Image Enhancement

Lecture 3

Qiufeng Wang

Qiufeng.Wang@xjtlu.edu.cn

Outline

- Introduction
  - AWGN, S&P noise
- Mapping function
- Histogram
  - Image histogram
  - Histogram Equalization
- Image Averaging
- Spatial filtering
  - LPF, HPF
- Practical filters
  - Averaging filter
  - edge detection filters
  - sharpening filters
  - Order Statistic Filters
- Convolution Neural Networks (CNN)

Introduction

- Images may suffer from the following degradations:
  - Poor contrast: due to poor illumination or finite sensitivity of the imaging device
  - Noisy image: due to electronic sensor noise, signal amplifier, atmospheric disturbances, or compression
  - Aliasing effects: due to sampling
  - Spatial blurring: due to aperture effects or motion
  - Others

Noisy image

- Additive white Gaussian noise (AWGN)

Original Image

Noisy Image

AWGN

1. Gaussian: probability density function

$$p(z) = \frac{1}{\sqrt{2\pi}\sigma} e^{-(z-\bar{z})^2/2\sigma^2}$$

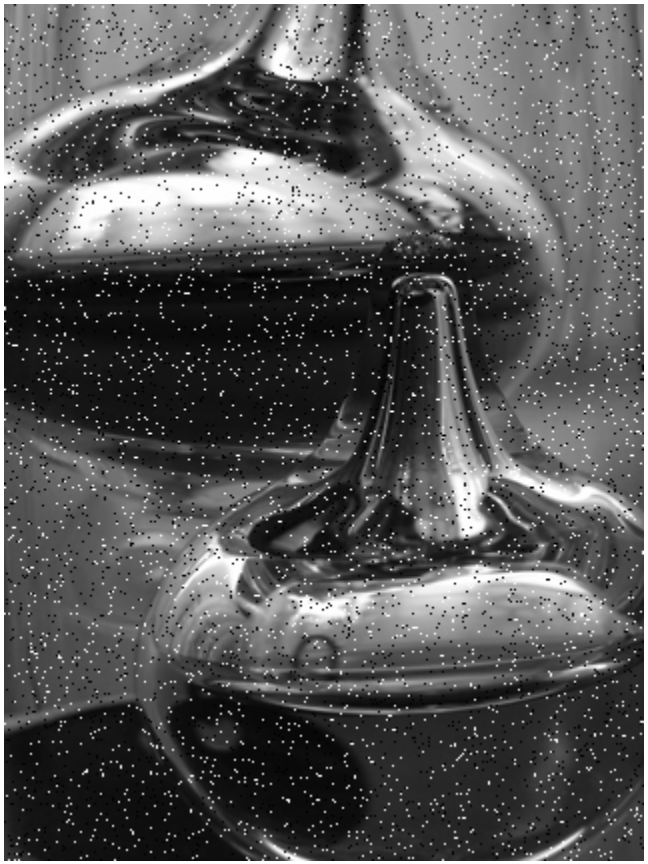
2. Code: AWGN

NewImage=
OldImage+K*Gaussian

```
void AddNoise(Mat img,double mu, double sigma,int k){
    Mat outImage;
    outImage.create(img.rows,img.cols,img.type());
    for(int x=0;x<img.rows;x++){
        for(int y=0;y<img.cols;y++){
            double temp = img.at<uchar>(x, y)
                    +k*generateGaussianNoise(mu,sigma);
            if(temp>PixcelMax)
                    temp=PixcelMax;
            else if(temp<PixcelMin)
                    temp=PixcelMin;
            outImage.at<uchar>(x, y) = temp;
        }
    }
    Filter(outImage,Filter::NXBJZ,3,3,1);
    imshow("Output", outImage);
    cvWaitKey(0);
}
```

Noisy image

- Salt and pepper noise
  - also known as impulse noise. This noise can be caused by sharp and sudden disturbances in the image signal



From wikipedia.org/

Salt and pepper noise

1. Randomly select position

2. Set it 0 or 255

```
void AddNoise(Mat img,double SNR ){
    Mat outImage;
    outImage.create(img.rows,img.cols,img.type());
    int SP = img.rows*img.cols;
    int NP = SP*(1-SNR);
    outImage = img.clone();
    for(int i=0; i<NP; i++) {

        int x = (int)(rand()*1.0/RAND_MAX* (double)img.rows);
        int y = (int)(rand()*1.0/RAND_MAX* (double)img.cols);
        int r = rand()%2;
       if(r){
            outImage.at<uchar>(x, y)=0;
        }
       else{
           outImage.at<uchar>(x, y)=255;
       }

    }
     Filter(outImage,Filter::NXBJZ,3,3,1);
    imshow("Output", outImage);
    cvWaitKey(0);
}
```

Image enhancement and restoration

- The goals of image enhancement are to:
  - Accentuate (filter) certain features for subsequent analysis or image display.
  - Discard unimportant image features.
  - Enhance otherwise hidden information
- On the other hand image restoration is a process that attempts to reconstruct or recover an image that has been degraded.
  - Remove noise
- Enhancement is the process of manipulating an image so that the result is more suitable than the original for a specific application.
  - Problem oriented
  - No general theory of measurement

Image enhancement and restoration

- Examples of image enhancement and restoration operations:
  - noise removal;
  - geometric distortion correction;
  - edge enhancement;
  - contrast enhancement;
  - image zooming;
- **Categories** of image enhancement
  - **Frequency domain**
  - Spatial domain (**pixel-domain**)

Pixel-domain image enhancement

- Mathematically pixel-domain image enhancement could be represented by the operator *T[.]*
- *T[.]* operates on the input image *im* to produce an output image *imo*, there are two approaches for this :
  - Non-adaptive to the data content
  - Adaptive to the data content

Pixel-domain image enhancement

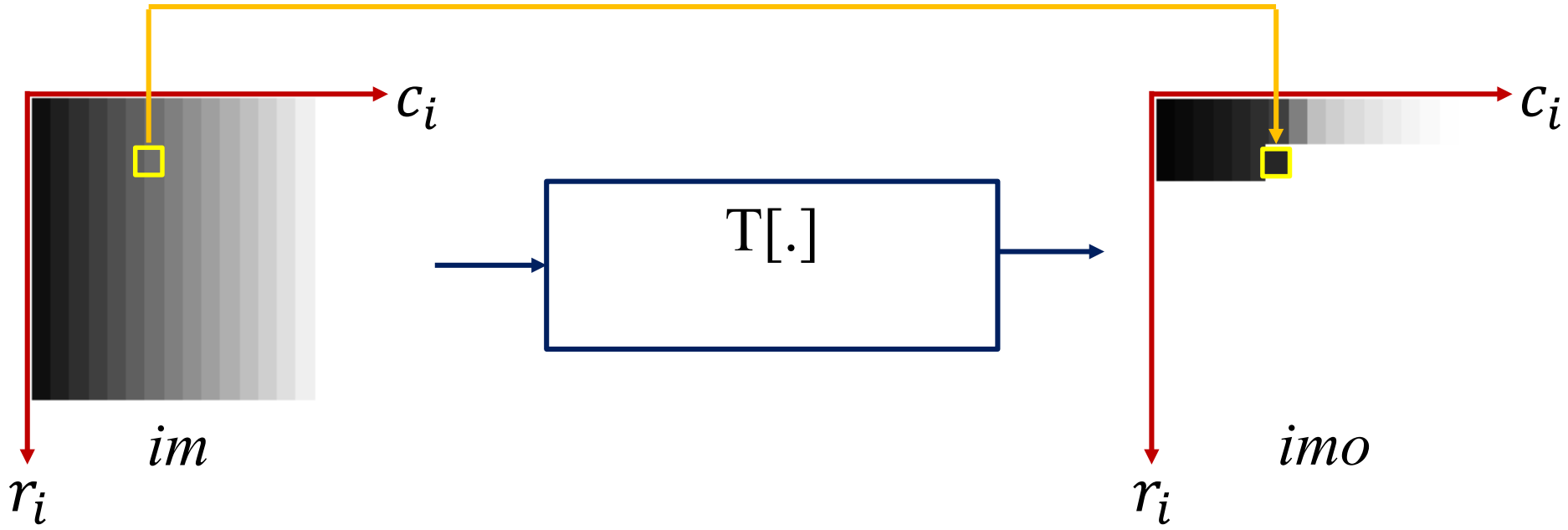
- Non-adaptive to image intensity
  - Usually used to compensate the image acquisition unit (e.g., gamma-compensation)
  - Usually used to overcome the limitation of the physical image acquisition unit (e.g., image negative)
- Adaptive to image intensity
  - Histogram based correction

Pixel-domain image enhancement

- *T[.]* in term of principle of operation could be classified as :
  - Pixel-to-pixel mapping
  - Block-to-pixel mapping
  - Block-to-block mapping
  - Image-to-image mapping

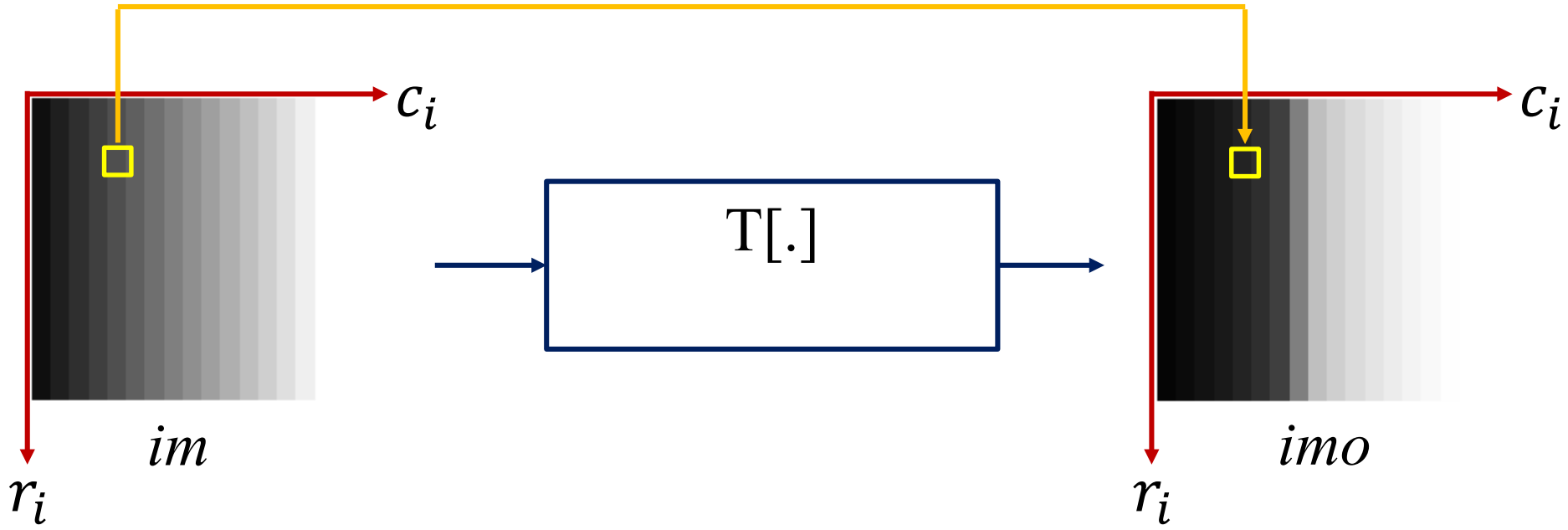
Pixel-domain image enhancement

- Pixel-to-pixel mapping
  - Operates on one pixel at a time
  - It is easy to implement



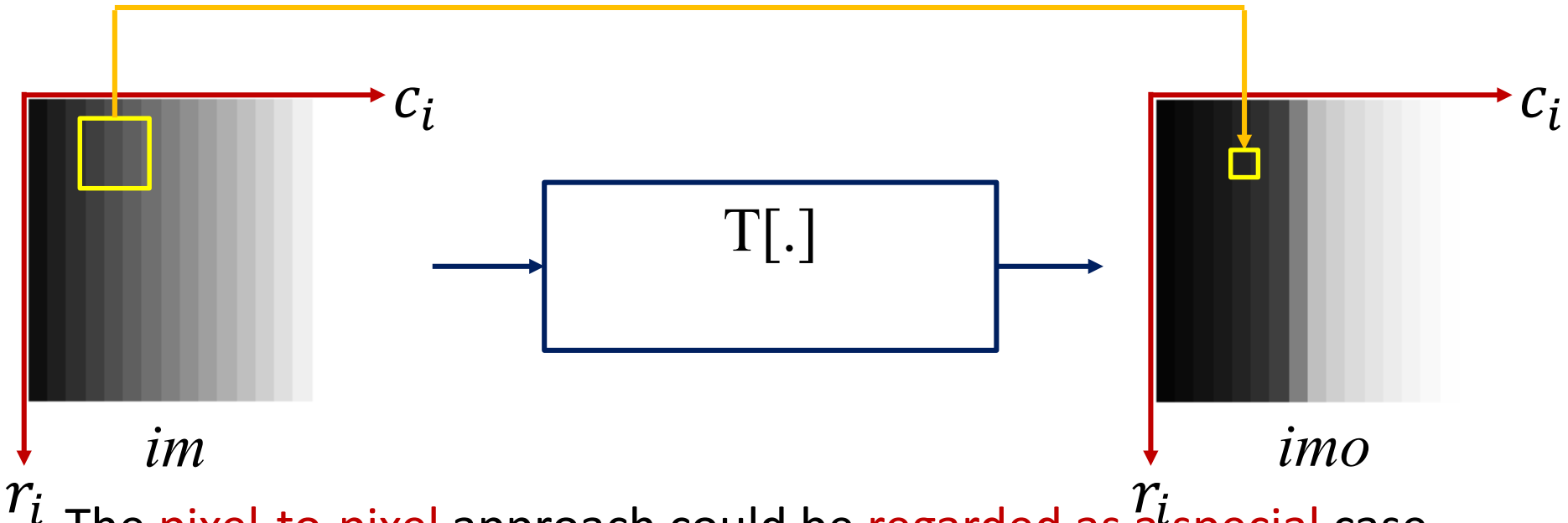
Pixel-domain image enhancement

- Pixel-to-pixel mapping
  - For gray images we could call it intensity transform (mapping function): $imo(r_i, c_i) = T[im(r_i, c_i)]$



Pixel-domain image enhancement

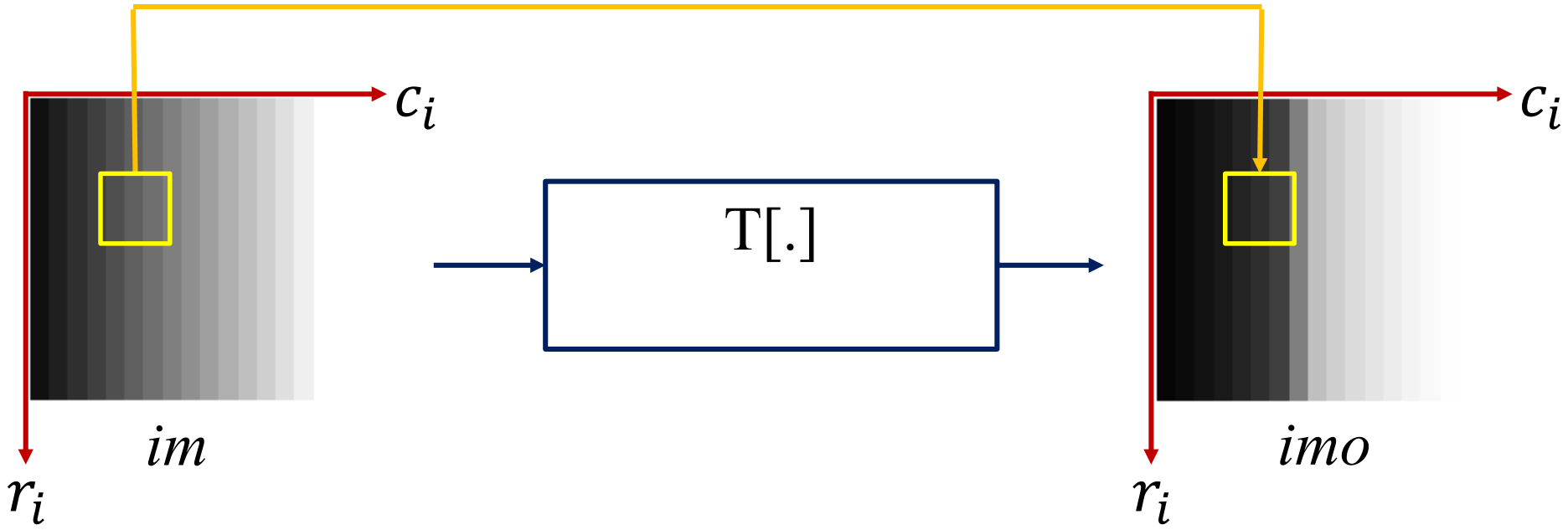
- Block-to-pixel mapping
  - Operates on a set of 'neighbourhoods' *N* of each pixel



The pixel-to-pixel approach could be regarded as a special case of the block-to-pixel approach, where the size of block is 1X1

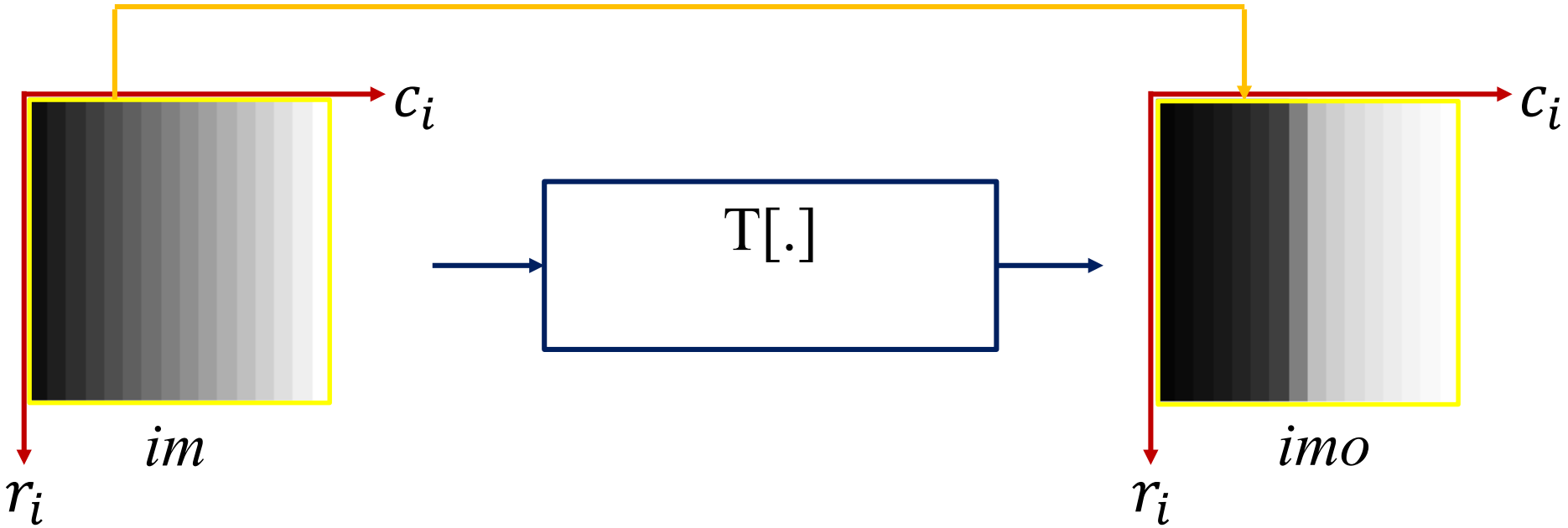
Pixel-domain image enhancement

- Block-to-block mapping
  - Operates on a set of pixels to produce a set of pixels



Pixel-domain image enhancement

- Image-to-image mapping
  - Operates on the whole set of pixels of the image



Is there any relationship between the block-to-block approach and the image-to-image approach ?

Pixel-domain image enhancement

- *T[.]* in term of principle of operation could be classified as :
  - Pixel-to-pixel mapping
  - Block-to-pixel mapping
  - Block-to-block mapping
  - Image-to-image mapping
- The previous approaches have different characteristics in term:
  - Complexity
  - Flexibility
  - Simplicity of the design stage

Examples

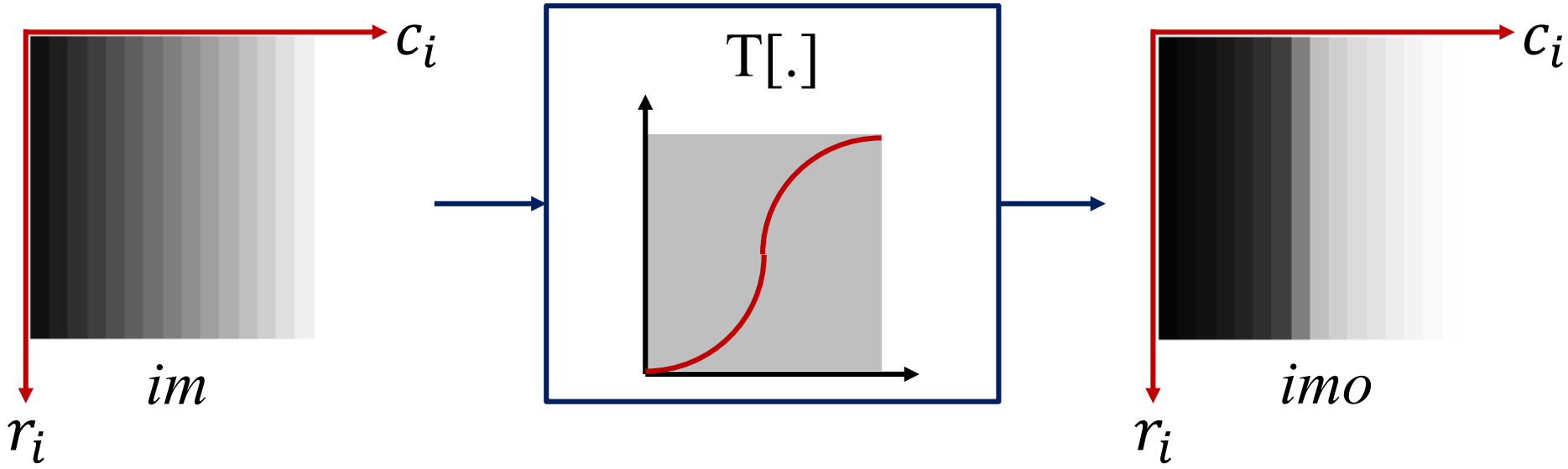
Contrast enhancement

- To produce an image of higher contrast than the original image
  - Contrast is the **difference** in luminance or color that makes an object (or its representation in an image or display) distinguishable**.**
  - human visual system is more sensitive to **contrast** than absolute luminance

Changes in the amount of contrast in a photo

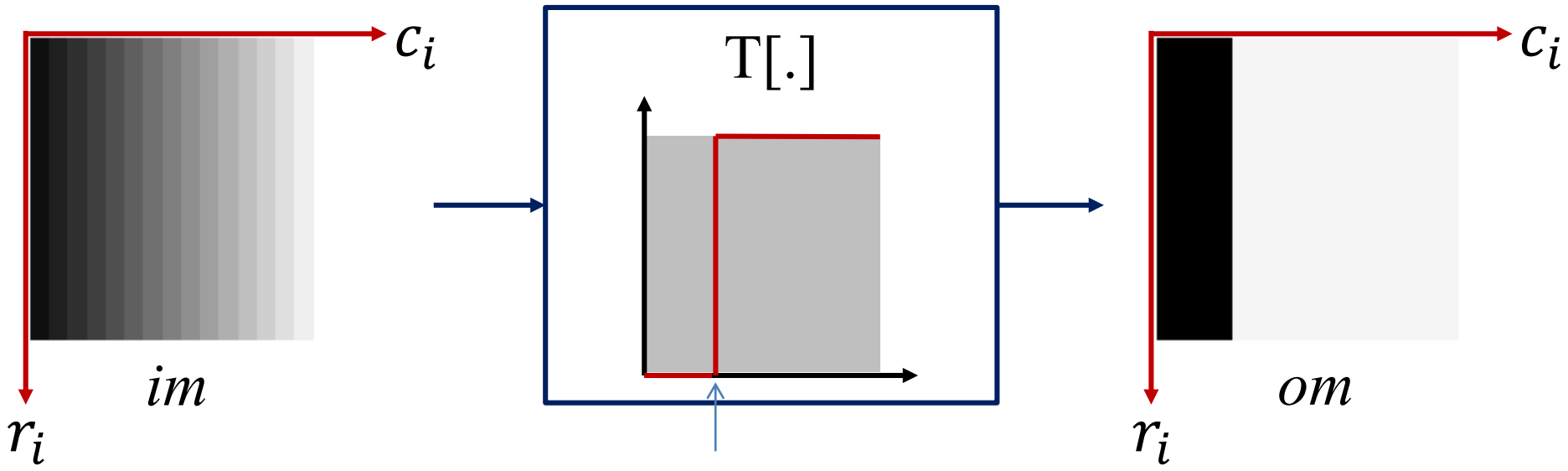
Contrast enhancement

- To produce an image of **higher contrast** than the original image



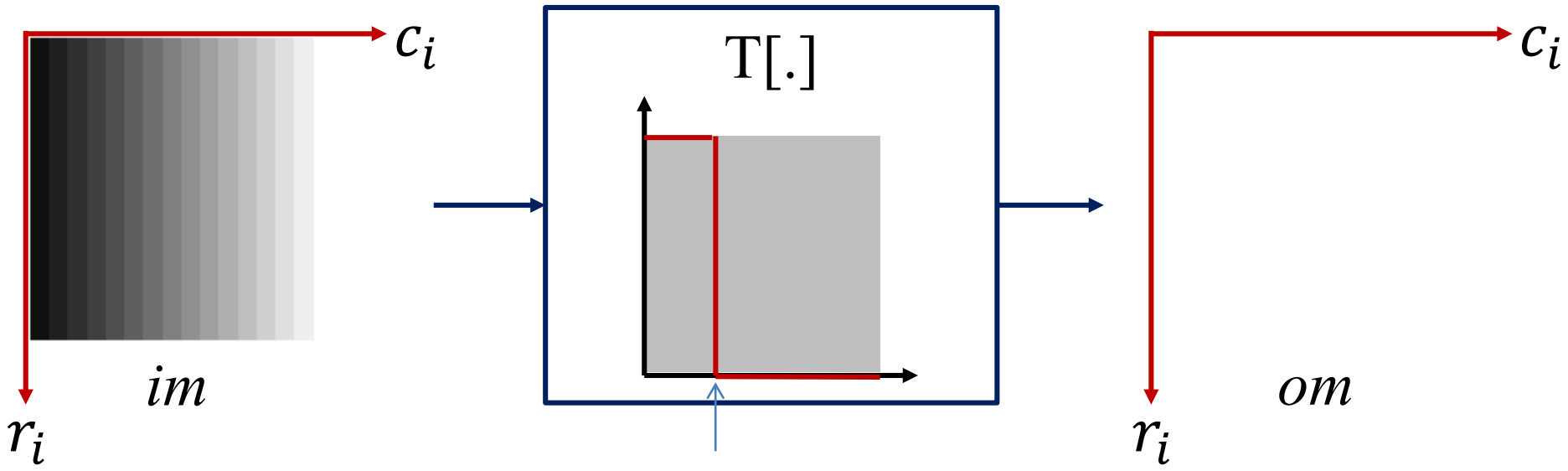
Thresholding

- Is used to produce a binary image



Thresholding

- What will happen with this mapping function ?

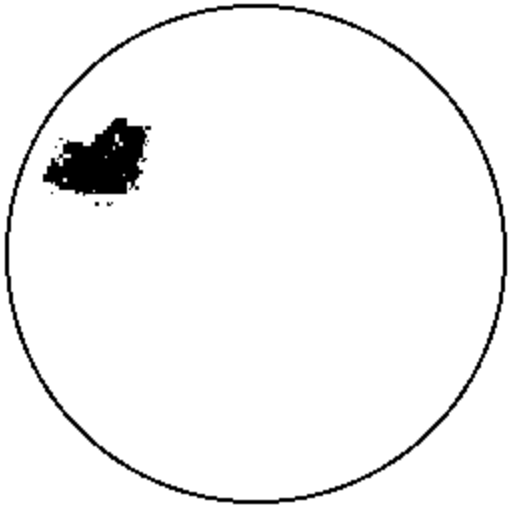


Thresholding

- Bleeding detection in the WCE (Wireless Capsule Endoscopy) images

Thresholding

- Bleeding detection in the WCE images

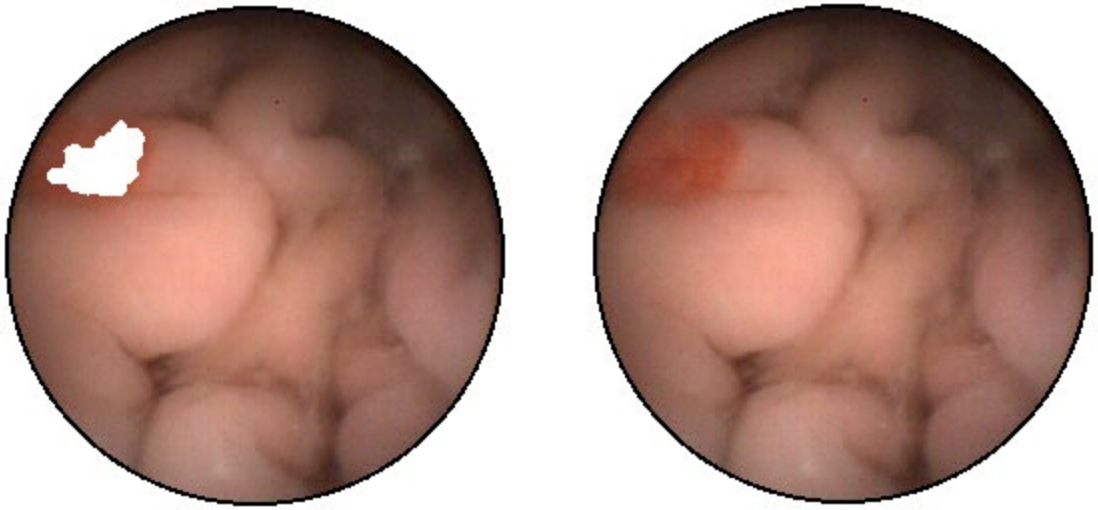


  - RGB → HSI (color space) → blood detection → threshold and masking

Thresholding

- Bleeding detection in the WCE images

Masked bleeding area Original image



Scene text detection and recognition

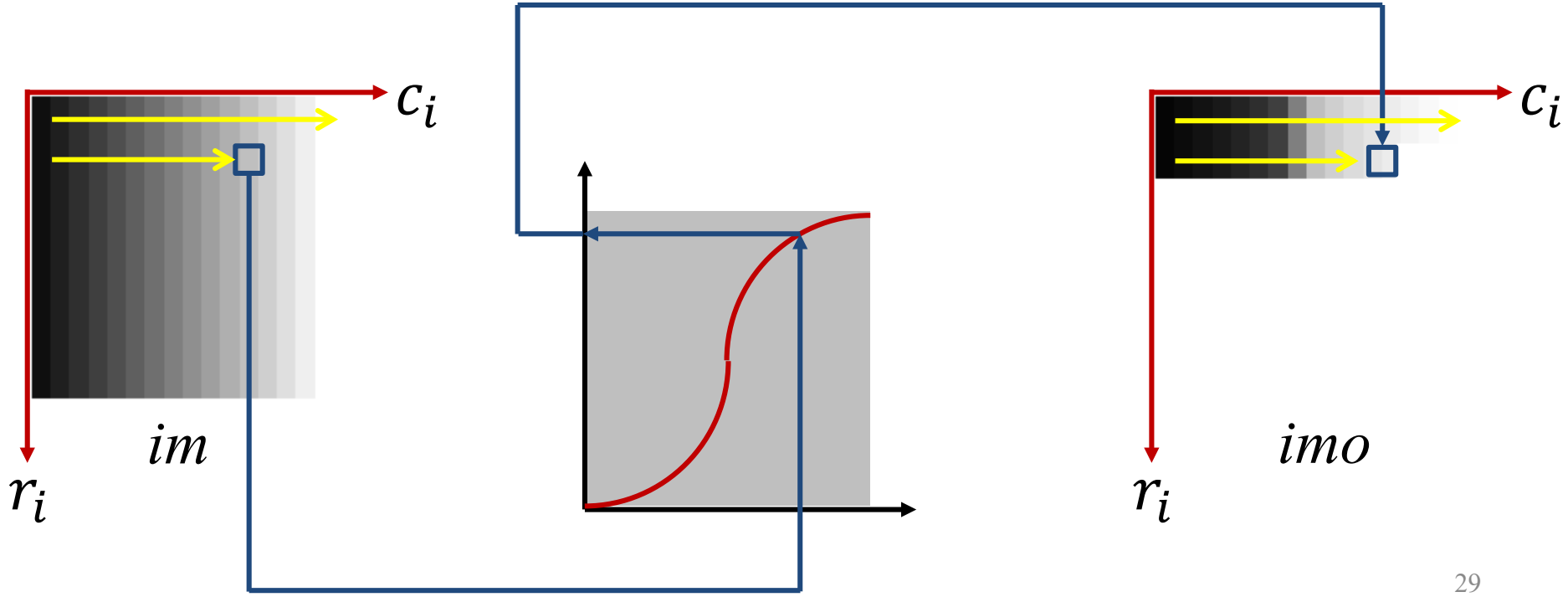
观前街

Guanqian Street

Text-to-speech

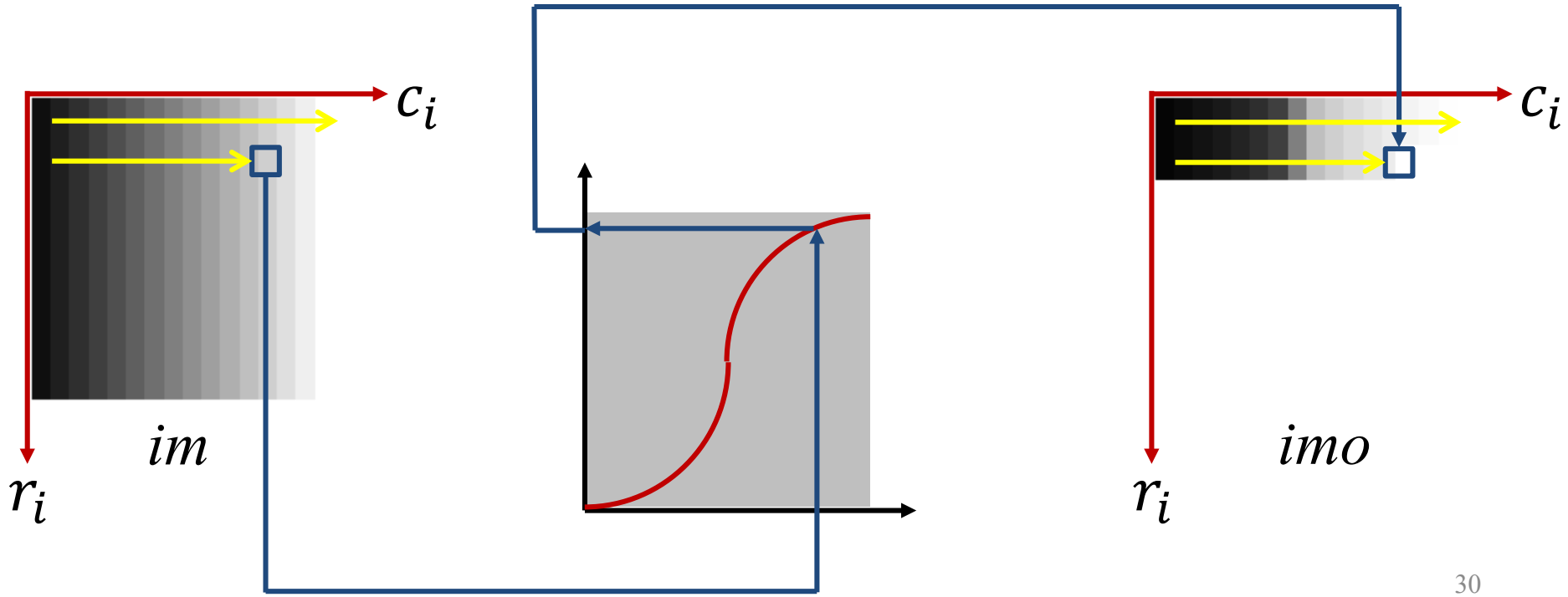
Graphical representation of Mapping function

- The operation $imo(r_i, c_i) = T[im(r_i, c_i)]$ could be described as a mapping function which is shown graphically :



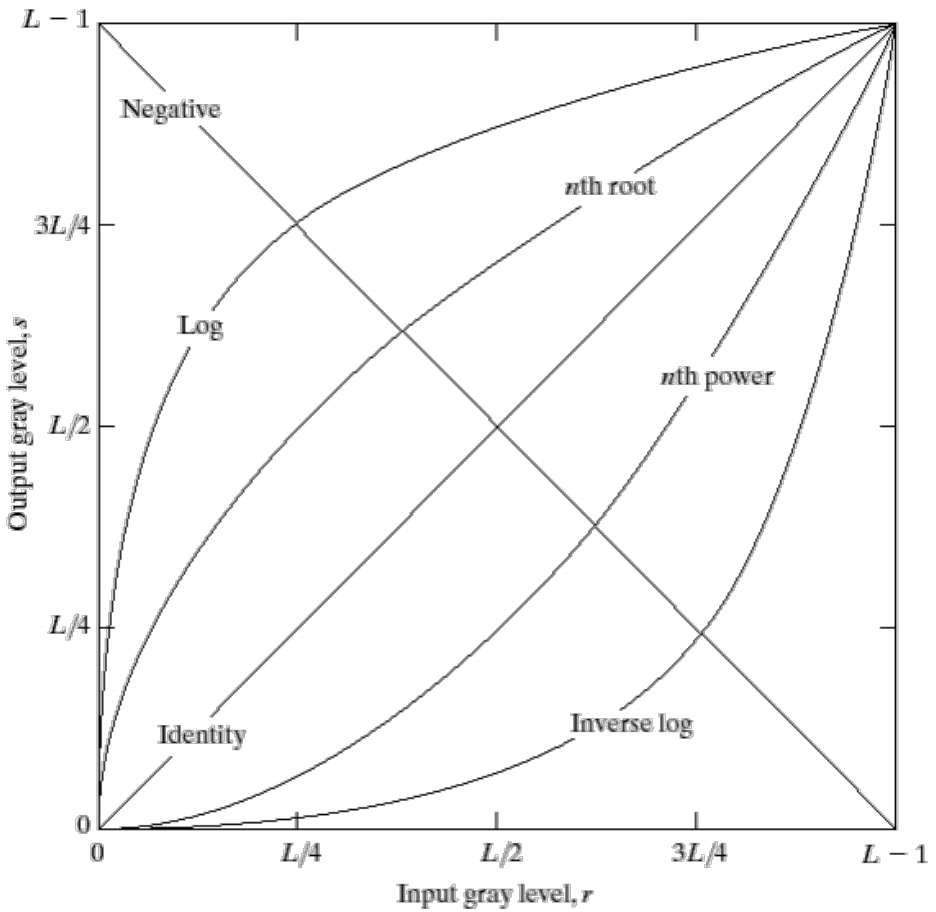
Graphical representation of Mapping function

- The operation $imo(r_i, c_i) = T[im(r_i, c_i)]$ could be described as a mapping function which is shown graphically :



Mapping function

Mapping function



- Some mathematically described mapping functions:
  - Negative and identity transform
  - Power-law transform
  - Log-transformation

Image negative

- Particularly suitable to enhance white/gray details embedded in dark regions

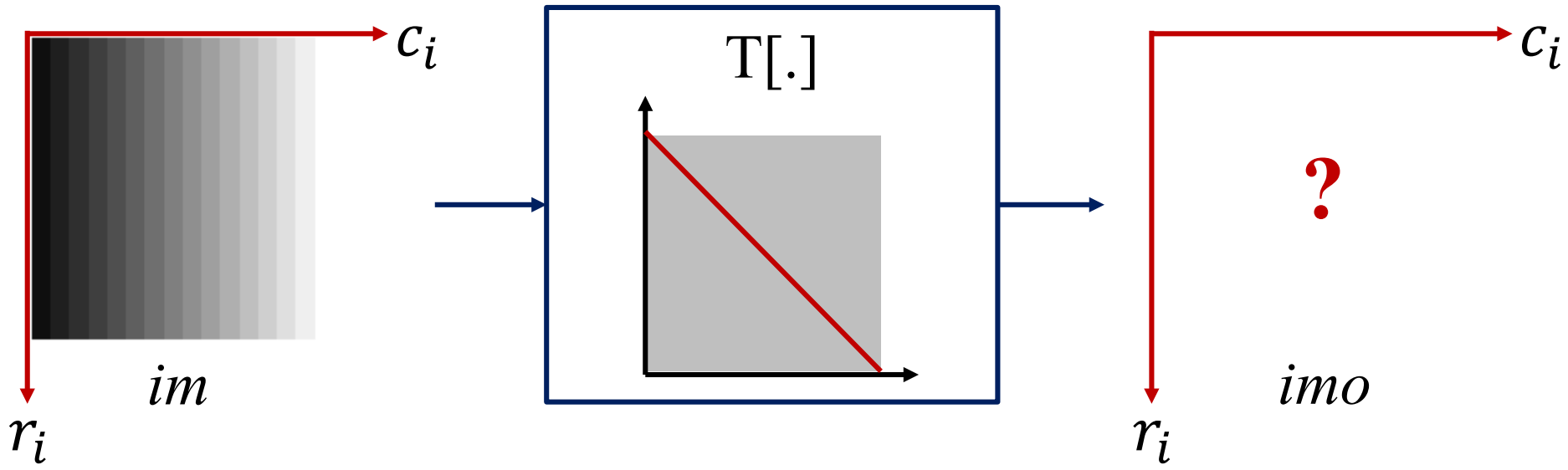
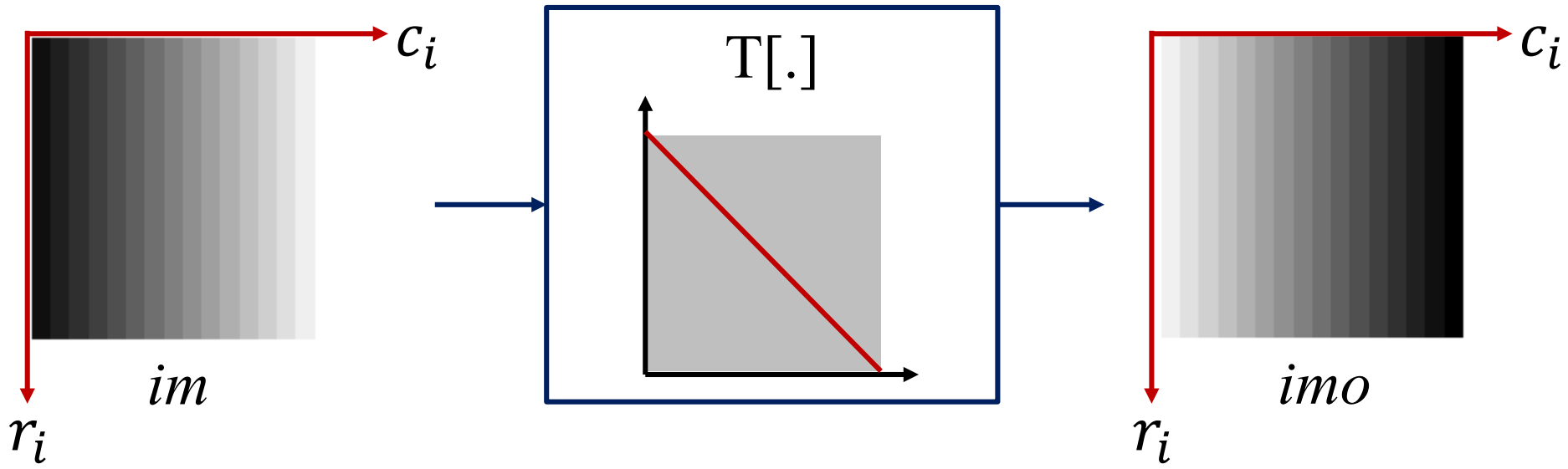


Image negative

- Particularly suitable to enhance white/gray details embedded in dark regions

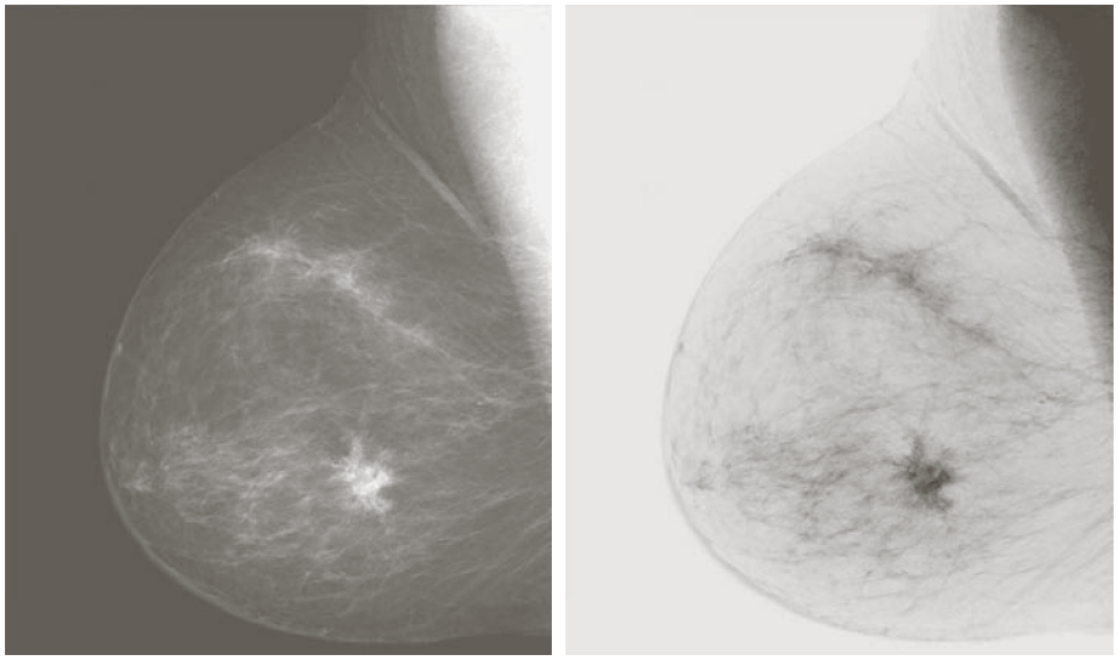


code

Image negative

- Digital mammogram showing a small lesion

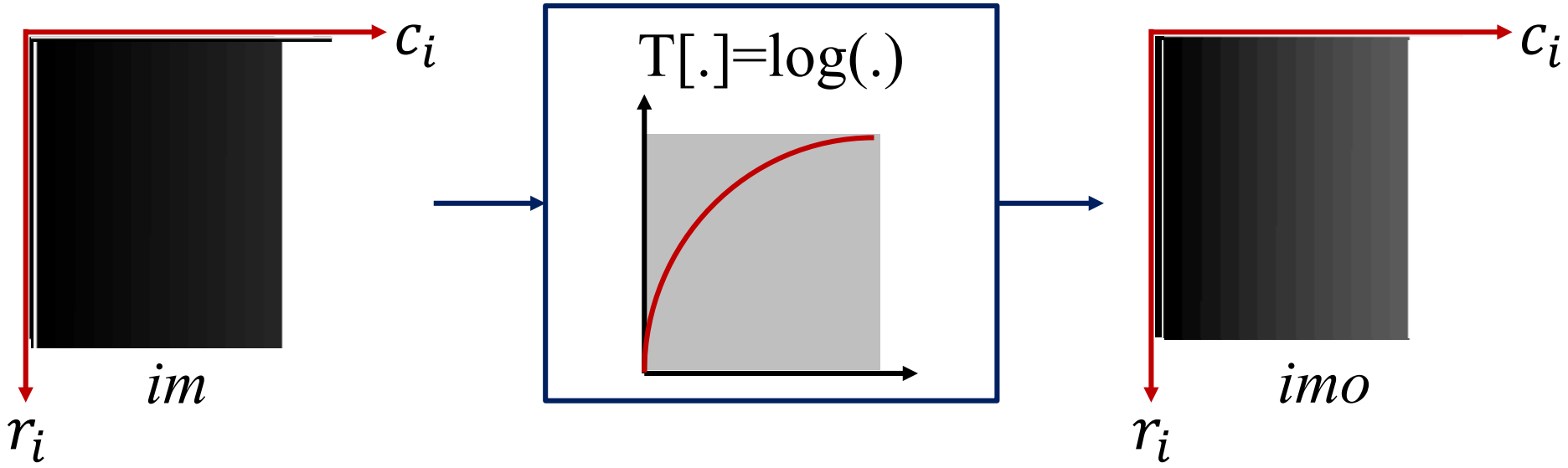
It is easier to see a dirty spot over white background



From: Gonzales & woods; Digital image processing 3rd

Log transforms

- This transform is used to expand the range of dark levels, while compressing the bright level range.
- It is used to compresses the dynamic range of images with large dynamic to make it suitable for display



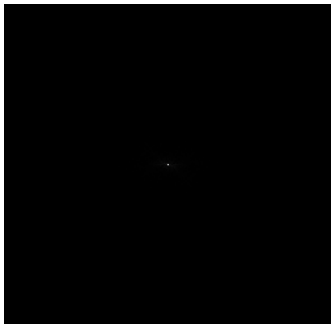
code

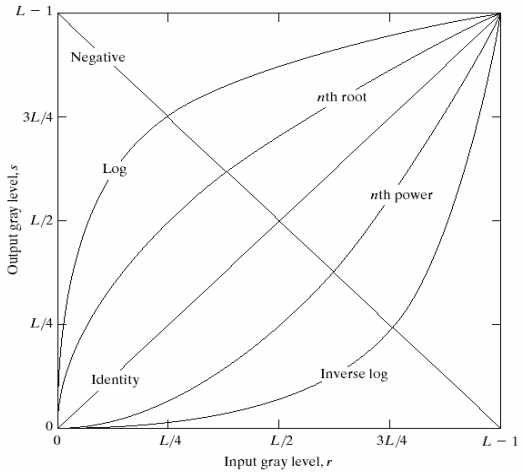
Log transforms

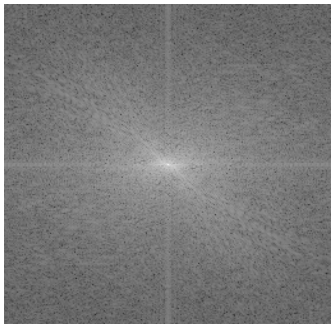
$$v = c\log(1+u)$$

lenna

FFT(lenna)





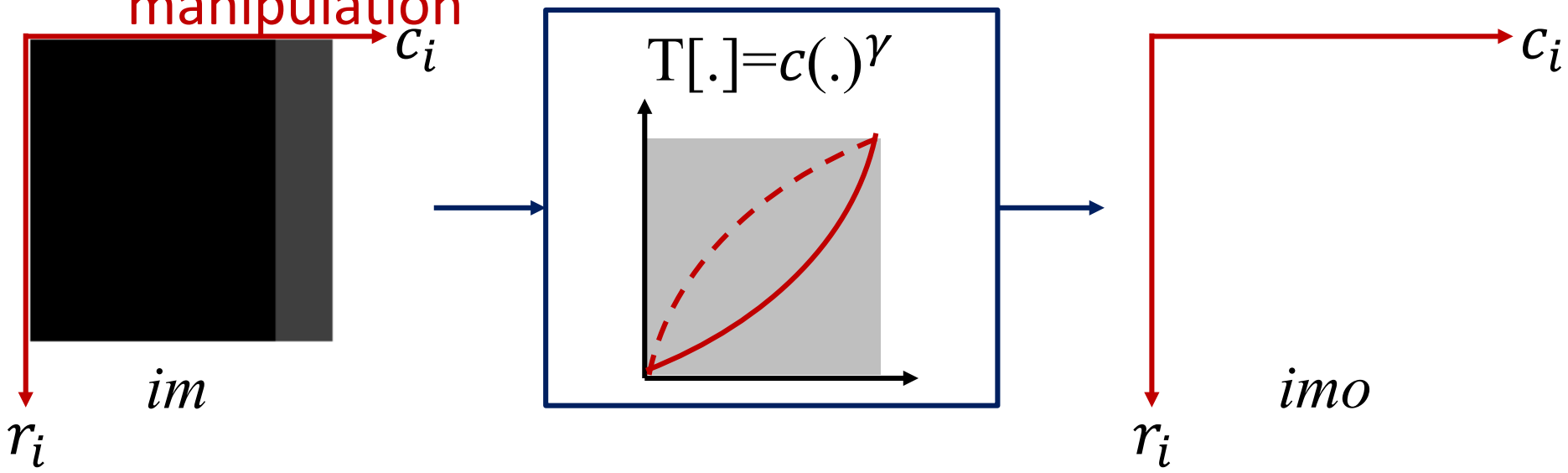


```
>> im = imread('lenna512.bmp')
>> a = abs(fftshift(fft2(double(im))));
>> max(max(a))
   32505395
```


code

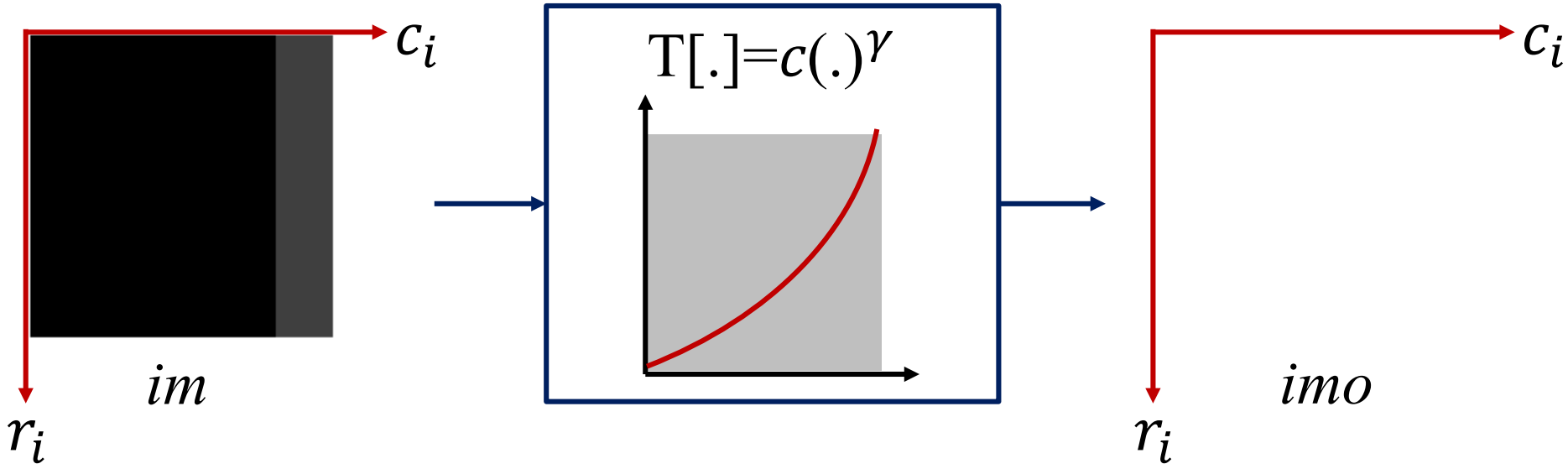
Power-law transforms

- This transform is good because we have a set of curves, rather than only one, which could be used to narrow certain range of gray levels and expand the other range ➔ are useful for general purpose contrast manipulation



Power-law transforms

- $\gamma > 1$
  - Compresses dark range
  - Expands bright values



Power-law transforms

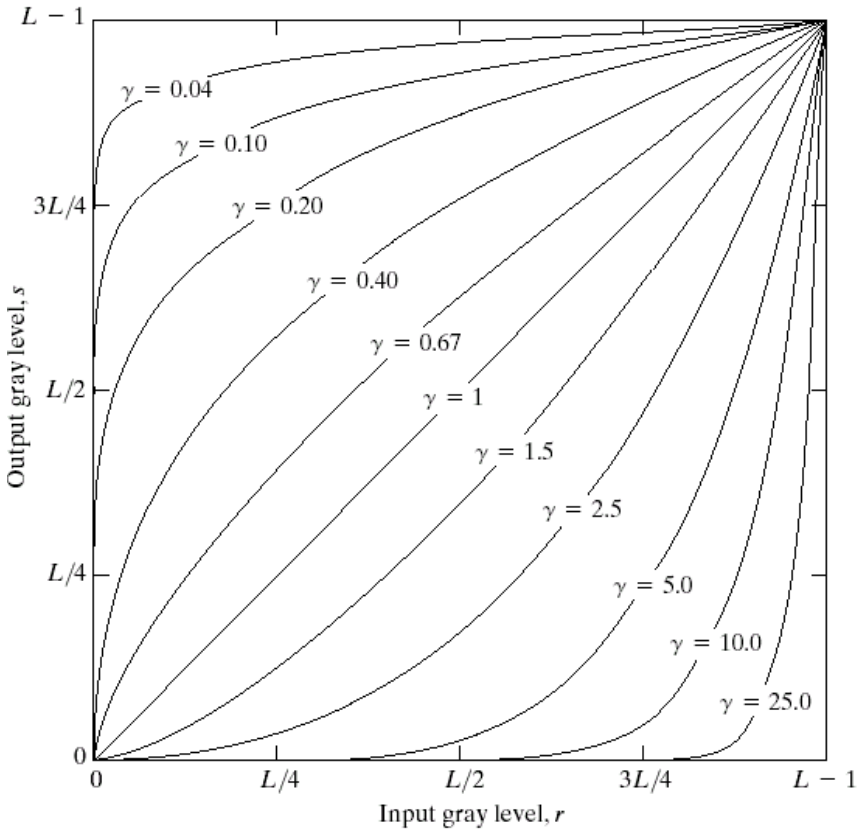
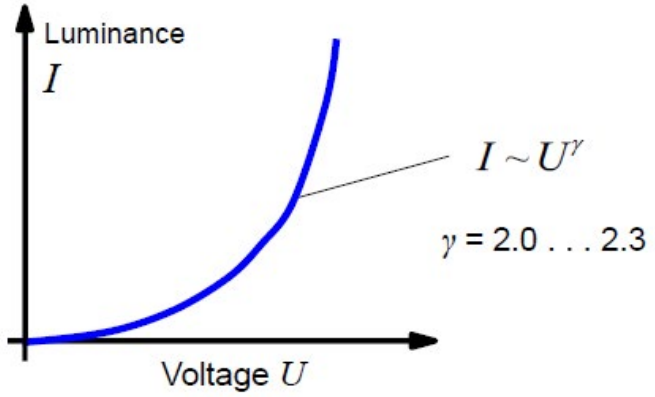


FIGURE 3.6 Plots of the equation $s = cr^{\gamma}$ for various values of γ ($c = 1$ in all cases).

Power-law transforms

- **Variety** of image capturing **devices**, printing, and display **respond** following a **power law**.
  - To **correct** the **power-law response** of this systems **gamma-correction** is used.
  - Example: CRT produce images darker than the intended input (the **intensity-to-voltage** response is power-law, γ=1.8~2.5)

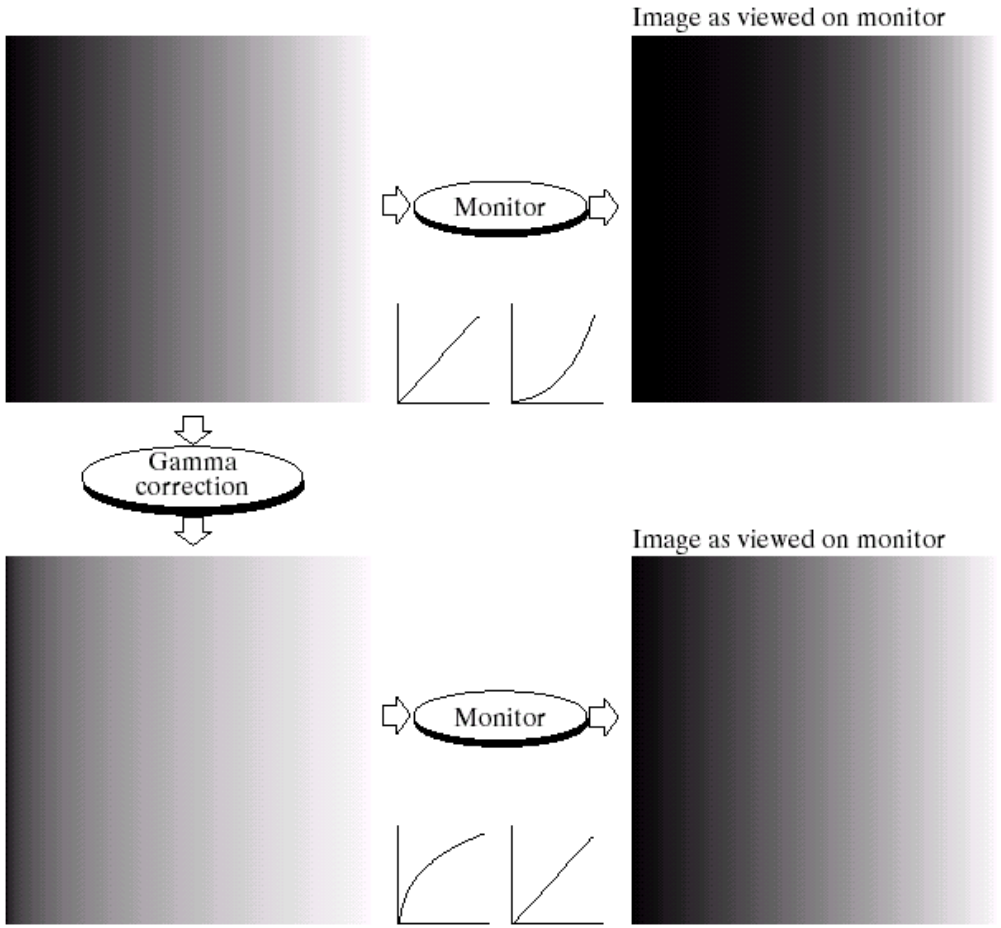


  - Example: Cameras γ = 1.0~1.7

Gamma-correction

a b
c d

FIGURE 3.7
(a) Linear-wedge gray-scale image. (b) Response of monitor to linear wedge. (c) Gamma-corrected wedge. (d) Output of monitor.



Gamma-correction

Original Image

Gamma Correction
c=1, γ=3

Gamma Correction
c=1, γ=4

Gamma Correction
c=1, γ=5

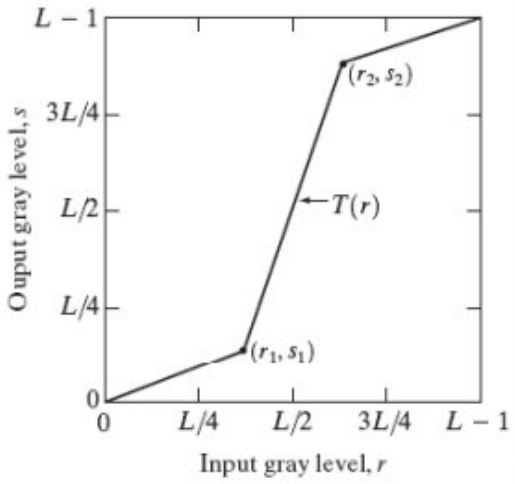
Piecewise-Linear Transformation Functions

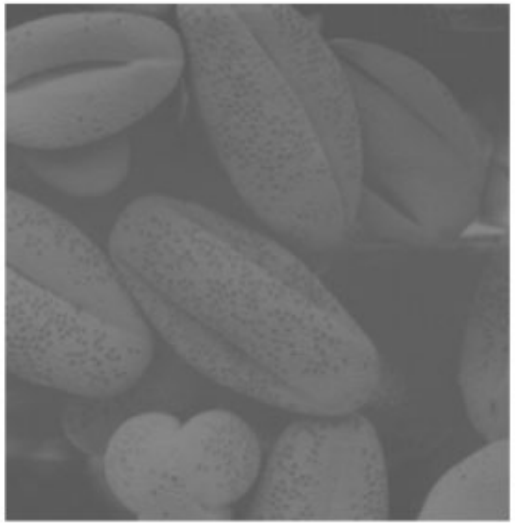
Piecewise-Linear Transformation Functions

- Different from previous functions which could be described mathematically, this category of functions is described graphically, this makes them suitable for variety of applications

- **Allows** to make the mapping function **arbitrary complex**
  - Contrast enhancing
  - Gray level slicing

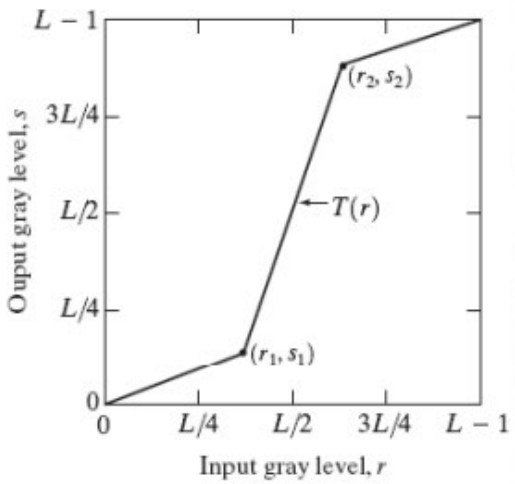
Contrast enhancing by stretching

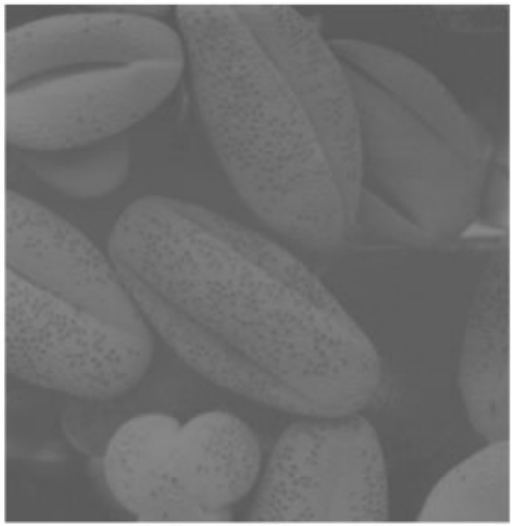


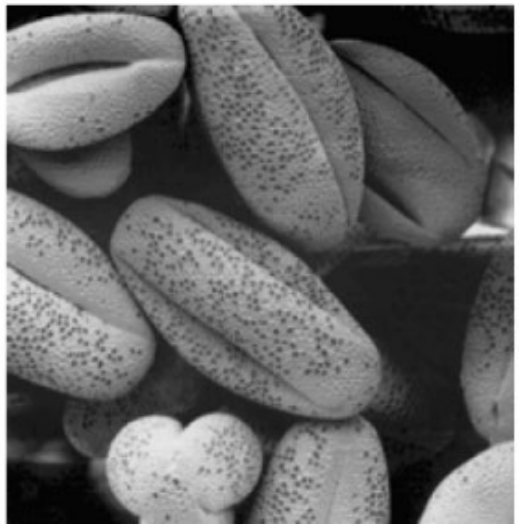


- The function is assumed single valued and monotonically increasing function, aims to **increase** the **dynamic range** of the gray levels

Contrast enhancing by stretching







- The function is assumed single valued and monotonically increasing function, aims to **increase** the **dynamic range** of the gray levels
- **How** to decide the **shape** of the mapping function ?

Gray level slicing

- Highlighting a specific range of gray levels.

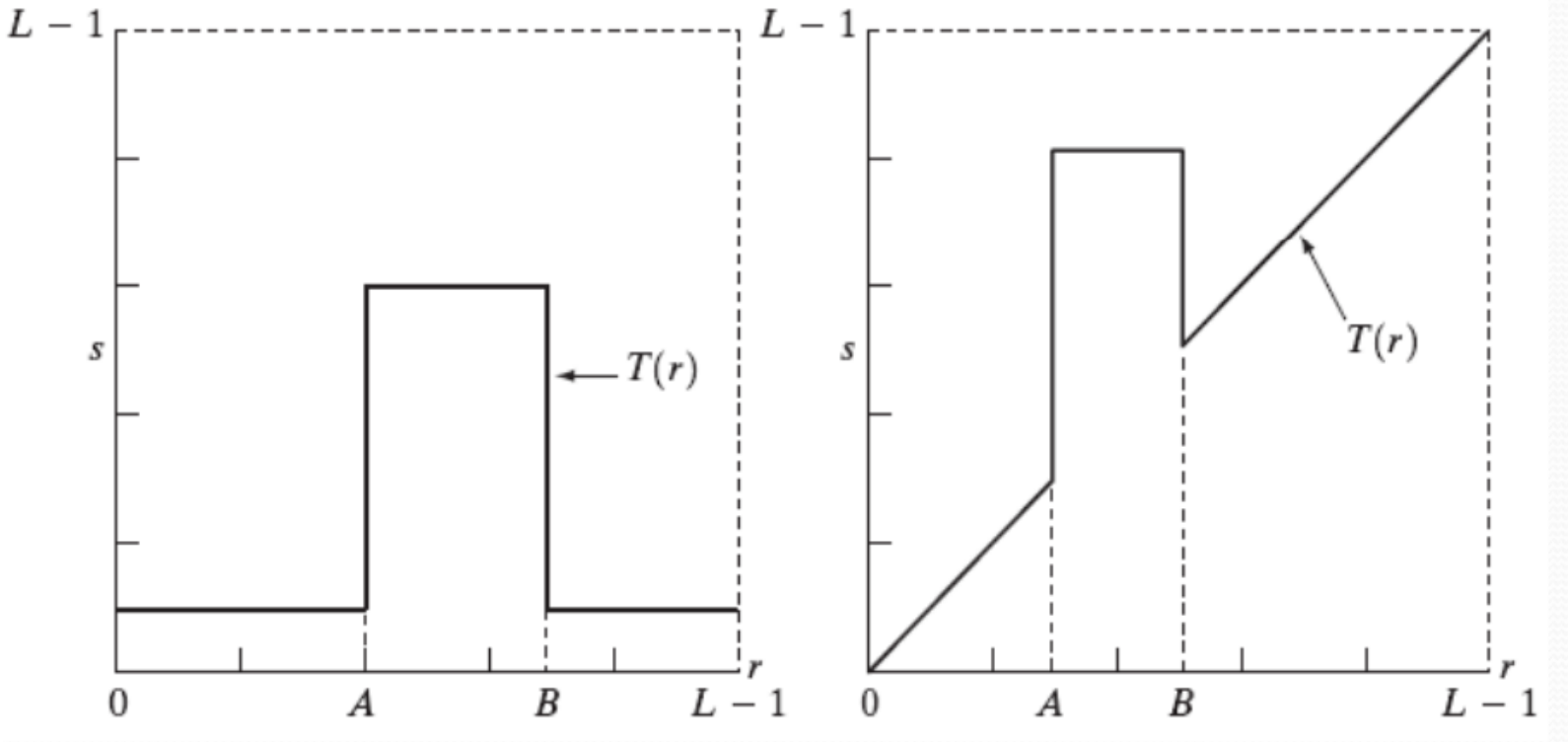
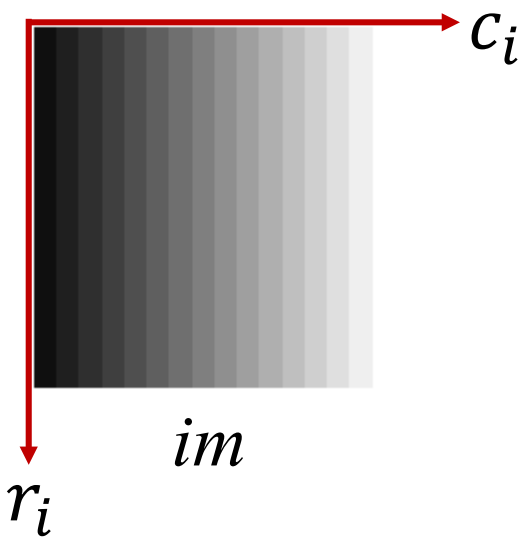
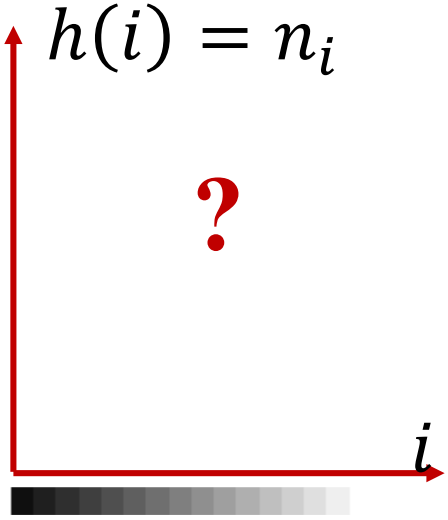


Image histogram

Image histogram

- Image histogram is a graphical representation of the distribution of gray levels in an image
- Is a discrete function $h(i) = n_i$ where i is a gray level and n_i is the number of pixels having that gray level





Normalized image histogram

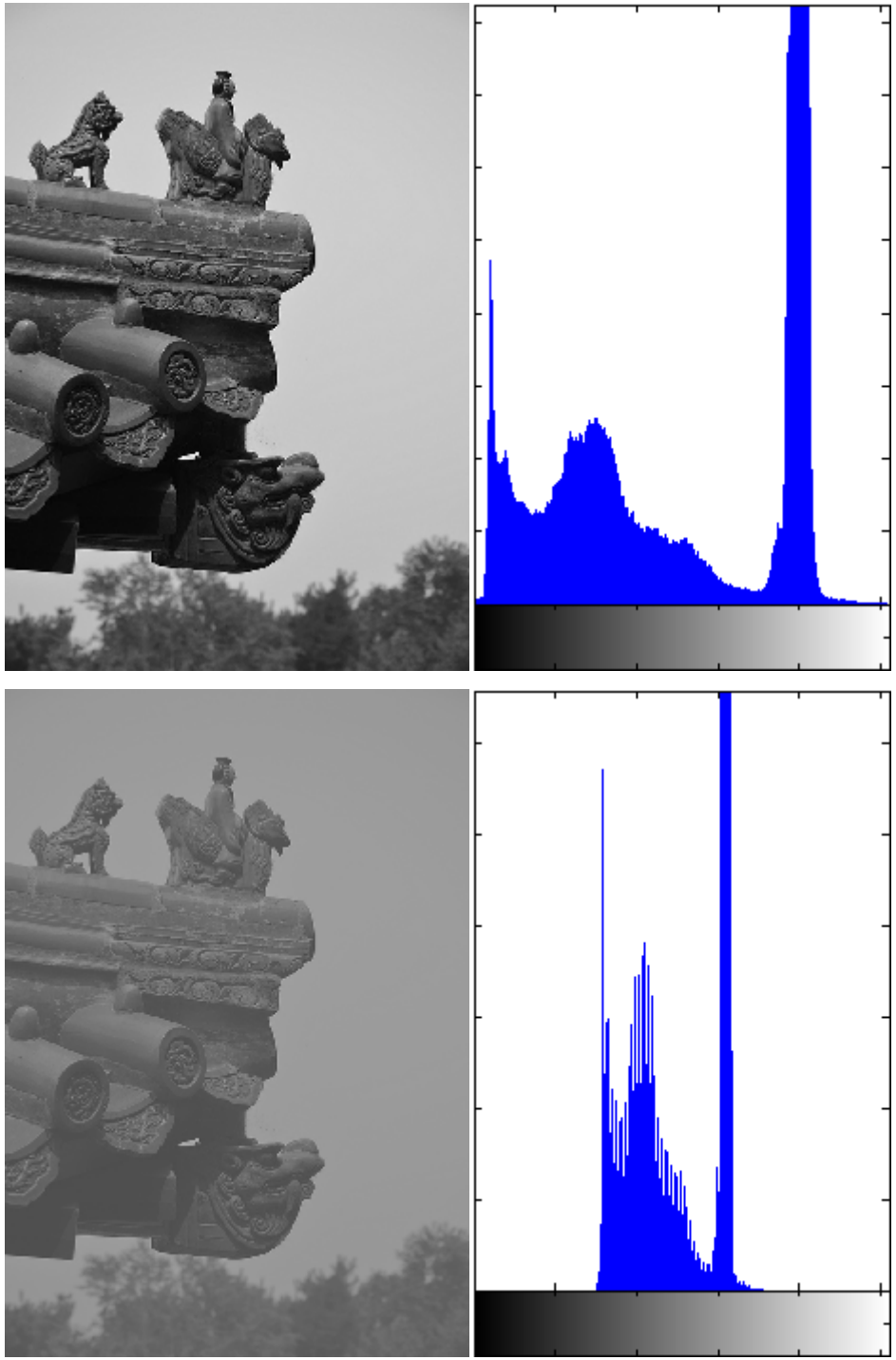
- In the Normalized Histogram, h_n, each value of the histogram is divided by the total number of pixels in the image $N = \sum_{\forall i} n_i$

$$h_n(i) = \frac{n_i}{N}$$

- When $N \to \infty$ then $h_n(i)$ represents the estimate of the probability of the gray level i.
  - In this case $h_n(i)$ becomes equivalent to the Probability Density Function (pdf) of a random *variable representing the gray level.*

Image histogram

- “unbalanced” histograms do not fully utilize the dynamic range
  - Low contrast image: narrow luminance range
  - Under-exposed image: concentrating on the dark side
  - Over-exposed image: concentrating on the bright side
- “balanced” histogram gives more pleasant look and reveals rich details

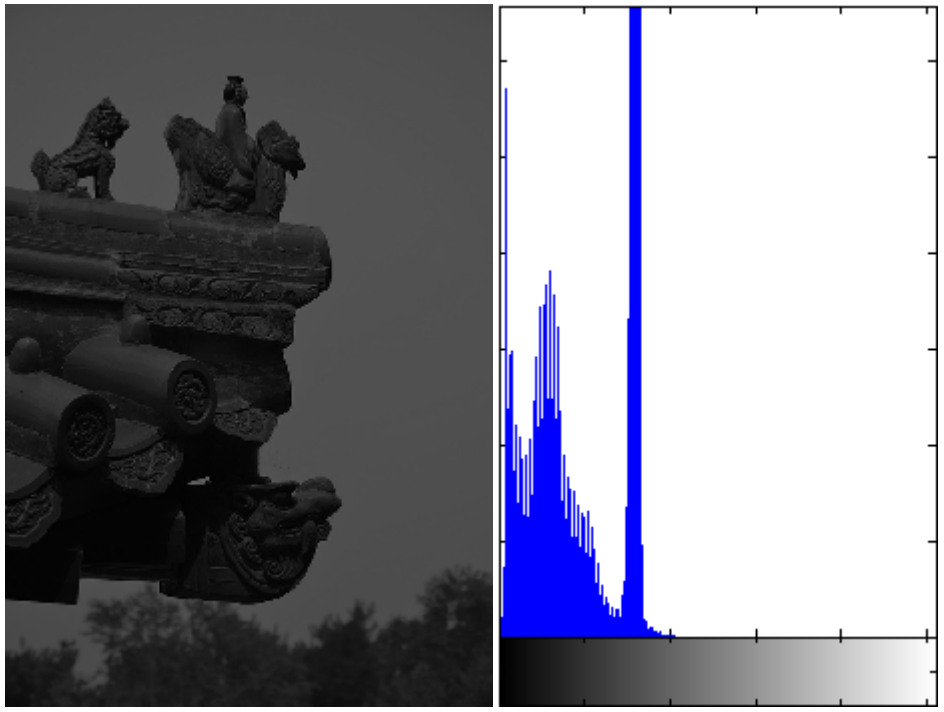


- High-contrast image
  - Covers the entire range of possible gray levels, tend to distribute them uniformly.
  - Exhibits a large variety of gray tones
  - We call it also “balanced” histogram

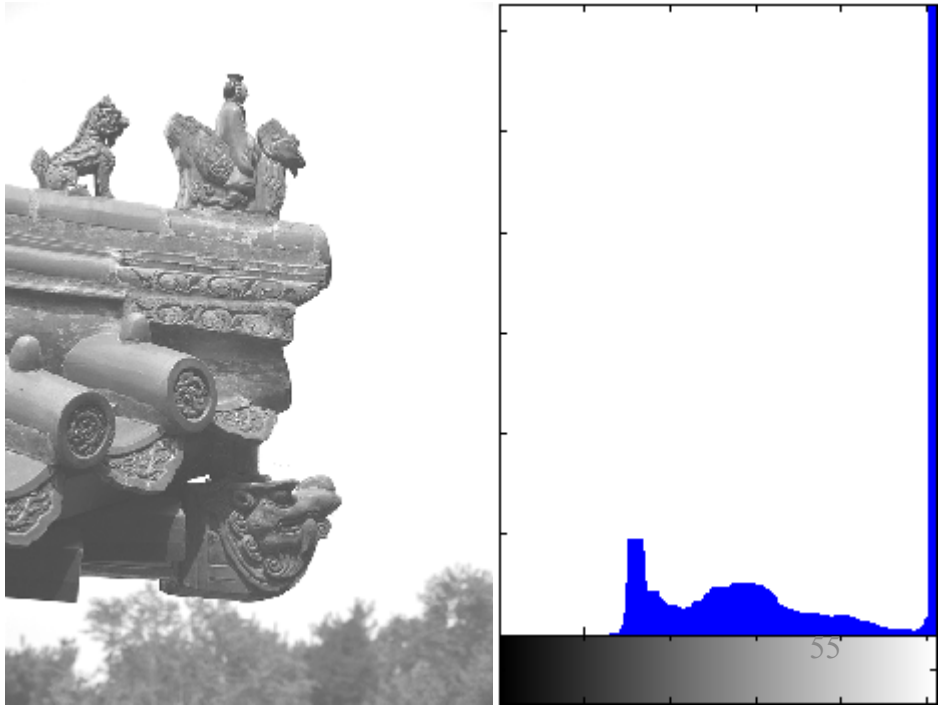
- Low-contrast image
  - The histogram is narrow and centered toward the middle of the gray scale.
  - So the image looks dull and washed-out.

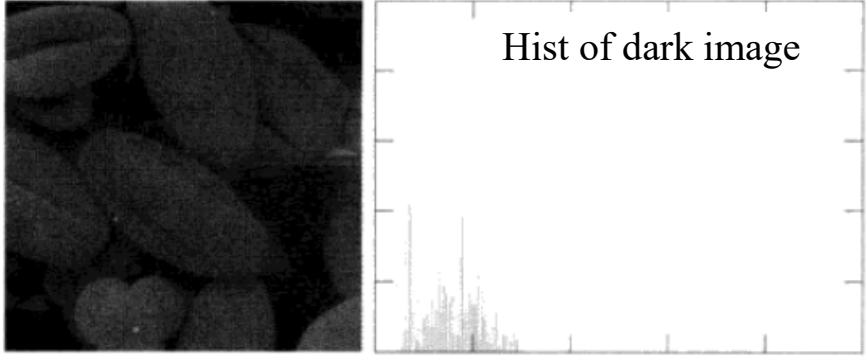
code

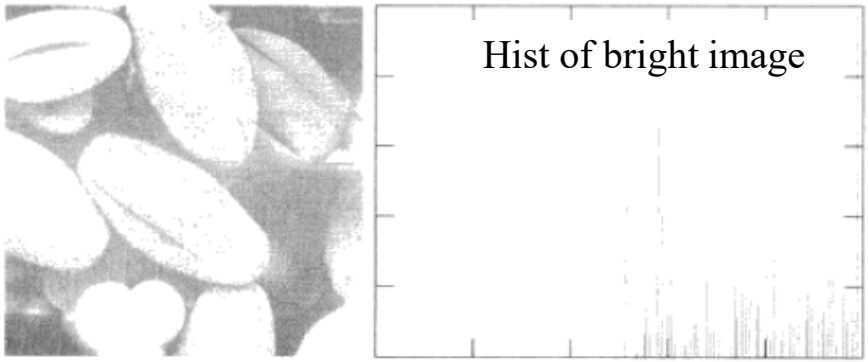
- Dark image
  - The histogram is biased toward the dark side of the gray scale.
  - In photography we call it: Under-exposed image

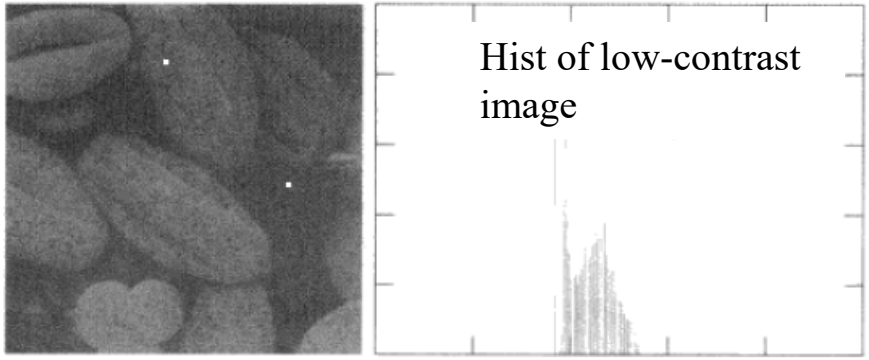


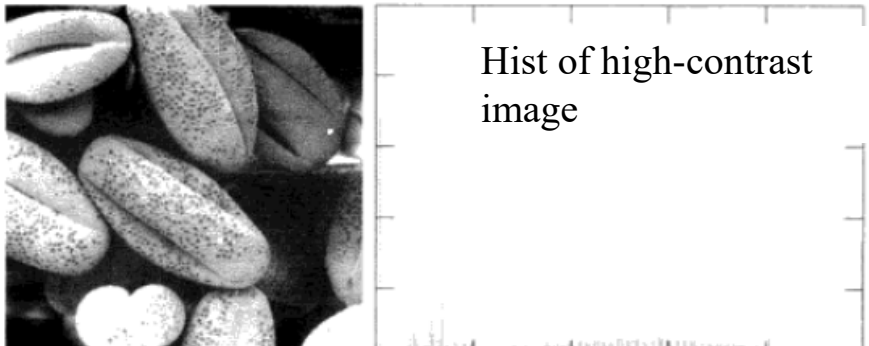
- Bright image
  - The histogram is biased toward the bright side of the gray scale.
  - In photography we call it: over-exposed image




Hist of dark image


Hist of bright image


Hist of low-contrast image

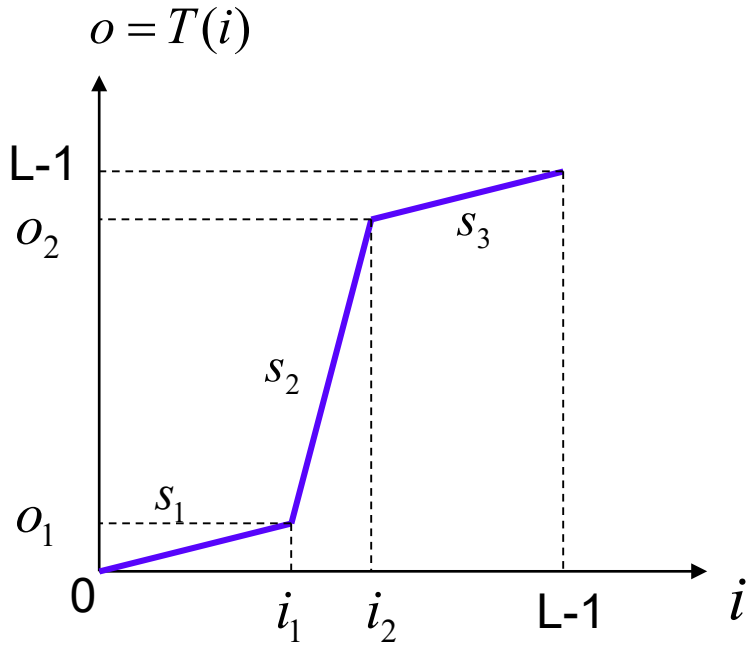

Hist of high-contrast image

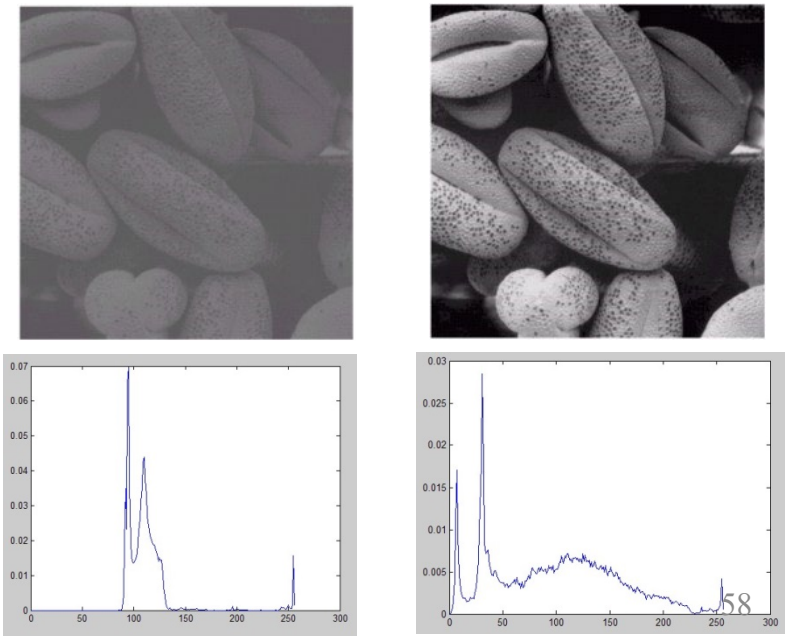
Low dynamic range

- A low dynamic range image could be due to the following reasons:
  - Poor illumination
  - Wrong setting of the lens aperture and/or shutter speed
- In a low contrast image the full dynamic range is not exploited
- Contrast enhancing by stretching
  - Is a simple technique to expand the histogram to fill entire gray-scale range

Contrast enhancing by stretching

- Piece-wise linear function could be designed based on the histogram of the original image,
  - The slope in the stretching region is greater than 1.
  - The slope in the compressing region is smaller than 1.





Histogram Equalization

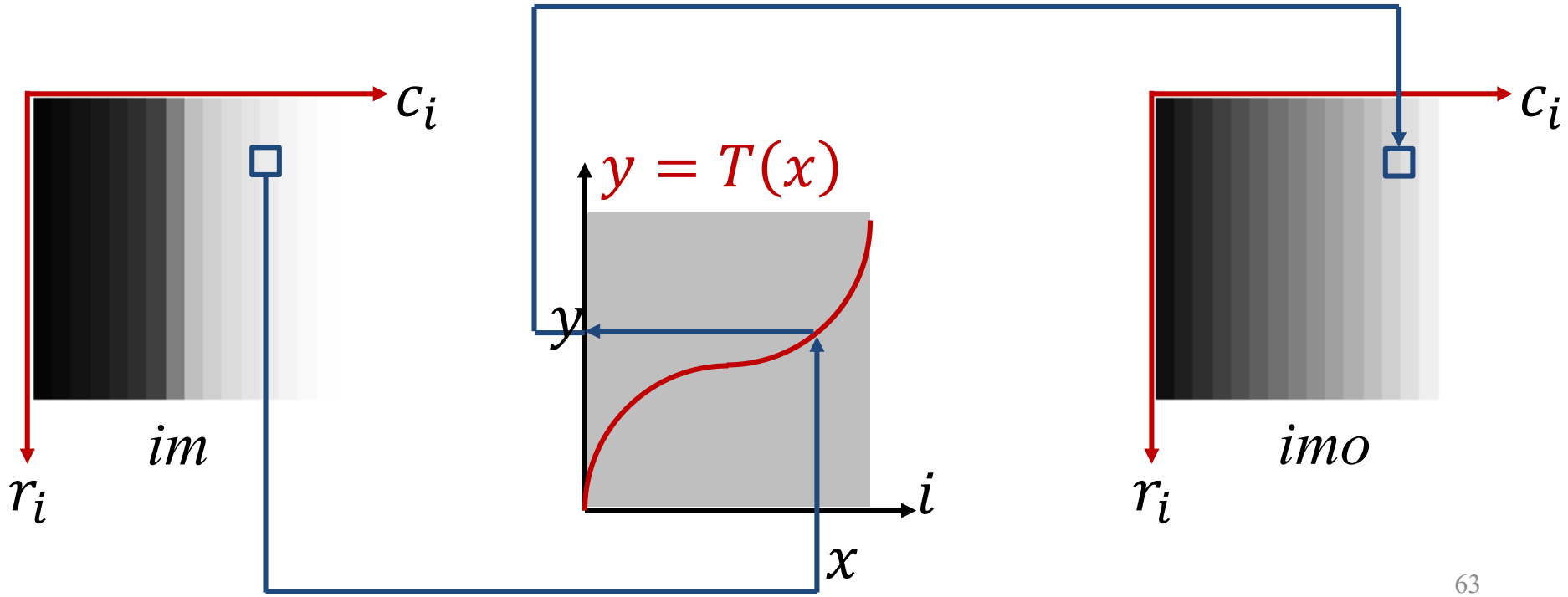
Histogram Equalization

- It is a technique to enhance contrast in 'natural' images, and it is a kind of nonlinear image-to-image mapping

- The target is to find gray level transformation function T to map the image intensity i such that the histogram of $T(i)$ is 'equalized'
  - An image with a perfectly flat histogram contains largest possible amount of information

Histogram Equalization

- The goal is to map each luminance level to a new value such that the output histogram has approximately uniform distribution.

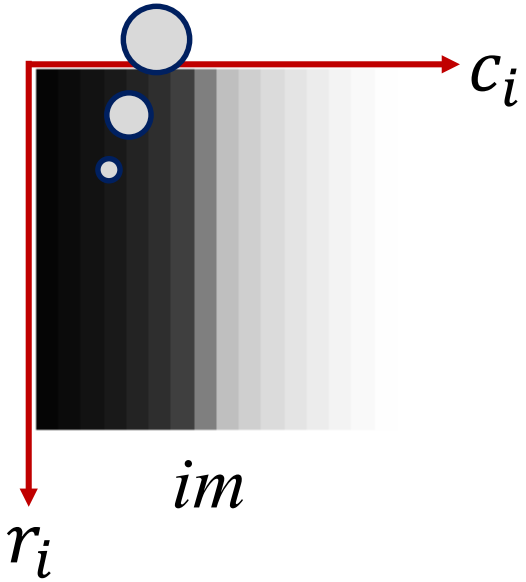


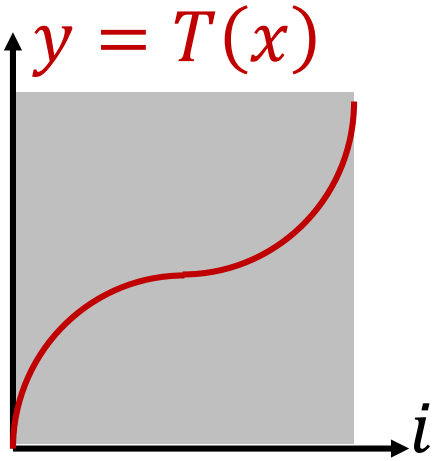
Histogram Equalization

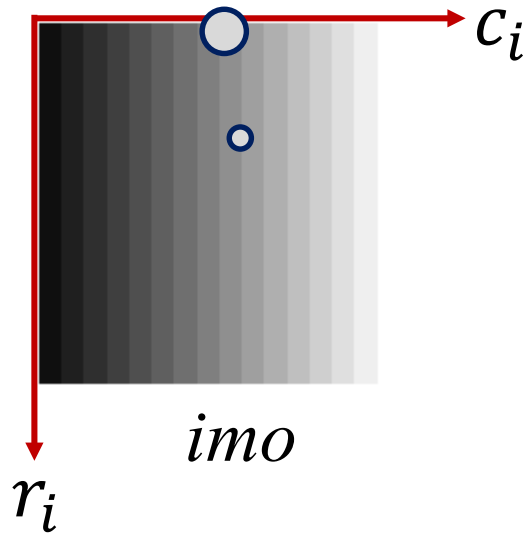
Here you distinguish more gray bands ➔ more information

Can you distinguish the various dark band ?

to map each lumina… such that the output i… has …ately uniform distribution.

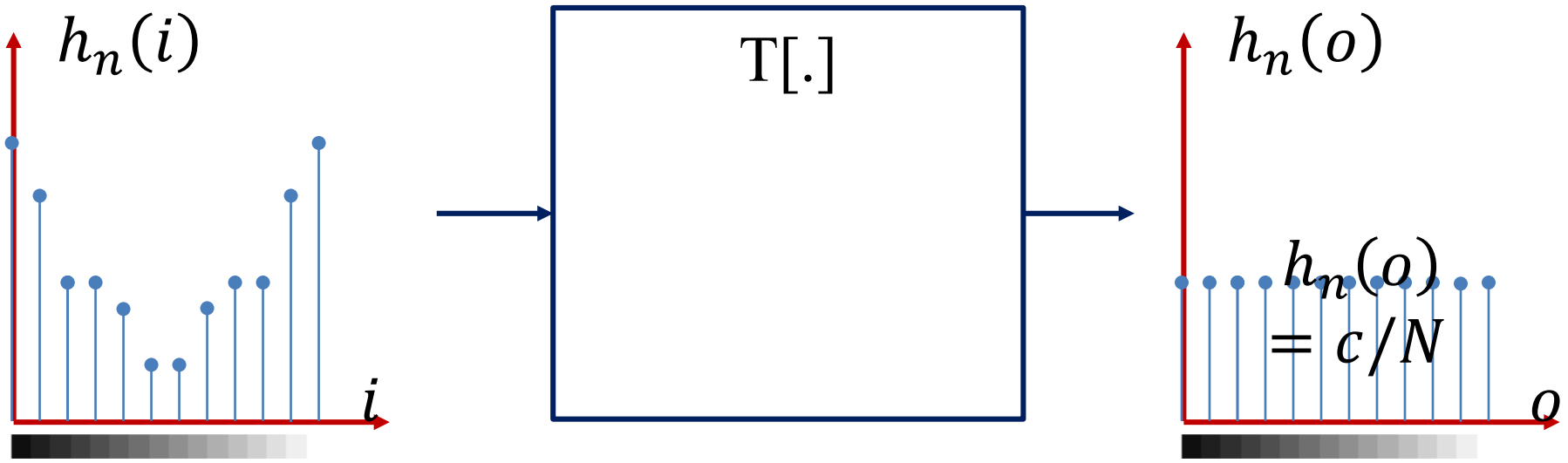






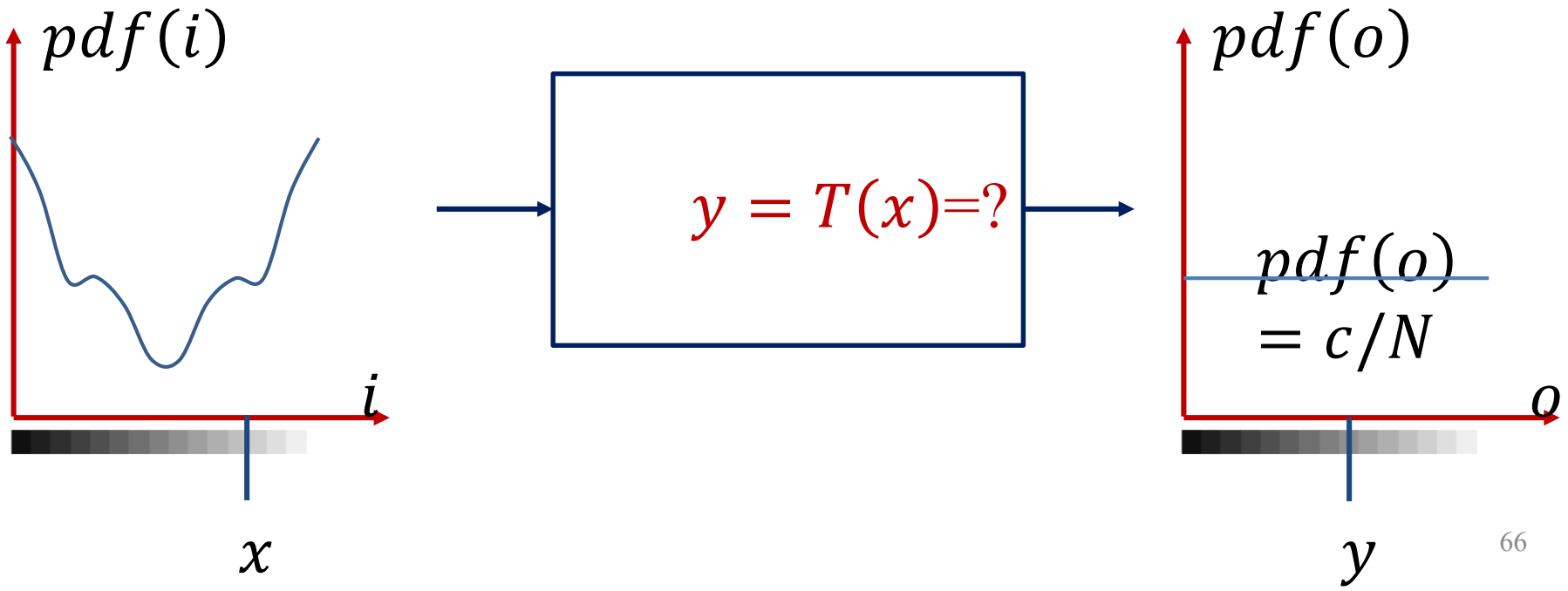
Histogram Equalization

- How to find $y = T(x)$



Histogram Equalization

- The first step to simplify this process is to assume that the functions are continuous.



Histogram Equalization

- To find $T(x)$ we have to notice that the number of re-mapped pixels is the same as the originals (preserving the quantities)

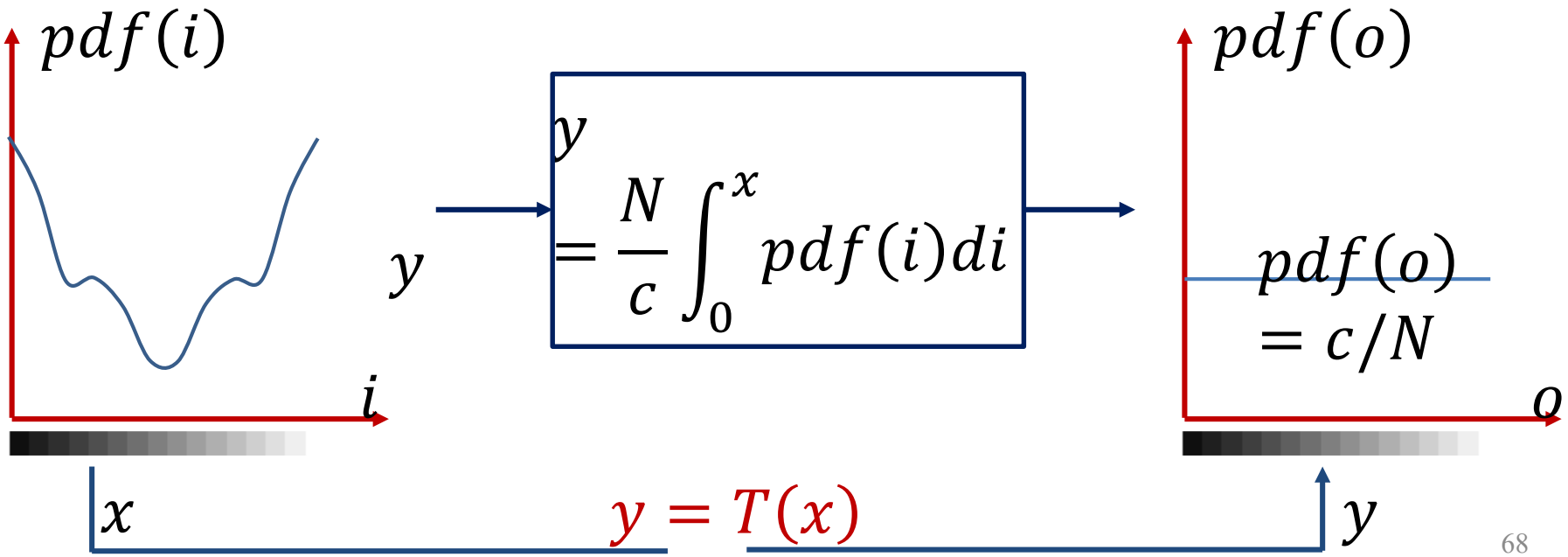
$$\int_0^x pdf(i)di = \int_0^y pdf(o)do$$
$$= \frac{c}{N}\int_0^y do$$
$$= \frac{c}{N}y$$

The cdf (Cumulative Distribution Function)

- Finally $\frac{c}{N}y = \int_0^x pdf(i)di$

Histogram Equalization

- Practically how to find $y = T(x)$



Histogram Equalization

- Discrete formulation

$$p_r(r_k) = \frac{n_k}{MN}, \quad k = 0, 1, 2, \cdots, L-1$$

$$s_k = T(r_k) = (L-1)\sum_{j=0}^{k} p_r(r_j) = \frac{(L-1)}{MN}\sum_{j=0}^{k} n_j, \quad k = 0, 1, 2, \cdots, L-1$$

Here: *MN* is the number of total pixels;

n_k is the number of pixels whose gray value equals *k*

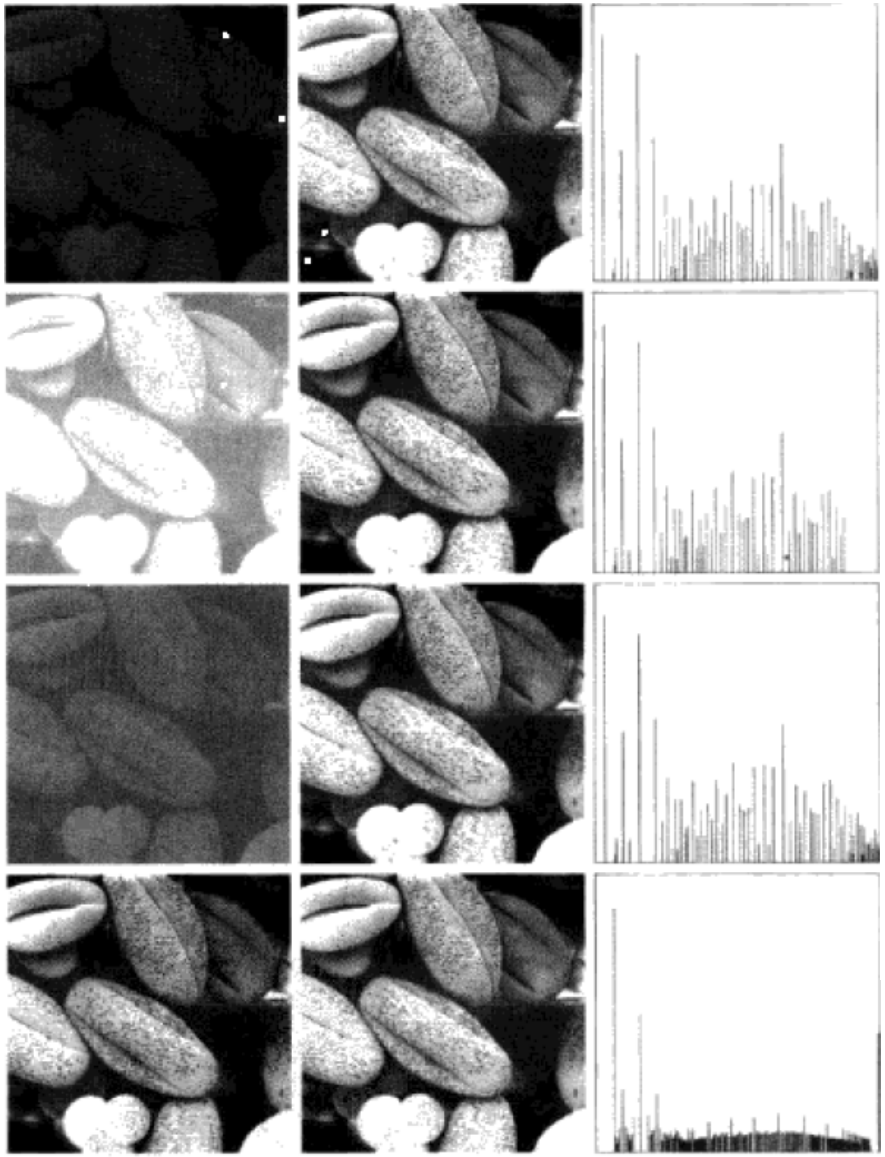
L: gray level

r_k is the gray value of the pixels in the original image

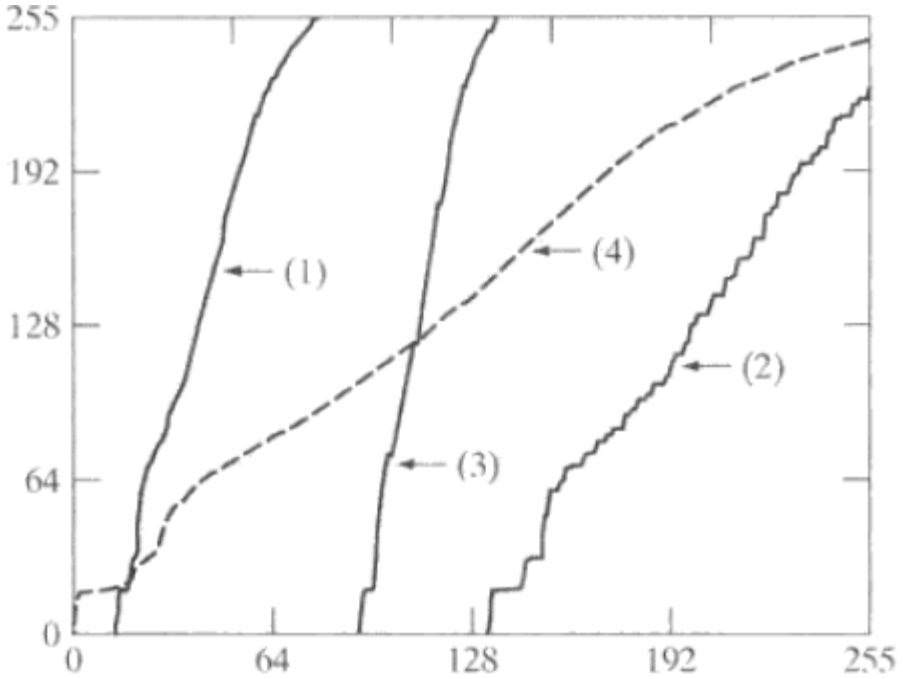
s_k is the gray value of the pixels in the new image

- Histogram equalization transformation function can be automatically obtained based on the information of the histogram of the input image.

Histogram Equalization



- Left column shows images with different contrast



Example

Image=0 2 5 6

1 2 4 5

1 3 3 0

Histogram:

Gray	Count	Prob.
0	2	0. 167
1	2	0. 167
2	2	0. 167
3	2	0. 167
4	1	0. 083
5	2	0. 167
6	1	0. 083
7	0	0

What is the variable?
What is the response?

Histogram Equalization

$$s_k = T(r_k) = (L-1)\sum_{j=0}^{k} p_r(r_j) = \frac{(L-1)}{MN}\sum_{j=0}^{k} n_j, \quad k = 0, 1, 2, \cdots, L-1$$

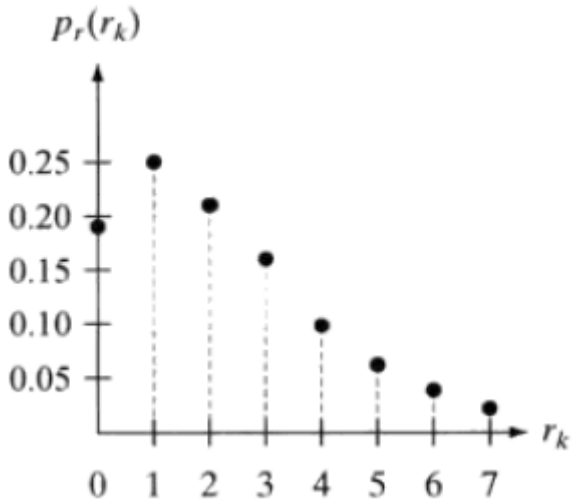
New Image=1 3 6 7

2 3 5 6

2 5 5 1

Example 2

r_k	n_k	$p_r(r_k) = n_k/MN$
$r_0 = 0$	790	0.19
$r_1 = 1$	1023	0.25
$r_2 = 2$	850	0.21
$r_3 = 3$	656	0.16
$r_4 = 4$	329	0.08
$r_5 = 5$	245	0.06
$r_6 = 6$	122	0.03
$r_7 = 7$	81	0.02

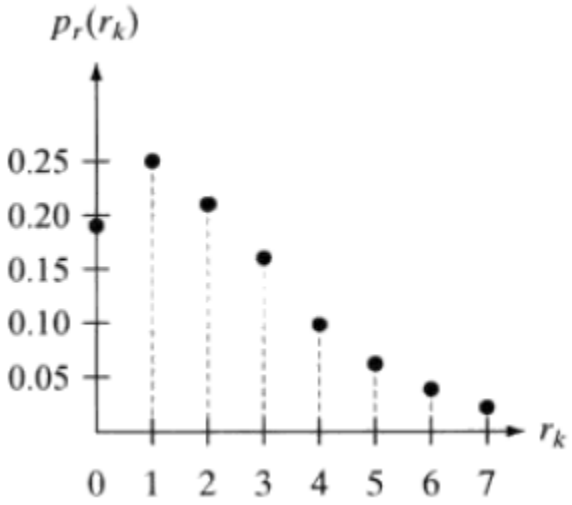


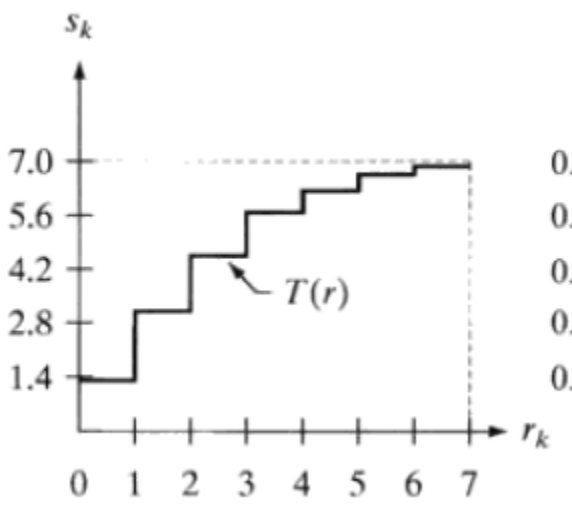
Example 2

$$s_0 = T(r_0) = 7\sum_{j=0}^{0} p_r(r_j) = 7\,p_r(r_0) = 1.33$$

$$s_1 = T(r_1) = 7\sum_{j=0}^{1} p_r(r_j) = 7\,p_r(r_0) + 7\,p_r(r_1) = 3.08$$

$$s_2 = 4.55,\ s_3 = 5.67,\ s_4 = 6.23,\ s_5 = 6.65,\ s_6 = 6.86,\ s_7 = 7.00$$





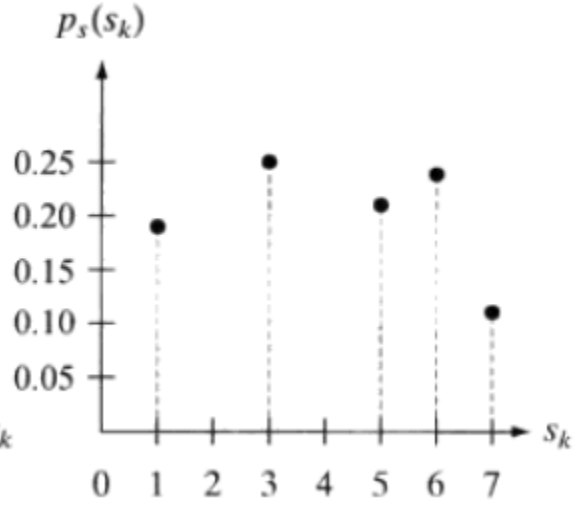


Image Averaging

- Consider a noisy image g(x,y) formed by addition of noise η(x,y) to an original image.
- $$g(x,y) = f(x,y) + \eta(x,y)$$
- Here we assume that at every pair of coordinates (x,y) the noise is uncorrelated and has zero average value.
- We try to reduce the noise content by adding a set of noisy images, $g_i(x,y)$.
- It can be shown that as the number of noisy images used in the averaging process increases, the noisy image resembles the original image f(x,y).

Image Averaging

- Let $\breve{g}(x,y)$ is obtained by averaging K different noisy images.
- $\breve{g}(x,y) = (1/K)\ \Sigma\ g_i(x,y)$ where i varies from 1 to K.
- Then
- $E\{\ \breve{g}(x,y)\} = f(x,y)$ and
- $\sigma^2\ \breve{g}(x,y) = (1/K)\ \sigma^2\ \eta(x,y)$
- Here $E\{\ \breve{g}(x,y)\}$ is the expected value of $\breve{g}$ and $\sigma^2\ \breve{g}(x,y)$ and $\sigma^2\ \eta(x,y)$ are the variances of $\breve{g}$ and η, all at coordinates (x,y).
- The standard deviation at any point in the average image is
- $\sigma_{\breve{g}(x,y)} = (1/\mathrm{sqrt}(K))\ \sigma_{\eta(x,y)}$
- Now we realize that as K increases, the variability (noise) of the pixel values at each location (x,y) decreases.

Original Image

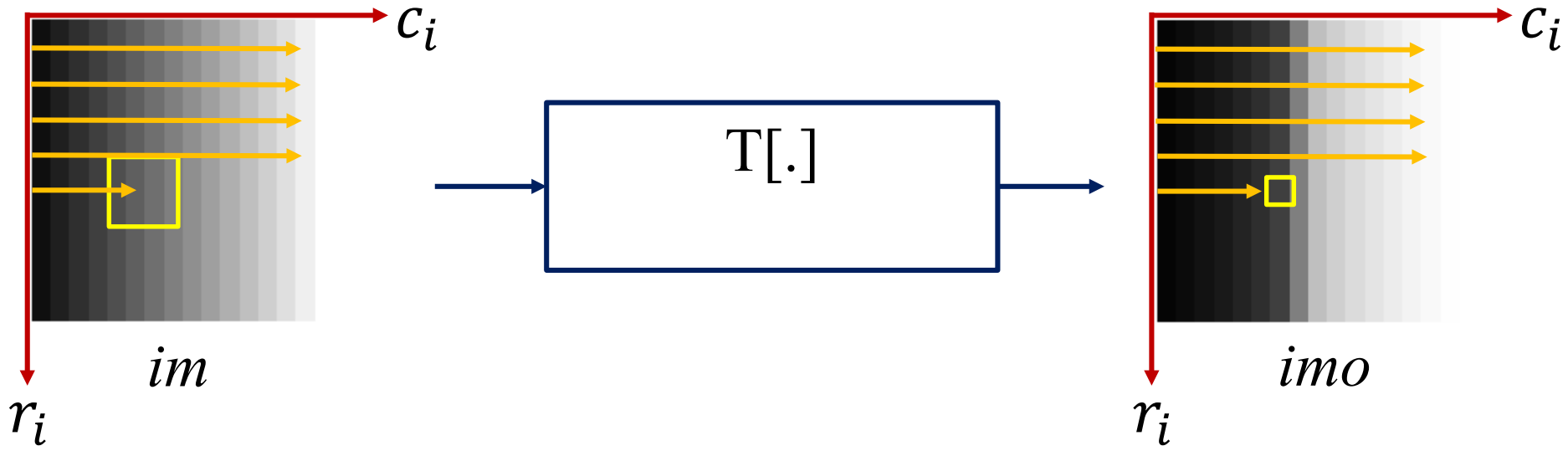
Noisy Image

Averaged Noisy Image

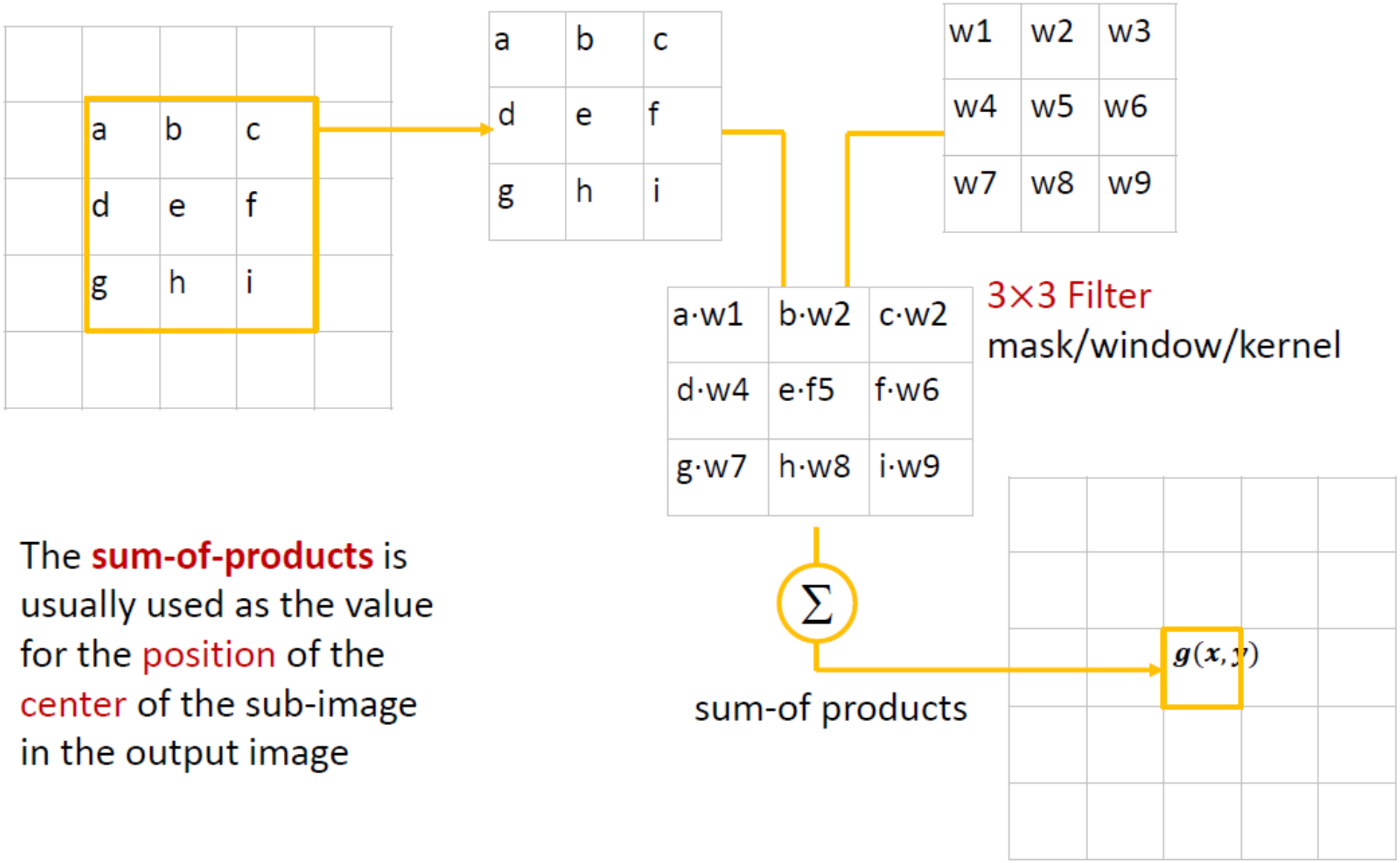
code

Spatial Filtering

- Spatial filtering is a block-to-pixel mapping
  - Operates on a **set of 'neighbourhoods'** *N* of each pixel, and it is normally computed in the **spatial domain** and **not** in **frequency** domain

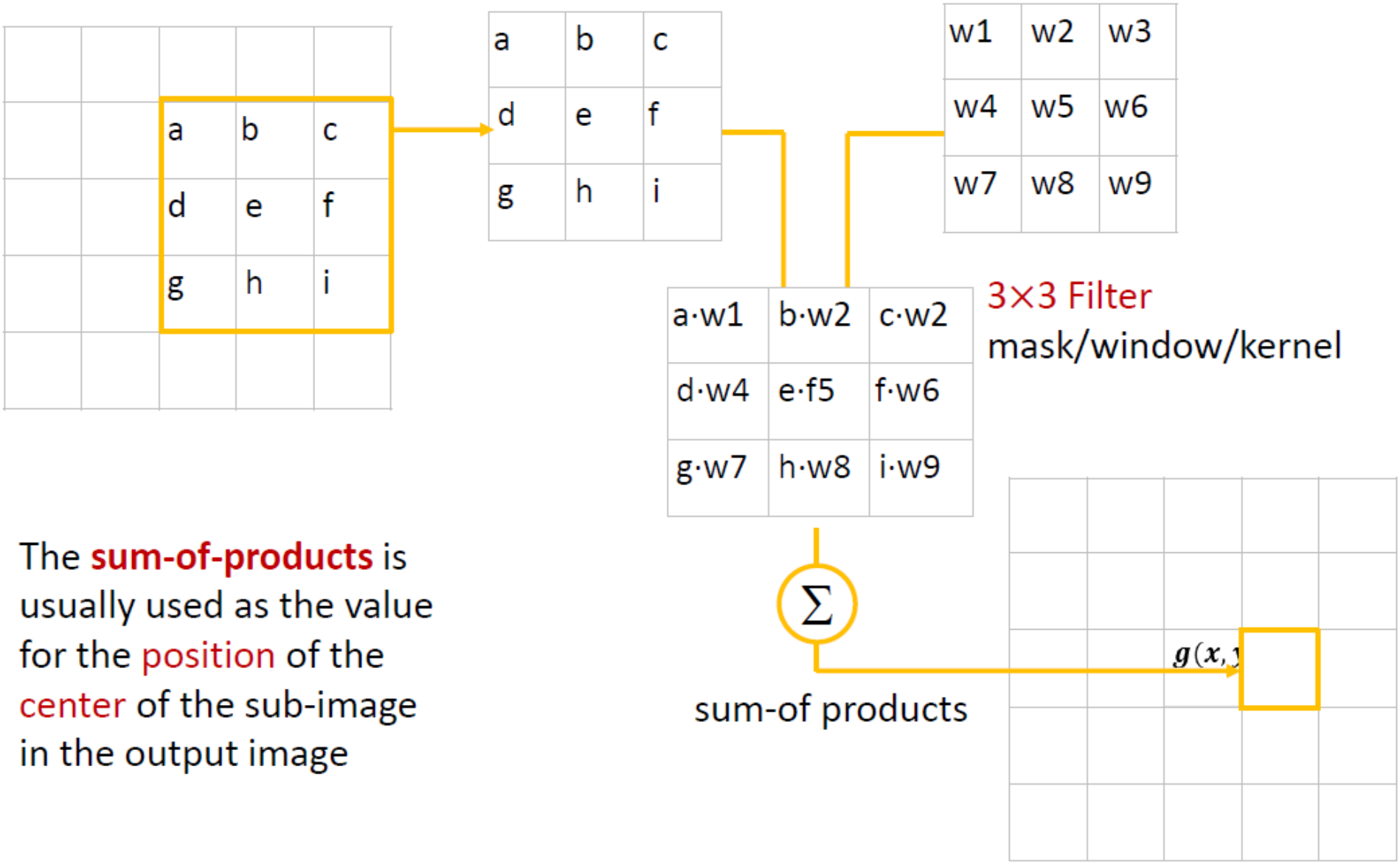


Spatial Filtering



The **sum-of-products** is usually used as the value for the position of the center of the sub-image in the output image

Spatial Filtering



The **sum-of-products** is usually used as the value for the position of the center of the sub-image in the output image

Spatial Filters

- Spatial convolution:
  - Convolution involves 'overlap – multiply – add' with 'convolution mask'

- Mathematically the convolution operator of an image h*(x,y)* with a filter $f(\theta,\vartheta)$ is :

$$g(x,y) = \sum_{\forall\theta}\sum_{\forall\vartheta} f(\theta,\vartheta)h(x-\theta,y-\vartheta)$$

Spatial Filters

- It is more natural to think about the center of the mask with odd sizes, whereas, even masks are awkward for implementation.

- Special consideration is given when the center of the filter approaches the boarder of the image; solutions: cropping, padding.

Matlab: Spatial-Filtering

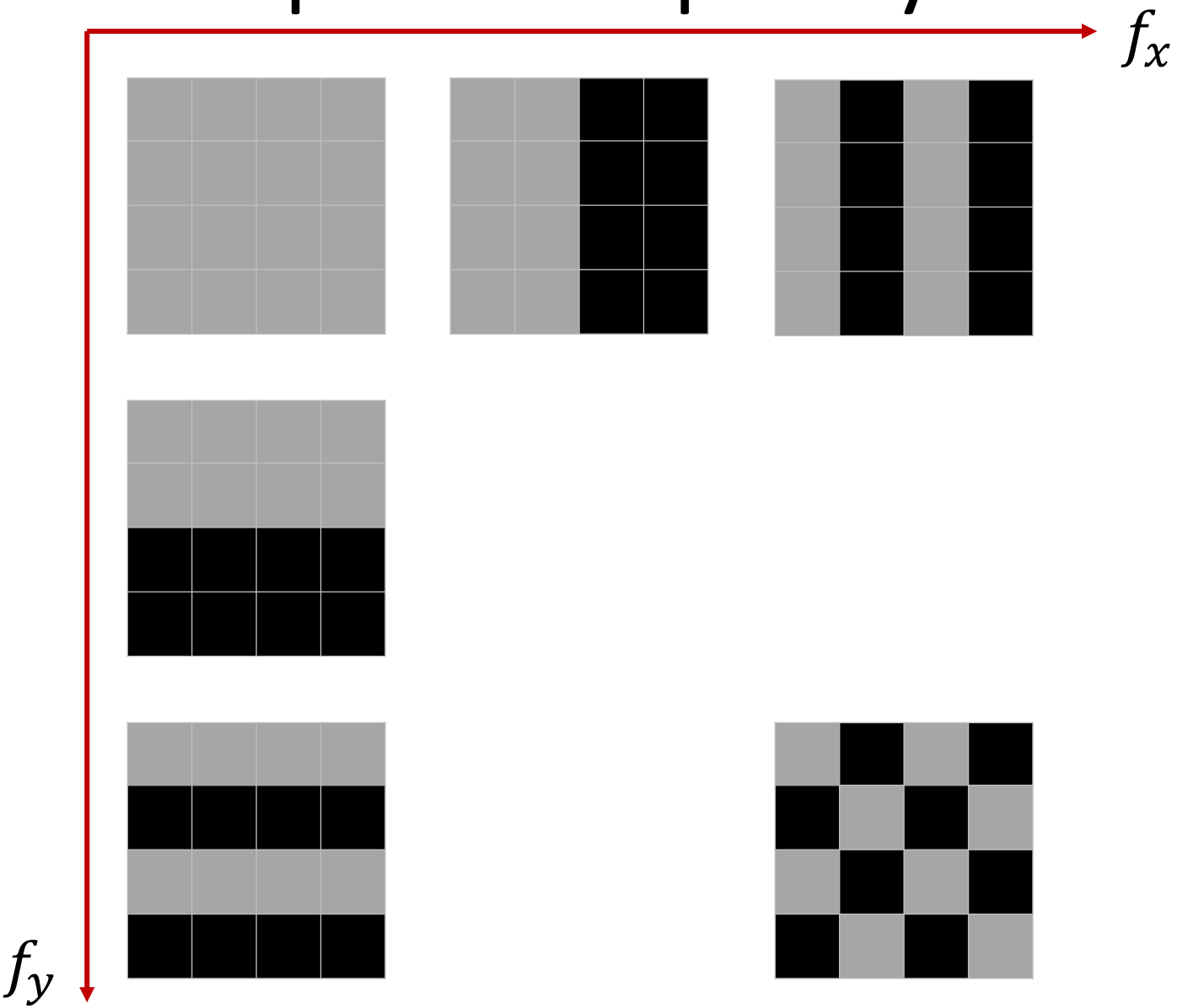
- Example from MATLAB help
  - » im = imread('peppers.png');
  - » h = fspecial('motion', 50, 45);
  - » imo= imfilter(im , h);
  - » figure, imshow(imo)

Spatial frequency

Spatial frequency

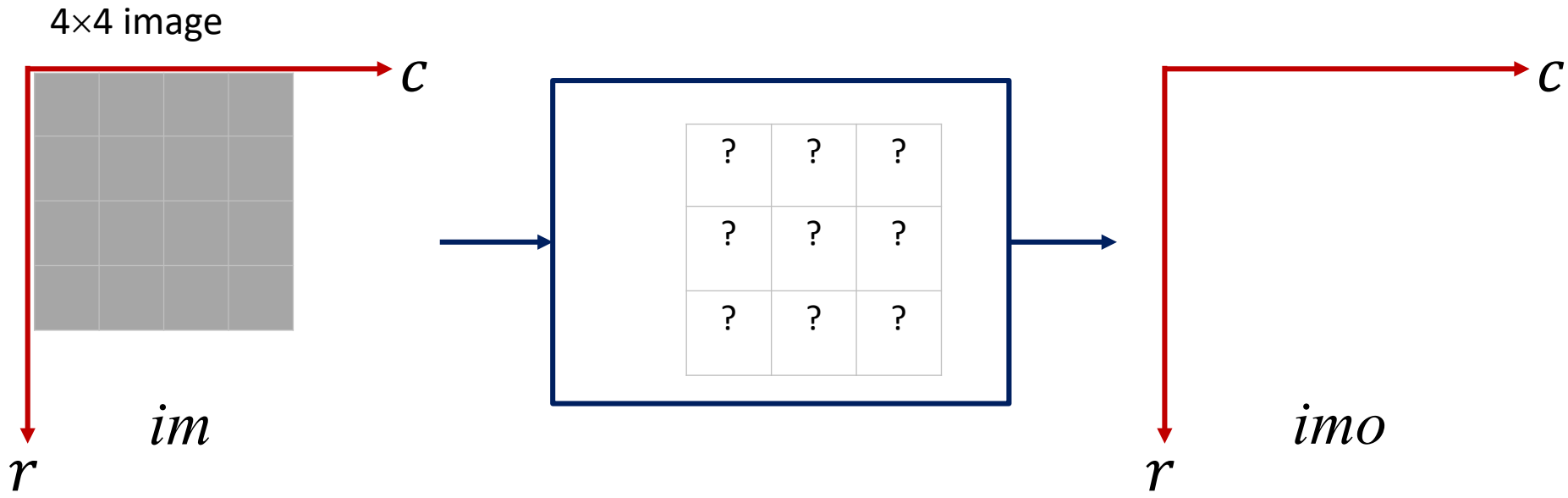
- Spatial Frequency refers to the rate of variations of image brightness with position in space.
  - High spatial frequencies correspond to rapidly varying brightness → fine detail.
  - Low spatial frequencies correspond to slowly changing brightness → flat areas or uniform texture areas.

Spatial frequency

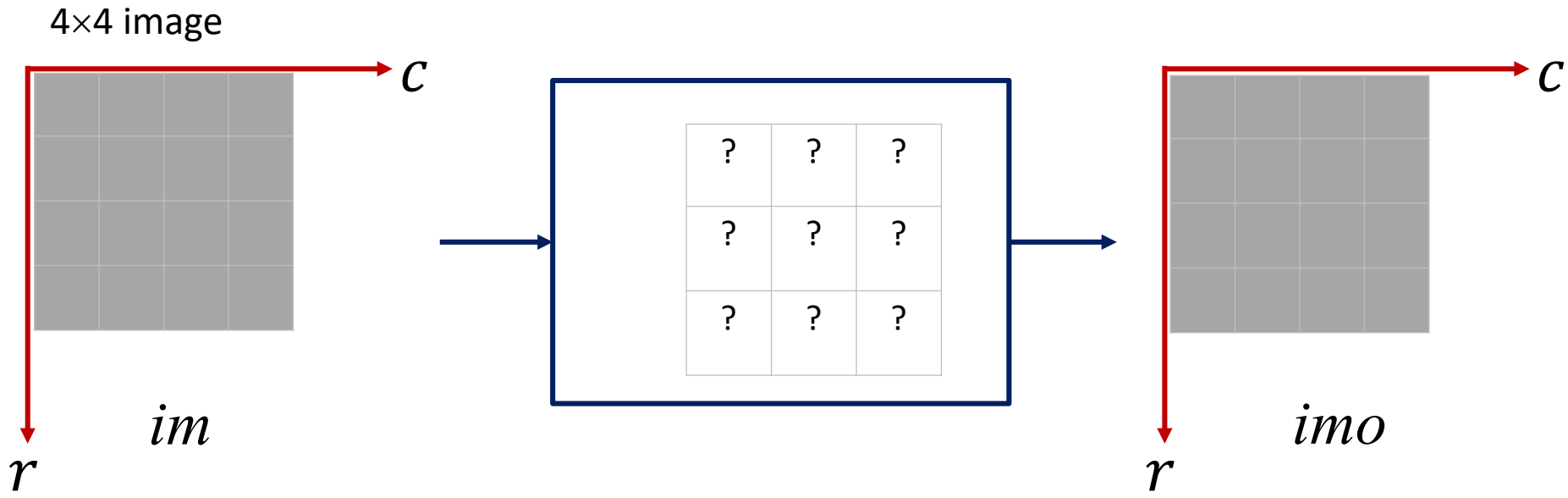


LPF and HPF filters

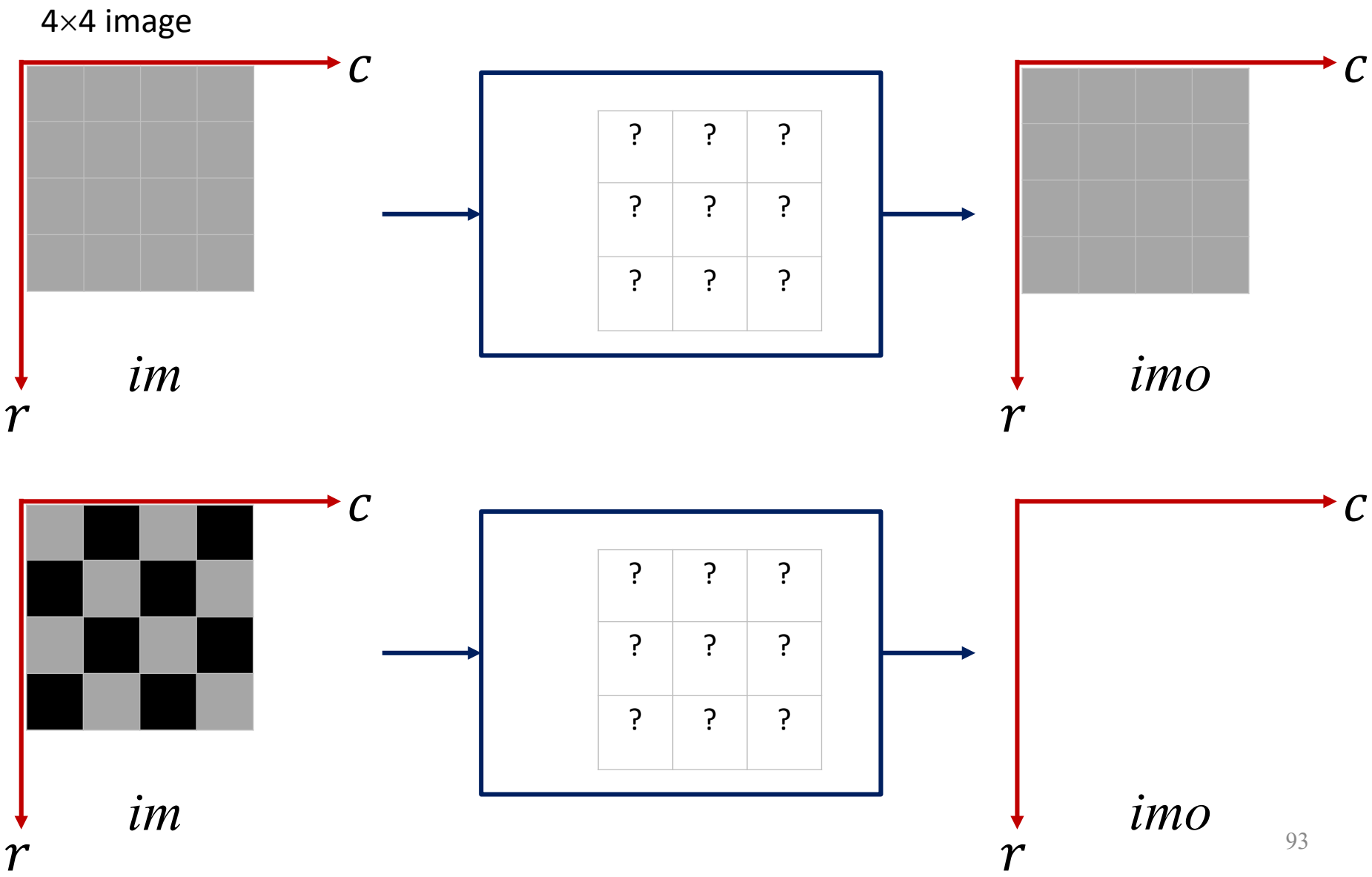
Low pass filters



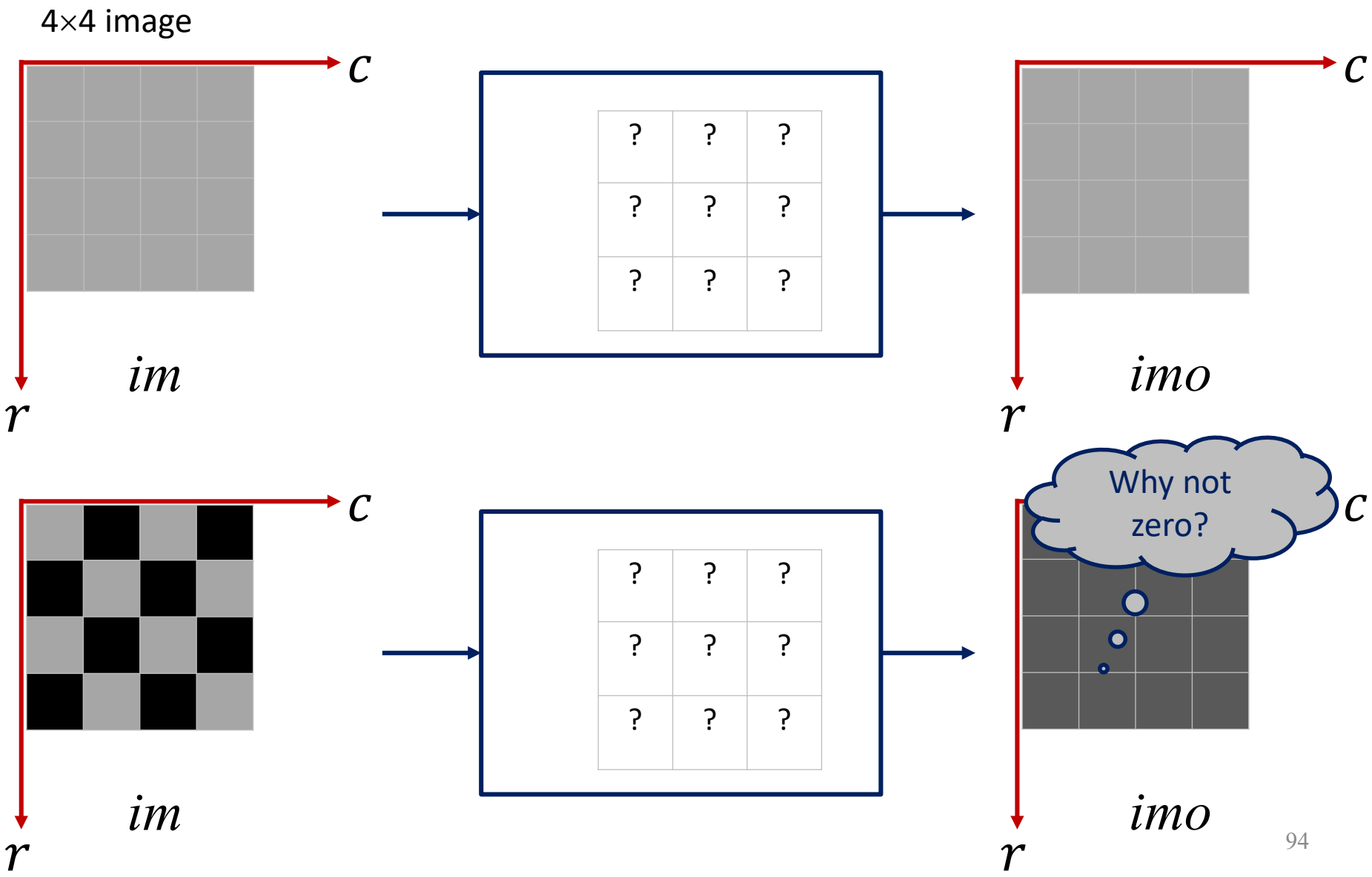
Low pass filters



Low pass filters



Low pass filters

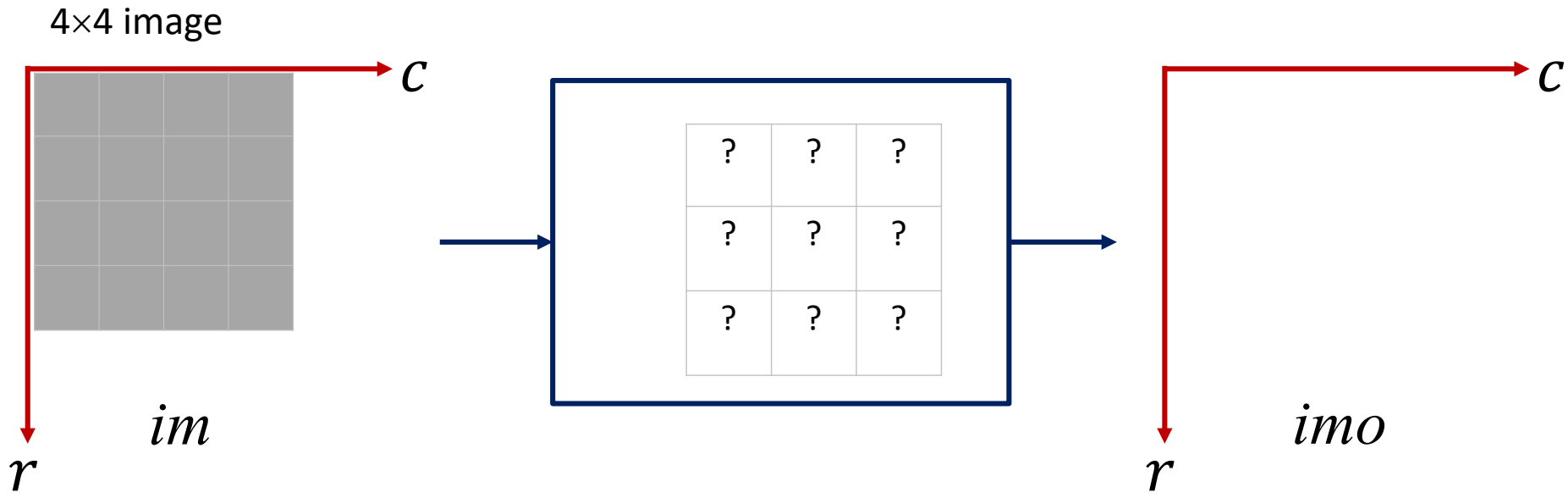


Low pass filters

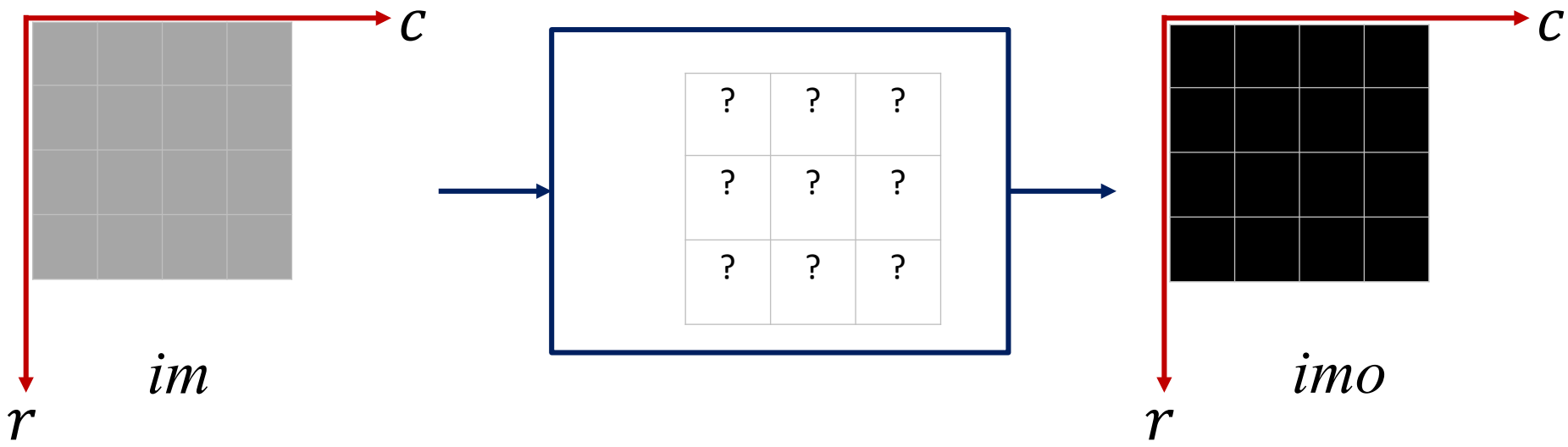
- Low pass filters **attenuate** a **specific** range of high frequency content of the image.
- High frequency content correspond to boundaries of the objects and edges, so LPF filtered images results in:
  - Smooth images.
  - Blurred borders.

- Give an example of a 3x3 LPF filter!

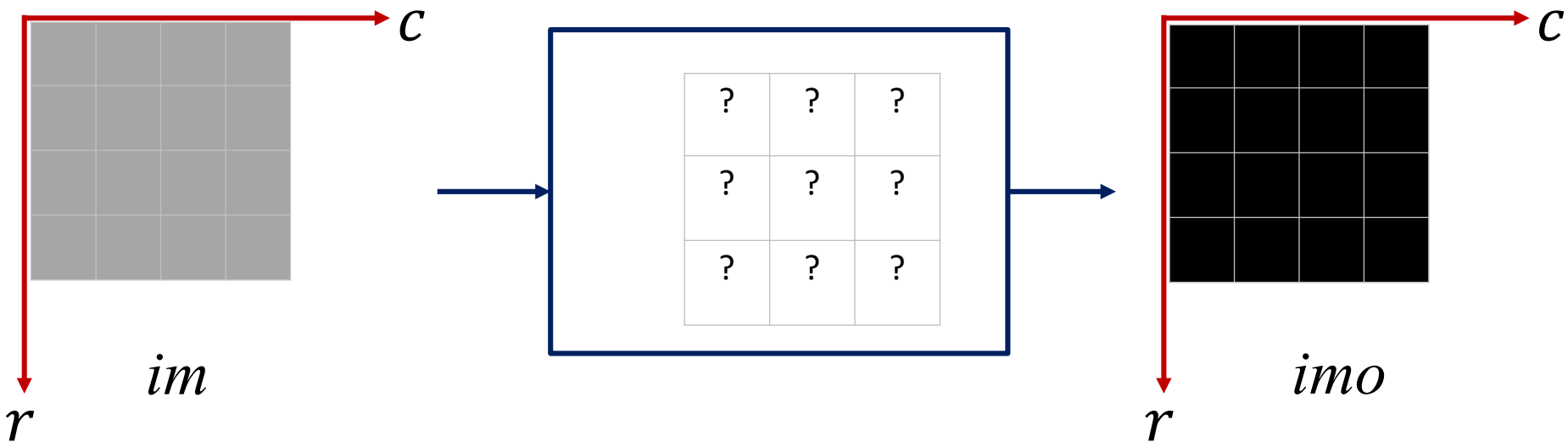
High pass filters

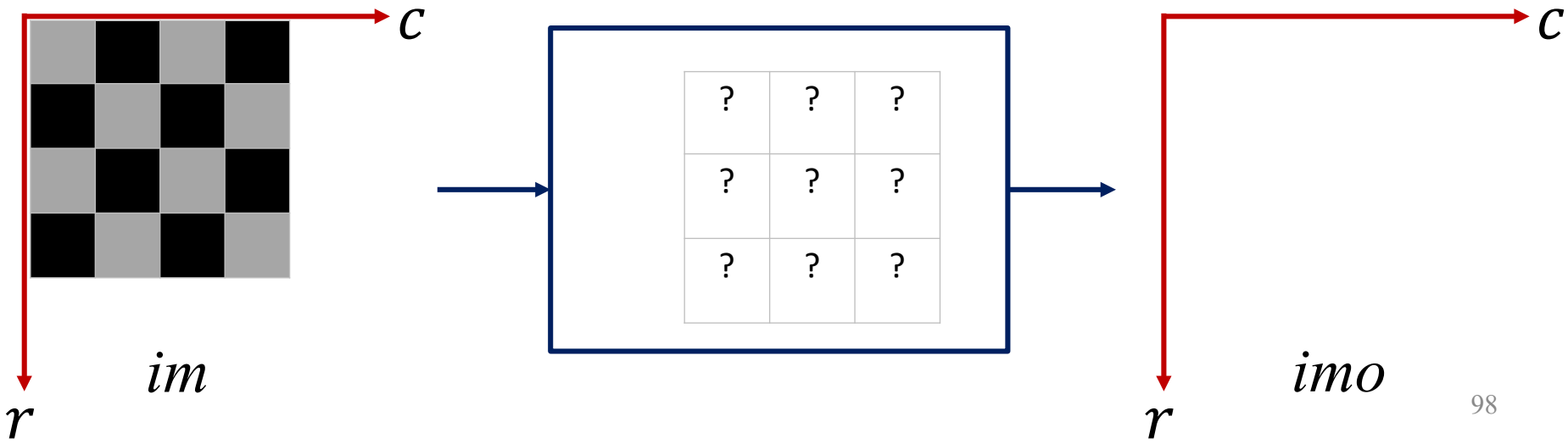


High pass filters

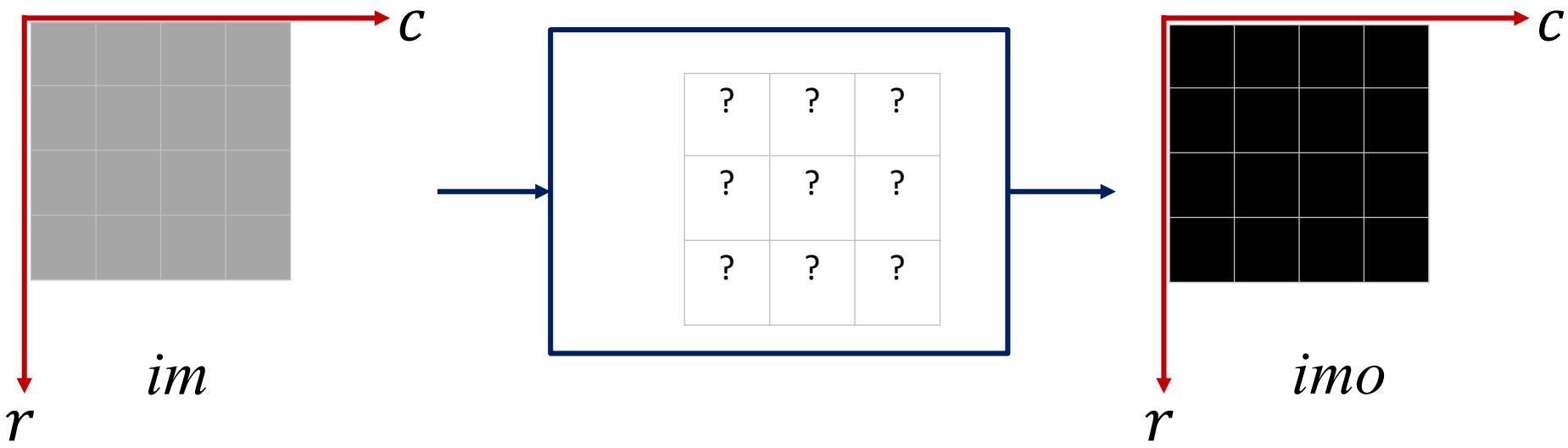


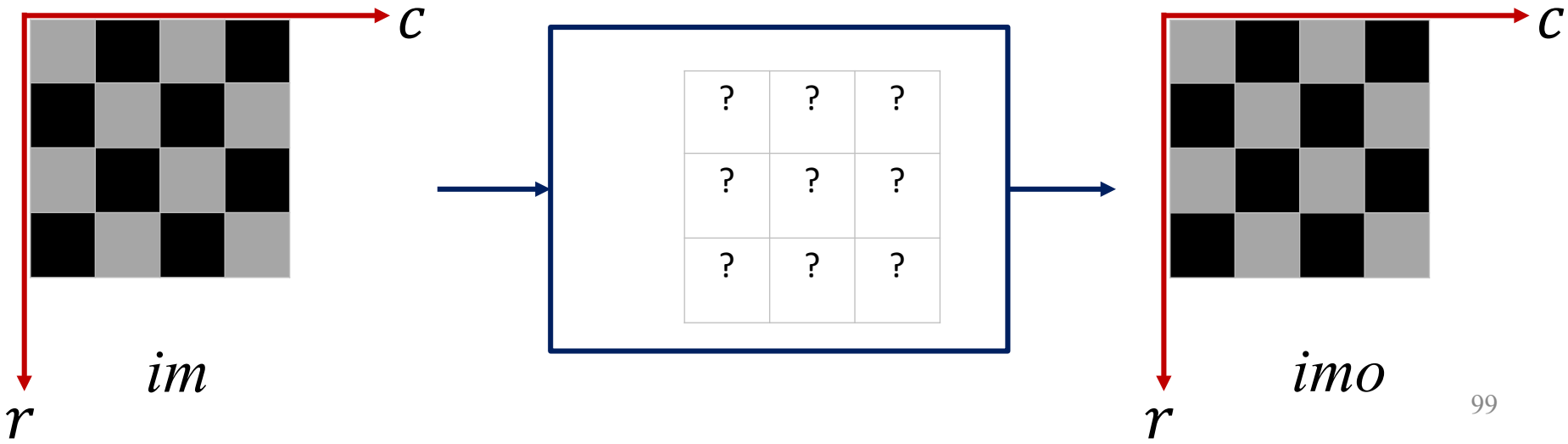
High pass filters





High pass filters





High pass filters

- High pass filters let the high frequency content of the image pass through the filter and block the low frequency content, so:
  - "Texture" information get removed so the results are darker images.
  - Edges get emphasized.

Practical filters: Averaging filter

Why 1/9 ?

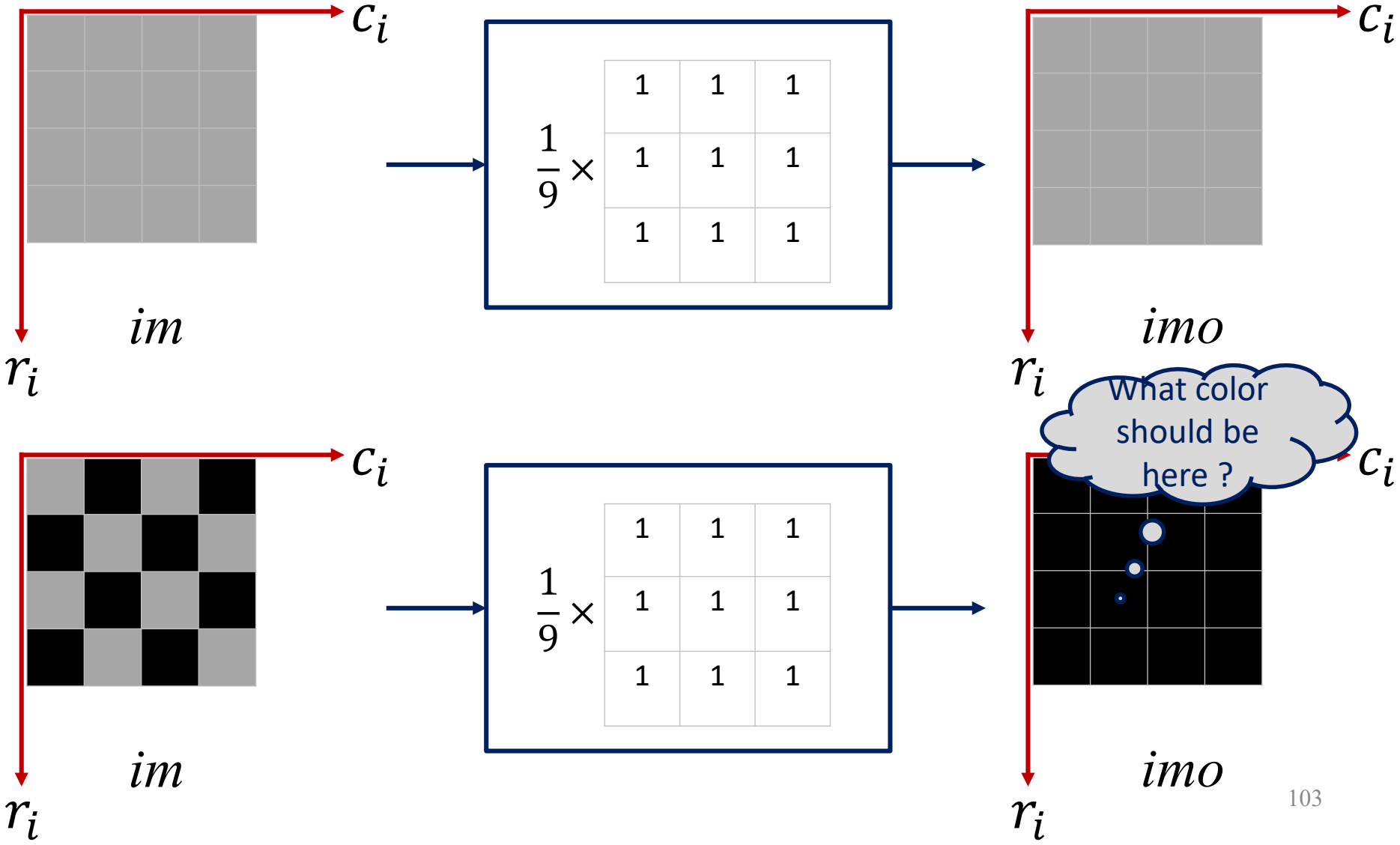
Averaging filter

- The 3×3 averaging filter:

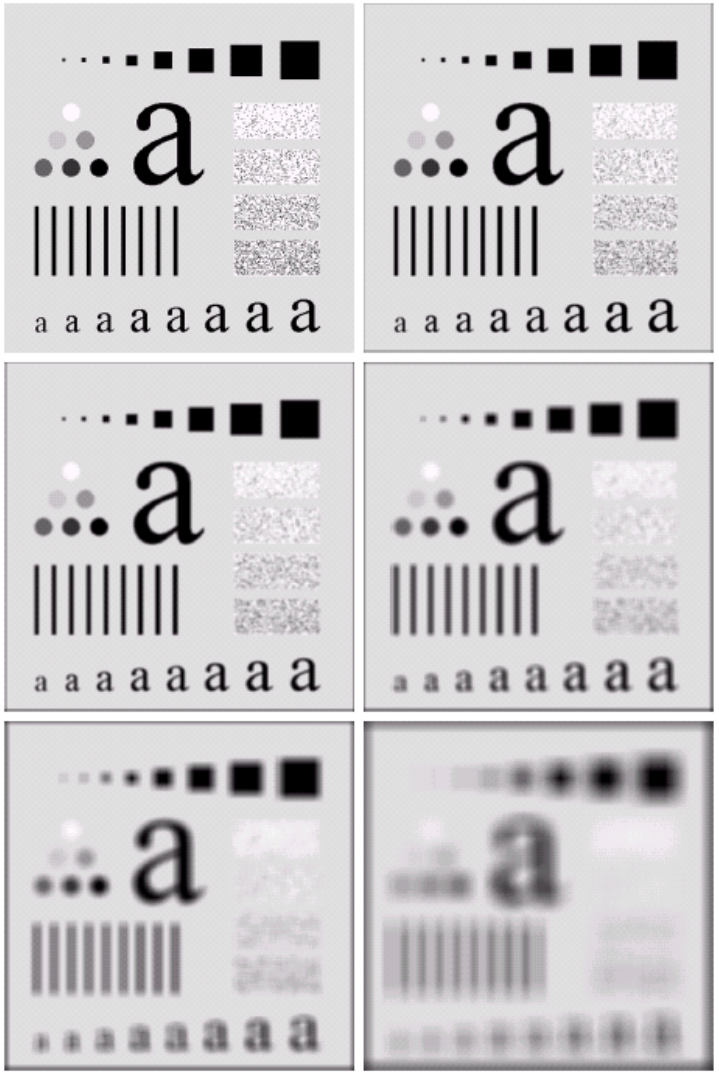
$$\frac{1}{9} \times \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

- This filter is 45° rotation isotropic (isotropic filter response is independent of the direction to which the filter is applied).
- It is a LPF

Averaging filter



Averaging filter



a b
c d
e f

FIGURE 3.35 (a) Original image, of size 500 × 500 pixels. (b)–(f) Results of smoothing with square averaging filter masks of sizes $n = 3, 5, 9, 15$, and 35, respectively. The black squares at the top are of sizes 3, 5, 9, 15, 25, 35, 45, and 55 pixels, respectively; their borders are 25 pixels apart. The letters at the bottom range in size from 10 to 24 points, in increments of 2 points; the large letter at the top is 60 points. The vertical bars are 5 pixels wide and 100 pixels high; their separation is 20 pixels. The diameter of the circles is 25 pixels, and their borders are 15 pixels apart; their gray levels range from 0% to 100% black in increments of 20%. The background of the image is 10% black. The noisy rectangles are of size 50 × 120 pixels.

Averaging filters are a kind of smoothing filters

- The **size of the mask** is related to the relative **size** of the **objects** that will be **blended with the background**.
- **Black border** is due to **padding** (expanding) the border of the original image with **zeroes**.

Smoothing filters

- Smoothing filters are used for :
  - Noise reduction
  - **Reduction** of **irrelevant detail** or false contours
  - Sometime for image blurring

- Undesirable side effect of smoothing filters
  - Blur edges (if blurring is not the main target)

Weighted average filters

- An example of 3×3 weighted average filter:

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

Which distance? De, D_4, D_8?

- The weights aim to give more importance to some pixels at the expanse of others.
  - In this example the importance reduces with the distance from the center of the filter.

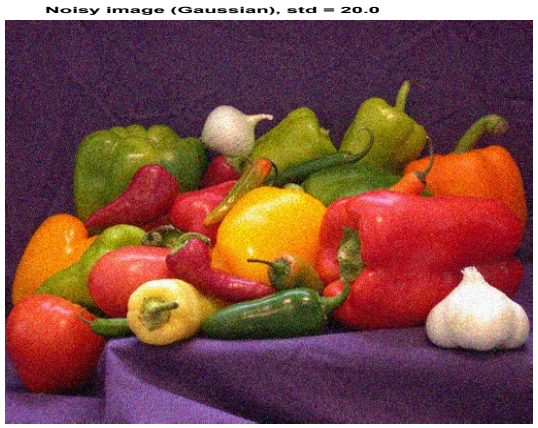
Weighted average filters

- **Due** to the **uneven weights** the weighted average filter **reduces blurring** in the smoothing process.

- Normalization
  - To **avoid changing** the overall **brightness** due to the use of LPF the entries of the filter must sum to one. So in the previous example 1/16 is a normalization factor.

Matlab: Spatial-Filtering

- Example from MATLAB

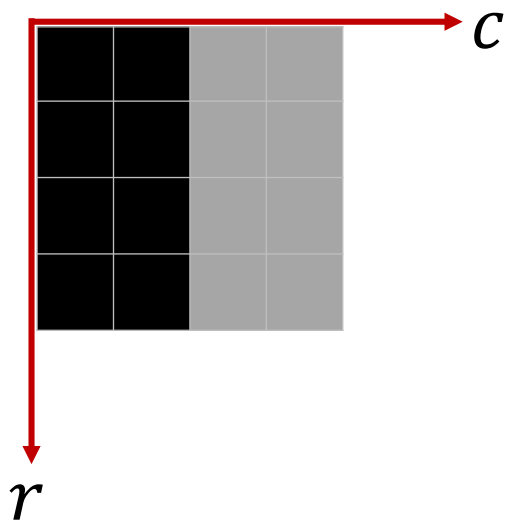


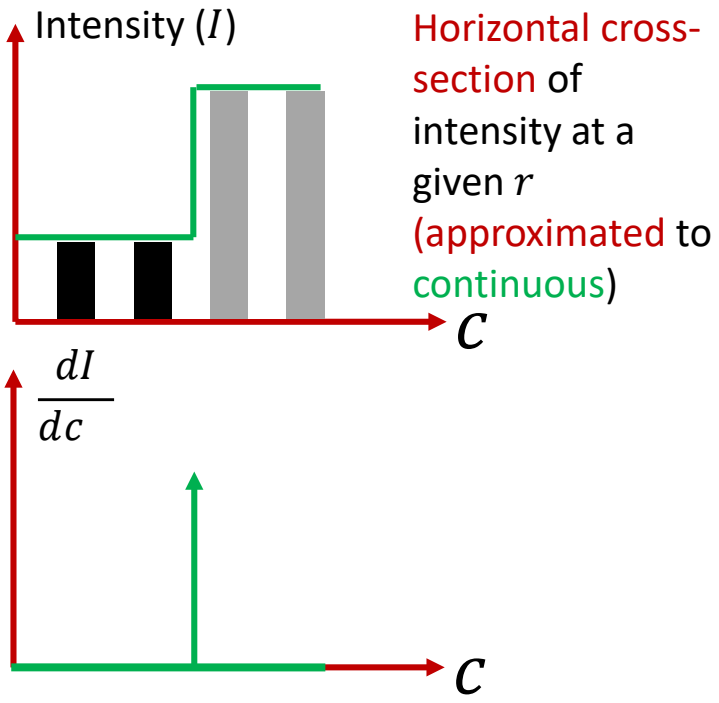
code

Practical filters: edge detection filters

Edge Detection Filters

- How to detect edges ?





Horizontal cross-section of intensity at a given r (approximated to continuous)

- First derivative (gradient: $\frac{\partial f}{\partial x}$) allows to detect changes of continuous functions

Edge Detection Filters

- In the discrete domain the derivative could be approximated as :

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

- These are fast to compute, but the result can only be an approximation to the true gradient, and is usually an underestimate.

Edge Detection Filters

- Sobel operators :

Detect horizontal edges

-1	-2	-1
0	0	0
1	2	1

Detect vertical edges

-1	0	1
-2	0	2
-1	0	1

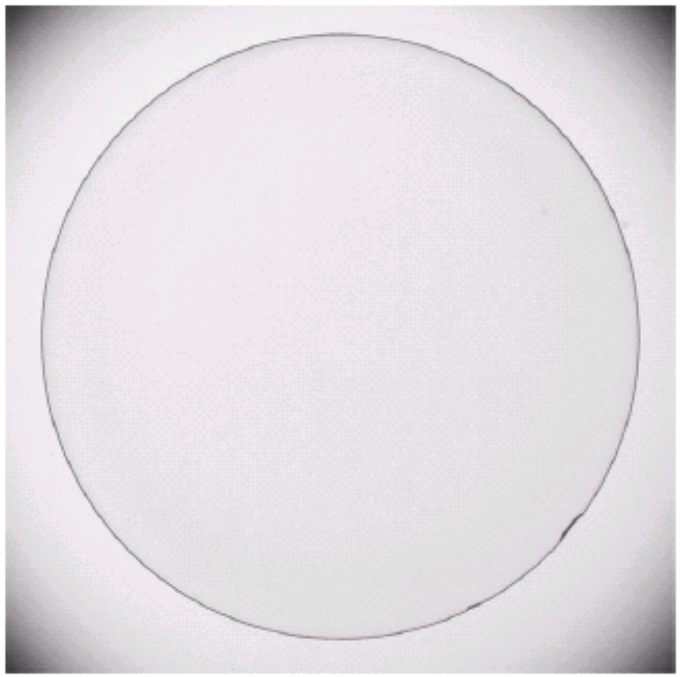
- The center point has more importance.
- Filter's coefficients sum to zero, indicating that a constant gray area would have zero response. Is it LPF or HPF filter ?

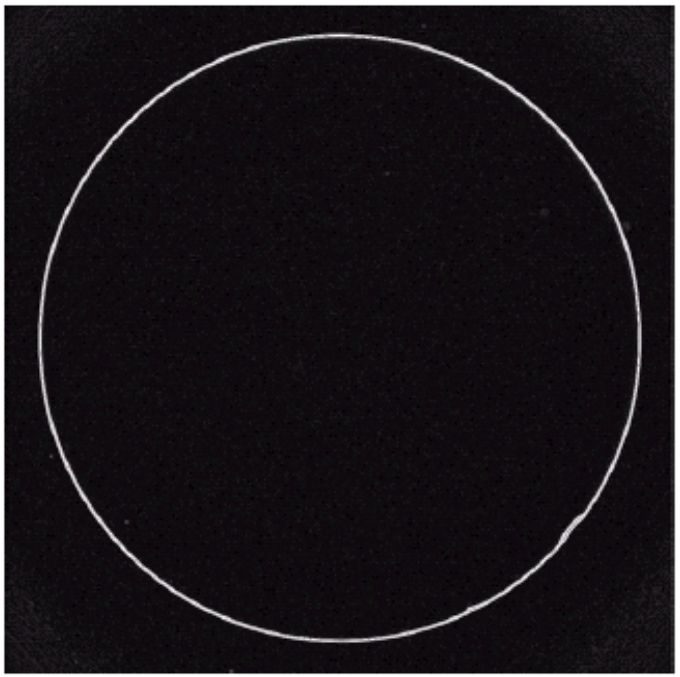
Edge Detection Filters

- Step 1: Convolution with gradient (Sobel) filter
  - Separately for horizontal and vertical directions
- Step 2: Magnitude of gradient
  - Norm of horizontal and vertical gradients
- Step 3: Thresholding
  - Threshold to detect edges

Edge Detection Filters

- Sobel operators :





a b

FIGURE 3.45 Optical image of contact lens (note defects on the boundary at 4 and 5 o'clock). (b) Sobel gradient. (Original image courtesy of Mr. Pete Sites, Perceptics Corporation.)

The horizontal and vertical edges are summed (ignoring sign) to give the result

Matlab: Edge Detection Filters

- Edge Detection in MATLAB

"All" edges

Horizontal edges

Vertical edges

Vertical edges

code

Practical filters: sharpening filters

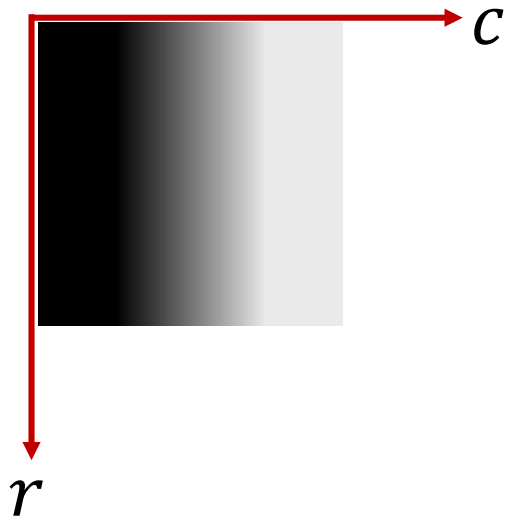
Sharpening Filters

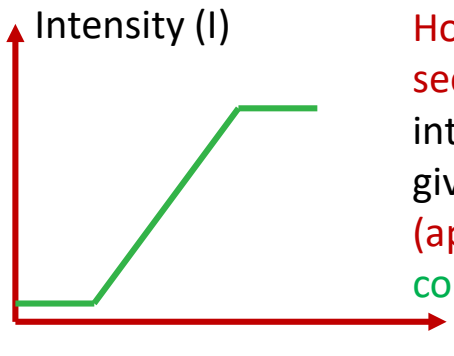
- The principal objective of sharpening is to :
  - highlight fine detail in an image and/or to
  - enhance blurred details

- Unlike blur when sharpening an image we want to accentuate high frequencies

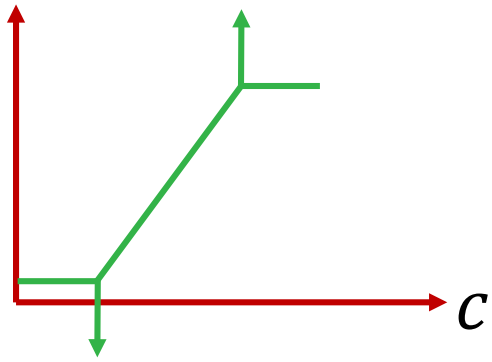
- How to sharpen an image ?

Sharpening Filters

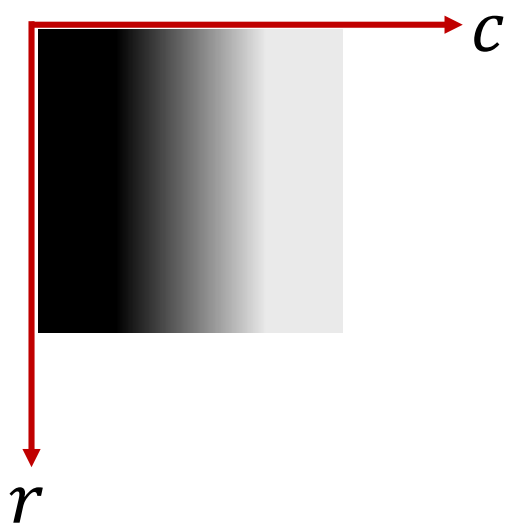


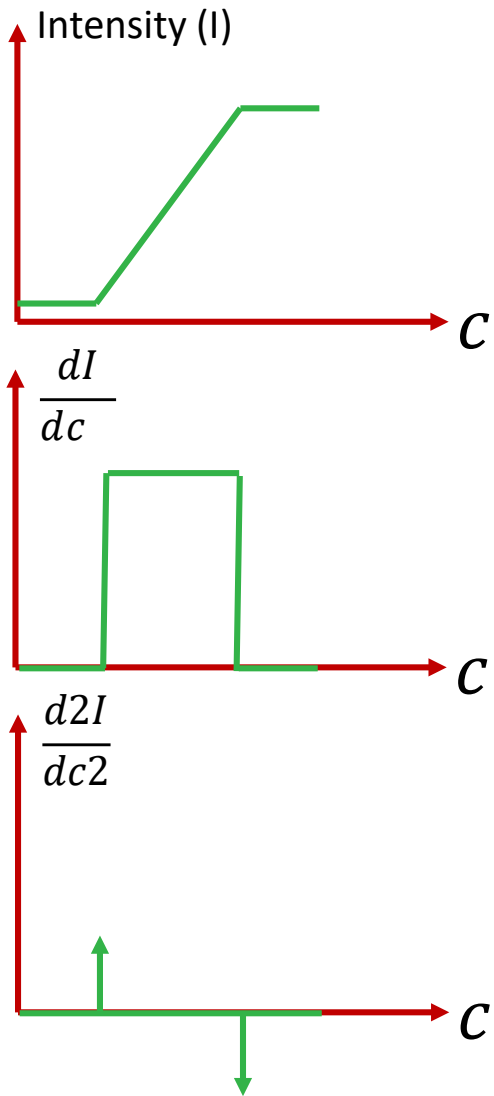


Horizontal cross-section of intensity at a given r (approximated to continuous)



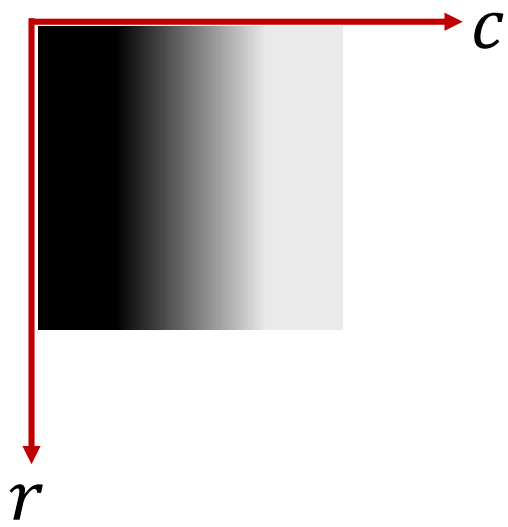
Sharpening Filters

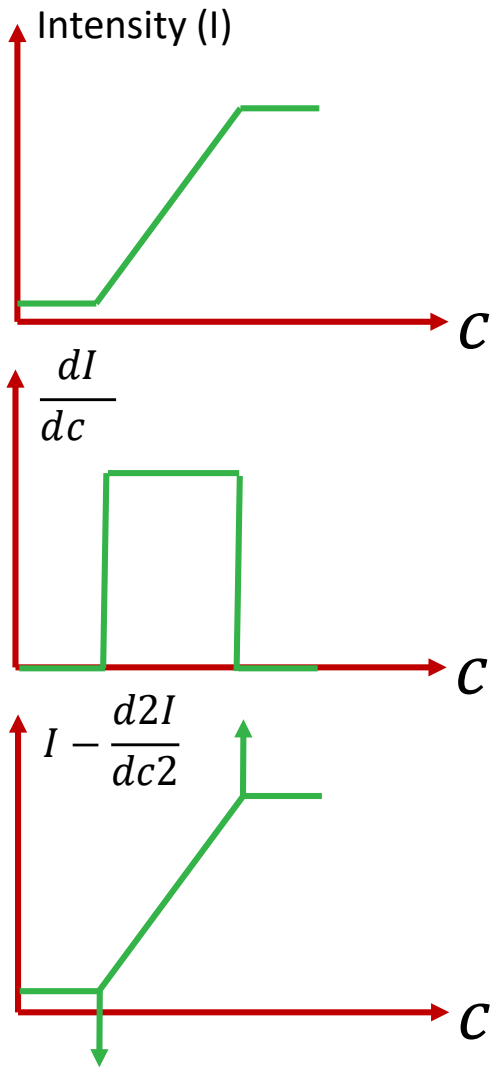




Can the second derivative enhance sharpness ?

Sharpening Filters





The second derivative allows to enhance sharpness

Sharpening Filters

- First derivative

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

- Second derivative

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

Sharpening Filter

Second derivative in discrete-domain:

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

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

Two dimensional Laplacian:

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

Sharpening Filters

- 90° rotation isotropic second derivative filter

0	1	0
1	-4	1
0	1	0

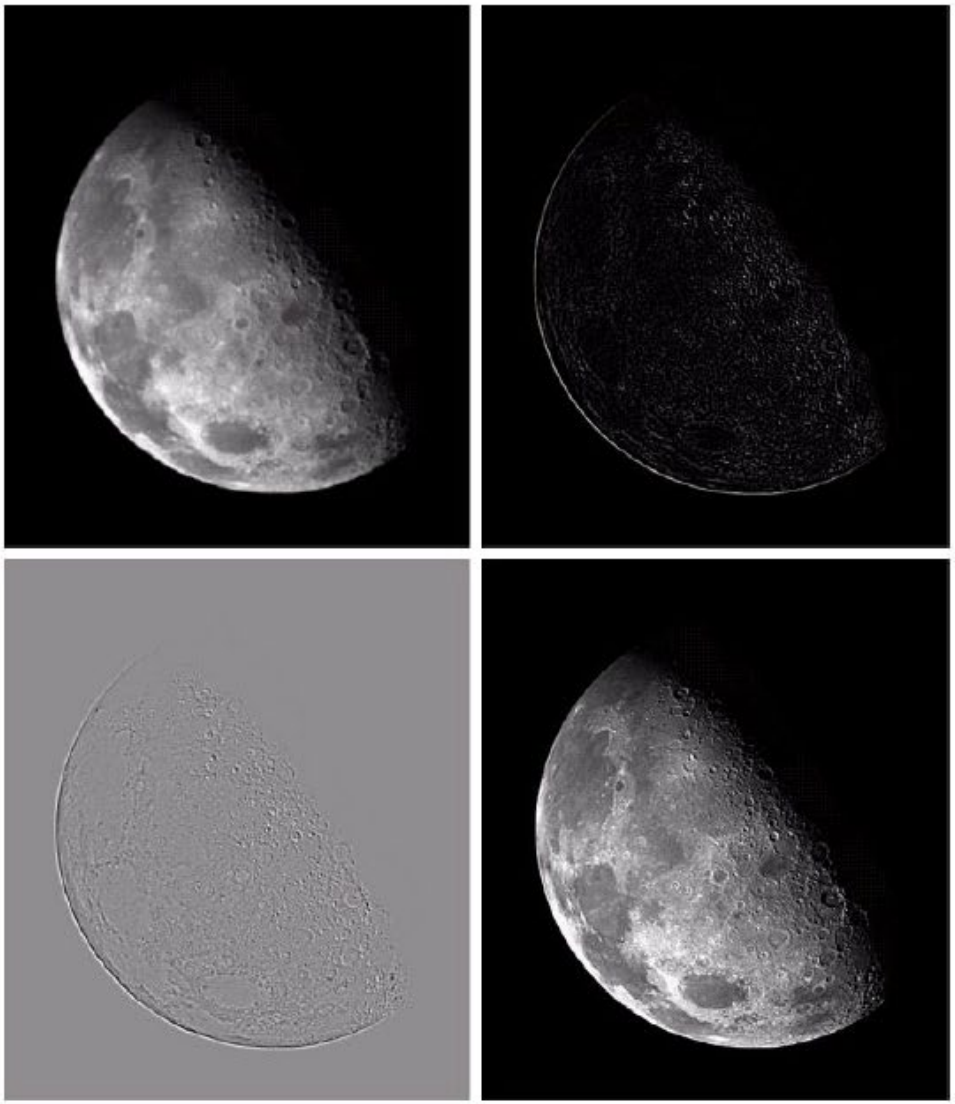
- 45° rotation isotropic second derivative filter

1	1	1
1	-8	1
1	1	1

Sharpening Filters

a b
c d

FIGURE 3.40
(a) Image of the North Pole of the moon.
(b) Laplacian-filtered image.
(c) Laplacian image scaled for display purposes.
(d) Image enhanced by using Eq. (3.7-5).
(Original image courtesy of NASA.)



- **Laplacian sharpening**

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

- **Simplification:**

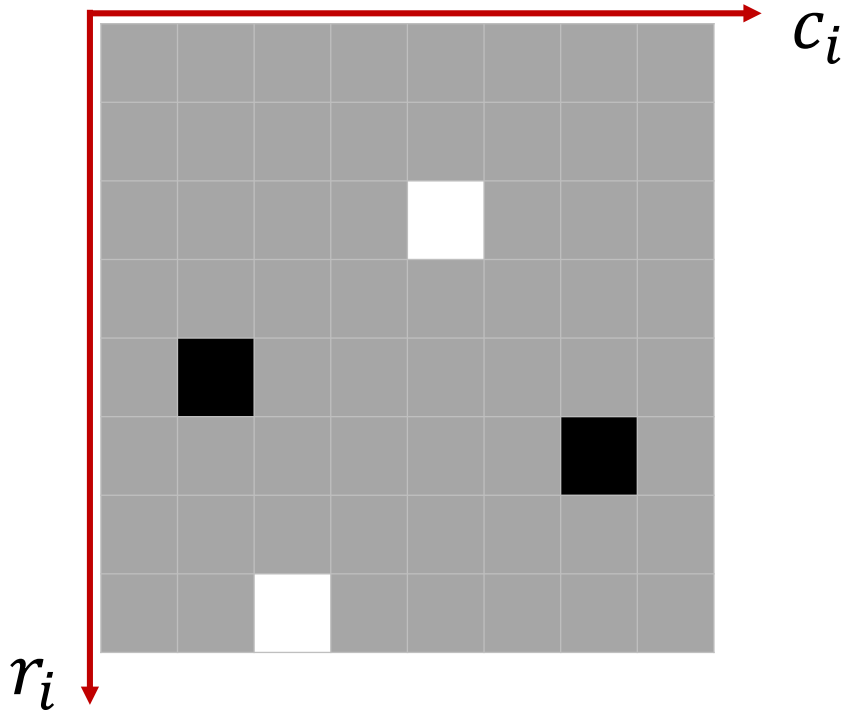
0	-1	0
-1	5	-1
0	-1	0

Sharpening Filters

```
originalRGB = imread('lenna512.bmp');
h = fspecial('average', [3 3]);
filteredRGB = imfilter(originalRGB, h, 'replicate');
figure, imshow(originalRGB) ;
figure, imshow(filteredRGB) ;
w = [ 0 1 0
    1 -4 1
    0  1 0
   ];
lap_image = imfilter(filteredRGB, w, 'replicate');
final_image = originalRGB - lap_image;
figure, imshow(final_image) ;
```

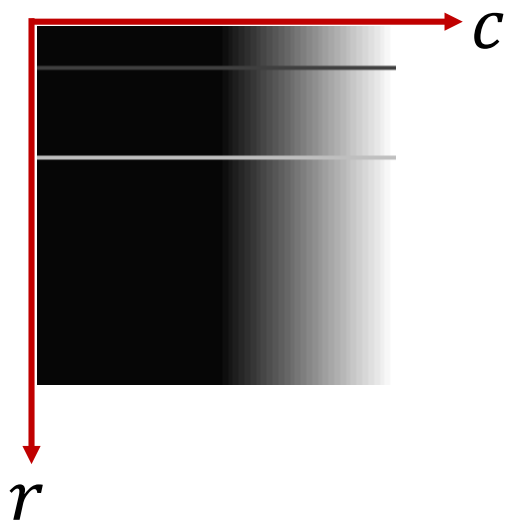
Problem: Salt & Paper noise

- How to remove S&P noise ?
- Is averaging (LPF) filter good for this kind of noise ?



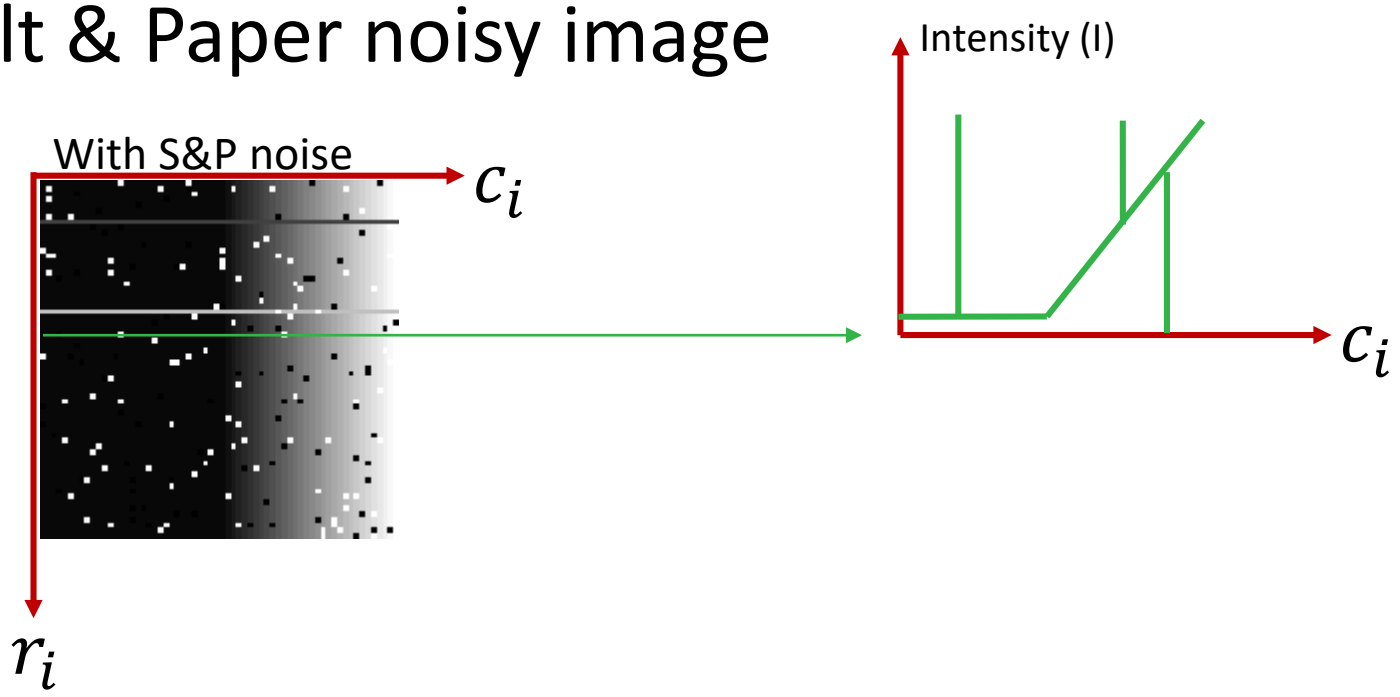
Case study: Salt & Paper noise

- Original image



Case study: Salt & Paper noise

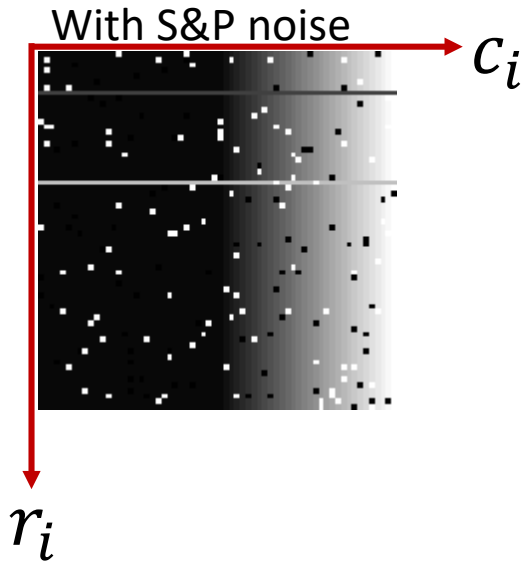
- Salt & Paper noisy image

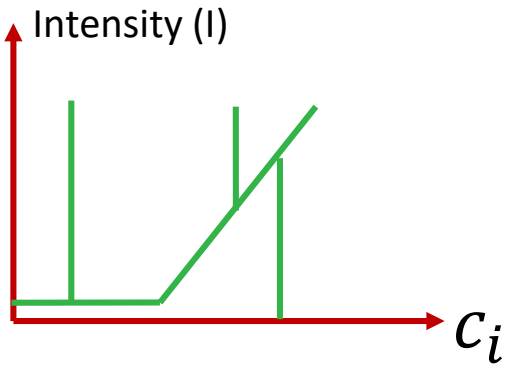


- Sometime we call it impulse/spike noise

Case study: Salt & Paper noise

- Salt & Paper noisy image





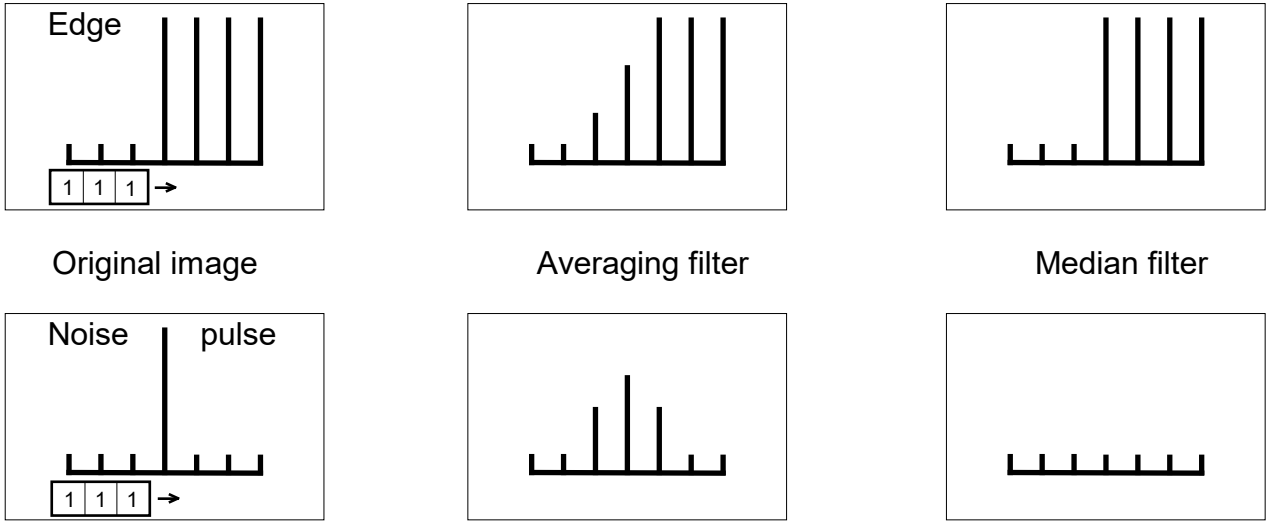
- How to remove S&P noise ?
- Is averaging (LPF) filter good to handle this kind of noise ?

Case study: Salt & Paper noise

- Observations:
  - The intensity of natural images is normally smooth, without sudden and sharp changes → locally data is "in order".
  - S&P noise (sometime related to "defective pixel") appear as sudden and sharp change of intensity, as if it "out-of-order".
  - To remove S&P noise we could use order-statistics filters.

Low Pass and Median Filters

- The low-pass filter can provide image smoothing and noise reduction, but subdues and blurs sharp edges.
- Median filters can provide noise filtering without blurring.

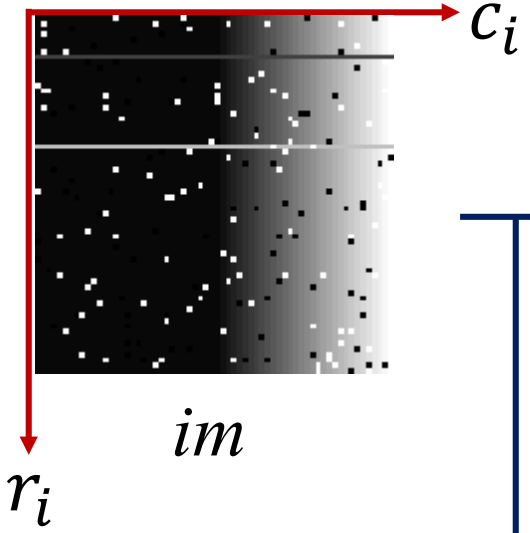


Order Statistic Filters

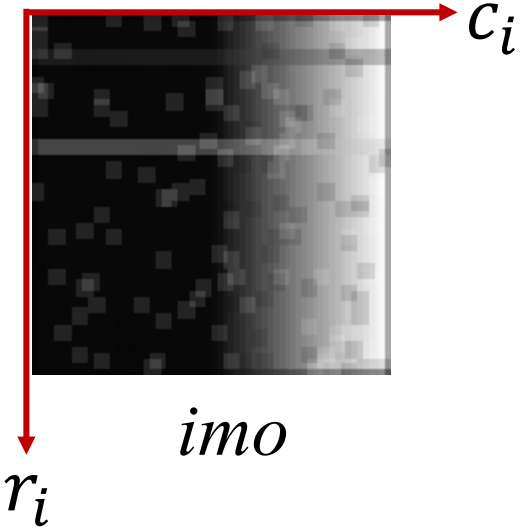
- Are **nonlinear** spatial filters.
- The median filter replaces the pixel at the center of the filter with the median value of the pixels falling beneath the mask, to do this it **orders** (ranks) the pixels of the sub-image and then replaces the value of the center pixel with the value determined by the ranking result.
- The median of a set of *k* pixel grey-levels is that grey level for which *k/2* are smaller or equal, and *k/2* are larger or equal.
- Note that the choice of neighborhood over which the median is computed has a significant influence over the filter's effect on edges, corners and other image discontinuities.

Order Statistic Filters

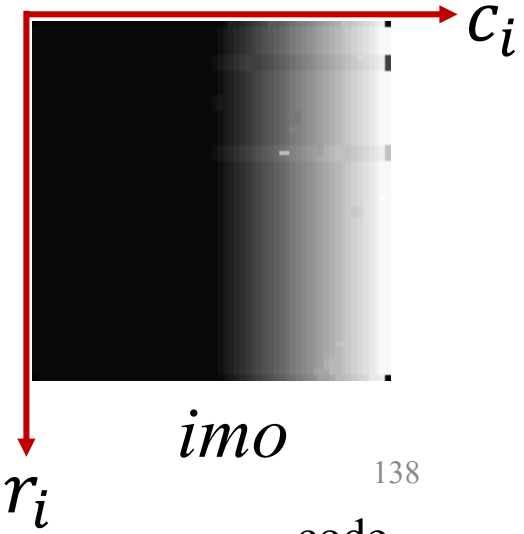
With 5% S&P noise



$$\frac{1}{9}\times\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}$$



3×3 median filter



code

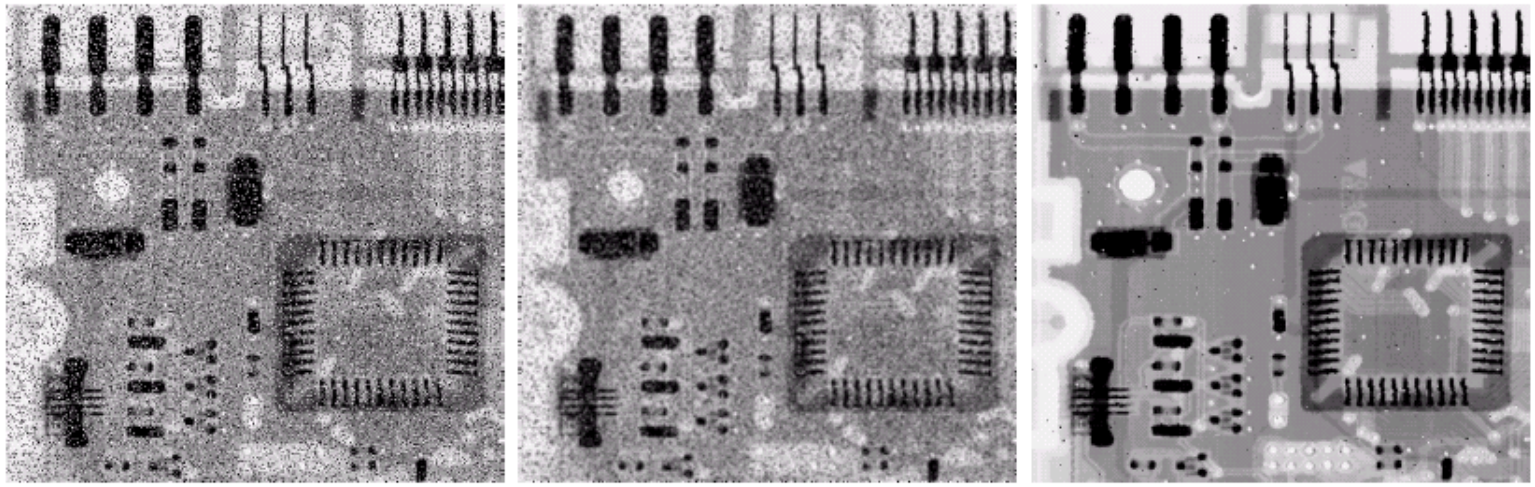
Order Statistic Filters

- Median filter does not blur the image but it rounds the corners.
- A median filter is more effective than convolution when the goal is to simultaneously reduce noise and preserve edges.

Order Statistic Filters

3×3 average filter

3×3 median filter



a b c

FIGURE 3.37 (a) X-ray image of circuit board corrupted by salt-and-pepper noise. (b) Noise reduction with a 3 × 3 averaging mask. (c) Noise reduction with a 3 × 3 median filter. (Original image courtesy of Mr. Joseph E. Pascente, Lixi, Inc.)

Order Statistic Filters

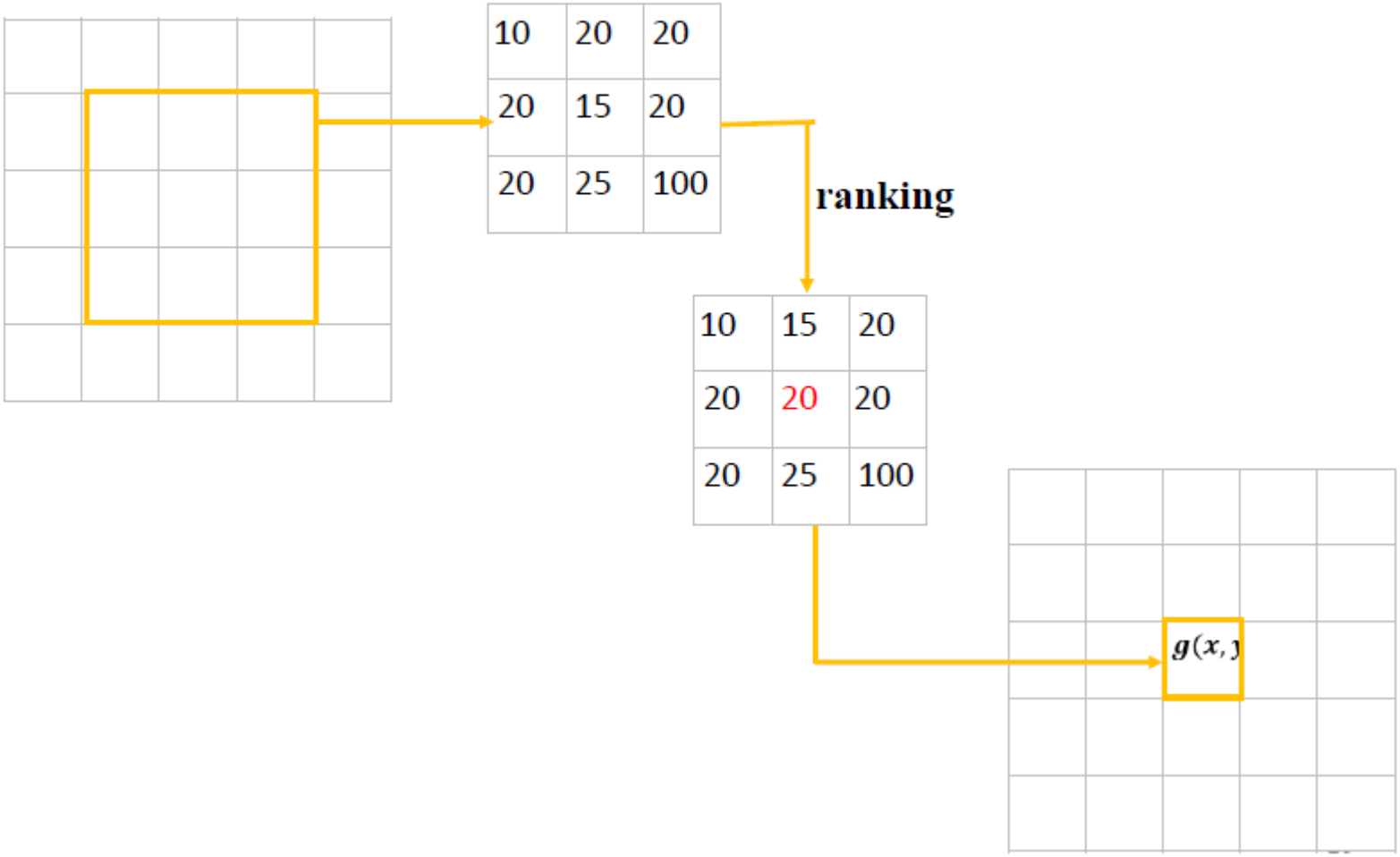
- Median filter eliminates isolated clusters of pixels that are light or dark with respect to their neighbors, and whose area is less than n^2/2.

3×3 Median filter [10 125 125 135 141 141 144 230 240] = 141

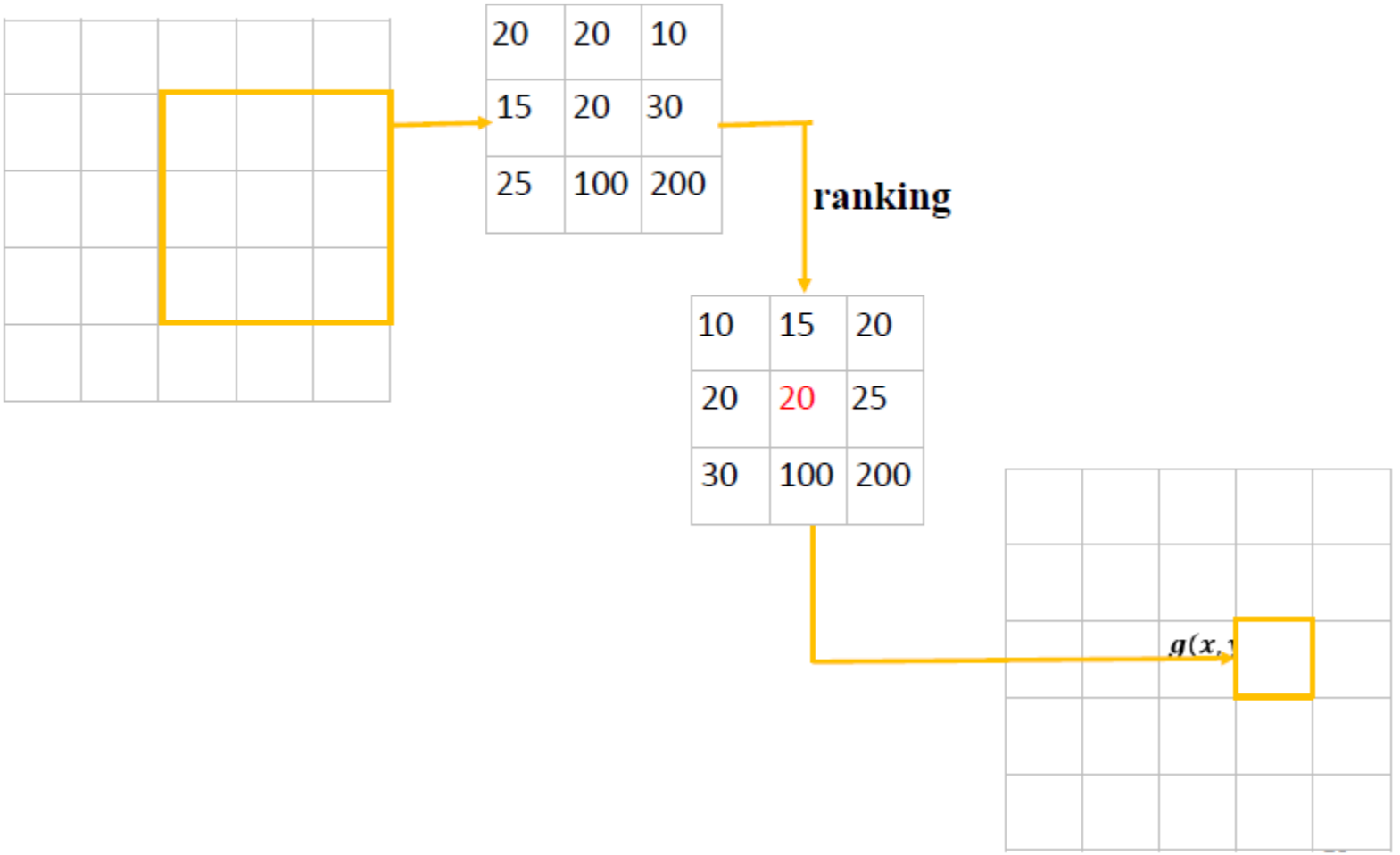
3×3 Max filter [10 125 125 135 141 141 144 230 240] = 240

3×3 Min filter [10 125 125 135 141 141 144 230 240] = 10

Order-Statistics filter
(median filter)



Order-Statistics filter
(median filter)



Order-Statistics filter (median filter)

original

salt & pepper

filtered image

code

In Summary

- Fixed parameters in the spatial filters
- Fixed rules in the order-statistics filters

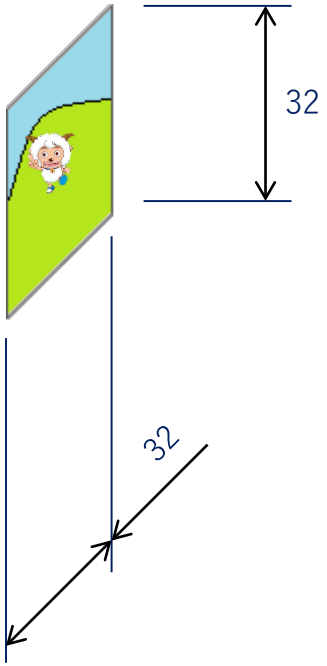
- How to make it flexible? Adaptable?
  - Learning these parameters
  - CNN

Self-Reading

Convolutional Neural Networks (CNN)

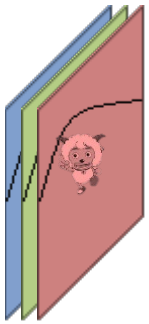
Convolutional Neural Networks

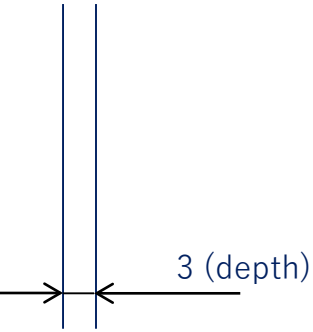
- Convolution layer



Convolutional Neural Networks

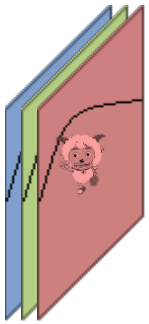
- Convolution layer





Convolutional Neural Networks

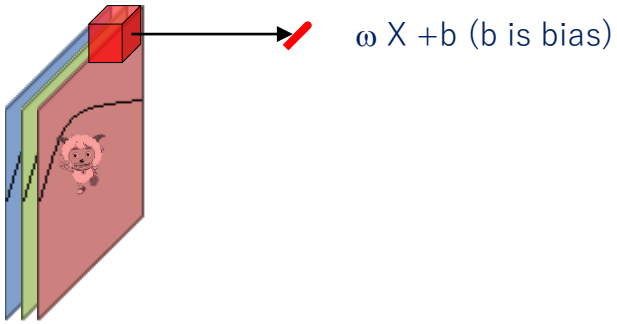
- Convolve the filter with the image



5x5x3 filter (ω)

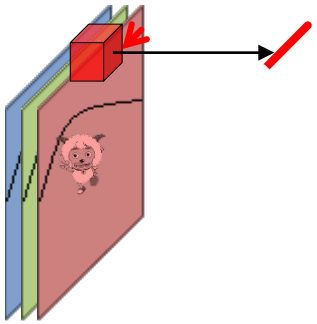
Convolutional Neural Networks

- Convolve the filter with the image



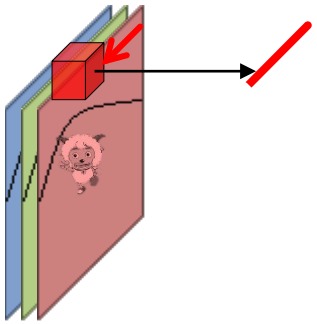
Convolutional Neural Networks

- Convolve the filter with the image



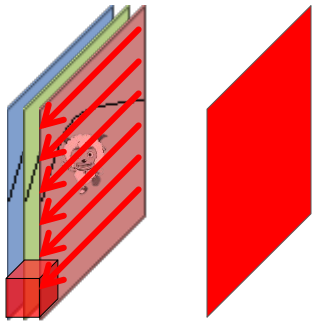
Convolutional Neural Networks

- Convolve the filter with the image



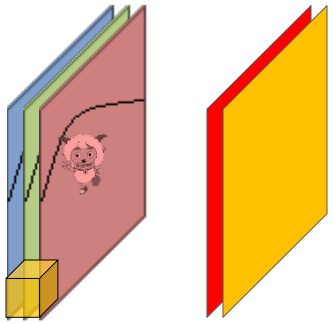
Convolutional Neural Networks

- Generate the first feature map in layer 1



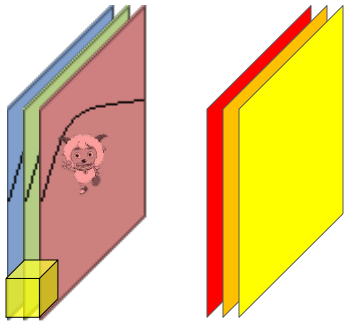
Convolutional Neural Networks

- Generate the second feature map in layer 1



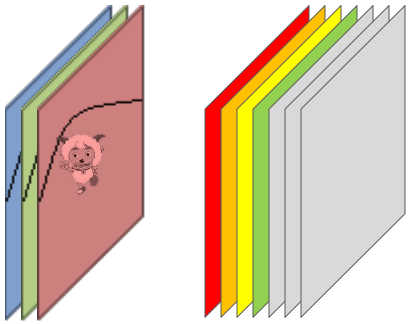
Convolutional Neural Networks

- Generate the third feature map in layer 1



Convolutional Neural Networks

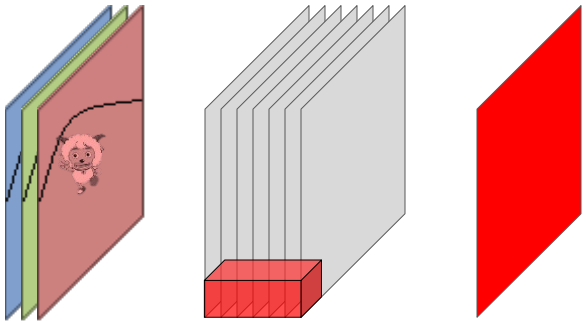
- Generating the first layer



Layer 1 : the features are stacked together to get a “new image” in this example it is an “image” with a depth of 6

Convolutional Neural Networks

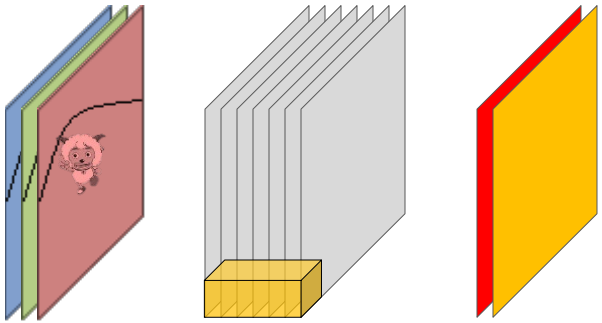
- Generate the first feature map in layer 2



Layer 1 Layer 2

Convolutional Neural Networks

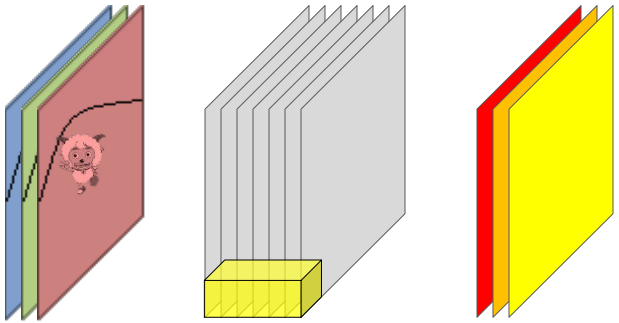
- Generate the second feature map in layer 2



Layer 1 Layer 2

Convolutional Neural Networks

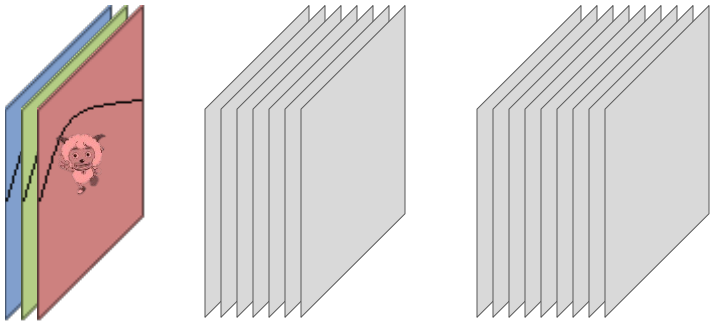
- Generate the third feature map in layer 2



Layer 1 Layer 2

Convolutional Neural Networks

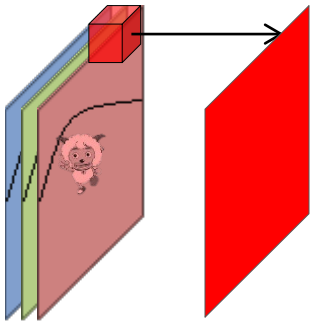
- Generating the second layer



Layer 1 Layer 2

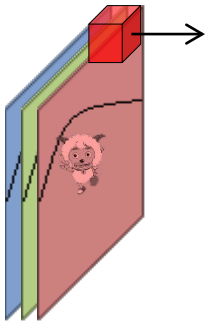
Convolutional Neural Networks

- Activation functions :



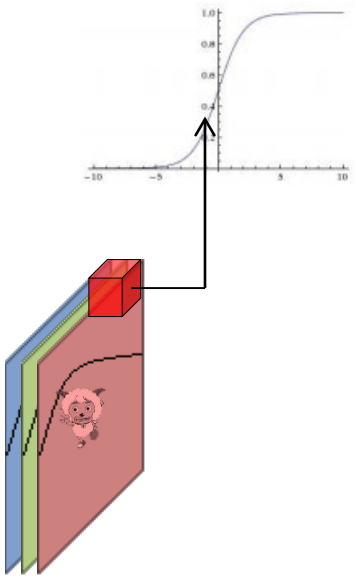
Convolutional Neural Networks

- Activation functions :
  - at the output of the convolution we might have a non-linear mapping function



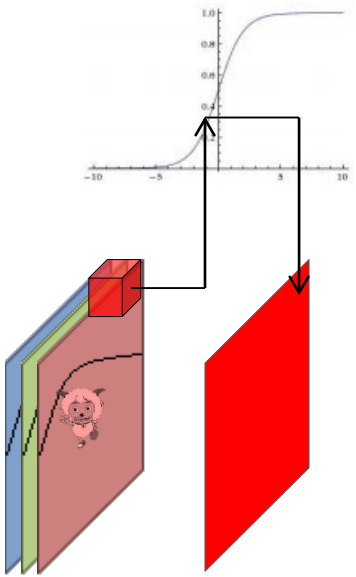
Convolutional Neural Networks

- Activation functions :



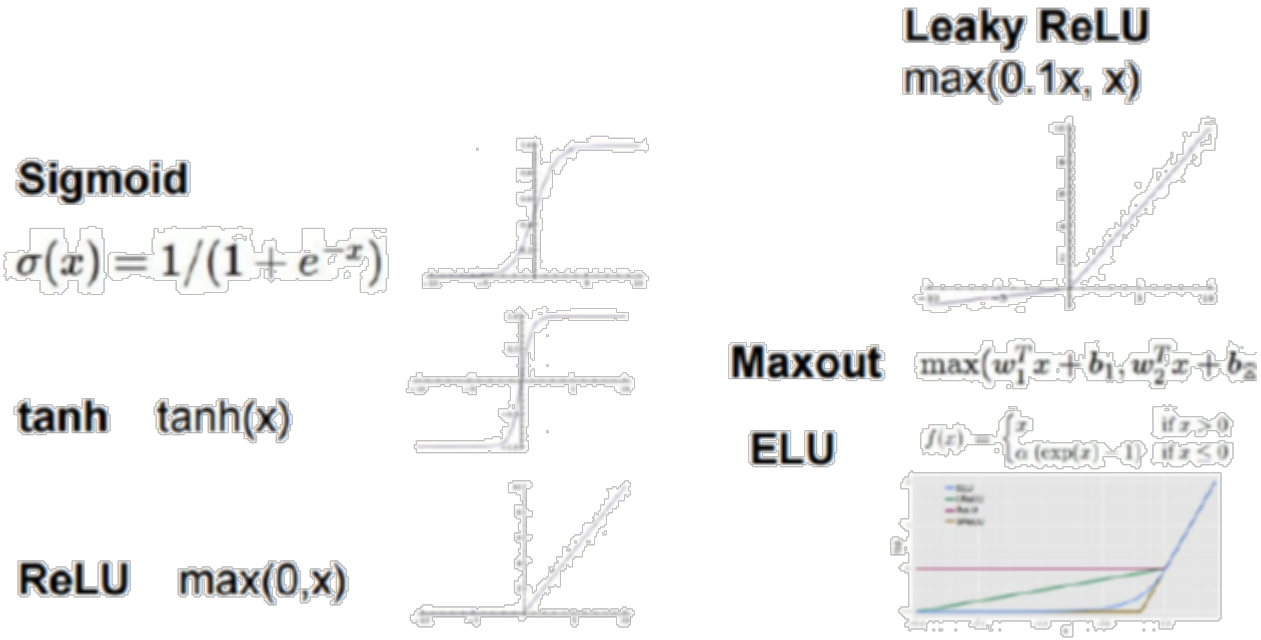
Convolutional Neural Networks

- Activation functions :



Convolutional Neural Networks

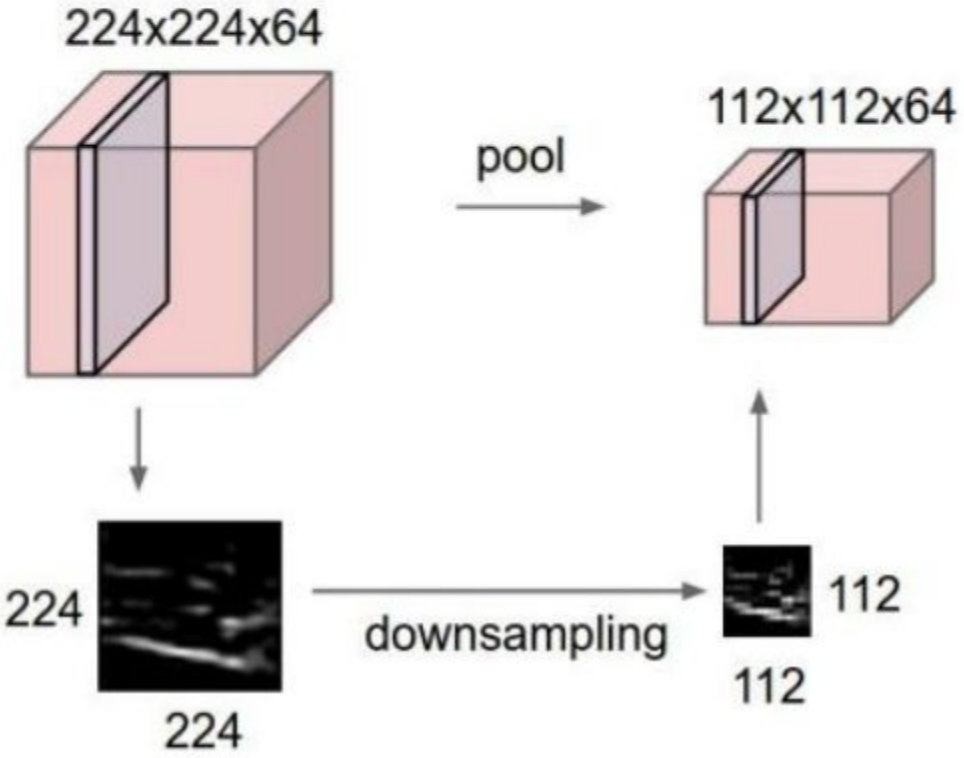
- Activation functions :



Convolutional Neural Networks

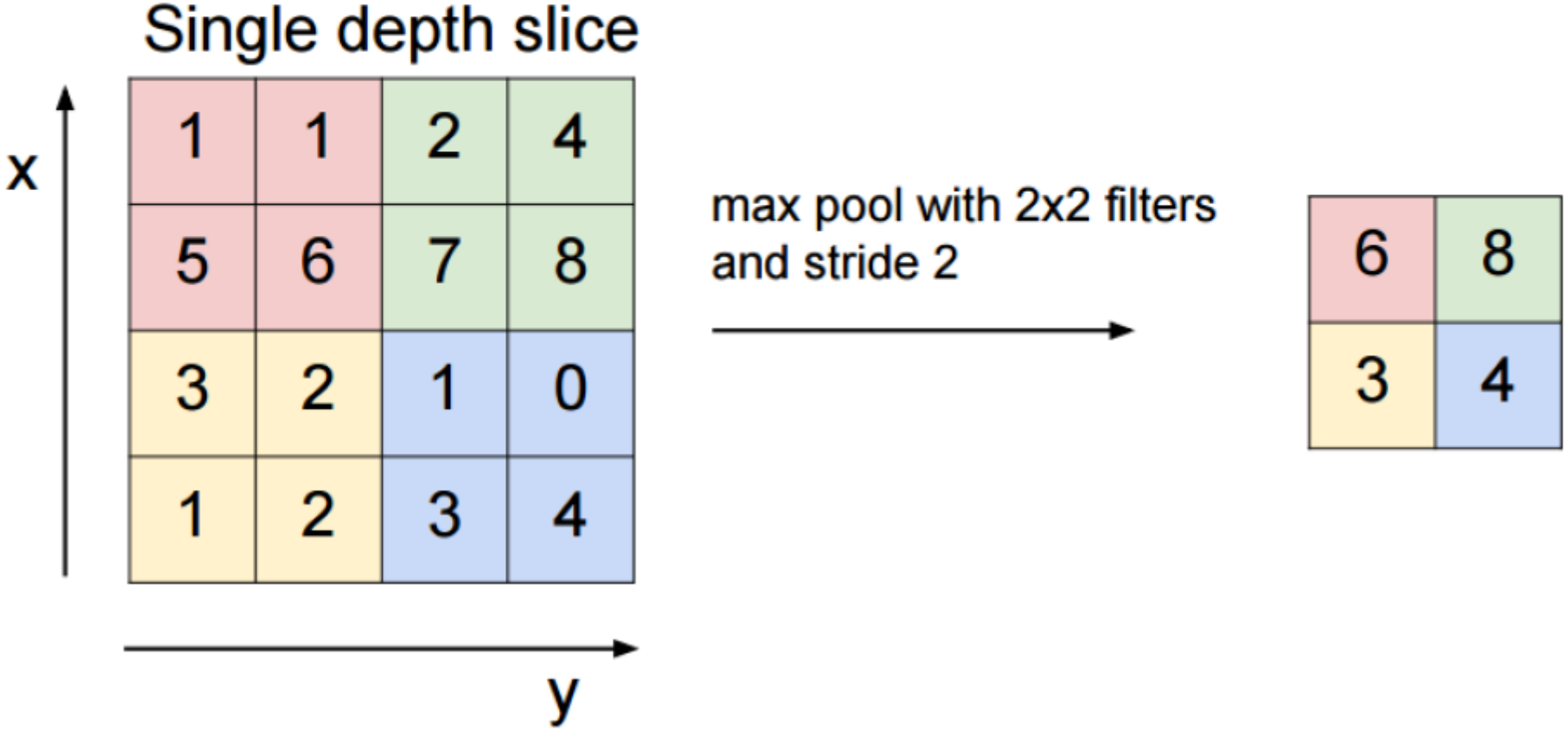
Pooling layer

- makes the representations smaller and more manageable
- operates over each activation map independently:



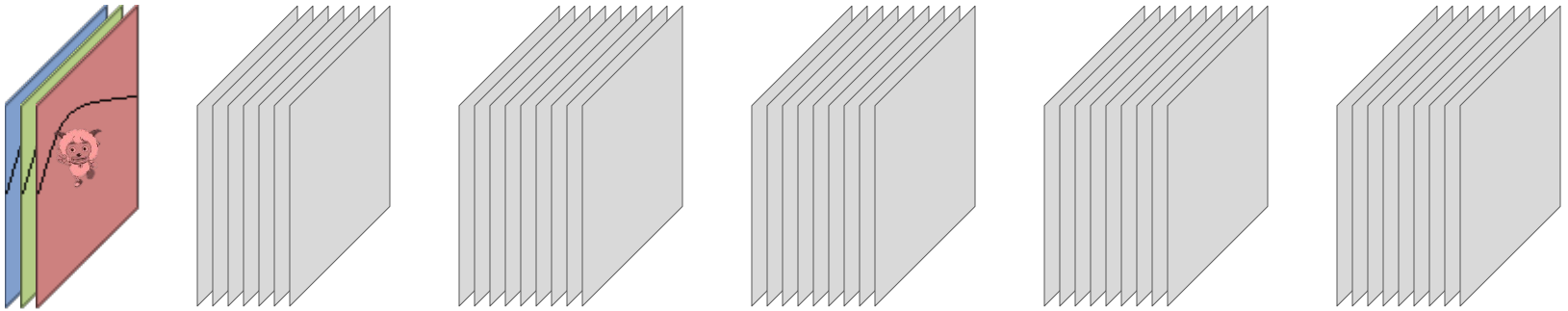
Convolutional Neural Networks

MAX POOLING



Convolutional Neural Networks

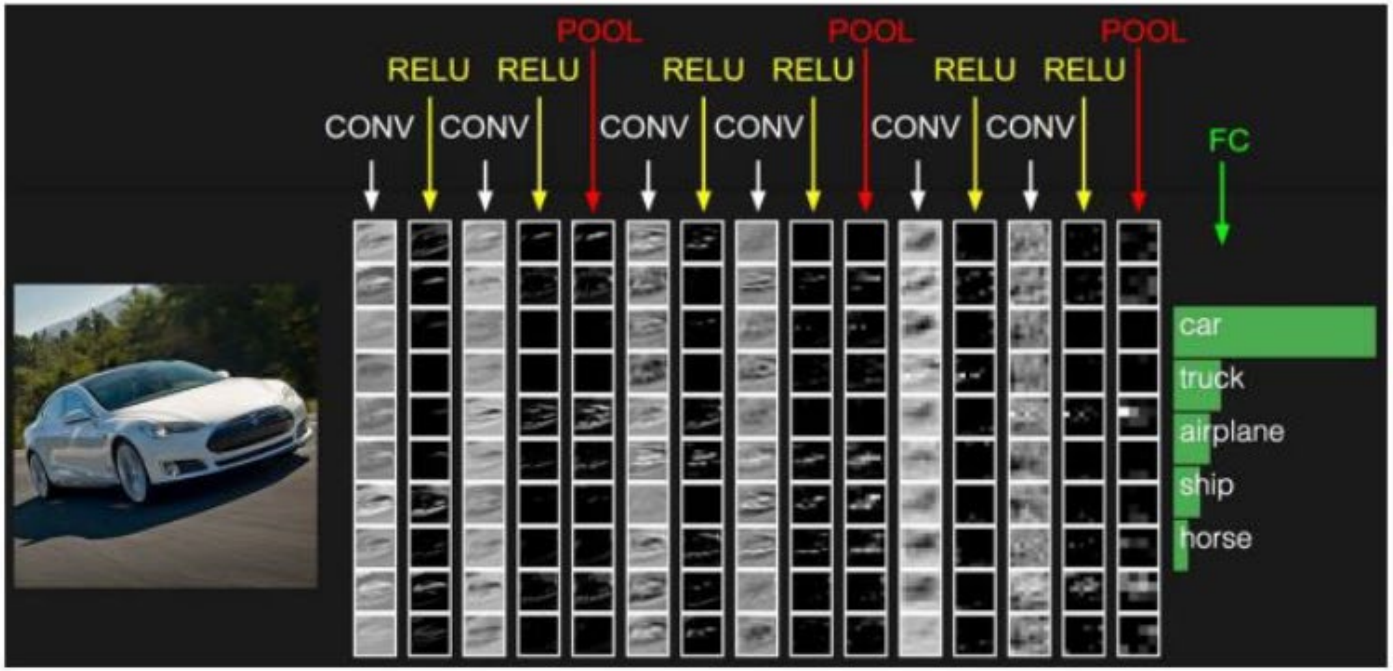
- ConvNet : is a sequence of convolution operations followed by activation functions



Convolutional Neural Networks

Fully Connected Layer (FC layer)

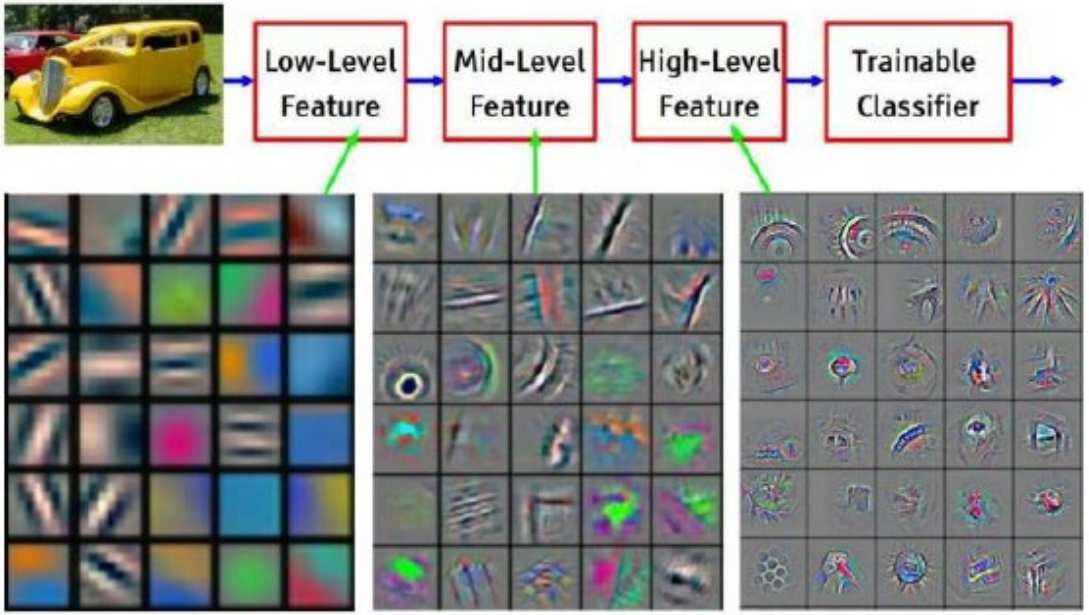
- Contains neurons that connect to the entire input volume, as in ordinary Neur Networks



Convolutional Neural Networks

Preview

[From recent Yann LeCun slides]

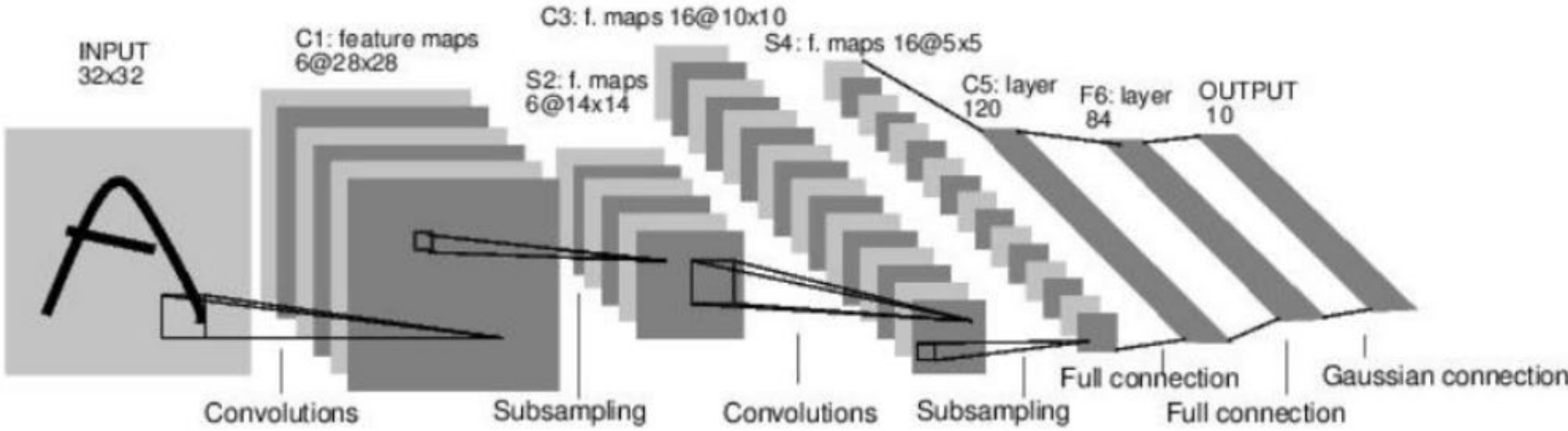


Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

Convolutional Neural Networks

Case Study: LeNet-5

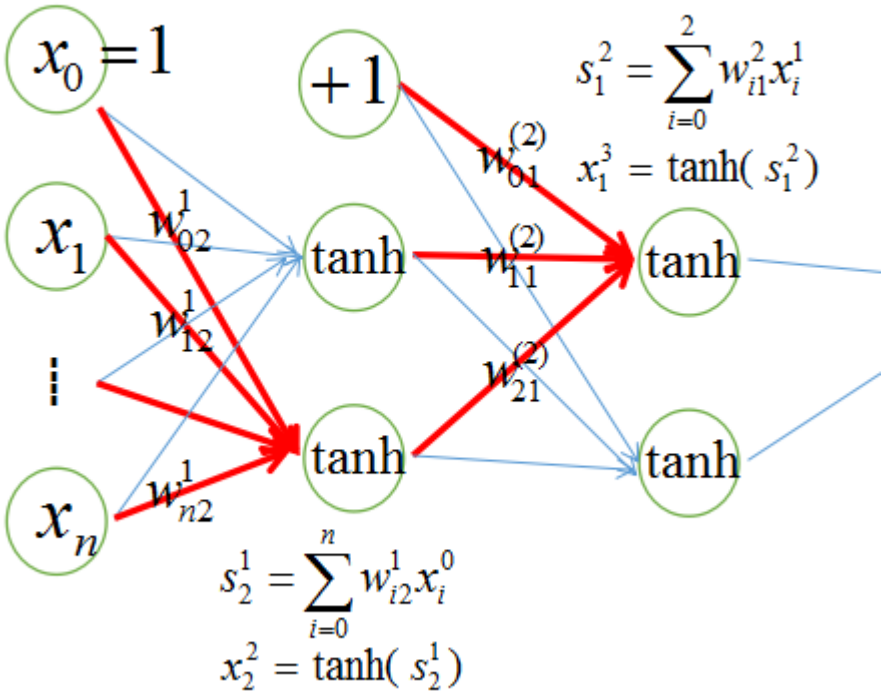
[LeCun et al., 1998]



Conv filters were 5x5, applied at stride 1
Subsampling (Pooling) layers were 2x2 applied at stride 2
i.e. architecture is [CONV-POOL-CONV-POOL-CONV-FC]

Neural Networks

- **Multilayer Perceptron (MLP)**



LeNet-5

- C1: Conv
  - Input :32*32
  - Kernel: 5*5
  - Kernel number: 6
  - Stride: 1
  - NewSize=(OldSize-KernelSize)/Stride+1
  - Parameters: (5*5+1)*6
  - Connections: (5*5+1)*6*28*28

LeNet-5

- S2: Down-sampling
  - Input :28*28
  - Sampling size: 2*2
  - Sampling method: sum*a+b
  - Stride: 2
  - NewSize=(OldSize-KernelSize)/Stride+1
  - Parameters: 2*6
  - Connections: (2*2+1)*6*14*14

LeNet-5

- C3: Conv
  - Input :last layer
  - Kernel size: 5*5
  - Kernel number: 16
  - Parameters: 6* (3*25+1) +6* (4*25+1) +3* (4*25+1) + (25*6+1) =1516
  - Connections:10*10*1516

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	X				X	X	X			X	X	X	X		X	X
1	X	X				X	X	X			X	X	X	X		X
2	X	X	X				X	X	X			X		X	X	X
3		X	X	X			X	X	X	X			X		X	X
4			X	X	X			X	X	X	X		X	X		X
5				X	X	X			X	X	X	X		X	X	X

TABLE I

EACH COLUMN INDICATES WHICH FEATURE MAP IN S2 ARE COMBINED

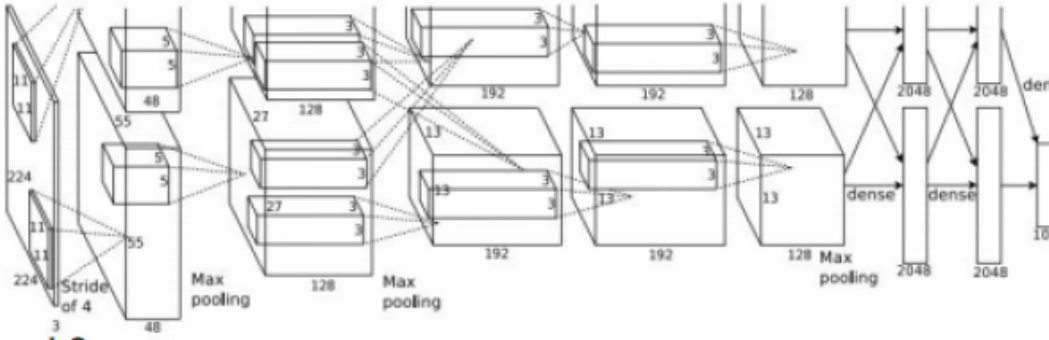
LeNet-5

- S4: Down-sampling
  - Parameters: 2*16
  - Connections:16*(2*2+1)*5*5
- C5: Full Connect
  - Parameters: 120*(16*5*5+1)
  - Connections:120*(16*5*5+1)
- F6: Full Connect
  - Parameters : 84*(120+1)
- Output
  - Gaussian
  - softmax

Convolutional Neural Networks

Case Study: AlexNet

[Krizhevsky et al. 2012]



Full (simplified) AlexNet architecture:
[227x227x3] INPUT
[55x55x96] CONV1: 96 11x11 filters at stride 4, pad 0
[27x27x96] MAX POOL1: 3x3 filters at stride 2
[27x27x96] NORM1: Normalization layer
[27x27x256] CONV2: 256 5x5 filters at stride 1, pad 2
[13x13x256] MAX POOL2: 3x3 filters at stride 2
[13x13x256] NORM2: Normalization layer
[13x13x384] CONV3: 384 3x3 filters at stride 1, pad 1
[13x13x384] CONV4: 384 3x3 filters at stride 1, pad 1
[13x13x256] CONV5: 256 3x3 filters at stride 1, pad 1
[6x6x256] MAX POOL3: 3x3 filters at stride 2
[4096] FC6: 4096 neurons
[4096] FC7: 4096 neurons
[1000] FC8: 1000 neurons (class scores)