

Course

« *Computer Vision* »

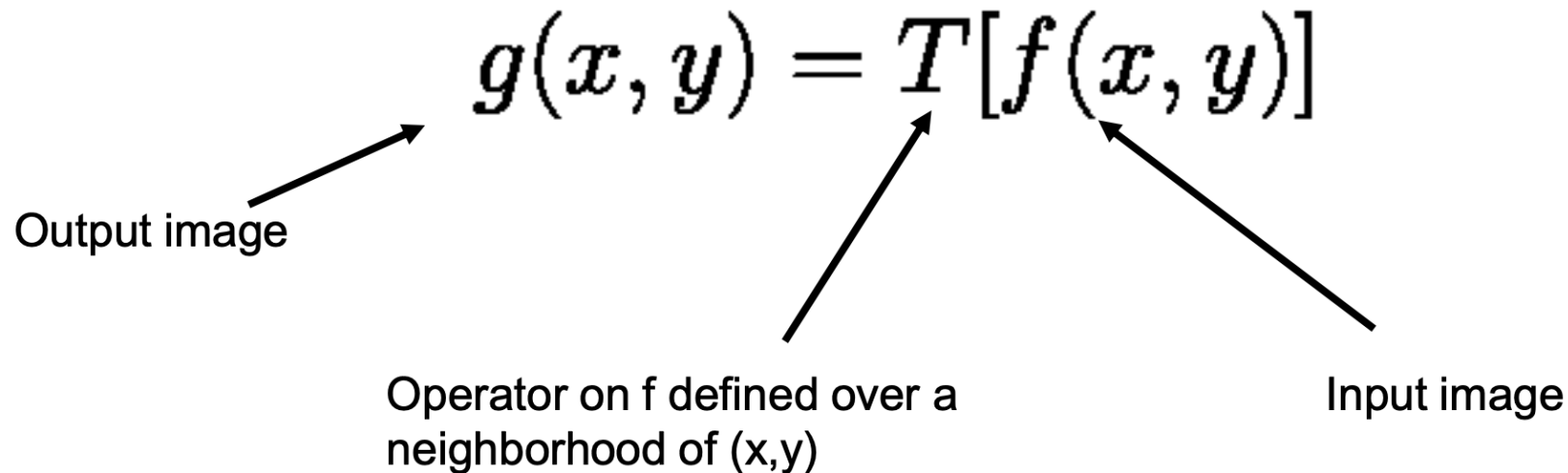
Sid-Ahmed Berrani

2024-2025



Image enhancement: Intensity transformation

- This part focuses on techniques that modify the intensity of pixels implemented in the **spatial domain** (i.e. the image plane containing the pixels of the image).
- A spatial domain process can be described by the expression:

$$g(x, y) = T[f(x, y)]$$


Output image

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

Input image

Image enhancement: Intensity transformation

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

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

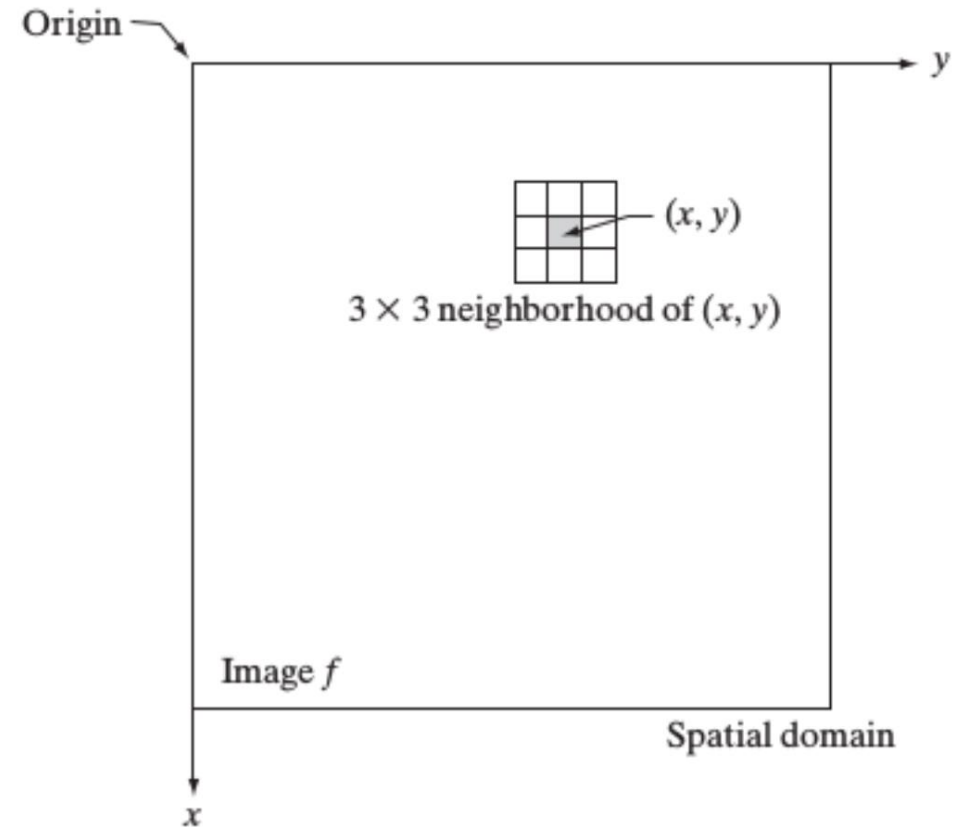


Image enhancement: Intensity transformation

- 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)

Image enhancement: Intensity transformation

- **The negative**

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

$$s = L - 1 - r$$

=> This processing enhances white or gray details embedded in dark regions

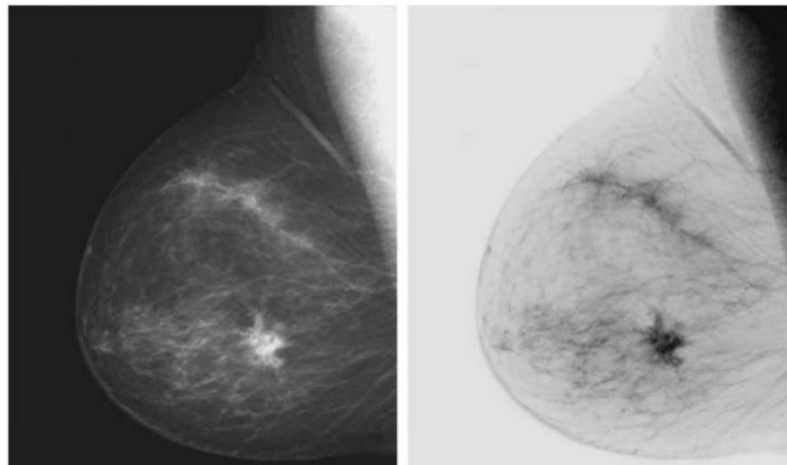


Image enhancement: Intensity transformation

- **Gain/Bias**

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

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

The two parameters $\alpha > 0$ and β are often called *gain* and *bias*.

α controls **contrast**

β controls **brightness**

Image enhancement: Intensity transformation

- Gain/Bias

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



Original Image



$\alpha = 2$



$\beta = 100$

Image enhancement: Intensity transformation

- **Log Transformations**

Log transformations are useful to **compress the dynamic range** for images with large variation in pixel values.

It consists in replacing each pixel value with its logarithm.

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

c is the scaling constant represented by the following expression (for 8-bit):

$$c = 255 / (\log(1 + \text{max_input_pixel_value}))$$

Image enhancement: Intensity transformation

- **Log Transformations**

The value of c is chosen such that we get the maximum output value corresponding to the bit size used. e.g for 8 bit images.

That is, c is chosen such that we get max value equal to 255.

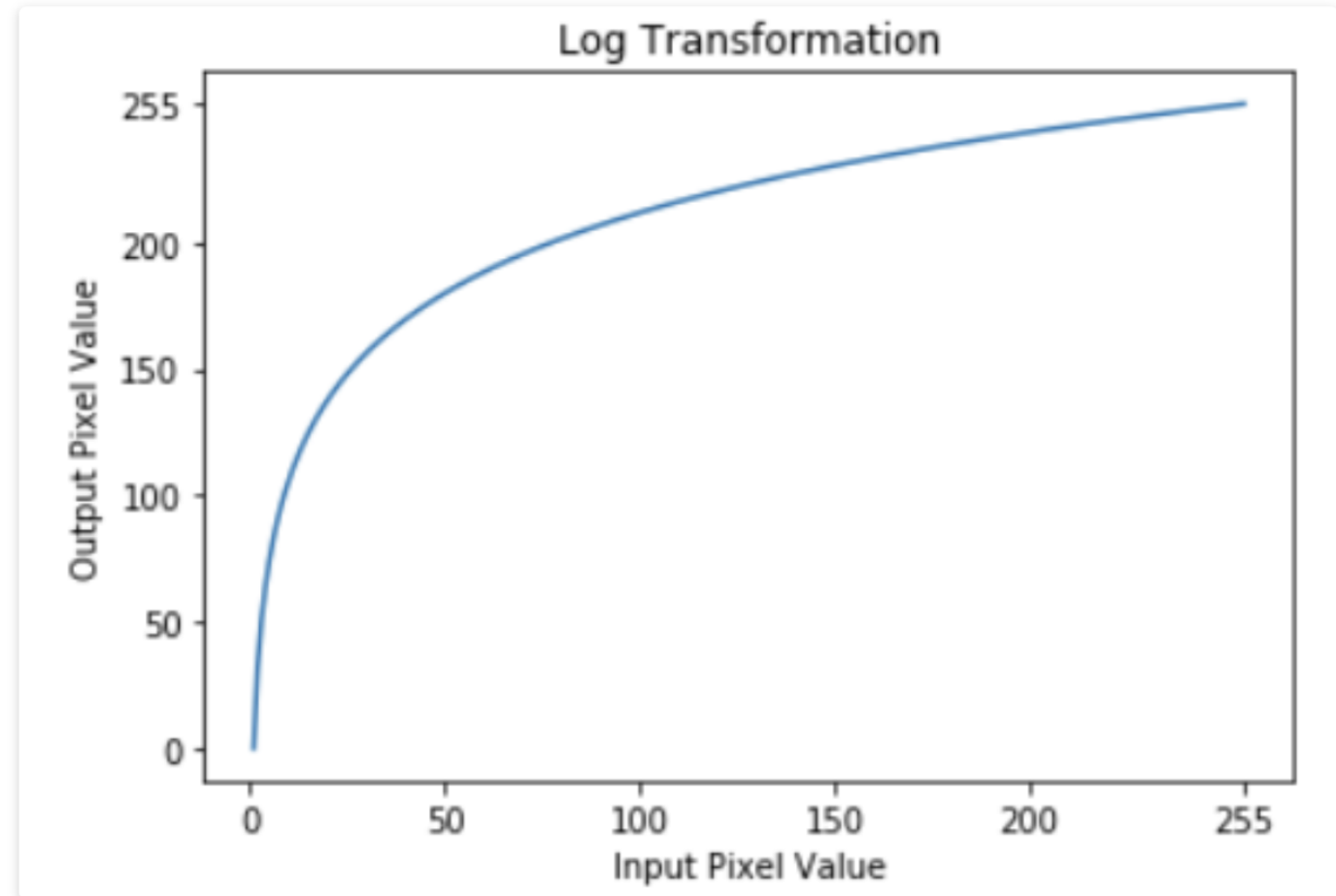


Image enhancement: Intensity transformation

- **Log Transformations**

=> Maps a wide range of high intensity values in input to a narrow range of output levels.

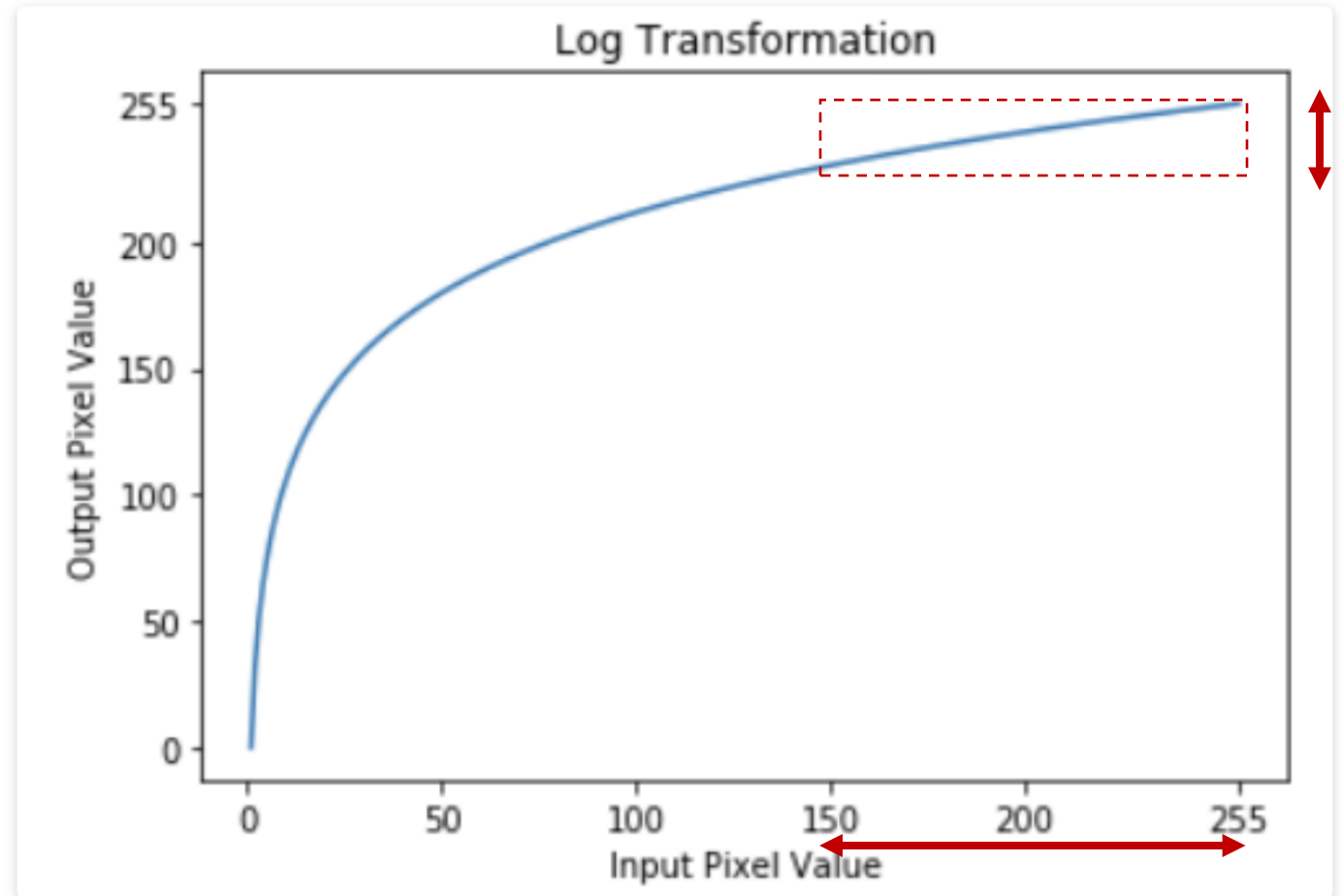


Image enhancement: Intensity transformation

- **Log Transformations**

=> low intensity values in the input image are mapped to a wider range of output levels..

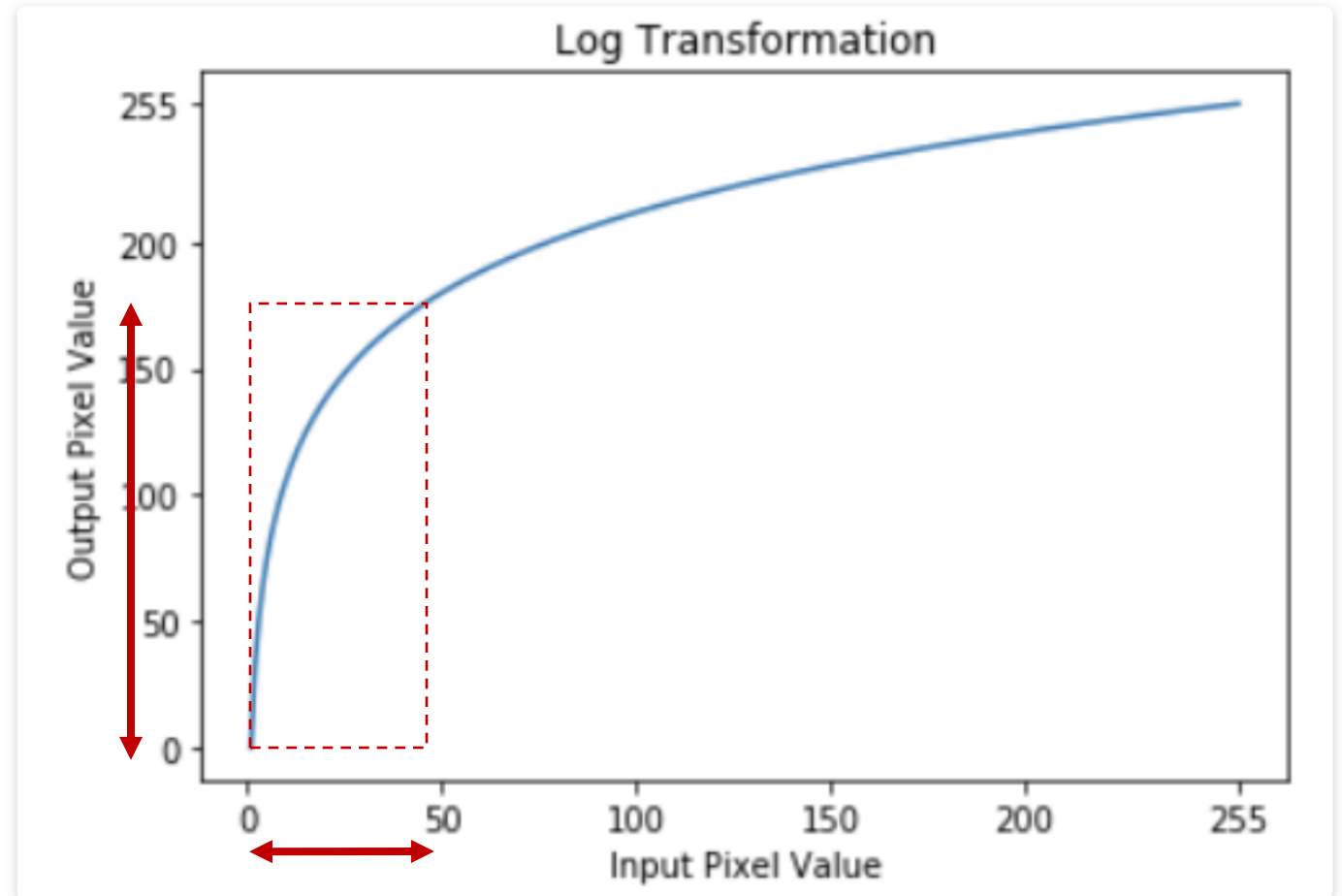


Image enhancement: Intensity transformation

- **Log Transformations: Applications**

- Expands the dark pixels in the image while compressing the brighter pixels
- Compresses the dynamic range (display of Fourier transform).

Recall that the dynamic range refers to the ratio of max and min intensity values.

When the dynamic range of the image is greater than that of displaying device(like in Fourier transform) => The lower values are suppressed.

Solution: The log transformation

In other terms, a logarithmic transform is appropriate when we want to enhance the low pixel values at the expense of loss of information in the high pixel values.

Image enhancement: Intensity transformation

- Log Transformations

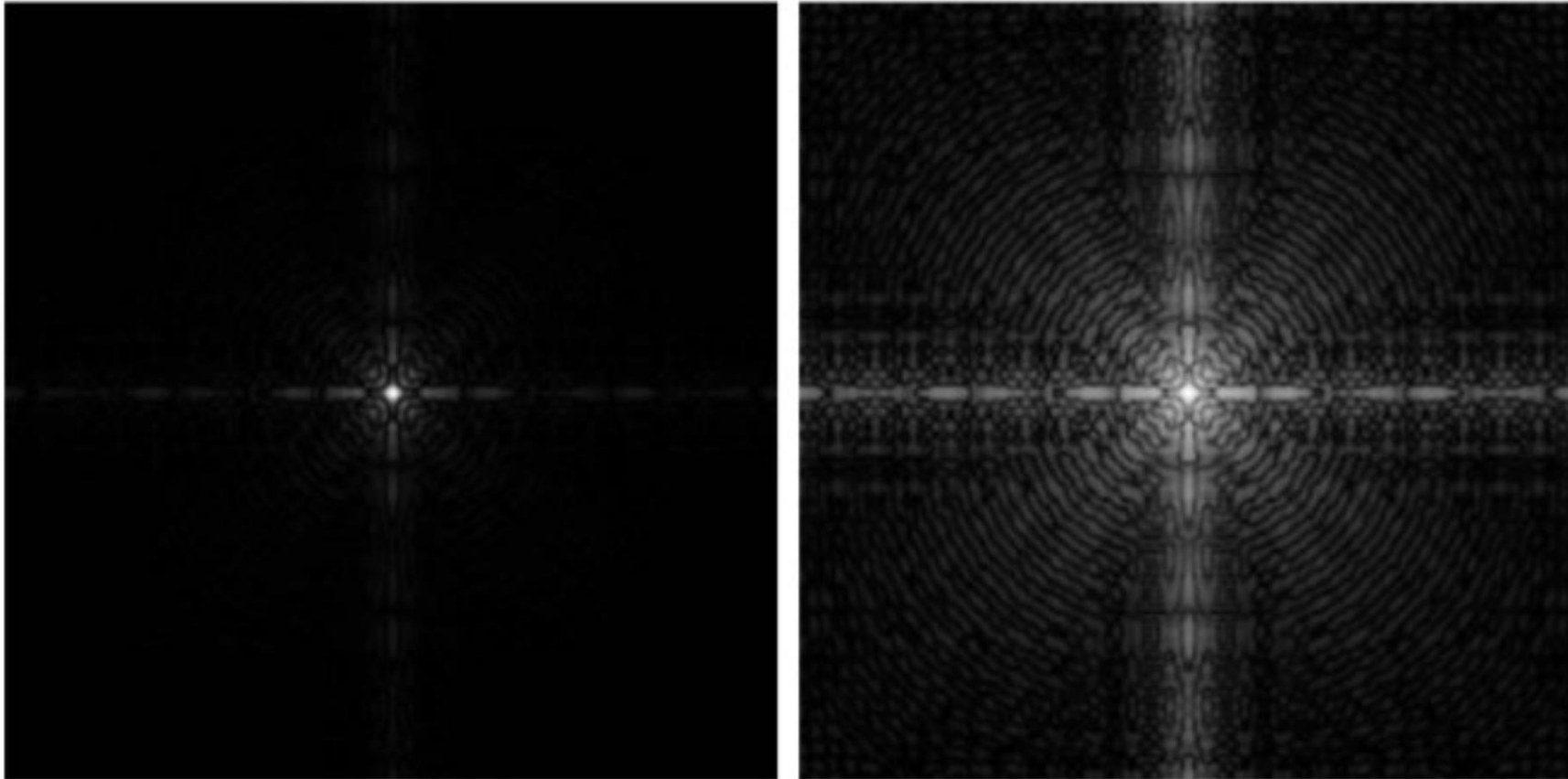


Image enhancement: Intensity transformation

- **Log Transformations: Applications**

But be careful if most of the details are present in the high pixel values...



Before



After

Image enhancement: Intensity transformation

- **Gamma (or power law) transformations**

They have the following basic form: $s = c \cdot r^\gamma$

where c and γ are positive constants.

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

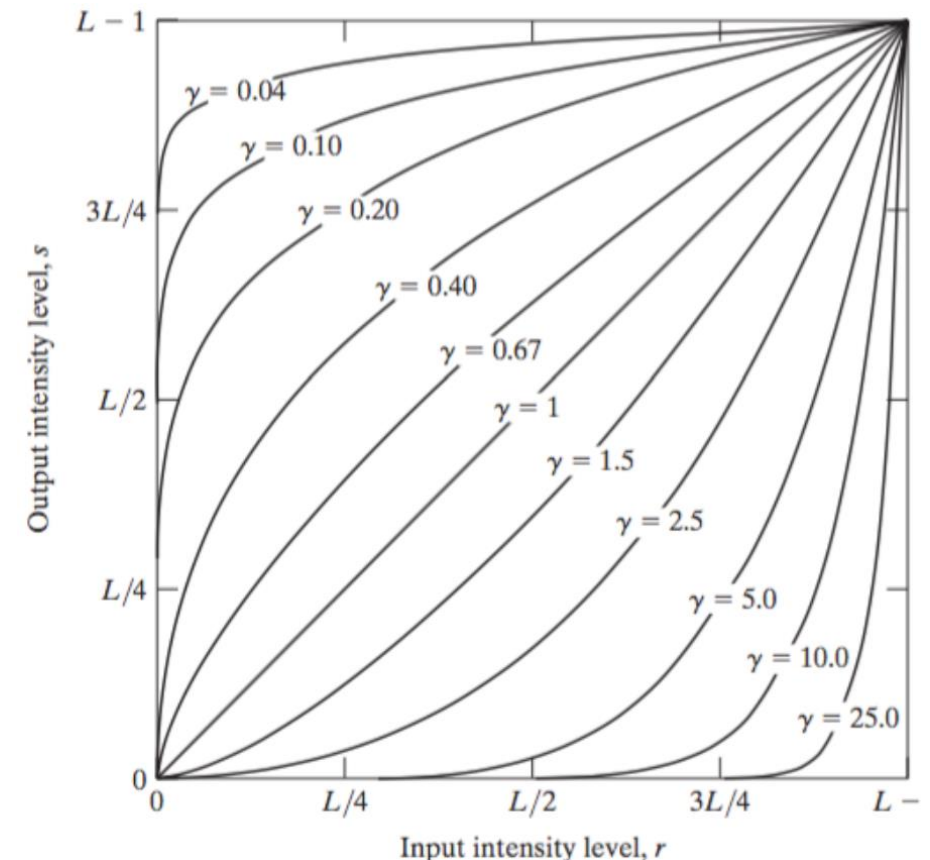


Image enhancement: Intensity transformation

- **Gamma (or power law) transformations**

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

- For example, old CRT monitors or modern projectors 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

Image enhancement: Intensity transformation

- **Gamma (or power law) transformations**

For example, old CRT monitors or modern projectors have an intensity-to-voltage response which is a power-law with exponents varying from 1.8 to 2.5

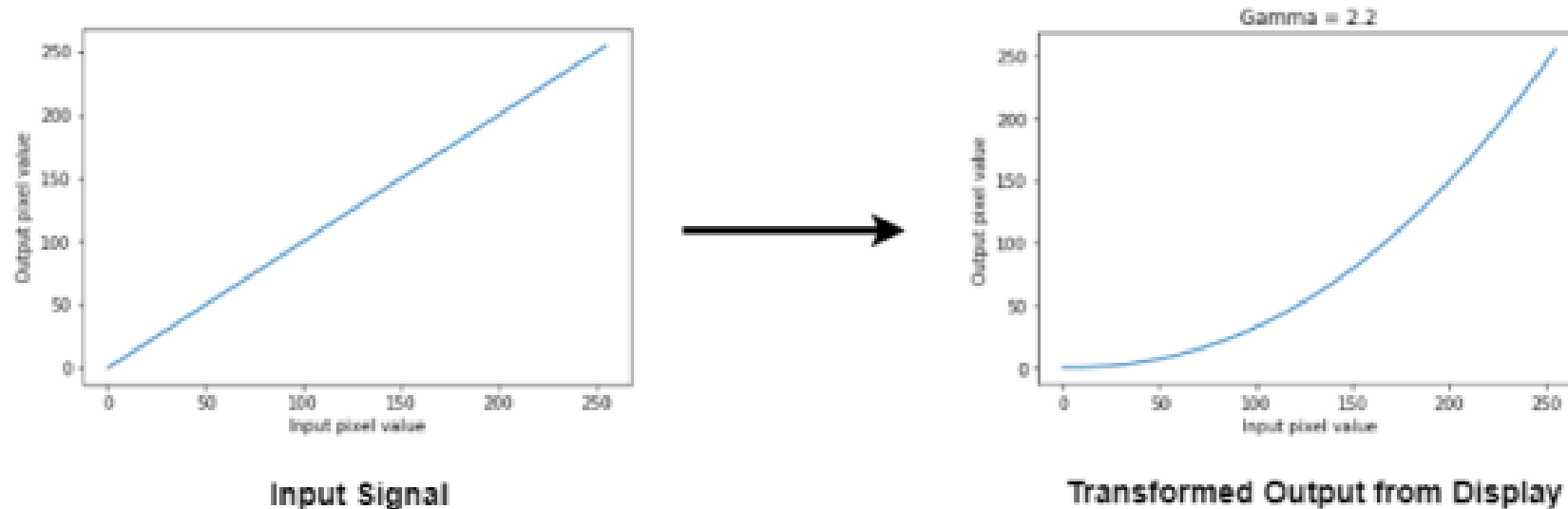


Image enhancement: Intensity transformation

- **Gamma (or power law) transformations**

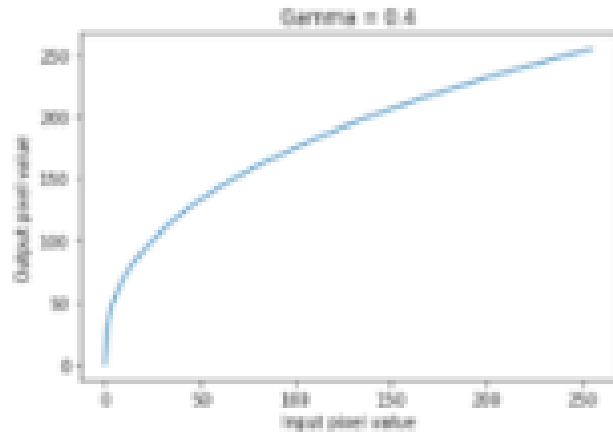
To correct this, we apply gamma correction to the input signal (we know the intensity and voltage relationship we simply take the complement) which is known as Image Gamma.

This gamma is automatically applied by the conversion algorithms like jpeg etc. thus the image looks normal to us.

This input cancels out the effects generated by the display and we see the image as it is. The whole procedure can be summed up as by the following figure

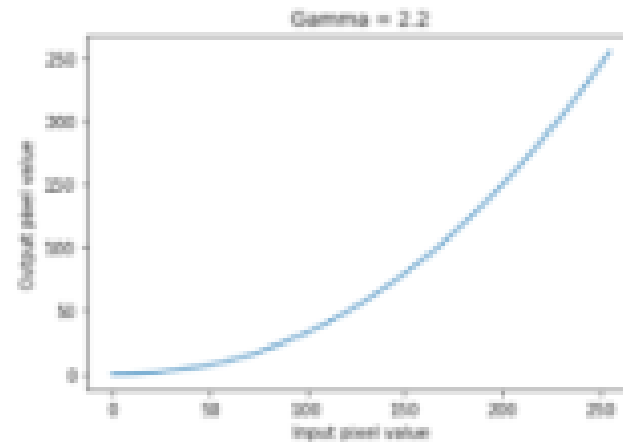
Image enhancement: Intensity transformation

- Gamma (or power law) transformations



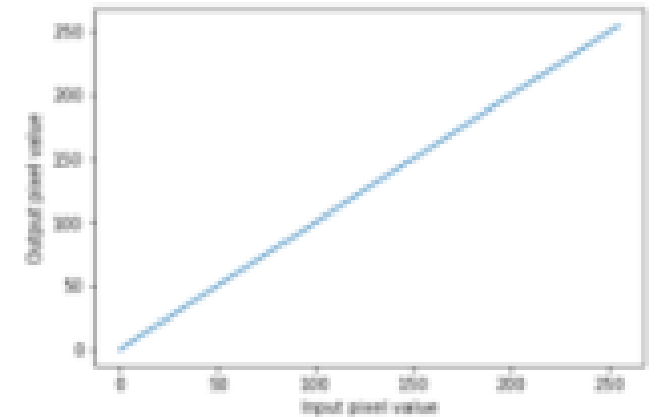
**gamma corrected
input signal**

+



Transformed Output from Display

=



Final Output

Image enhancement: Intensity transformation

- Gamma (or power law) transformations



Original Image



Image enhancement: Intensity transformation

- Gamma (or power law) transformations

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

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

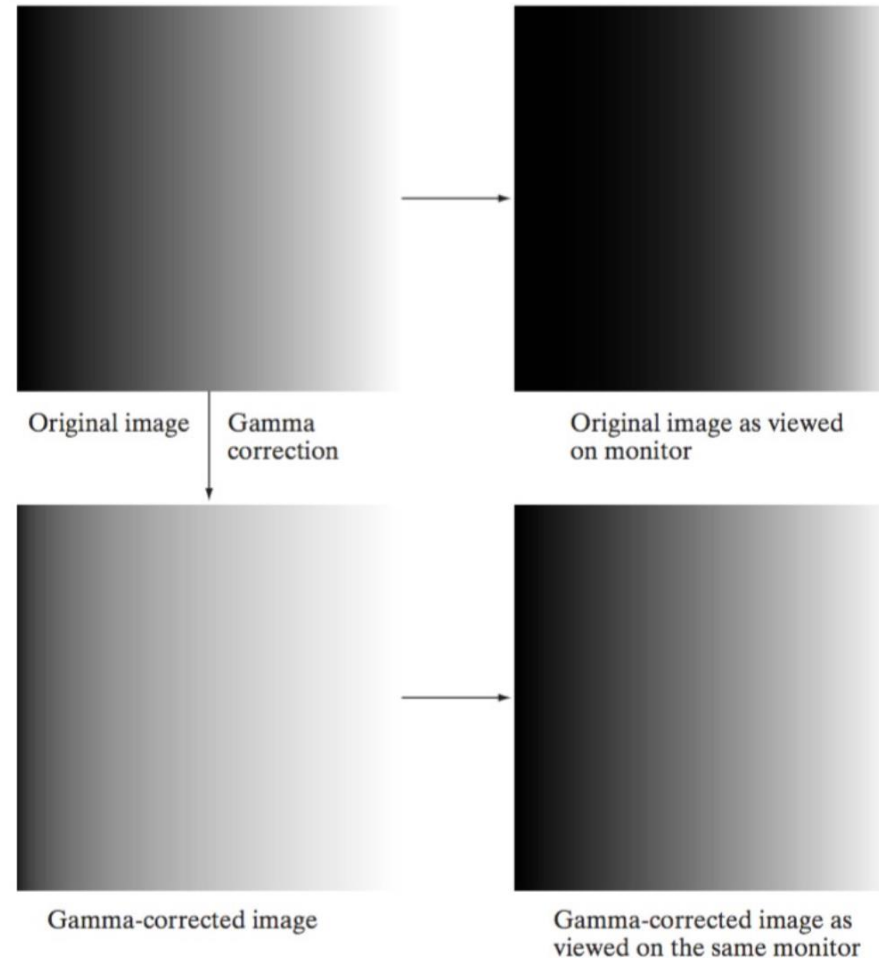


Image enhancement: Intensity transformation

- Gamma (or power law) transformations

=> *Also useful for contrast manipulation.*

Image is
washed out



Corrected with
 $\gamma = 3$



Corrected with
 $\gamma = 4$



Corrected with
 $\gamma = 5$



Image enhancement: Intensity transformation

- **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 (with a sigmoid shape), also called **Contrast-Stretching**.

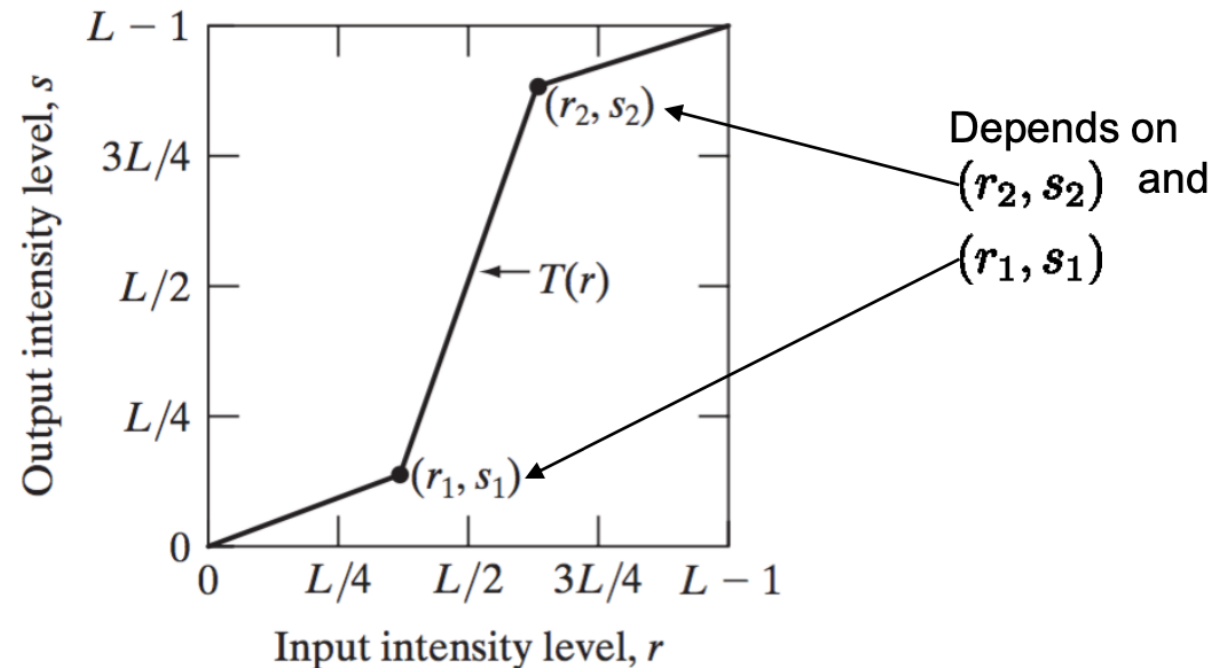


Image enhancement: Intensity transformation

- **Contrast Enhancement**

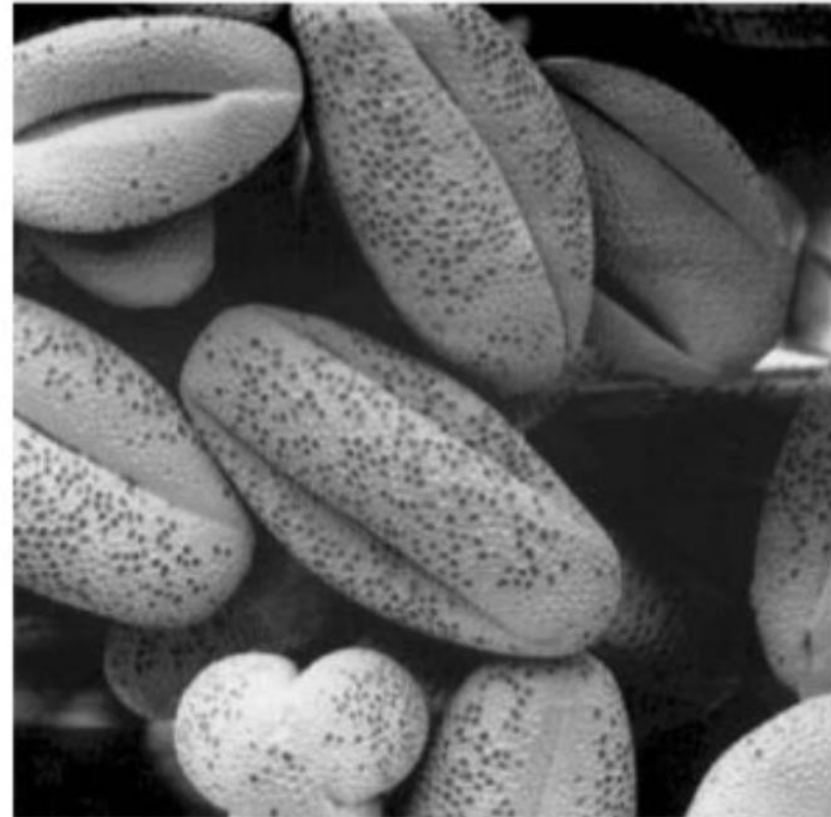
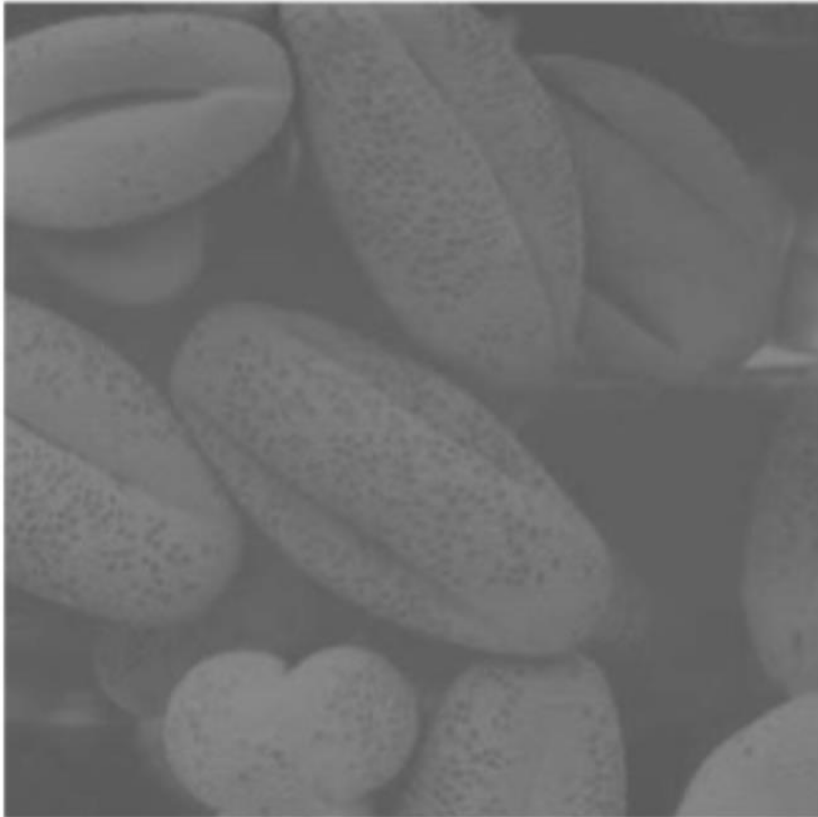
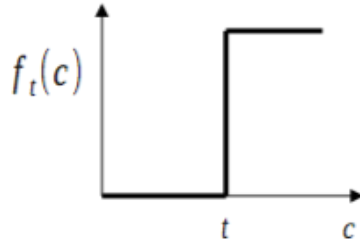


Image enhancement: Intensity transformation

- **Thresholding**

An extreme case of contrast enhancement is:

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


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 enhancement: Intensity transformation

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

=> How can we automatically determine the best values of these parameters?

- One effective tool is the **Image Histogram**
- It 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 enhancement: Intensity transformation

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

=> How can we automatically determine the best values of these parameters?

- One effective tool is the **Image Histogram**
- It 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 enhancement: Intensity transformation

- **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^{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

⇒ each histogram component is an estimate of the probability of the occurrence of the intensity r_k

Image enhancement: Intensity transformation

- Image Histogram $h(r_k) = n_k$

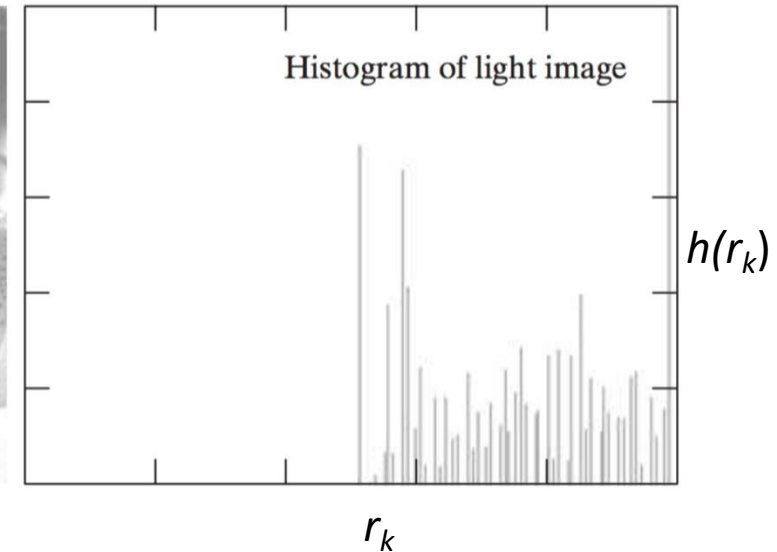
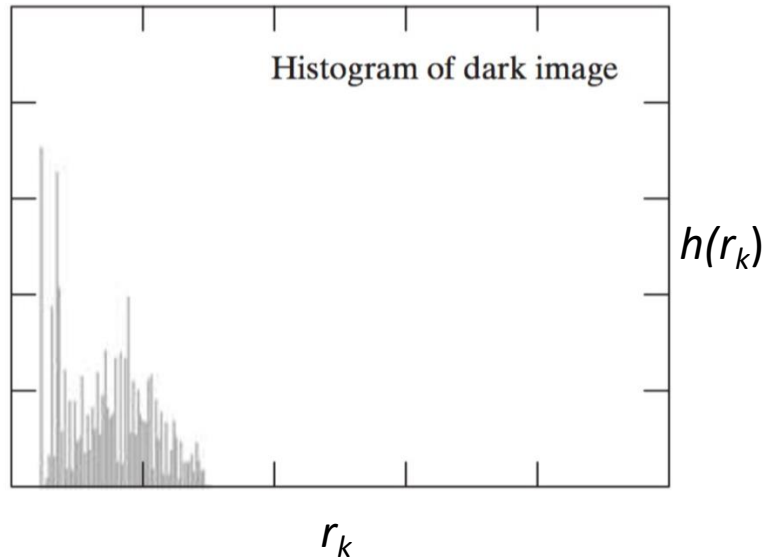
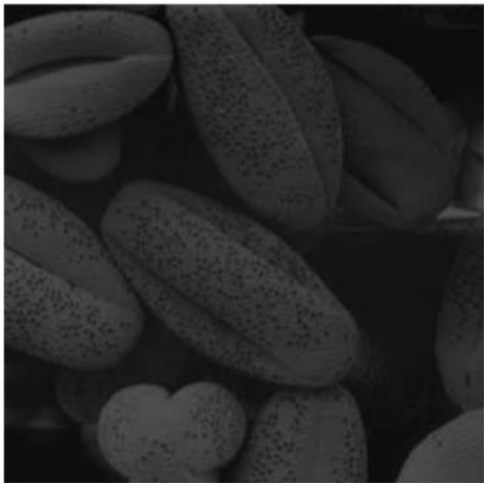


Image enhancement: Intensity transformation

- Image Histogram $h(r_k) = n_k$

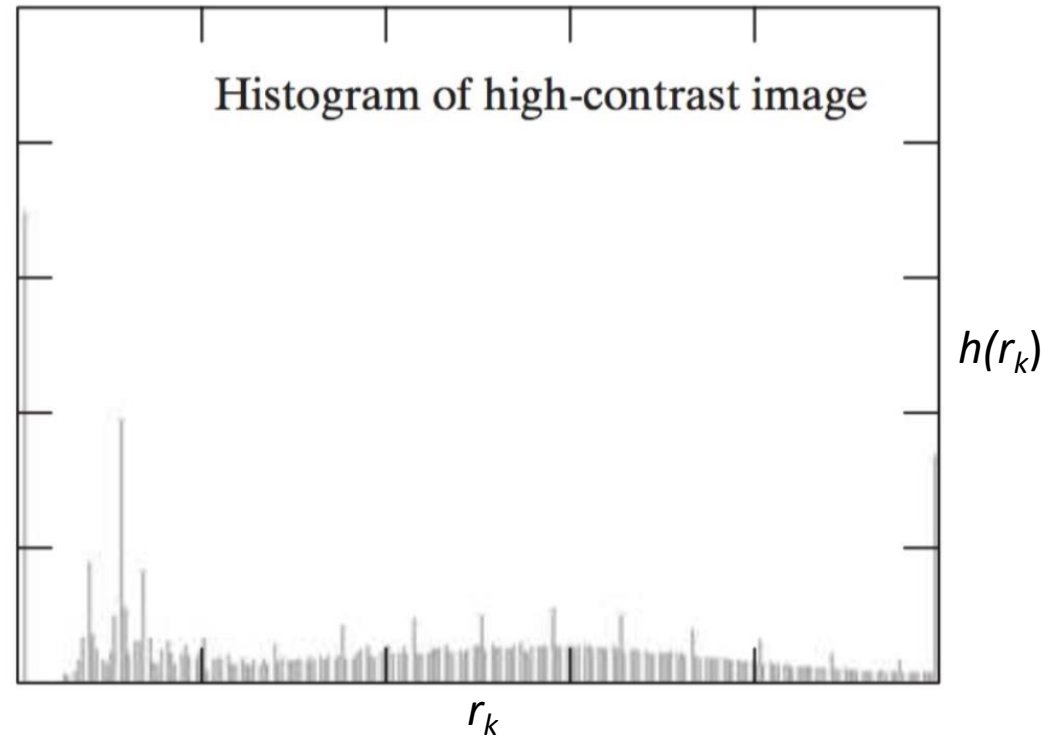
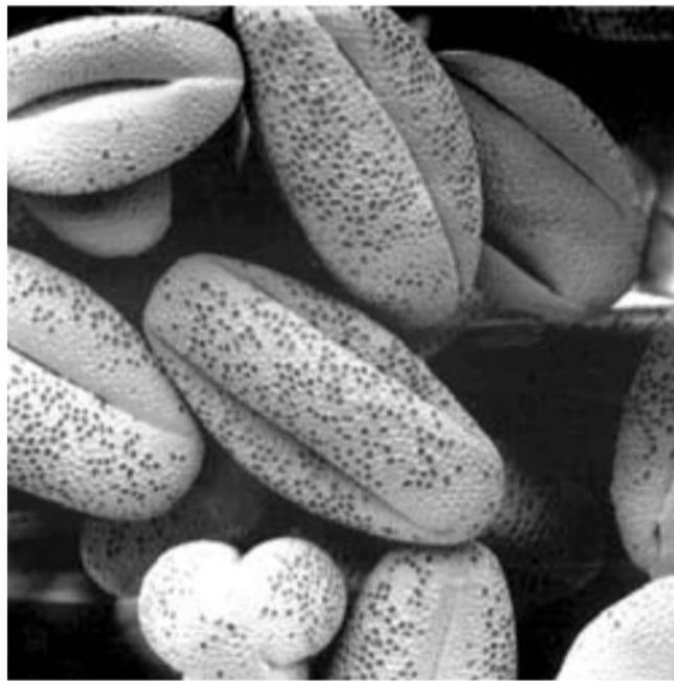
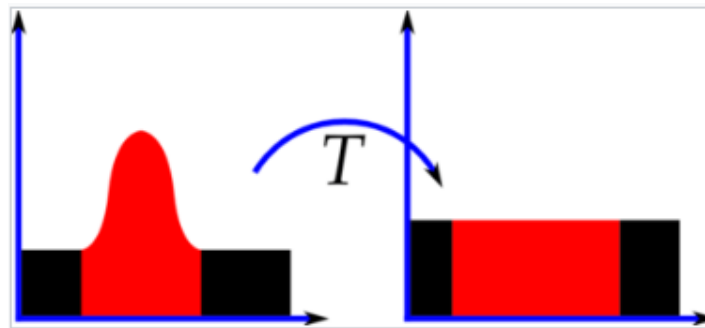


Image enhancement: Intensity transformation

Histogram equalization

As the name suggests, stretches the histogram to fill the dynamic range and at the same time tries to keep the histogram uniform.



$$s = T(r)$$

⇒ When enhancing an image, ideally we would like to maximize the dynamic range of the image to catch both the dark and bright details.

Image enhancement: Intensity transformation

Histogram equalization

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:

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

(T must be continuous, differentiable and strictly monotonic)

Image enhancement: Intensity transformation

Histogram equalization

Cumulative distrib. Function of r

The used function (T) is the following:

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

where L is the maximum intensity value (for n bit image, $L = 2^n$)

$$\Rightarrow p_s(s) = \frac{1}{L - 1}$$

Uniform distribution

$$\text{In the discrete case: } s_k = T(r_k) = (L - 1) \sum_{j=0}^k p_r(r_j) = \frac{L - 1}{MN} \sum_{j=0}^k h(r_j)$$

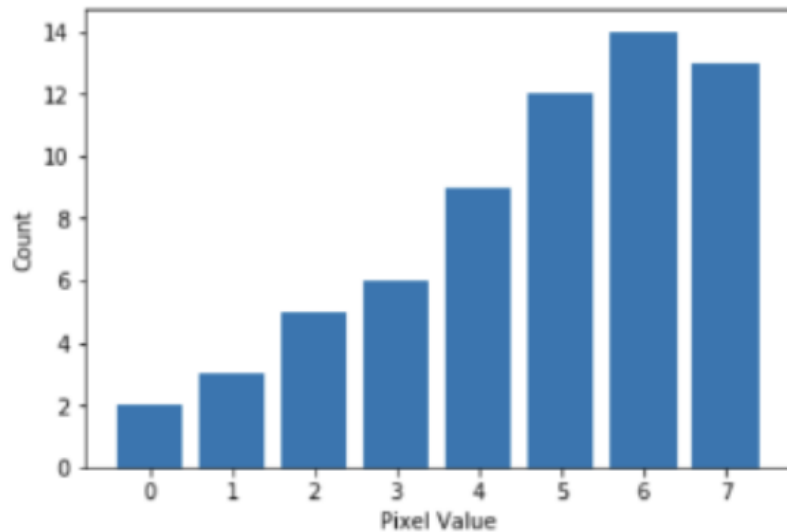
Sum of the first k components of the input image histogram

\Rightarrow We obtain a **(quasi)** flat histogram of the output image.

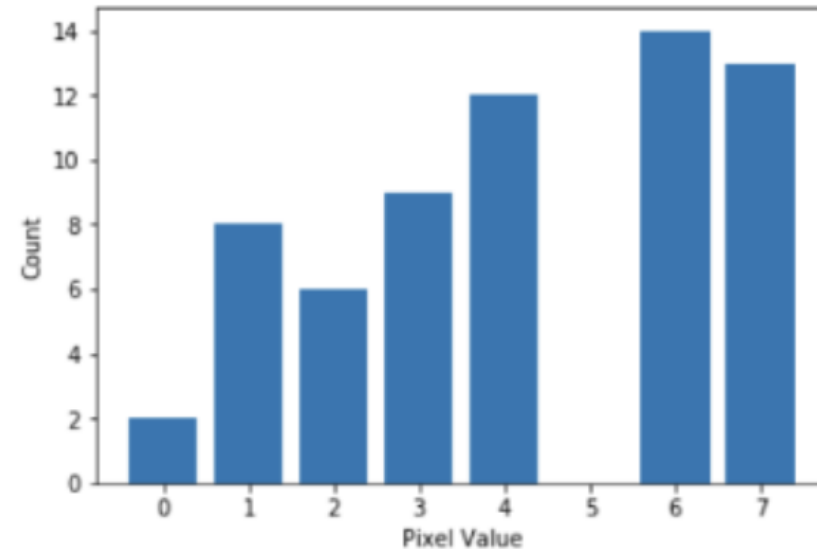
Image enhancement: Intensity transformation

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!



Original



Equalized

Image enhancement: Intensity transformation

Histogram equalization

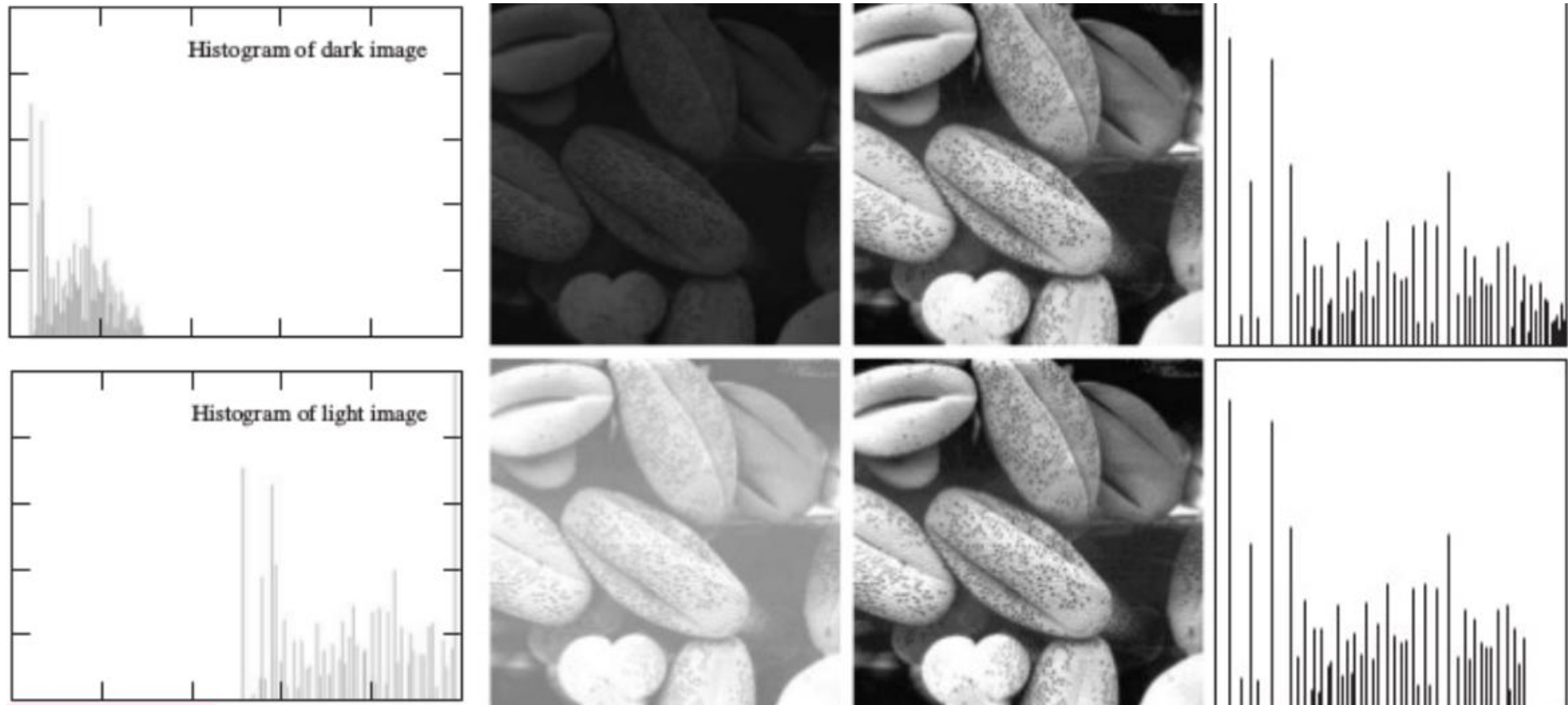


Image enhancement: Intensity transformation

Histogram equalization

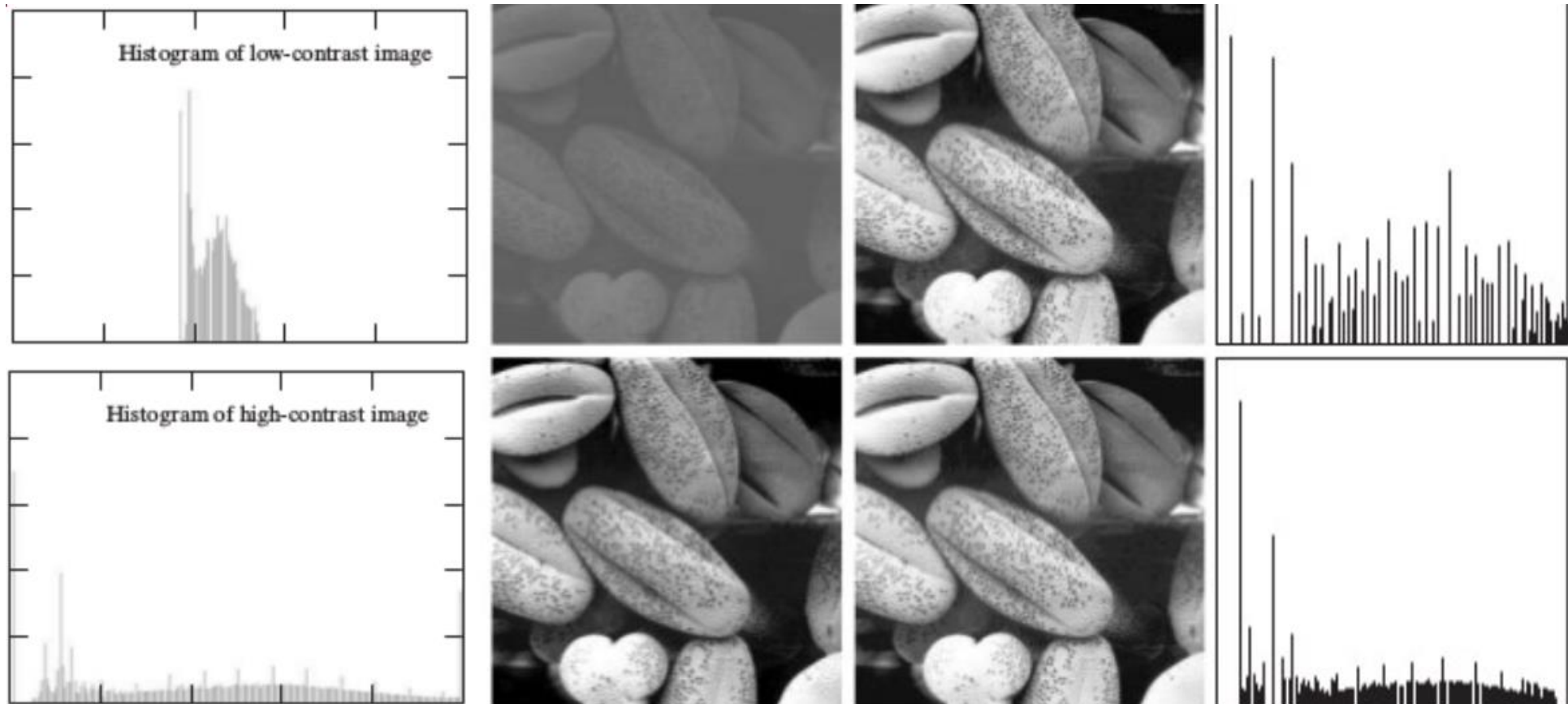
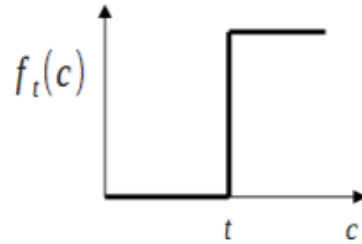


Image enhancement: Intensity transformation

Histogram for thresholding

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



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



Image enhancement: Intensity transformation

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.

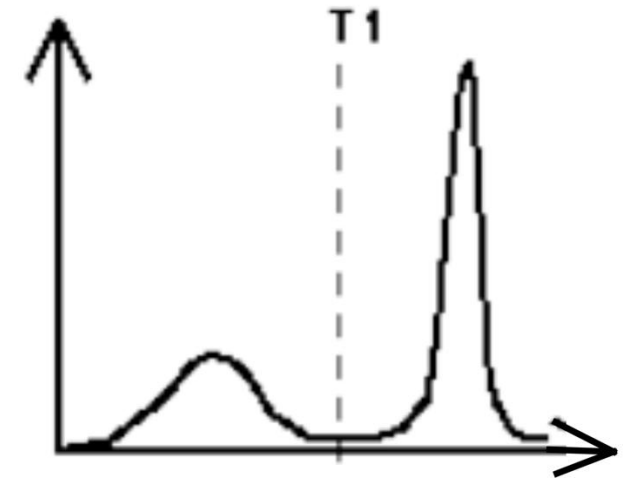
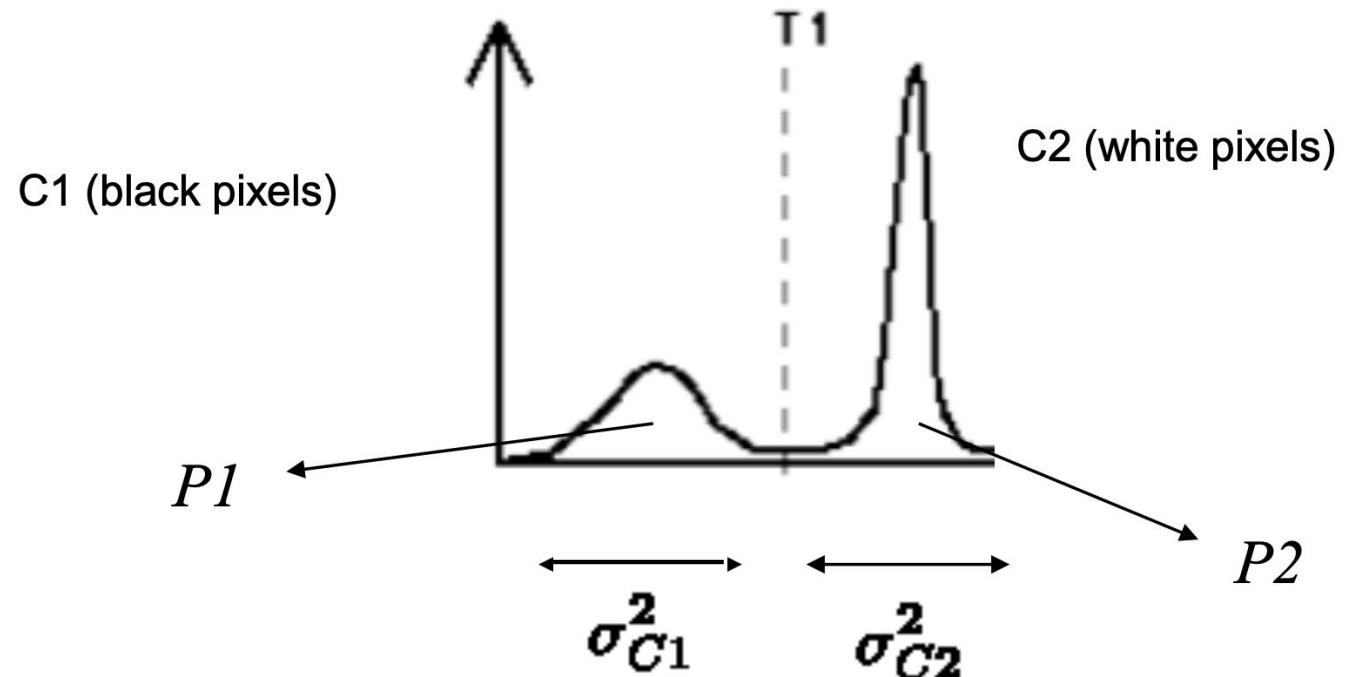


Image enhancement: Intensity transformation

Otsu Thresholding

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

$$\operatorname{argmin}_T P_1 \sigma_{C1}^2 + P_2 \sigma_{C2}^2$$



Spatial filtering

It is a neighborhood operator

It uses a collection of pixel values in the vicinity of a given pixel to determine its final output value.

In a linear spatial filter, the output pixel value is a weighted sum of pixel values within a neighborhood N .

Each set of weights => define a new filter

It is a **spatial filtering**.

The underlying mechanism is the ***convolution***.

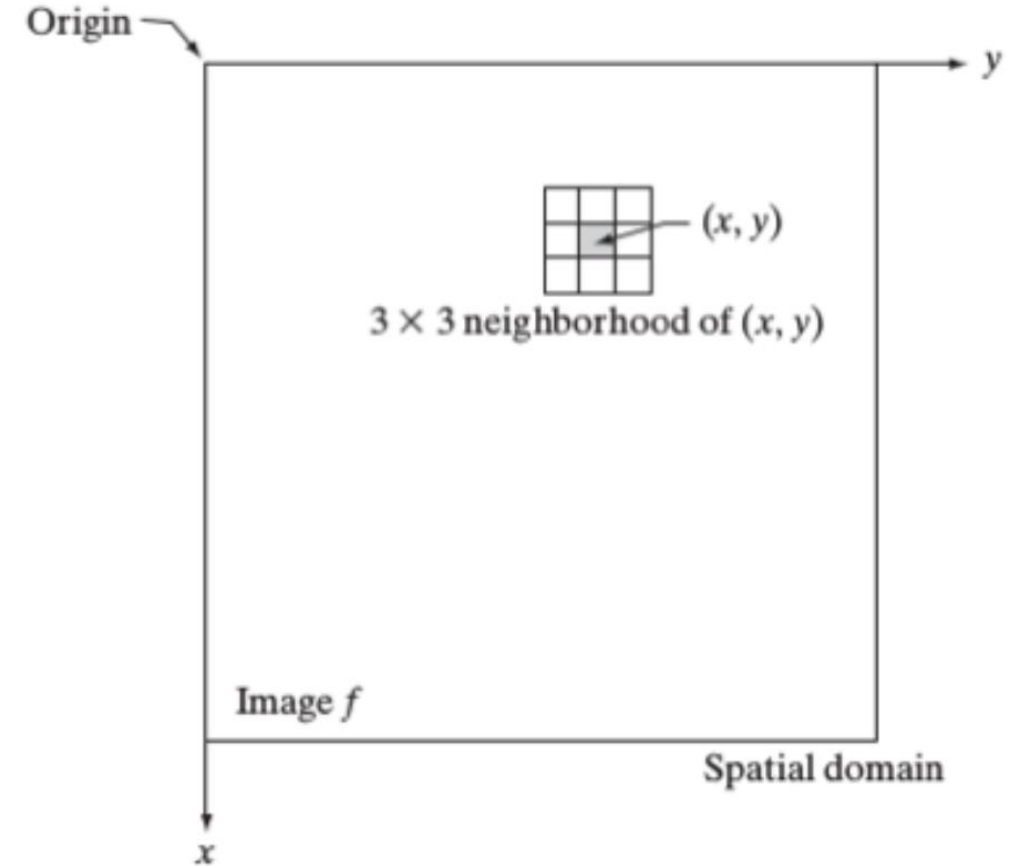
Spatial filtering

It creates a new pixel with coordinates equal to the center of the neighborhood and whose value is the result of the filtering operation.

Filter is defined in terms of a coefficient matrix W

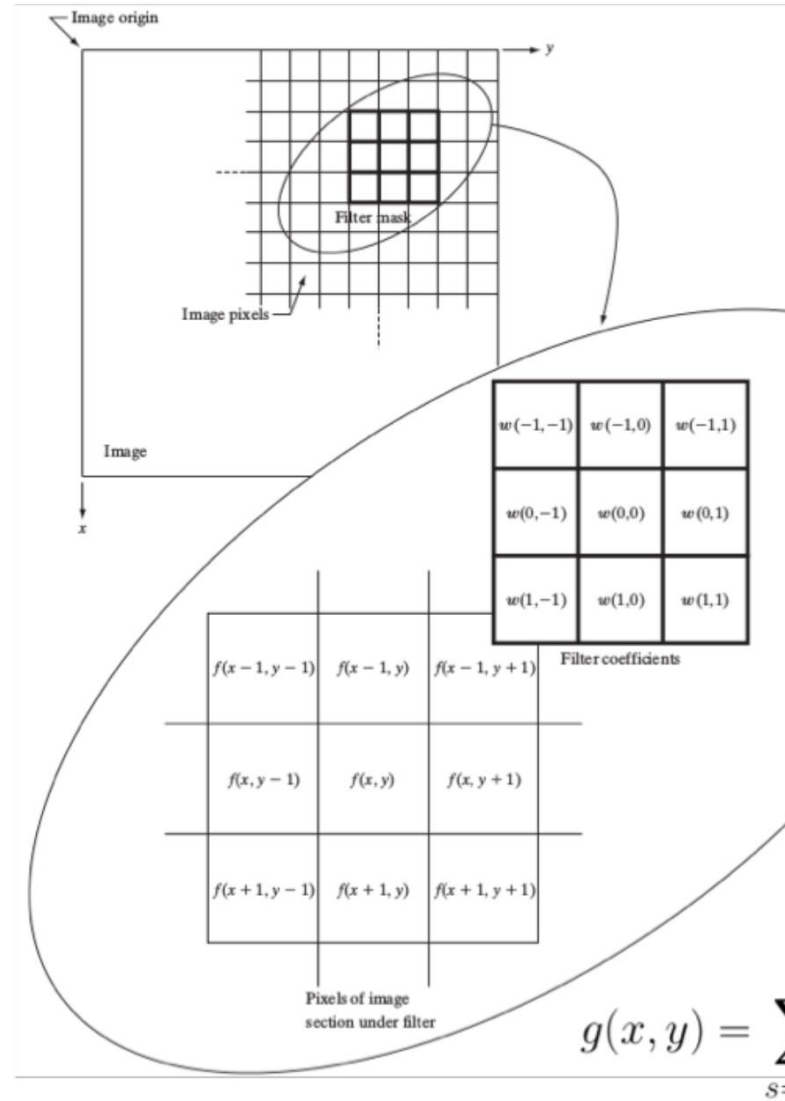
The performed operation is the sum of products of the filter coefficients and the image pixels encompassed by the filter.

$$g(x, y) = \sum_{s=-a}^a \sum_{t=-b}^b w(s, t) f(x + s, y + t)$$



Spatial filtering

- When applying a linear filter, the same set of weights is applied on the whole image.
- Different sets of weights => different processes.



Example of a filter acting on a 3x3 neighborhood:

$$g(x, y) = w(-1, -1)f(x-1, y-1) + \\ + w(-1, 0)f(x-1, y) + \dots \\ \dots \\ + w(1, 1)f(x+1, y+1)$$

$$g(x, y) = \sum_{s=-a}^a \sum_{t=-b}^b w(s, t) f(x+s, y+t)$$

Spatial filtering

An example: The average filter

A linear filter that averages all pixels within $2k+1 \times 2k+1$ neighborhood:

$$R_{ij} = \frac{1}{(2k+1)^2} \sum_{u=i-k}^{i+k} \sum_{v=j-k}^{j+k} f_{uv}$$

The filter (kernel) is a matrix of size $2k+1 \times 2k+1$ where all the components of the filters are equal to 1.

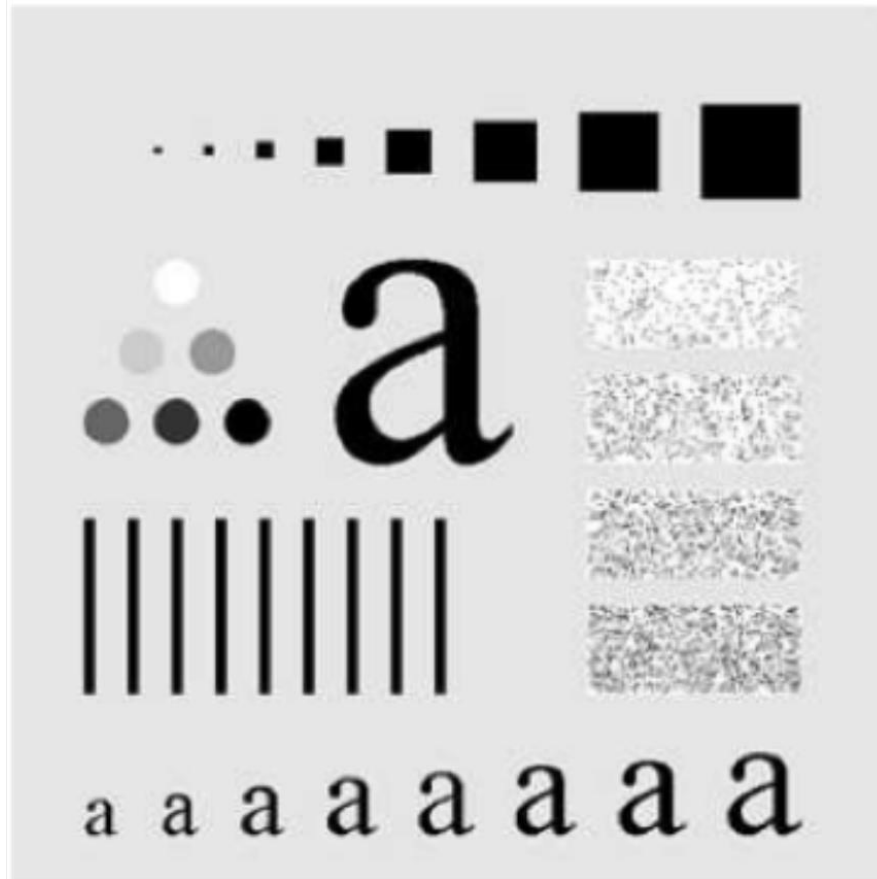
 $\frac{1}{9} \times$

1	1	1
1	1	1
1	1	1

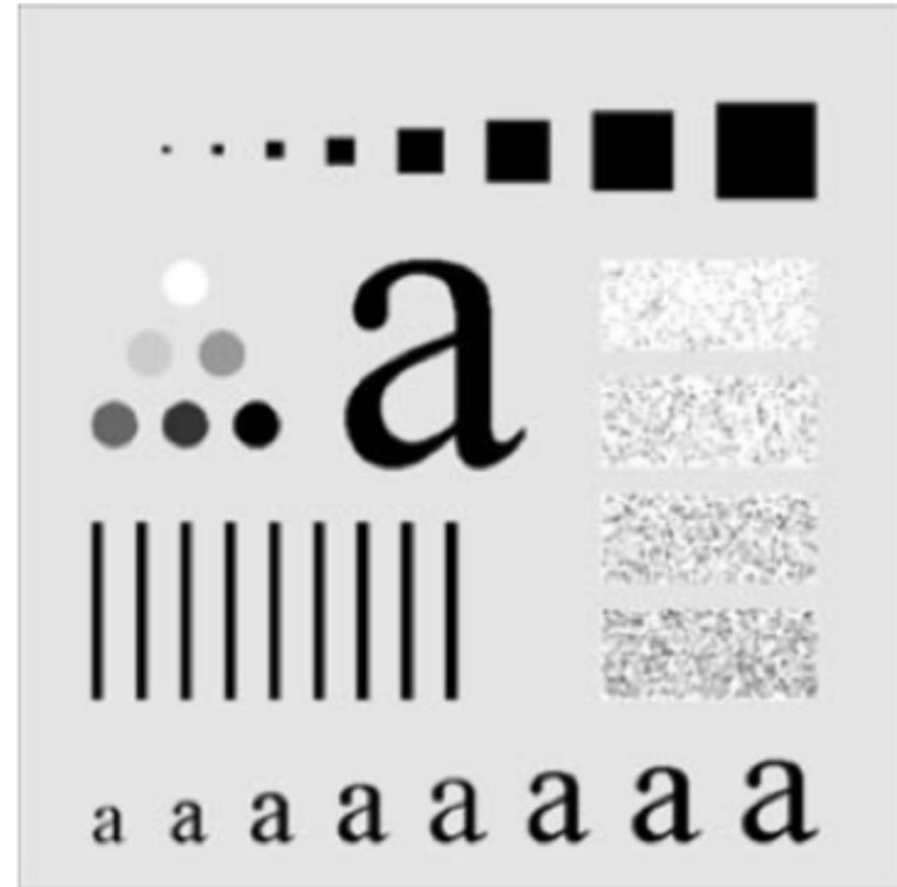
Spatial filtering

An example: The average filter

Original image



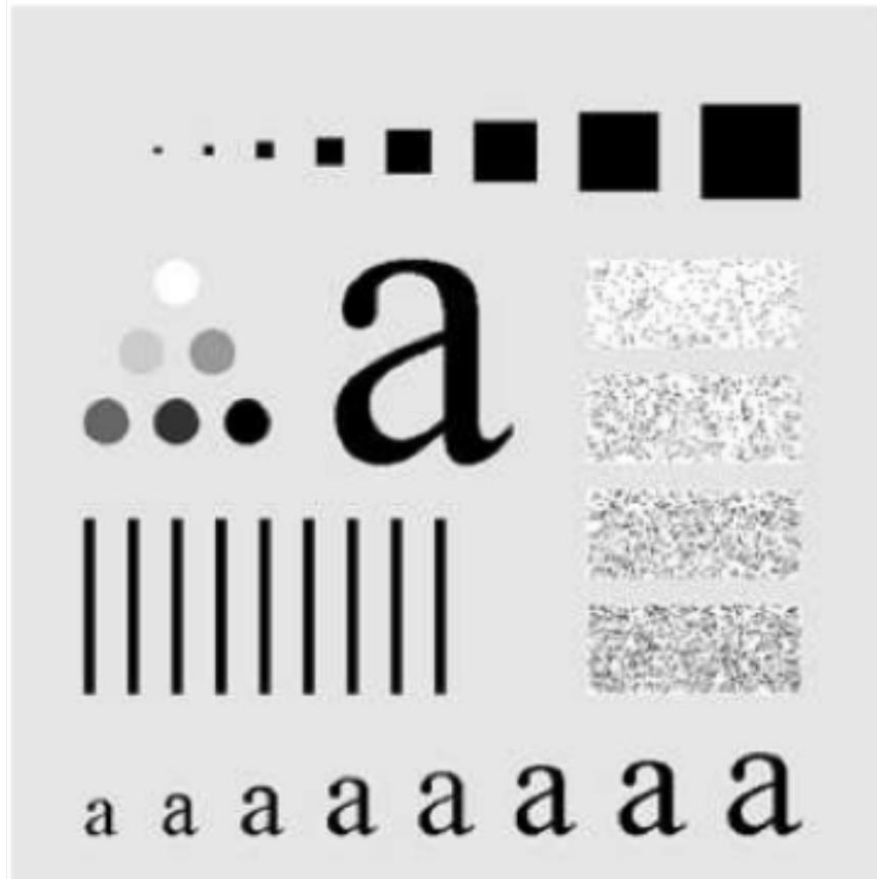
3x3 average filter



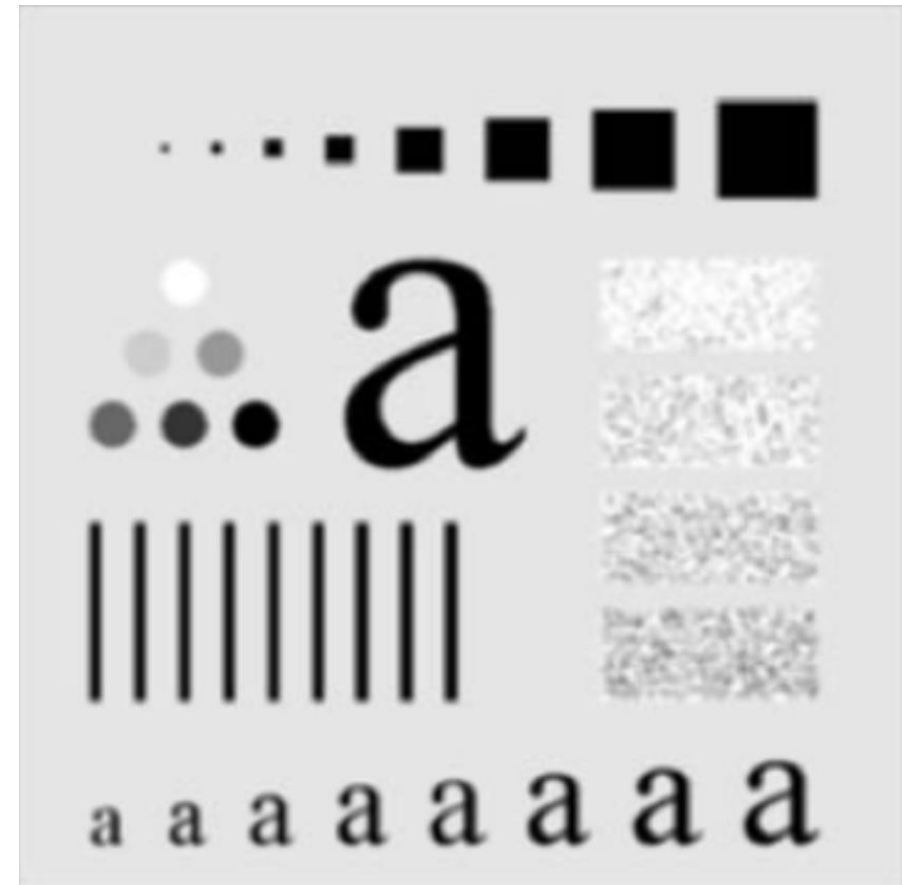
Spatial filtering

An example: The average filter

Original image



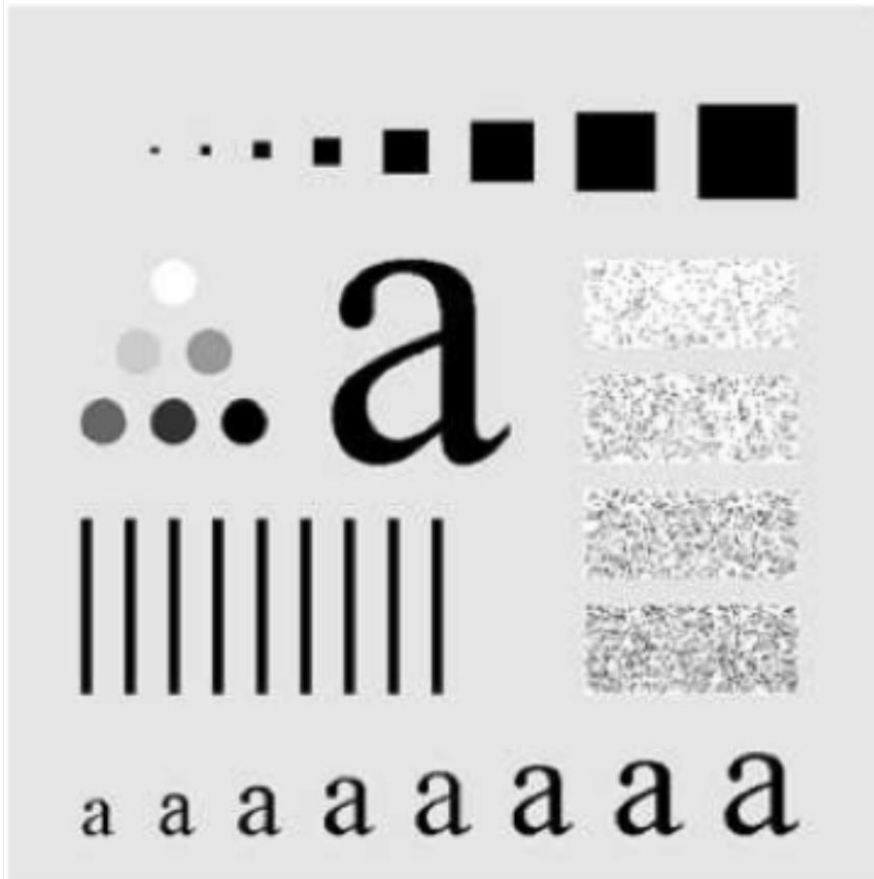
5x5 average filter



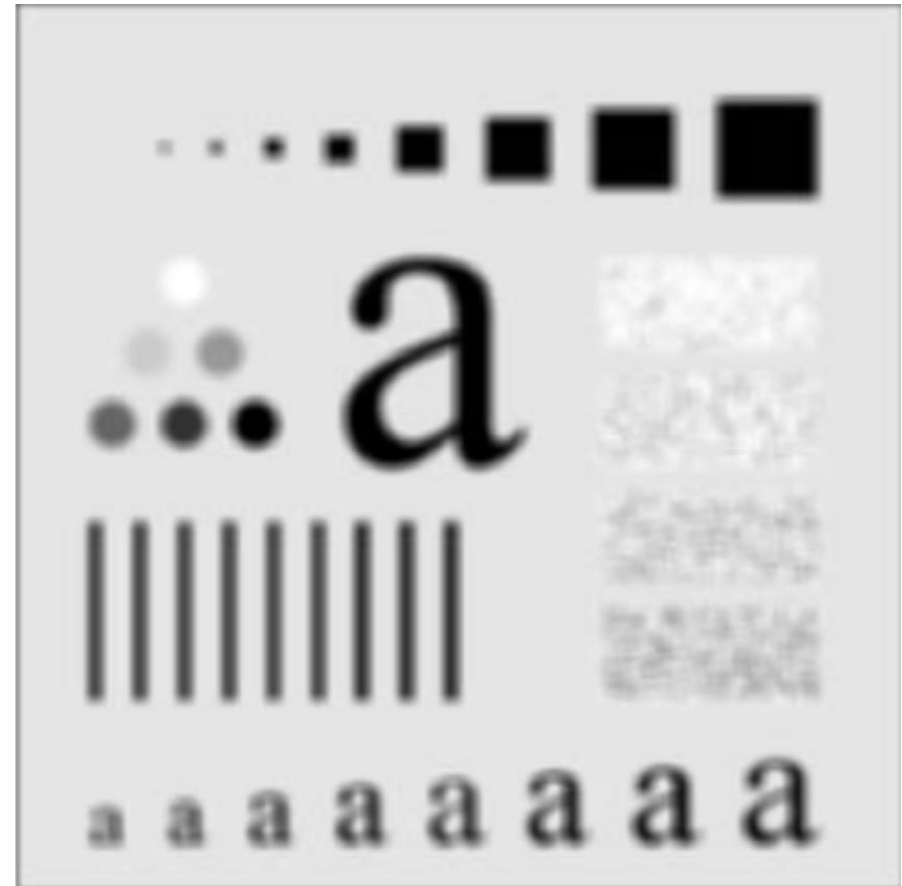
Spatial filtering

An example: The average filter

Original image



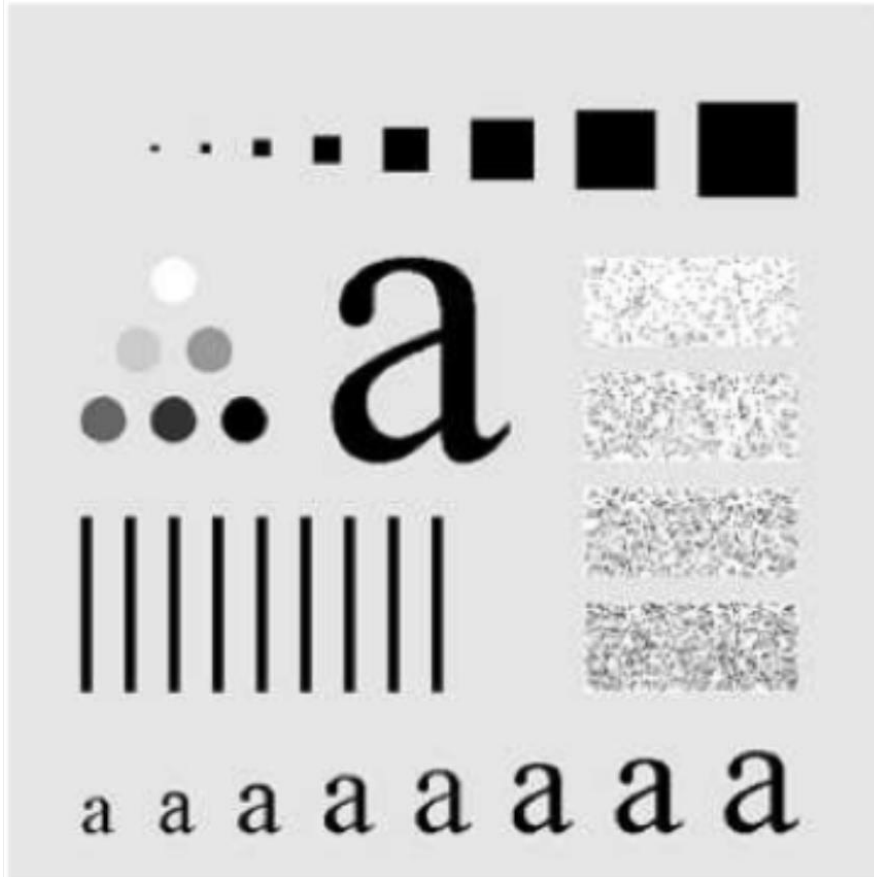
9x9 average filter



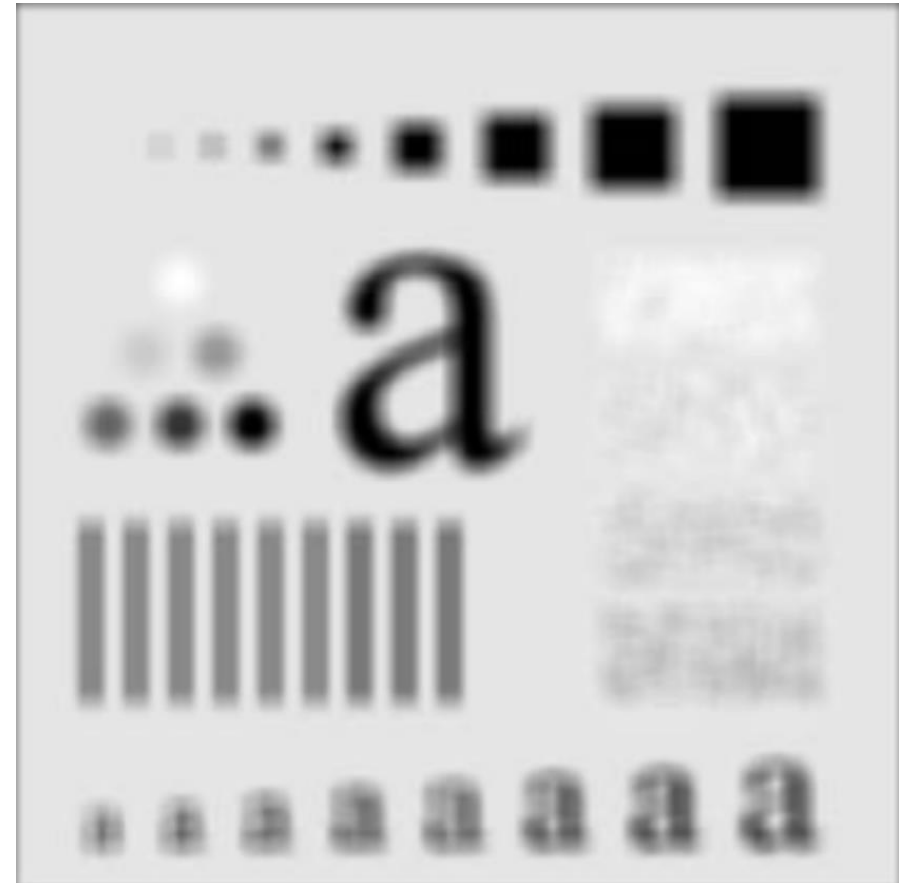
Spatial filtering

An example: The average filter

Original image



15x15 average filter



Spatial filtering

Properties of filters, in general

- **Superposition:** $R(f + g) = R(f) + R(g)$.

The response to the sum of stimuli is the sum of the individual responses.

- **Scaling:** $R(kf) = kR(f)$.

The response to a scaled stimulus is a scaled version of the response to the original stimulus.

- **Linear:** Superposition + scaling.
- **Shift invariance:** The response to a *translated* stimulus is just a translation of the response to the stimulus.

Spatial linear filtering

What it is it used for?

- **Noise Reduction** – Removes unwanted noise while preserving essential image features.
- **Edge Detection** – Enhances edges by applying filters like Sobel, Prewitt, or Laplacian.
- **Image Sharpening** – Enhances fine details and textures by emphasizing high-frequency components.
- **Smoothing/Blurring** – Reduces details or noise using averaging or Gaussian filters.
- **Feature Enhancement** – Enhances specific patterns or structures, such as textures.
- **Embossing and Edge Enhancement** – Creates relief effects by highlighting directional intensity changes.
- **Image Restoration** – Improves image quality by removing artifacts or blurring effects.

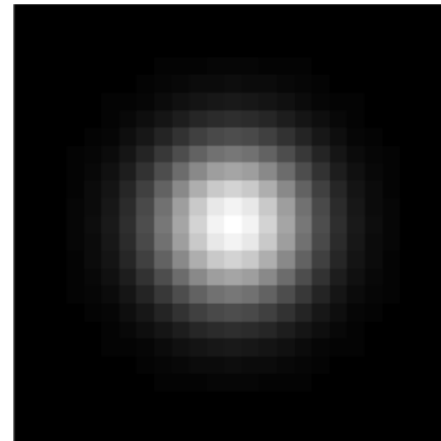
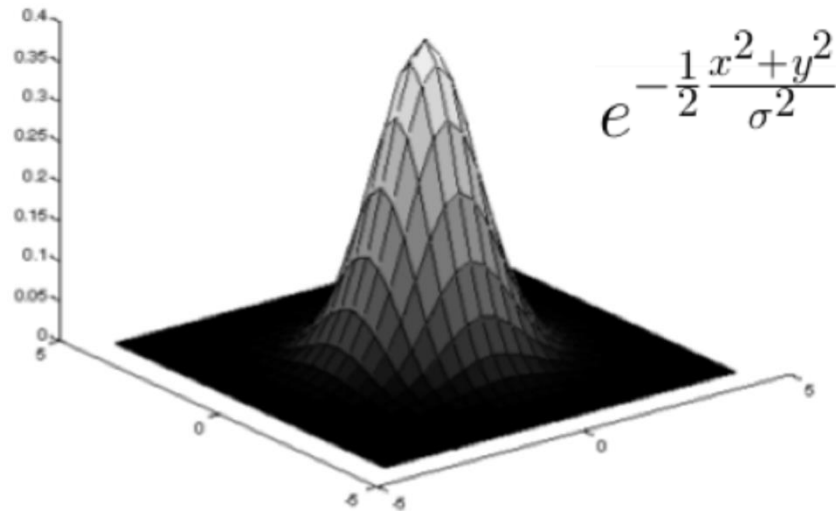
Smoothing filters

Smoothing filters are used for blurring / noise reduction.

The **average filter** is an example.

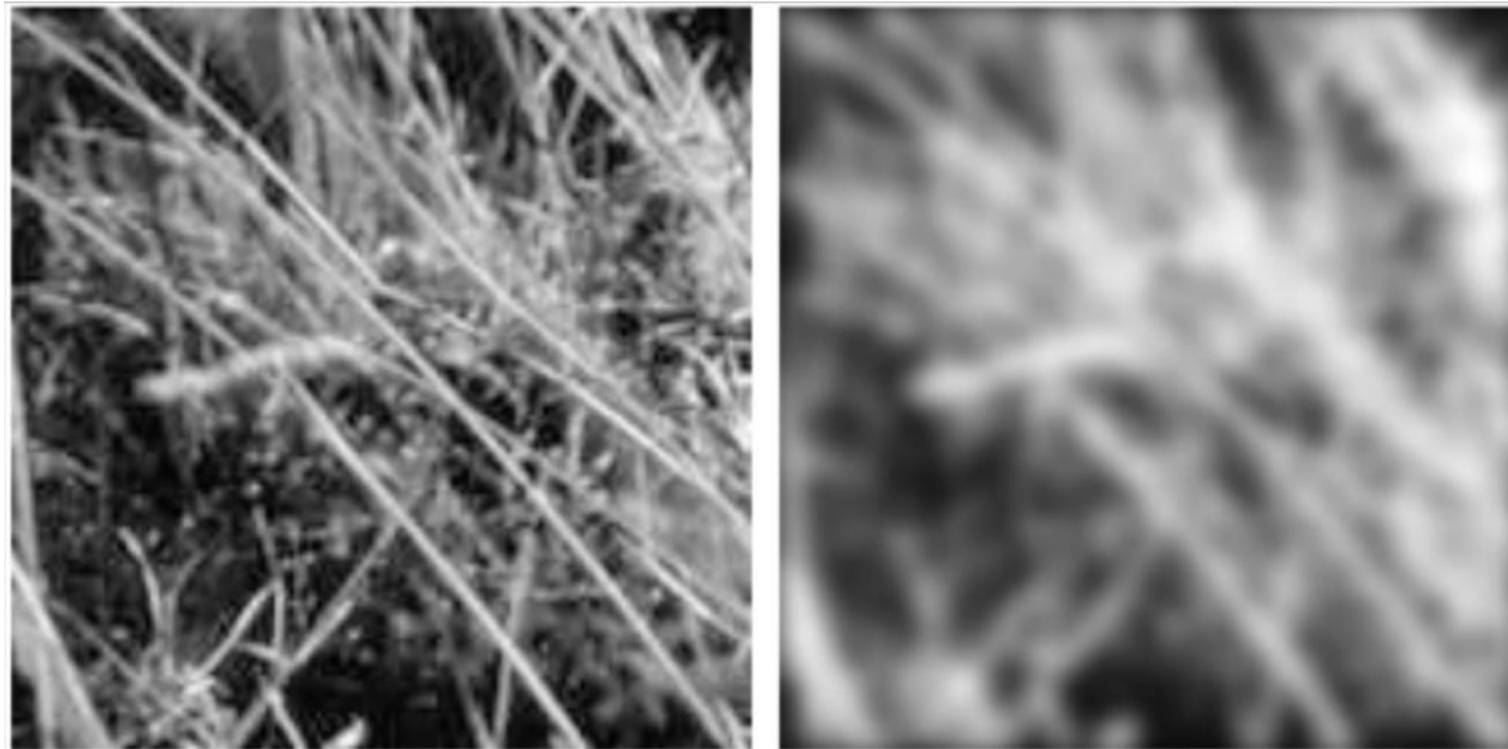
The **Gaussian filter** is another smoothing filter:

- It is a weighted average filter.



Smoothing filters

$$e^{-\frac{1}{2} \frac{x^2+y^2}{\sigma^2}}$$



The **Gaussian filter** => A crucial parameter: σ

Smoothing filters

The **Gaussian filter**:

- If σ is very small, the smoothing will have a very little effect (the weights for all pixels off the center will be very small).
- For a larger σ , the neighboring pixels will have larger weights – The average will be strongly biased toward a consensus of the neighbors.
- A kernel that has a very large σ will cause much of the image detail to disappear, along with the noise.
- A discrete smoothing kernel H is obtained by constructing a $2k+1 \times 2k+1$ array whose i, j^{th} value is:

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

Sharpening filters

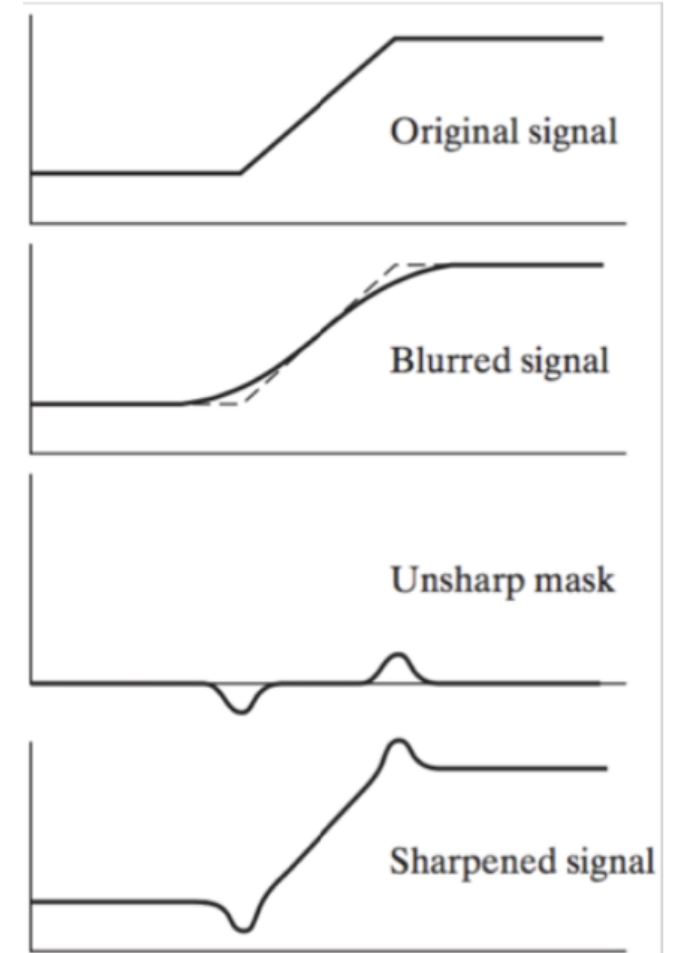
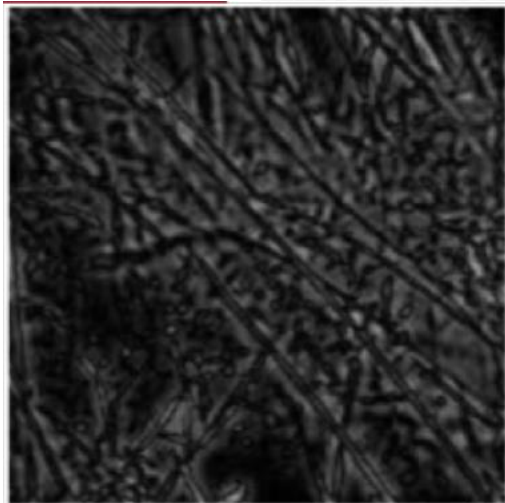
The principal objective of sharpening is to highlight transitions in intensity.

One simple approach is called **unsharp masking** (or high-boost filtering) and consist of the following:

1. Blur the original image
2. Subtract the blurred image from the original to obtain a mask
3. Add the mask to the original (multiplied by a constant for high-boosting)

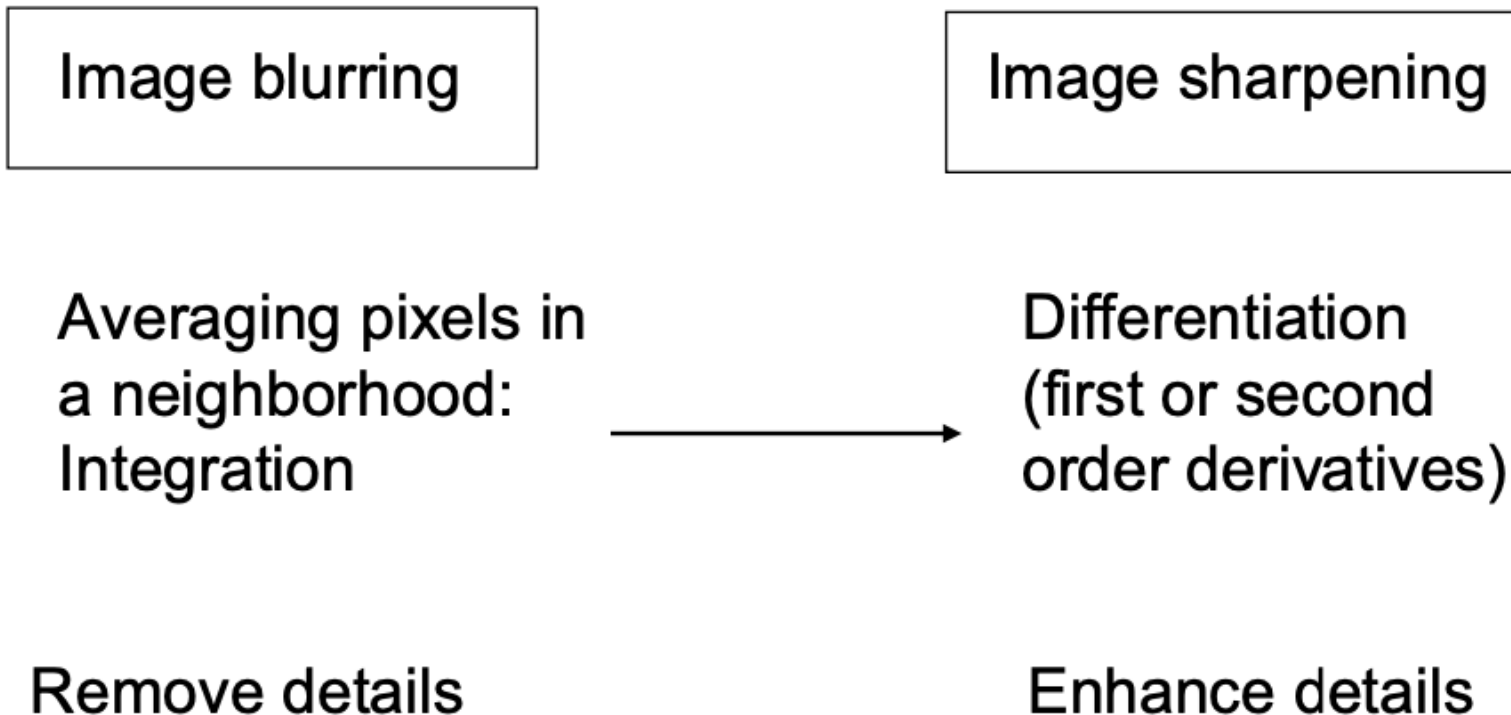
Sharpening filters

Unsharp masking



Sharpening filters

In general, sharpening filter are based on the concept of differentiation.



Sharpening filters

Derivatives of a digital discrete functions are defined in term of differences.

First-order derivative of a one-dimensional function:

$$\frac{df}{dx} = \lim_{\epsilon \rightarrow 0} \frac{f(x + \epsilon) - f(x)}{\epsilon} \approx f(x + 1) - f(x)$$

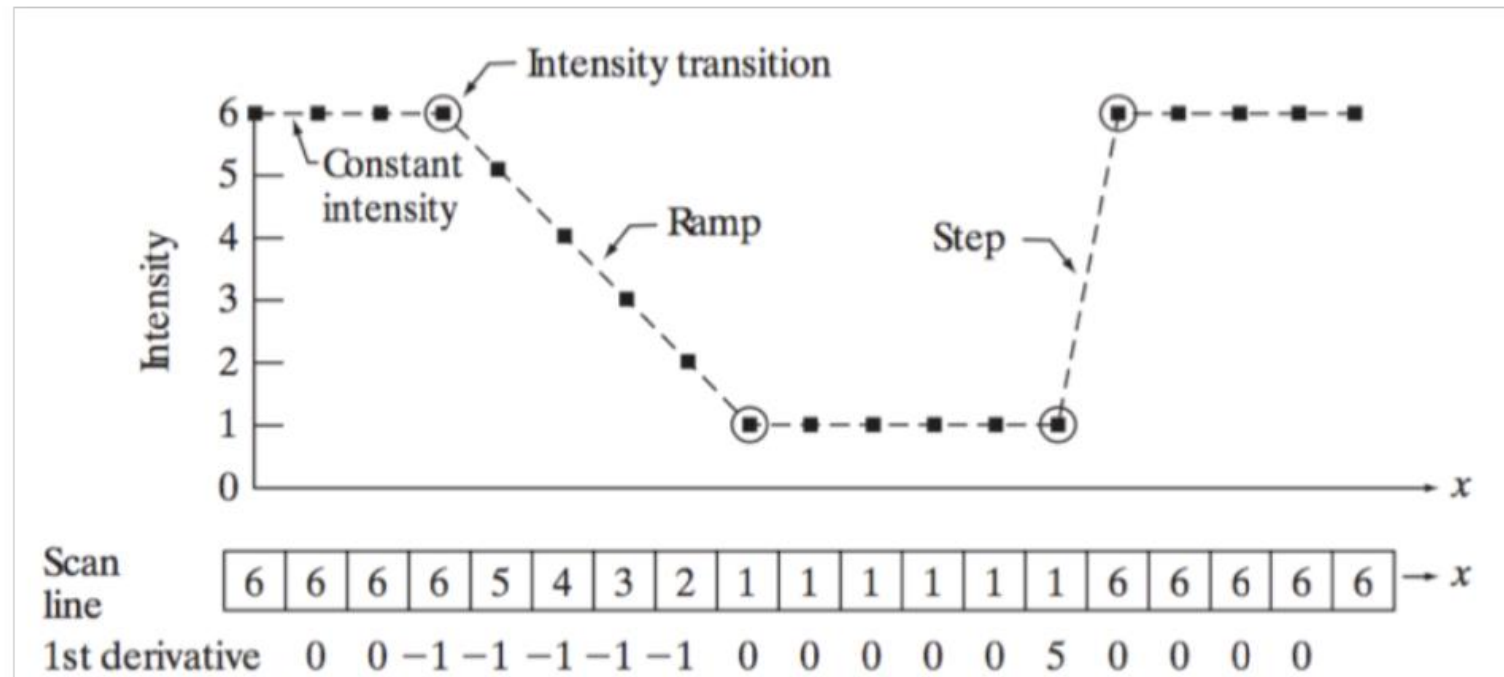
Second-order derivative:

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

Sharpening filters

Derivatives of a digital discrete functions are defined in term of differences.

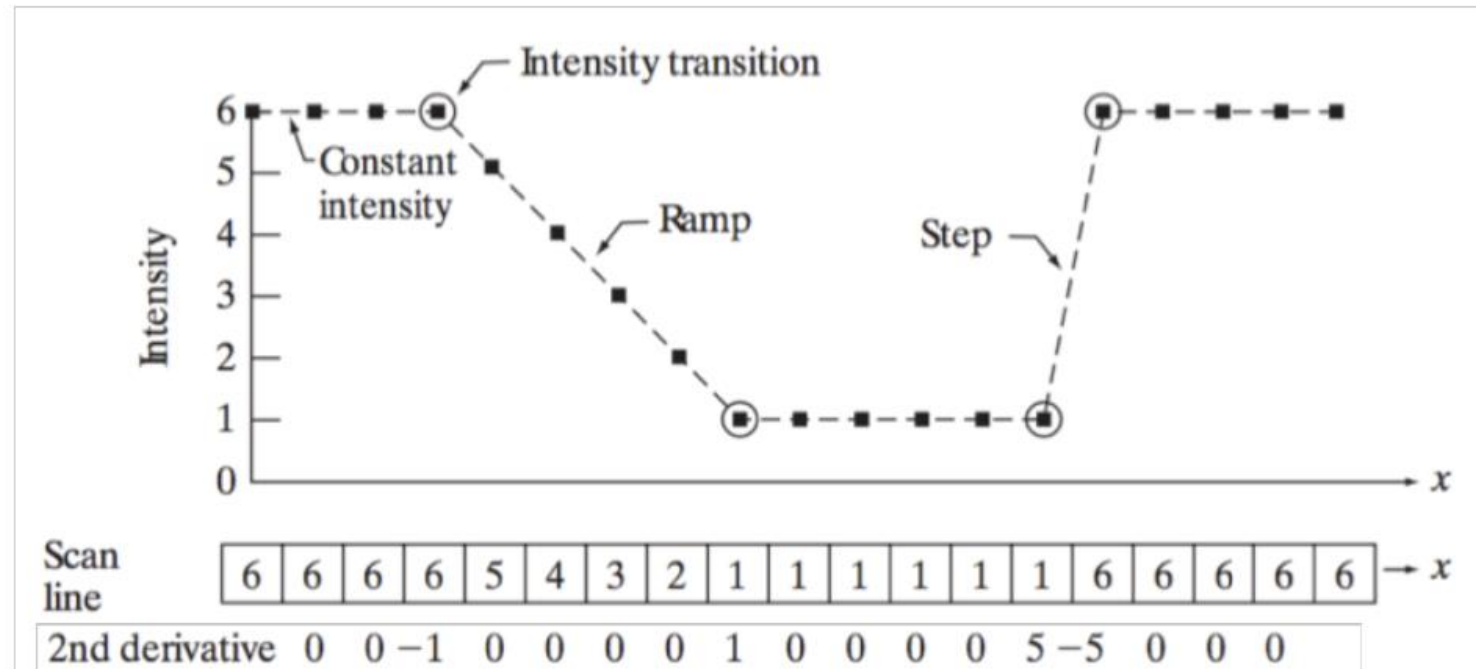
First-order derivative of a one-dimensional function: $\frac{df}{dx} = \lim_{\epsilon \rightarrow 0} \frac{f(x + \epsilon) - f(x)}{\epsilon} \approx f(x + 1) - f(x)$



Sharpening filters

Derivatives of a digital discrete functions are defined in term of differences.

Second-order derivative: $\frac{d^2 f}{d^2 x} \approx f(x+1) + f(x-1) - 2f(x)$



Sharpening filters

First-order derivative:

- would result in thick edges because the derivative is nonzero along a ramp.

Second-order derivative:

- would produce a double edge one pixel thick, separated by zeros

=> Enhances fine detail much better than the first derivative, and are also easier to implement!

⇒ The **Laplacian** is the simplest *isotropic* second-order derivative operator.

(isotropic means here that the operator behaves the same in all directions. The laplacian does not favor any particular direction (horizontal, vertical, or diagonal) when computing the rate of change in intensity.)

Sharpening: The Laplacian filter

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

In discrete form, the Laplacian can be expressed in term of finite differences:

$$\begin{aligned}\nabla^2 f(x, y) = & f(x + 1, y) + f(x - 1, y) + \\ & + f(x, y + 1) + f(x, y - 1) - 4f(x, y)\end{aligned}$$

Sharpening: The Laplacian filter

Since derivatives are linear operators, the Laplacian is a linear operator and hence can be implemented as a convolution with a proper filter mask:

0	1	0
1	-4	1
0	1	0

This filter gives an isotropic result for increments of 90°

1	1	1
1	-8	1
1	1	1

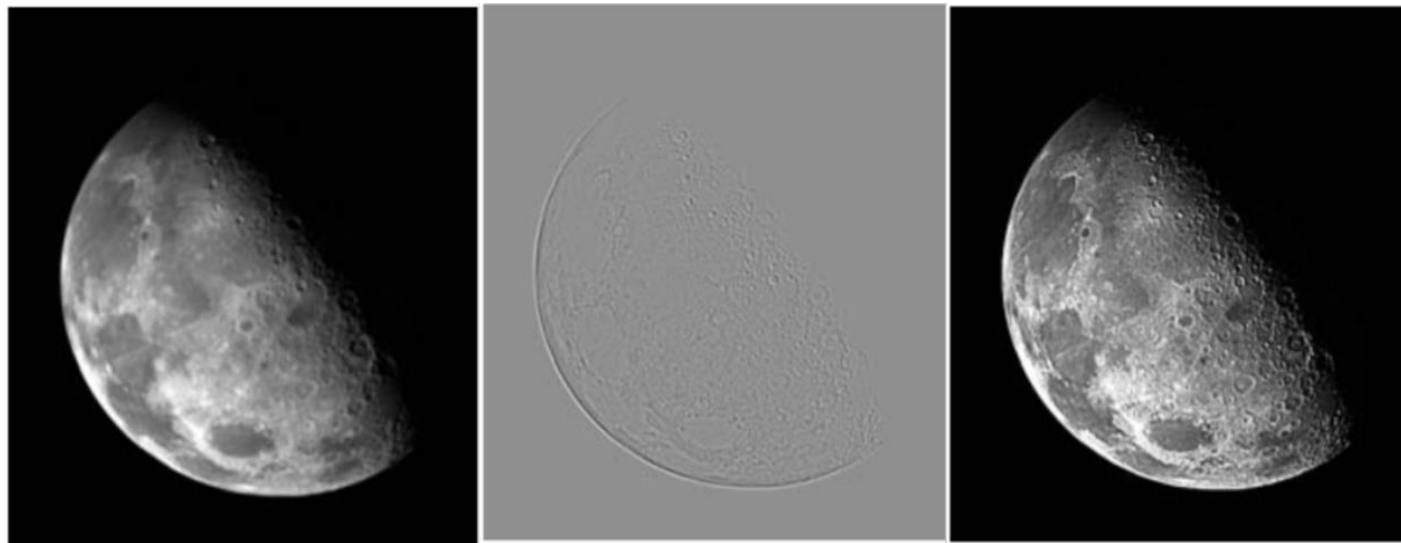
This filter gives an isotropic result for increments of 45°

The diagonal directions can be incorporated in the definition of the digital Laplacian by adding two more terms, one for each of the two diagonal directions.

Sharpening: The Laplacian filter

- The Laplacian operator highlights intensity discontinuities in an image and deemphasizes regions with slowly varying intensity levels.
- If we add the effect of the laplacian operator to the original image we are effectively sharpening the details while preserving background slowly-varying gradients.

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



Spatial filtering: Non-linear filters

Order-statistic filters are non-linear spatial filters whose response is based on:

1. ordering the pixels contained in the image area encompassed by the filter
2. replacing the value of the center pixel with the value determined by the ranking result.

This category of filters is non-linear and cannot be performed as a convolution

Spatial filtering: Non-linear filters

Order-statistic filters

Median-filter: replaces the value of a pixel by the median of the intensity values in the neighborhood of that pixel (the original value of the pixel is included in the computation of the median)

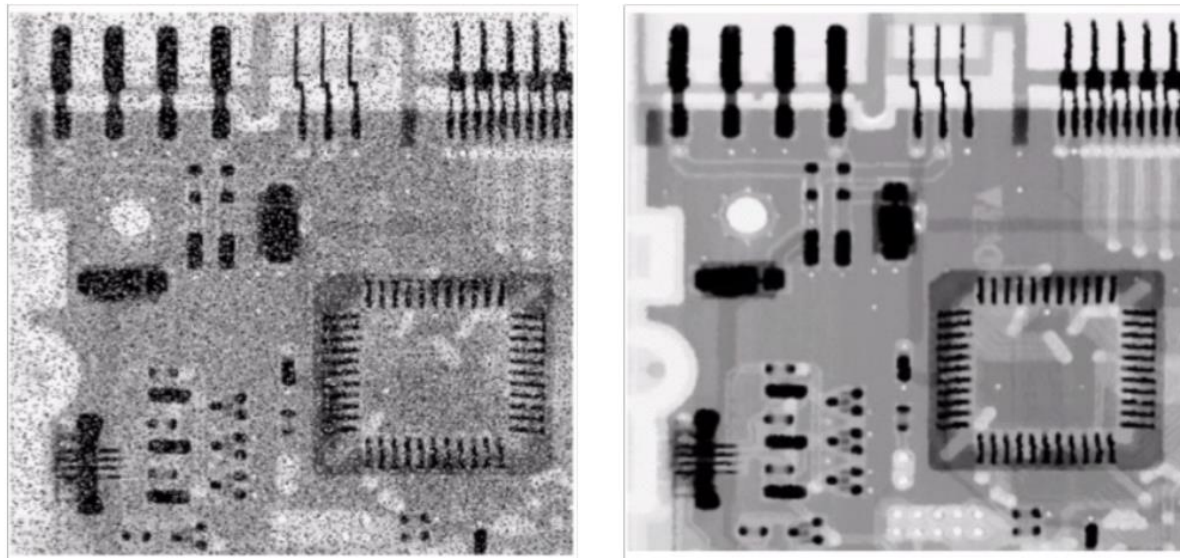
Max-filter: Replaces the value of a pixel with the brightest intensity value in the neighborhood

Min-filter: Replaces the value of a pixel with the darkest intensity value in the neighborhood

Spatial filtering: Non-linear filters

α -trimmed mean filter

- It uses a robust estimate of the mean (that does not take into account outliers):
Eliminate the top and bottom $\alpha/2$ values.
Take the average of the remaining pixels.



Spatial filtering: Non-linear filters

Bilateral filtering

- It is a non-linear, edge-preserving smoothing technique used in image processing.
- Unlike traditional filters that blur edges, it considers both **spatial proximity** and **intensity similarity** when averaging pixel values.
- **Spatial Weighting:** Nearby pixels have a higher influence (Gaussian function based on distance).
- **Intensity Weighting:** Pixels with similar intensity values have a higher influence (Gaussian function based on intensity difference).

Spatial filtering: Non-linear filters

Bilateral filtering

- It considers both **spatial proximity** and **intensity similarity** when averaging pixel values.

$$I'(x) = \frac{1}{W_p} \sum_{x_i \in \mathcal{N}(x)} I(x_i) \cdot f_s(\|x_i - x\|) \cdot f_r(|I(x_i) - I(x)|)$$

$I'(x)$ is the filtered intensity at pixel x .

$I(x_i)$ is the intensity of a neighboring pixel x_i .

$\mathcal{N}(x)$ is the neighborhood around x .

$$f_s(\|x_i - x\|) = \exp\left(-\frac{\|x_i - x\|^2}{2\sigma_s^2}\right)$$

$$f_r(|I(x_i) - I(x)|) = \exp\left(-\frac{|I(x_i) - I(x)|^2}{2\sigma_r^2}\right)$$

$$W_p = \sum_{x_i \in \mathcal{N}(x)} f_s(\|x_i - x\|) \cdot f_r(|I(x_i) - I(x)|)$$

Spatial filtering

What will be seen in Lab 3:

- Other filters.
- Noise.
- Noise removal using filters.
- How to assess the effectiveness of the filters (qualitative/quantitative evaluations):
 - PSNR
 - SSIM