Digital Image Processing

Lecture 4
Image Segmentation 1

Rahul JAIN

Spring 2022

Outline

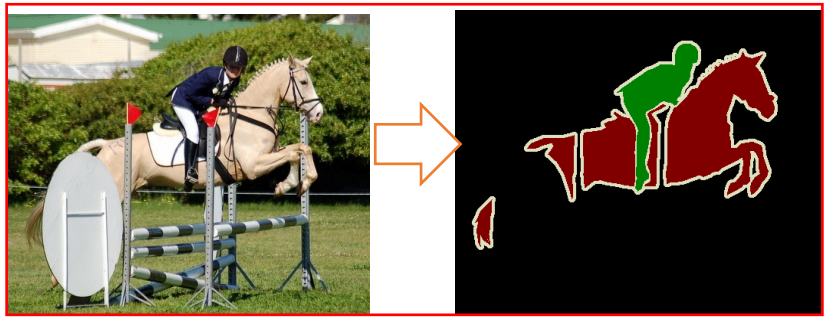
- Image Segmentation:
  - What is image segmentation
- Edge discontinuity detection-based segmentation of an image
- Segmentation using the following edge detection algorithm:
  - Marr-Hildreth Edge Detector
  - Canny Edge Detector

Image segmentation

- Let R represent the entire region occupied by an image
- We may view image segmentation process that partitions R into n subregions, $R_1, R_2, \ldots, R_n$ such that
  - $\bigcup_{i=1}^{n} R_i = R$: the segmentation must be complete, every pixel must be in the region
  - R_i is a connected set, for $i = 0,1,2, \ldots, n$: points in a region be connected in some predefined sense (e.g. the points must be 8-connected)
  - $R_i \cap R_j = \phi$: for all i and j, $i \neq j$: regions must be disjoint
  - $Q(R_i)$: TRUE for $i = 0,1,2, \ldots, n$: all pixels in R_i have the same intensity (or standard deviation of intensity values are specified constant)
  - $Q(R_i \cup R_j)$ = FALSE for any adjacent regions R_i and R_j: indicates two adjacent regions R_i and R_j must be different in the sense of predicate Q

Image segmentation

- Image Segmentation is the process of subdividing an image into its constituent regions or objects
- In simple terms, segmentation is the process of assigning labels to pixels



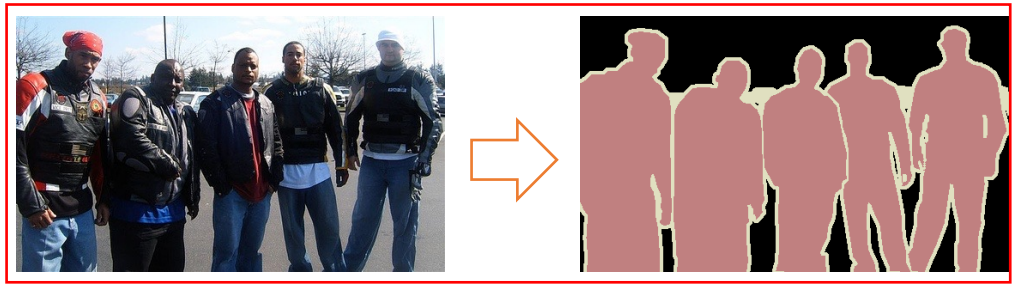
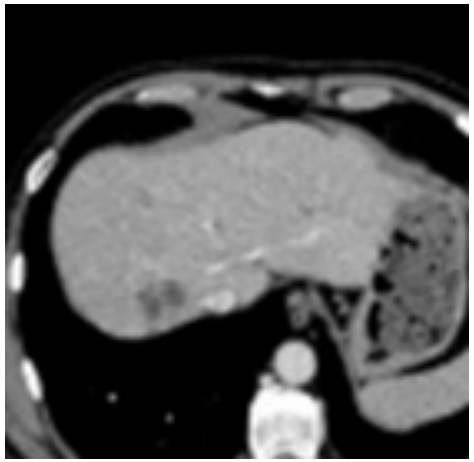


Image segmentation

- Level of subdivision depends on the problem being solved
- For a self-driving system, the goal is to identify pedestrians and vehicles on a road

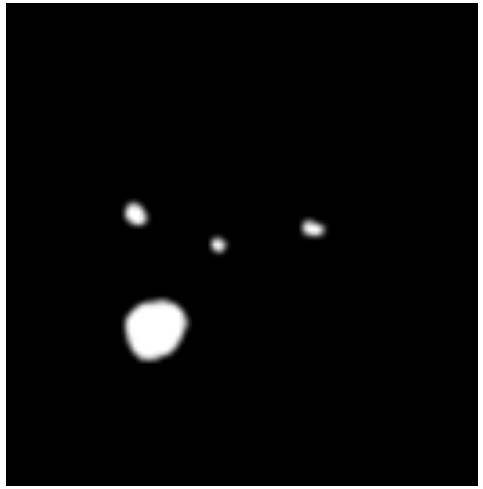
Image segmentation

- Segmentation is an important method for image processing and can help with object recognition
- The segmentation algorithm for grayscale images will be discussed
- Example: liver segmentation in grayscale image



Liver Image

Liver Mask

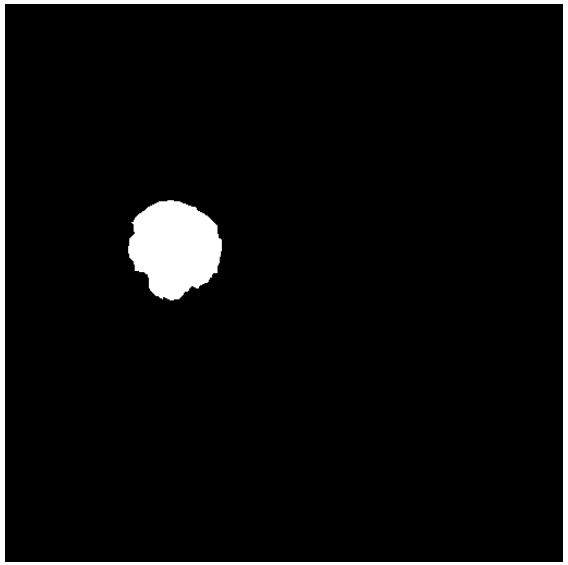


Tumor Mask

Image segmentation

- Tumor segmentation and tumor detection

Liver Tumor Image



Tumor Segmentation

Tumor Detection

Approaches for Image segmentation

- Gray scale image segmentation algorithms are based on two basic properties of intensity values (gray-level values):
  - **Discontinuity** - In this approach an image is partitioned based on abrupt changes in the intensity
  - **Similarity** – Partition an image into regions that are similar according to a set of predefined criteria.
- Today we will discuss discontinuity-based image segmentation

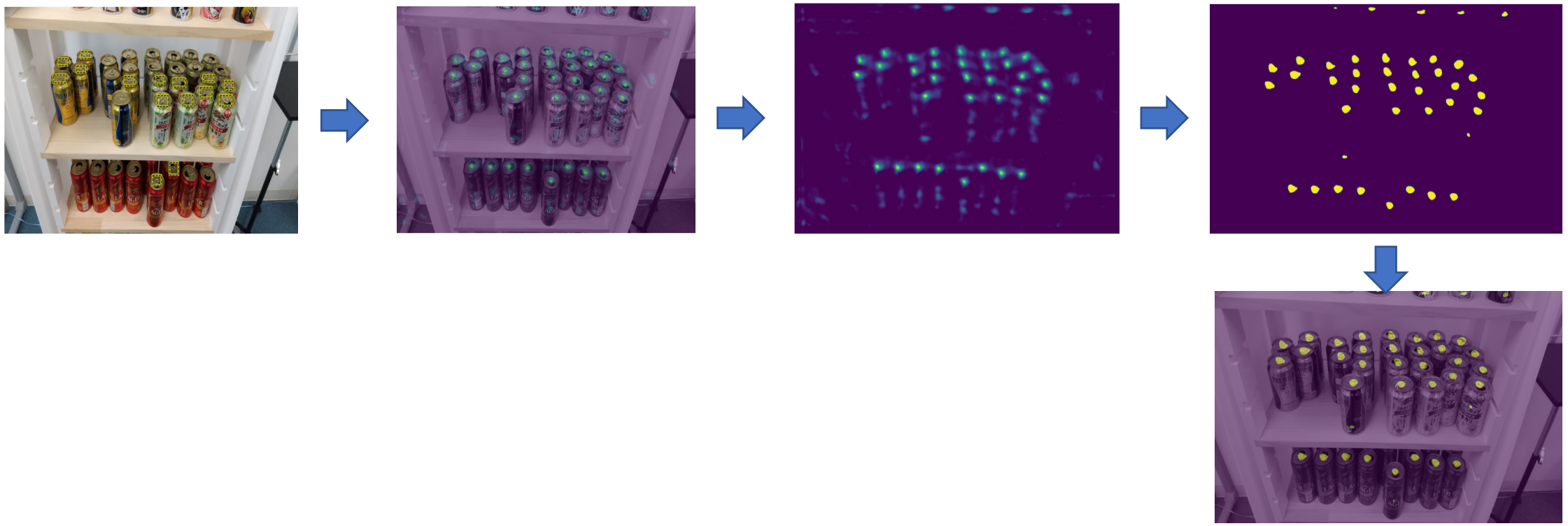
Discontinuity-based image partitioning

- The strategy for discontinuity is to partition an image based on abrupt (local) changes in gray-level value
- We can detect gray-level discontinuities in a digital image using three basic types of gray-level discontinuities: points, lines, and edges detection

Thresholding

- Thresholding

If $f(x, y) > T$ then $f(x, y) = 0$ else $\mathrm{f}(x, y) = 255$



Derivatives

- It is intuitive that abrupt, local changes in intensity can be detected using derivatives
- Many edge finding operators are based on the differentiation
- Apply the continuous derivates to a discrete image

$$\frac{df}{dx} = \lim_{h \to 0} \left(\frac{f(x+h) - f(x)}{h} \right)$$

- Since in an image the smallest possible value of *h* is 1 (the difference between index value of two adjacent pixels)
- A discrete function of the derivative expression is:

$$f(x+1) - f(x)$$

Derivatives

- The derivatives of a digital function f are defined in terms of differences

- The first-order derivative of a one-dimensional function $f(x)$:

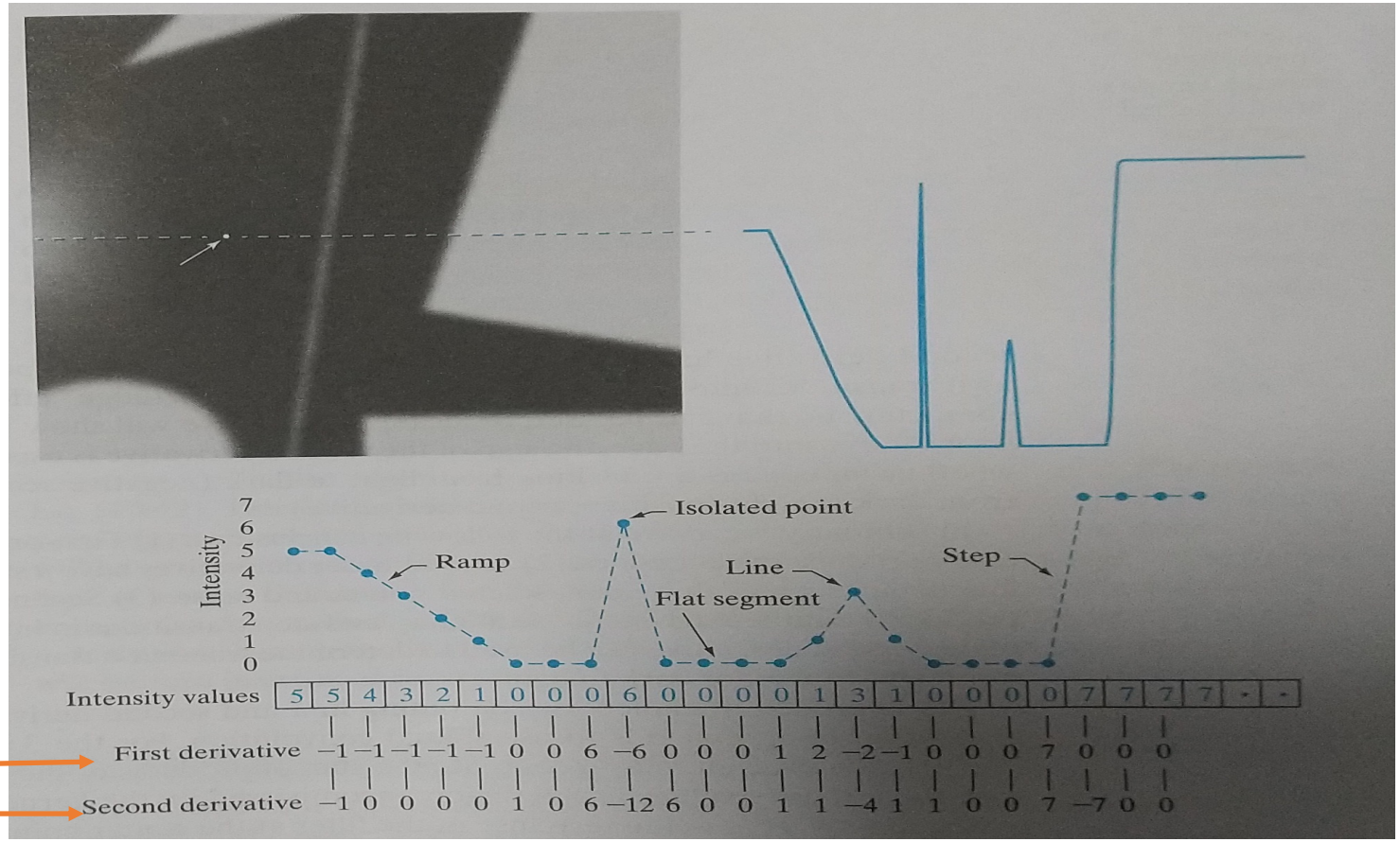
$$\frac{\partial f}{\partial x} = f(x+1) - f(x)$$

- The second-order derivative of $f(x)$:

$$\frac{\partial^2 f}{\partial x^2} = f(x+1) + f(x-1) - 2f(x)$$

Derivatives

- The approach of choice for computing first and second order derivatives at every pixel location in an image is to use spatial convolution.



$$\frac{\partial f}{\partial x} = f(x+1) - f(x)$$

$$\frac{\partial^2 f}{\partial x^2} = f(x+1) + f(x-1) - 2f(x)$$

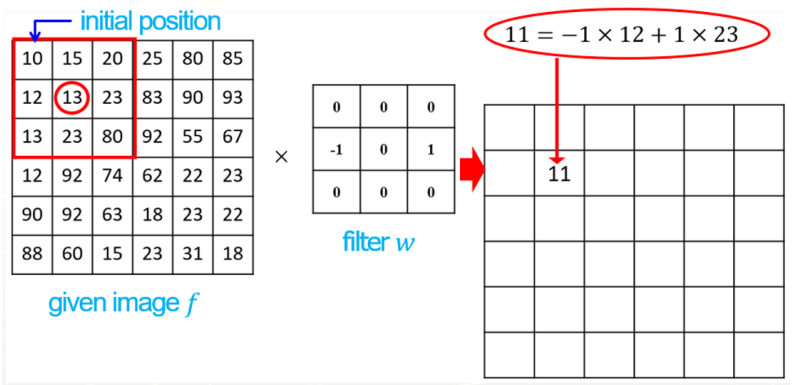
Image Source: Digital Image Processing by Rafeal et al.

Point detection using mask

- Points detection: The detection of isolated points using a mask (filter).

  We can detect a point at the location on which the mask is centered if: $|R| \geq T$ (T is a non-negative threshold)

  The procedure basically measures the weighted difference between the center point and its neighbor



  For X-ray image using the Laplacian operation [[-1,-1,-1],[-1,8,-1],[-1,-1,-1]] you can emphasize intensity on the points

Line detection

- The next level of complexity is line detection
- In this approach a mask is moved around an image
- We simply run the mask through the image and threshold the absolute value of the result
- It would respond more strongly to lines (one pixel thick) oriented horizontally

-1	-1	-1	-1	-1	2	-1	2	-1	2	-1	-1
2	2	2	-1	2	-1	-1	2	-1	-1	2	-1
-1	-1	-1	2	-1	-1	-1	2	-1	-1	-1	2
Horizontal			+45°			Vertical			-45°		

Line masks

Suggetsed reading:
Payam S.Rahmdel et al.: A Review of Hough Transform and Line Segment Detection Approaches

Edge detection

- The discontinuity approach involves segmenting an image based on abrupt (local) changes in gray-level value
- Edge detection is a widely used common image segmentation approach for detection meaningful discontinuities in gray level
- We can use edges to isolate objects from their background
- There are three edge models according to the intensity profile of pixels
  - Step edge
  - Ramp edge
  - Roof edge

Edge detection

Ways to model edges:

Step edges:

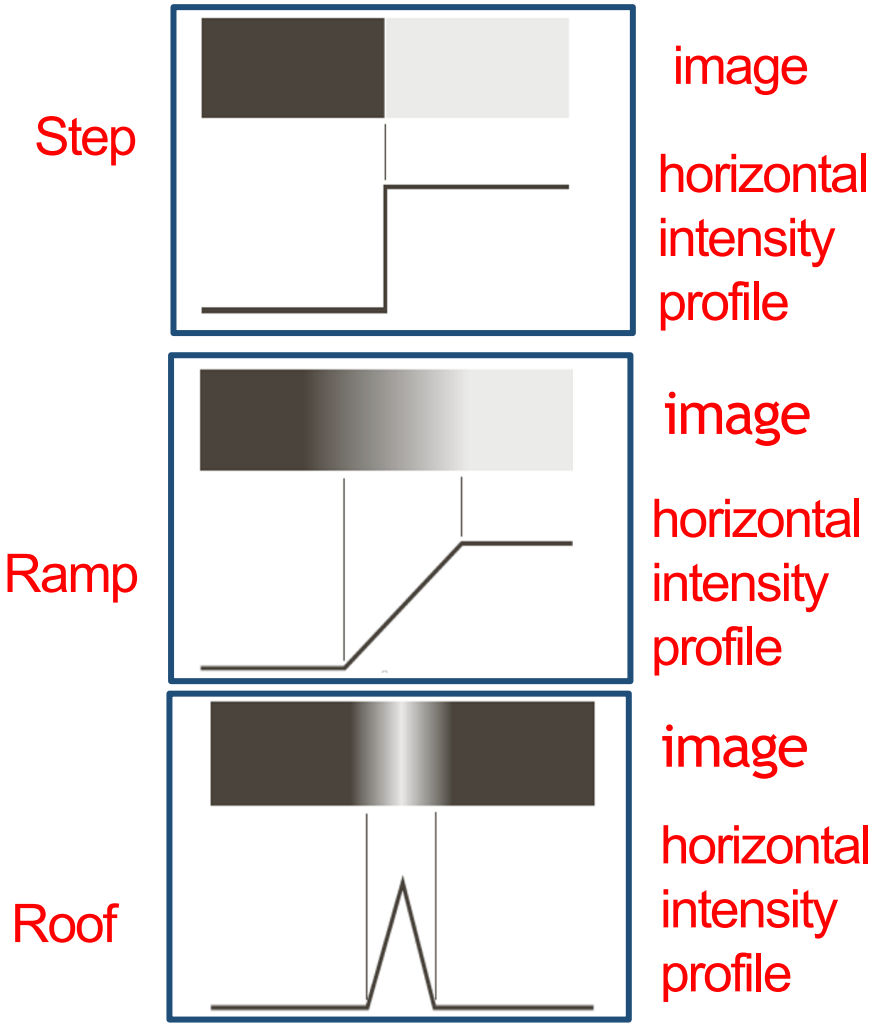
- Involve a transition between two intensity levels that occurs across a single-pixel distance
- Where the gray values change suddenly

Ramp edges:

- Because edges are usually blurred and noisy, edges are more closely modeled as an intensity ramp in practice
- Where the gray values change slowly

Roof edges:

- It models a line through a region



Edge detection

- Edge detection: An edge point is any point contained in the ramp and an edge segment would then be a set of such points that are connected

51	52	53	59
54	52	53	62
50	52	53	68
55	52	53	55

50	53	155	160
51	53	160	170
52	53	167	190
51	53	162	155

Blocks of pixels

- The grey values in the second and third columns have a distinct difference.
- This magnitude of difference is easily discernible by the human eye
- The goal of edge detection algorithms is to pick out the edges of an image

Derivatives

- We use partial derivatives for a two-dimensional image
- An important expression is *gradient* which is the vector defined by

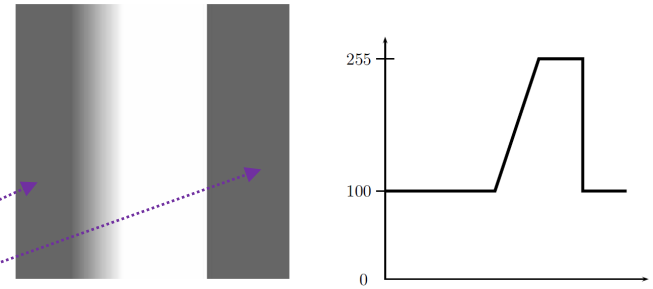
$$\left[\frac{df}{dx} \quad \frac{df}{dy}\right]$$

which for a function $f(x, y)$ points in the direction of its greatest increase

- The direction of that is given by $\tan^{-1}\left({}^{df/dy}/_{df/dx}\right)$
- Its magnitude can be given by: $\sqrt{\left(\frac{df}{dx}\right)^2+\left(\frac{df}{dy}\right)^2}$

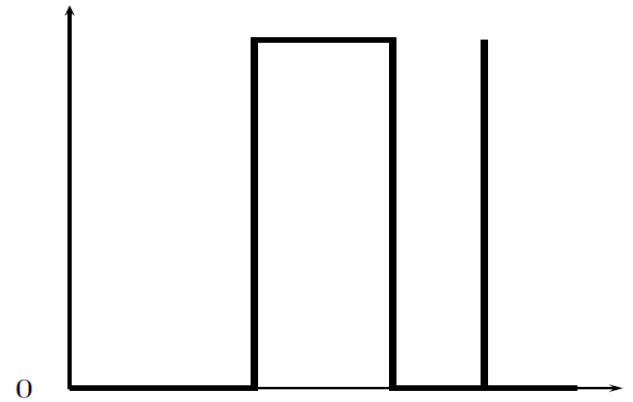
Derivatives

- The 1st and 2nd derivatives:
- Suppose the function which provides the profile is $f(s)$
- Then its derivates $f'(x)$



Edges and their profile

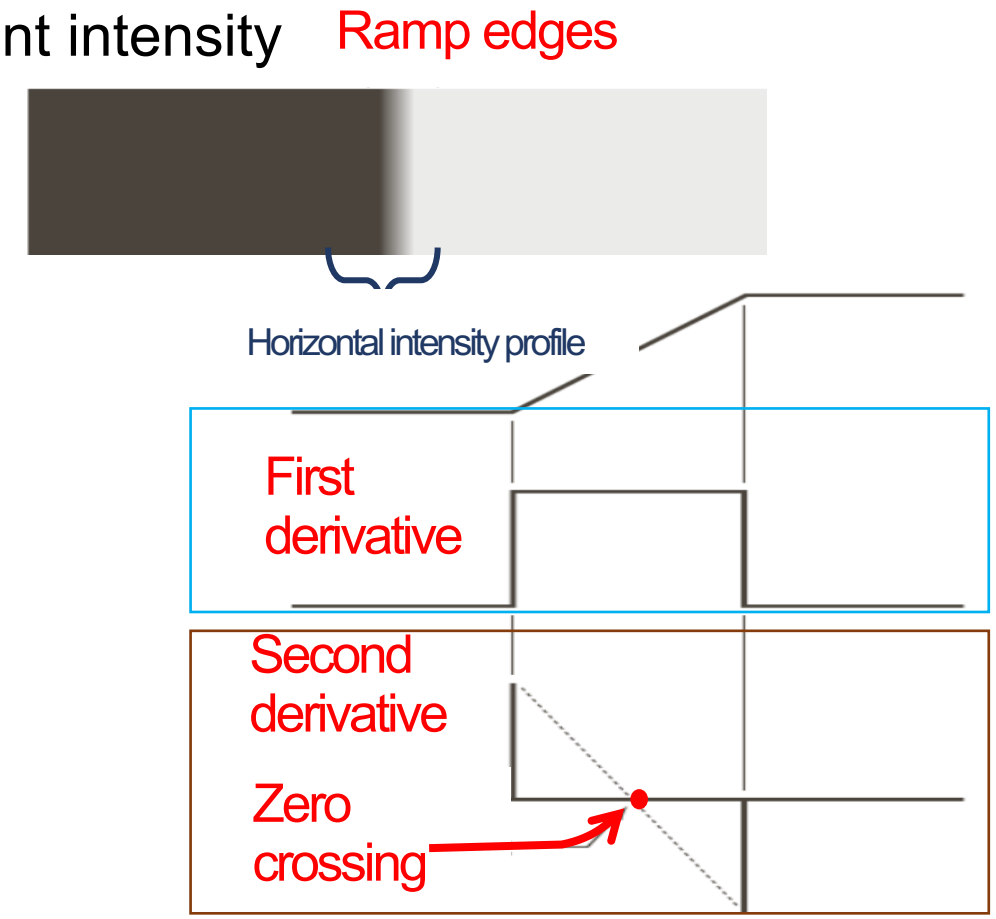
- The derivate returns zero for all constant section of the profile
- In this example, it returns non-zero value only those parts of the image in which difference occur



The derivative of the edge profile

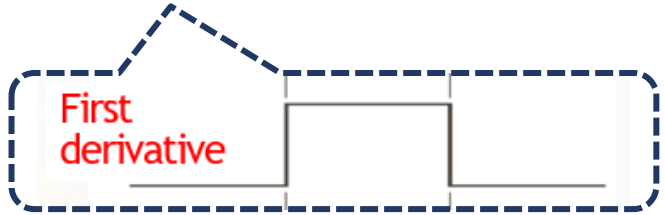
Derivatives

- The first derivative is positive at the start of the ramp and at other places along it
- The first derivative is zero at in areas of the constant intensity
- The second derivative is (dark to light)
  - positive at the beginning of the ramp,
  - negative at the end of the ramp, and
  - zero at the point of constant intensity
- The sign will be reversed when an edge transits (light to dark)

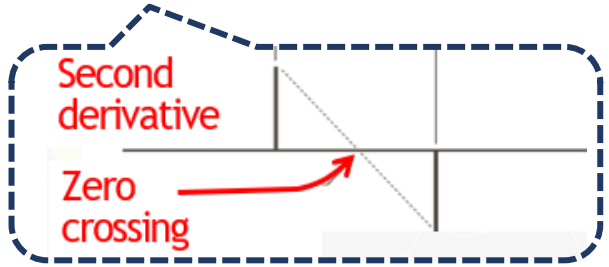


Derivatives

- The first derivative
  - Its magnitude can be used to detect the presence of edges
  - It usually produces thicker edges in an image



- The second derivative
  - It produces two values for every edge in an image
  - Its zero crossings can be used for locating the centers of thick edges
  - It has a stronger response to fine details, such as thin lines, isolated points and noise.



Basic steps in edge detection

- Three fundamental steps performed in edge detection:

1. Image smoothing for noise reduction: Noise reduction improves edge identification
2. Detection of edge points: Extract all points that are potential candidates to become edge points
3. Edge localization: Select from the candidate edge points that are true members of the set point set comprising an edge

Noise issue in edge detection

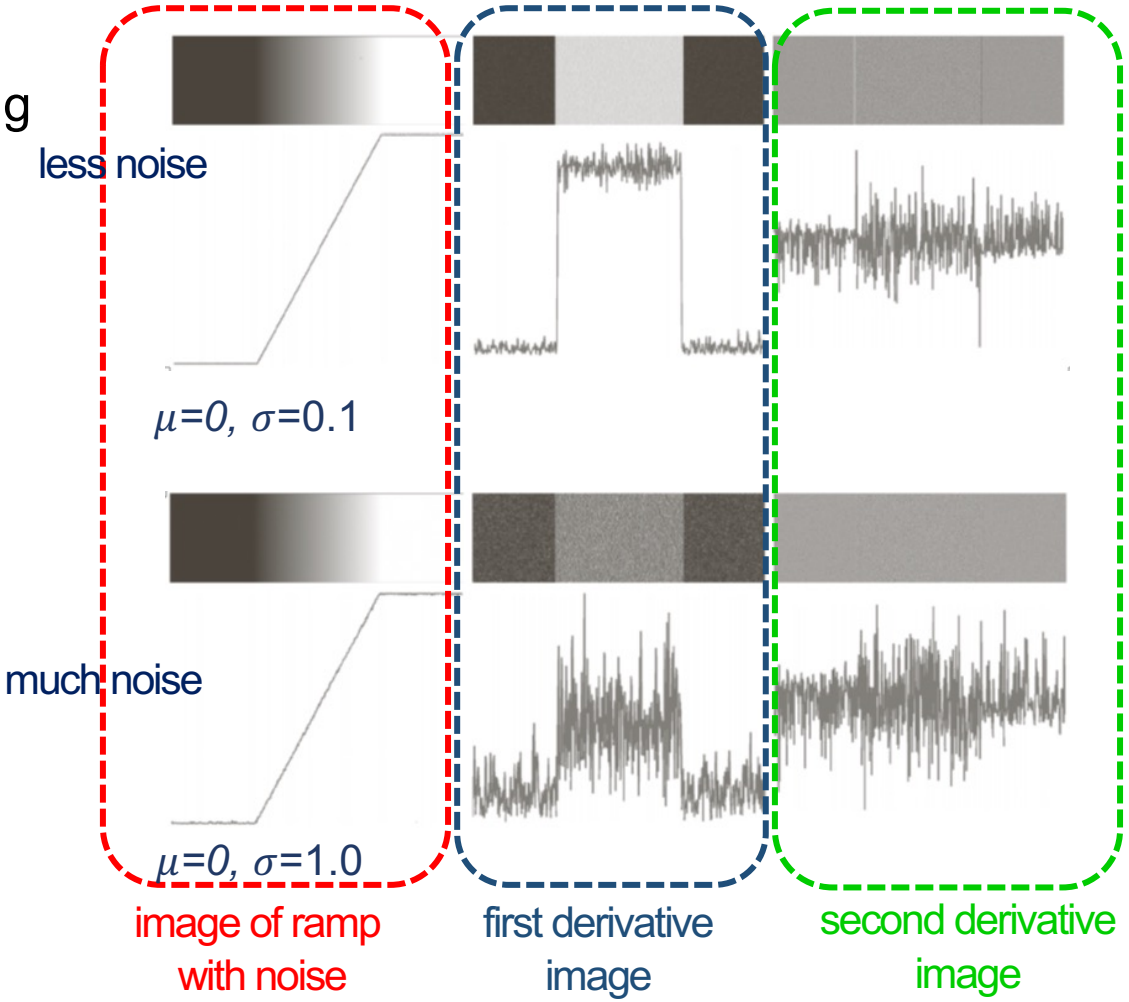
- Noise issue:
  - In edge detection, noise is a major issue
  - Noise has a significant impact on the 1st and 2nd order derivatives used for detecting edges

- The images is affected by Gaussian noise with

  μ= 0, σ = 0.1, 1.0

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2}$$

- Both derivatives are noise sensitive
- The second derivative is more noise-sensitive than the first



Edge detection methods

- An edge detection operator should have two key characteristics:
  - It should be a differential operator that is able to compute an approximation of the first or second derivatives
  - It should be an operator that can identify edges of various scales, such as blurry edges and sharply focused edges
- The Laplacian operator is one of the most satisfying operators that meets these requirements

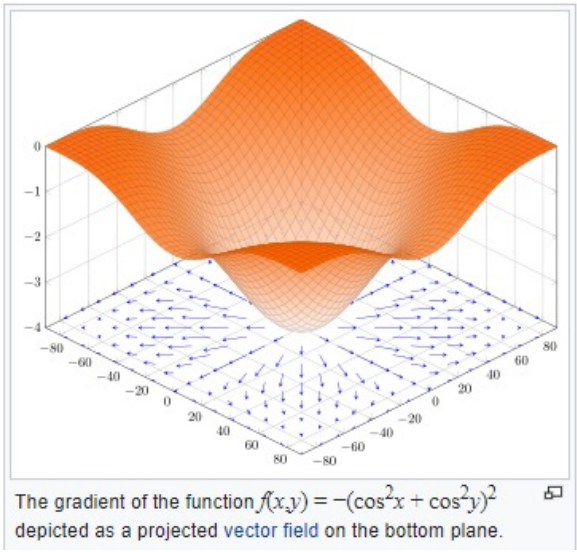
Laplacian operator

- Laplacian operator is based on second-order derivative
- It is **used to find edges in an image**
- It emphasize the intensity discontinuities in an image
- It is an isotropic filter; which means it is rotation invariant
  - Its response is independent of the direction of the discontinuities in the image to which the filter is applied

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

f is a twice-differentiable real-valued function

divergence $\nabla = \frac{d}{d_{x_1}}, \ldots, \frac{d}{d_{x_n}}$



The gradient of the function $f(x,y) = -(\cos^2 x + \cos^2 y)^2$ depicted as a projected vector field on the bottom plane.

Image Source: https://en.wikipedia.org/wiki/Gradient

Laplacian operator

Laplacian operator:

For a function (image) $f(x,y)$ of two variables: $\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$

- In the x-direction, we have

$$\frac{\partial^2 f}{\partial x^2} = f(x+1,y) + f(x-1,y) - 2f(x,y)$$

- Similarly, in the y-direction we have

$$\frac{\partial^2 f}{\partial y^2} = f(x,y+1) + f(x,y-1) - 2f(x,y)$$

The discrete Laplacian of two variables is:

$$\nabla^2 f = f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1) - 4f(x,y)$$

Laplacian operator

Implementation of the Laplacian

- This equation can be implemented using the filter masks:

(a)
$$\begin{matrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{matrix}$$

(a) Filter mask used to implement equation: $\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$

(b)
$$\begin{matrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{matrix}$$

(b) Filter mask used to implement an extension of this equation

(c)
$$\begin{matrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{matrix}$$

(d)
$$\begin{matrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{matrix}$$

- The signs of coefficients are opposite in masks
- They result in negative images of the masks

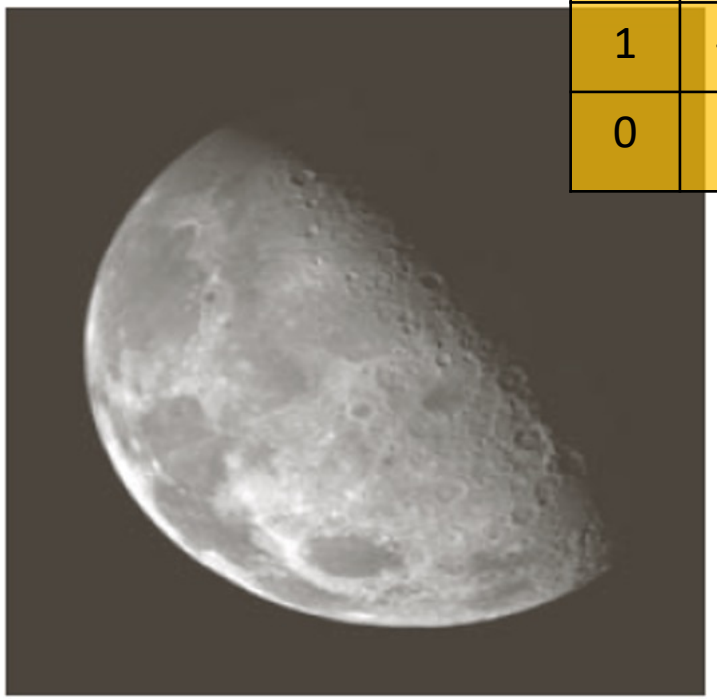
(c)(d) Two other implementations of the Laplacian found frequently in practice

Notice that all the mask coefficients sum to zero.

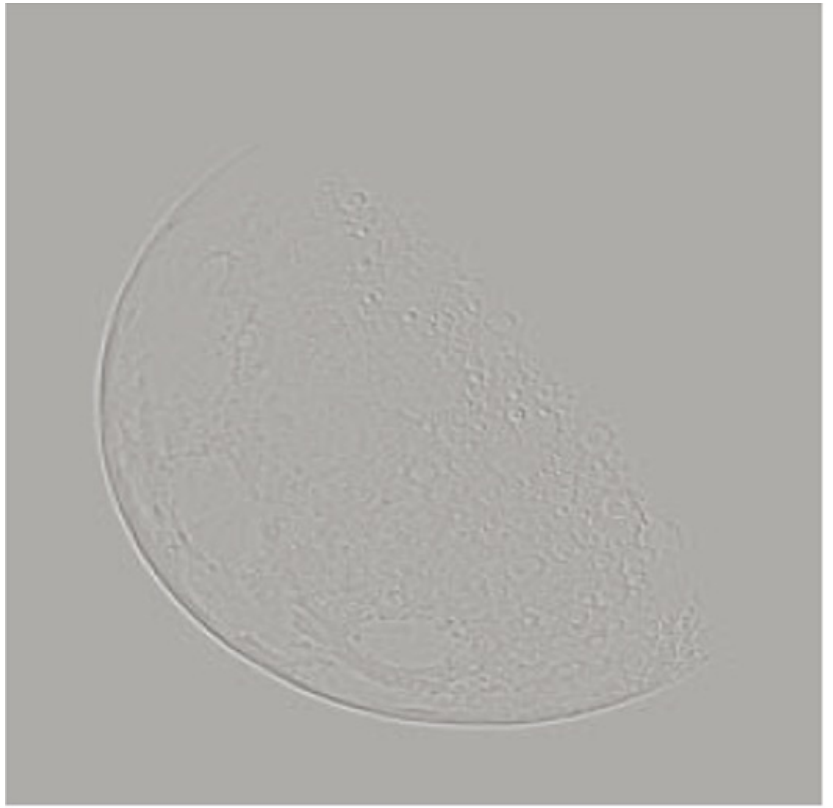
Laplacian operator

Laplacian filter

0	1	0
1	-4	0
0	1	0



Blurred image of the North pole of the moon



Scaled Laplacian image

You can combine the gradient with thresholding

Large sections of the image are black because the Laplacian contains both positive and negative values and all negative values are clipped at 0 by the display

Laplacian of Gaussian

- The Laplacian operator's formula is:

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

If *G*, is a 2-D Gaussian function

- Gaussian function can be defined by:

$$G(x, y) = e^{-\frac{x^2+y^2}{2\sigma^2}}$$

where σ is standard deviation (also called space constant)

Laplacian of Gaussian

- Substitute Gaussian function of *G* in Laplacian equation to find the expression of $\nabla^2 G$:

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

$$= \frac{\partial}{\partial x}\left[\frac{-x}{\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}\right] + \frac{\partial}{\partial y}\left[\frac{-y}{\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}\right]$$

$$= \left[\frac{x^2}{\sigma^4} - \frac{1}{\sigma^2}\right] e^{-\frac{x^2+y^2}{2\sigma^2}} + \left[\frac{y^2}{\sigma^4} - \frac{1}{\sigma^2}\right] e^{-\frac{x^2+y^2}{2\sigma^2}}$$

- Collecting terms gives the final expression:

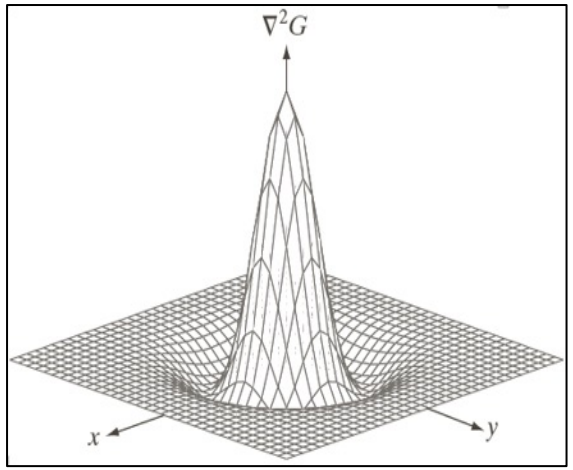
$$\nabla^2 G(x,y) = \left[\frac{x^2 + y^2 - 2\sigma^2}{\sigma^4}\right] e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Laplacian of Gaussian

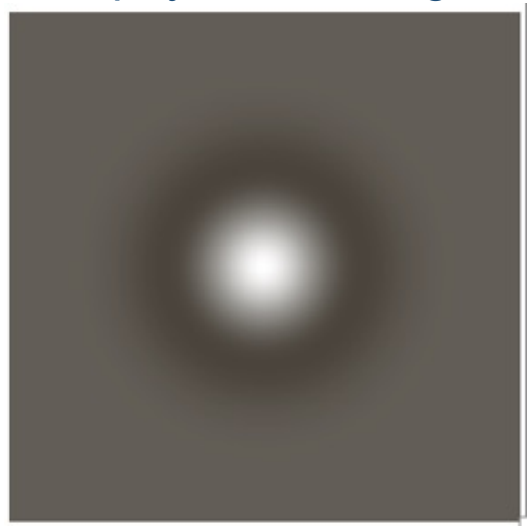
- The Laplacian of a Gaussian (LoG):

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

3D plot of the negative of the LoG



Negative of the LoG displayed as an image



Laplacian of Gaussian

- There are two zero crossings

Zero crossings occur at a circle:

$$x^2 + y^2 = 2\sigma^2$$

The LoG can be used for

- Smoothing the image
- Detecting zero crossings (edges)

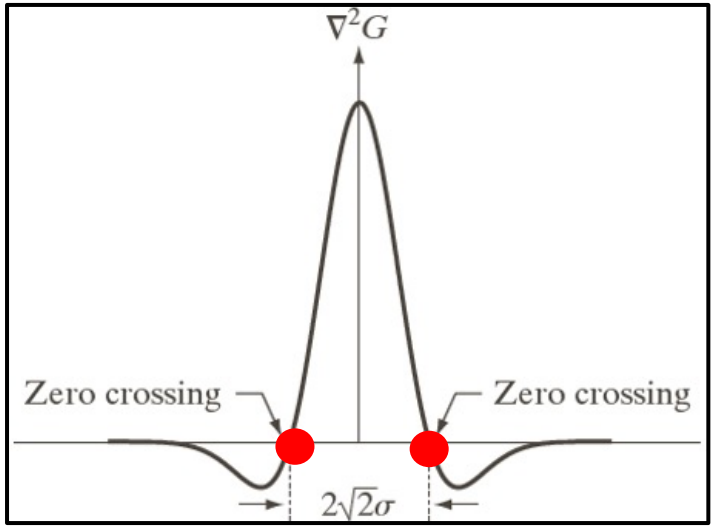
at the same time.

- Use a 5 × 5 filter to approximate the shape of LoG
- This approximation is not unique
- Filters of any size can be generated by sampling by equation

$\nabla^2 G(x,y) = \left[\frac{x^2+y^2-2\sigma^2}{\sigma^4}\right] e^{-\frac{x^2+y^2}{2\sigma^2}}$ and scaling the coefficients

so that they sum to zero

2-D plot of the negative of the LoG



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

5 × 5 LoG filter

The Marr-Hildreth Algorithm

- The Marr-Hildreth algorithm consists of two parts:

Part 1. Convolving the LoG filter with an input image, f

$$g(x,y) = \nabla^2[G(x,y)] \star f(x,y)$$

Part 2. Finding the zero crossings of $g(x,y)$ to determine the locations of edges in f

- Because of the linear processes,

$$g(x,y) = \nabla^2\left[G(x,y) \star f(x,y)\right]$$

We can smooth the image first with a Gaussian filter and then compute the Laplacian of the result.

The Marr-Hildreth Algorithm

- The Marr-Hildreth algorithm:

  Step 1. Filter the input image with an $n \times n$ Gaussian lowpass filter

  Step 2. Compute the Laplacian of the image resulting from step 1 using for example, the 3×3 mask

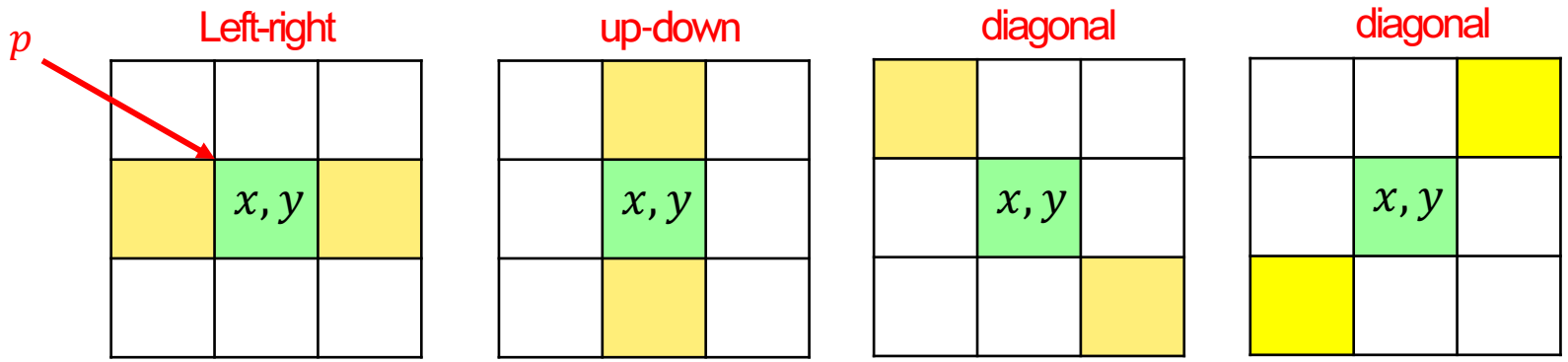
  Step 3. Find the zero crossings of the image from Step 2

1	1	1
1	-8	1
1	1	1

Laplacian mask (b)

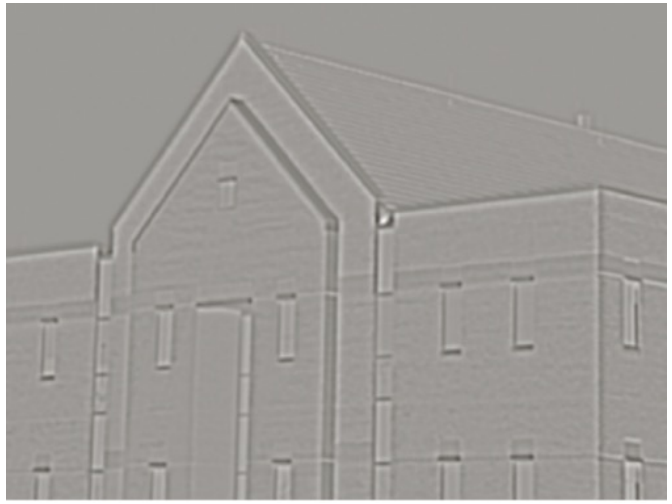
The Marr-Hildreth algorithm

- How to find zero crossings at any pixel, p, of $g(x, y)$?
- Take a 3×3 neighborhood centered at p as an example
- p is a zero crossing if it meets the following two conditions
  - The signs of at least two of p's opposing (left/right, up/down and two diagonals) neighboring pixels must differ
  - The absolute value of their numerical difference must exceed a threshold



Example of using Marr-Hildreth Edge Detector

Original building image of size 834×1114



LoG image

- The result of the convolution, using a Gaussian filter with $\sigma = 4$ and $n = 25$.
- Intensity values are scaled to the range [0,1].

Example of using Marr-Hildreth Edge Detector

The original image

Detected edges

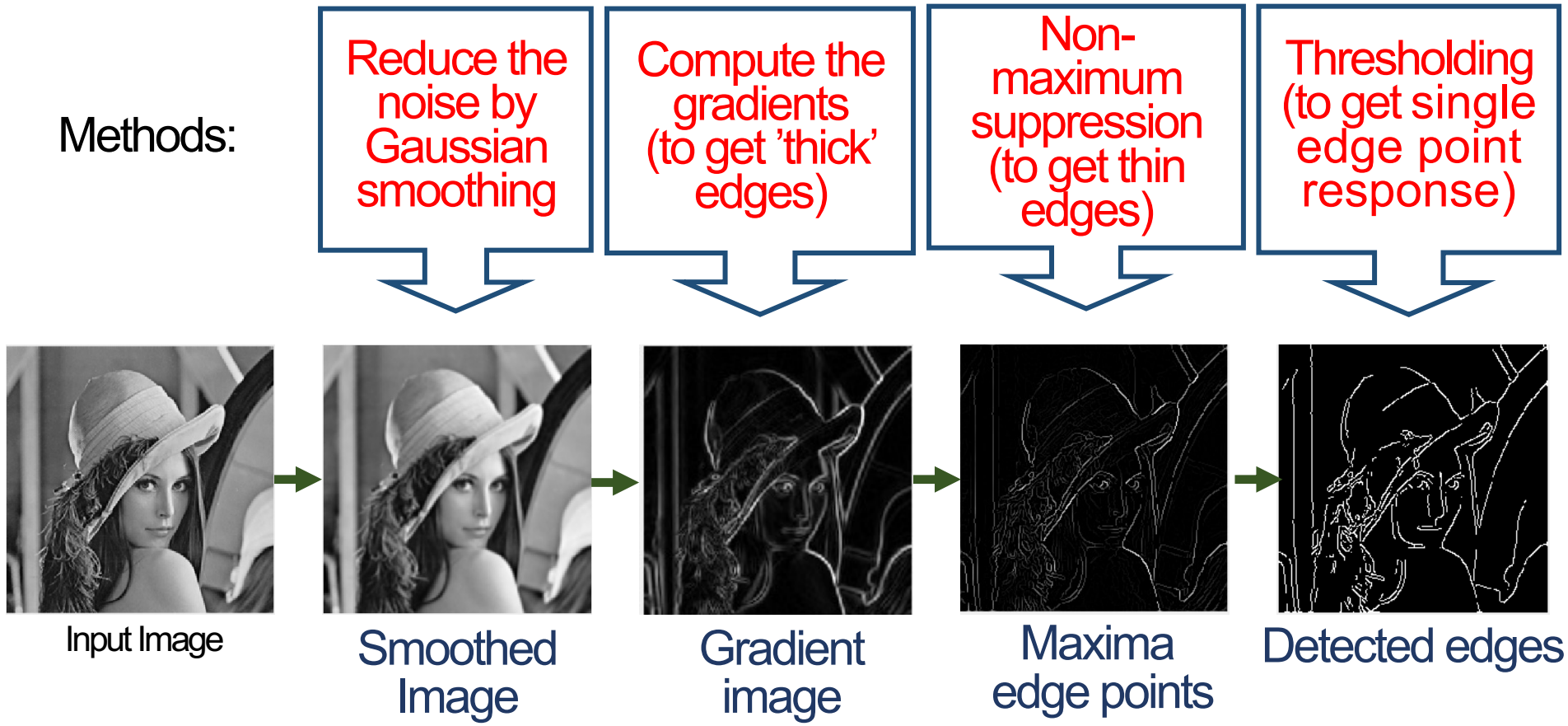
- A threshold of 4% of the maximum value of the LoG image is used to find zero crossings
- The generated edges have a thickness of 1 pixel

The Canny Edge detector

- The performance of Canny edge detector is superior to other methods
- The Canny's approach is based on three basic objectives.
  - Low error rate - All edges should be found, and there should be no false responses
  - Edge points should be well localized - The edges must be located as close as possible to the true edges
  - Single edge point response - The detector should return only one point for each true edge point

The Canny Edge detector

- Steps of Canny edge detector algorithm:



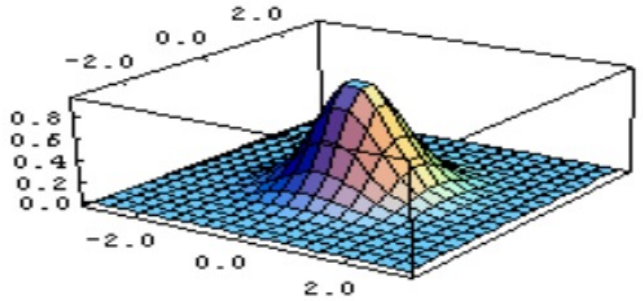
Smoothing

- The output of a smoothing (linear spatial filter) is simply the average of the pixels contained in the neighborhood of the filter mask
- These filters are called averaging filters
- Smoothing filters are also referred as lowpass filter
- Smoothing filters are used for blurring and for noise reduction
  - Blurring is useful in preprocessing tasks such as removal of small details from an image
  - Bridging the gap between small gaps and lines

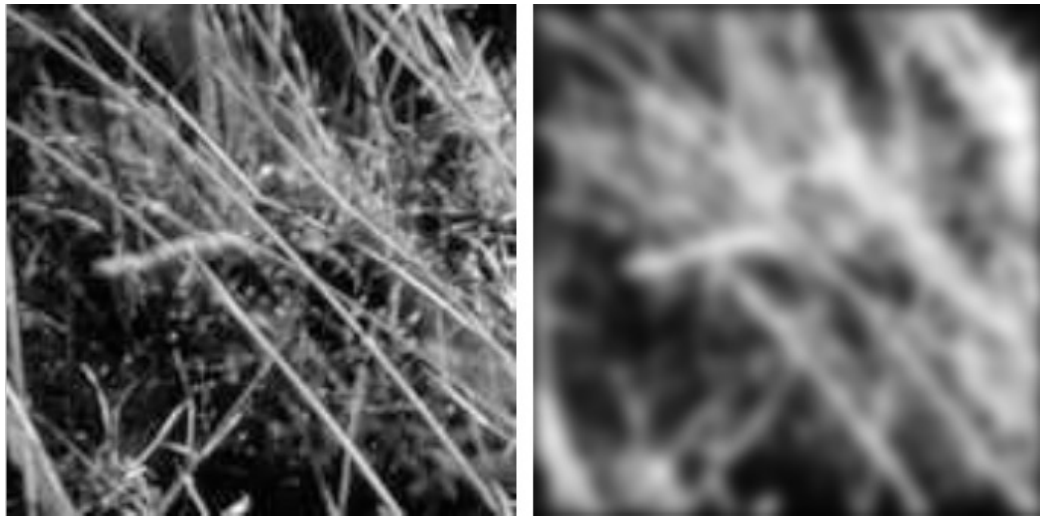
Smoothing using Gaussian filter

- This filter is an approximation of a Gaussian function:

$$G_{2D}(x, y, \sigma) = \frac{1}{2\pi\sigma^2} exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)$$



exp (exponential functions: e – 2.7182818…)
σ determines the width of the Gaussian kernel
x is the distance from the origin in the horizontal axis
y is the distance from the origin in the vertical axis



Smoothed image by a Gaussian filter

$\frac{1}{16} \times$

1	2	1
2	4	2
1	2	1

Gaussian filter

- A 5×5 2-D Gaussian filter sampled from Eq.(4.2)
- The coefficients must sum to 0

$$G(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}}$$

$f(x,y)$

★

0.023247	0.033824	0.038328	0.033824	0.023247
0.033824	0.049214	0.055766	0.049214	0.033824
0.038328	0.055766	0.063191	0.055766	0.038328
0.033824	0.049214	0.055766	0.049214	0.033824
0.023247	0.033824	0.038328	0.033824	0.023247

A 5×5 Gaussian filter with $\sigma = 2$

$f_s(x,y)$

Gaussian filter

- How to generate 5 × 5 Gaussian filter?

(-2,-2)	(-1,-2)	(0,-2)	(1,-2)	(2,-2)
(-2,-1)	(-1,-1)	(0,-1)	(1,-1)	(2,-1)
(-2,-0)	(-1,0)	(0,0)	(1,0)	(2,0)
(-2,1)	(-1,1)	(0,1)	(1,1)	(2,1)
(-2,2)	(-1,2)	(0,2)	(1,2)	(2,2)

← Set the range of pixel position (x, y) to [-2,1,0,1,2]

$$G(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}}$$ ← Compute the coefficient by Gaussian function

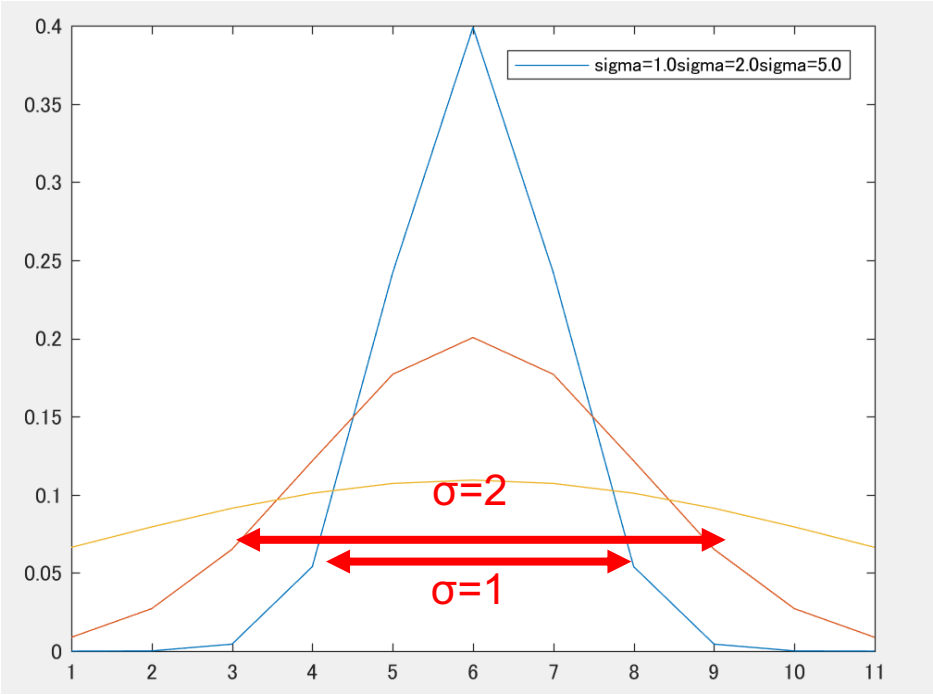
0.023247	0.033824	0.038328	0.033824	0.023247
0.033824	0.049214	0.055766	0.049214	0.033824
0.038328	0.055766	0.063191	0.055766	0.038328
0.033824	0.049214	0.055766	0.049214	0.033824
0.023247	0.033824	0.038328	0.033824	0.023247

Normalize the filter so that the sum of coefficient to zero by

$$G'(x,y) = \frac{G(x,y)}{\sum G(x,y)}$$

Gaussian filter

- When σ gets bigger, the effect of smoothing becomes stronger and a larger filter $[N, N]$ is necessary



σ=1: window size around 5x5

σ=2: window size around 7x7

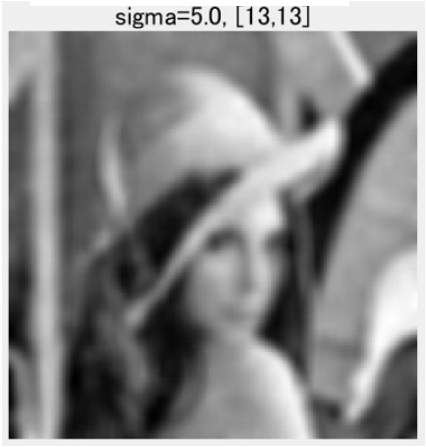
σ=5: need bigger window

Original image

σ=1.0 [5,5]

σ = 2.0 [7,7]

σ = 5.0 [13,13]



The Canny Edge detector

Step 1. Smooth the input image (reduction of noise)

- Convolving the Gaussian function G and the input image f, and obtain a smoothed image, f_s

$$f_s(x,y) = G(x,y) \bigstar f(x,y)$$

$$G(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}}$$

- This is implemented by filtering f with a Gaussian filter

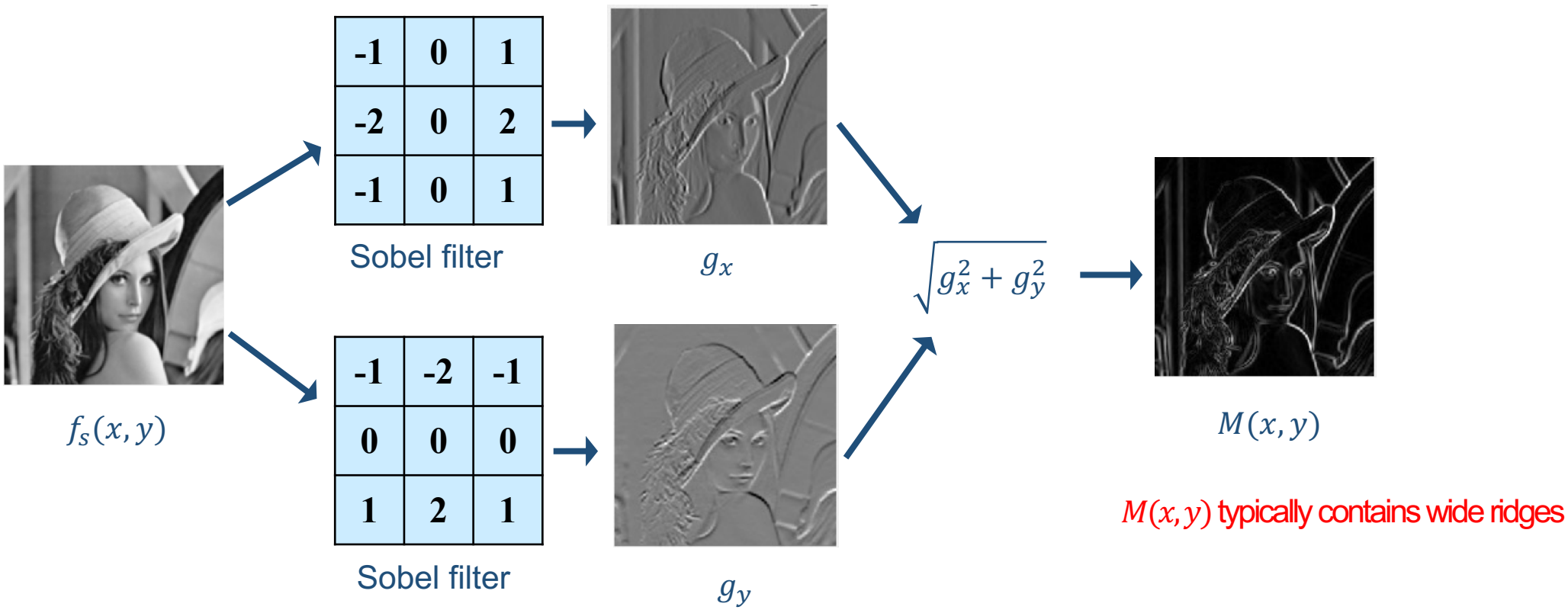
$f(x,y)$  $f_s(x,y)$

The Canny Edge detector

Step 2. Compute the gradients

(a) Compute magnitude of gradients of smoothed image by

$$M(x,y) = \sqrt{g_x^2 + g_y^2} \qquad \textit{i.e.} \quad \left(g_x = \frac{\partial f_s}{\partial x}, g_y = \frac{\partial f_s}{\partial y}\right)$$



$M(x,y)$ typically contains wide ridges

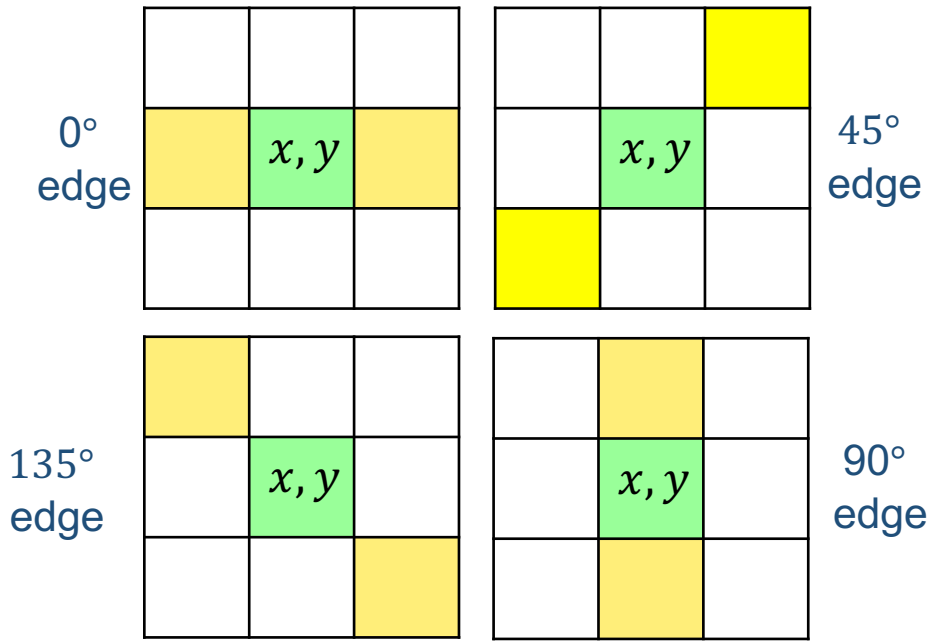
The Canny Edge detector

Step 2. Compute the gradients

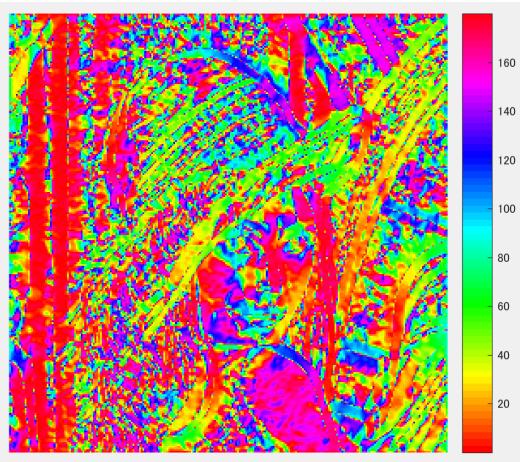
(b) Compute the direction of gradient image by:

$$\alpha(x, y) = \tan^{-1}\left[\frac{g_y}{g_x}\right]$$

(c) Discretize the directions into four groups:



$\alpha(x, y)$



0° : α< 22.5 ||α>=157.5

45° : 22.5 <α<= 67.5

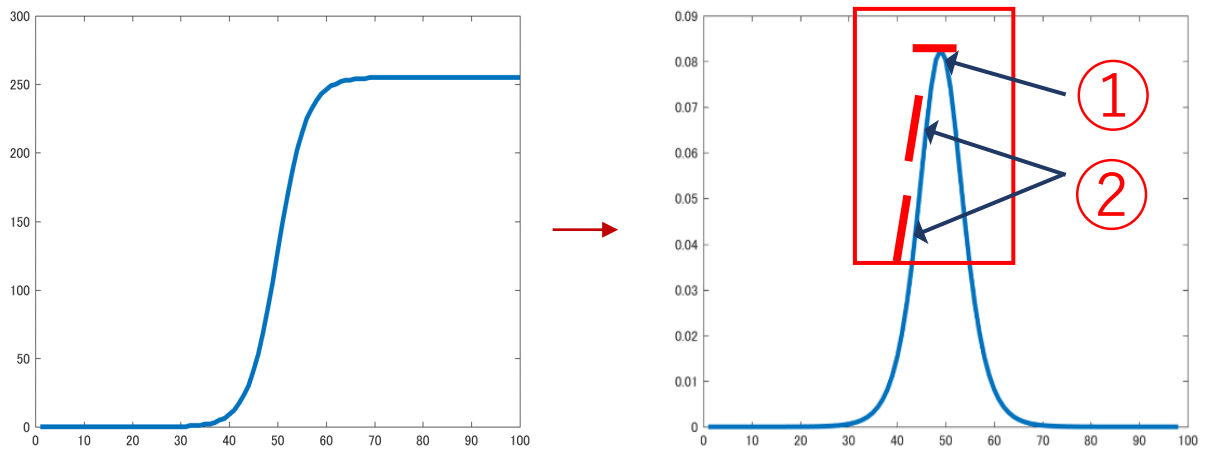
90° : 67 .5 <α<= 112.5

135° : 112.5 <α< 157.5

The Canny Edge detector

Step 3. Apply non-maximum suppression to get thin edges

the area with big magnitude of gradients



1D function

gradient

① the gradient is maximum

→ keep it as an edge point

② gradients of neighbors are bigger

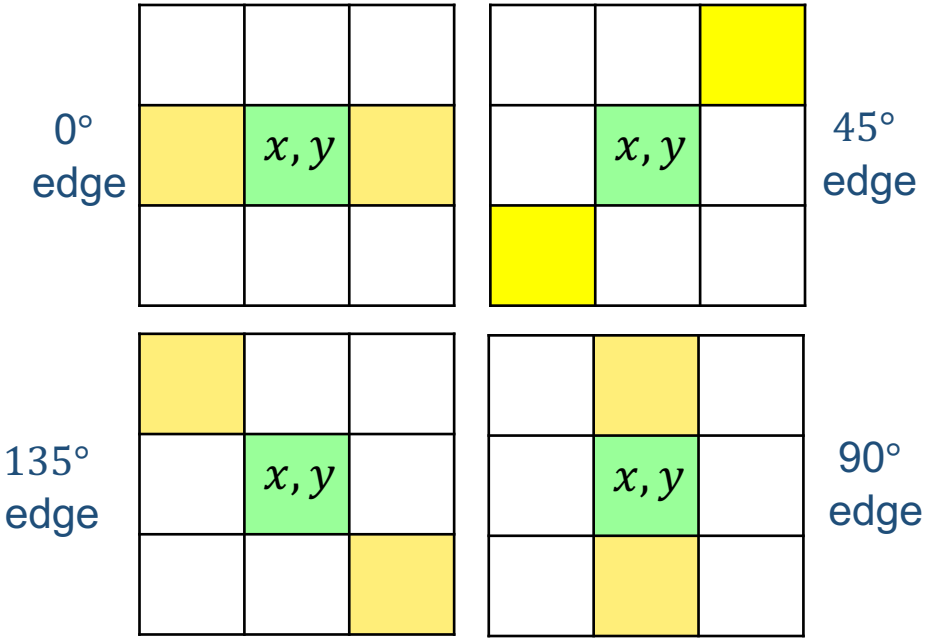
→ suppress it

The Canny edge detector

Step 3. Non-maximum suppression (thinning edges by detecting maximum position in gradient direction)

Algorithm: If $M(x, y)$ is less than at least one of its two neighbors *in the gradient direction*, let it be zero (suppression): $g_N(x, y) = 0$

otherwise keep it: $g_N(x, y) = M(x, y)$



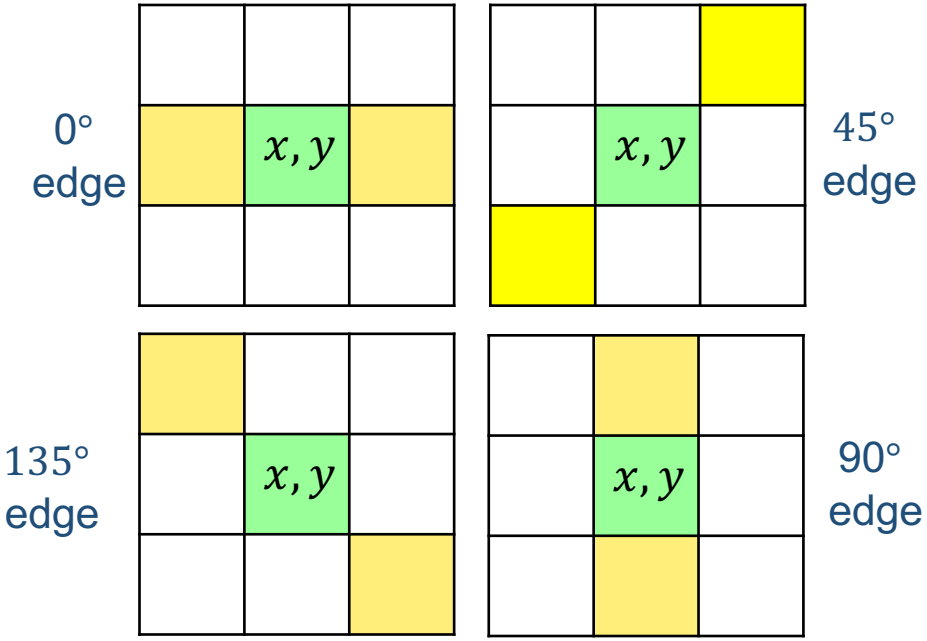
The Canny edge detector

Step 3. Non-maximum suppression (The 'thick' edges extracted are thinned by non-maximum suppression)

Algorithm: If $M(x, y)$ is less than at least one of its two neighbors *in the gradient direction*,

let it be zero (suppression): $g_N(x, y) = 0$

otherwise keep it: $g_N(x, y) = M(x, y)$



Less than the neighbor

30	30	70
30	50	30
20	30	30

magnitude

30	30	70
30	0	30
20	30	30

suppression

Larger than two neighbors

30	40	30
30	50	40
20	30	30

magnitude

30	40	30
30	50	40
20	30	30

remaining

The Canny edge detector

- Right: the output image after applying non-maximum suppression

$M(x, y)$

thinning

$g_N(x, y)$

The Canny edge detector

Step 4. Hysteresis threshold: reduce false edge points by thresholding

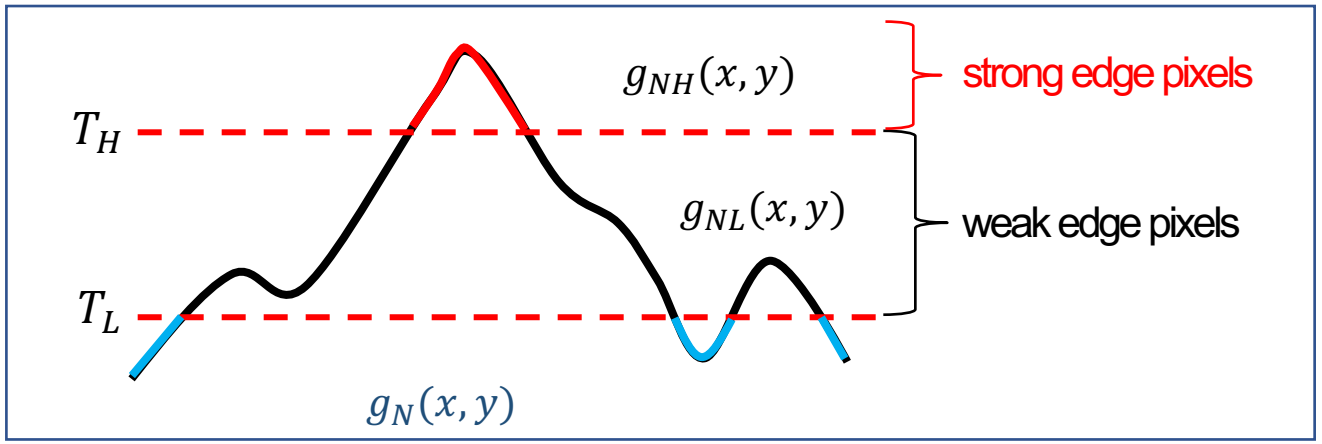
(a) Use two thresholds: T_H and T_L ($T_H > T_L$) to create two additional images:

$$g_{NH}(x,y) = g_N(x,y) \geq T_H$$

$$g_{NL}(x,y) = g_N(x,y) \geq T_L$$

(b) Eliminate from $g_{NL}(x,y)$ all the strong edge pixels:

$$g_{NL}(x,y) = g_{NL}(x,y) - g_{NH}(x,y)$$



The Canny edge detector

Step 4. Hysteresis threshold: reduce false edge points by thresholding

(c) Algorithm:

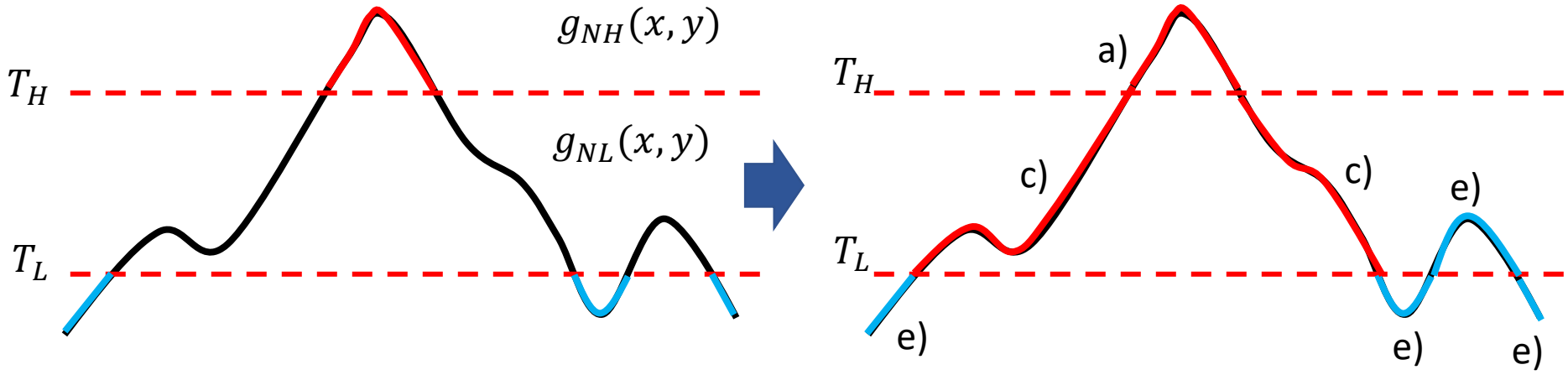
1) Mark all strong pixels in $g_{NH}(x,y)$ as edge points

2) Locate the next unvisited edge pixel, p, in $g_{NH}(x,y)$

3) Check the 8-connectd neighbors of p, mark the neighbors in $g_{NL}(x,y)$ as edge pixels

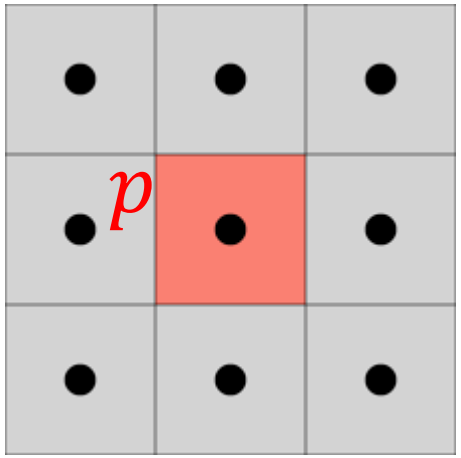
4) Repeat the steps 2-3 until all nonzero pixels in $g_{NH}(x,y)$ are visited

5) Set all pixels in $g_{NL}(x,y)$ that were not marked as edge pixels to 0

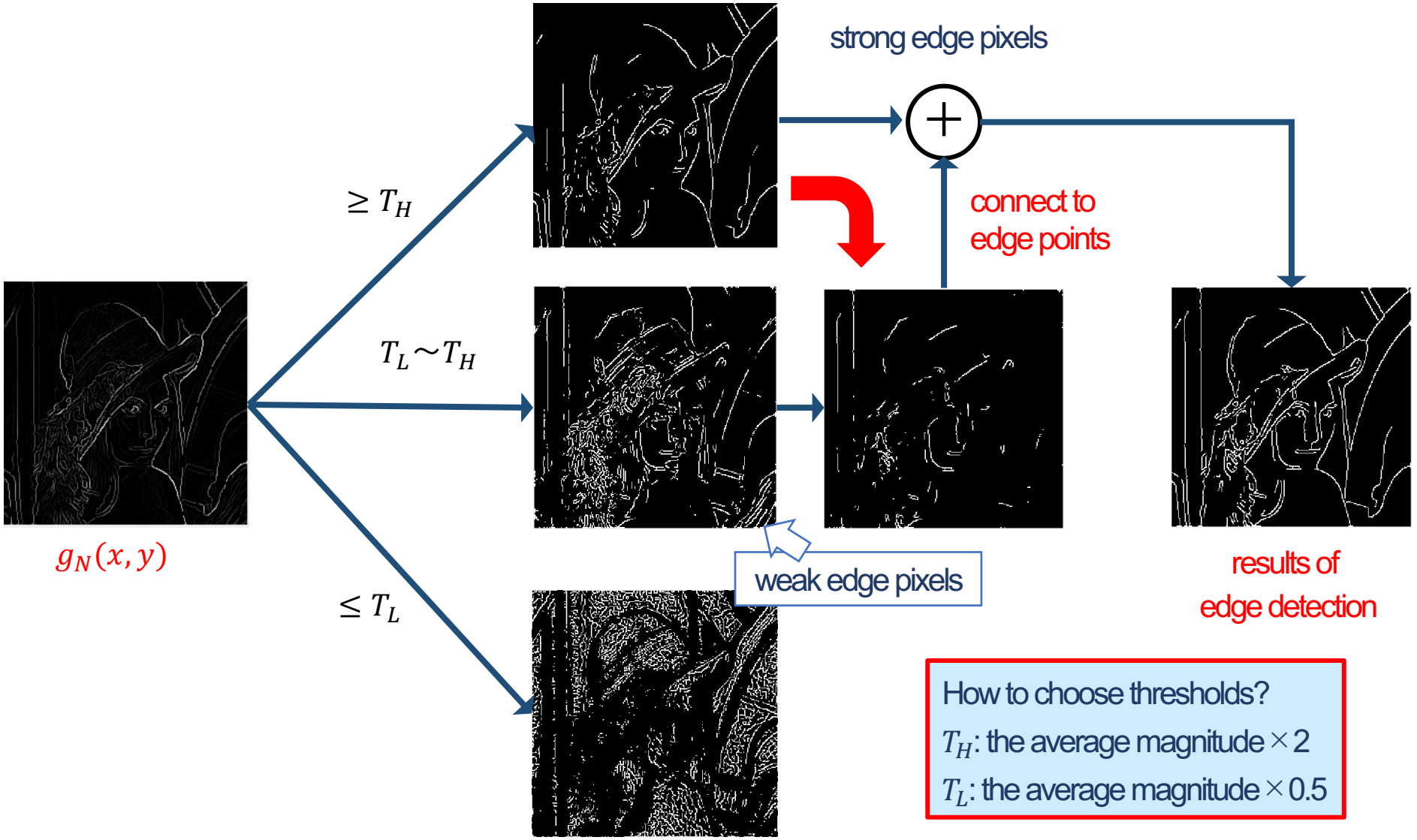


The Canny Edge Detector

- 8-connected pixels are neighbors to every pixel that touches one of their edges or corners
- These pixels are connected horizontally, vertically, and diagonally



The Canny Edge Detector

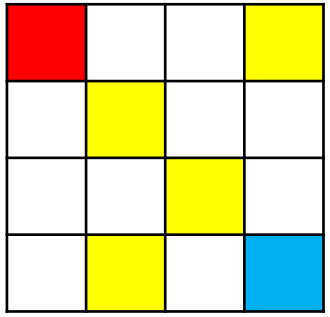


How to choose thresholds?

T_H: the average magnitude $\times 2$

T_L: the average magnitude $\times 0.5$

The Canny Edge Detector

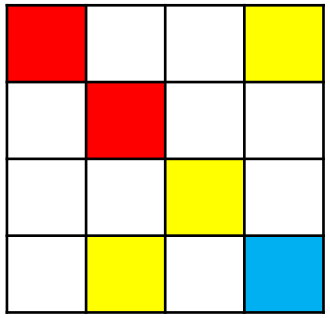


S1. Set all strong edge pixels (red) as valid edge pixels (red).

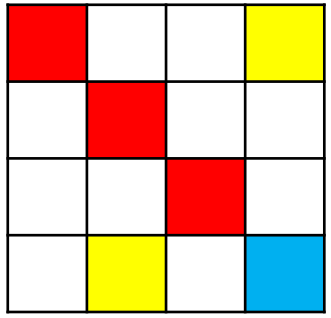
: Strong edge pixels

: Weak edge pixels

: Gradients $< T_L$

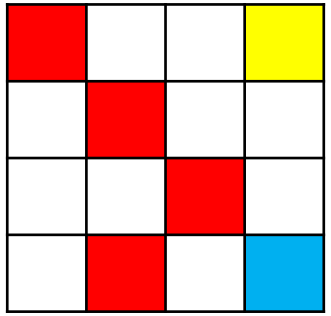


S2. Set a weak edge pixel (yellow) as valid edge pixel if it has an edge pixel as a neighbor.



S3. repeat S2 until no color change occurs.

result



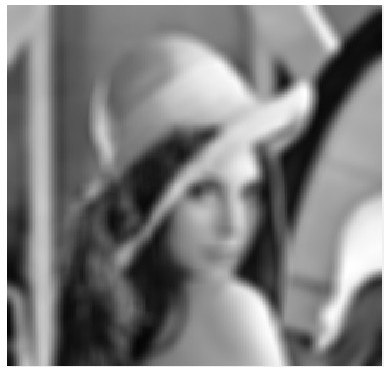
The Canny Edge Detector

- Results with various Gaussian parameters

Gaussian σ = 0.1, [5,5]

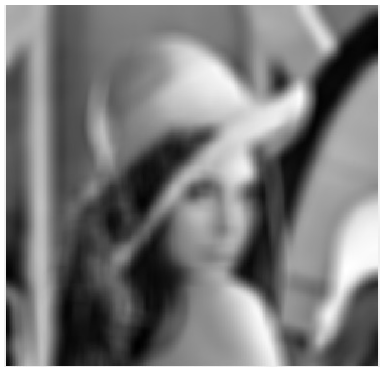
Gaussian σ = 1, [5,5]

Gaussian σ = 2, [5,5]



Gaussian σ = 5, [11,11]

Gaussian σ = 10, [13,13]



The Canny Edge Detector

- Results with various thresholds.
  - All the cases with Gaussian σ=1, [5,5]

T_H: average magnitude × 2
T_L: average magnitude × 0.5

T_H: average magnitude × 1
T_L: average magnitude × 0.5

T_H: average magnitude × 2
T_L: average magnitude × 1

Thank you for your attention