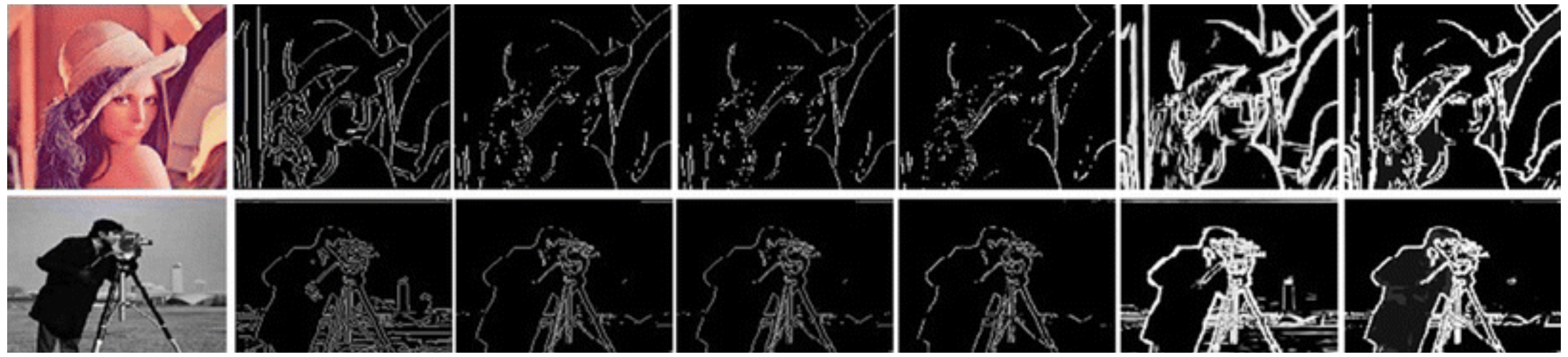
THE UNIVERSITY OF BRITISH COLUMBIA

CPSC 425: Computer Vision



Lecture 12: Edge Detection (cont.)

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

Menu for Today (**October 5, 2020**)

Topics:

- **Canny** Edges
- Image **Boundaries**

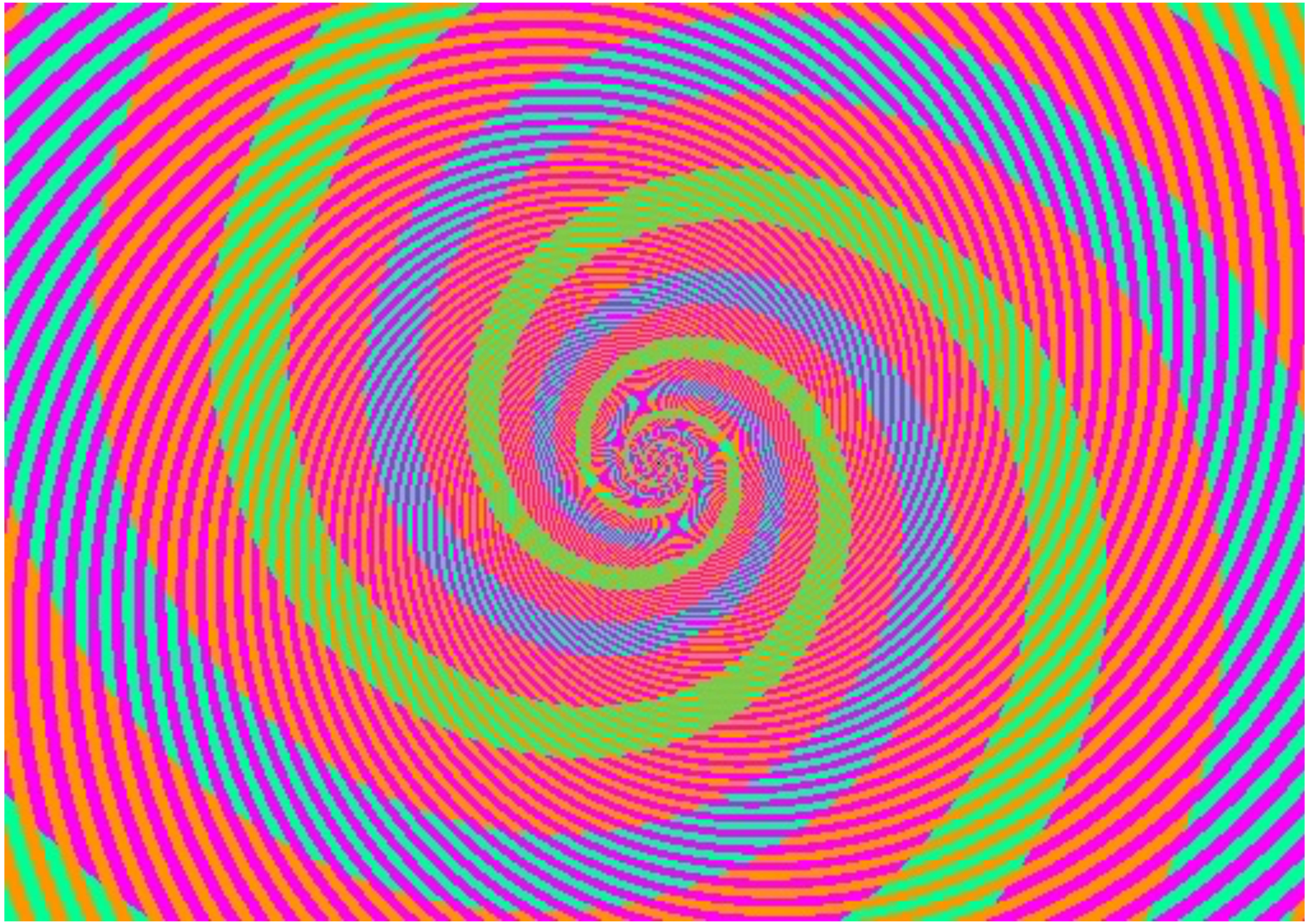
Redings:

- **Today's** Lecture: Forsyth & Ponce (2nd ed.) 5.1 - 5.2
- **Next** Lecture: Forsyth & Ponce (2nd ed.) 5.3.0 - 5.3.1

Reminders:

- **Assignment 2**: Scaled Representations, Face Detection and Image Blending
- **Quiz 1** correct answers are posted
- **Midterm** prep questions will be available this week (initially without Answers)

Today’s “**fun**” Example:



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

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

Lecture 11: Re-cap

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

— depth discontinuity

— surface orientation discontinuity

— reflectance discontinuity

— illumination boundaries

Lecture 11: Re-cap

Edge: a location with high gradient (derivative)

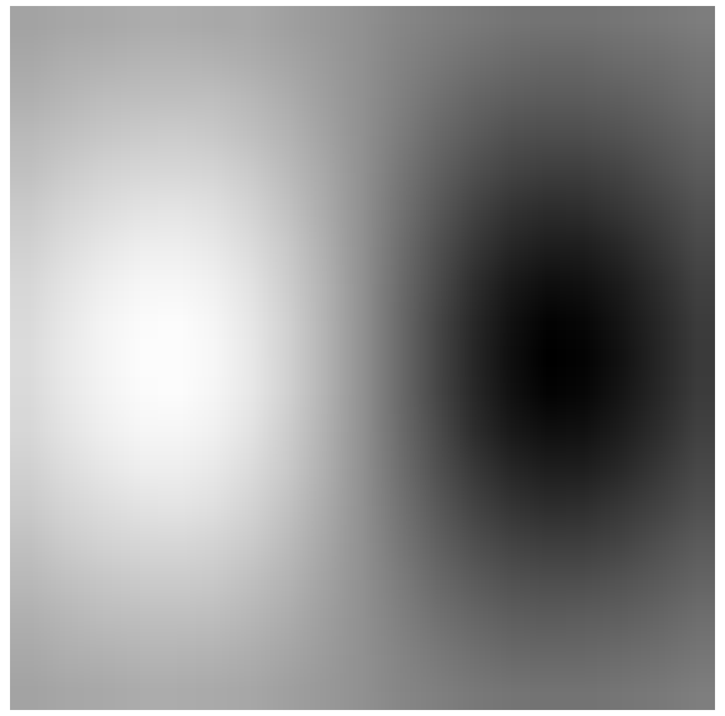
Need smoothing to reduce noise prior to taking derivative

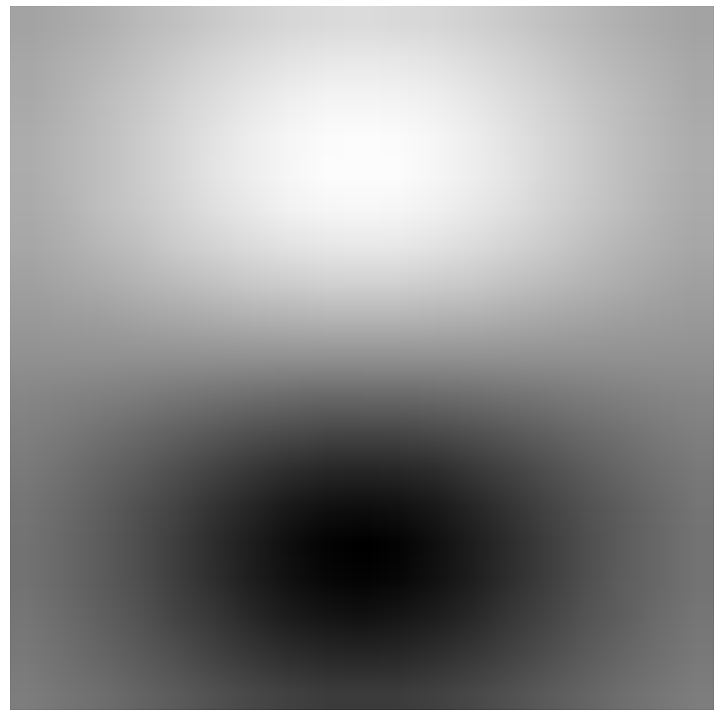
Need two derivatives, in x and y direction

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

Let $\otimes$ denote convolution

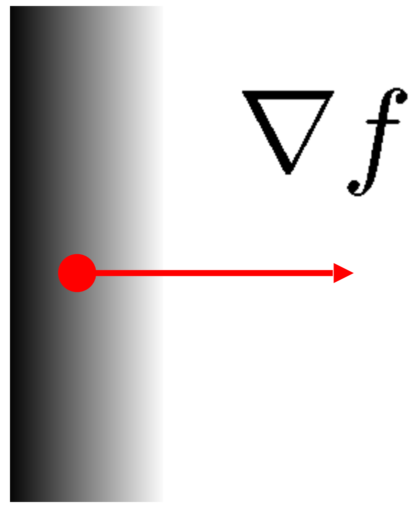
$$D \otimes (G \otimes I(X,Y)) = (D \otimes G) \otimes I(X,Y)$$



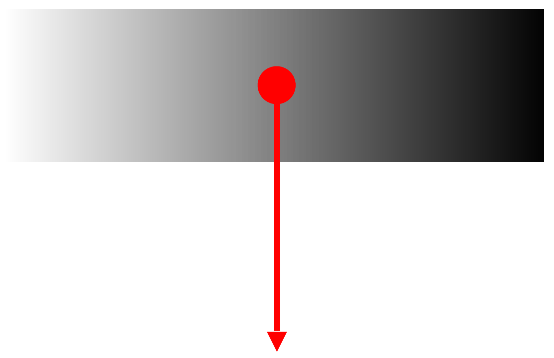


Lecture 11: Re-cap

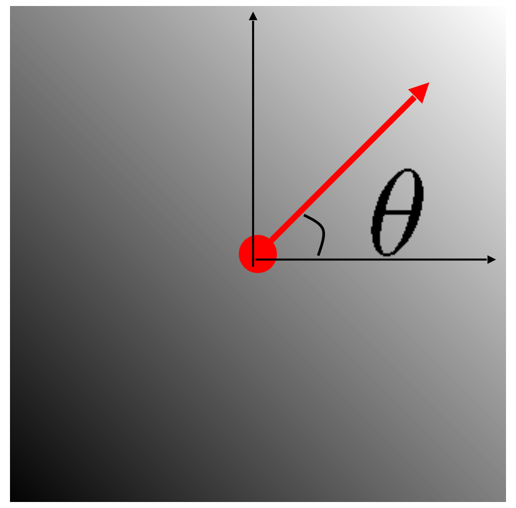
The gradient of an image: $\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$



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



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



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

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

Lecture 11: Re-Cap **Sobel** Edge Detector

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

2. **Threshold** to obtain edges

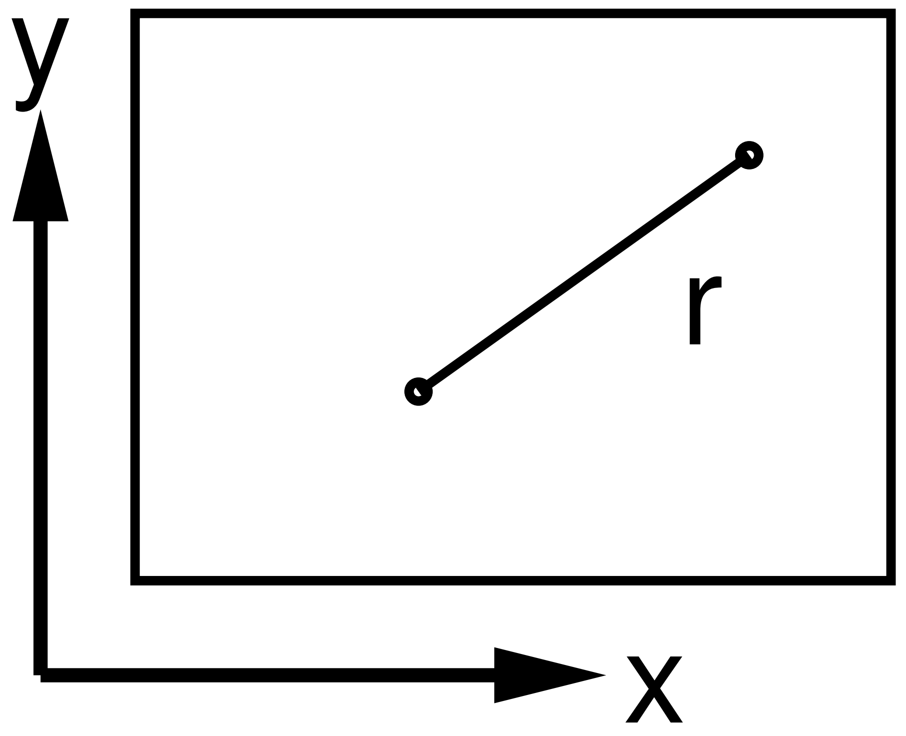
Original Image

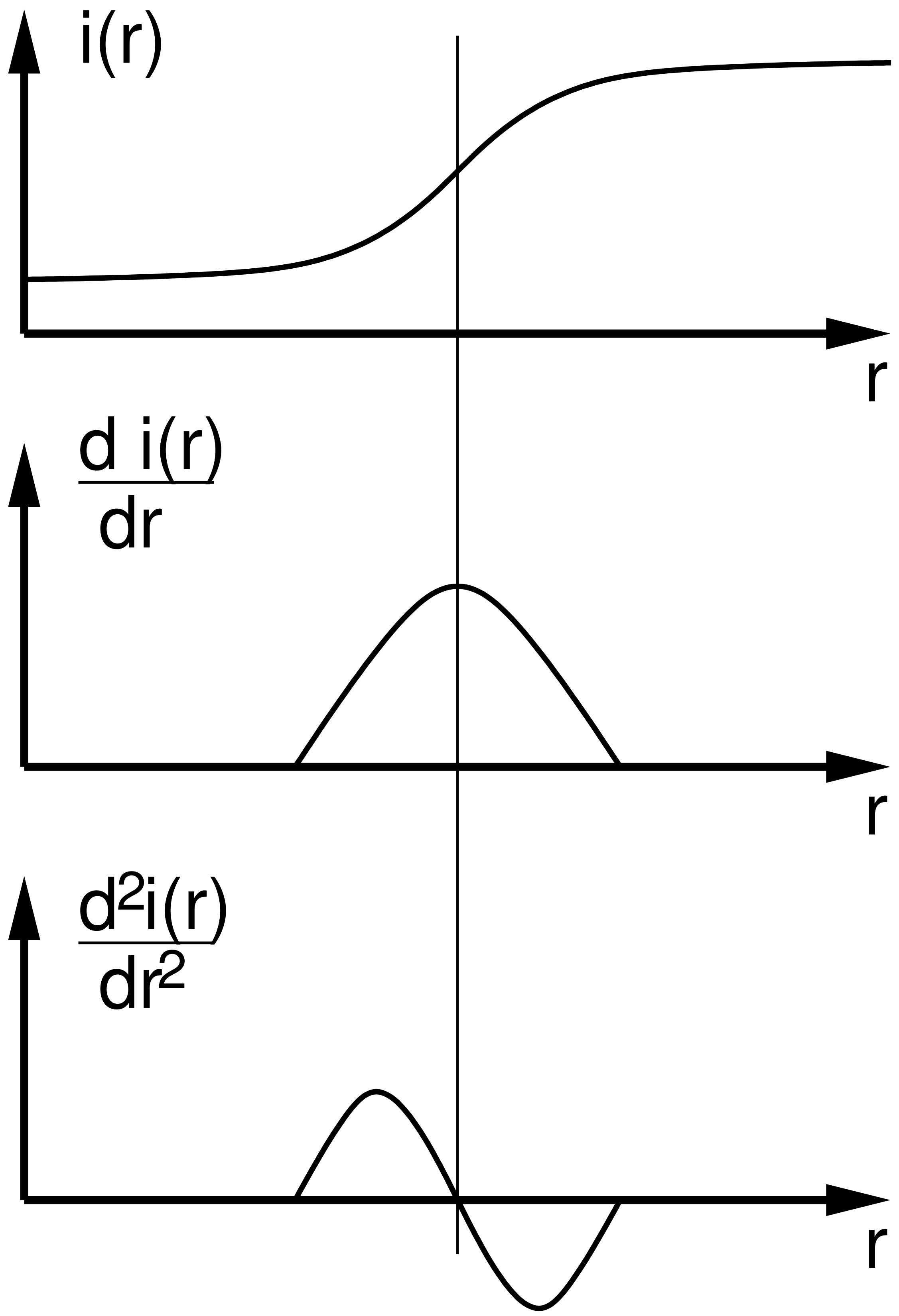
Sobel Gradient

Sobel Edges

Thresholds are brittle, we can do better!

Two Generic Approaches for **Edge** Detection





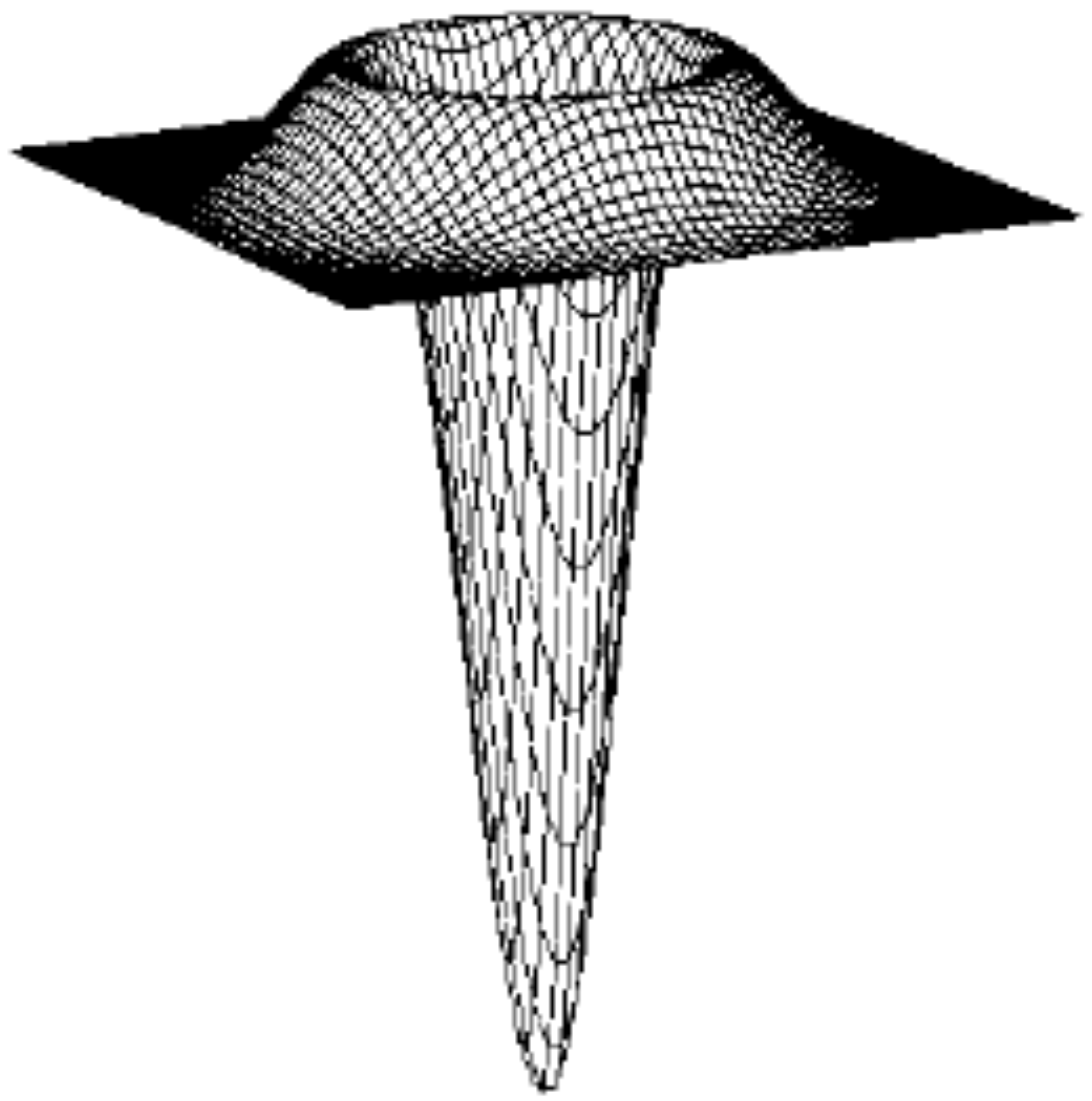
Two generic approaches to **edge point detection**:

— (significant) local extrema of a first derivative operator

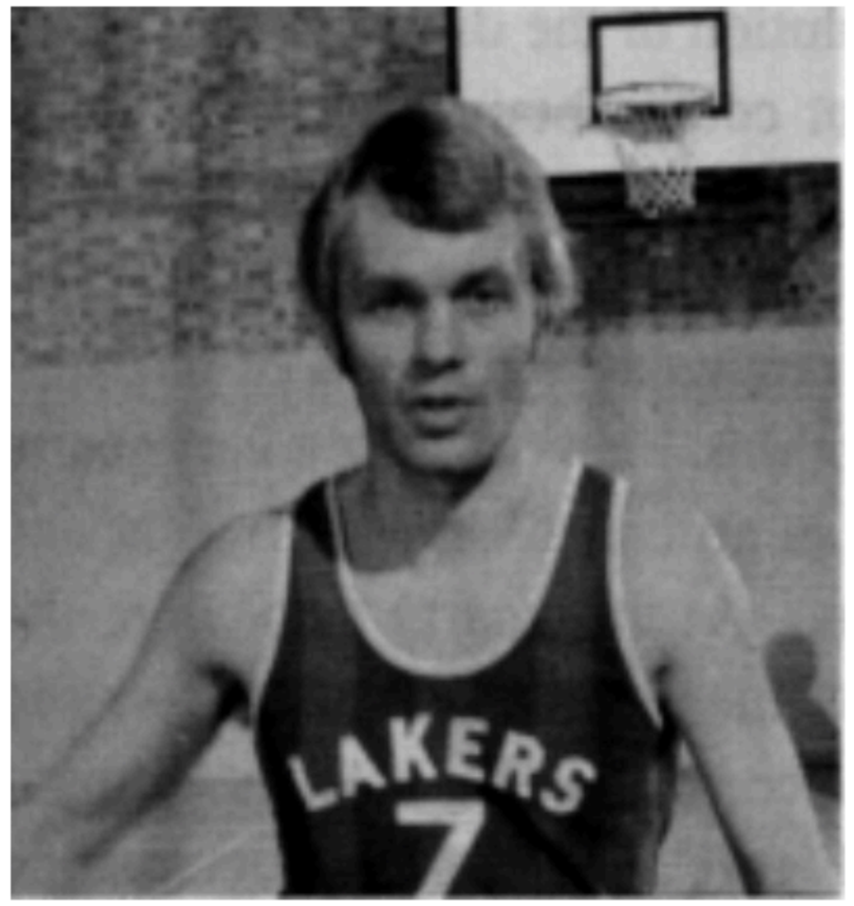
— zero crossings of a second derivative operator

Lecture 11: Marr / Hildreth **Laplacian of Gaussian**

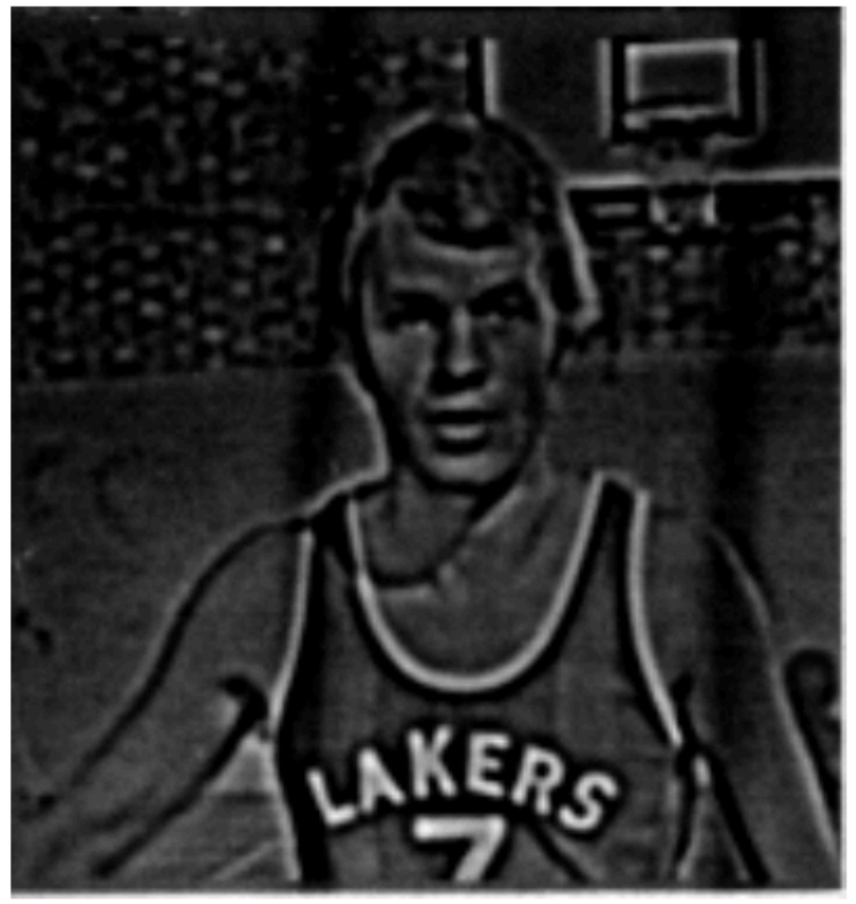
A “**zero crossings** of a second derivative operator” approach



Lecture 11: Marr / Hildreth **Laplacian of Gaussian**



Original Image



LoG Filter

Zero Crossings

Scale (σ)

Image From: A. Campilho

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

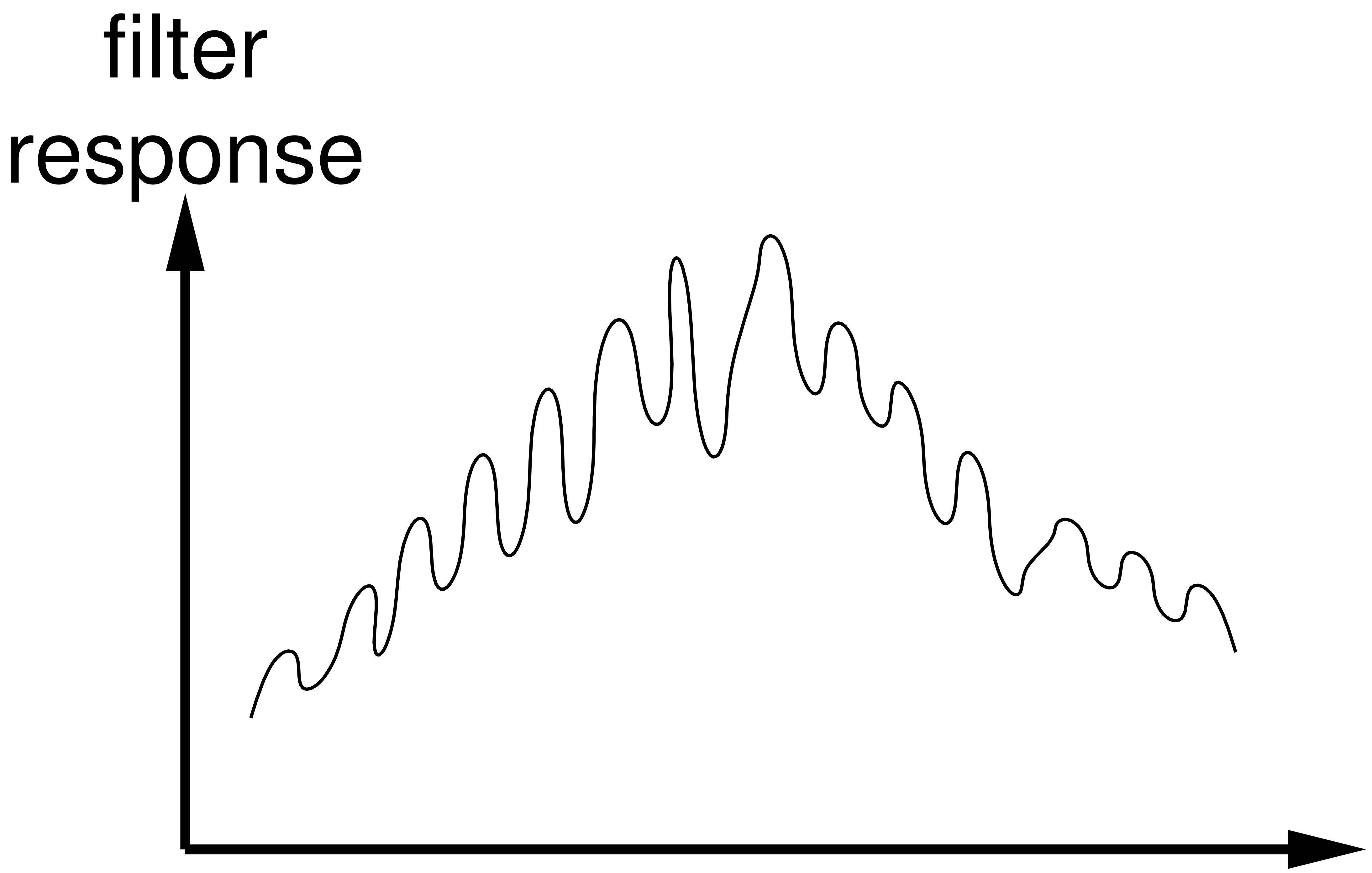
Design Criteria:

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)

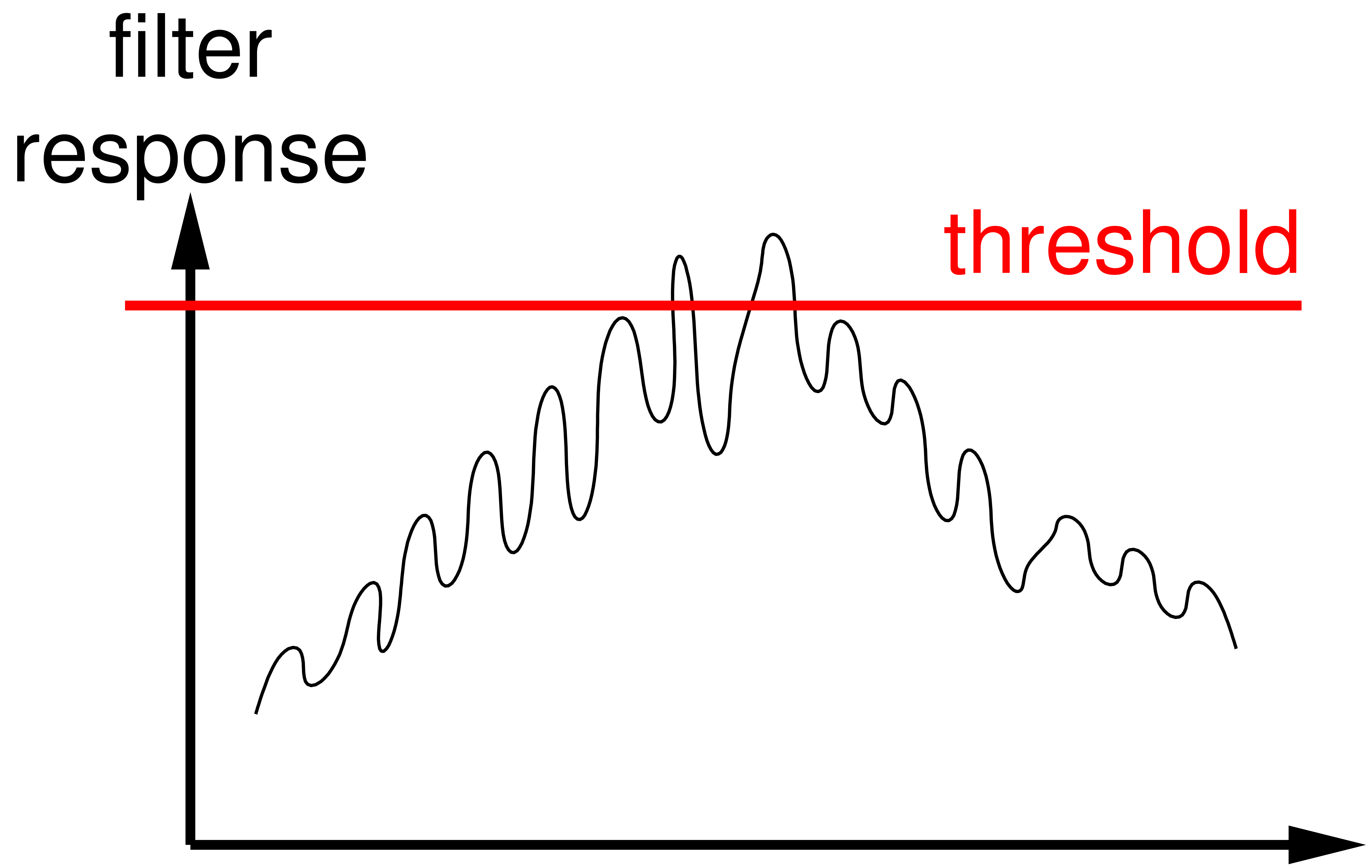
Example: Edge Detection



Question: How many edges are there?

Question: What is the position of each edge?

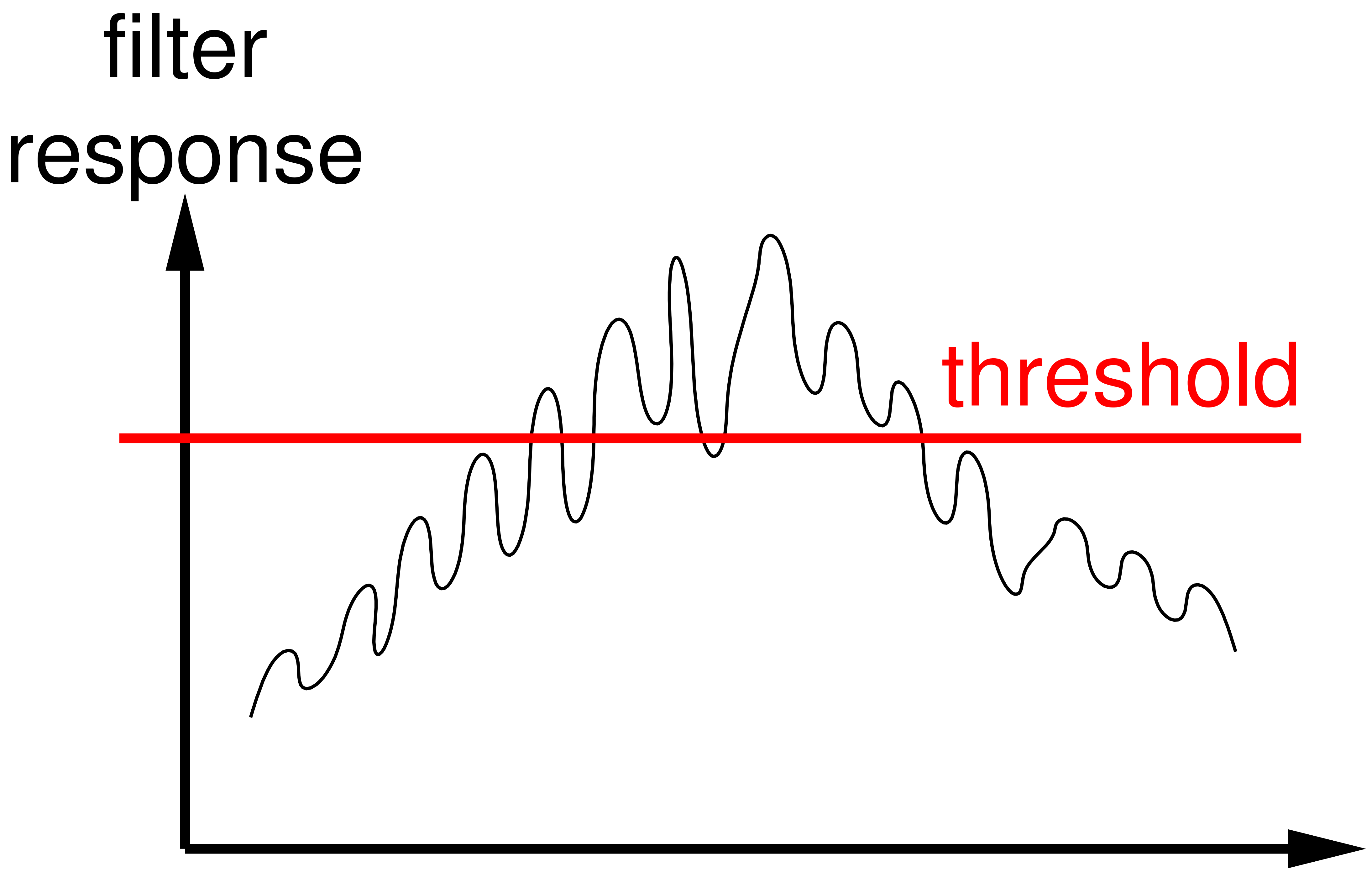
Example: Edge Detection



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

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

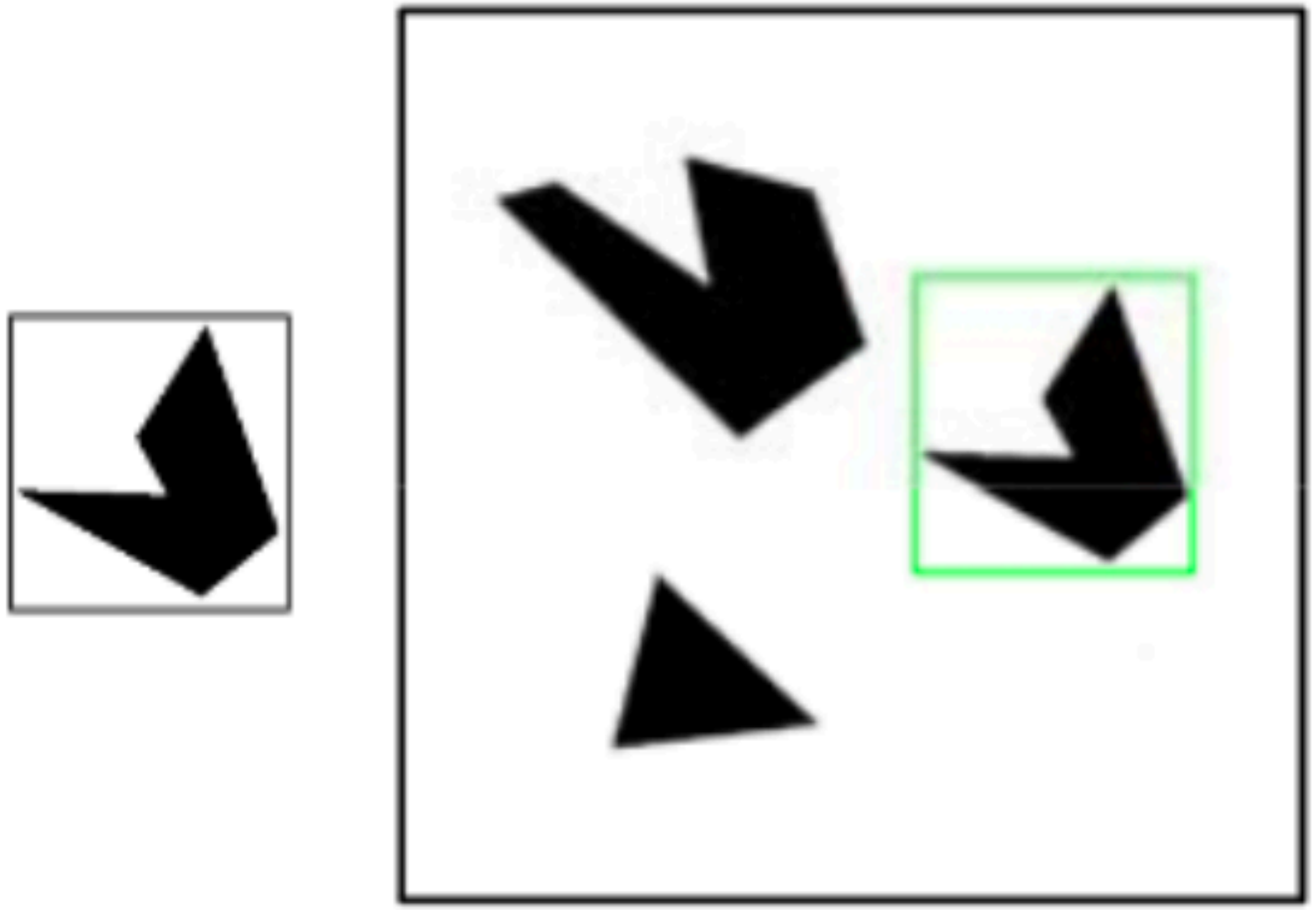
4. **Linking** and thresholding
   — Low, high edge-strength thresholds
   — Accept all edges over low threshold that are connected to edge over high threshold

Non-maxima Suppression

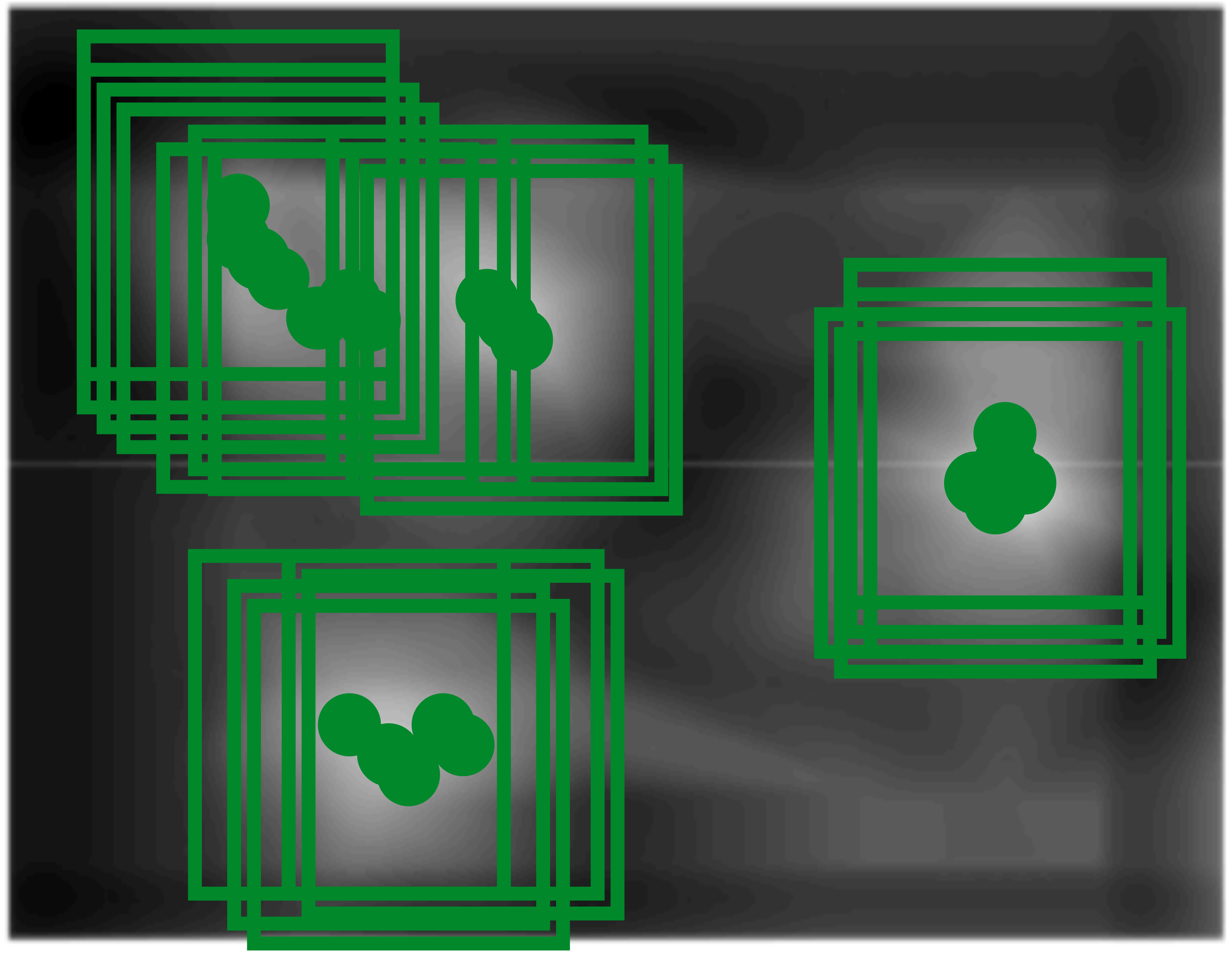
Idea: suppress near-by similar detections to obtain one “true” result

Non-maxima Suppression

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



Detected template

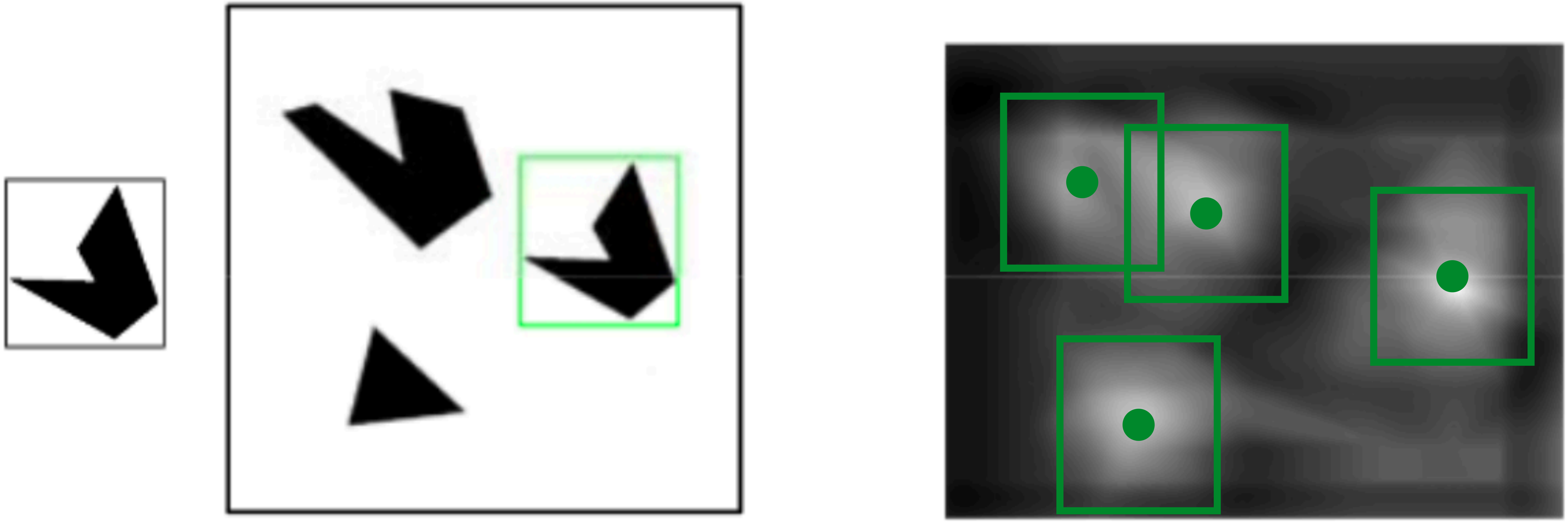


Correlation map

Slide Credit: Kristen Grauman

Non-maxima Suppression

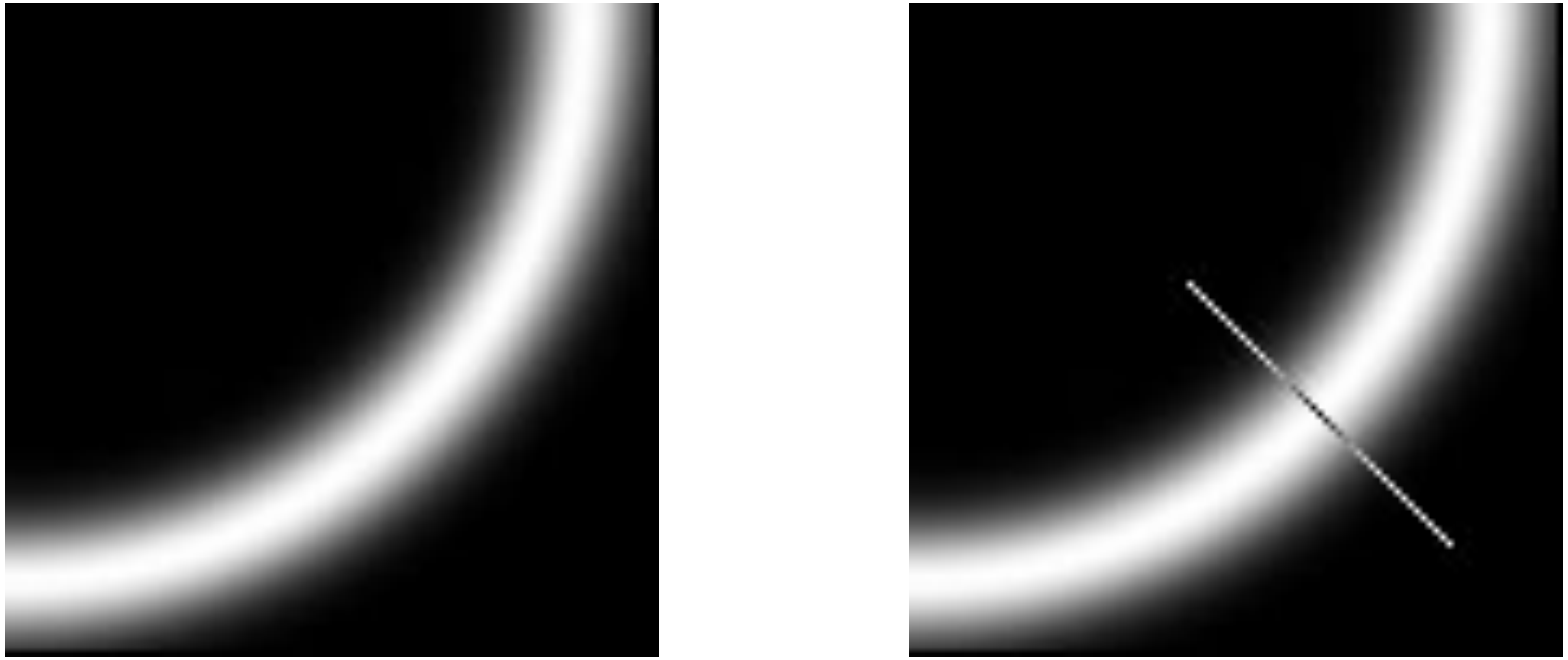
Idea: suppress near-by similar detections to obtain one “true” result



Detected template　　　　Correlation map

Slide Credit: Kristen Grauman

Non-maxima Suppression

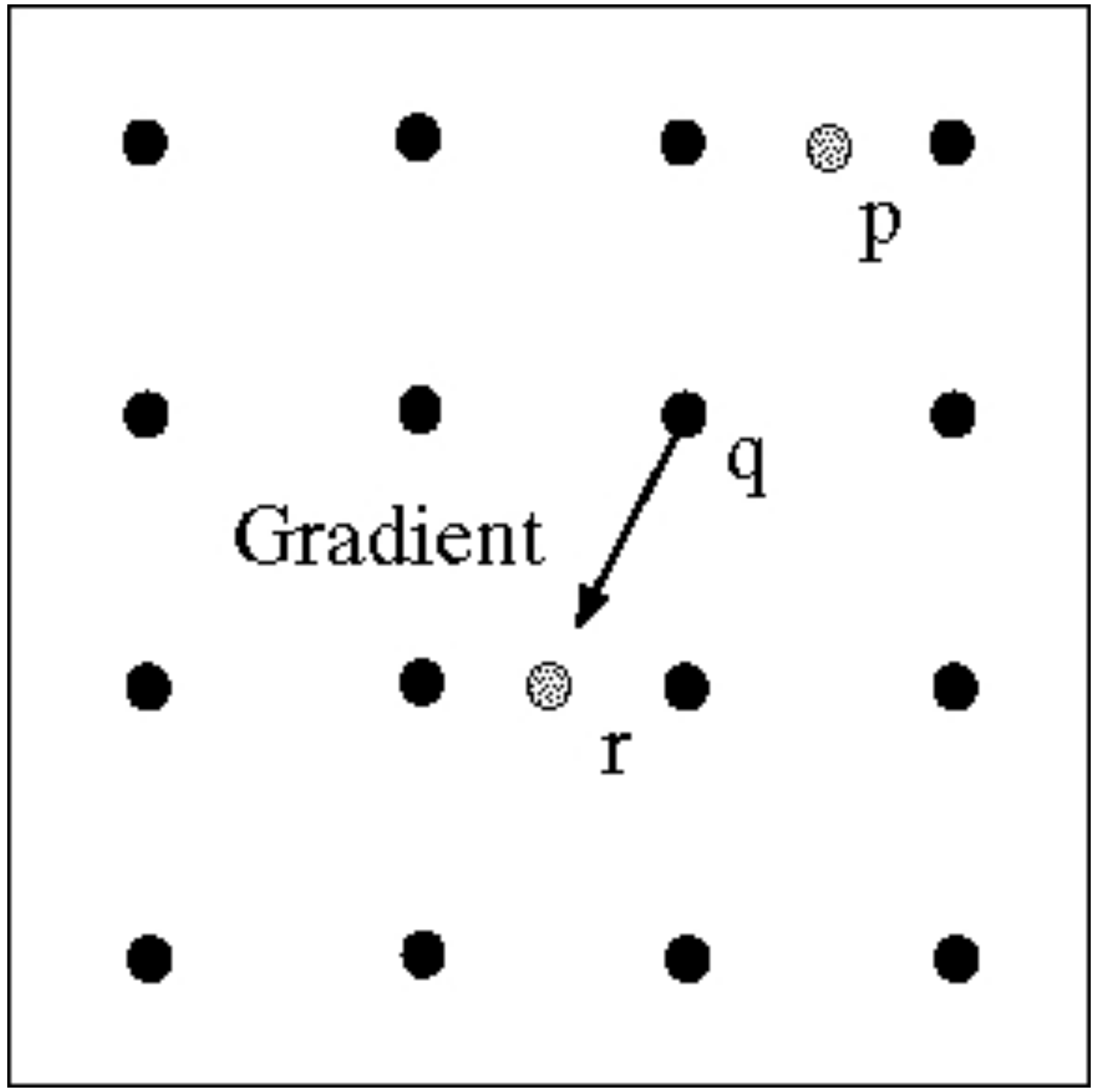


Forsyth & Ponce (1st ed.) Figure 8.11

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

Non-maxima Suppression

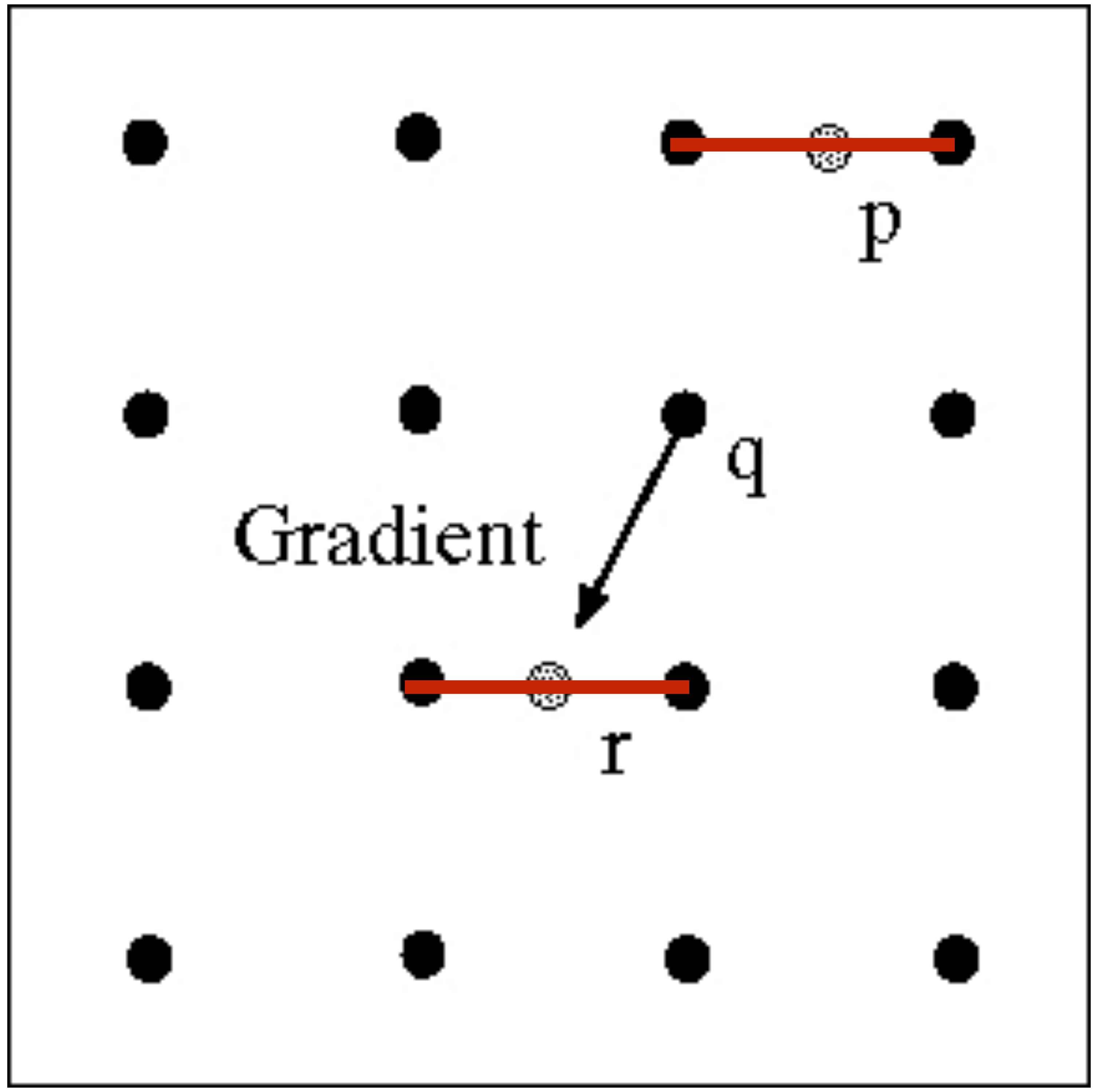
Value at *q* must be larger than interpolated values at *p* and *r*



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

Non-maxima Suppression

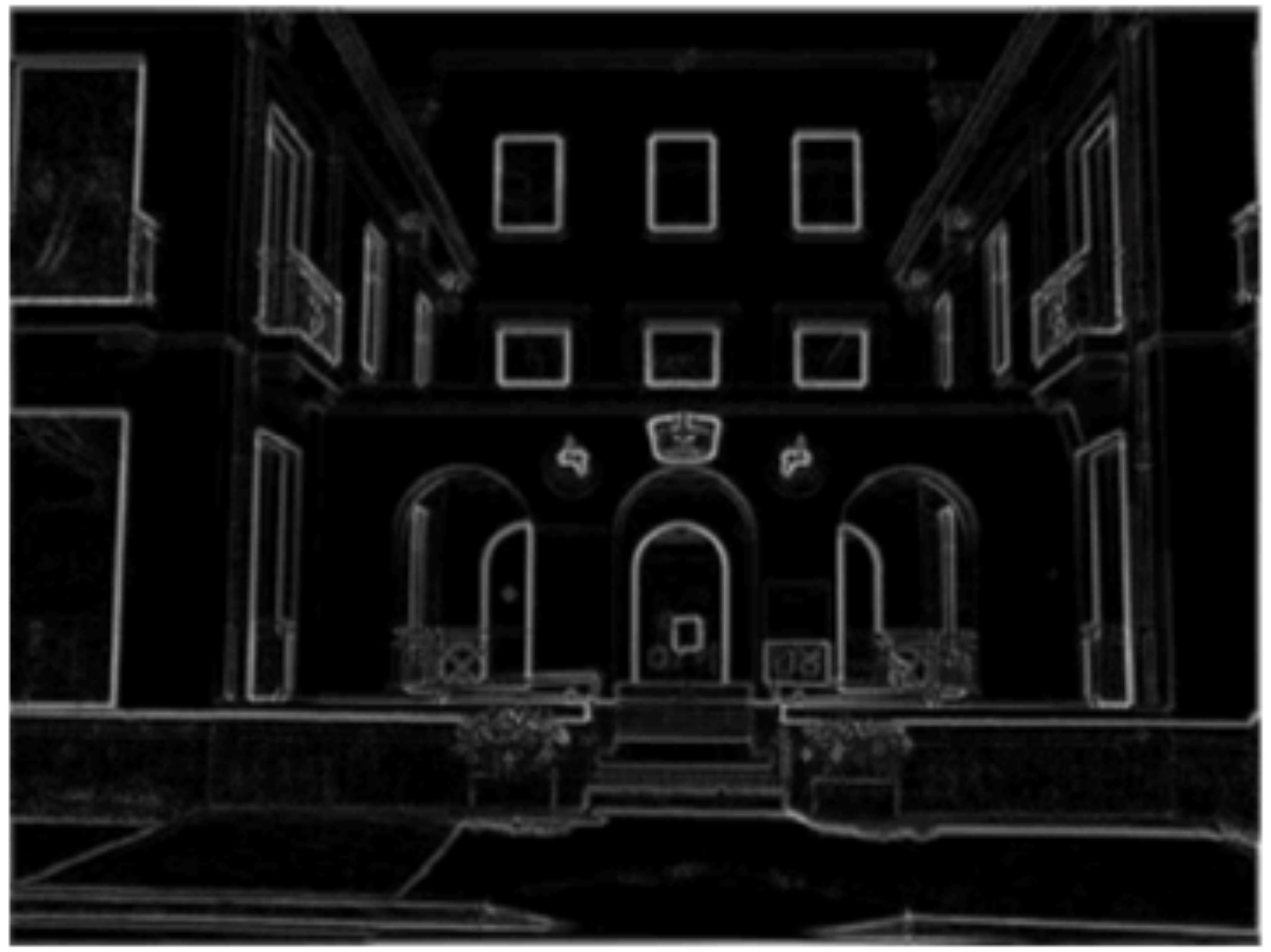
Value at *q* must be larger than interpolated values at *p* and *r*



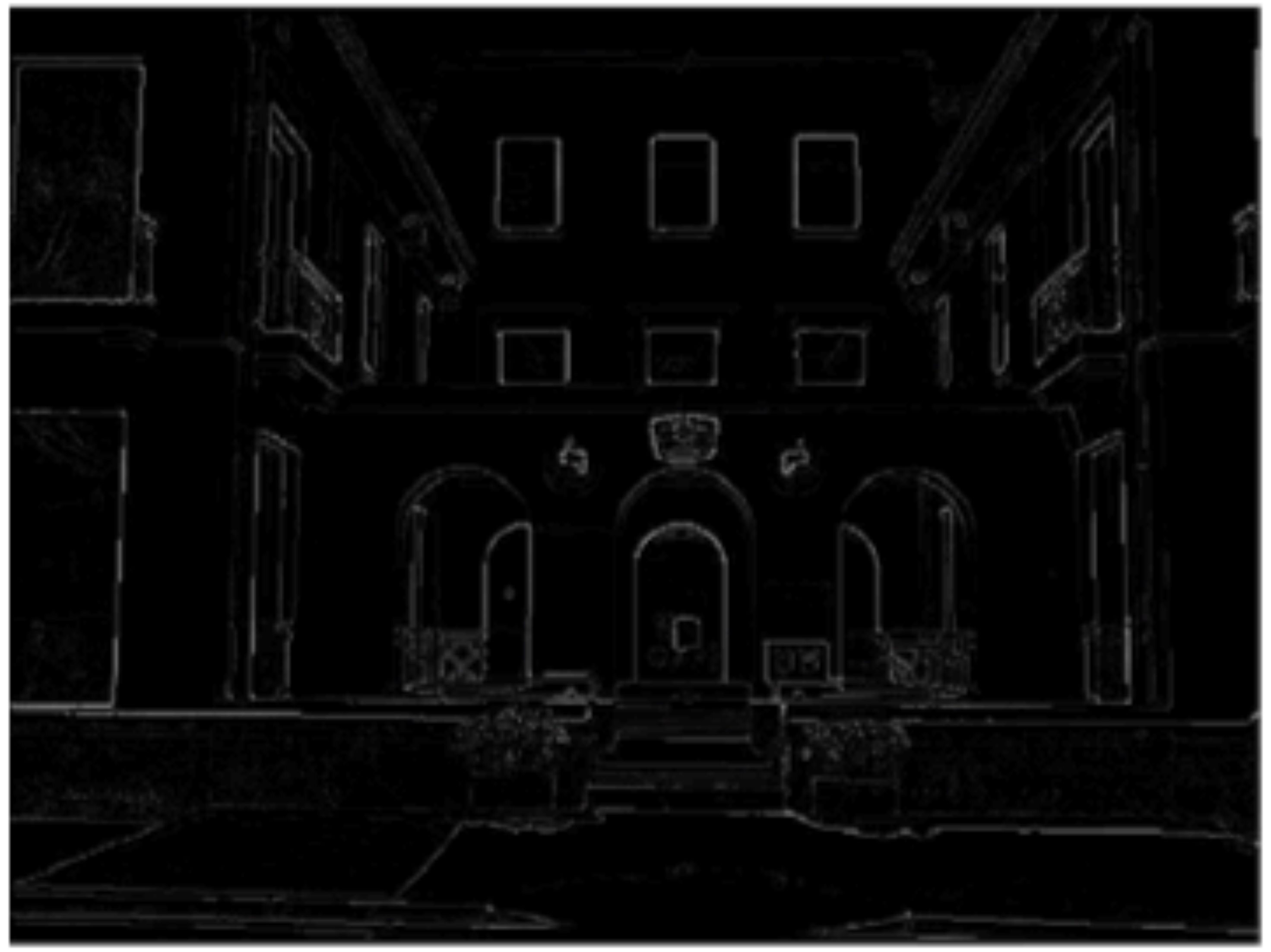
Forsyth & Ponce (2nd ed.) Figure 5.5 left

Example: Non-maxima Suppression

Original Image



Gradient Magnitude



courtesy of G. Loy

Non-maxima Suppression

Slide Credit: Christopher Rasmussen

Example

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

Example

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

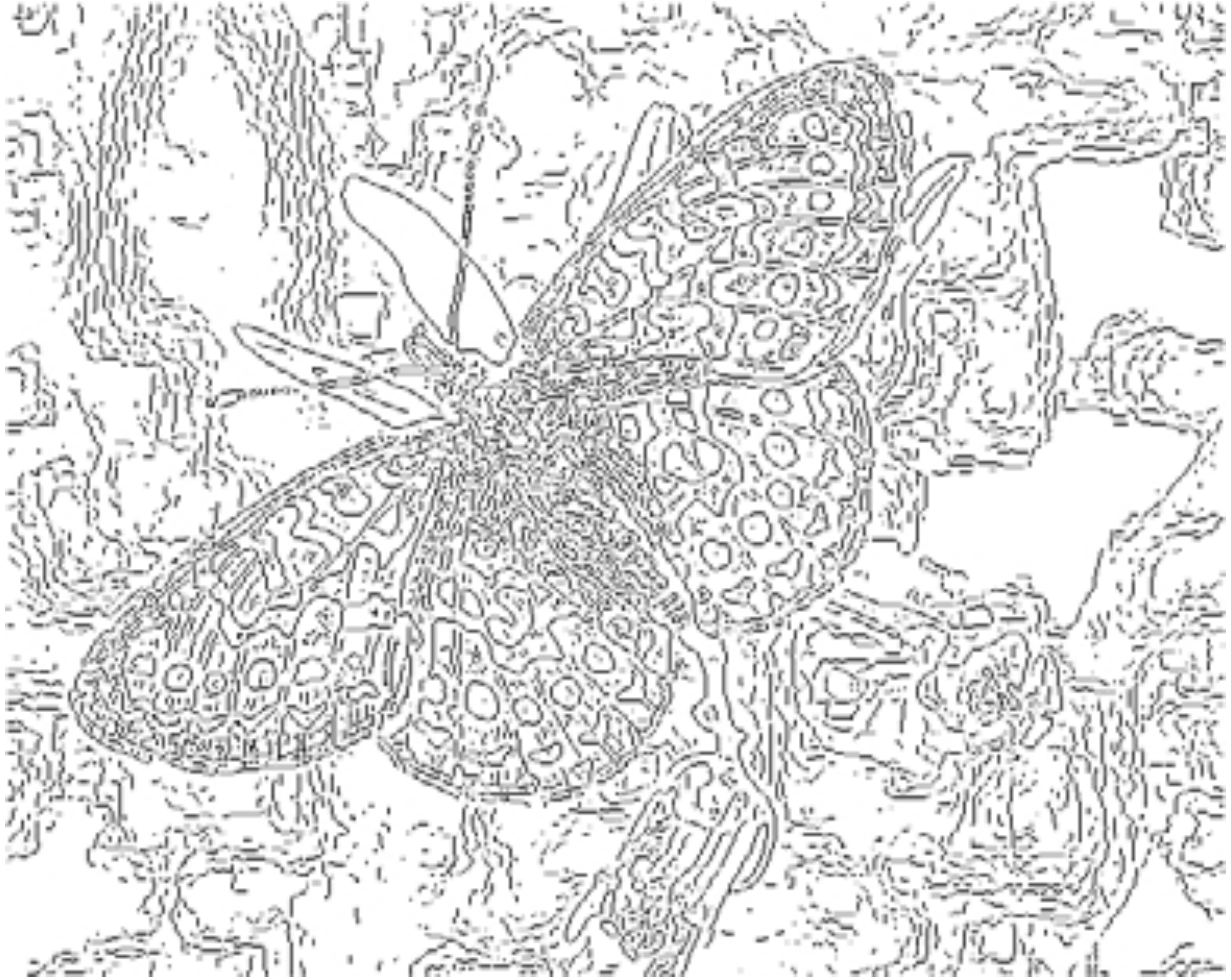


Figure 8.13 bottom left

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

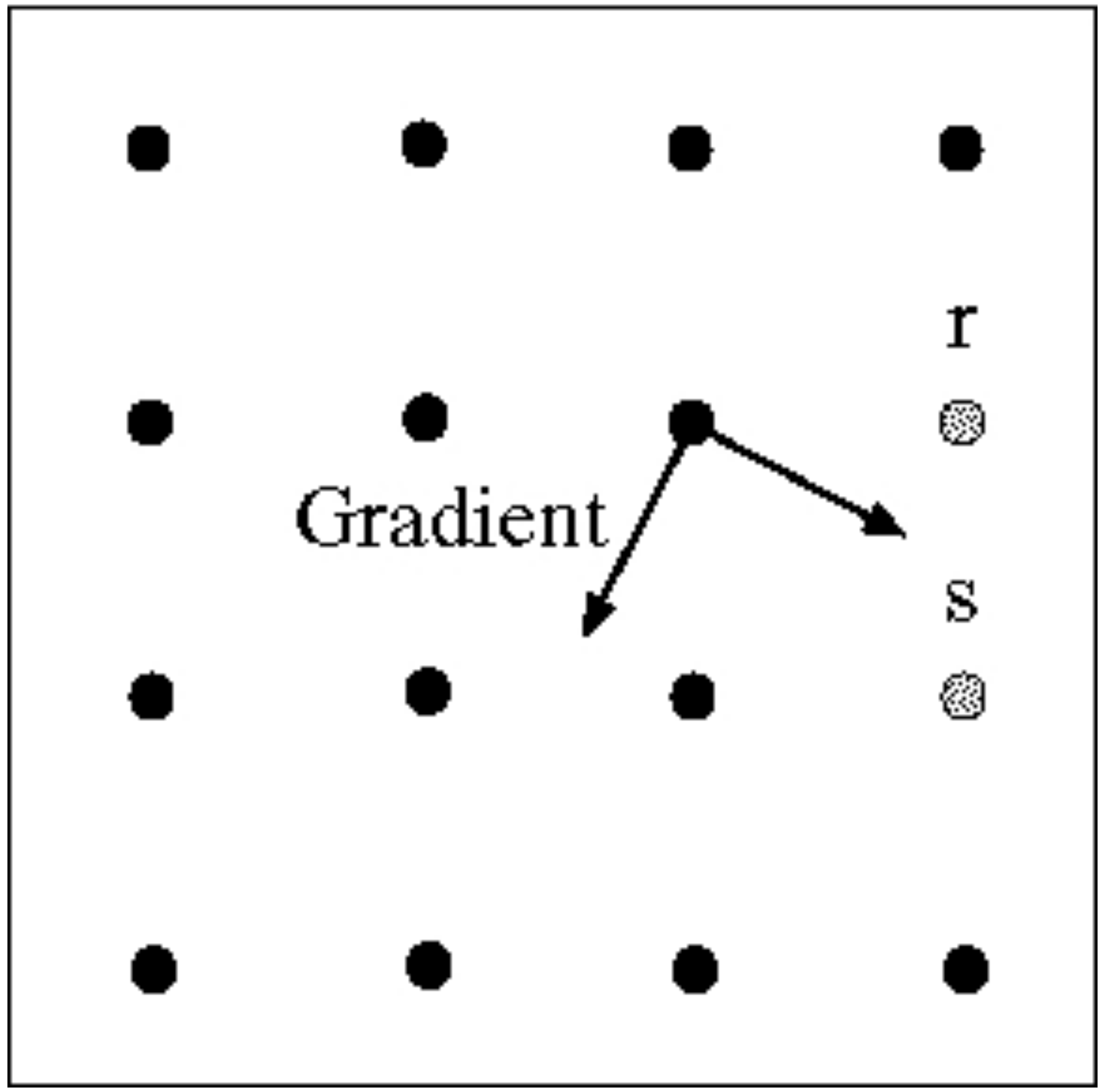
Example

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

Figure 8.13 bottom right

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

Linking Edge Points



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 khigh to find strong edges to start edge chain

— Use klow 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**?

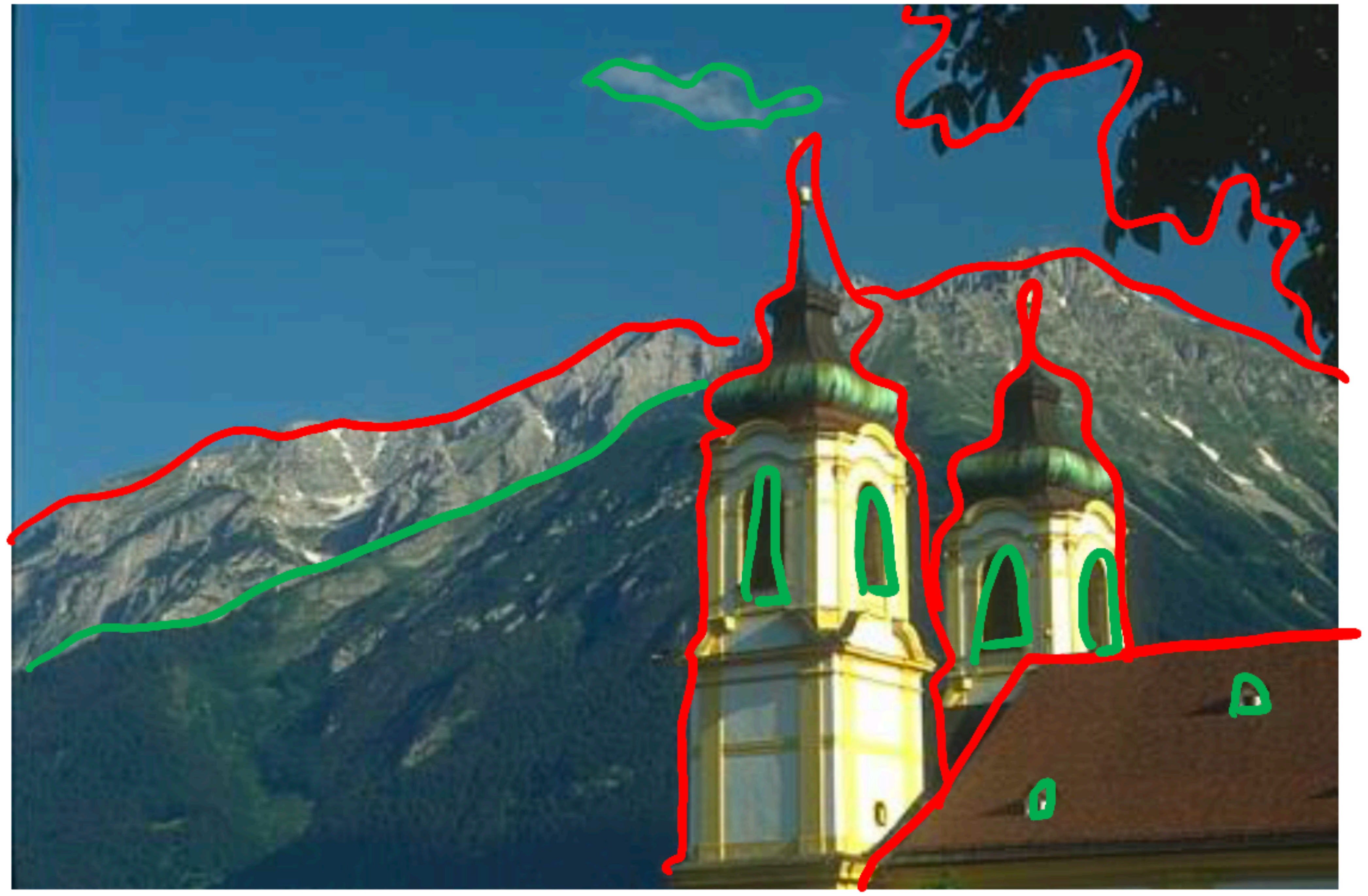
Edges are a property of the 2D image.

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**?

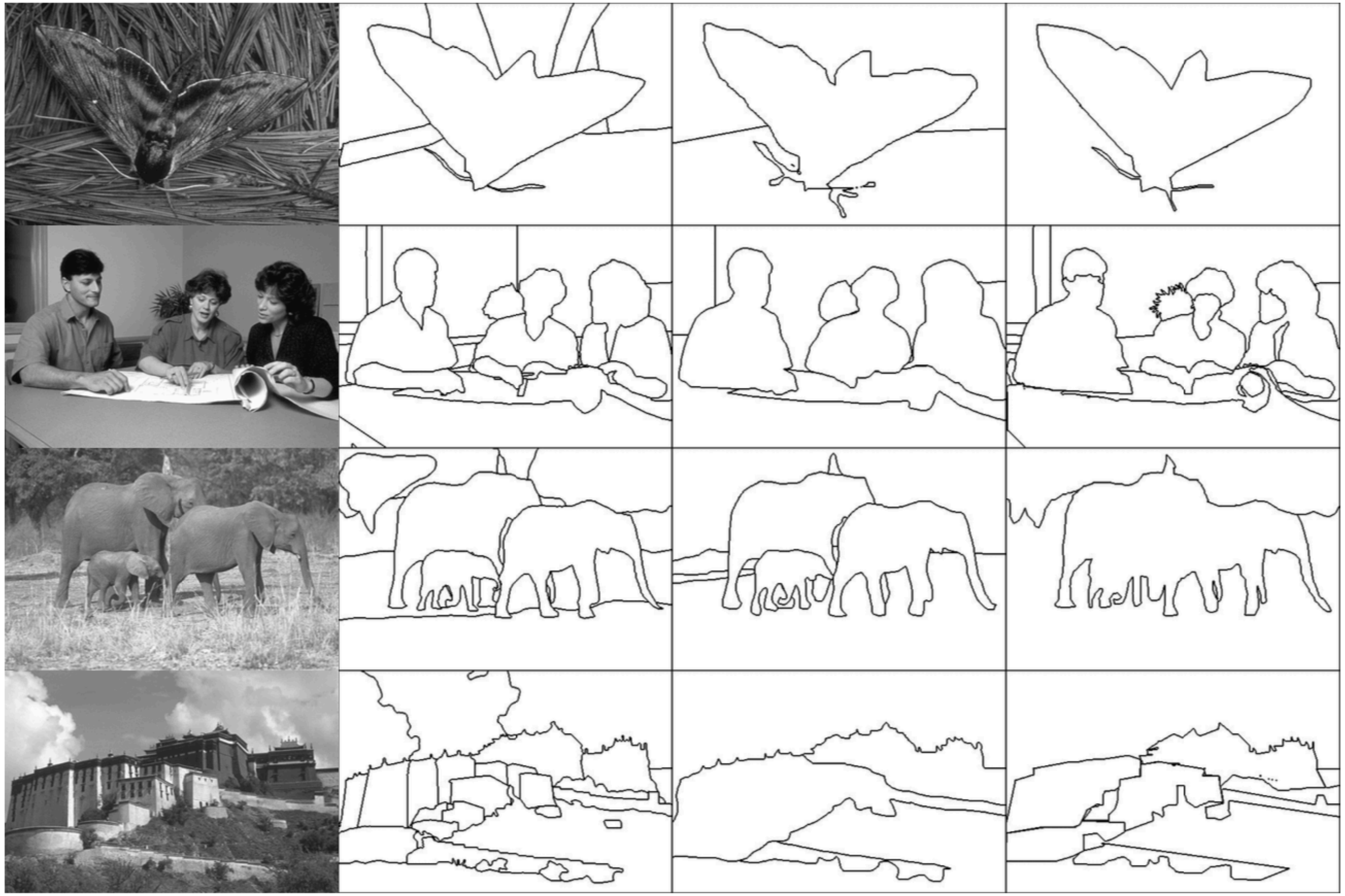


Figure Credit: Martin et al. 2001

How do humans perceive **boundaries**?

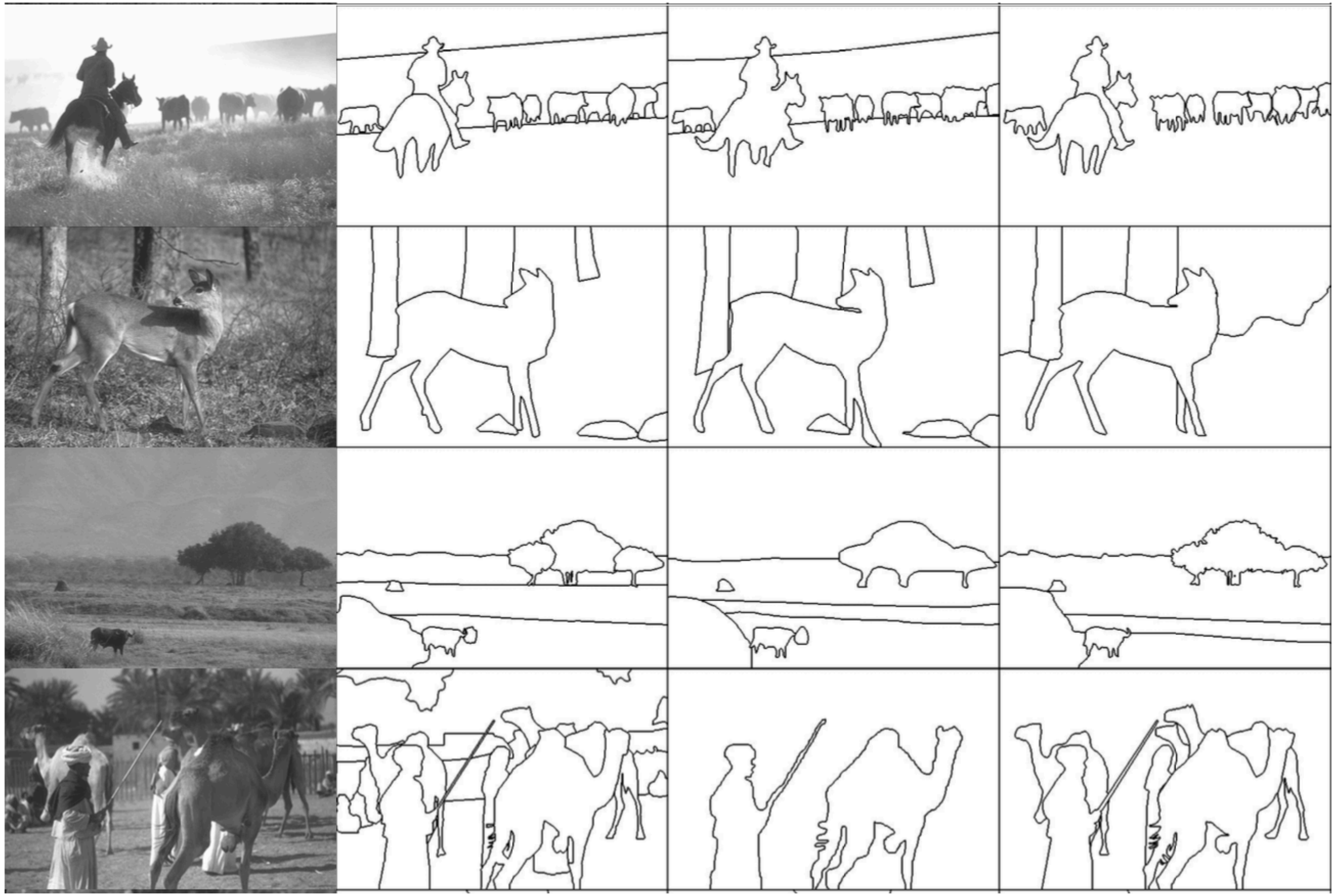
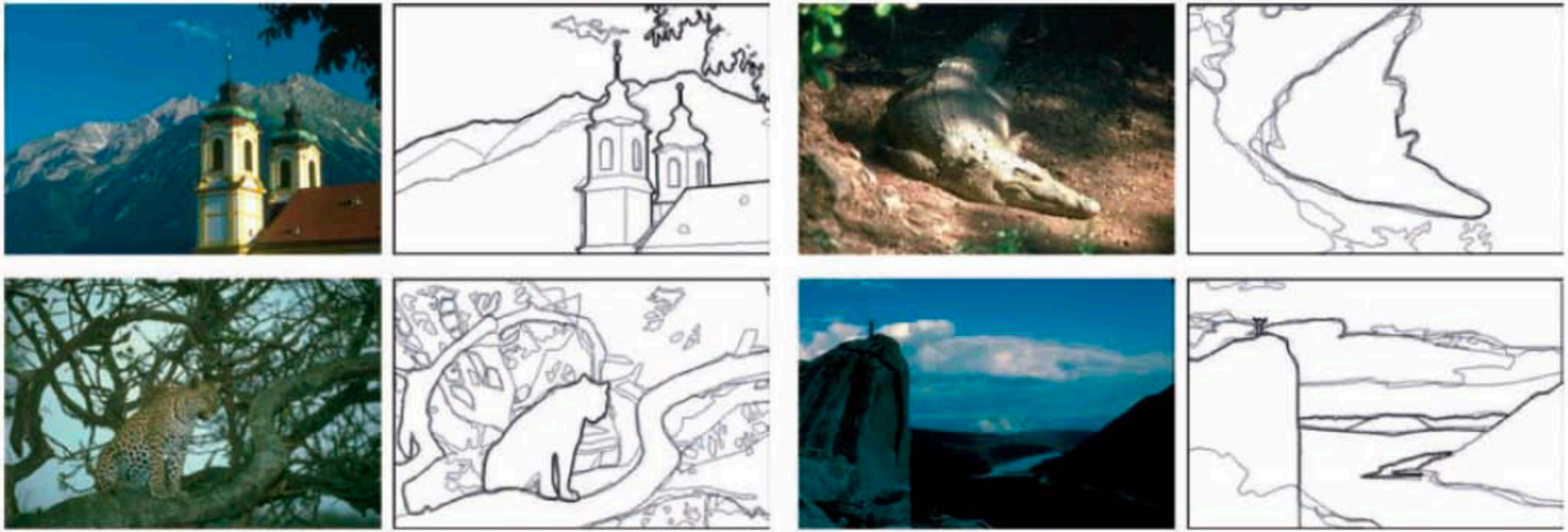


Figure Credit: Martin et al. 2001

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

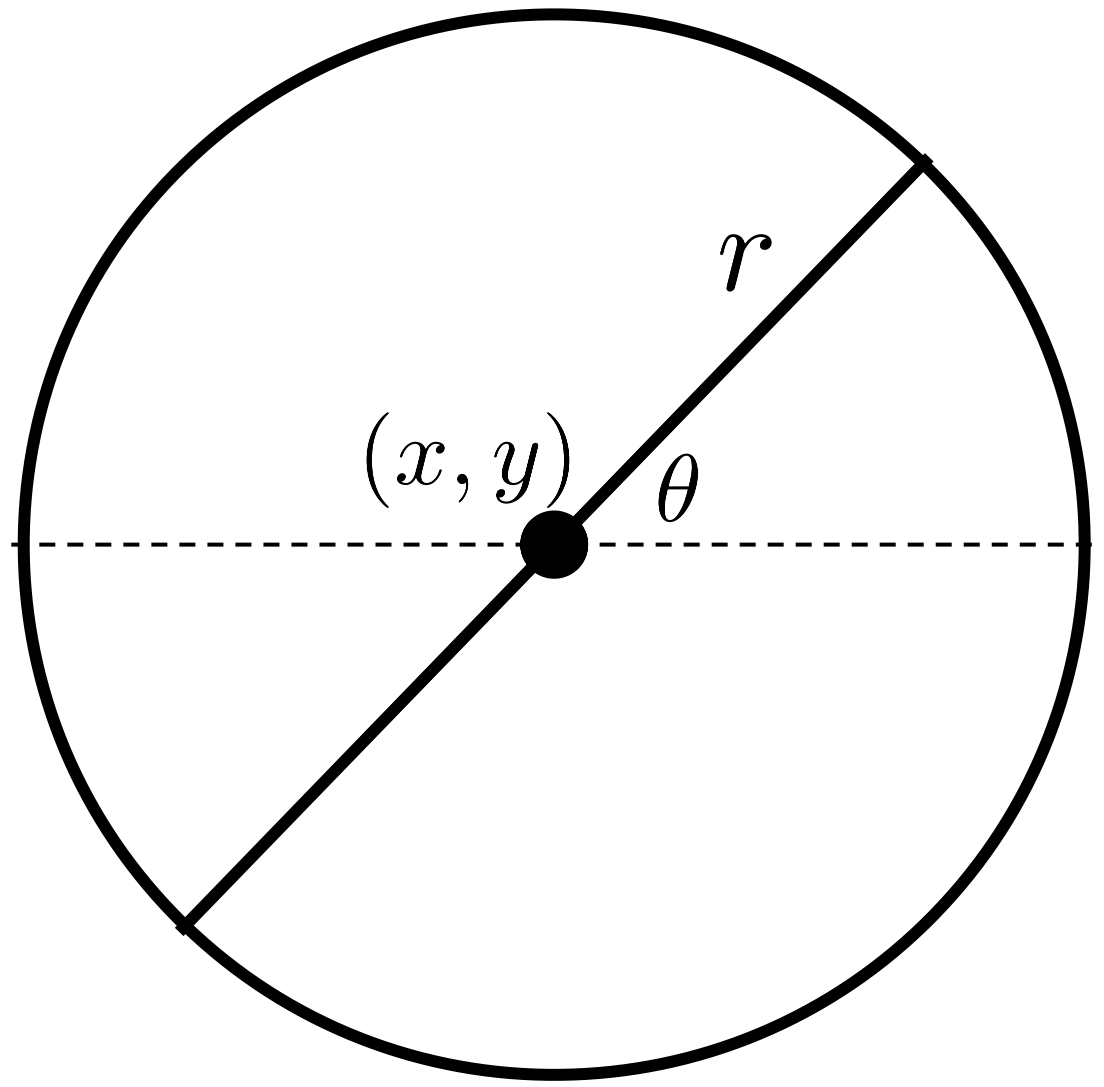
Boundary Detection: Example Approach

— 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



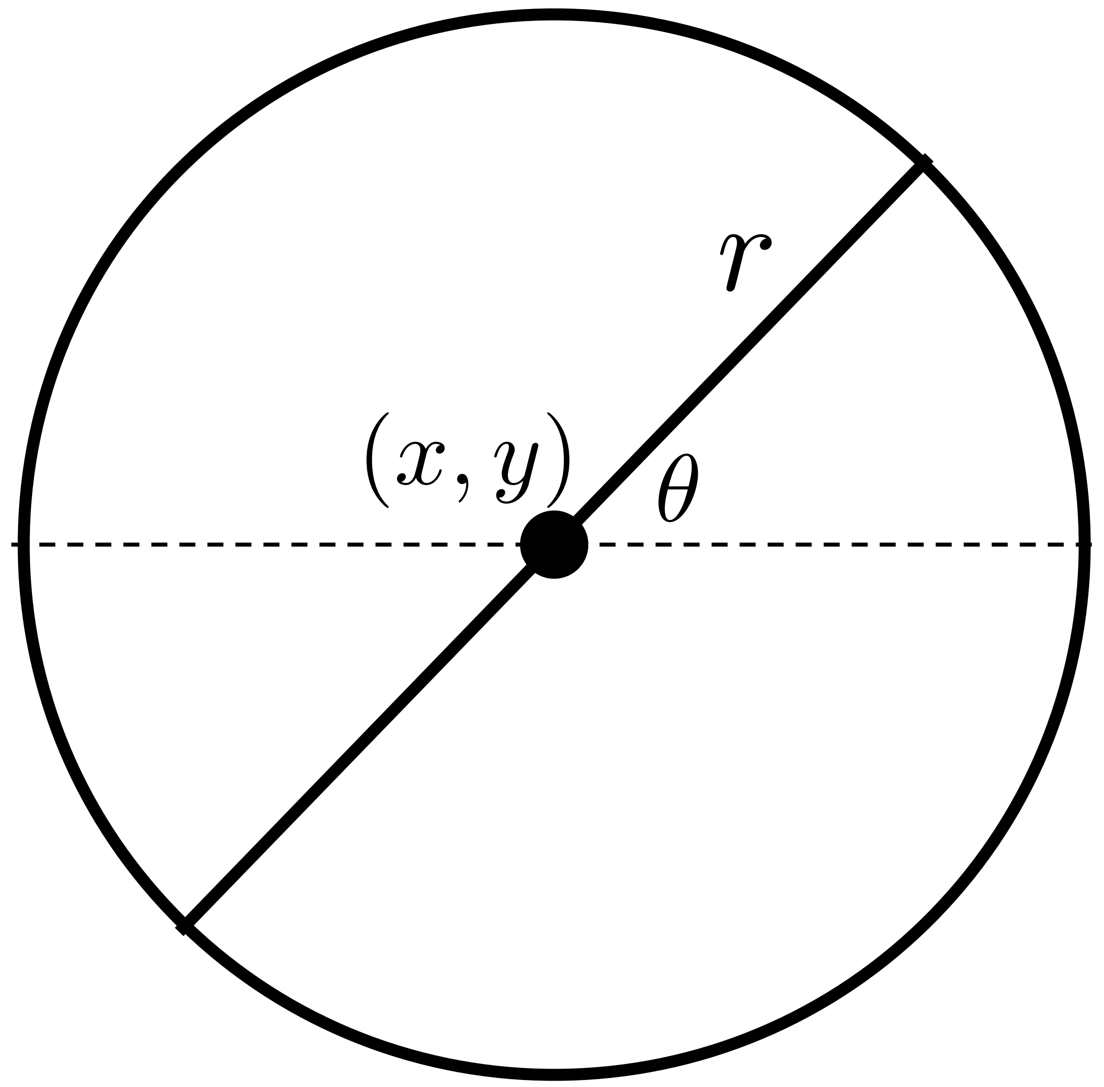
Boundary Detection: Example Approach

— 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)

Boundary Detection:

Features:

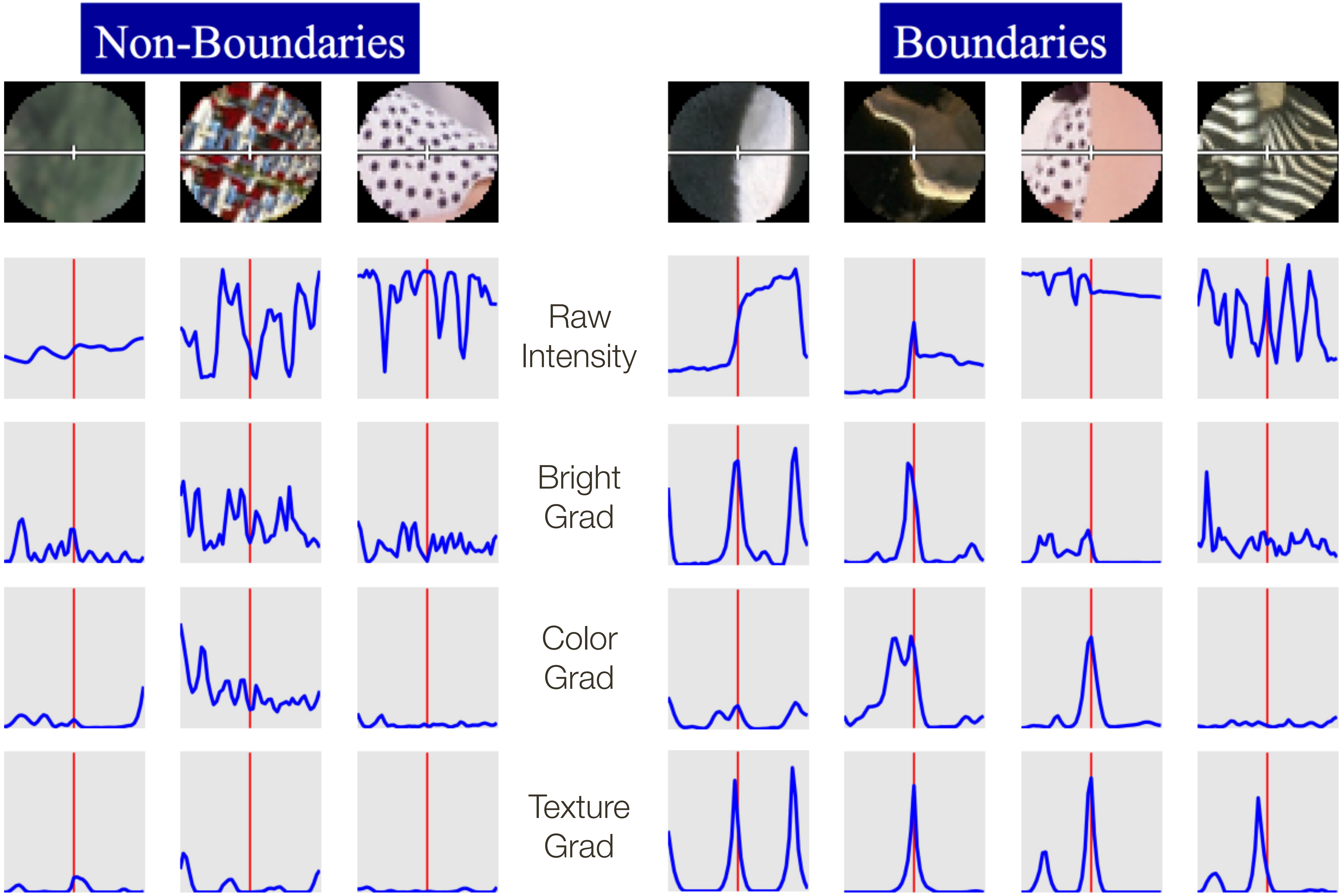
— Raw Intensity

— 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**:

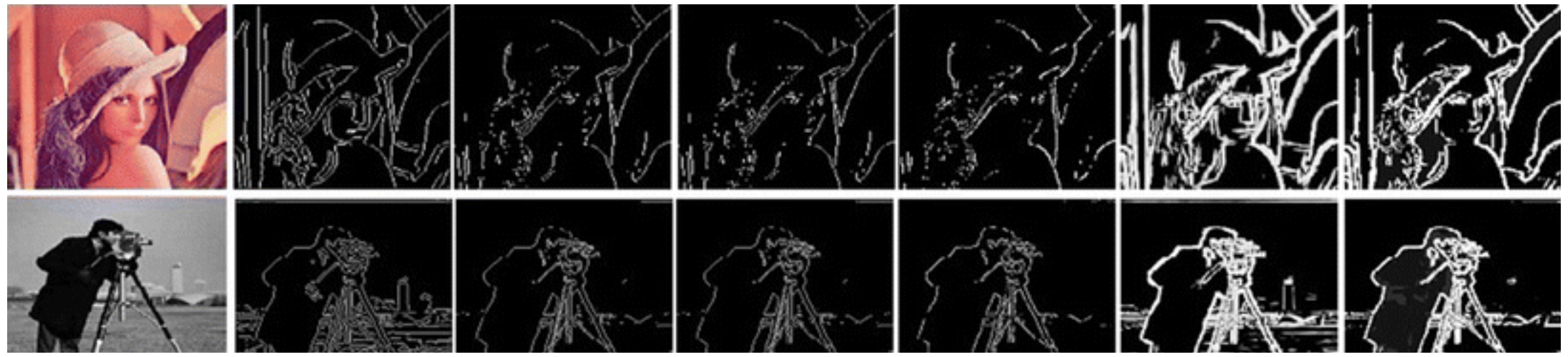
— local extrema of a first derivative operator → **Canny**

— 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

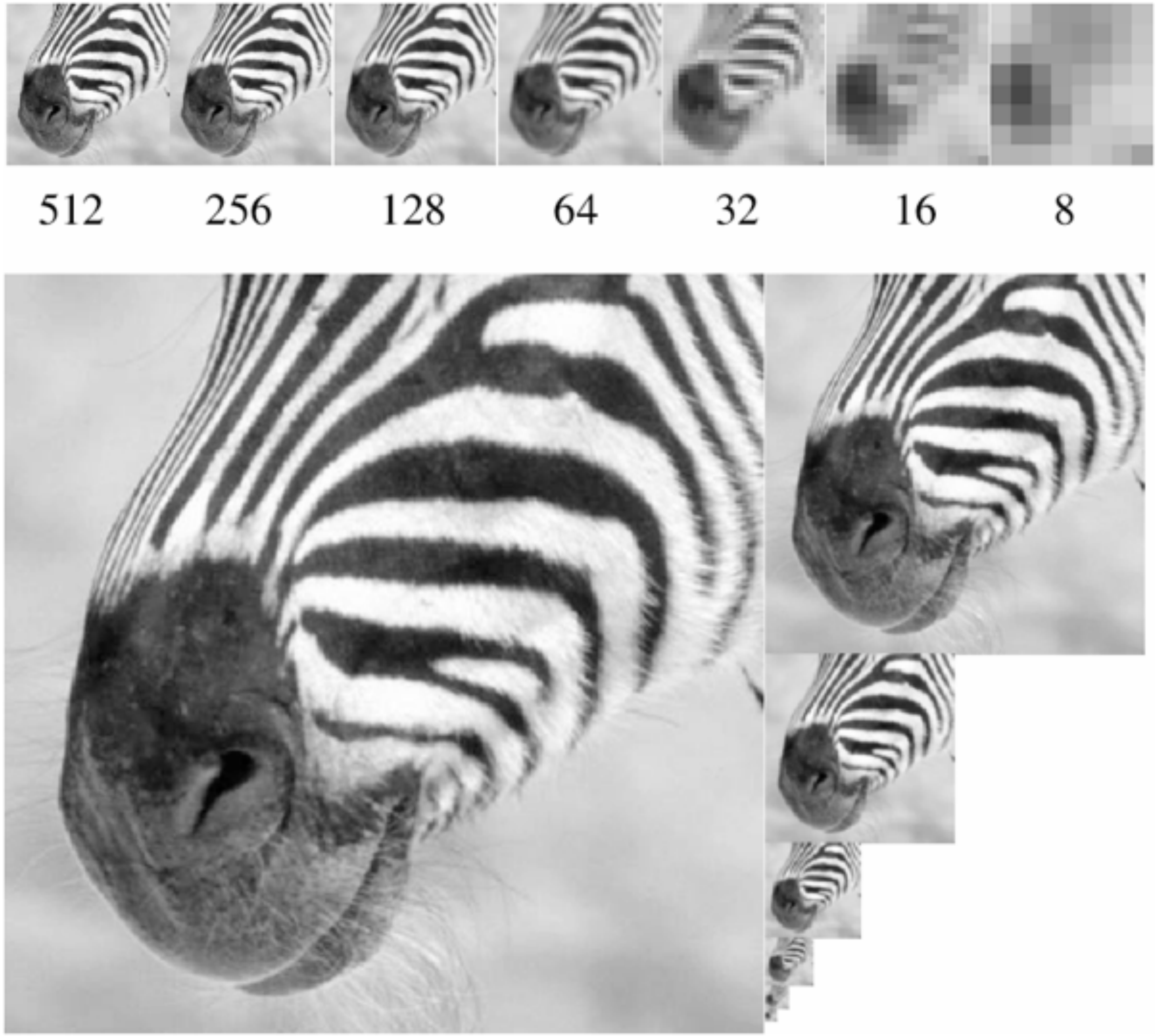
THE UNIVERSITY OF BRITISH COLUMBIA

CPSC 425: Computer Vision



Lecture 12: Laplacian Pyramids (aside for HW2)

Gaussian Pyramid



Forsyth & Ponce (2nd ed.) Figure 4.17

What happens to the details?

— They get smoothed out as we move to higher levels

What is preserved at the higher levels?

— Mostly large uniform regions in the original image

How would you reconstruct the original image from the image at the upper level?

— That's not possible

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Laplacian Pyramid

Building a **Laplacian** pyramid:

— Create a Gaussian pyramid

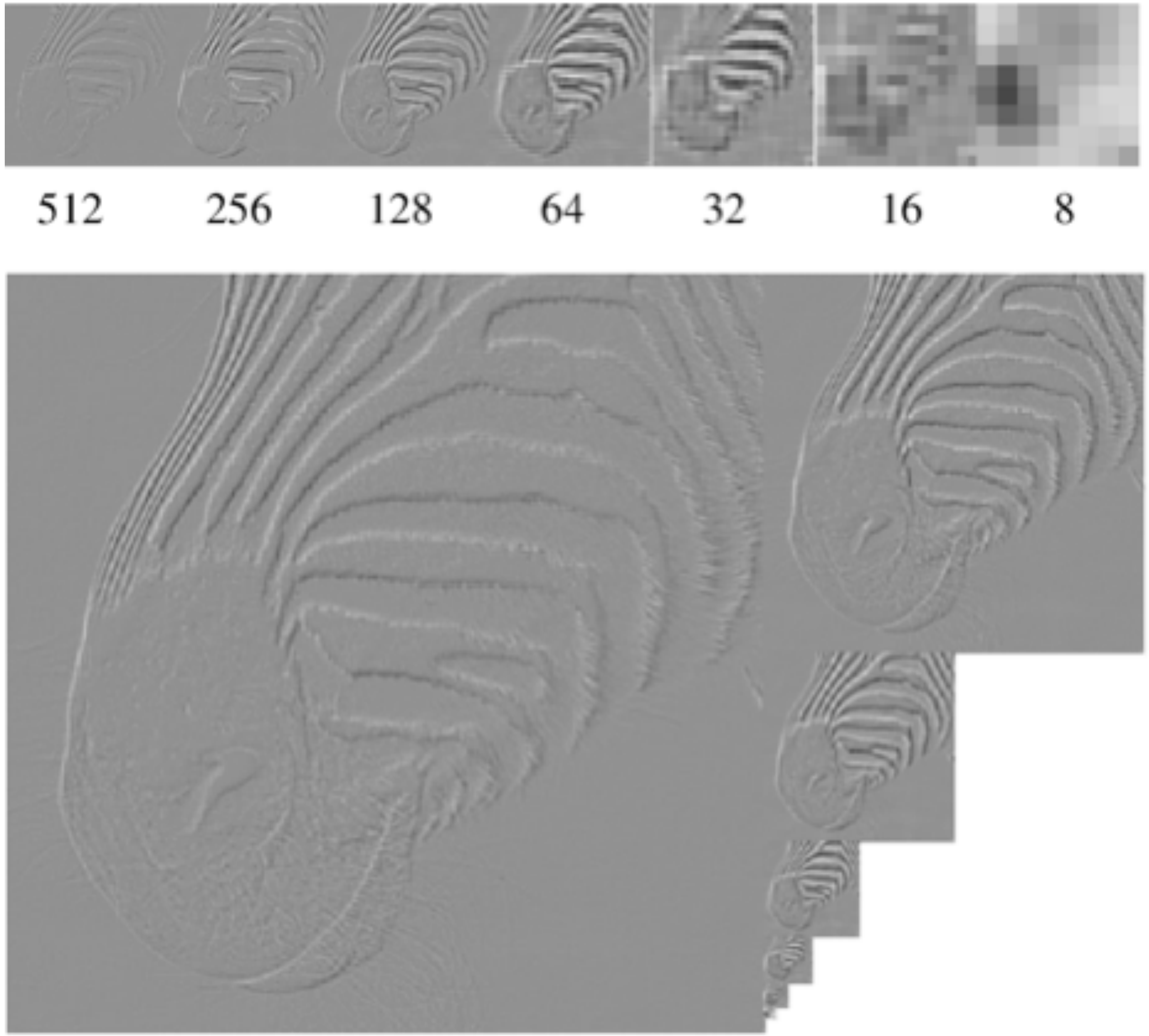
— Take the difference between one Gaussian pyramid level and the next (before subsampling)

Properties

— Also known as the difference-of-Gaussian (DOG) function, a close approximation to the Laplacian

— It is a band pass filter – each level represents a different band of spatial frequencies

Laplacian Pyramid



At each level, retain the residuals instead of the blurred images themselves.

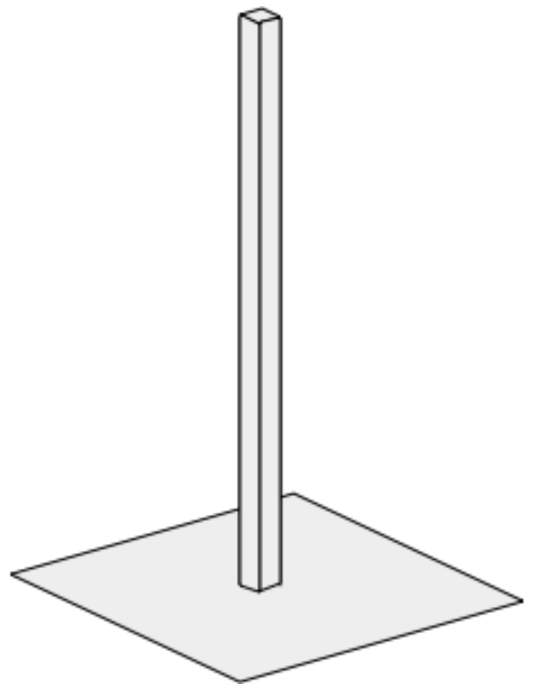
Why is it called Laplacian Pyramid?

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

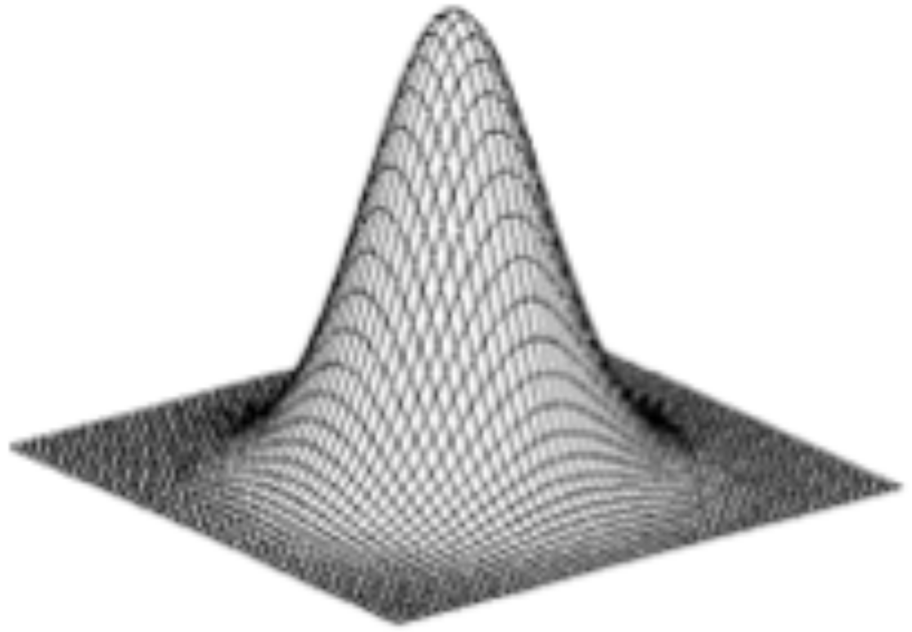
Why **Laplacian** Pyramid?

-

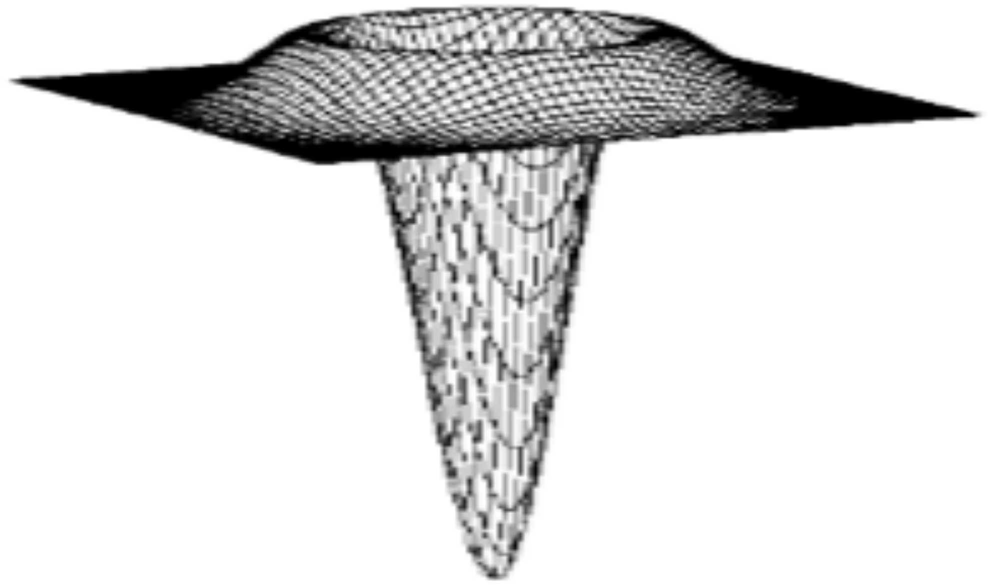
=



-



≈



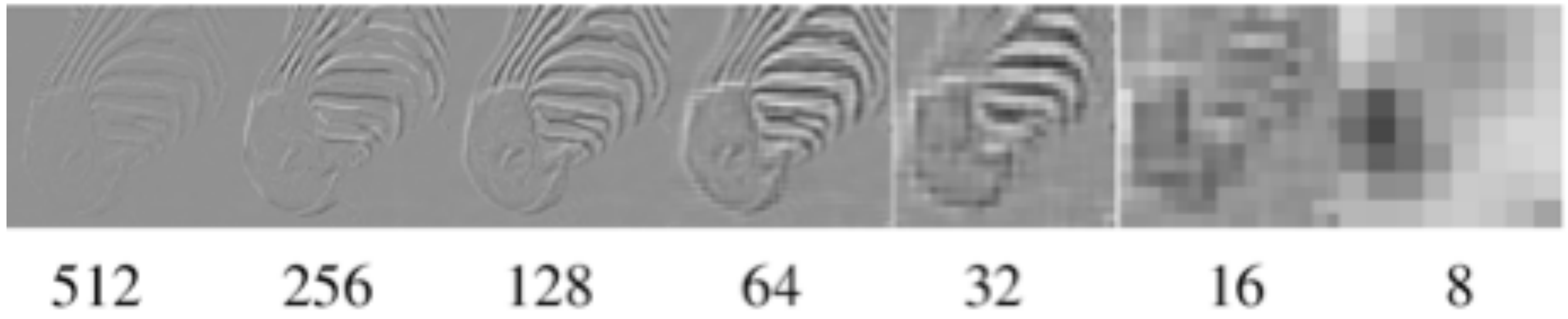
unit

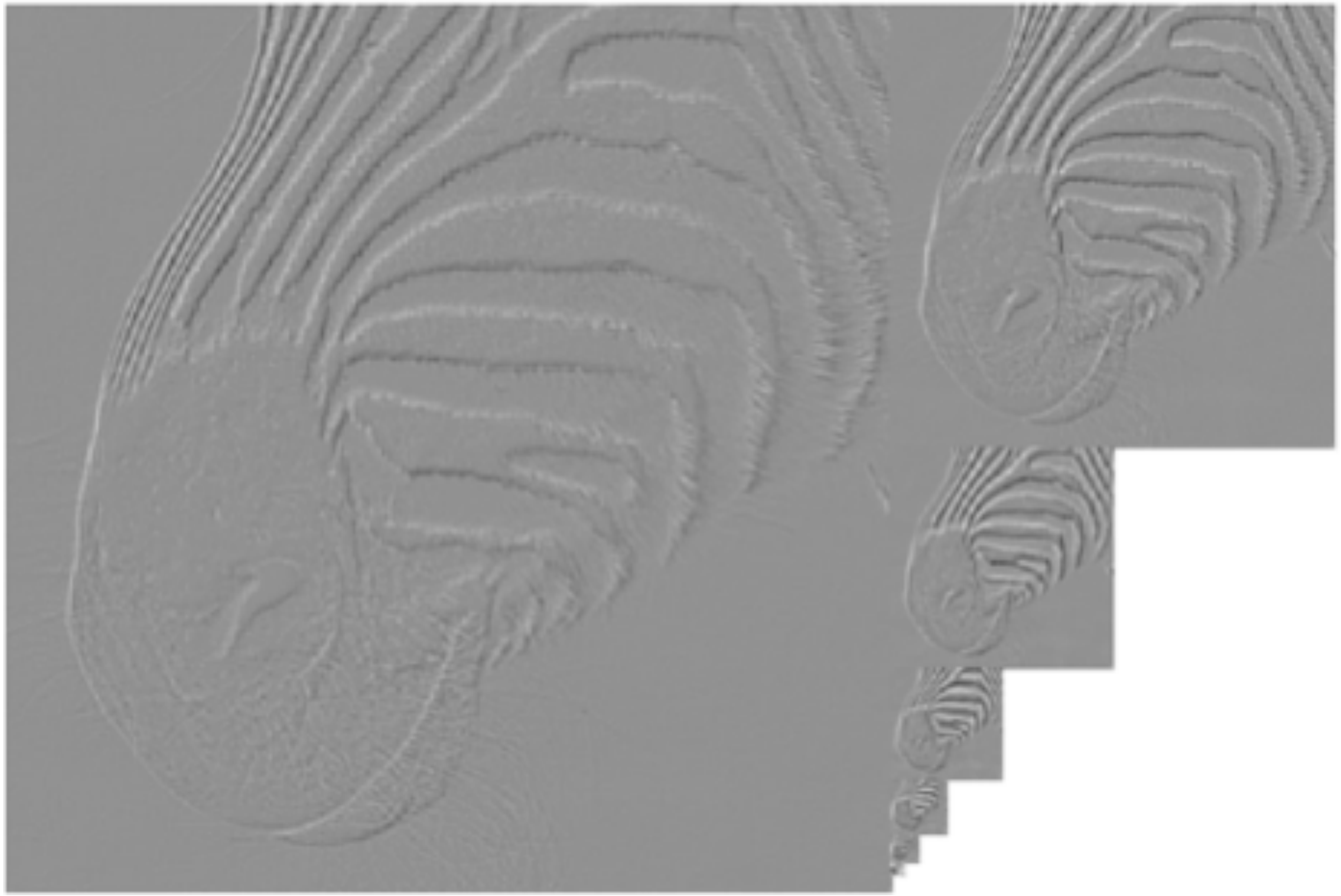
Gaussian

Laplacian

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Laplacian Pyramid





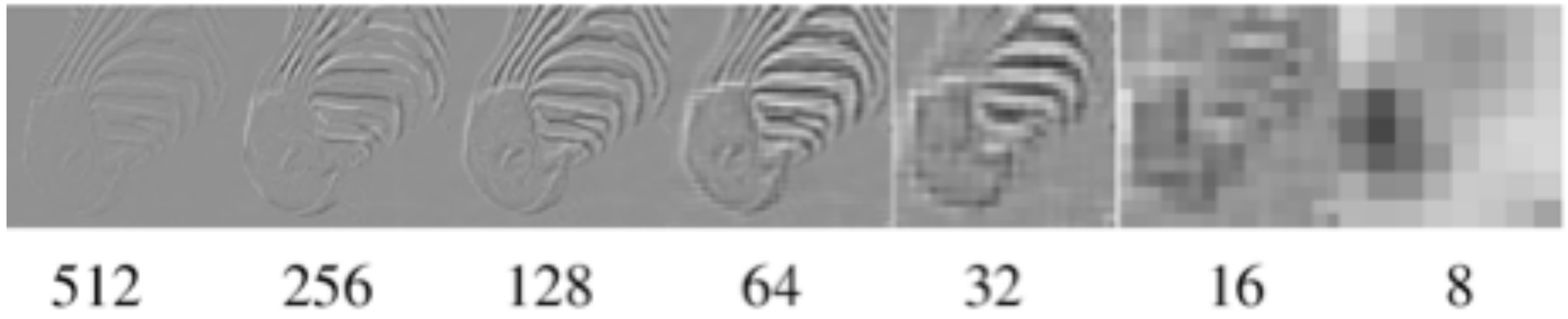
At each level, retain the residuals instead of the blurred images themselves.

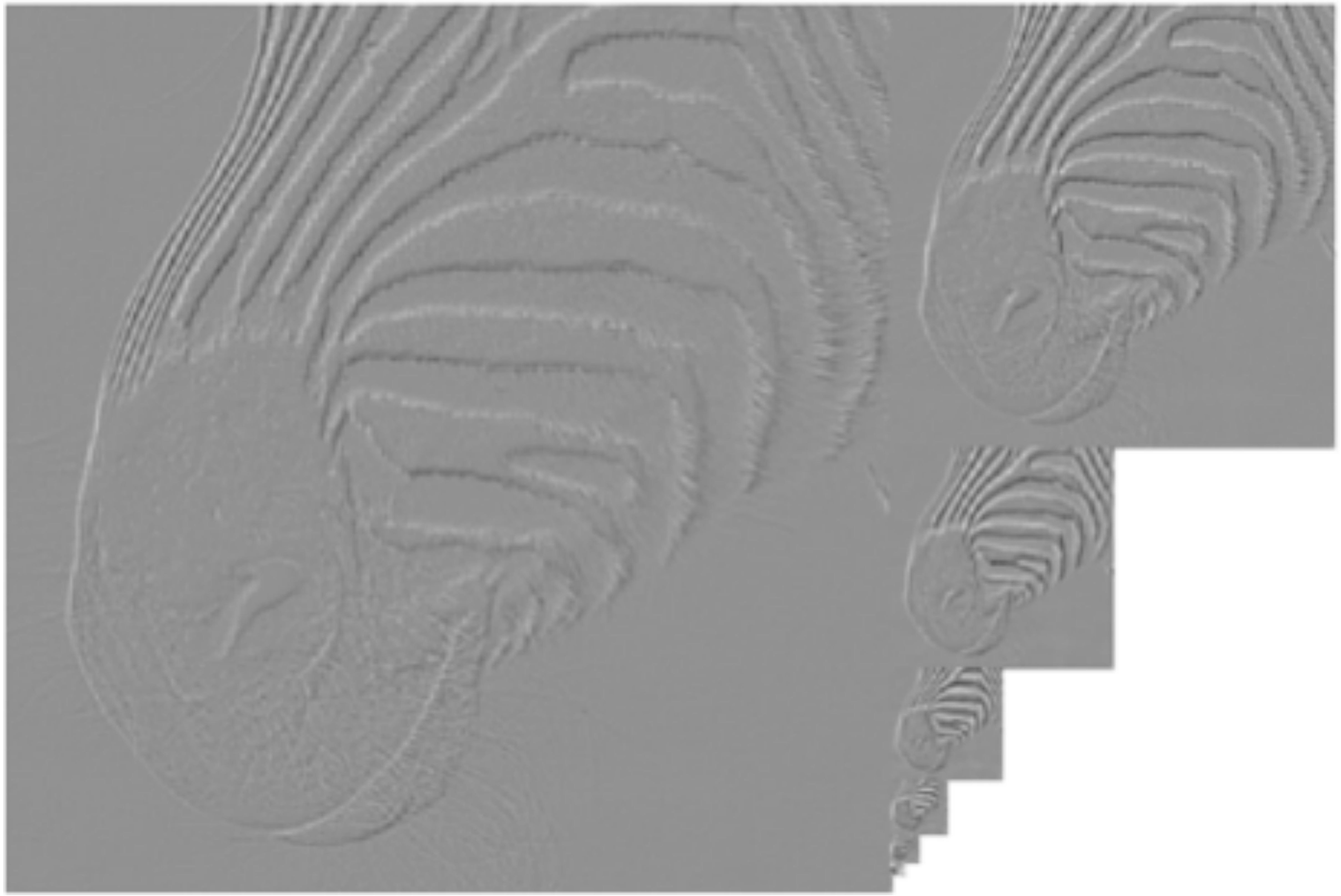
Why is it called Laplacian Pyramid?

Can we reconstruct the original image using the pyramid?
— Yes we can!

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Laplacian Pyramid





At each level, retain the residuals instead of the blurred images themselves.

Why is it called Laplacian Pyramid?

Can we reconstruct the original image using the pyramid?
— Yes we can!

What do we need to store to be able to reconstruct the original image?

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Let's start by just looking at **one level**

=

+

level 0

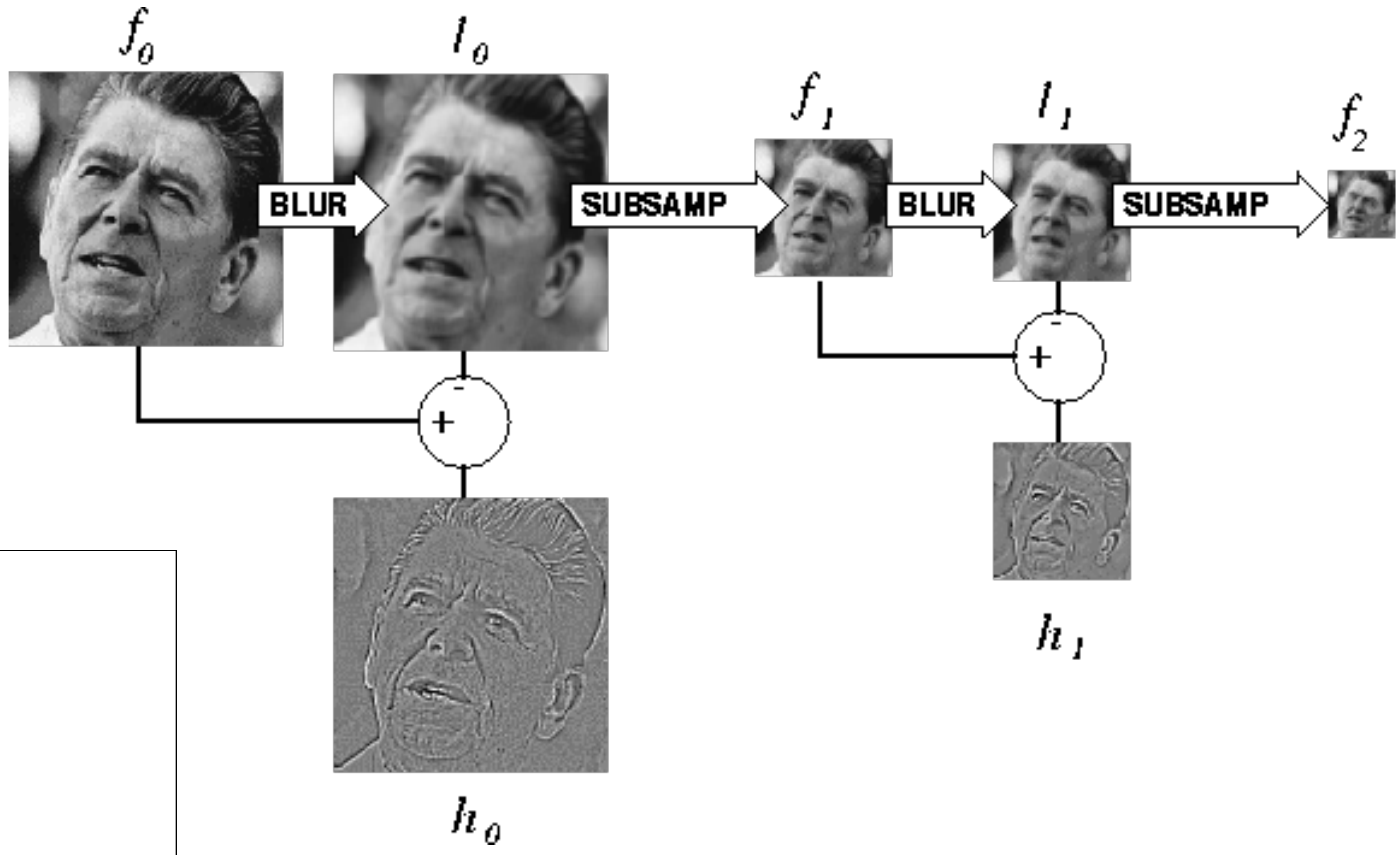
level 1 (upsampled)

residual

Does this mean we need to store both residuals and the blurred copies of the original?

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Constructing a **Laplacian** Pyramid

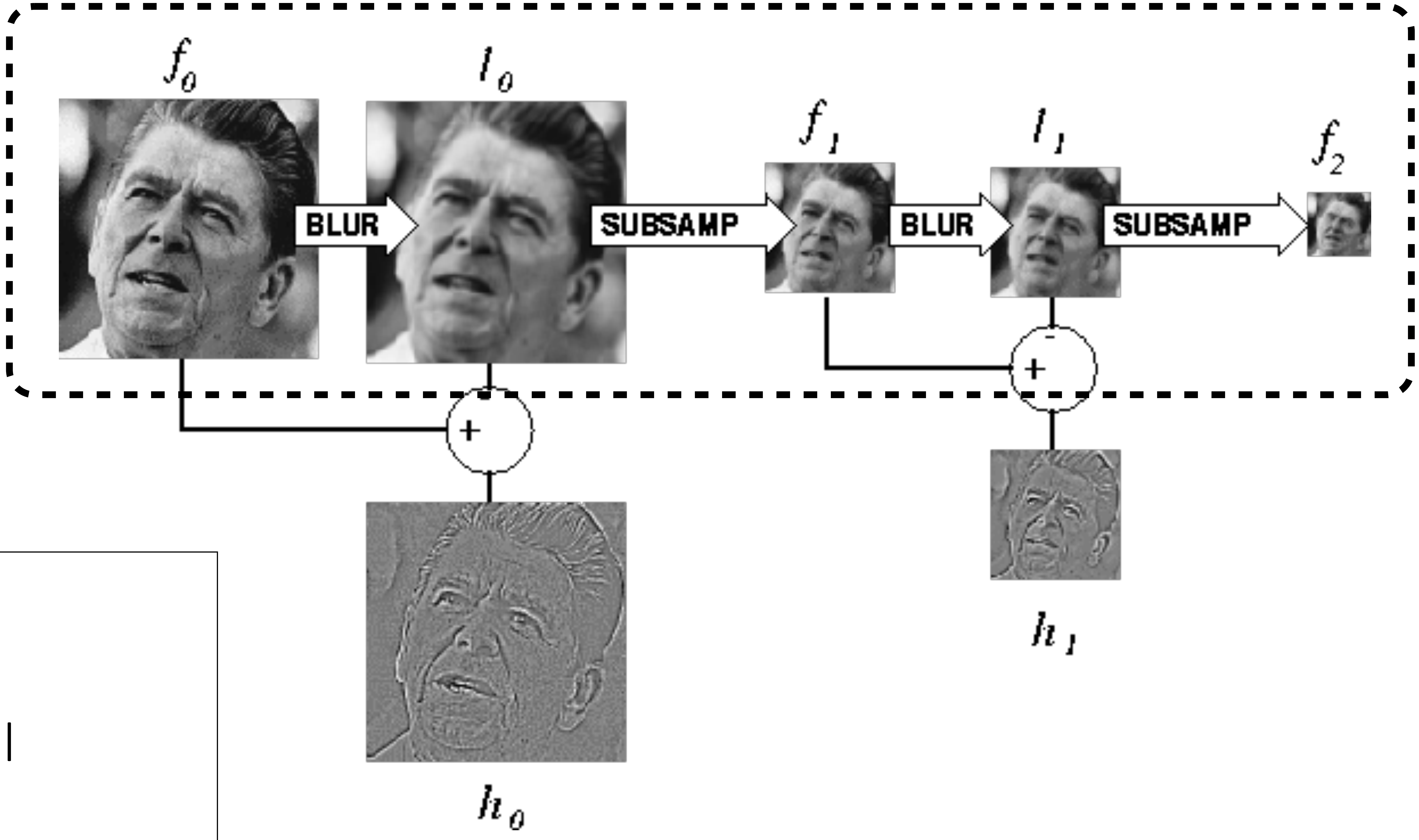


Algorithm

repeat:
    filter
    compute residual
    subsample
until min resolution reached

Constructing a **Laplacian** Pyramid

What is this part?



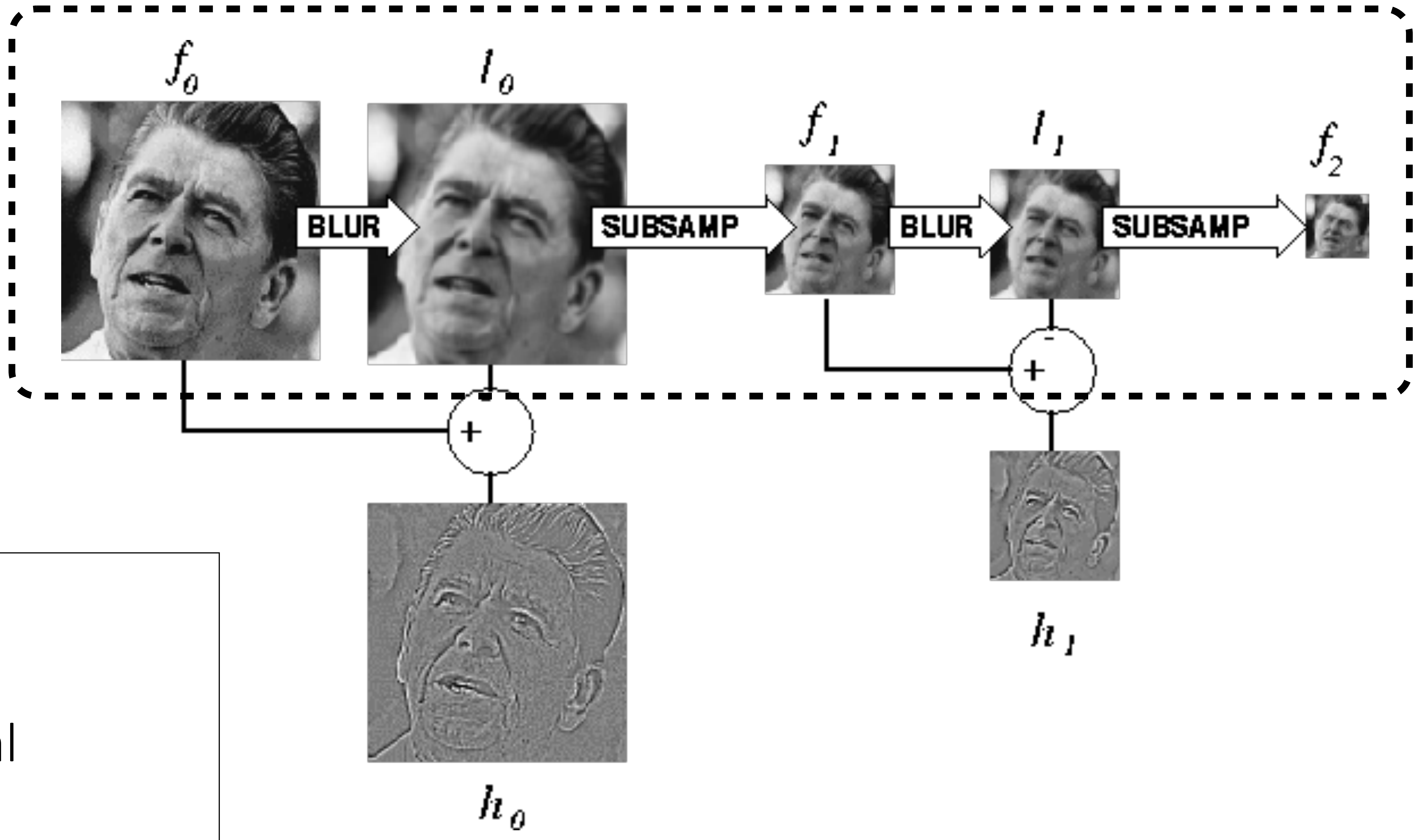
Algorithm

repeat:
  filter
  compute residual
  subsample
until min resolution reached

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Constructing a **Laplacian** Pyramid

It's a Gaussian Pyramid

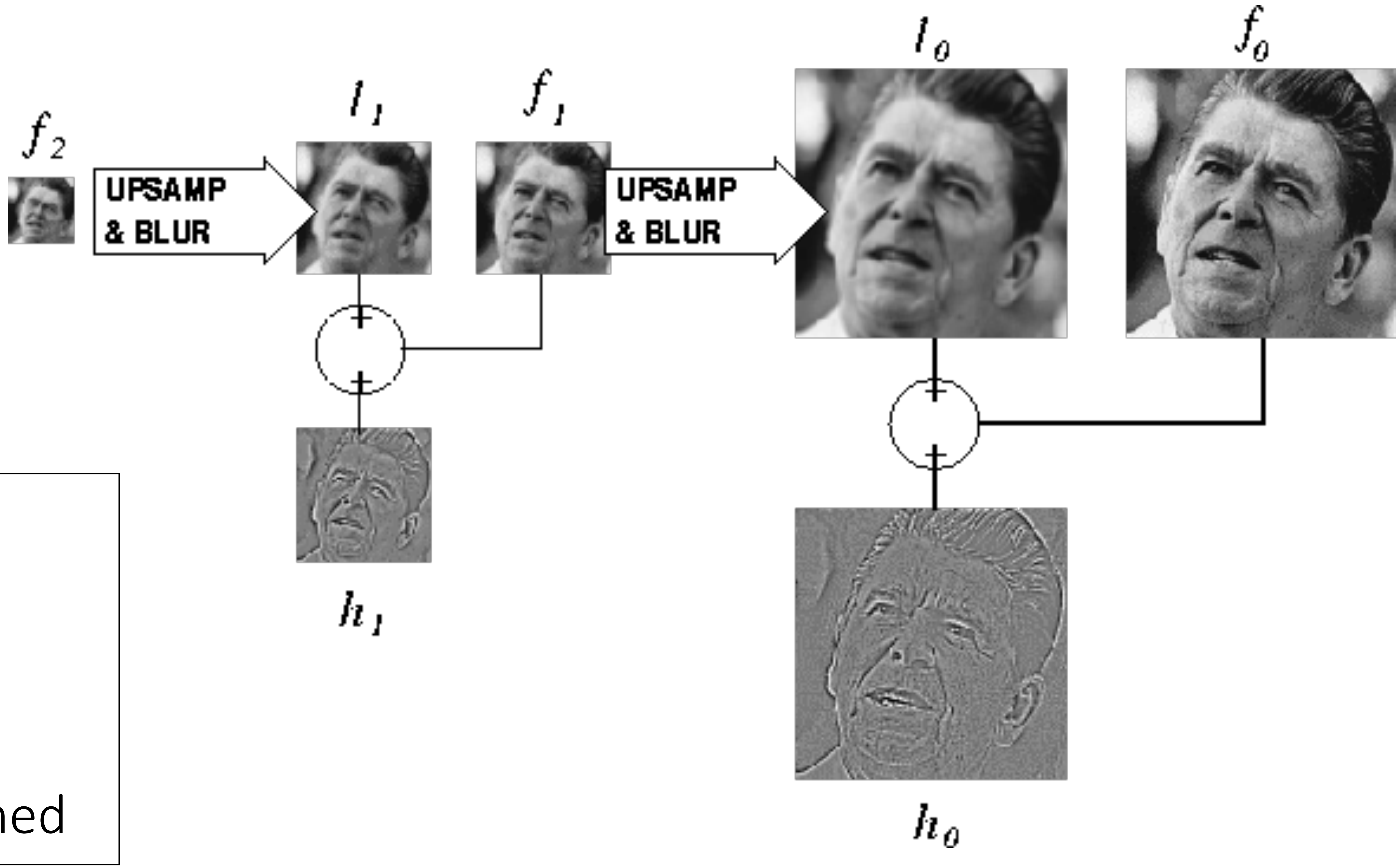


Algorithm

repeat:
  filter
  compute residual
  subsample
until min resolution reached

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Reconstructing the Original Image



Algorithm

repeat:

upsample

sum with residual

until orig resolution reached

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

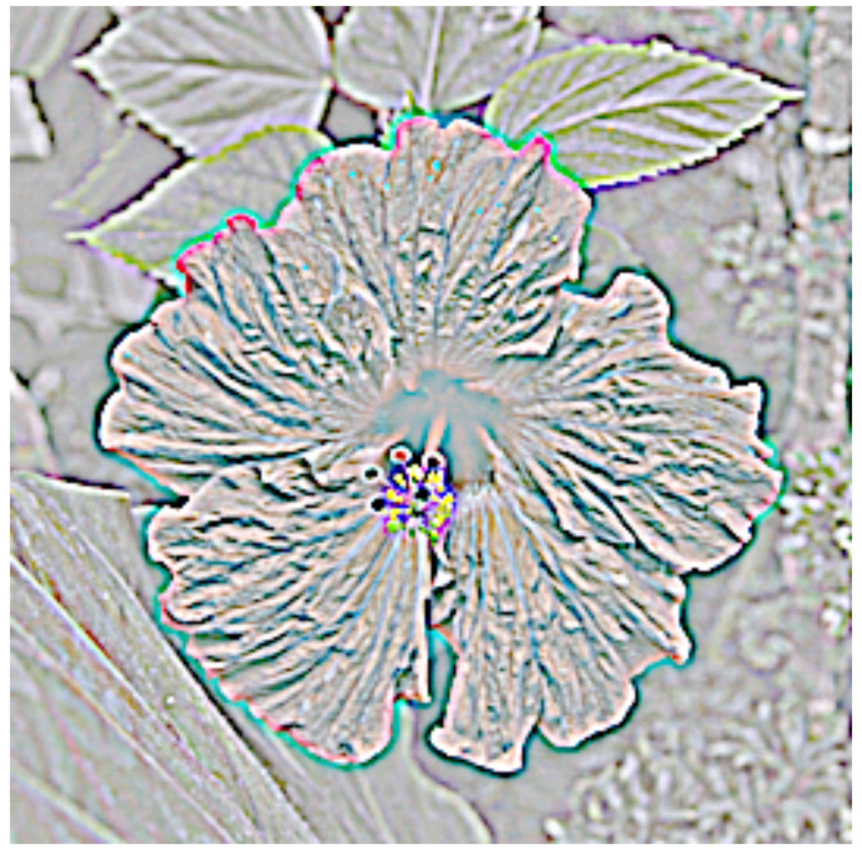
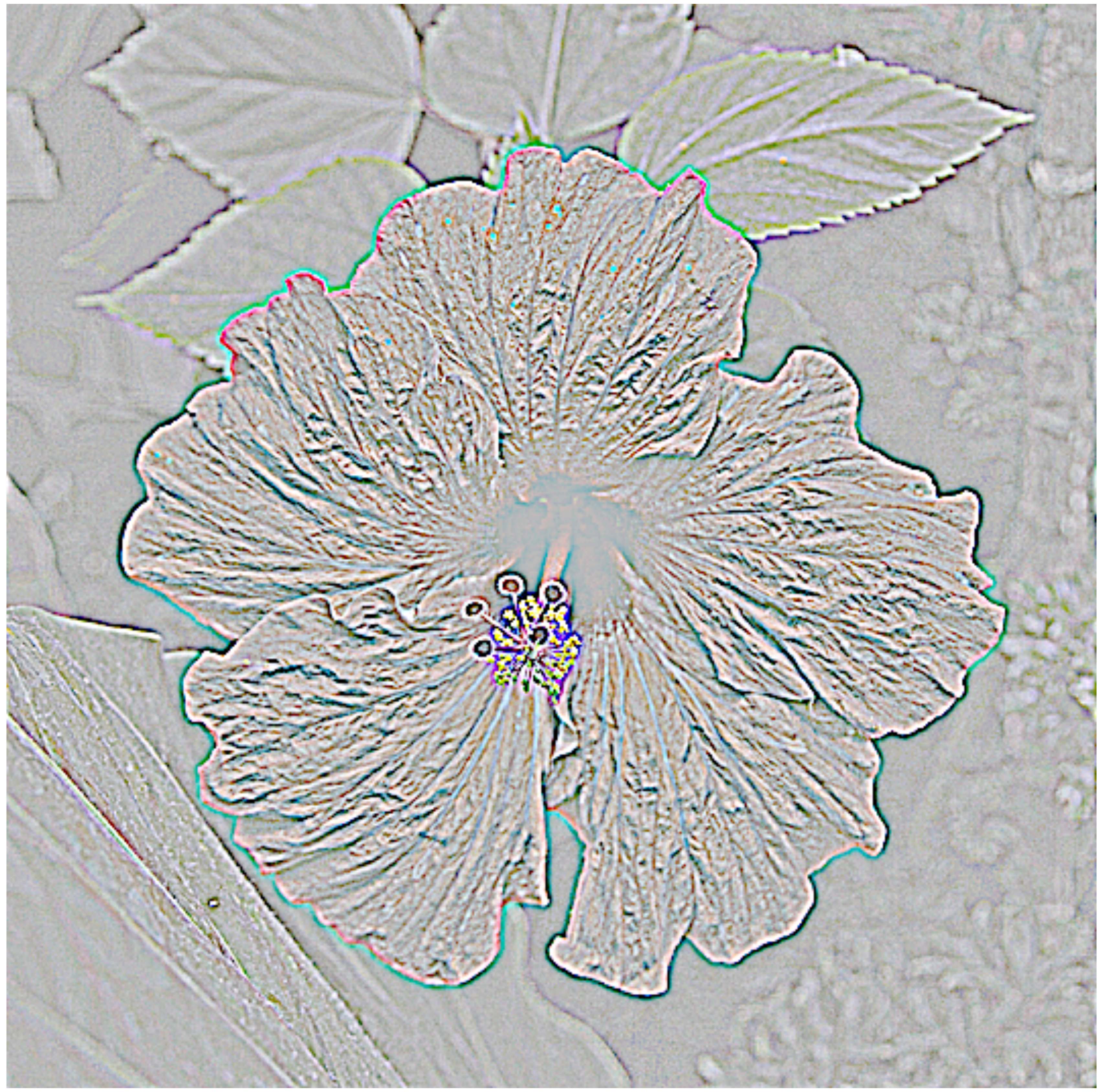
Gaussian vs **Laplacian** Pyramid




Shown in opposite order for space

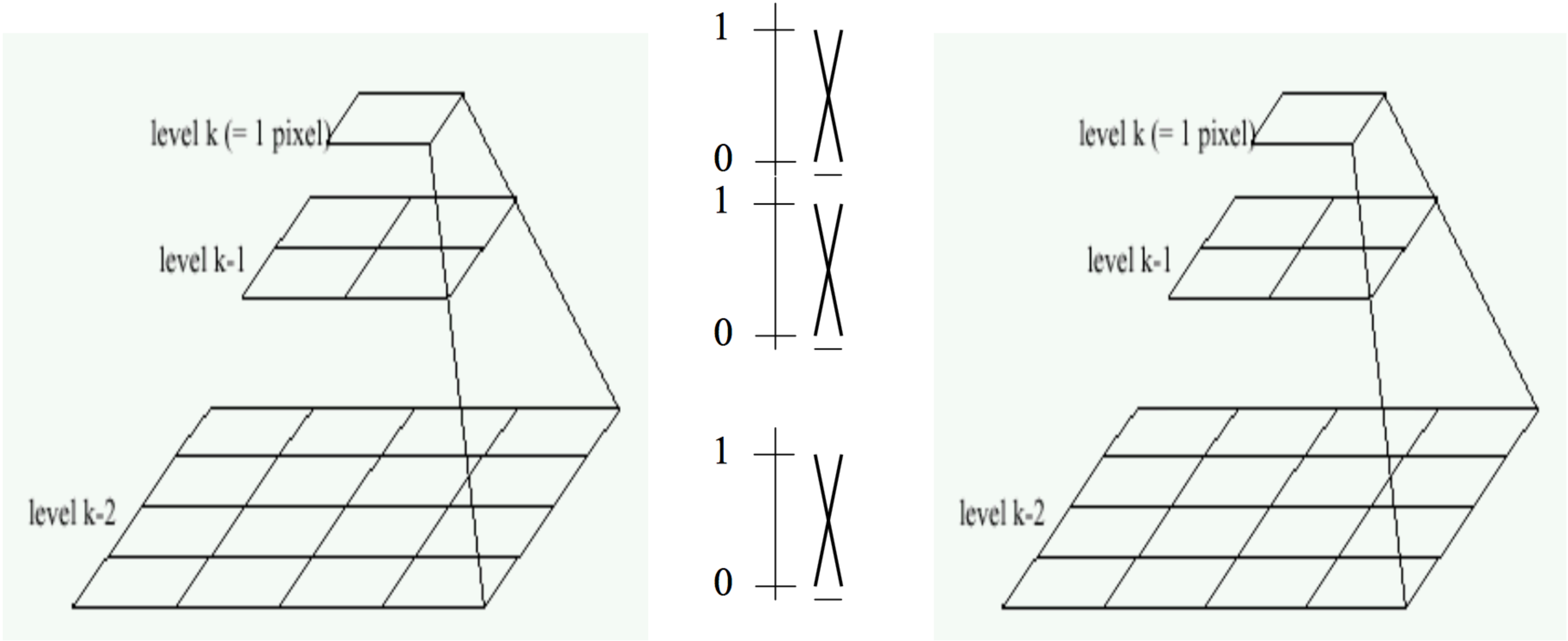
Which one takes more space to store?



Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

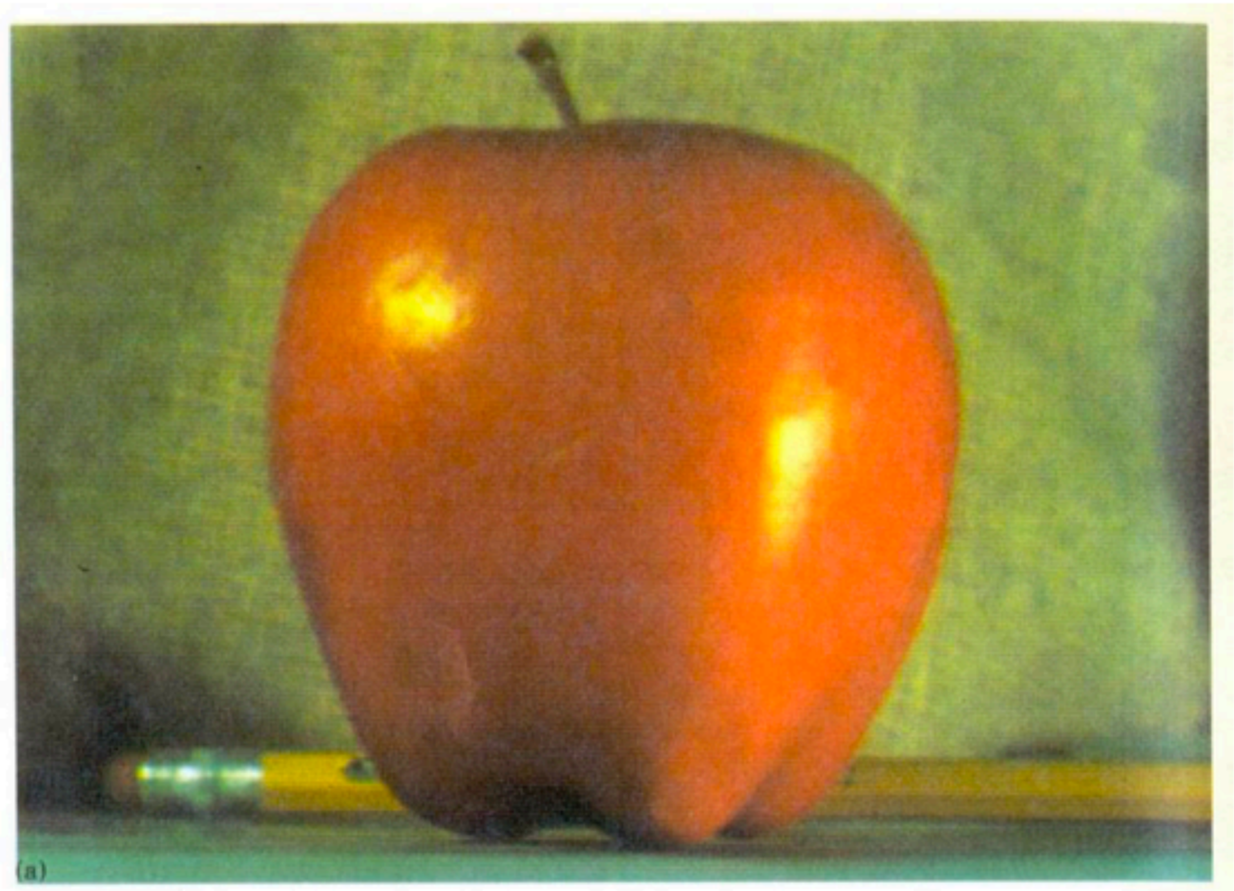
Aside: Image Blending

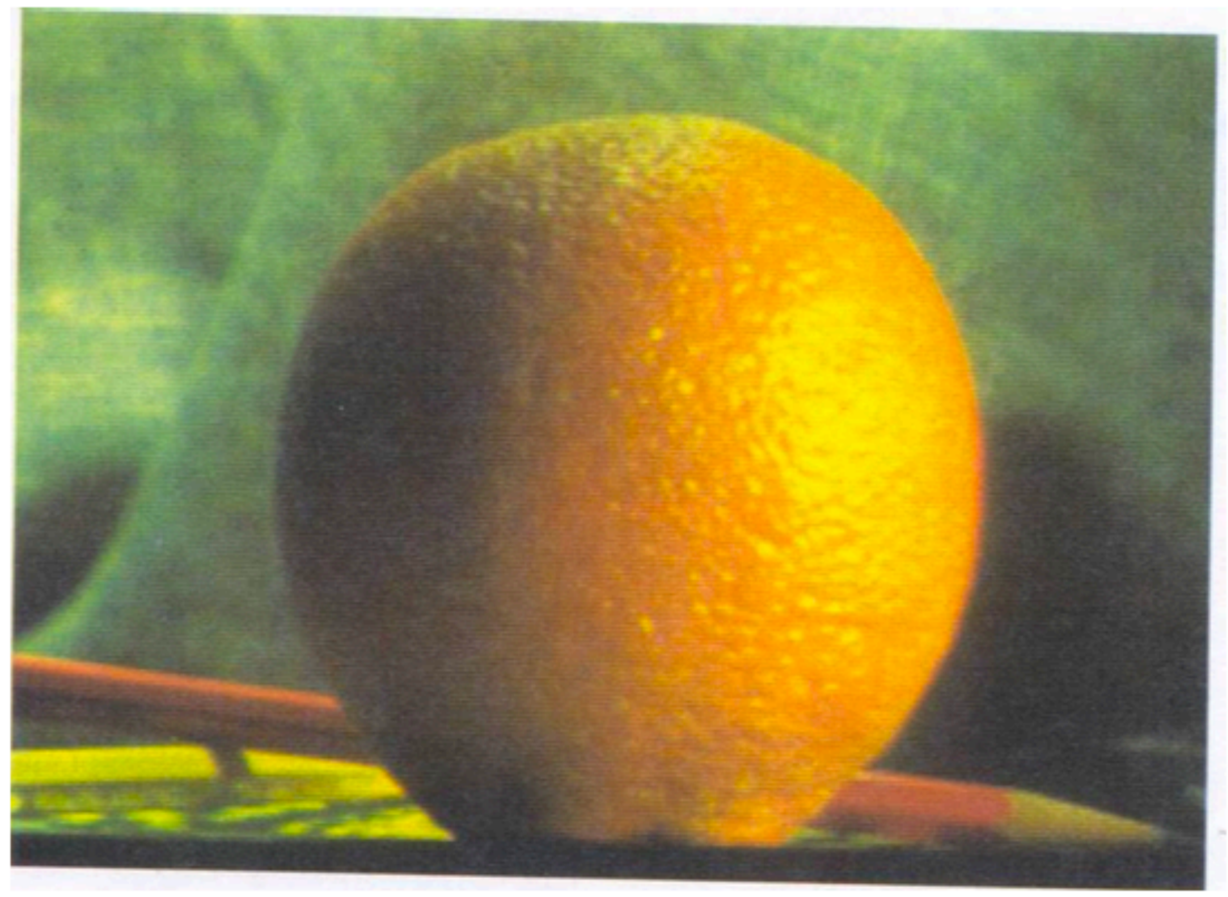


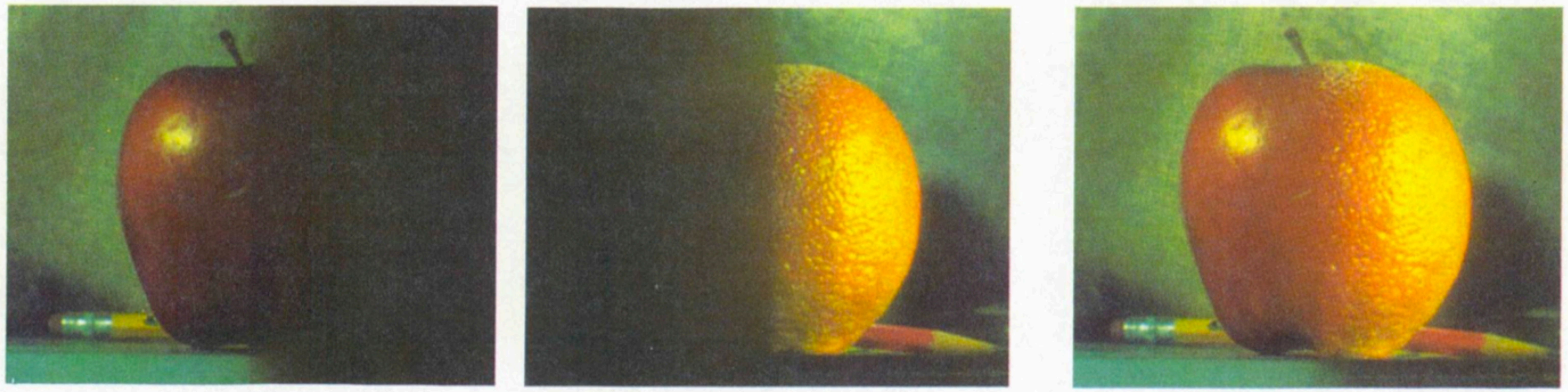
Left pyramid blend Right pyramid

Burt and Adelson, “A multiresolution spline with application to image mosaics,”ACM Transactions on Graphics, 1983, Vol.2, pp.217-236.

Aside: Image Blending







(d) (h) (l)

Burt and Adelson, “A multiresolution spline with application to image mosaics,”ACM Transactions on Graphics, 1983, Vol.2, pp.217-236.

Aside: Image Blending

Algorithm:

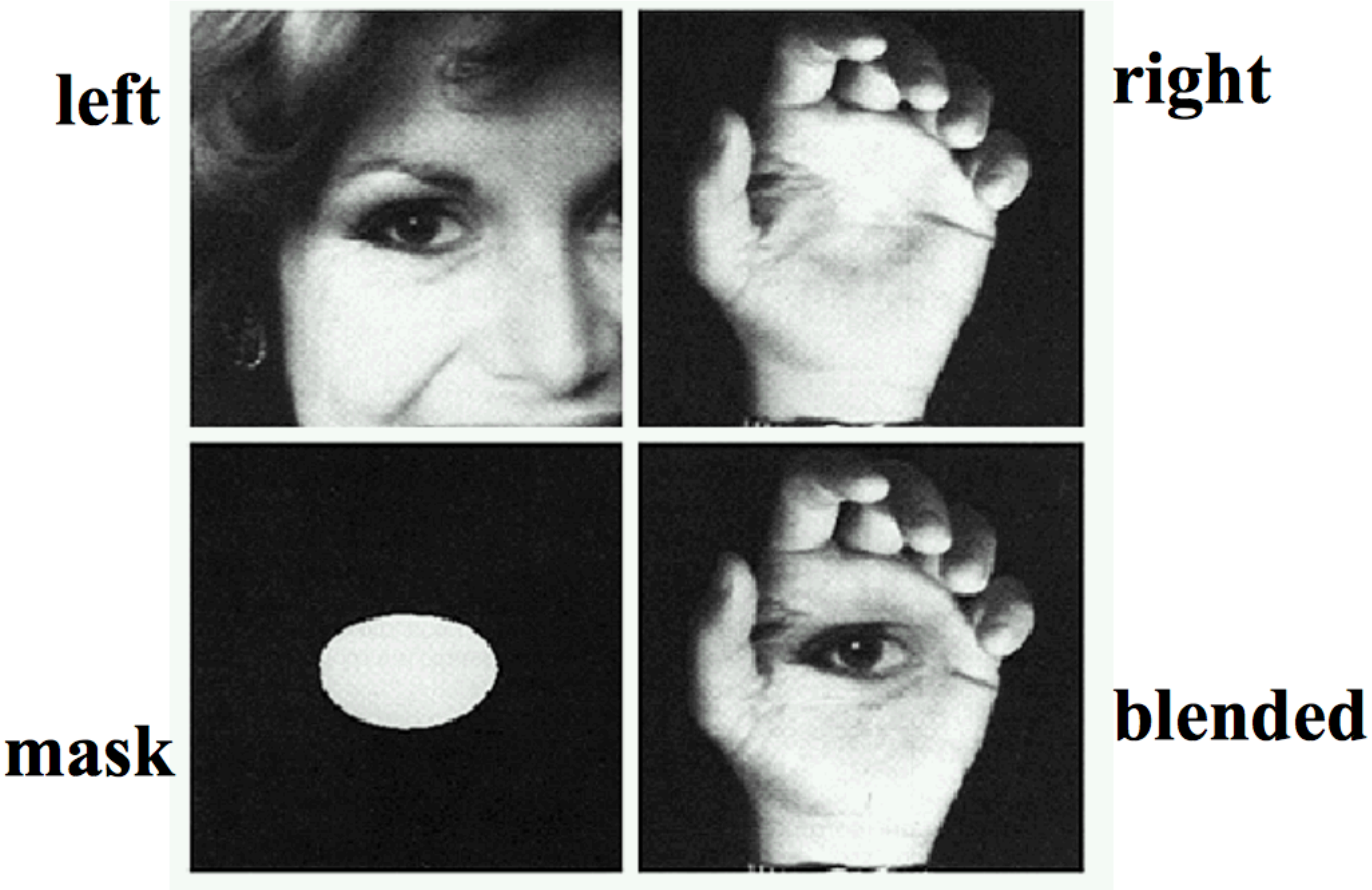
1. Build Laplacian pyramid LA and LB from images A and B

2. Build a Gaussian pyramid GR from mask image R (the mask defines which image pixels should be coming from A or B)

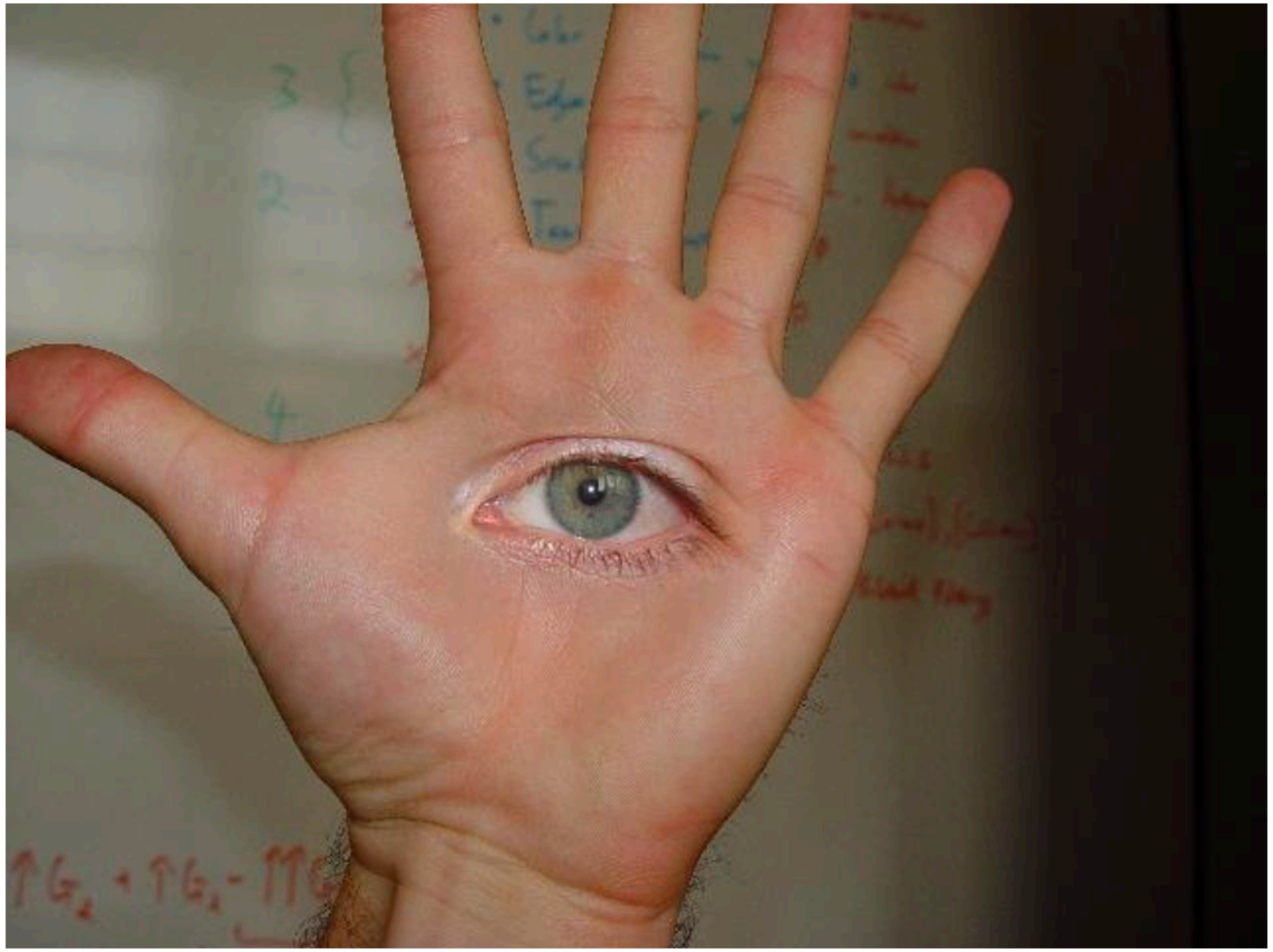
3. From a combined (blended) Laplacian pyramid LS, using nodes of GR as weights: $LS(i,j) = GR(i,j) * LA(i,j) + (1-GR(i,j)) * LB(i,j)$

4. Reconstruct the final blended image from LS

Aside: Image Blending



Aside: Image Blending



© david dmartin (Boston College)

Aside: Image Blending

© Chris Cameron