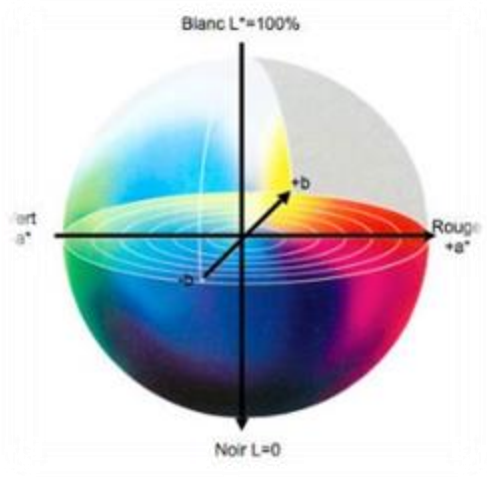
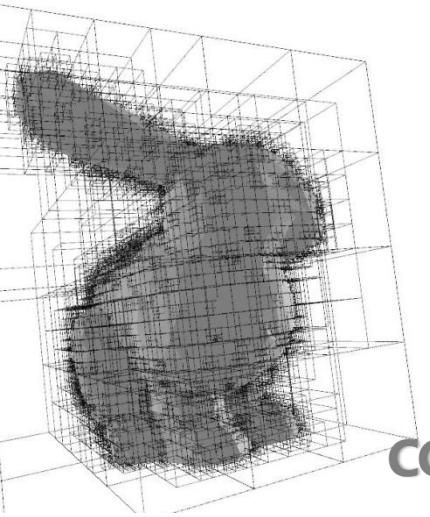
Image-Based Graphics





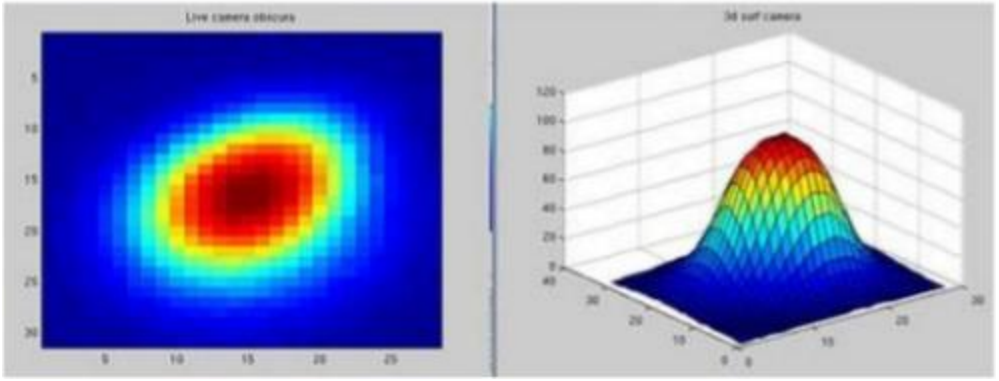
Fundamentals of Image Processing

Outline

- Pixels and colours
- Point operations
- Histograms
- Filtering and morphology
- Low-level features of images

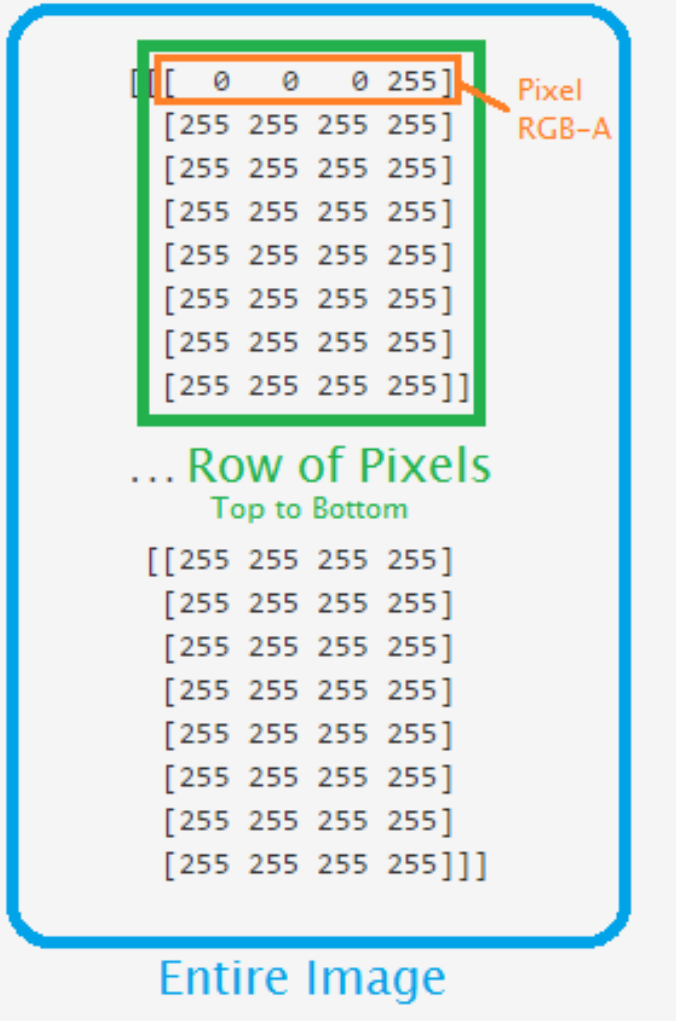
What is an image?

- A rectangular grid of **pixels**
- Each pixel has a color
- It can be represented using a two dimensional function f(x,y) where x and y are the spatial coordinates of each point in the image and the amplitude of f is the intensity of that point.



Pixels

- The word is a combination of *pix*, for picture, and *element*.
  - The word *pix* appeared in *Variety* magazine headlines in 1932, as an abbreviation for the word *pictures*, in reference to movies.
  - The concept of a "picture element" dates to the earliest days of television, the earliest publication of the term *picture element* itself was in *Wireless World* magazine in 1927
- **In computer, it is represented by a multi-channel array of values**



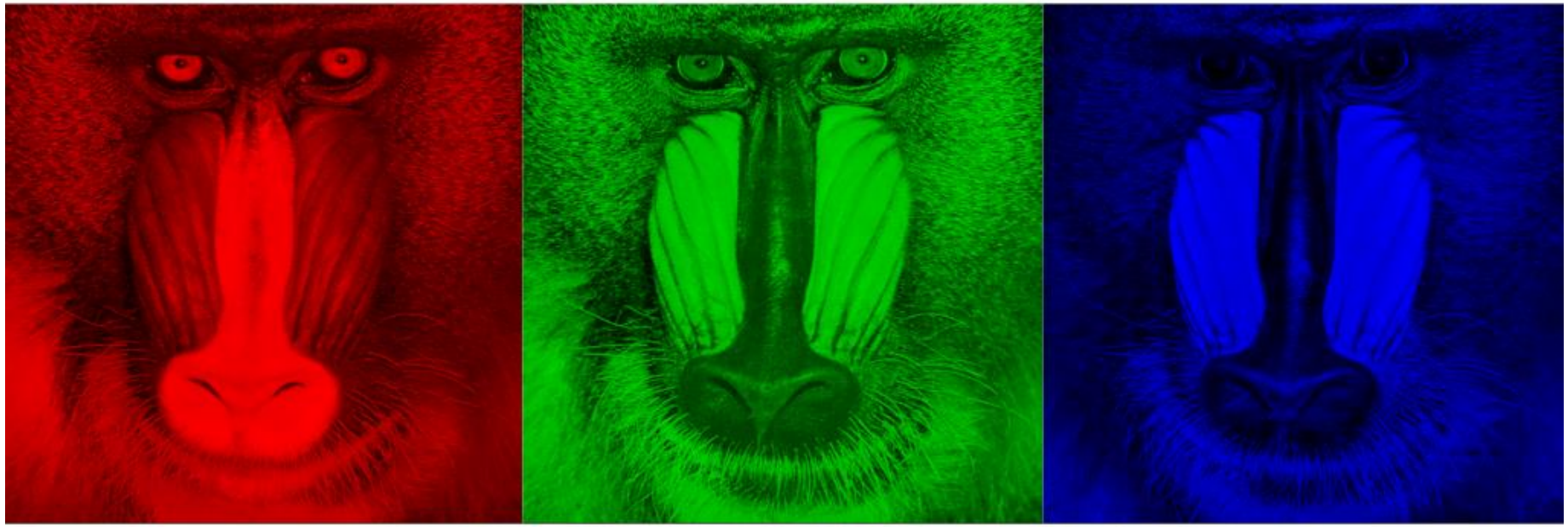
Gray vs. Color

- **Gray scale image:** Each pixel is shade of Gray, it is from 0 (black) to 255 (white), store by one byte (8 bits)
  - [83, 129, 250, 23, 0, 59, ….]

- **Color image:** Each pixel corresponds to 3 values, or channels, which represents **R**ed, **G**reen, **B**lue,
  - 8 bits (1Byte) for each
  - [(23,33,200), (50, 150, 20), (255, 0, 128), …]
  - **Interestingly, it is called RGB, but normally stored in the order of BGR (IMPORTANT when programming!)**

Colours-RGB

- RGB uses additive color mixing, because it describes what kind of *light* needs be *emitted* to produce a given color. RGB stores individual values
  - RGBA is RGB with an additional channel, alpha, for transparency.

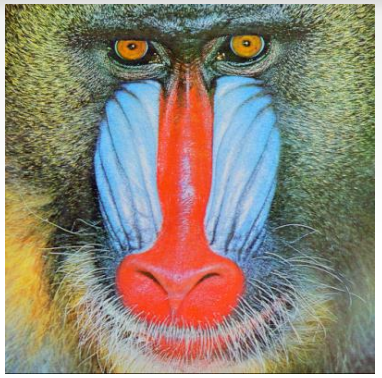


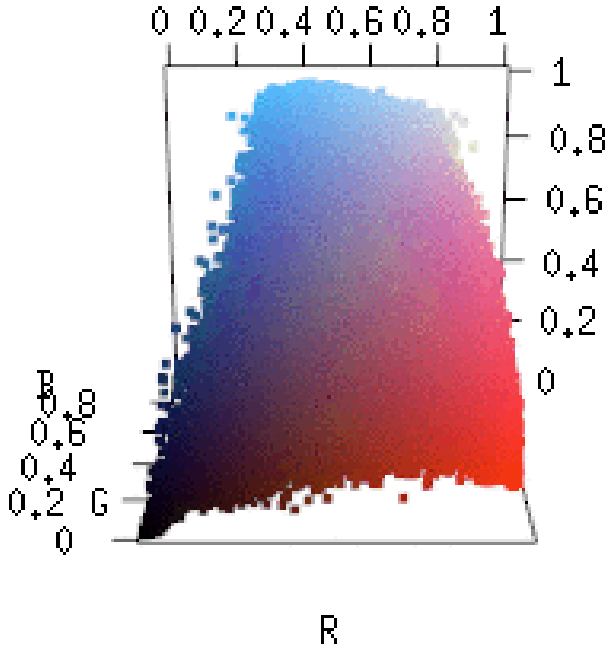
Colours-RGB

- **RGB is a 3D space**
- **No right answer for comparing two colors**
  - Sum of Squared Differences (SSD):

$$SSD(\boldsymbol{p}, \boldsymbol{q}) = \sum_{i \in Channels} [I^i(\boldsymbol{p}) - I^i(\boldsymbol{q})]^2$$

For RGB colors, you just add up the SSD in three channels





Transform color image to gray image

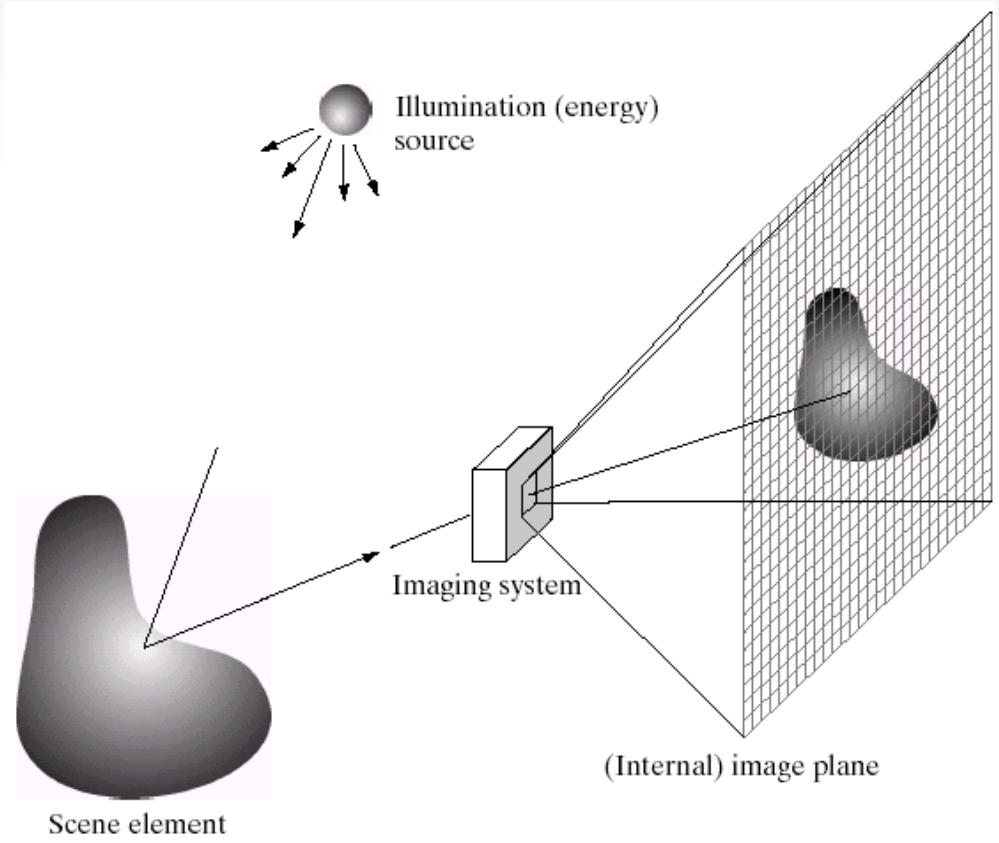
- **For every pixel, using luminosity method :**
  - Intensity = 0.21 R + 0.72 G + 0.07 B

Color

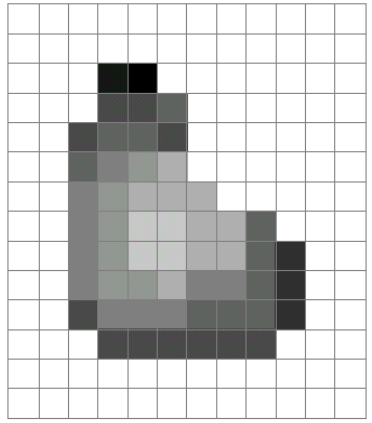
Average

Luminosity

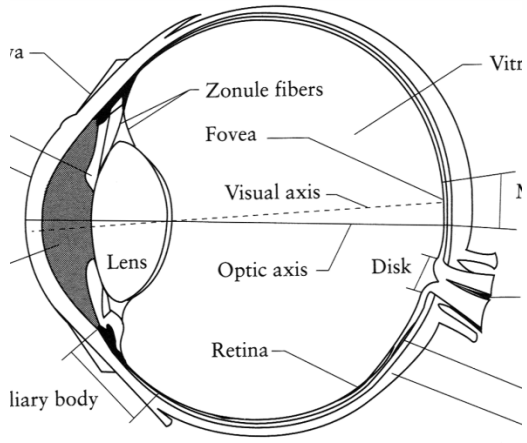
How do we get an image?



Film

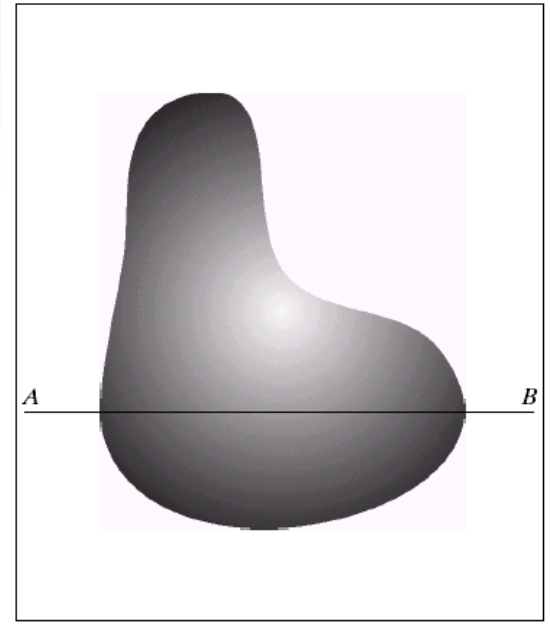


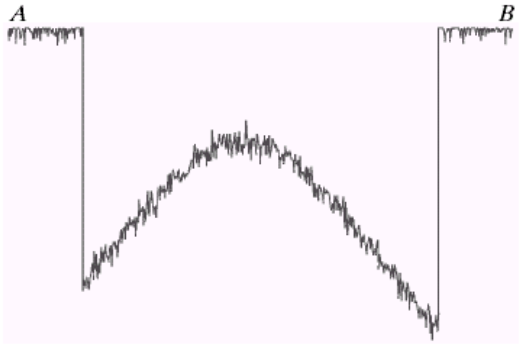
Digital Camera

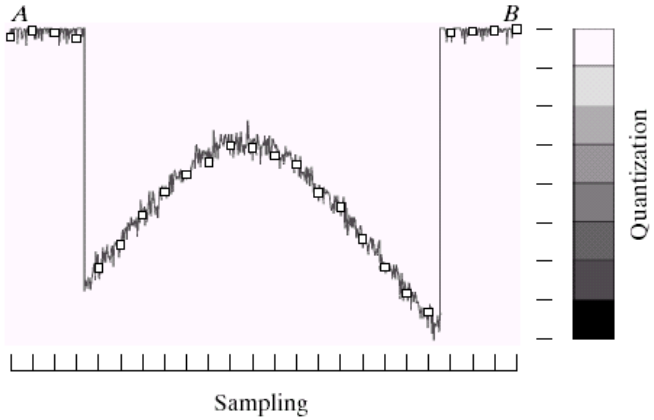


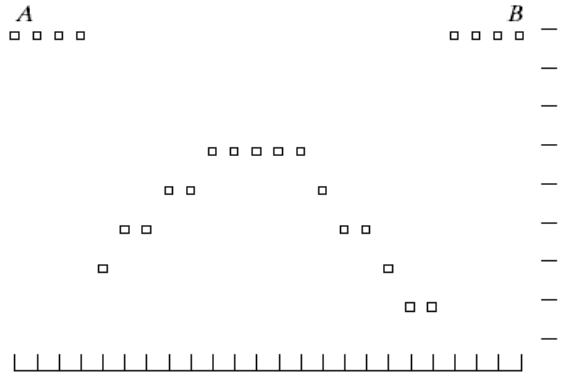
The Eye

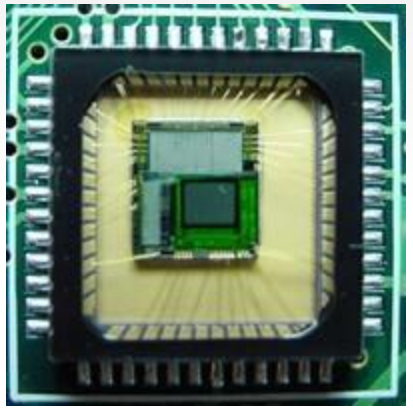
Sampling and Quantization











CMOS sensor

CCD sensor

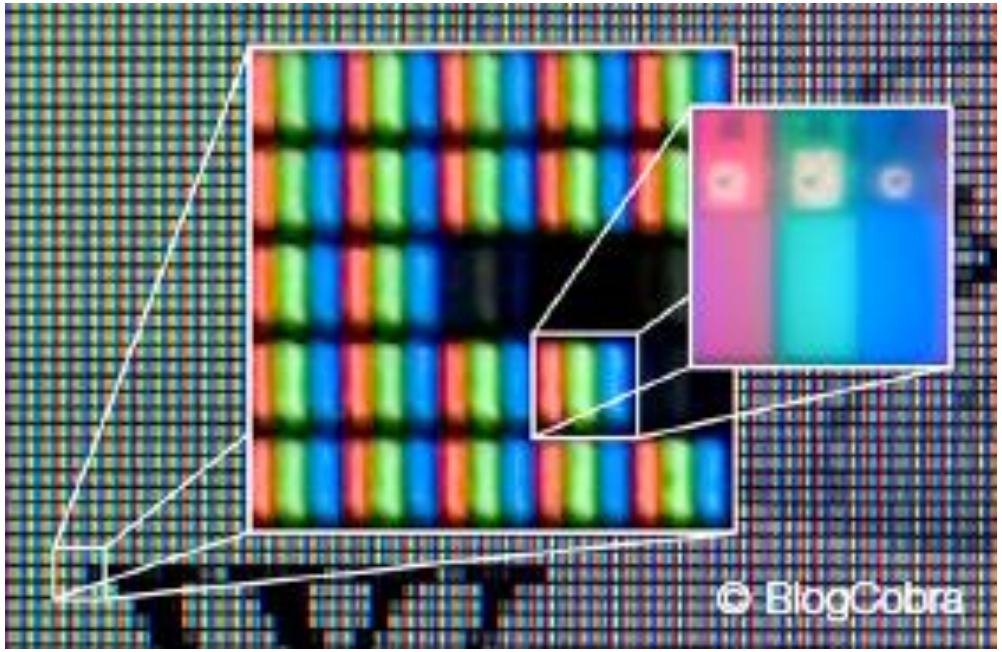
Quantization

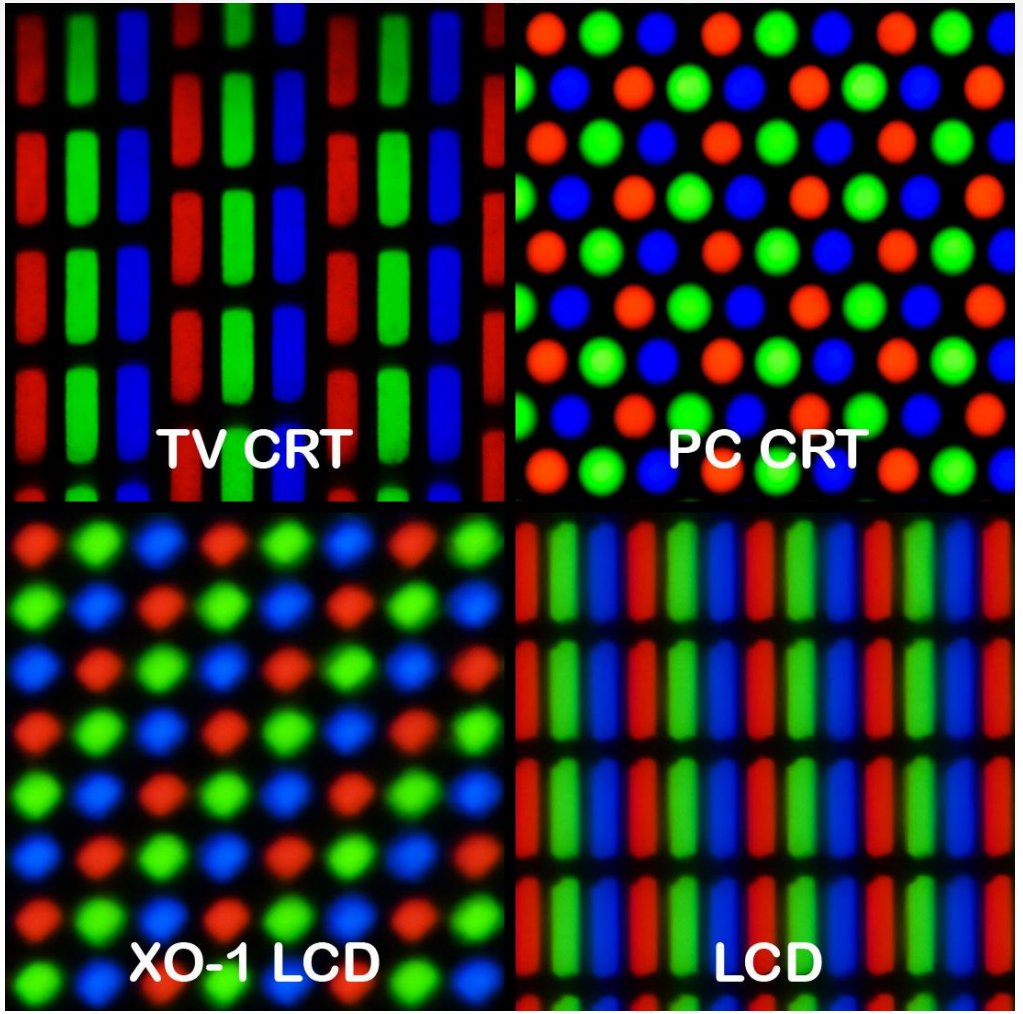
- The numbers of gray levels of the following eight images are 256, 128, 64, 32, 16, 8, 4, and 2, respectively.

Pixel art

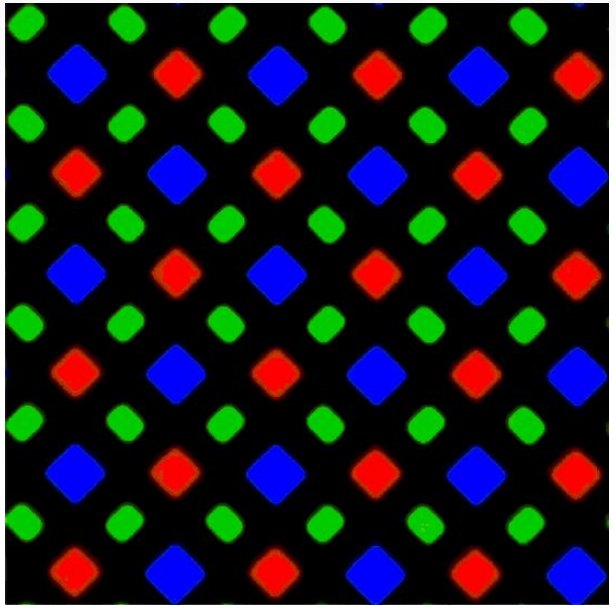
Pixel art

How do we show the pixels?

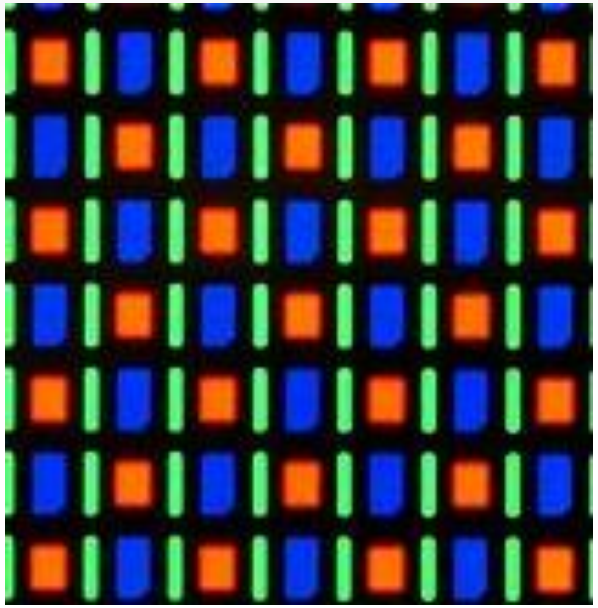




How do we show the pixels?

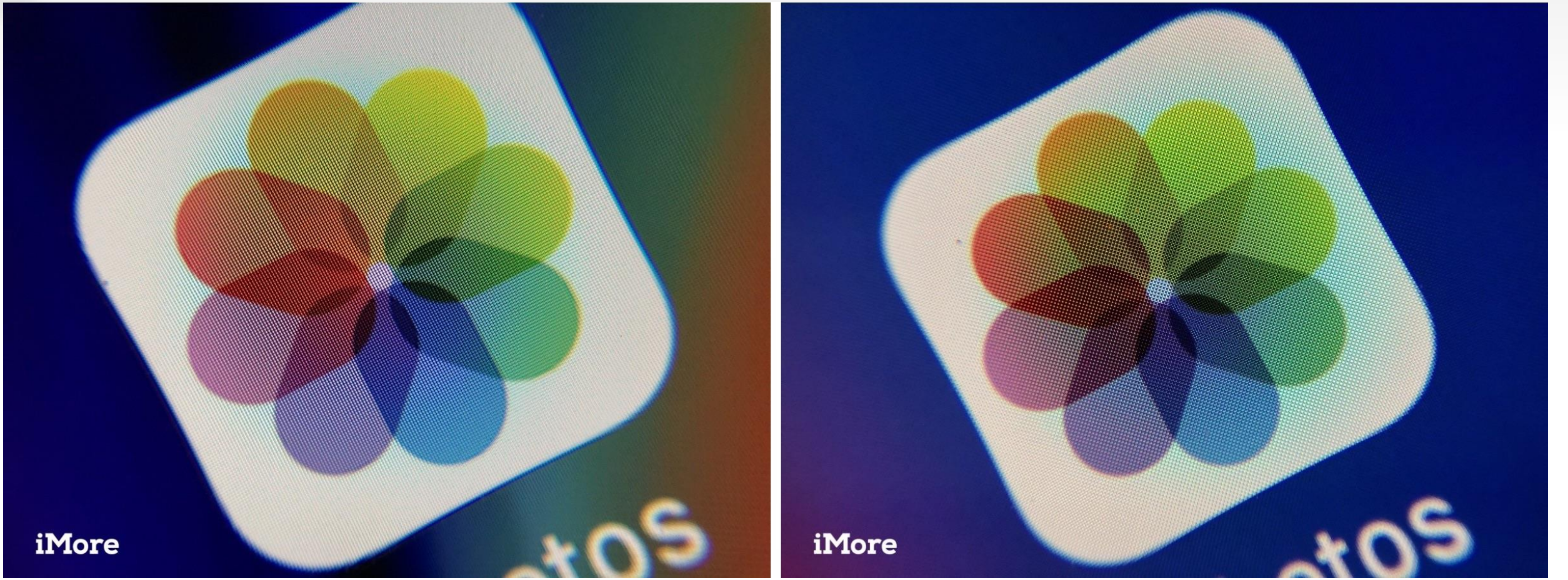


OLED screen on iPhone X
Diamond Sub-Pixels



AMOLED screen on the
Google Nexus One

How do we show the pixels?



iPhone 8 Plus's RGB stripe pixel layout (left) and iPhone X's diamond PenTile pixels (right).

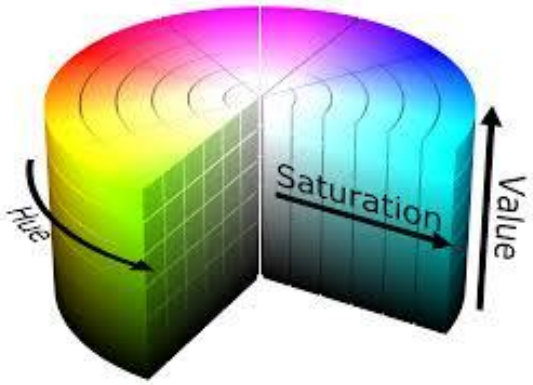
Other colour spaces - HSL

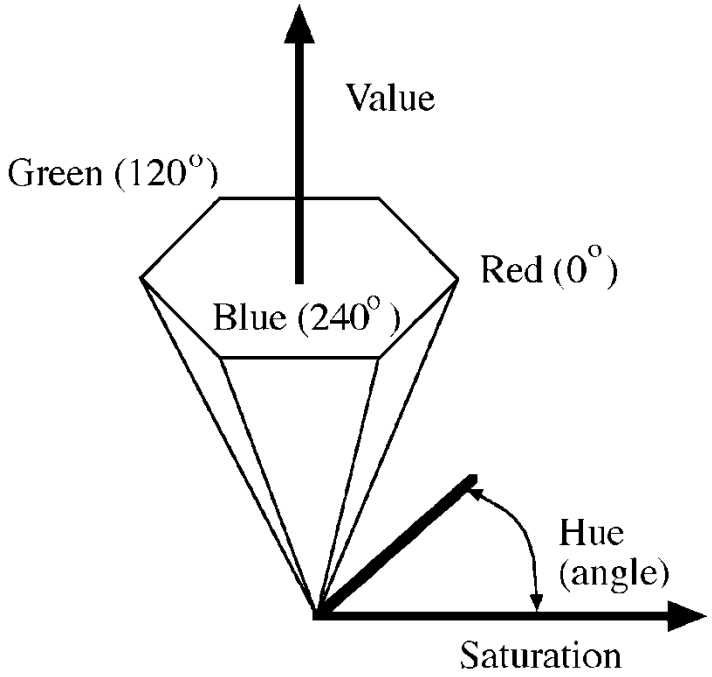
- **HSL** (**Hue, Saturation, Lightness**) and **HSV** (**Hue, Saturation, Value**) are two alternative representations of the RGB color model

$$V \leftarrow max(R, G, B)$$

$$S \leftarrow \begin{cases} \frac{V - min(R,G,B)}{V} & \text{if } V \neq 0 \\ 0 & \text{otherwise} \end{cases}$$

$$H \leftarrow \begin{cases} 60(G - B)/(V - min(R, G, B)) & \text{if } V = R \\ 120 + 60(B - R)/(V - min(R, G, B)) & \text{if } V = G \\ 240 + 60(R - G)/(V - min(R, G, B)) & \text{if } V = B \end{cases}$$

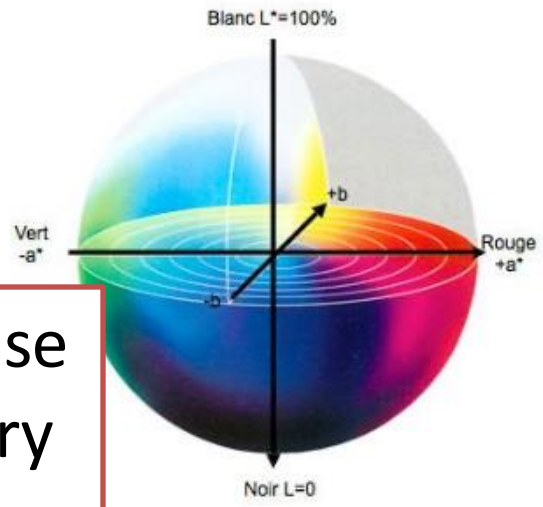




If H<0 then H←H+360 . On output 0≤V≤1, 0≤S≤1, 0≤H≤360, The values are then converted to the destination data type: 8-bit images: V←255V,S←255S,H←H/2(to fit to 0 to 255)

Other colour spaces - Lab

- The ***Lab*** color space describes mathematically all perceivable colors in the three dimensions
  - *L* for lightness and *a* and *b* for the color components green–red and blue–yellow.



Lab color space

OpenCV supports these colour conversion very well!

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \leftarrow \begin{bmatrix} 0.412453 & 0.357580 & 0.180423 \\ 0.212671 & 0.715160 & 0.072169 \\ 0.019334 & 0.119193 & 0.950227 \end{bmatrix} \cdot \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

$$X \leftarrow X/X_n, \text{where} X_n = 0.950456$$

$$Z \leftarrow Z/Z_n, \text{where} Z_n = 1.088754$$

$$L \leftarrow \begin{cases} 116 * Y^{1/3} - 16 & \text{for } Y > 0.008856 \\ 903.3 * Y & \text{for } Y \le 0.008856 \end{cases}$$

$$a \leftarrow 500(f(X) - f(Y)) + delta$$

$$b \leftarrow 200(f(Y) - f(Z)) + delta$$

$$f(t) = \begin{cases} t^{1/3} & \text{for } t > 0.008856 \\ 7.787t + 16/116 & \text{for } t \le 0.008856 \end{cases}$$

$$delta = \begin{cases} 128 & \text{for 8-bit images} \\ 0 & \text{for floating-point images} \end{cases}$$

8-bit images: L←L∗255/100,a←a+128,b←b+128

Contrast Ratio

- **Brightness discrimination experiment**
  - Can you see the internal rectangle?

Visibility threshold

$K = \Delta I / I \approx 1\sim2\%$

Weber fraction

I is luminance

Same blue channel difference value:

Contrast Ratio

- Luminance ratio between two successive quantization levels should be at visibility threshold
- K is Weber's fraction

$$\frac{I_{max}}{I_{min}} = (1+K)^{N-1}$$

- For K is between 0.01~0.02, N = 256, $\frac{I_{max}}{I_{min}}$=13~156
- Typical display contrast ratio (max : min)
  - Modern flat panel display in dark room 1000:1
  - Cathode ray tubes (CRTs) 100:1
  - Print on paper 10:1
- If we use more bits for a channel, say 9 bits, N = 512

$\frac{I_{max}}{I_{min}}$=24813 when K = 0.02

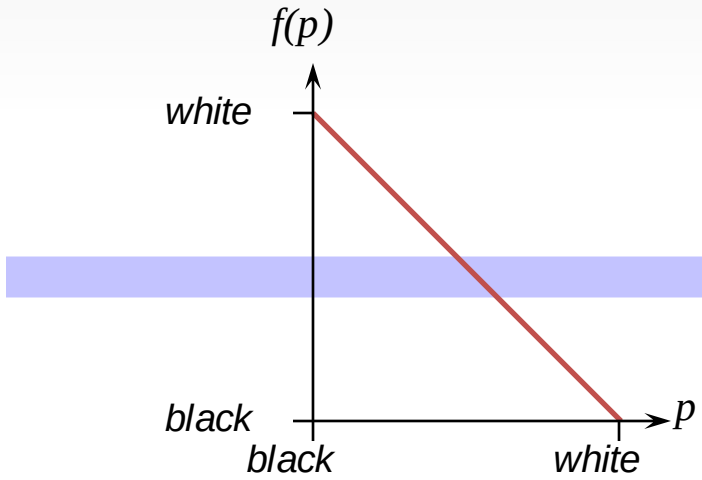
Point operations

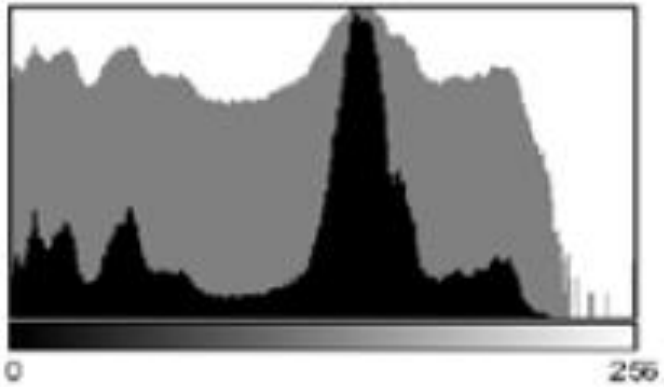
- **Each pixel's value is modified**
- **the modification function only takes that pixel's value into account**

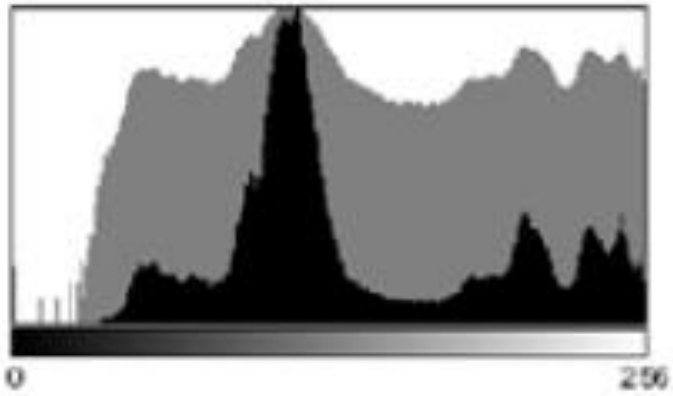
$$p'(i, j) = f\{p(i, j)\}$$

  - where $p(i,j)$ is the value of the pixel and $p'(i,j)$ is the modified value
  - the modification function, $f(p)$, can perform any operation that maps one intensity value to another

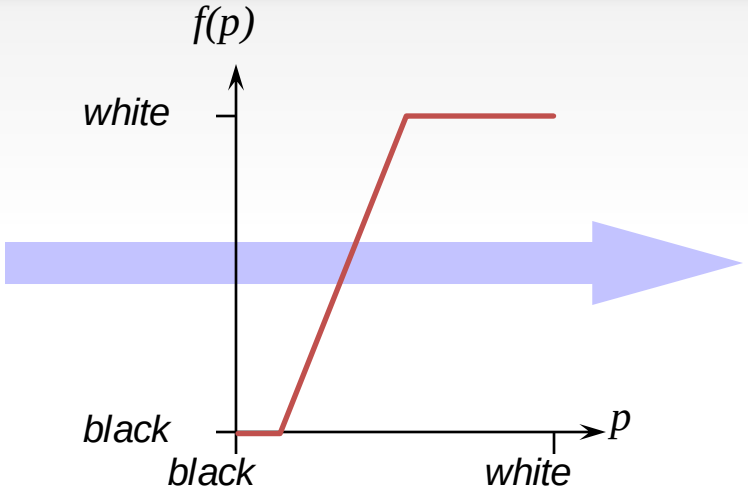
Point processing: inverting an image

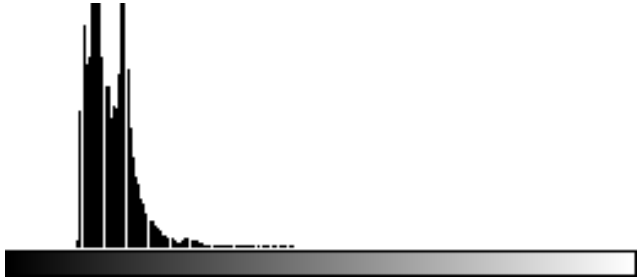




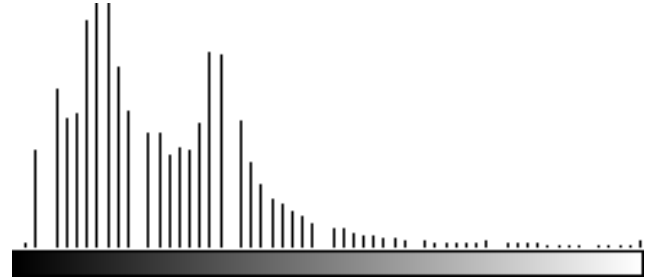


Point processing: improving contrast





dark histogram

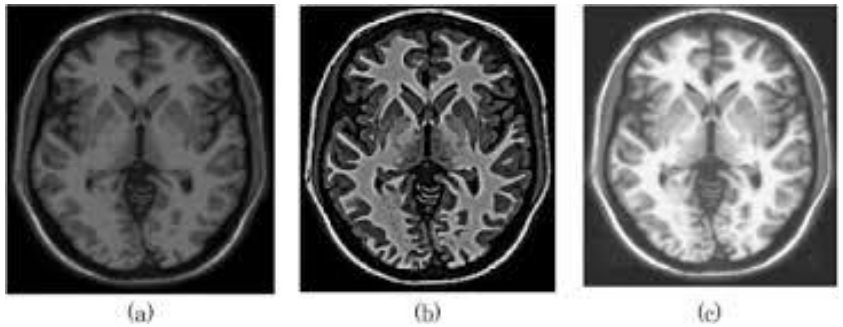


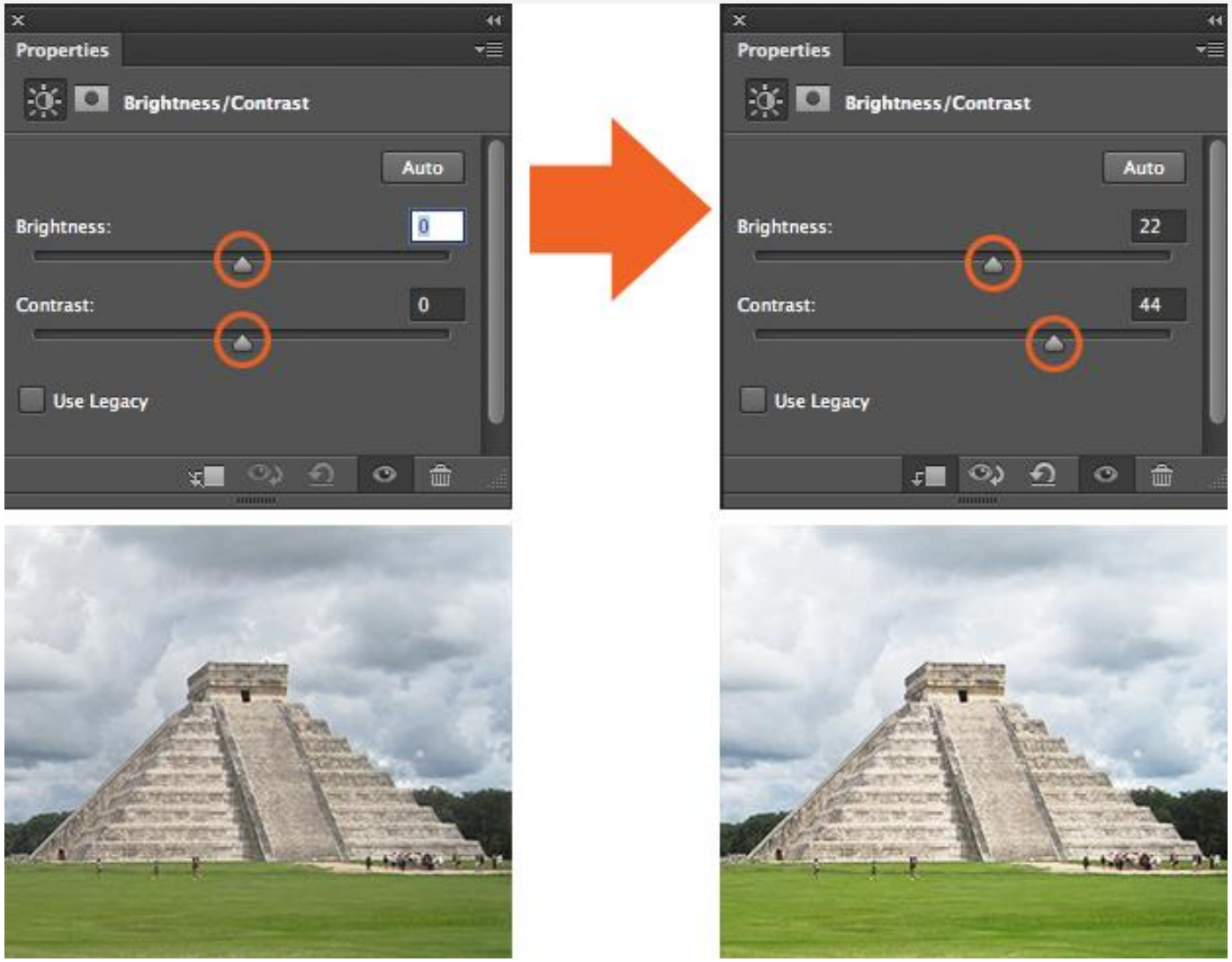
improved histogram

Improve visual quality

- Combine these operations properly, you will get a much better visual quality

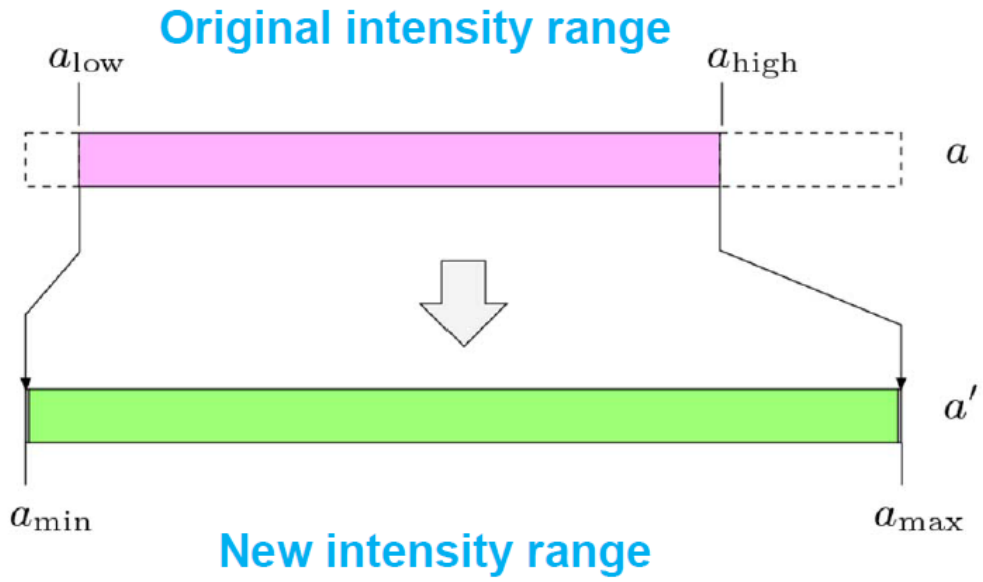
- Not only useful for entertainment! It has many important applications, like medical image enhancement





Automatic Contrast Adjustment

- **Point operation that modifies pixel intensities such that available range of values is fully covered**
- **Algorithm:**
  - Find the highest and lowest pixel intensities
  - Linear stretching of intensity range



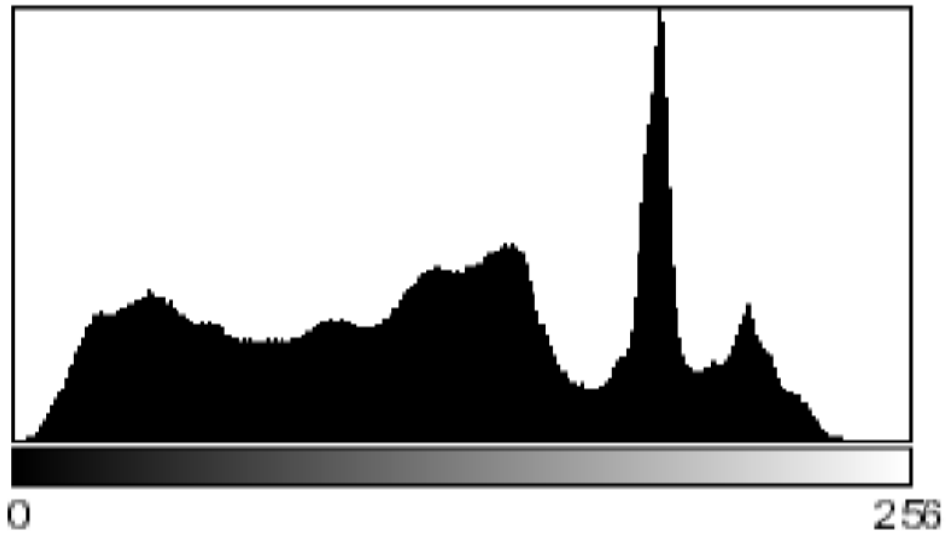
$$f_{\text{ac}}(a) = a_{\text{min}} + (a - a_{\text{low}}) \cdot \frac{a_{\text{max}} - a_{\text{min}}}{a_{\text{high}} - a_{\text{low}}}$$

If $a_{min} = 0$ and $a_{max} = 255$

$$f_{\text{ac}}(a) = (a - a_{\text{low}}) \cdot \frac{255}{a_{\text{high}} - a_{\text{low}}}$$

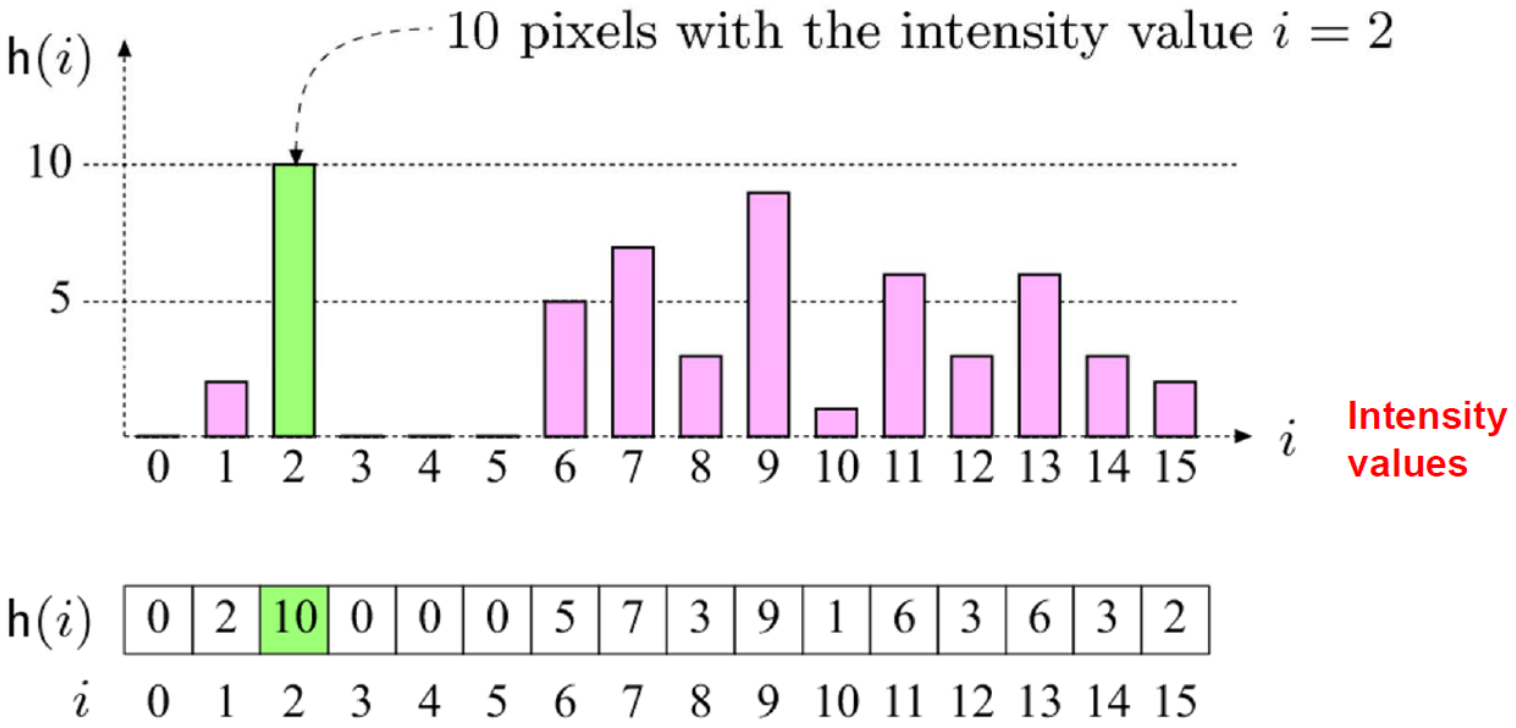
Histograms

- **Histograms plots how many times (frequency) each intensity value in image occurs.**
  - Example: An image has 256 distinct gray levels (8 bits), the **Histogram** (right) shows frequencies for the 256 gray levels



Histograms

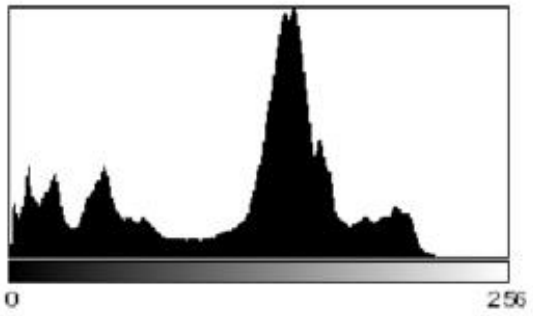
- ***Maximum Intensity Value*** **= 16, 10 pixels have value = 2**

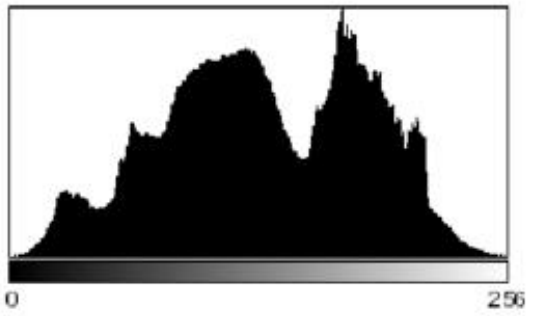


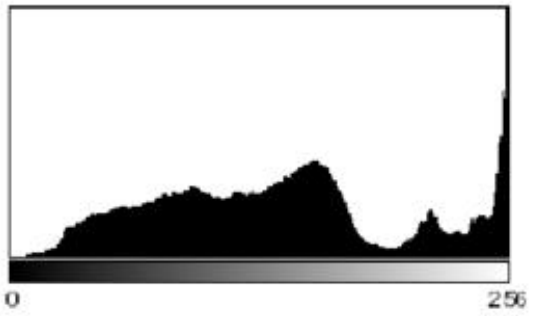
Detecting Bad Exposure

- **Exposure:**

  Are intensity values spread (good) out or bunched up (bad)?

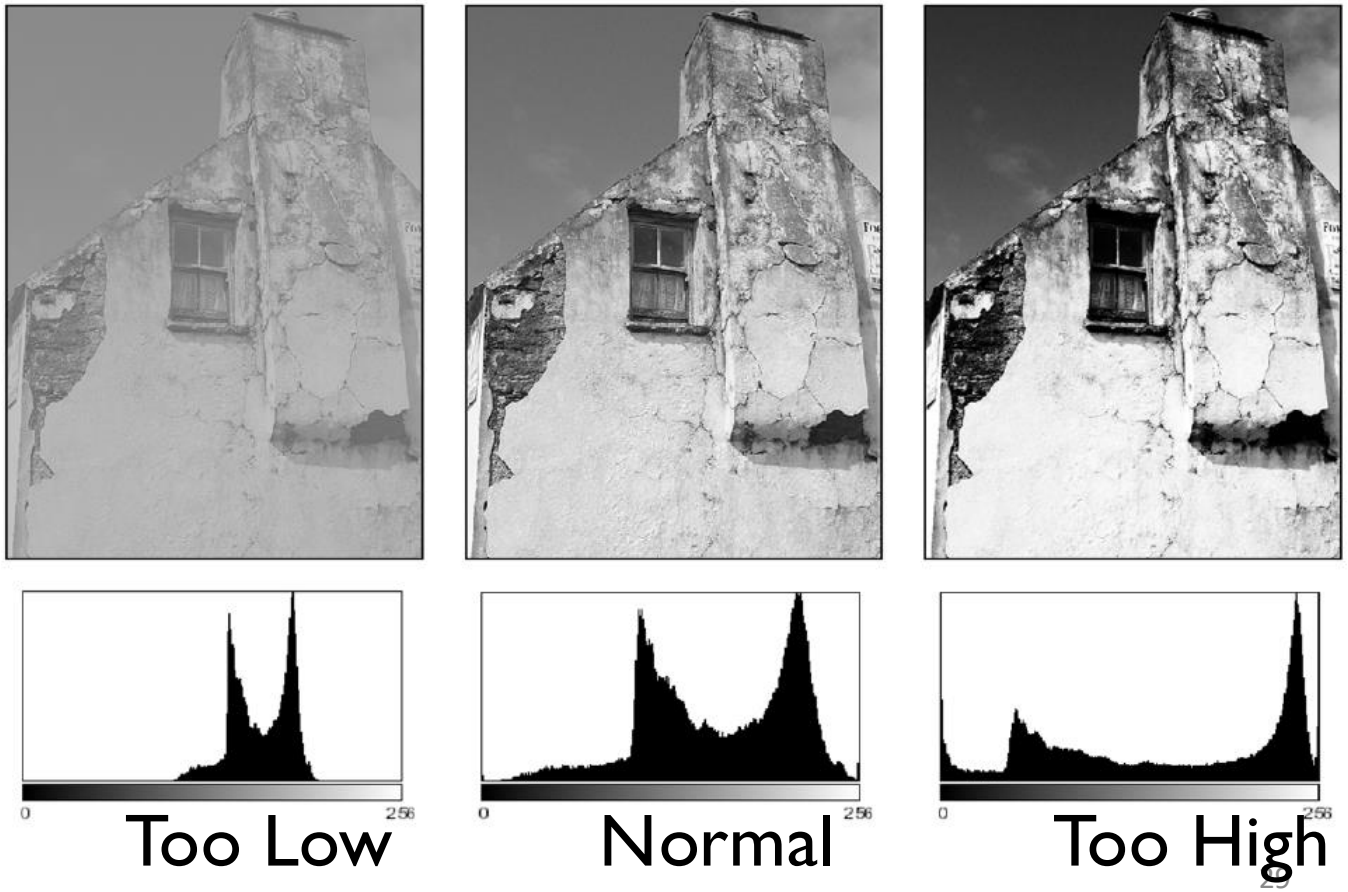






Histograms and Contrast

- **The contrast of a image indicates how easily objects can be distinguished**
  - **High contrast image:** many distinct intensity values
  - **Low contrast:** image uses few intensity values



Histograms and Dynamic Range

- **Dynamic Range: Number of distinct pixels in image**

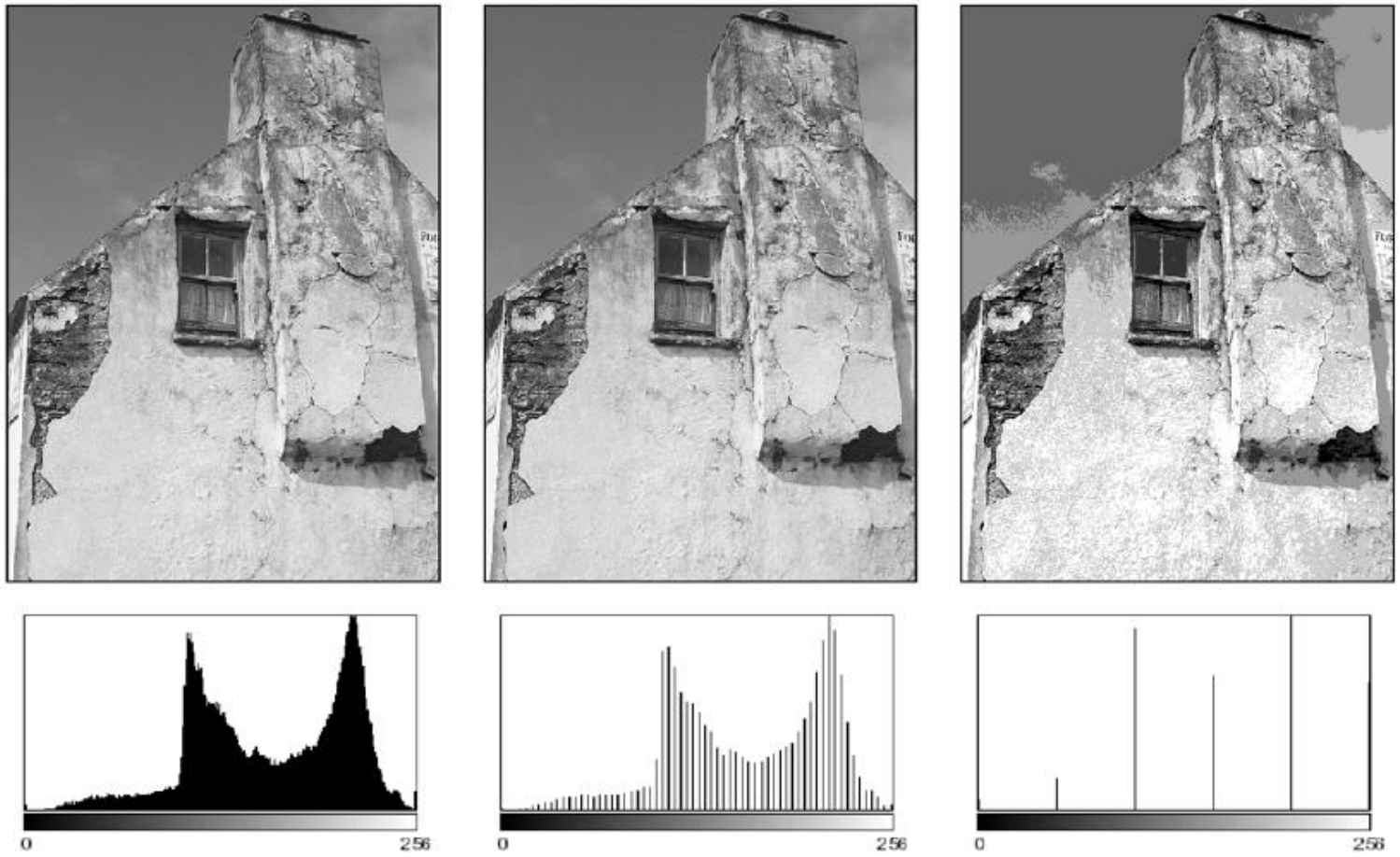
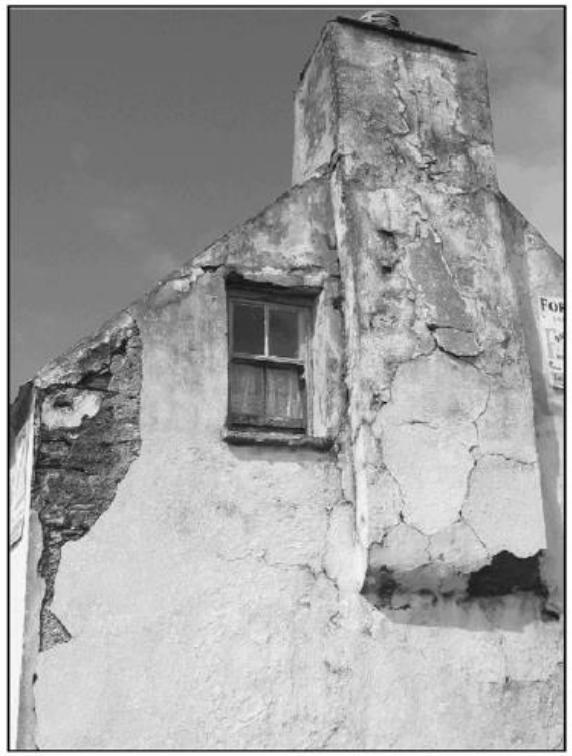
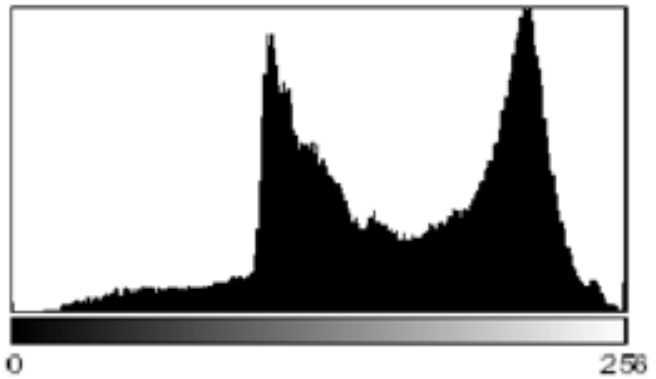


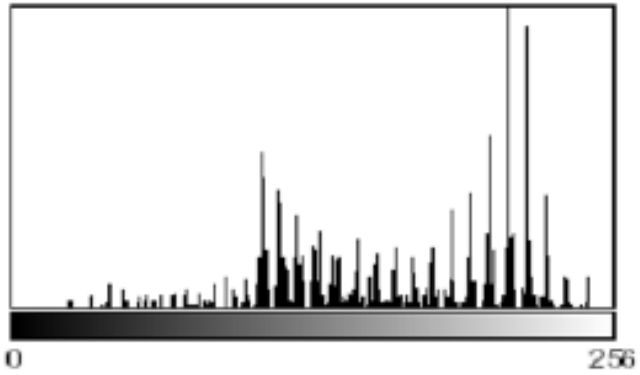
Image Defects and Histograms

- **Histograms show impact of image compression**
  - Example: in GIF compression, dynamic range is reduced to only few intensities (quantization)





Original Histogram

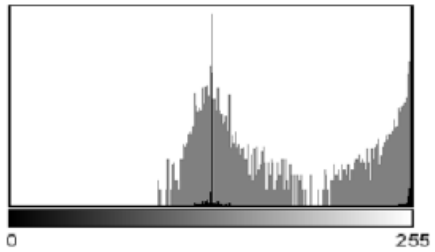


Histogram after GIF conversion

Image Defects and Histograms

- **Histograms show impact of image compression**
  - Example: Effect of JPEG compression on line graphics:

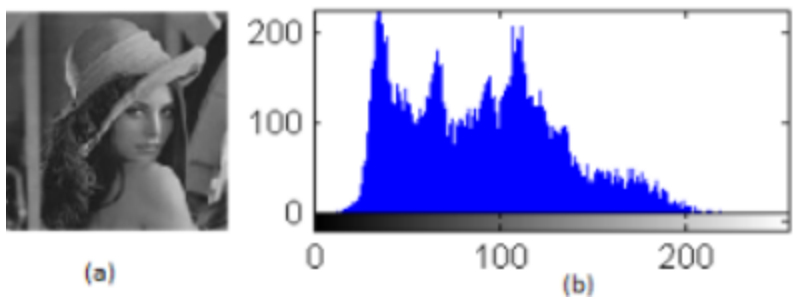
Original histogram has only 2 intensities (gray and white)

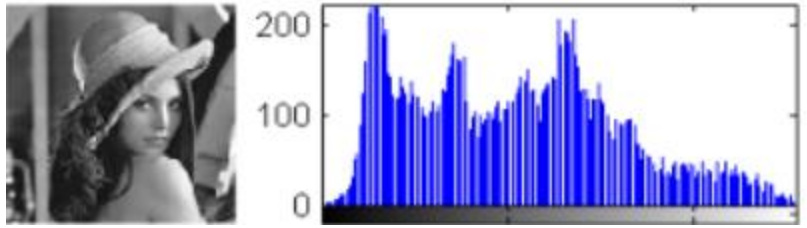


JPEG image appears dirty, fuzzy and blurred Its Histogram contains gray values not in original

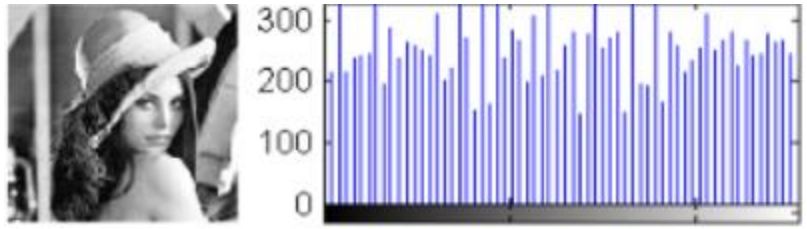
Histogram Equalization

- **Histogram equalization** is a method in image processing of contrast adjustment using the image's histogram.





Automatic Contrast Adjustment



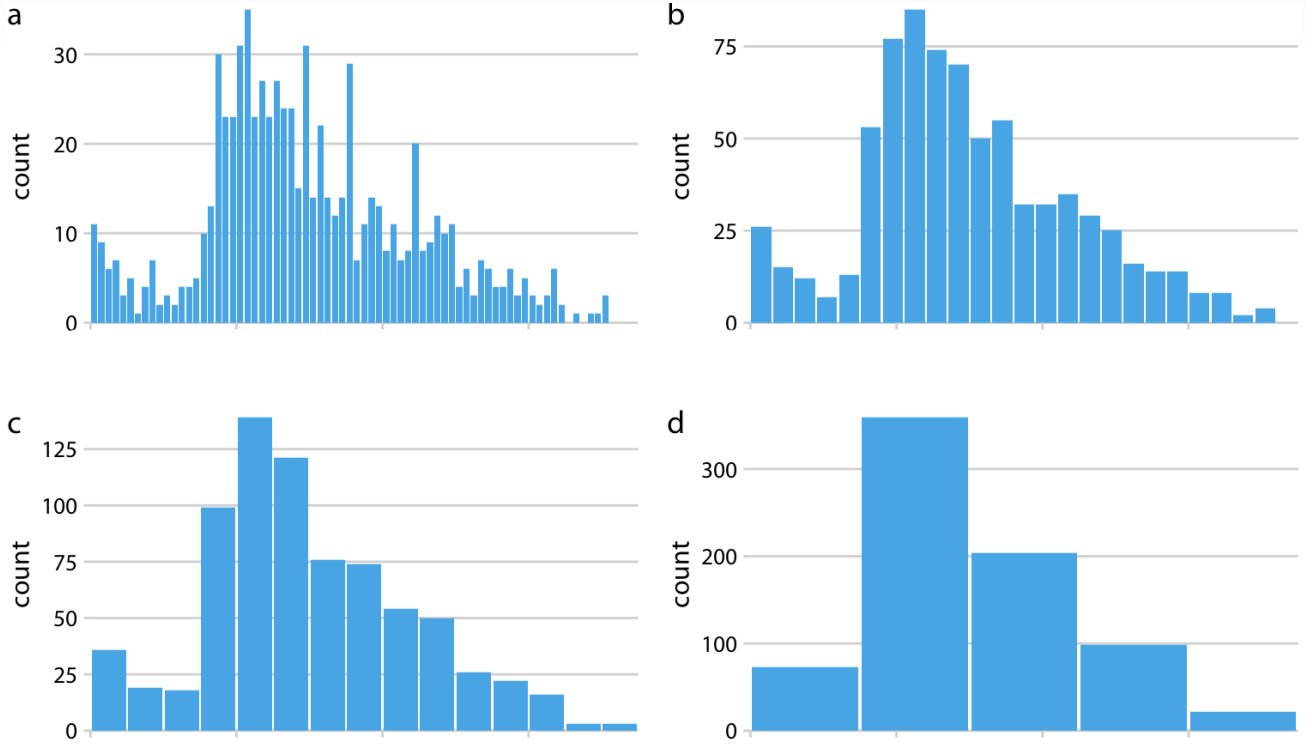
Histogram Equalization

The purposes of Histogram Equalization:

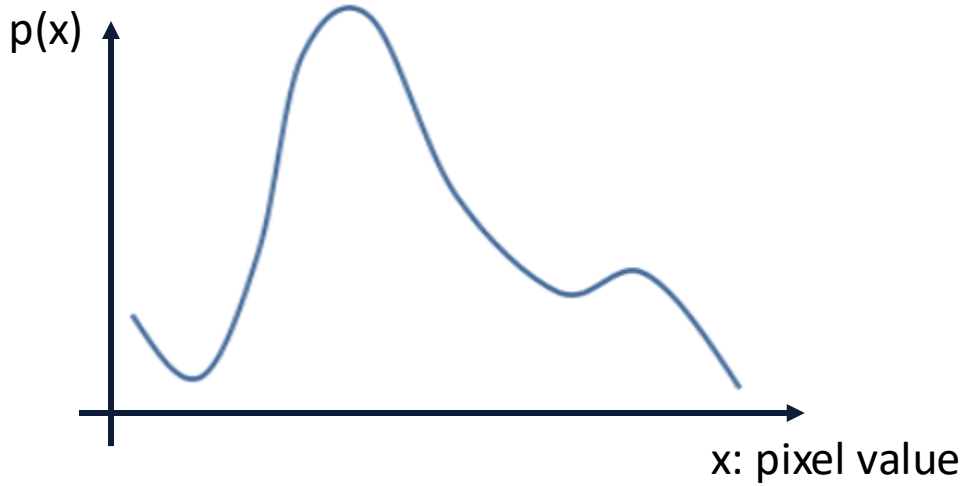
--to equally make use of all available gray levels in the dynamic range

Histogram Equalization

- A mathematical view of a histogram:

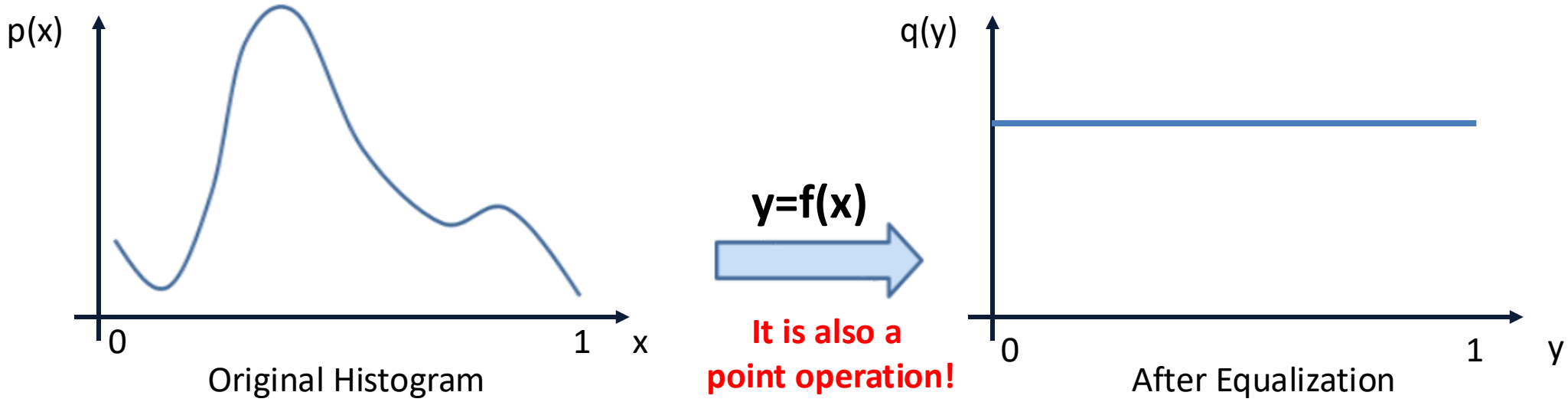


Think of the histogram as a
Probability Density Function



The Objective

- Assume the pixel values are continuous in the normalized range of (0, 1), and a mapping function y=f(x) maps x to y also in the same range.
- p(x) and q(y) are the probability density functions for x and y



What is f(x) ??

Histogram Equalization

$\int_0^1 p(x)dx = \int_0^1 q(y)dy = 1$ (p(x), q(y)-Probability Density Functions)

- Also assume all pixels within the gray scale interval of the input image are mapped to the corresponding range of the output :

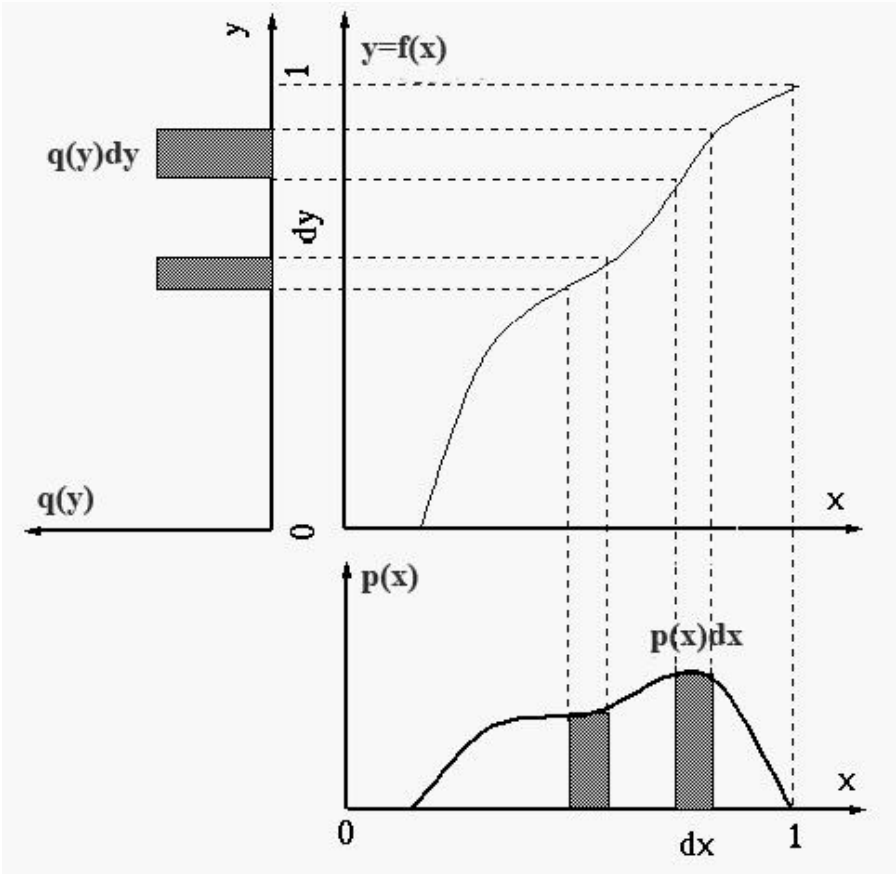
$$\mathrm{q}(y)dy = p(x)dx$$

- For the distribution function q(y), it should be equalized, which needs to be a constant 1, so:

$$q(y) = 1$$

$$\mathrm{q}(y)dy = dy = p(x)dx,$$

$$or \quad \frac{dy}{dx} = p(x)$$



Histogram Equalization

- Integrating both sides, we get the mapping function y = f(x) for the histogram equalization:

$$y = f(x) = \int_0^x p(u)du$$

- We can use P(x) to represent the **cumulative distribution** of the gray levels of the input image:

$$P(x) = \int_0^x p(u)du\,, \qquad P(0) = 0, \qquad P(1) = 1$$

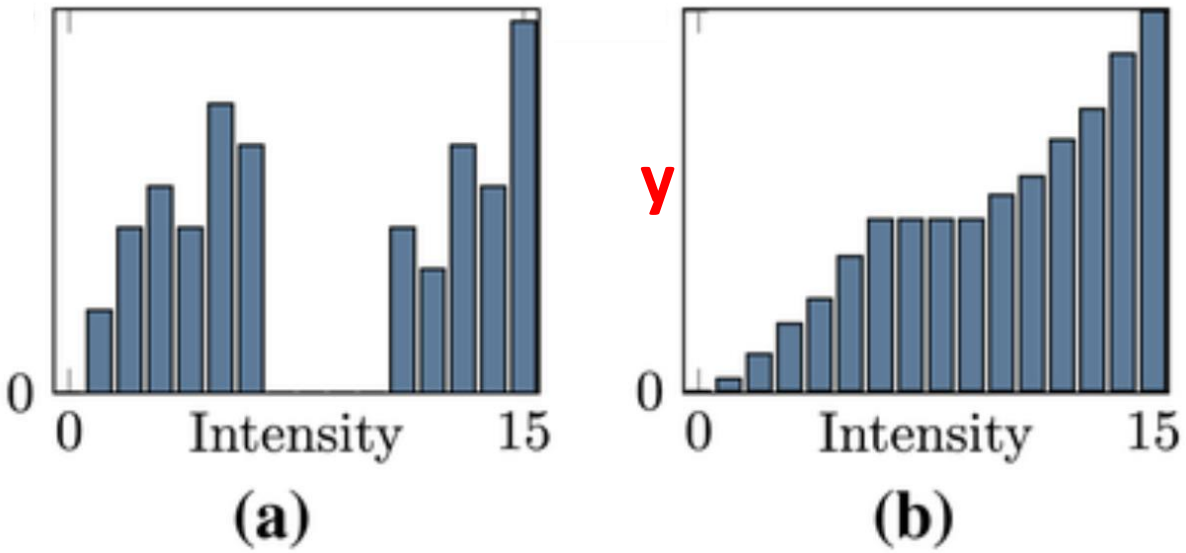
P(x) is the function y = f(x) we are looking for! Given an original pixel value x, the cumulative probability of x (the probability of the pixel value being less than x) in the original histogram function p(x) will be the new value y for that pixel!

Histogram Equalization

- **We use the cumulative distribution function's value at x to as the new value y!**

The Original Histogram (Probability Density)

x: intensity value



The Cumulative Distribution (Probability Density)

$$P(x) = \int_0^x p(u)du$$

Histogram Equalization

Implementation -- back projection

- Consider a discrete grayscale image $\{x\}$ and let n_i be the number of occurrences of gray level n. The probability of an occurrence of a pixel of level n in the image is

$$p_n = \frac{\text{number of pixels with intensity } n}{\text{total number of pixels}} \qquad n = 0, 1, ..., L-1.$$

$$g_{i,j} = \text{floor}((L-1)\sum_{n=0}^{f_{i,j}} p_n)$$

In implementation, we add an **extra normalization step** to make full use of 0-255

Histogram Equalization

Original Image

$$\begin{bmatrix} 52 & 55 & 61 & 66 & 70 & 61 & 64 & 73 \\ 63 & 59 & 55 & 90 & 109 & 85 & 69 & 72 \\ 62 & 59 & 68 & 113 & 144 & 104 & 66 & 73 \\ 63 & 58 & 71 & 122 & 154 & 106 & 70 & 69 \\ 67 & 61 & 68 & 104 & 126 & 88 & 68 & 70 \\ 79 & 65 & 60 & 70 & 77 & 68 & 58 & 75 \\ 85 & 71 & 64 & 59 & 55 & 61 & 65 & 83 \\ 87 & 79 & 69 & 68 & 65 & 76 & 78 & 94 \end{bmatrix}$$

Value	Count	Value	Count	Value	Count	Value	Count	Value	Count
52	1	64	2	72	1	85	2	113	1
55	3	65	3	73	2	87	1	122	1
58	2	66	2	75	1	88	1	126	1
59	3	67	1	76	1	90	1	144	1
60	1	68	5	77	1	94	1	154	1
						104	2		
						106	1		
						109	1		

Histogram

Cannot guarantee the utilization of the intensity level 0.
E.g. If L = 256, CDF(min)/Total > 0.004

Implementation using the CDF table

$$g_{i,j} = floor\left(\frac{CDF(f_{i,j})}{\text{Total Pixel Num}} \times (L-1)\right)$$

$$g_{i,j} = floor\left(\frac{(CDF(f_{i,j}) - CDF_{min})}{(CDF_{max} - CDF_{min})} \times (L-1)\right)$$

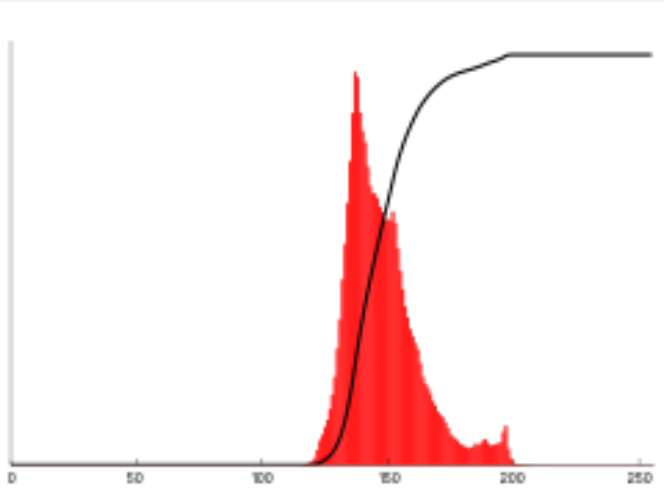
Value	CDF	Value	CDF	Value	CDF	Value	CDF	Value	CDF
52	1	64	19	72	40	85	51	113	60
55	4	65	22	73	42	87	52	122	61
58	6	66	24	75	43	88	53	126	62
59	9	67	25	76	44	90	54	144	63
60	10	68	30	77	45	94	55	154	64
61	14	69	33	78	46	104	57		
62	15	70	37	79	48	106	58		
63	17	71	39	83	49	109	59		

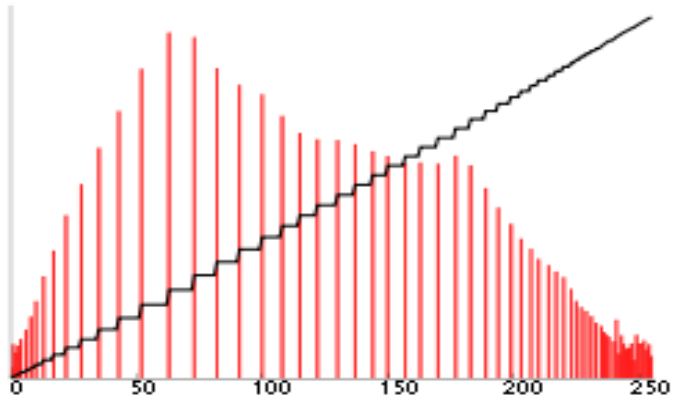
Cumulative Distribution Function (CDF)

$$\begin{bmatrix} 0 & 12 & 53 & 93 & 146 & 53 & 73 & 166 \\ 65 & 32 & 12 & 215 & 235 & 202 & 130 & 158 \\ 57 & 32 & 117 & 239 & 251 & 227 & 93 & 166 \\ 65 & 20 & 154 & 243 & 255 & 231 & 146 & 130 \\ 97 & 53 & 117 & 227 & 247 & 210 & 117 & 146 \\ 190 & 85 & 36 & 146 & 178 & 117 & 20 & 170 \\ 202 & 154 & 73 & 32 & 12 & 53 & 85 & 194 \\ 206 & 190 & 130 & 117 & 85 & 174 & 182 & 219 \end{bmatrix}$$

Result

Histogram Equalization



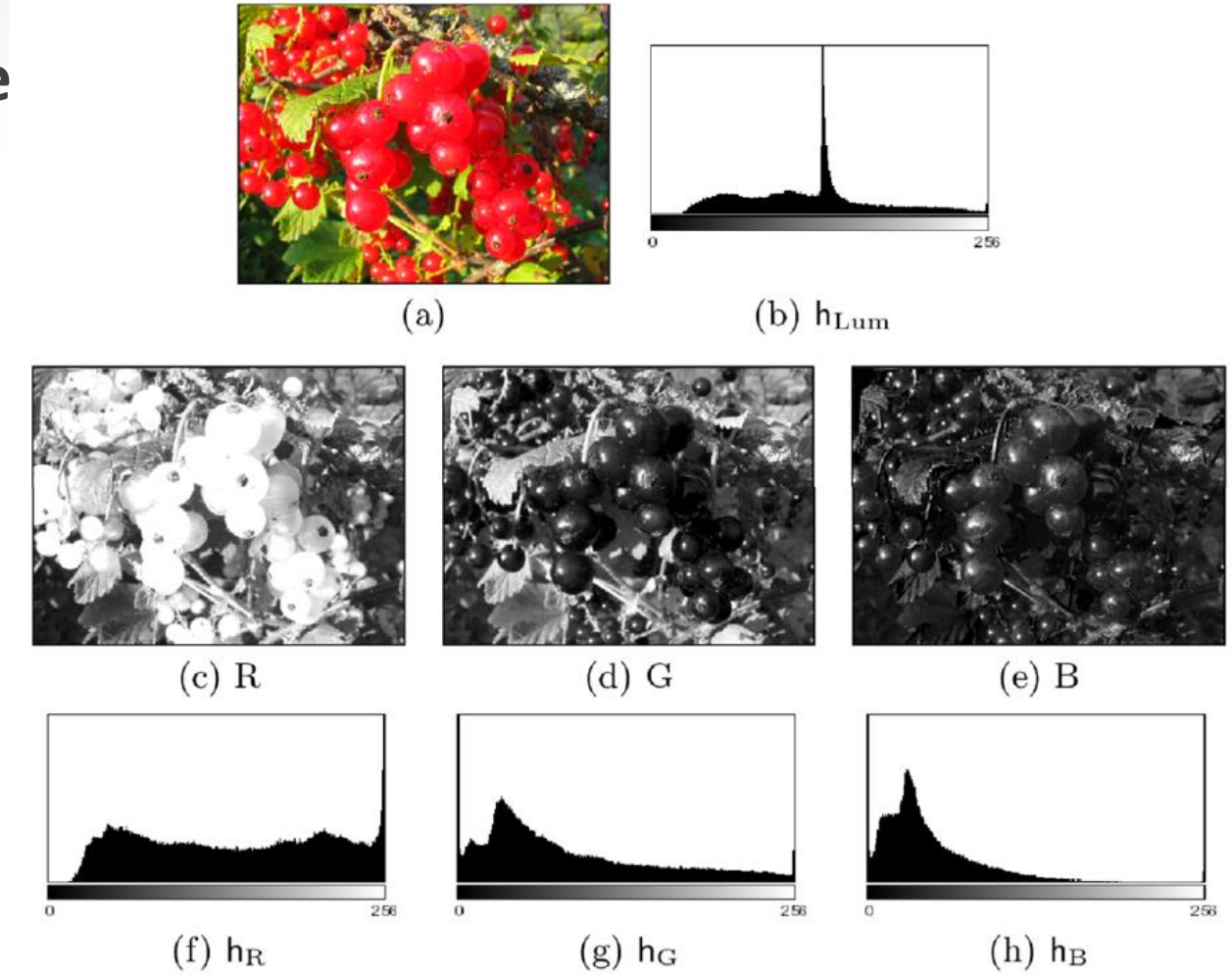


Color Image Histograms

1. Intensity histogram:
   - Convert color image to gray scale

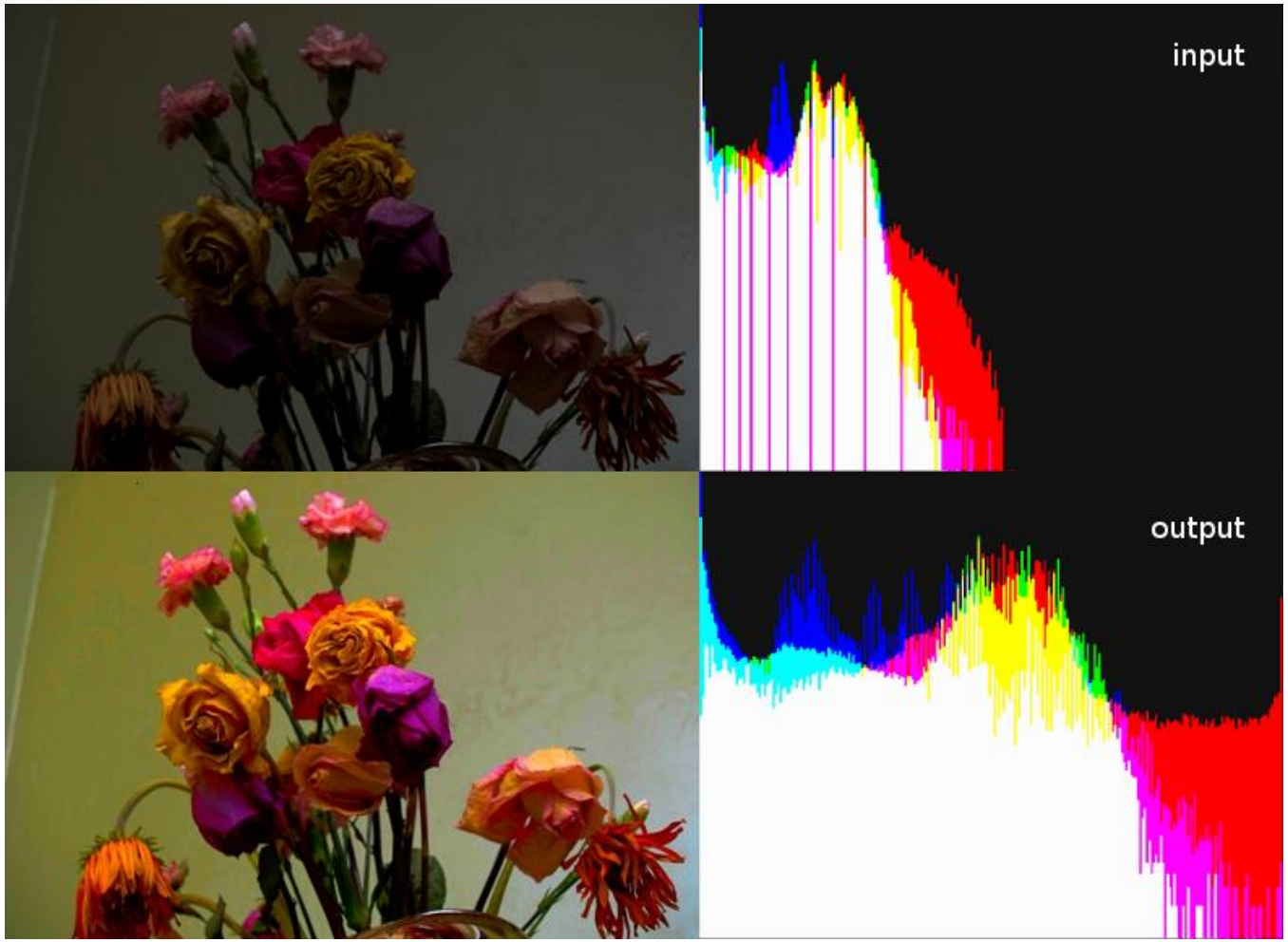
2. Individual Color
   - Channel Histograms:
     3 histograms (R,G,B)

All the operations on histograms can be done on each channel



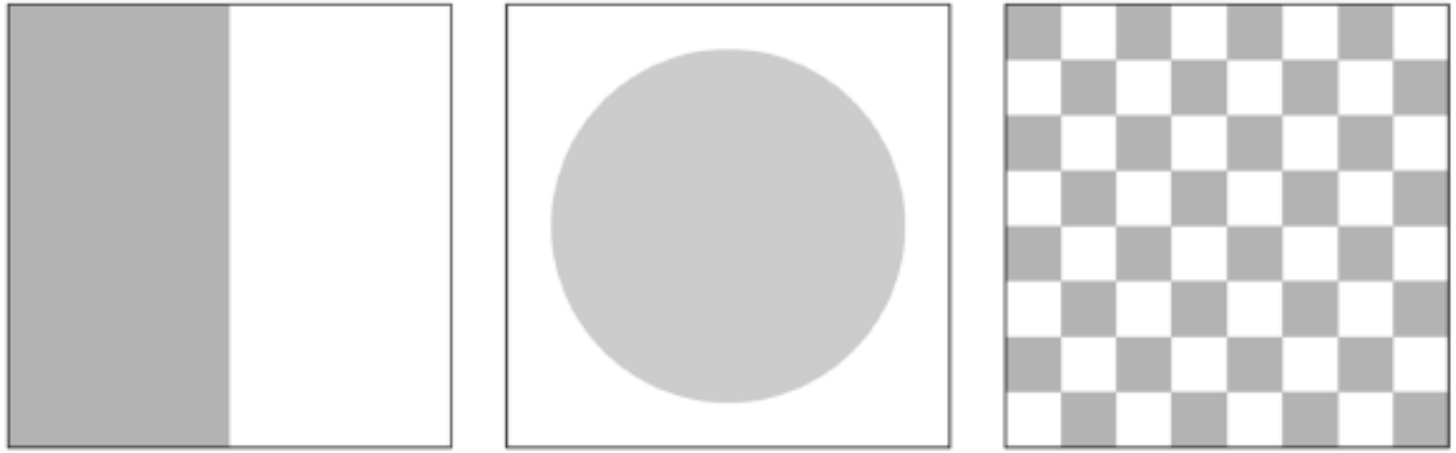
Color Image Histograms

- Color image histogram equalization:



Histograms

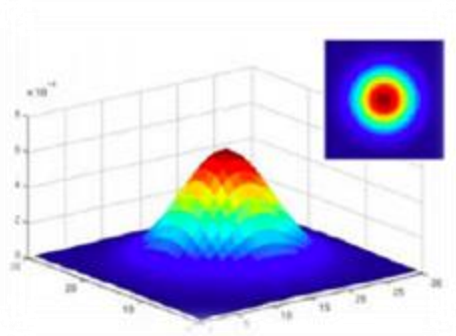
- Different images can have same histogram
- 3 images below have same histogram

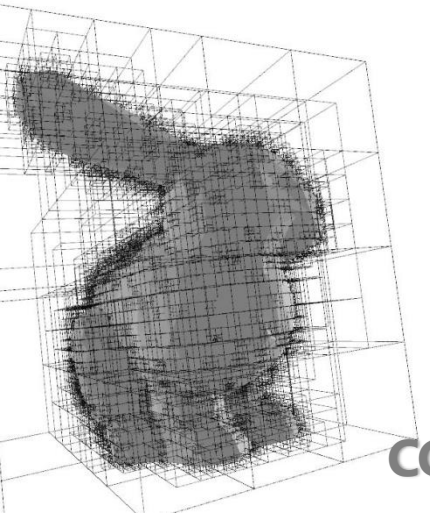


Histograms only have statistical information

- Half of pixels are gray, half are white
  - Same histogram = same statistics
  - Can we reconstruct image from histogram? No!

Image-Based Computer Graphics





Fundamentals of Image Processing

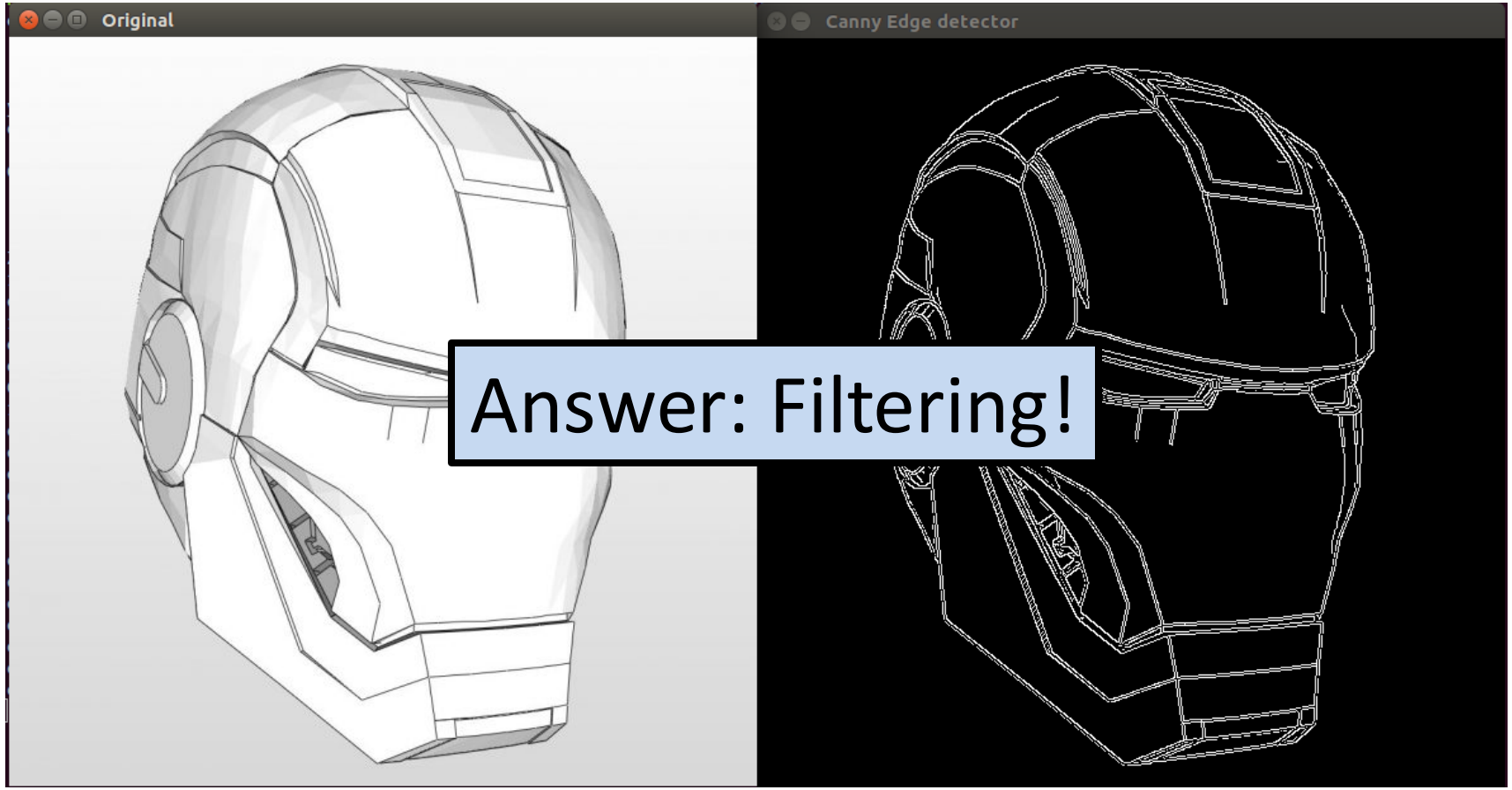
Morphology, Filtering and Pyramids

CGRA 352

How to remove the noises?

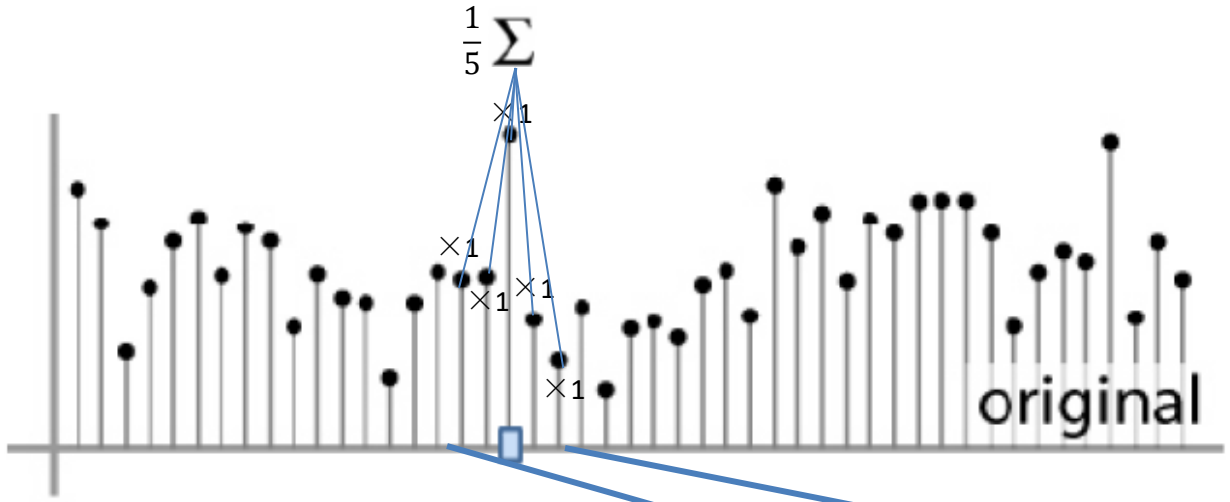
How to find the exactly same bird?

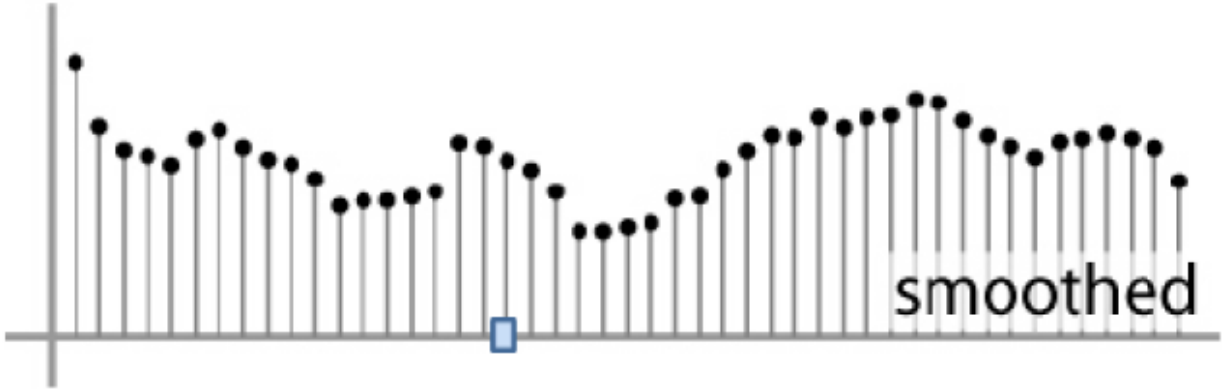
How to extract all these edges?



Denoising

- The simplest thing: replace each pixel by the average of neighbors.
- 1-D example:



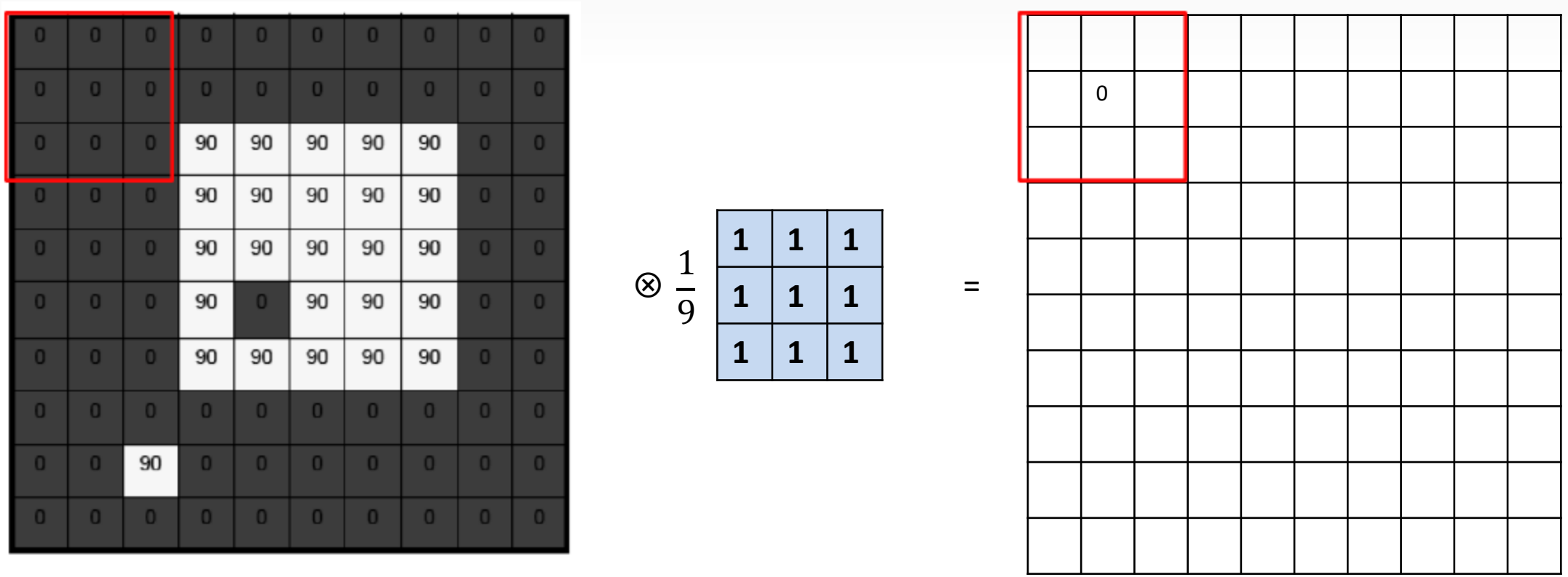


[Image from S. Marschner]

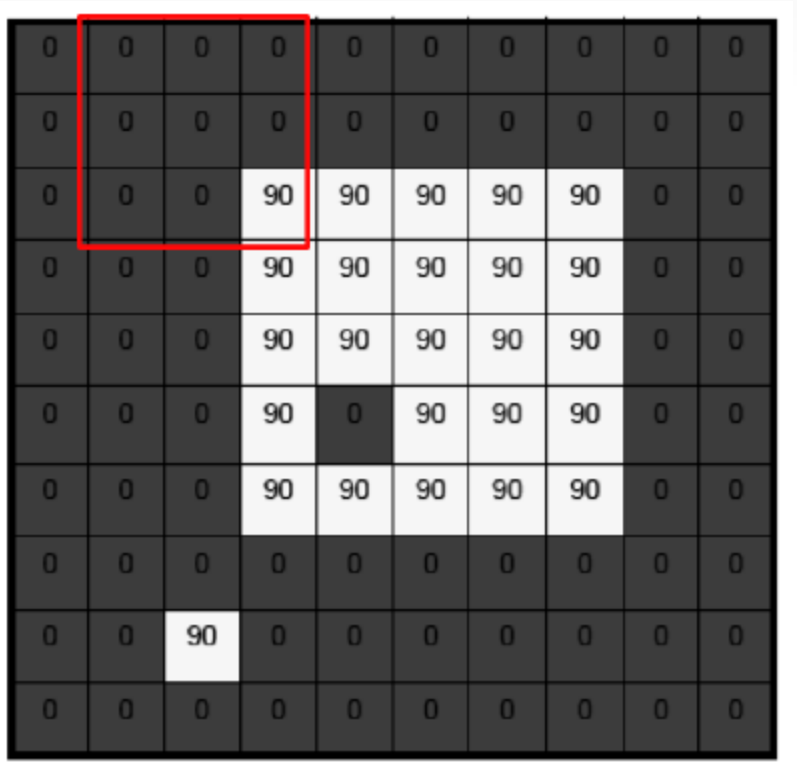
Filtering template $\frac{1}{5}$ 1 1 1 1 1

Center

Averaging in 2D image



Averaging in 2D image



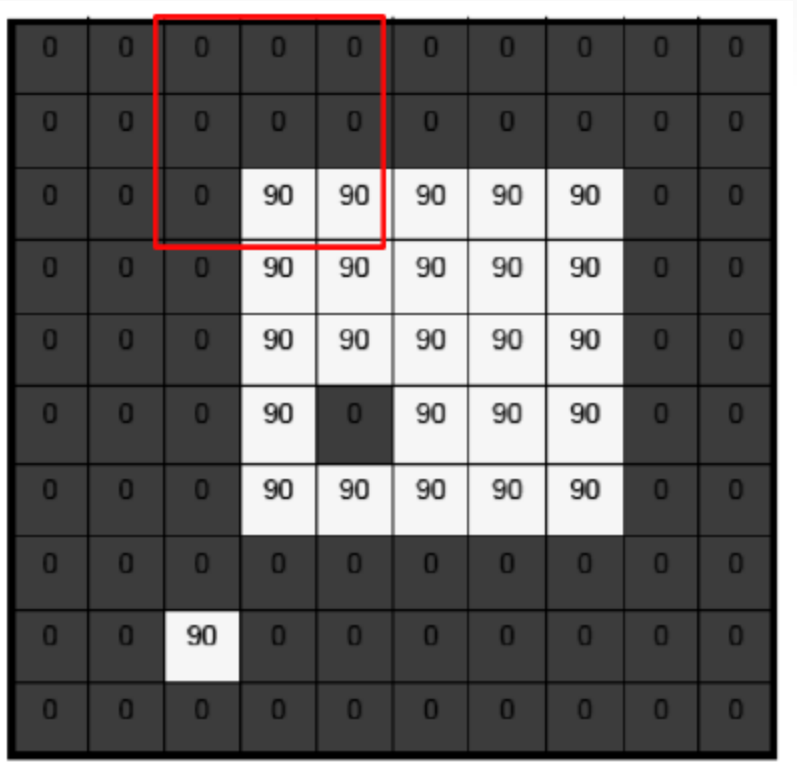
$\otimes \frac{1}{9}$

1	1	1
1	1	1
1	1	1

=

	0	10							

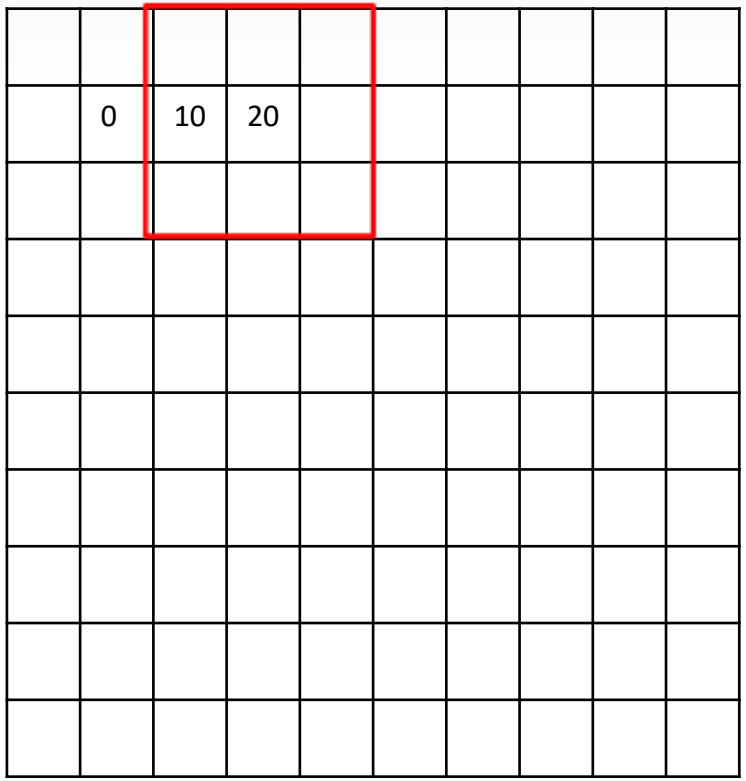
Averaging in 2D image



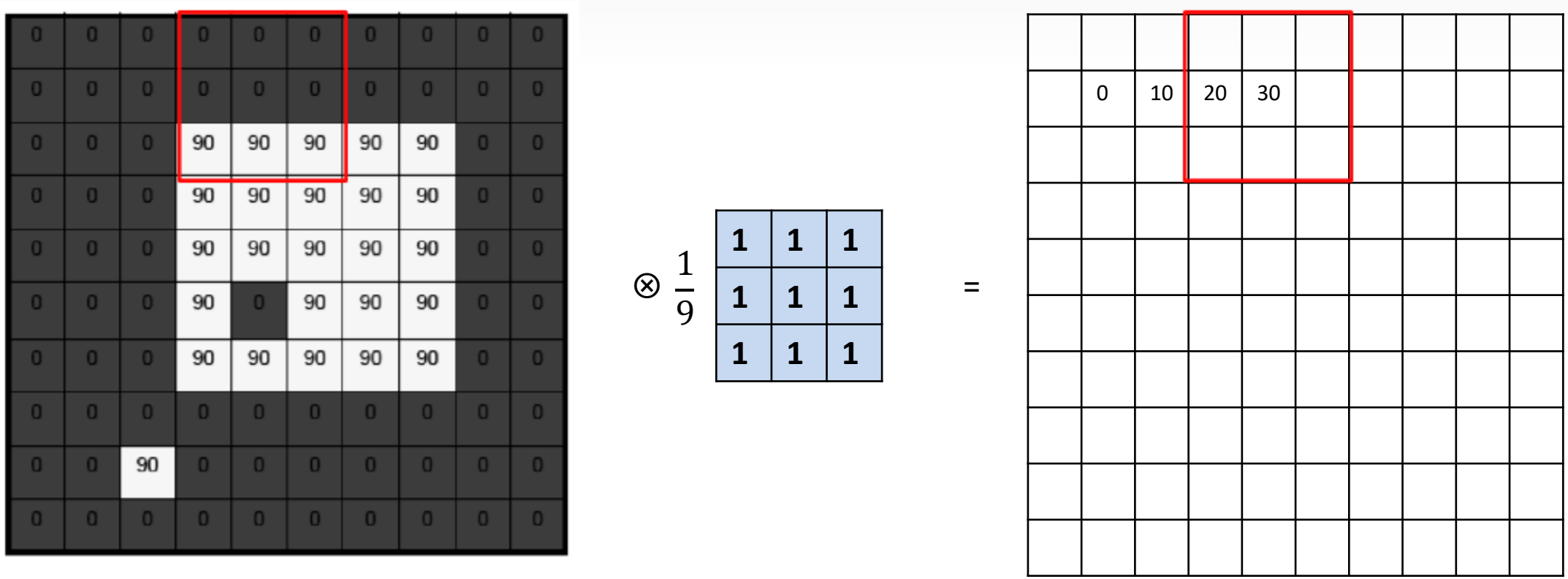
$\otimes \frac{1}{9}$

1	1	1
1	1	1
1	1	1

=



Averaging in 2D image



Averaging in 2D image

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$\otimes \frac{1}{9}$

1	1	1
1	1	1
1	1	1

=

	0	10	20	30	30	30	20	10	
	0	20	40	60	60	60	40	20	
	0	30	60	90	90	90	60	30	
	0	30	50	80	80	90	60	30	
	0	30	50	80	80	90	60	30	
	0	20	30	50	50	60	40	20	
	10	20	30	30	30	30	20	10	
	10	10	10	0	0	0	0	0	

Filtering: Convolution

- For the averaging thing, it is equally weighted combinations of pixels in a small neighboring group:

$$G(i,j) = \frac{1}{(2k+1)^2} \sum_{u=-k}^{k} \sum_{v=-k}^{k} I(i+u, j+v)$$

- For general filtering, it is determined as a weighted sum of input pixel value

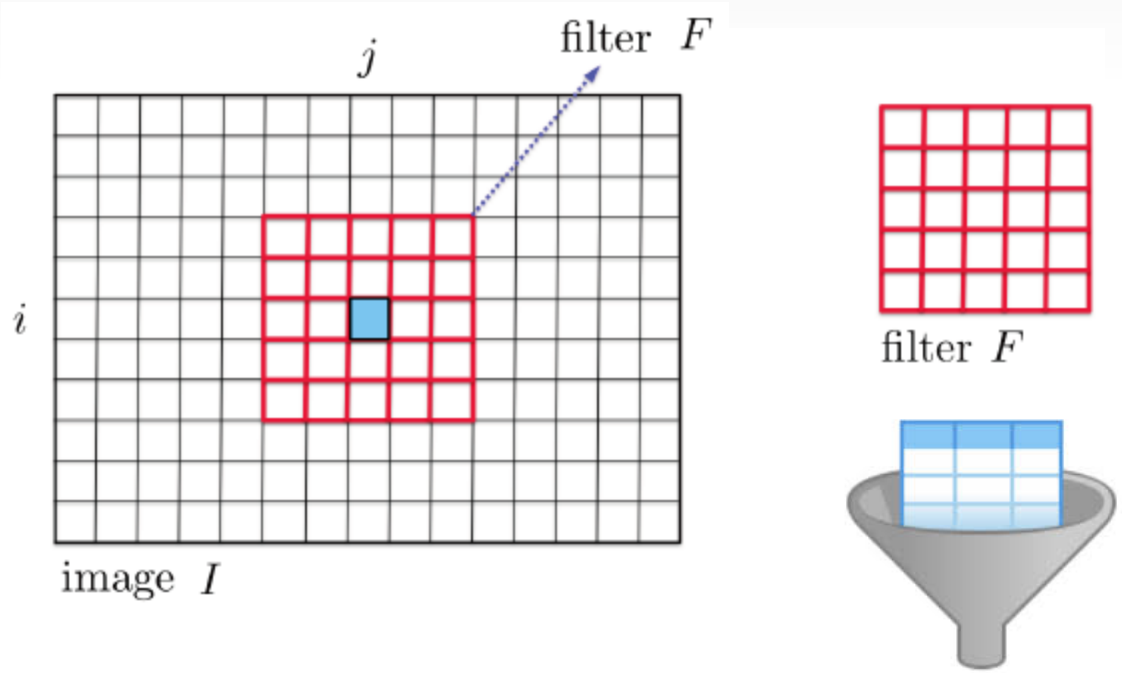
$$G(i,j) = \sum_{u=-k}^{k} \sum_{v=-k}^{k} F(u,v) * I(i+u, j+v)$$

The entries of the **weight kernel** or **mask** *F(u, v)* are called the **filter coefficients.**

This operation is also called **CONVOLUTION**, just as in deep learning

Filtering templates

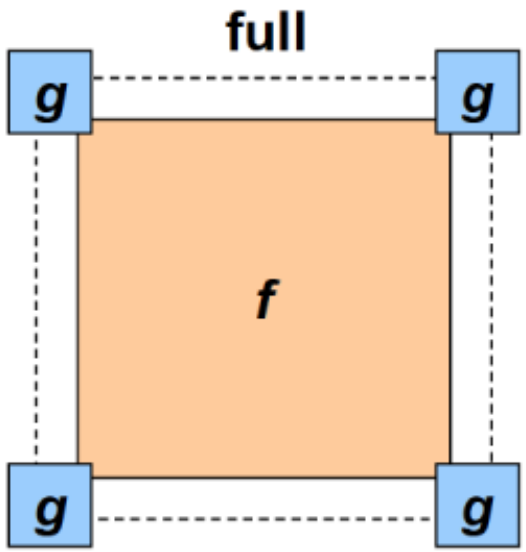
- We can define different templates for different situations

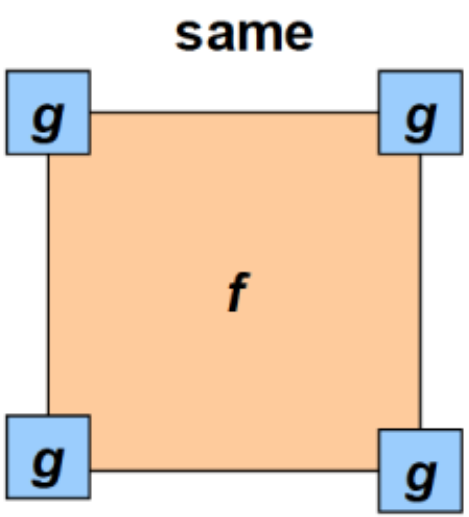


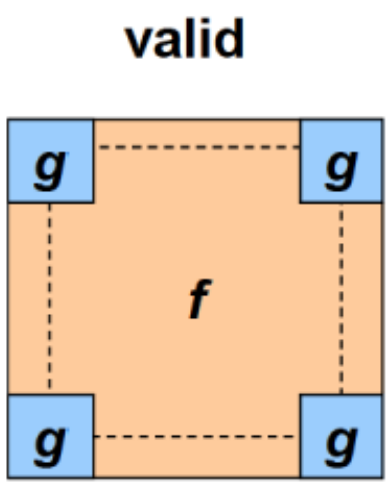
Filter output only one pixel value

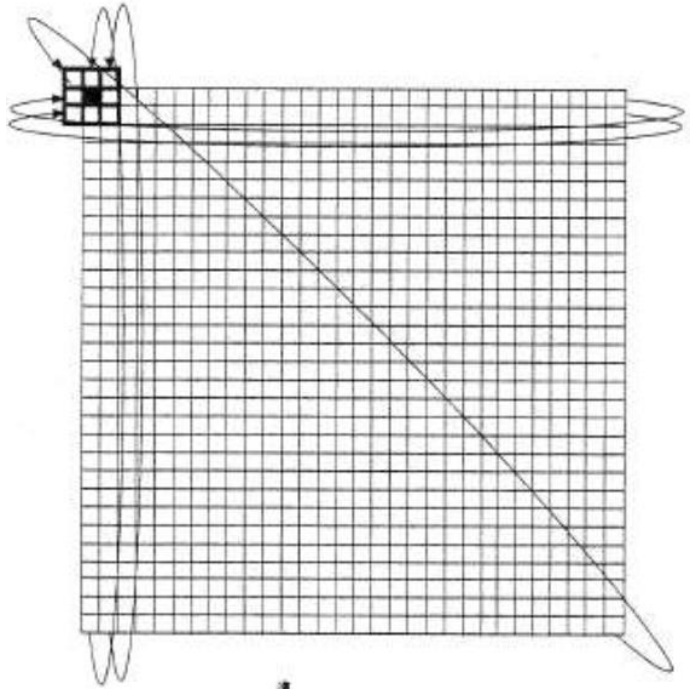
Filtering templates

- **What about boundary pixels?**
  - Different ways to handle it.



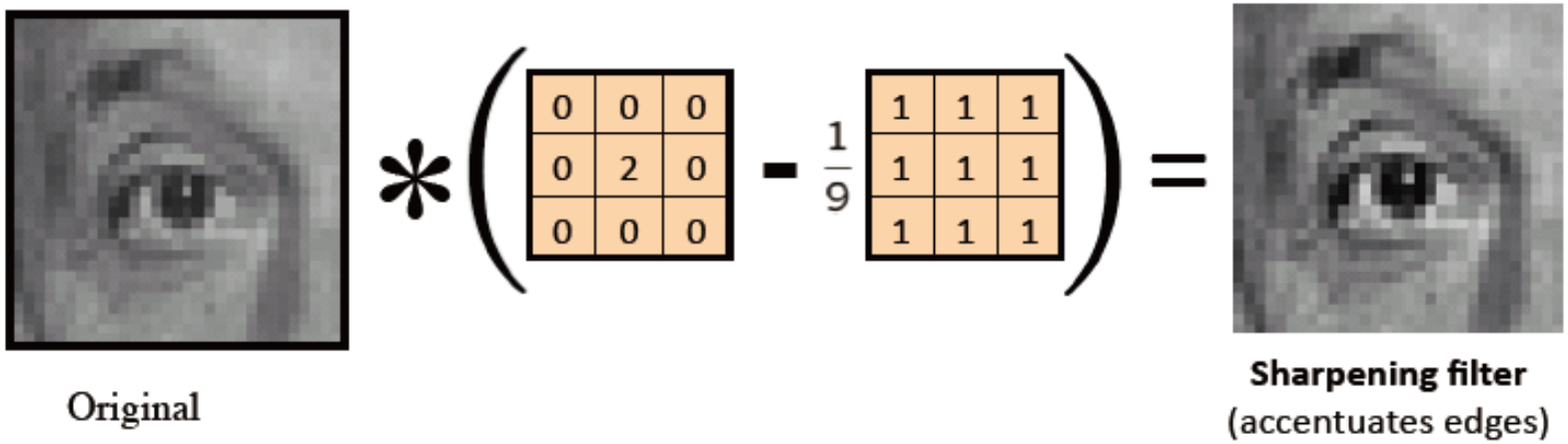






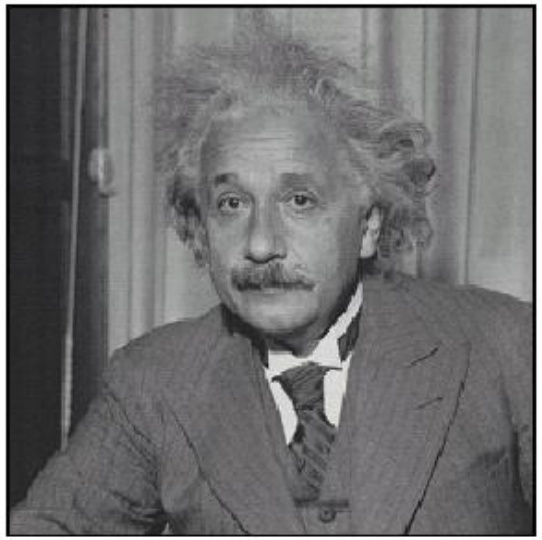
Recycle or just pad with zeros

Filtering Applications

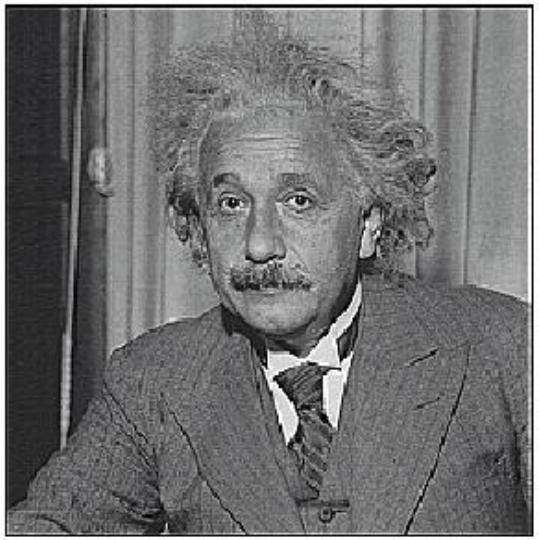


Original

Sharpening filter
(accentuates edges)



before

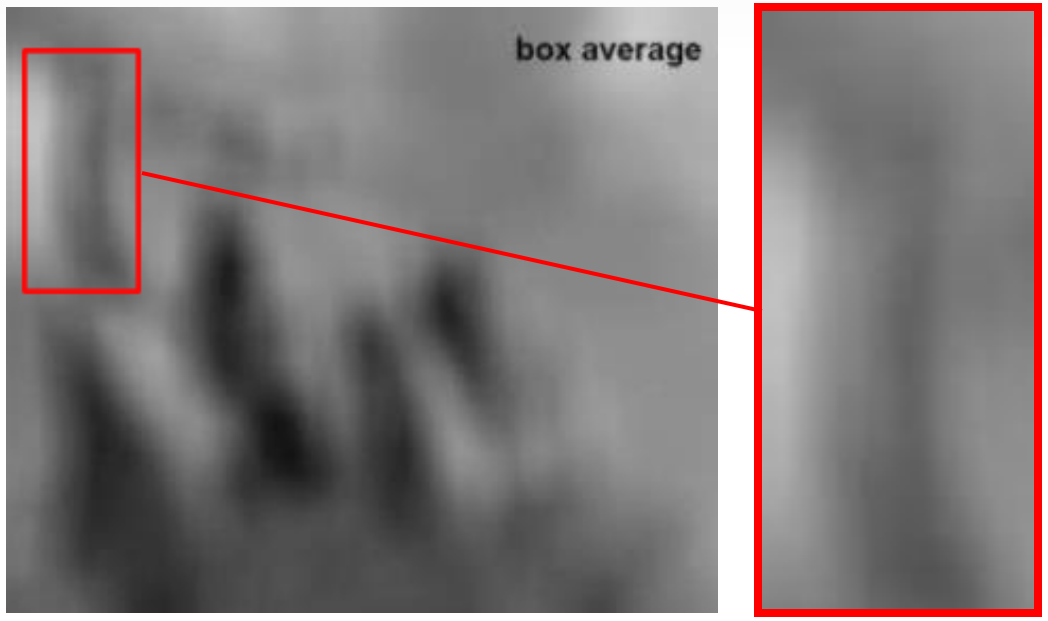


after

[Source: D. Lowe]

Filtering Applications

- Smoothing by averaging (or box averaging)



- We actually have a mathematically correct smoothing filter
  - Gaussian filter

Filtering Applications

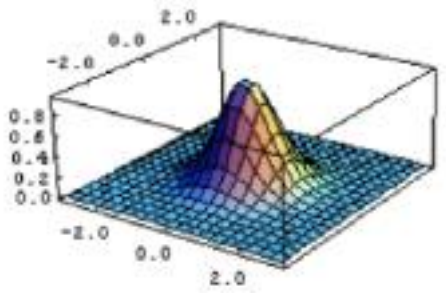
- **Gaussian filter kernel:**
  - 3 by 3 version:

$$\frac{1}{16}\begin{bmatrix}1 & 2 & 1\\ 2 & 4 & 2\\ 1 & 2 & 1\end{bmatrix}$$

$F(i,j)$

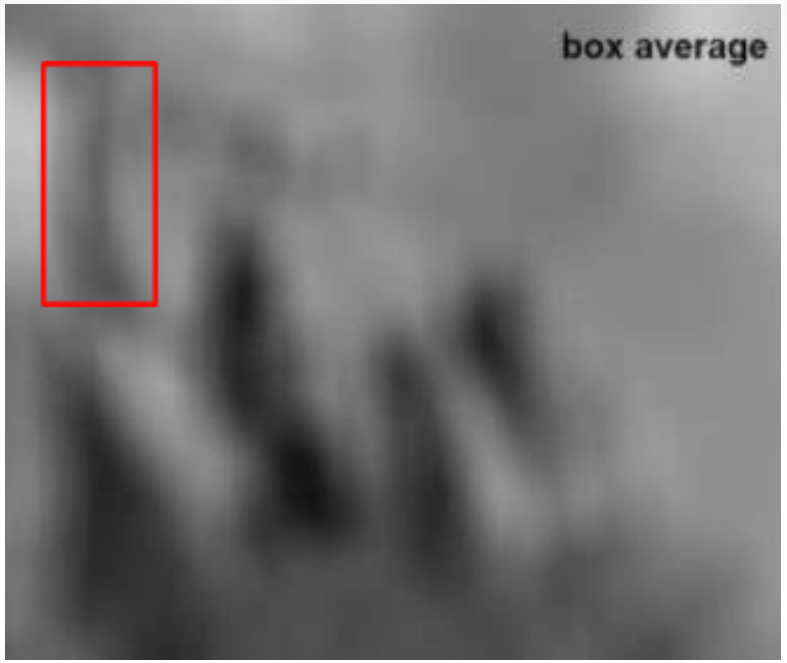
This kernel is an approximation of a 2d Gaussian function:

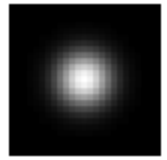
$$h(u,v) = \frac{1}{2\pi\sigma^2}e^{-\frac{u^2+v^2}{\sigma^2}}$$



- **Removes high-frequency components from the image (low-pass filter)**
  - **In Fourier Transform** of an image, the 2D image is treated as a 2D signal
    - decomposed into components in different frequency
    - Sharp edges, where the image changes rapidly, are modeled by high frequencies
    - Applying Gaussian filter can remove the high frequencies, but box filter cannot.

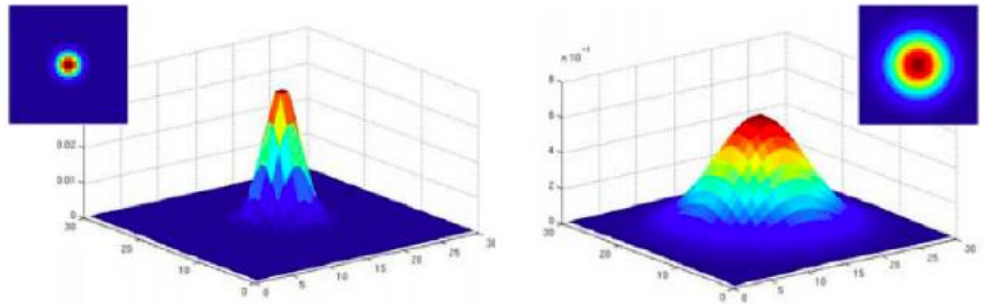
Smoothing by Gaussian





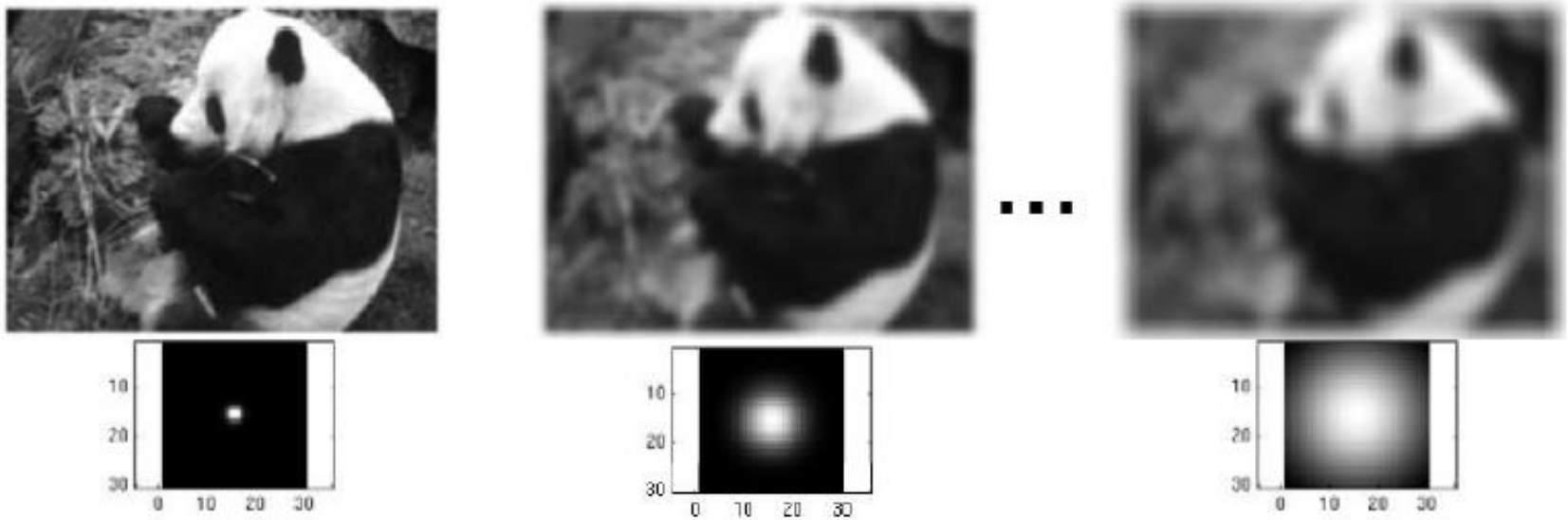
Smoothing by Gaussian

- **Different parameters for the Gaussian functions**
  - We can get different level of smoothness



σ = 2 with 30 x 30 kernel

σ = 5 with 30 x 30 kernel



[Source: K. Grauman]

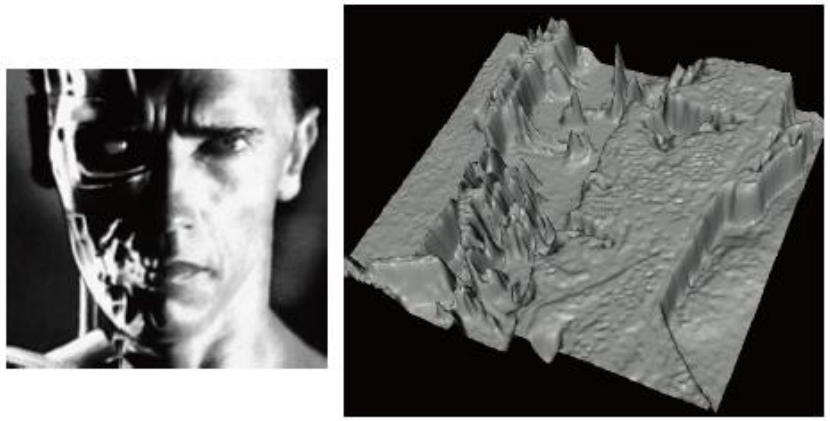
Edge Detection

- **Classical definition of the edge detection problem: localization of large local changes in the grey level image → large graylevel gradients**
  - Can be extended to color images

- **Goal: identify objects in images**
  - also feature extraction, 3D reconstruction, motion recognition, image restoration, registration
- **Contours are very important perceptual cues!**
  - They provide a first saliency map for the interpretation of image semantics

Edge Detection

- **What causes an edge?**
  - Depth discontinuity, object boundaries
  - Change in surface orientation: shape
  - Cast shadows
- **Edges look like steep cliffs**



Images as Functions

Derivatives of images

- If image f was continuous, then compute the partial derivative as

$$\frac{df(x,y)}{dx} = \lim_{\varepsilon \to 0} \frac{f(x+\varepsilon, y) - f(x,y)}{\varepsilon}$$

- In the discrete domain, 1st-order forward discrete derivative:

$$\frac{df(x,y)}{dx} \approx \frac{f(x+1,y) - f(x,y)}{1}$$

- So The partial derivative of image ***f(x,y)*** with respect to the variable ***x*** can be computed as:

$$(-1) * f(x,y) + (1) * f(x+1,y)$$

Filter for getting partial derivatives

- It can just be computed by applying the filter:

-1	1

$$(-1) * f(x,y) + (1) * f(x+1,y)$$

- Partial derivative for y:

-1
1

- The horizontal derivative using the filter $[-1,1]$

Filter for getting partial derivatives

- It can just be computed by applying the filter:

-1	1

$$(-1) * f(x,y) + (1) * f(x+1,y)$$

- Partial derivative for y:

-1
1

- The vertical derivative using the filter $[-1,1]^T$

Filter for getting partial derivatives

Other approximations

- Other templates for getting approximated derivatives

Roberts

$$\begin{pmatrix} [0] & 1 \\ -1 & 0 \end{pmatrix} \quad \begin{pmatrix} [1] & 0 \\ 0 & -1 \end{pmatrix}$$

Central Difference

$$\begin{pmatrix} 0 & 0 & 0 \\ -1 & [0] & 1 \\ 0 & 0 & 0 \end{pmatrix} \quad \begin{pmatrix} 0 & -1 & 0 \\ 0 & [0] & 0 \\ 0 & 1 & 0 \end{pmatrix}$$

Prewitt

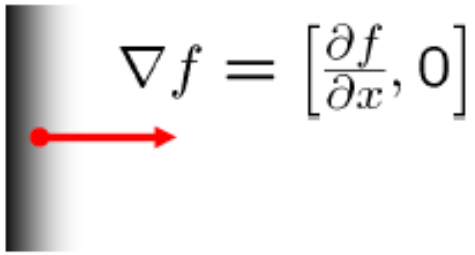
$$\begin{pmatrix} -1 & 0 & 1 \\ -1 & [0] & 1 \\ -1 & 0 & 1 \end{pmatrix} \quad \begin{pmatrix} -1 & -1 & -1 \\ 0 & [0] & 0 \\ 1 & 1 & 1 \end{pmatrix}$$

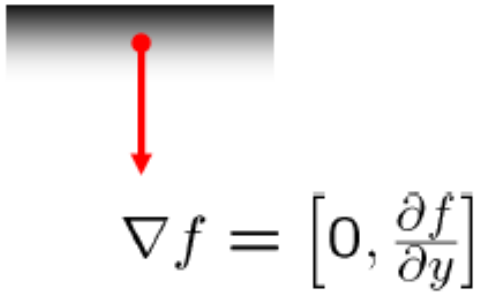
Sobel

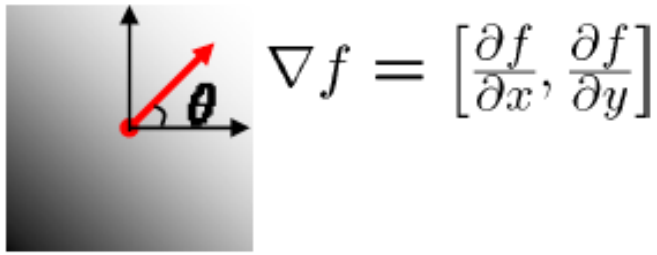
$$\begin{pmatrix} -1 & 0 & 1 \\ -2 & [0] & 2 \\ -1 & 0 & 1 \end{pmatrix} \quad \begin{pmatrix} -1 & -2 & -1 \\ 0 & [0] & 0 \\ 1 & 2 & 1 \end{pmatrix}$$

Image Gradient

- **The gradient of an image:** $\nabla f = [\frac{df}{dx}, \frac{df}{dy}]$
- **The gradient points in the direction of most rapid change in intensity**







- **The gradient direction (orientation of edge normal) is given by:**

$$\theta = \tan^{-1}(\frac{df}{dy} / \frac{df}{dx})$$

Image Gradient

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

$$||\nabla f|| = \sqrt{(\frac{df}{dx})^2 + (\frac{df}{dy})^2}$$

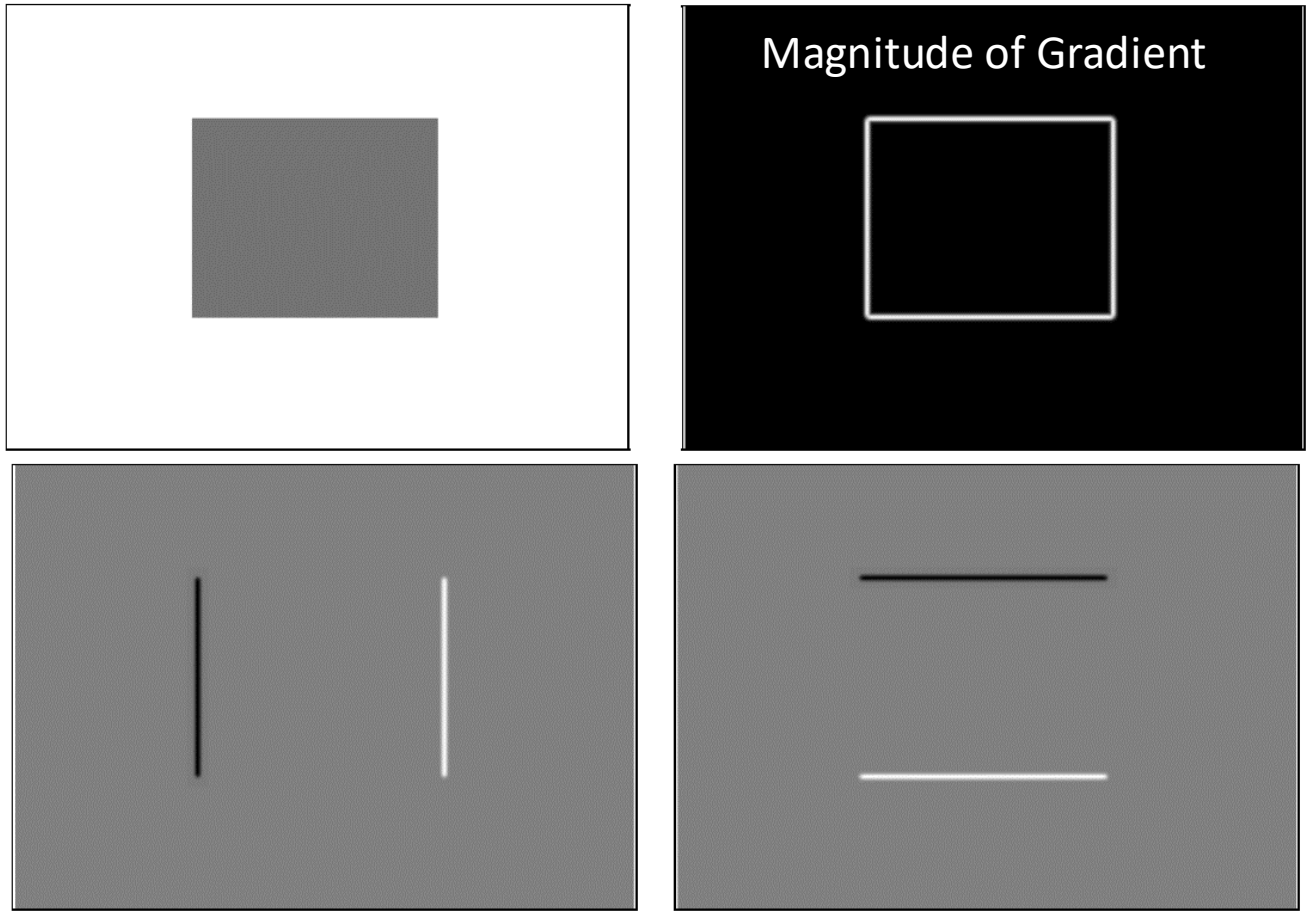


Image Gradient

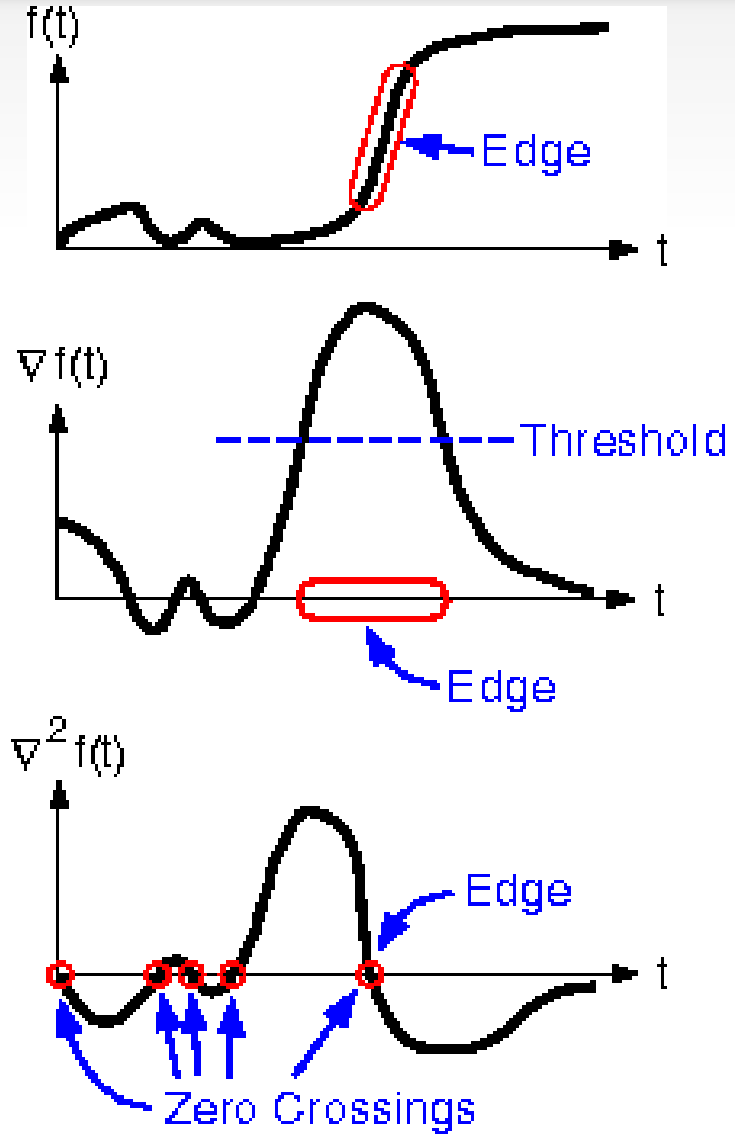
- **Thresholding gradient magnitude to get edges:**

Gradient Magnitude
(Gradient Map)

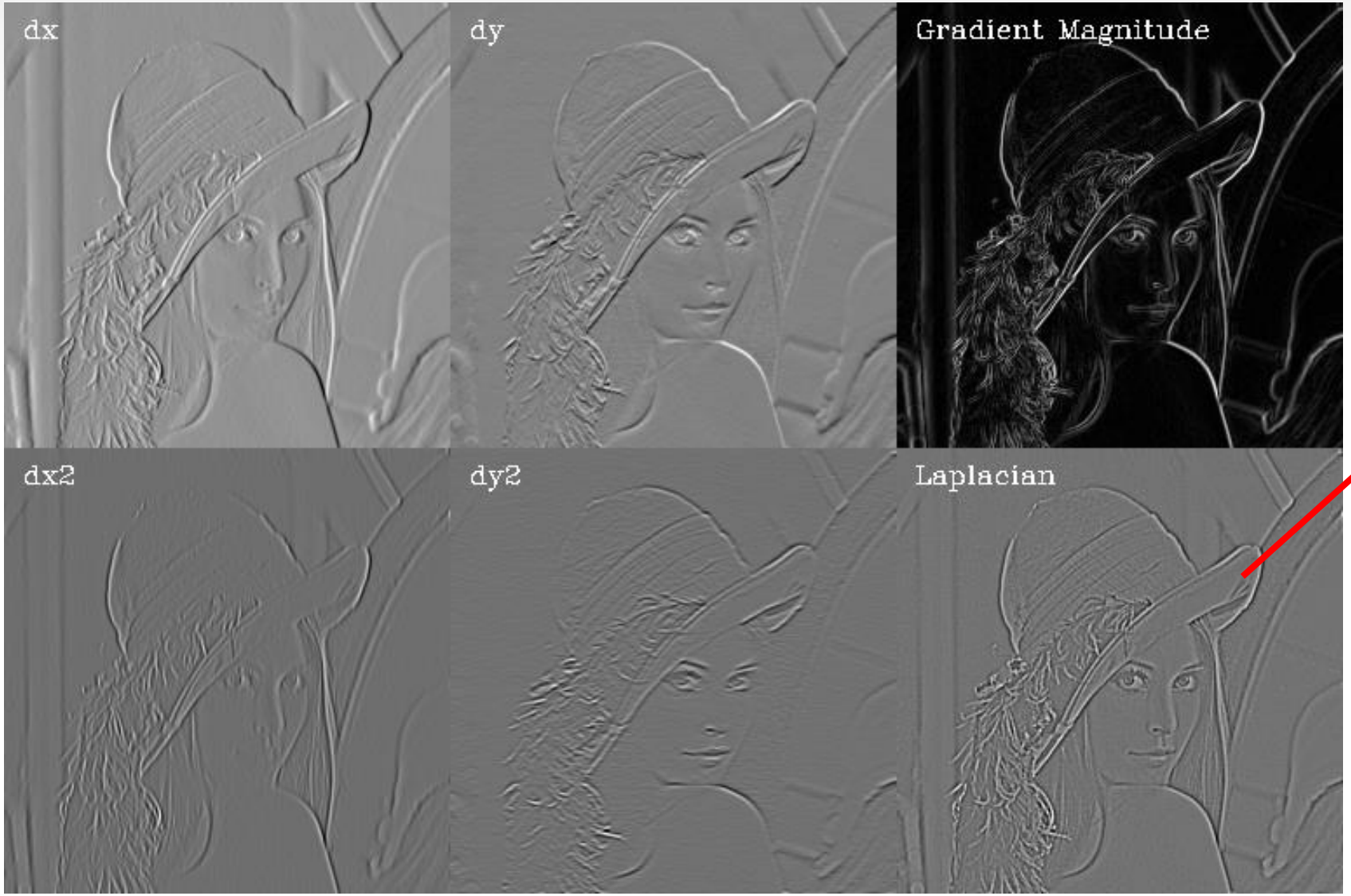
Edges

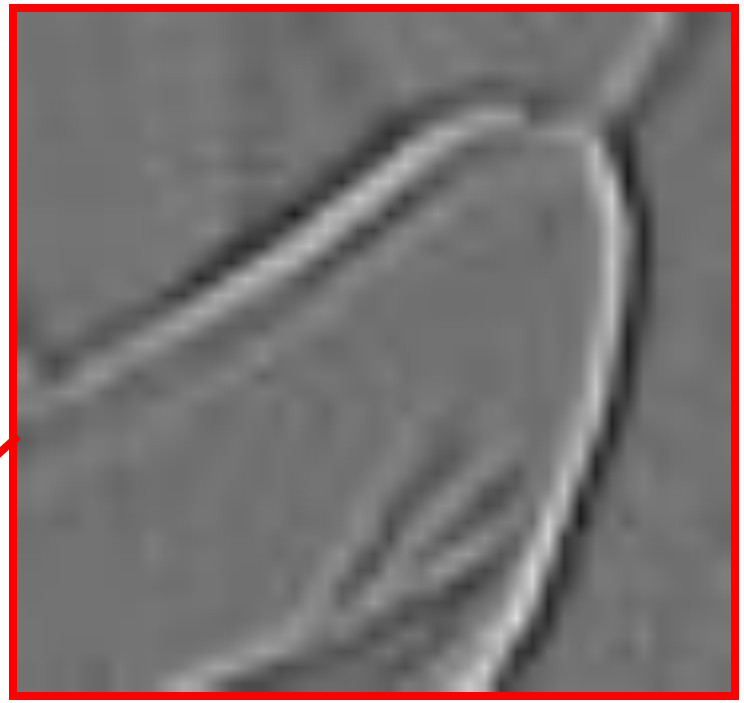
Laplacian filter – second derivatives

- However, directly thresholding the gradient magnitude, will get a “thick” edge
- Laplacian filter is for detecting edge by second derivative
- Zero crossing points will be the edge



Laplacian filter





Laplacian filter

First order finite difference: $\delta_h[f](x) = f(x + \frac{1}{2}h) - f(x - \frac{1}{2}h)$

Second order derivative calculated by finite difference

When h approaches 0

$$f''(x) \approx \frac{\delta_h^2[f](x)}{h^2} = \frac{f(x+h) - 2f(x) + f(x-h)}{h^2}.$$

$$It\ is\ I(x+1) + I(x-1) - 2I(x)$$

Laplace filter

1	-2	1

For 2D

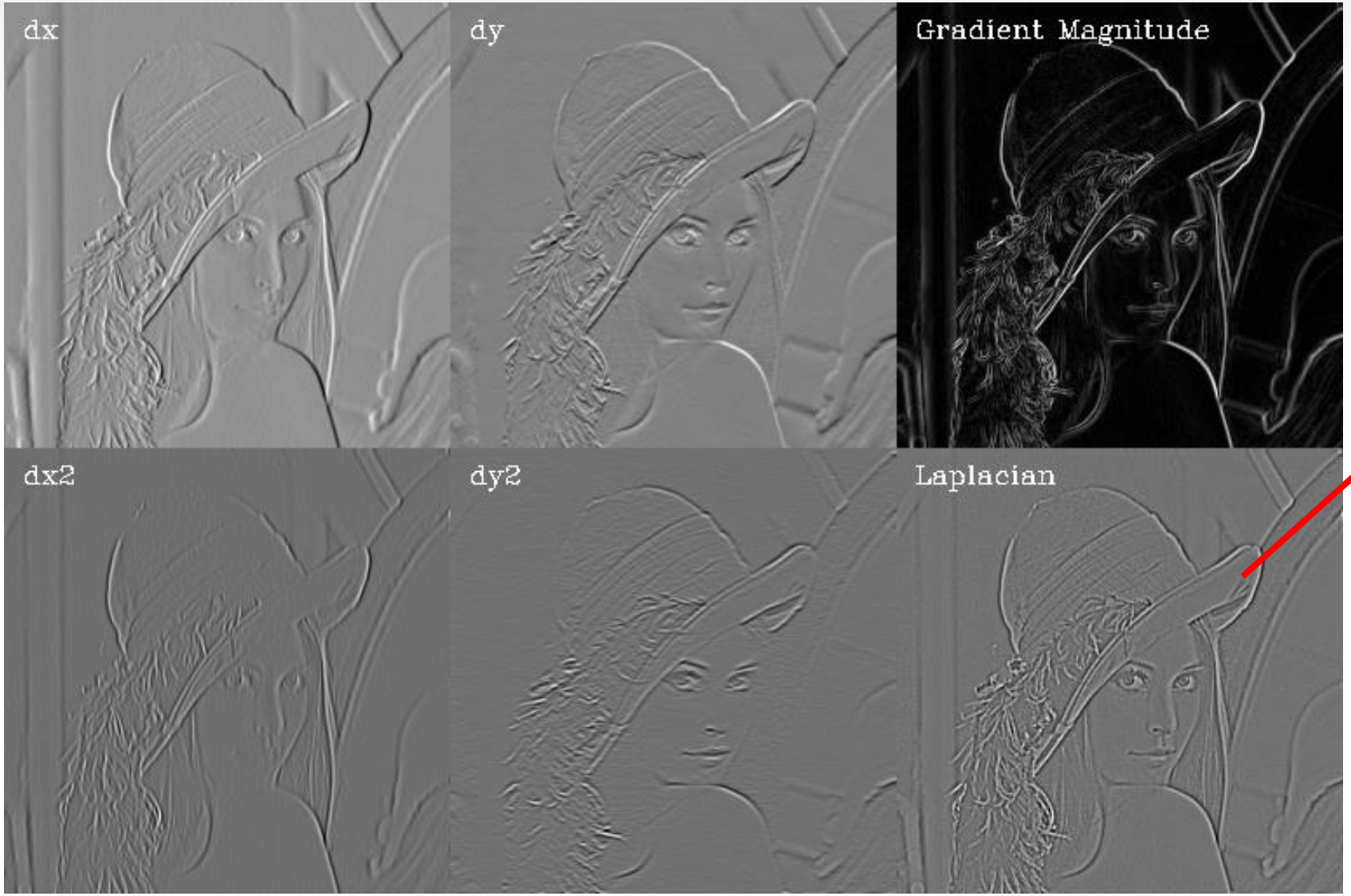
$$\Delta f = \frac{\partial f^2}{\partial^2 x} + \frac{\partial f^2}{\partial^2 y} = I(x+1,y) + I(x-1,y) - 2I(x,y) + I(x,y+1) + I(x,y-1) - 2I(x,y)$$

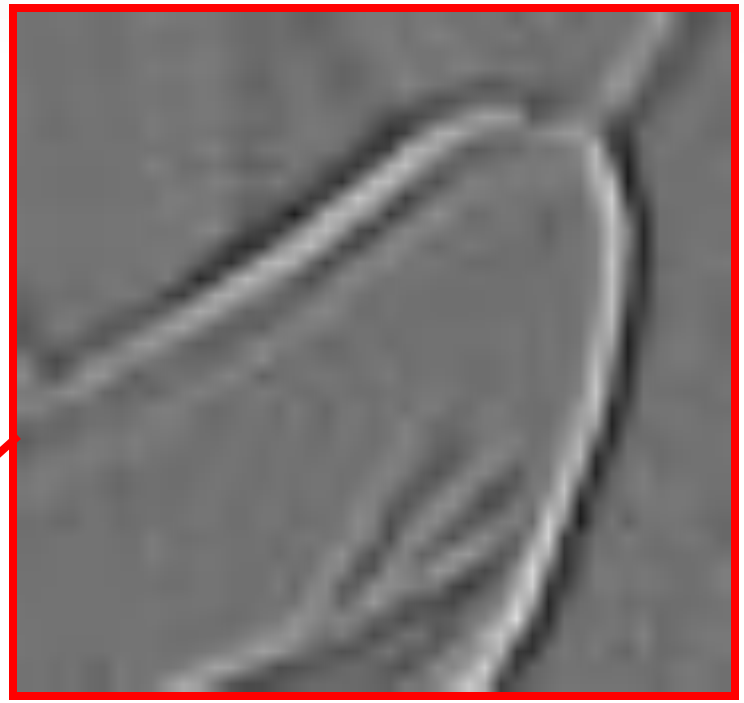
$$= I(x+1,y) + I(x-1,y) + I(x,y+1) + I(x,y-1) - 4I(x,y)$$

0	1	0
1	-4	1
0	1	0

2D Laplace filter

Laplacian filter





Laplacian filter

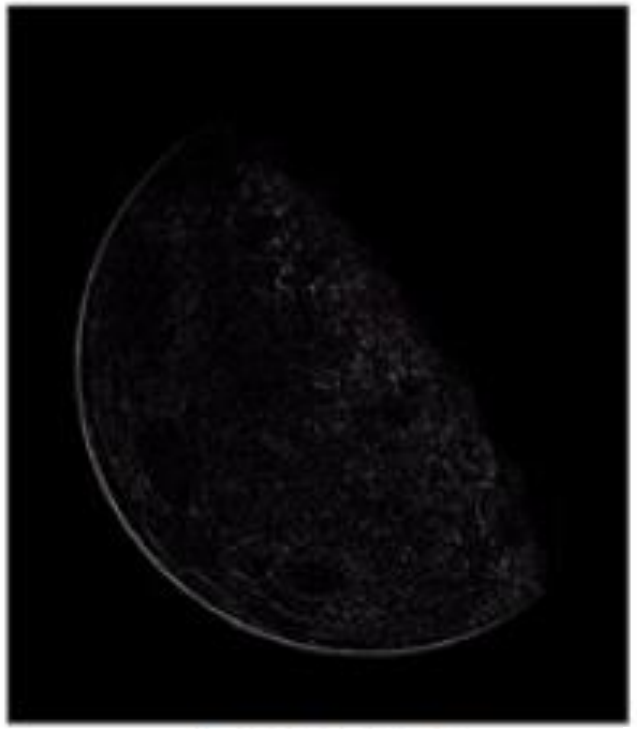
Filtered Result
(Absolute value visualisation)

Laplacian filter

- A simple enhancement using Laplacian filter: Sharpen

Original Image

−



Laplacian Filtered Image

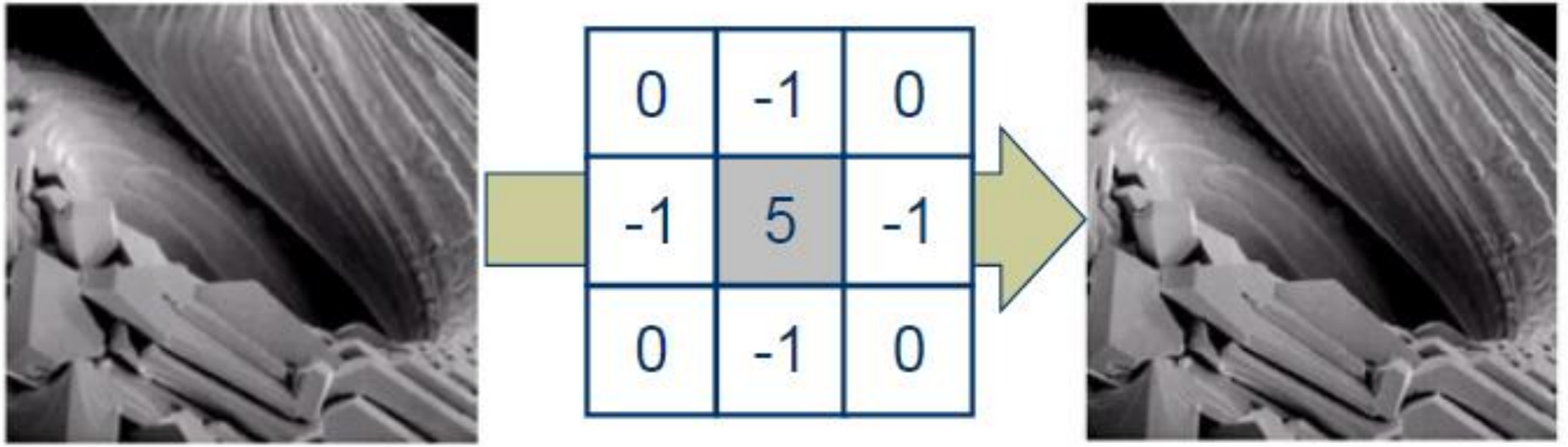
=

Sharpened Image

0	1	0
1	-4	1
0	1	0

2D Laplace filter

Laplacian Filter for Sharpening



$$g(x,y) = f(x,y) - \nabla^2 f$$

$$= f(x,y) - [f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1) - 4f(x,y)]$$