Medical Imaging Research Laboratory
www.creatis.insa-lyon.fr

Image pre-processing

Patrick Clarysse, DR CNRS,
CREATIS, UMR CNRS 5220, Inserm 1206, Lyon.
Contact: patrick.clarysse@creatis.insa-lyon.fr

Created: October 2020, Modified October 2022

Operations

- Image restoration and correction
  - **Intensity inhomogeneity** correction: usually refers to slow intensity variations over the image domain
  - Image denoising. **Noise** = basic signal distortion which hinders the process of image observation and information extraction. Types of noise:
    - Additive white Gaussian noise (acquisition and transmission)
    - Impulse (salt and pepper)
    - Quantization
    - Poisson
    - Speckle

Restoration: Image denoising

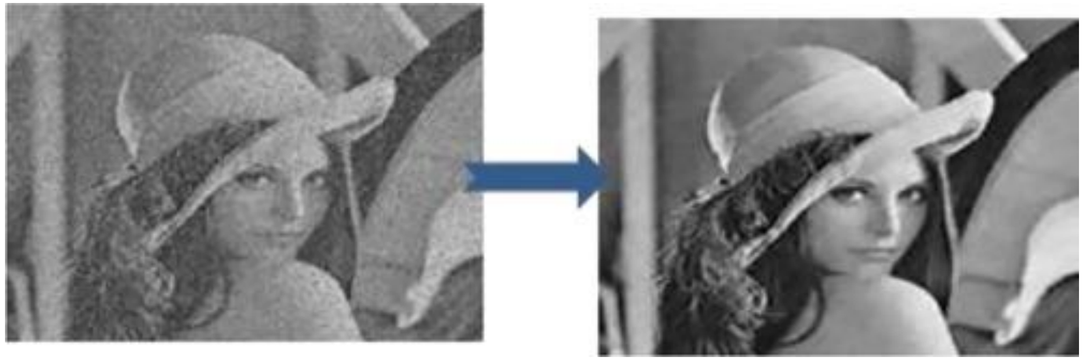


Image denoising

- Formulation

$$f(\mathbf{x}) = u(\mathbf{x}) + n(\mathbf{x})$$

Where $f(\mathbf{x}), u(\mathbf{x})$ and $n(\mathbf{x})$ are respectively the observed, the true and the noise at location $\mathbf{x} = (x, y)$

- Methods
  - **Linear Translational Invariant Filtering**: averaging or mean or box filtering, Gaussian filter, Weiner filter, Least Mean Square filters, Bilateral filter,
  - **Non-linear filtering**: Median, Anisotropic Diffusion, Rank filter, Steering Kernel Regression (SKR), Metric Steering Kernel Regression and Trained filters

Simplest approach

- Linear Translational Invariant filtering is averaging or mean or box filtering which generates output at each pixel as the average of neighbouring pixels in a given window

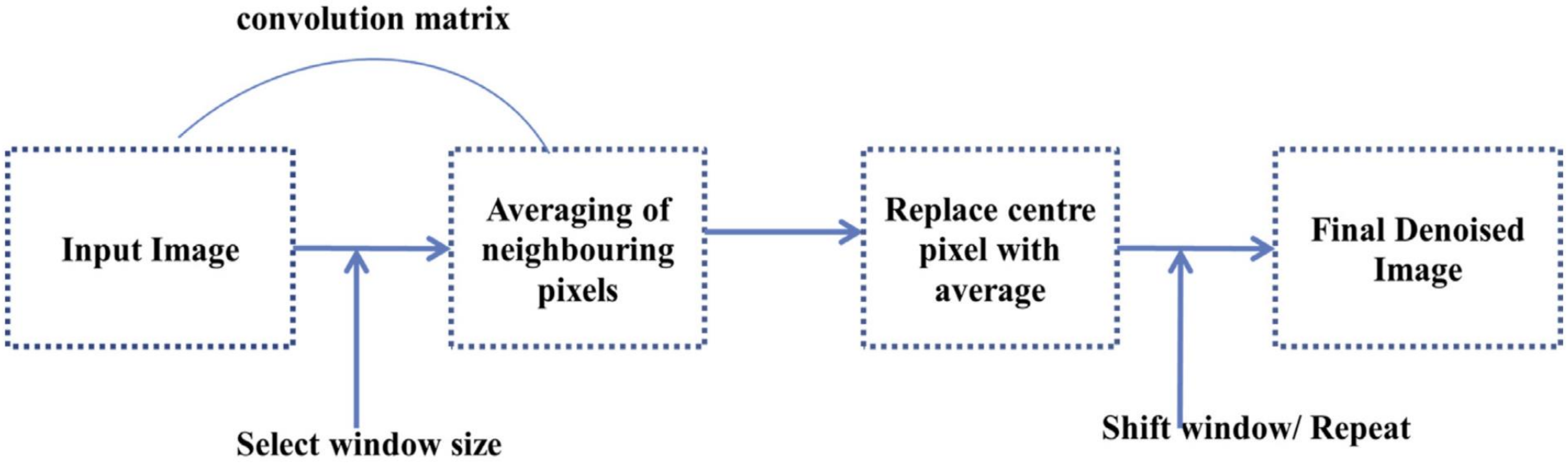
217	191	191
217	89	127
127	242	217

→

217	191	191
217	180	127
127	242	217

(217+191+191+217+89+
127+127+242+217)/9
=1618/9=180

General principle: averaging filter

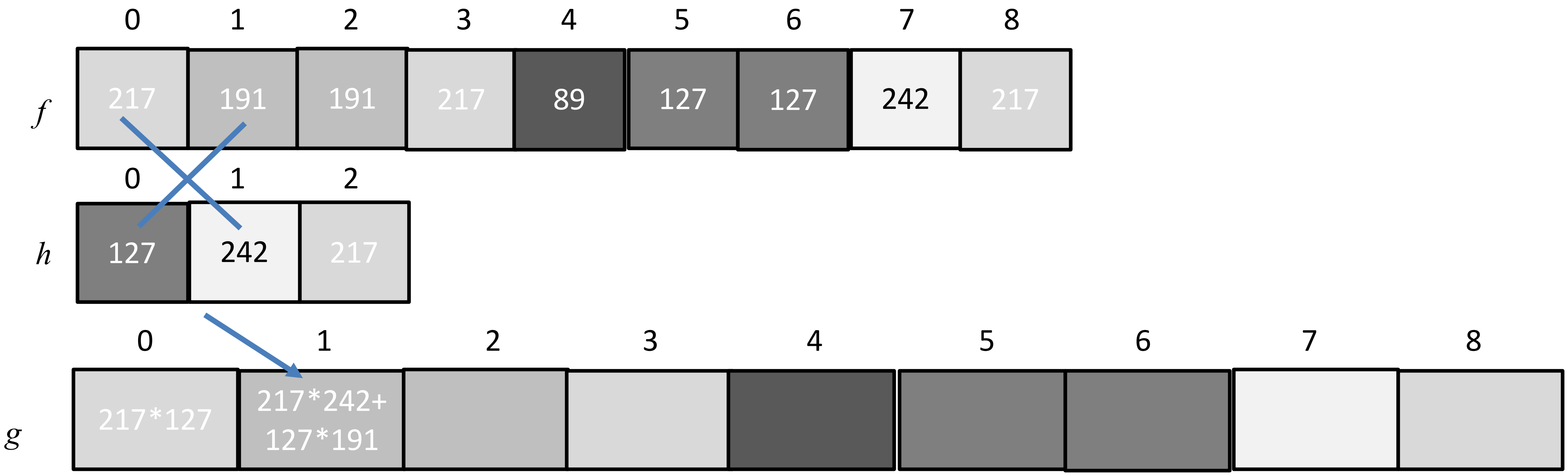


Convolution of f and h defined on the set Z of integers

$$(f * h)(k) = \sum_j f(j)\, h(k-j) = \sum_j f(k-j)\, h(k)$$

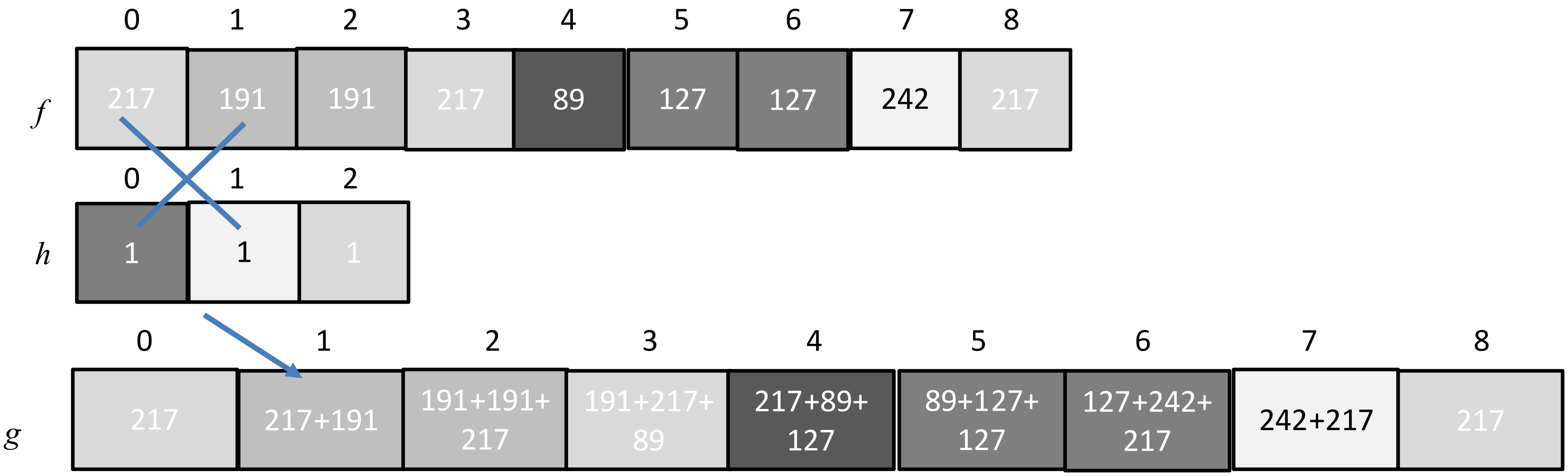
$$g(0) = f(0) * h(0)$$
$$g(1) = f(0) * h(1) + f(1) * h(0)$$
$$g(2) = f(0) * h(2) + f(1) * h(1) + f(2) * h(0)$$
.
.
$$g(n) = f(0) * h(n) + f(1) * h(n-1) + f(2) * h(n-2) + \cdots + f(n) * h(0) \ldots$$
$$g(2n) = f(n) * h(n)$$

With f and h of size n



$$g = (f * h)(k) = \sum_j f(j)\, h(k-j) = \sum_j f(k-j)\, h(k)$$

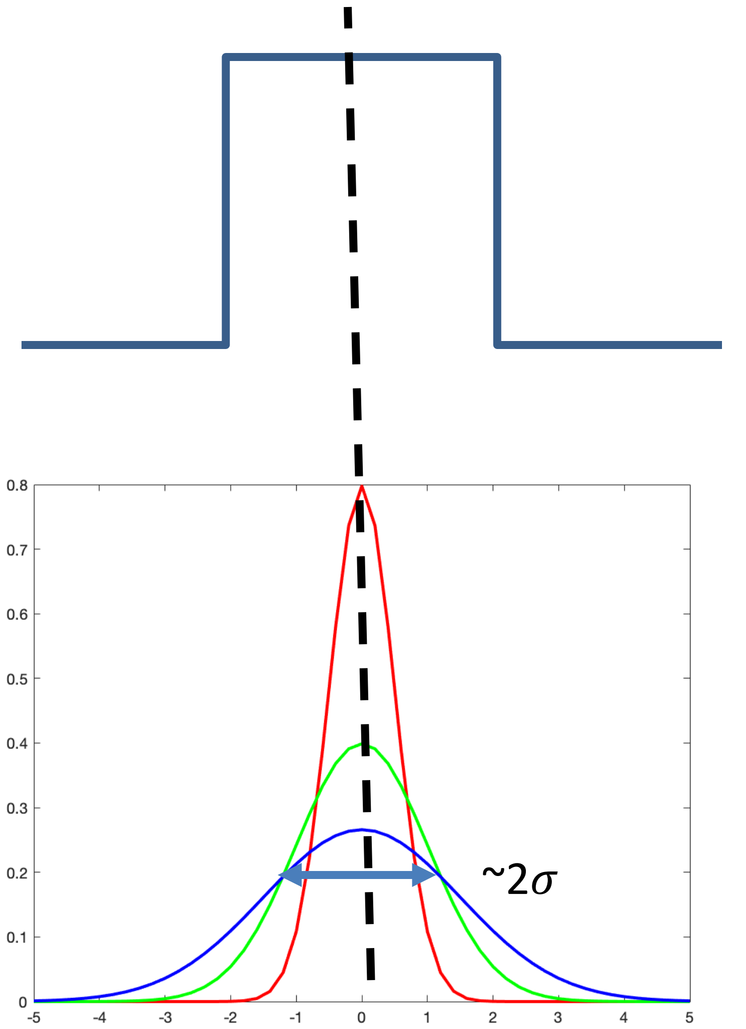
Special cases



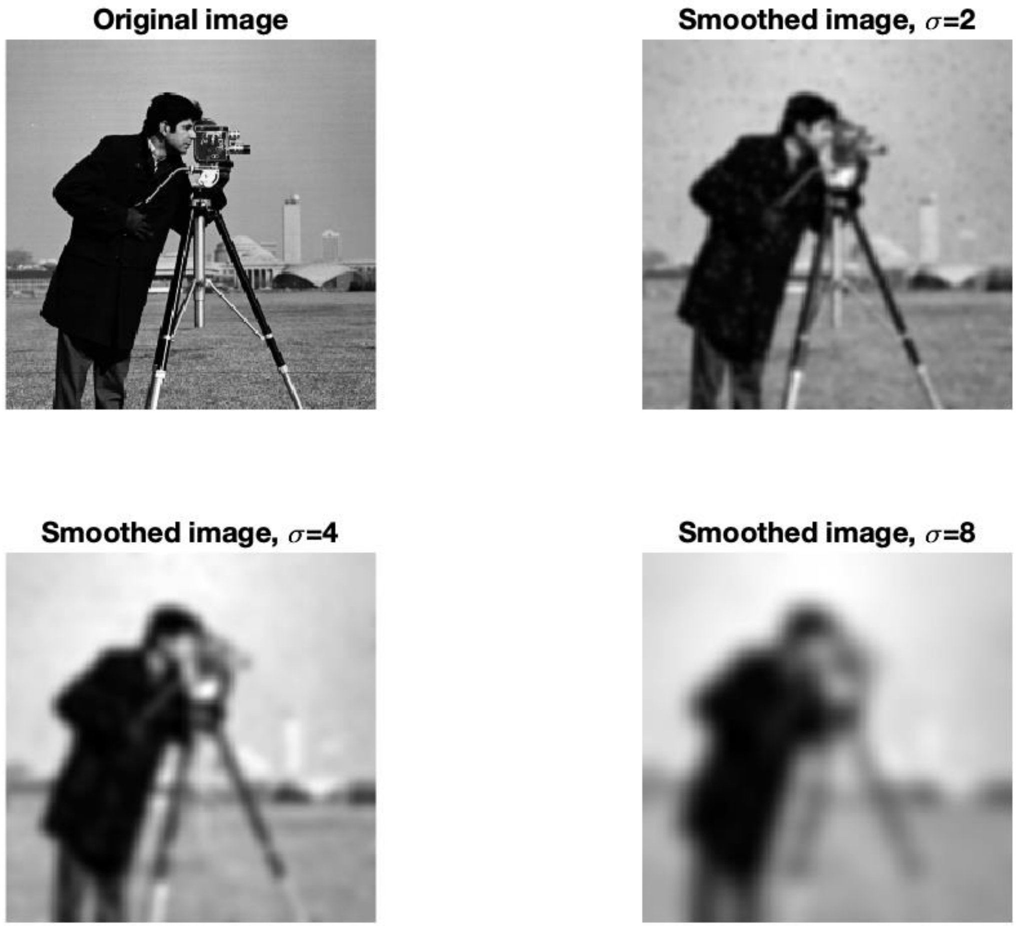
Special cases

Mean filter

Gaussian filter



Gaussian filtering



Denoising examples

Salt & pepper

noice corrupted

Average filter

Gaussian filter

Median filter

Well adapted to this kind of noise (as well as morphological filters)

Generation of noise

- Salt & pepper

```
[rows, columns] = size(im);
imnoise = im;

for i = 1:rows %for loops iterate through every pixel
  for j = 1:columns
     noise_check = randi(noise_percent); %creates a random number between 1 and noise_percent
     if noise_check == noise_percent    %if the random number = noise_percent (1/noise_percent
chance of any given pixel being noisy)
        noise_value = randi(256);    %creates a random noise value to replace the pixel
        imnoise(i,j) = noise_value; %replaces the original pixel value with the random noise
     end
  end
end
```

original image

noice corrupted

The noise is Gaussian (normally) distributed with a mean of zero and standard deviation of 25.

Mean filtering

Gaussian filtering

Median filtering

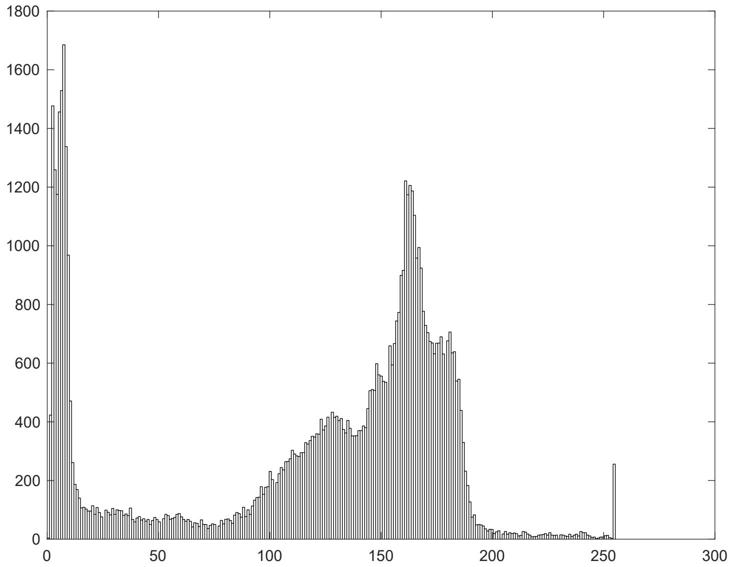
Image enhancement

- Contrast adjustment
- Image filtering
- Morphological operations
- Debluring

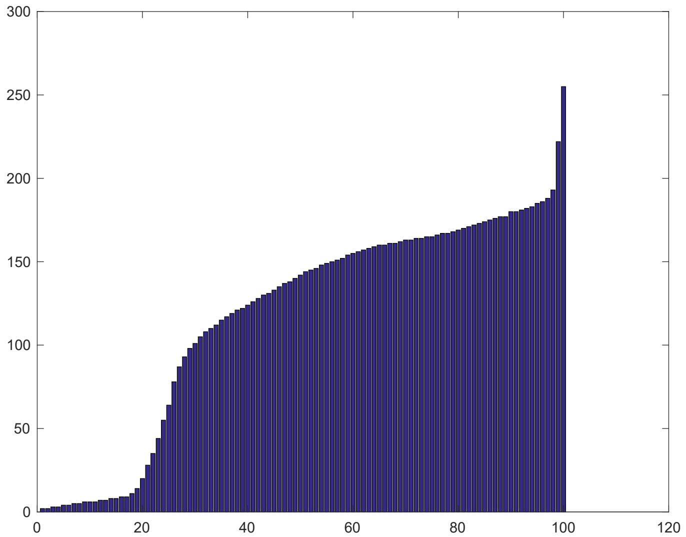
Contrast adjustment: stretching

- Also called normalization (linear transformation)
- Define the output range: $[R_{min,} R_{max}]$ (e.g. 0-255)
- Given an image I, search for the minimum and maximum intensities: $I_{min,} I_{max}$
- The scale change is defined by the pixel per pixel operation:
- $P_{out} = (P_{in} - I_{min})(R_{max} - R_{min} / I_{max} - I_{min}) + R_{min}$

- **Remark**: The problem with this is that a single outlying pixel with either a very high I_{max} or very low I_{min} value can severely affect the result and this could lead to very unrepresentative scaling. Therefore a more robust approach is to first take a histogram of the image, and then select I_{min} and I_{max} at, say, the 5th and 95th percentile in the histogram (that is, 5% of the pixel in the histogram will have values lower than I_{min}, and 5% of the pixels will have values higher than I_{max}). This prevents outliers affecting the scaling so much.

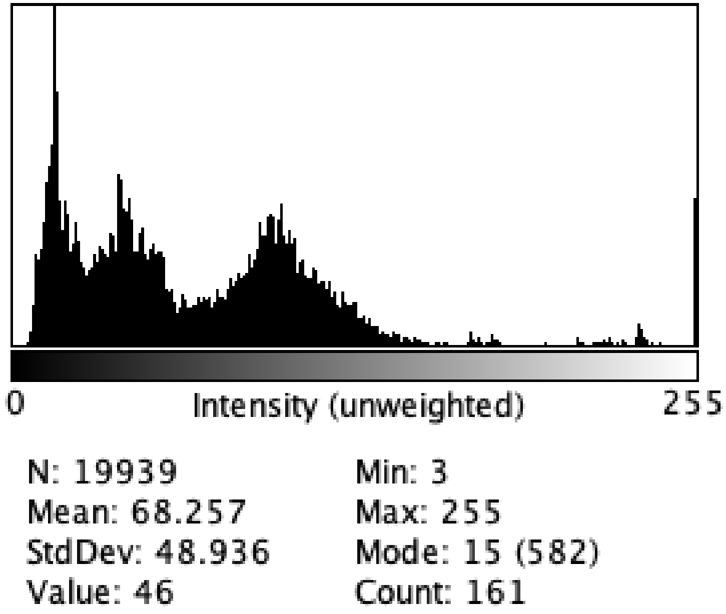


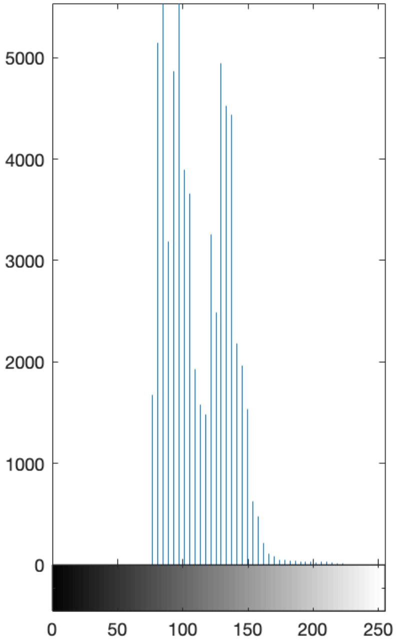
Histogram

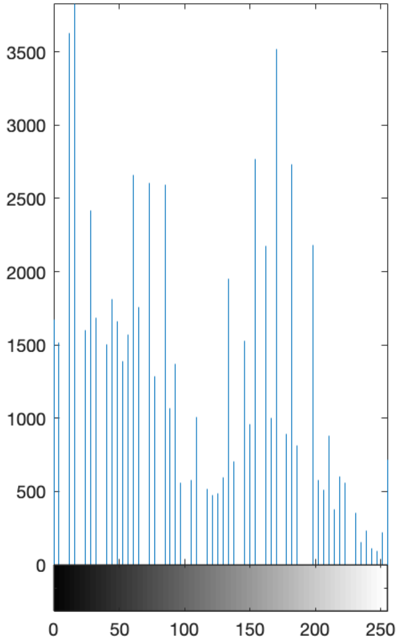


Percentile plot

Contrast adjustment: stretching





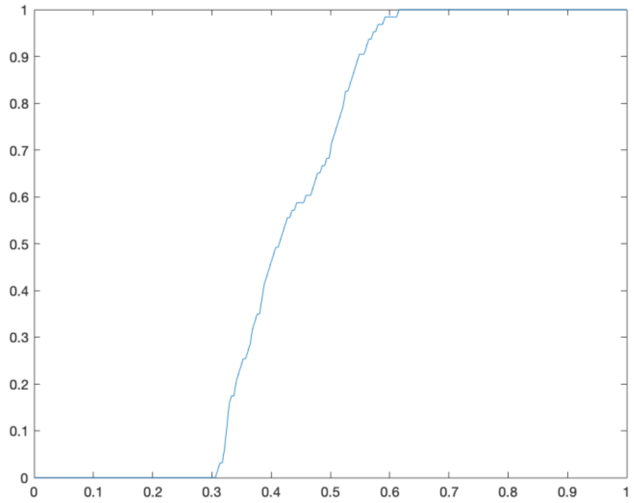


Stat I: 74, 224,110.3, 109
Stat J: 0, 255, 98.7, 94

Histogram equalization

➔ transforming the intensity values so that the histogram of the output image approximately matches a specified histogram

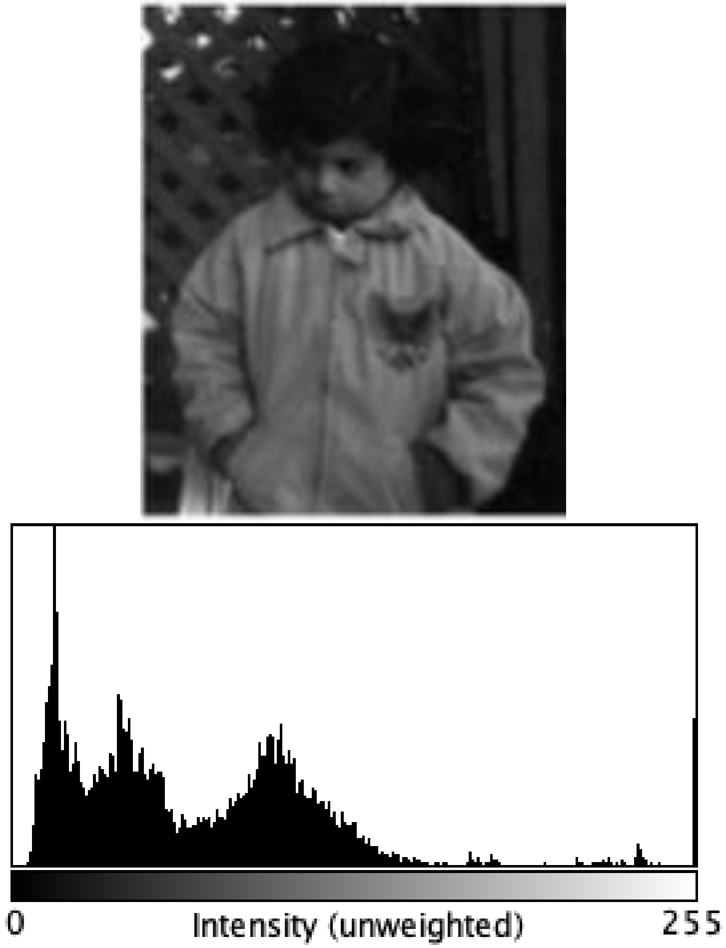
Example: match to a flat histogram



Output values

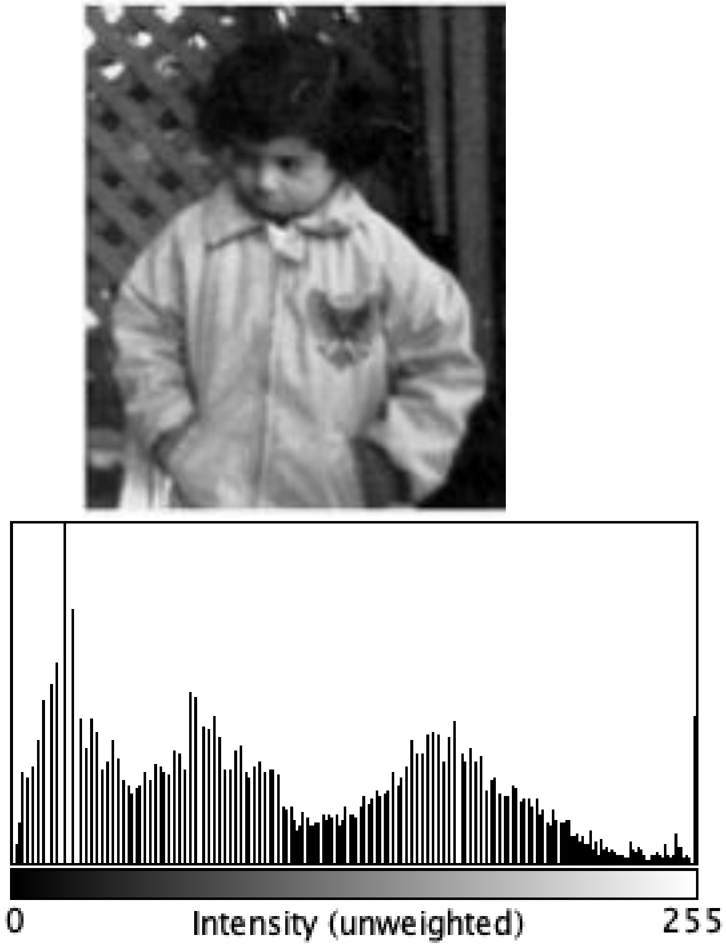
Input values

Contrast adjustment: Histogram equalization



N: 19939
Mean: 68.257
StdDev: 48.936
Value: 46

Min: 3
Max: 255
Mode: 15 (582)
Count: 161



N: 19939
Mean: 107.219
StdDev: 68.001
Value: ---

Min: 0
Max: 255
Mode: 19 (582)
Count: ---