforms, one for each ray (of known length)

Slide credit: Kristen Grauman

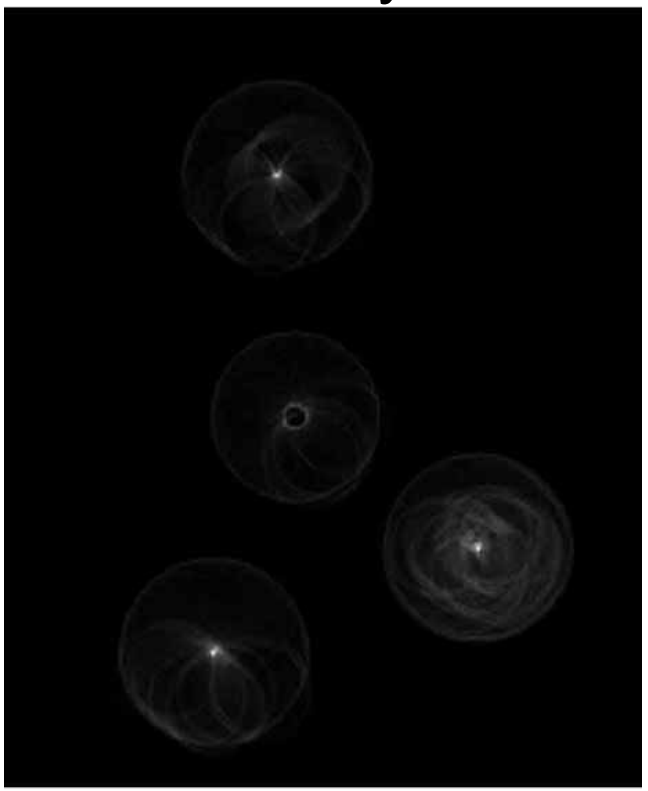
Circle detection using Hough transform

Original image

Edges



Votes ray 2

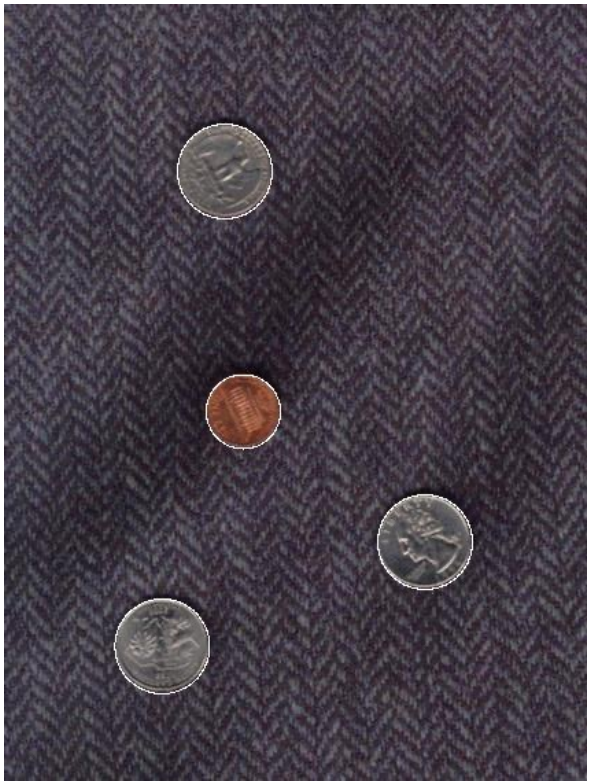


Use two Hough transforms, one for each ray (of known length)

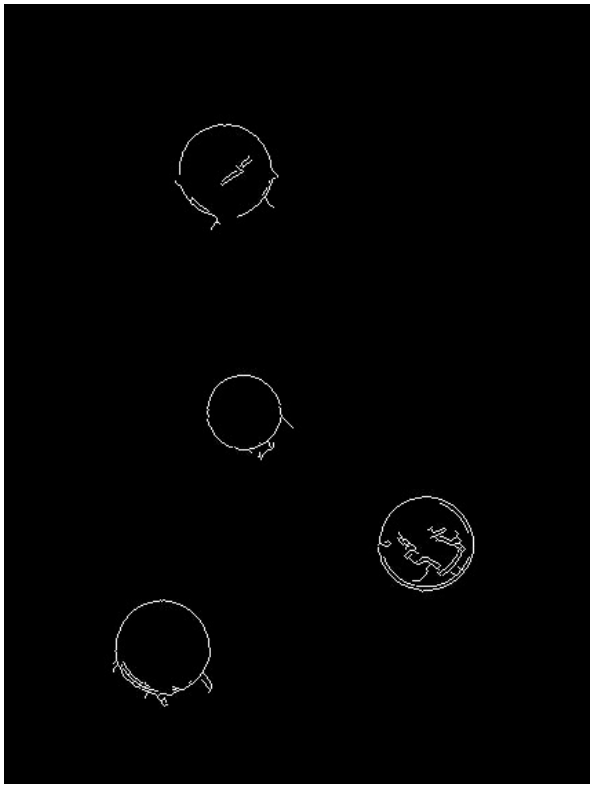
Slide credit: Kristen Grauman

Circle detection using Hough transform

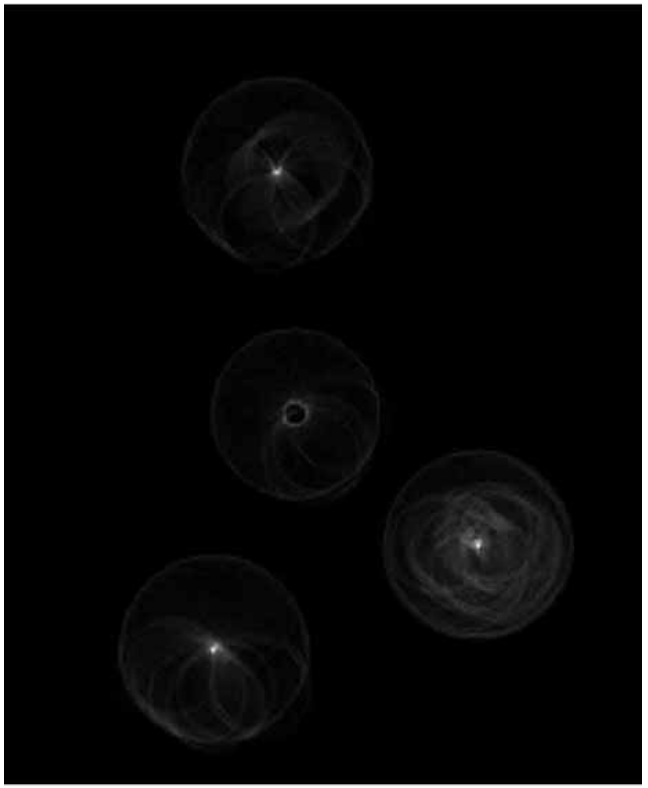
Combined detections



Edges



All votes



Slide credit: Kristen Grauman

Hough transform: Pros and cons

- Pros
  - Can deal with non-locality and occlusion
  - Can detect multiple instances of a model
  - Some robustness to noise: noise points unlikely to contribute consistently to any single bin
- Cons
  - Complexity of search time increases exponentially with the number of model parameters
  - Non-target shapes can produce spurious peaks in parameter space
  - It's hard to pick a good grid size

Slide credit: Kristen Grauman