



浙江大学爱丁堡大学联合学院

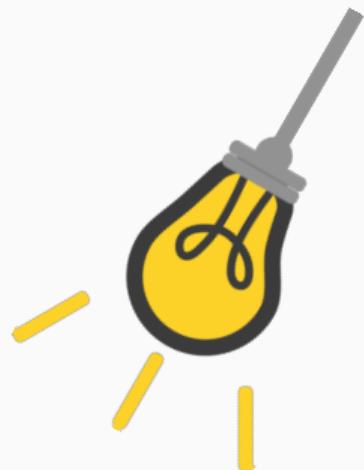
ZJU-UoE Institute

Image histograms

Nicola Romanò - nicola.romano@ed.ac.uk

Learning objectives

- Define and produce an image histogram
- Use histograms to interpret image quality
- Apply point operations to manipulate image histograms



Histograms

Brightness and contrast

Point operations

What is a histogram?

"A histogram is an approximate representation of the distribution of numerical data." ([Wikipedia](#))

Given a variable x , in the interval $[x_{min}; x_{max}]$, we divide this interval (or a suitably larger one) into non-overlapping **bins** and count the number of values of x falling into each bin.

Generally, we choose bins to be of equal size.

Histogram example

Given $x = 5, 7, 10, 22, 35, 88, 26, 74, 22, 95$

We can choose the interval $[0; 99]$, and divide it into bins of size 10 $[0 : 9][10; 19] \dots [90 : 99]$

Histogram example

Given $x = 5, 7, 10, 22, 35, 88, 26, 74, 22, 95$

We can choose the interval $[0; 99]$, and divide it into bins of size 10 $[0 : 9][10; 19] \dots [90 : 99]$

We now count the occurrences of x in each bin.

$$[0 : 9] \rightarrow 2$$

$$[10 : 19] \rightarrow 1$$

...and so on

Histogram example

Given $x = 5, 7, 10, 22, 35, 88, 26, 74, 22, 95$

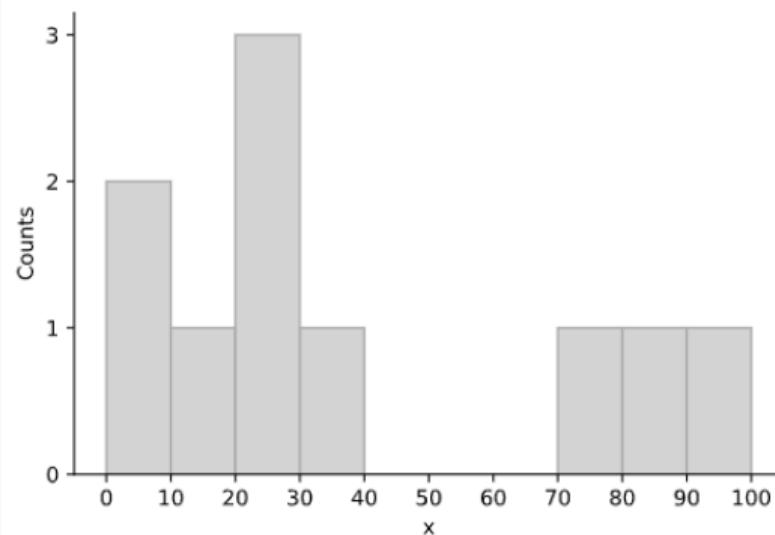
We can choose the interval $[0 : 99]$, and divide it into bins of size 10 $[0 : 9][10 : 19] \dots [90 : 99]$

We now count the occurrences of x in each bin.

$$[0 : 9] \rightarrow 2$$

$$[10 : 19] \rightarrow 1$$

...and so on



$x =$

5	7	10	22	35	88	26	74	22	95
---	---	----	----	----	----	----	----	----	----

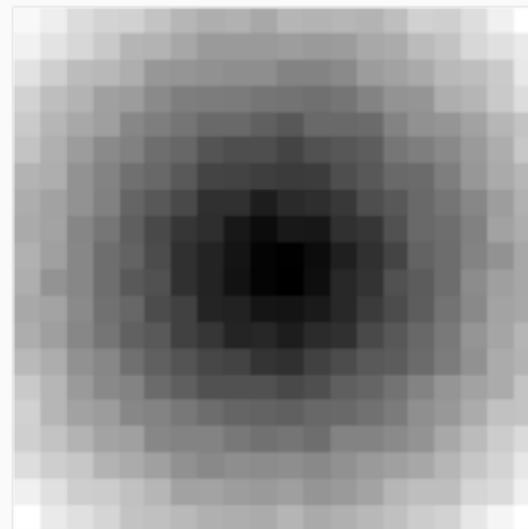
Histogram of an image

Images are just matrices, so we can count the occurrences of each pixel value in the image.

Example

An 8-bit image, 20 x 20

```
> print(img.shape)
(20, 20)
> print(np.min(img))
13
> print(np.max(img))
177
```



Histogram of an image

Images are just matrices, so we can count the occurrences of each pixel value in the image.

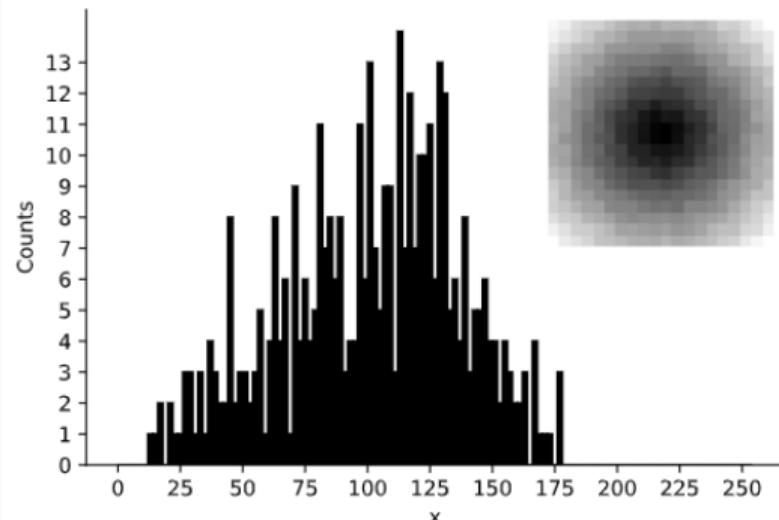
Example

An 8-bit image, 20 x 20

```
> print(img.shape)
(20, 20)
> print(np.min(img))
13
> print(np.max(img))
177
```

Maximum dynamic range [0, 255]

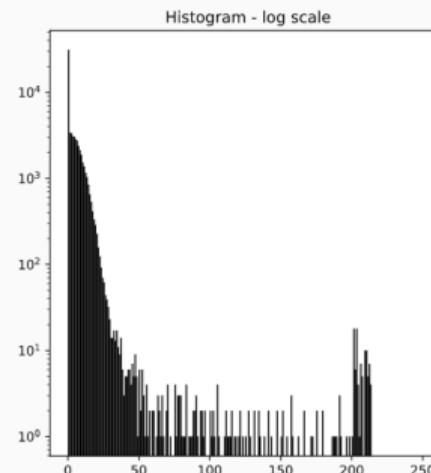
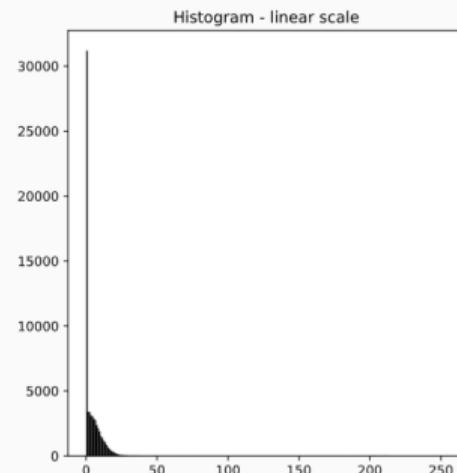
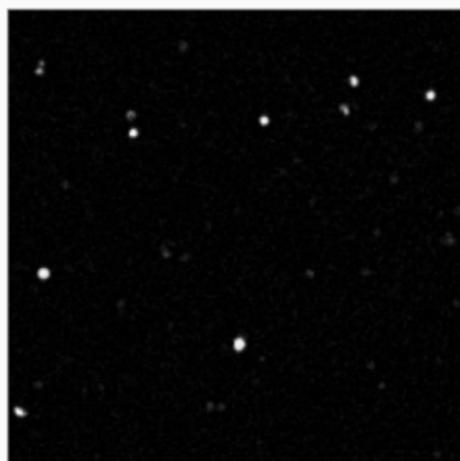
We count the occurrences of each pixel with intensity between 0 and 255.



What can you tell about the image from looking at the histogram?

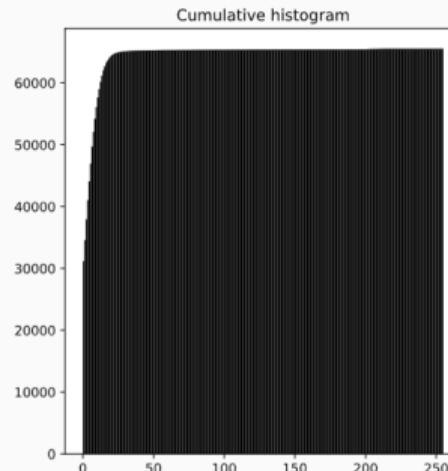
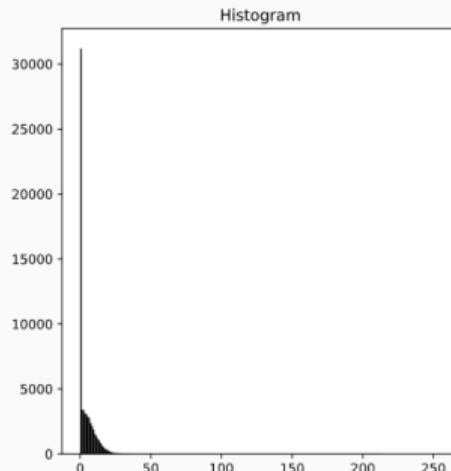
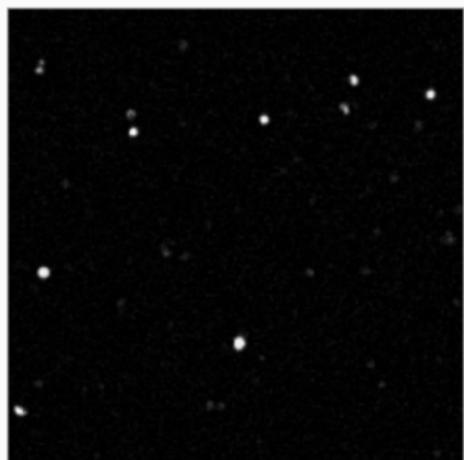
Logarithmic scale

We can plot the histogram on a logarithmic scale. Useful when images contain many pixels of a specific intensity (e.g. cells on a black background).



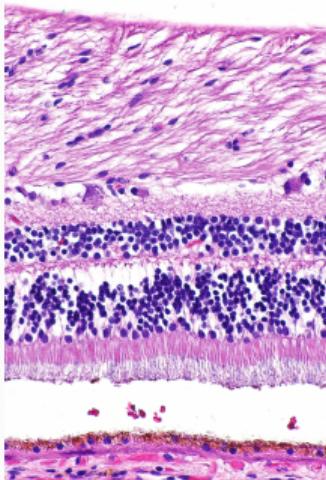
Cumulative histogram

Alternatively, we can plot a cumulative histogram (see later for uses). Each bar is the count of pixels with intensity between 0 and the corresponding bin.

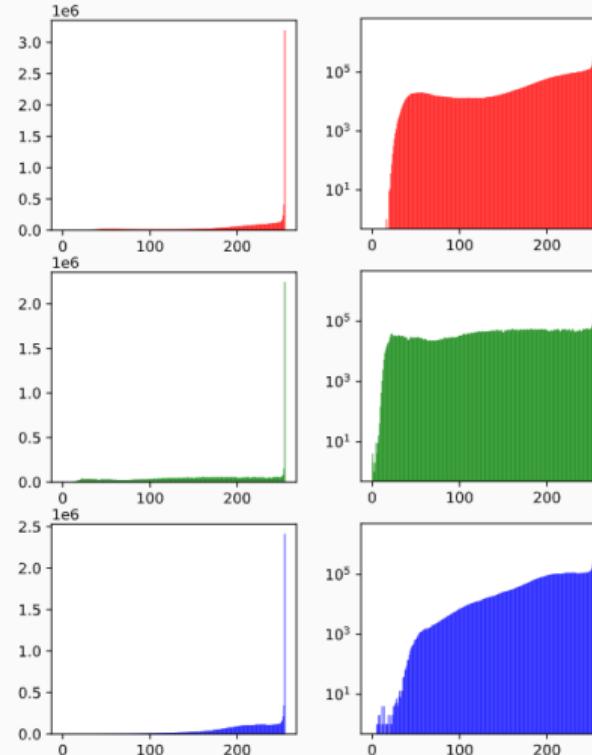


Histograms of RGB images

For RGB (or multi-channel) images, it makes sense to have a single histogram per channel



H&E staining of retina (cell nuclei stained blue-purple
and extracellular material stained pink)
Librepath - CC BY-SA 3.0



RGB histogram - left linear, right logarithmic.

Histograms are destructive

It is important to remember that producing a histogram is a many-to-one operation! We can generate a histogram from an image, but cannot recreate the image from the histogram.



These three images have the same histogram!

Drawing a histogram using Matplotlib

The Matplotlib `hist` function makes it very easy to plot a histogram

```
import matplotlib.pyplot as plt
img = io.imread("cells.tif")

fig, ax = plt.subplots(1, 2, figsize=(12, 6))
# Linear
ax[0].hist(img.ravel(), bins=range(255), color="black")
ax[0].set_title("Histogram - linear scale")
```

The `bins` parameter accepts either a sequence defining the edges of the bins, or a single value with the number of desired bins.

Drawing a histogram using Matplotlib

The Matplotlib `hist` function makes it very easy to plot a histogram

```
import matplotlib.pyplot as plt
img = io.imread("cells.tif")

fig, ax = plt.subplots(1, 2, figsize=(12, 6))
# Linear
ax[0].hist(img.ravel(), bins=range(255), color="black")
ax[0].set_title("Histogram - linear scale")
# Log
ax[1].hist(img.ravel(), bins=range(255), color="black", log=True)
ax[1].set_title("Histogram - log scale")
plt.show()
```

The `bins` parameter accepts either a sequence defining the edges of the bins, or a single value with the number of desired bins.

Outline

Histograms

Brightness and contrast

Point operations

Brightness

The **brightness** of an image is the average intensity of all its pixels

$$B(I) = \frac{1}{w * h} \sum_{r=1}^h \sum_{c=1}^w I(r, c)$$

Where w and h are the width and height of the image, and $I(r, c)$ is the intensity of the pixel at row r and column c .

Brightness

The **brightness** of an image is the average intensity of all its pixels

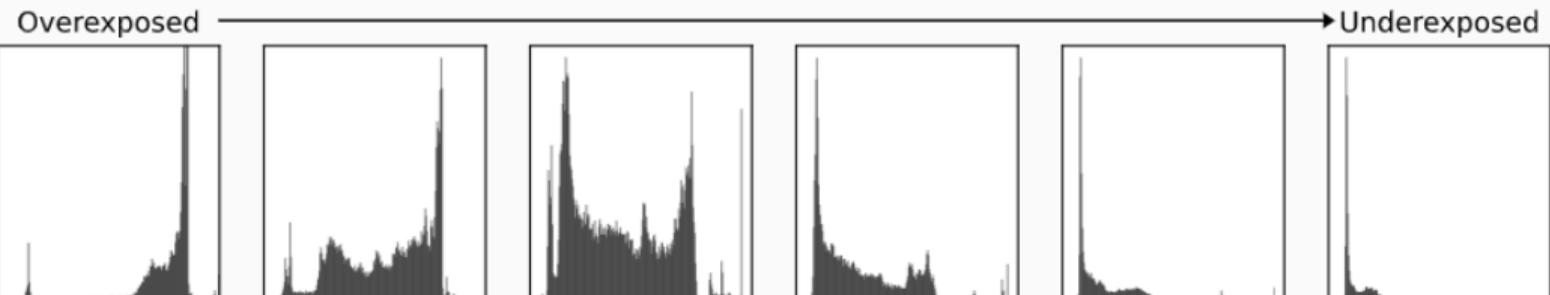
$$B(I) = \frac{1}{w * h} \sum_{r=1}^h \sum_{c=1}^w I(r, c)$$

Where w and h are the width and height of the image, and $I(r, c)$ is the intensity of the pixel at row r and column c .

Brightness example

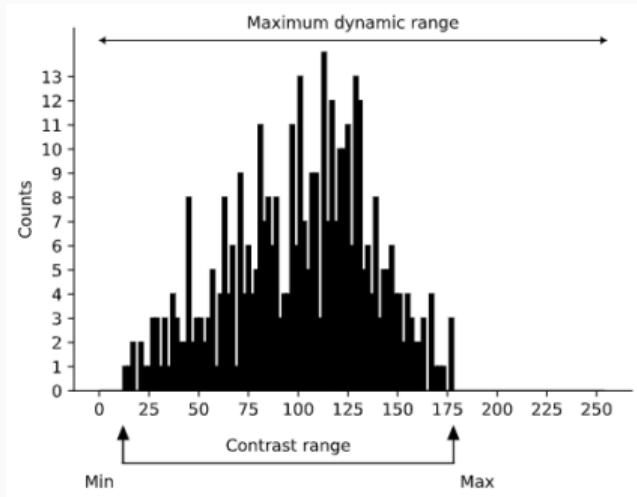


Same X-ray, decreasing brightness - Image from Veldkamp et al., 2009

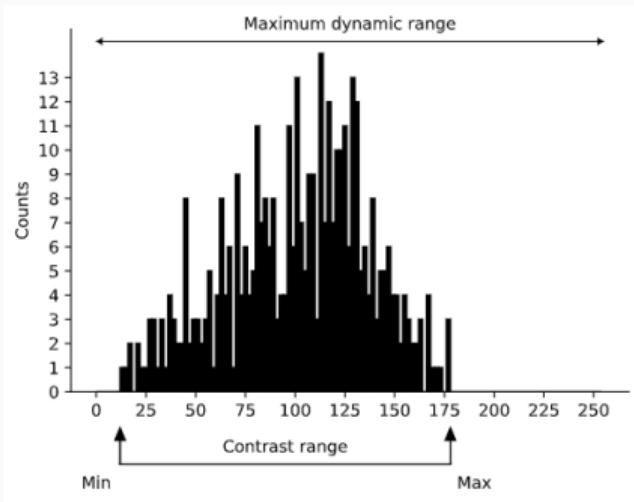


Contrast

Contrast is the difference in luminance or colour that makes an object distinguishable (Wikipedia). It measures the relationship between light and dark pixels.



Contrast



Contrast is the difference in luminance or colour that makes an object distinguishable (Wikipedia). It measures the relationship between light and dark pixels. Many different definitions in the literature (see Peli, *J. Opt. Soc. Am. A*, 1990)

Weber contrast (range $0 \rightarrow \infty$)

$$\frac{I_{obj} - I_{bg}}{I_{bg}}$$

Michelson contrast (range $0 \rightarrow 1$)

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

RMS contrast

$$\sqrt{\frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})^2}$$

where x is the normalised intensity $0 \leq x \leq 1$

Outline

Histograms

Brightness and contrast

Point operations

Point operations and neighborhood operations

Manipulating pixel intensities allows enhancement of images.

Two types of operations:

- Point operations - Change pixel intensity based only on its value
- Neighborhood operations - Change pixel intensity based on the intensity of the pixel and its neighbours.

Point operations and neighborhood operations

Manipulating pixel intensities allows enhancement of images.

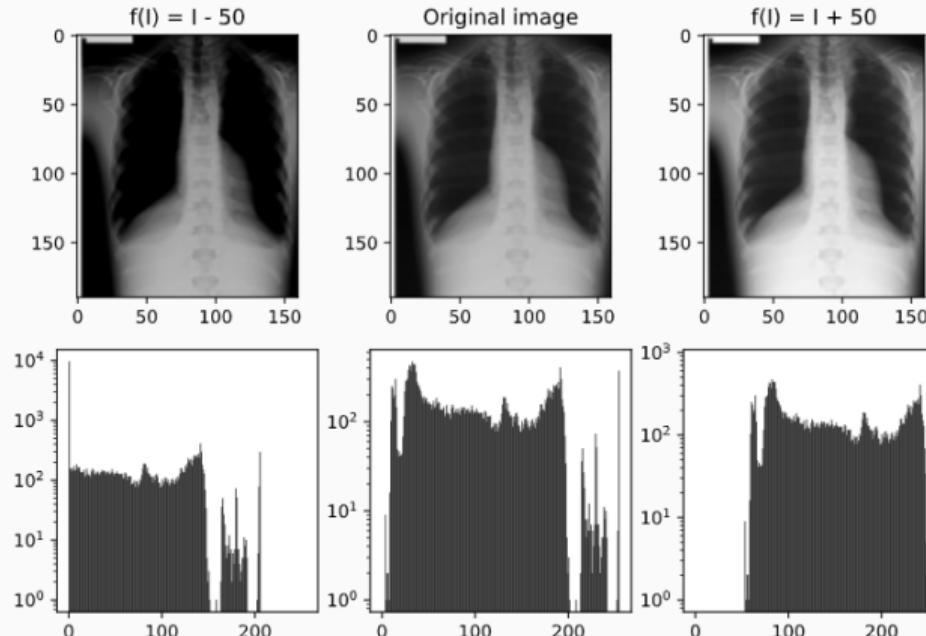
Two types of operations:

- Point operations - Change pixel intensity based only on its value
- Neighborhood operations - Change pixel intensity based on the intensity of the pixel and its neighbours.

So, the resulting intensity I' of a point operation is $I'_{(x,y)} = f(I_{x,y})$.

Manipulating brightness

We can easily change an image brightness by using a point operation $f(I_{x,y}) = I_{x,y} + \text{offset}$, which shifts the image histogram to the left or to the right.



What is happening to the histograms?

Example code for brightness manipulation

```
from skimage import img_as_float, img_as_ubyte

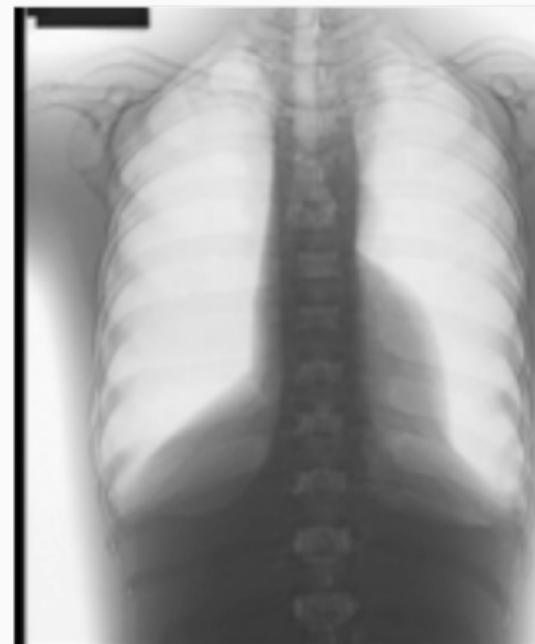
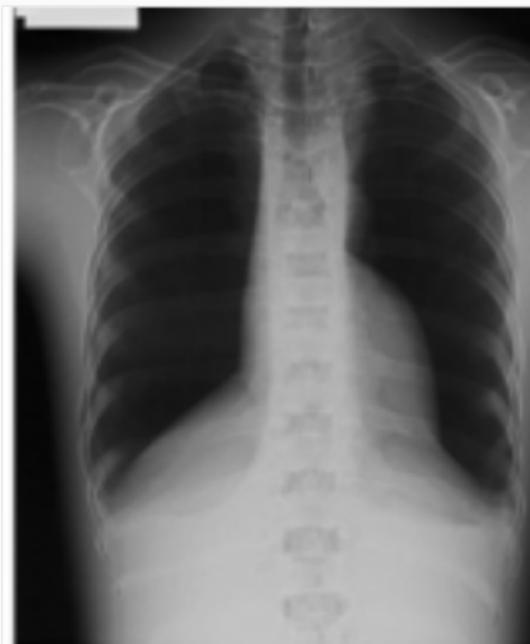
def change_brightness(img, offset):
    """
    Changes the brightness of an image
    img: the image
    offset: the brightness offset to apply
    Returns: the modified image
    """

    img2 = img_as_float(img)
    img2 += offset/255
    img2 = np.clip(img2, 0, 1)
    return (img_as_ubyte(img2))

img_2 = change_brightness(img, 10)
```

Your turn!

Can you think of a point operation to invert an image? What happens to the histogram?
Write your own Python function to do this!



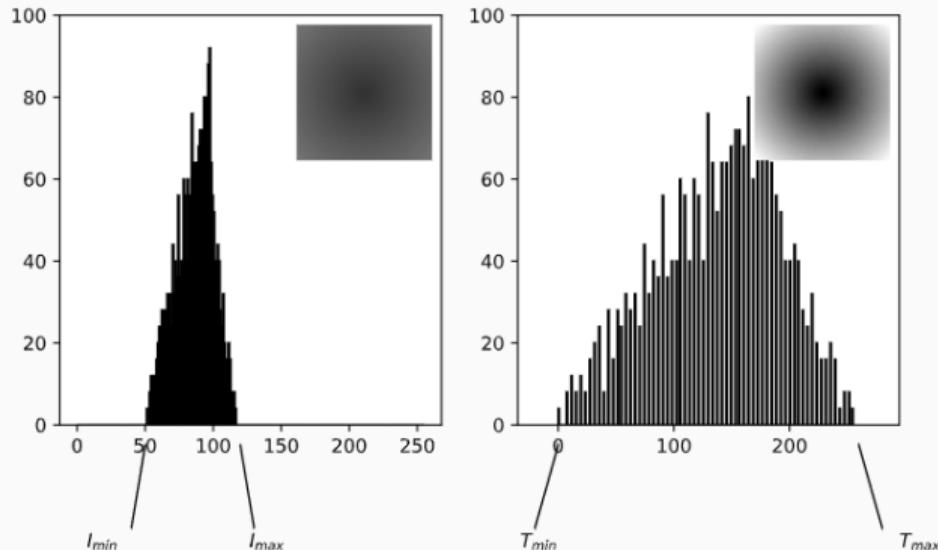
Manipulating contrast - histogram stretching

A simple way of increasing the contrast of an image is to stretch its histogram.

Simplest form:

$$I' = (I - I_{min}) \frac{T_{max} - T_{min}}{I_{max} - I_{min}} + max$$

Where I_{min} and I_{max} are the minimum and maximum intensity of the image, and T_{min} and T_{max} are the minimum and maximum intensity of the target histogram.



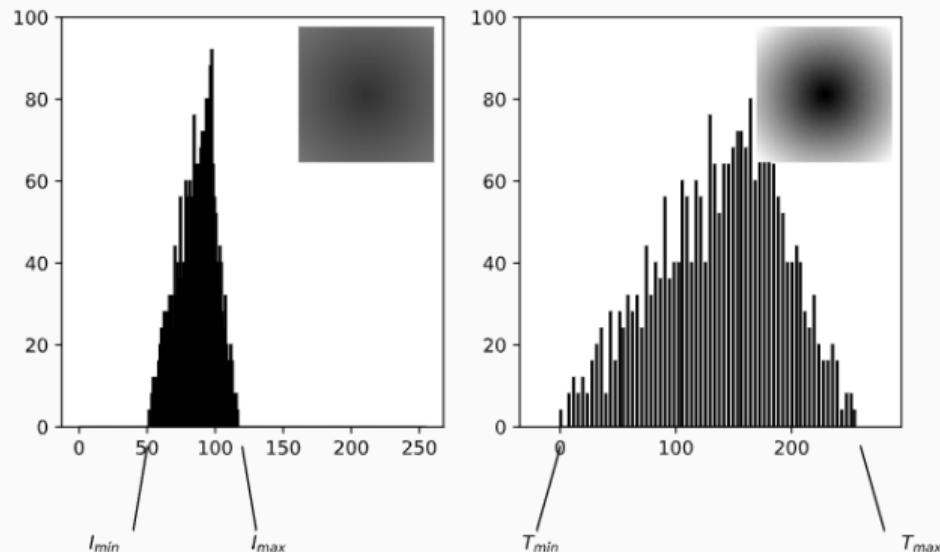
Manipulating contrast - histogram stretching

A simple way of increasing the contrast of an image is to stretch its histogram.

Simplest form:

$$I' = (I - I_{min}) \frac{T_{max} - T_{min}}{I_{max} - I_{min}} + max$$

Where I_{min} and I_{max} are the minimum and maximum intensity of the image, and T_{min} and T_{max} are the minimum and maximum intensity of the target histogram.



This calculation can become meaningless if we have even a single very light or dark pixel, so we can use the 2nd and 98th percentiles instead of I_{min} and I_{max} values.

Histogram stretching in Python

```
import numpy as np
from skimage.exposure import rescale_intensity

img_rescale = rescale_intensity(img, in_range=(0, 100), out_range=(0, 255))
```

Histogram stretching in Python

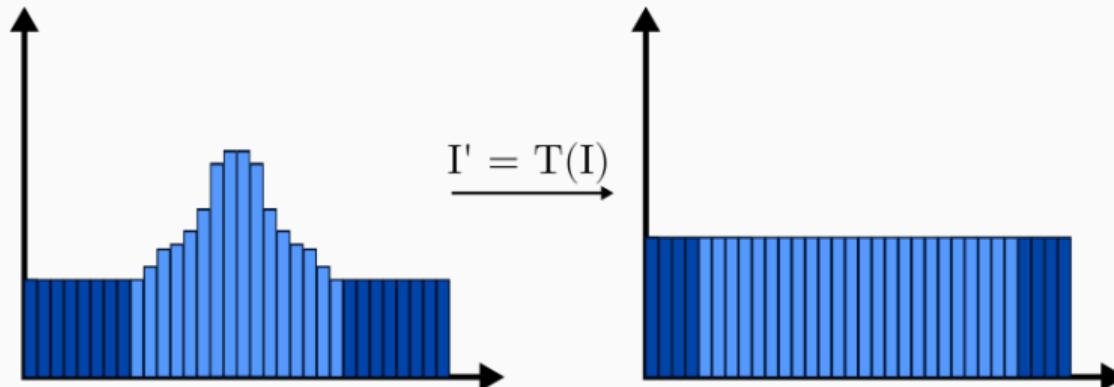
```
import numpy as np
from skimage.exposure import rescale_intensity

img_rescale = rescale_intensity(img, in_range=(0, 100), out_range=(0, 255))
# Or, use the percentile version:
# Calculate the 2nd and 98th percentiles
p2, p98 = np.percentile(img, (2, 98))
img_rescale = rescale_intensity(img, in_range=(p2, p98), out_range=(0, 255))
```

Histogram equalization

Histogram equalization increases contrast by manipulating pixel intensities.

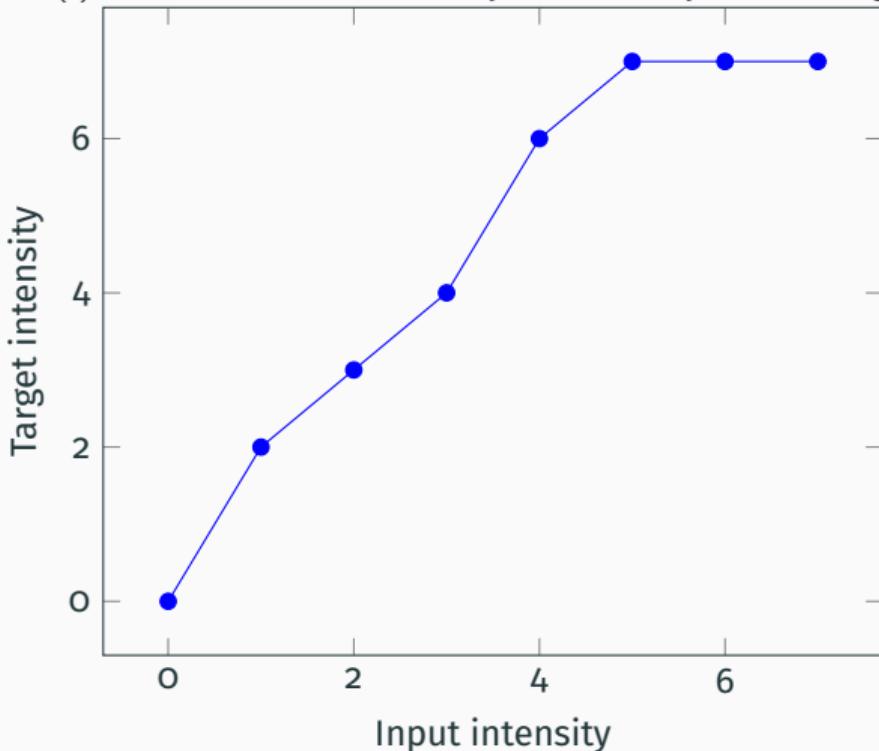
The idea is to generate a **histogram transfer function** $T(I)$ that will map the image intensities to a target, uniform histogram.



Choosing $T(I)$

How do we choose $T(I)$?

$T(I)$ should be monotonically (not strictly) increasing, to preserve the intensity rank of the pixels.

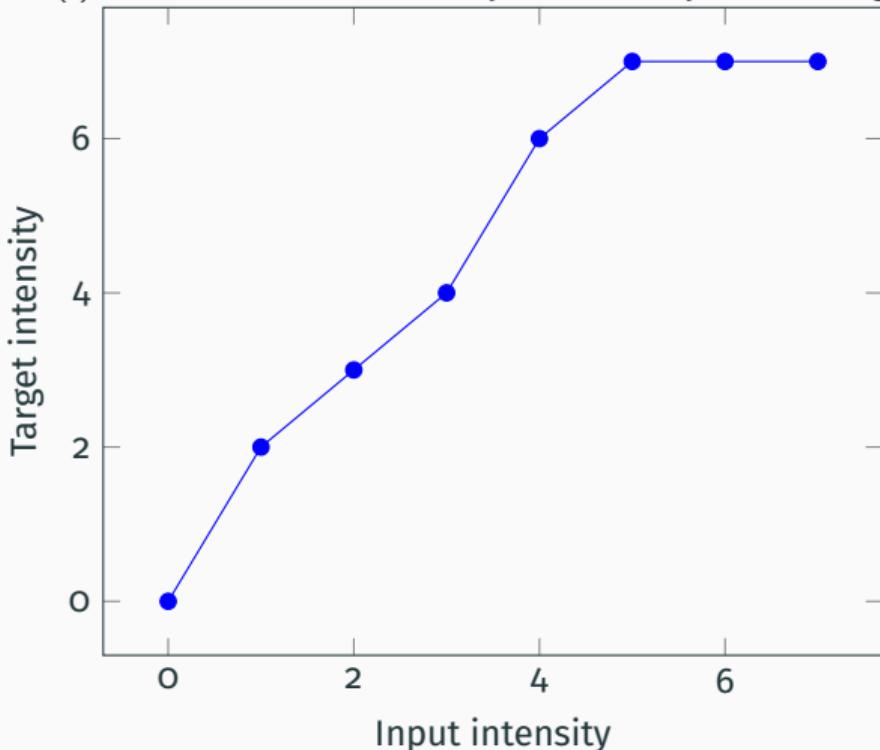


In this example we would change any pixel of the original image with intensity 3 to intensity 4, any with intensity 6 to intensity 7 and so on.

Choosing $T(I)$

How do we choose $T(I)$?

$T(I)$ should be monotonically (not strictly) increasing, to preserve the intensity rank of the pixels.

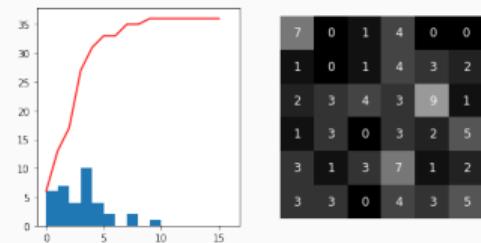


In this example we would change any pixel of the original image with intensity 3 to intensity 4, any with intensity 6 to intensity 7 and so on. It turns out that using the CDF (or the cumulative histogram, for the discrete case) is the solution! Look here if you want a mathematical proof!

Histogram equalization - an example

Assume you have a 4-bit image (dynamic range 0-15) with the following histogram

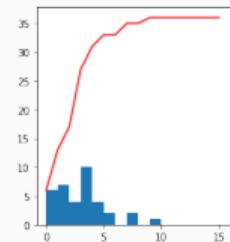
Pixel value	Number of pixels	Cumulative
0	6	6
1	7	13
2	4	17
3	10	27
4	4	31
5	2	33
6	0	33
7	2	35
8	0	35
9	1	36
10	0	36
11	0	36
12	0	36
13	0	36
14	0	36
15	0	36



Histogram equalization - an example

Assume you have a 4-bit image (dynamic range 0-15) with the following histogram

Pixel value	Number of pixels	Cumulative	Cum. normalized (h_n)
0	6	6	0.17
1	7	13	0.36
2	4	17	0.47
3	10	27	0.75
4	4	31	0.86
5	2	33	0.92
6	0	33	0.92
7	2	35	0.97
8	0	35	0.97
9	1	36	1.00
10	0	36	1.00
11	0	36	1.00
12	0	36	1.00
13	0	36	1.00
14	0	36	1.00
15	0	36	1.00

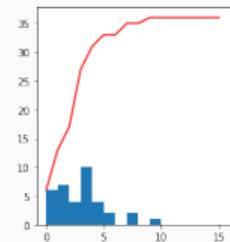


7	0	1	4	0	0
1	0	1	4	3	2
2	3	4	3	9	1
1	3	0	3	2	5
3	1	3	7	1	2
3	3	0	4	3	5

Histogram equalization - an example

Assume you have a 4-bit image (dynamic range 0-15) with the following histogram

Pixel value	Number of pixels	Cumulative	Cum. normalized (h_n)	New pixel value $\lfloor(h_n * I_{max})\rfloor$
0	6	6	0.17	2
1	7	13	0.36	5
2	4	17	0.47	7
3	10	27	0.75	11
4	4	31	0.86	12
5	2	33	0.92	13
6	0	33	0.92	13
7	2	35	0.97	14
8	0	35	0.97	14
9	1	36	1.00	15
10	0	36	1.00	15
11	0	36	1.00	15
12	0	36	1.00	15
13	0	36	1.00	15
14	0	36	1.00	15
15	0	36	1.00	15

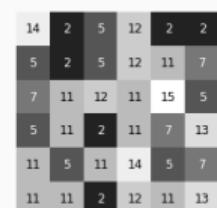
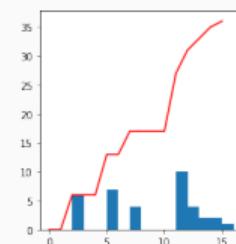
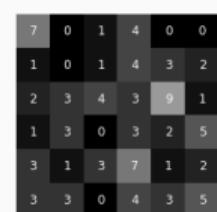
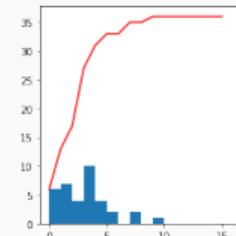


7	0	1	4	0	0
1	0	1	4	3	2
2	3	4	3	9	1
1	3	0	3	2	5
3	1	3	7	1	2
3	3	0	4	3	5

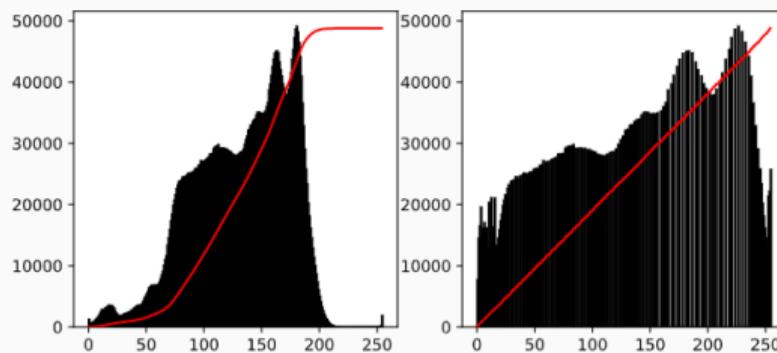
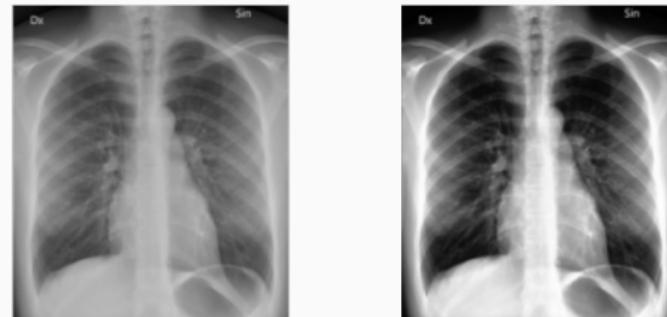
Histogram equalization - an example

Assume you have a 4-bit image (dynamic range 0-15) with the following histogram

Pixel value	Number of pixels	Cumulative	Cum. normalized (h_n)	New pixel value $\lfloor(h_n * I_{max})\rfloor$
0	6	6	0.17	2
1	7	13	0.36	5
2	4	17	0.47	7
3	10	27	0.75	11
4	4	31	0.86	12
5	2	33	0.92	13
6	0	33	0.92	13
7	2	35	0.97	14
8	0	35	0.97	14
9	1	36	1.00	15
10	0	36	1.00	15
11	0	36	1.00	15
12	0	36	1.00	15
13	0	36	1.00	15
14	0	36	1.00	15
15	0	36	1.00	15



On a real image...



Note how, in practice, we often do not have a uniform histogram.

Beyond histogram equalization...

Histogram equalization can result in increased noise and "unrealistic" looking images. Useful e.g. for X-rays, but other strategies exist

Beyond histogram equalization...

Histogram equalization can result in increased noise and "unrealistic" looking images. Useful e.g. for X-rays, but other strategies exist

Adaptive histogram equalization (AHE) computes several histograms of different regions of the image to improve local contrast. Drawback: increased noise in areas with low contrast.

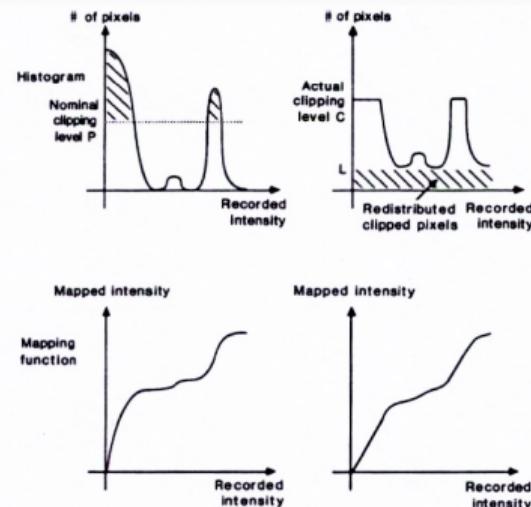
Beyond histogram equalization...

Histogram equalization can result in increased noise and "unrealistic" looking images. Useful e.g. for X-rays, but other strategies exist

Adaptive histogram equalization (AHE) computes several histograms of different regions of the image to improve local contrast. Drawback: increased noise in areas with low contrast.

Contrast Limited Adaptive Equalization (CLAHE) solves this by reducing the amount of contrast enhancement. This is done by clipping the histogram to a maximum before calculating the cumulative histogram.

This is implemented in the
`skimage.exposure.equalize_adapthist` function.



Histogram equalization - Scikit image

```
from skimage.exposure import equalize_hist, equalize_adapthist  
  
img_equalized = equalize_hist(img)  
# Or, with CLAHE  
img_equalized_CLAHE = equalize_adapthist(img, clip_limit=0.03)
```

Original



equalize_hist



equalize_adapthist (clip_limit 0.02)



equalize_adapthist (clip_limit 0.04)



- Histograms are an important tool for inspecting image quality
- Optimal brightness and contrast can help reveal specific image features
- Manipulation of image histograms e.g. through point operations is one important tool for image enhancement