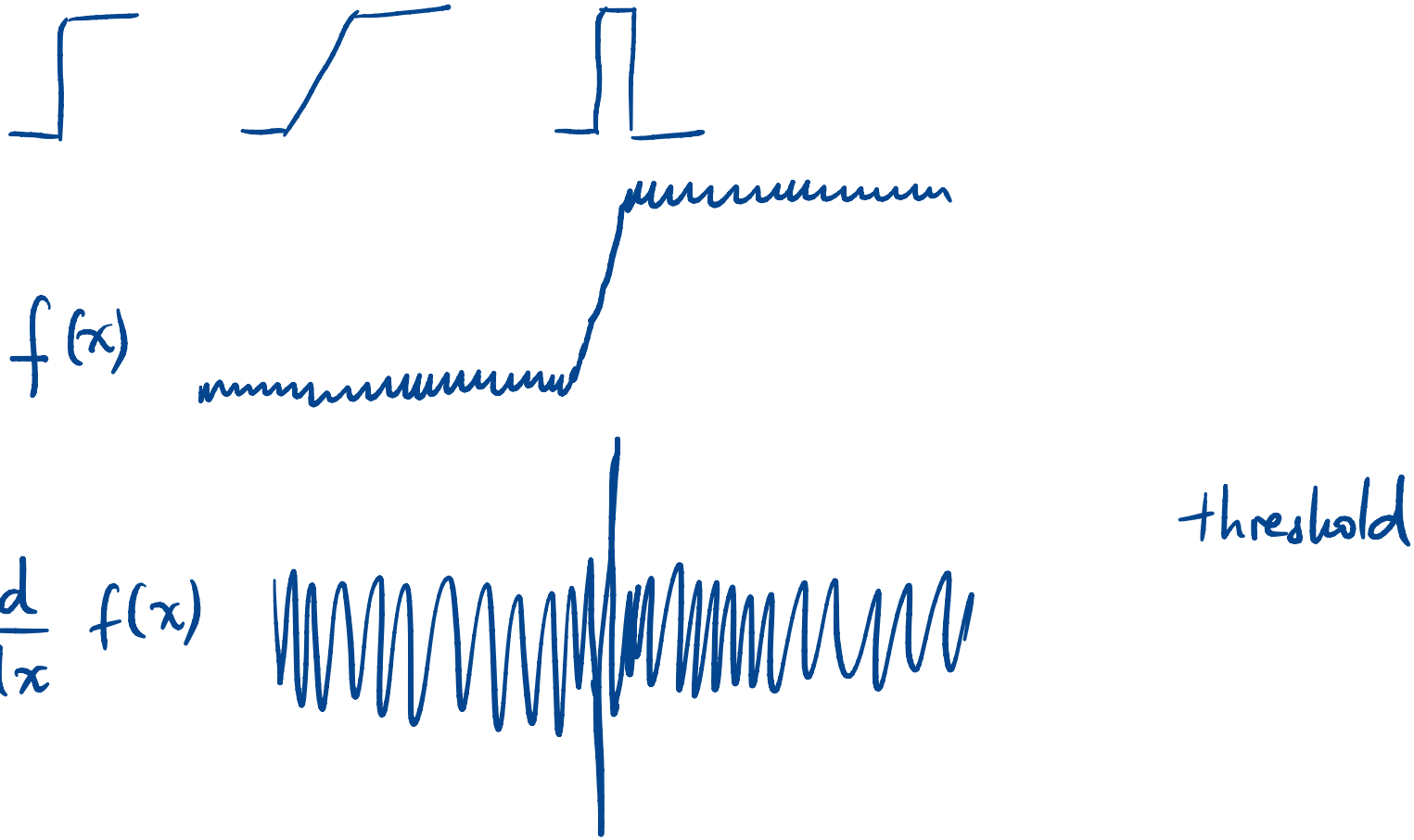

University of Colorado
Boulder

Deep Learning Applications for Computer Vision

Lecture 8: An Algorithm for Edge Detection

Effects of noise

Plotting intensity as a function of position – it's a signal!



Example

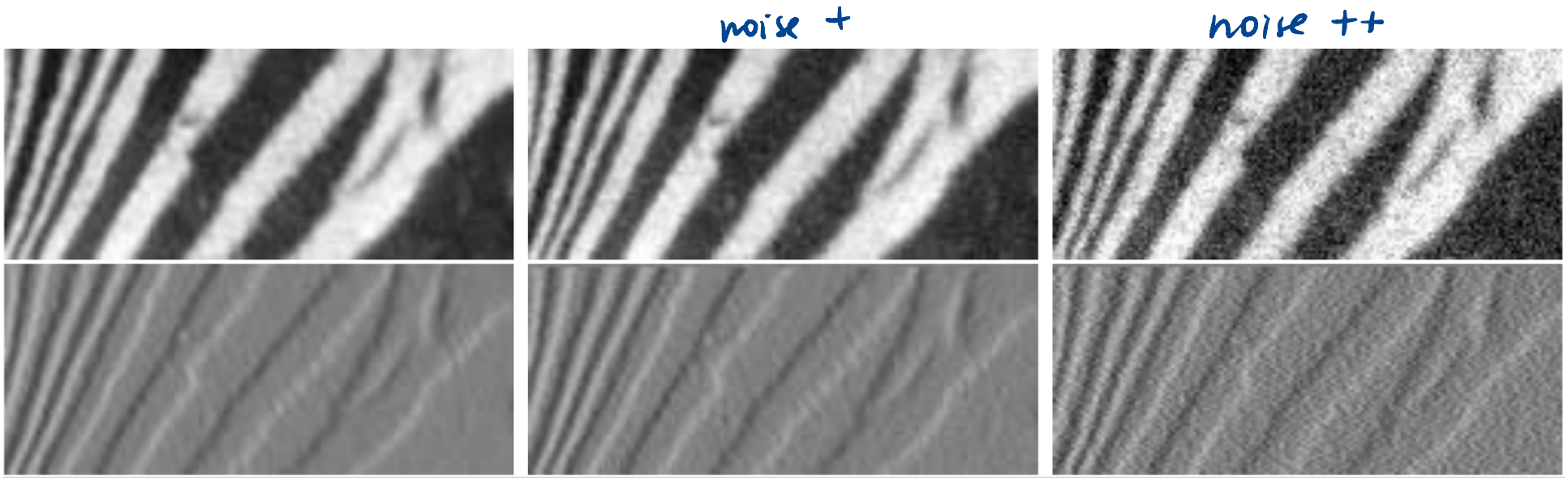


Figure 4.4 – Chapter 4, Forsyth & Ponce, *Computer Vision – A Modern Approach*

Adding noise to images, then applying derivatives:

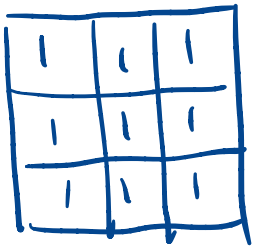
- the noise is amplified by the derivatives
- if a pixel is very different than its neighbor, a derivative filter will respond strongly to that difference

Solution: smooth first

Smoothing will reduce noise.

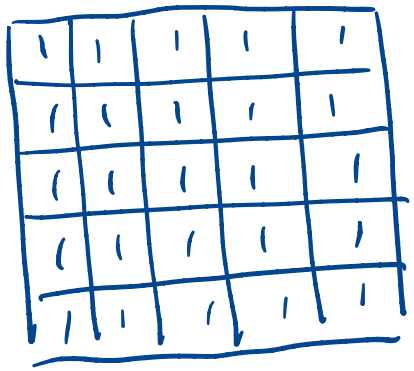
Important parameter: filter size/scale

- more pixels involved – more blurring



3 x 3

$\frac{1}{9}$



5 x 5

$\frac{1}{25}$

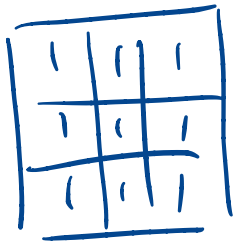
Solution: smooth first

Smoothing will reduce noise.

Important parameter: filter size/scale

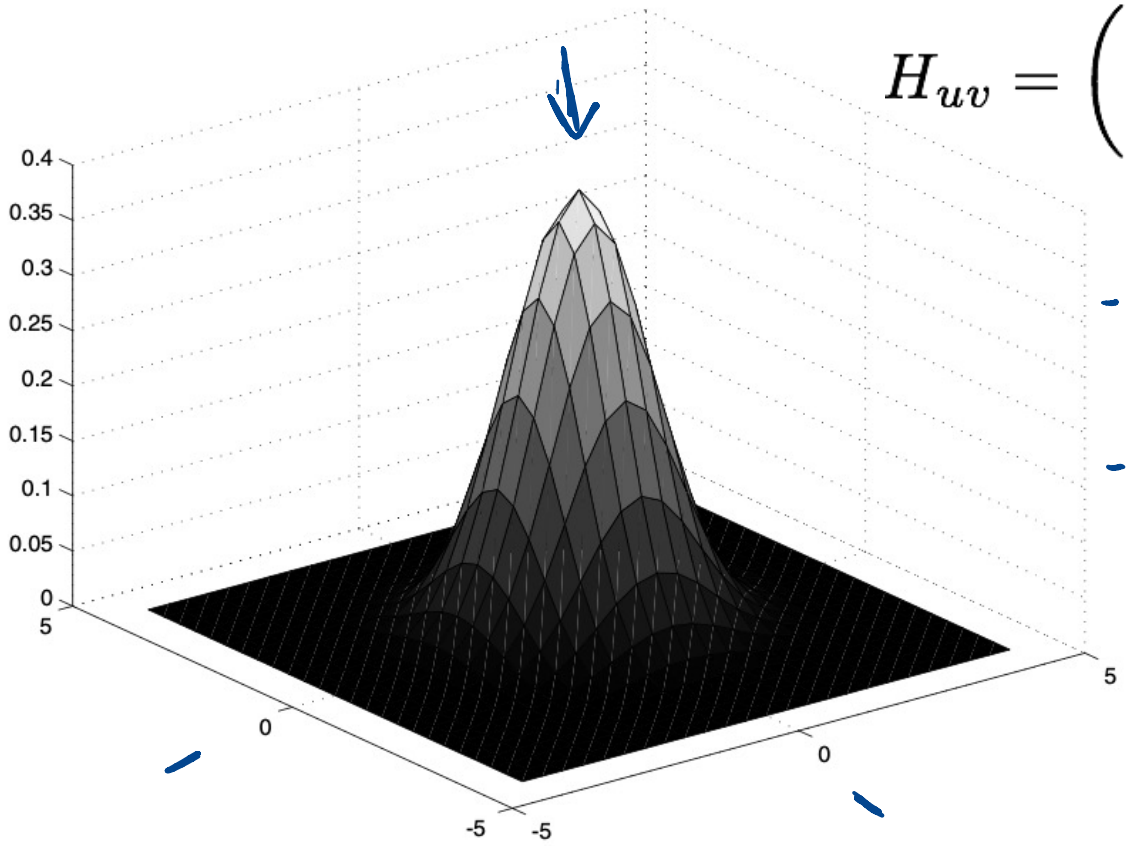
- more pixels involved – more blurring

Option A: Use a box filter (averaging)



Solution: smooth first

Option B: Use a Gaussian filter



$$H_{uv} = \left(\frac{1}{2\pi\sigma^2}\right) \exp\left(\frac{-\left[u^2+v^2\right]}{2\sigma^2}\right)$$

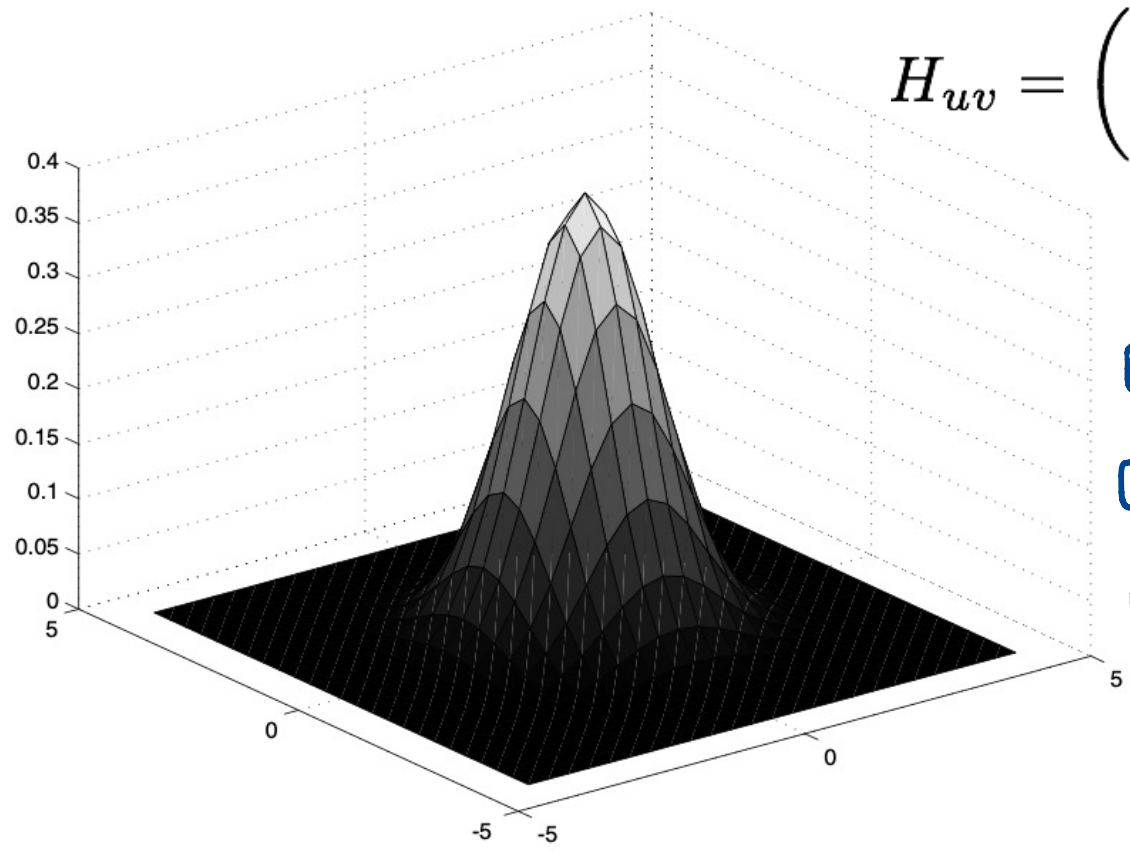
- strong response in the center
- lesser contributions from distant neighbors

Figure 4.2 The symmetric Gaussian kernel in 2D

Solution: smooth first

Option B: Use a Gaussian filter

$$H_{uv} = \left(\frac{1}{2\pi\sigma^2}\right) \exp\left(\frac{-\left[u^2+v^2\right]}{2\sigma^2}\right)$$



σ – std. deviation

σ small

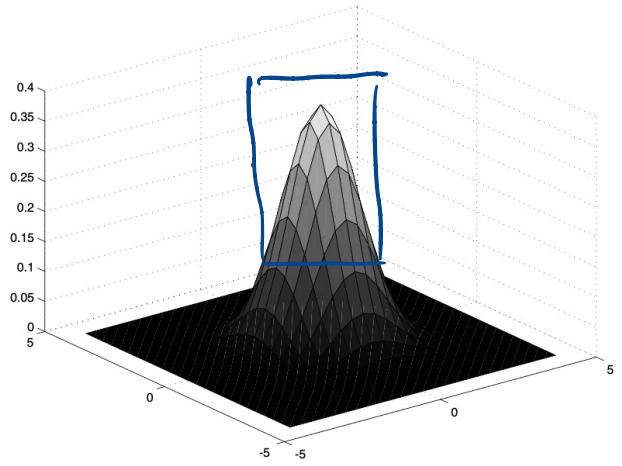
σ large

σ v. large

Figure 4.2 The symmetric Gaussian kernel in 2D

A Discrete Gaussian Kernel

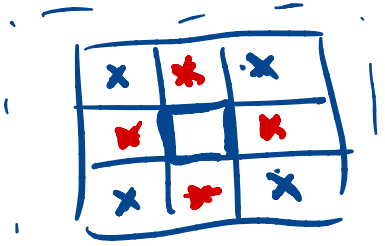
Option B: Use a Gaussian filter



$$H_{ij} = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{((i-k-1)^2 + (j-k-1)^2)}{2\sigma^2}\right)$$

$(2k+1)$ by $(2k+1)$ array

$k=1 \Rightarrow$ 3×3 kernel



$i=1 \quad i=3$

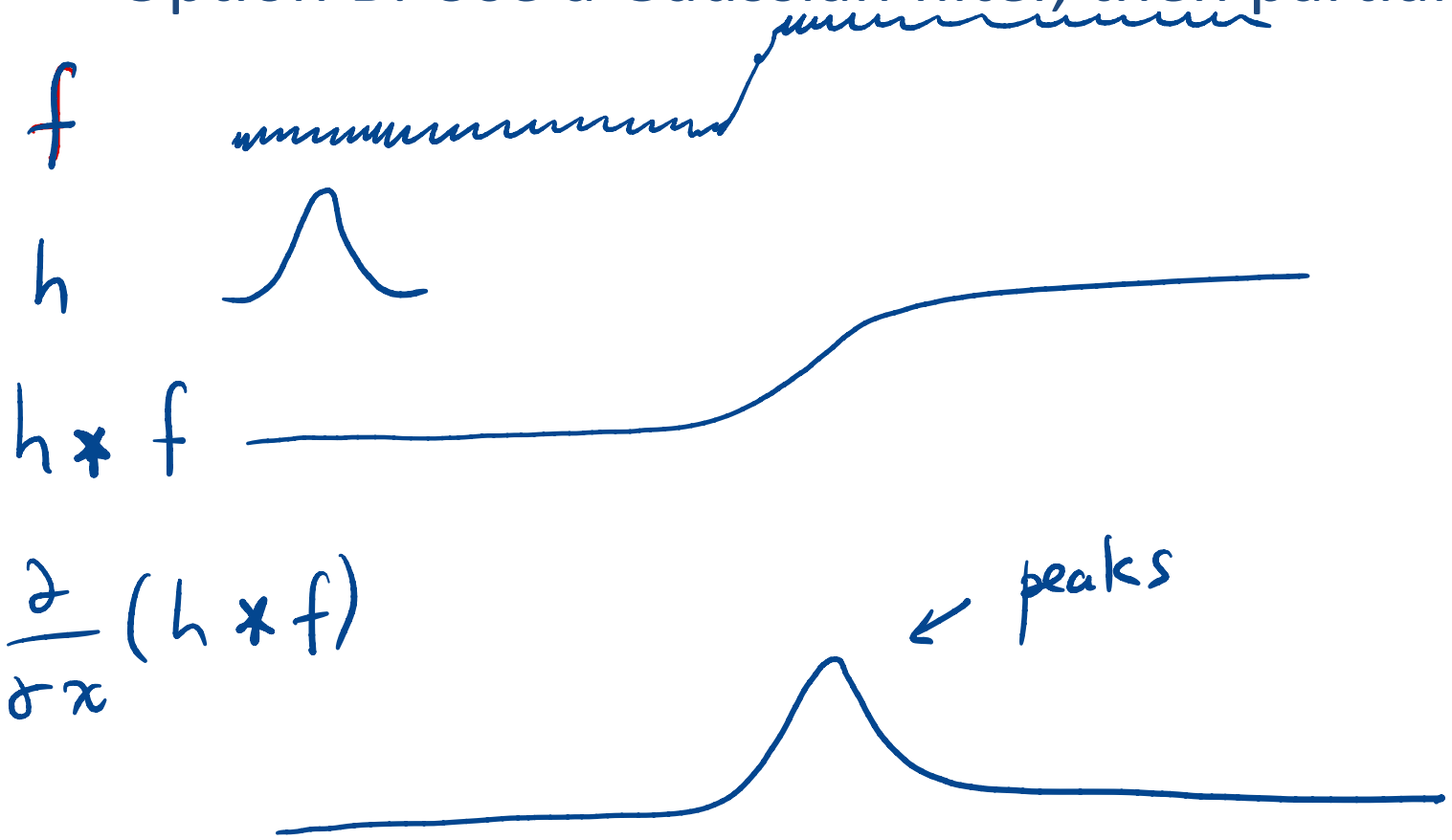
$j=1 \quad j=3$

Figure 4.2 The symmetric Gaussian kernel in 2D

Gaussian kernel:

Solution: smooth first

Option B: Use a Gaussian filter, then partial derivatives



Option C: Convolve with the derivative of Gaussian filter first

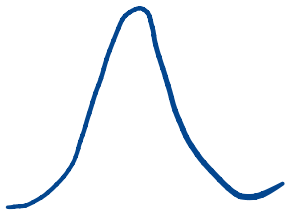
$$\frac{\partial (G_\sigma * * I)}{\partial x} = (\frac{\partial G_\sigma}{\partial x}) * * I$$

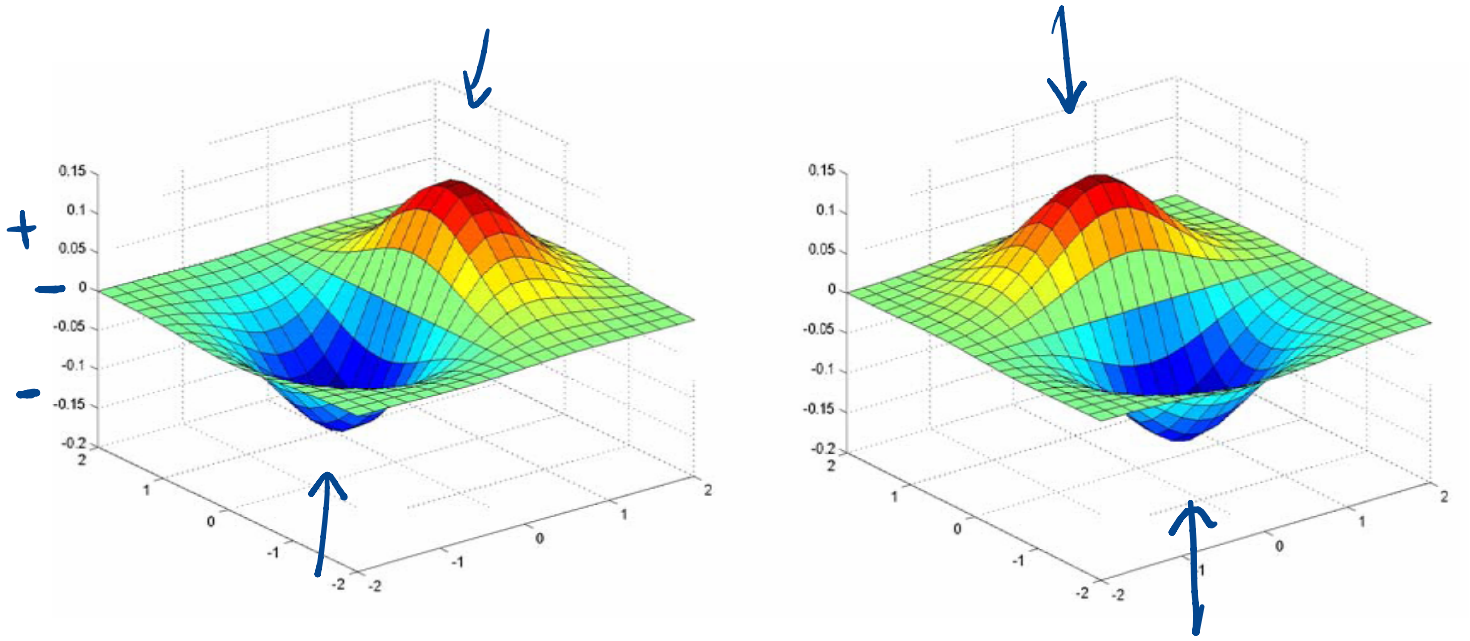
1) Apply Gaussian filter
2) Partial derivatives

1) Partial derivatives of Gaussian filter
2) Convolve the image with the result

Option C: Convolve with the derivative of Gaussian filter first

$$\frac{\partial \left(G_\sigma * * I\right)}{\partial x} = \left(\frac{\partial G_\sigma}{\partial x}\right) * * I$$





Option C: Convolve with the derivative of Gaussian filter

$$\frac{\partial \left(G_\sigma * * I\right)}{\partial x} = \left(\frac{\partial G_\sigma}{\partial x}\right) * * I$$

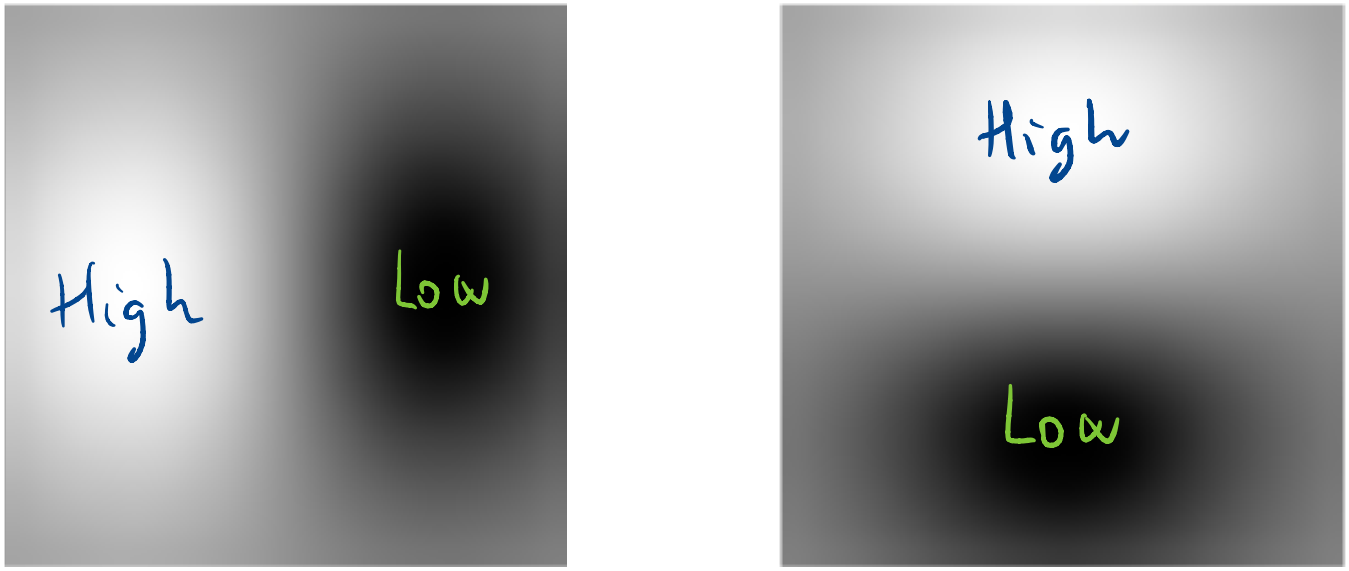
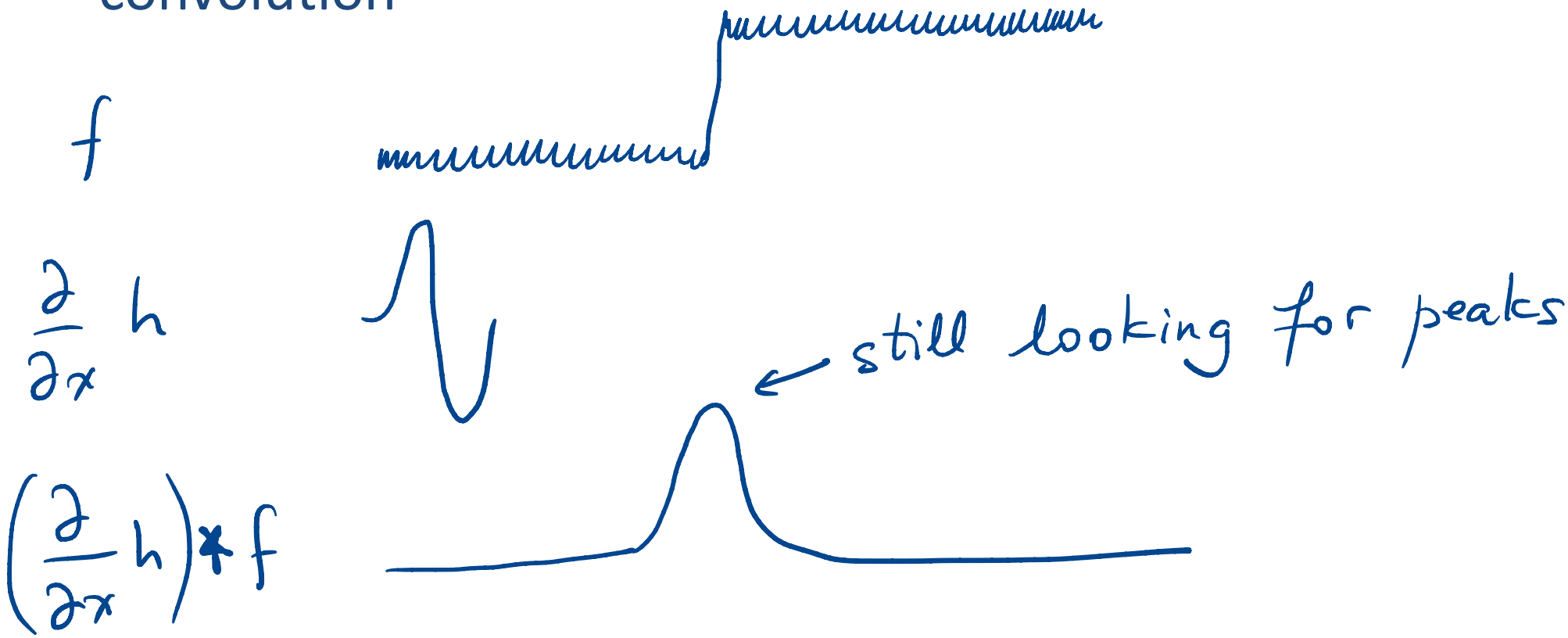


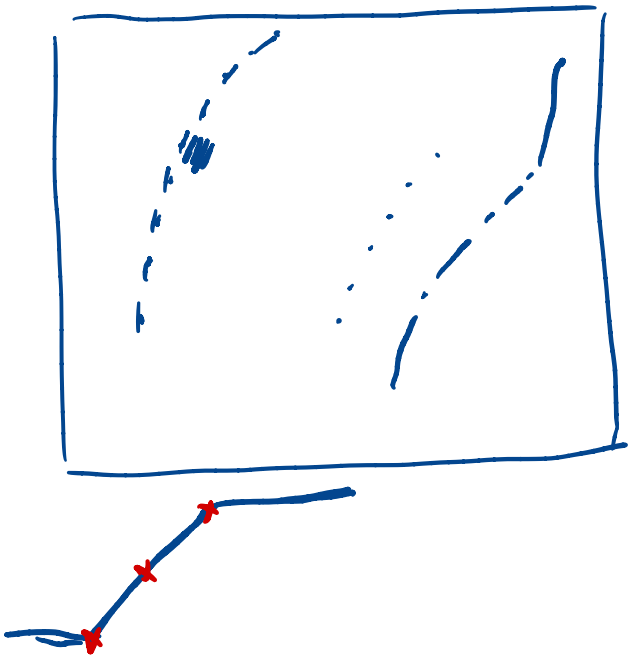
FIGURE 4.15: Filter kernels look like the effects they are intended to detect. On the left, a smoothed derivative of Gaussian filter that looks for large changes in the x-direction (such as a dark blob next to a light blob); on the right, a smoothed derivative of Gaussian filter that looks for large changes in the y-direction.

Option C: Derivative of Gaussian filter first, then convolution



Gradient-based Edge Detection

1) Some smoothing
2) Some partial derivatives
3) Compute magnitude of gradient — use threshold
4) Thin edges
5) Connected / continuous edges

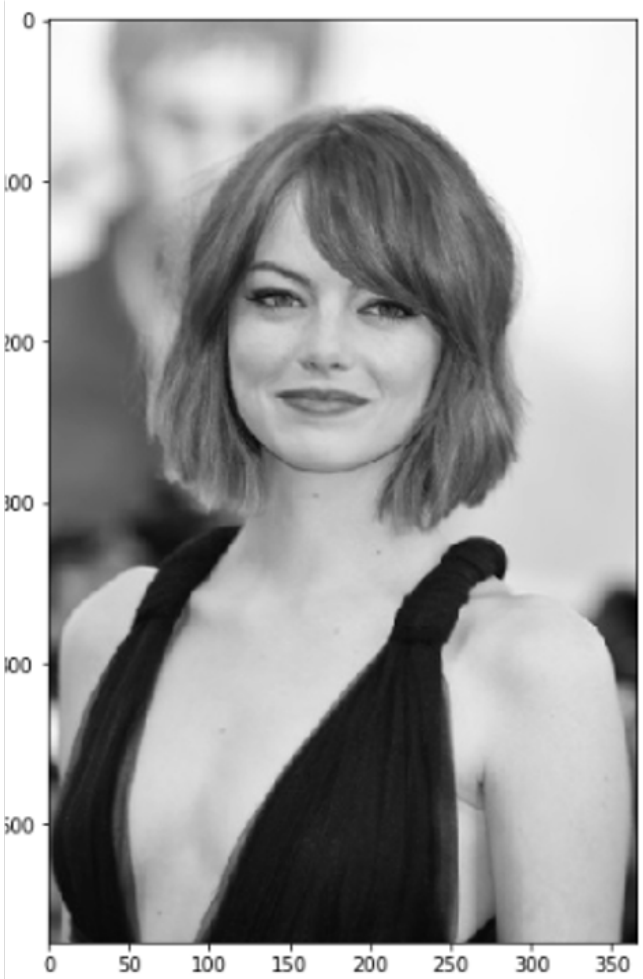


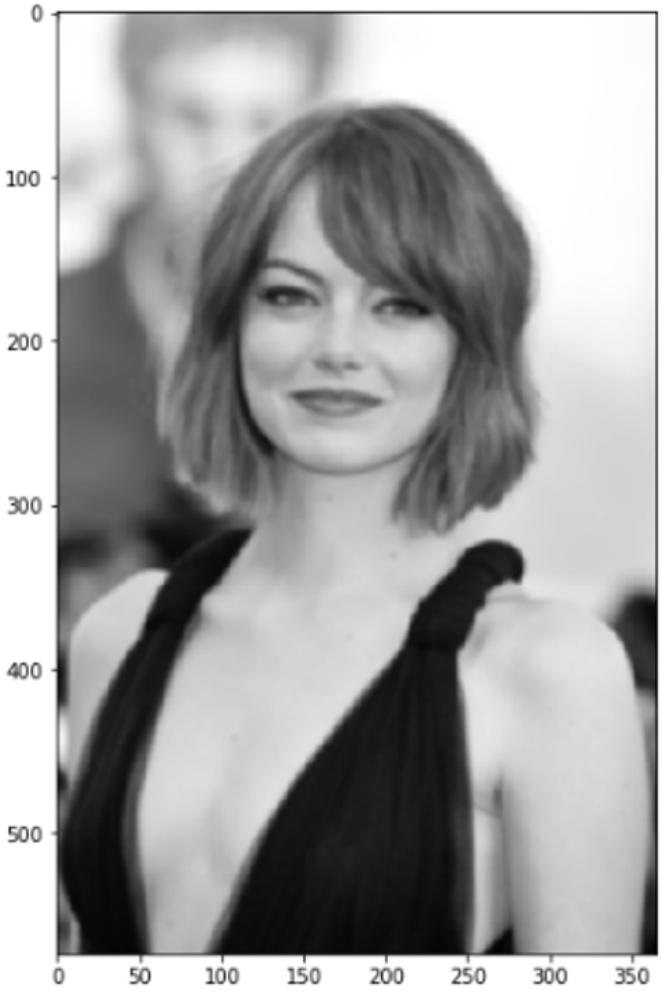
Optimal Edge Detection – Canny, 1986

1. Filter image with x, y derivatives of Gaussian ✓
2. Find magnitude and orientation of gradient ✓
3. *Non-maximum suppression*:
   – Thin multipixel wide "ridges" down to single pixel width
4. *Thresholding* and linking (*hysteresis*):
   – Define two thresholds: low and high
   – Use the high threshold to start edge curves and the low threshold to continue them

Optimal Edge Detection – Canny, 1986

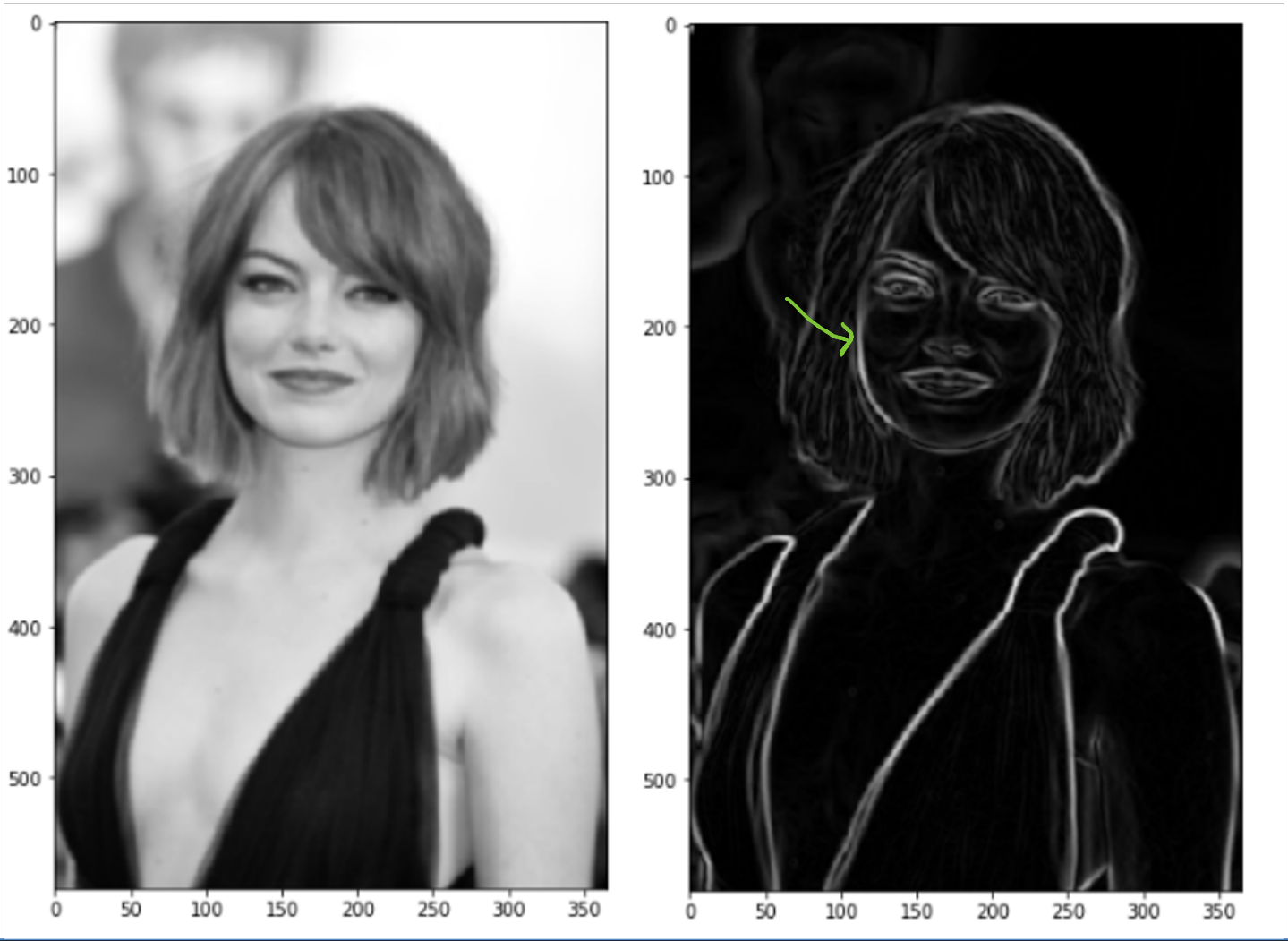
1. Filter image with x, y derivatives of Gaussian





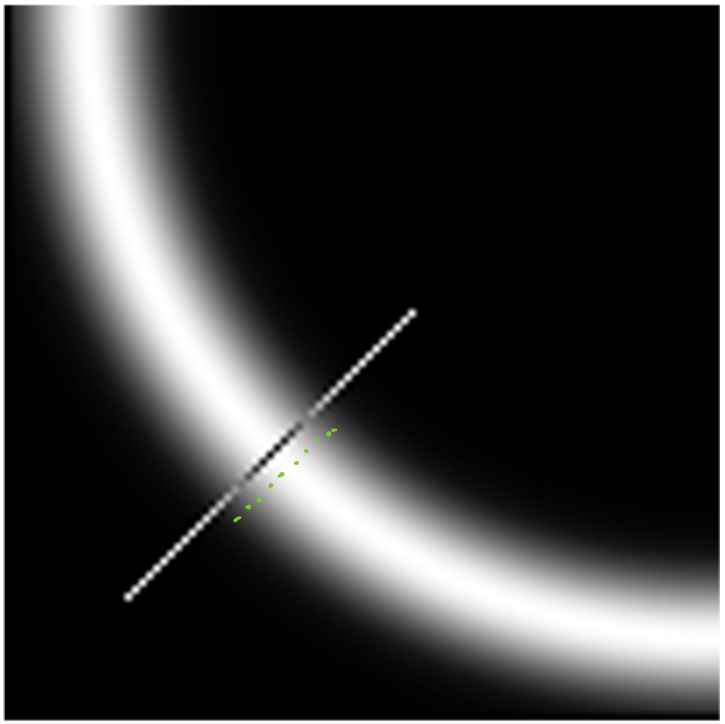
Optimal Edge Detection – Canny 1986

2. Find magnitude and orientation of gradient



Non-maximum suppression

- Check if pixel is local maximum along gradient direction



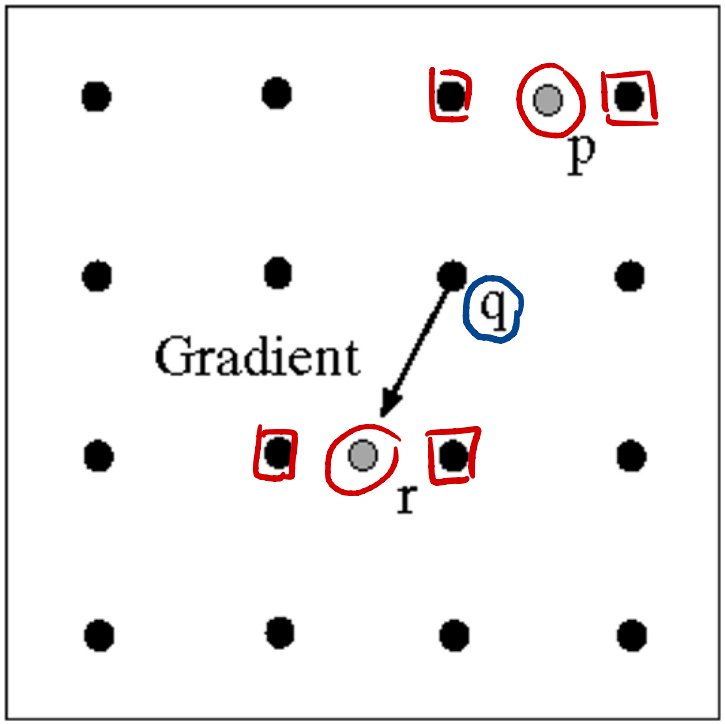
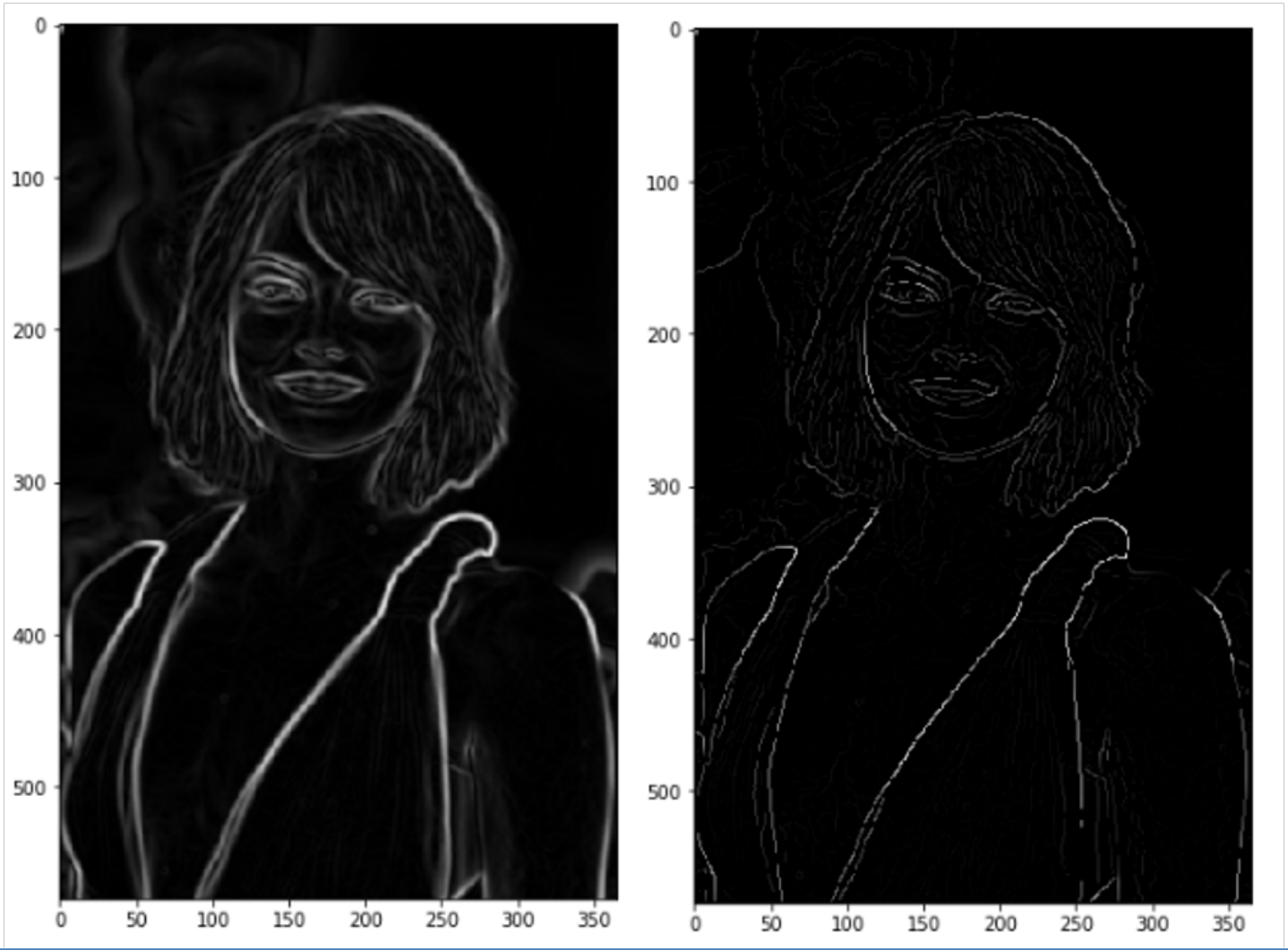


Figure 5.5 (left): Chapter 5, Forsyth and Ponce, *Computer Vision – A Modern Approach*

Non-maximum suppression

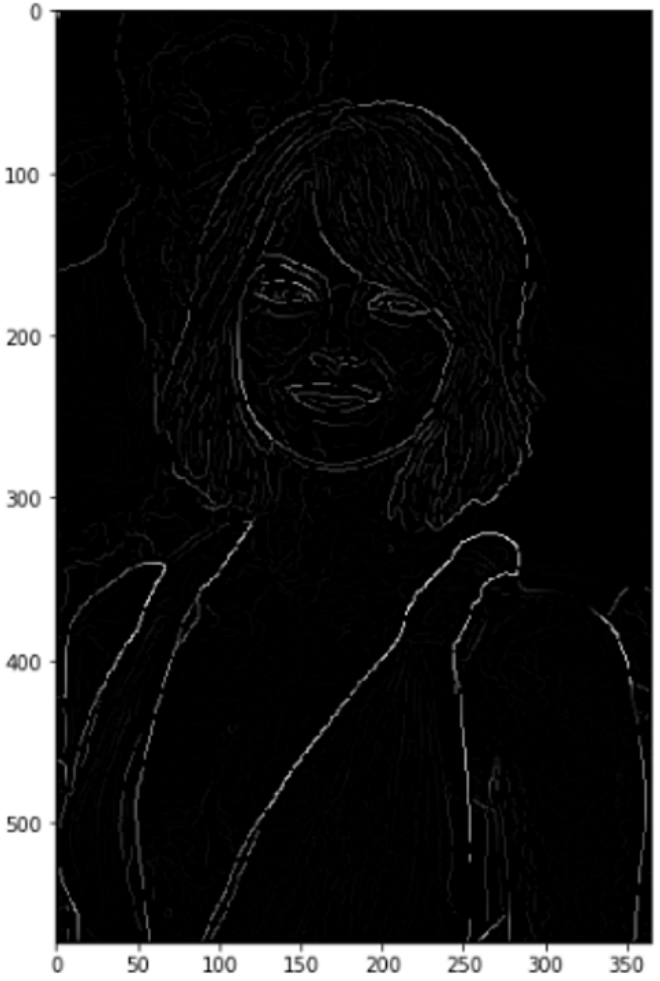
- Check if pixel is local maximum along gradient direction

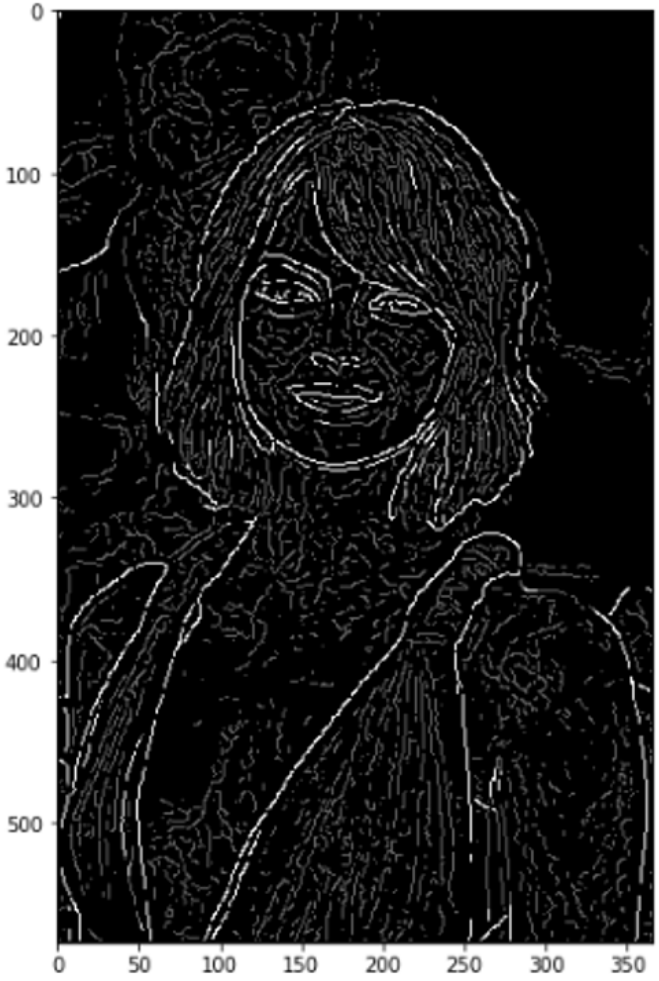


Thresholding

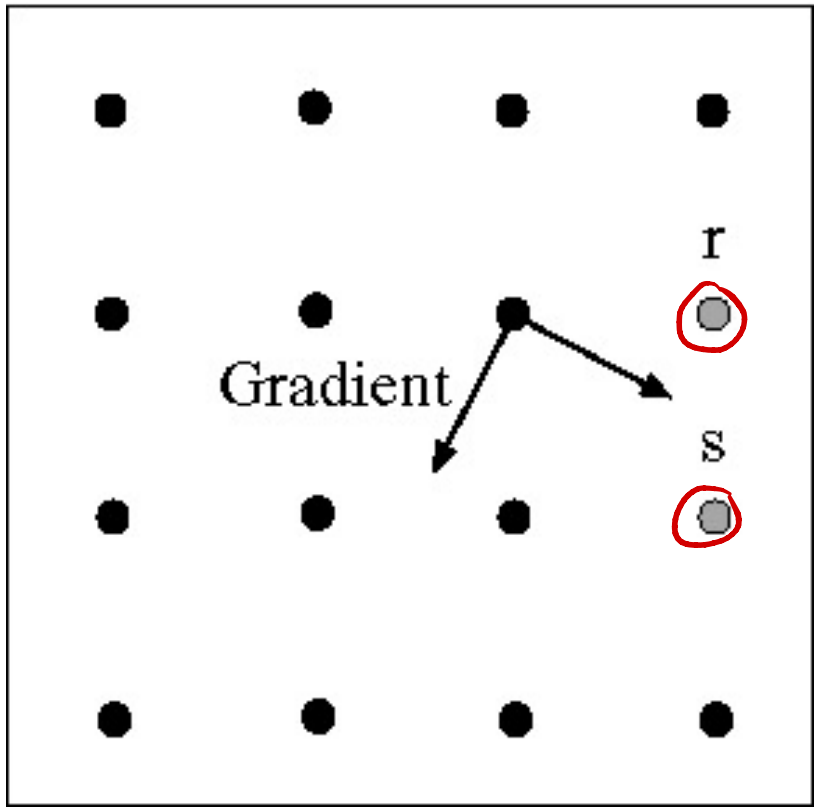
high ↗ strong edge
low ↘ discarded : no edge
I weak edge pixels

- Too low: false contours. Too high: suppressed contours





Hysteresis Thresholding



Predicting the next edge point

Assume the marked point is an edge point. We then construct the tangent to the edge curve (which is normal to the gradient at that point) and use this to predict the next points (here, either r or s).

Figure 5.5 (right): Chapter 5, Forsyth and Ponce, *Computer Vision – A Modern Approach*

Hysteresis