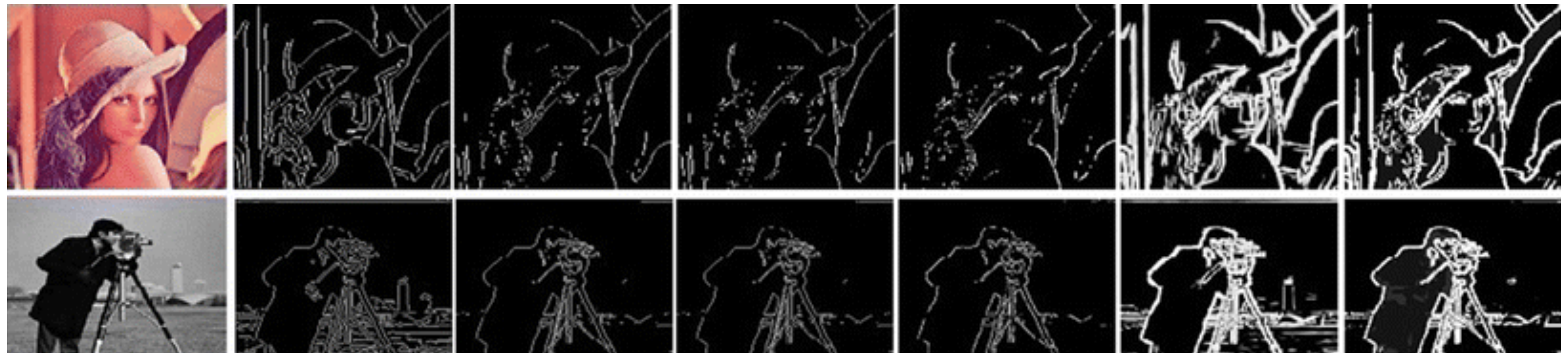
THE UNIVERSITY OF BRITISH COLUMBIA

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 (**October 7, 2024**)

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

— **Quiz 2** will be released **today**

— Lecture **videos** — stay tuned for some changes on Canvas

Today's "**fun**" Example: Colour Constancy

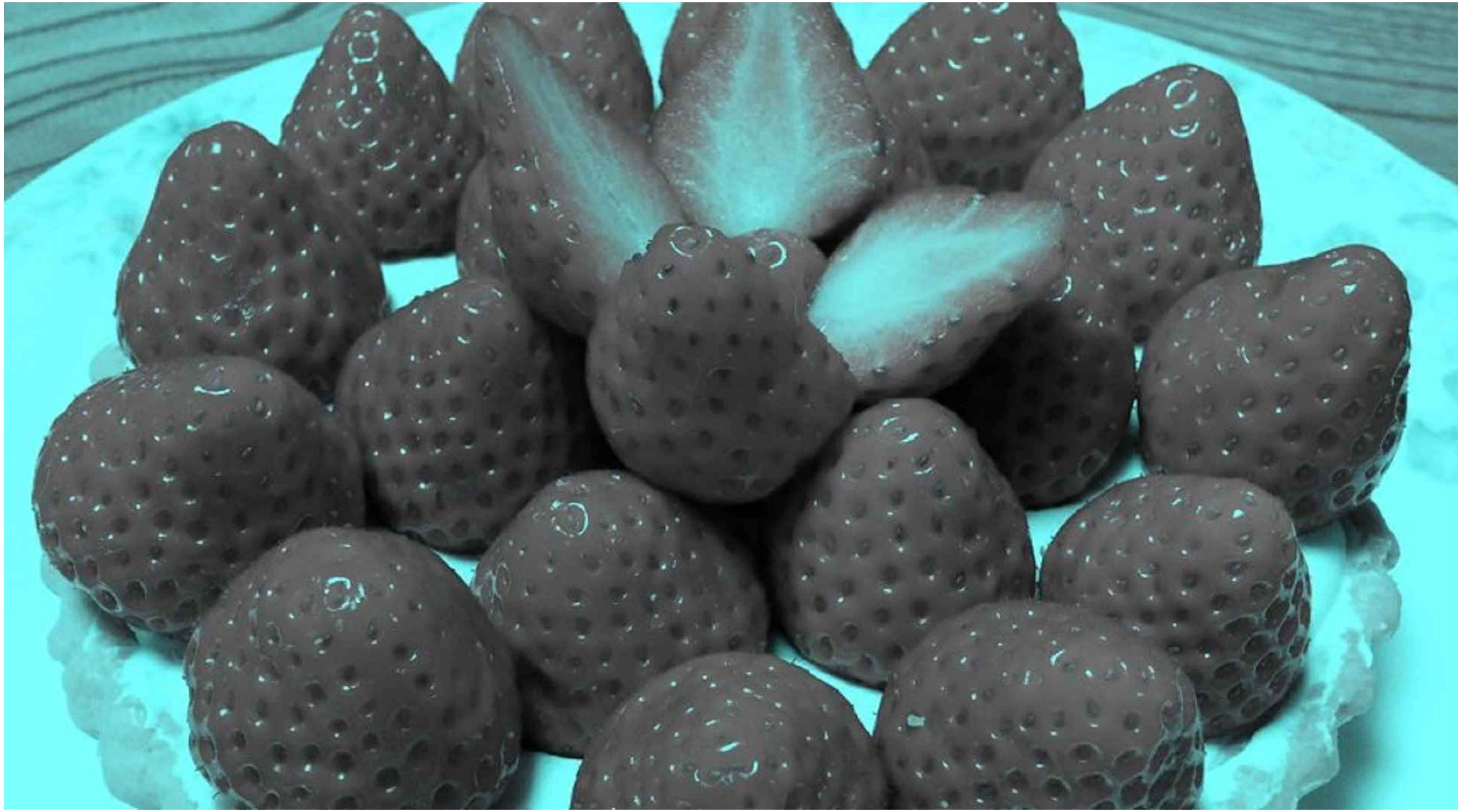


Image Credit: Akiyosha Kitoaka

Today's "**fun**" Example: Colour Constancy

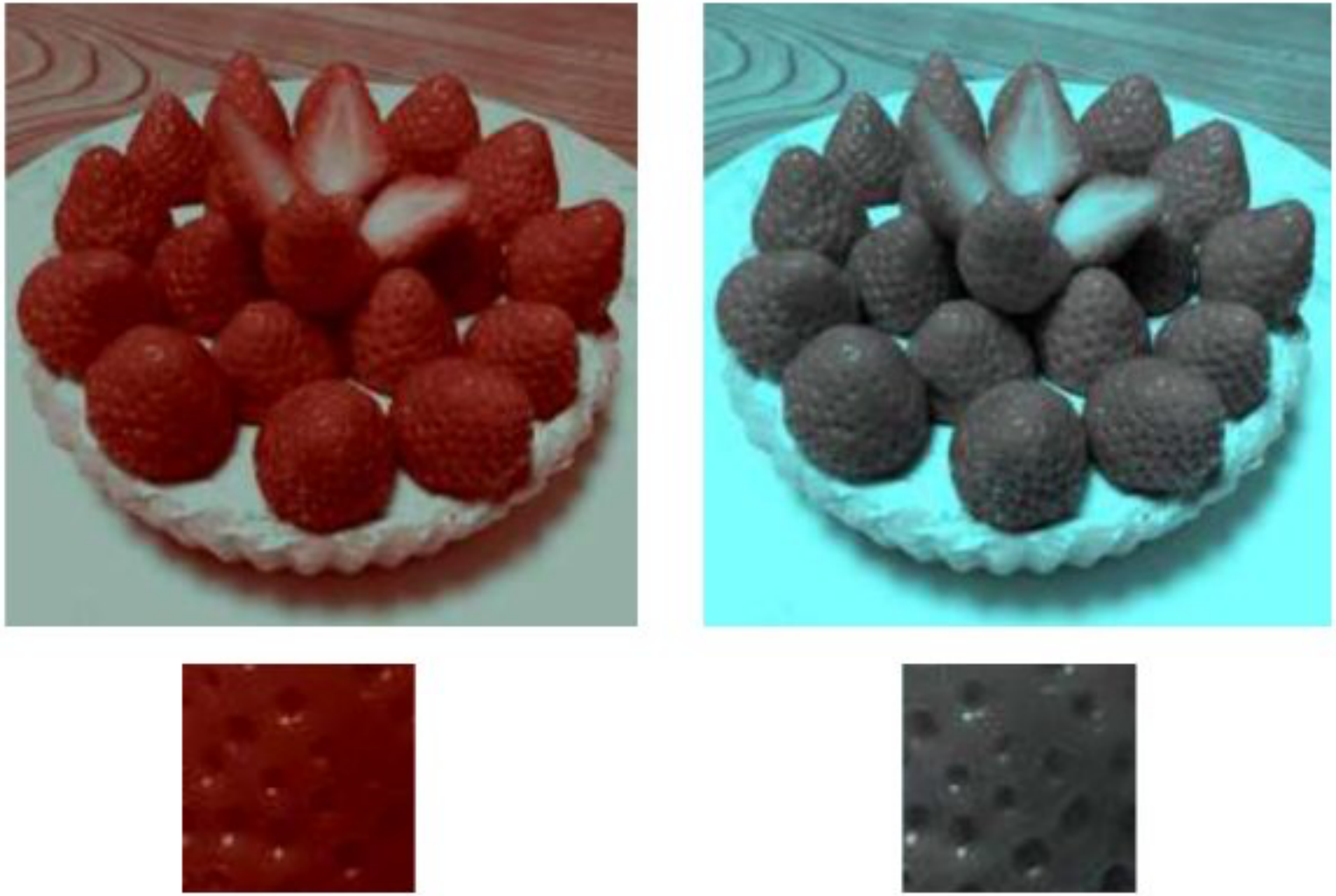


Image Credit: Akiyosha Kitoaka

Today's "**fun**" Example: Colour Constancy

— Some people see a white and gold dress.

— Some people see a blue and black dress.

— Some people see one interpretation and then switch to the other

https://www.nytimes.com/interactive/2015/02/28/science/white-or-blue-dress.html

Today's "**fun**" Example: Colour Constancy

— Some people see a white and gold dress.

— Some people see a blue and black dress.

— Some people see one interpretation and then switch to the other

Two pieces of the dress

Average colors

The basic pattern of the dress

https://www.nytimes.com/interactive/2015/02/28/science/white-or-blue-dress.html

Today's "**fun**" Example: Colour Constancy

IS THE DRESS IN SHADOW?

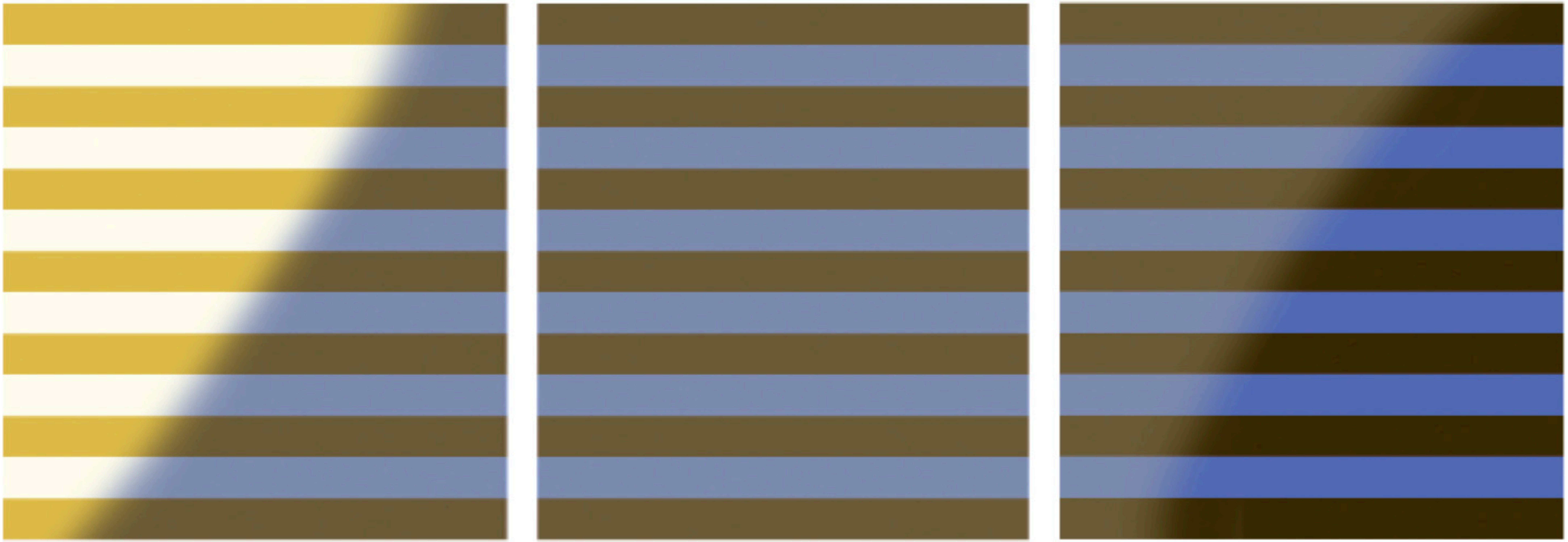
If you think the dress is in shadow, your brain may remove the blue cast and perceive the dress as being white and gold.

THE DRESS IN THE PHOTO

If the photograph showed more of the room, or if skin tones were visible, there might have been more clues about the ambient light.

IS THE DRESS IN BRIGHT LIGHT?

If you think the dress is being washed out by bright light, your brain may perceive the dress as a darker blue and black.



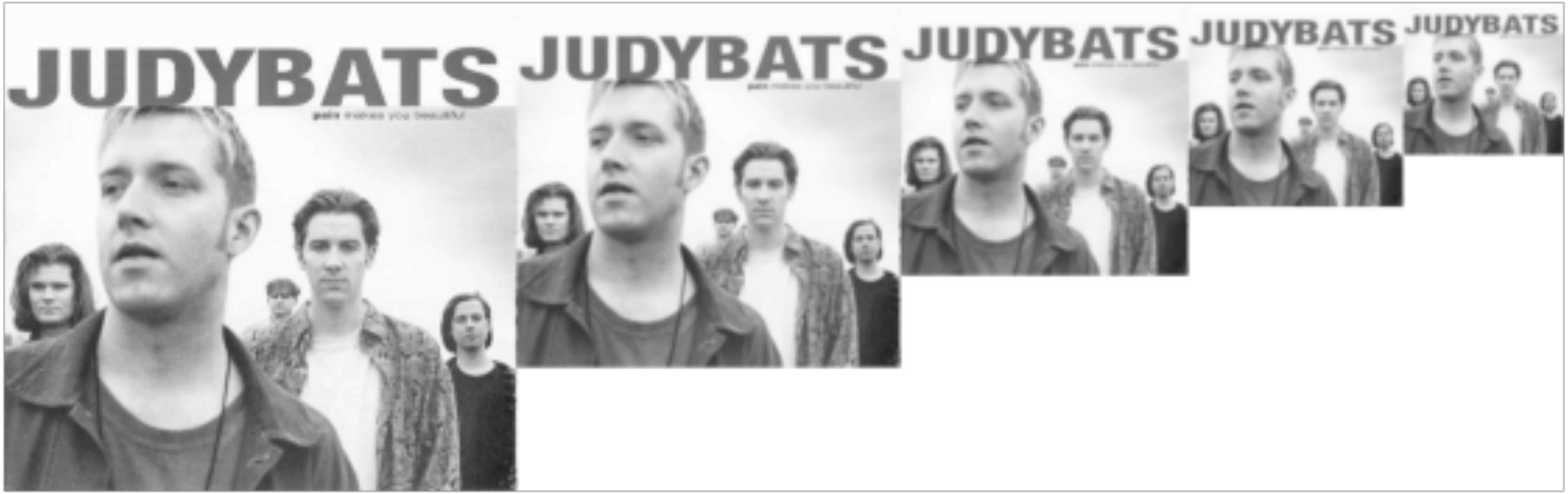
https://www.nytimes.com/interactive/2015/02/28/science/white-or-blue-dress.html

Today's "**fun**" Example: Colour Constancy

https://www.nytimes.com/interactive/2015/02/28/science/white-or-blue-dress.html

Lecture 8: Re-cap **Multi-Scale** Template Matching

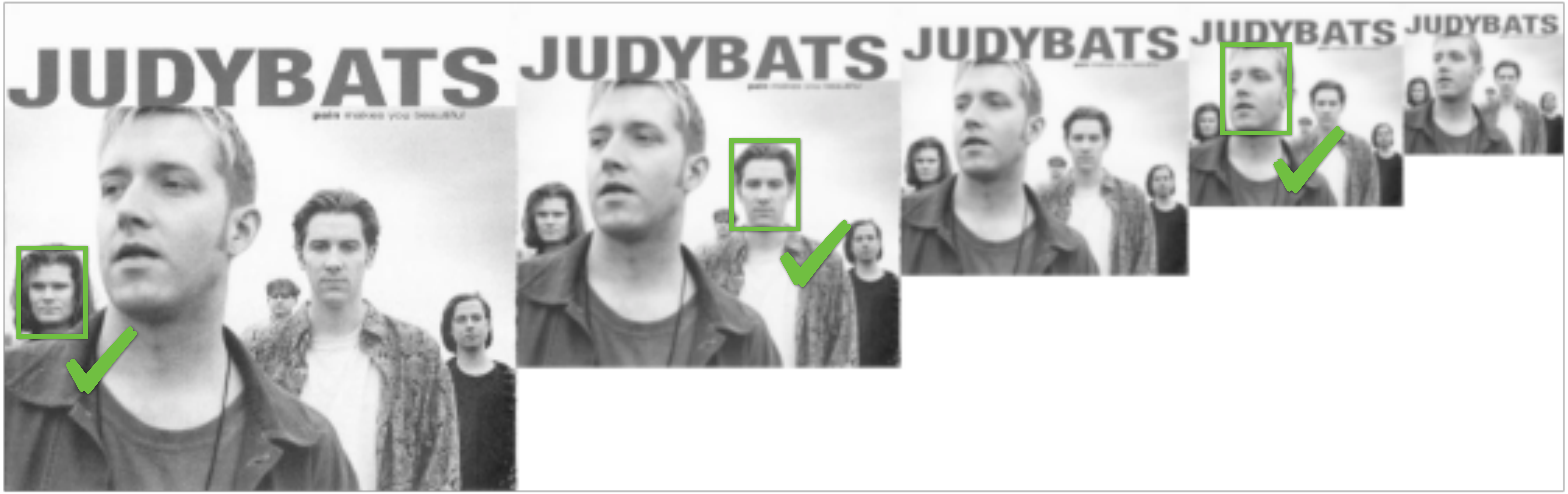
Correlation with a **fixed-sized image** only detects faces at **specific scales**



= Template

Lecture 8: Re-cap **Multi-Scale** Template Matching

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



 = Template

Lecture 8: Re-cap **Scaled Representations**

Gaussian Pyramid

— Each level represents a **low-pass** filtered image at a different scale

— Generated by successive Gaussian blurring and downsampling

— Useful for image resizing, sampling

Laplacian Pyramid

— Each level is a **band-pass** image at a different scale

— Generated by differences between successive levels of a Gaussian Pyramid

— Used for pyramid blending, feature extraction etc.

From Template Matching to **Local Feature Detection**

Image Template

Test **Image**

From Template Matching to **Local Feature Detection**

Image Template

Edge Template

Test **Image**

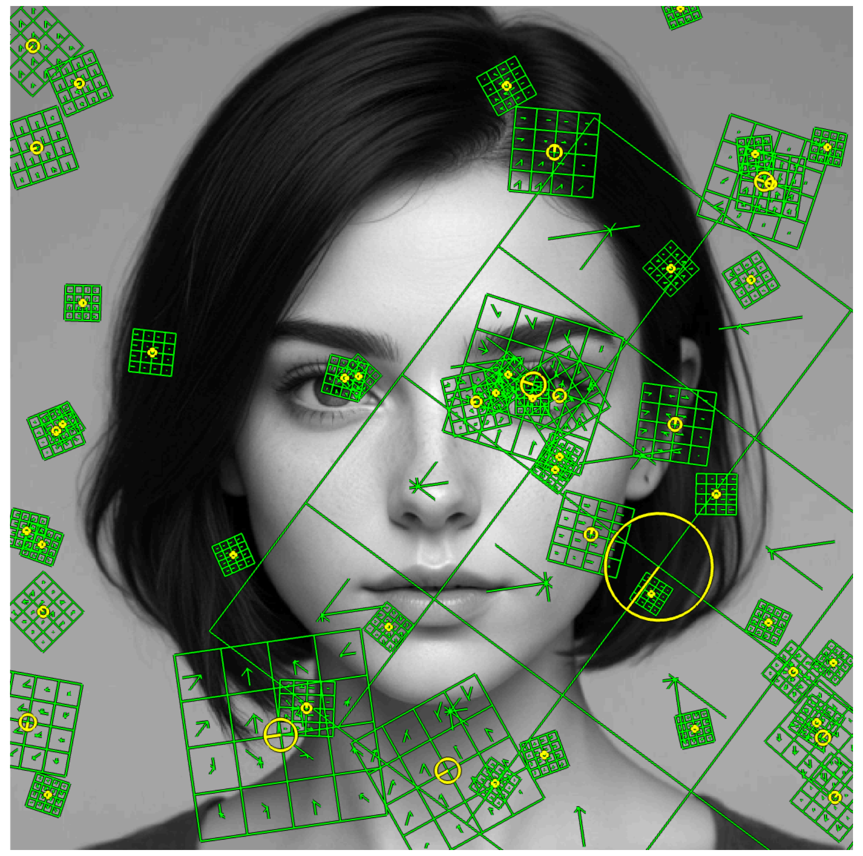
Test **Edge** Image

From Template Matching to **Local Feature Detection**

Image Template

Edge Template

Interest Points



Test **Image**

Test **Edge** Image

From Template Matching to **Local Feature Detection**

- Move from global template matching to **local template matching**
- Local template matching also called local **feature detection**
- Obvious local features to detect are **edges** and **corners**

Edge Detection

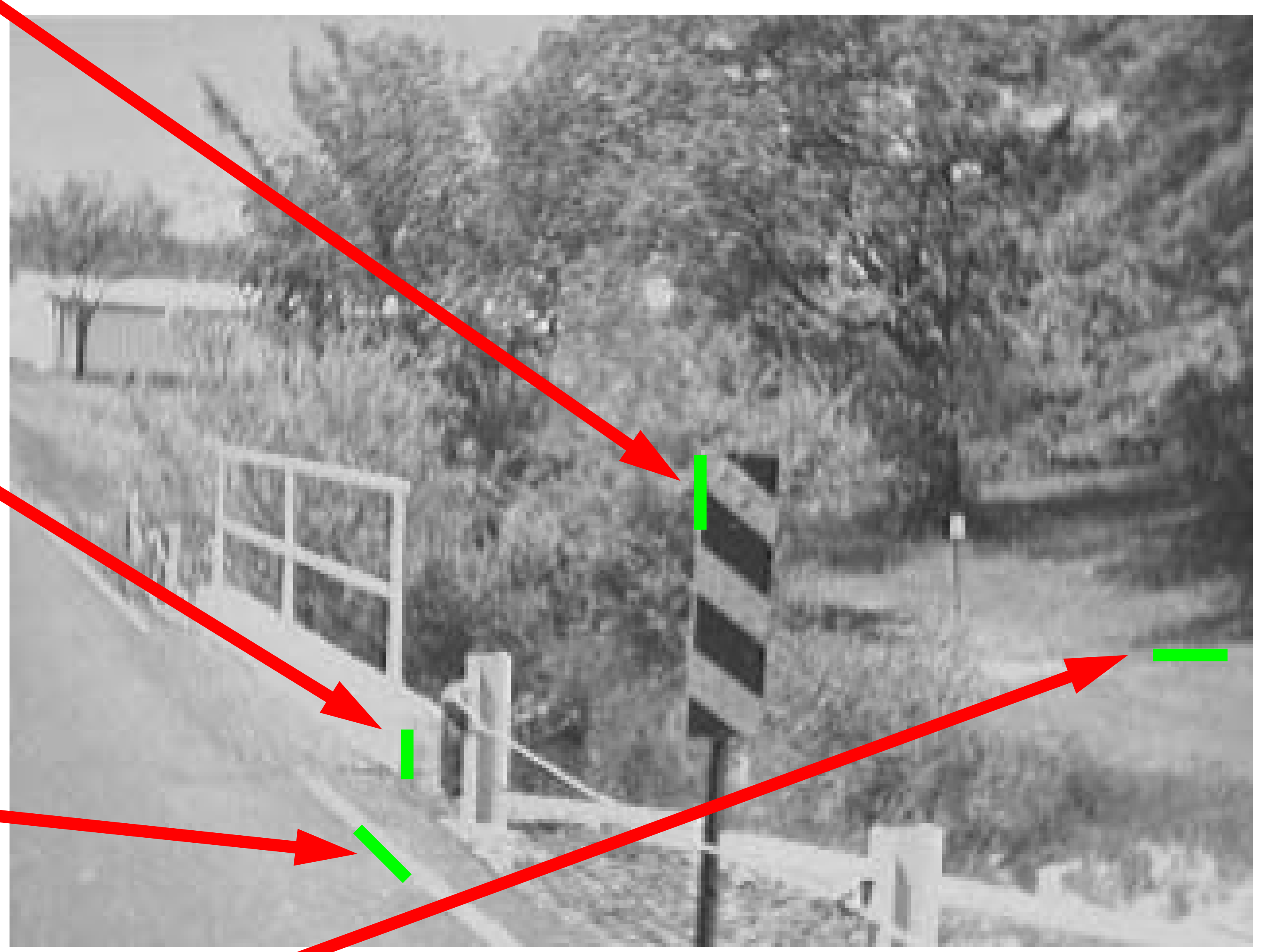
Goal: Identify sudden changes in image intensity

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

Estimating **Derivatives**

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

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

Estimating **Derivatives**

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

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

A (discrete) approximation is (**forward** difference):

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

Estimating **Derivatives**

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

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

A (discrete) approximation is (**backward** difference):

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

Estimating **Derivatives**

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

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

A (discrete) approximation is (**central** difference):

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

Estimating **Derivatives** (most common)

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

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

A (discrete) approximation is (**forward** difference):

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

Estimating **Derivatives** (most common)

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

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

A (discrete) approximation is (**forward** difference):

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

$$\boxed{-1} \boxed{1}$$

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

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

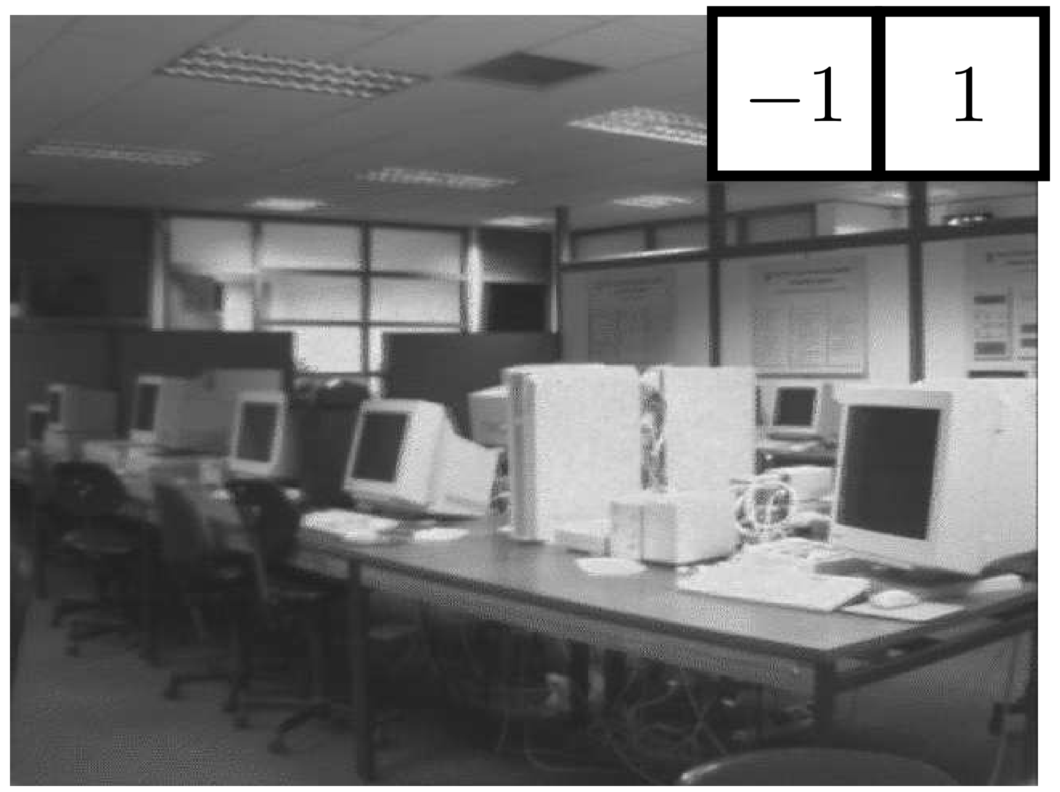
-1	1

from **right**

convolution

1	-1

Estimating **Derivatives**



"**forward** difference" implemented as

correlation

-1	1

from **left**

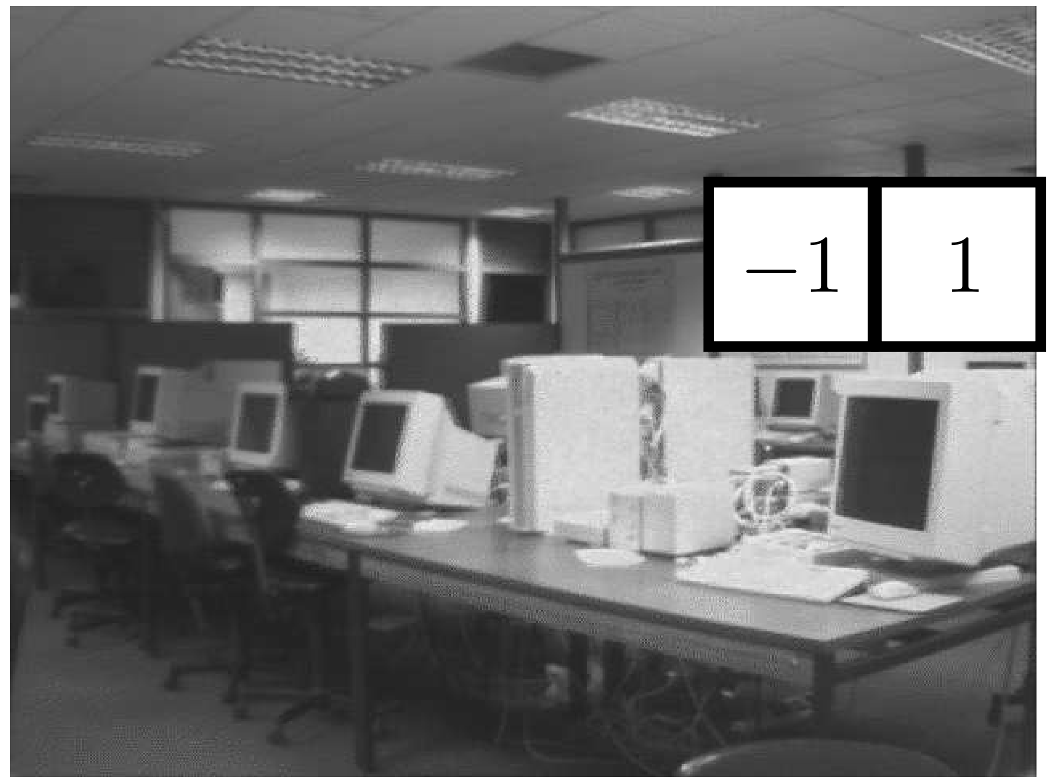
"**backward** difference" implemented as

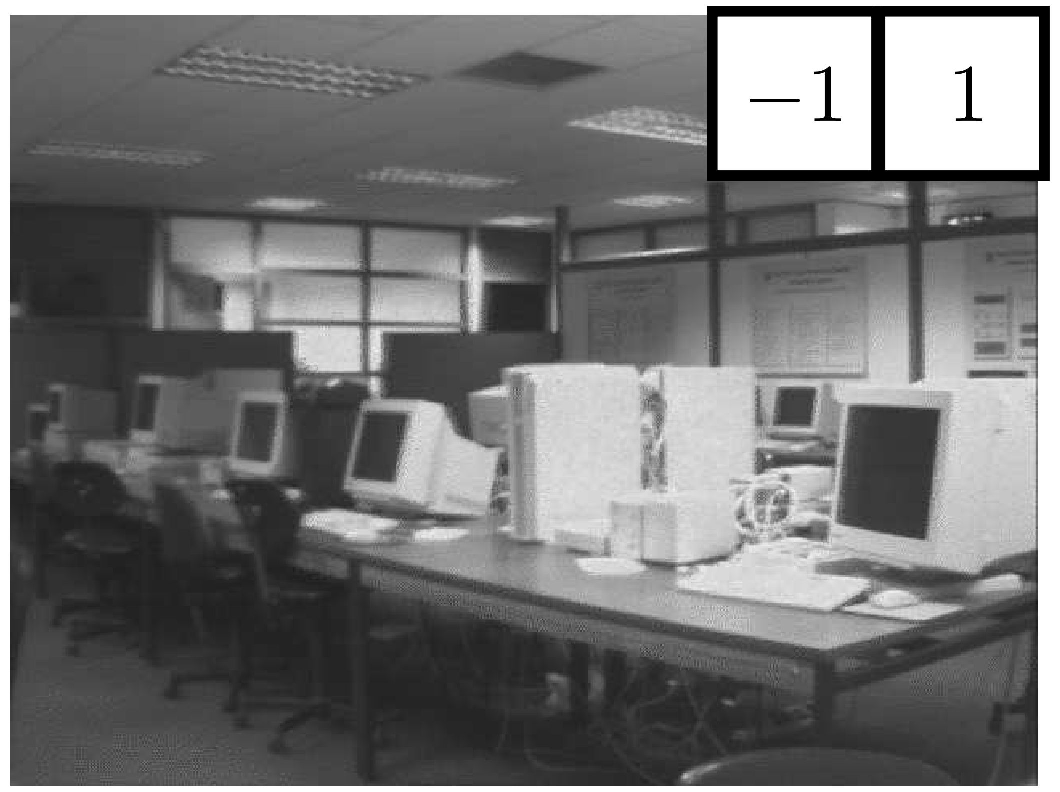
correlation

-1	1

from **right**

Estimating **Derivatives**





"**forward** difference" implemented as

correlation

$$\begin{array}{|c|c|}\hline -1 & 1 \\ \hline \end{array}$$

from **left**

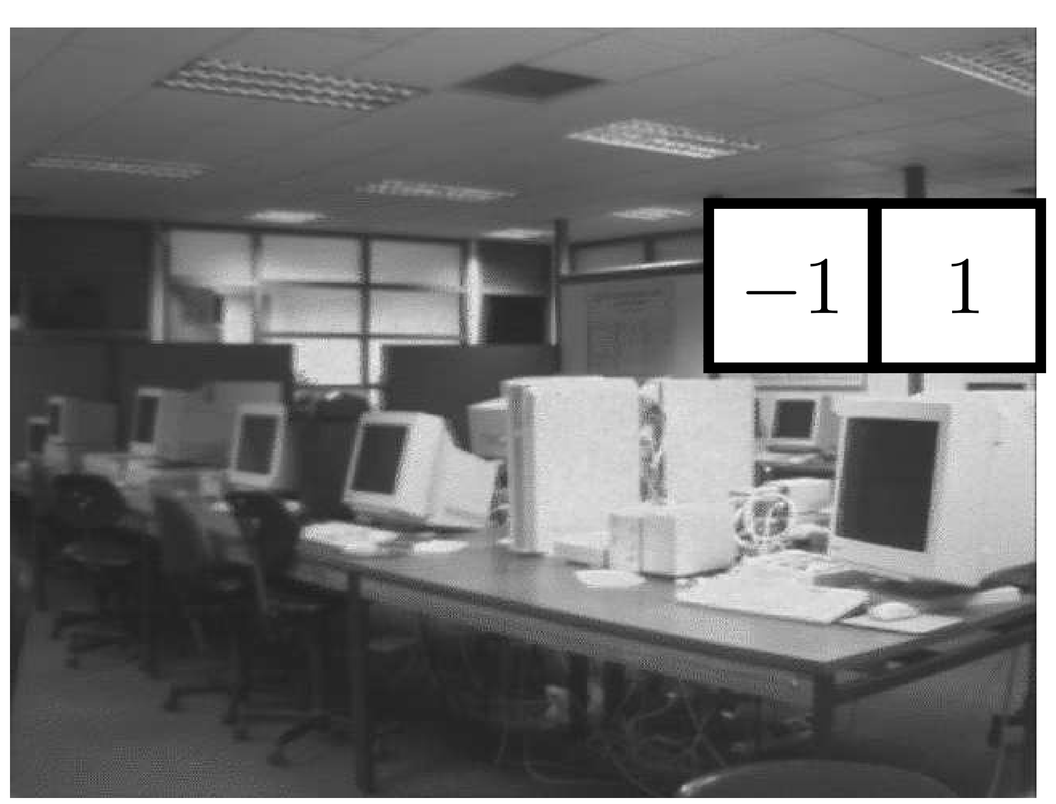
"**backward** difference" implemented as

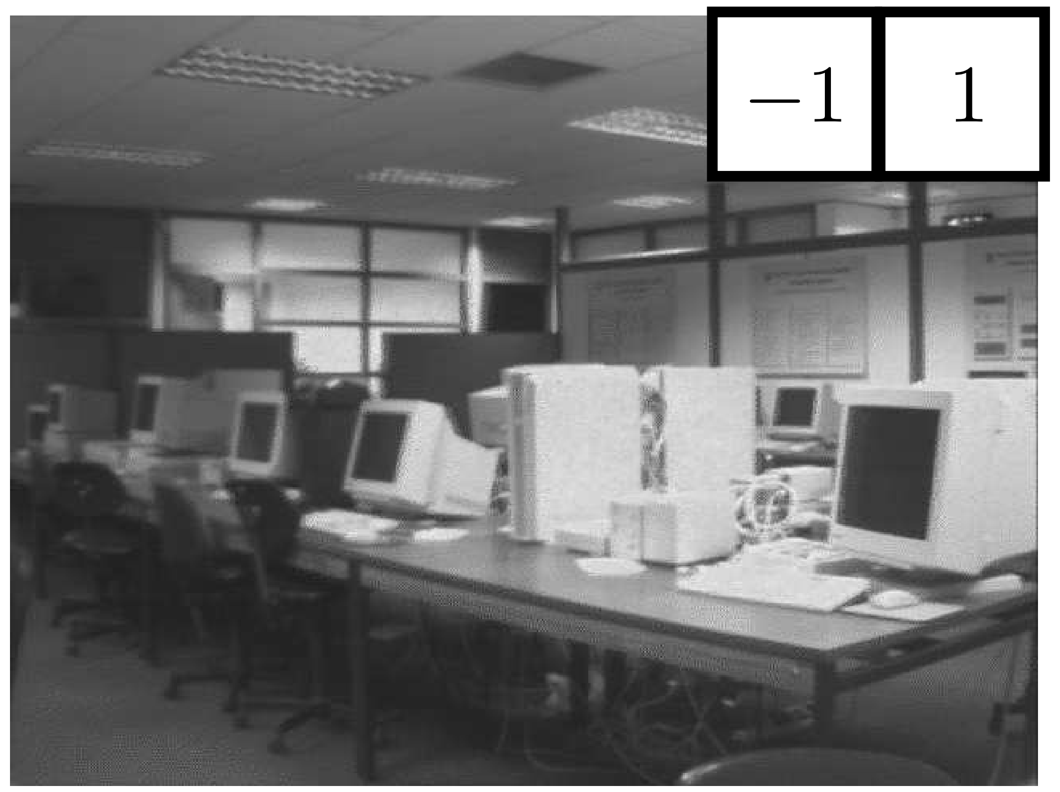
correlation

$$\begin{array}{|c|c|}\hline -1 & 1 \\ \hline \end{array}$$

from **right**

Estimating **Derivatives**





"**forward** difference" implemented as

correlation

-1	1

from **left**

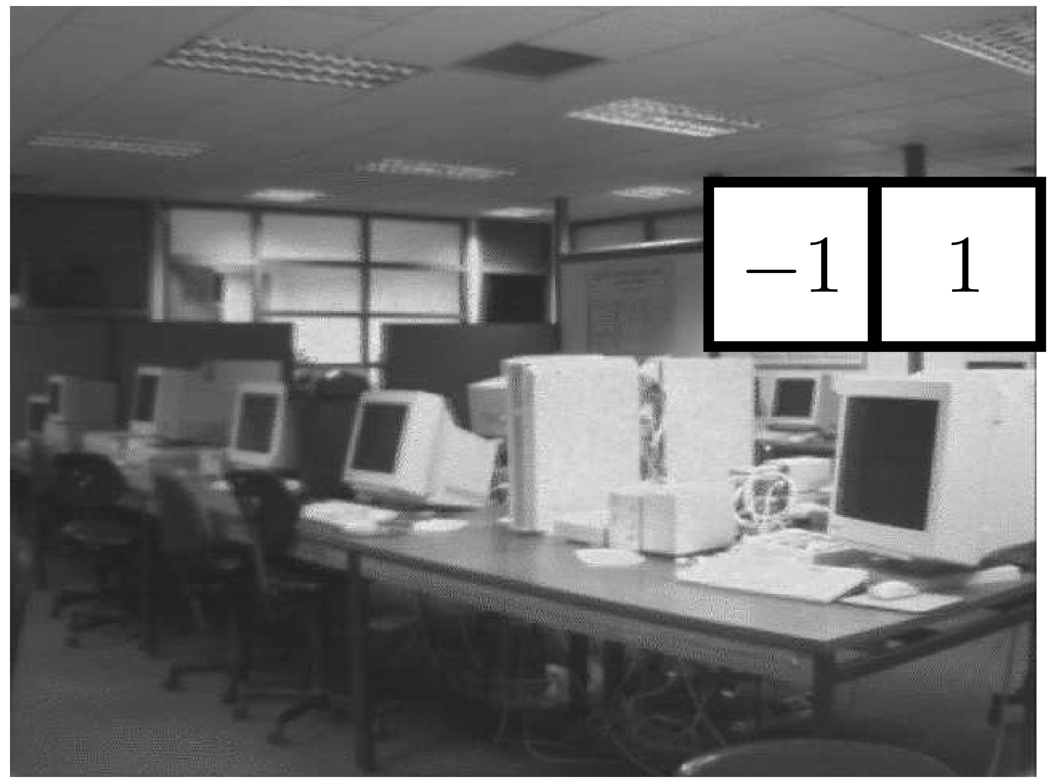
"**backward** difference" implemented as

correlation

-1	1

from **right**

Estimating **Derivatives**



"**forward** difference" implemented as

correlation

-1	1

from **left**

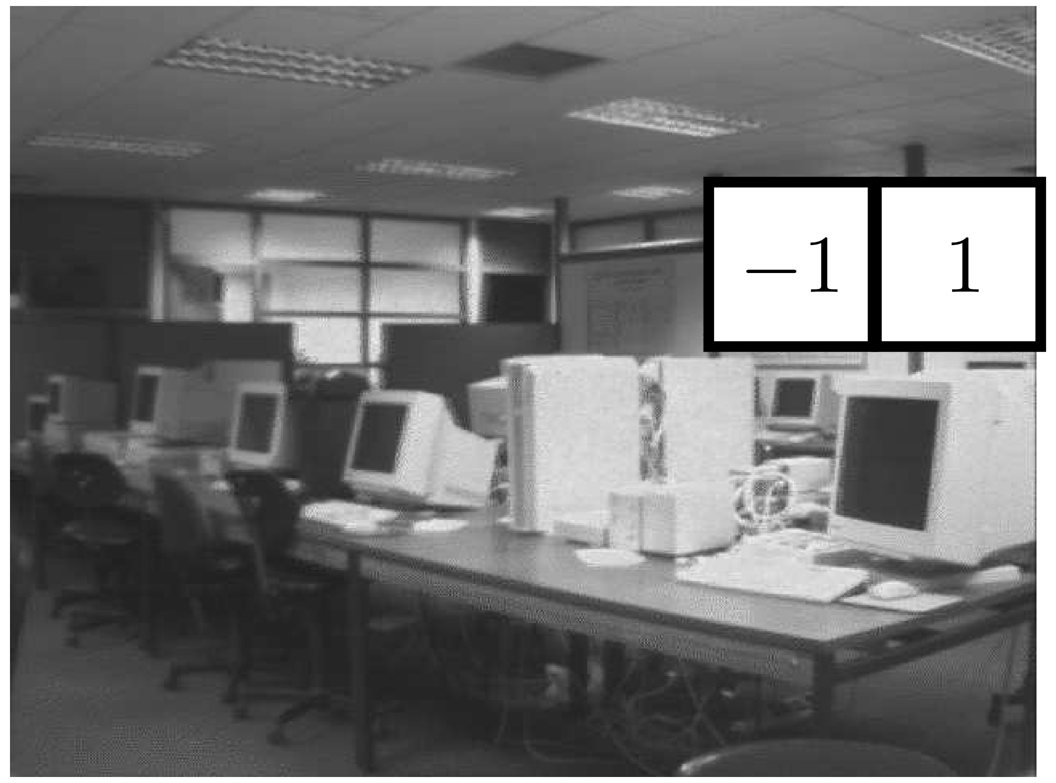
"**backward** difference" implemented as

correlation

-1	1

from **right**

Estimating **Derivatives**



"**forward** difference" implemented as

correlation

$$\begin{array}{|c|c|}\hline -1 & 1 \\ \hline \end{array}$$

from **left**

"**backward** difference" implemented as

correlation

$$\begin{array}{|c|c|}\hline -1 & 1 \\ \hline \end{array}$$

from **right**

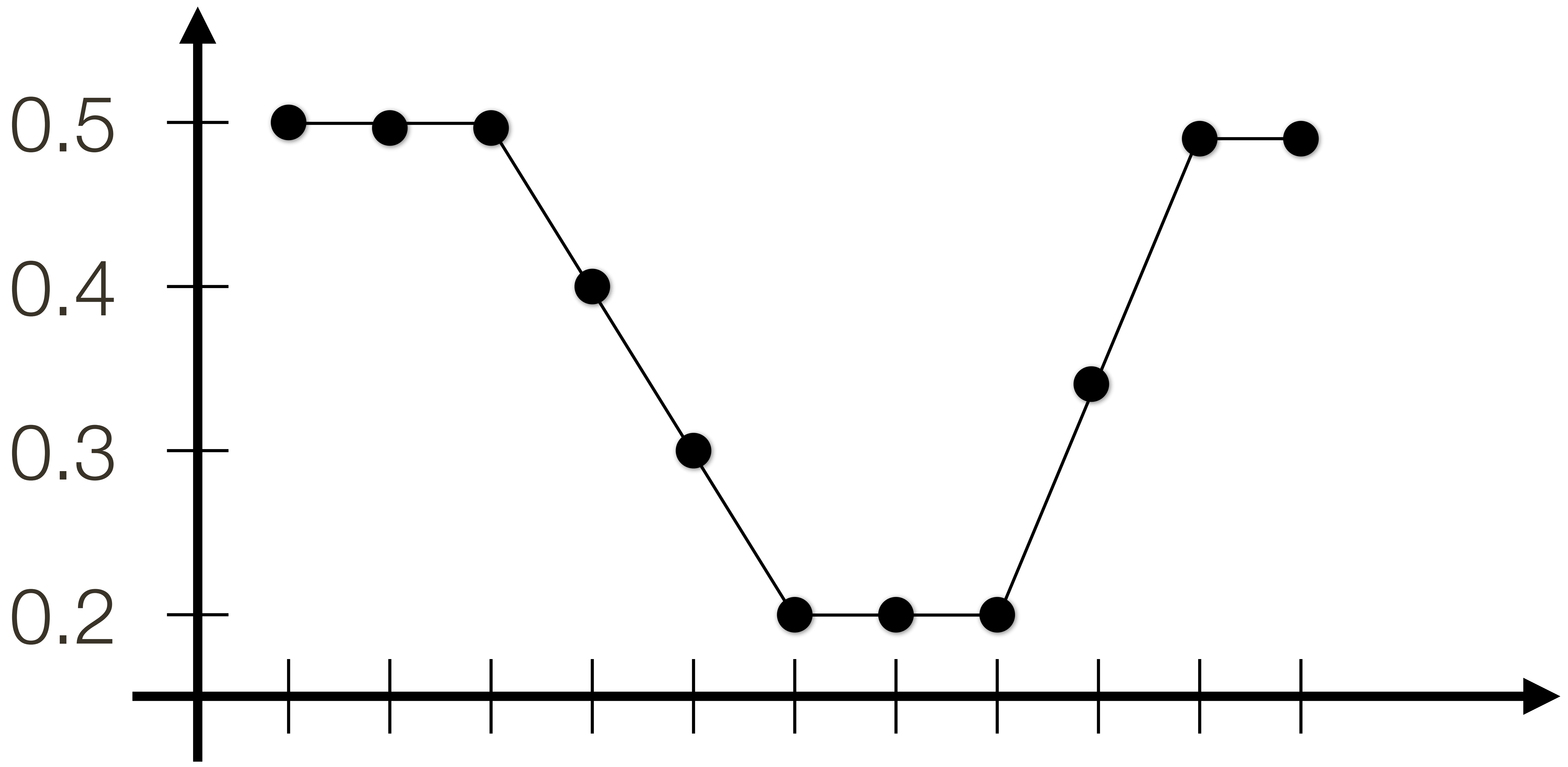
Estimating **Derivatives**

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

-1
1

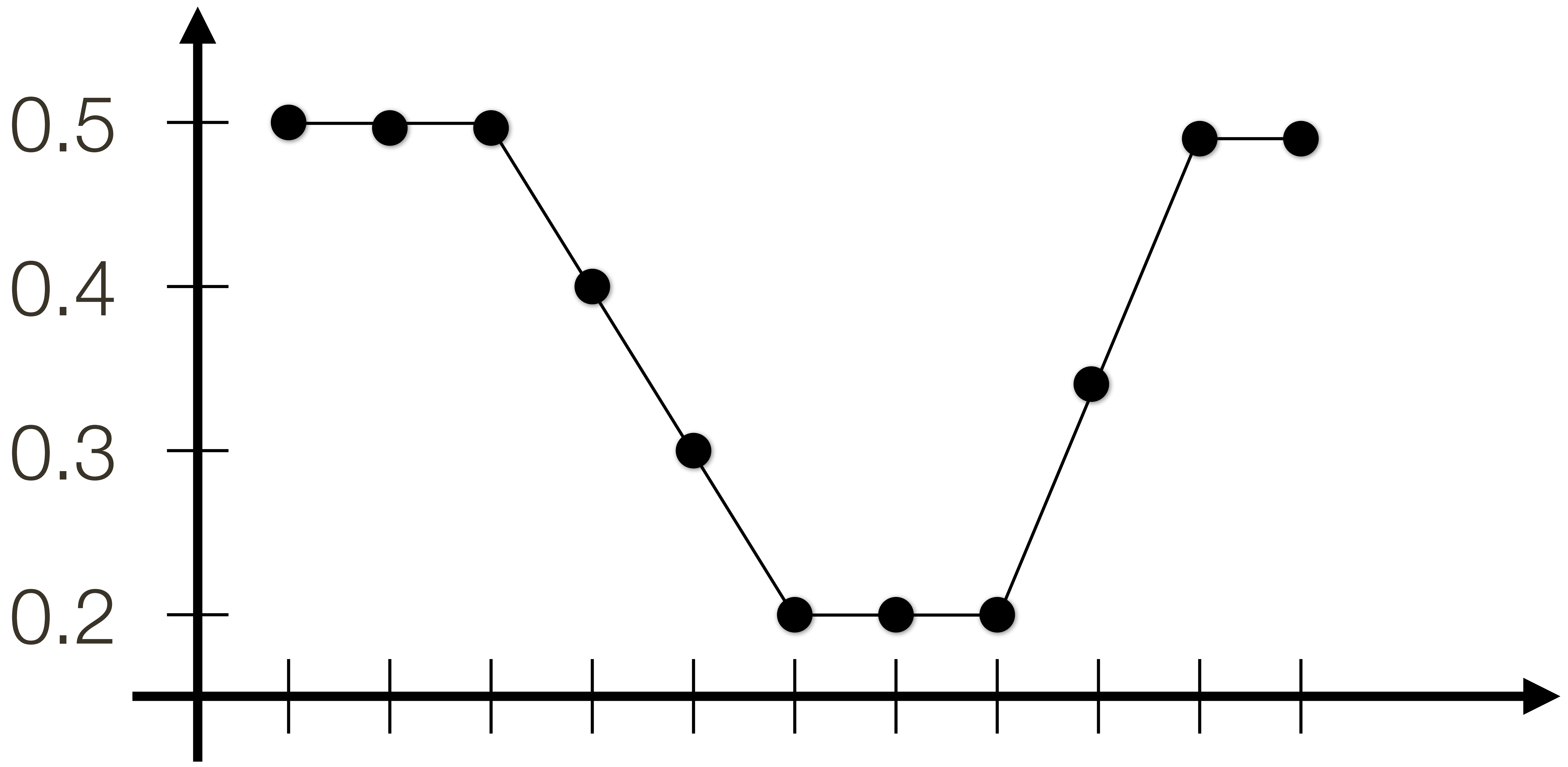
Example 1D

-1	1



Example 1D

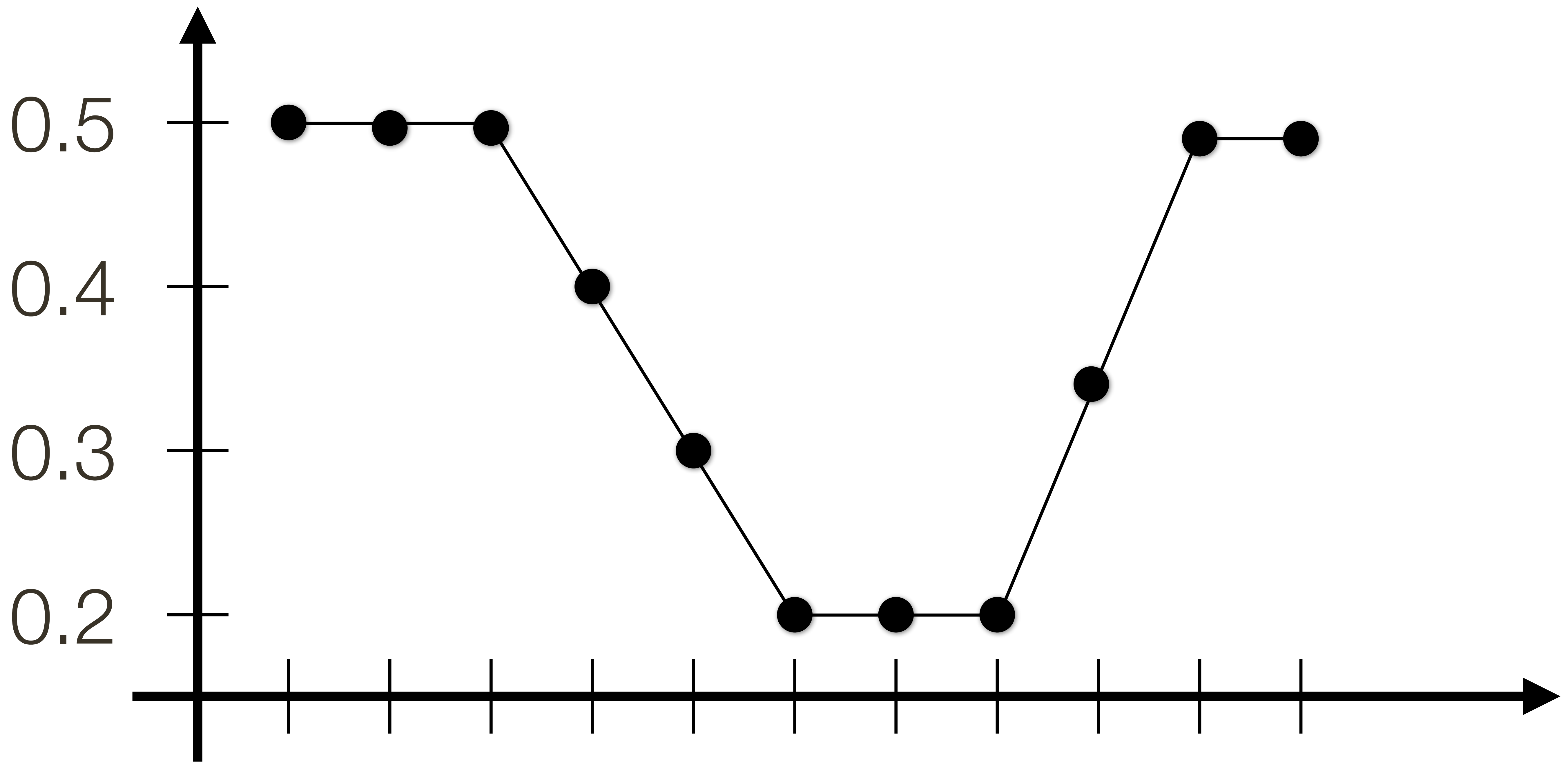
-1	1



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

-1	1

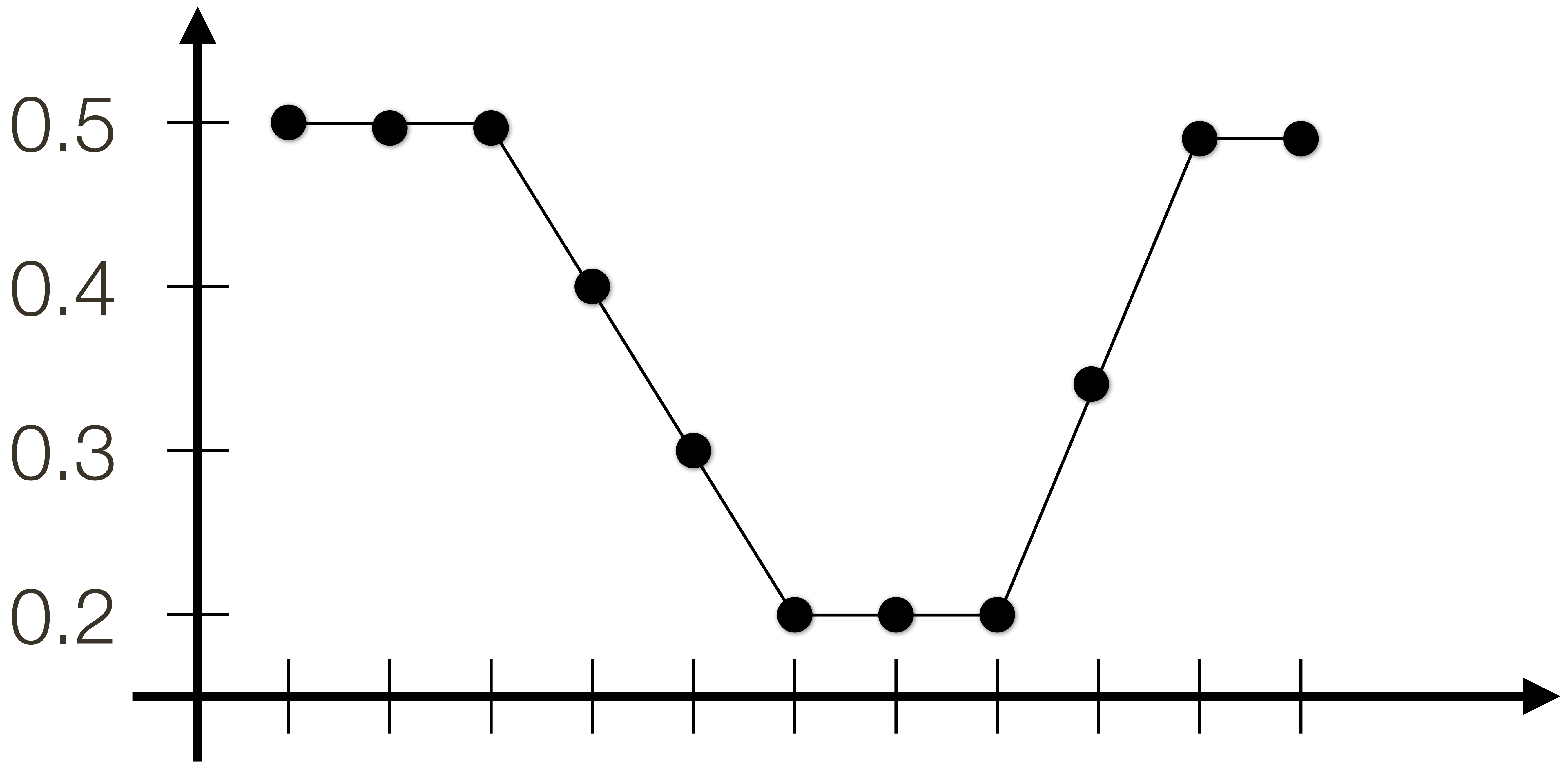


Signal	0.5	0.5	0.5	0.4	0.3	0.2	0.2	0.2	0.35	0.5	0.5

Derivative

Example 1D

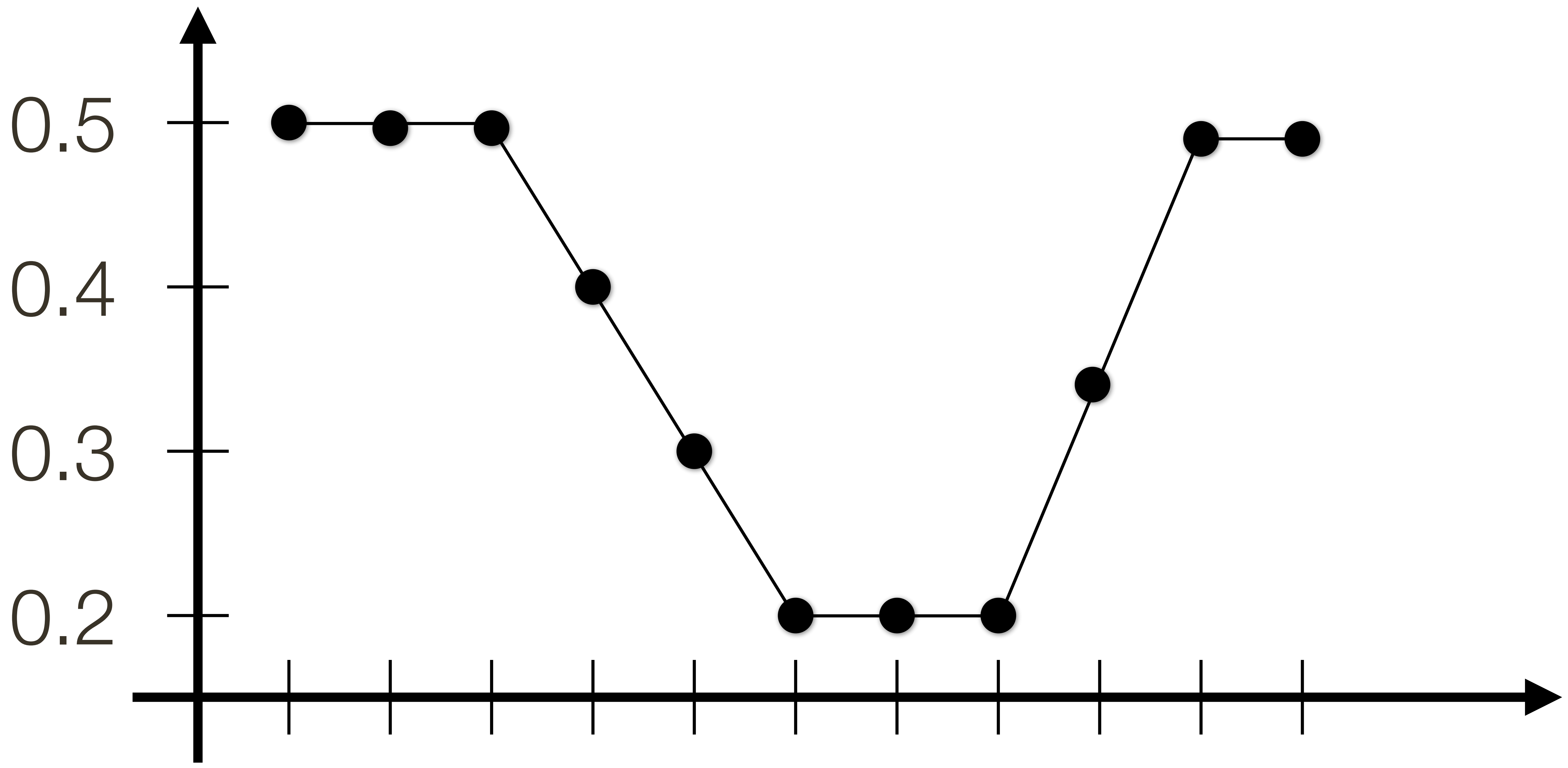
-1	1



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

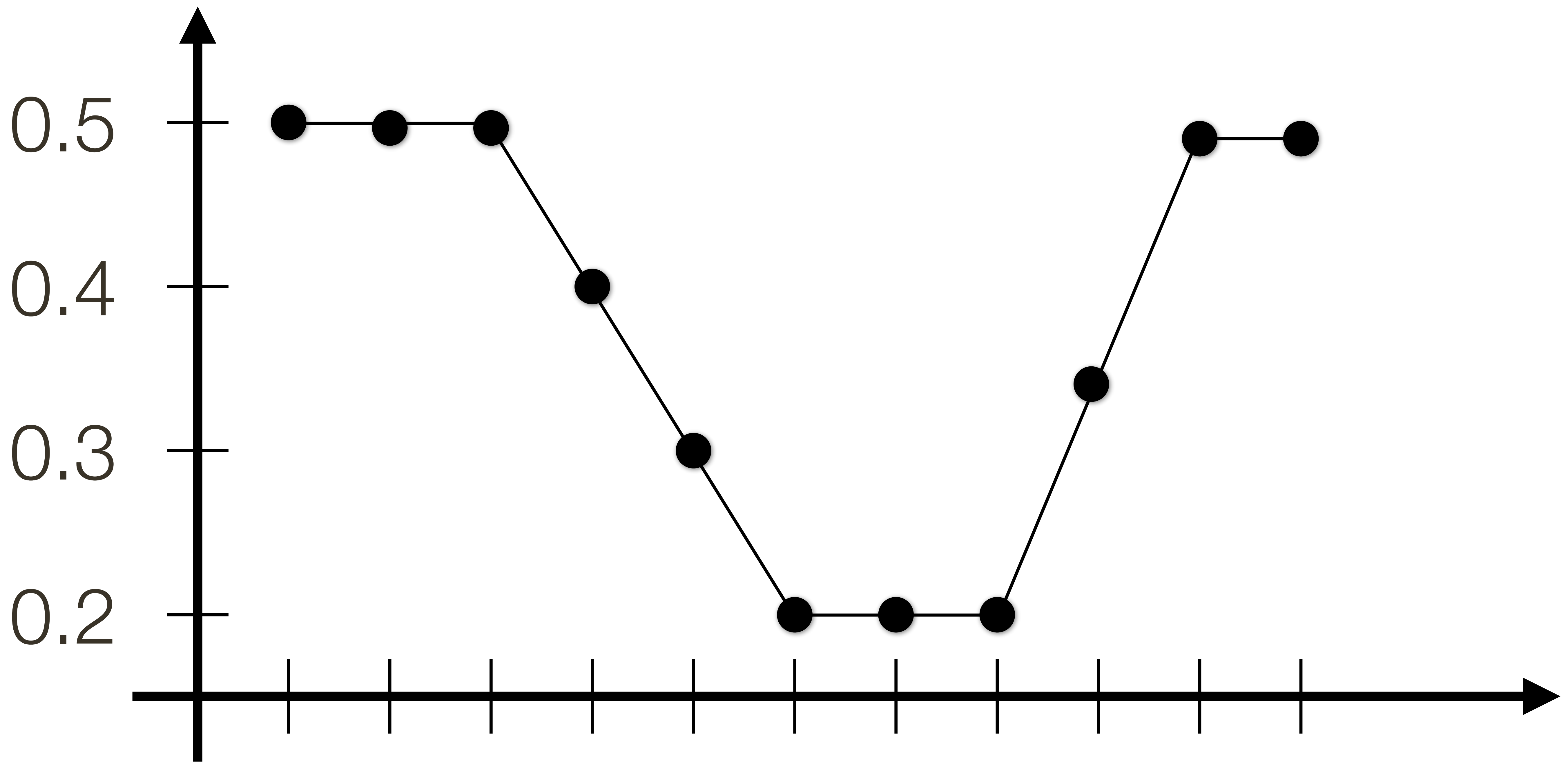
-1	1



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

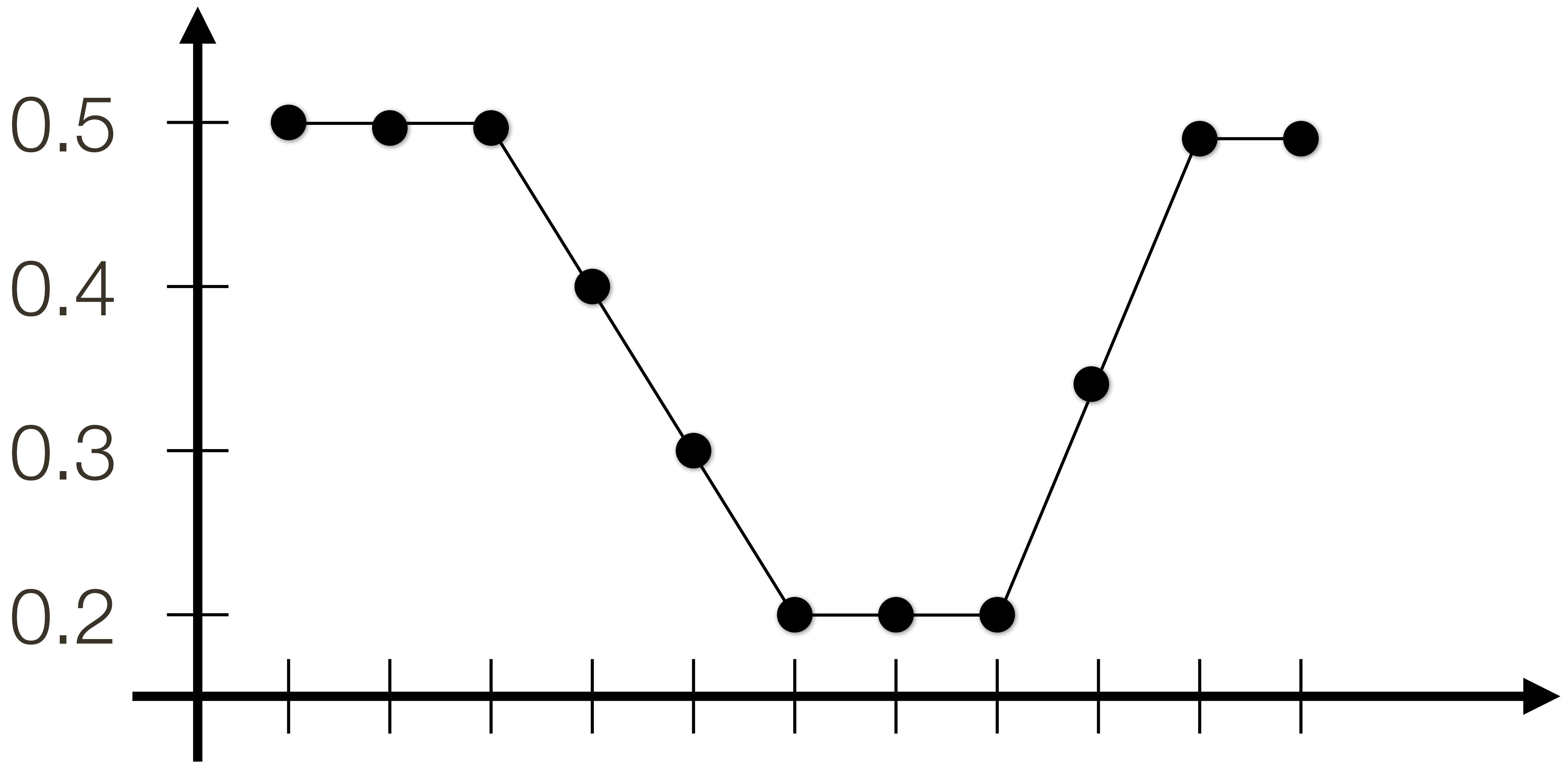
-1	1



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

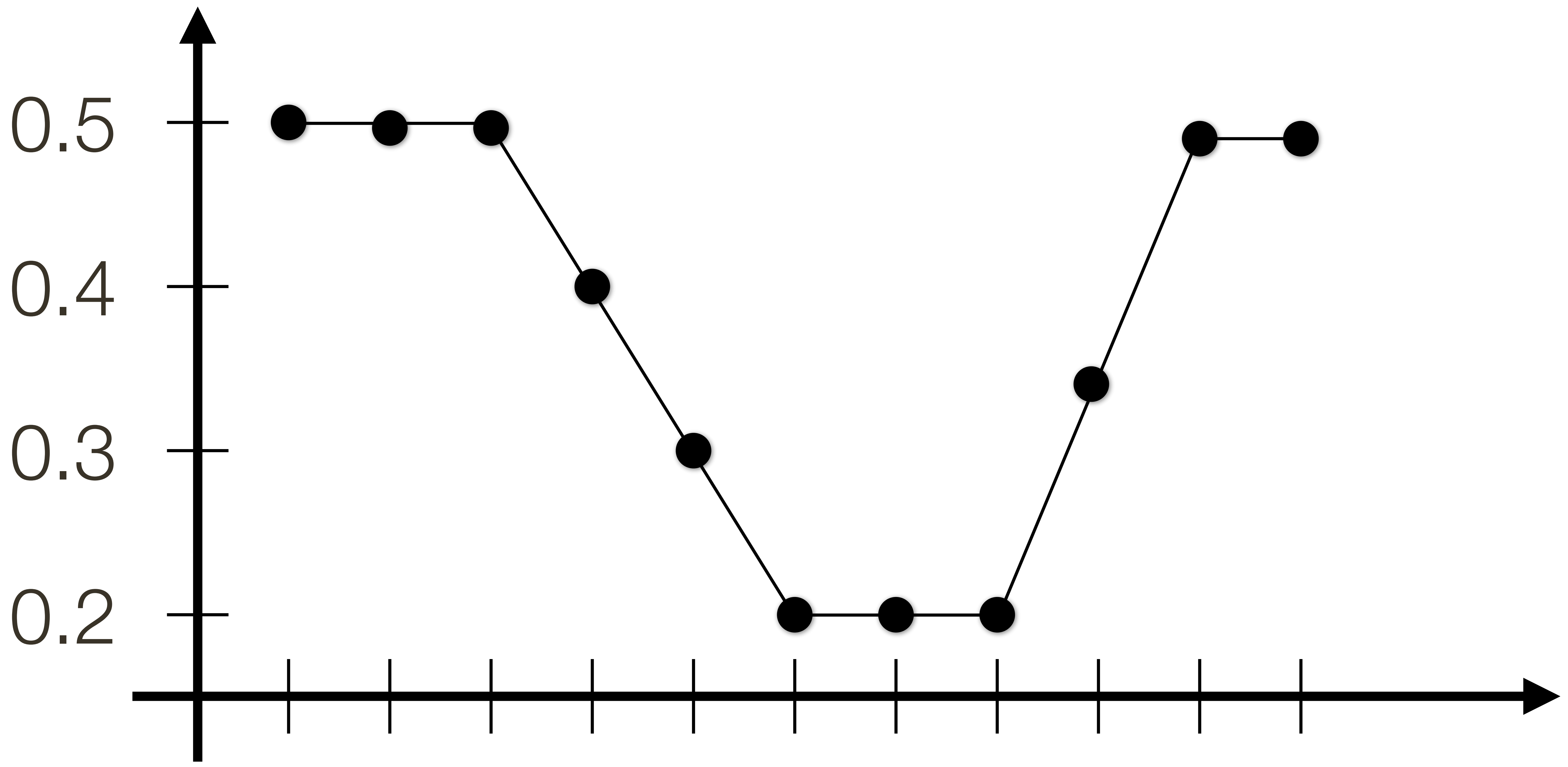
-1	1



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

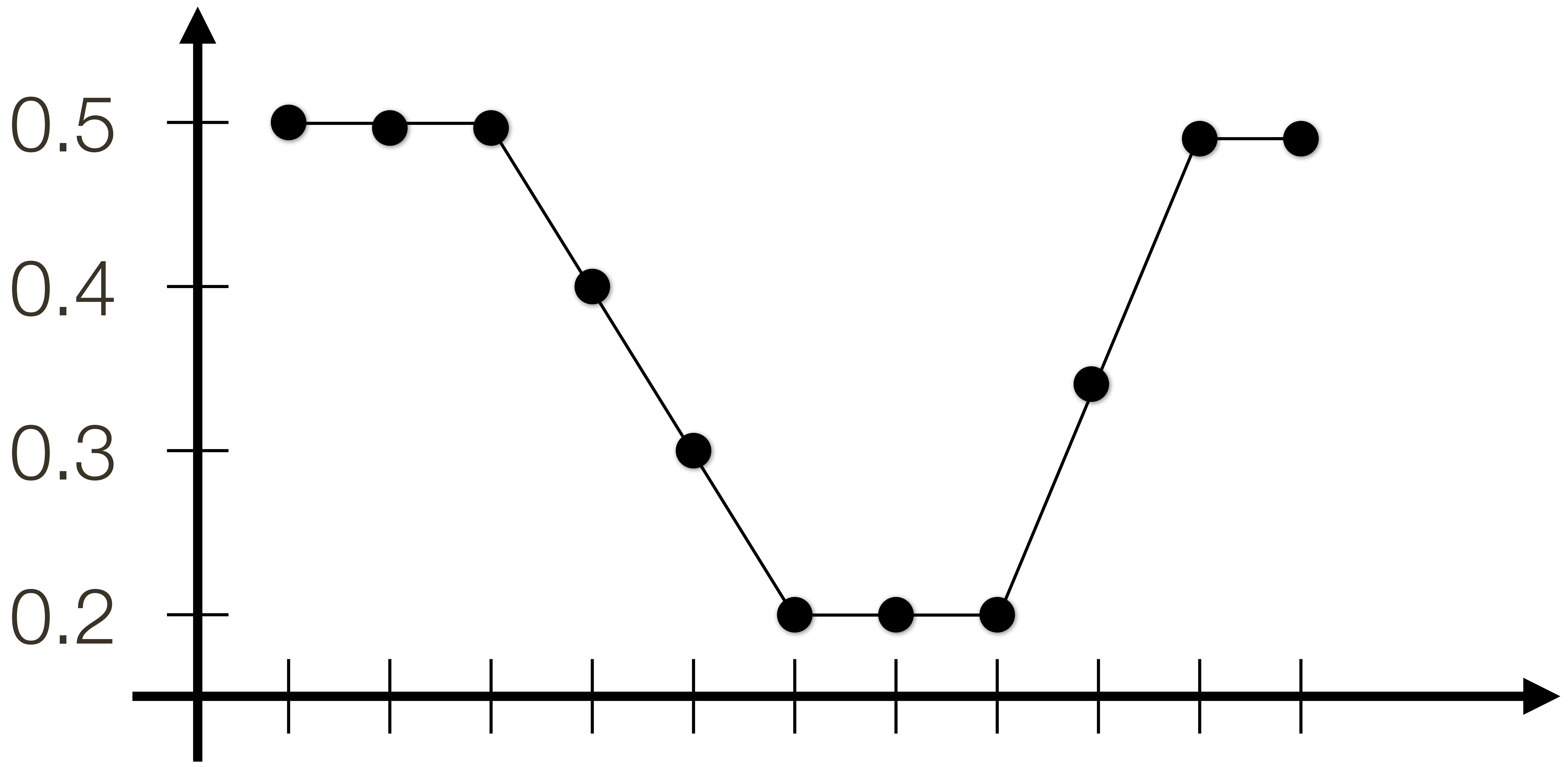
-1	1



Signal	0.5	0.5	$\boxed{0.5}$	$\boxed{0.4}$	0.3	0.2	0.2	0.2	0.35	0.5	0.5
Derivative	0.0	0.0	-0.1								

Example 1D

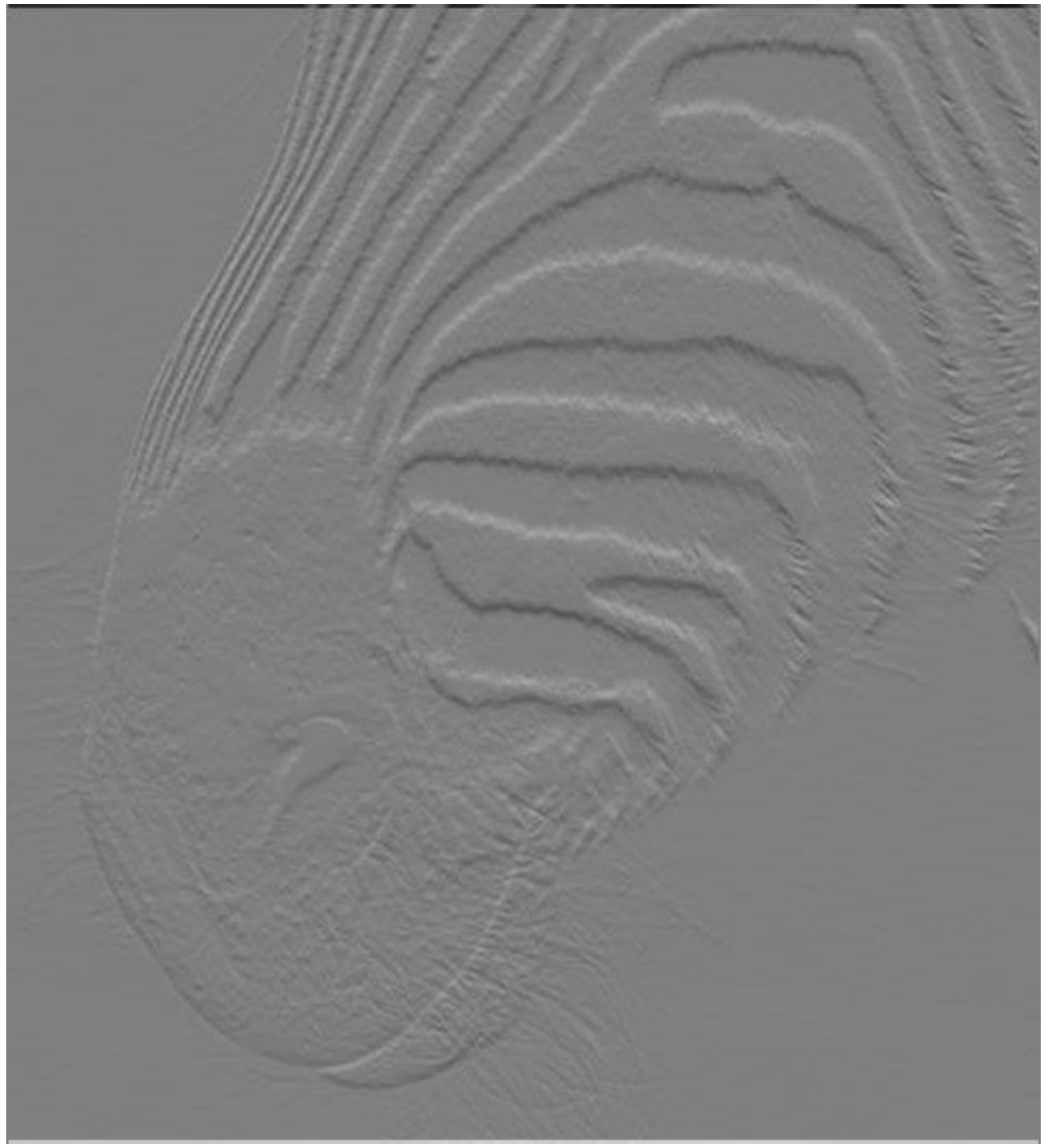
-1	1



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**

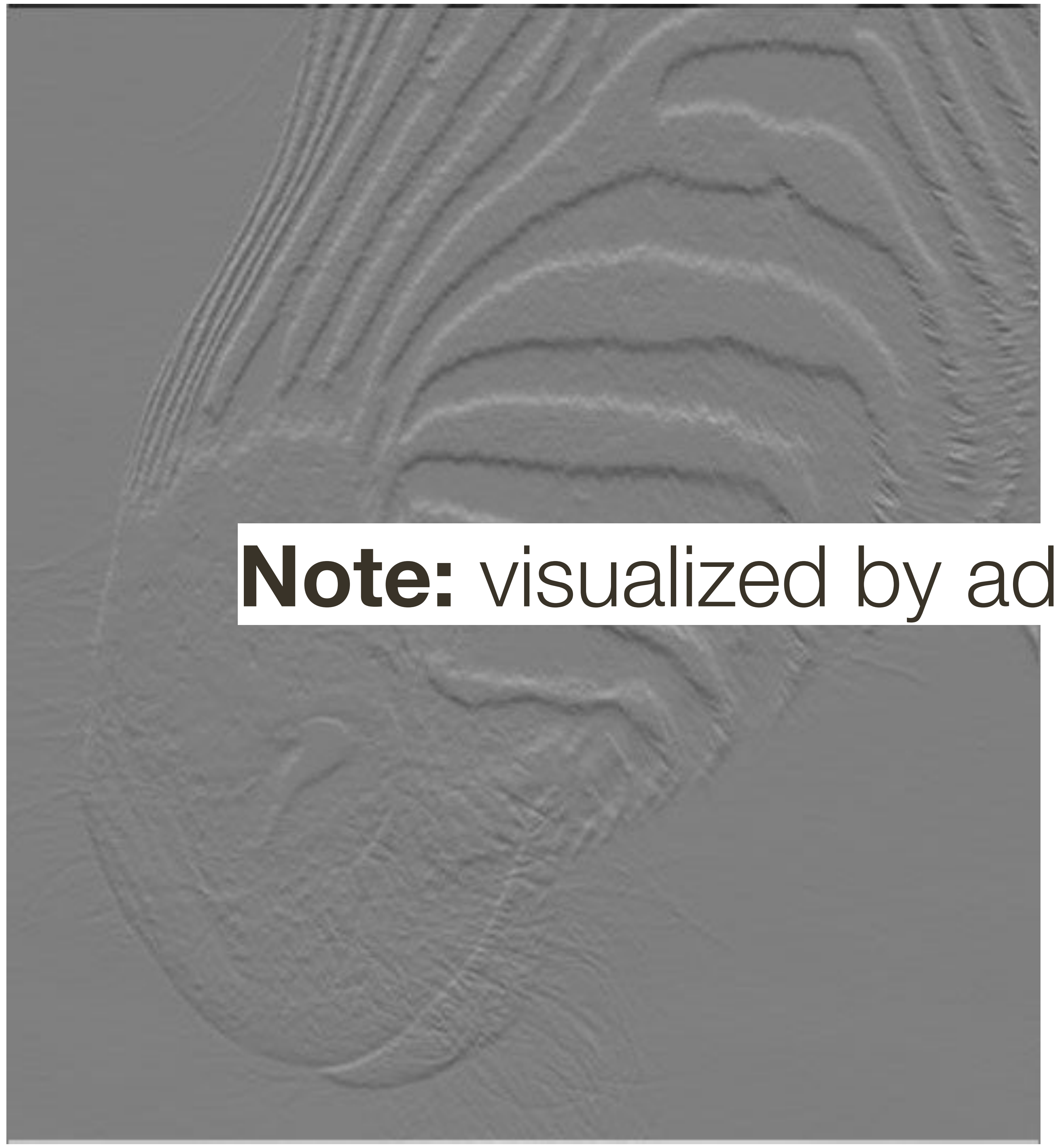
Derivative in Y (i.e., vertical) direction



Forsyth & Ponce (1st ed.) Figure 7.4 (top left & top middle)

Estimating **Derivatives**

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

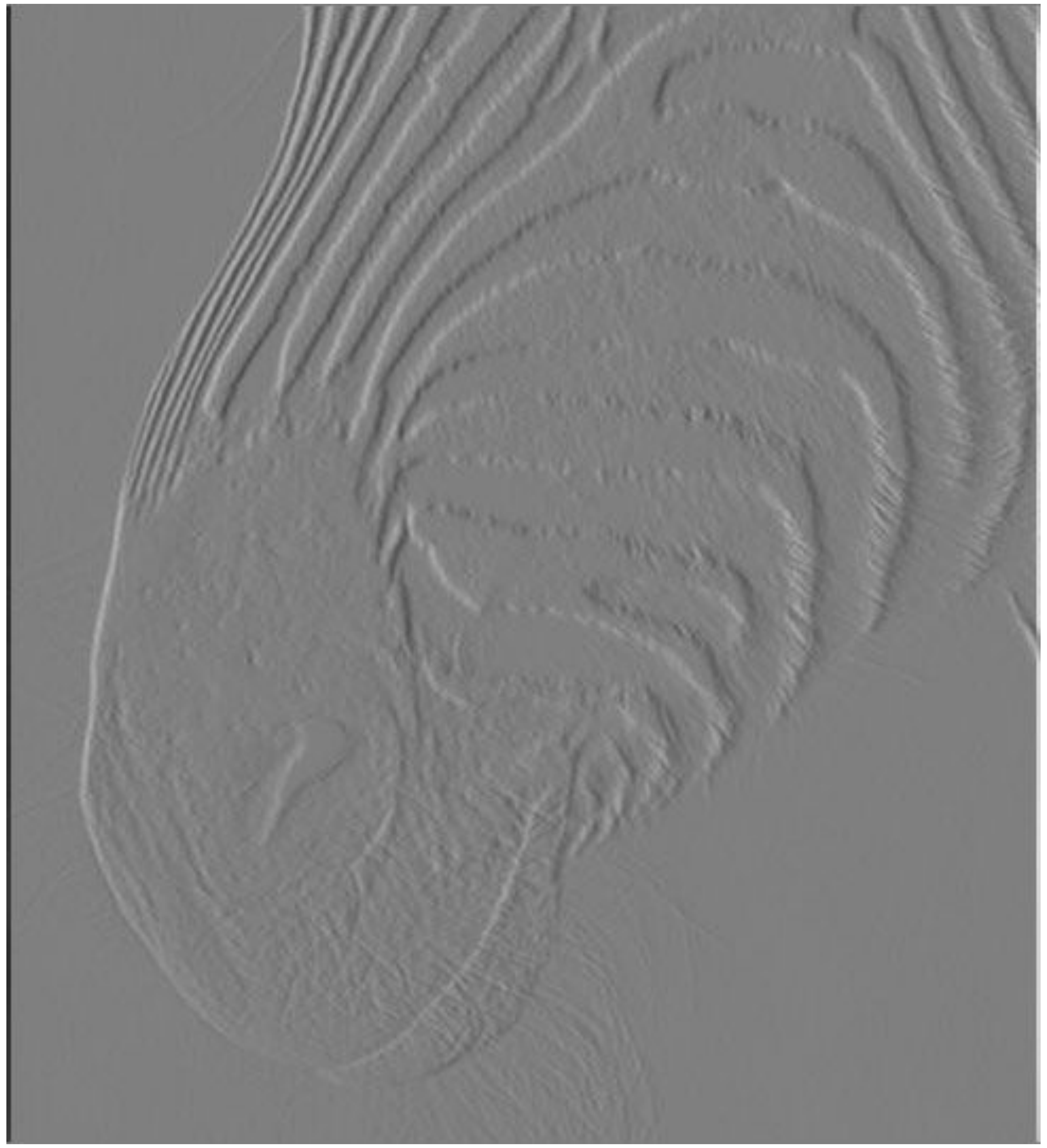


Note: visualized by adding 0.5/128

Forsyth & Ponce (1st ed.) Figure 7.4 (top left & top middle)

Estimating **Derivatives**

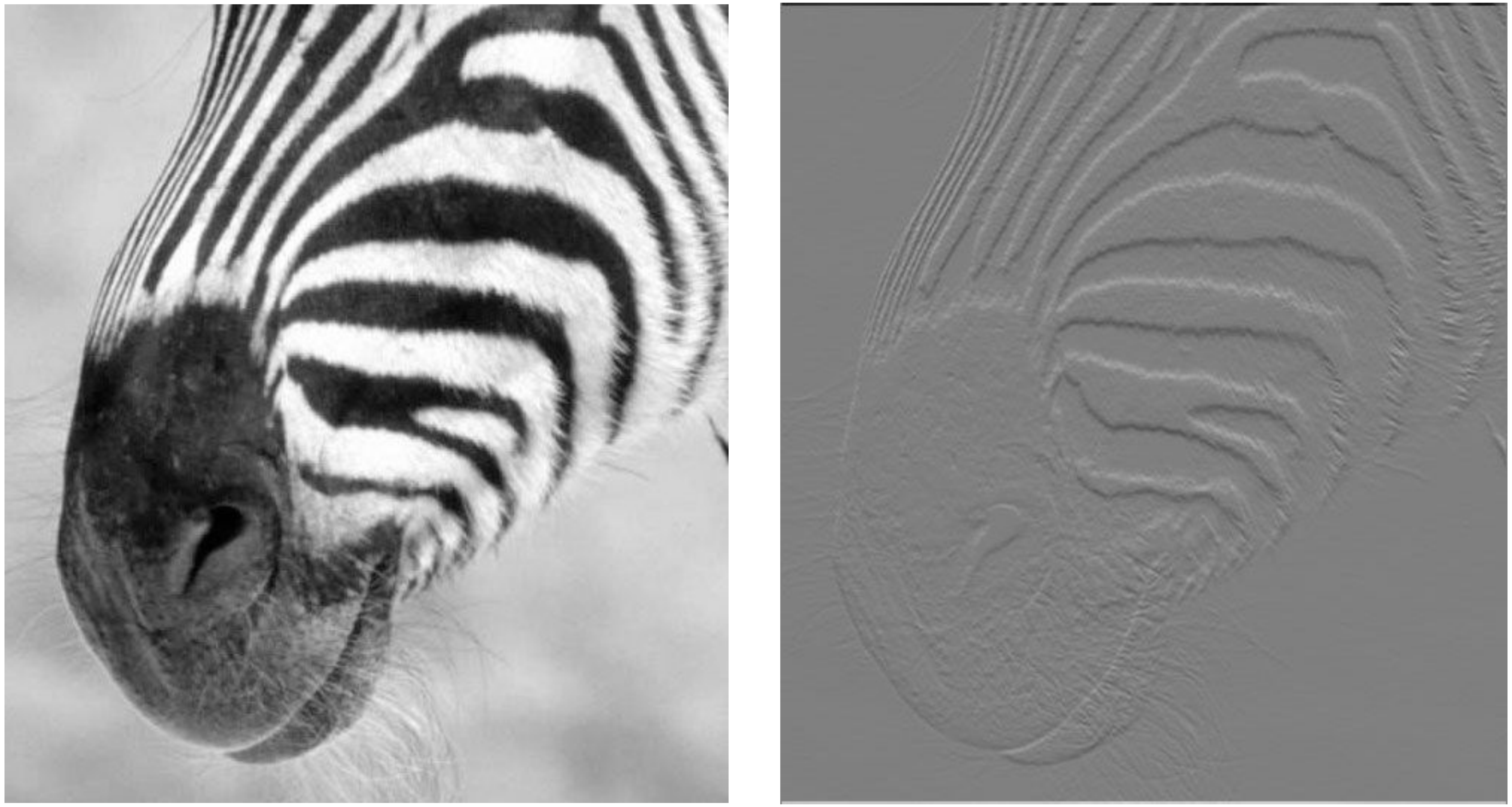
Derivative in X (i.e., horizontal) direction



Forsyth & Ponce (1st ed.) Figure 7.4 (top left & top right)

Estimating **Derivatives**

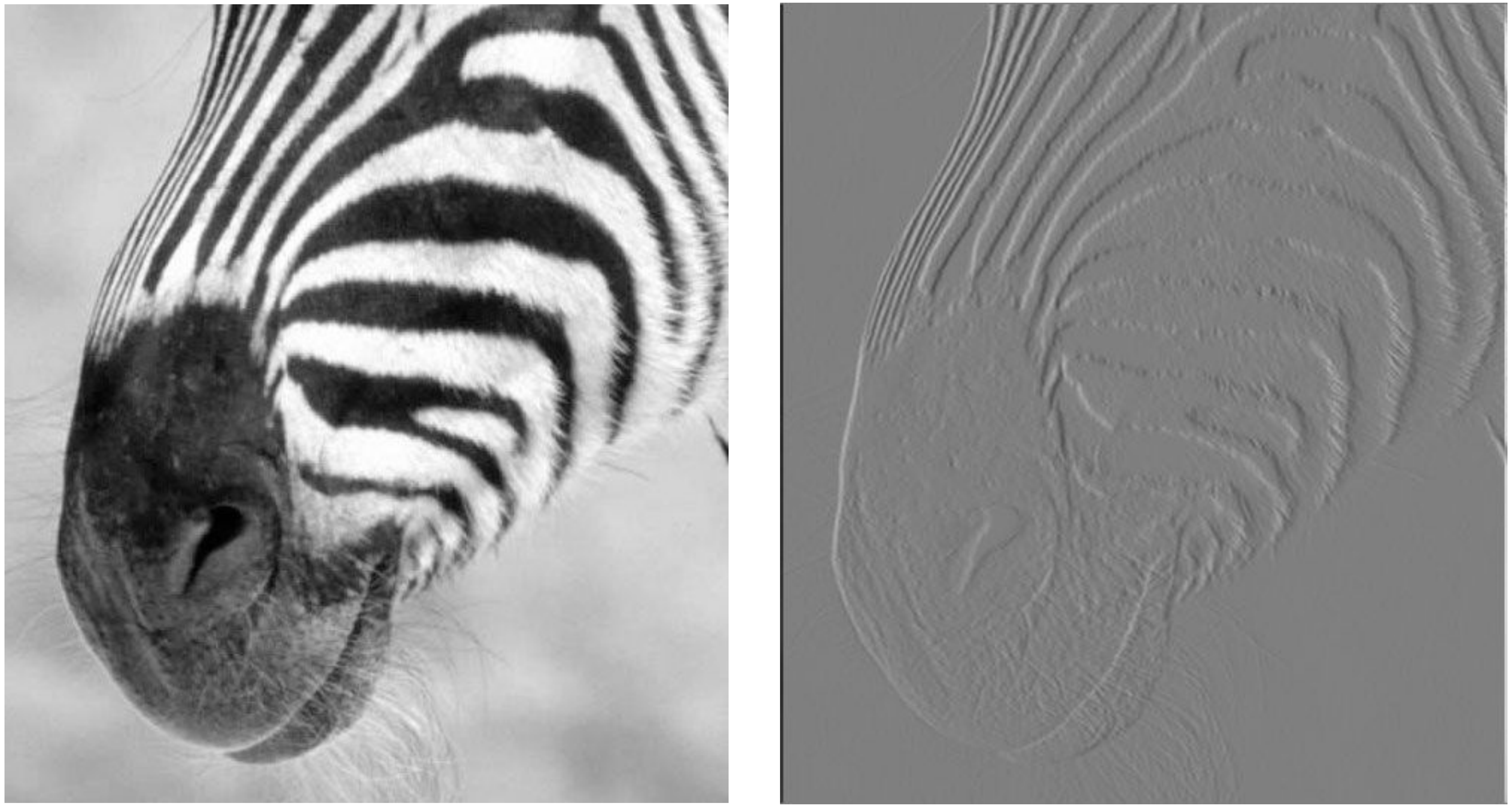
Derivative in Y (i.e., vertical) direction



Forsyth & Ponce (1st ed.) Figure 7.4 (top left & top middle)

Estimating **Derivatives**

Derivative in X (i.e., horizontal) direction



Forsyth & Ponce (1st ed.) Figure 7.4 (top left & top right)

A Sort **Exercise**

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

(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

A Sort **Exercise**: Derivative in X Direction

-1	1

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

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

A Sort **Exercise**: Derivative in X Direction

-1	1

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

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

0					

A Sort **Exercise**: Derivative in X Direction

-1	1

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

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

0					

A Sort **Exercise**: Derivative in X Direction

-1	1

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

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

0	0				

A Sort **Exercise**: Derivative in X Direction

-1	1

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

(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

0					
0	0	0	0	0	
0	0	0	0	0	

A Sort **Exercise**: Derivative in X Direction

-1	1

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

(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

0	-0.4				
0	0	0	0	0	
0	0	0	0	0	

A Sort **Exercise**: Derivative in X Direction

-1	1

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

(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

0	-0.4	-0.3	-0.3	0	
0	-0.4	-0.3	-0.3	0	
0	0	0	0	0	
0	0	0	0	0	

A Sort **Exercise**: Derivative in Y Direction

$$\begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

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

(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

A Sort **Exercise**: Derivative in Y Direction

$$\begin{array}{|c|}\hline 1 \\ \hline -1 \\ \hline \end{array}$$

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

(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

0					

A Sort **Exercise**: Derivative in Y Direction

$$\begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

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

(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

0	0	0	0	0	0

A Sort **Exercise**: Derivative in Y Direction

$$\begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

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

(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

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

Estimating **Derivatives**

-1	1

Question: Why, in general, should the weights of a filter used for differentiation sum to 0?

Estimating **Derivatives**

-1	1

Question: Why, in general, should the weights of a filter used for differentiation sum to 0?

Answer: Think of a constant image, $I(X, Y) = k$. The derivative is 0. Therefore, the weights of any filter used for differentiation need to sum to 0.

Estimating **Derivatives**

-1	1

Question: Why, in general, should the weights of a filter used for differentiation sum to 0?

Answer: Think of a constant image, $I(X, Y) = k$. The derivative is 0. Therefore, the weights of any filter used for differentiation need to sum to 0.

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

Estimating **Derivatives**

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.

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

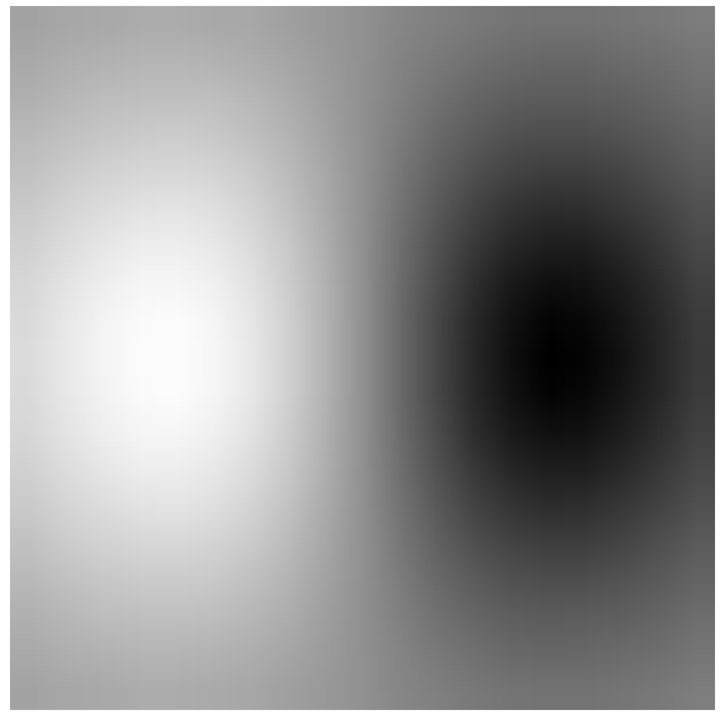
We can use **derivative of Gaussian** filters

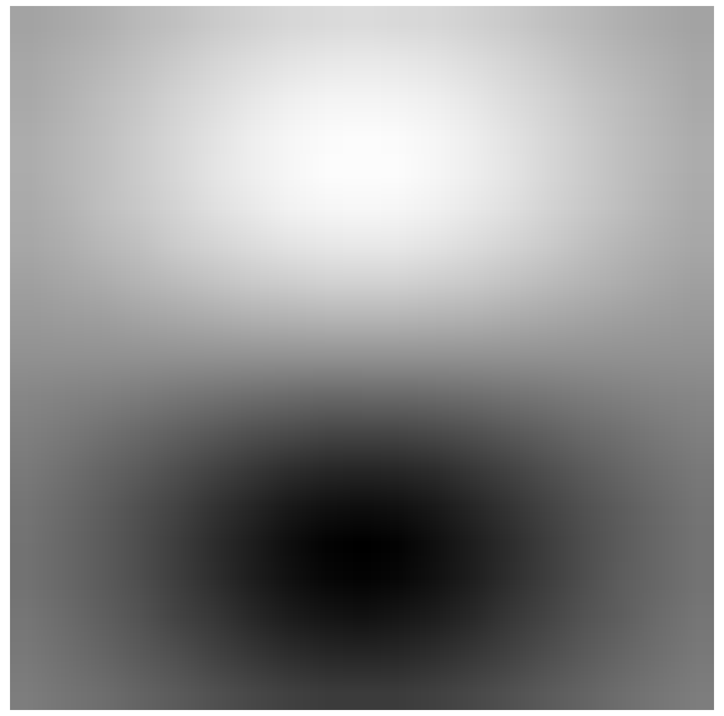
— because differentiation is convolution, and

— convolution is associative

Let $\otimes$ denote convolution

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

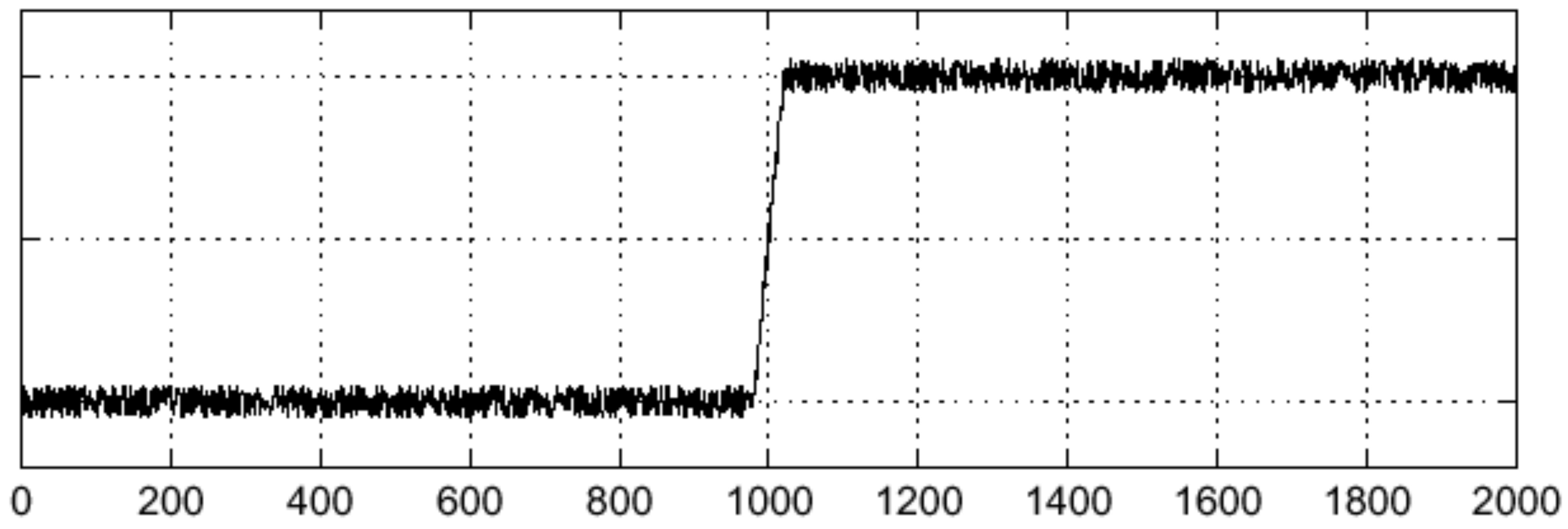




1D **Example**

Lets consider a row of pixels in an image:

$I(X, 245)$

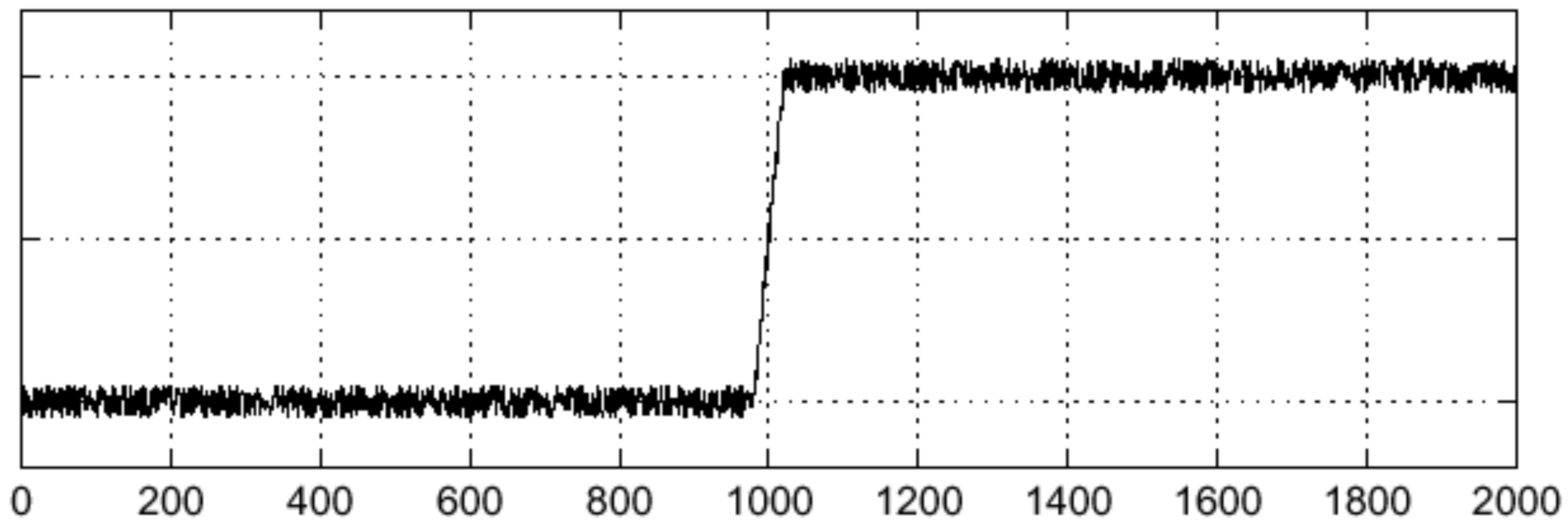


Where is the edge?

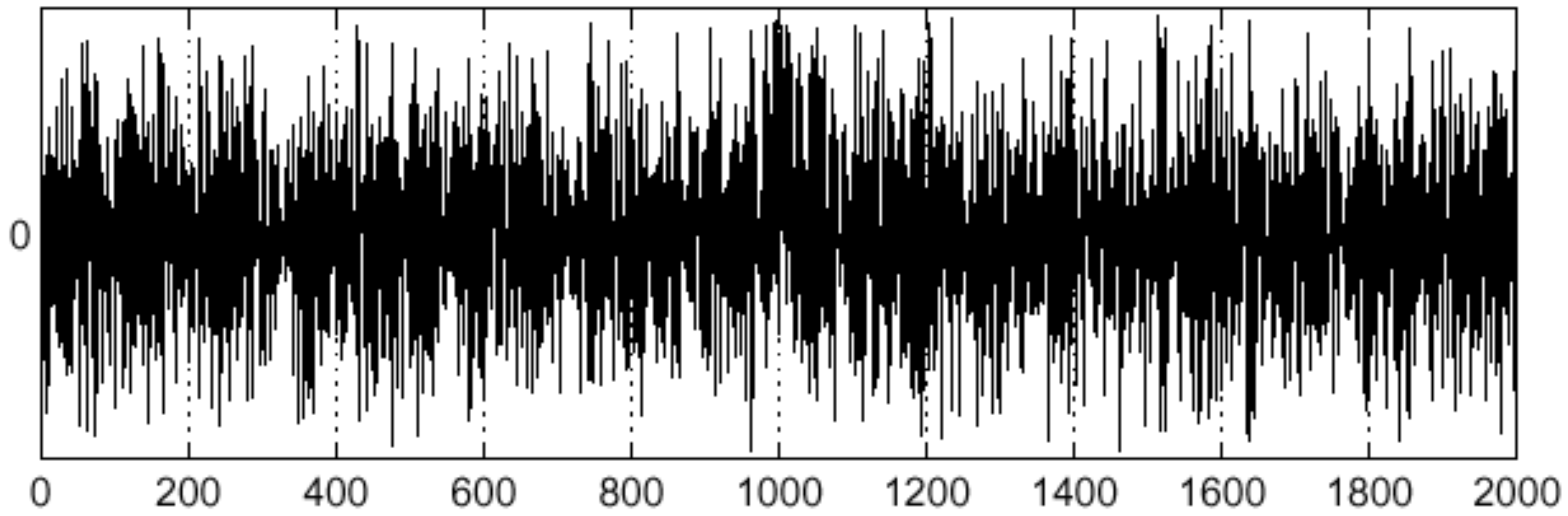
1D **Example**: Derivative

Lets consider a row of pixels in an image:

$$I(X, 245)$$



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



Where is the edge?

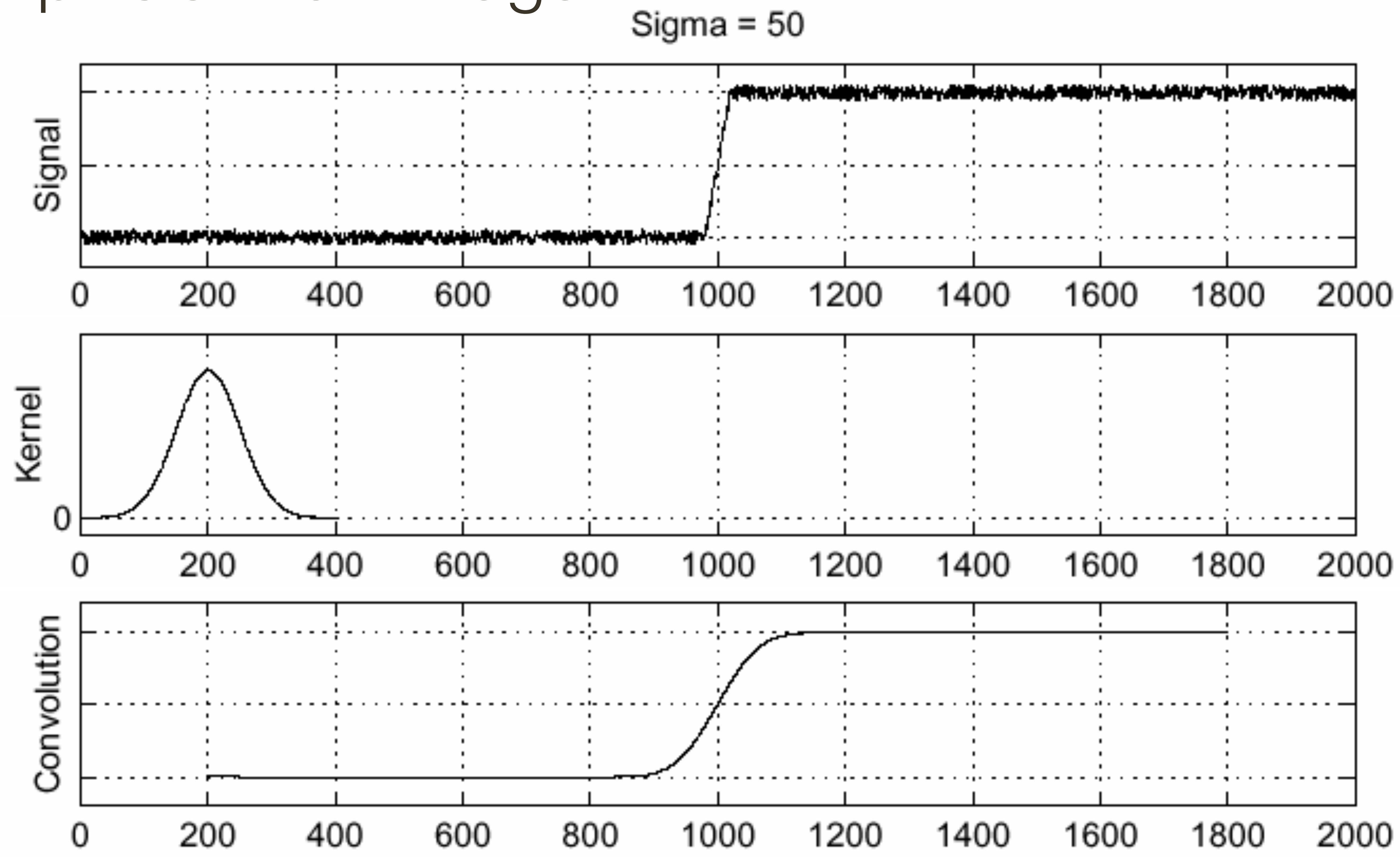
1D **Example**: Smoothing + Derivative

Lets consider a row of pixels in an image:

$$I(X, 245)$$

$$G$$

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



1D **Example**: Smoothing + Derivative

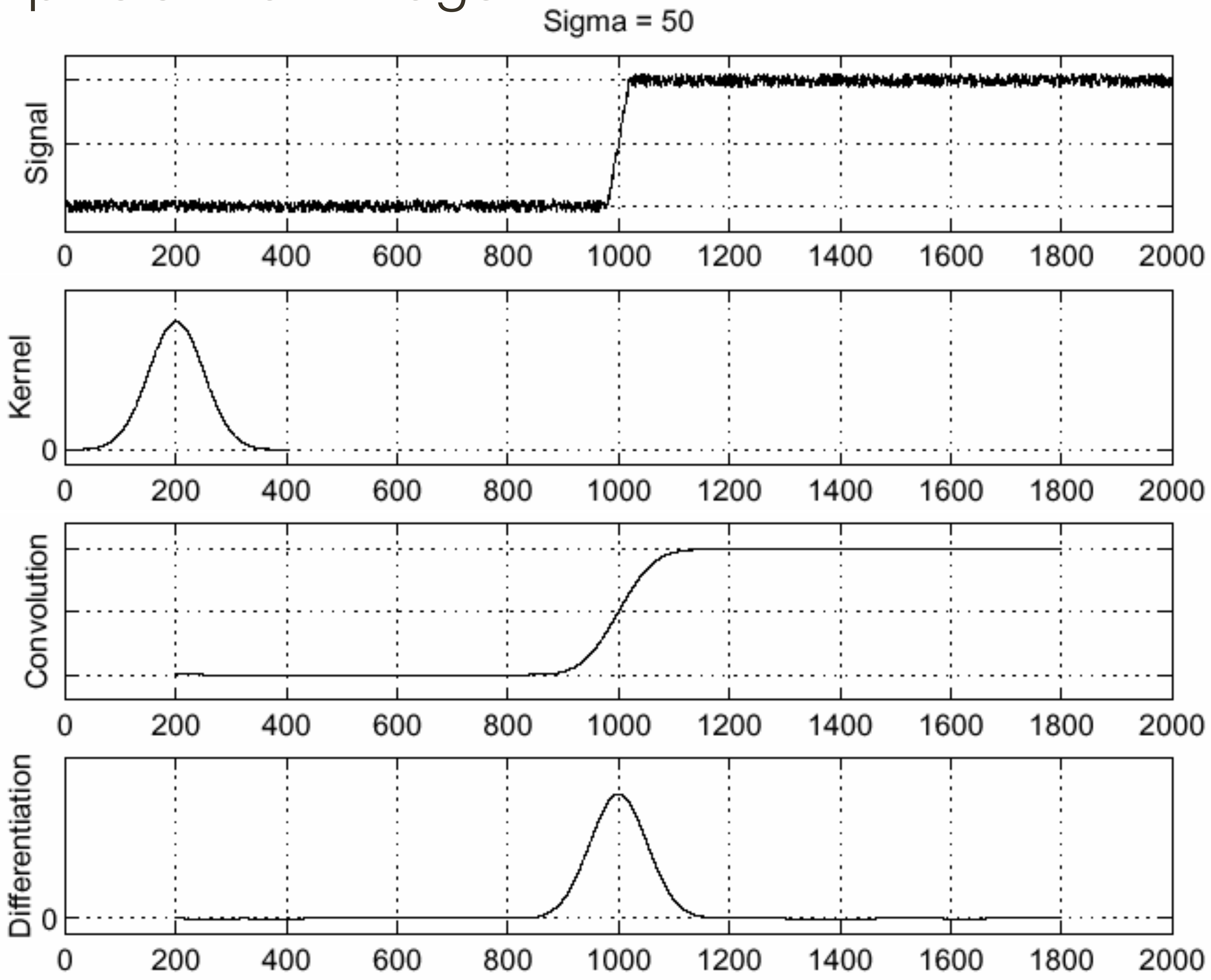
Lets consider a row of pixels in an image:

$$I(X, 245)$$

$$G$$

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

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



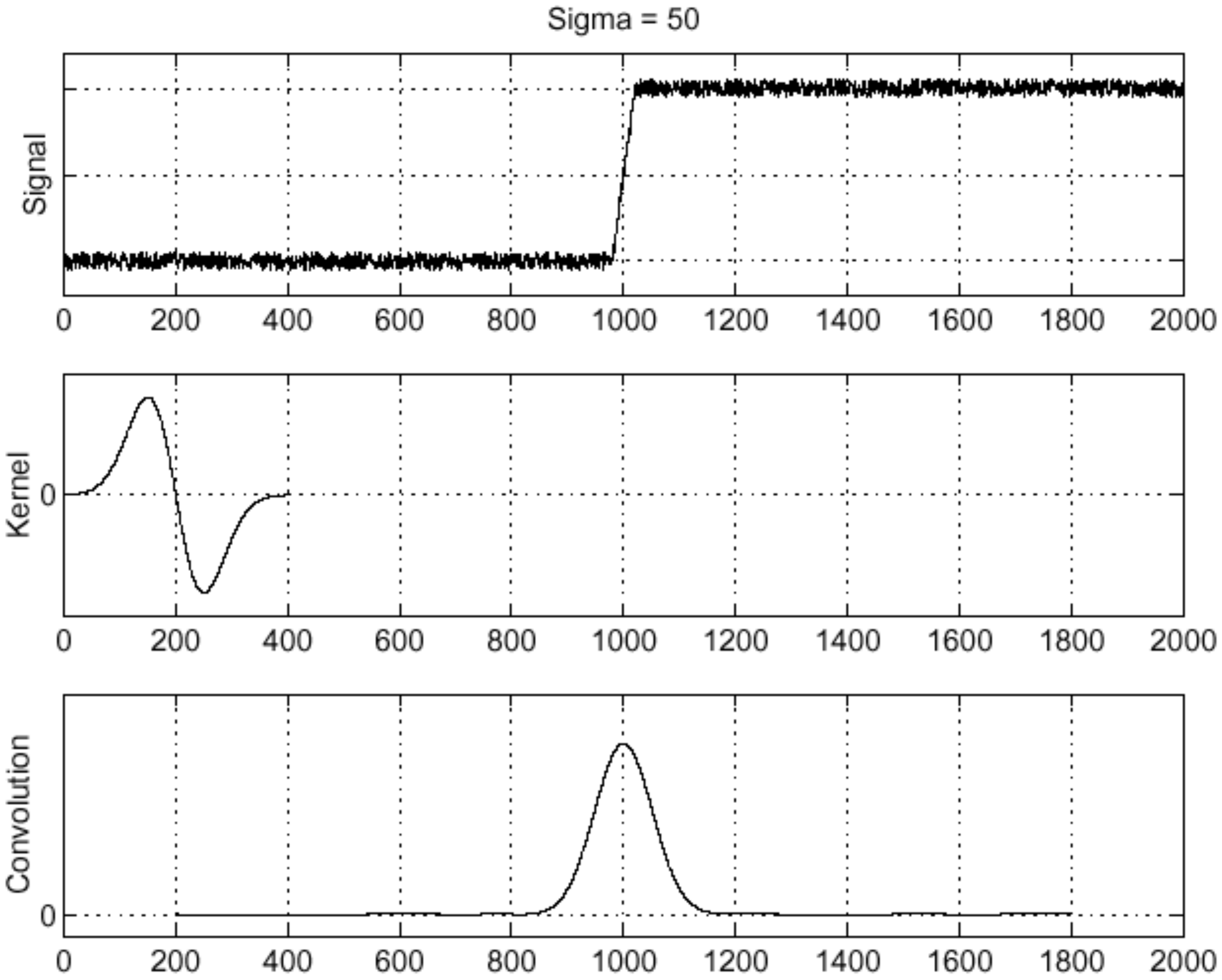
1D **Example**: Smoothing + Derivative (efficient)

Lets consider a row of pixels in an image:

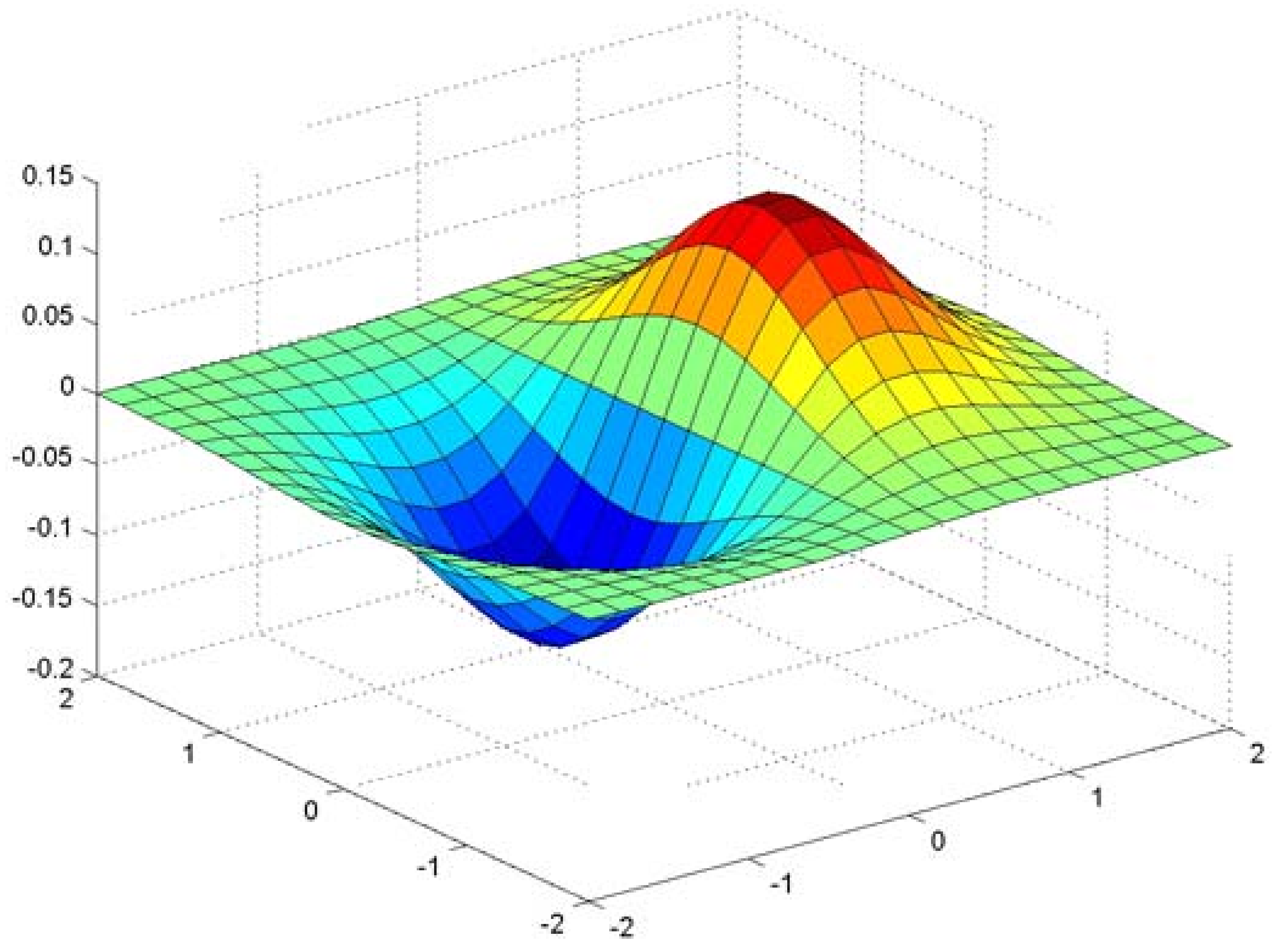
$$I(X, 245)$$

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

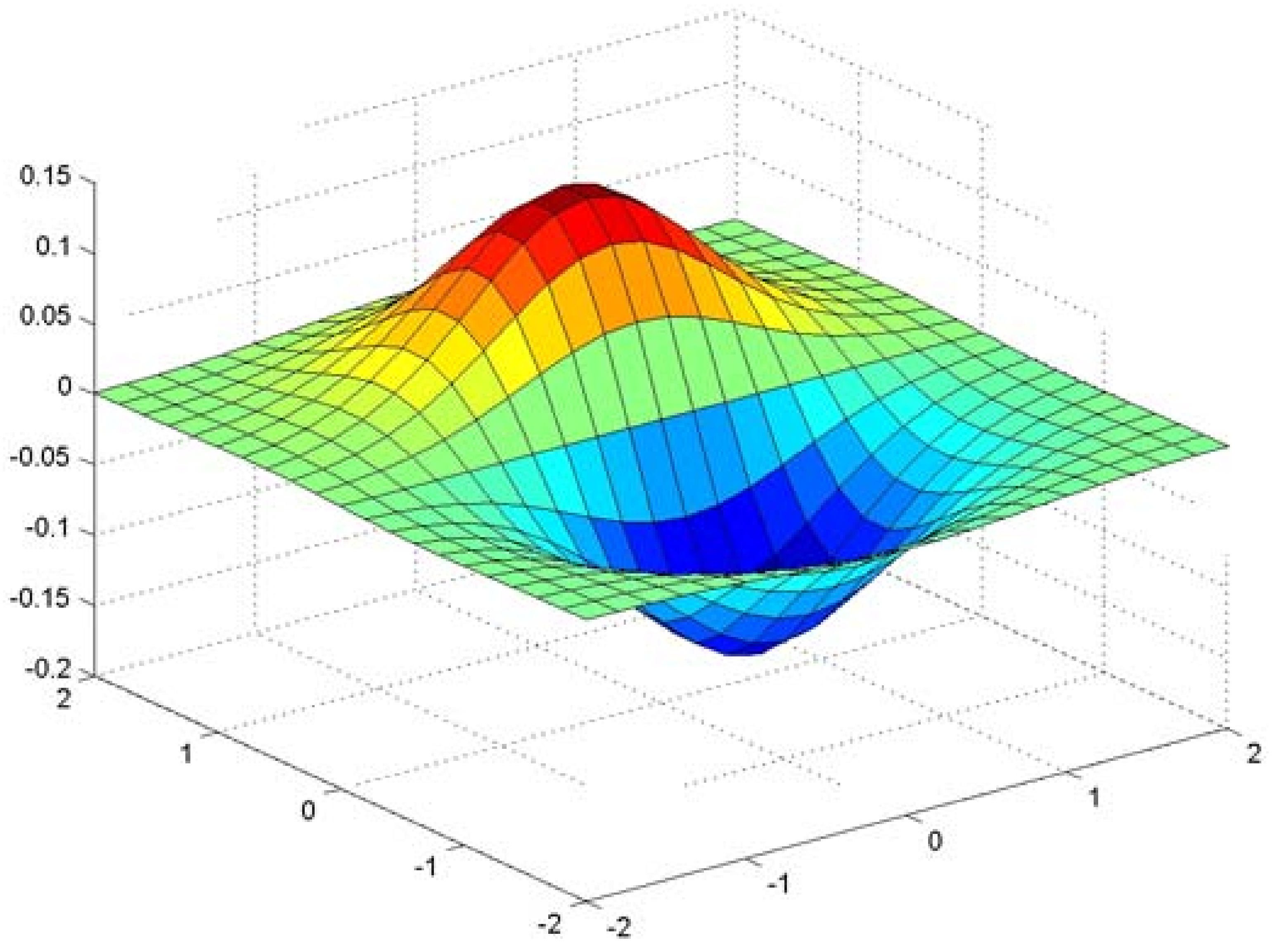
$$\frac{\partial G}{\partial x} \otimes I(X, Y)$$



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



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



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

Slide Credit: Christopher Rasmussen

Gradient **Magnitude**

Let $I(X, Y)$ be a (digital) image

Let $I_x(X, Y)$ and $I_y(X, Y)$ be estimates of the partial derivatives in the x and y directions, respectively.

Call these estimates I_x and I_y (for short) The vector $[I_x, I_y]$ is the **gradient**

The scalar $\sqrt{I_x^2 + I_y^2}$ is the **gradient magnitude**

Image **Gradient**

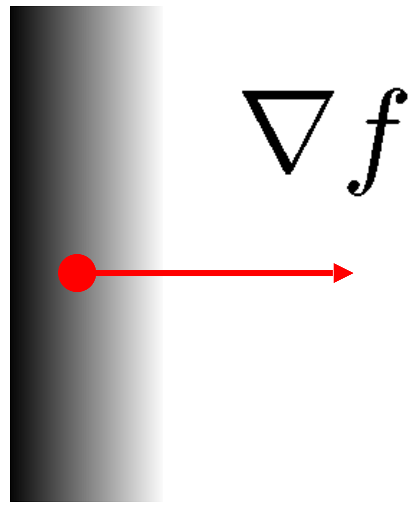
The gradient of an image: $\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$

Image **Gradient**

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

Image **Gradient**

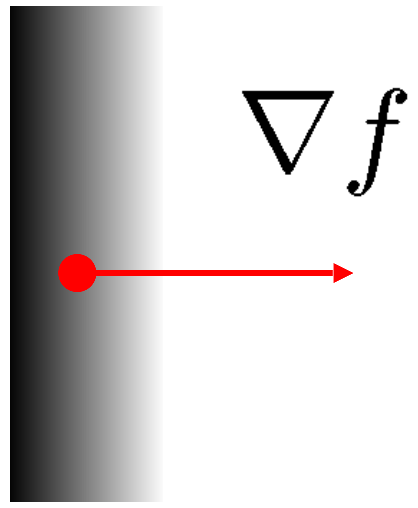
The gradient of an image: $\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$



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

Image **Gradient**

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



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

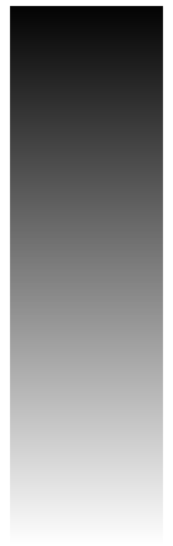
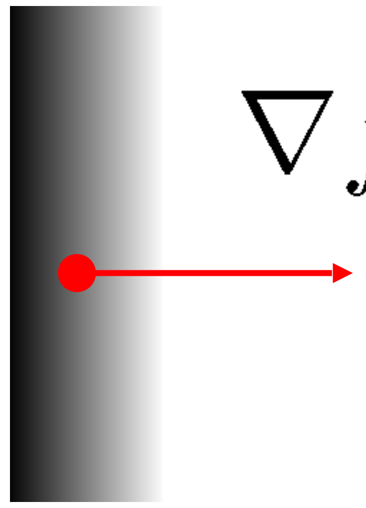
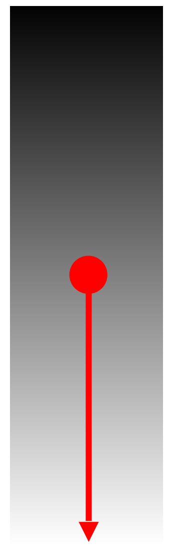


Image **Gradient**

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



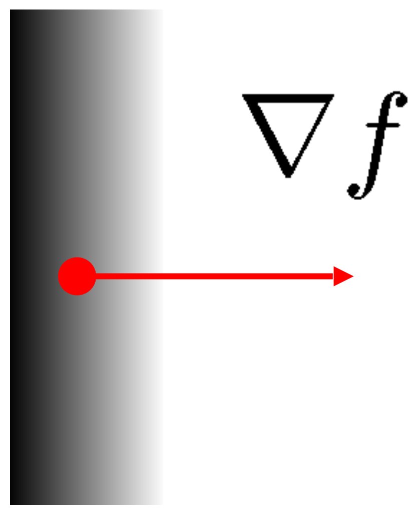
$\nabla f = \left[\frac{\partial f}{\partial x}, 0\right]$



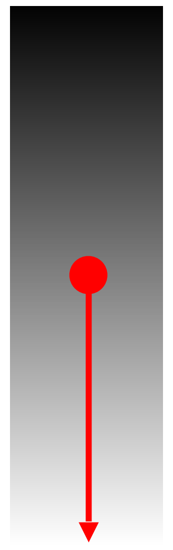
$\nabla f = \left[0, \frac{\partial f}{\partial y}\right]$

Image **Gradient**

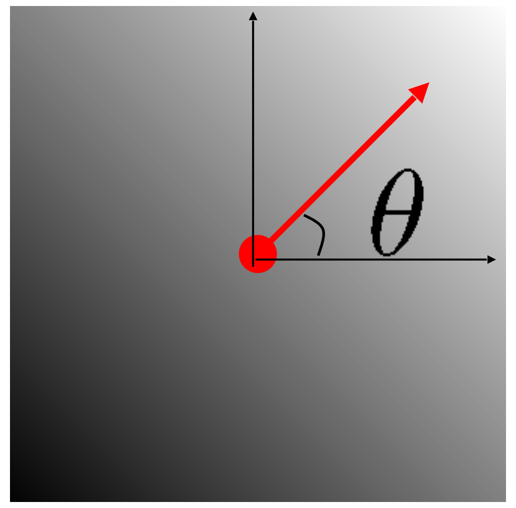
The gradient of an image: $$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$



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



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

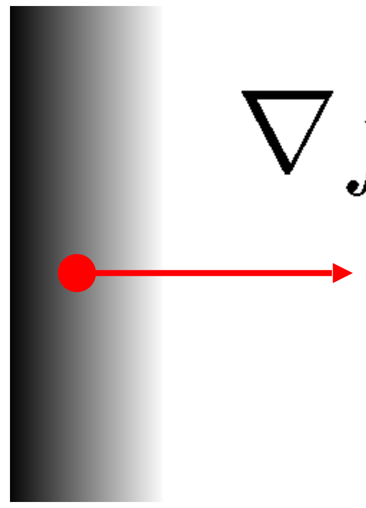


$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$

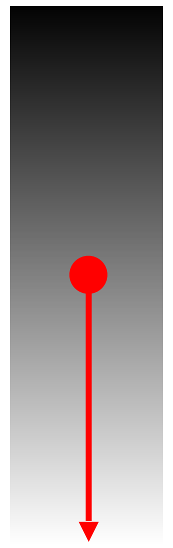
The gradient points in the direction of most rapid **increase of intensity**:

Image **Gradient**

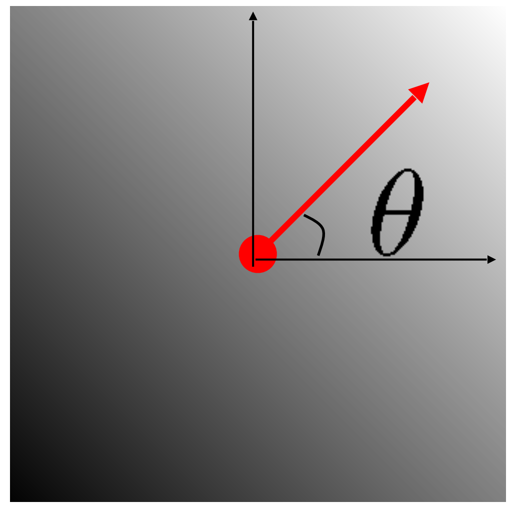
The gradient of an image: $$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$



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



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



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

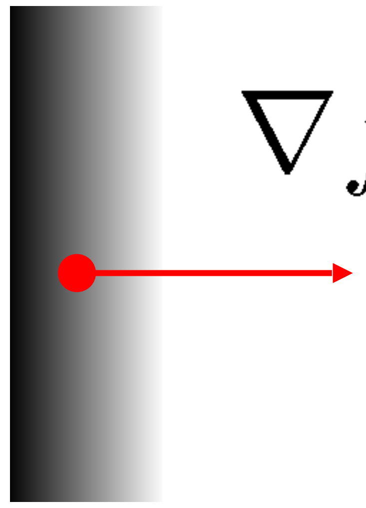
The gradient points in the direction of most rapid **increase of intensity**:

The **gradient direction** is given by:

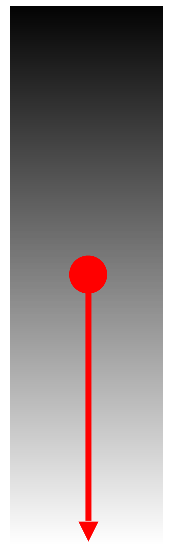
(how is this related to the direction of the edge?)

Image **Gradient**

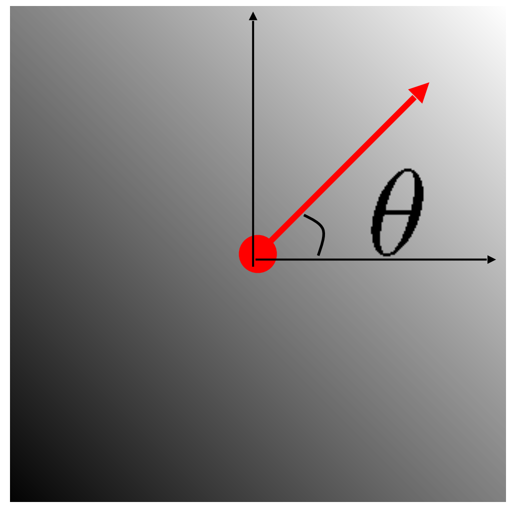
The gradient of an image: $\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$



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



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



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

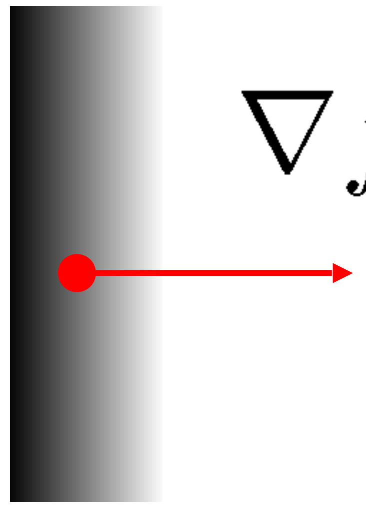
The gradient points in the direction of most rapid **increase of intensity**:

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

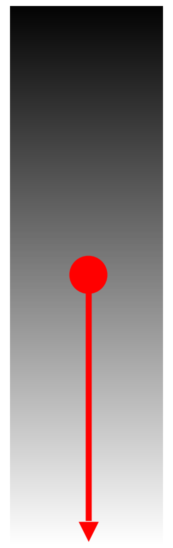
(how is this related to the direction of the edge?)

Image **Gradient**

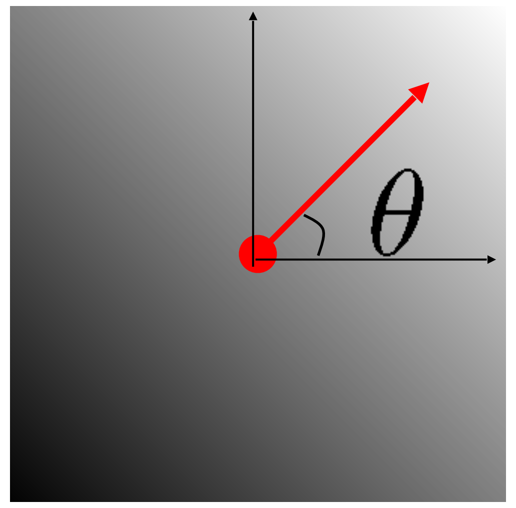
The gradient of an image: $$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$



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



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



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

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

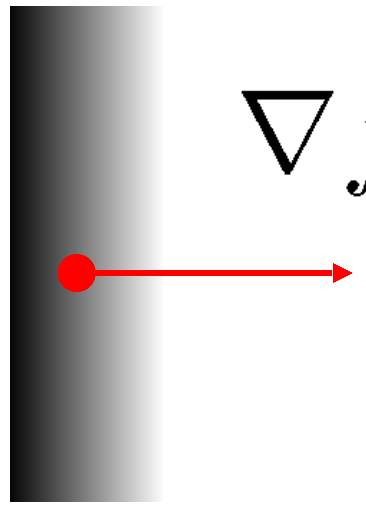
The **gradient direction** is given by:

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

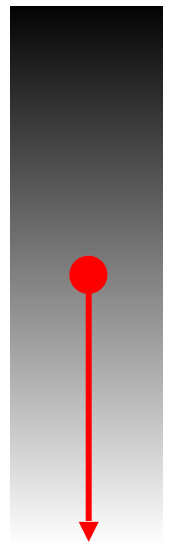
The edge strength is given by the **gradient magnitude**:

Image **Gradient**

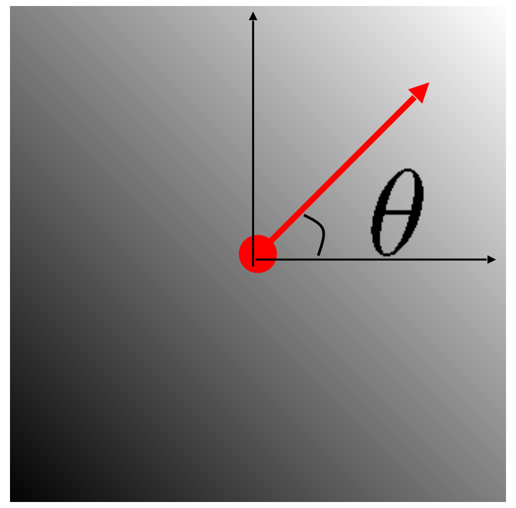
The gradient of an image: $\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$



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



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



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

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

Image **Gradient**

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

By looking at the **gradient magnitude** we can reason about the **strength of the edge** and by looking at the **gradient direction** we can reason about the **direction of the edge**

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

The **gradient direction** is given by:

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

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

Image **Gradient**

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

By looking at the **gradient magnitude** we can reason about the **strength of the edge** and by looking at the **gradient direction** we can reason about the **direction of the edge**

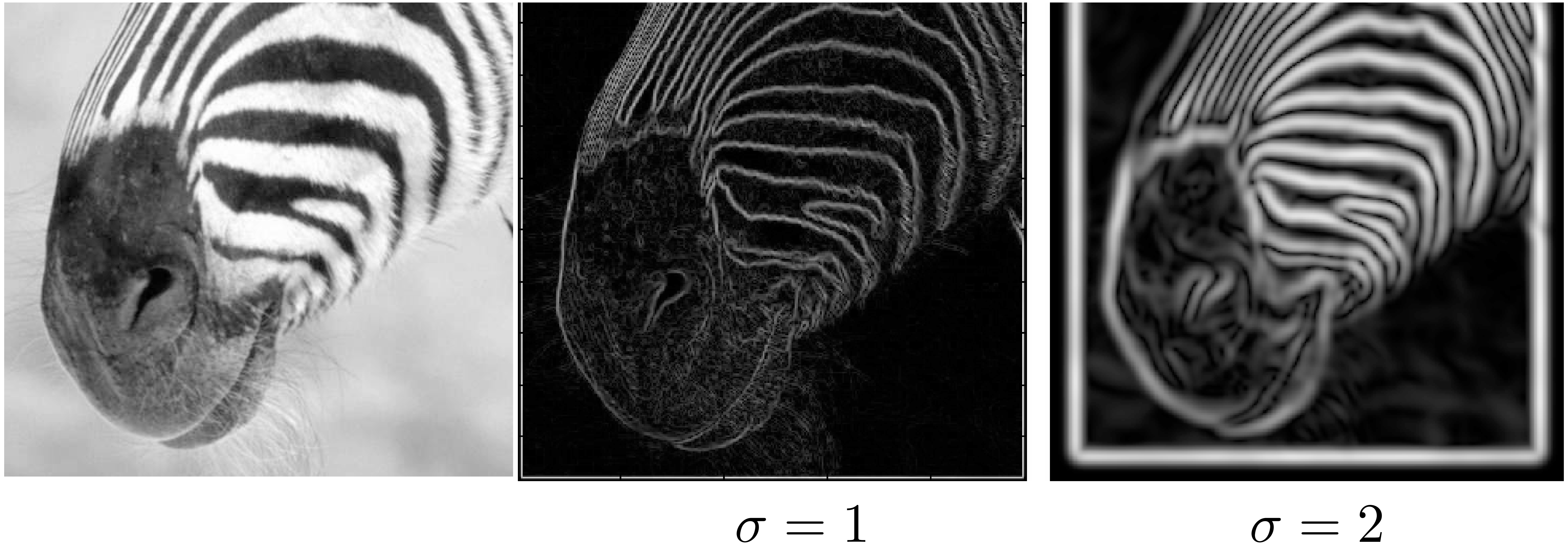
The gradient points in the direction of most rapid **increase of intensity**:

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

Gradient **Magnitude**



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.

$$\begin{bmatrix} \boxed{-1 \quad 0 \quad 1} \\ -2 \quad 0 \quad 2 \\ -1 \quad 0 \quad 1 \end{bmatrix}$$

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

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!

Comparing **Edge** Detectors

Comparing **Edge** Detectors

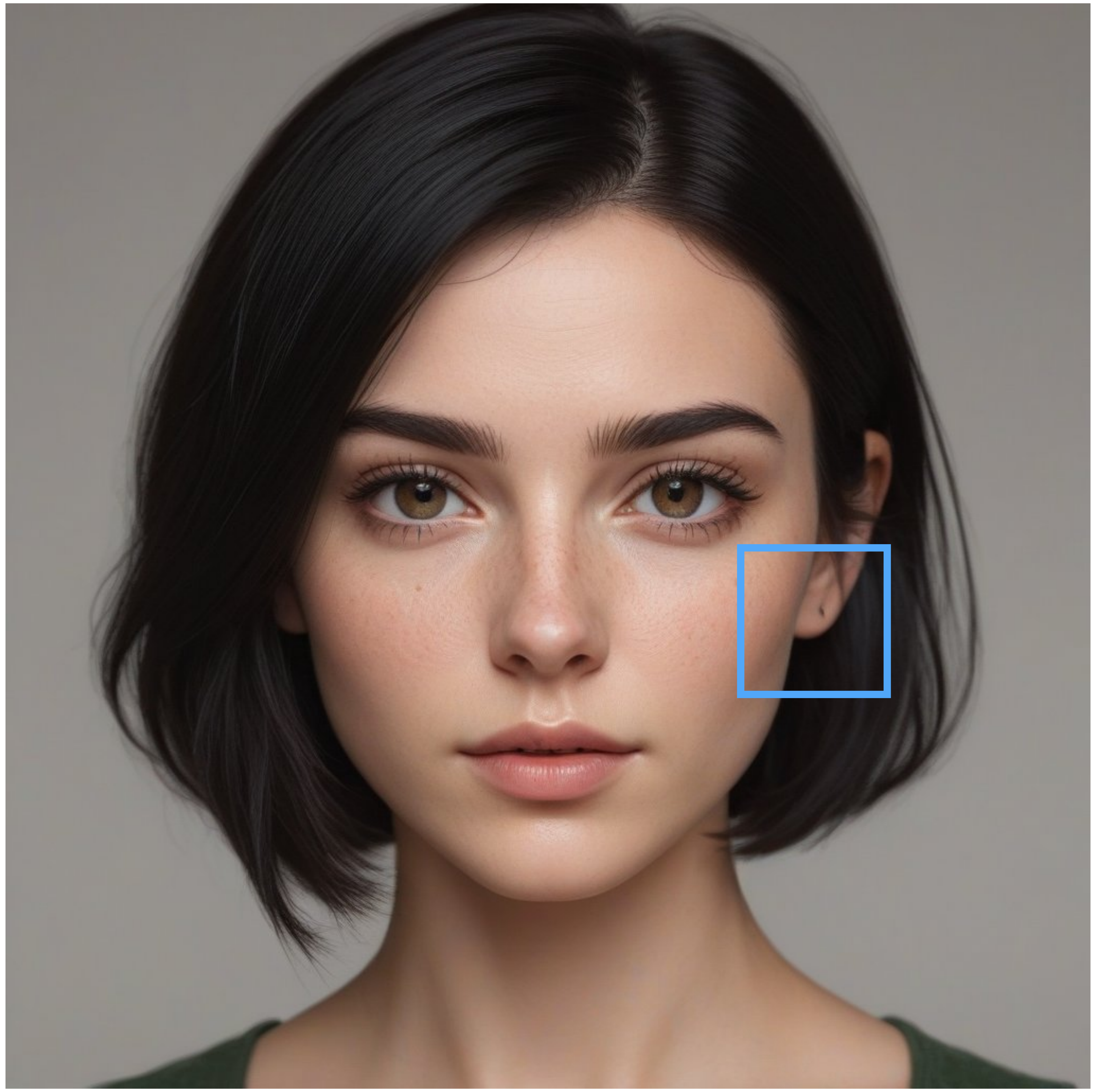
Good detection: minimize probability of false positives/negatives (spurious/missing) edges

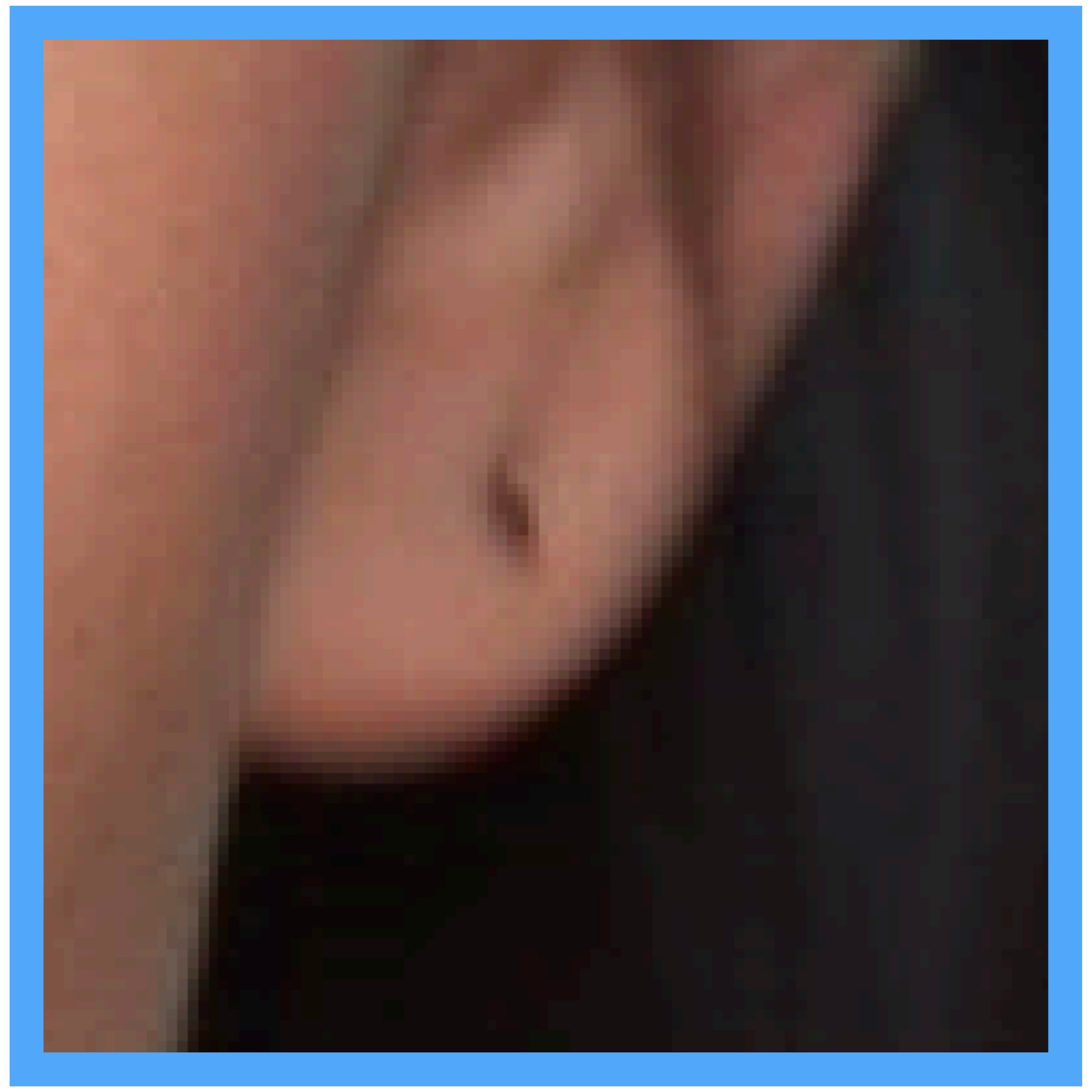
Good localization: found edges should be as close to true image edge as possible

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

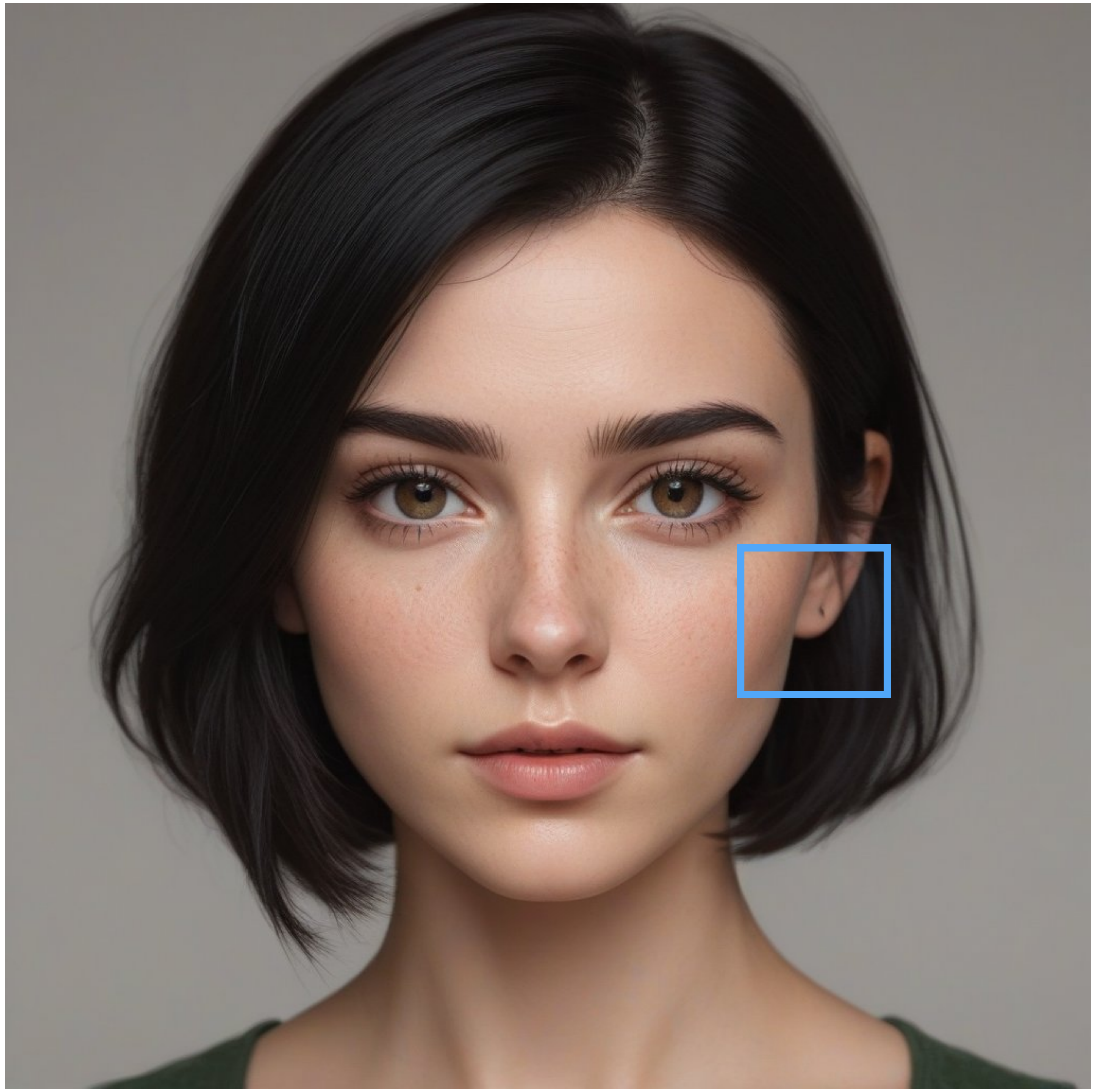
Comparing **Edge** Detectors

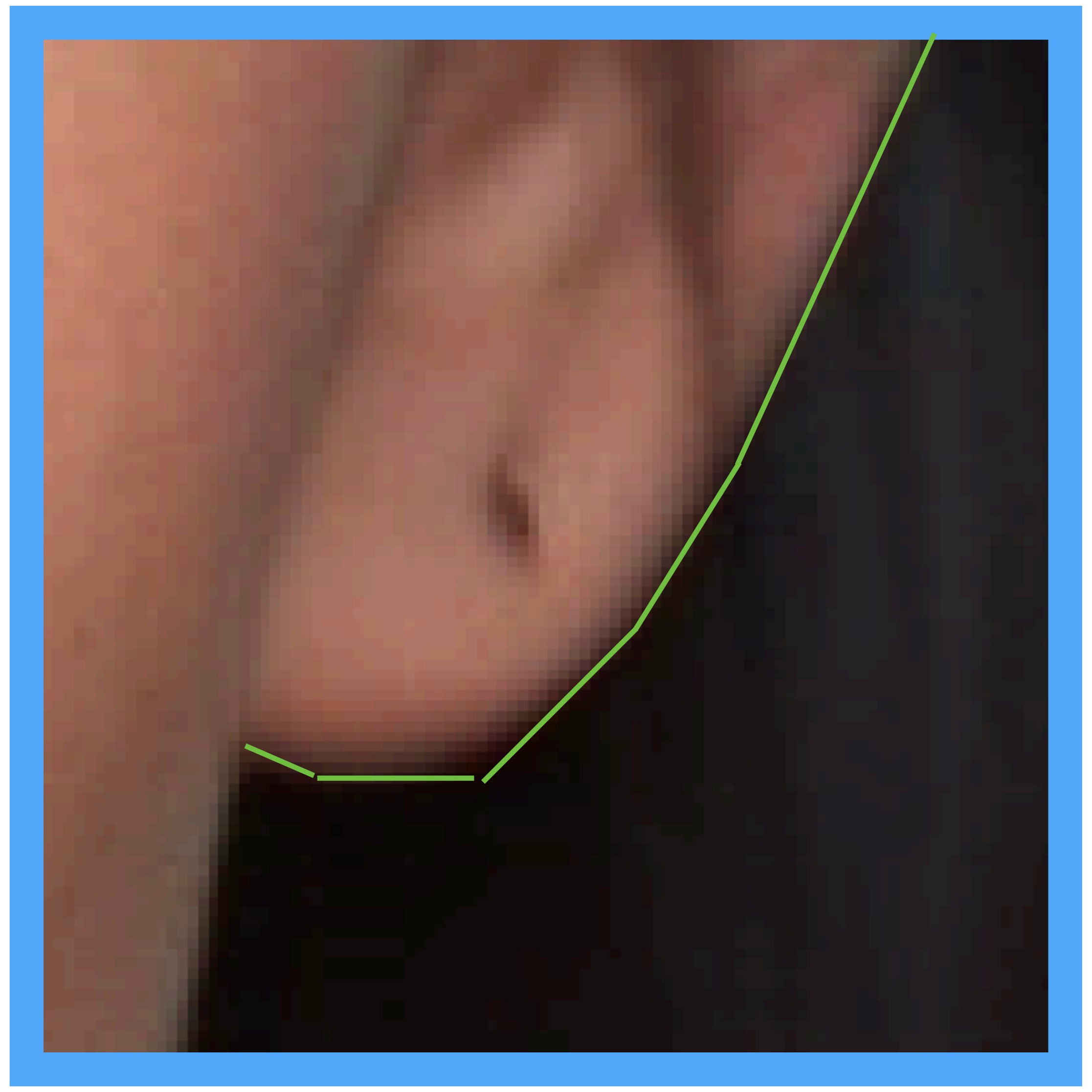
Comparing **Edge** Detectors



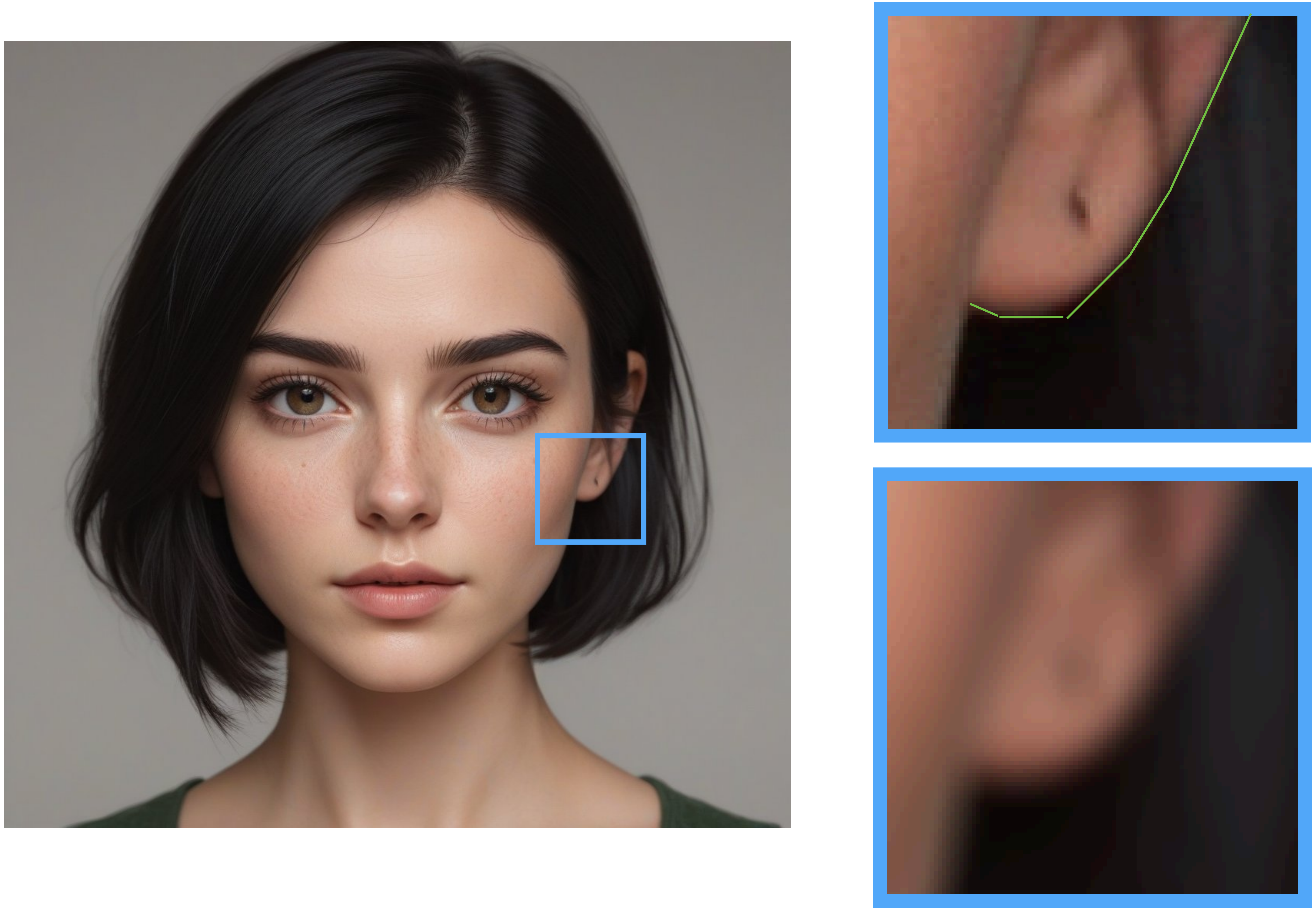


Comparing **Edge** Detectors

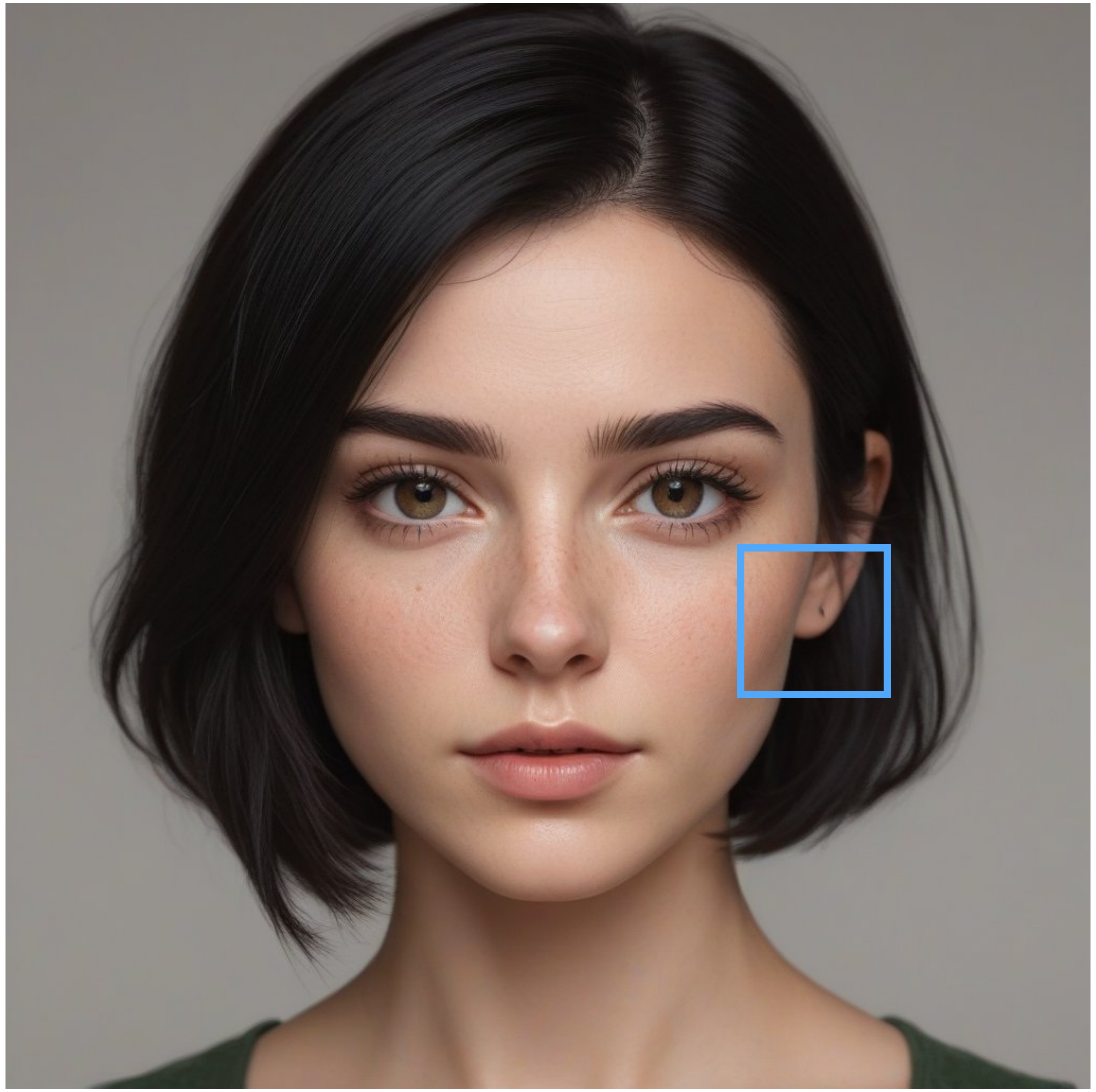


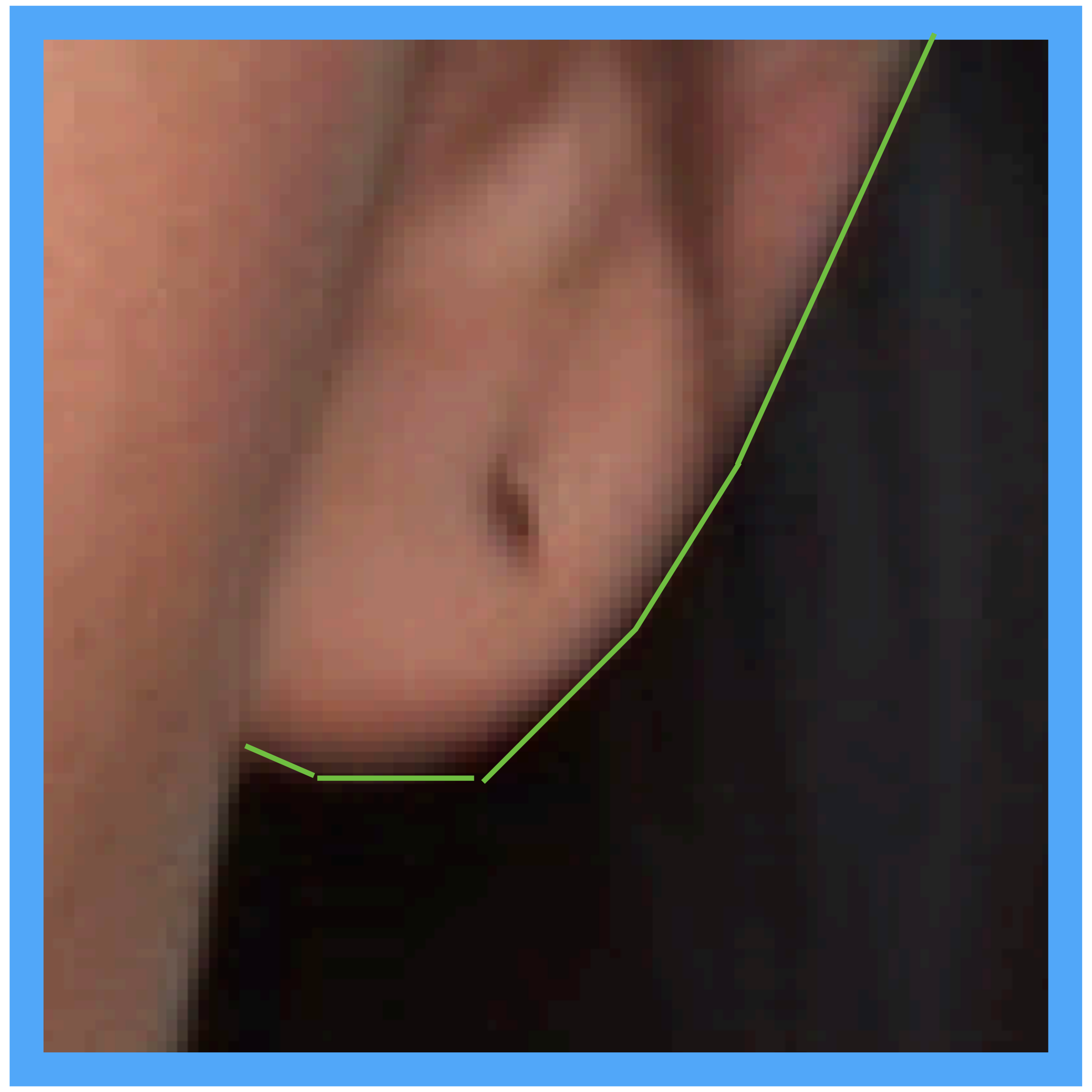


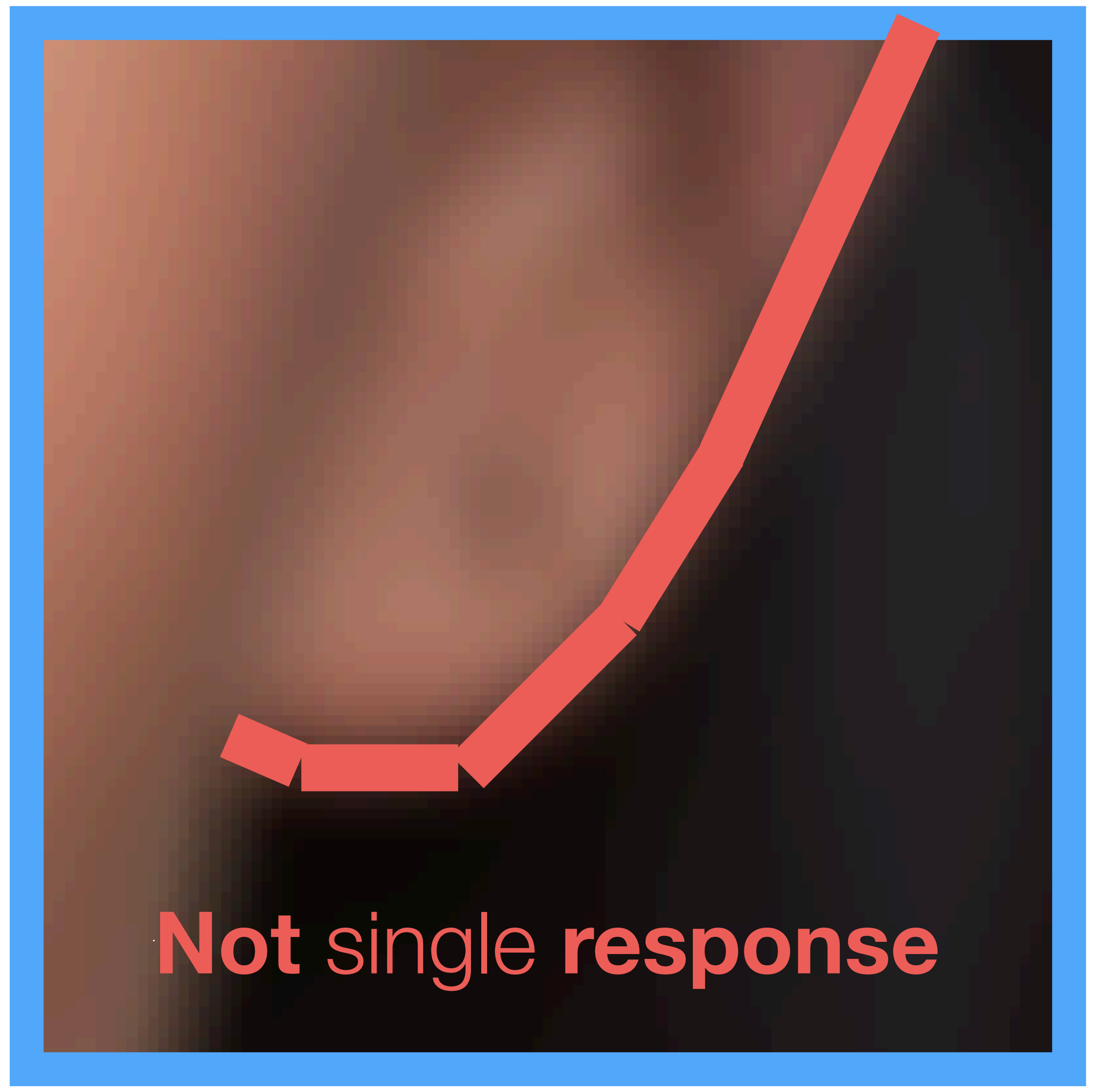
Comparing **Edge** Detectors



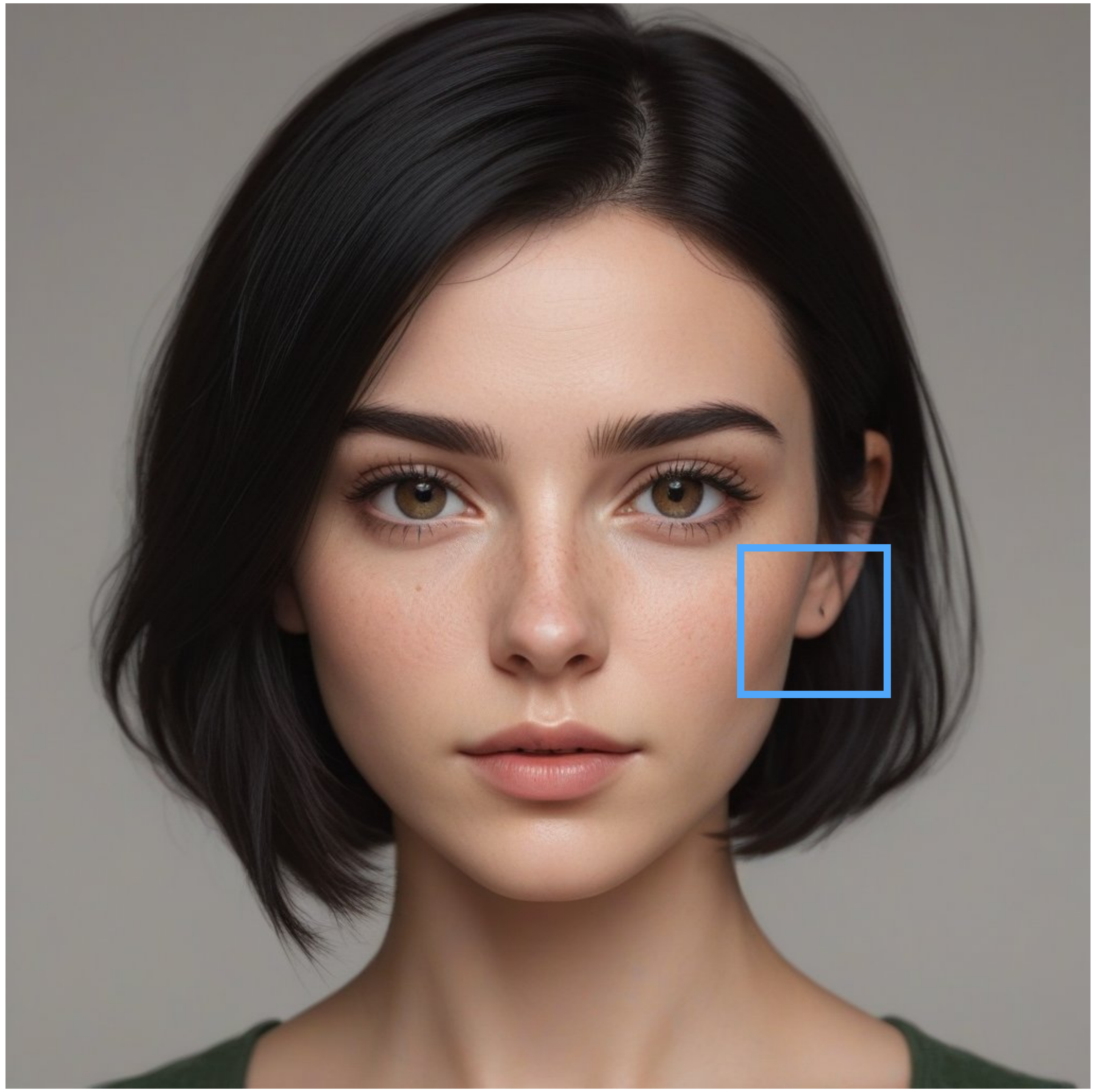
Comparing **Edge** Detectors

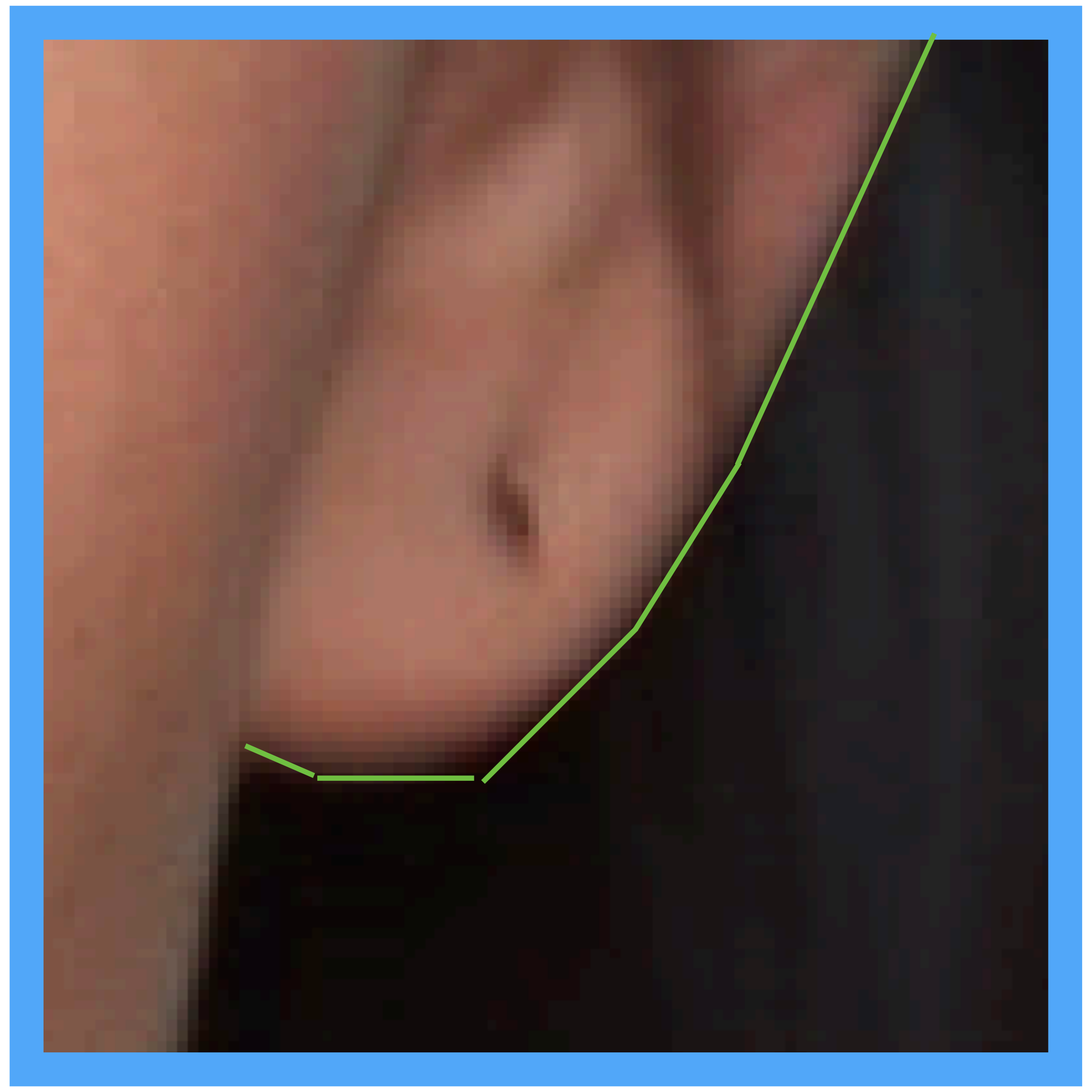


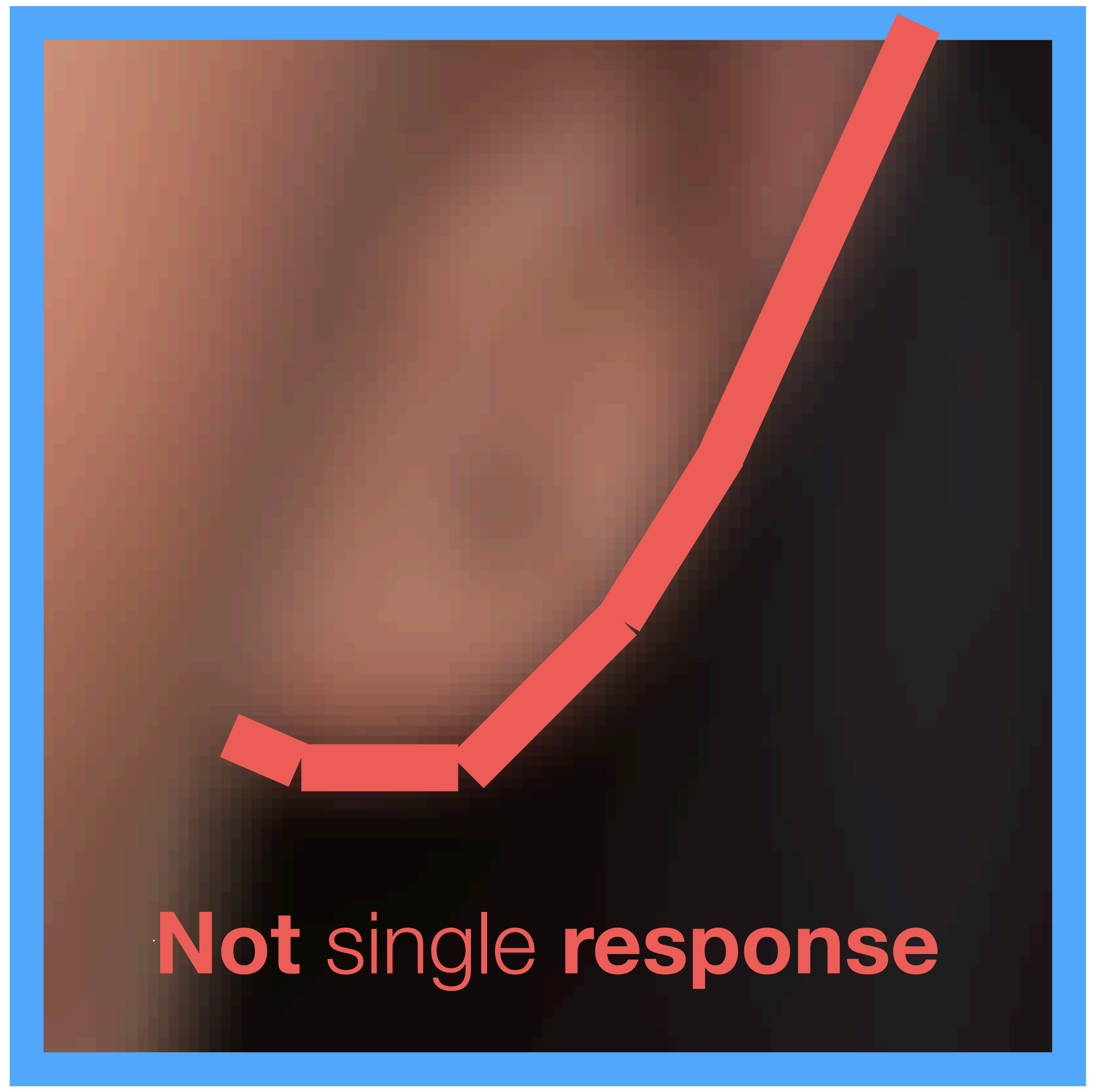


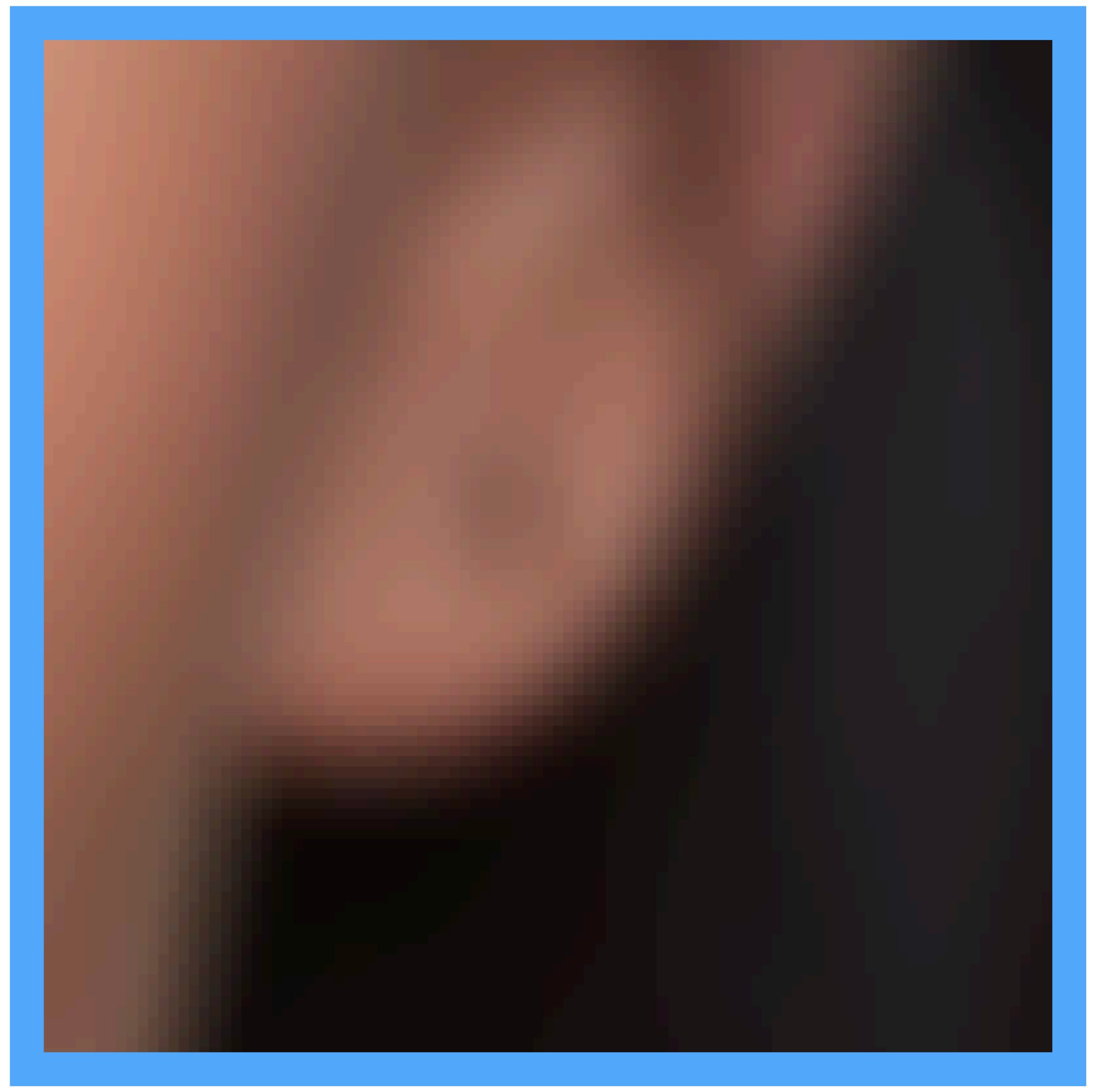


Comparing **Edge** Detectors

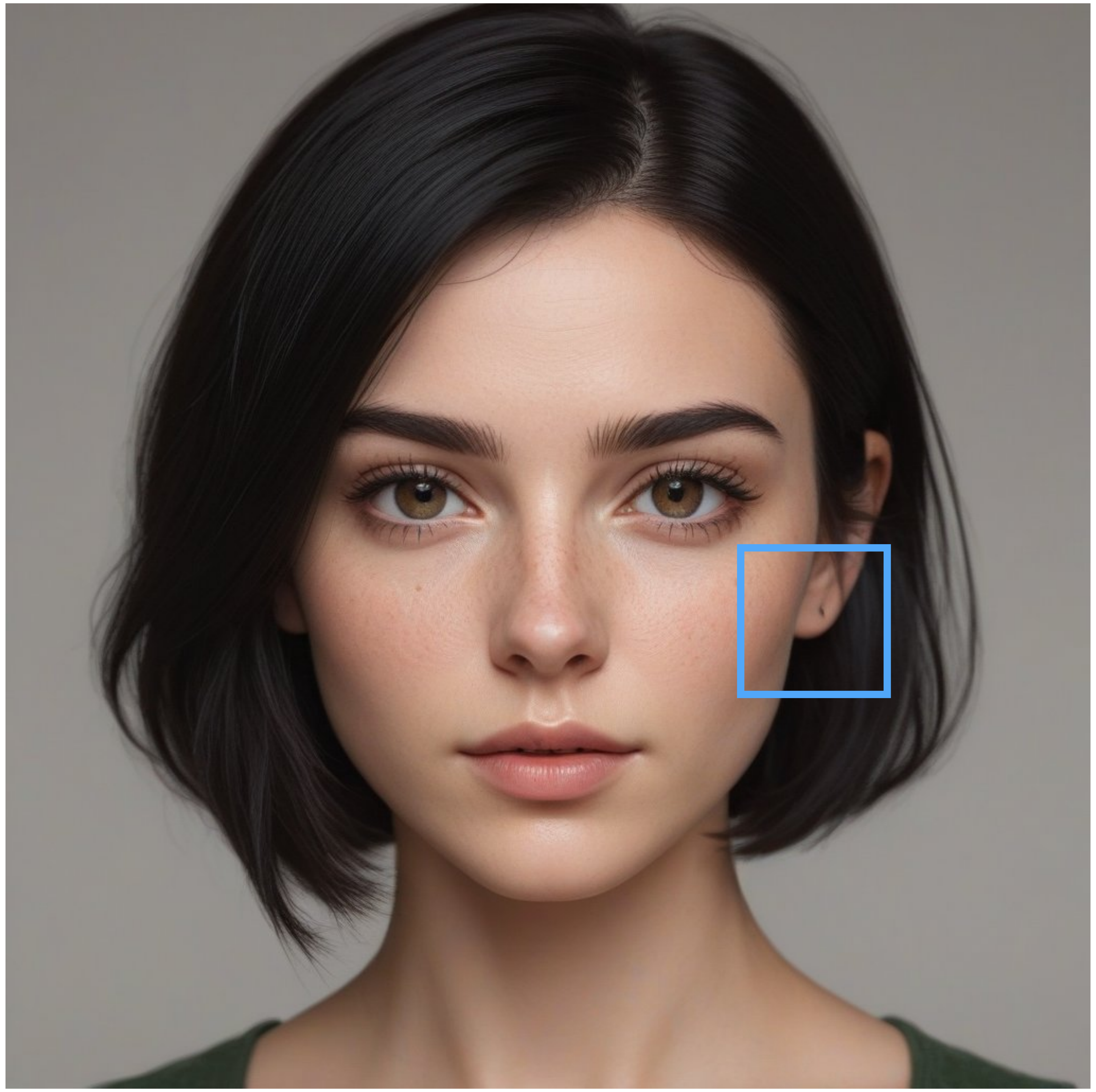


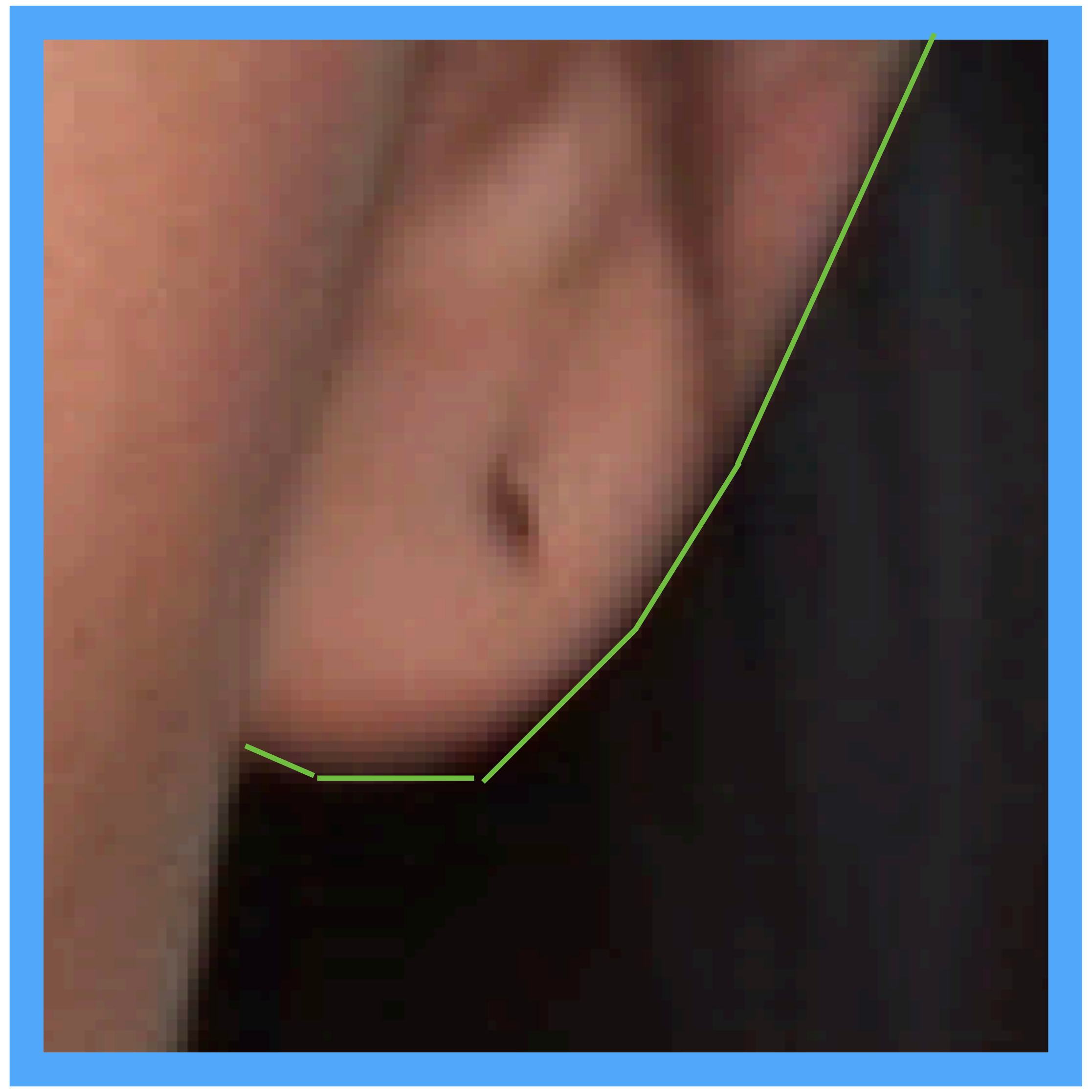


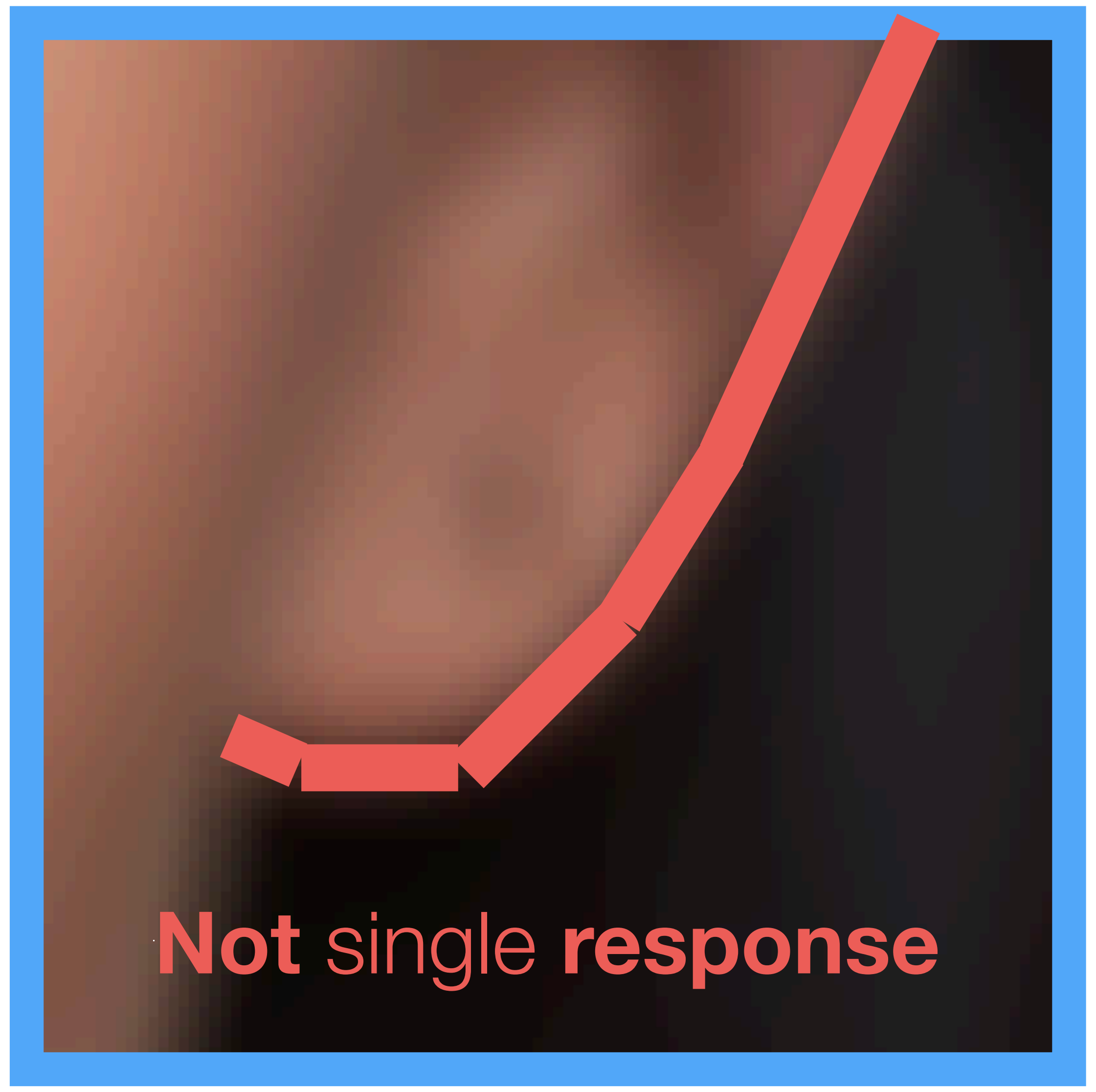


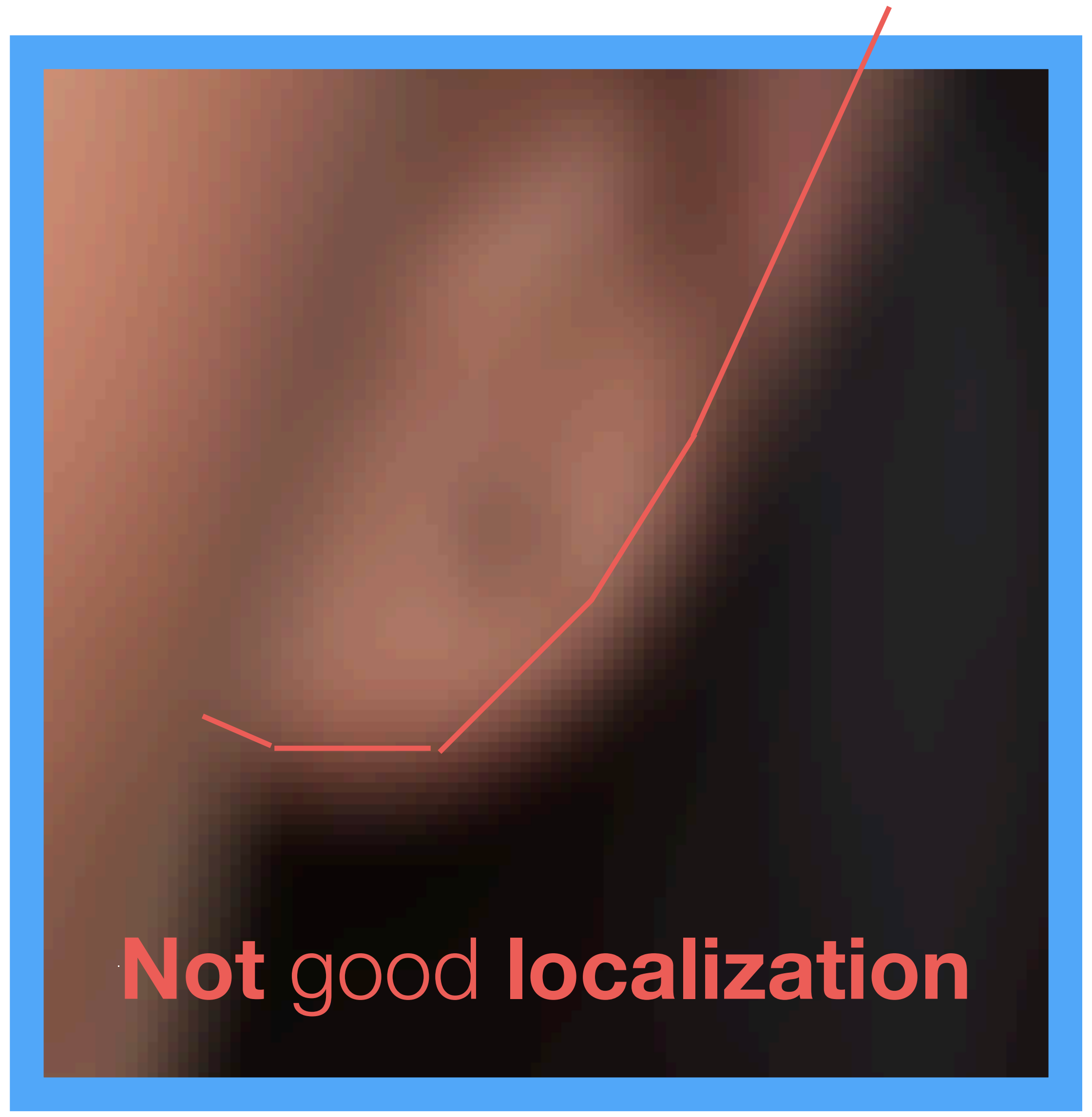


Comparing **Edge** Detectors

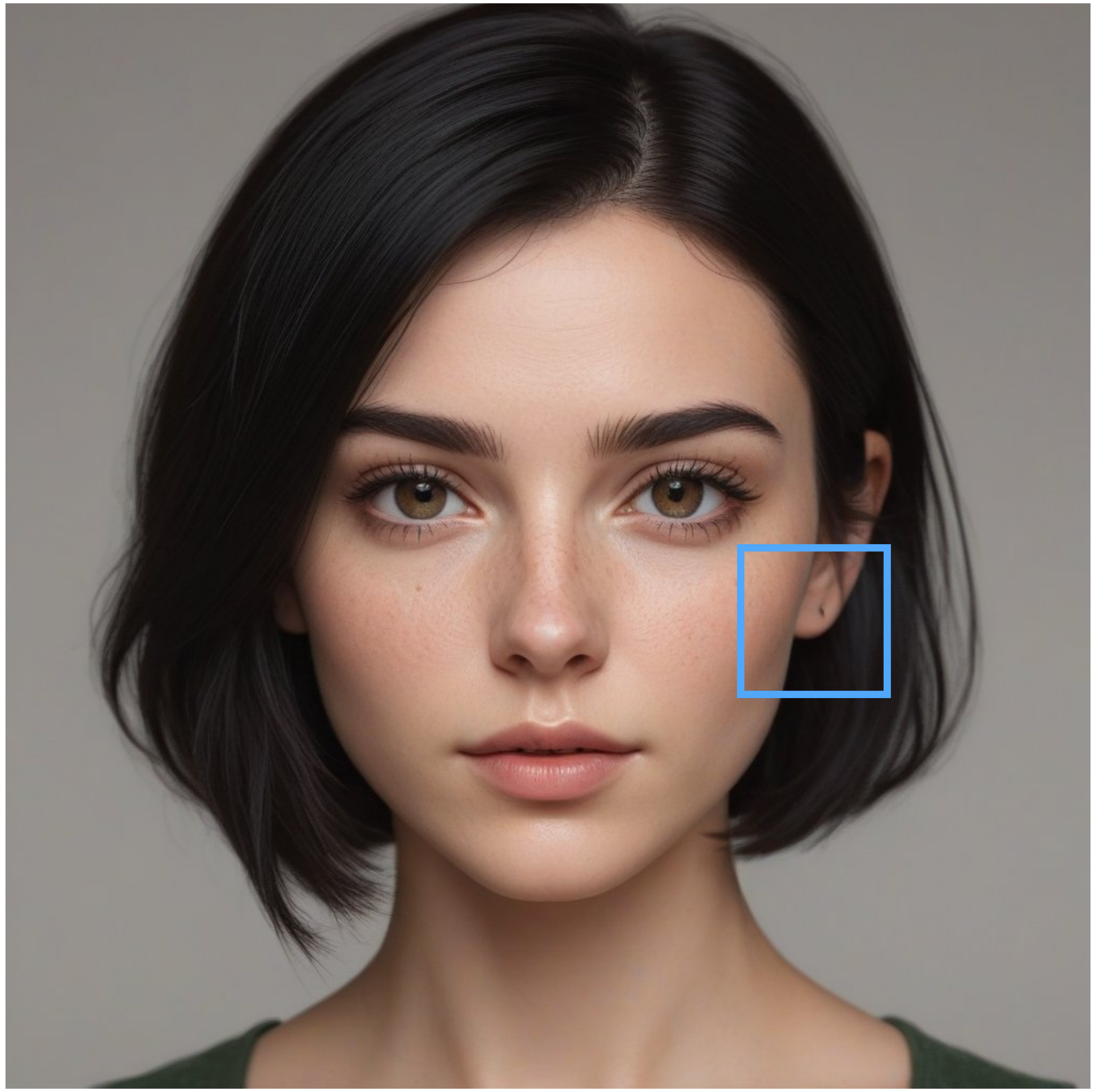


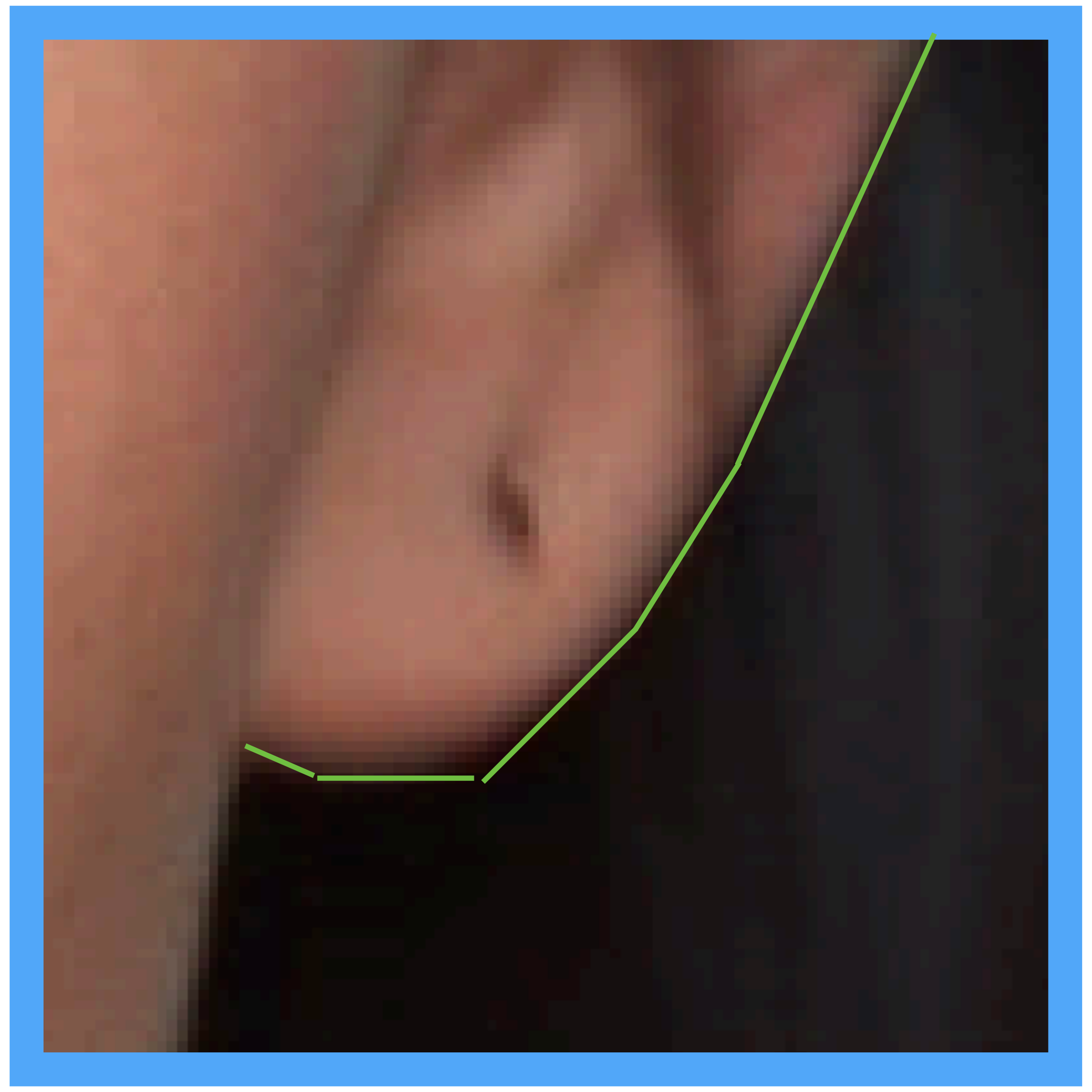


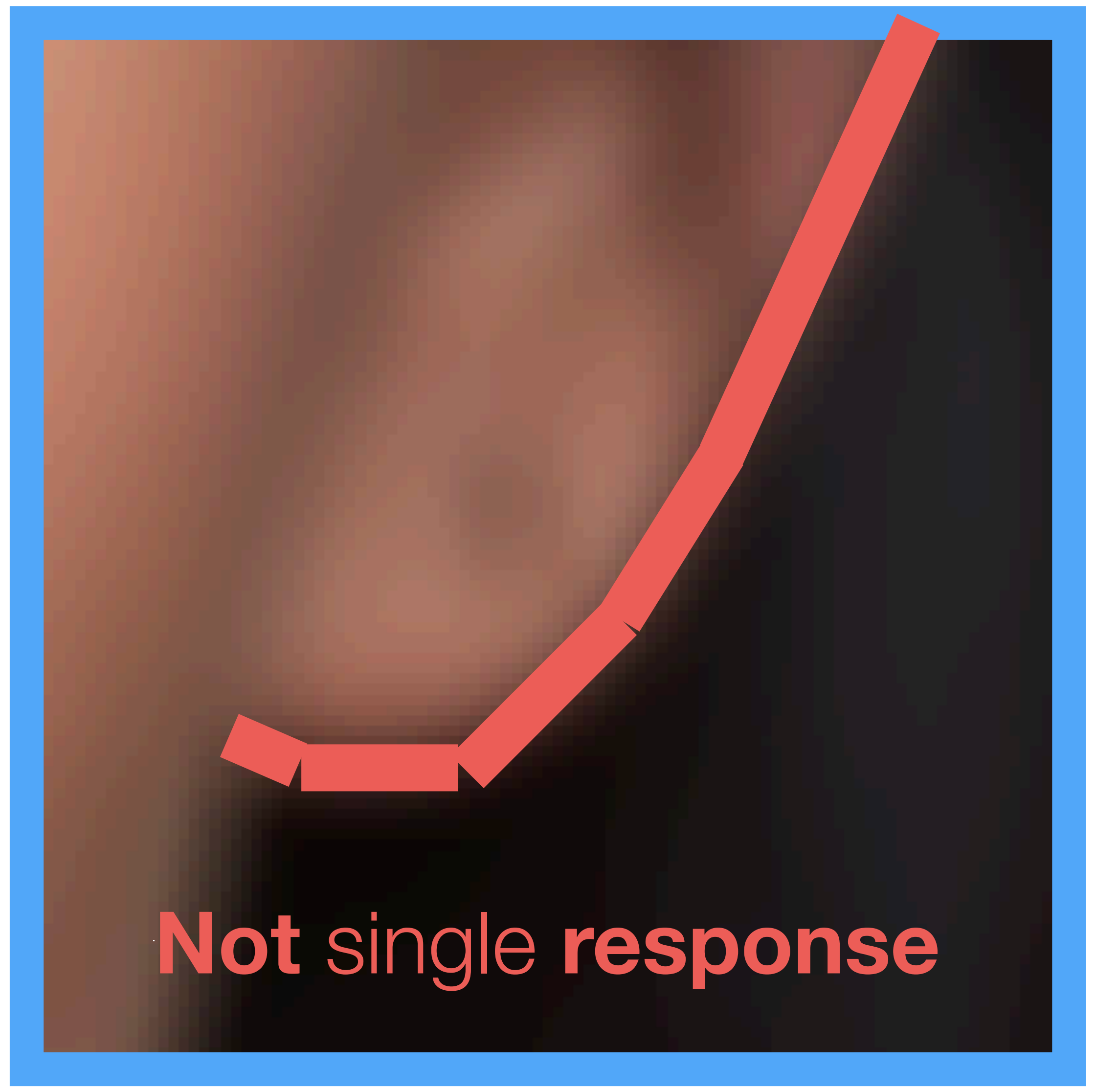


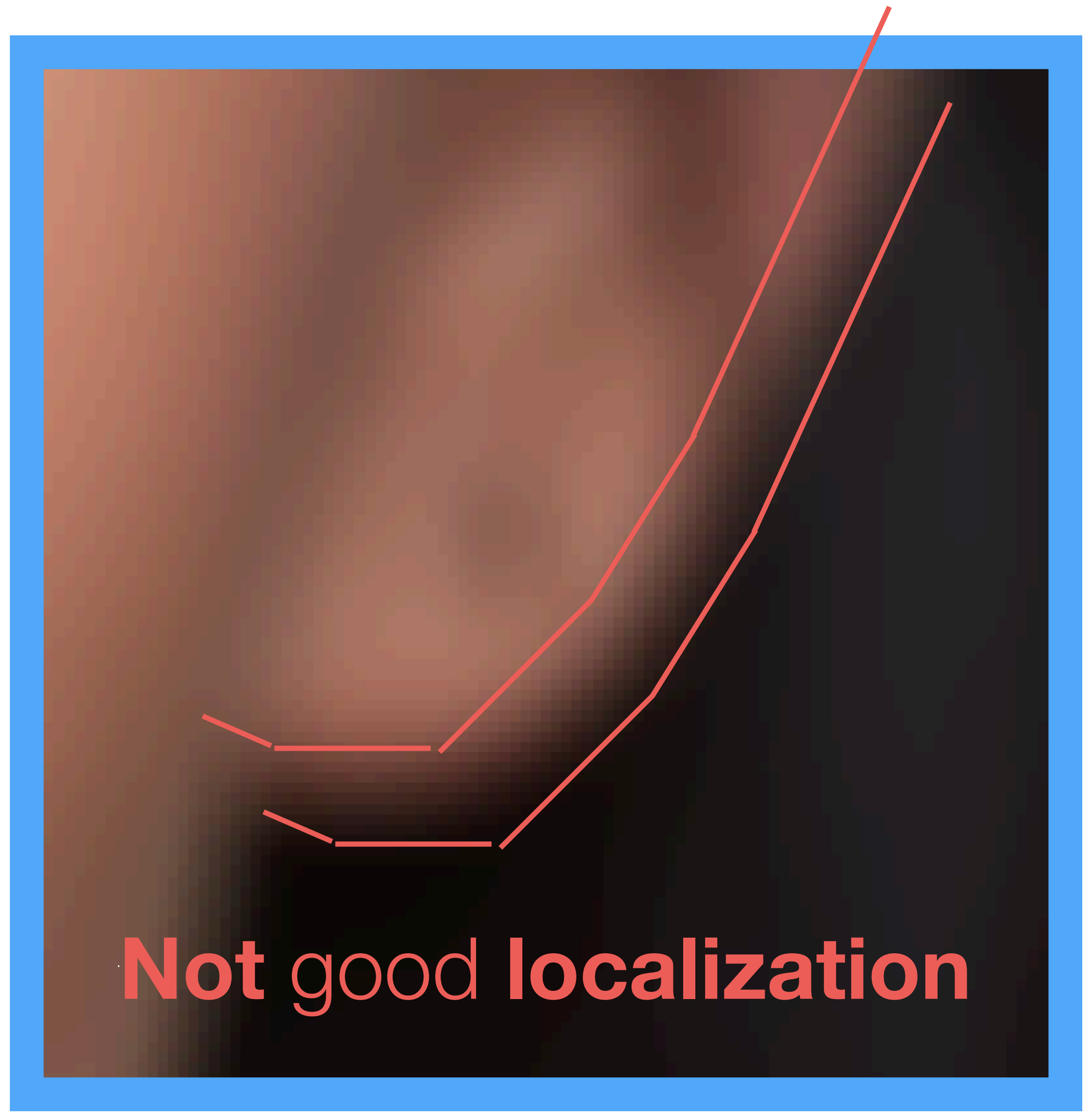


Comparing **Edge** Detectors

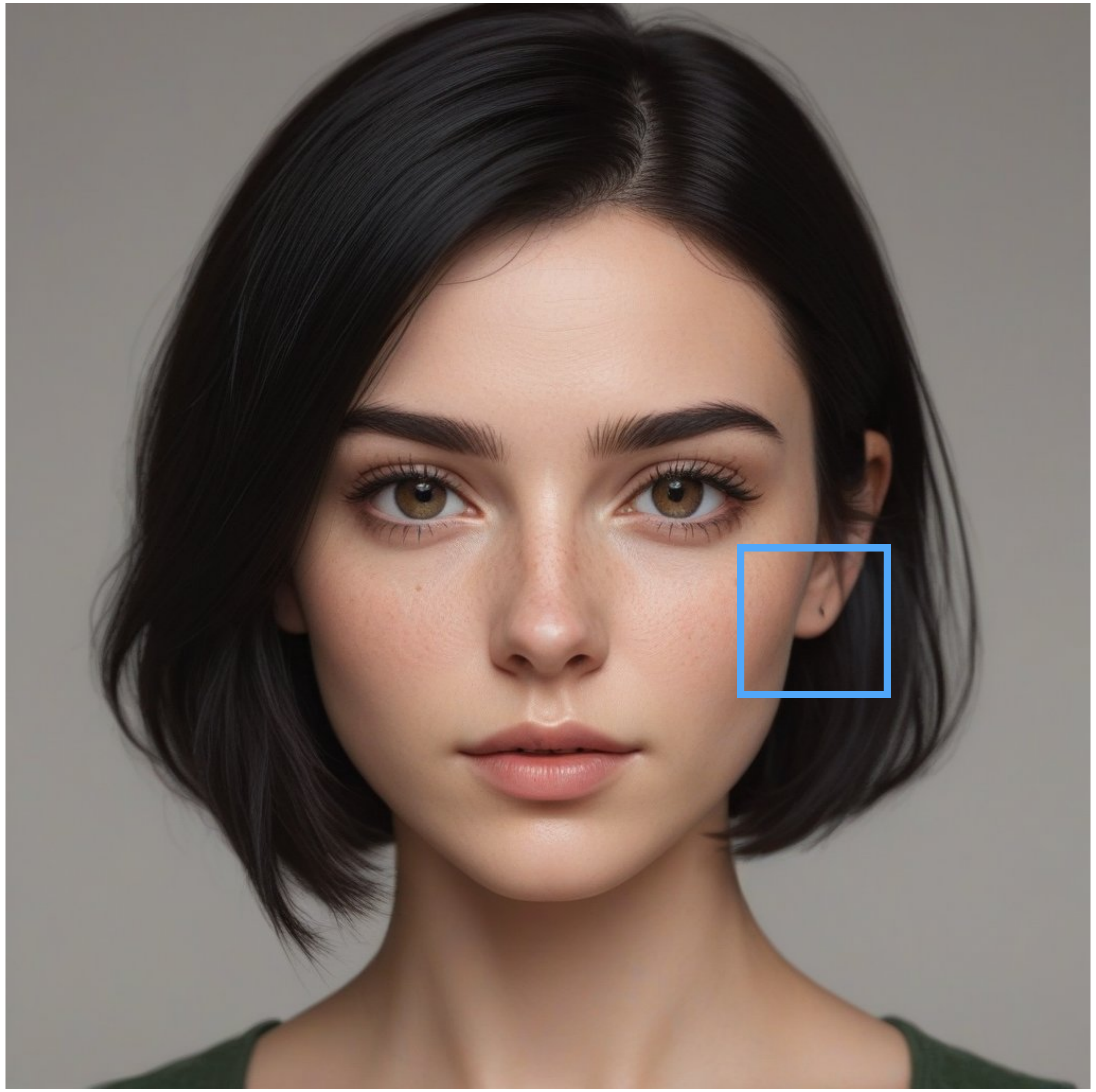


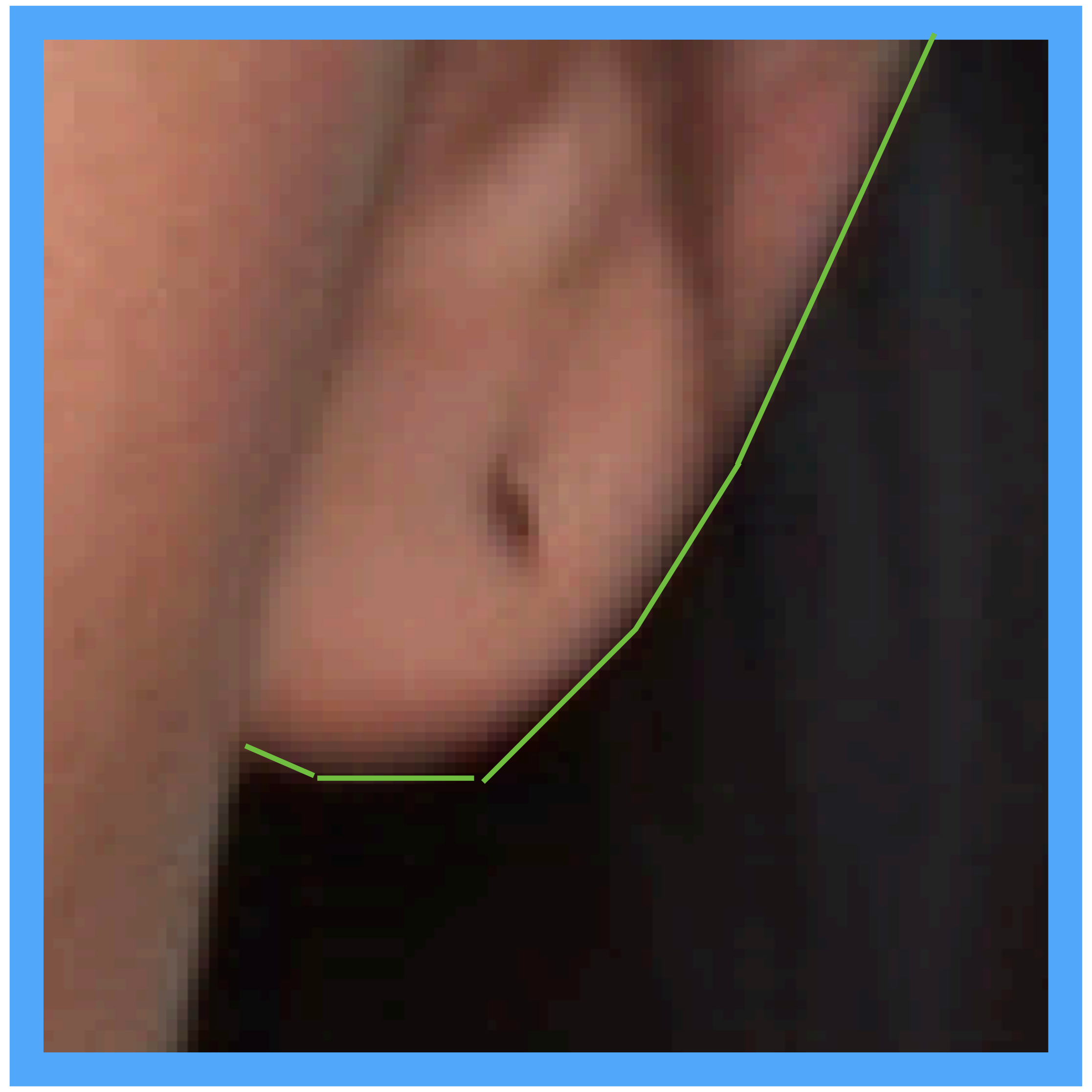


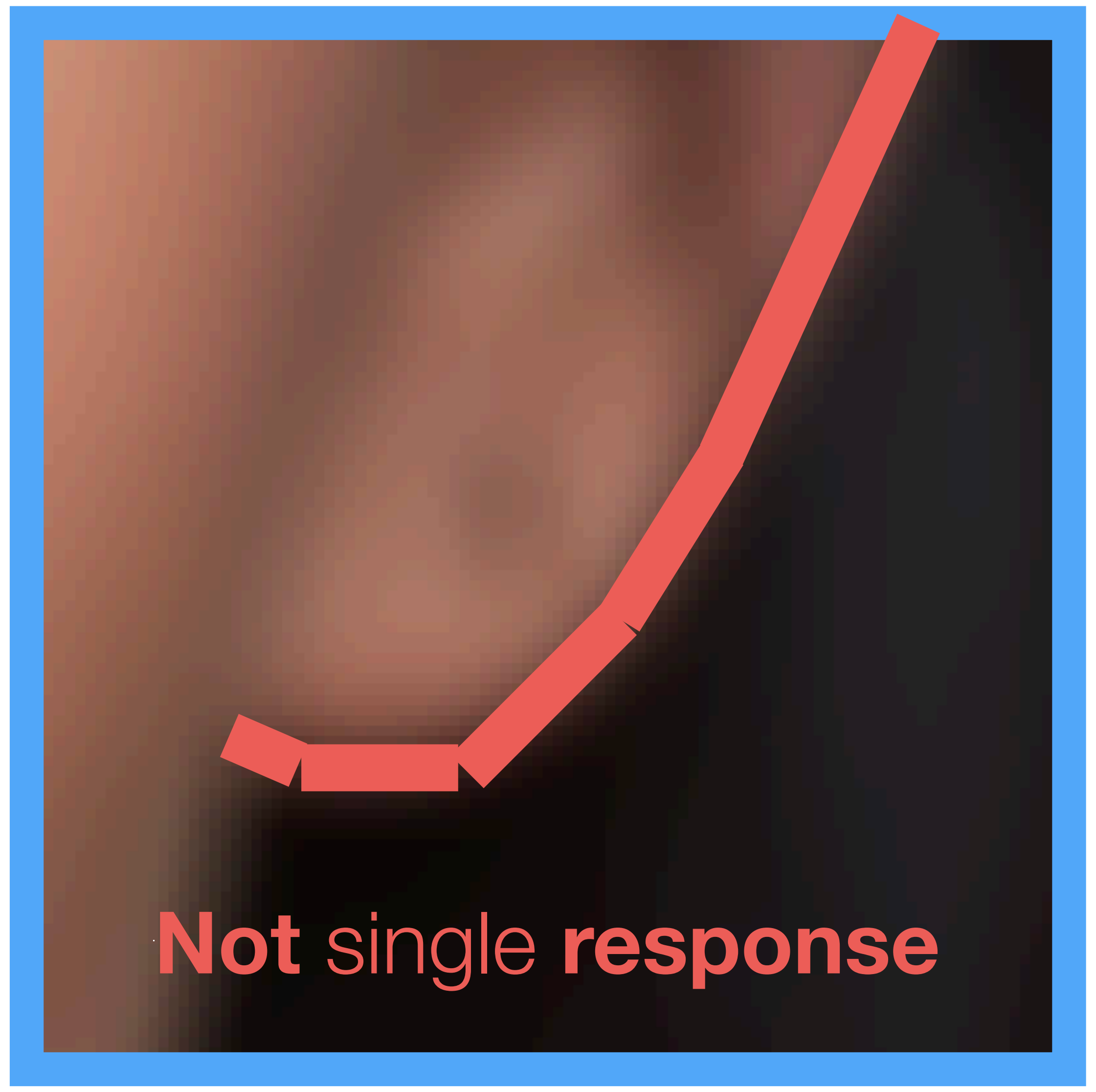


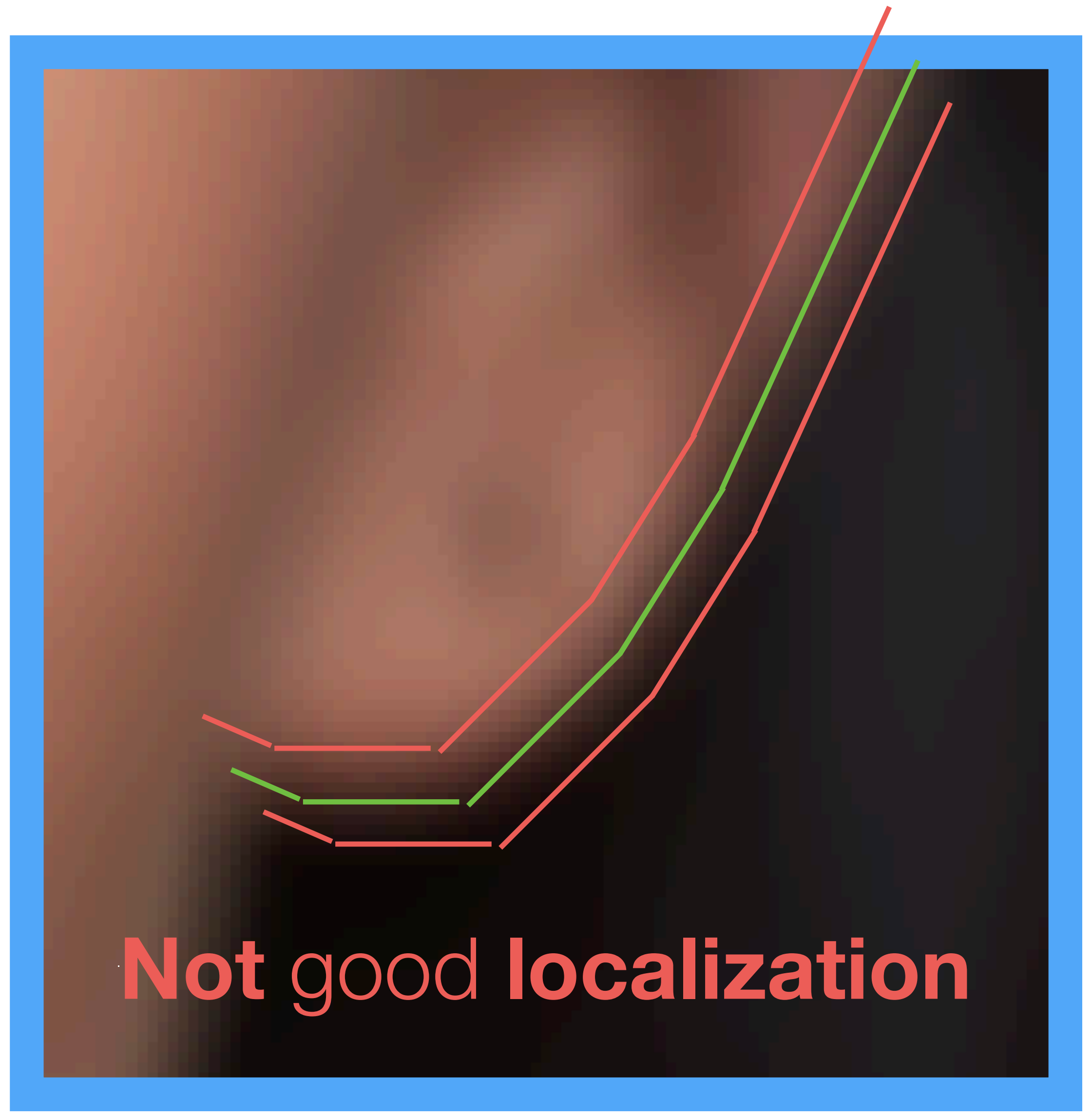


Comparing **Edge** Detectors









Comparing **Edge** Detectors

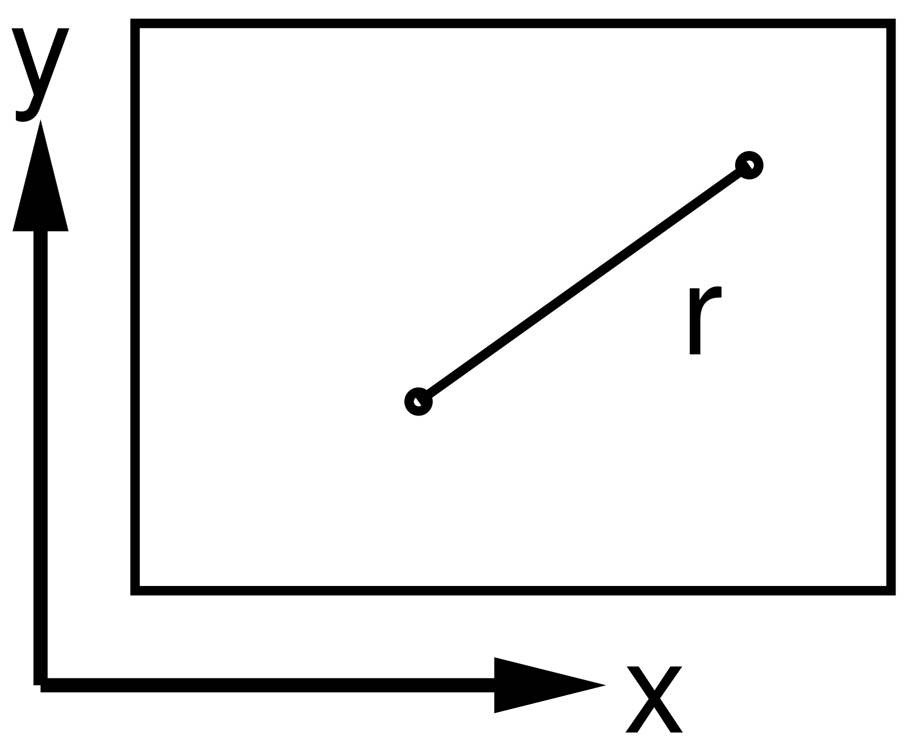
Good detection: minimize probability of false positives/negatives (spurious/missing) edges

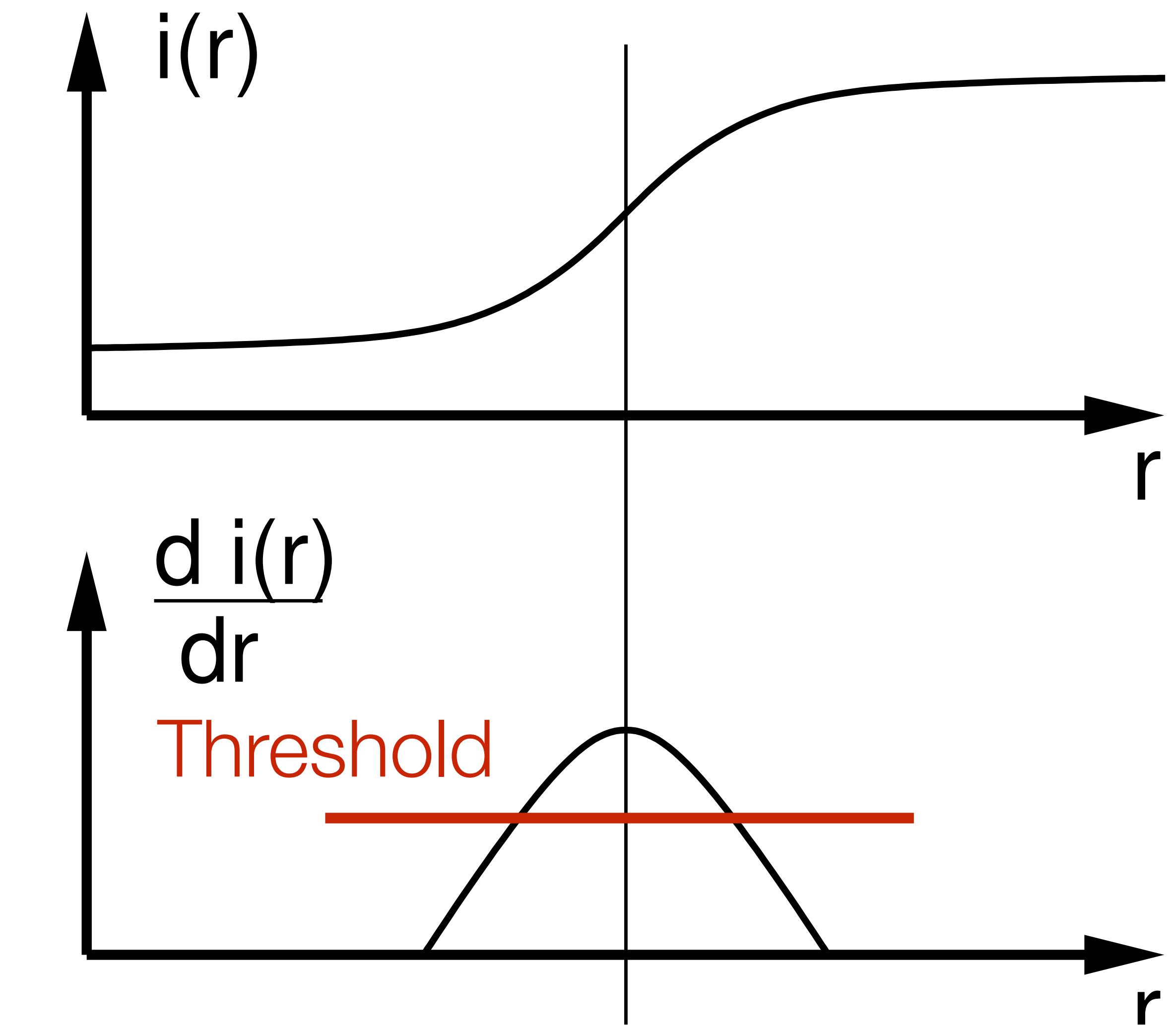
Good localization: found edges should be as close to true image edge as possible

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

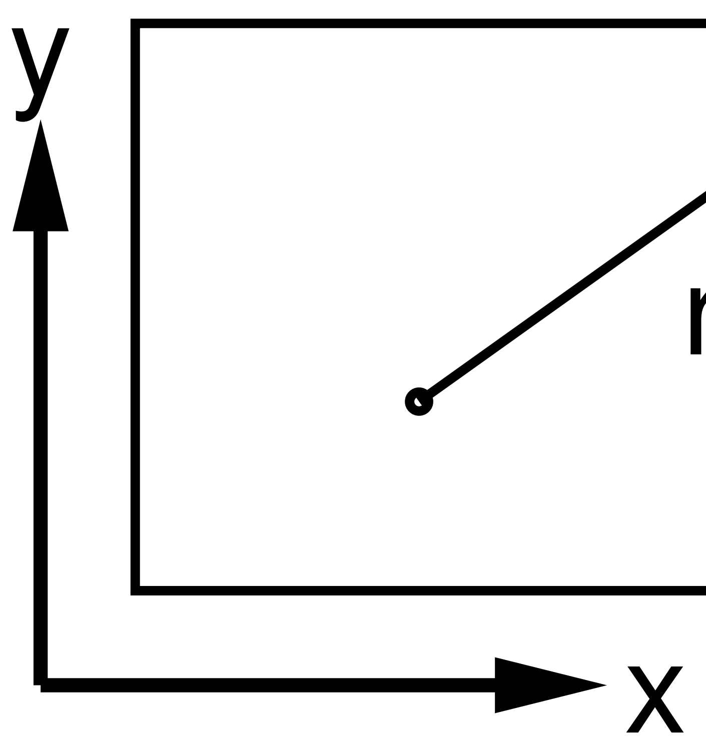
	Approach	Detection	Localization	Single Resp	Limitations
Sobel	Gradient Magnitude Threshold	Good	Poor	Poor	Results in Thick Edges

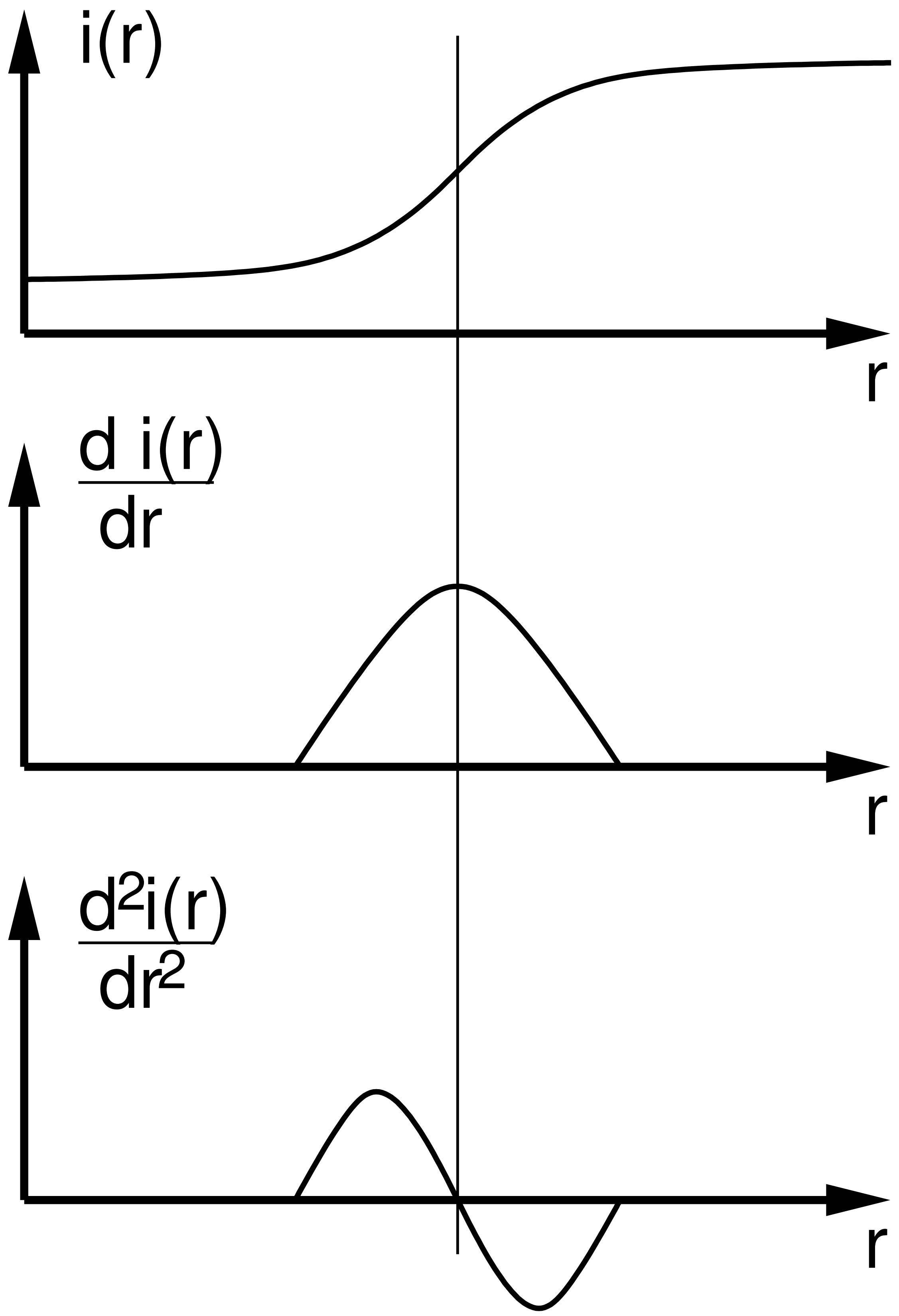
Two Generic Approaches for **Edge** Detection





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

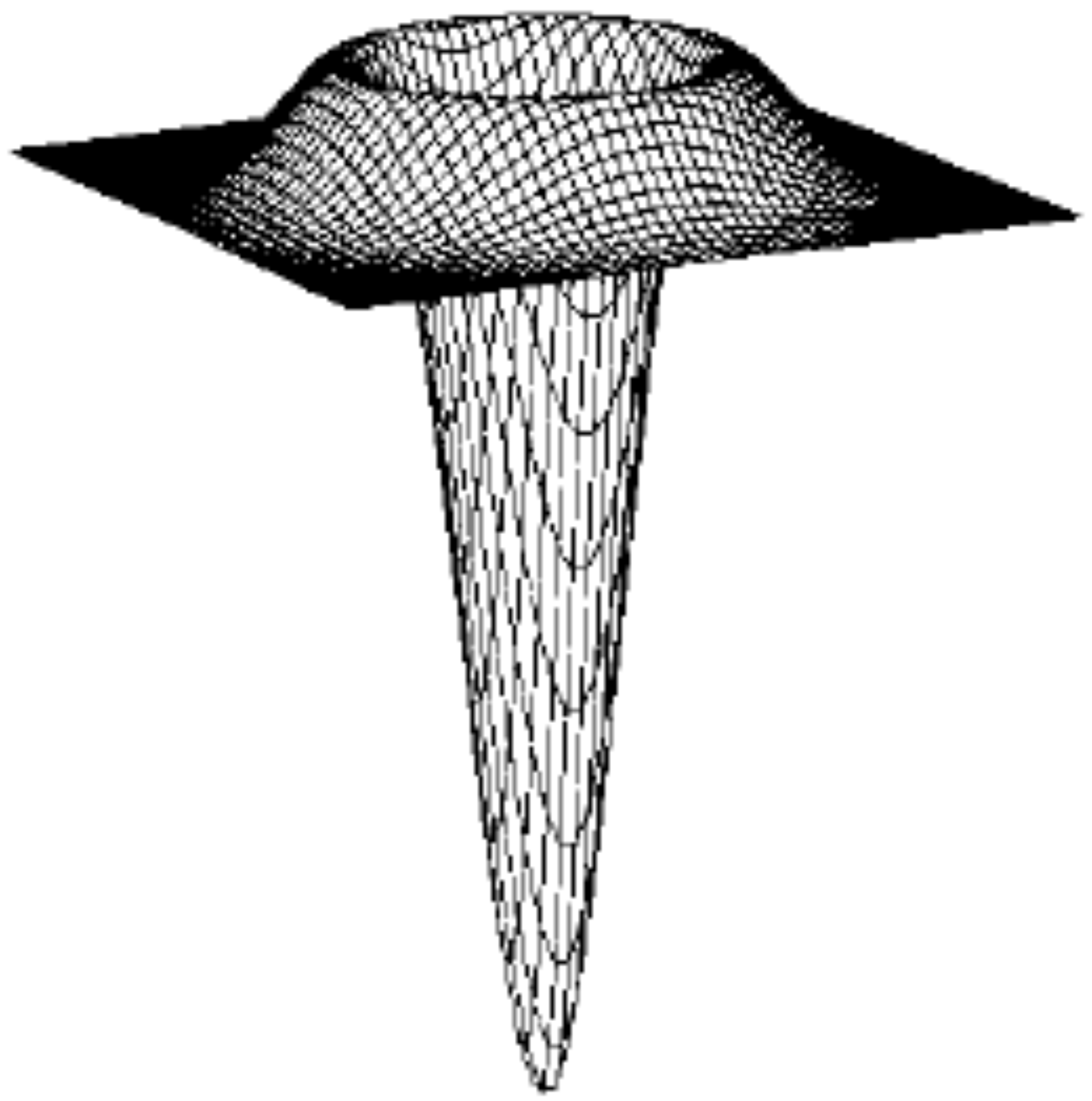
$$\nabla^2 f(x,y) = \frac{\partial^2 f(x,y)}{\partial x^2} + \frac{\partial^2 f(x,y)}{\partial y^2}$$

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

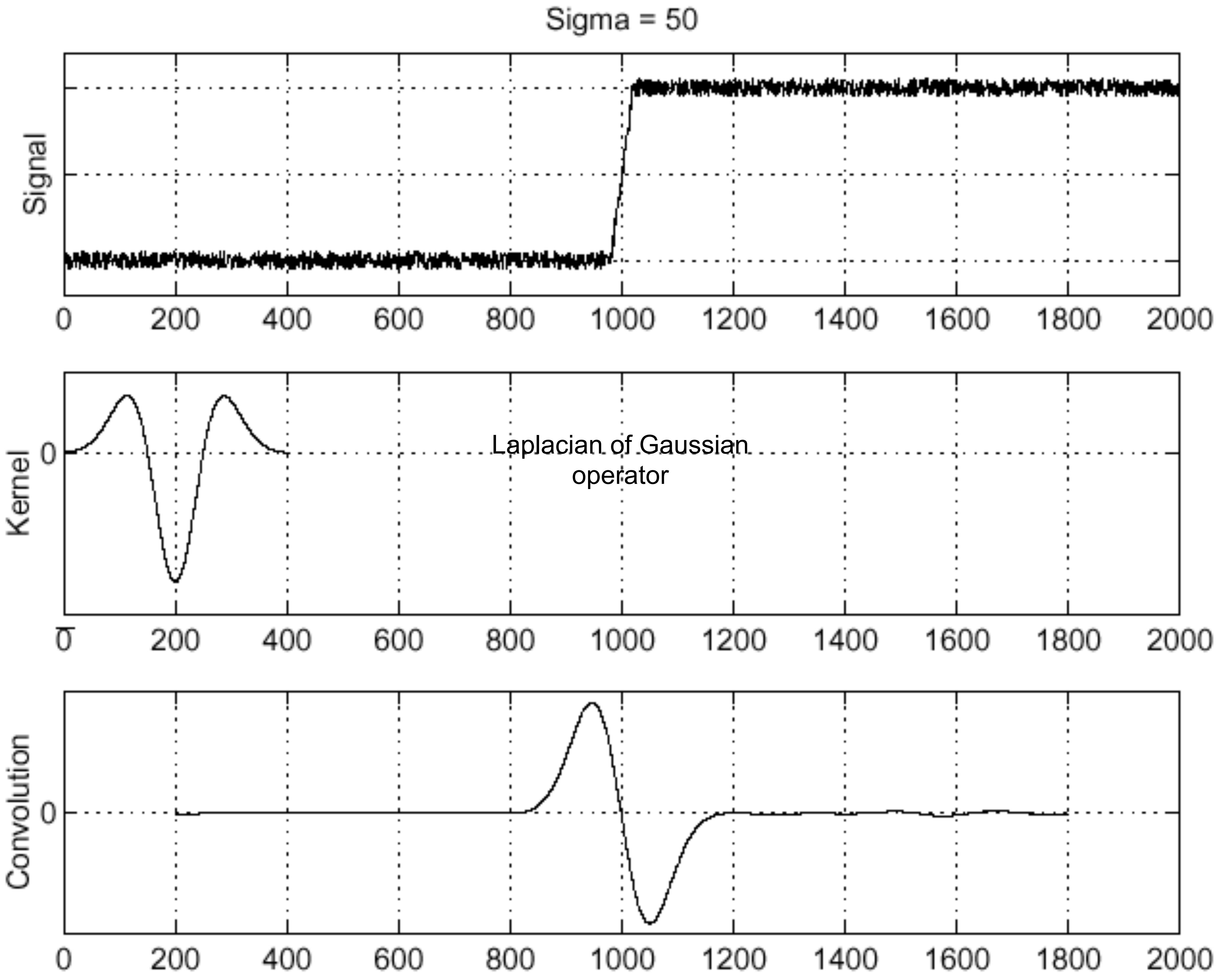
1D **Example**: Continued

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

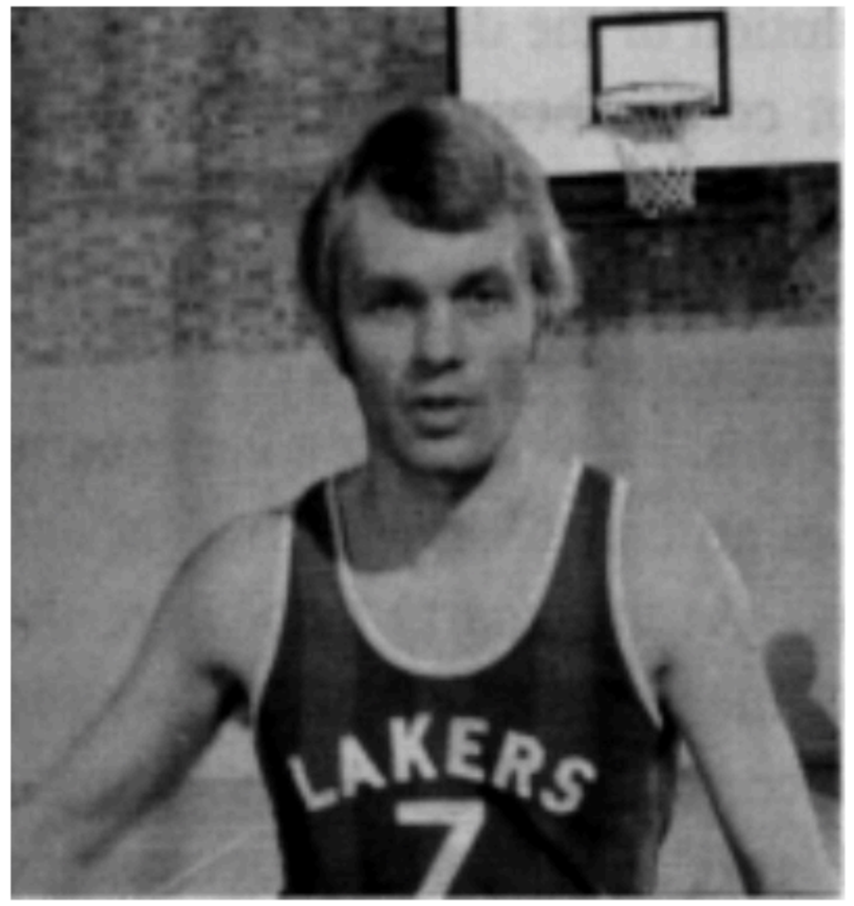
17 x 17 LoG filter

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

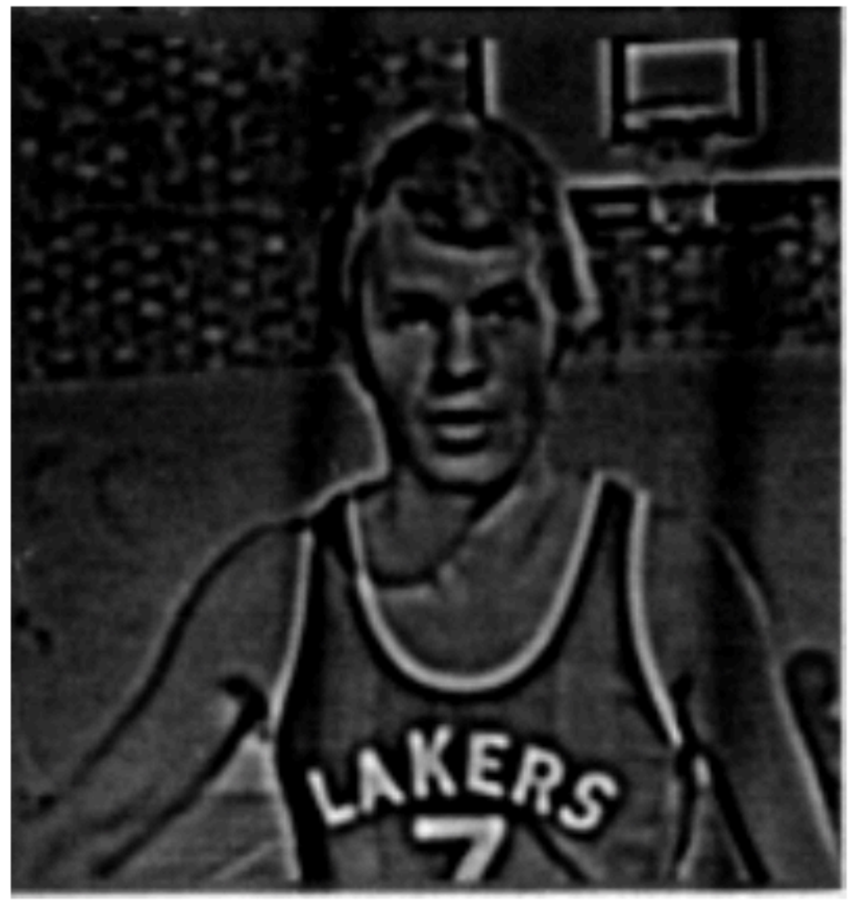
Scale (σ)

Image From: A. Campilho

Marr / Hildreth **Laplacian of Gaussian**



Original Image



LoG Filter

Zero Crossings

Scale (σ)

Image From: A. Campilho

Assignment 1: High Frequency Image

original

\-

smoothed
(5x5 Gaussian)

=

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

Assignment 1: High Frequency Image

original

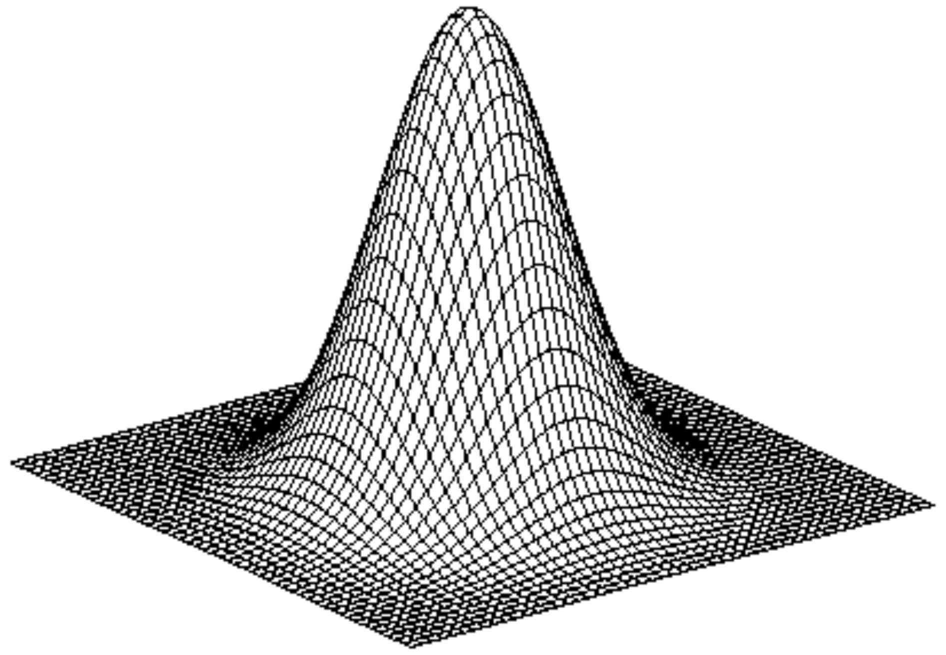
-

smoothed
(5x5 Gaussian)

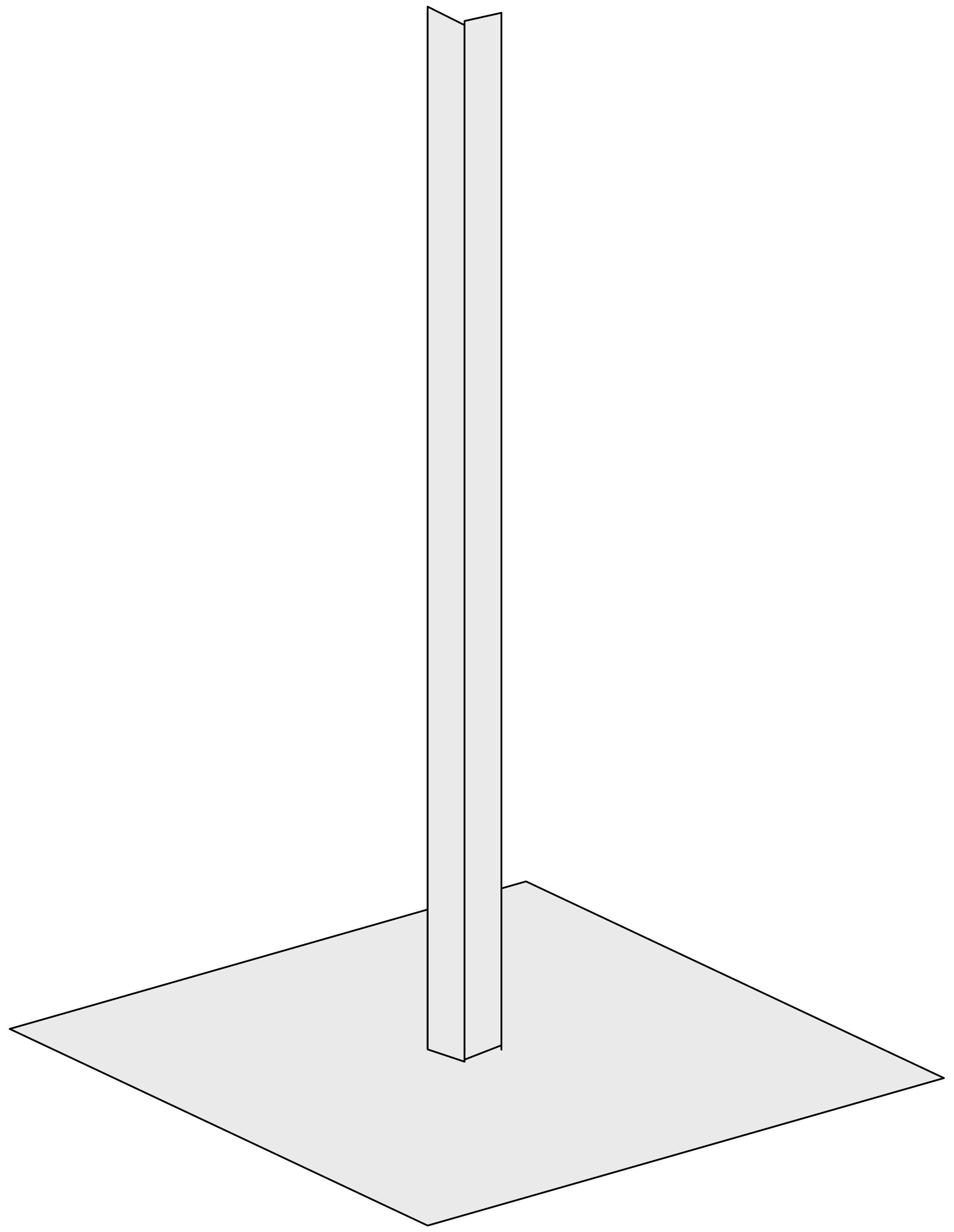
=

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

Assignment 1: High Frequency Image

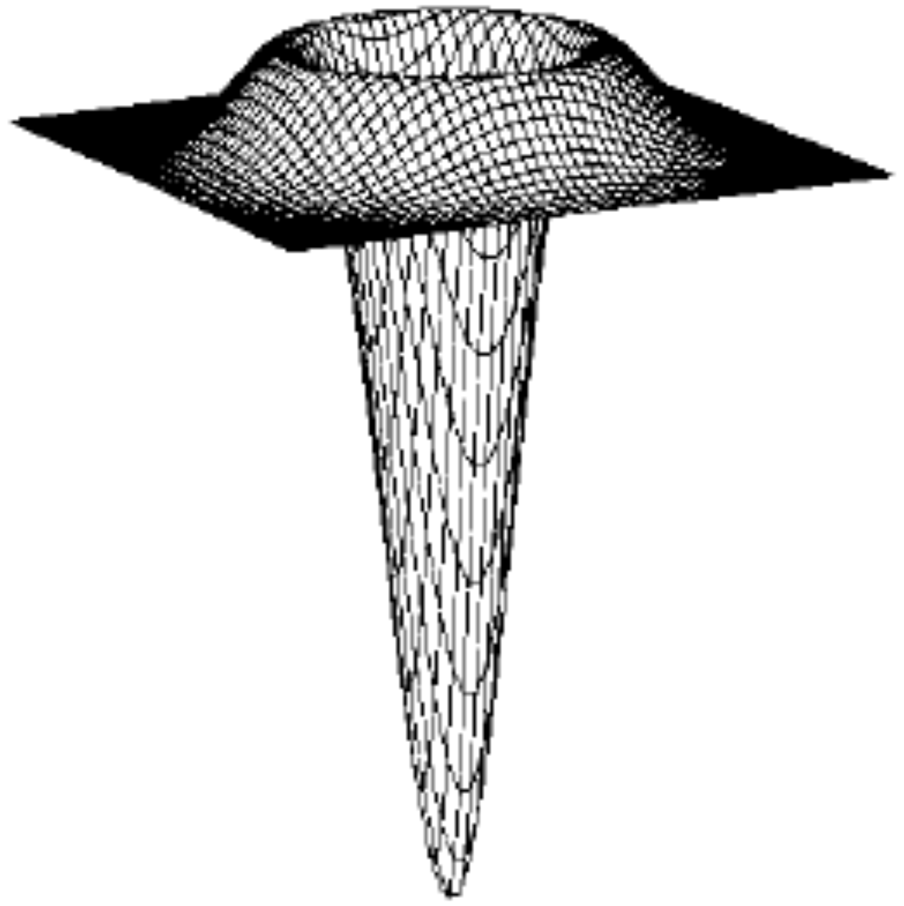


Gaussian

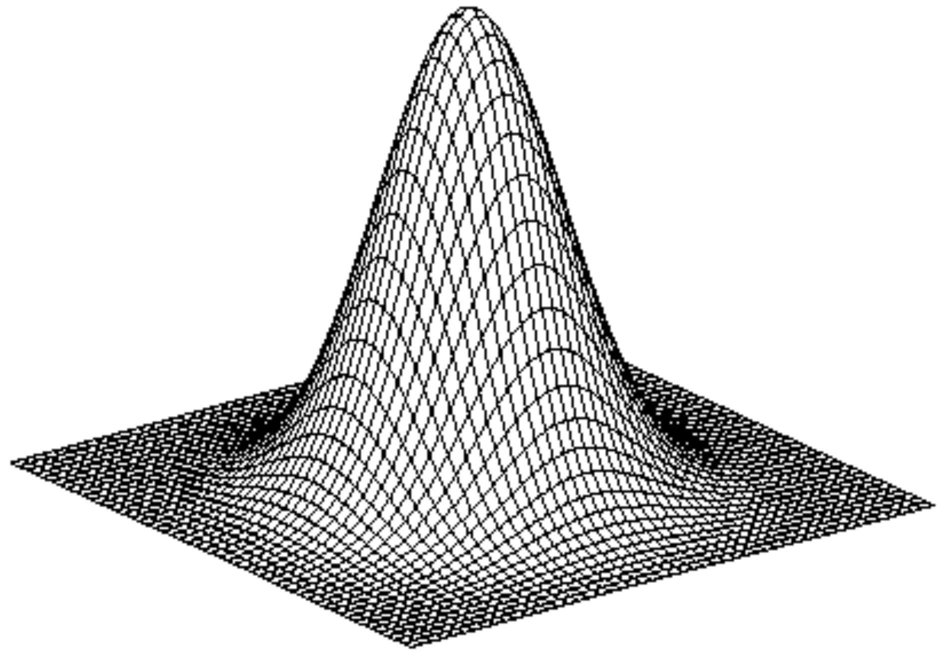


delta function

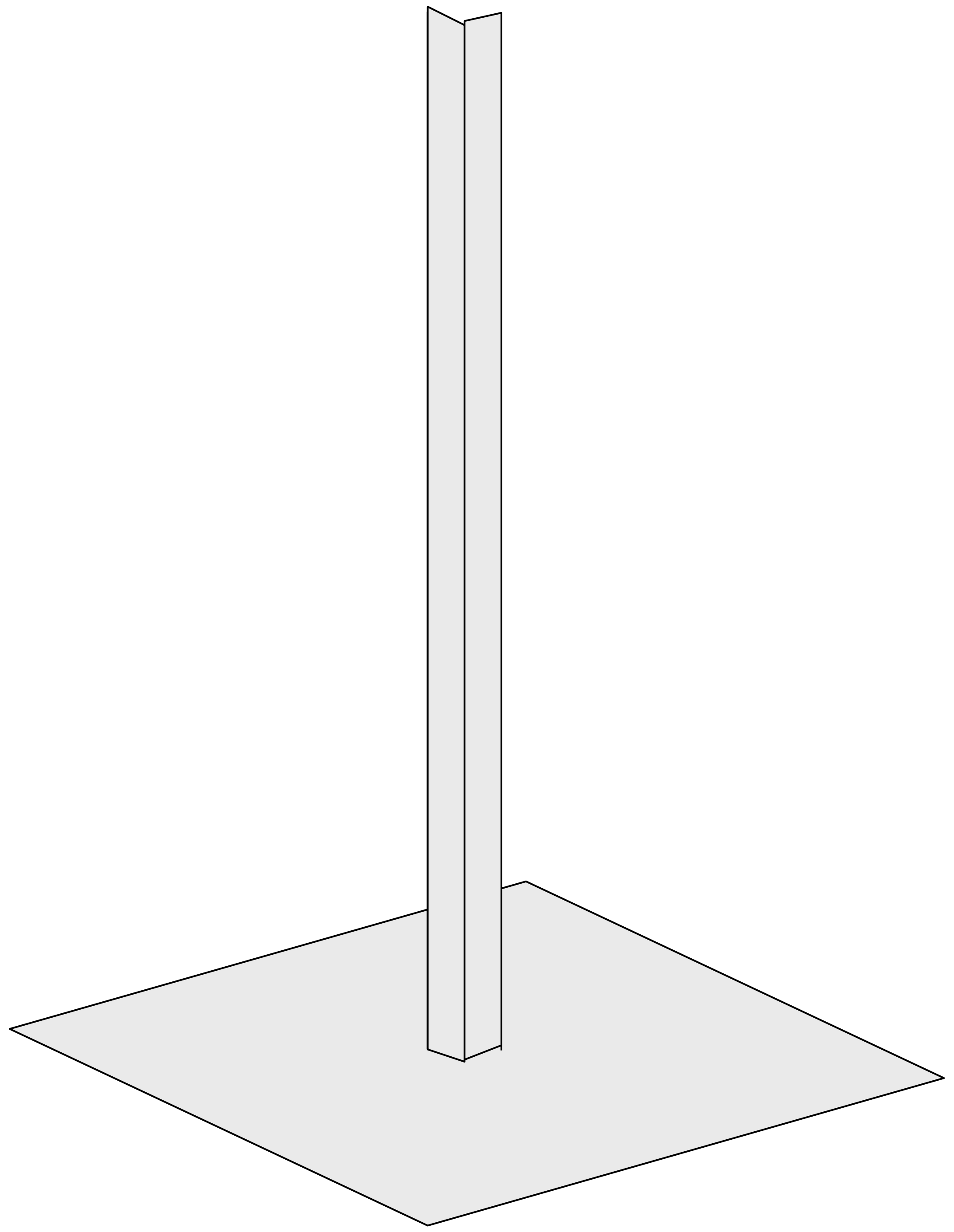
Laplacian of Gaussian



Assignment 1: High Frequency Image

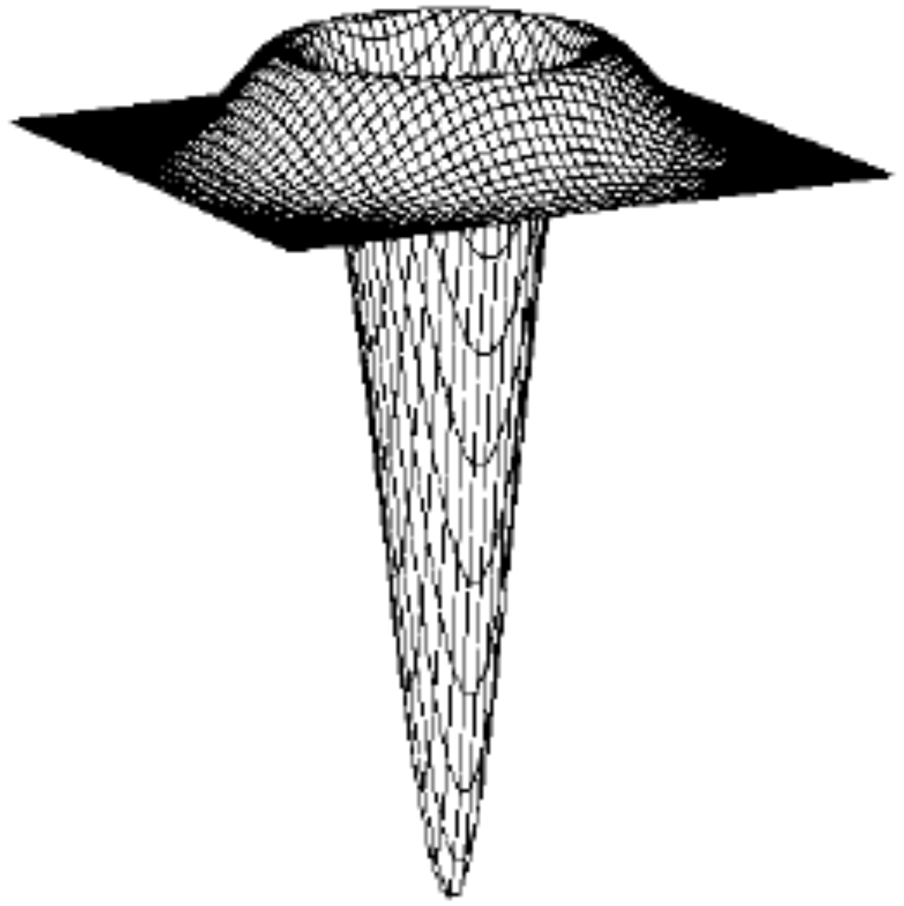


Gaussian



delta function

Laplacian of Gaussian



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

Comparing **Edge** Detectors

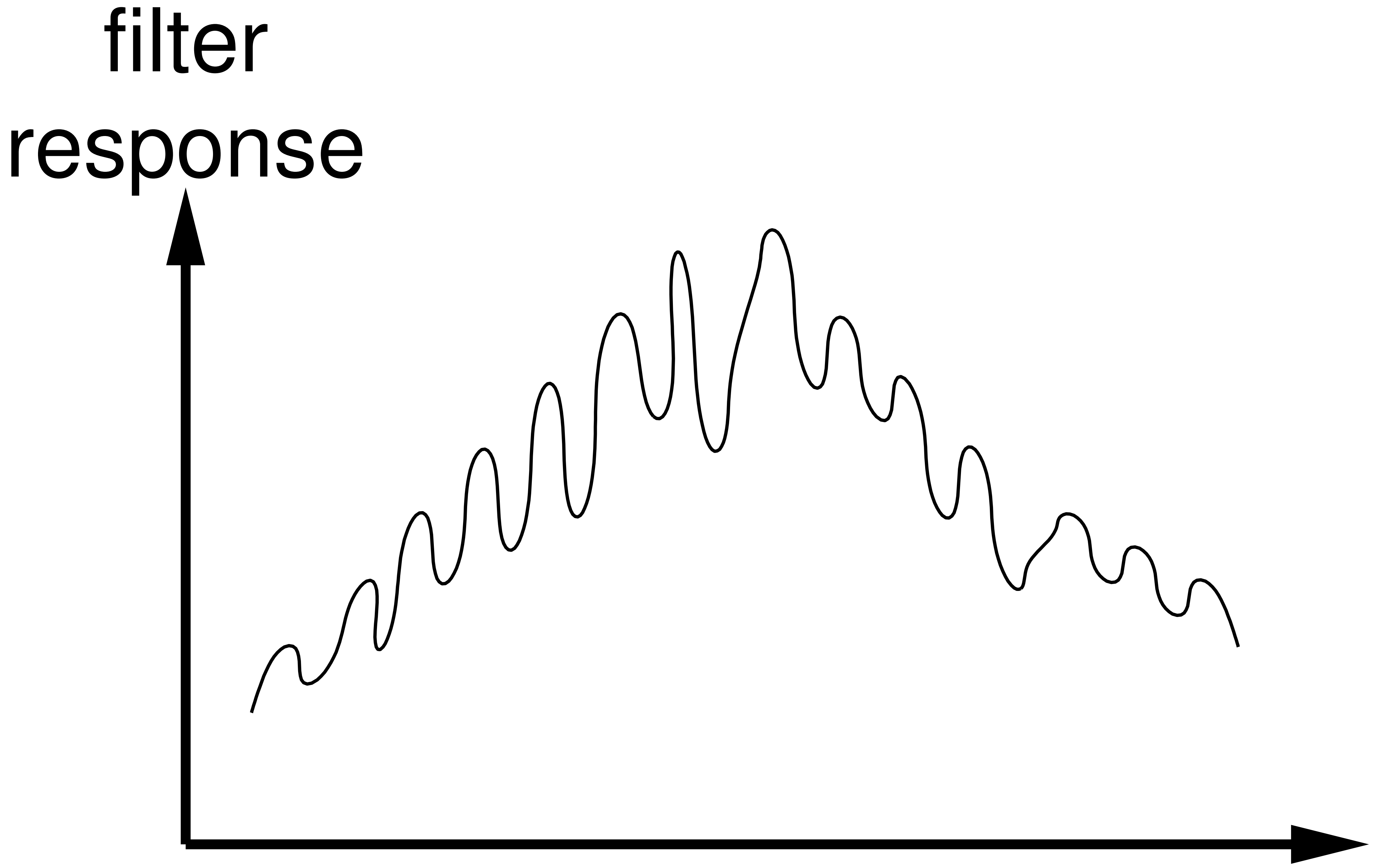
Good detection: minimize probability of false positives/negatives (spurious/missing) edges

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	

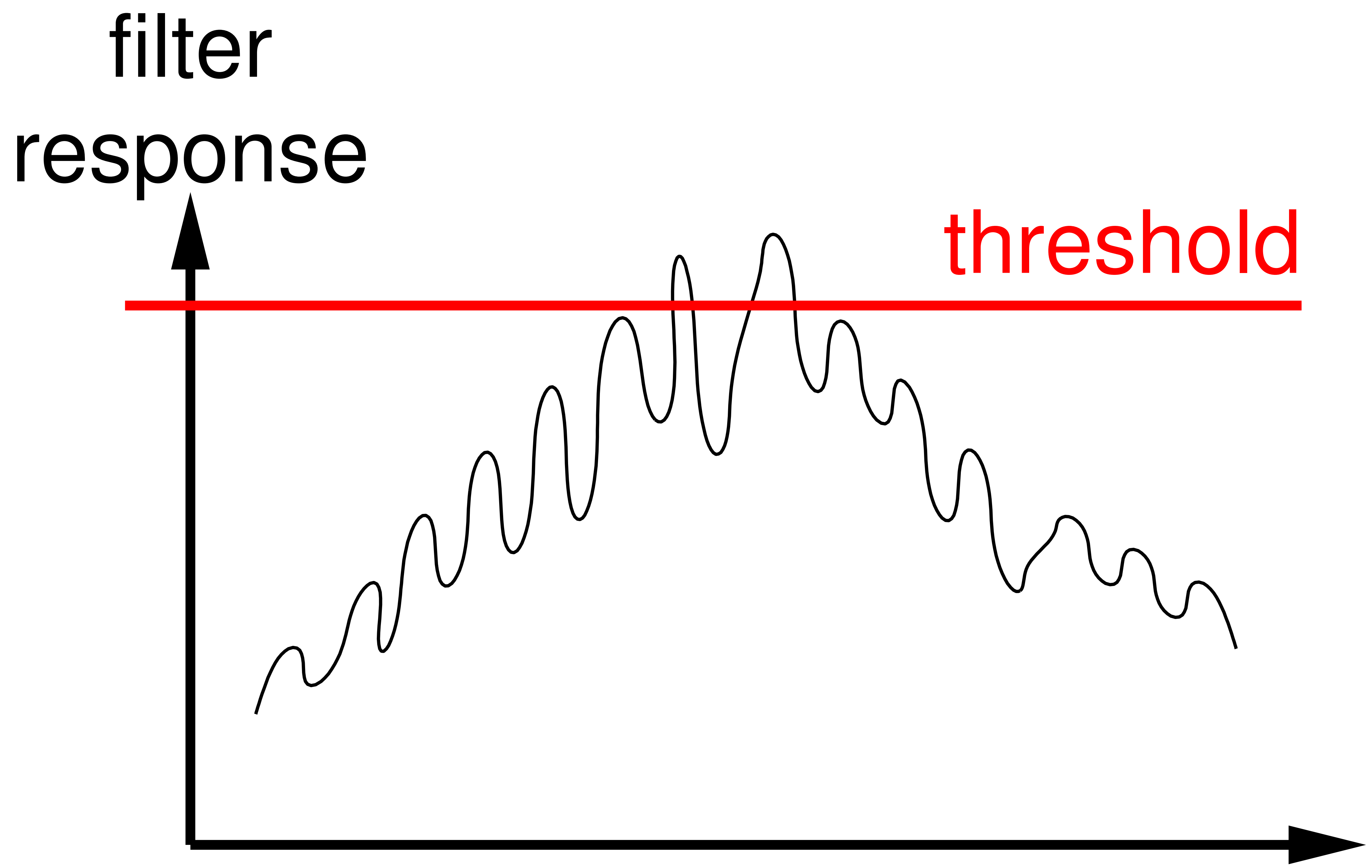
Example: Edge Detection



Question: How many edges are there?

Question: What is the position of each edge?

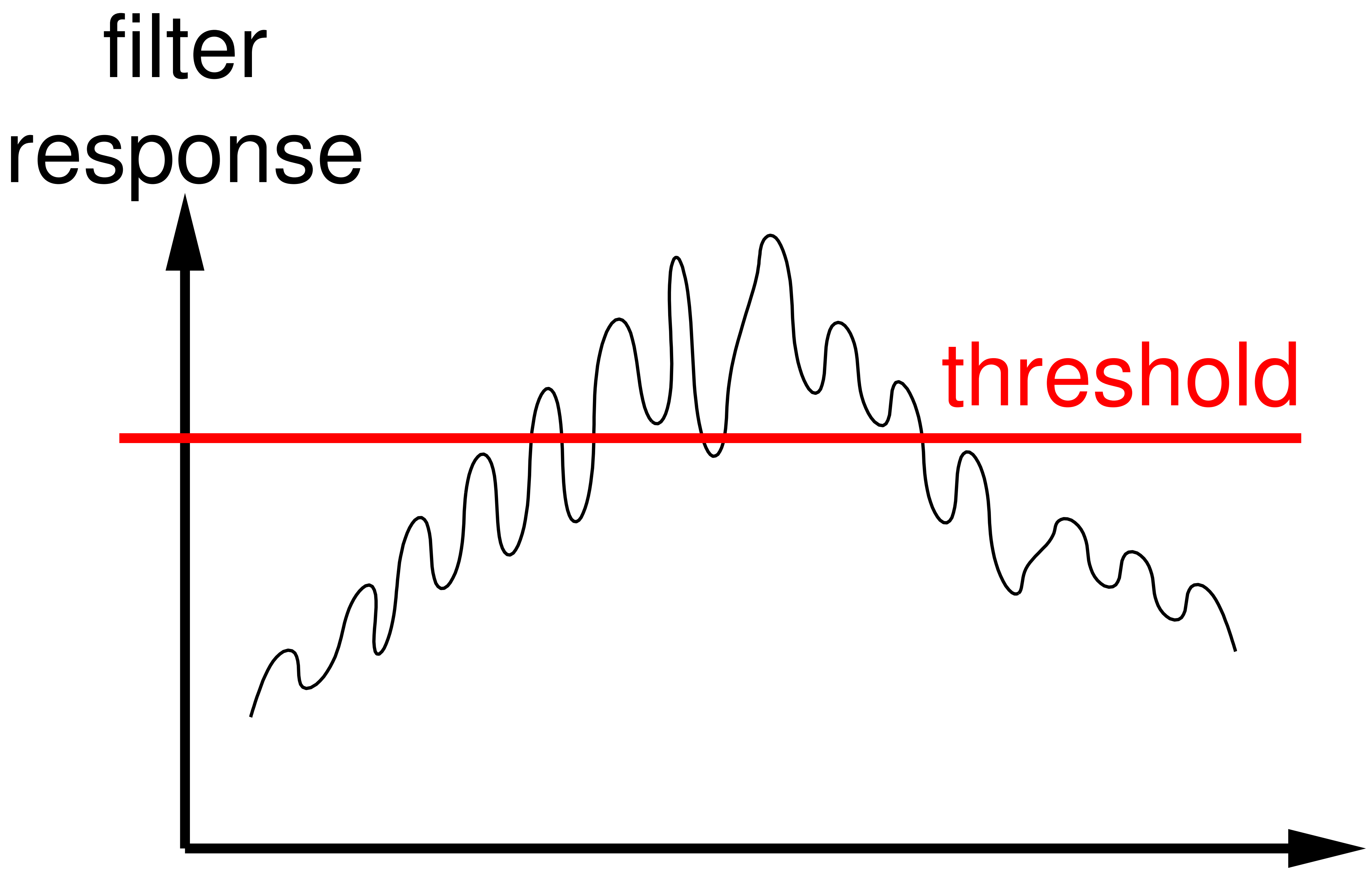
Example: Edge Detection



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

Steps:

1. Apply **directional derivatives** of Gaussian

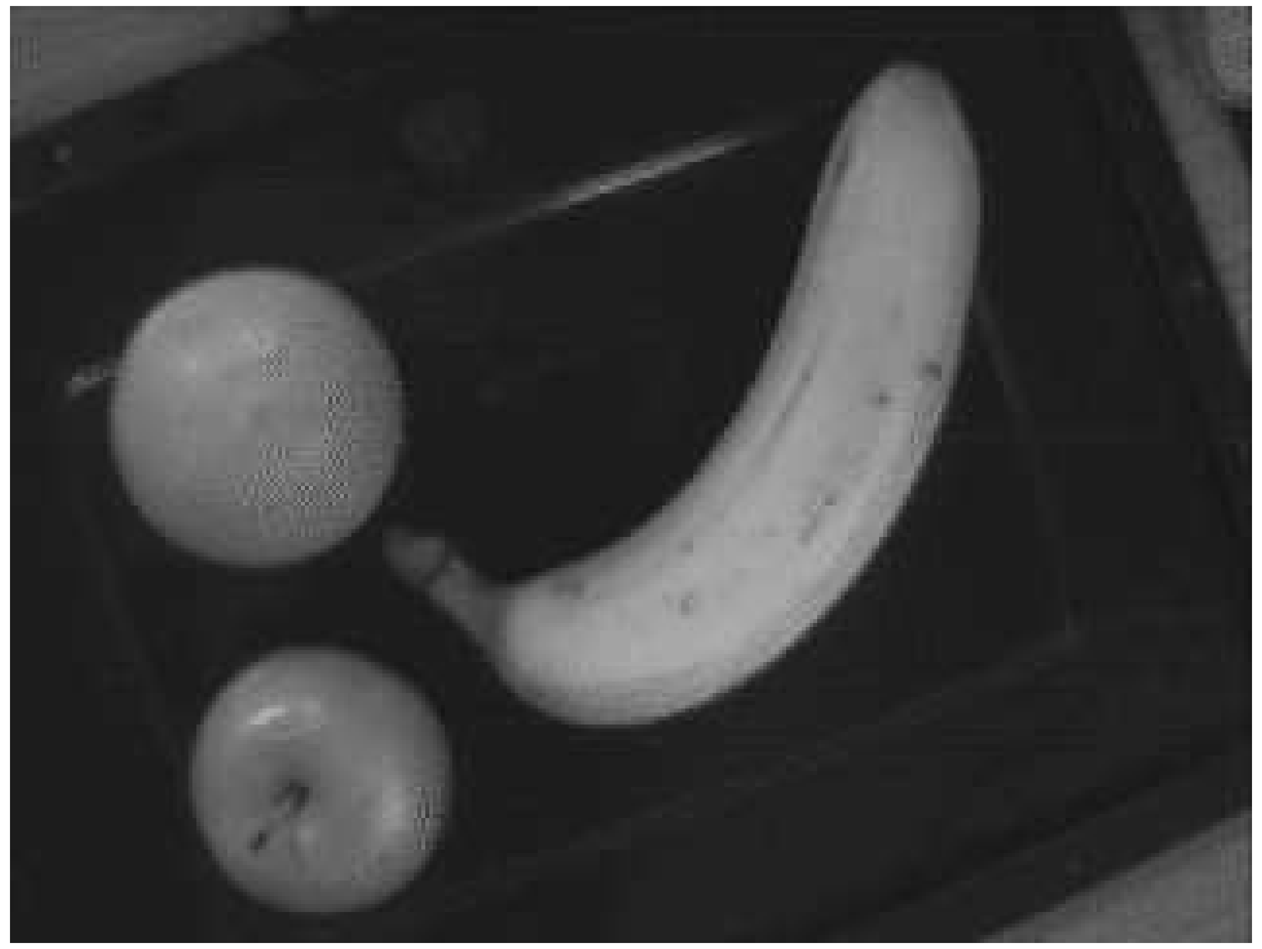
2. Compute **gradient magnitude** and **gradient direction**

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

Canny Edge Detector

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 $\pm\nabla I$)

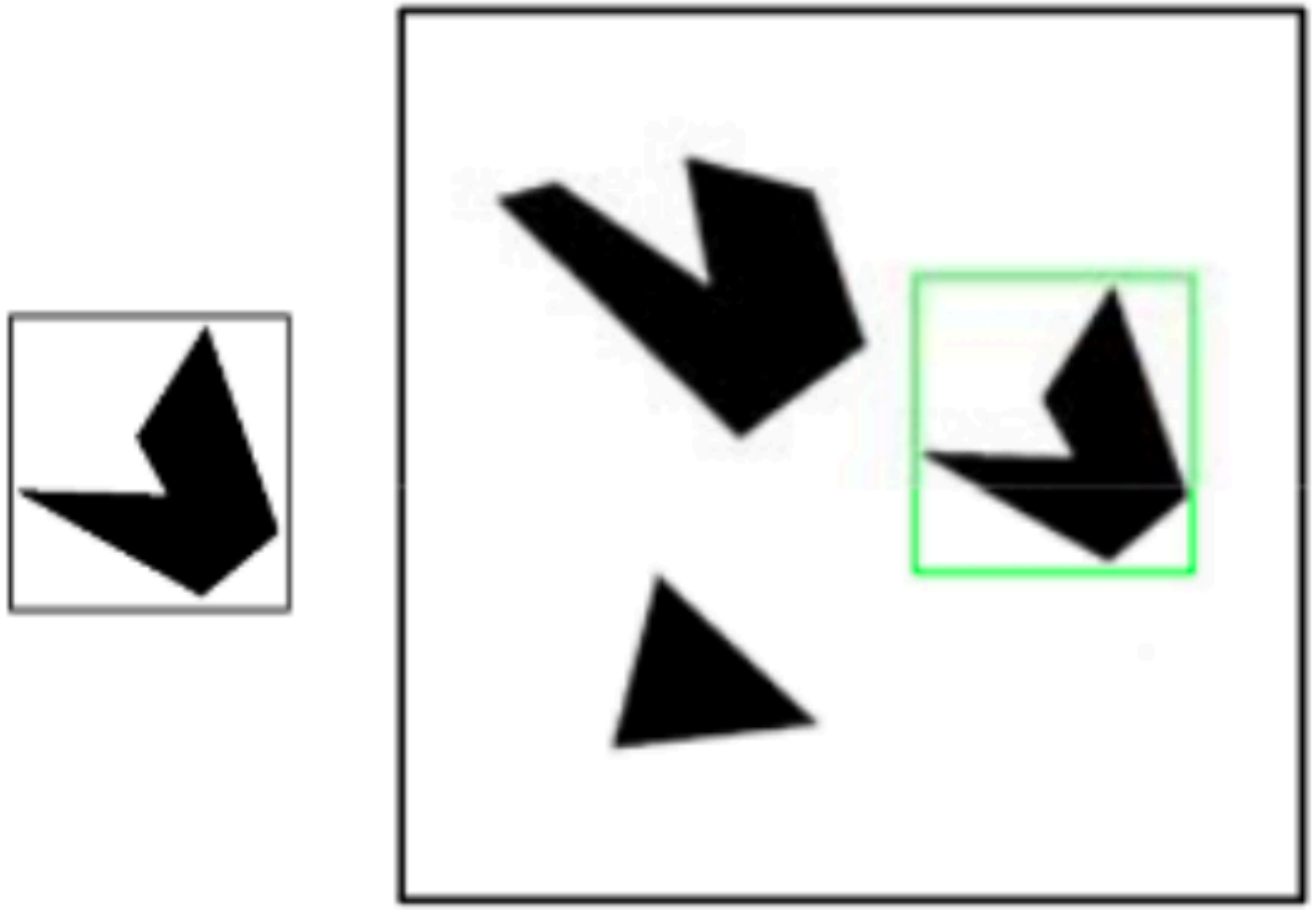
[Canny 1986]

Non-maxima Suppression

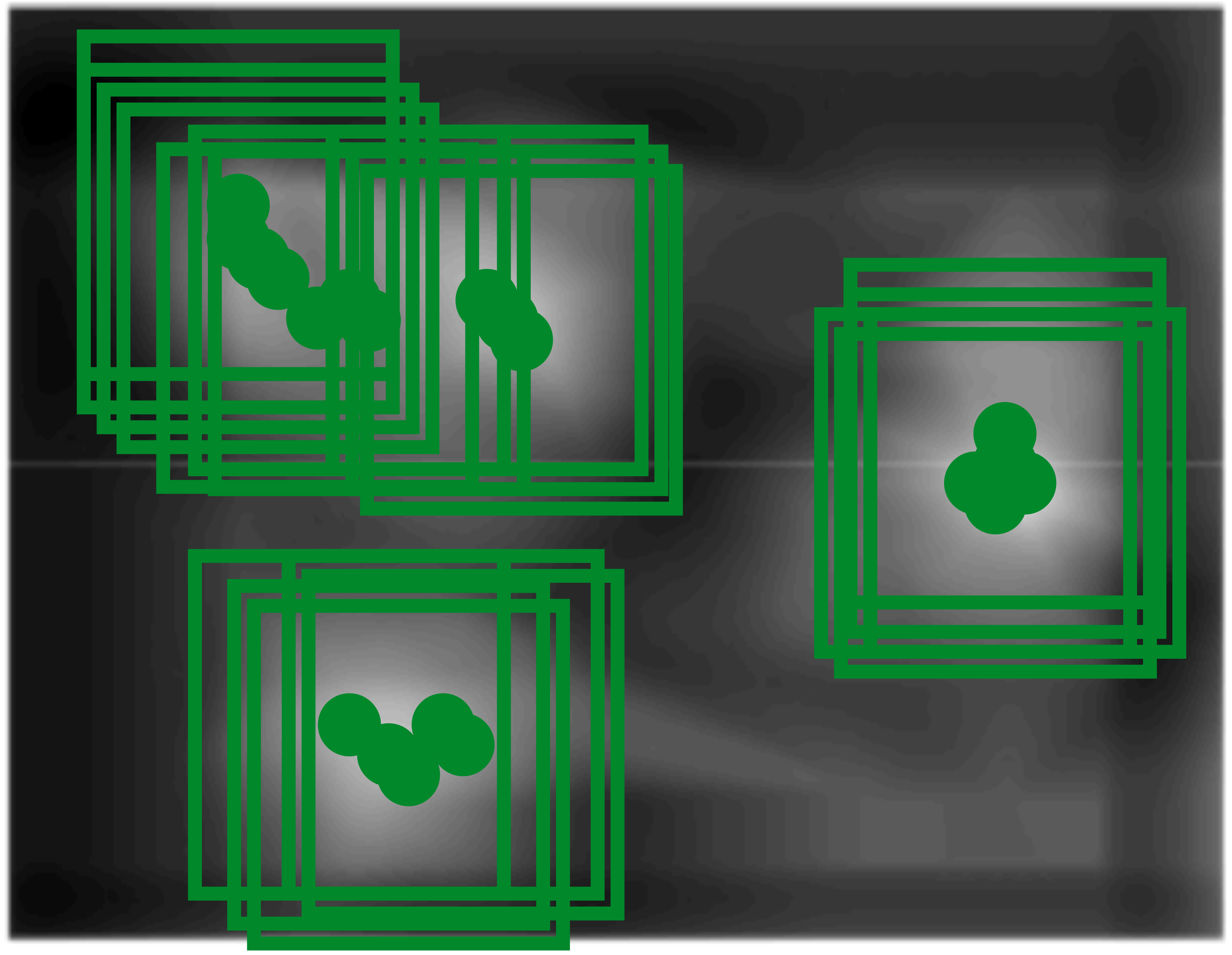
Idea: suppress near-by similar detections to obtain one “true” result

Non-maxima Suppression

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



Detected template

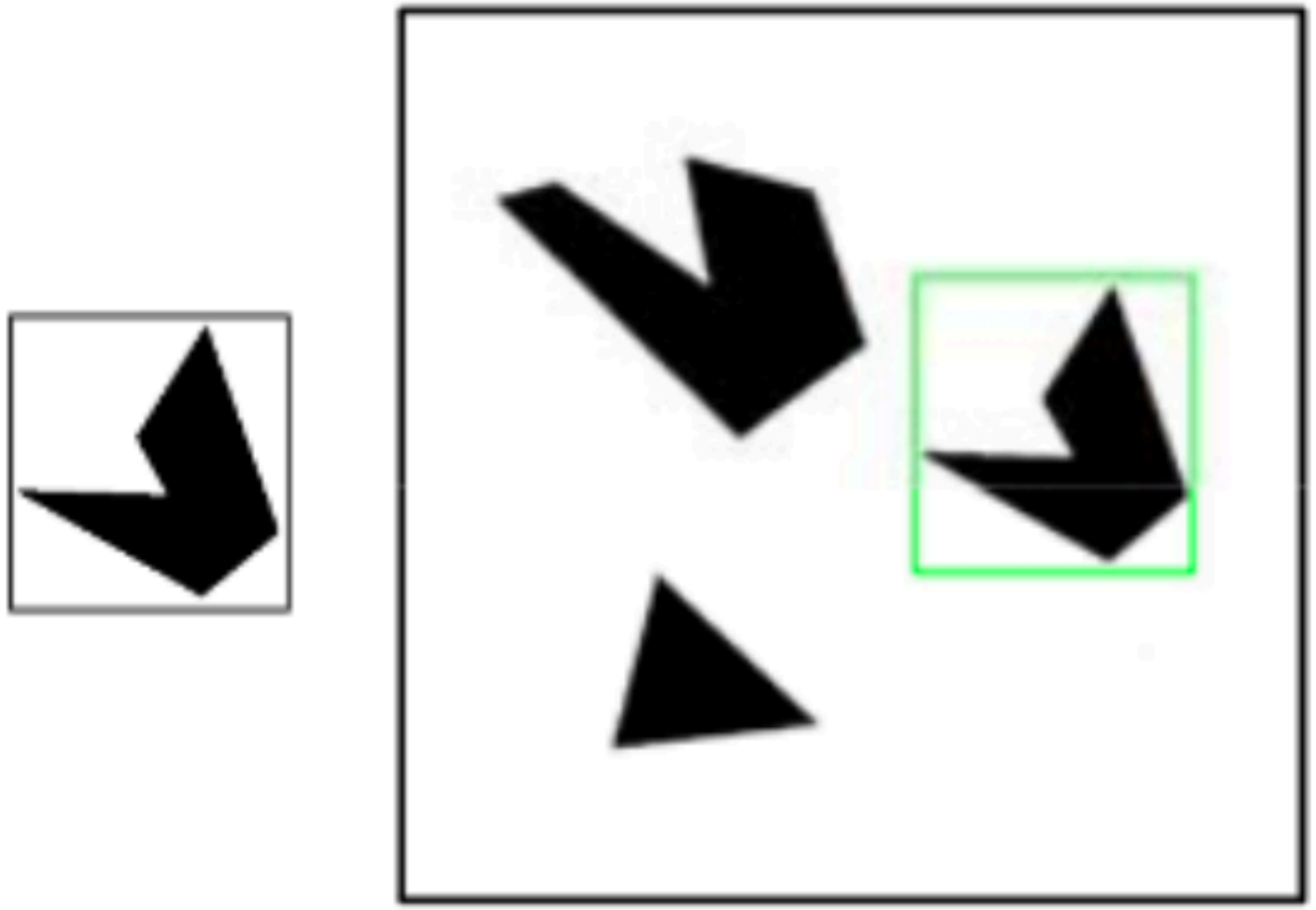


Correlation map

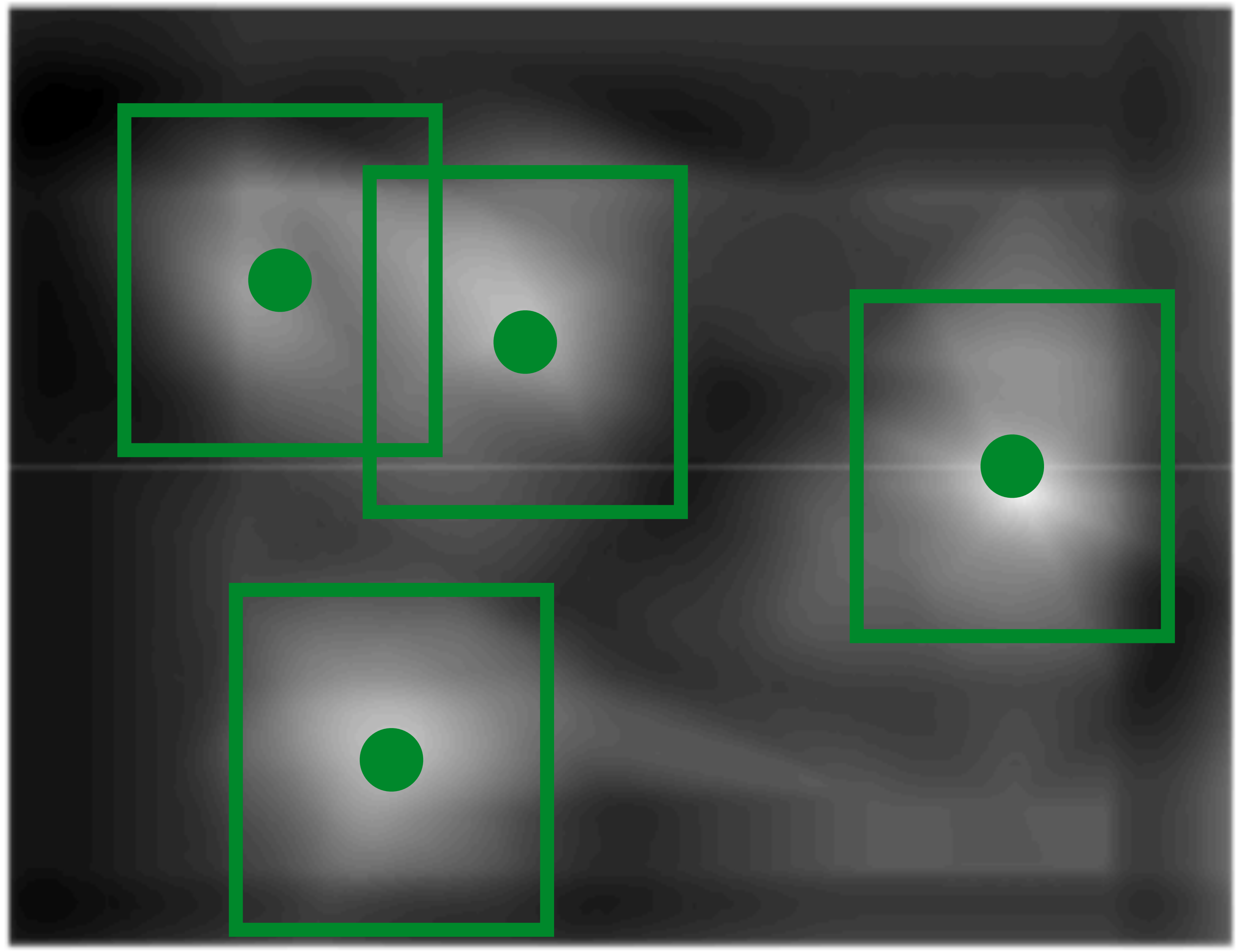
Slide Credit: Kristen Grauman

Non-maxima Suppression

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



Detected template

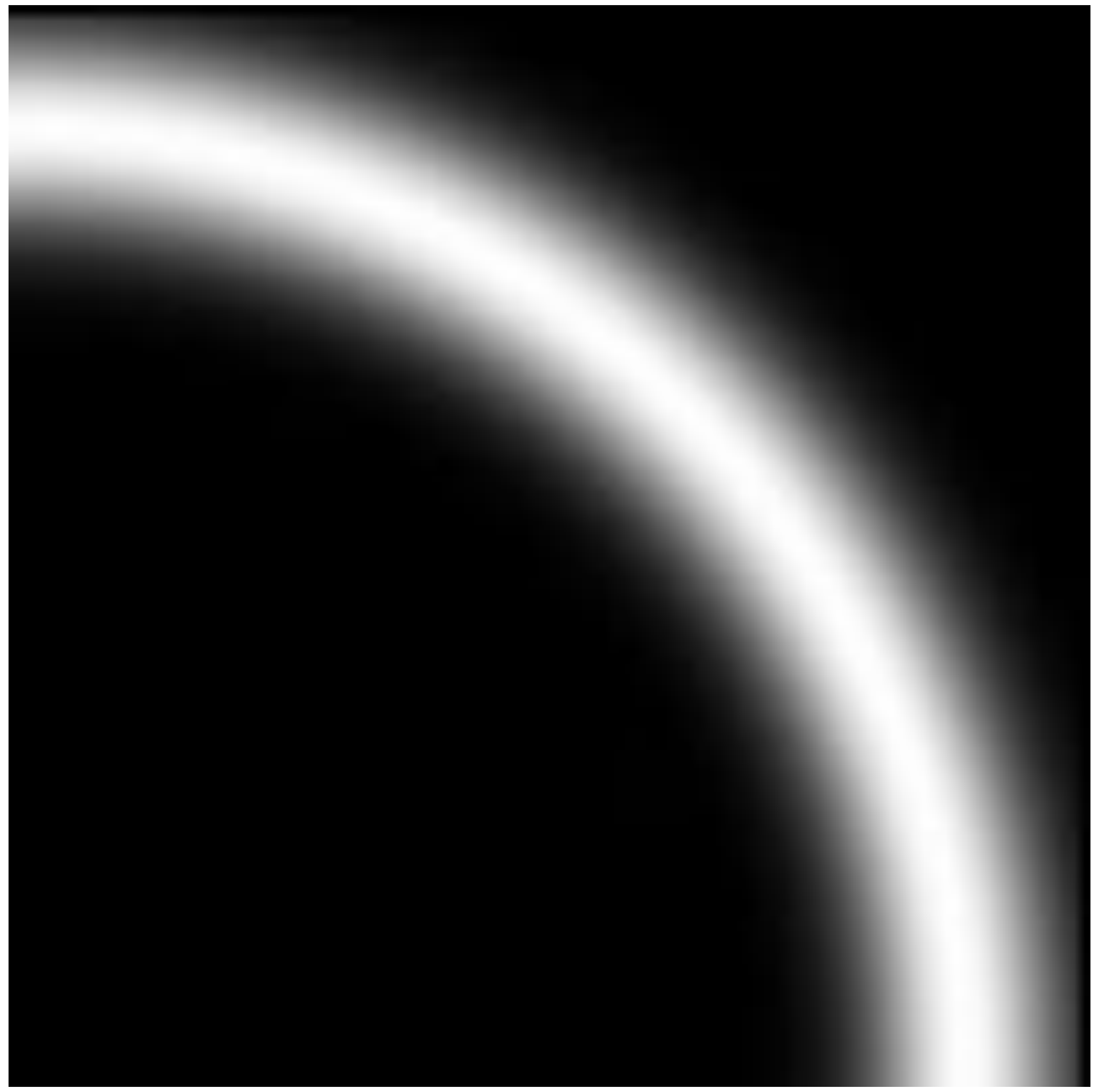


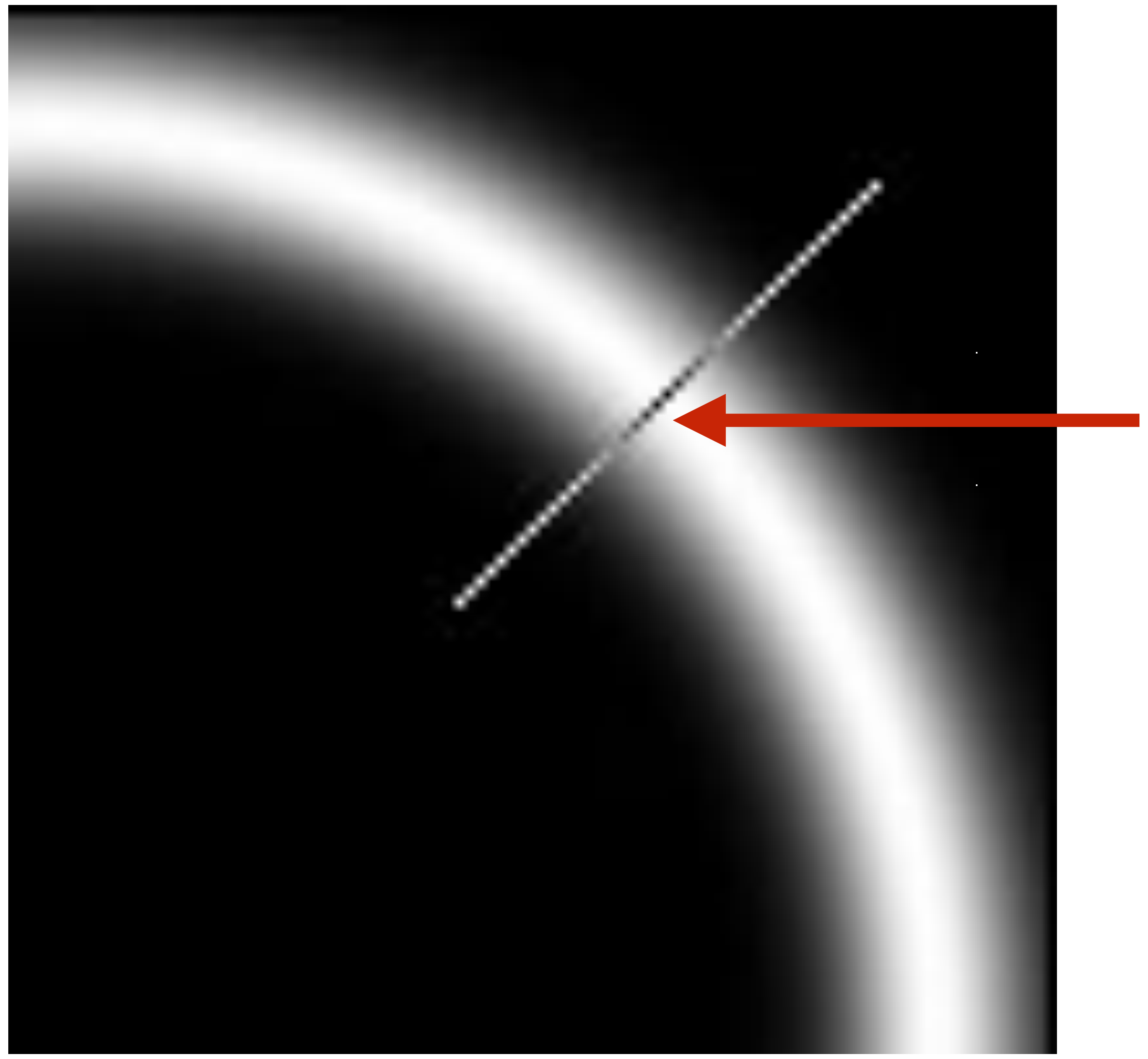
Correlation map

Slide Credit: Kristen Grauman

Non-maxima Suppression

Gradient magnitude



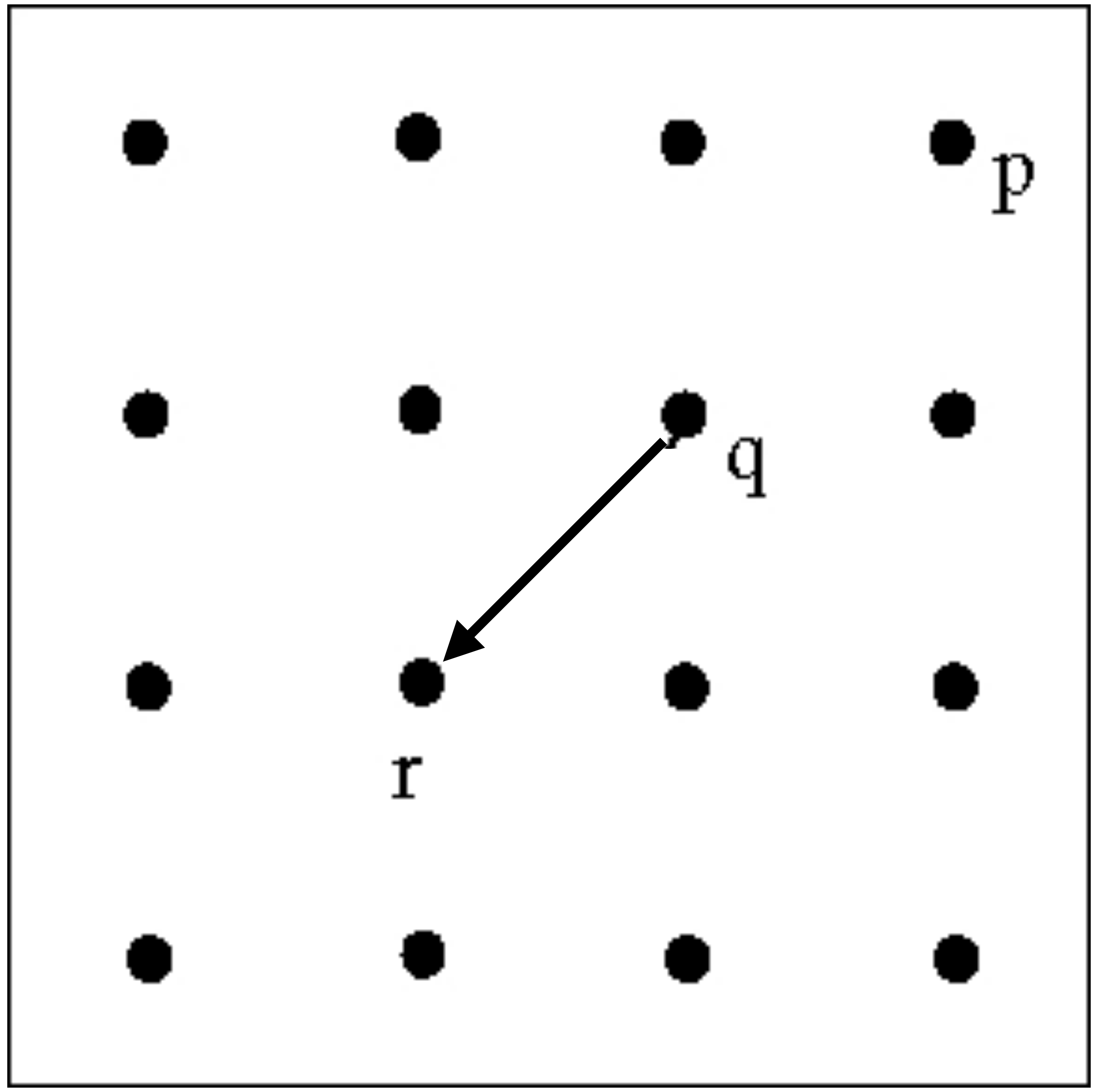


Forsyth & Ponce (1st ed.) Figure 8.11

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

Non-maxima Suppression

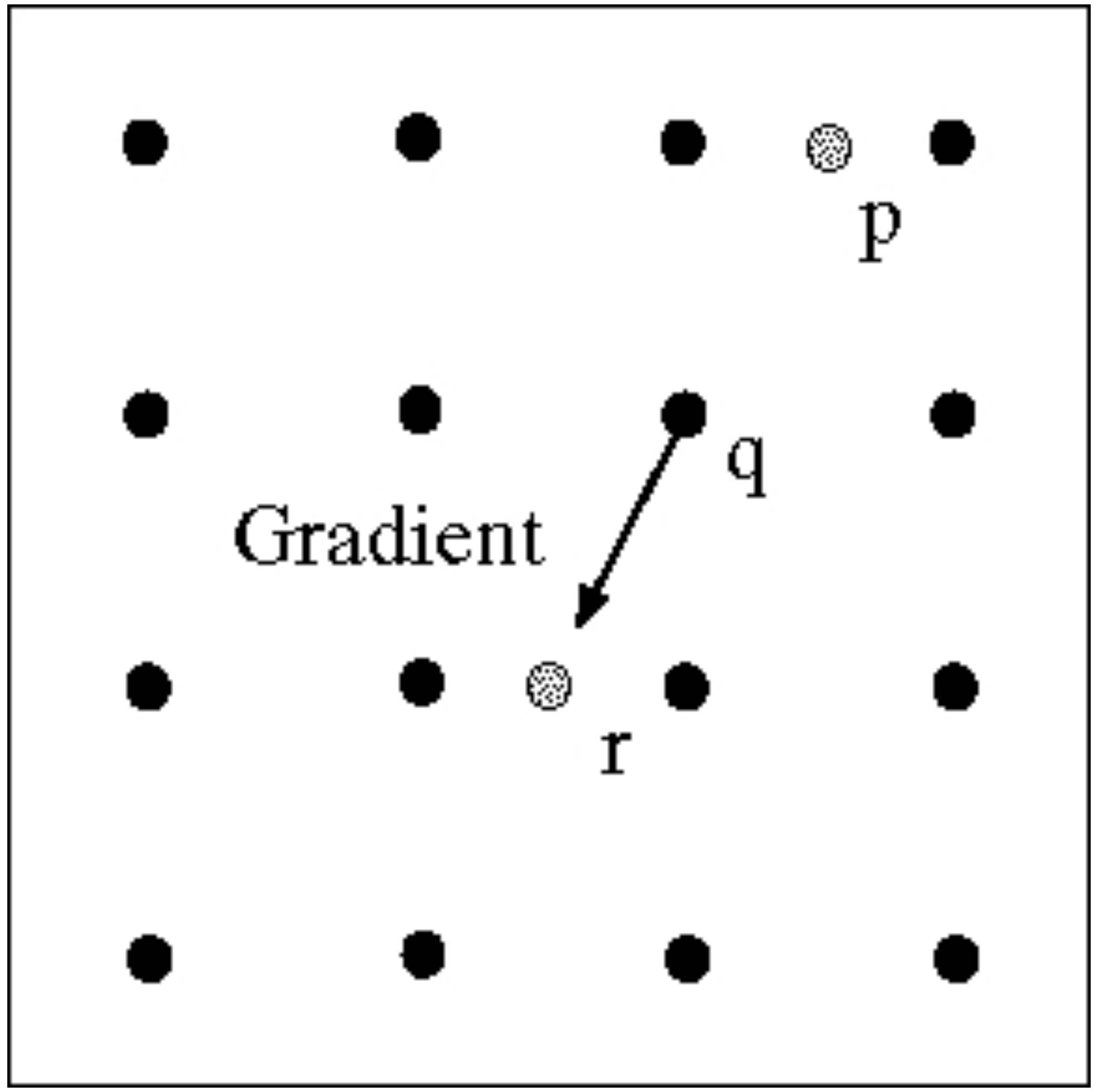
Value at *q* must be larger than interpolated values at *p* and *r*



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

Non-maxima Suppression

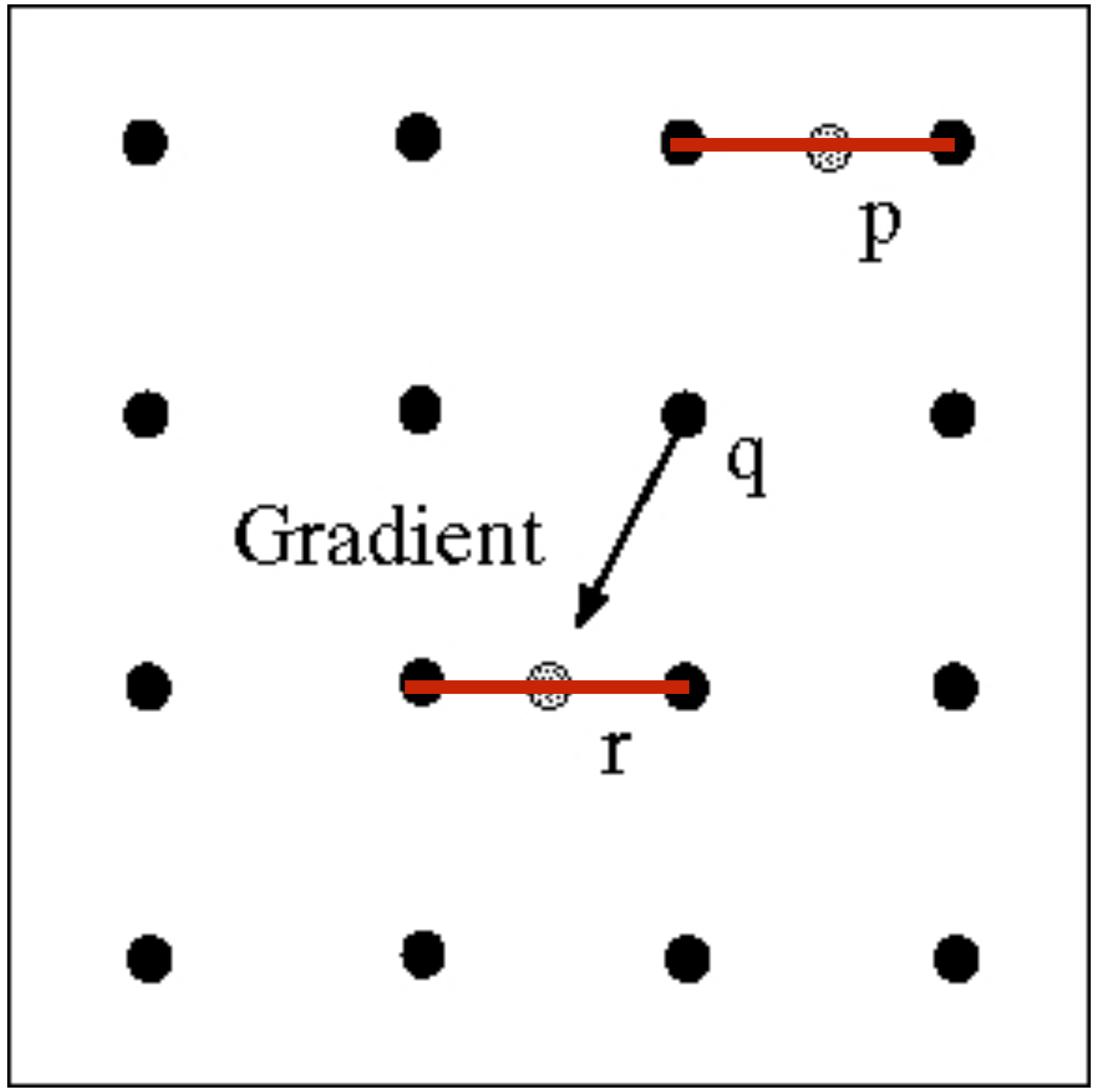
Value at *q* must be larger than interpolated values at *p* and *r*



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

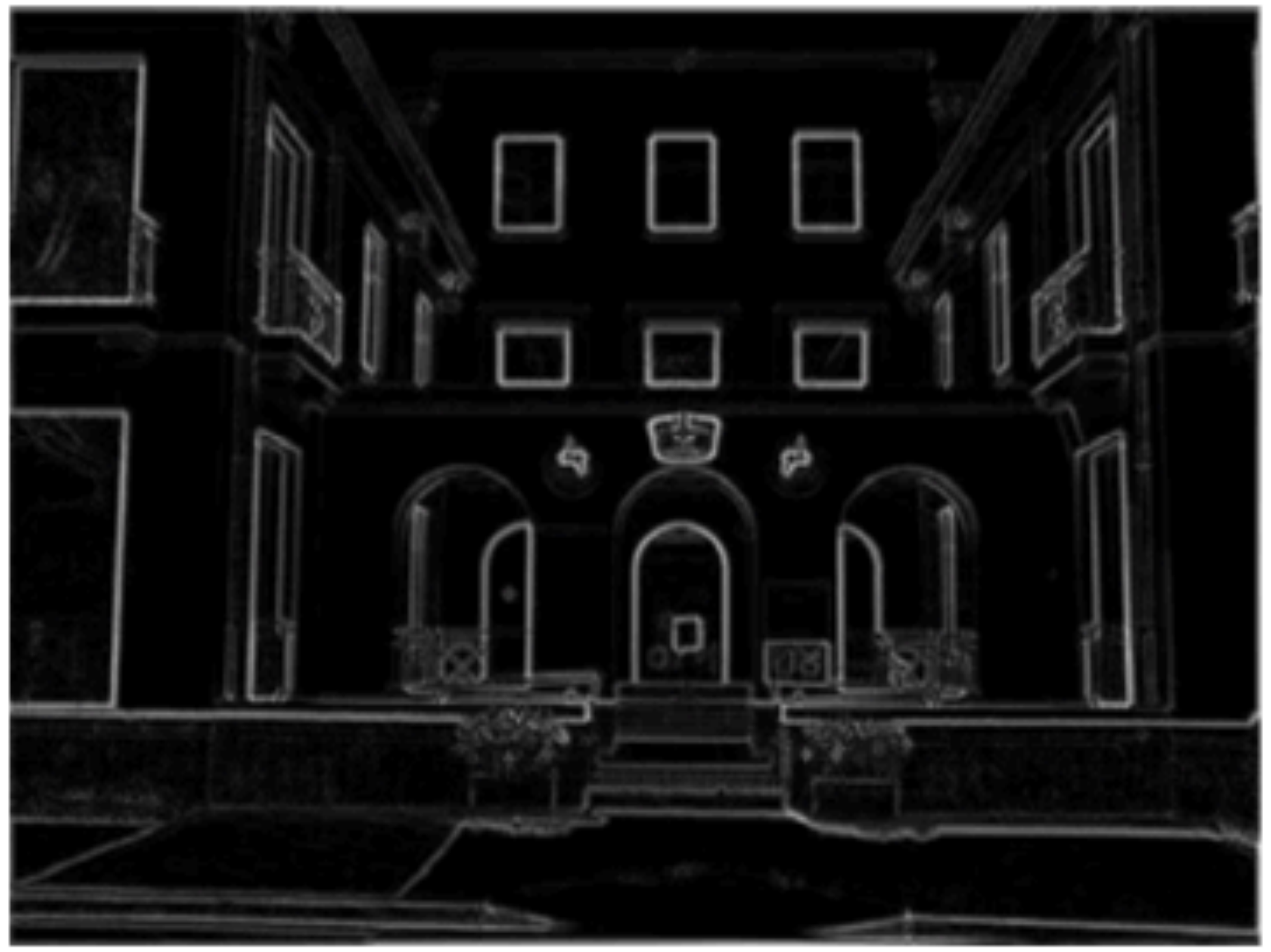
Non-maxima Suppression

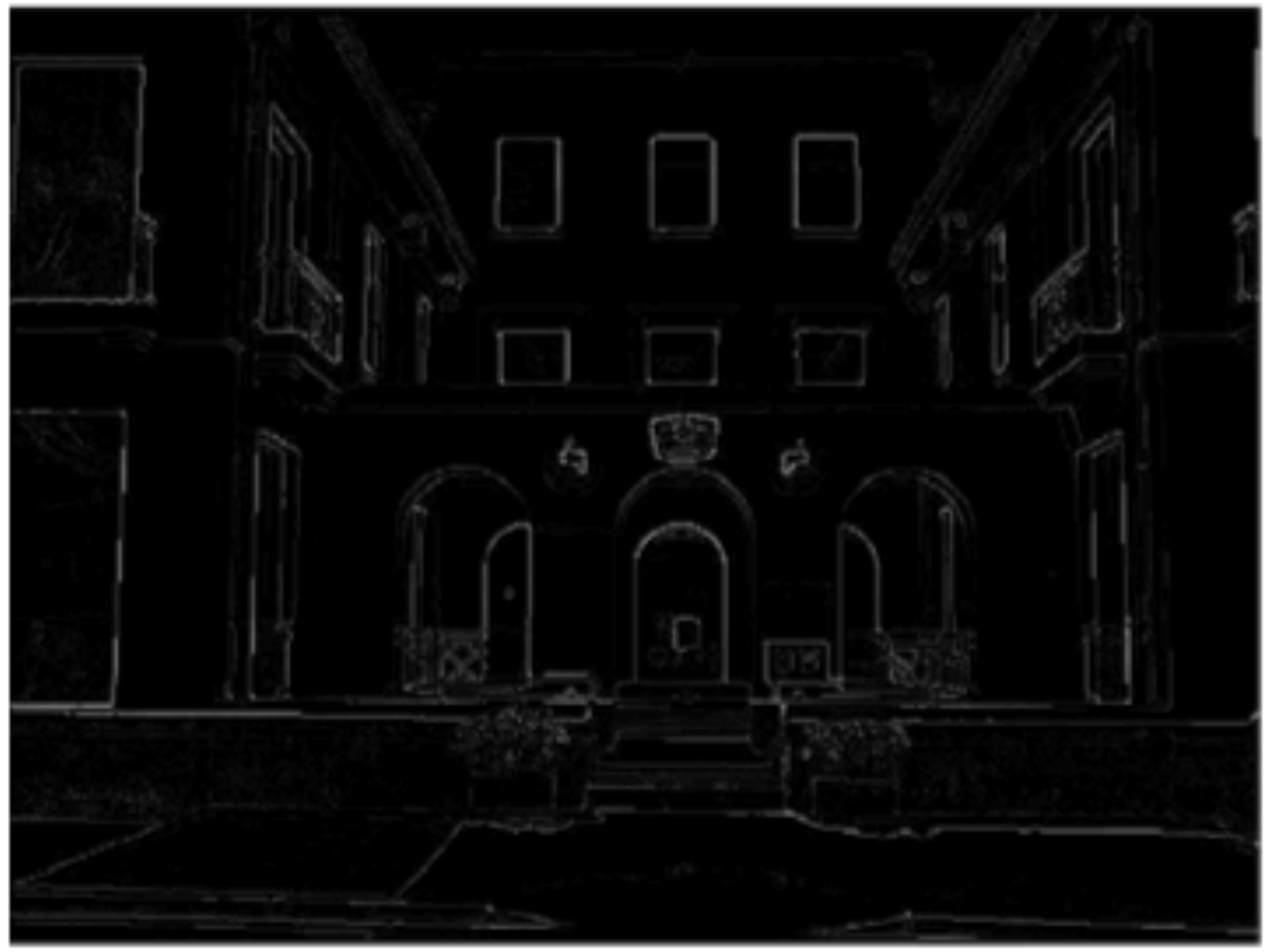
Value at *q* must be larger than interpolated values at *p* and *r*



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

Example: Non-maxima Suppression





courtesy of G. Loy

Original Image

Gradient Magnitude

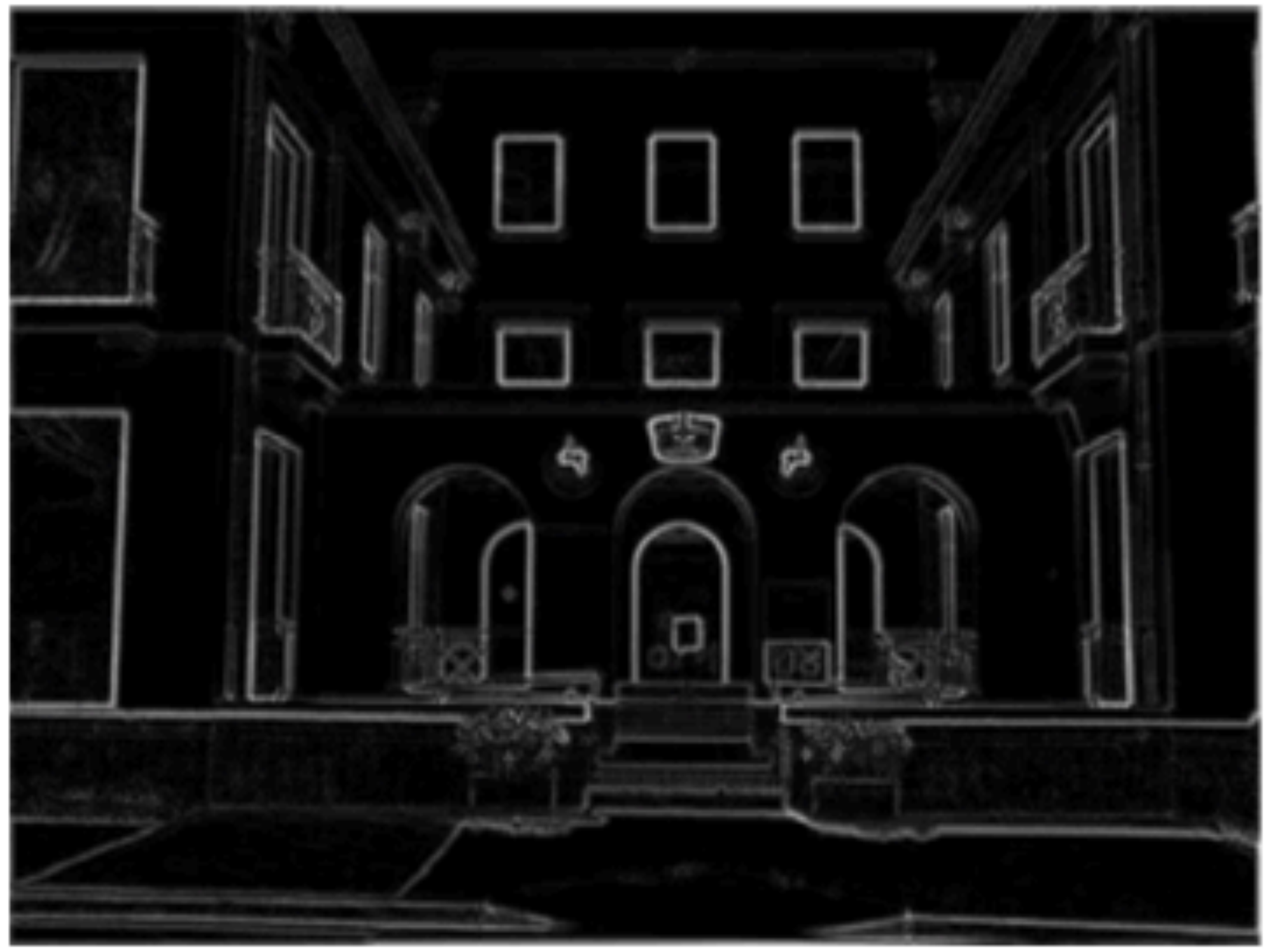
Non-maxima Suppression

Slide Credit: Christopher Rasmussen

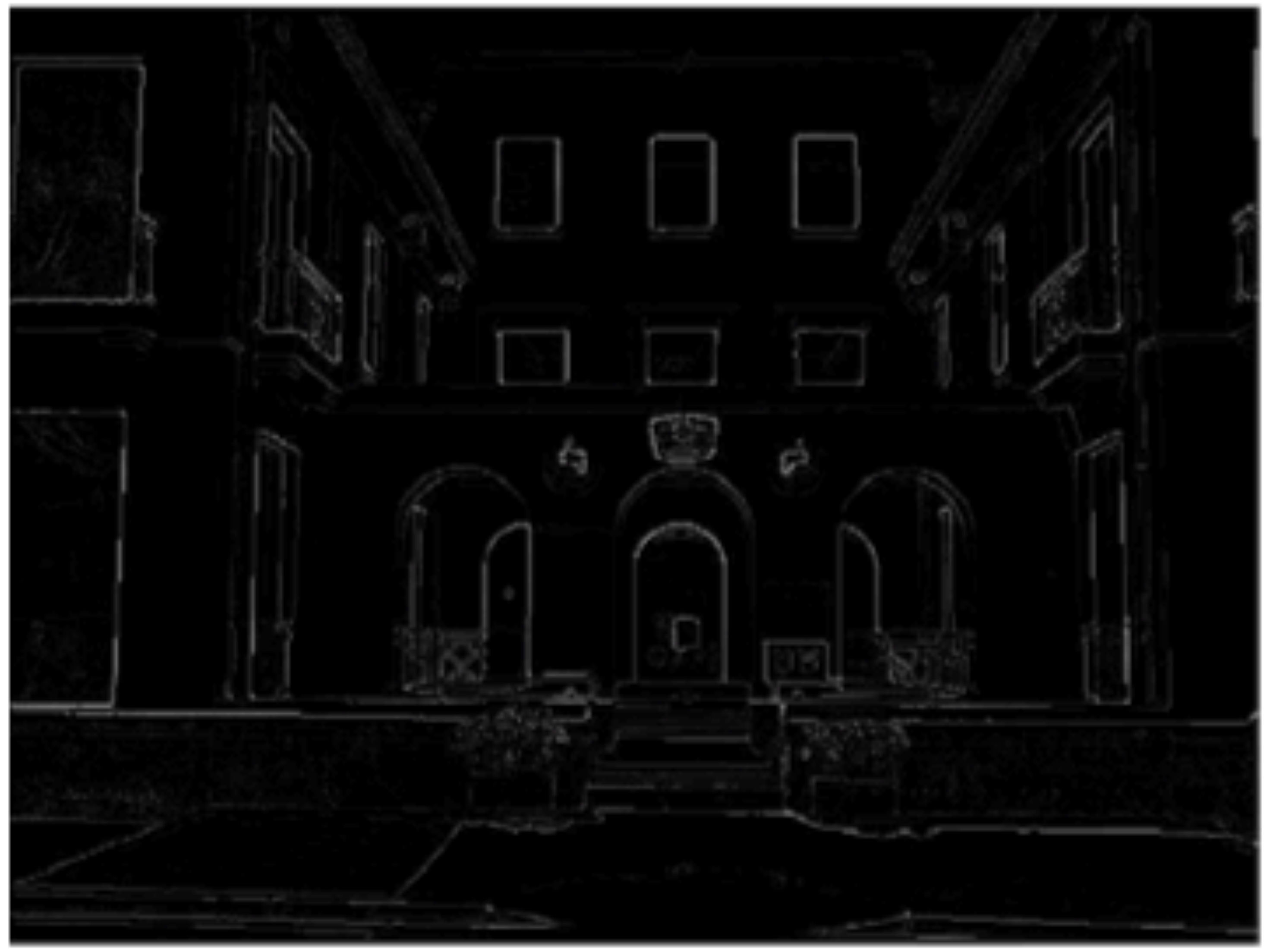
Example: Non-maxima Suppression

We only found **local extrema** of gradient magnitude, but some extrema may be small, others large in value, which do we keep? Threshold?

Original Image



Gradient Magnitude



courtesy of G. Loy

Non-maxima Suppression

Example

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

Example

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

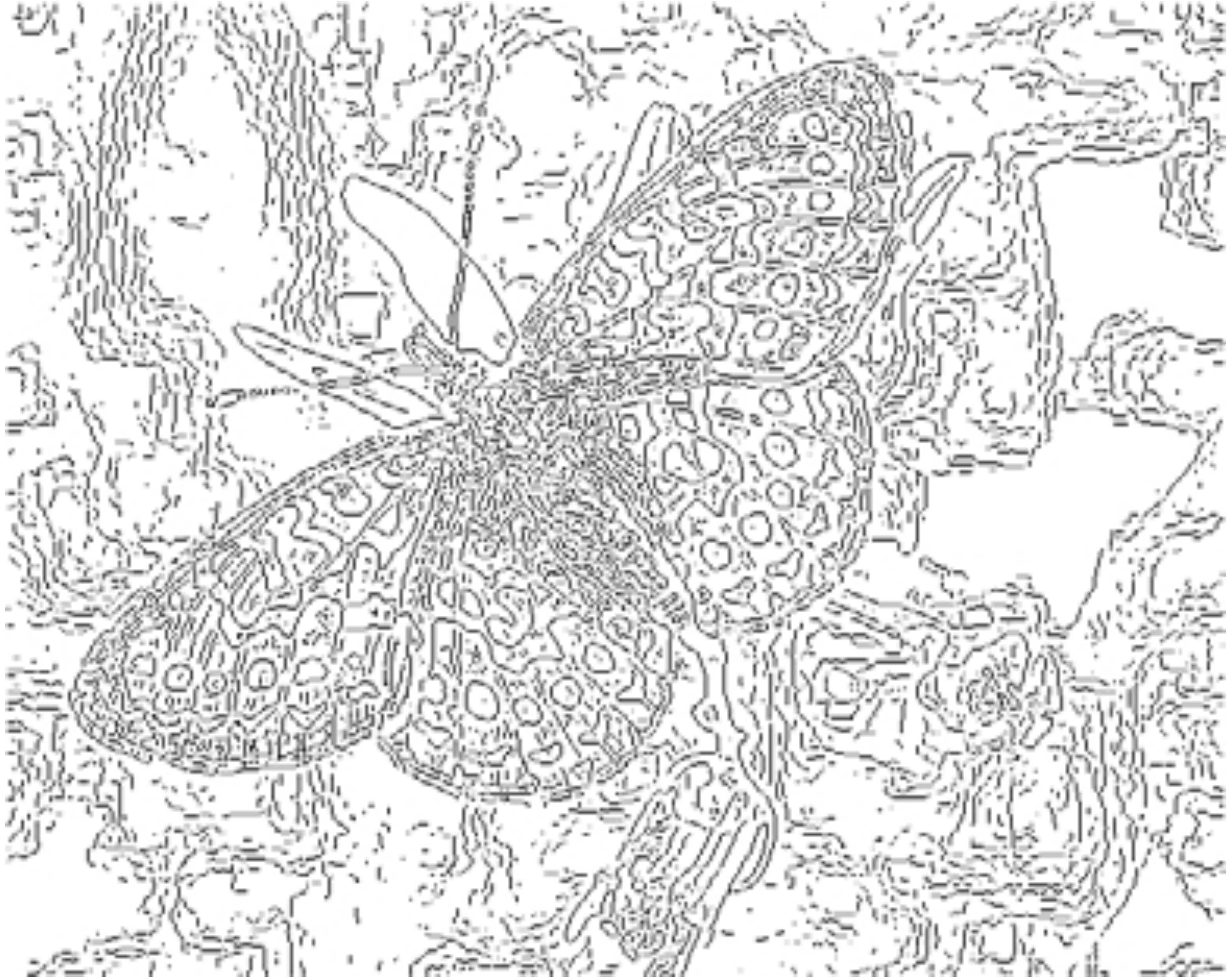


Figure 8.13 bottom left

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

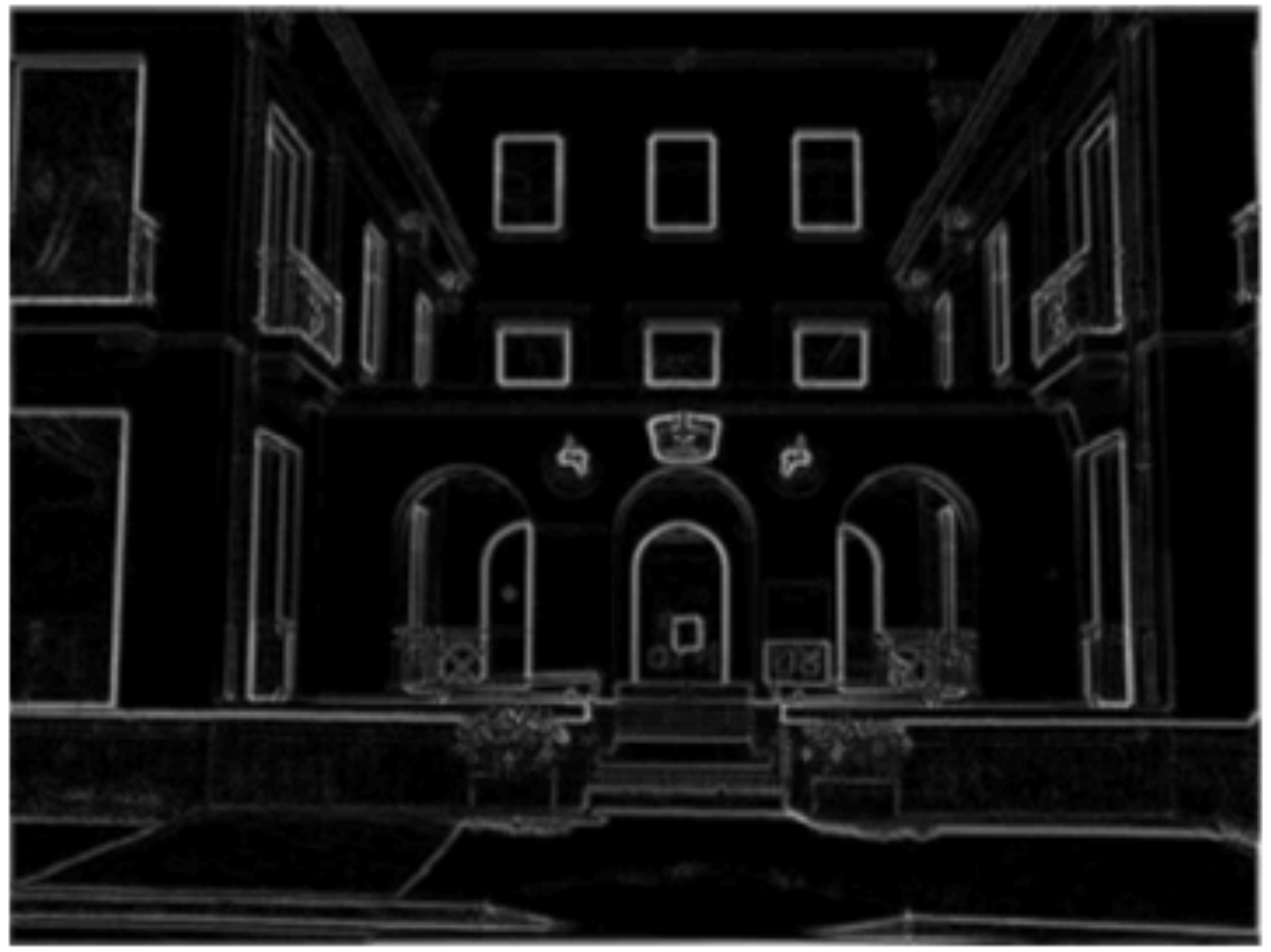
Figure 8.13 bottom right

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

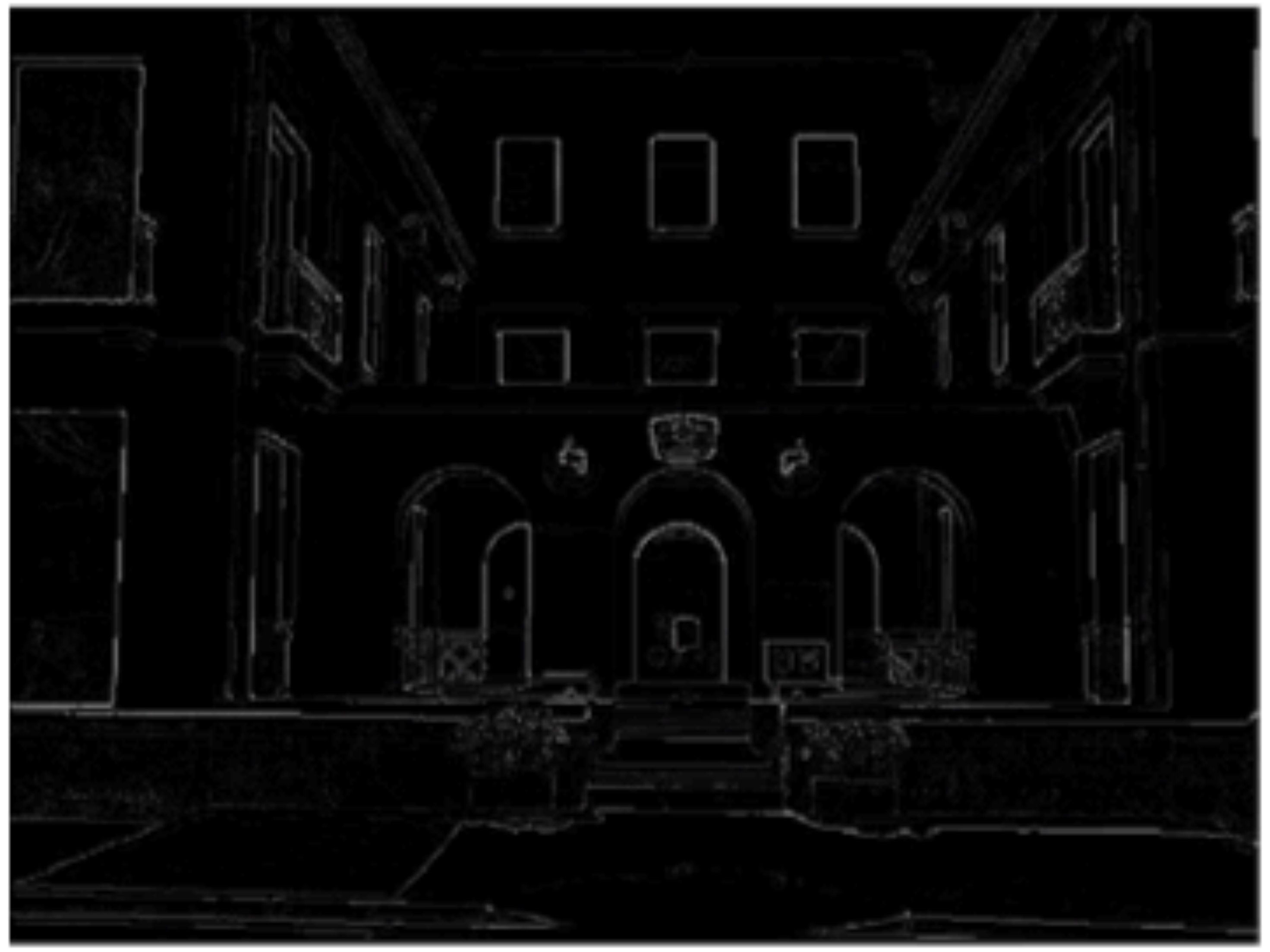
Linking Edge Points

We only found **local extrema** of gradient magnitude, but some extrema may be small, others large in value, which do we keep? Threshold?

Original Image



Gradient Magnitude



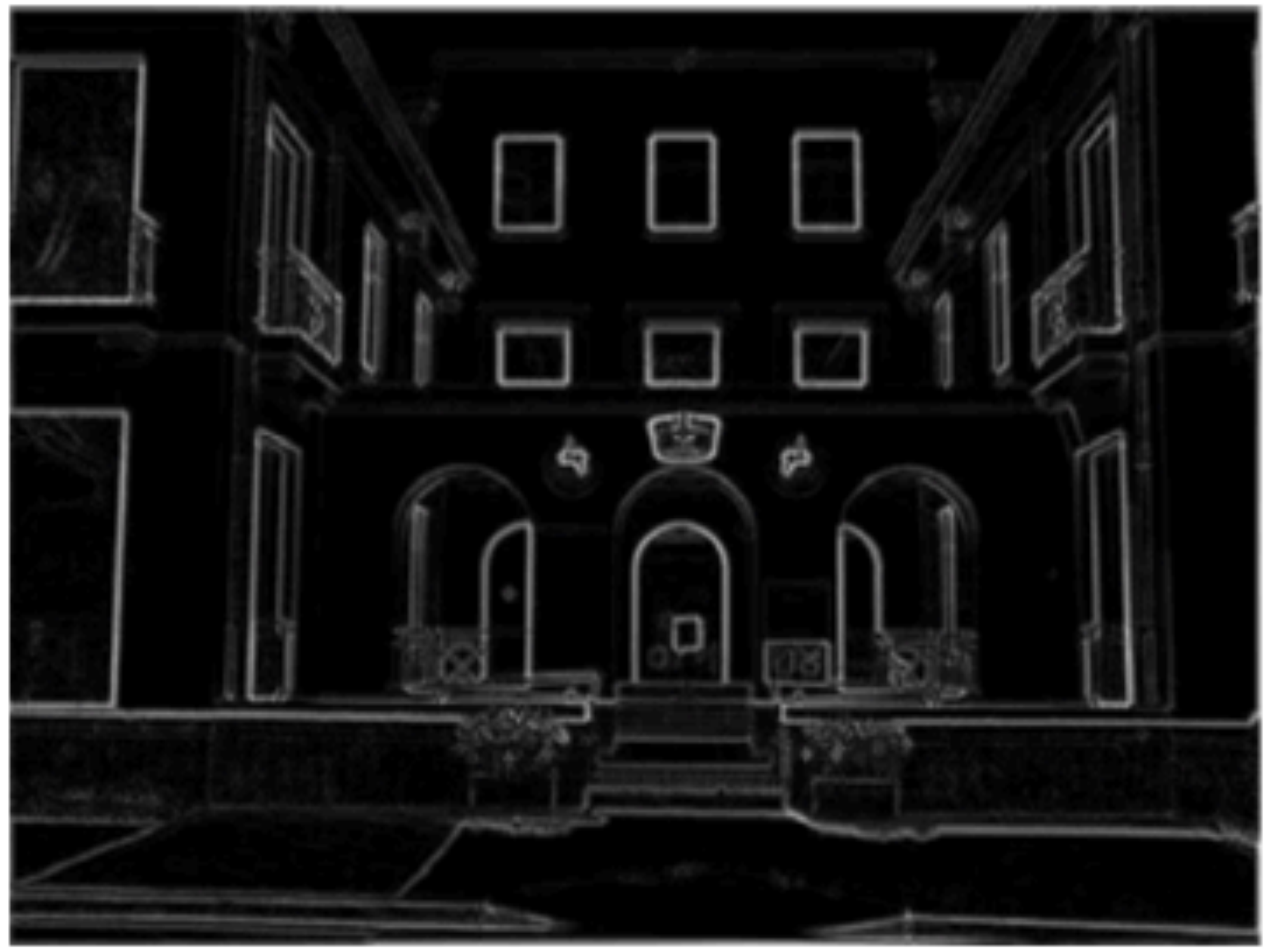
courtesy of G. Loy

Non-maxima Suppression

Linking Edge Points

We only found **local extrema** of gradient magnitude, but some extrema may be small, others large in value, which do we keep? Threshold?

Original Image

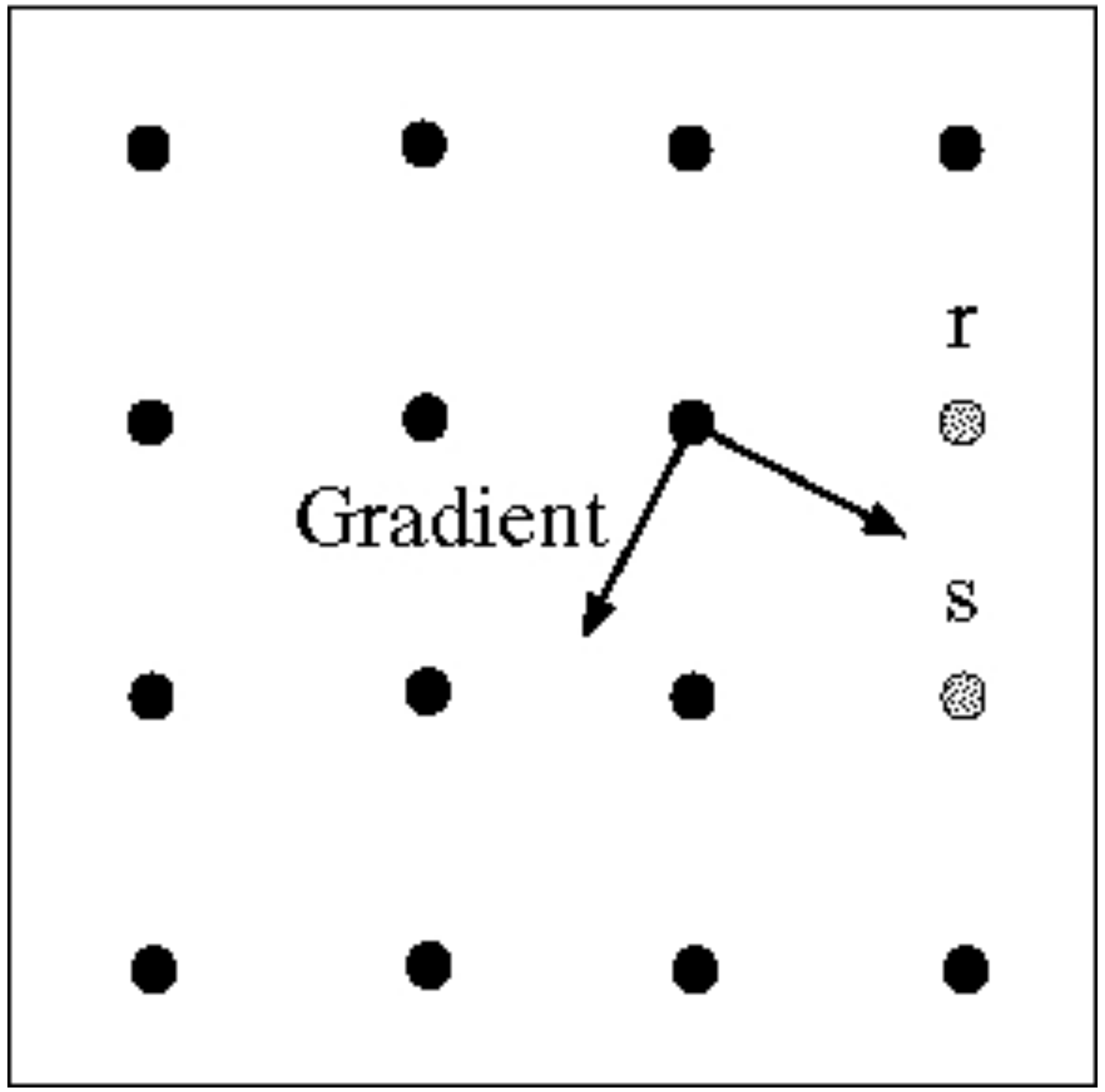


Gradient Magnitude

courtesy

Non-maxima Suppression

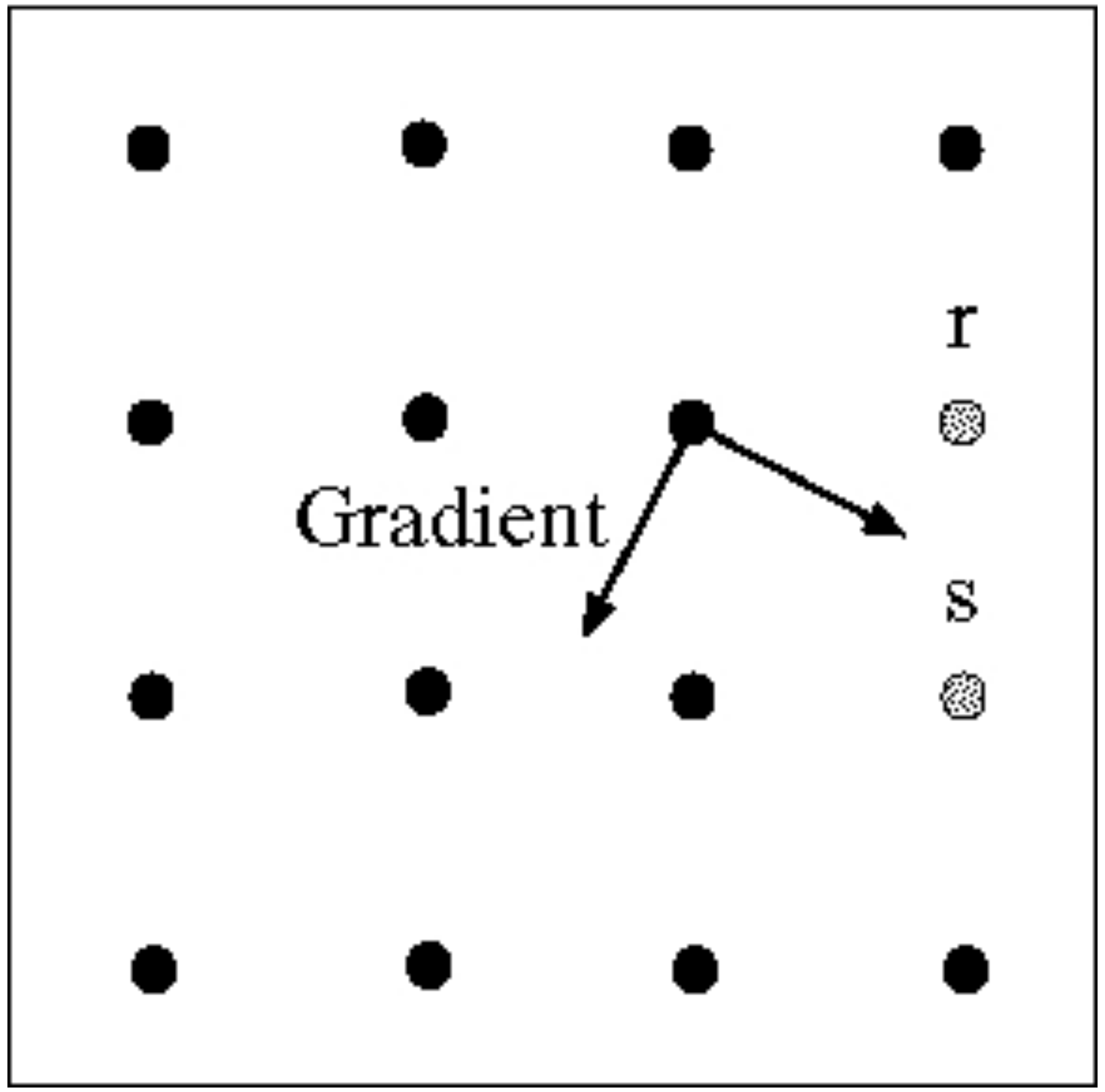
Linking Edge Points



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*)

Linking Edge Points



$\text{gradient magnitude} > \mathbf{k}_{high}$

— definitely **edge** pixel

$\mathbf{k}_{low} < \text{gradient magnitude} < \mathbf{k}_{high}$

— maybe an edge pixel

$\text{gradient magnitude} < \mathbf{k}_{low}$

— definitely **not** edge pixel

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 $\mathbf{k}_{high}$ to find strong edges to start edge chain

— Use $\mathbf{k}_{low}$ to find weak edges which continue edge chain

Typical ratio of thresholds is (roughly):

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

Canny Edge Detector

Original Image

Strong + connected **Weak** Edges

Strong Edges

Weak Edges

courtesy of G. Loy

How do humans perceive **boundaries**?

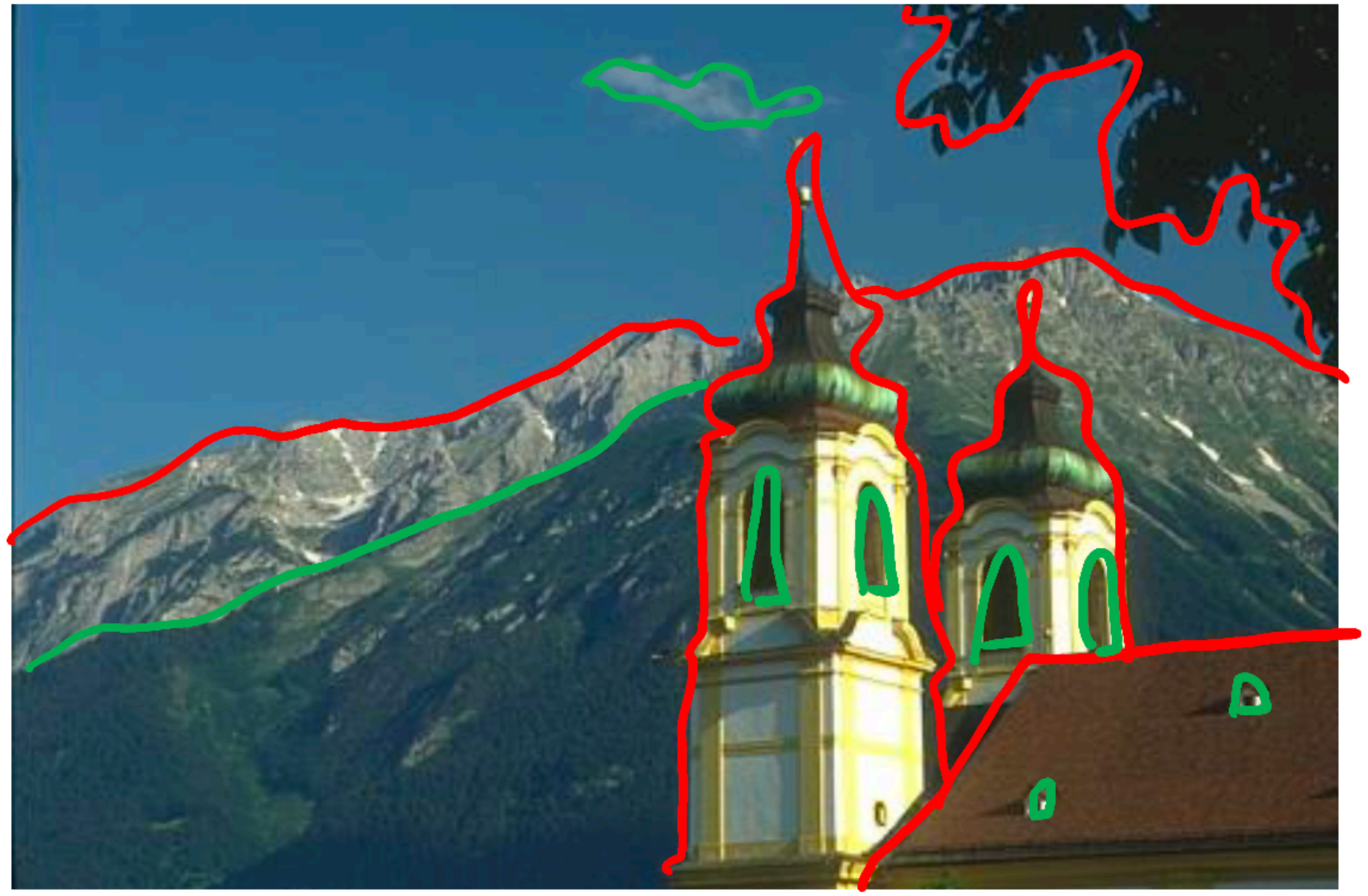
Edges are a property of the 2D image.

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

Traditional Edge Detection

Generally lacks **semantics** (i.e., too low-level for many task)

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**?

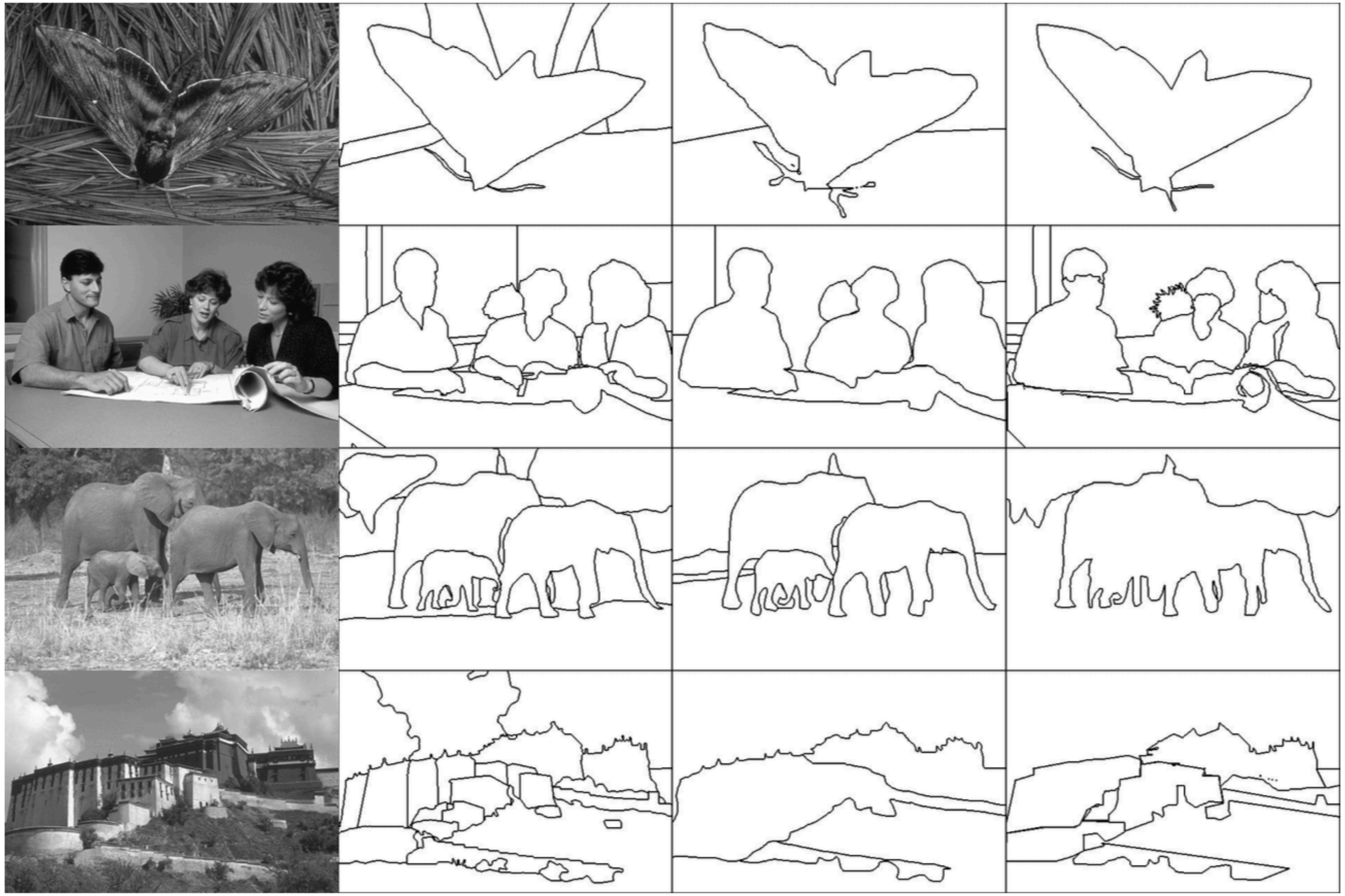


Figure Credit: Martin et al. 2001

How do humans perceive **boundaries**?

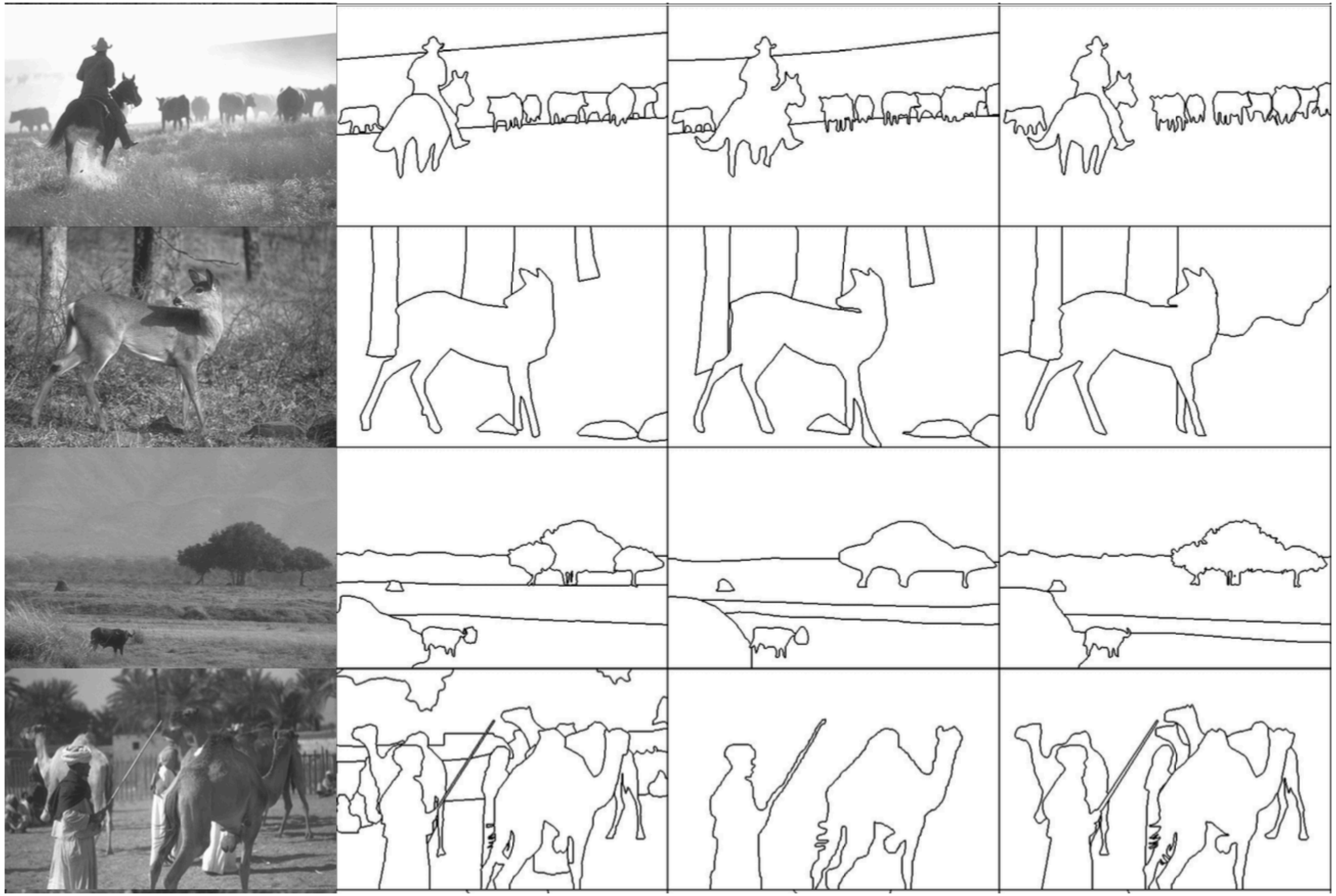
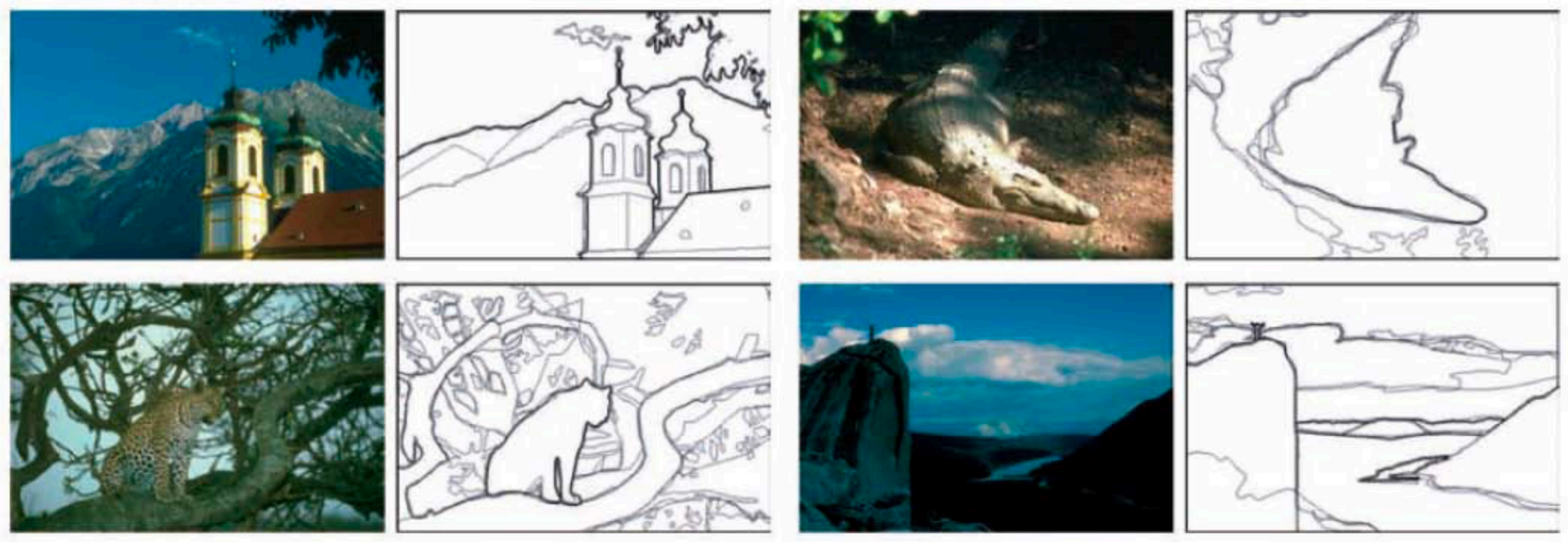


Figure Credit: Martin et al. 2001

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

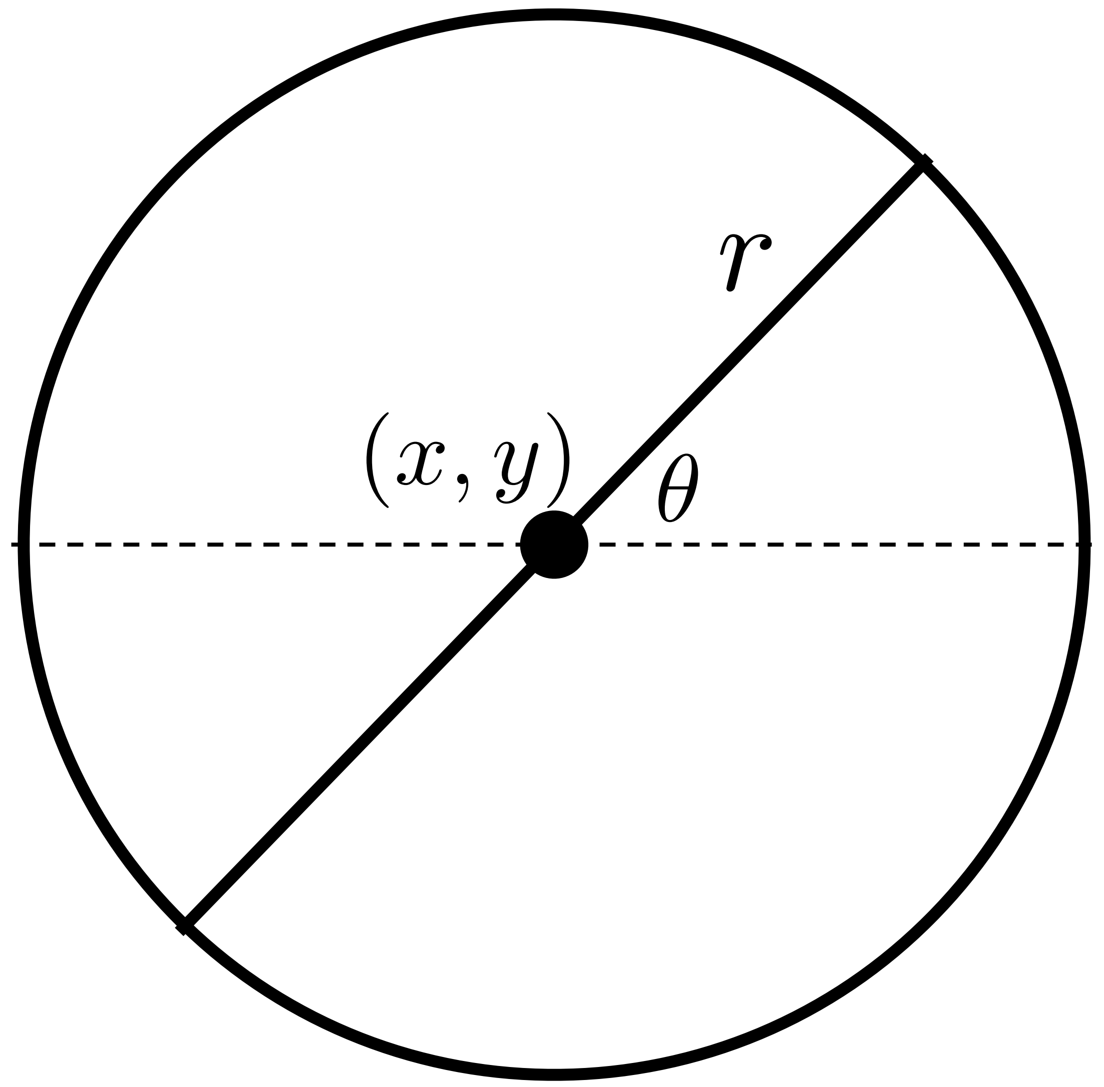
Boundary Detection: Example Approach

— Consider circular windows of radii r at each pixel (x, y) cut in half by an oriented line through the middle

— Compare visual features on both sides of the cut

— If features are very **different** on the two sides, the cut line probably corresponds to a boundary

— Notice this gives us an idea of the orientation of the boundary as well



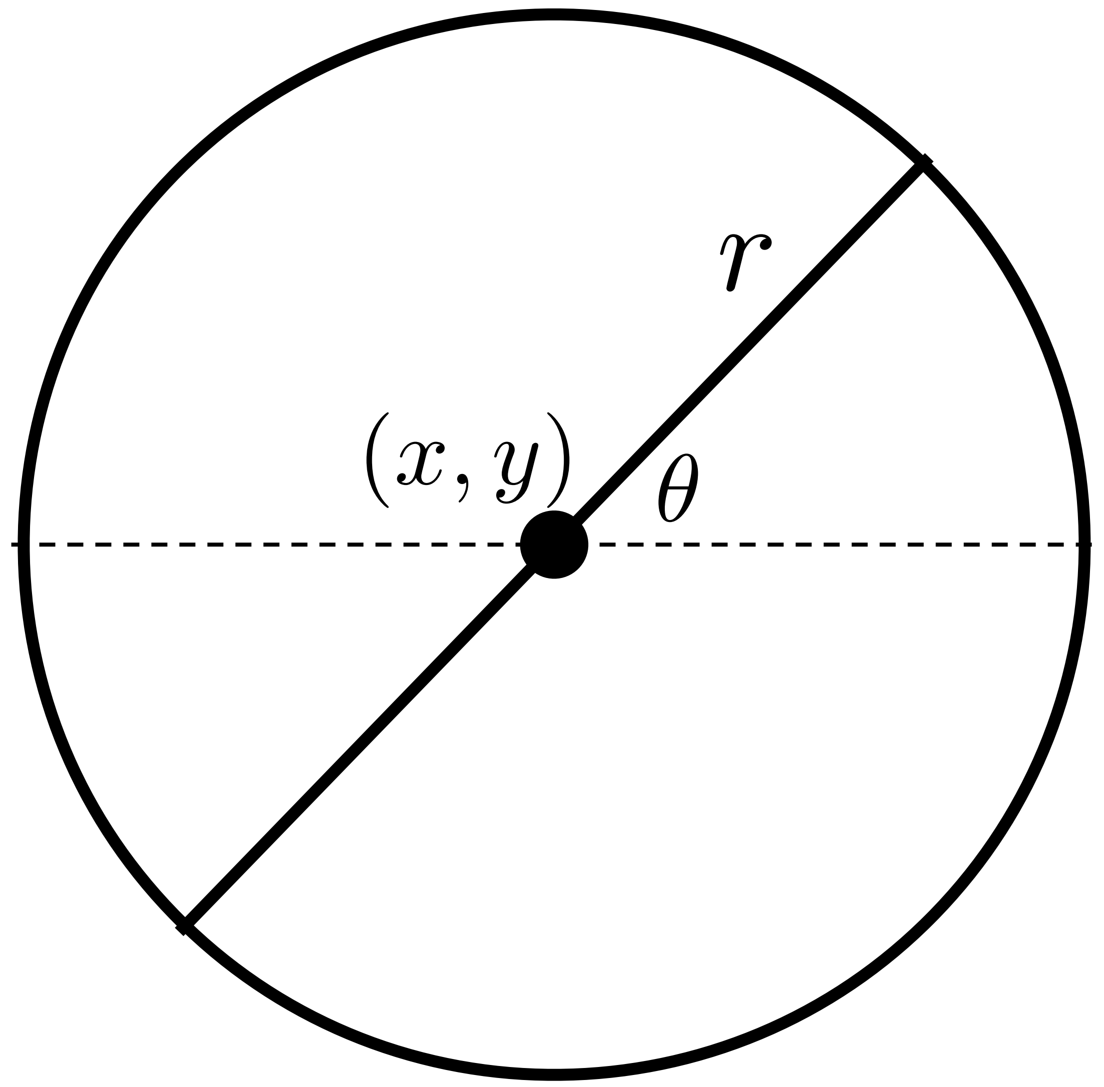
Boundary Detection: Example Approach

— Consider circular windows of radii r at each pixel (x, y) cut in half by an oriented line through the middle

— Compare visual features on both sides of the cut

— If features are very **different** on the two sides, the cut line probably corresponds to a boundary

— Notice this gives us an idea of the orientation of the boundary as well



Implementation: consider 8 discrete orientations (θ) and 3 scales (r)


r
(x, y)
θ


r
(x, y)
θ


r
(x, y)
θ


r
(x, y)
θ


r
(x, y)
θ

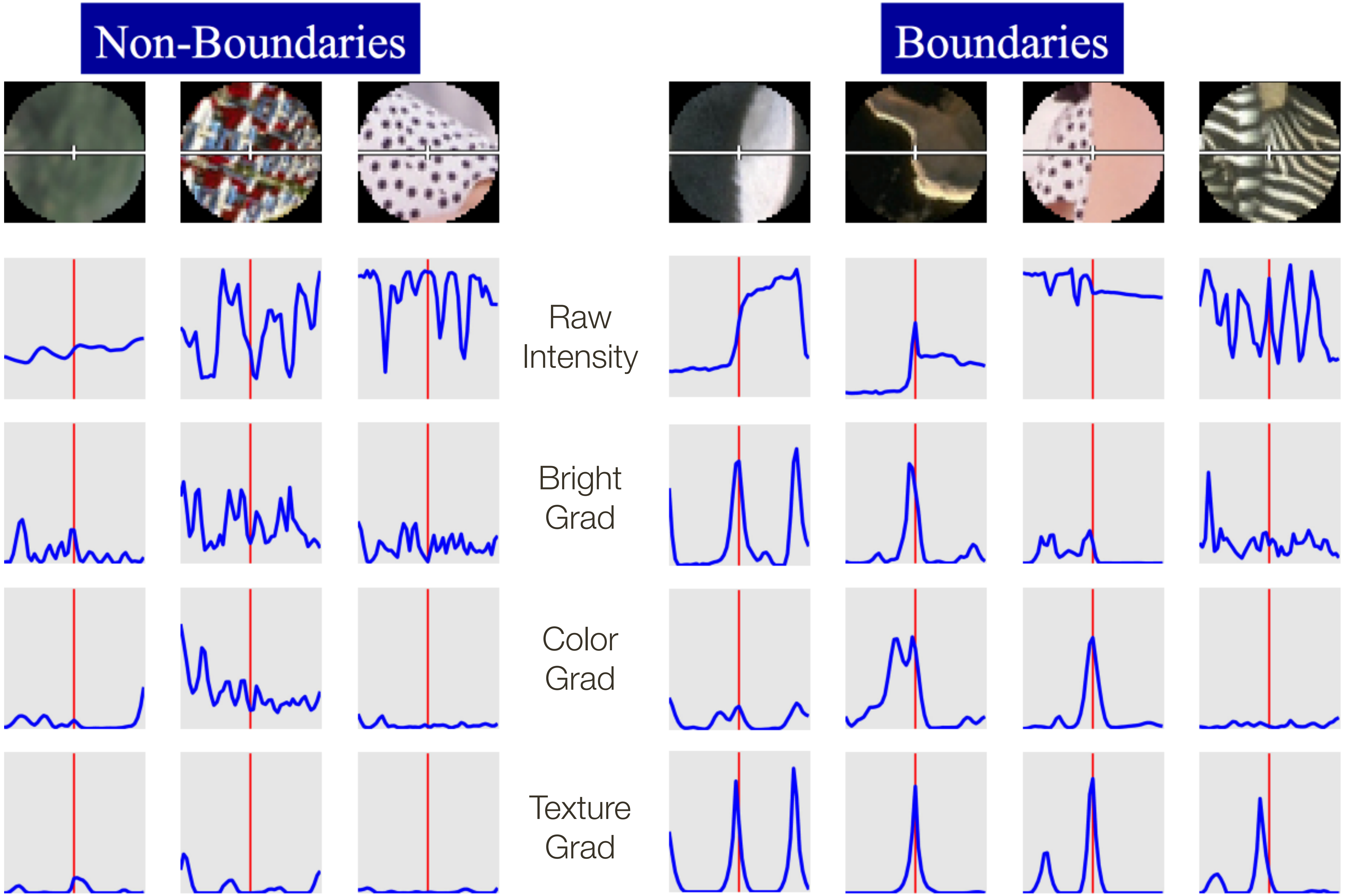
An **edge exists** if there is a **large difference between the distributions**

r
(x, y)
θ
r
r
(x, y)
θ
θ
r
(x, y)
θ
r
r
(x, y)
θ
θ

Boundary Detection:

Features:

- Raw Intensity
- Orientation Energy
- Brightness Gradient
- Color Gradient
- Texture gradient



Boundary Detection:

For each **feature** type

— Compute non-parametric distribution (histogram) for left side

— Compute non-parametric distribution (histogram) for right side

— Compare two histograms, on left and right side, using statistical test

Use all the histogram similarities as features in a learning based approach that outputs probabilities (Logistic Regression, SVM, etc.)

Boundary Detection: Example Approach

Figure Credit: Szeliski Fig. 4.33. **Original**: Martin et al. 2004

Summary

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

— depth discontinuity

— surface orientation discontinuity

— reflectance discontinuity

— illumination boundaries

Two generic approaches to **edge detection**:

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

— zero crossings of a second derivative operator → **Marr/Hildreth**

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