



Università  
Ca' Foscari  
Venezia

# Computer Vision

## Intensity transformations

Filippo Bergamasco ([filippo.bergamasco@unive.it](mailto:filippo.bergamasco@unive.it))

<http://www.dais.unive.it/~bergamasco>

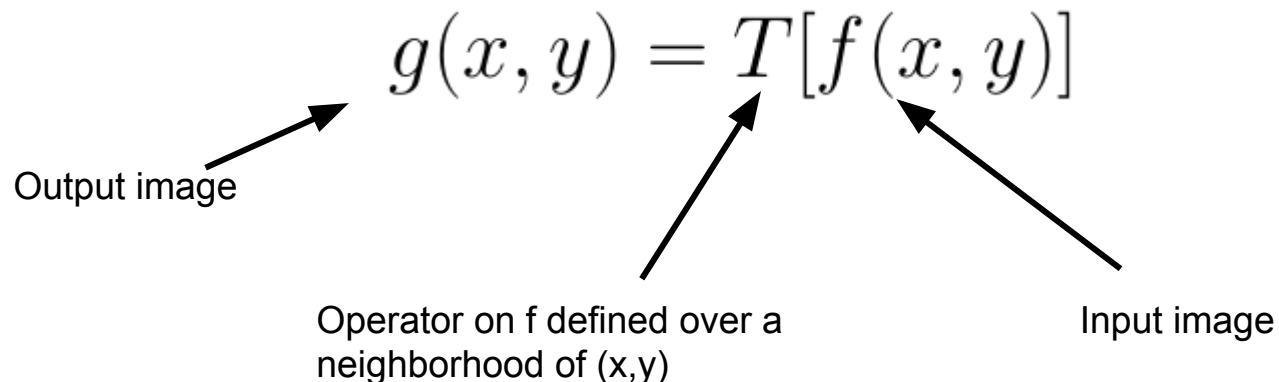
DAIS, Ca' Foscari University of Venice

Academic year 2016/2017

# Introduction

We will discuss techniques that modify the intensity of pixels implemented in the spatial domain (ie. the image plane containing the pixels of the image)

A spatial domain process can be described by the expression:

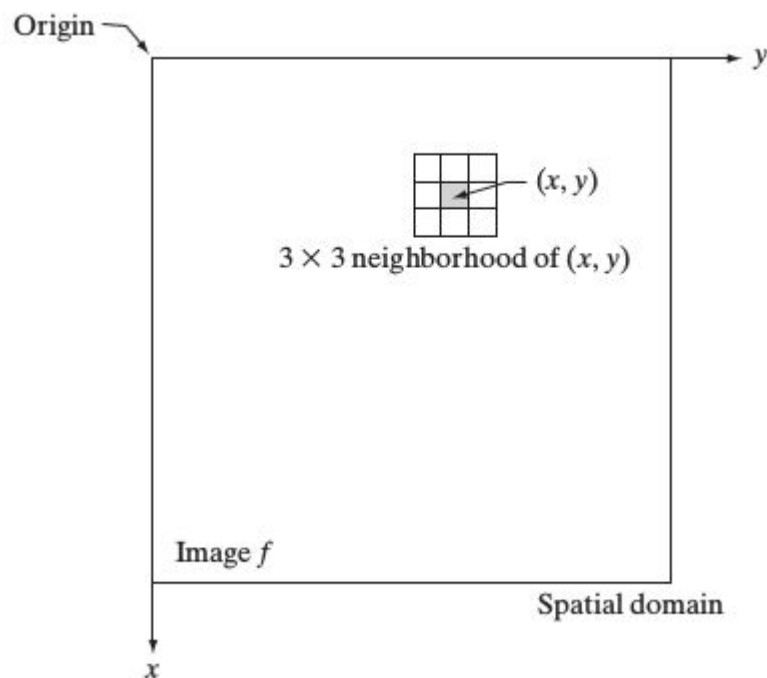
$$g(x, y) = T[f(x, y)]$$
A diagram illustrating the spatial domain process equation. Three arrows point from text labels to the equation. One arrow points from "Output image" to  $g(x, y)$ . Another arrow points from "Operator on f defined over a neighborhood of (x,y)" to  $T$ . A third arrow points from "Input image" to  $f(x, y)$ .

Output image

Operator on f defined over a neighborhood of (x,y)

Input image

# Introduction



Typically, the neighborhood of  $(x, y)$  is:

- Rectangular
- Centered on  $(x, y)$
- Much smaller than the size of the image



# Intensity transformations

When the neighborhood has size 1x1,  $g(x,y)$  depends only on the value of  $f$  at  $(x,y)$

$T$  is an **intensity transformation function**

$$s = T(r)$$

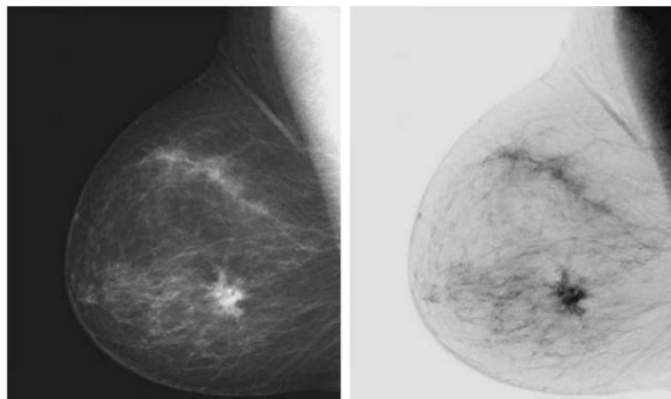
Where  $s$  and  $r$  are the intensity of  $g()$  and  $f()$  at a generic point  $(x,y)$

# Negative

The negative of an image with intensity levels in the range  $[0 \dots L-1]$  is obtained by the following expression:

$$s = L - 1 - r$$

This processing enhances white or gray details embedded in dark regions





# Gain/Bias

Two commonly used point processes are multiplication and addition with a constant:

$$s = \alpha r + \beta$$

The two parameters  $\alpha > 0$  and  $\beta$  are often called *gain* and *bias* and control **contrast** and **brightness** respectively.

# Gain/Bias



Original Image



$$\alpha = 2$$



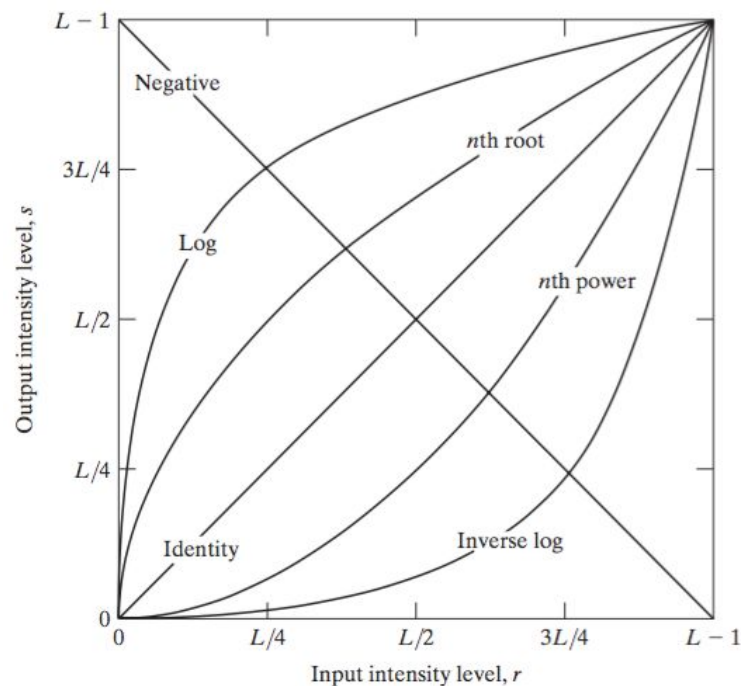
$$\beta = 100$$

# Log Transformations

Log transformations are useful to compress the dynamic range of images (difference between the brightest and darkest pixel intensity) for images with large variation in pixel values

$$s = c \log(1 + r)$$

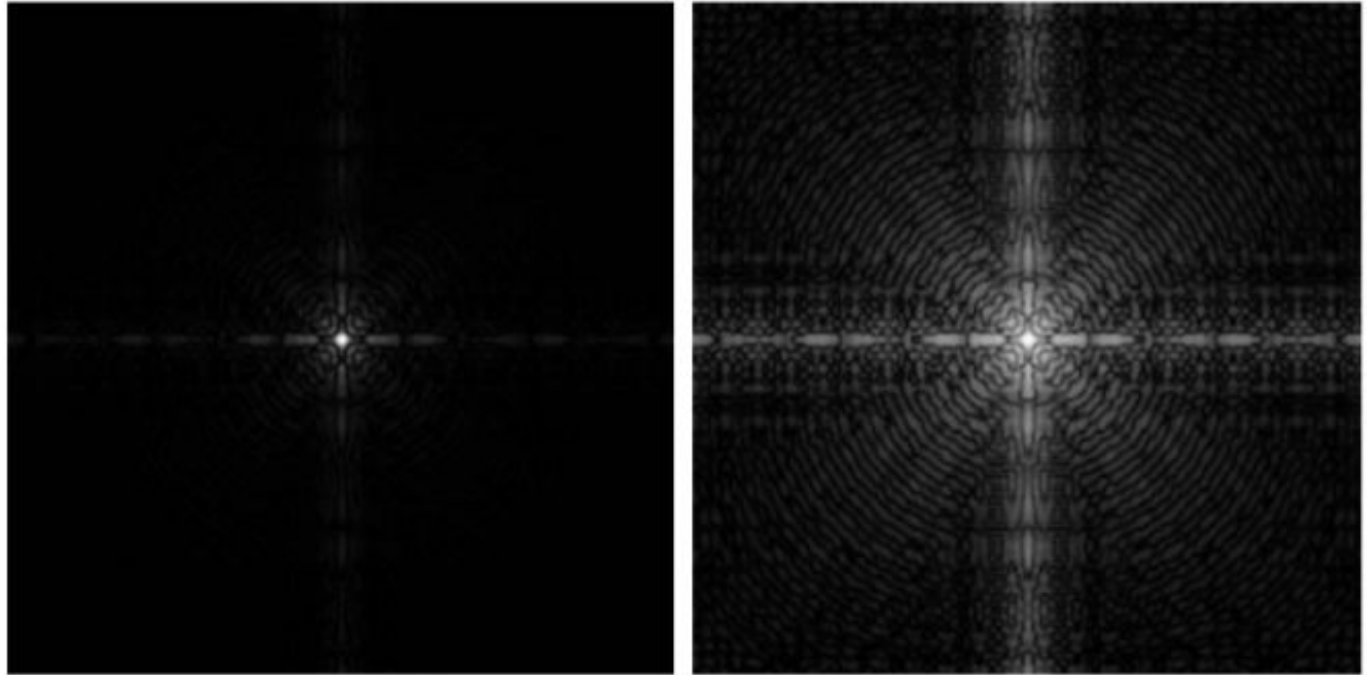
Arbitrary constant





# Log Transformations

$$s = c \log(1 + r)$$



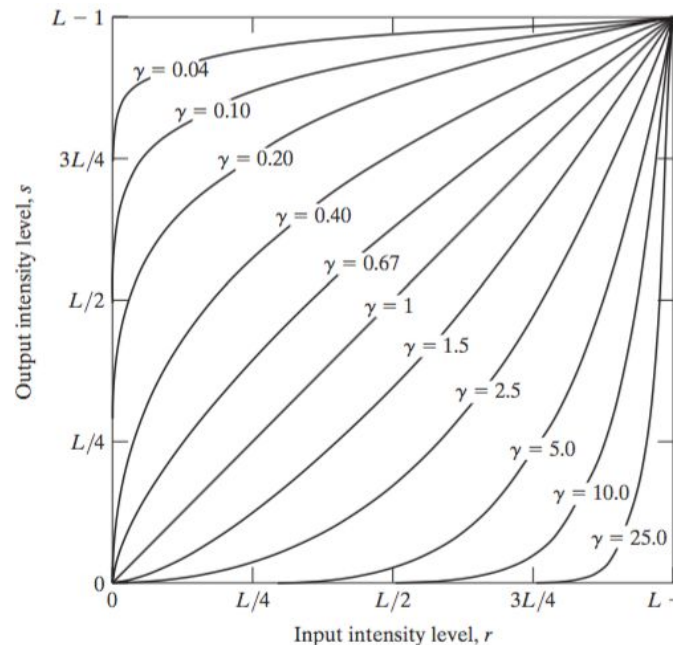
Fourier power spectrum is a nice example of a high dynamic range data

# Gamma Transformations

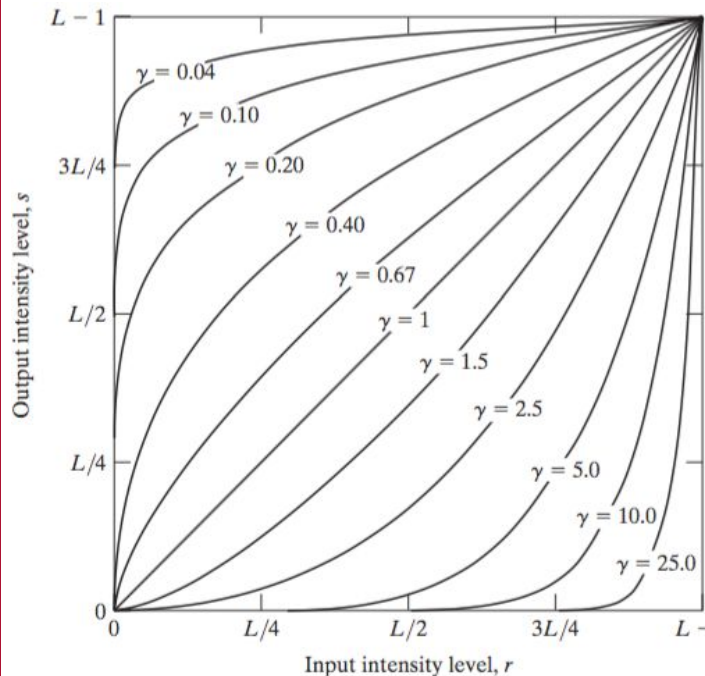
Gamma or power law transformations have the following basic form:

$$s = c r^{\gamma}$$

With  $c$  and  $\gamma$  positive constants.



# Gamma Transformations



- Compress values similar to log transformation but more flexible due to the  $\gamma$  parameter
- Curves generated with a positive  $\gamma$  have the opposite effect of those with negative values
- Identity transformation when  $c=\gamma=1$



# Gamma Transformations

Gamma correction is useful because many image capture/printing/display devices have a power-law response (not linear!).

- For example, old CRT monitors have an intensity-to-voltage response which is a power-law with exponents varying from 1.8 to 2.5
- By using gamma correction we can remove this effect to obtain a response that is similar to the original image

# Gamma Transformations

Monitor response is a  
power-law with  $\gamma = 2.5$

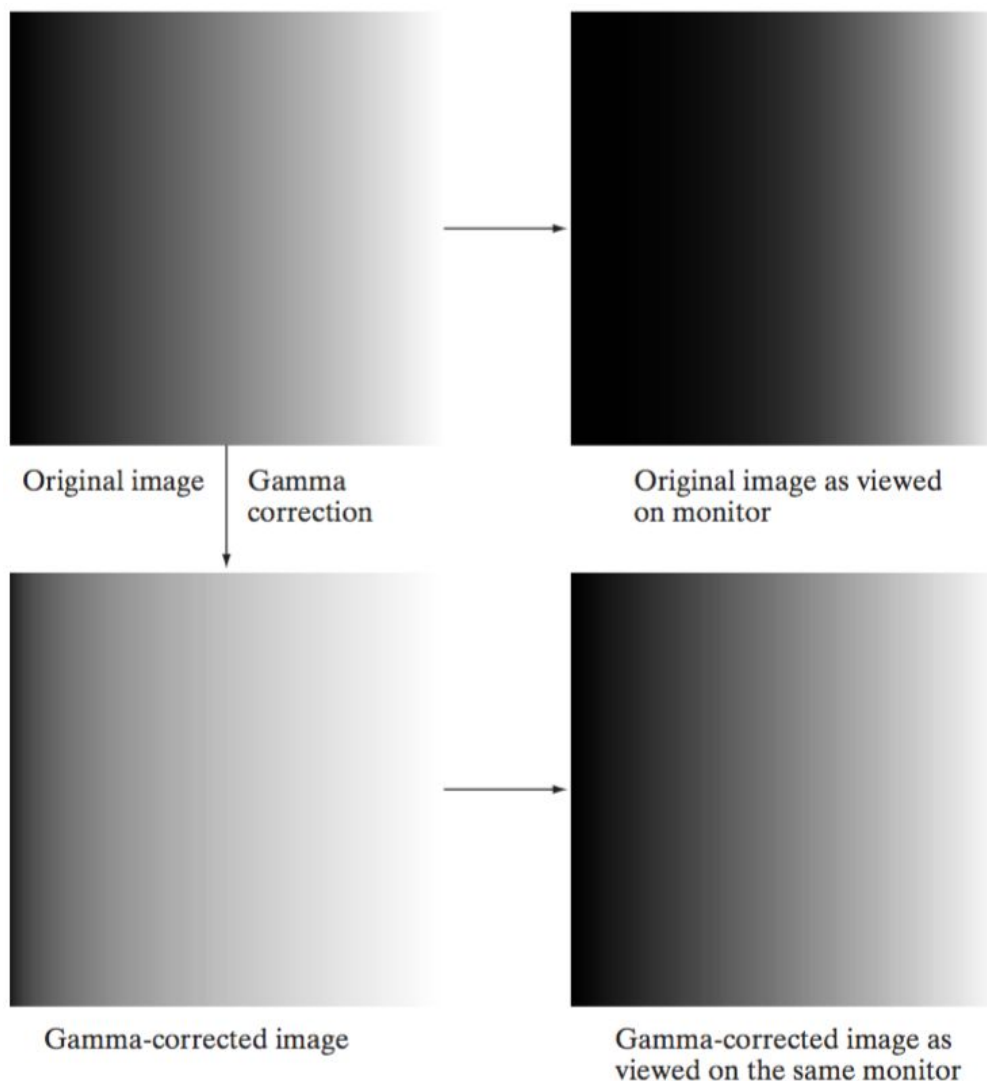


Image is pre-processed by  
applying a gamma  
transformation with  
 $\gamma = 1/(2.5) = 0.4$

# Gamma Transformations

In addition to gamma correction, gamma transformations are useful for general purpose contrast manipulation:

Image  
appears too  
dark



Corrected with  
 $\gamma = 0.6$



Corrected with  
 $\gamma = 0.4$



Corrected with  
 $\gamma = 0.3$



# Gamma Transformations

In addition to gamma correction, gamma transformations are useful for general purpose contrast manipulation:

Image is  
washed out



Corrected with  
 $\gamma = 3$



Corrected with  
 $\gamma = 4$



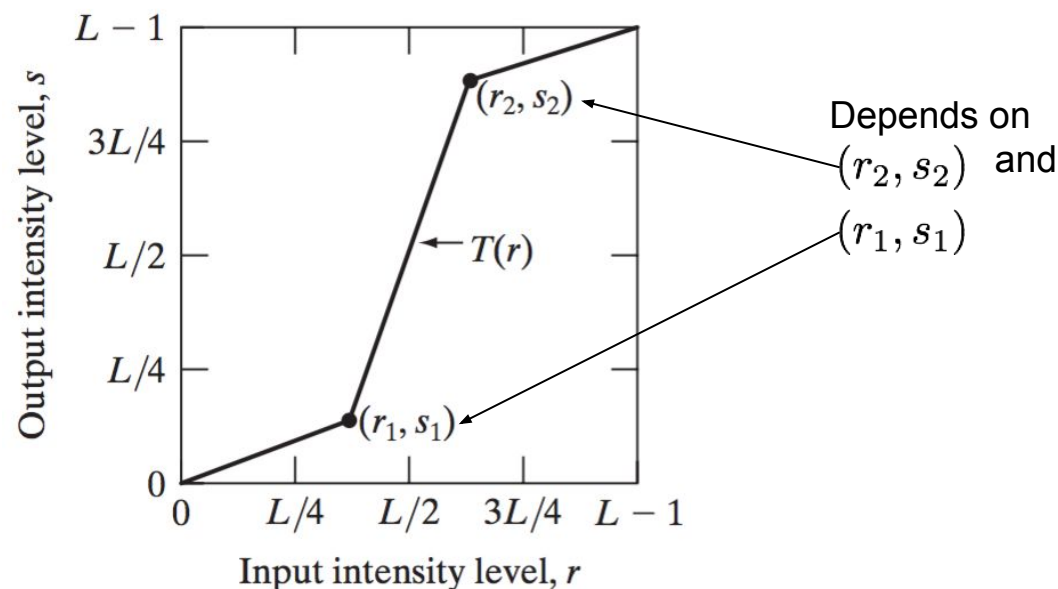
Corrected with  
 $\gamma = 5$



# Contrast Enhancement

A whole family of transformations are defined using piecewise-linear functions.

One of the simplest and most useful piecewise-linear transformation is contrast enhancement

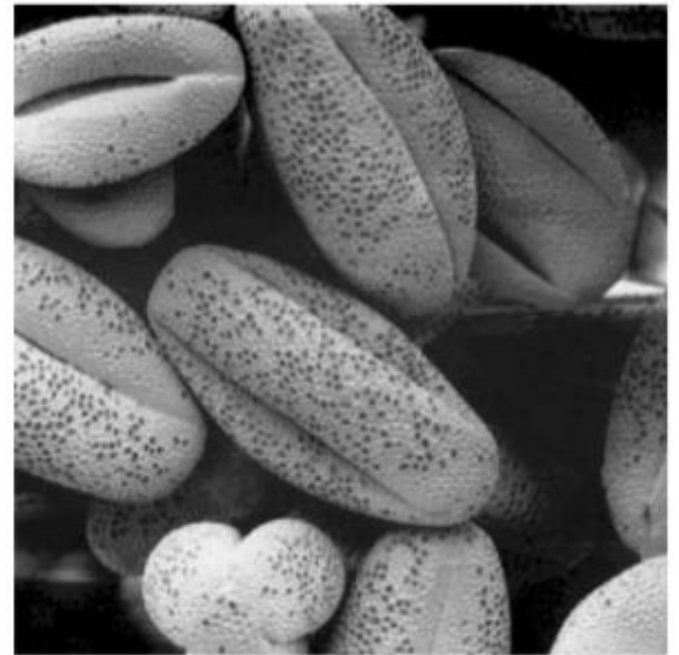
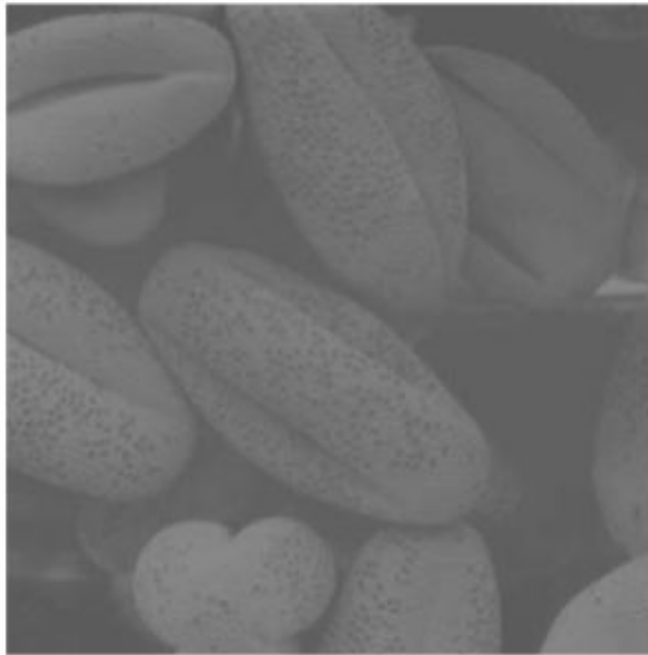






Università  
Ca' Foscari  
Venezia

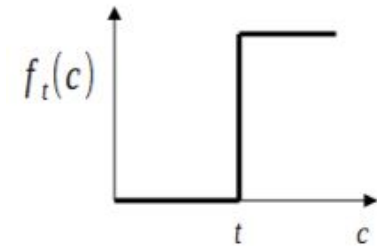
# Contrast Enhancement



# Thresholding

An extreme case of contrast enhancement is the following:

$$s = \begin{cases} 0 & \text{if } r \leq t \\ 1 & \text{if } r > t \end{cases}$$



Where  $t$  is a constant defined for the whole image. If  $t$  depends on the spatial coordinates it is often referred as adaptive thresholding





# Image Histogram

All the function described so far can improve the appearance of an image by varying some *parameters*

**How can we automatically determine their best values?**

One effective tool is the Image Histogram that allows us to analyze problems in the intensity (or color) distribution of an image

Without spatial information we can assimilate  $I(x,y)$  as a random intensity emitter.

**The image histogram is the empirical distribution of image intensities**



# Image Histogram

Let  $[0 \dots L-1]$  be the intensity levels of an image.

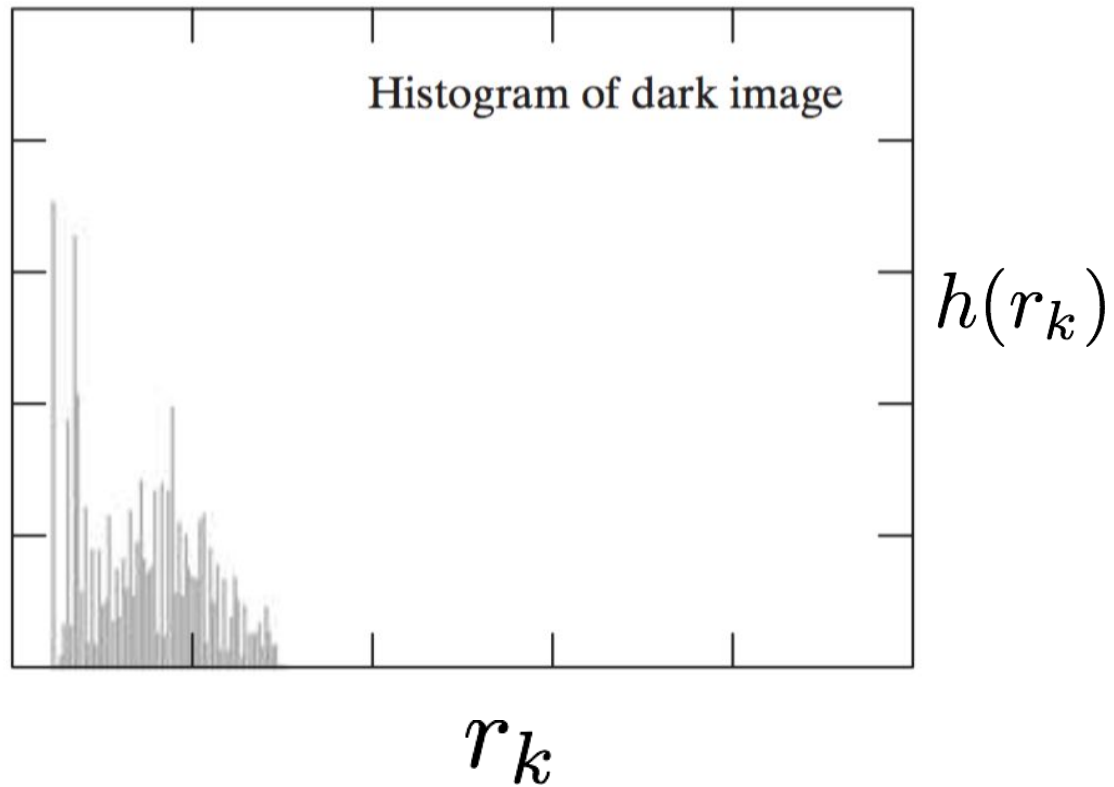
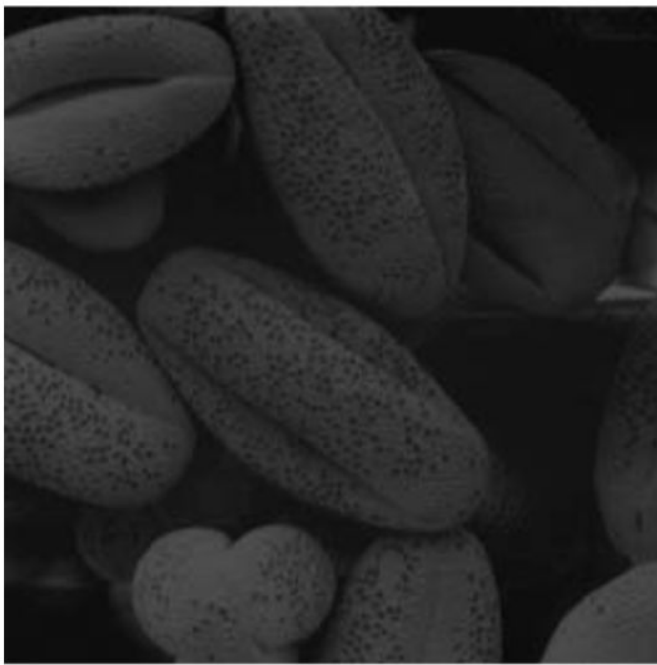
The image histogram is a discrete function

$$h(r_k) = n_k$$

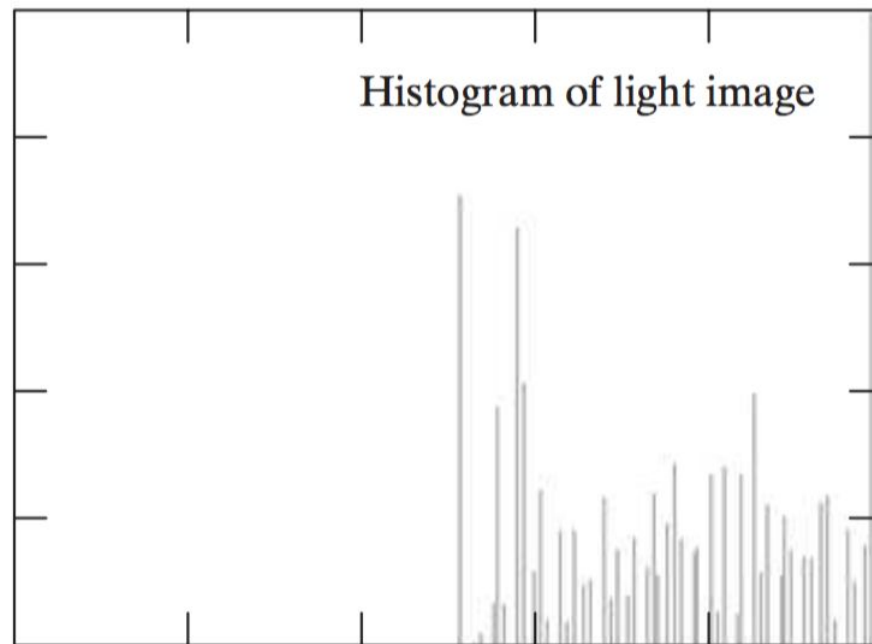
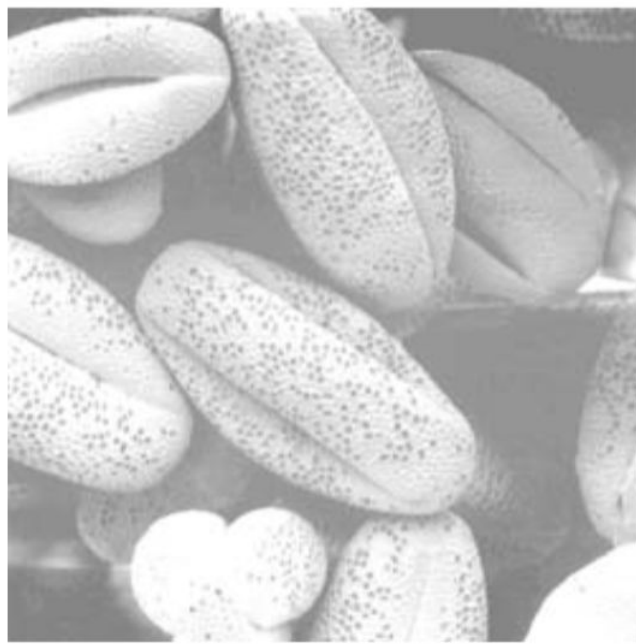
Where  $r_k$  is the  $k^{\text{th}}$  intensity value and  $n_k$  is the number of pixels in the image with intensity  $r_k$

Usually the histogram is normalized by dividing each component to the total number of pixels. This way **each histogram component is an estimate of the probability of the occurrence of the intensity  $r_k$**

# Image Histogram



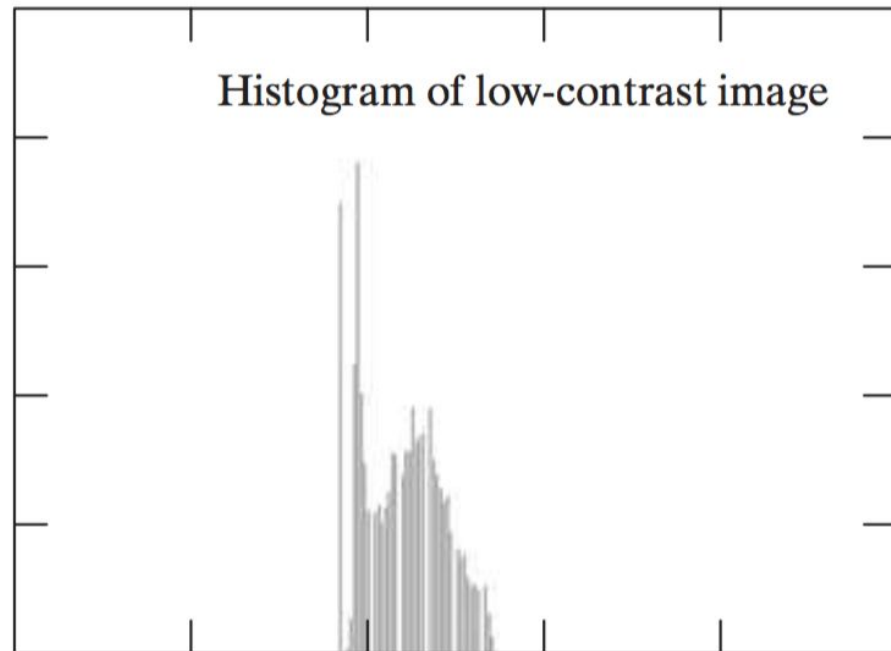
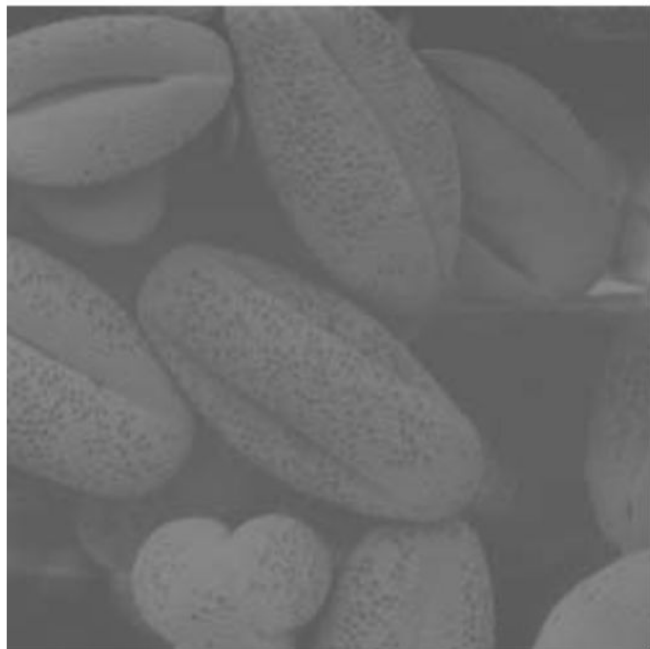
# Image Histogram



$$h(r_k)$$

$$r_k$$

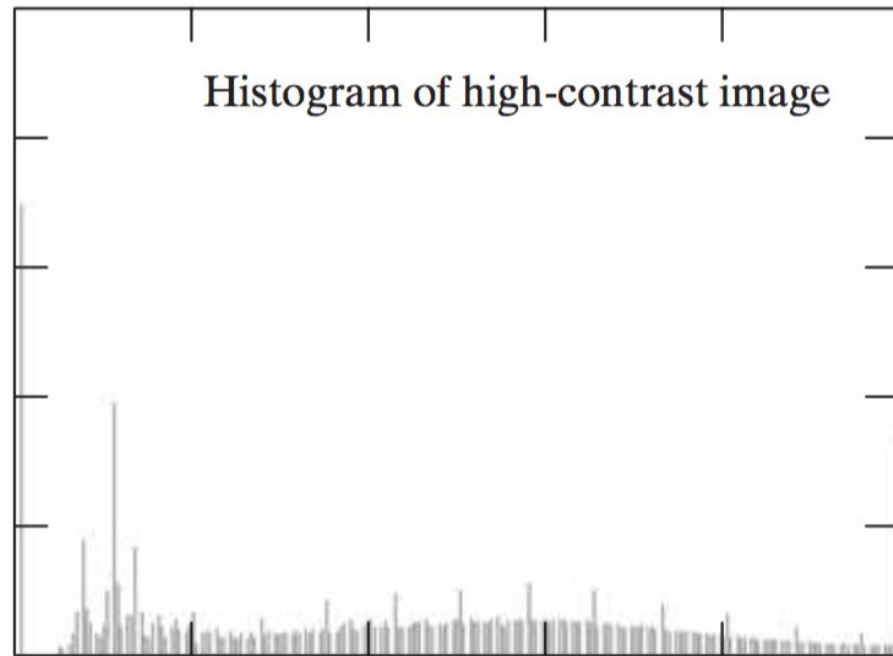
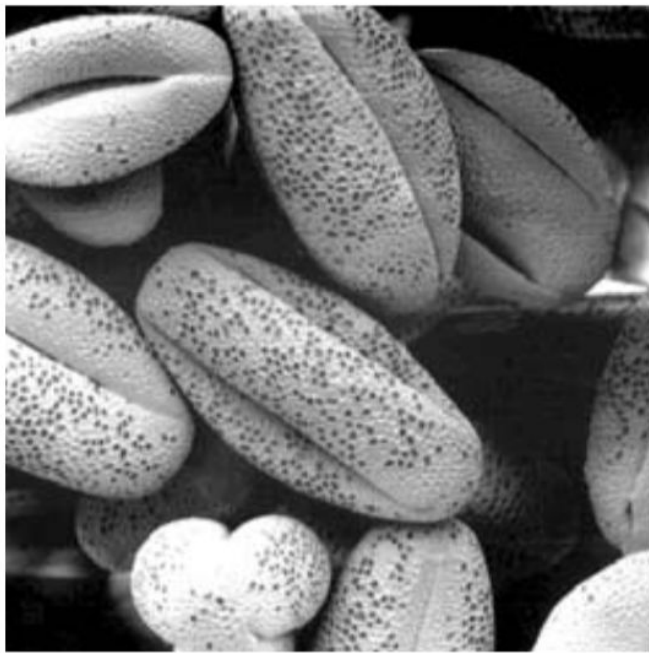
# Image Histogram



$h(r_k)$

$r_k$

# Image Histogram



$$h(r_k)$$

$$r_k$$





# Histogram equalization

When enhancing an image ideally we would like to brighten the dark values and darken the light ones.

- Choosing the correct parameters requires human intervention... how can we automate the process?

A popular answer is to find a mapping function

$$s = T(r)$$

so that the resulting histogram of  $s$  is flat (uniform distribution).



# Histogram equalization

More formally, intensity levels of an image may be viewed as random variables in interval  $[0 \dots L-1]$ .

Image histogram of  $s$  is an estimate of the **PDF** of  $s$  ( $p_s(s)$ ) and histogram of  $r$  is an estimate of  $p_r(r)$ .


From probability theory, when we apply a function  $s=T(r)$  to a random variable  $r$  we got:

$$p_s(s) = p_r(r) \left| \frac{dr}{ds} \right|$$

# Histogram equalization

If we use the function:

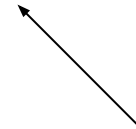
$$T(r) = (L - 1) \int_0^r p_r(w) dw$$

CDF of  $r$  

We have:

$$\frac{ds}{dr} = \frac{dT(r)}{dr} = (L-1) \frac{d}{dr} \left[ \int_0^r p_r(w) dw \right] = (L-1) p_r(r)$$

$$p_s(s) = p_r(r) \frac{dr}{ds} = p_r(r) \frac{1}{(L-1)p_r(r)} = \frac{1}{L-1}$$

Uniform distribution! 

# Histogram equalization

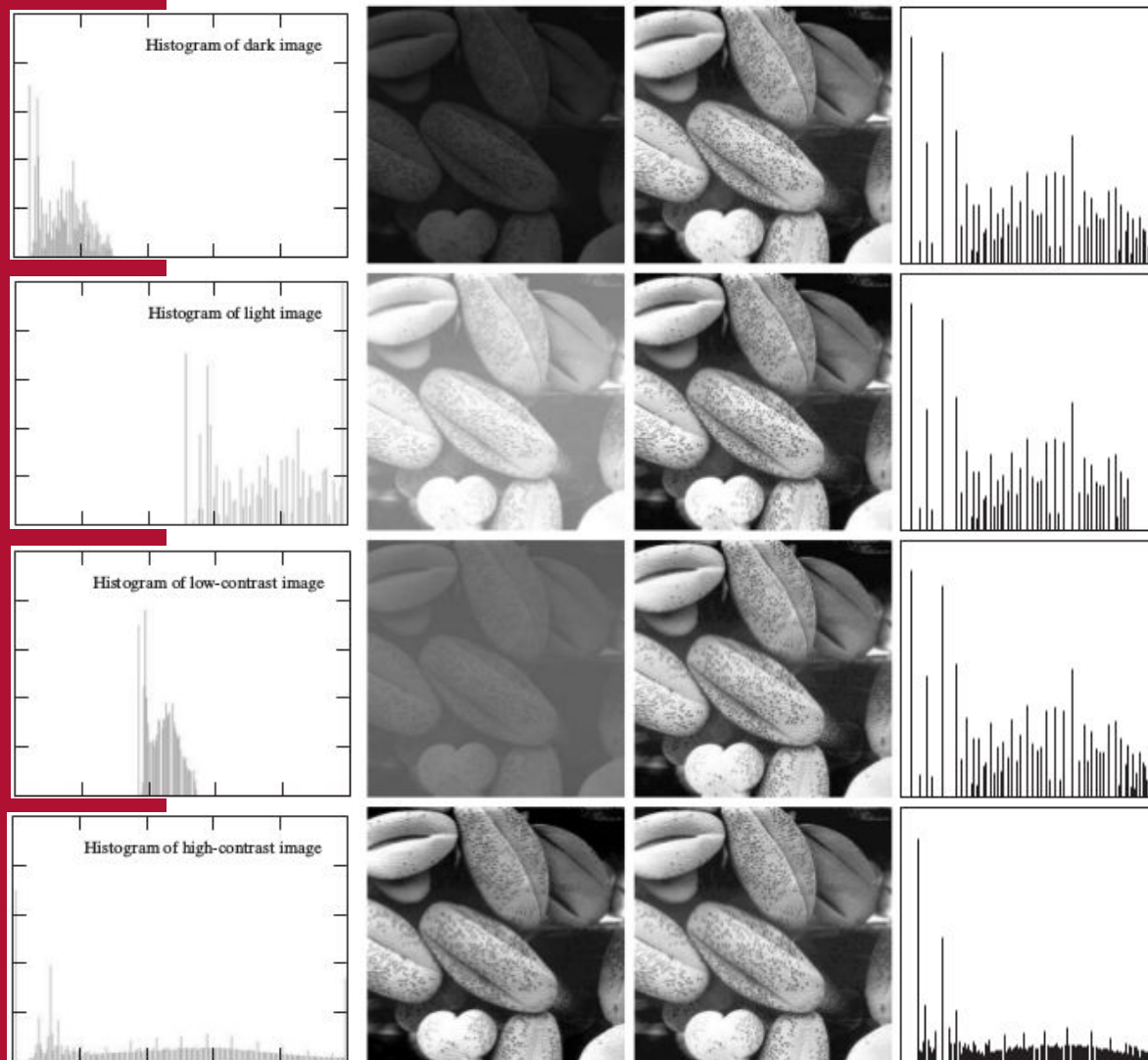
In the discrete case, if we apply the function

$$s_k = T(r_k) = (L-1) \sum_{j=0}^k p_r(r_j) = \frac{L-1}{MN} \sum_{j=0}^k h(r_j)$$

**Sum of the first k components  
of the input image histogram**

We obtain a (quasi) flat histogram of the output image

# Histogram equalization



Since histogram is a discrete approximation of a PDF, the resulting histogram is in general not perfectly flat.

It still remains a good approximation



# Histogram matching

It is useful sometimes to be able to specify the shape of the histogram that we wish the processed image to have (instead of a simple flat one).

Histogram equalization discussed so far can be used also for histogram matching

Suppose that we have an input image with intensities described by  $r$  with PDF  $p_r(r)$  and a *specified* PDF described by  $z$  with a given PDF  $p_z(z)$

# Histogram matching

Let  $s$  be a random variable with the following property:

$$s = T(r) = (L - 1) \int_0^r p_r(w) dw$$

(ie.  $s$  is the equalized version of  $r$ )

We define a function  $G(z)$  as following:

$$G(z) = (L - 1) \int_0^z p_z(t) dt$$

Since  $G(z)=T(r)$ , and both are monotonically increasing, we have  $z = G^{-1}[T(r)] = G^{-1}(s)$



# Histogram matching

Algorithm:

1. Compute the PDF (normalized histogram) of the input image  $p_r(r)$
2. Use the *specified* PDF  $p_z(z)$  to obtain the function  $G(z)$
3. Obtain the inverse transformation  $z = G^{-1}(s)$
4. Equalize the input image. Apply the function  $G^{-1}(s)$  to the equalized image to obtain the corresponding output image.

When all pixels are processed, the PDF of the output image will be equal to the specified PDF



# Histogram for thresholding

Let's go back to the thresholding operation, which is a common step in many cv applications:

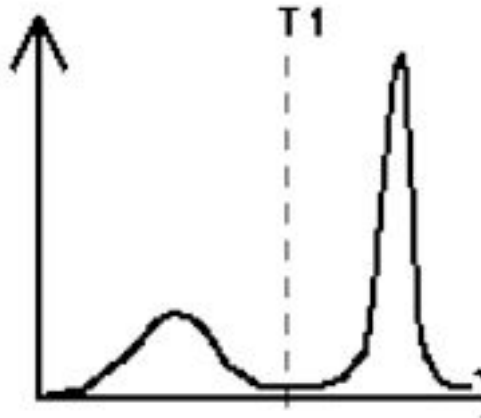


When using global thresholding a common problem is to automatically find a good threshold  $t$  that separates well dark from bright areas

$$s = \begin{cases} 0 & \text{if } r \leq t \\ 1 & \text{if } r > t \end{cases}$$

# Histogram for thresholding

Image histogram can give us useful clues on the threshold level



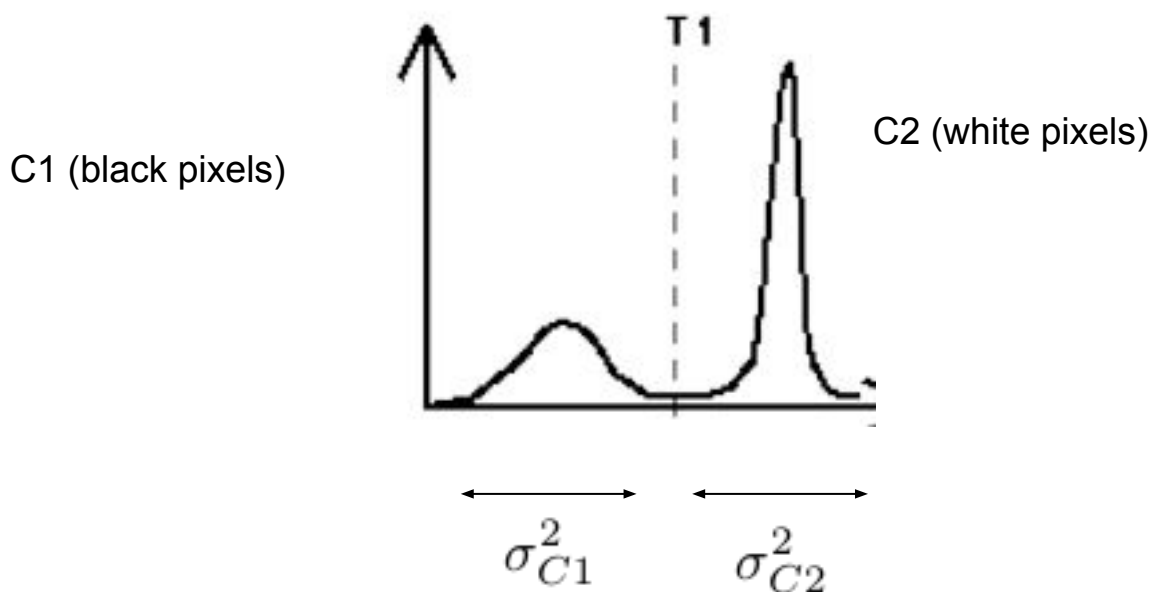
If an image is separable through thresholding there will be a range of intensity with low probability

Thresholding is essentially a **clustering problem** in which two clusters (black and white pixels) are sought

# Otsu Thresholding

The idea is to find the optimum threshold so that the variance of each class (**within-class variance**) is minimized

$$\operatorname{argmin}_T P_1 \sigma_{C_1}^2 + P_2 \sigma_{C_2}^2$$



# Otsu Thresholding

$$\operatorname{argmin}_T P_1 \sigma_{C_1}^2 + P_2 \sigma_{C_2}^2$$

Probability that a pixel is  
assigned to C1 given a  
threshold T

$$P_1(T) = \sum_{i=0}^T p_i$$

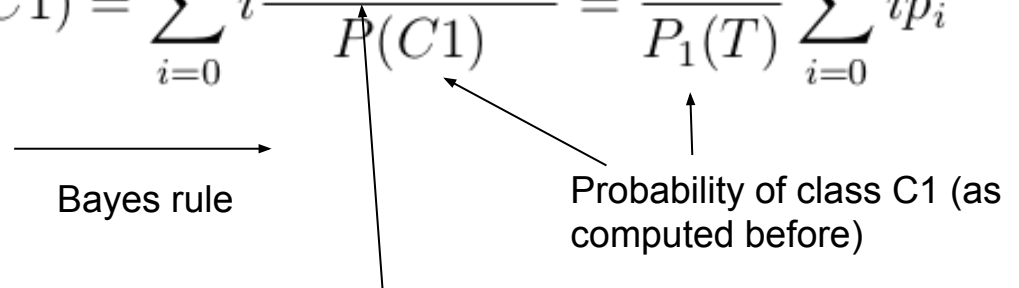
Probability that a pixel is  
assigned to C2 given a  
threshold T

$$P_2(T) = \sum_{i=T+1}^{L-1} p_i = 1 - P_1(T)$$

# Otsu Thresholding

Mean intensity values for the pixels assigned to C1:

$$m_1(T) = \sum_{i=0}^T i P(i|C1) = \sum_{i=0}^T i \frac{P(C1|i)P(i)}{P(C1)} = \frac{1}{P_1(T)} \sum_{i=0}^T i p_i$$



Bayes rule

Always 1 because we are dealing only with values  $i$  from C1

Probability of class C1 (as computed before)

Mean intensity values for the pixels assigned to C2:

$$m_2(T) = \sum_{i=T+1}^{L-1} i P(i|C2) = \frac{1}{P_2(T)} \sum_{i=T+1}^{L-1} i p_i$$

# Otsu Thresholding

C1 class variance:

$$\sigma_{C1}^2(T) = \frac{1}{P_1(T)} \sum_{i=0}^T (i - m_1(T))^2 p_i$$

C2 class variance:

$$\sigma_{C2}^2(T) = \frac{1}{P_2(T)} \sum_{i=T+1}^{L-1} (i - m_2(T))^2 p_i$$

# Otsu Thresholding

What is the best threshold?

Operatively, we can try all the possible  $T$  from 0 to  $L-1$  and keep the threshold for which

$$P_1(T)\sigma_{C_1}^2(T) + P_2(T)\sigma_{C_2}^2(T)$$

Is minimum.

**Problem:** Is computationally expensive to compute

$$\sigma_{C_1}^2(T), \sigma_{C_2}^2(T)$$

**Solution:** Otsu demonstrated that the optimal  $T$  that minimizes the *within-class-variance* also maximizes the *between-class-variance*

# Otsu Thresholding

Between-class-variance:

$$\sigma_B^2(T) = \frac{[m_G P_1(T) - m(T)]^2}{P_1(T)[1 - P_1(T)]}$$

$$m_G = \sum_{i=0}^{L-1} i p_i$$

Global mean

$$m(T) = \sum_{i=0}^T i p_i$$

Cumulative mean

Optimum  $T$  can be now easily computed efficiently



# Otsu Thresholding

Otsu Algorithm for optimal global thresholding:

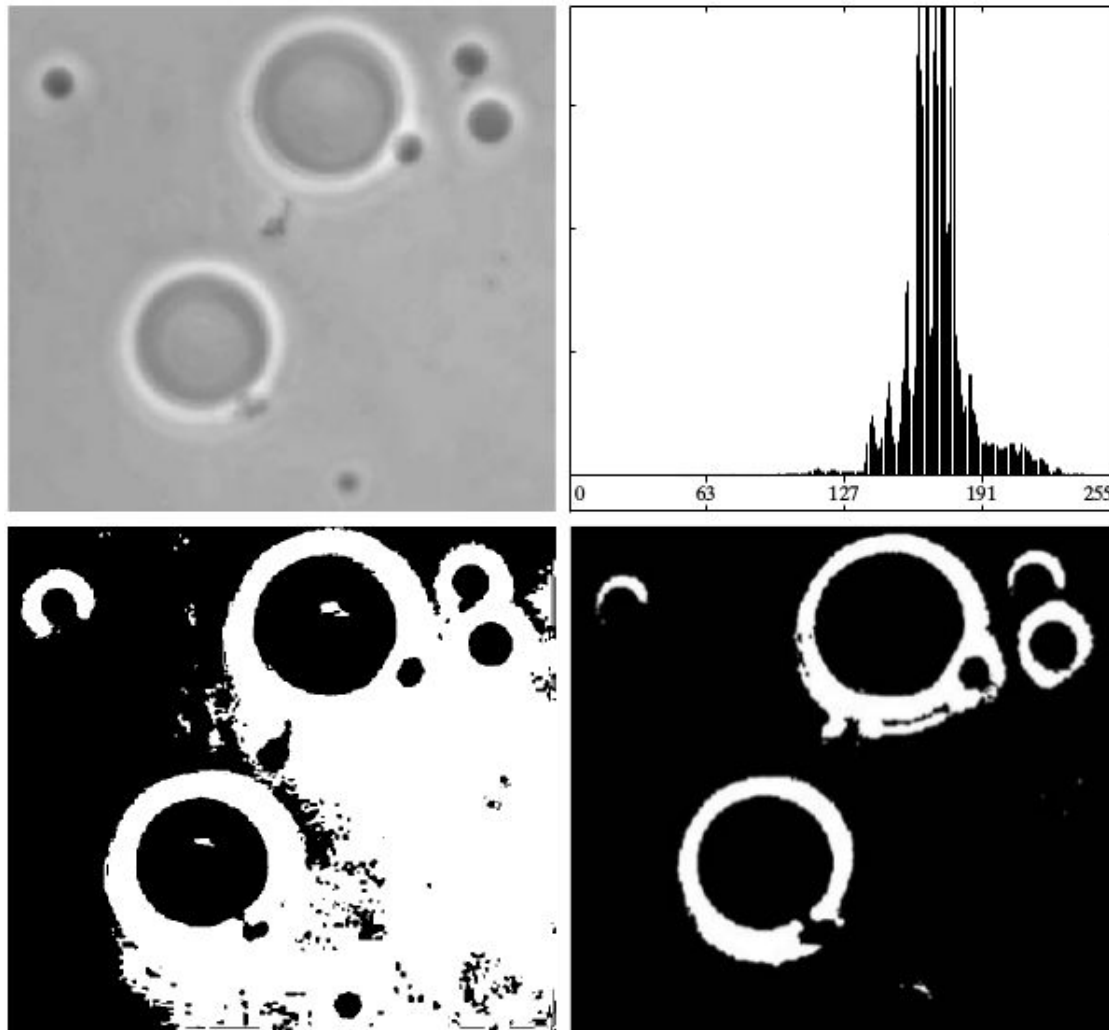
1. Compute the normalized histogram of the input image.  
Denote each component of the histogram as

$$p_i, i = 0, 1, \dots, L - 1$$

2. Compute the cumulative sums  $P_1(T) \quad \forall T = 0 \dots L - 1$
3. Compute the cumulative means  $m(T) \quad \forall T = 0 \dots L - 1$
4. Compute the global intensity mean  $m_G$
5. Compute the between class variance  $\sigma_B^2(T) \quad \forall T = 0 \dots L - 1$
6. Apply threshold with a value of  $T$  for which  $\sigma_B^2(T)$  is maximum

Otsu thresholding iterates on image histogram and not on image pixels as other global methods!

# Otsu Thresholding



K-means based

Otsu