Digital Image Processing

Fundamental notions and point processing

Erchan Aptoula

Spring 2016-2017

Now that we have an idea about **WHAT** digital images and digital image processing is, let's take a look into the **HOW** of digital image processing.

For the sake of simplicity, every image $f: \mathbb{Z}^2 \rightarrow \mathbb{Z}$ from now on is assumed **digital** (on a 2D Cartesian grid with integer pixel intensities), unless stated otherwise.

Digital images are processed/transformed through **operators**; e.g.

$$\mathrm{O}[f(x,y)] = g(x,y)$$

One of the most important operator properties is **linearity**;

Operator O is **linear** if it satisfies **additivity**:

$$\mathrm{O}(f+g) = O(f) + O(g), \text{ where } f \text{ and } g \text{ are two images.}$$

AND homogeneity:

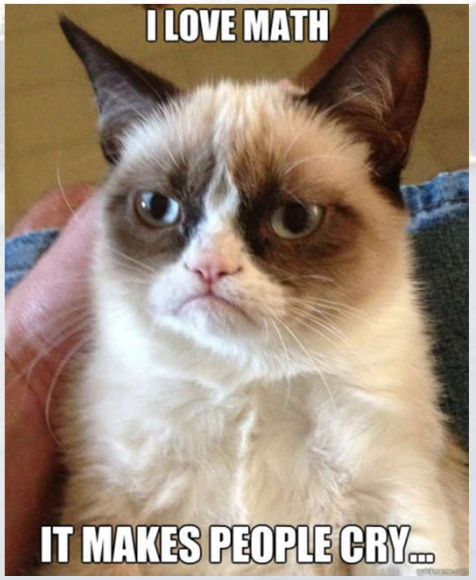
$$\mathrm{O}(af) = aO(f)$$

where a is a constant.

Example of a linear operator: summation $\Sigma(f)$

Example of a non-linear operator: $\max(f)$

Engineers are fond of linearity and linear operators. Why?



Another important operator property is **shift or translation invariance**; Operator O is shift invariant if given:

$$O(f(x,y)) = g(x,y)$$

we have:

$$O\big(f(x-k,y-m)\big) = g(x-k,y-m)$$

i.e., if the input is shifted/translated by a certain amount, the output will be shifted by the same amount. Thus, *the location of the origin of the coordinate system is irrelevant*.

We'll revisit linear image processing in more detail after a few weeks.

Point processing refers to operators, the output of which for any given location $p = (x, y)$ in an image depends only on the value $f(p)$ of the image at the said location.

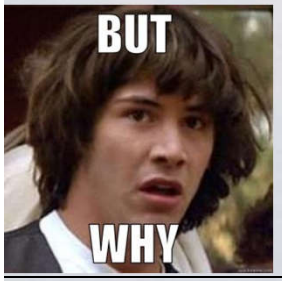
For instance **arithmetic** and **logical** operations.

$$a(x, y) = f(x, y) + g(x, y)$$

$$s(x, y) = f(x, y) - g(x, y)$$

$$p(x, y) = f(x, y) \times g(x, y)$$

$$d(x, y) = f(x, y) \div g(x, y)$$



…would anyone want to add two images?

…for instance for **noise averaging**.

Say you have a telescope and you take images of deep space. Your images will be probably very noisy due to all the particles and radiation between you and the object many light-years away:

$$f(x,y) = g(x,y) + n(x,y)$$

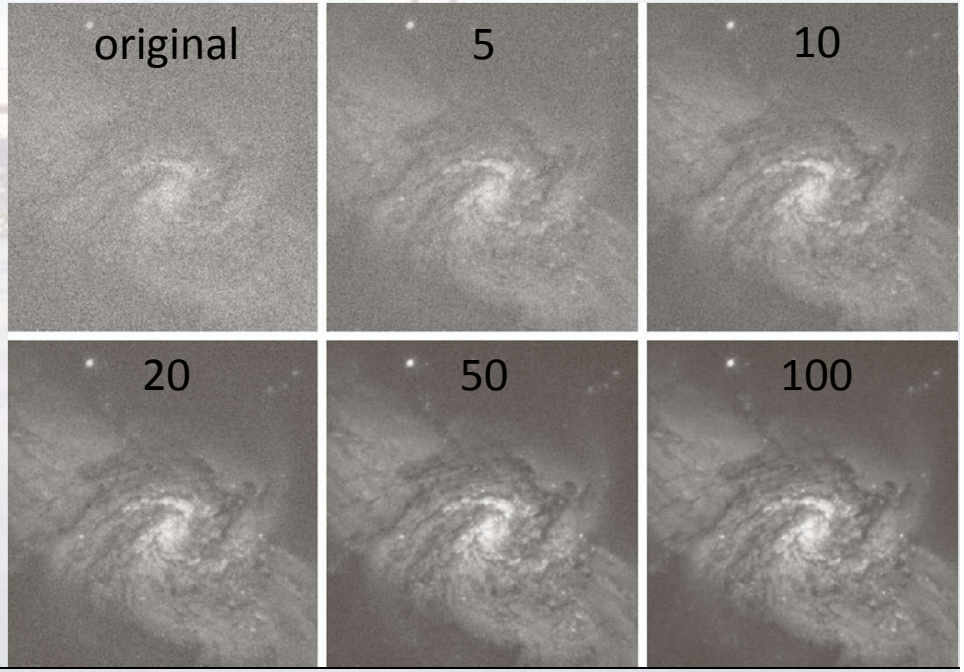
where g is the noise-free image and n the **uncorrelated** noise.

Now, assume you acquire N aligned images: $f_1(x,y), f_2(x,y), \ldots, f_N(x,y)$ of the same scene. They are all noisy. And you calculate their average:

$$\bar{f}(x,y) = \frac{1}{N}\sum_{i=1}^{N} f_i(x,y) = g(x,y) + \frac{1}{N}\sum_{i=1}^{N} n(x,y) = g(x,y) + \bar{n}(x,y)$$

If the noise has zero mean this means that the more samples you acquire, the more your approach a noise-free version of f.

In practice this approach's main problem is the perfect alignment of the samples.



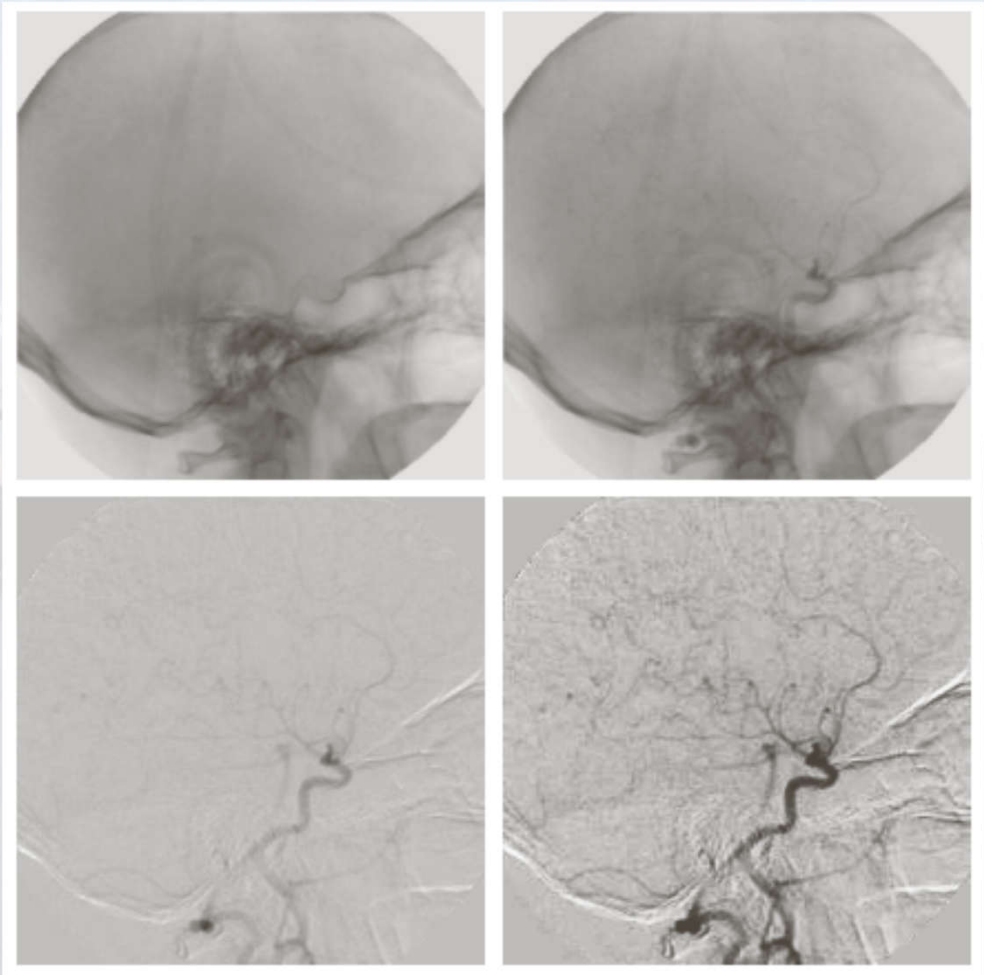
Another motivation for arithmetically combining images is **High Dynamic Range Imaging** (HDRI)!

Non-HDR cameras (i.e. LDR) have a limited exposure range.

HDR cameras capture **several** exposures of the same scene with **narrower** ranges, and **combine** them.

-8 f-stops -2 f-stops +2 f-stops +4 f-stops

Subtraction can be used for highlighting the differences between similar images:

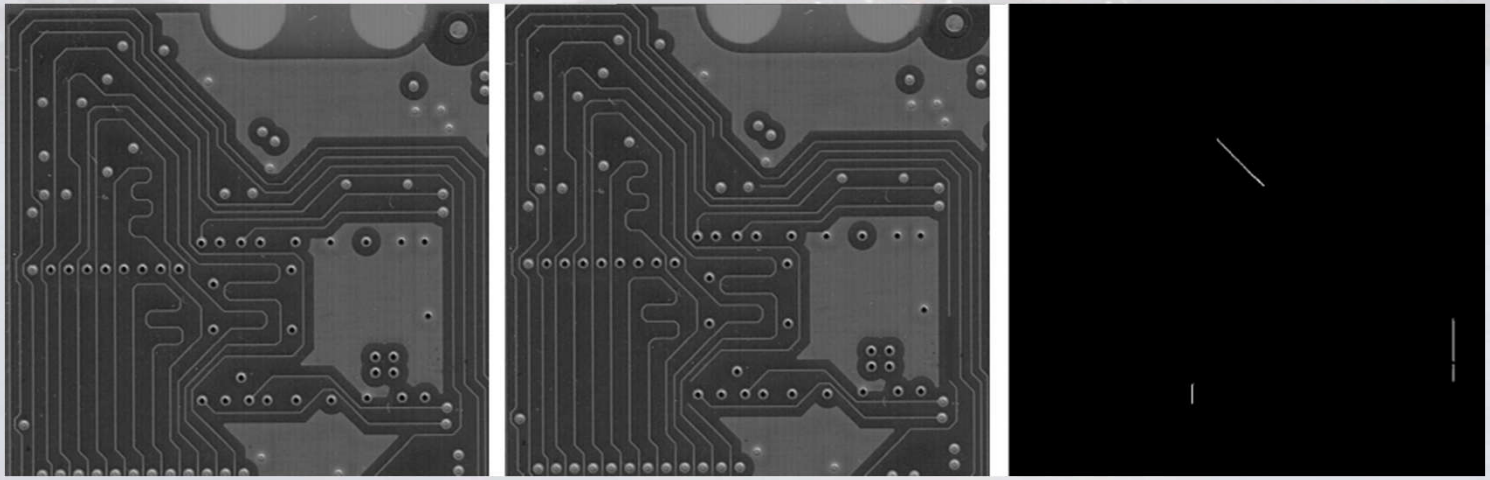


a b
c d

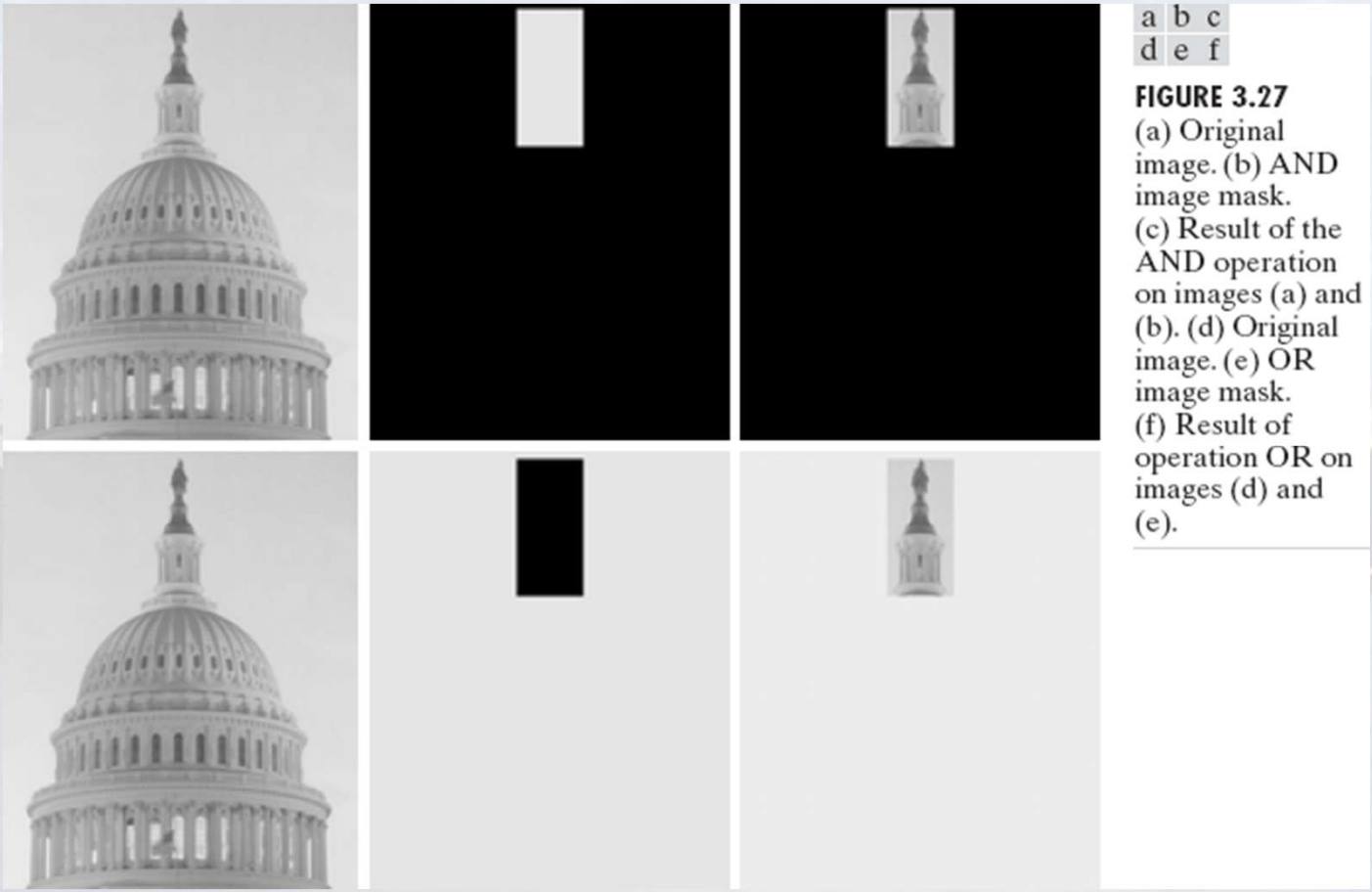
FIGURE 2.28 Digital subtraction angiography. (a) Mask image. (b) A live image. (c) Difference between (a) and (b). (d) Enhanced difference image. (Figures (a) and (b) courtesy of The Image Sciences Institute, University Medical Center, Utrecht, The Netherlands.)

Subtraction also serves for foreground detection from video sequences by computing frame differences:

As well as for defect detection, if you have a reference mask:



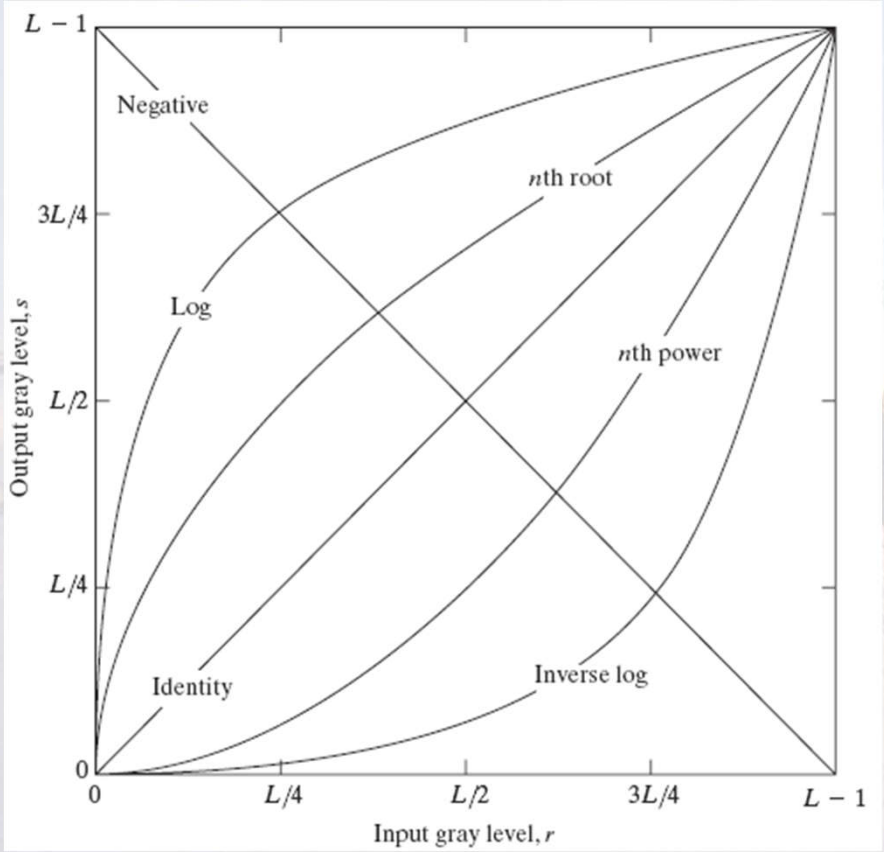
You can also apply logical operators on a pixel-wise basis, e.g. AND, OR, XOR, NOT:



a b c
d e f

FIGURE 3.27
(a) Original image. (b) AND image mask. (c) Result of the AND operation on images (a) and (b). (d) Original image. (e) OR image mask. (f) Result of operation OR on images (d) and (e).

Another group of operators are “intensity transformations” operating only on the input pixel intensity for calculating the output pixel intensity.



Power-law transformations are of particular significance, as they are used for **gamma correction**.

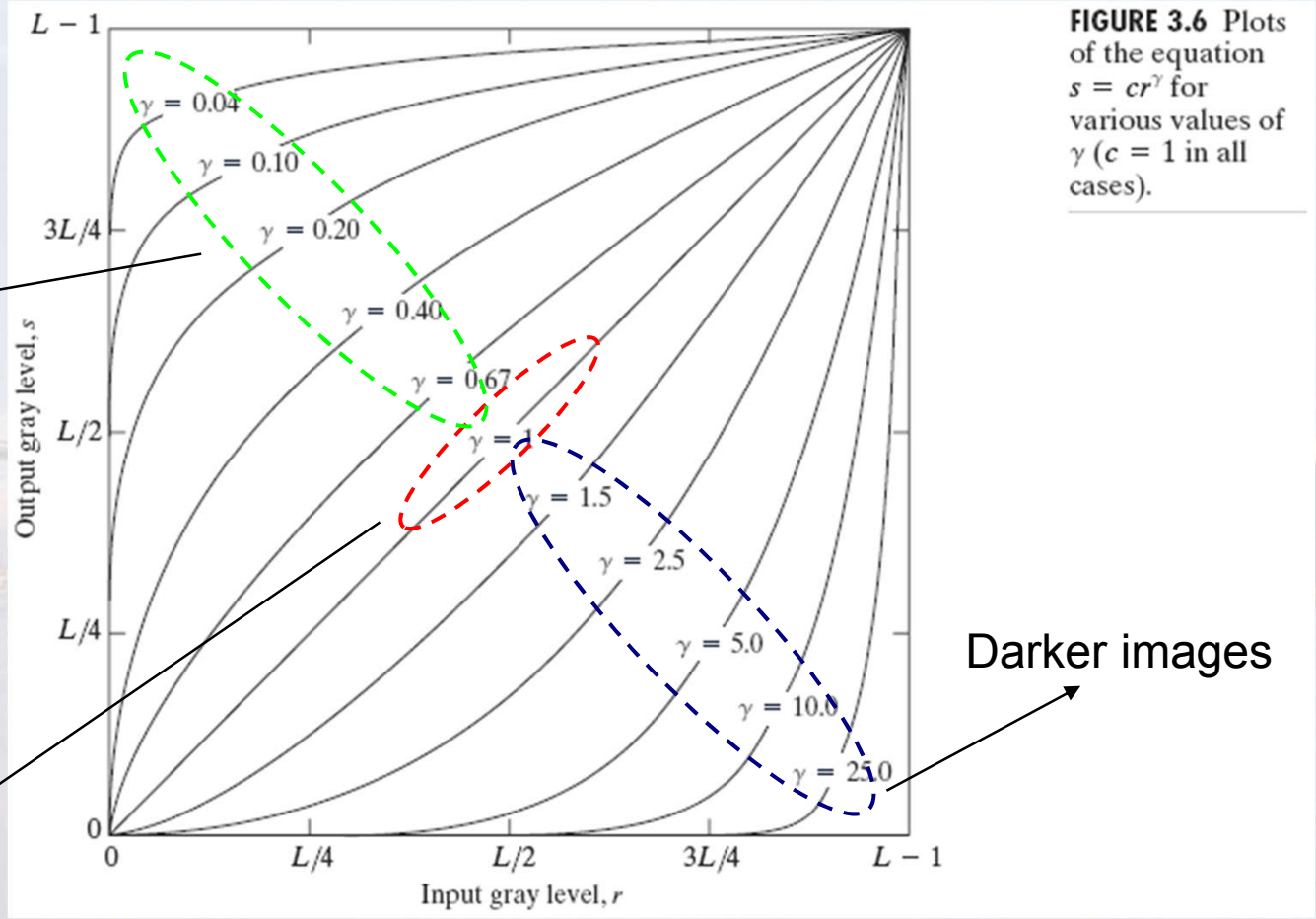


FIGURE 3.6 Plots of the equation $s = cr^{\gamma}$ for various values of γ ($c = 1$ in all cases).

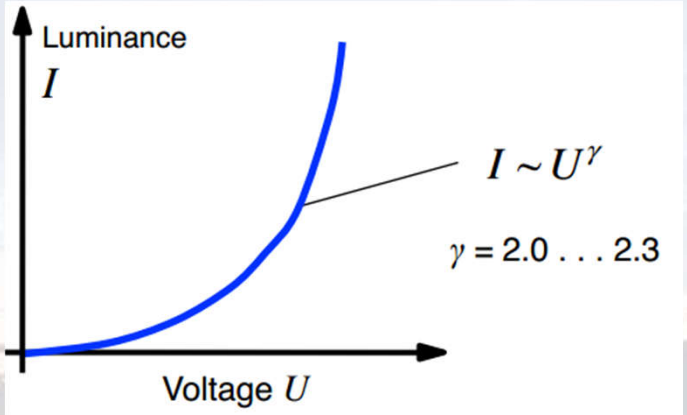
Brighter images

Darker images

Lazy man operator

What's **gamma correction**?

Cathode ray tubes (CRT) are non-linear:



Which is why cameras are equipped with a gamma pre-distortion circuit:

$$U \sim I^{1/\gamma}$$

Another intensity transformation is "**bit-plane slicing**"

a b c
d e f
g h i

FIGURE 3.14 (a) An 8-bit gray-scale image of size 500 × 1192 pixels. (b) through (i) Bit planes 1 through 8, with bit plane 1 corresponding to the least significant bit. Each bit plane is a binary image.

Geometric transformations modify the spatial relationships between pixels of an image.

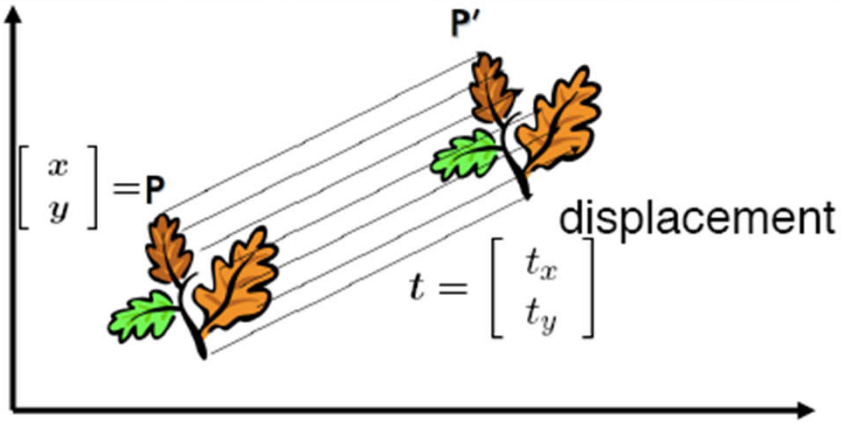
$$(v, w) = T\{(x, y)\}$$

Where (x, y) is the pixel in the original image, and (v, w) its new location in the transformed image. The most common is the **affine transform** (1990), that has the general form:

$$[v\ w\ 1] = T[x\ y\ 1] = \begin{bmatrix} t_{11} & t_{12} & t_{13} \\ t_{21} & t_{22} & t_{23} \\ 0 & 0 & 1 \end{bmatrix} [x\ y\ 1]$$

This transform can rotate, translate and scale, shear pixels depending on the chosen T.

Translation

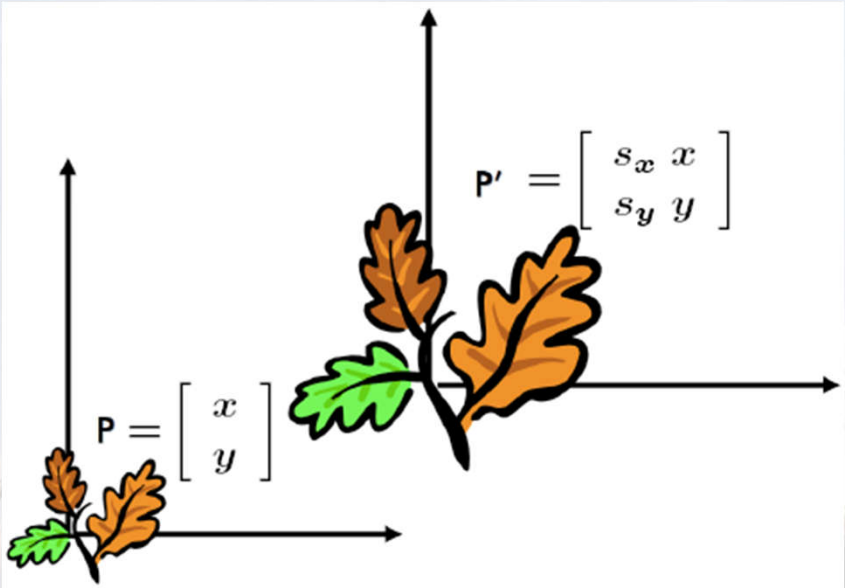


homogeneous coordinates

$$P' = P + \boldsymbol{t} = \begin{bmatrix} x + t_x \\ y + t_y \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

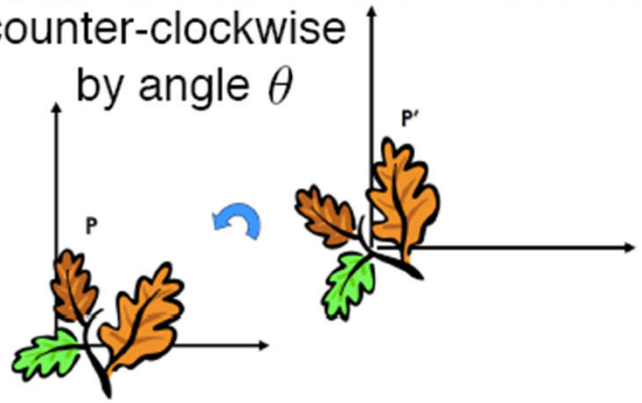
$$\Rightarrow \quad P' \rightarrow \begin{bmatrix} x + t_x \\ y + t_y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

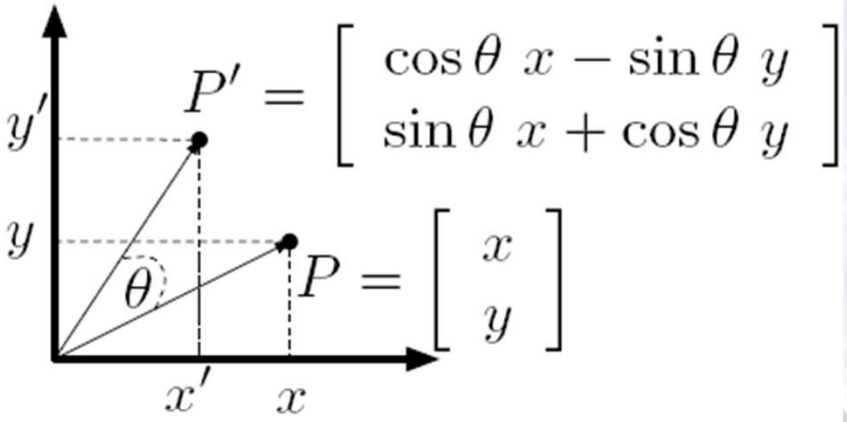
Scaling



$$\begin{bmatrix} s_x\, x \\ s_y\, y \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\text{scaling matrix}} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Rotation





$$P' = \begin{bmatrix} \cos\theta\ x - \sin\theta\ y \\ \sin\theta\ x + \cos\theta\ y \end{bmatrix}$$

$$P = \begin{bmatrix} x \\ y \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{R} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

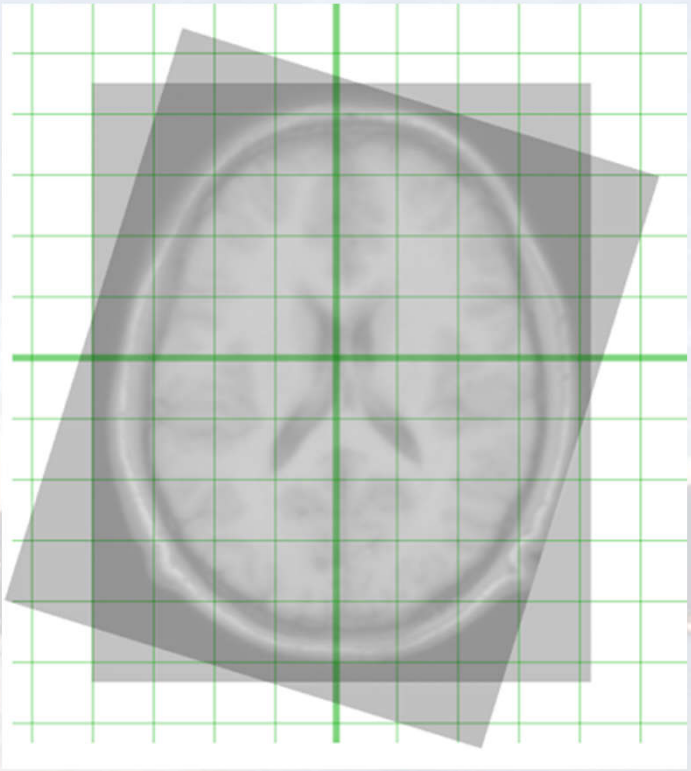
rotation matrix

Compound geometric transform

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\text{rotation matrix}} \underbrace{\begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}}_{\text{translation matrix}} \underbrace{\begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\text{scaling matrix}} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} R\,S & \boldsymbol{t} \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Image registration is a problem encountered often in medical and military image processing applications.



- Input and output images are available but the transformation function is unknown.

Goal: *estimate the transformation function* and use it to register the two images.

- One of the principal approaches for image registration is to use ***control points***.

The corresponding points are known precisely in the input and output (**reference**) images.

Thus it becomes just a question of estimating the unknown transform matrix.

???

$$\begin{bmatrix} t_{11} & t_{12} & t_{13} \\ t_{21} & t_{22} & t_{23} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}$$

$$x \cdot t_{11} + y \cdot t_{12} + 1 \cdot t_{13} + 0 \cdot t_{21} + 0 \cdot t_{22} + 0 \cdot t_{23} = x'$$

$$0 \cdot t_{11} + 0 \cdot t_{12} + 0 \cdot t_{13} + x \cdot t_{21} + y \cdot t_{22} + 1 \cdot t_{23} = y'$$

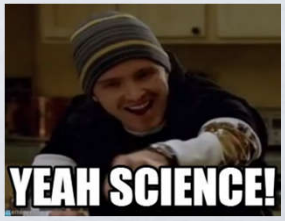
$$\begin{bmatrix} x & y & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x & y & 1 \end{bmatrix} \begin{bmatrix} t_{11} \\ t_{12} \\ t_{13} \\ t_{21} \\ t_{22} \\ t_{23} \end{bmatrix} = \begin{bmatrix} x' \\ y' \end{bmatrix}$$

With all the control points in place, we have a nice system of equations.

This is one of the reasons you were taught numerical analysis.

$$\underbrace{\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_1 & y_1 & 1 \\ x_2 & y_2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_2 & y_2 & 1 \\ \vdots & & & & & \\ x_N & y_N & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_N & y_N & 1 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} t_{11} \\ t_{12} \\ t_{13} \\ t_{21} \\ t_{22} \\ t_{23} \end{bmatrix}}_{X} = \underbrace{\begin{bmatrix} x'_1 \\ y'_1 \\ x'_2 \\ y'_2 \\ \vdots \\ x'_N \\ y'_N \end{bmatrix}}_{B}$$

If $\det(A) \neq 0 \Rightarrow X = A^{-1}B$



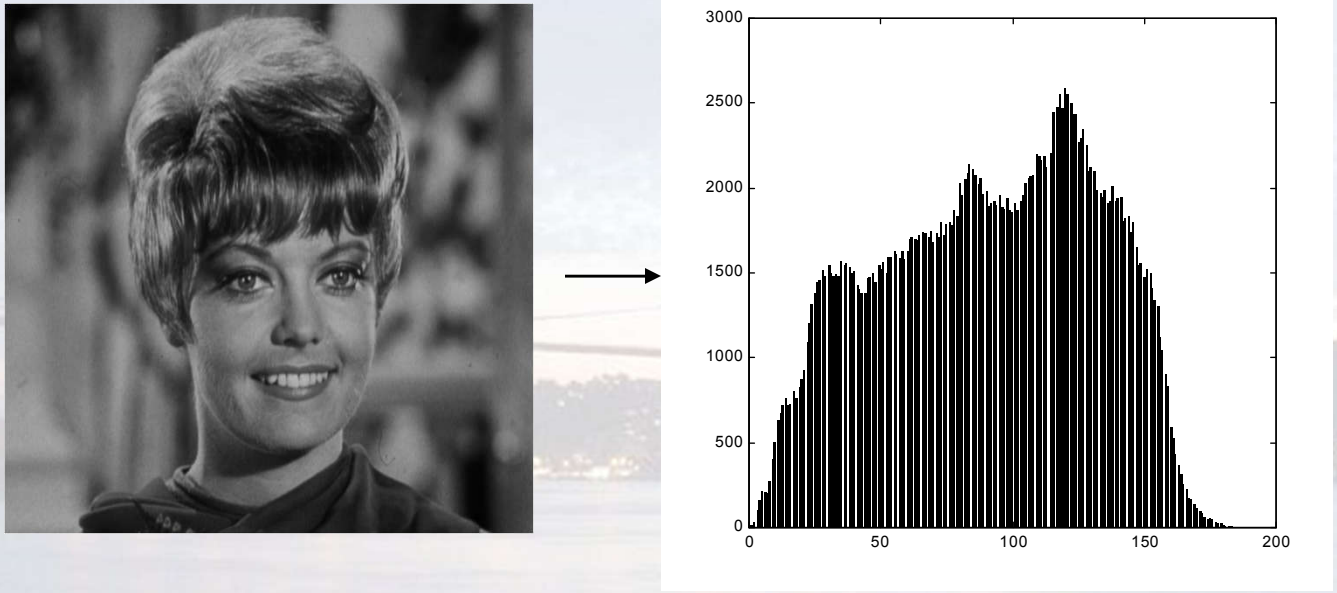
input

transformed input

inverse transform

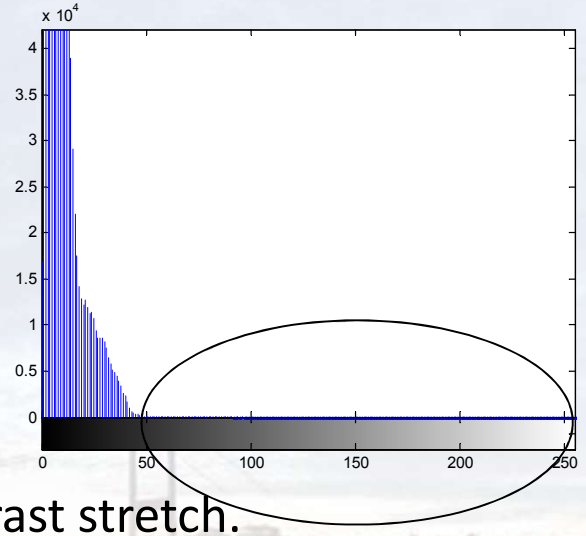
estimation error

The histogram of an image represents the occurrence frequencies of its gray levels.

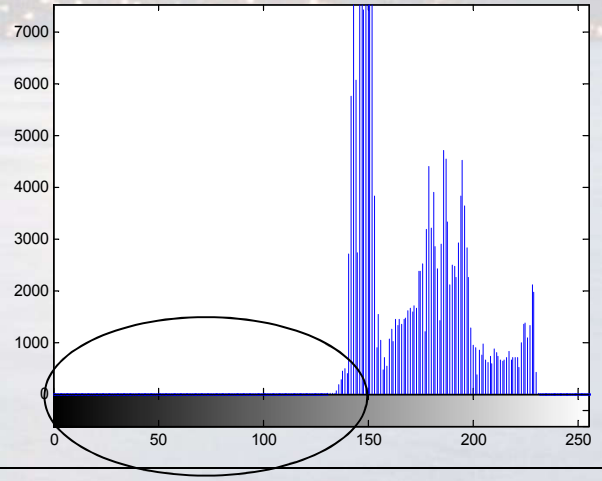


If the histogram bins are further normalized to [0,1] by dividing every bin with the total number of pixels, then the histogram becomes the **probability distribution function (pdf)** of the gray levels of the image!

The histogram helps to determine under-exposed



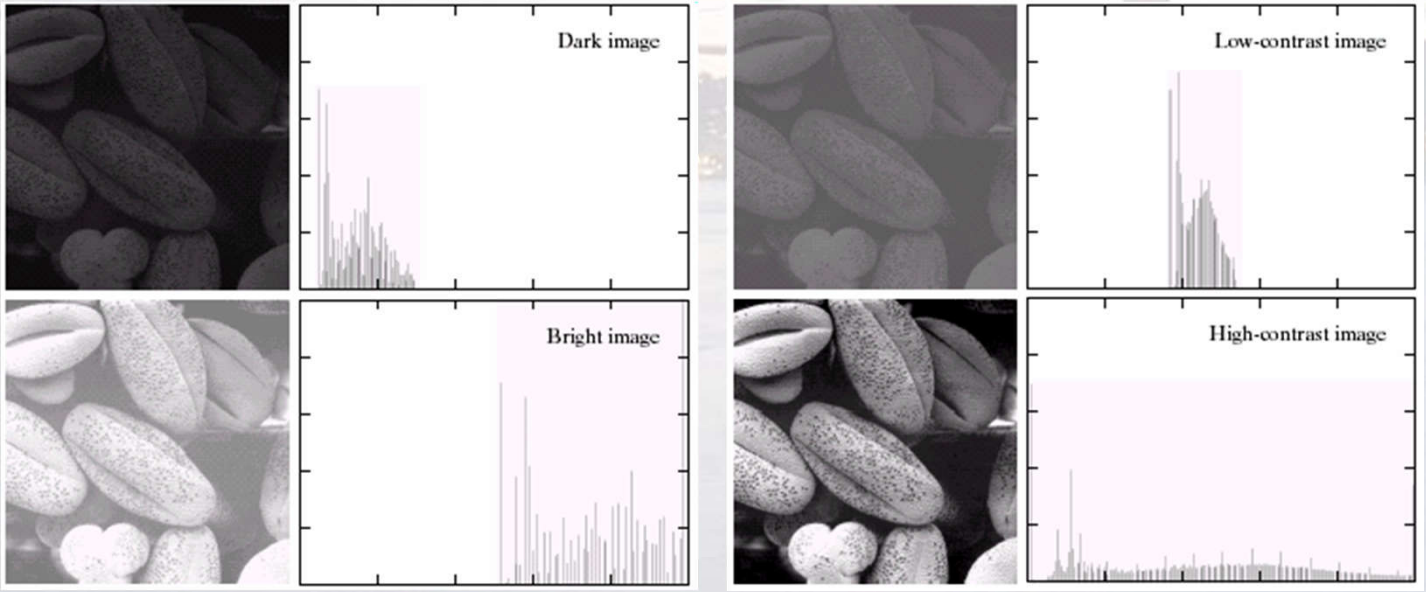
and over-exposed images, easily rectifiable with a contrast stretch.



Contrast is a measure of the "range" of an image; i.e. how spread its intensities are. It has many formal definitions; e.g. Michelson's:

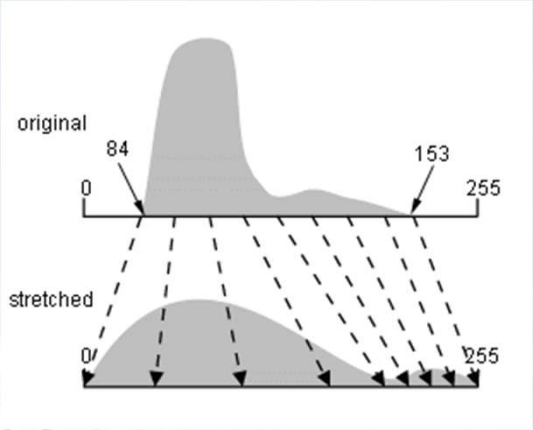
$$\frac{I_{max} - I_{min}}{I_{max} + I_{min}}$$

Contrast is strongly tied to an image's overall visual quality. Ideally, we'd like images to use the entire range of values available to them.



Contrast stretching is a straightforward way of accomplishing it.

$$g(x,y) = K\frac{f(x,y) - \min[f(x,y)]}{\max[f(x,y)] - \min[f(x,y)]}$$

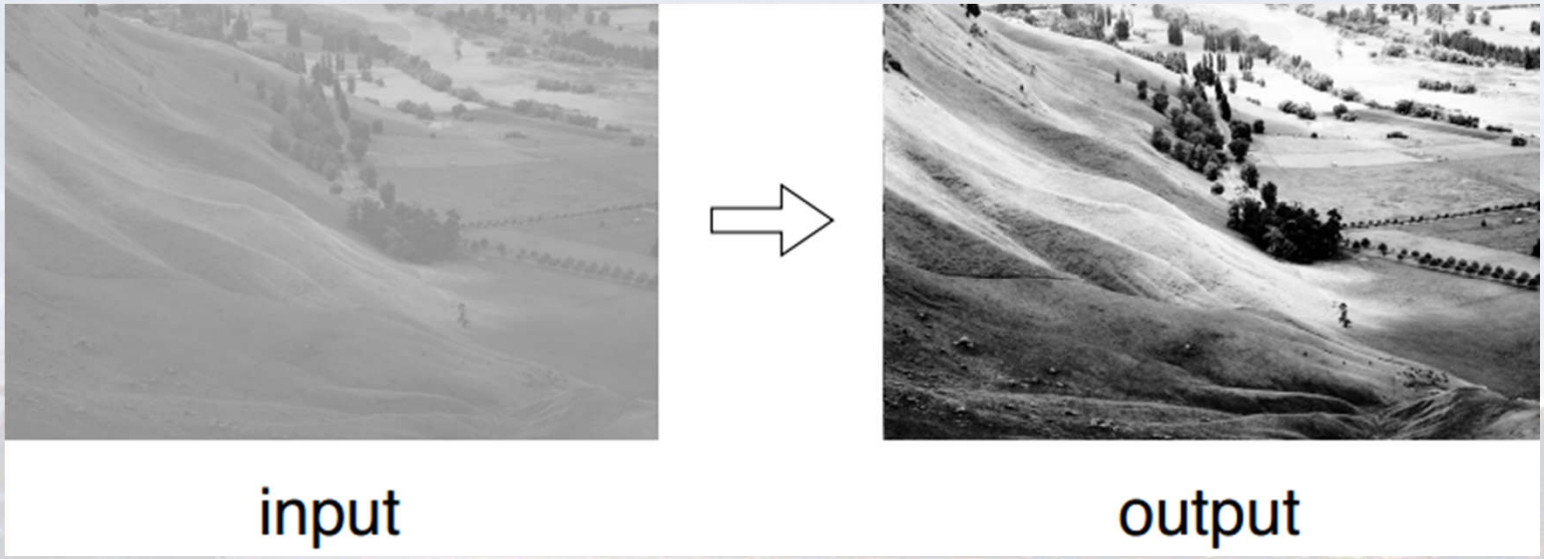


Straightforward, yet **unreliable**; why?

input

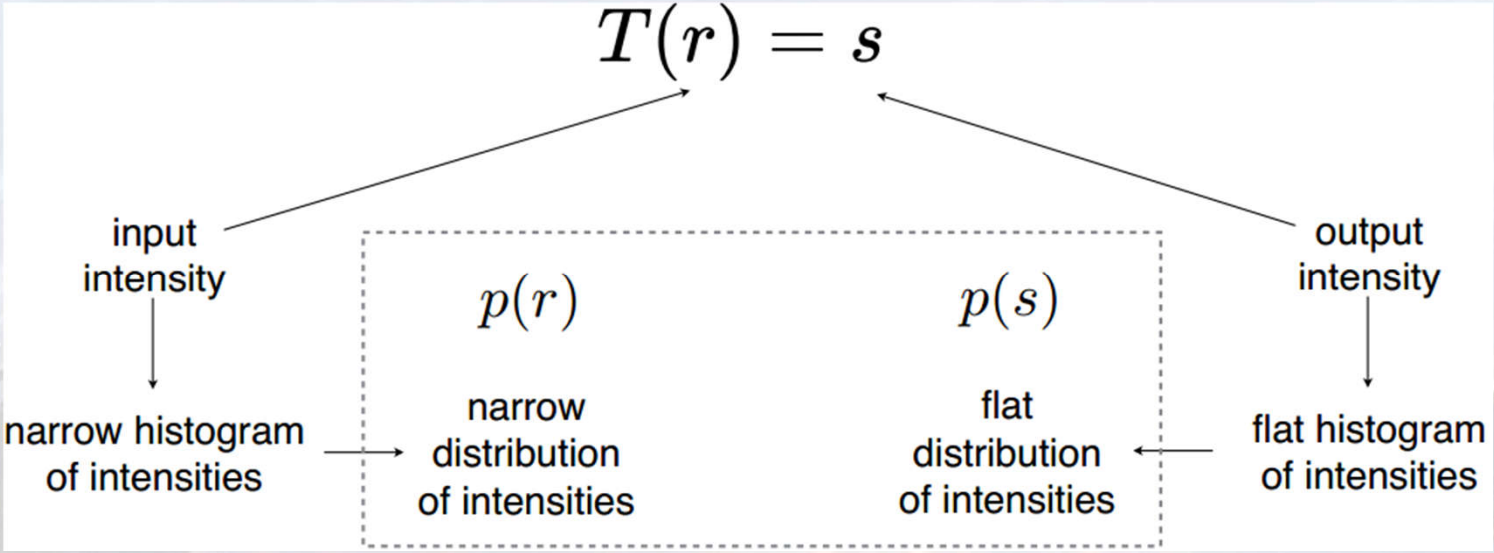
output

A better alternative is **histogram equalization.**



- Increases the contrast by spreading out the intensity distribution.
- Often produces artifacts.

Histogram equalization is a non linear transformation, applied to every pixel of the input, so that the output has a **uniform** distribution of gray values; i.e. a flat histogram!



But how do we convert an arbitrary pdf into a constant one?

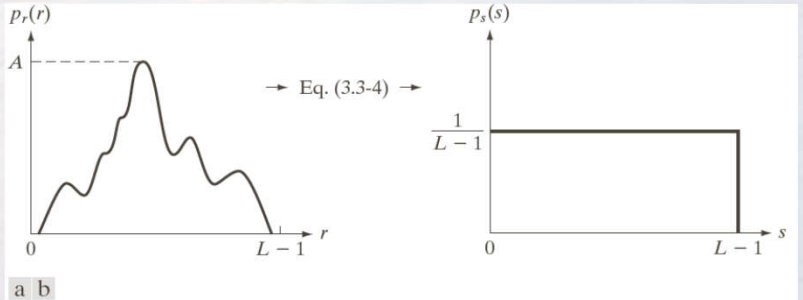


FIGURE 3.18 (a) An arbitrary PDF. (b) Result of applying the transformation in Eq. (3.3-4) to all intensity levels, r. The resulting intensities, s, have a uniform PDF, independently of the form of the PDF of the r's.

It's easy! Let's look at the ideal, continuous case first.

- The original and transformed intensities can be considered respectively as random variables r and s in [0, L-1]
- Let $p_r(r)$ and $p_s(s)$ be their pdf's.
- Let $s = T(r)$, and T be monotonically increasing in [0, L-1], differentiable and invertible.

Goal: find a constant $p_s(s)$

Recall from basic probability theory:

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

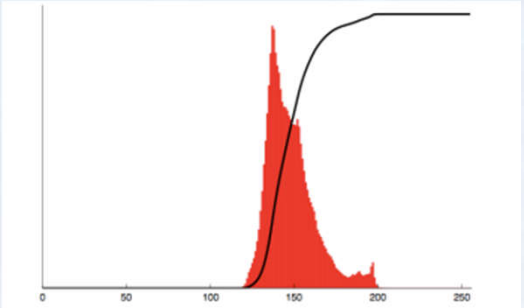
If R is a random variable with pdf p_R, then

$$P(r_1 < R < r_2) = \int_{r_1}^{r_2} p_R(r)dr$$

is its cumulative distribution function (cdf).

Consider this function: $\boldsymbol{s = T(r) = (L-1)\int_0^r p_r(w)dw}$

We recognize that it is the cdf of r. Since a pdf is positive, the cdf as its area, is also positive, and monotonically increasing as well as differentiable. Conditions verified!



intensity-level histogram (red)
cumulative histogram (black)

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

$p_s(s) = p_r(r)\left|\frac{dr}{ds}\right| = p_r(r)\left|\frac{1}{(L-1)p_r(r)}\right| = \frac{1}{L-1}$ It's constant! i.e. T suits our goal!

Now let's transfer the cumulative probability function to the discrete case.

- In the discrete world, images have only discrete amplitudes, each with its own probability:

$$P_0 = \frac{n_0}{n}, P_1 = \frac{n_1}{n}, \dots, P_{L-1} = \frac{n_{L-1}}{n}$$

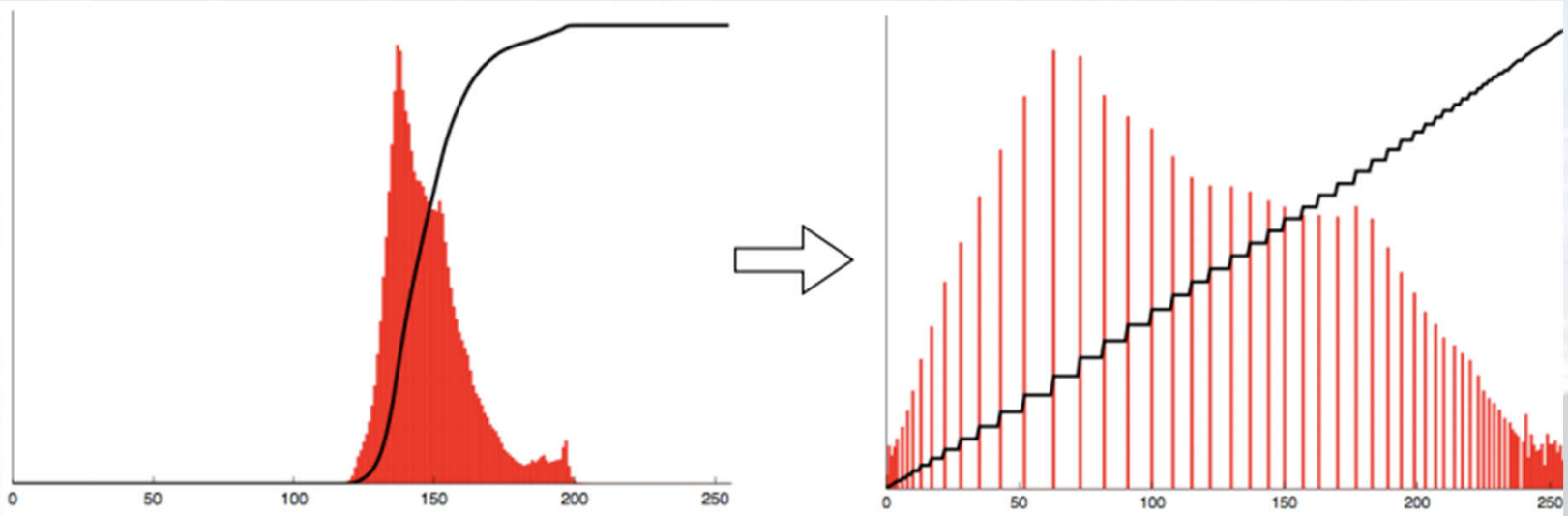
- And our transform function's discrete form

$$s_k = T(r_k) = (L-1)\sum_{j=0}^{k} P_j$$

Original image *Bay*

... after histogram equalization

And the effect of histogram equalization on the histogram:



intensity-level histogram (red); cumulative histogram (black)

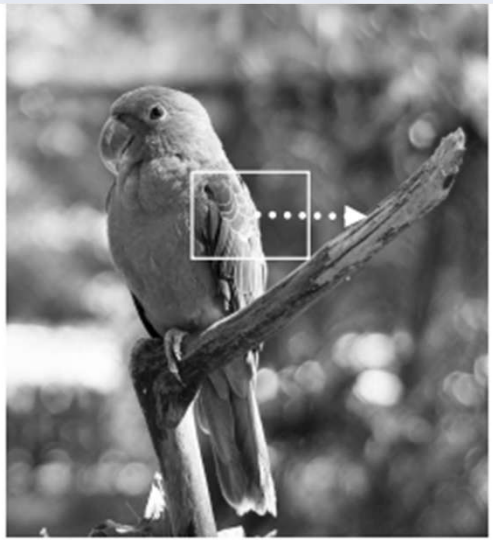
Original image

Histogram equalization

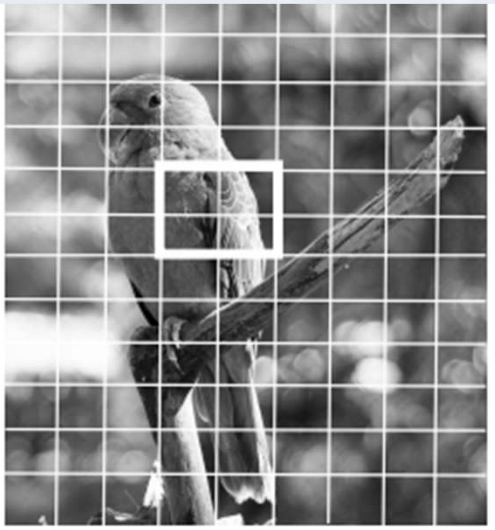
Histogram stretching by removing 2% percentile

Adapted from Selim Aksoy, Bilkent

A more sophisticated approach is to apply locally adaptive histogram processing.



Sliding window approach: different histogram (and mapping) for every pixel



Tiling approach: subdivide into overlapping regions, mitigate blocking effect by smooth blending between neighboring tiles

Original image
Parrot

Global histogram equalization

Adaptive histogram equalization, 8x8 tiles

Adaptive histogram equalization, 16x16 tiles

Fundamental notions and point processing: Histograms

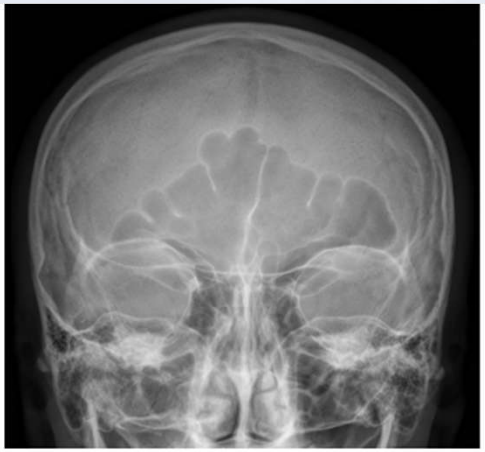
Original image
Dental Xray

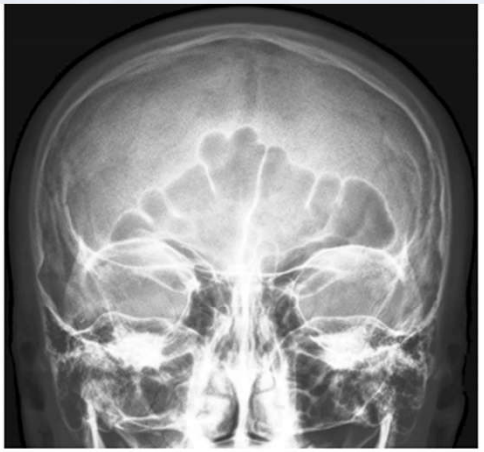
Global histogram equalization

Adaptive histogram equalization, 8x8 tiles

Adaptive histogram equalization, 16x16 tiles

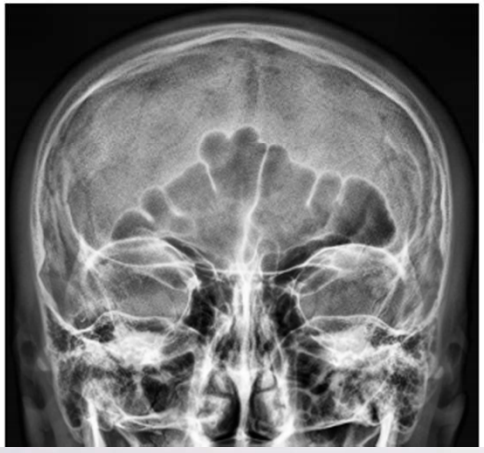
Original image
Skull Xray

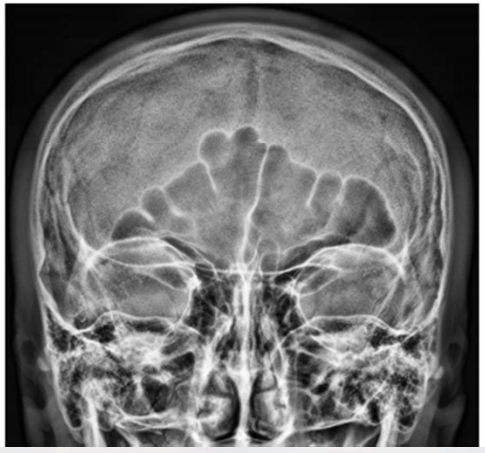




Global histogram equalization

Adaptive histogram equalization, 8x8 tiles





Adaptive histogram equalization, 16x16 tiles

In case a histogram is available, it can be also used for calculating some basic image statistics such as the mean and the variance:

Mean: $$m = \sum_{i=0}^{L-1} r_i p(r_i) = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y)$$

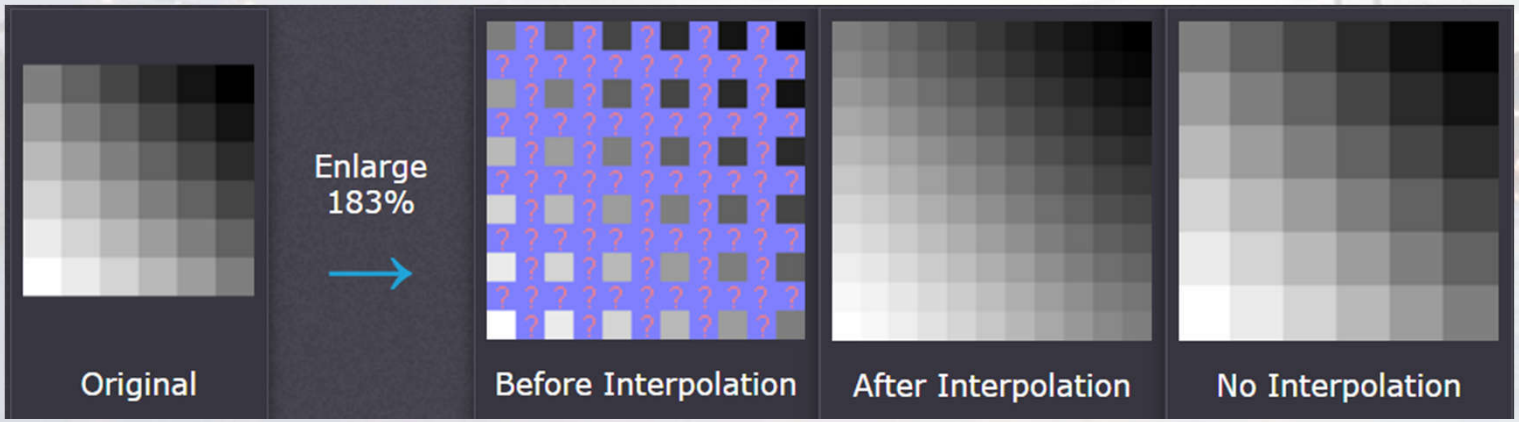
Variance: $$\sigma^2 = u_2(r) = \sum_{i=0}^{L-1} (r_i - m)^2 p(r_i) = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} [f(x,y) - m]^2$$

Moreover, as we'll see afterwards, a histogram can also serve as a rudimentary image descriptor; equipped with perfect **rotation** and **translation invariance**.

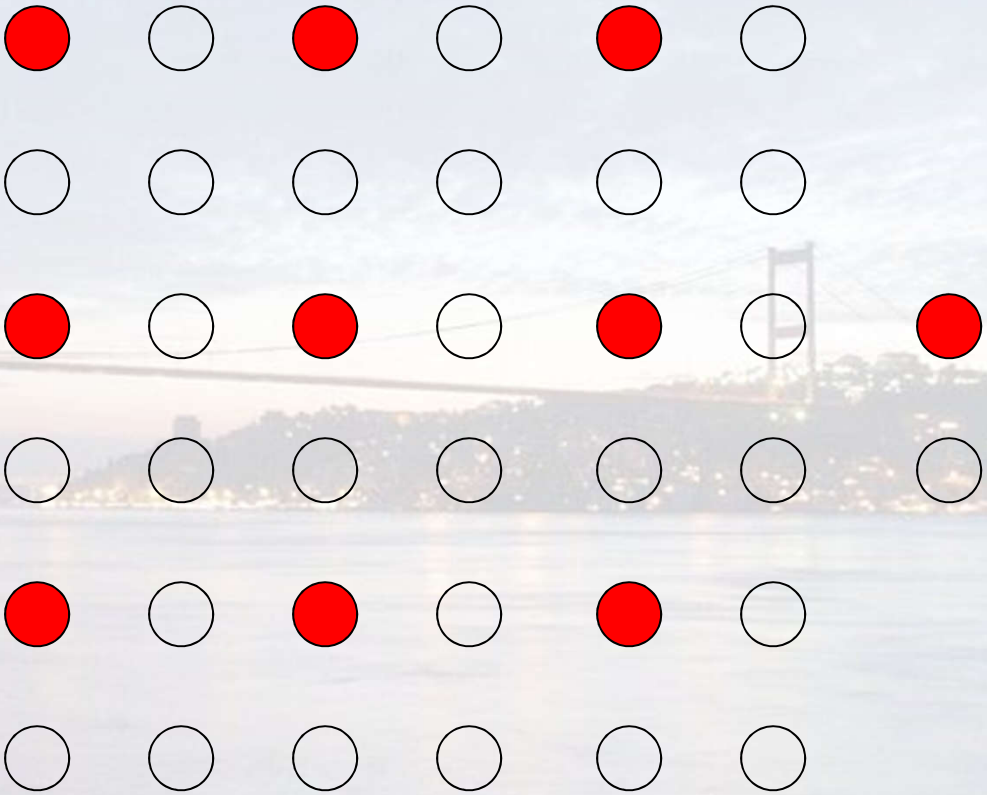
Interpolation is one of the fundamental tools of digital image processing, as it bridges the continuous and digital worlds.

Interpolation is all about guessing the values of pixels at missing and usually **non-integer** locations; it is required very often during geometric transformations.

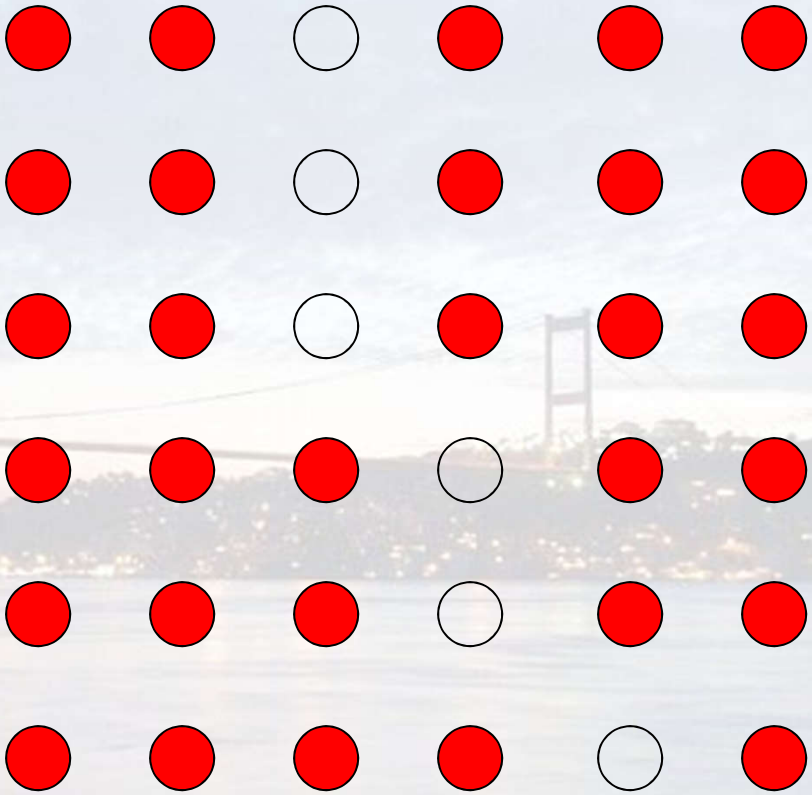
You use it in fact every time that you perform digital zoom on your cameras:



Zooming



Inpainting



Non-damaged Damaged

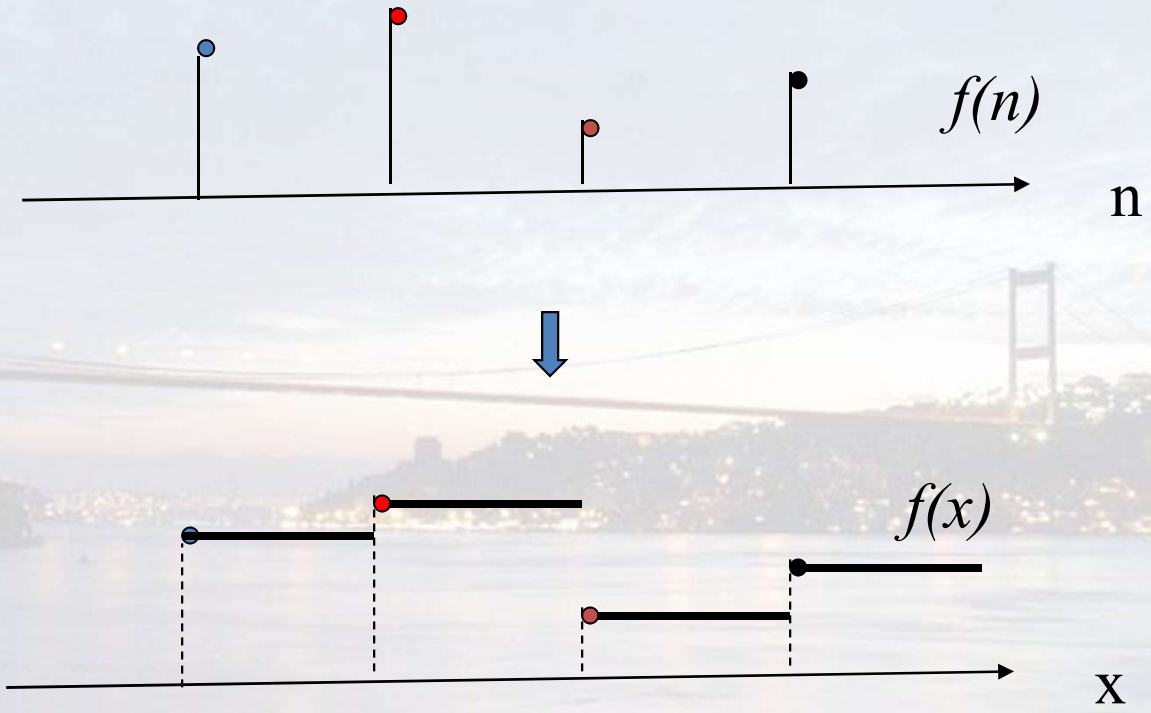
Upsampling or zooming: how to make this  10 times larger ?

The most straightforward way is to replicate every row and column 10 times ☺

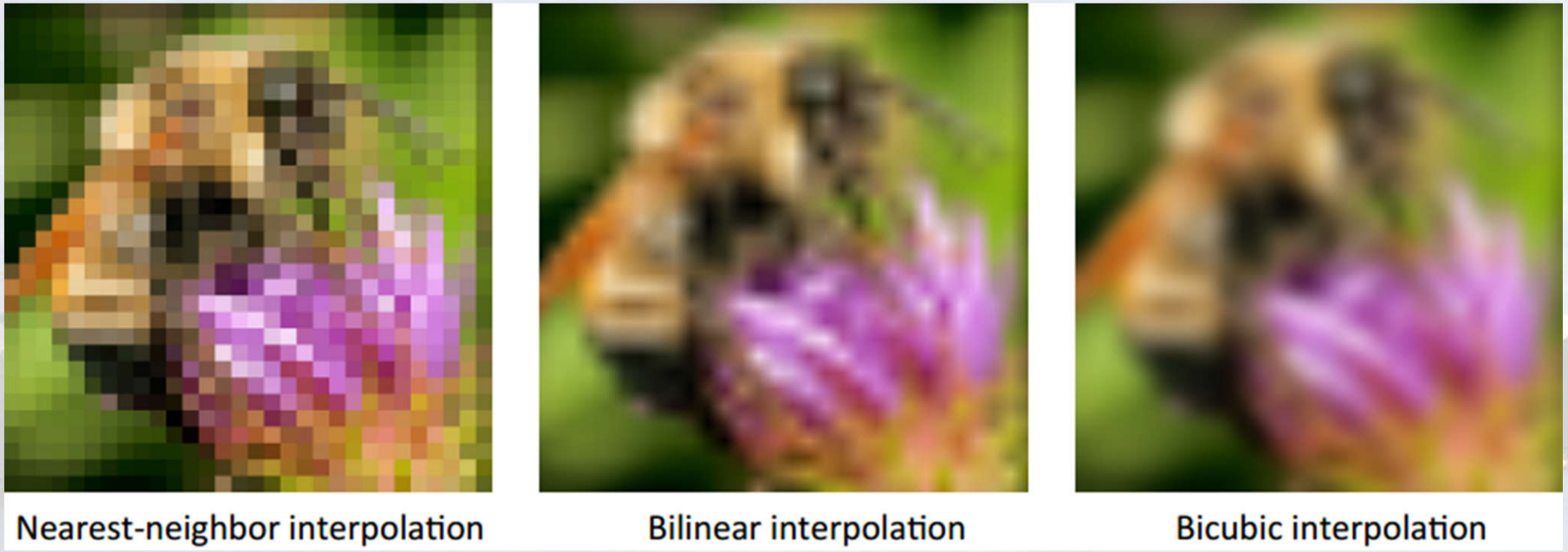
This is also known as:

- Nearest neighbor interpolation
- Zero-order interpolation

At 1D

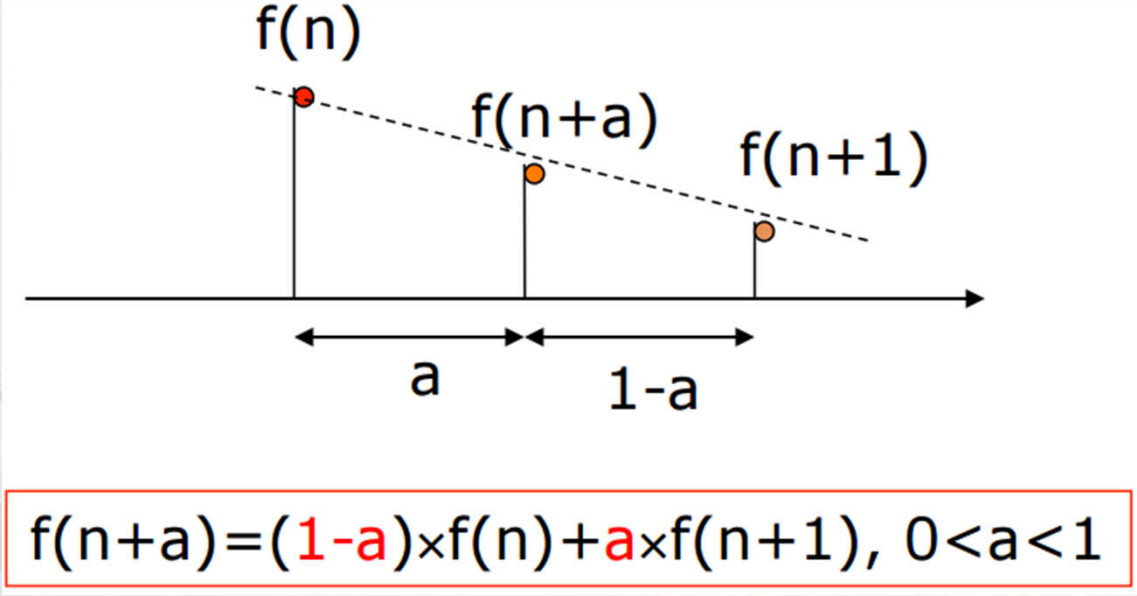


Other alternatives are bilinear and bicubic.



Nearest-neighbor interpolation | Bilinear interpolation | Bicubic interpolation

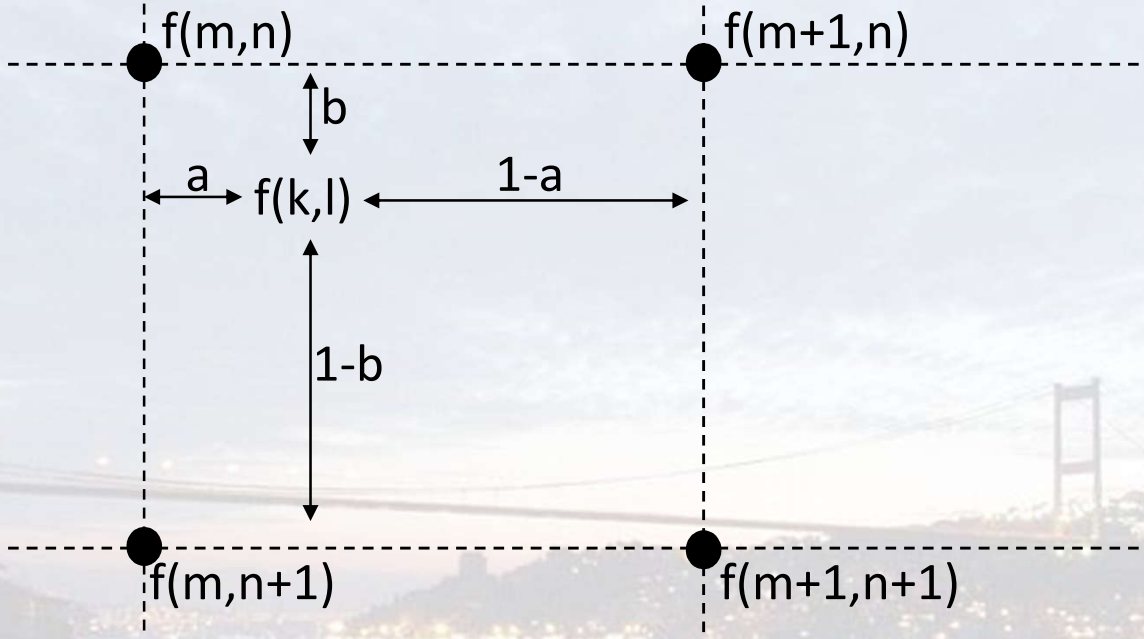
Let’s look at 1D linear interpolation: the closer the pixel, the higher its weight.



$$f(n+a)=(1-a)\times f(n)+a\times f(n+1),\ 0<a<1$$

If a=0.5, then it’s just the average.

The same idea at 2 dimensions: bilinear interpolation.



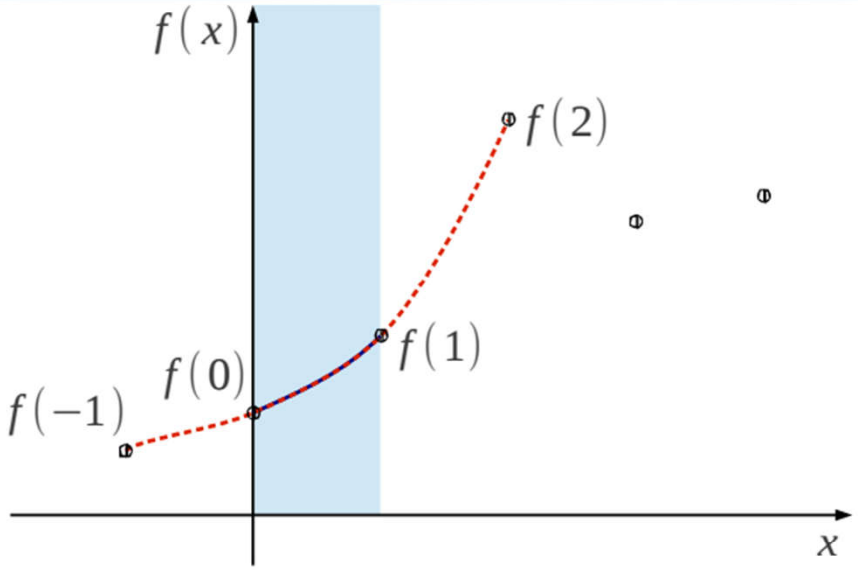
$$f(x,y) = (1-a)(1-b)f(m,n) + (1-a)bf(m,n+1) + a(1-b)f(m+1,n) + abf(m+1,n+1)$$

More generally: $f(x,y) = ax + by + cxy + d$ where the 4 coefficients are determined from the 4 equations corresponding to every neighbor f(,).

Bicubic interpolation uses the same logic as bilinear, yet with more neighbors.

At 1D:

Uses a 3rd degree polynomial instead of a line for interpolating the missing values.



Model: $f(x) = \sum_{i=0}^{3} a_i x^i = a_3 x^3 + a_2 x^2 + a_1 x^1 + a_0$

$$\begin{cases} f(-1) = a_3 \cdot (-1)^3 + a_2 \cdot (-1)^2 + a_1 \cdot (-1)^1 + a_0 \cdot (-1)^0 \\ f(0) = a_3 \cdot 0^3 + a_2 \cdot 0^2 + a_1 \cdot 0^1 + a_0 \cdot 0^0 \\ f(1) = a_3 \cdot 1^3 + a_2 \cdot 1^2 + a_1 \cdot 1^1 + a_0 \cdot 1^0 \\ f(2) = a_3 \cdot 2^3 + a_2 \cdot 2^2 + a_1 \cdot 2^1 + a_0 \cdot 2^0 \end{cases}$$

4 equations

4 unknowns

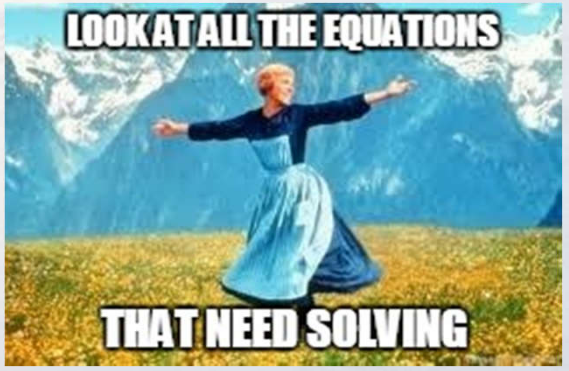
At 2D, bicubic interpolation uses 16 neighbors to determine the interpolated value:

$$f(x,y) = \sum_{i=0}^{3}\sum_{i=0}^{3} a_{ij}x^i y^j$$

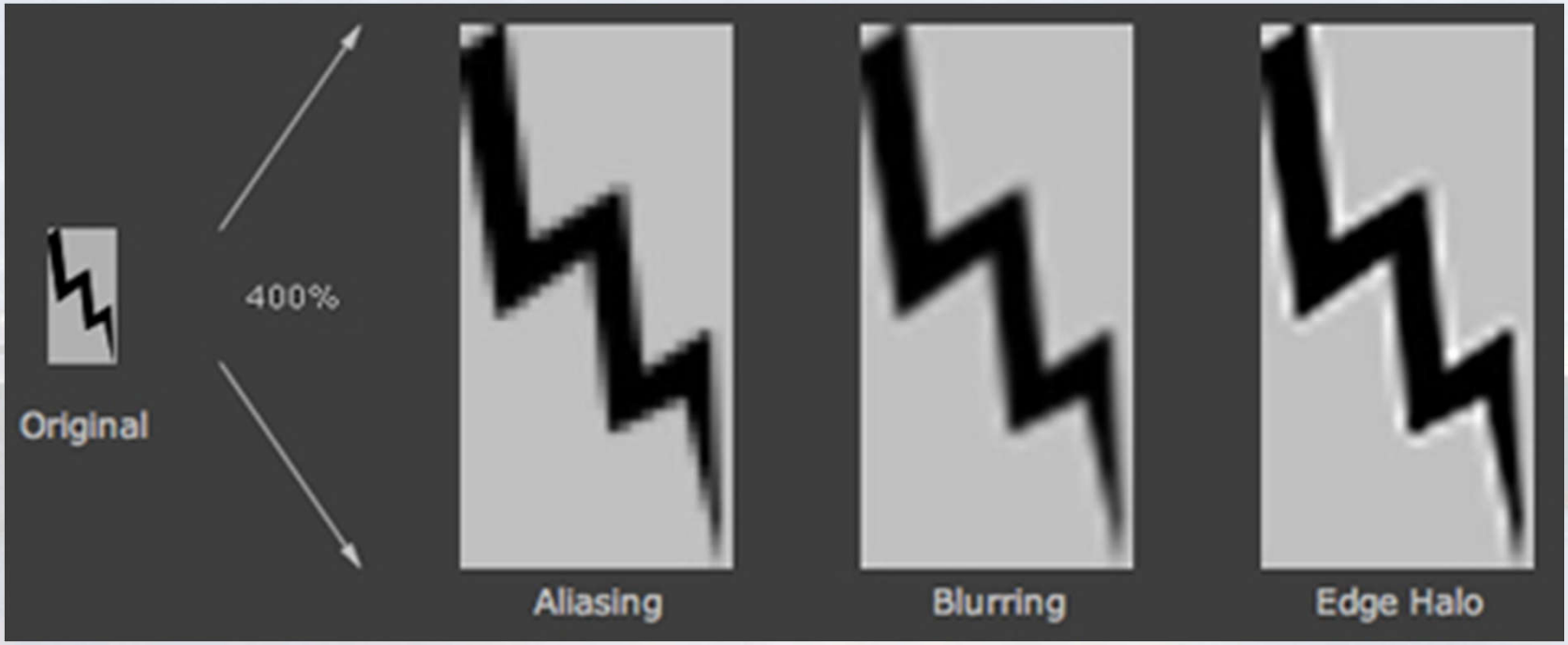
where the 16 coefficients are determined from the 16 equations. Bicubic interpolation is widely used in many commercial editing programs; e.g. Adobe Photoshop and Corel Photo Paint.

PS: There is also Sinc (Lanczos) interpolation.

It's similar to bicubic in practice.



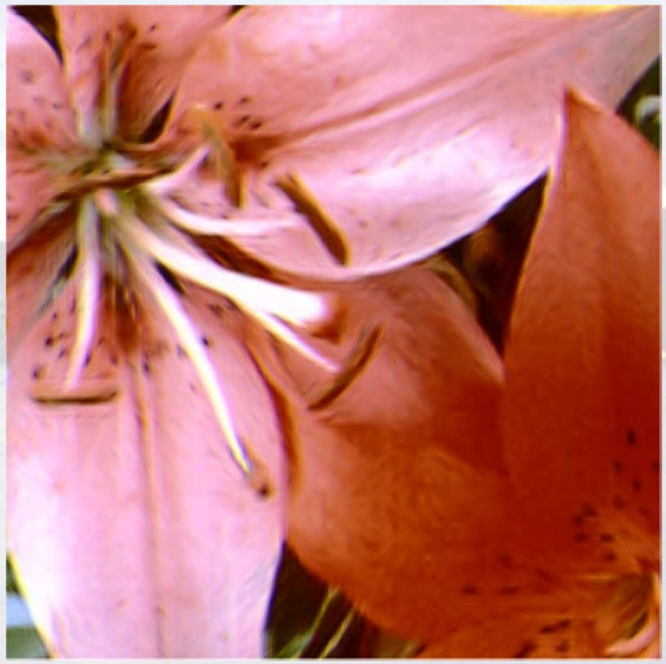
Bilinear or bicubic, they both introduce artifacts into the images and blur their edges.



Edge-directed interpolation techniques have been also reported; They often provide the best SNR performances.

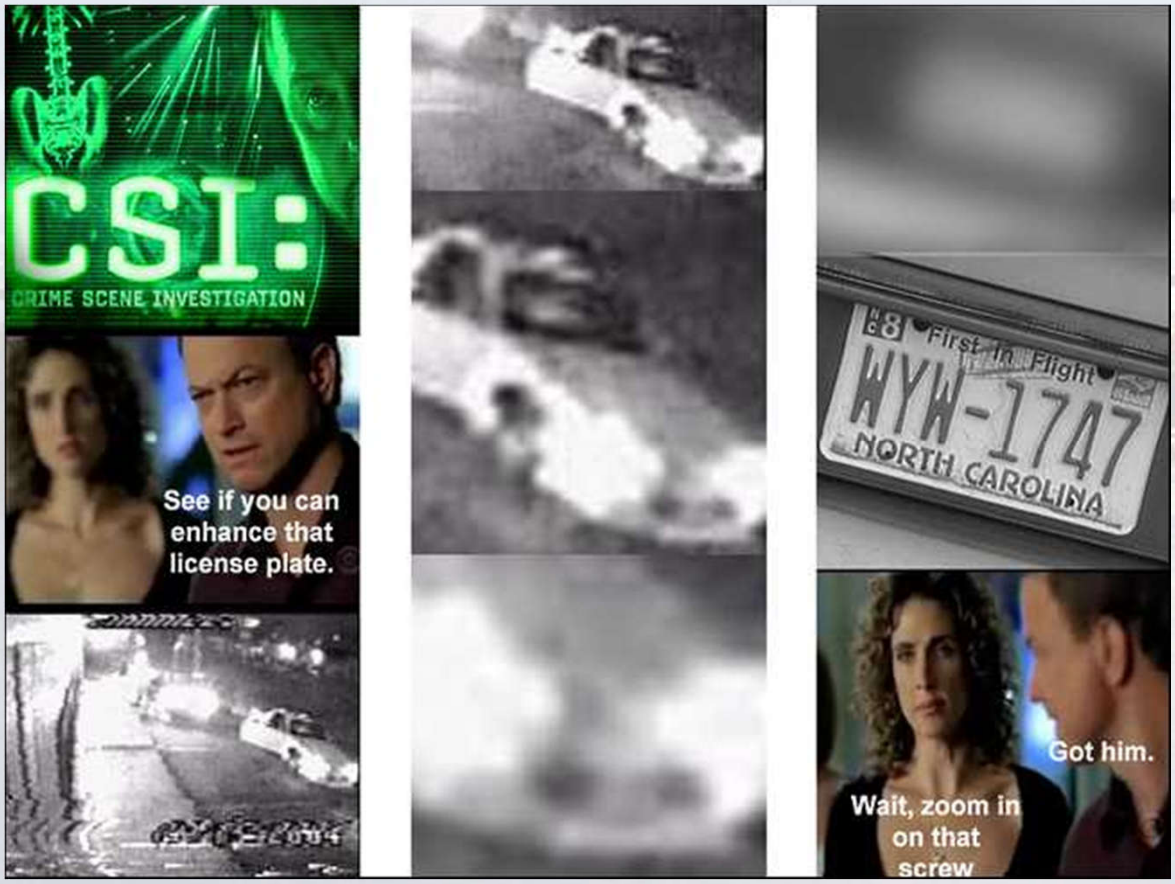
The main idea is not to smooth perpendicularly to edges but to smooth parallel to edges, and thus preserve edges.

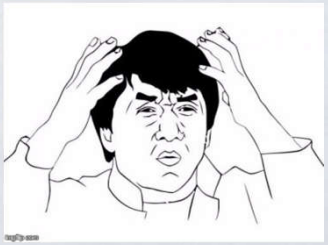
Bicubic



EDI (Li & Orchard, 2000)

Optical zoom (or **true zoom**) is realized through the lens of a camera, and preserves sharpness and resolution, by providing a closer view of the subject. Digital zoom cannot introduce new features into your images, at best it preserves existing ones.





Point processing

Read sections 2.6, 3.1-3.3 from DIP.