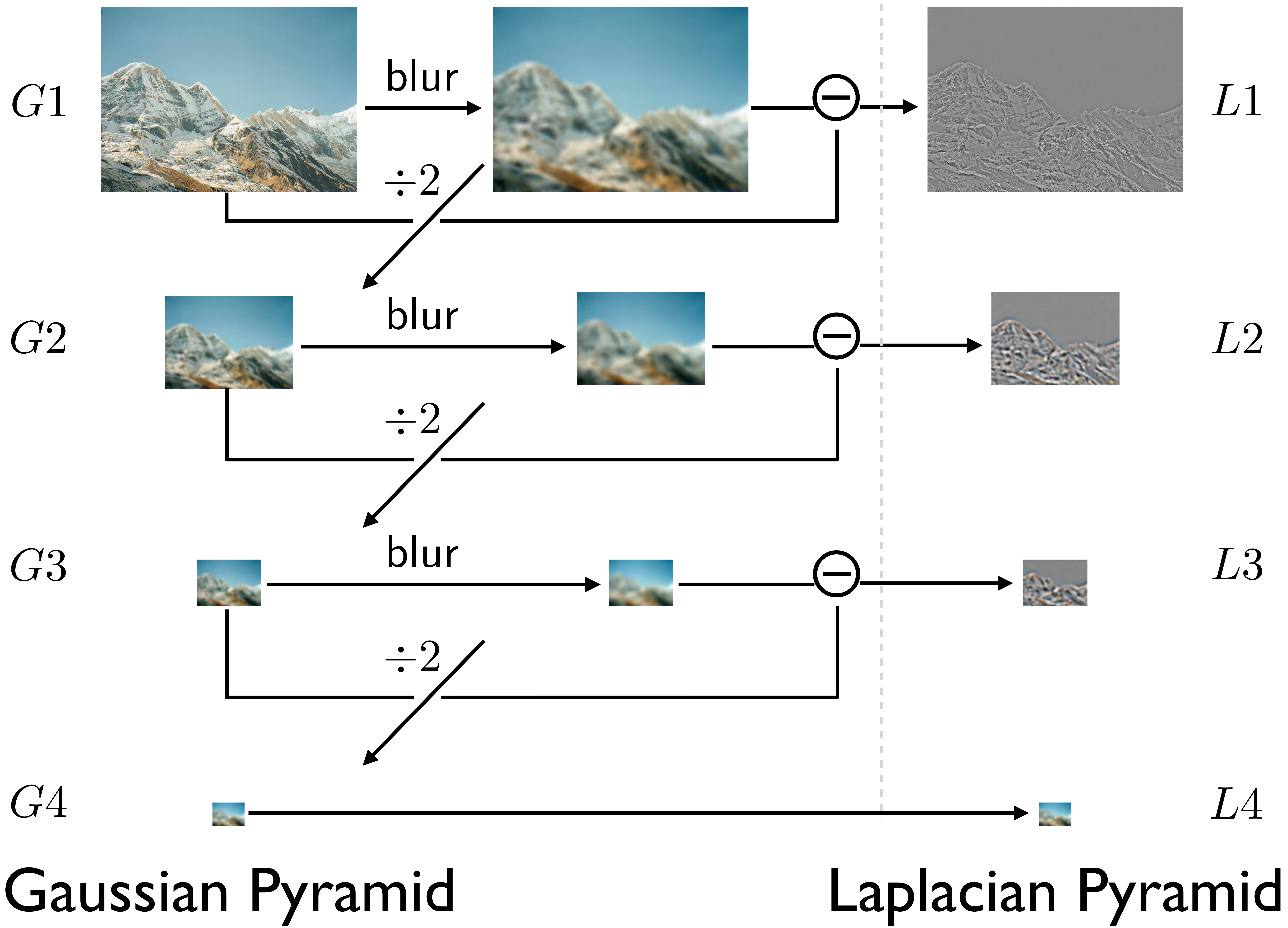
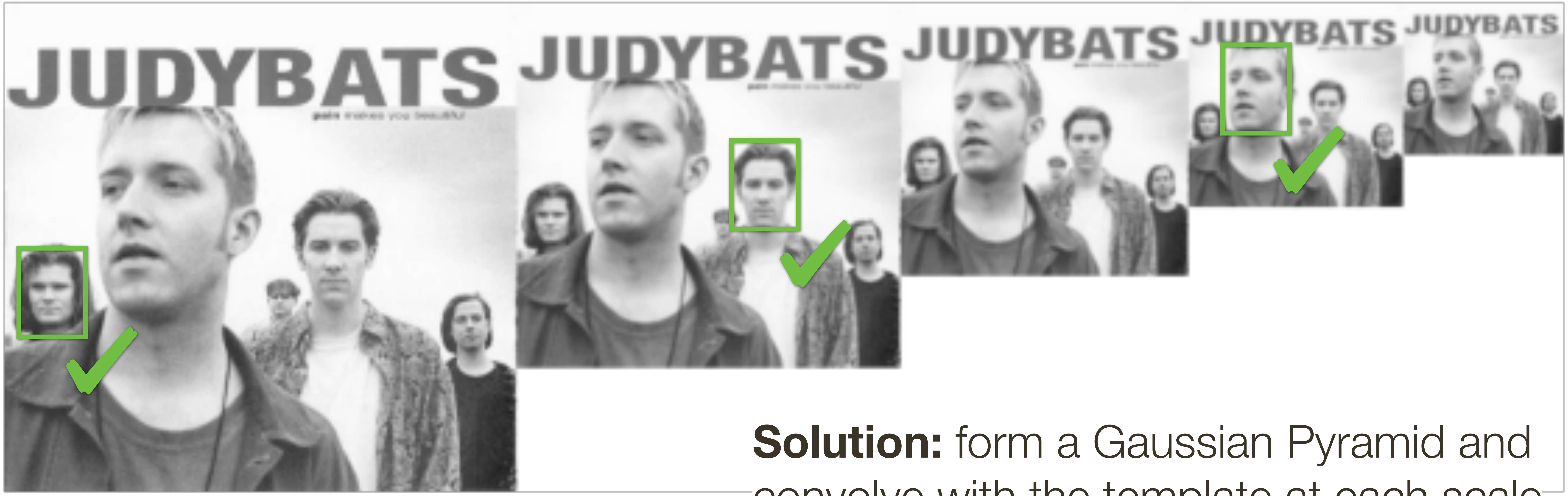

$G1$
blur
$\div 2$
$L1$
$G2$
blur
$\div 2$
$L2$
$G3$
blur
$\div 2$
$L3$
$G4$
$L4$
Gaussian Pyramid
Laplacian Pyramid

Recap: **Multi-Scale** Template Matching

Correlation with a **fixed-sized image** only detects faces at **specific scales**



Solution: form a Gaussian Pyramid and convolve with the template at each scale

Q. **Why scale** the **image** and not the **template**?

= Template

Improving Template Matching

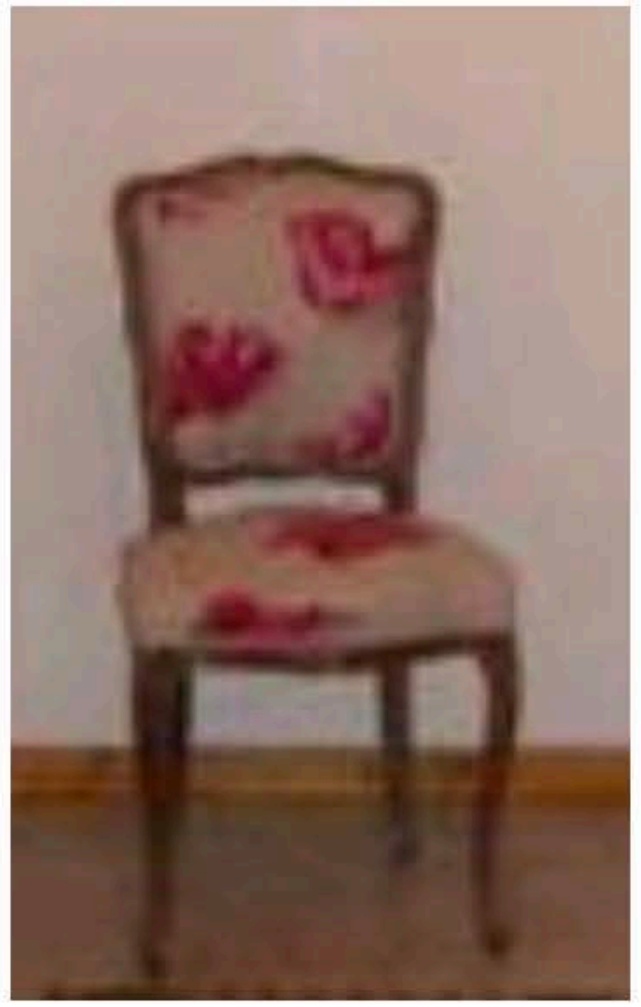
Consider the problem of finding images of an elephant using a template

An elephant looks different from different viewpoints
— from above (as in an aerial photograph or satellite image)
— head on
— sideways (i.e., in profile)
— rear on

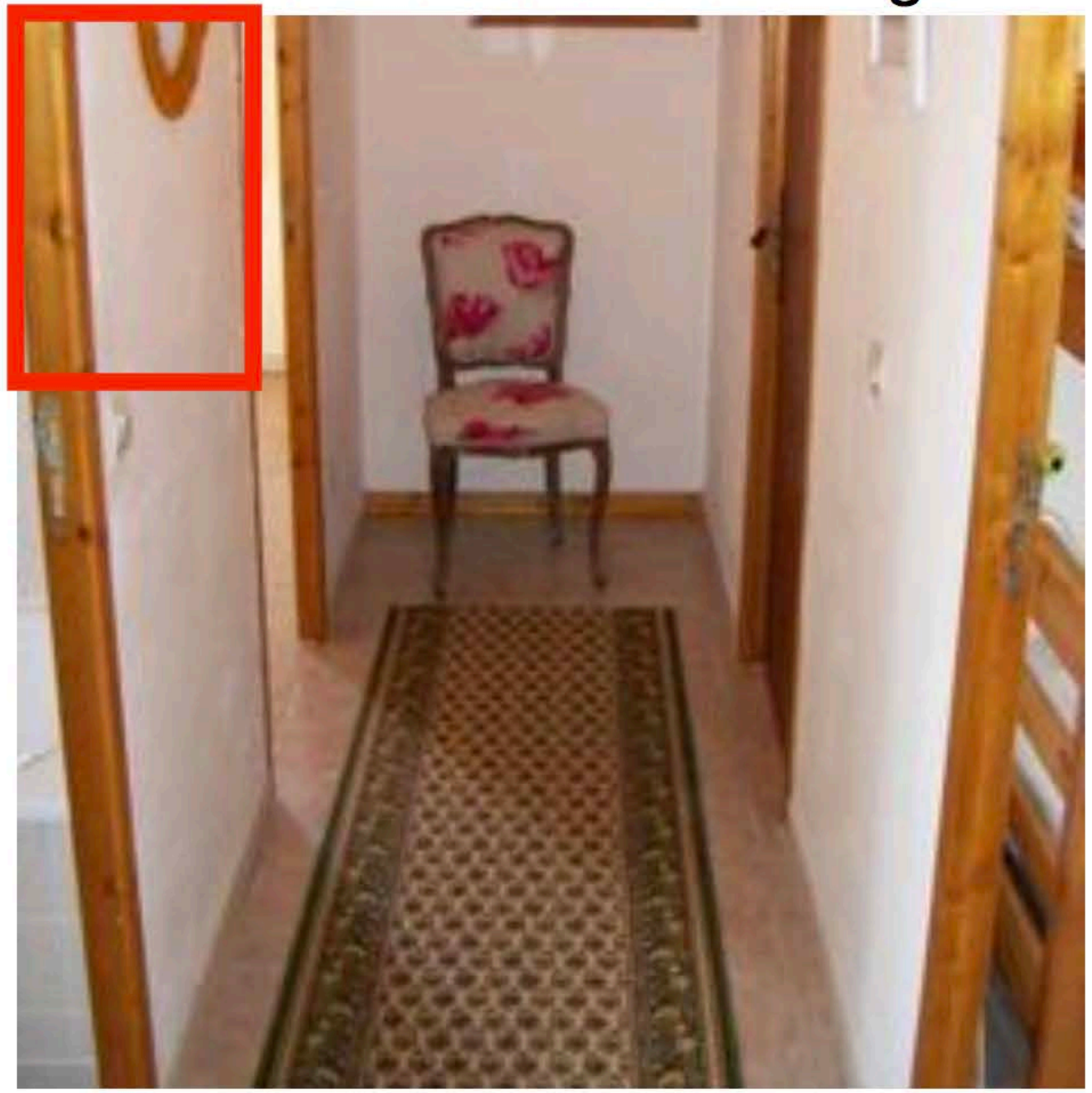
What happens if parts of an elephant are obscured from view by trees, rocks, other elephants?

Improving Template Matching

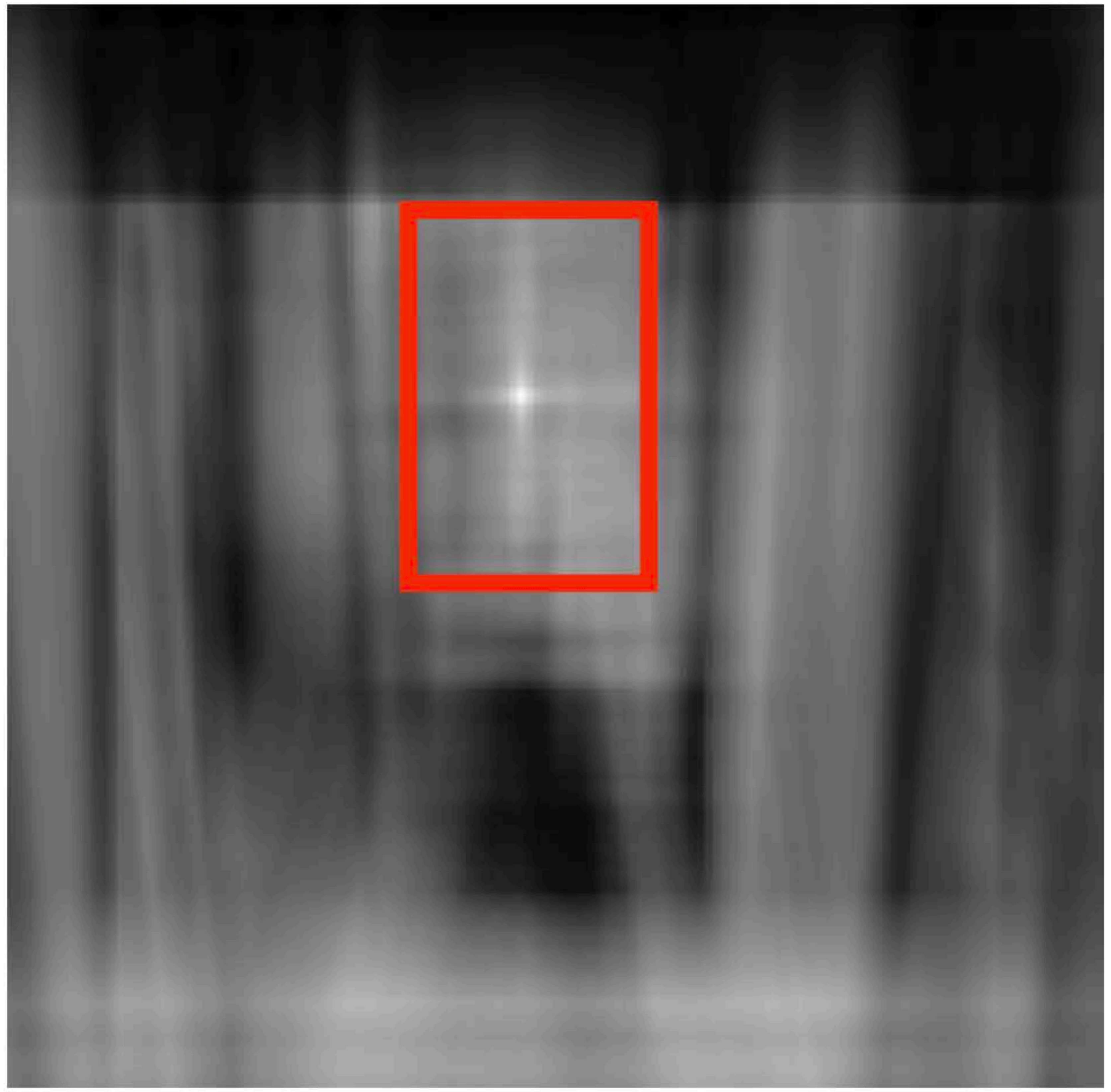
This is a chair



Find the chair in this image

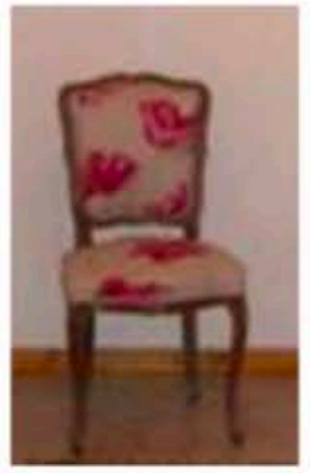


Output of normalized correlation

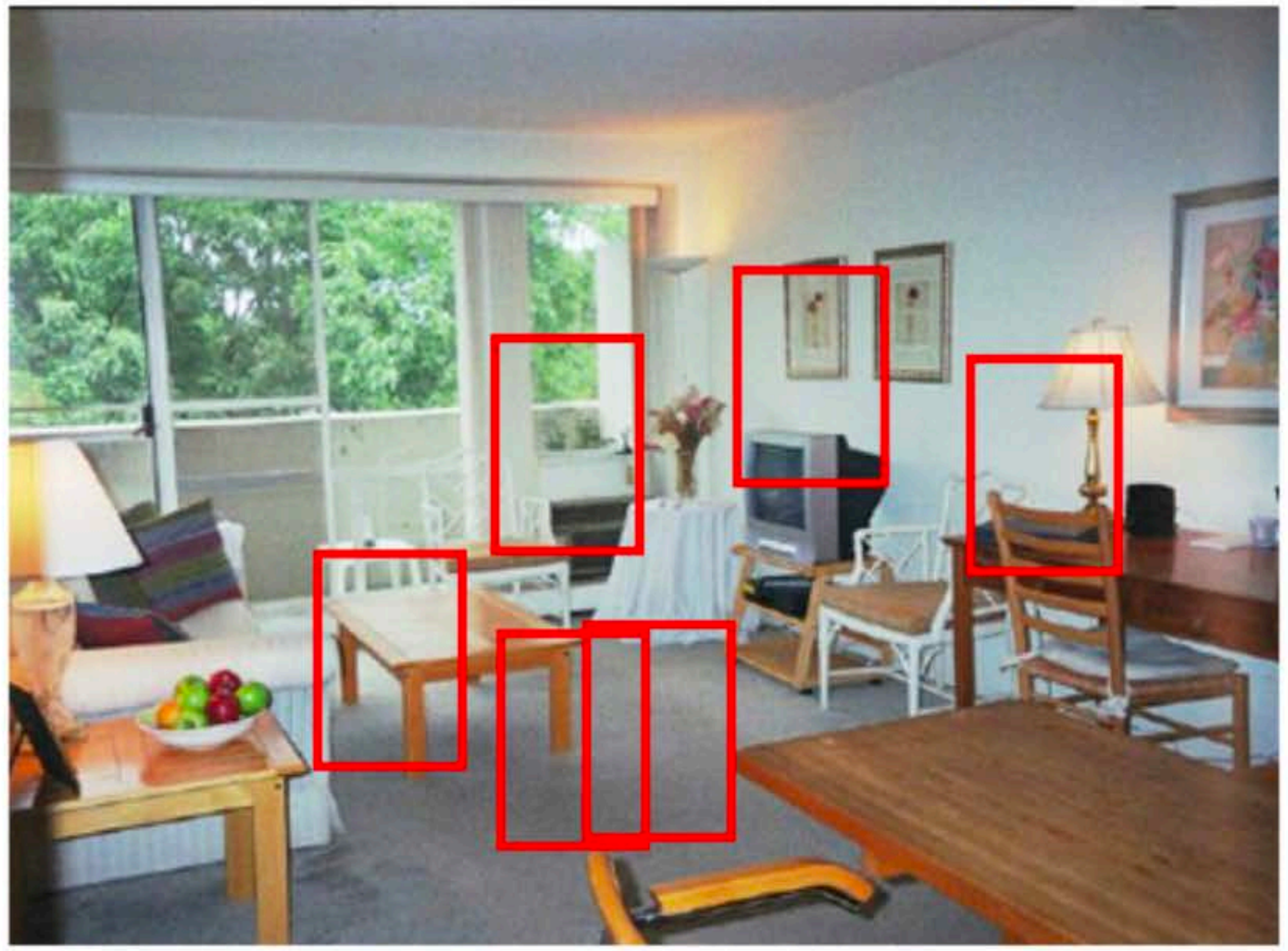


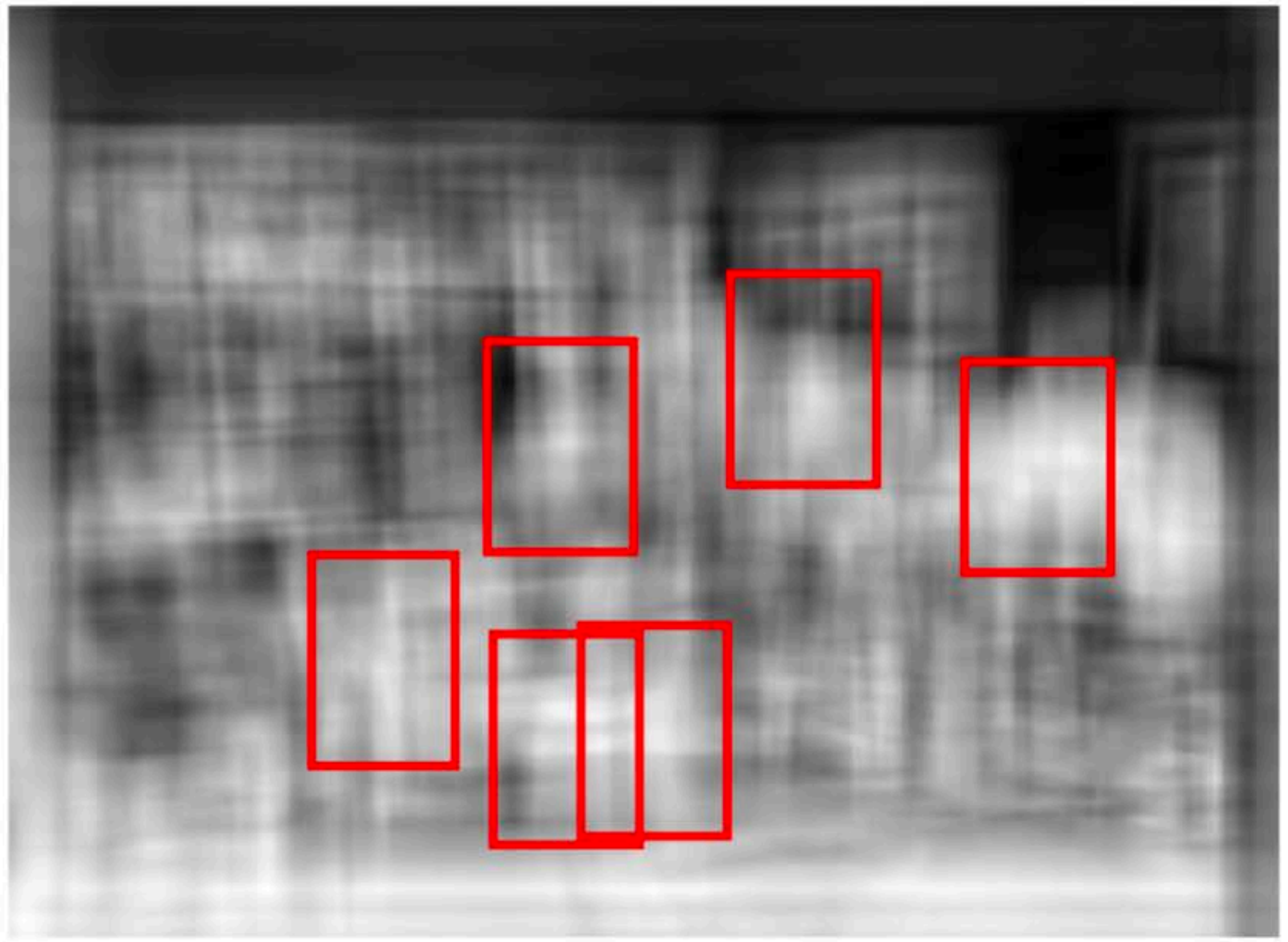
Slide Credit: Li Fei-Fei, Rob Fergus, and Antonio Torralba

Improving Template Matching



Find the chair in this image





Pretty much garbage
Simple template matching is not going to make it

Slide Credit: Li Fei-Fei, Rob Fergus, and Antonio Torralba

Improving Template Matching

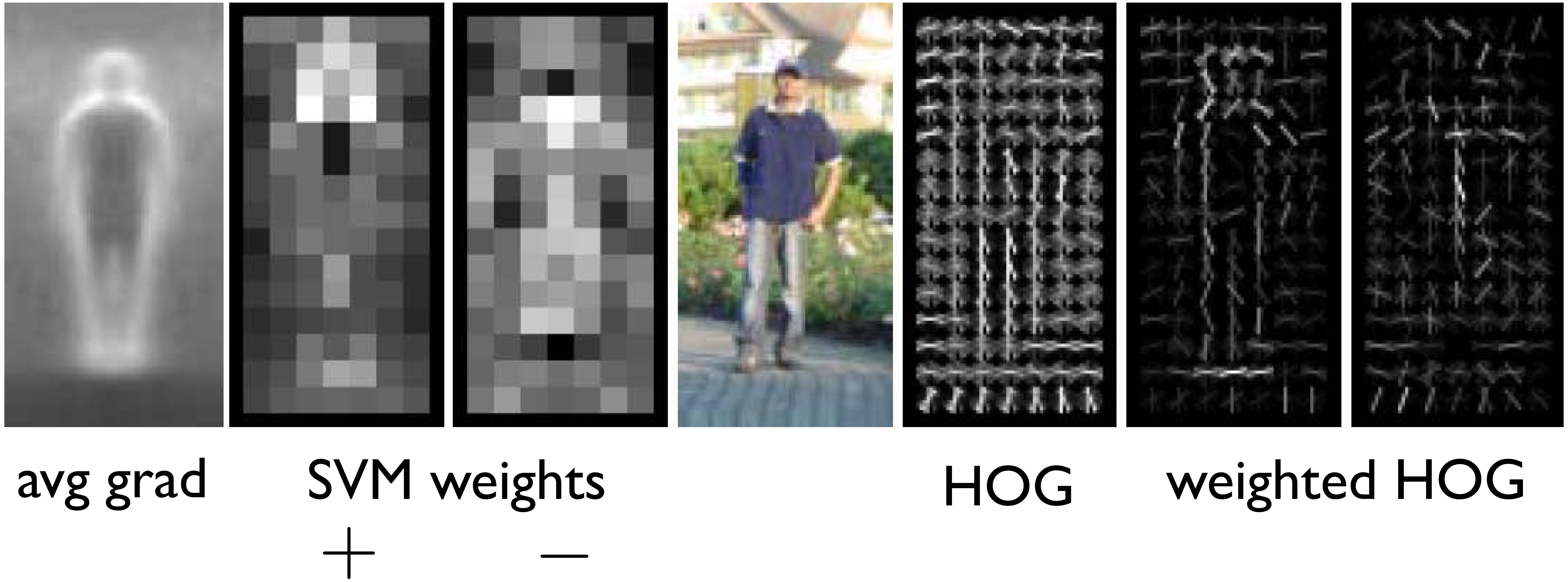
Improved detection algorithms make use of **image features**

These can be **hand coded** or **learned**

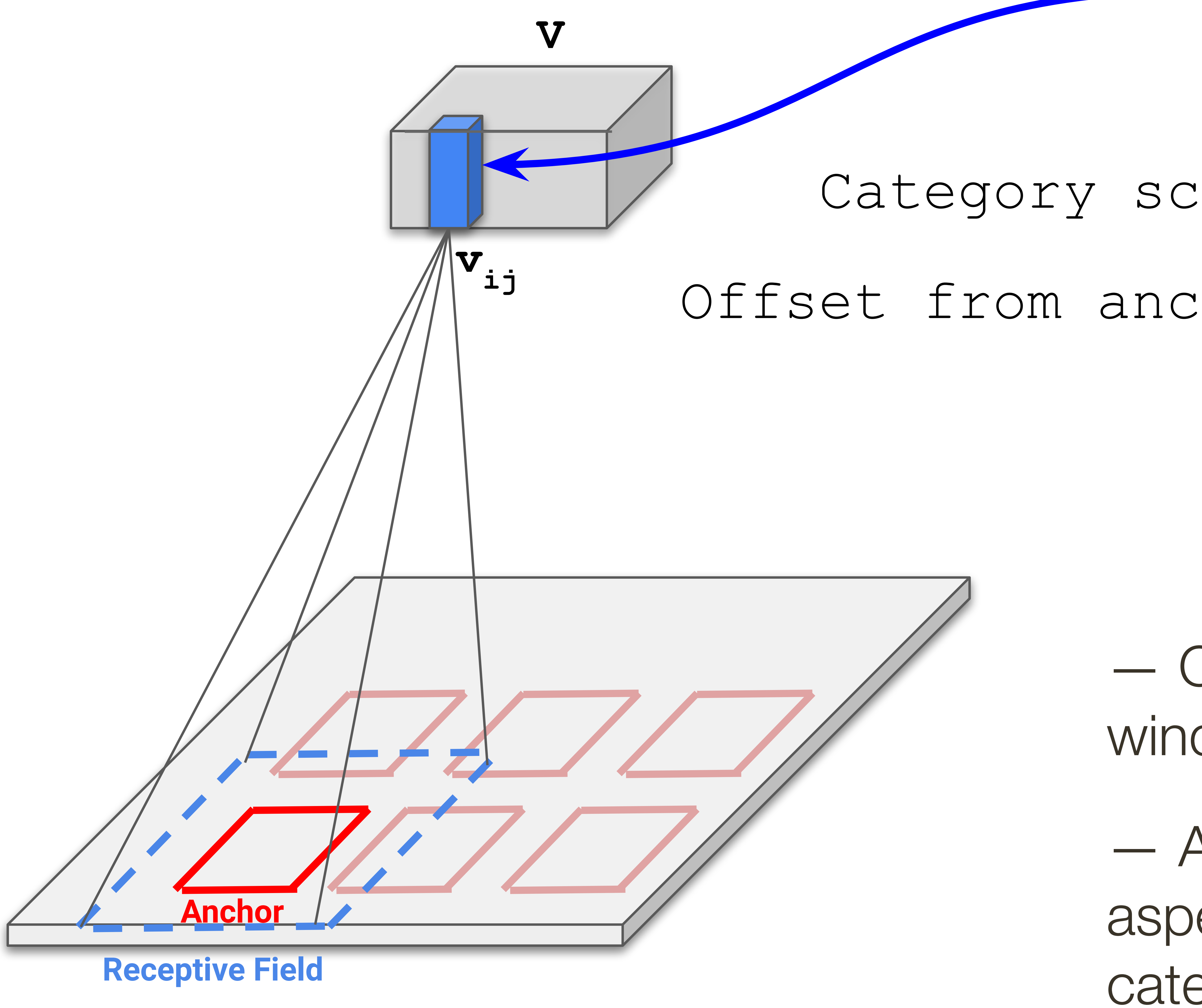
Template Matching with HOG

Template matching can be improved by using better features, e.g., Histograms of Gradients (HOG) [Dalal Triggs 2005]

The authors use a Learning-based approach (Support Vector Machine) to find an optimally weighted template



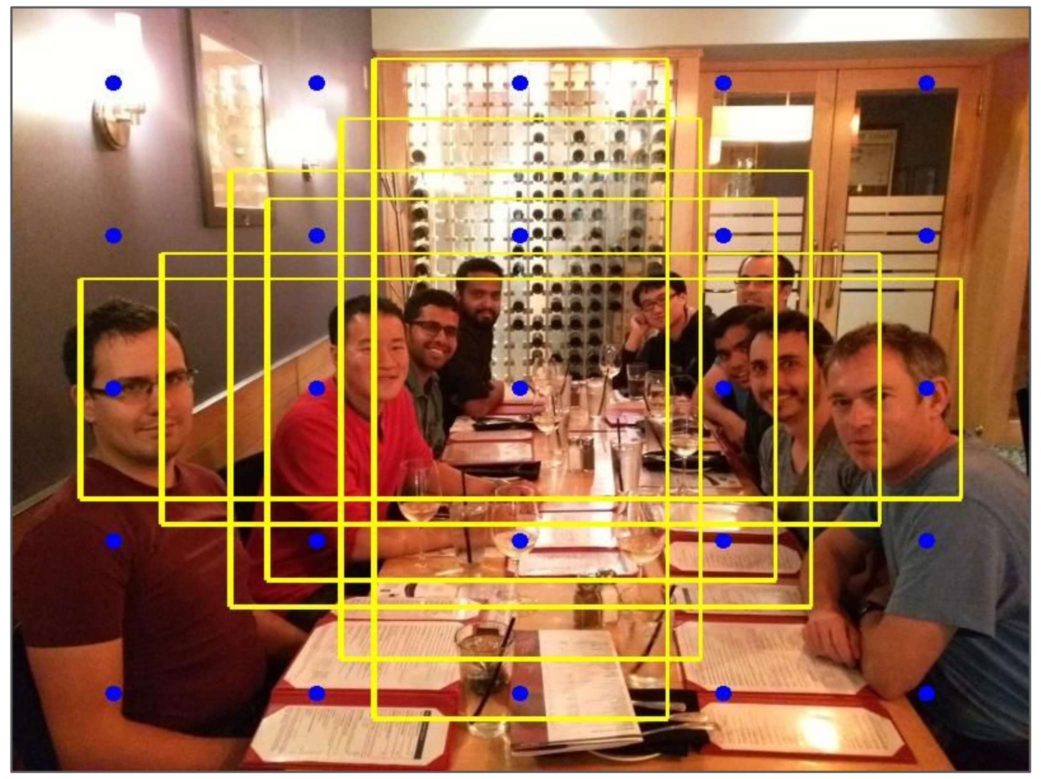
Convnet Object Detection



Think of each feature vector $\mathbf{v}_{ij}$ as corresponding to a sliding window (anchor).

$$\text{Category score} = \text{SoftMax}(W^{cls} \cdot \mathbf{v}_{ij})$$

$$\text{Offset from anchor} = W^{loc} \cdot \mathbf{v}_{ij}$$



— Convnet based object detectors resemble sliding window template matching in feature space

— Architectures typically involve multiple scales and aspect ratios, and regress to a 2D offset in addition to category scores

[Images: Jonathan Huang]

THE UNIVERSITY OF BRITISH COLUMBIA

CPSC 425: Computer Vision

Lecture 9: Edge Detection

(unless otherwise stated slides are taken or adopted from **Bob Woodham, Jim Little** and **Fred Tung**)

Menu for Today

Topics:

- Edge **Detection**
- **Canny** Edge Detector
- Image **Boundaries**

Readings:

- **Today's** Lecture: Szeliski 7.1-7.2, Forsyth & Ponce 5.1 - 5.2

Reminders:

- **Assignment 2**: Scaled Representations, Face Detection and Image Blending
- **Midterm:** Feb 26th 3:30 pm **in class**

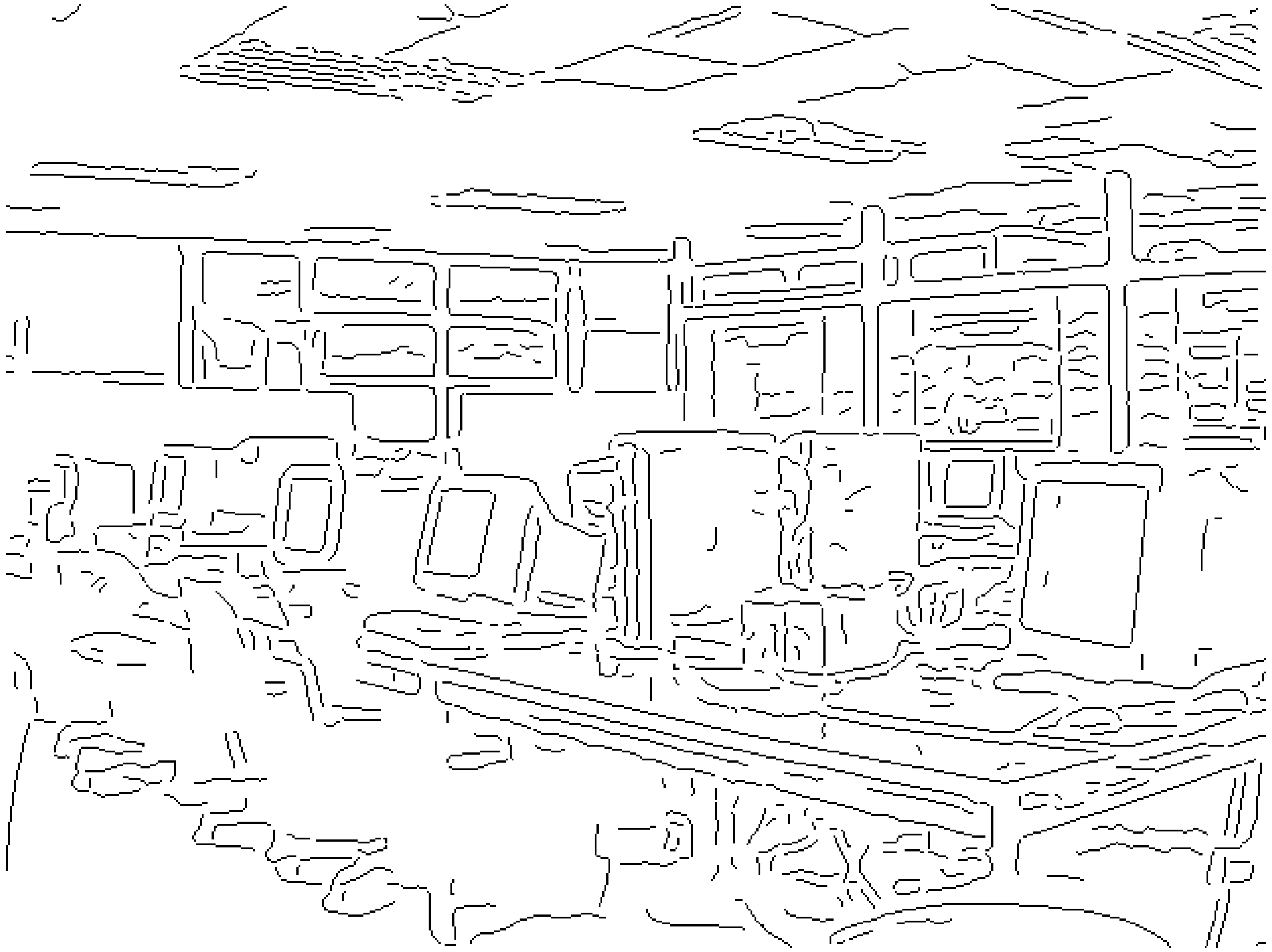
Learning Goal

Understand that gradients are useful

Gradient —> Edges

Edge Detection

One of the first algorithms in Computer Vision



Edge Detection

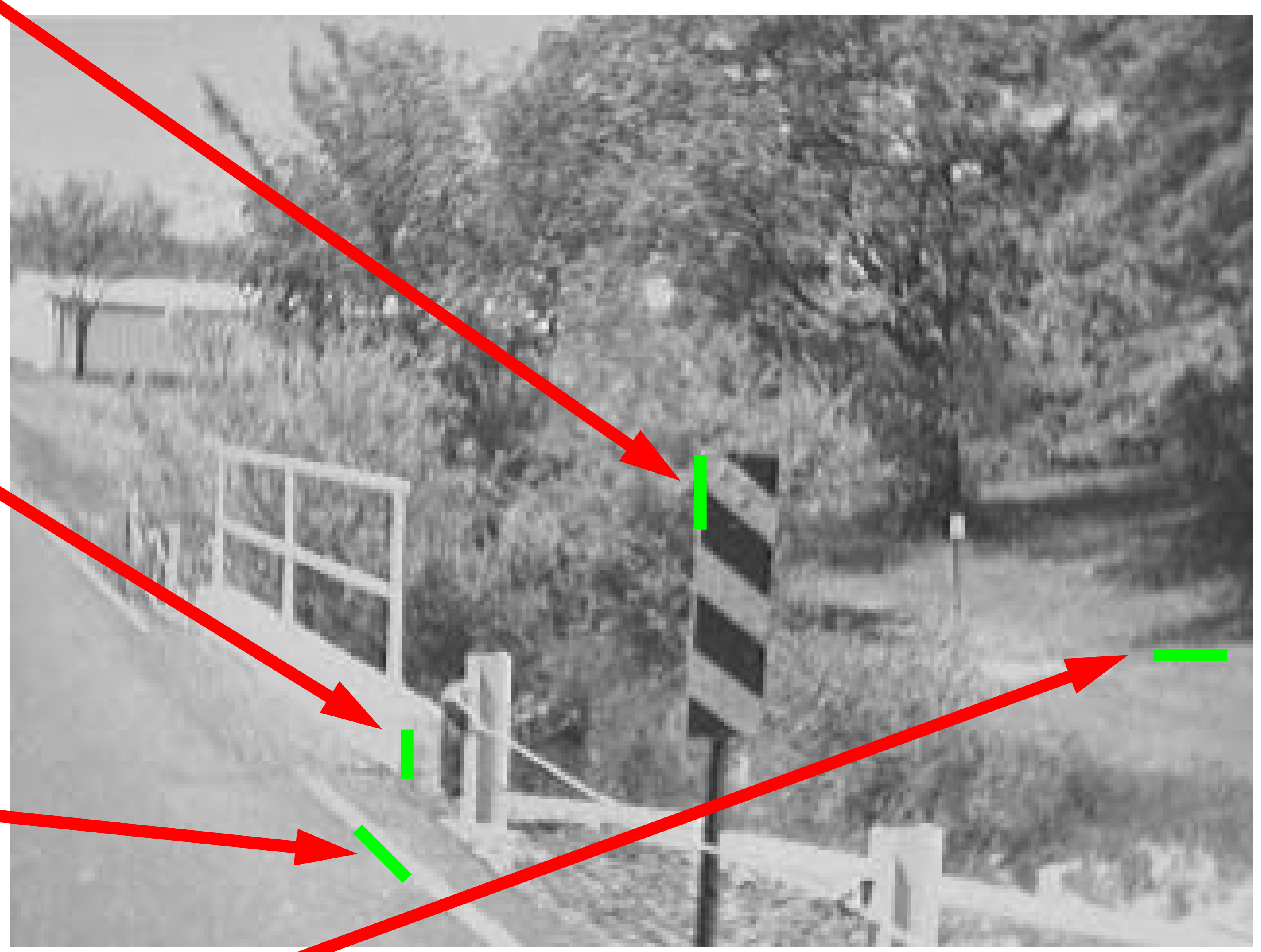
Goal: Identify sudden changes in image intensity

This is where most shape information is encoded

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

What Causes **Edges**?

- Depth discontinuity
- Surface orientation discontinuity
- Reflectance discontinuity (i.e., change in surface material properties)
- Illumination discontinuity (e.g., shadow)



Slide Credit: Christopher Rasmussen

Derivative Definition

9.1

Estimating **Derivatives**

Recall, for a 2D (continuous) function, f(x,y)

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

Differentiation is linear and shift invariant, and therefore can be implemented as a convolution

Estimating **Derivatives**

Recall, for a 2D (continuous) function, f(x,y)

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

Differentiation is linear and shift invariant, and therefore can be implemented as a convolution

A (discrete) approximation is

$$\frac{\partial f}{\partial X} \approx \frac{F(X+1, Y) - F(X, Y)}{\Delta X}$$

Estimating **Derivatives**

Recall, for a 2D (continuous) function, f(x,y)

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

Differentiation is linear and shift invariant, and therefore can be implemented as a convolution

A (discrete) approximation is

$$\frac{\partial f}{\partial X} \approx \frac{F(X+1, Y) - F(X,Y)}{\Delta X}$$

-1	1

Estimating **Derivatives**

A (**discrete**) approximation is

$$\frac{\partial f}{\partial X} \approx \frac{F(X+1,Y) - F(X,Y)}{\Delta X}$$

"**forward** difference" implemented as

correlation

-1	1

convolution

1	-1

from **left**

Estimating **Derivatives**

A (**discrete**) approximation is

$$\frac{\partial f}{\partial X} \approx \frac{F(X+1,Y) - F(X,Y)}{\Delta X}$$

"**forward** difference" implemented as

correlation

-1	1

from **left**

convolution

1	-1

"**backward** difference" implemented as

correlation

-1	1

from **right**

convolution

1	-1

Estimating **Derivatives**

A (**discrete**) approximation is

$$\frac{\partial f}{\partial X} \approx \frac{F(X+1,Y) - F(X,Y)}{\Delta X}$$

"**forward** difference" implemented as

correlation

-1	1

from **left**

convolution

1	-1

"**backward** difference" implemented as

correlation

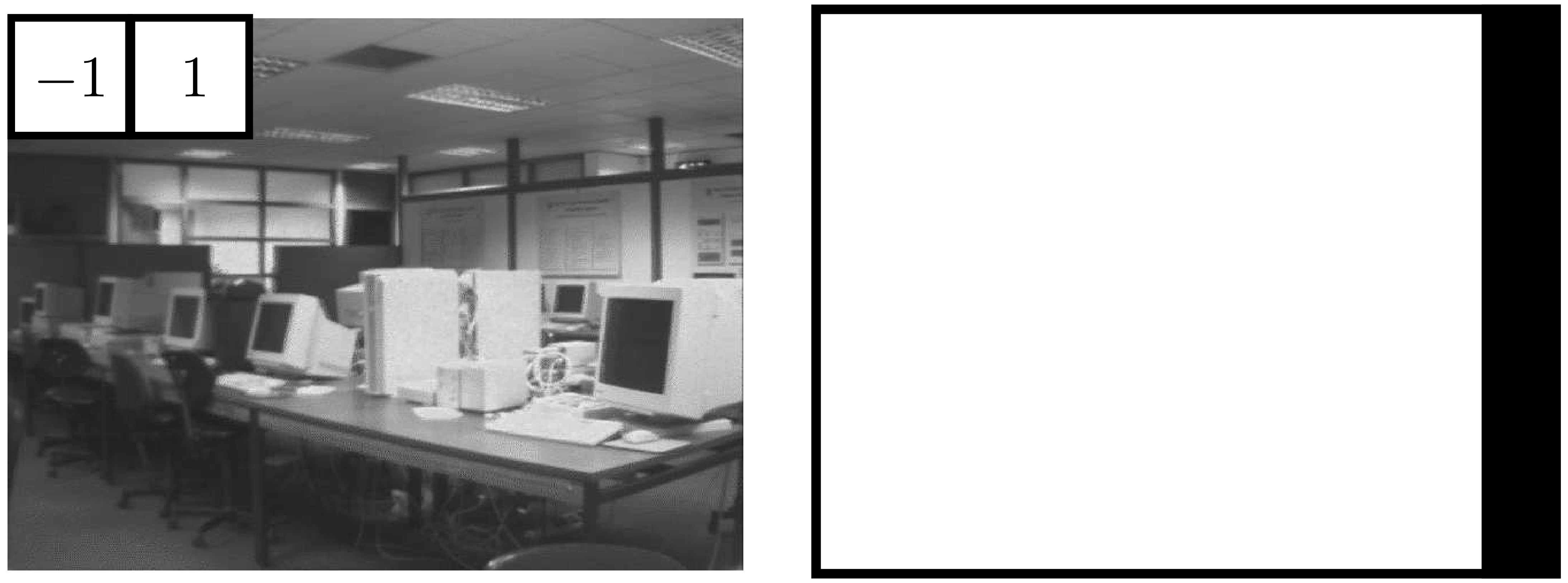
-1	1

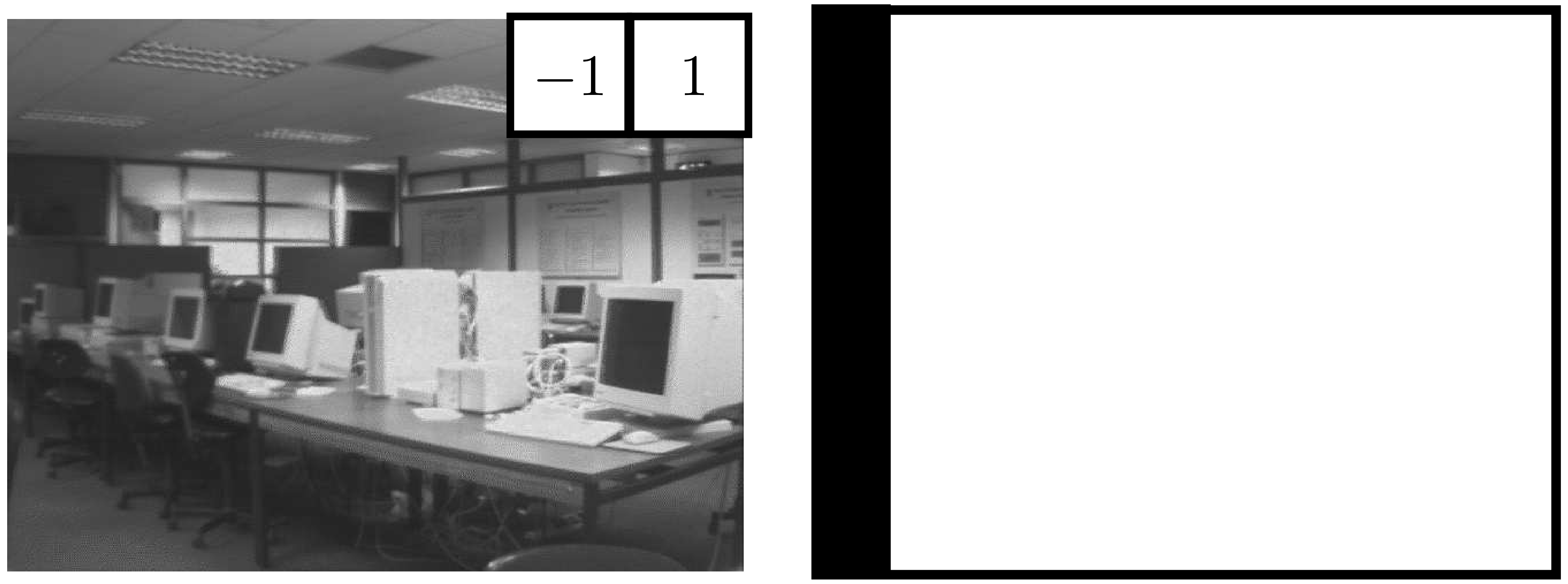
from **right**

convolution

1	-1

Estimating **Derivatives**





“**forward** difference” implemented as

correlation

-1	1

from **left**

“**backward** difference” implemented as

correlation

-1	1

from **right**

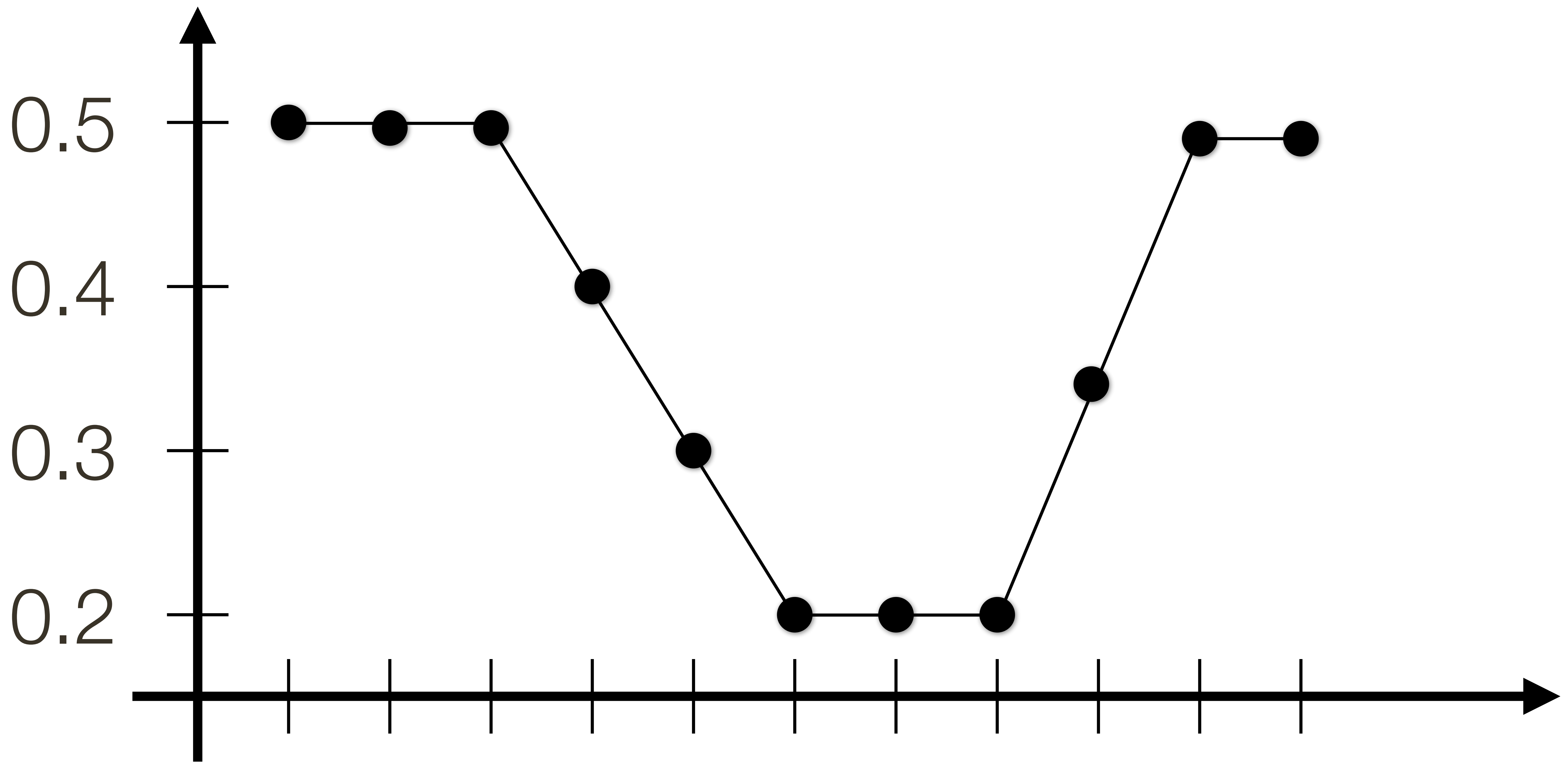
Estimating **Derivatives**

A similar definition (and approximation) holds for $\frac{\partial f}{\partial y}$

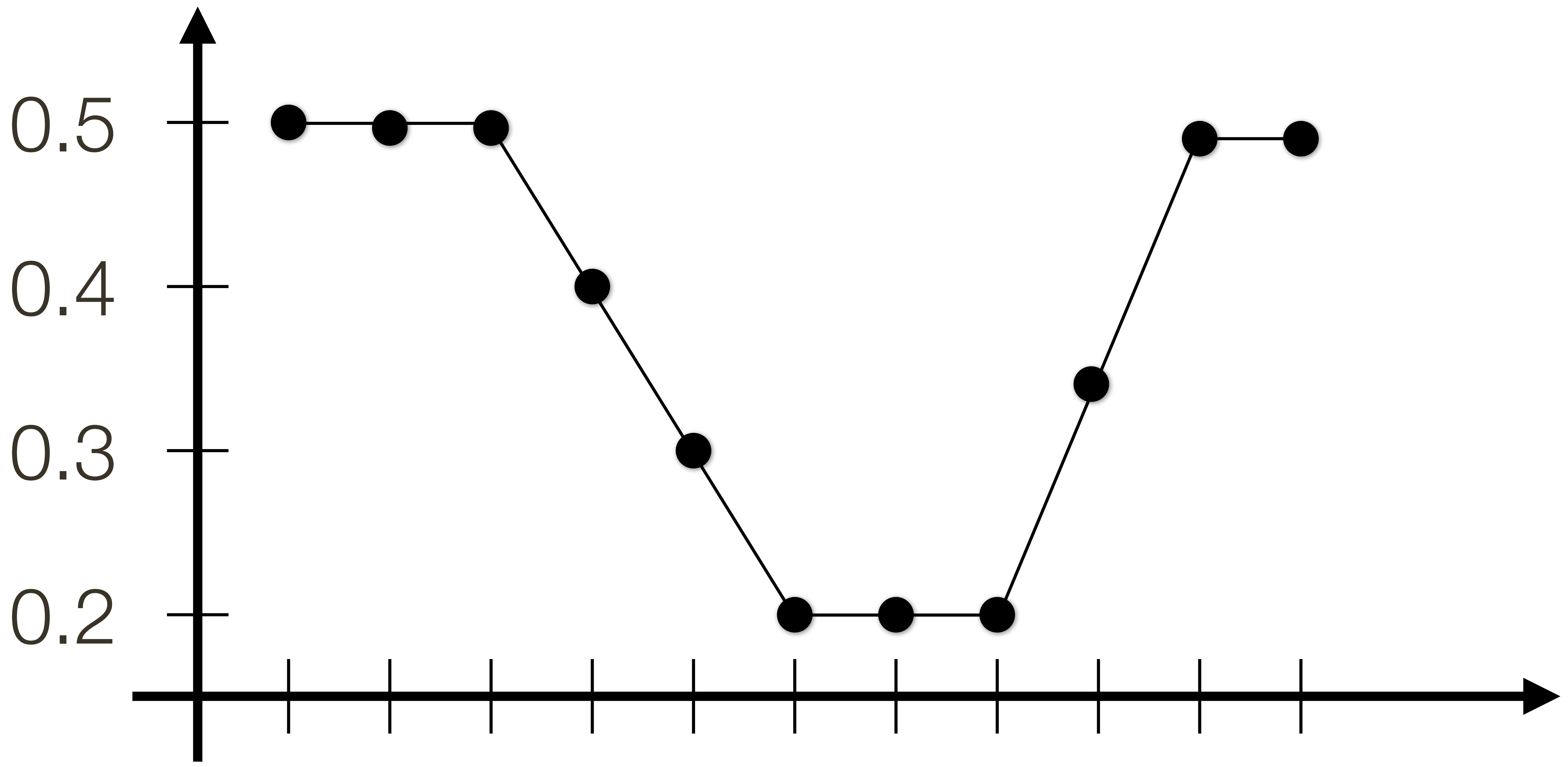
Image **noise** tends to result in pixels not looking exactly like their neighbours, so simple “finite differences” are sensitive to noise.

The usual way to deal with this problem is to **smooth** the image prior to derivative estimation.

Example 1D

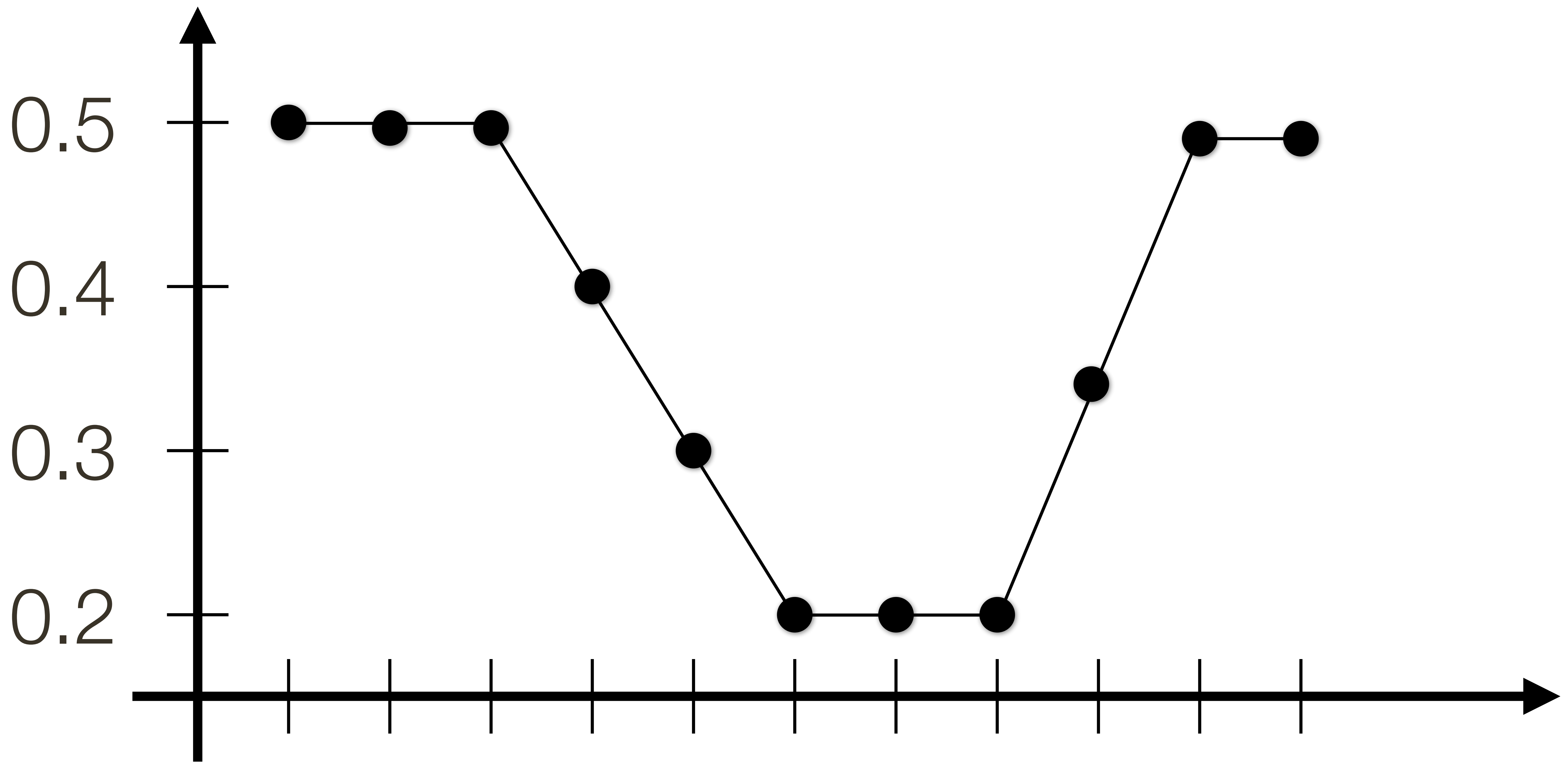


Example 1D



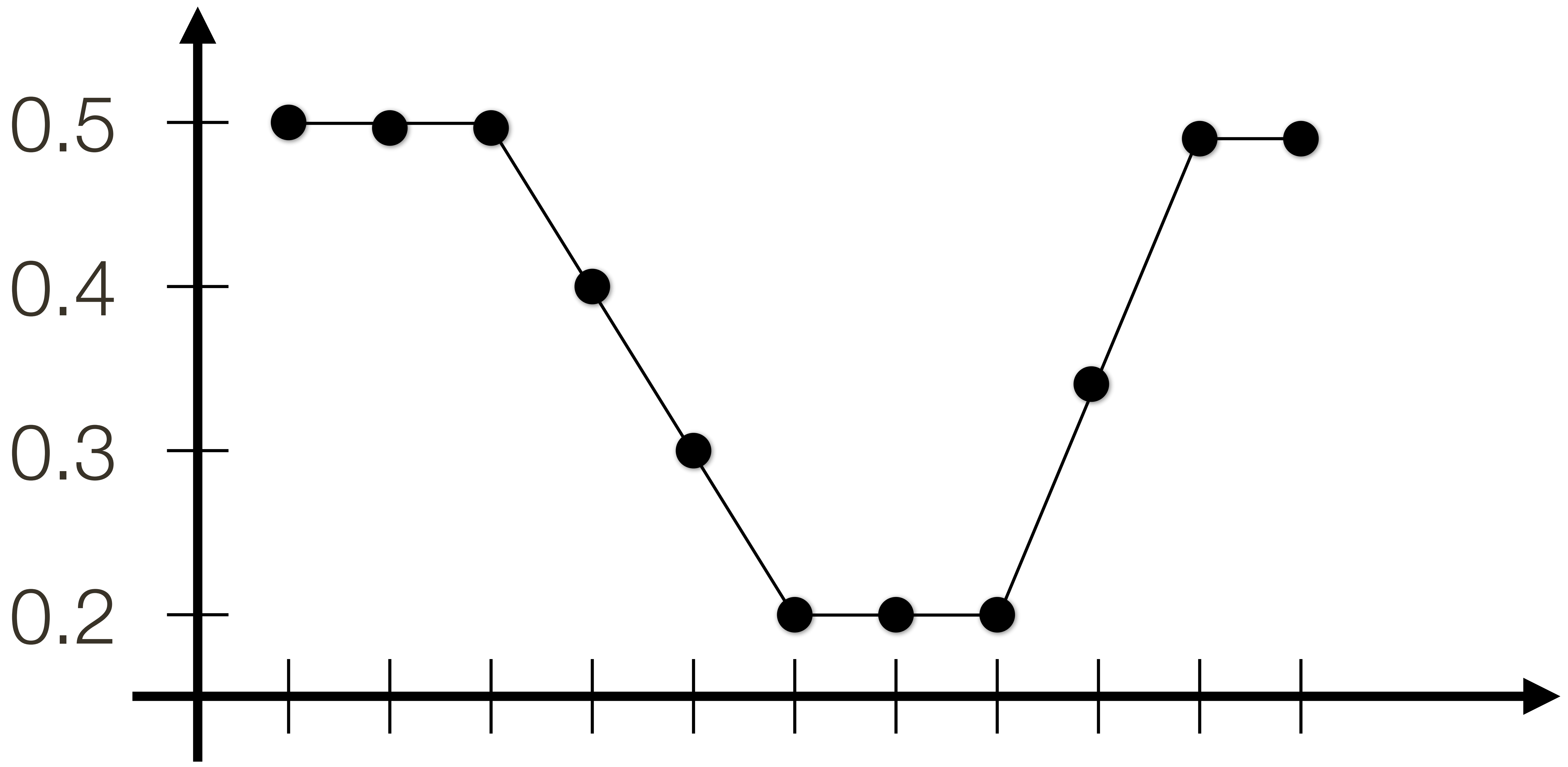
Signal	0.5	0.5	0.5	0.4	0.3	0.2	0.2	0.2	0.35	0.5	0.5

Example 1D



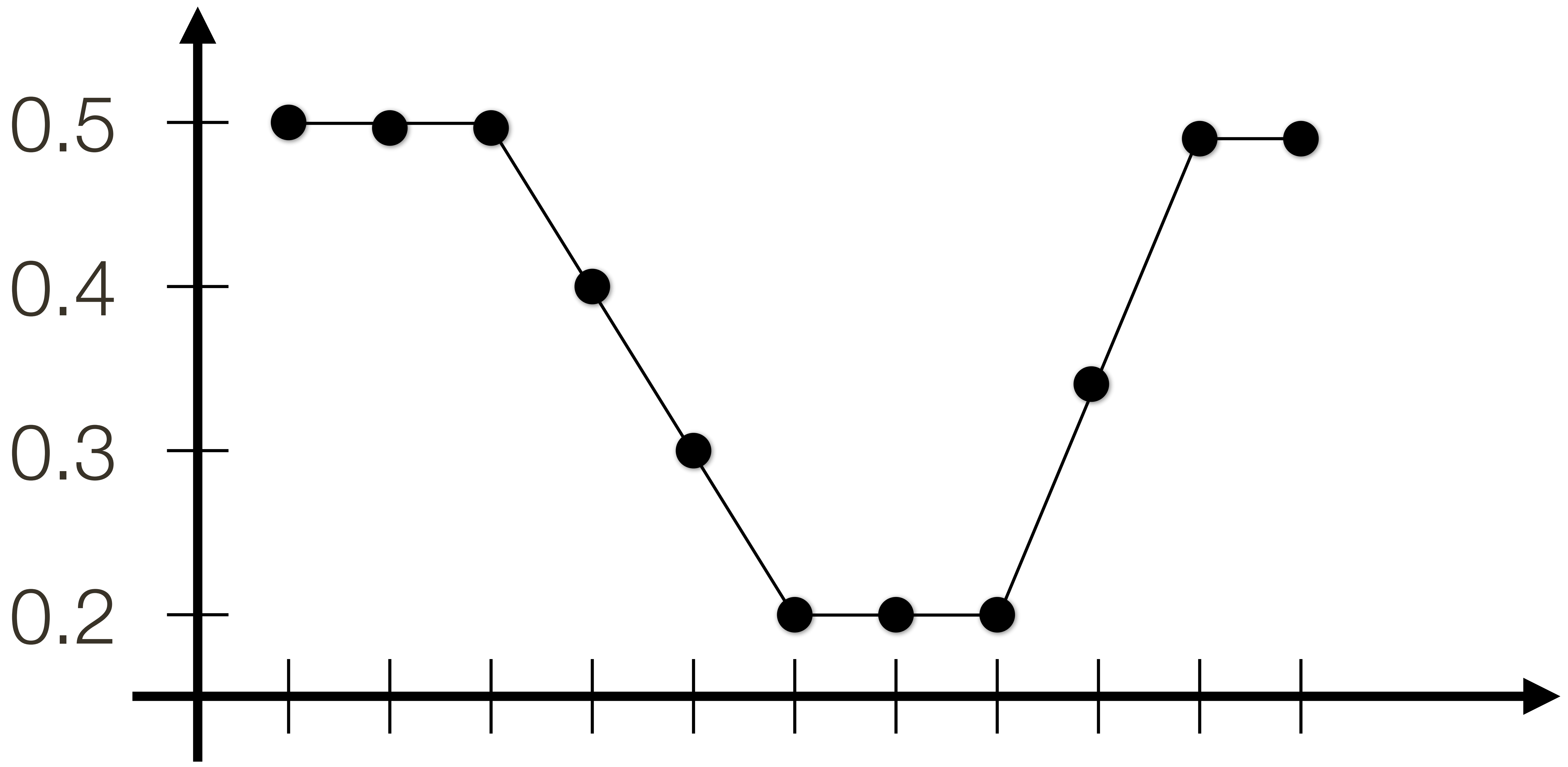
Signal	0.5	0.5	0.5	0.4	0.3	0.2	0.2	0.2	0.35	0.5	0.5
Derivative											

Example 1D



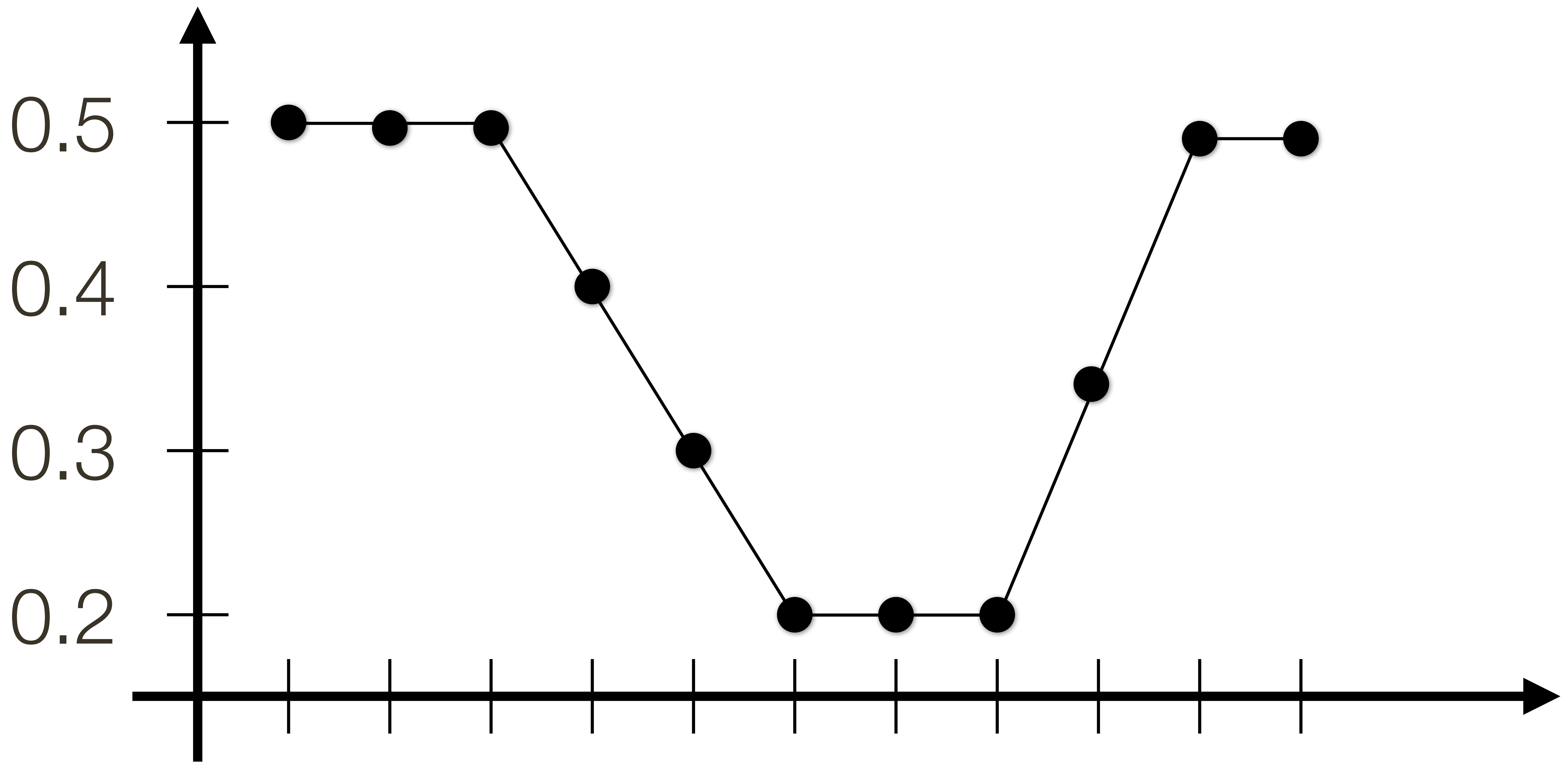
Signal	0.5	0.5	0.5	0.4	0.3	0.2	0.2	0.2	0.35	0.5	0.5
Derivative	0.0										

Example 1D



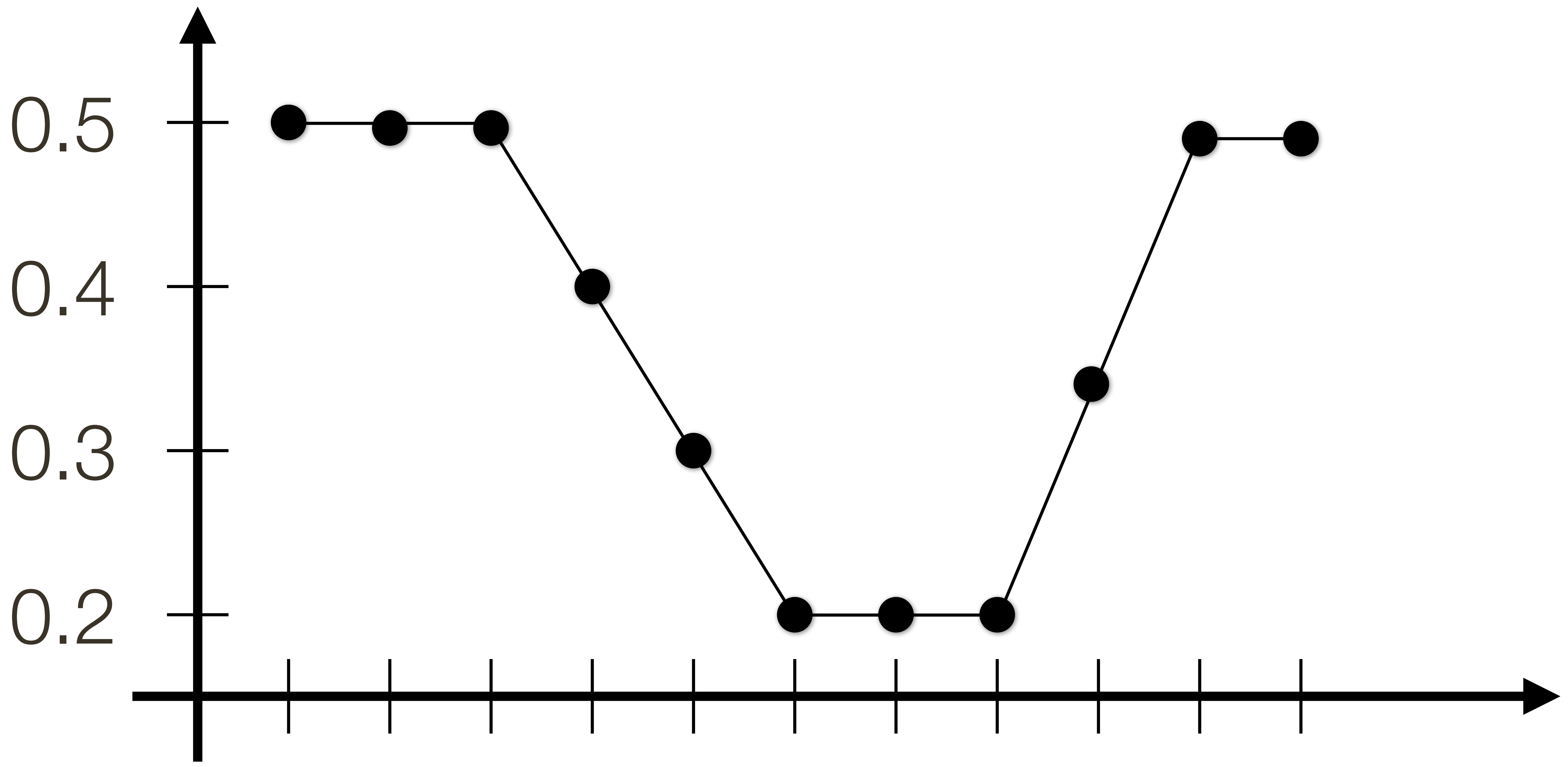
Signal	0.5	0.5	0.5	0.4	0.3	0.2	0.2	0.2	0.35	0.5	0.5
Derivative	0.0										

Example 1D



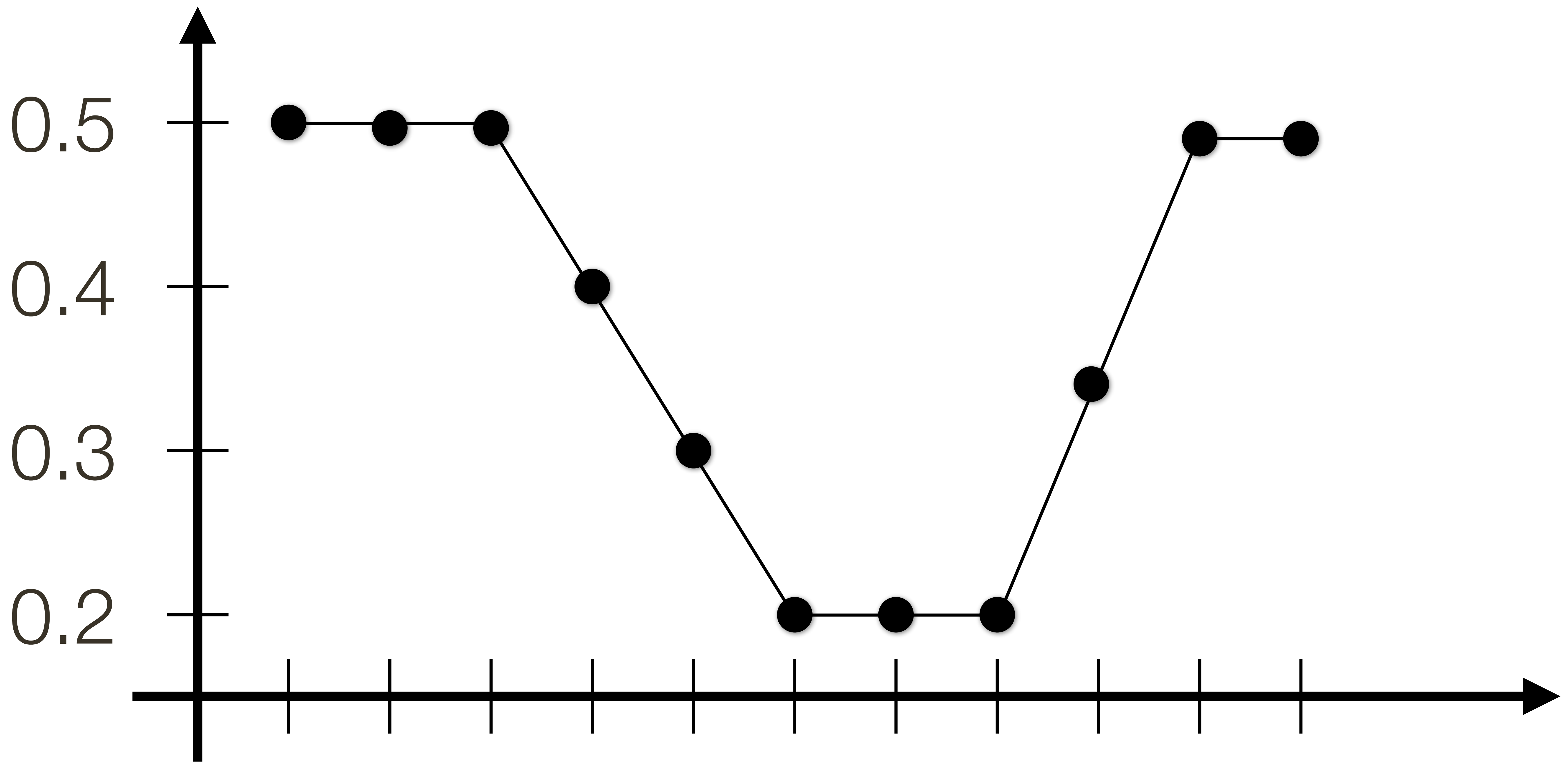
Signal	0.5	0.5	0.5	0.4	0.3	0.2	0.2	0.2	0.35	0.5	0.5
Derivative	0.0	0.0									

Example 1D



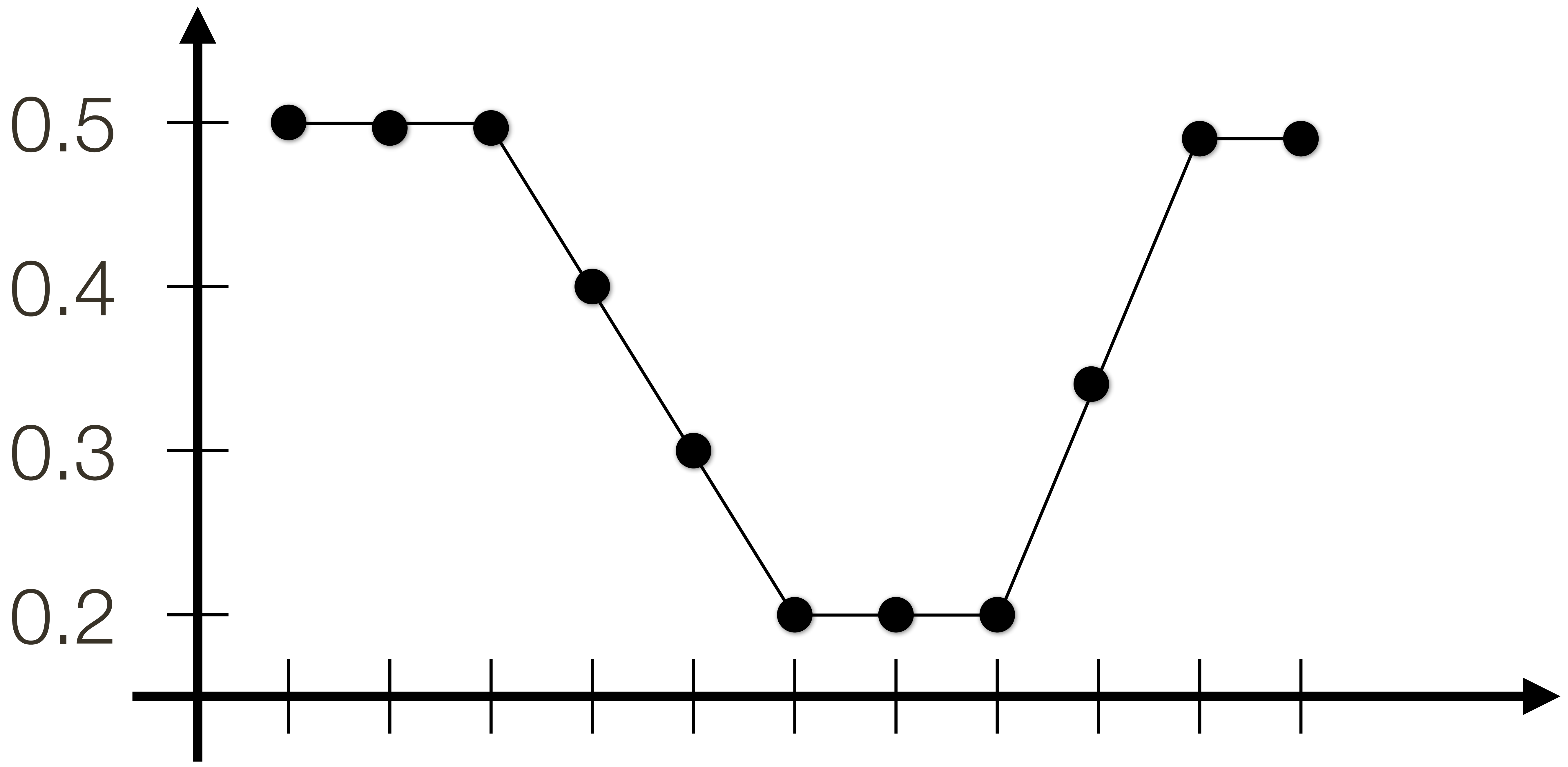
Signal	0.5	0.5	0.5	0.4	0.3	0.2	0.2	0.2	0.35	0.5	0.5
Derivative	0.0	0.0									

Example 1D



Signal	0.5	0.5	0.5	0.4	0.3	0.2	0.2	0.2	0.35	0.5	0.5
Derivative	0.0	0.0	-0.1								

Example 1D

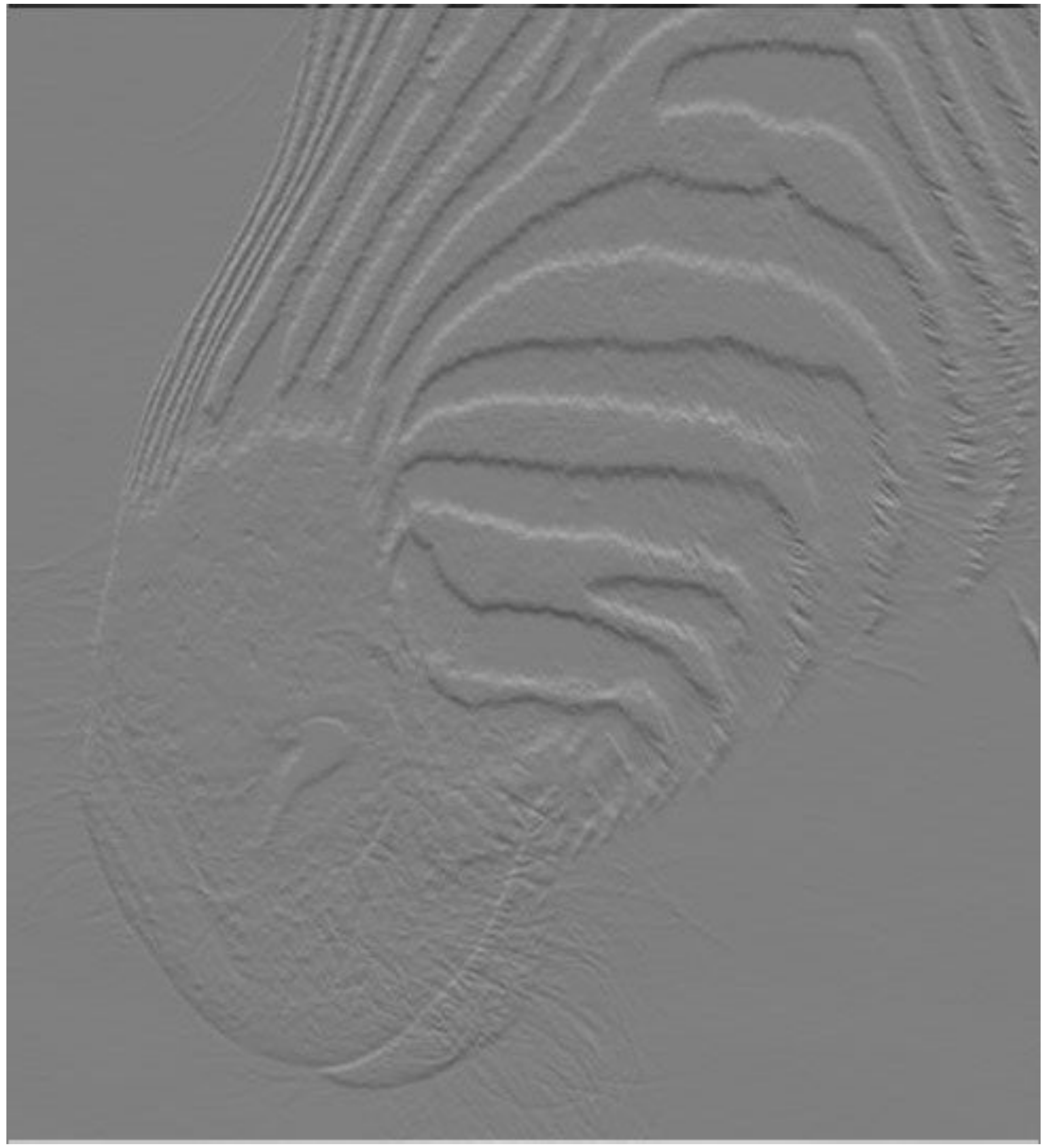


Signal	0.5	0.5	0.5	0.4	0.3	0.2	0.2	0.2	0.35	0.5	0.5
Derivative	0.0	0.0	-0.1	-0.1	-0.1	0.0	0.0	0.15	0.15	0.0	X

Estimating **Derivatives**

Derivative in Y (i.e., vertical) direction

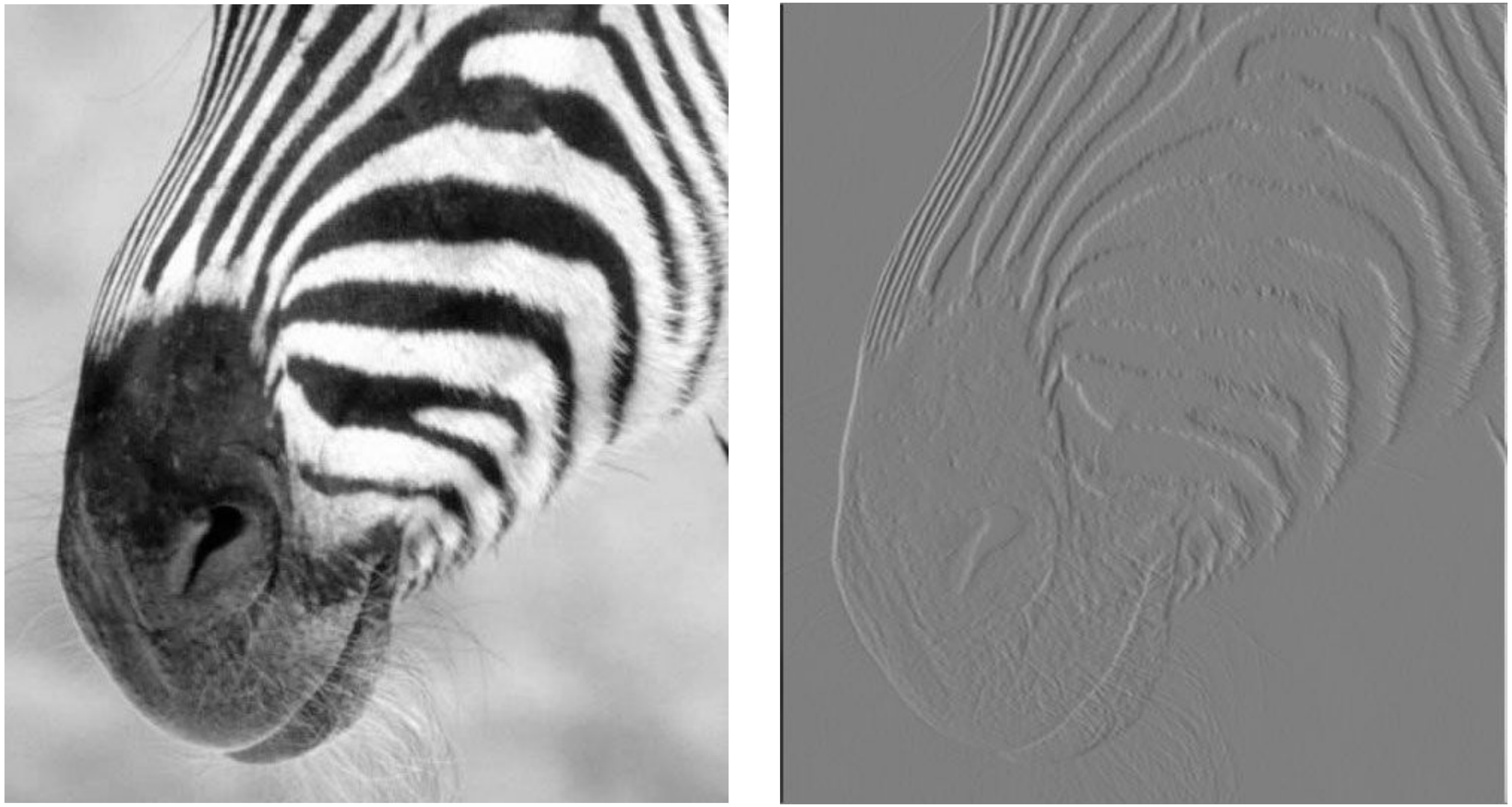
(**Note:** visualized by adding 0.5/128)



Forsyth & Ponce (1st ed.) Figure 7.4

Estimating **Derivatives**

Derivative in X (i.e., horizontal) direction (**Note:** visualized by adding 0.5/128)



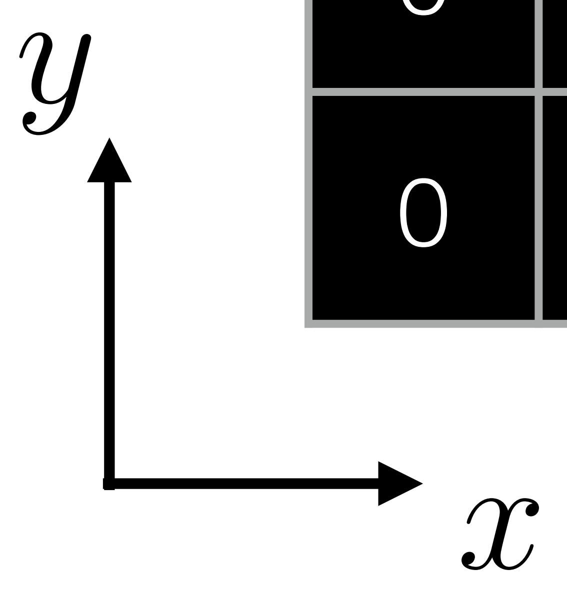
Forsyth & Ponce (1st ed.) Figure 7.4

Example: 2D Derivatives

Use the “first forward difference" to compute the image derivatives in X and Y

9.2 Compute two arrays, one of $\frac{\partial f}{\partial x}$ values and one of $\frac{\partial f}{\partial y}$ values

1	1	0.6	0.3	0	0
1	1	0.6	0.3	0	0
0	0	0	0	0	0
0	0	0	0	0	0



Estimating **Derivatives**

Q: Why should the weights of a filter used for differentiation sum to 0?

Estimating **Derivatives**

Q: Why should the weights of a filter used for differentiation sum to 0?

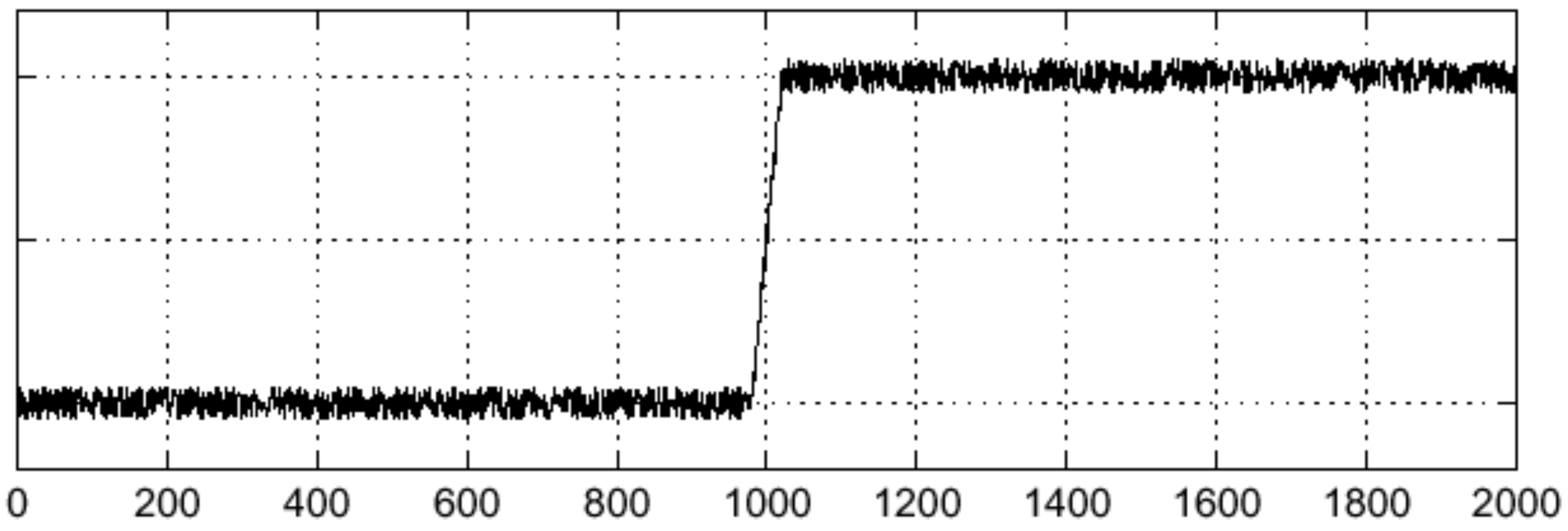
e.g. a constant image, $I(X,Y) = k$ has derivative = 0. Therefore, the weights of any filter used for differentiation need to sum to 0.

$$\sum_{i=1}^{N} f_i \cdot k = k \sum_{i=1}^{N} f_i = 0 \implies \sum_{i=1}^{N} f_i = 0$$

Edge Detection: 1D **Example**

Lets consider a row of pixels in an image:

$I(X, 245)$

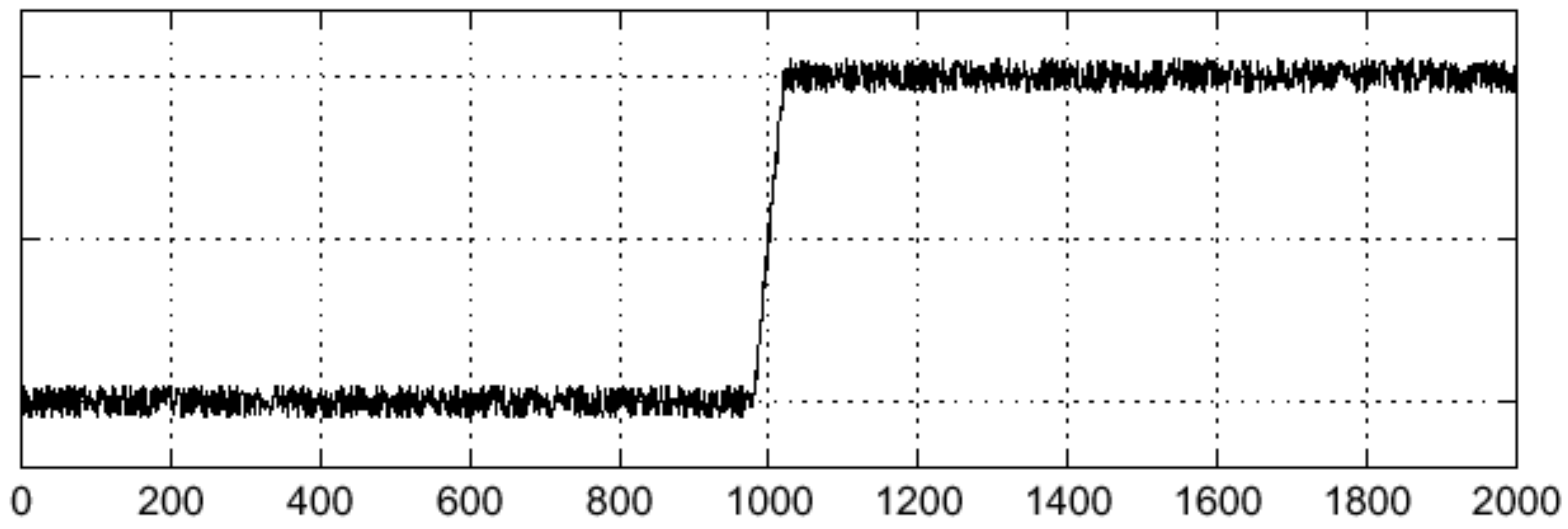


Where is the edge?

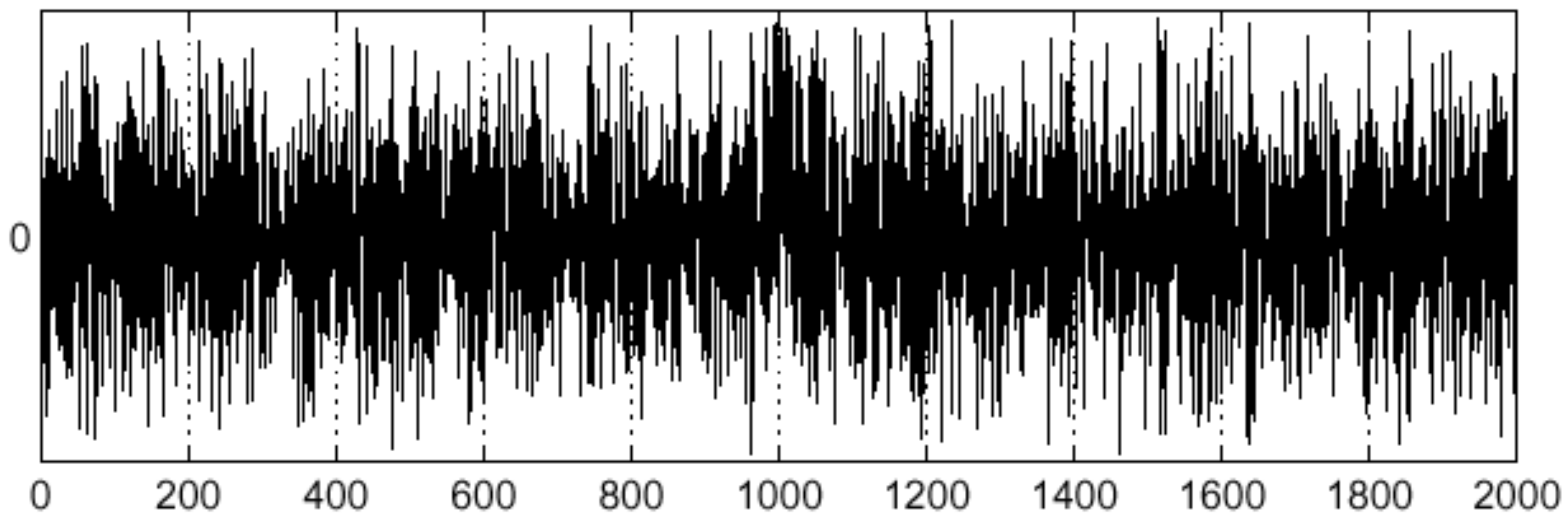
Edge Detection: 1D **Example**

Lets consider a row of pixels in an image:

$I(X, 245)$



$\frac{\partial I(X, 245)}{\partial x}$



Where is the edge?

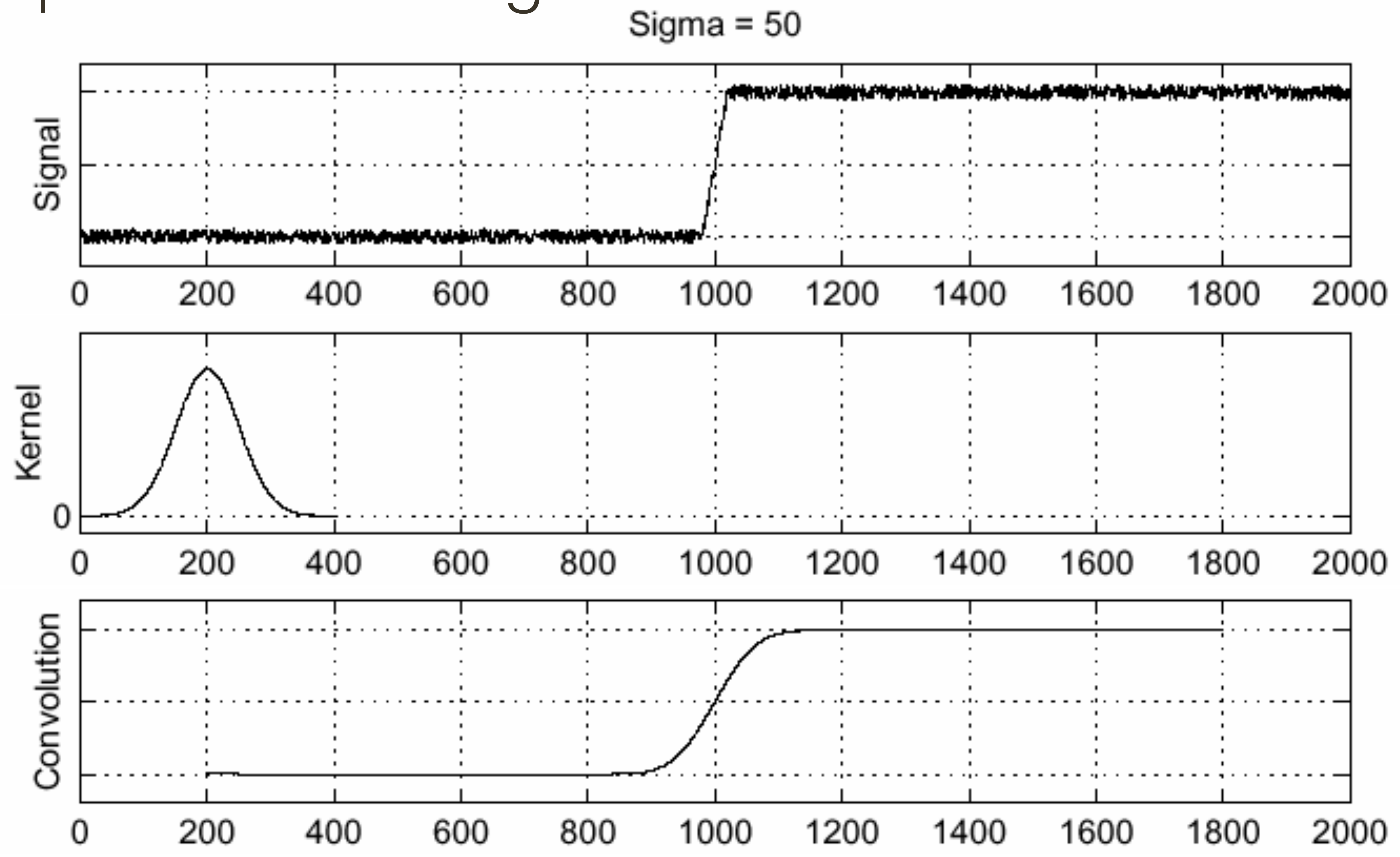
1D **Example**: Smoothing + Derivative

Lets consider a row of pixels in an image:

$I(X, 245)$

G

$G \otimes I(X, Y)$



1D **Example**: Smoothing + Derivative

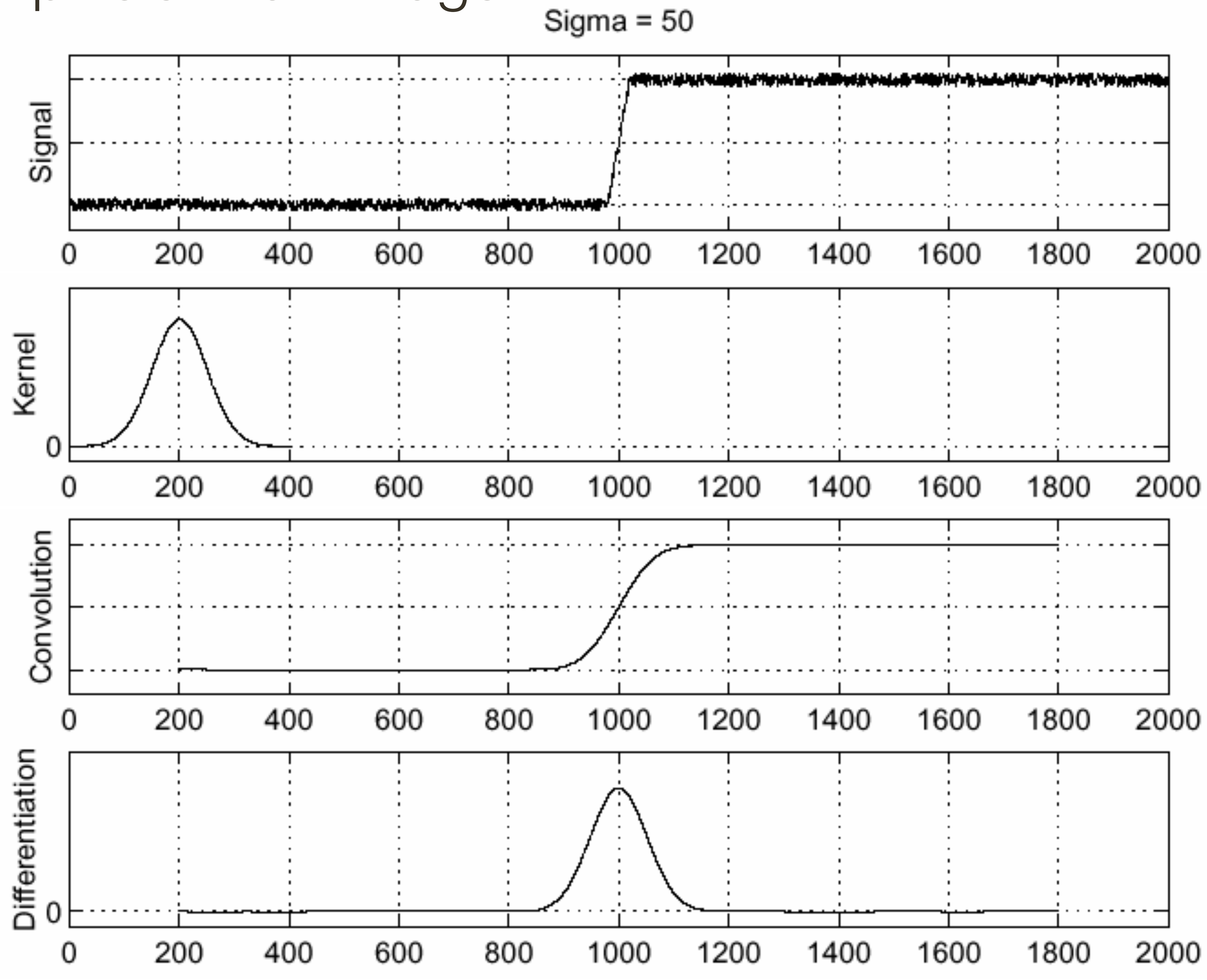
Lets consider a row of pixels in an image:

$I(X, 245)$

G

$G \otimes I(X, Y)$

$$\frac{\partial G \otimes I(X,Y)}{\partial x}$$



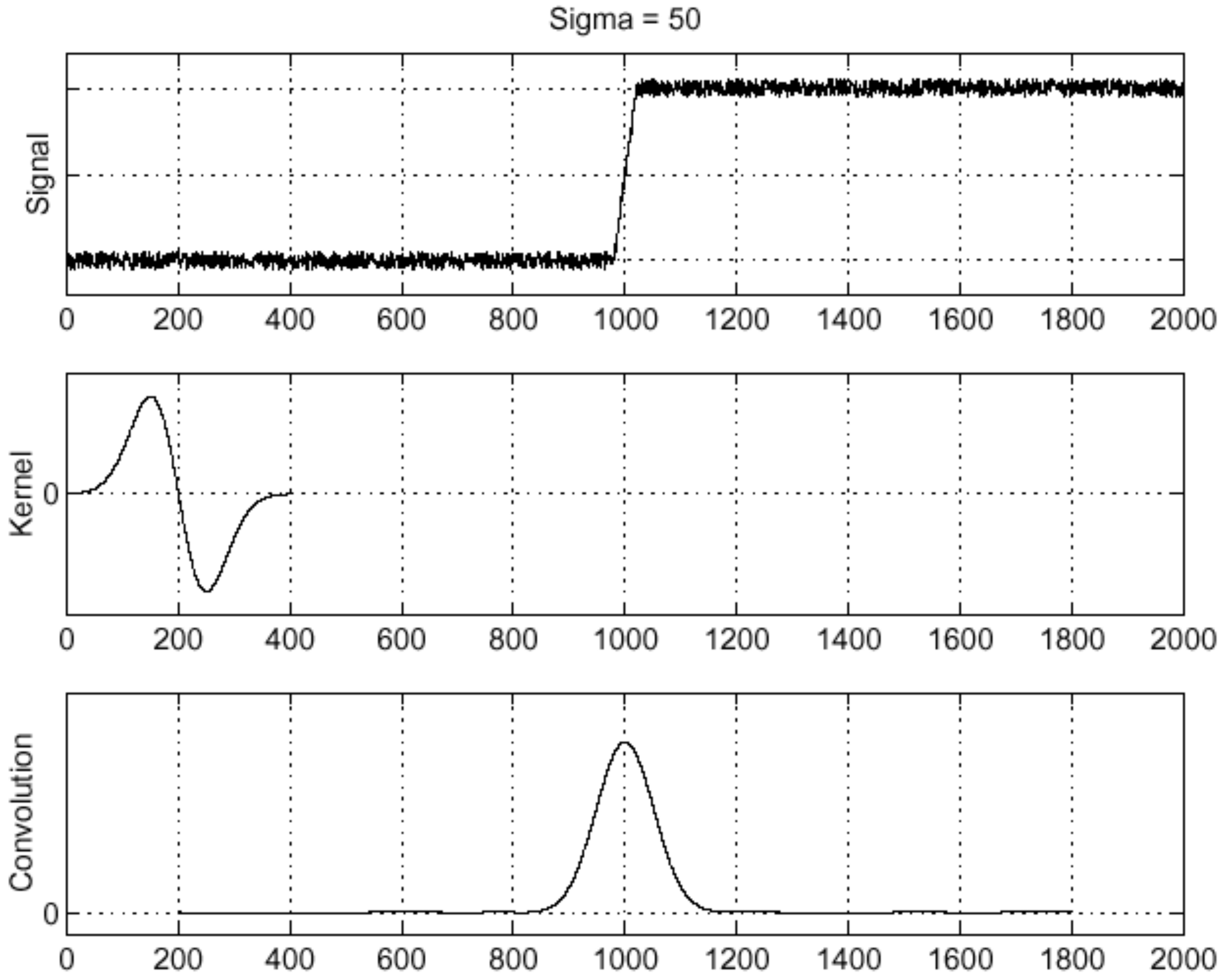
1D **Example**: Smoothing + Derivative

Lets consider a row of pixels in an image:

$$I(X, 245)$$

$$\frac{\partial G}{\partial x}$$

$$\frac{\partial G}{\partial x} \otimes I(X, Y)$$



Smoothing and Differentiation

Edge: a location with high gradient (derivative)

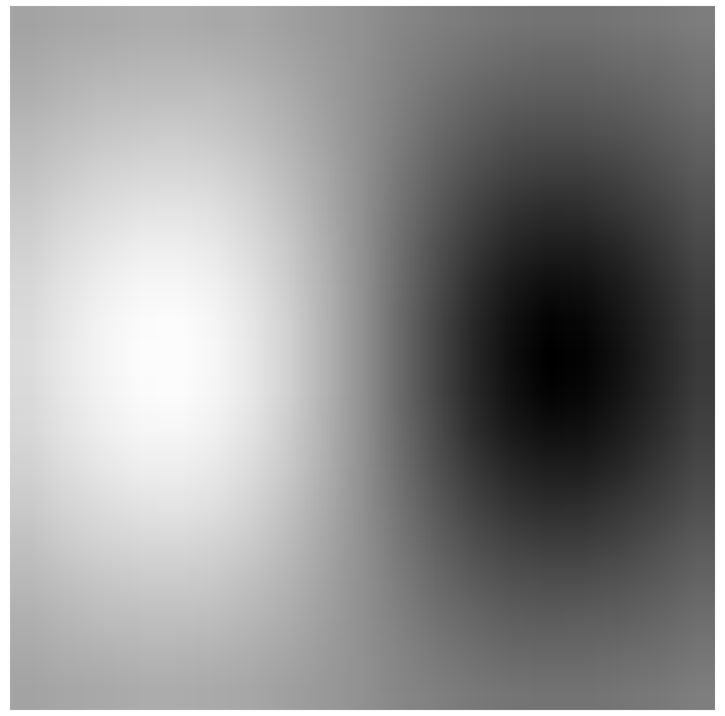
Need smoothing to reduce noise prior to taking derivative

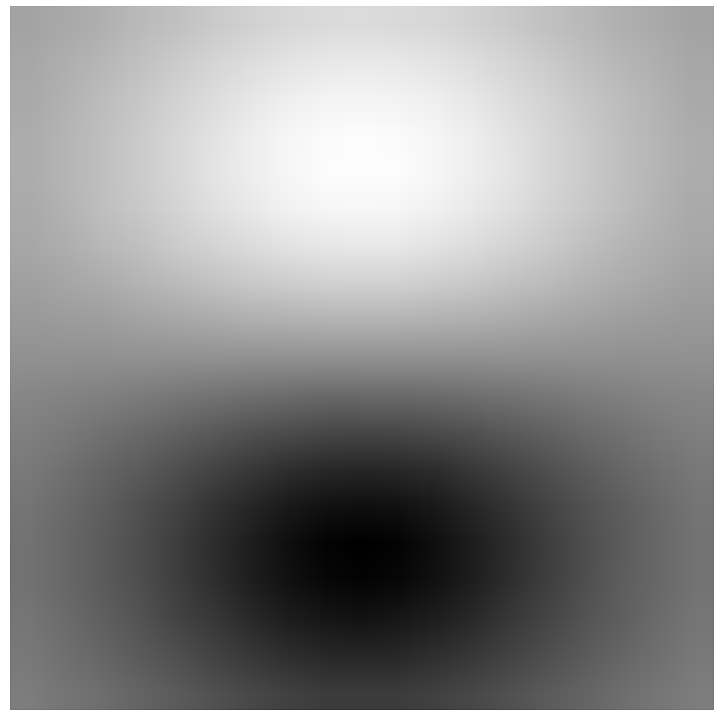
Need two derivatives, in x and y direction

We can use **derivative of Gaussian** filters
— because differentiation is convolution, and
— convolution is associative

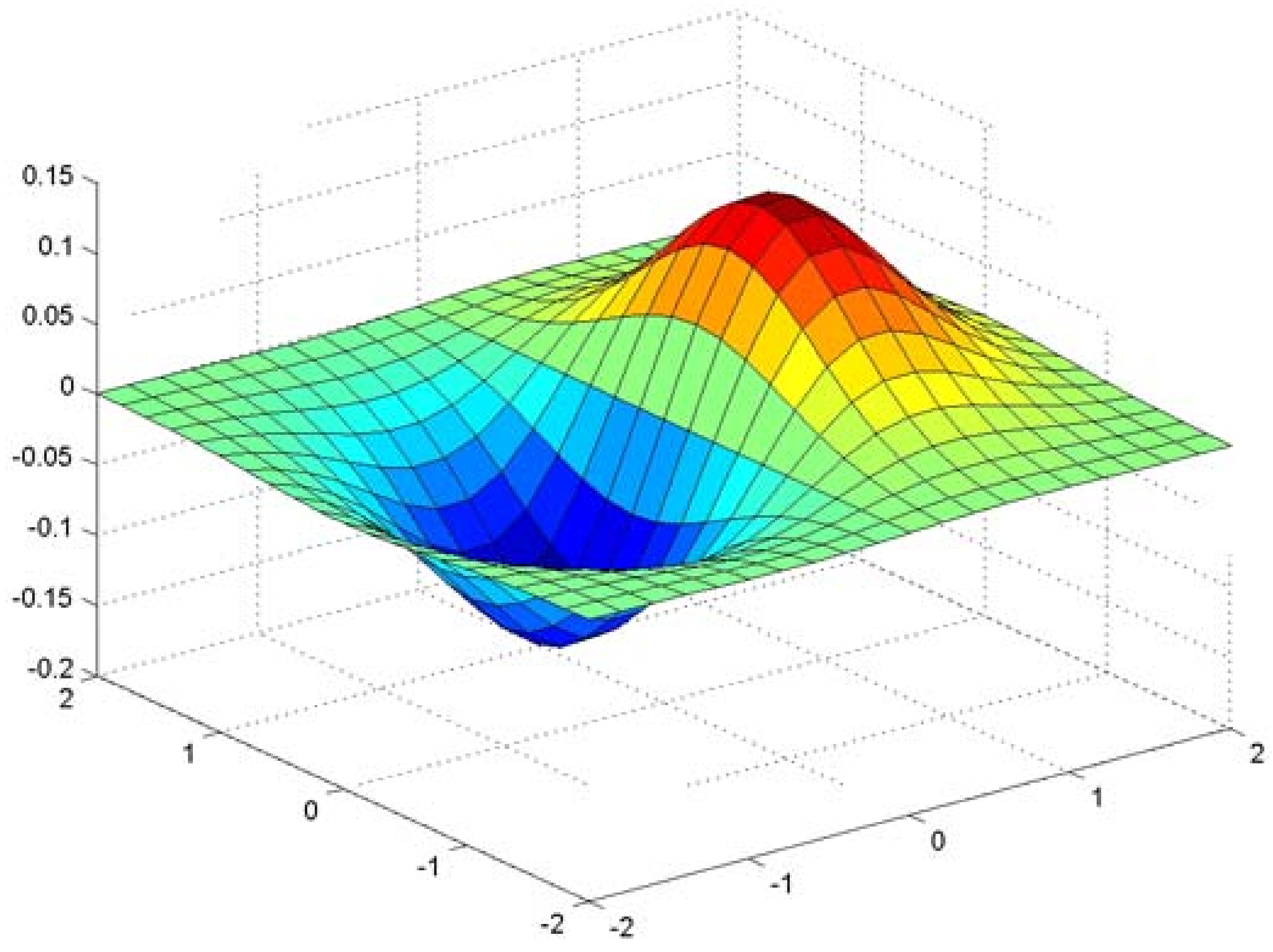
Let $\otimes$ denote convolution

$$D \otimes (G \otimes I(X,Y)) = (D \otimes G) \otimes I(X,Y)$$

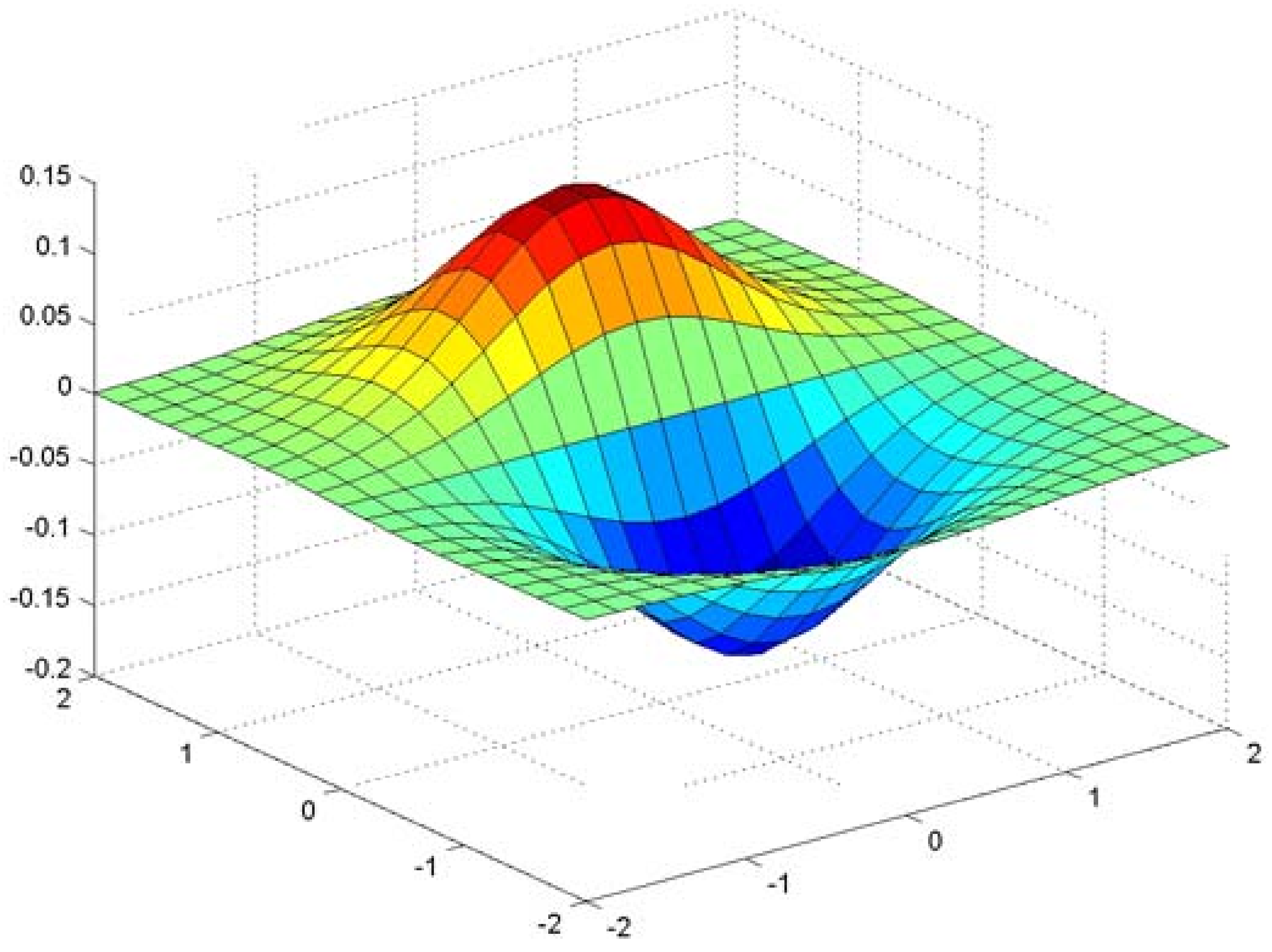




Partial **Derivatives** of **Gaussian**



$$\frac{\partial}{\partial x} G_\sigma$$



$$\frac{\partial}{\partial y} G_\sigma$$

Slide Credit: Christopher Rasmussen

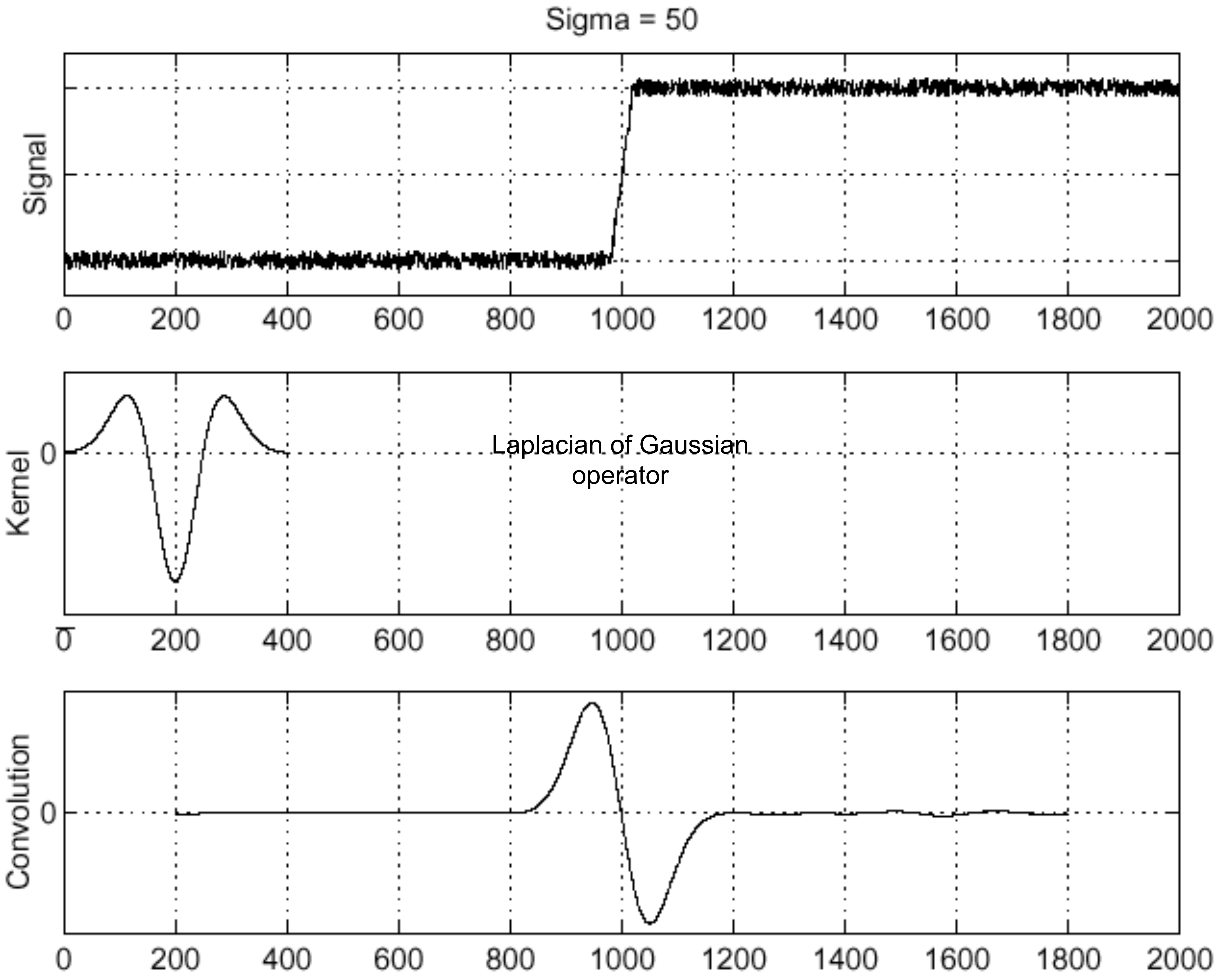
1D **Example**: Continued

Lets consider a row of pixels in an image:

$$I(X, 245)$$

$$\nabla^2 G$$

$$\nabla^2 G \otimes I(X, Y)$$



Zero-crossings of bottom graph

Derivative Approximations: Forward, Backward, Centred

9.3

Sobel Edge Detector

1. Use **central differencing** to compute gradient image (instead of first forward differencing). This is more accurate.

$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

2. **Threshold** to obtain edges

Original Image

Sobel Gradient

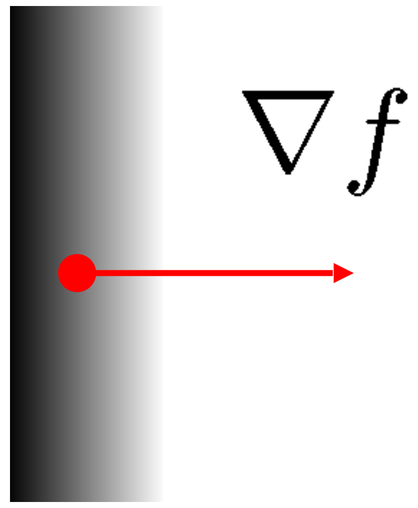
Sobel Edges

2D Image **Gradient**

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

2D Image **Gradient**

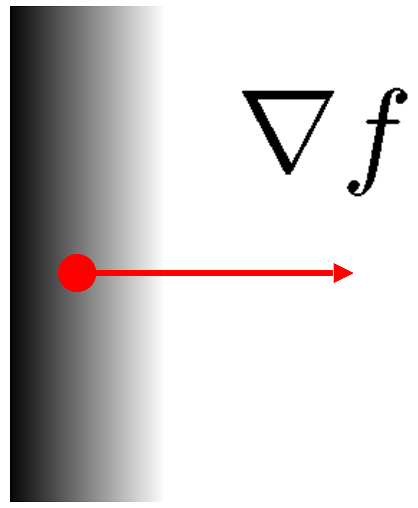
The gradient of an image: $\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$



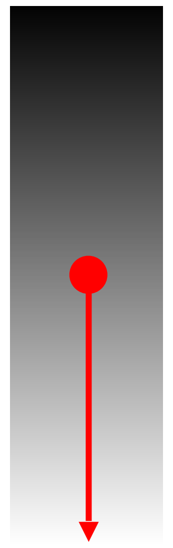
$\nabla f = \left[\frac{\partial f}{\partial x}, 0\right]$

2D Image **Gradient**

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



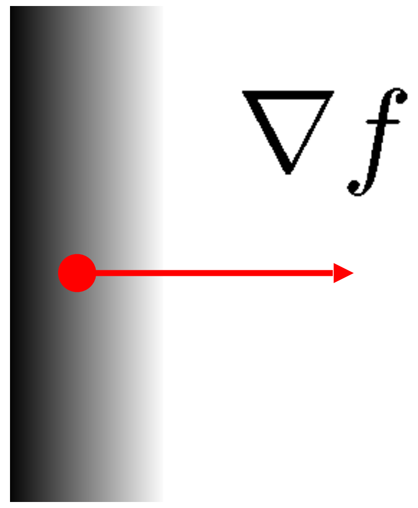
$$\nabla f = \left[\frac{\partial f}{\partial x}, 0\right]$$



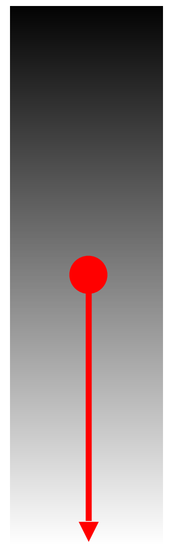
$$\nabla f = \left[0, \frac{\partial f}{\partial y}\right]$$

2D Image **Gradient**

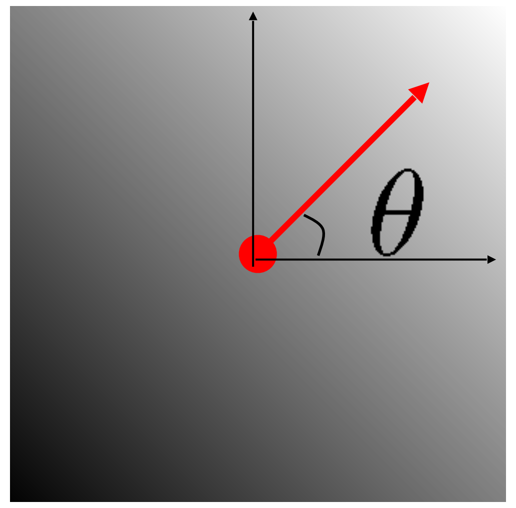
The gradient of an image: $\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$



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



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

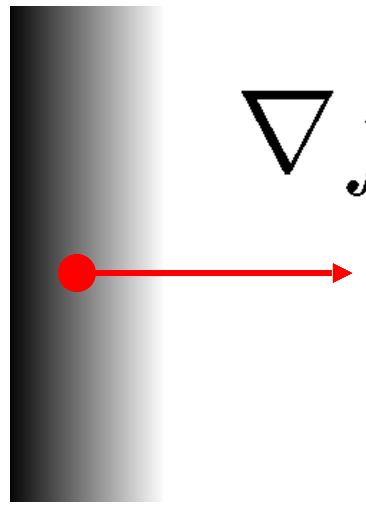


$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$

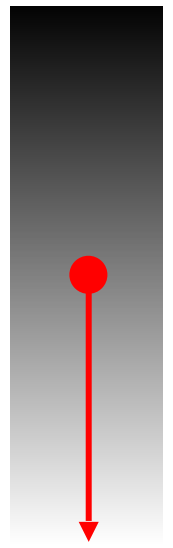
The gradient points in the direction of most rapid **increase of intensity**:

2D Image **Gradient**

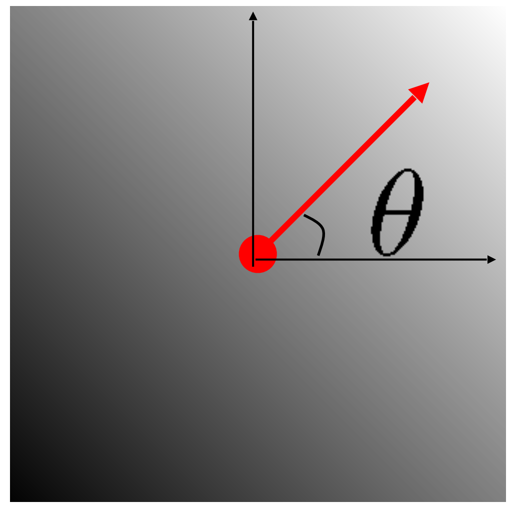
The gradient of an image: $\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$



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



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



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

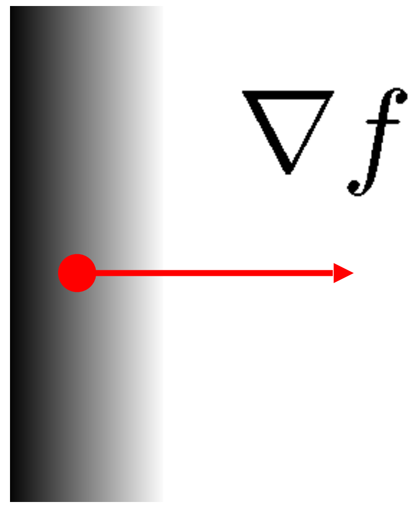
The gradient points in the direction of most rapid **increase of intensity**:

The **gradient direction** is given by: $\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$

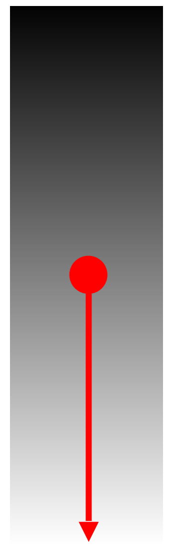
(how is this related to the direction of the edge?)

2D Image **Gradient**

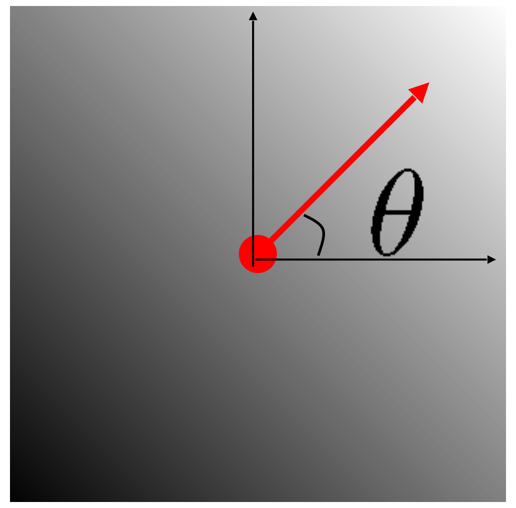
The gradient of an image: $\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$



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



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



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

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

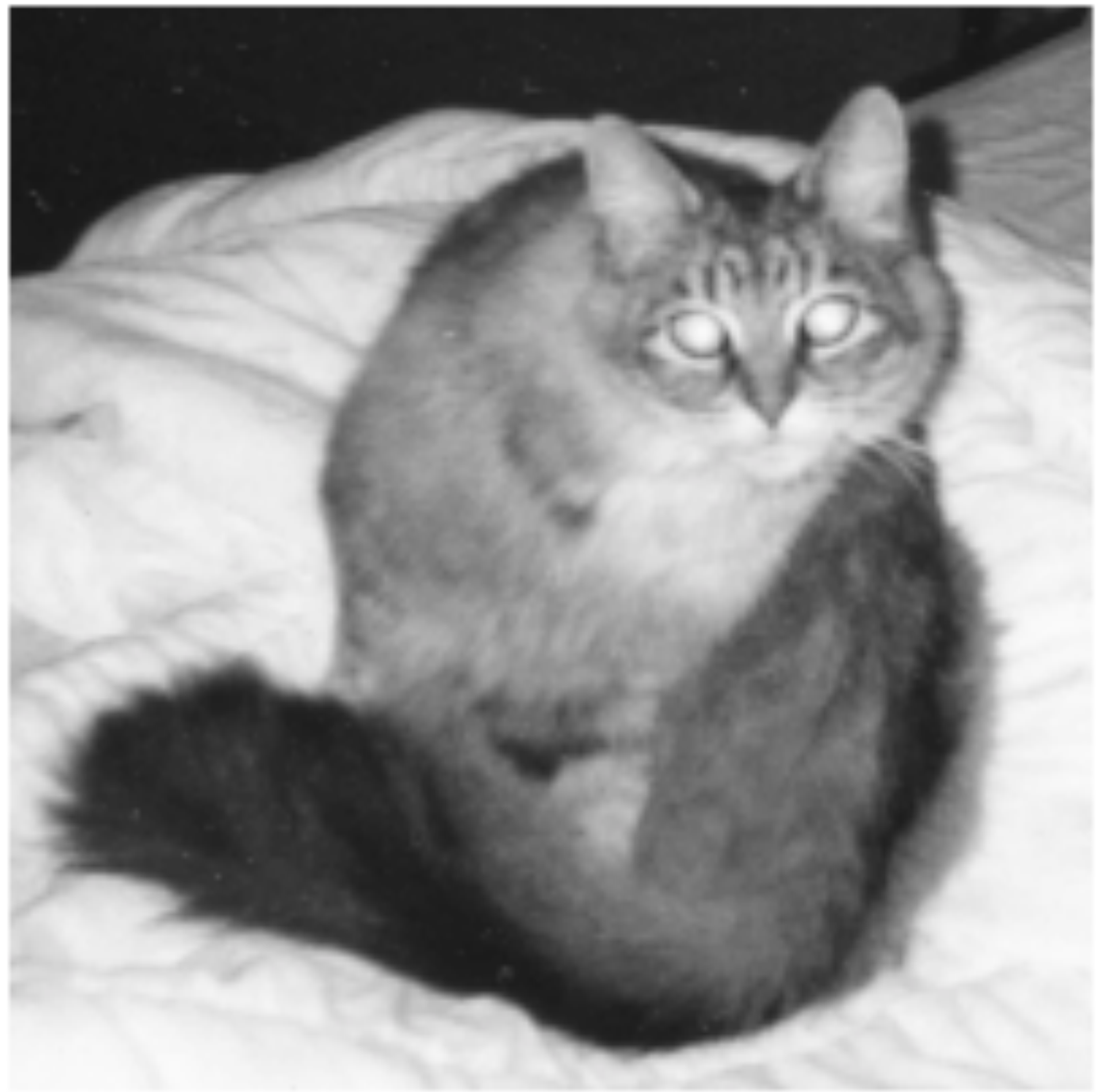
The **gradient direction** is given by: $\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$

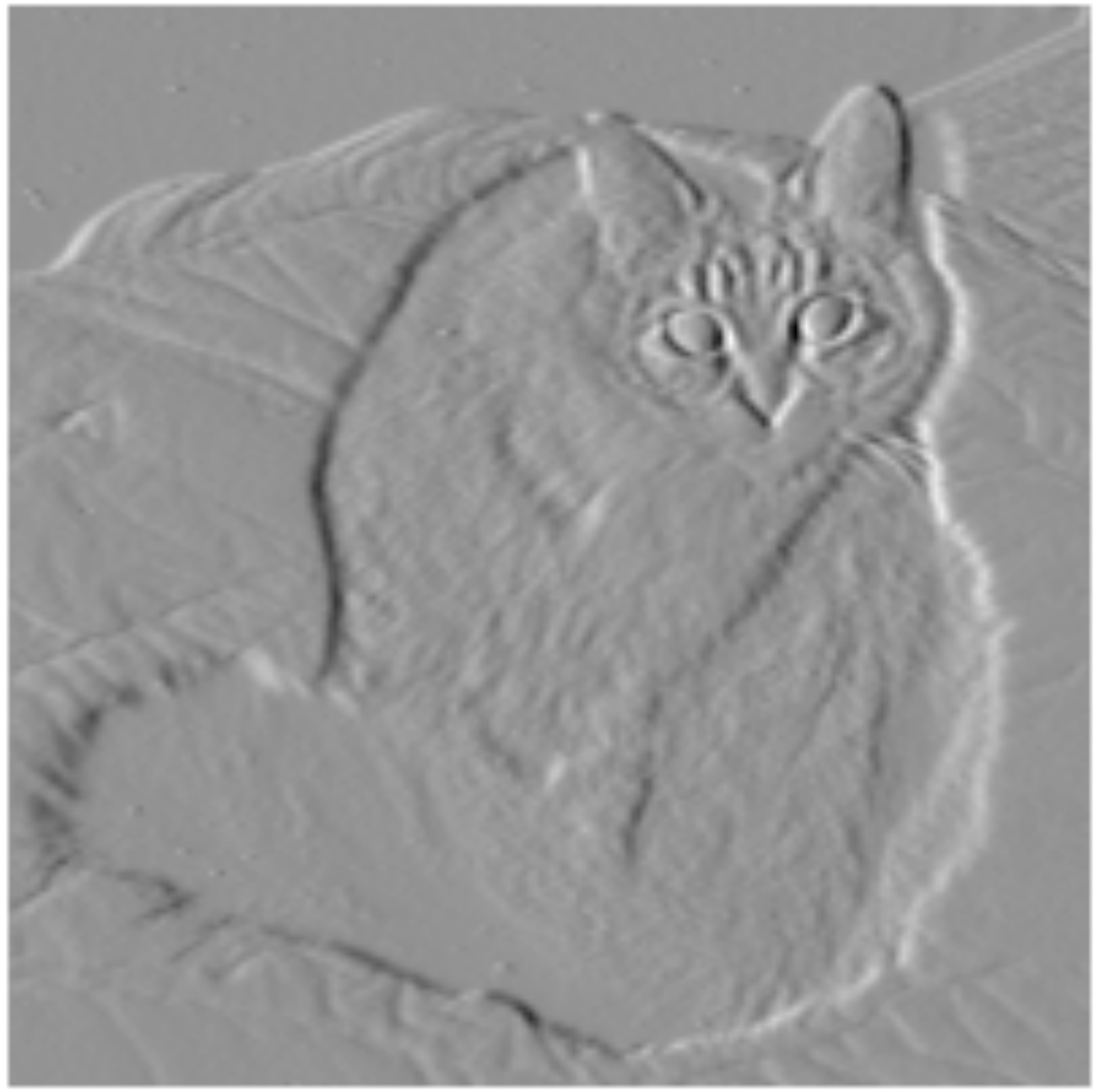
(how is this related to the direction of the edge?)

The edge strength is given by the **gradient magnitude**: $\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$

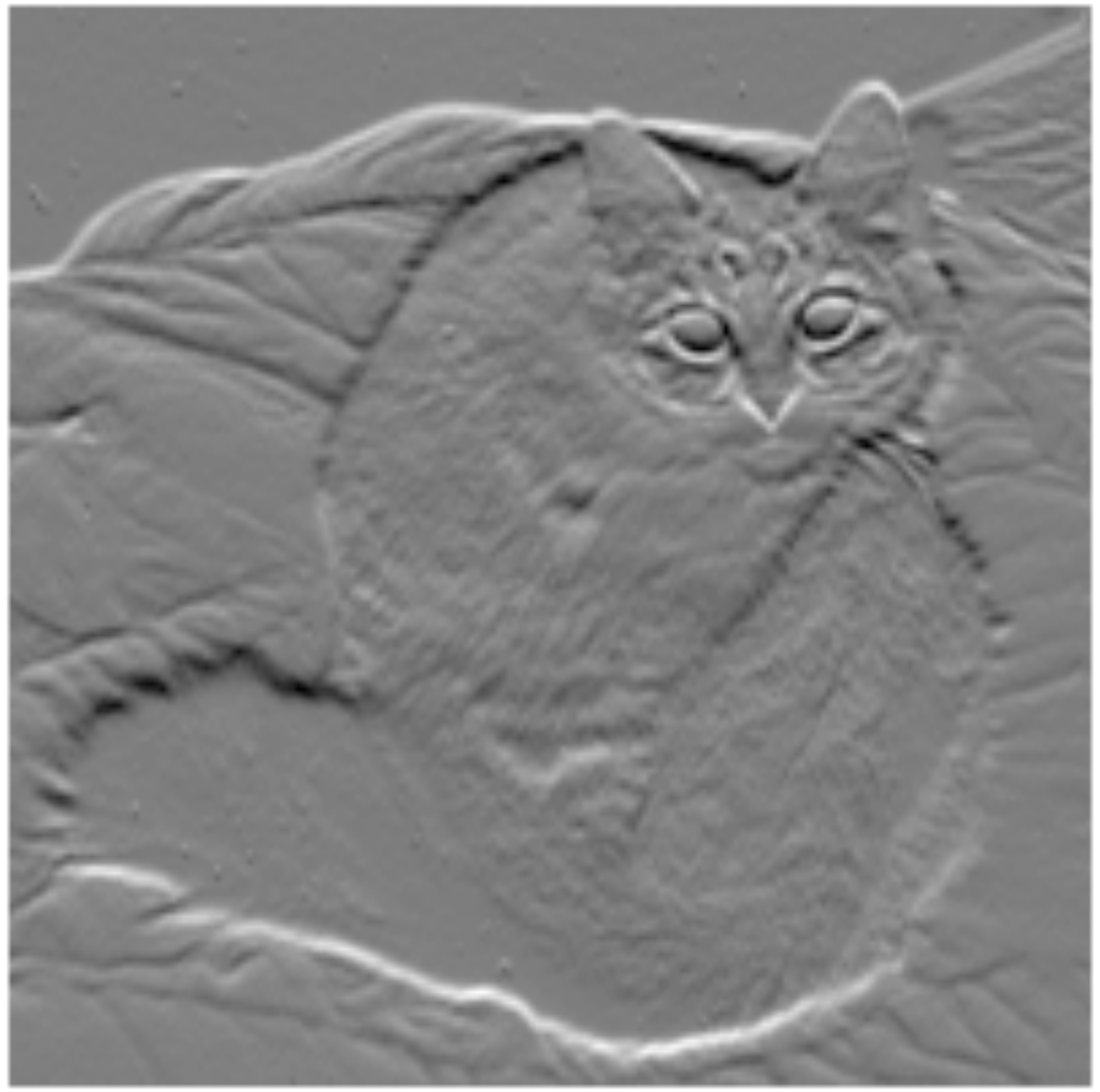
2D Edge Detection

Smooth image and convolve with [-1 1]





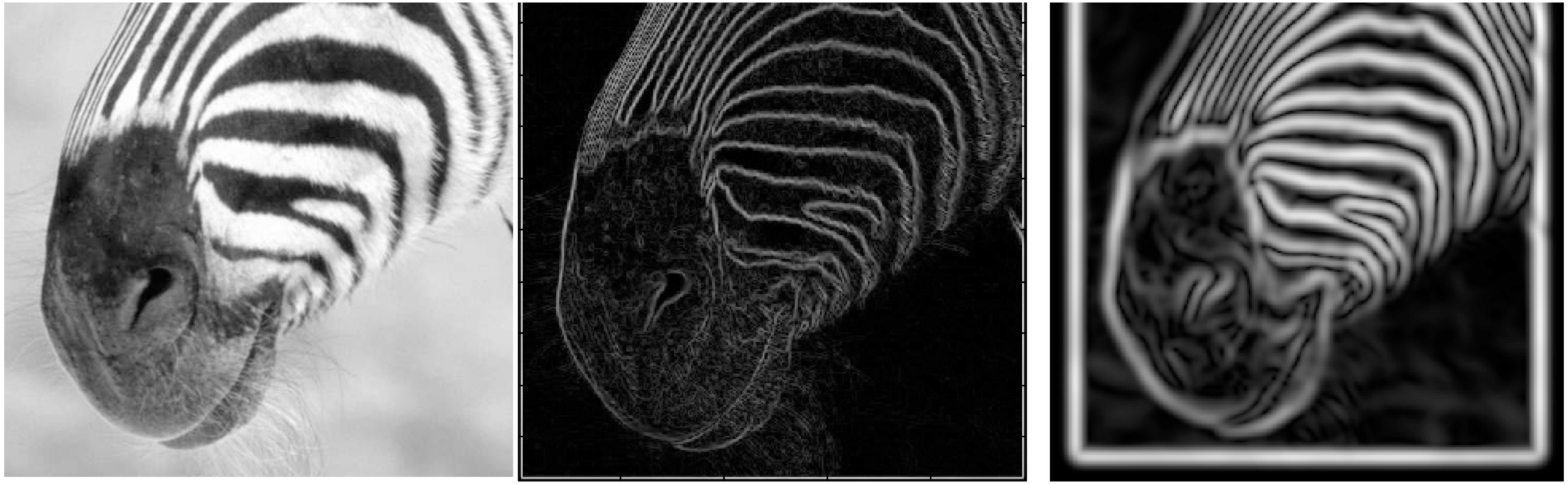
g_x



g_y

2D gradient: $\nabla I = \begin{bmatrix} g_x \\ g_y \end{bmatrix}$

Gradient **Magnitude**



$\sigma = 1$ $\sigma = 2$

Forsyth & Ponce (2nd ed.) Figure 5.4

Increased **smoothing**:

— eliminates noise edges

— makes edges smoother and thicker

— removes fine detail

Sobel Edge Detector

1. Use **central differencing** to compute gradient image (instead of first forward differencing). This is more accurate.

$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

2. **Threshold** to obtain edges

Original Image

Sobel Gradient

Sobel Edges

Sobel Edge Detector

1. Use **central differencing** to compute gradient image (instead of first forward differencing). This is more accurate.

$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

2. **Threshold** to obtain edges

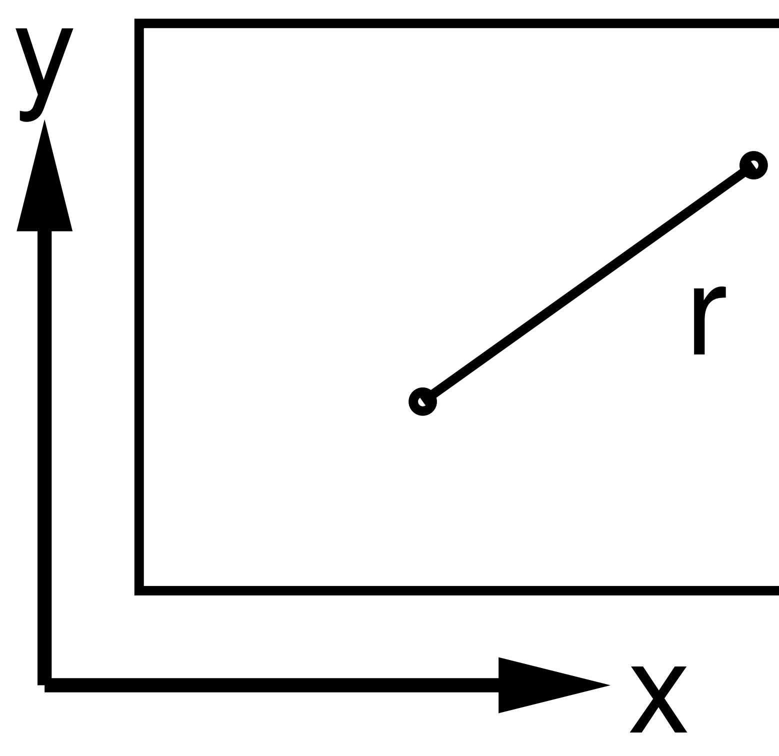
Original Image

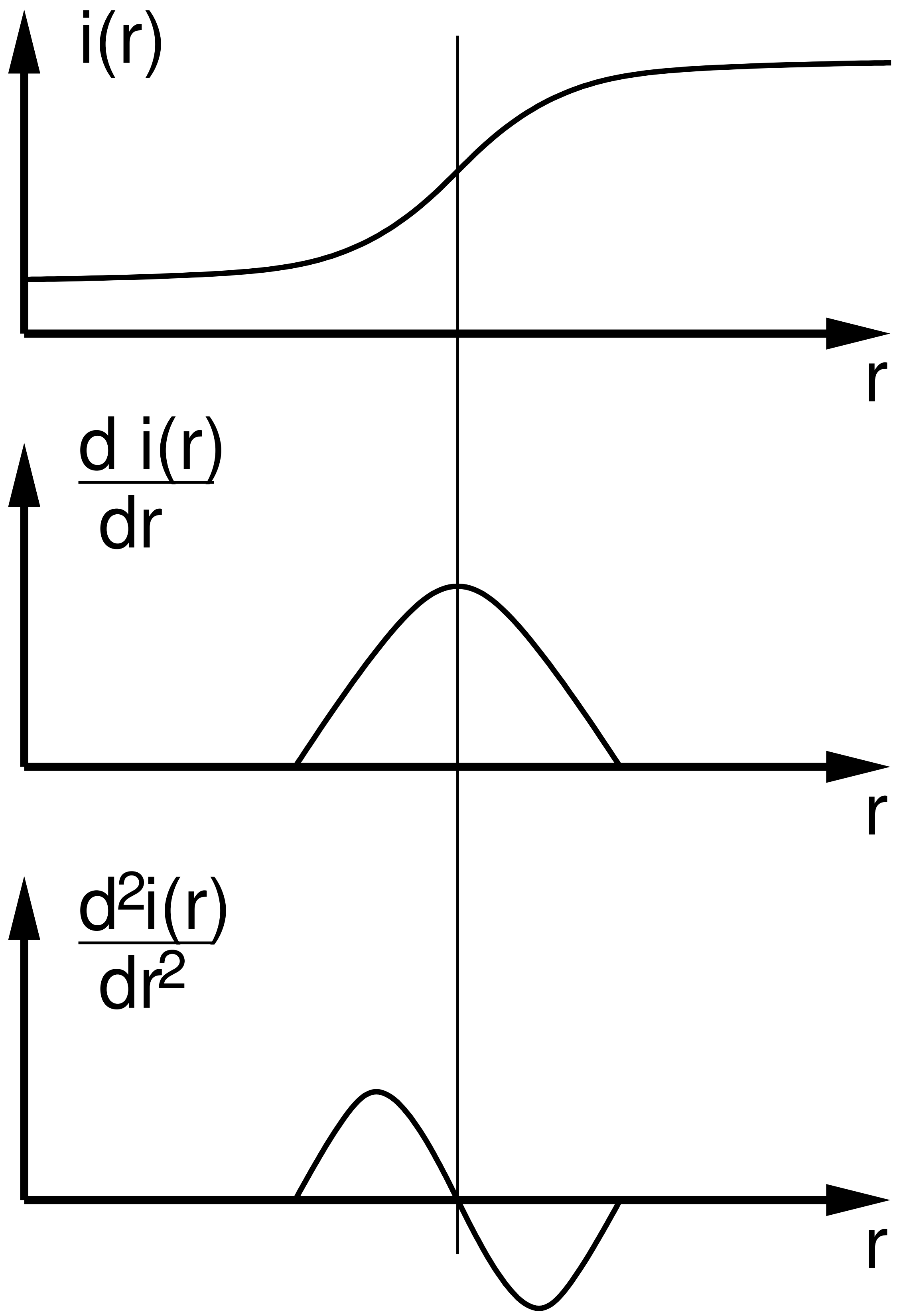
Sobel Gradient

Sobel Edges

Thresholds are brittle, we can do better!

Two Generic Approaches for **Edge** Detection





Two generic approaches to **edge point detection**:

— (significant) local extrema of a first derivative operator

— zero crossings of a second derivative operator

Marr / Hildreth **Laplacian of Gaussian**

A "**zero crossings** of a second derivative operator" approach

Steps:

1. Gaussian for smoothing

2. Laplacian (∇^2) for differentiation where

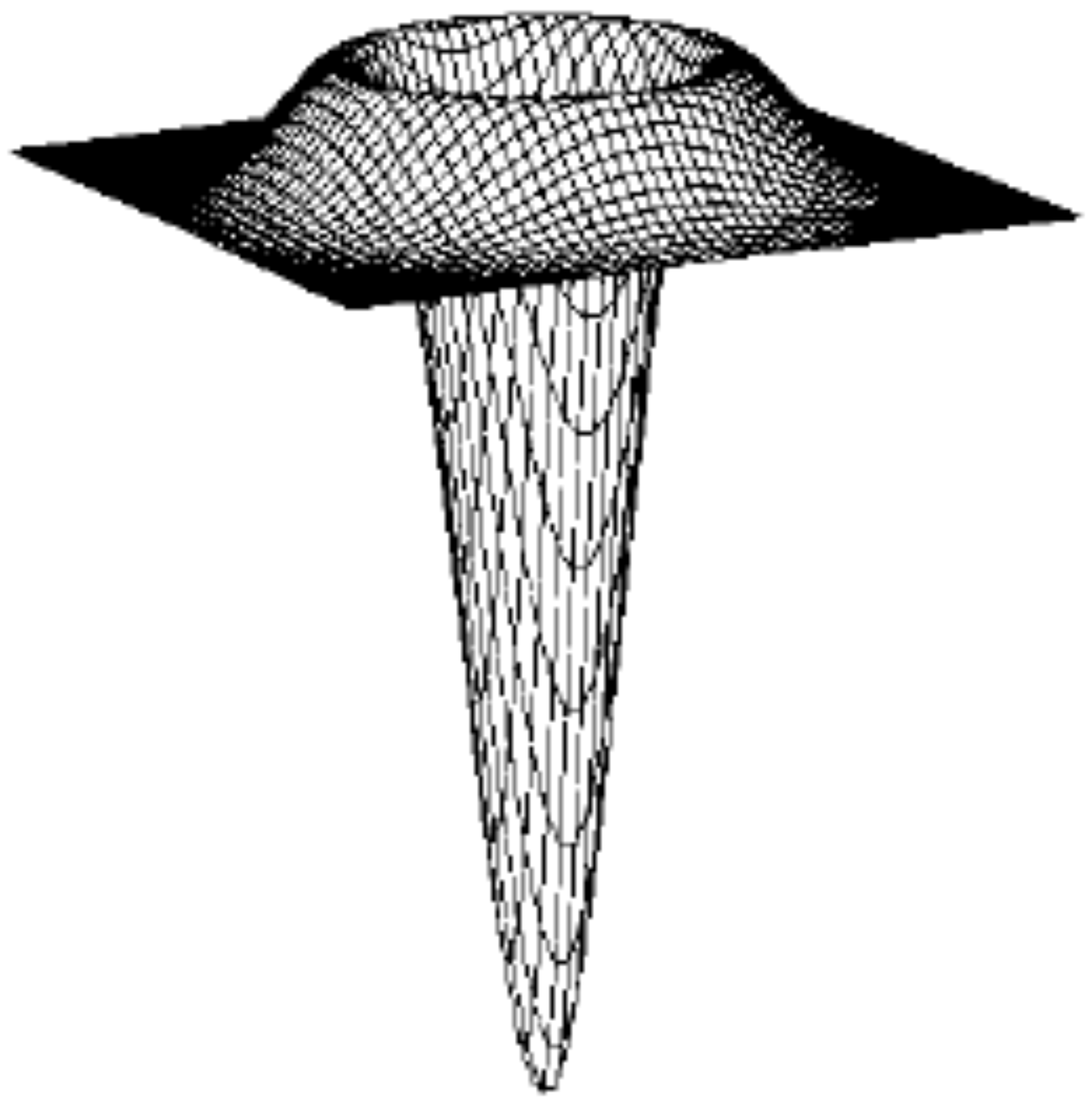
$$\nabla^2 f(x,y) = \frac{\partial^2 f(x,y)}{\partial x^2} + \frac{\partial^2 f(x,y)}{\partial y^2}$$

3. Locate zero-crossings in the Laplacian of the Gaussian ($\nabla^2 G$) where

$$\nabla^2 G(x,y) = \frac{-1}{2\pi\sigma^4}\left[2 - \frac{x^2+y^2}{\sigma^2}\right]\exp^{-\frac{x^2+y^2}{2\sigma^2}}$$

Marr / Hildreth **Laplacian of Gaussian**

Here's a 3D plot of the Laplacian of the Gaussian ($\nabla^2 G$)



. . . with its characteristic "Mexican hat" shape

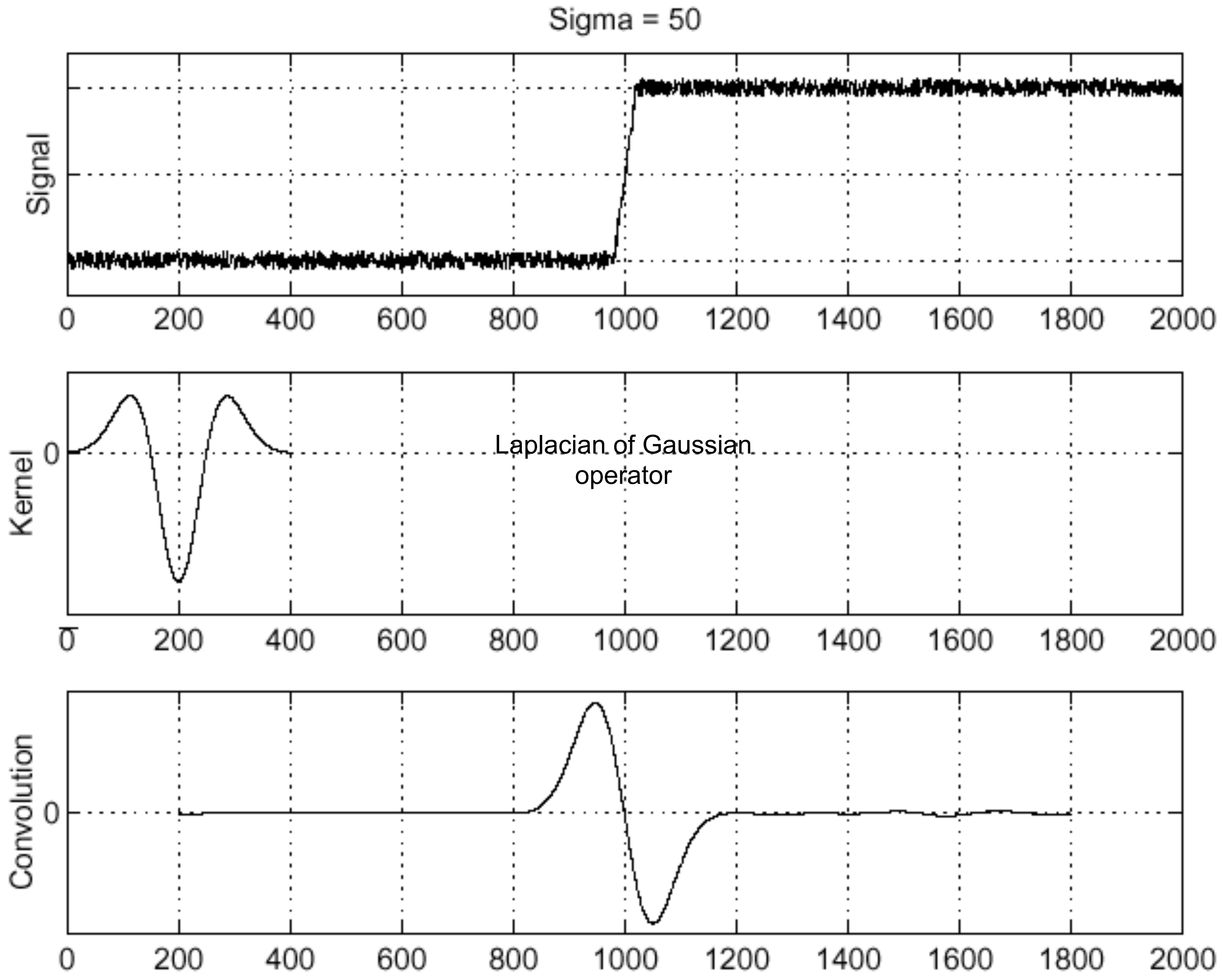
1D **Example**: Continued

Lets consider a row of pixels in an image:

$$I(X, 245)$$

$$\nabla^2 G$$

$$\nabla^2 G \otimes I(X, Y)$$



Where is the edge?

Zero-crossings of bottom graph

Marr / Hildreth **Laplacian of Gaussian**

5 x 5 LoG filter

0	0	-1	0	0
0	-1	-2	-1	0
-1	-2	16	-2	-1
0	-1	-2	-1	0
0	0	-1	0	0

17 x 17 LoG filter

0	0	0	0	0	0	-1	-1	-1	-1	-1	0	0	0	0	0
0	0	0	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	0	0
0	0	-1	-1	-1	-2	-3	-3	-3	-3	-3	-2	-1	-1	-1	0
0	0	-1	-1	-2	-3	-3	-3	-3	-3	-3	-3	-2	-1	-1	0
0	-1	-1	-2	-3	-3	-3	-2	-3	-2	-3	-3	-3	-2	-1	-1
0	-1	-2	-3	-3	-3	0	2	4	2	0	-3	-3	-3	-2	-1
-1	-1	-3	-3	-3	0	4	10	12	10	4	0	-3	-3	-3	-1
-1	-1	-3	-3	-2	2	10	18	21	18	10	2	-2	-3	-3	-1
-1	**-1**	**-3**	**-3**	**-3**	**4**	**12**	**21**	**24**	**21**	**12**	**4**	**-3**	**-3**	**-3**	**-1**
-1	-1	-3	-3	-2	2	10	18	21	18	10	2	-2	-3	-3	-1
-1	-1	-3	-3	-3	0	4	10	12	10	4	0	-3	-3	-3	-1
0	-1	-2	-3	-3	-3	0	2	4	2	0	-3	-3	-3	-2	-1
0	-1	-1	-2	-3	-3	-3	-2	-3	-2	-3	-3	-3	-2	-1	-1
0	-1	-1	-2	-3	-3	-3	-2	-3	-2	-3	-3	-3	-2	-1	-1
0	0	-1	-1	-1	-2	-3	-3	-3	-3	-3	-2	-1	-1	-1	0
0	0	0	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	0	0

Scale (σ)

Image From: A. Campilho

Marr / Hildreth **Laplacian of Gaussian**

Image From: A. Campilho

Assignment 1: High Frequency Image

original

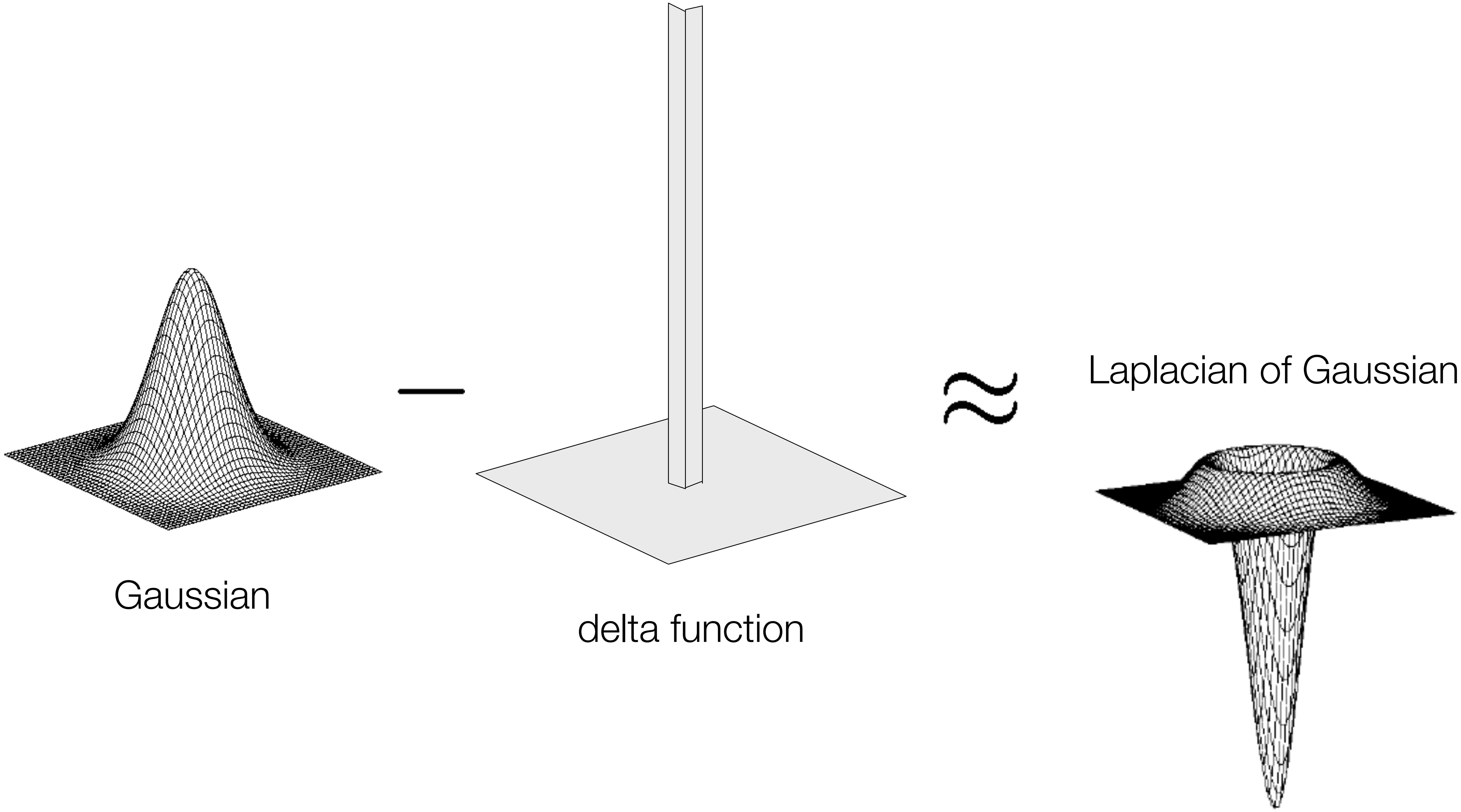
-

smoothed
Gaussian)

=

original - smoothed
(scaled by 4, offset +128)

Assignment 1: High Frequency Image



Comparing **Edge** Detectors

Comparing **Edge** Detectors

Good detection: minimize probability of false positives/negatives (spurious/missing) edges

Good localization: found edges should be as close to true image edge as possible

Single response: minimize the number of edge pixels around a single edge

Comparing **Edge** Detectors

Good detection: minimize probability of false positives/negatives (spurious/missing) edges

Good localization: found edges should be as close to true image edge as possible

Single response: minimize the number of edge pixels around a single edge

	Approach	Detection	Localization	Single Resp	Limitations
Sobel	Gradient Magnitude Threshold	Good	Poor	Poor	Results in Thick Edges
Marr / Hildreth	Zero-crossings of 2nd Derivative (LoG)	Good	Good	Good	Smooths Corners
Canny	Local extrema of 1st Derivative	Best	Good	Good	

Canny Edge Detector

A "**local extrema of a first derivative operator**" approach

Design Criteria:

1. good detection
   — low error rate for omissions (missed edges)
   — low error rate for commissions (false positive)

2. good localization

3. one (single) response to a given edge
   — (i.e., eliminate multiple responses to a single edge)

Canny Edge Detector

Steps:

1. Apply **directional derivatives** of Gaussian

2. Compute **gradient magnitude** and **gradient direction**

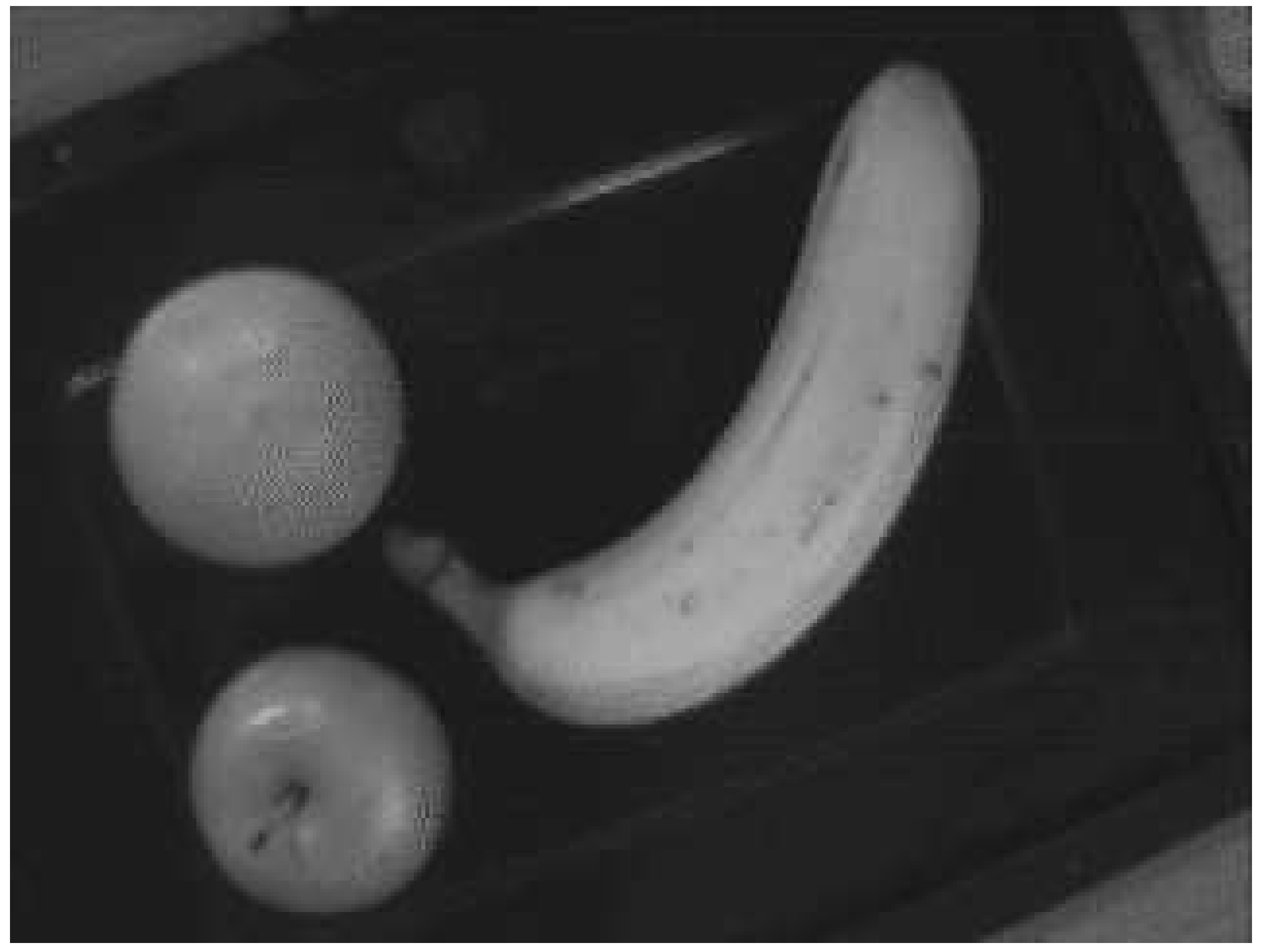
3. **Non-maximum** suppression
   — thin multi-pixel wide “ridges” down to single pixel width

4. **Linking** and thresholding
   — Low, high edge-strength thresholds
   — Accept all edges over low threshold that are connected to edge over high threshold

2D Edge Detection

Optional subtitle

Look at the magnitude of the smoothed gradient $|\nabla I|$



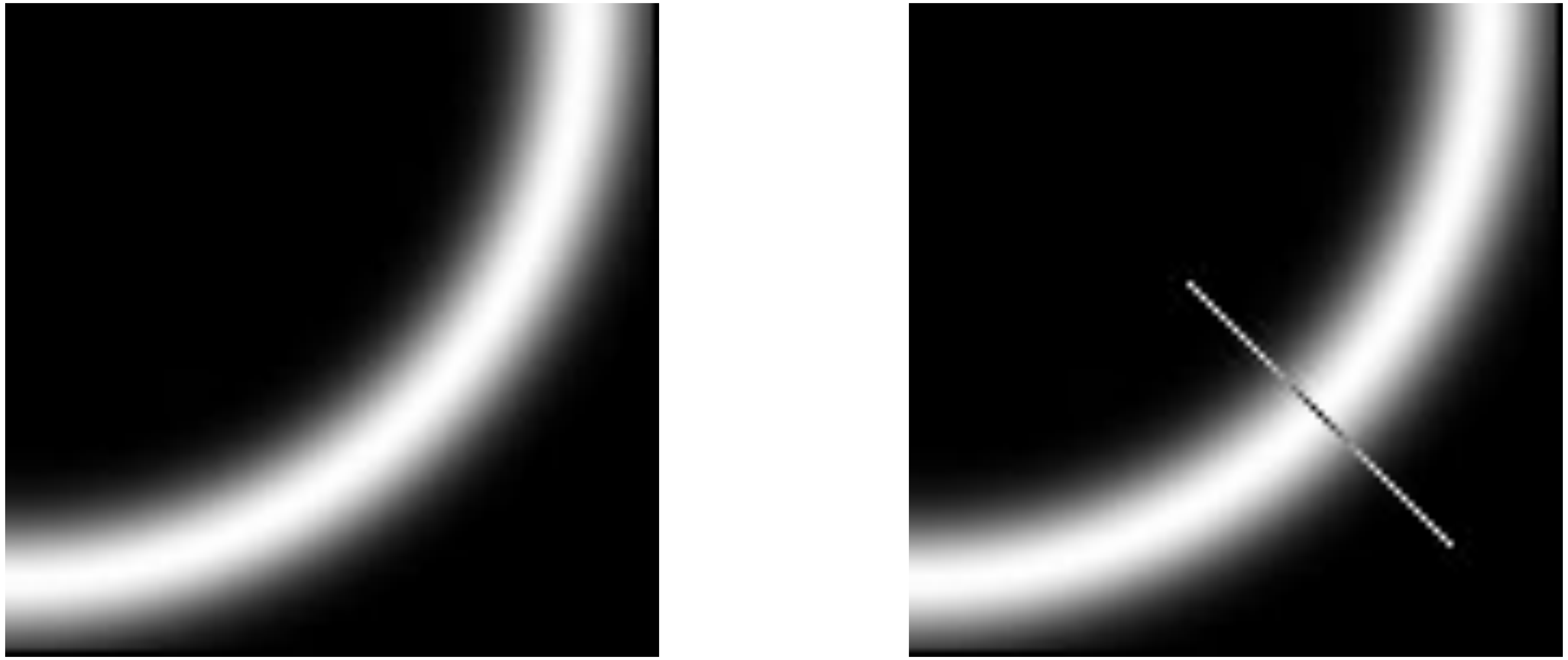
$$|\nabla I| = \sqrt{g_x^2 + g_y^2}$$

Non-maximal suppression (keep points where $|\nabla I|$ is a maximum in directions $\pm\nabla I$)

[Canny 1986]

Non-maxima Suppression

Idea: suppress near-by similar detections to obtain one "true" result

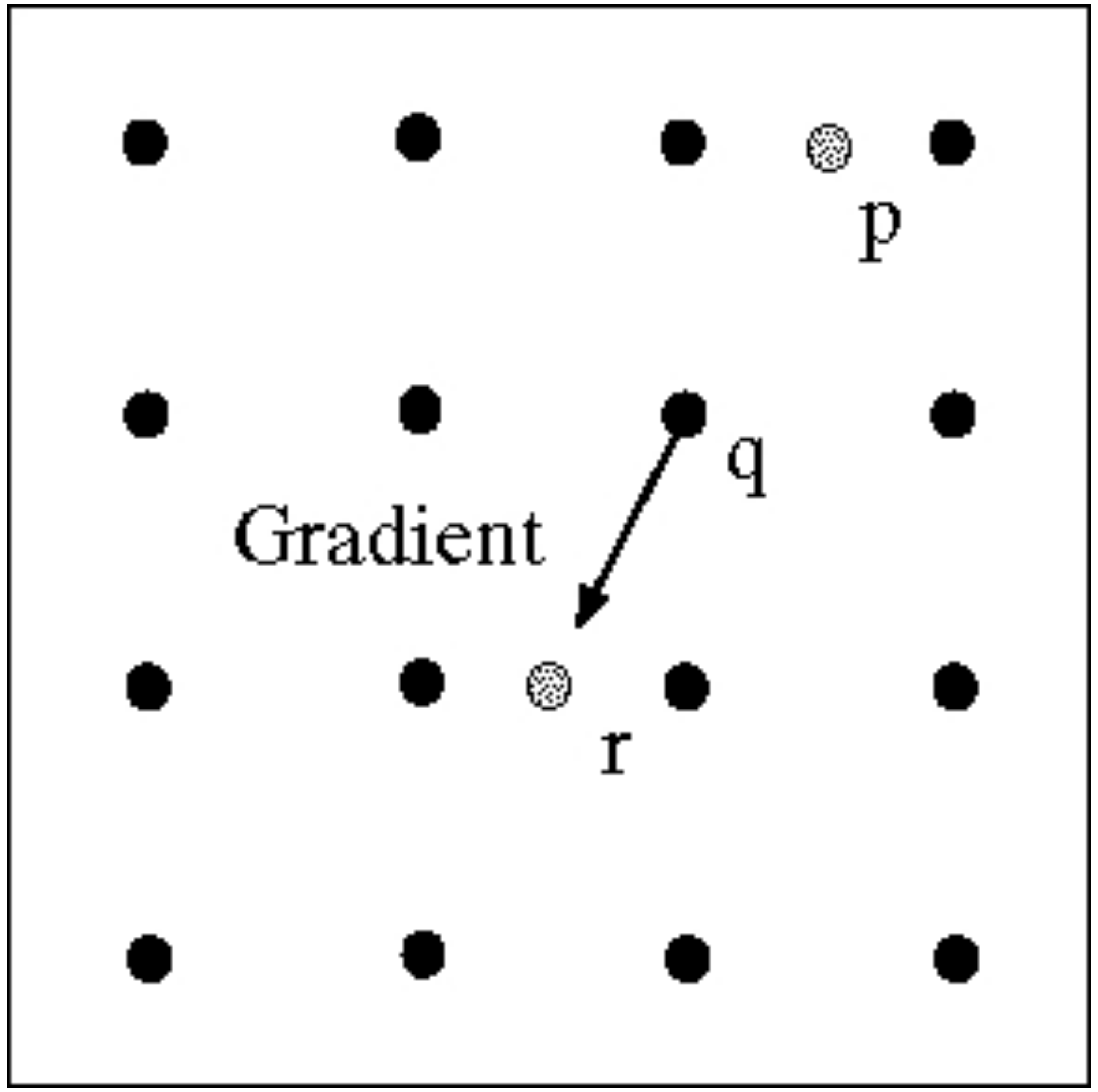


Non-maximal suppression (keep points where $|\nabla I|$ is a maximum in directions $\pm\nabla I$)

Select the image **maximum point** across the width of the edge

Non-maxima Suppression

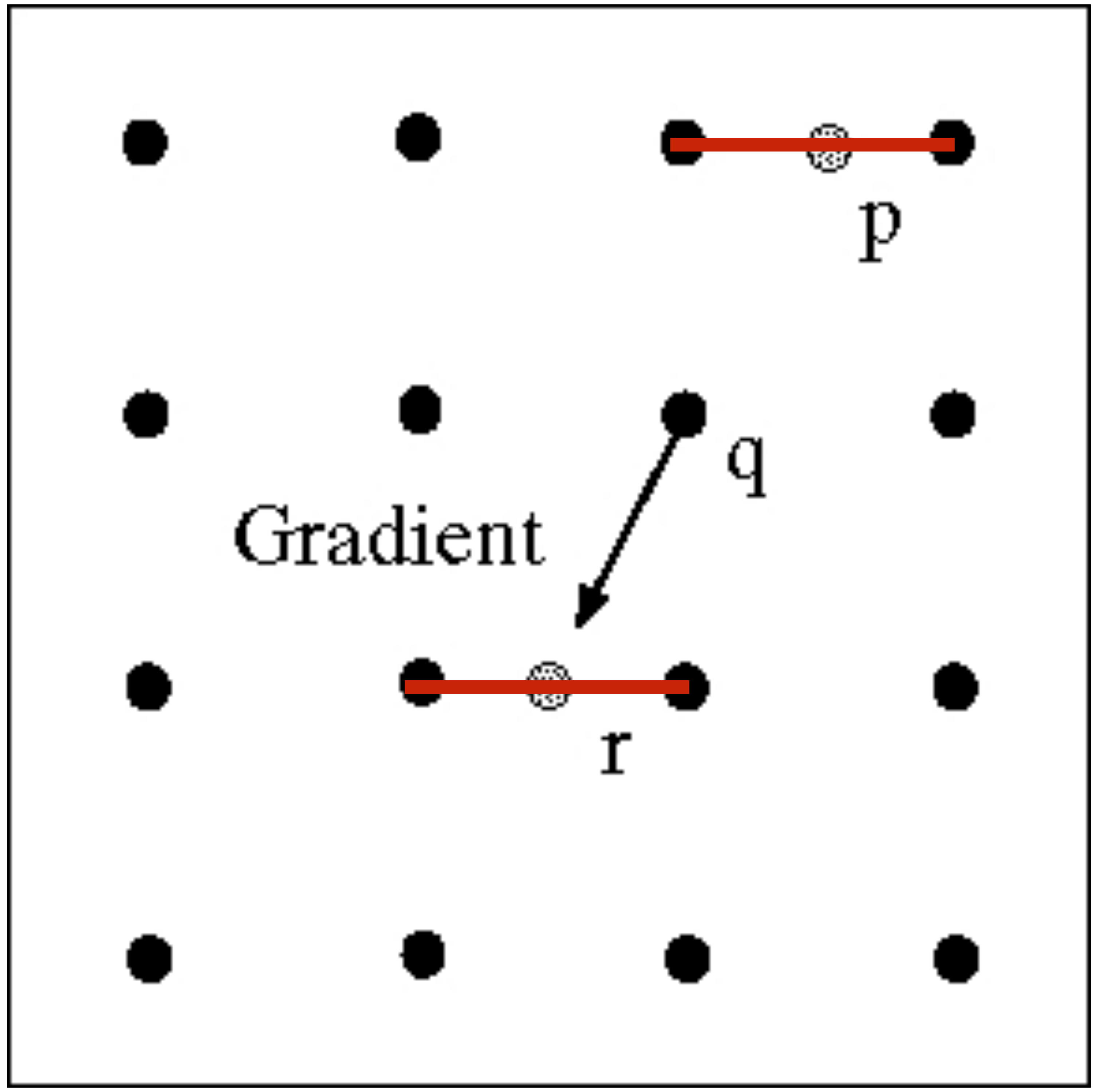
Value at *q* must be larger than interpolated values at *p* and *r*



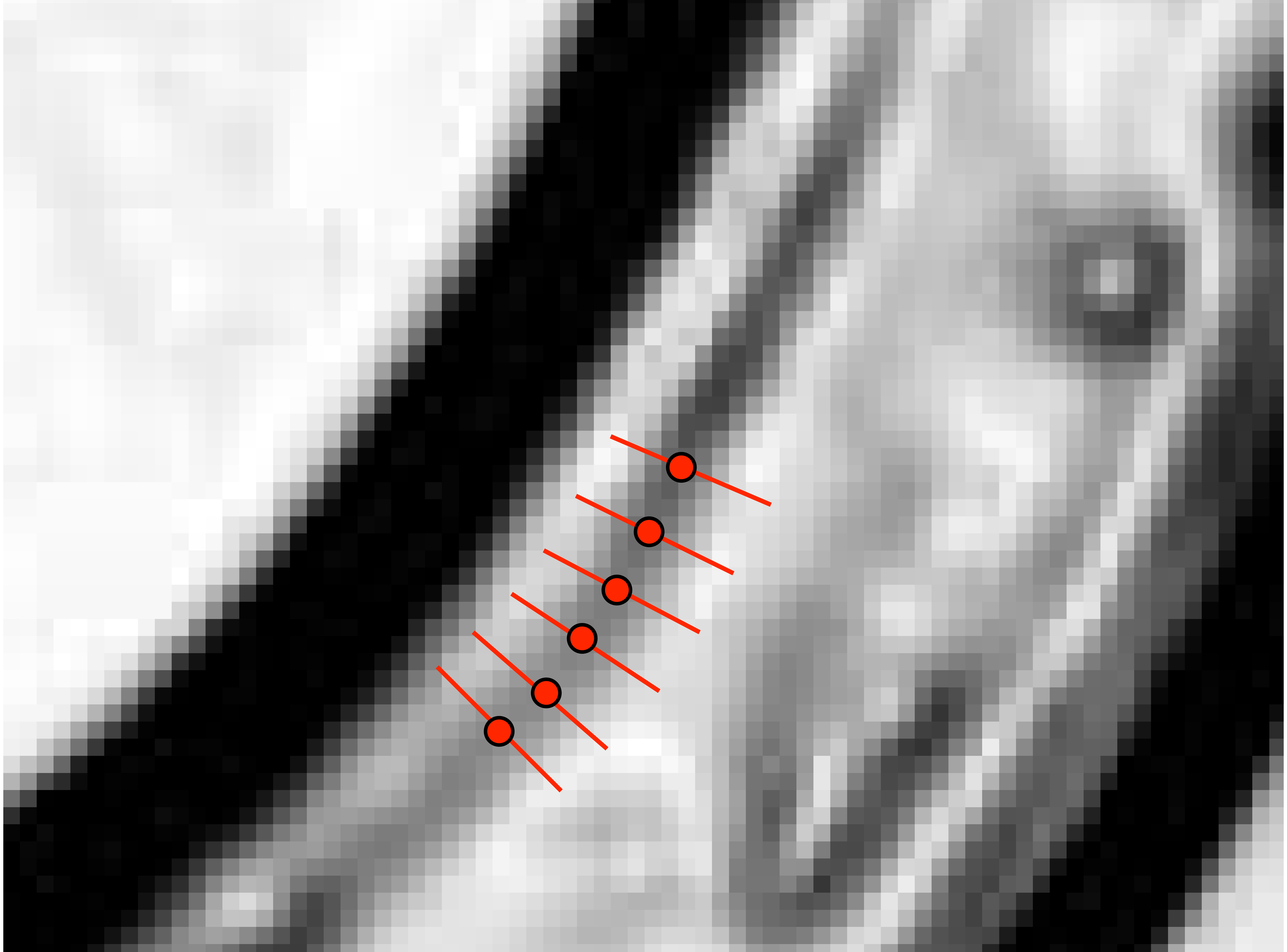
Forsyth & Ponce (2nd ed.) Figure 5.5 left

Non-maxima Suppression

Value at *q* must be larger than interpolated values at *p* and *r*

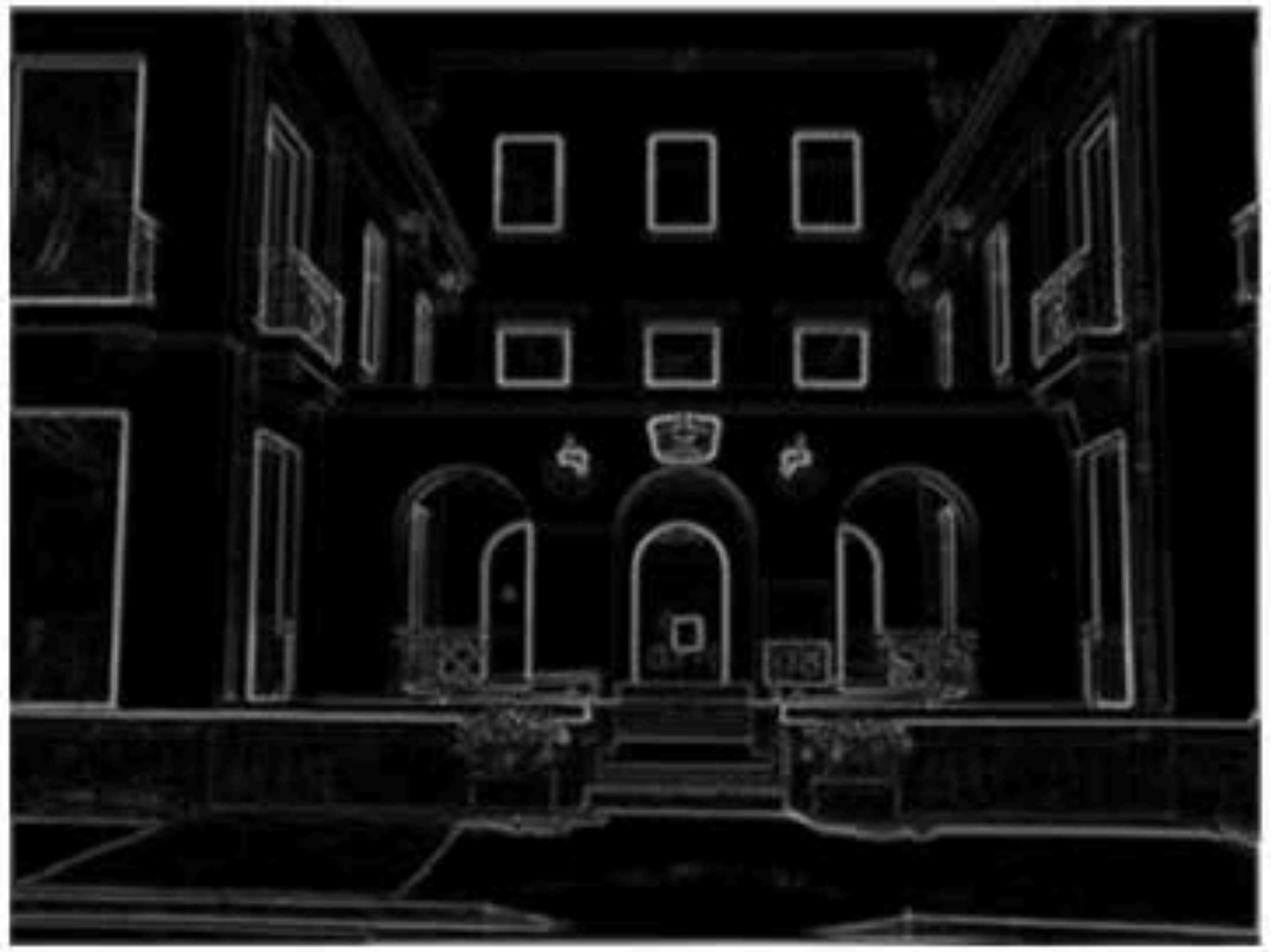


Forsyth & Ponce (2nd ed.) Figure 5.5 left



Example: Non-maxima Suppression

Original Image

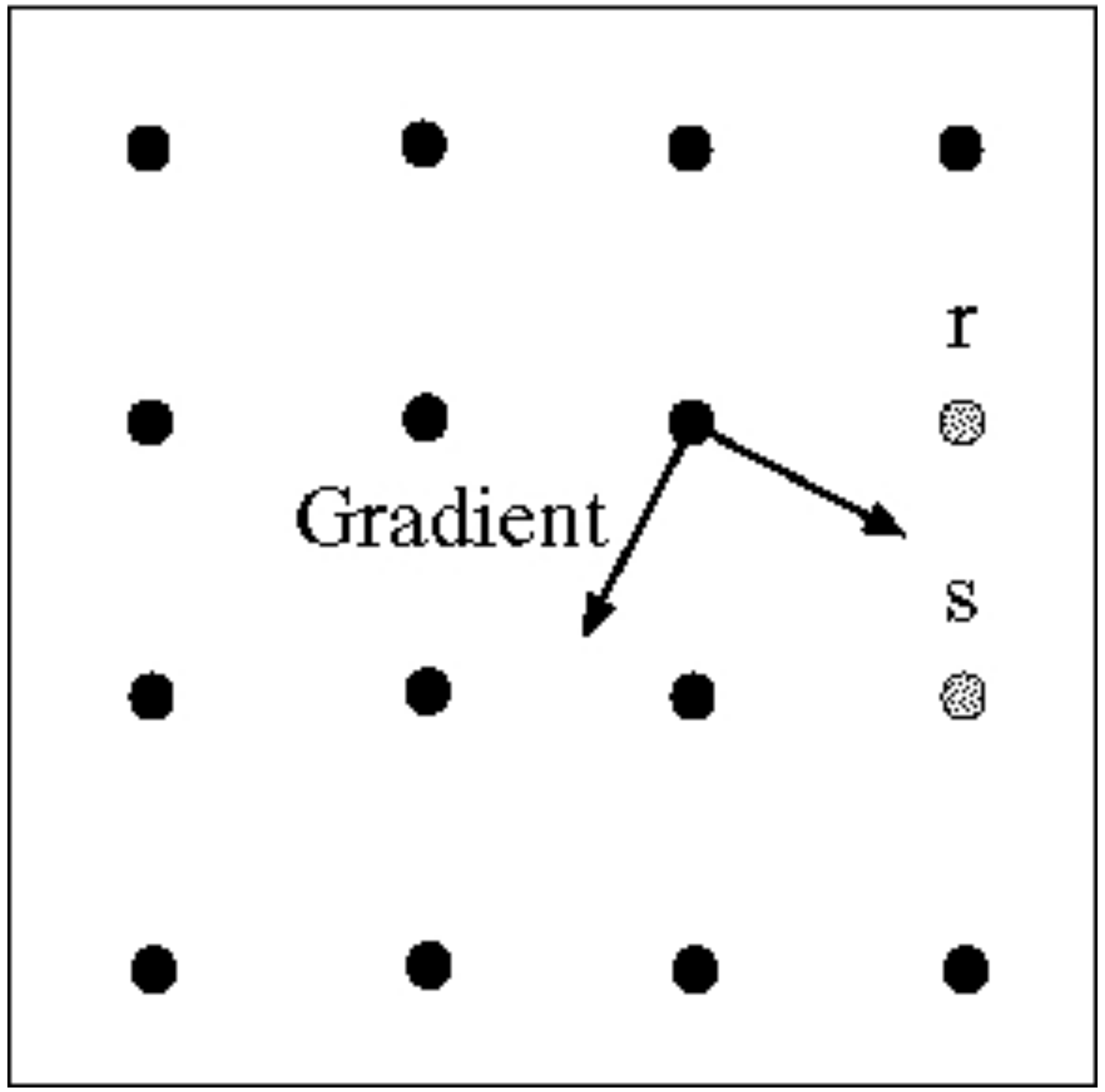


Gradient Magnitude

Non-maxima
Suppression

Slide Credit: Christopher Rasmussen

Linking Edge Points



Forsyth & Ponce (2nd ed.) Figure 5.5 right

Assume the marked point is an **edge point**. Take the normal to the gradient at that point and use this to predict continuation points (either *r* or *s*)

Edge **Hysteresis**

One way to deal with broken edge chains is to use hysteresis

Hysteresis: A lag or momentum factor

Idea: Maintain two thresholds $\mathbf{k}_{high}$ and $\mathbf{k}_{low}$

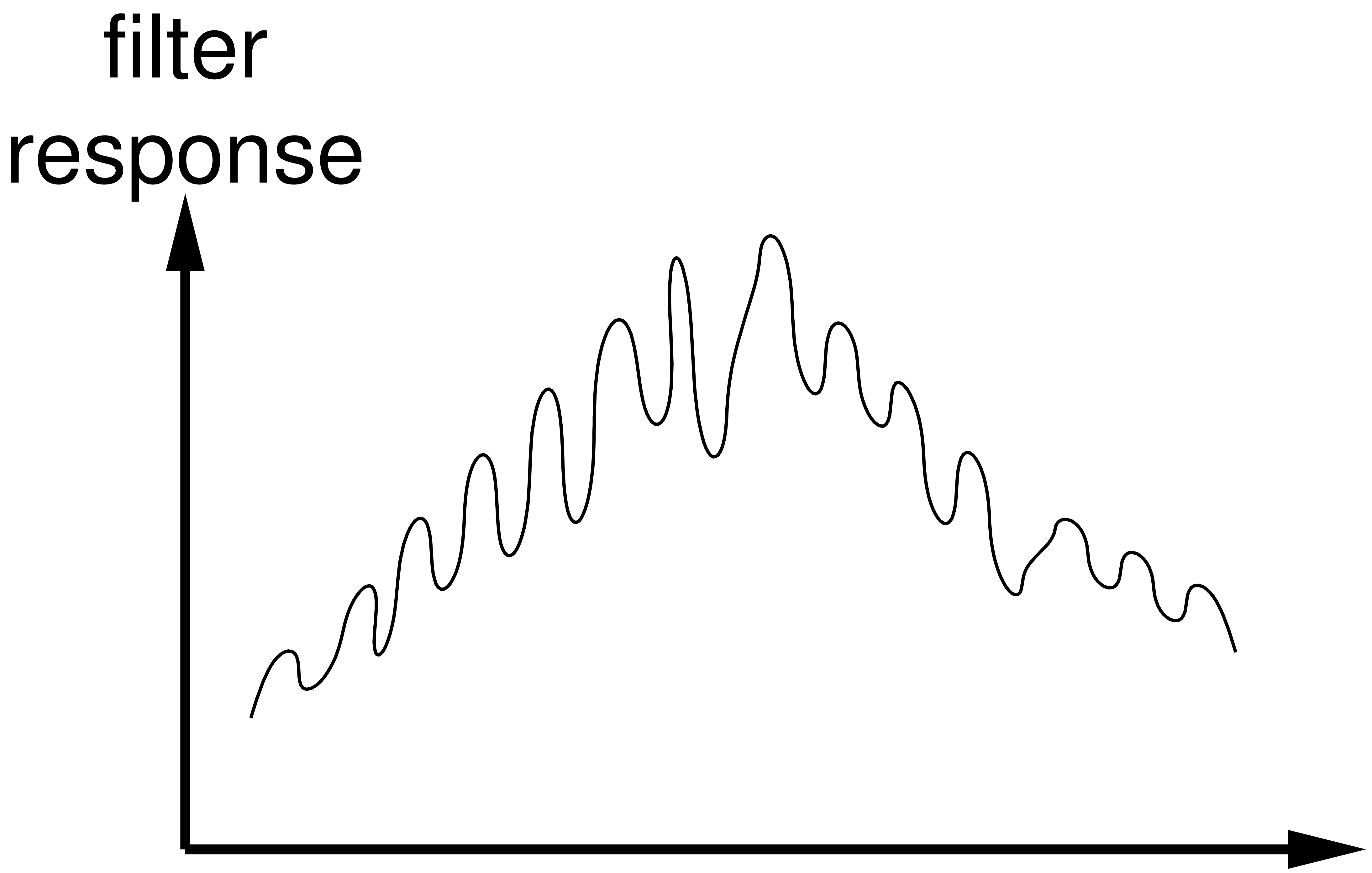
— Use k_{high} to find strong edges to start edge chain

— Use k_{low} to find weak edges which continue edge chain

Typical ratio of thresholds is (roughly):

$$\frac{\mathbf{k}_{high}}{\mathbf{k}_{low}} = 2$$

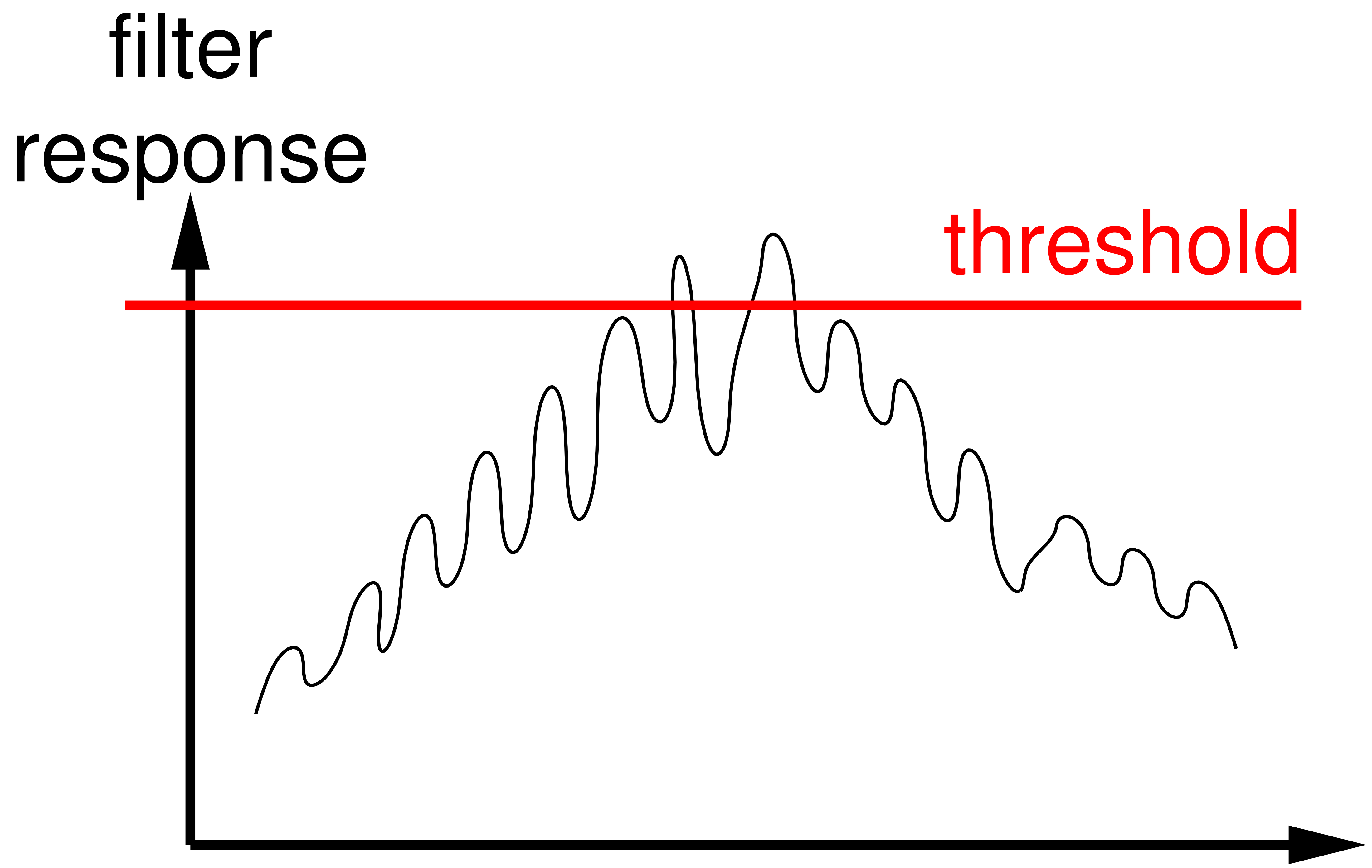
Example: Edge Detection



Question: How many edges are there?

Question: What is the position of each edge?

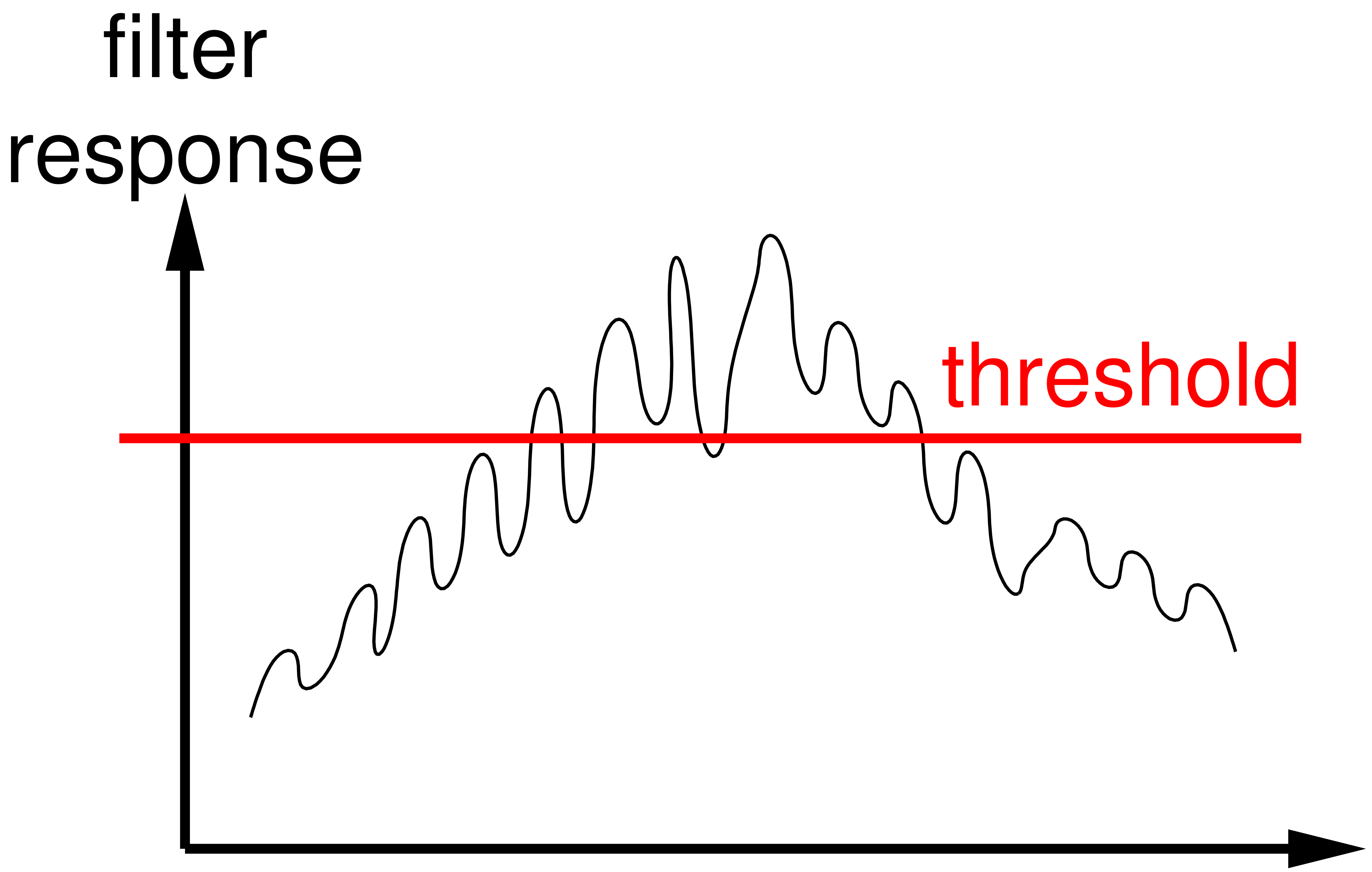
Example: Edge Detection



Question: How many edges are there?

Question: What is the position of each edge?

Example: Edge Detection



Question: How many edges are there?

Question: What is the position of each edge?

Canny Edge Detector

Original
Image

Strong +
connected
Weak Edges

Strong
Edges

Weak
Edges

courtesy of G. Loy

2D Edge Detection

Optional subtitle

Threshold the gradient magnitude with two thresholds: T_{high} and T_{low}

Edges start at edge locations with gradient magnitude > T_{high}

Continue tracing edge until gradient magnitude falls below T_{low}

[Canny 1986]

Example

Forsyth & Ponce (1st ed.) Figure 8.13 top

Example

Forsyth & Ponce (1st ed.) Figure 8.13 top

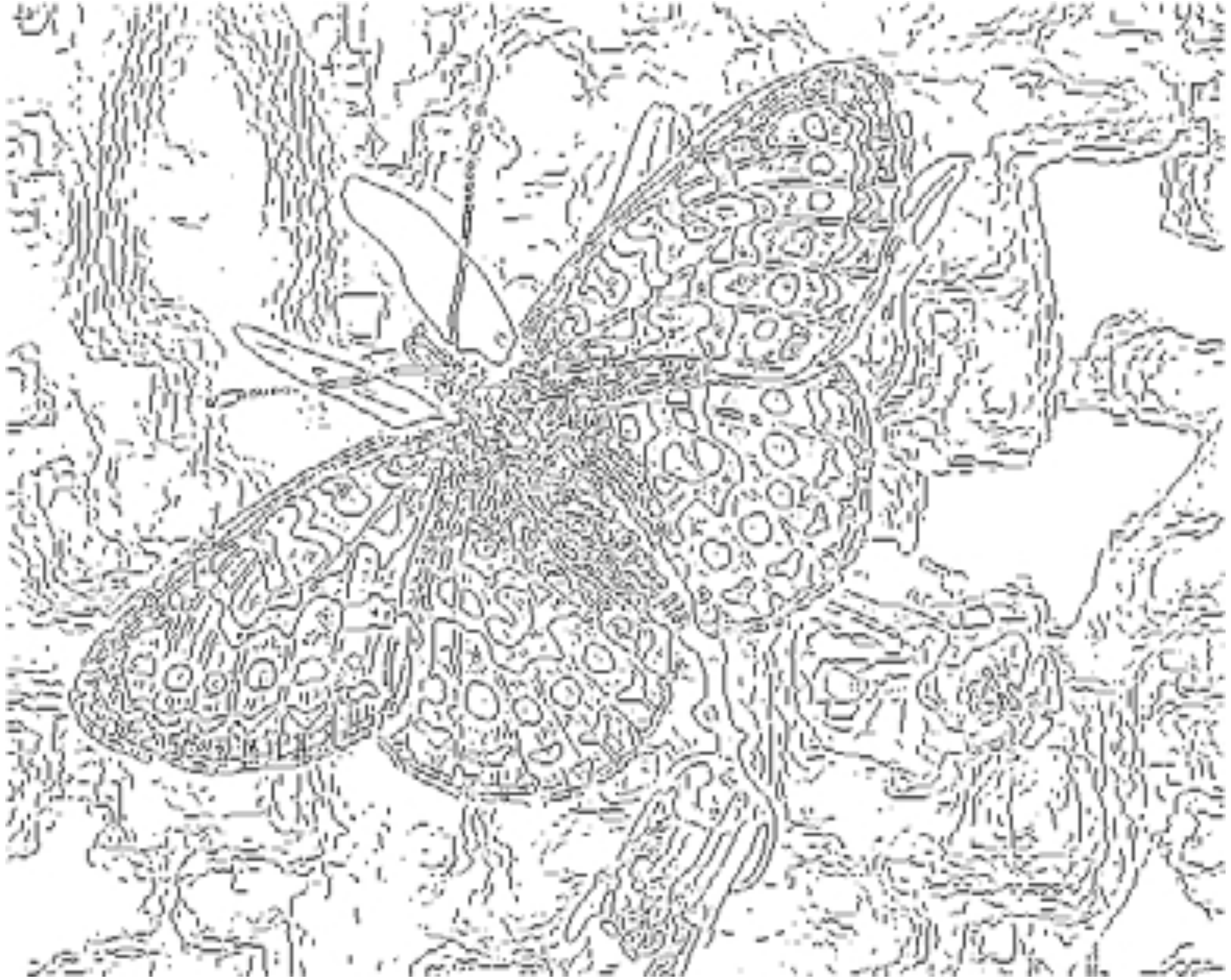


Figure 8.13 bottom left

Fine scale ($\sigma = 1$), high threshold

Example

Forsyth & Ponce (1st ed.) Figure 8.13 top

Figure 8.13 bottom middle

Fine scale ($\sigma = 4$), high threshold

Example

Forsyth & Ponce (1st ed.) Figure 8.13 top

Figure 8.13 bottom right

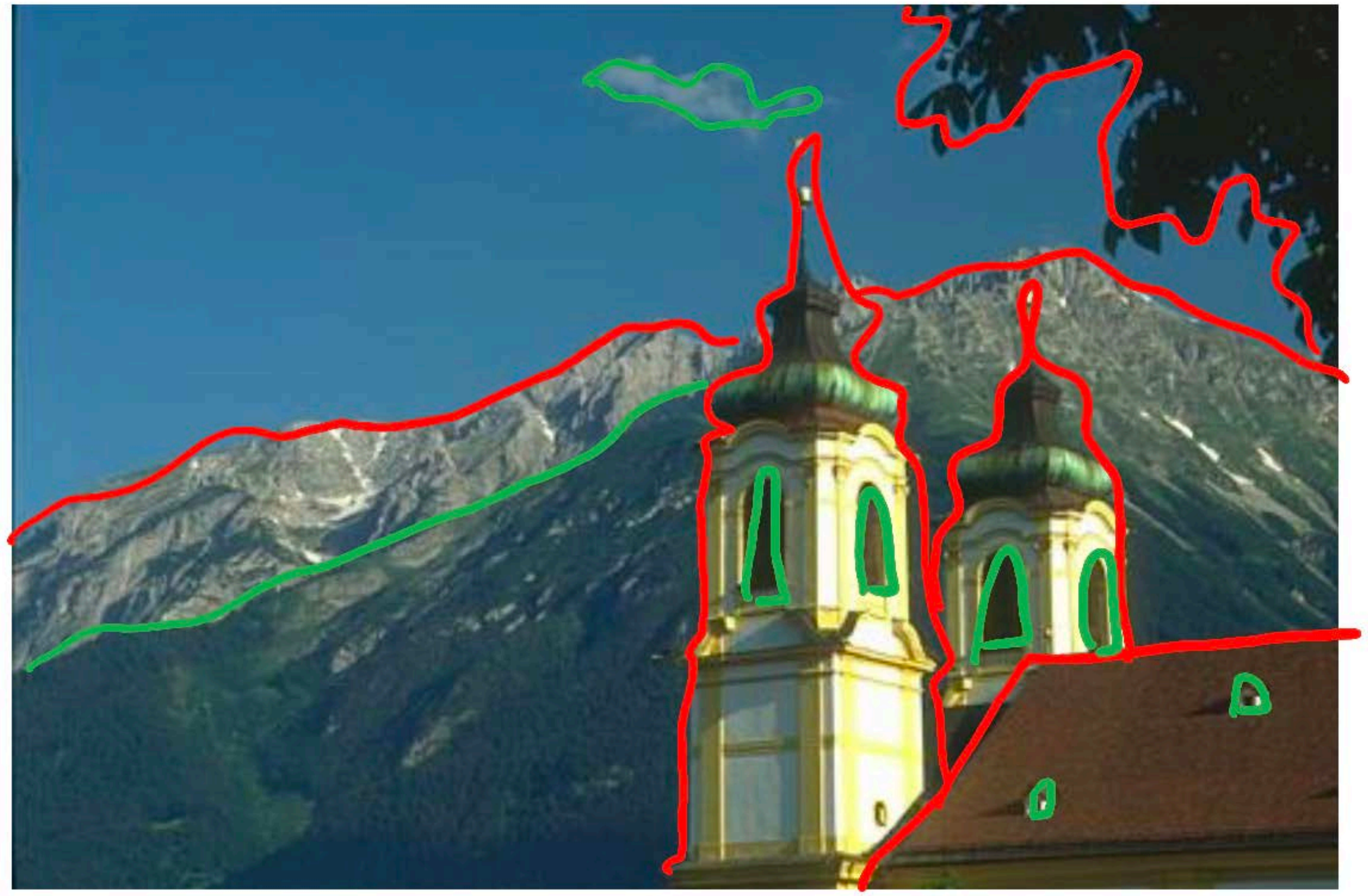
Fine scale ($\sigma = 4$), low threshold

How do humans perceive **boundaries**?

Edges are a property of the 2D image.

It is interesting to ask: How closely do image edges correspond to boundaries that humans perceive to be salient or significant?

How do humans perceive **boundaries**?



"Divide the image into some number of segments, where the segments represent 'things' or 'parts of things' in the scene. The number of segments is up to you, as it depends on the image. Something between 2 and 30 is likely to be appropriate. It is important that all of the segments have approximately equal importance."

(Martin et al. 2004)

How do humans perceive **boundaries**?

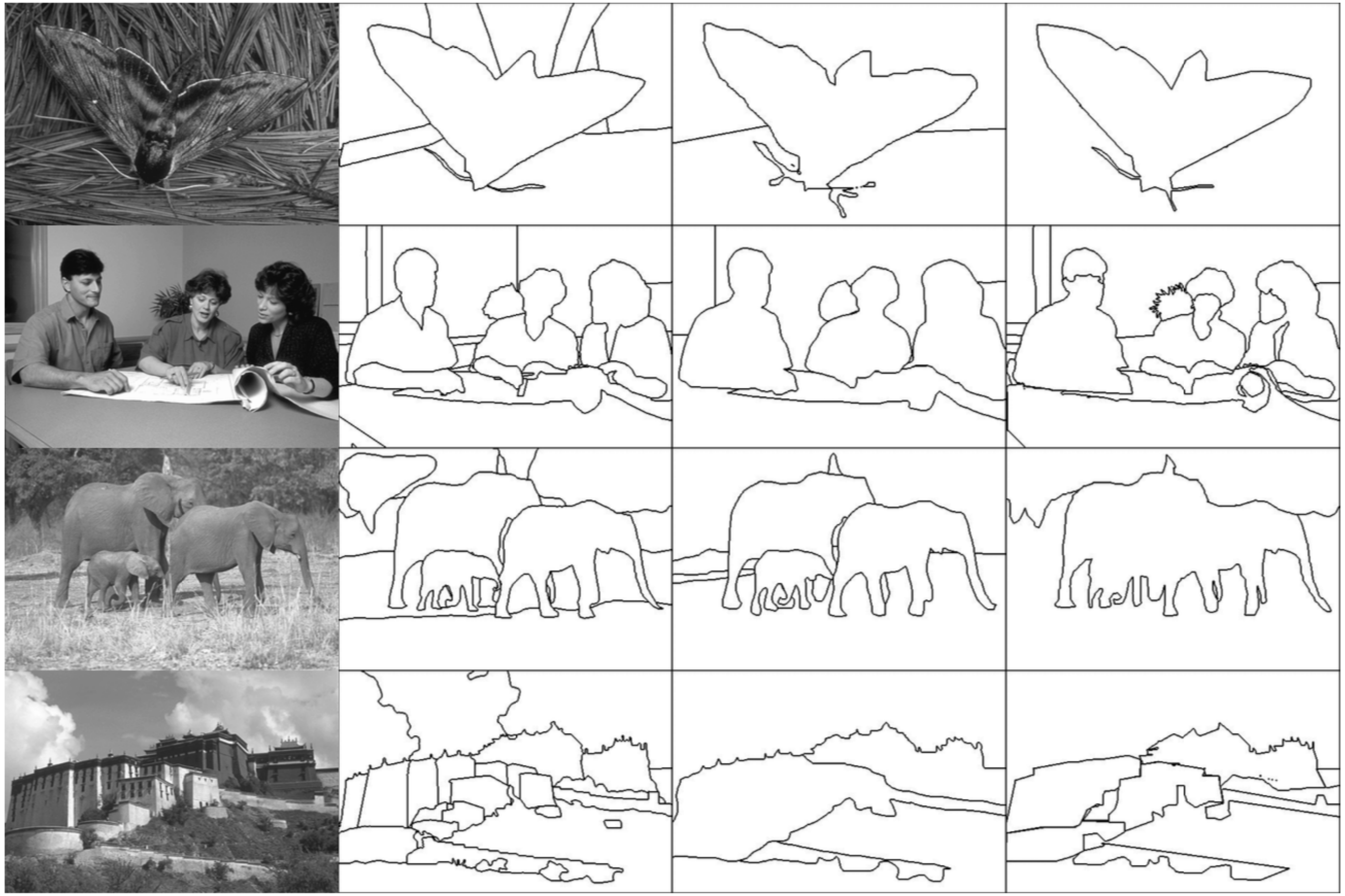


Figure Credit: Martin et al. 2001

How do humans perceive **boundaries**?

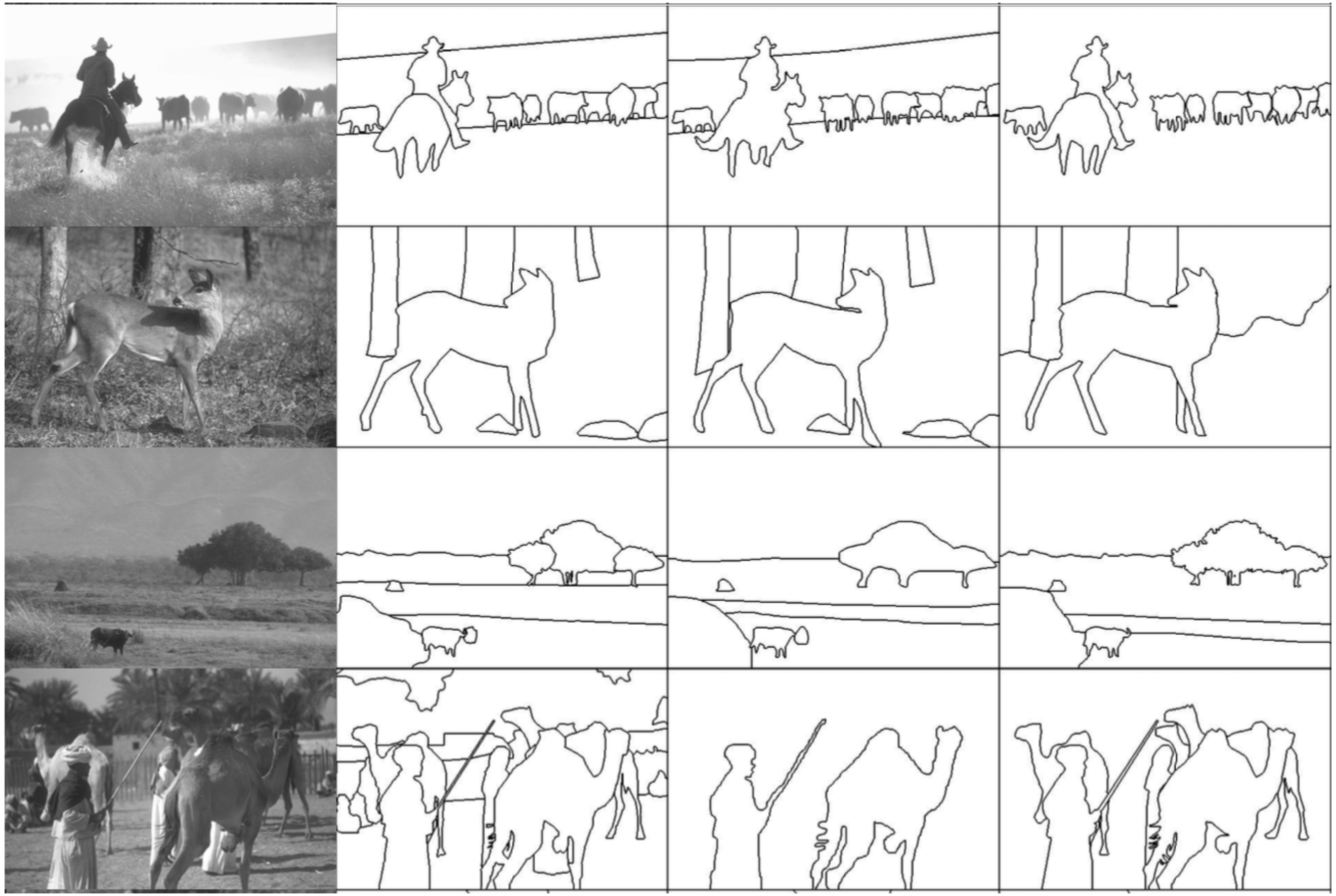
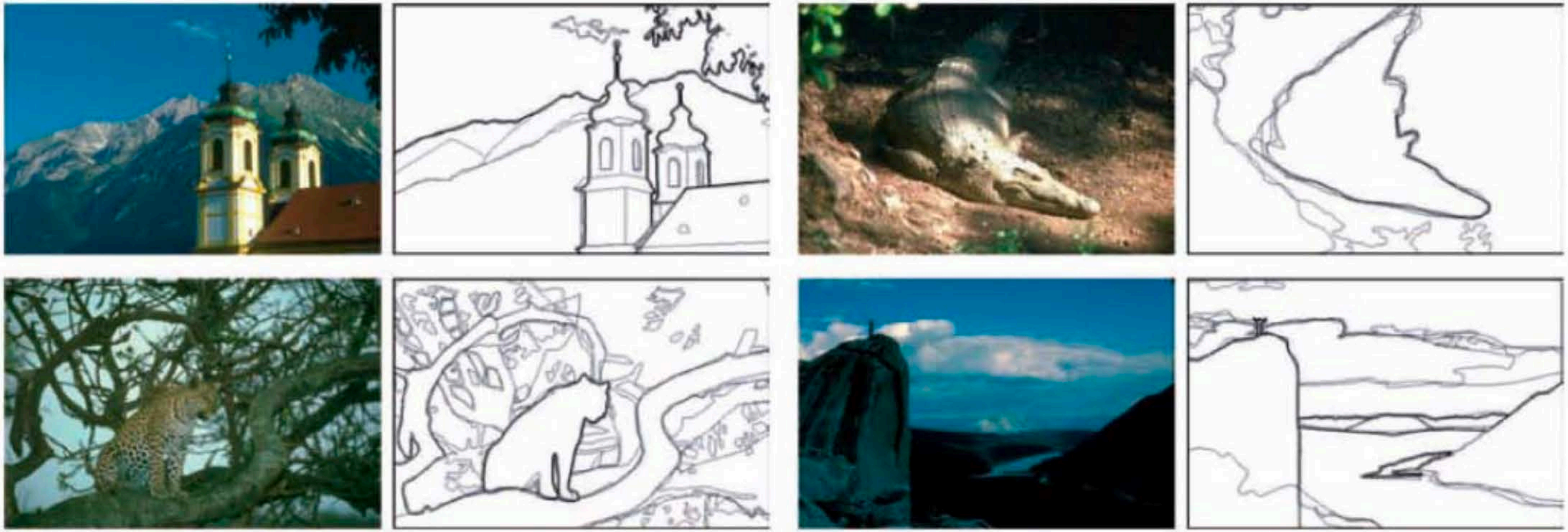


Figure Credit: Martin et al. 2001

How do humans perceive **boundaries**?



Each image shows multiple (4-8) human-marked boundaries. Pixels are darker where more humans marked a boundary.

Figure Credit: Szeliski Fig. 4.31. **Original**: Martin et al. 2004

Boundary Detection

We can formulate **boundary detection** as a high-level recognition task

— Try to learn, from sample human-annotated images, which visual features or cues are predictive of a salient/significant boundary

Many boundary detectors output a **probability or confidence** that a pixel is on a boundary

Summary

Physical properties of a 3D scene cause “**edges**” in an image:

— depth discontinuity

— surface orientation discontinuity

— reflectance discontinuity

— illumination boundaries

Basic approaches to **edge detection**:

—Smooth image to a desired scale and extract image gradients

—local extrema of a first derivative operator → **Canny**

Many algorithms consider “**boundary detection**” as a high-level recognition task and output a probability or confidence that a pixel is on a human-perceived boundary