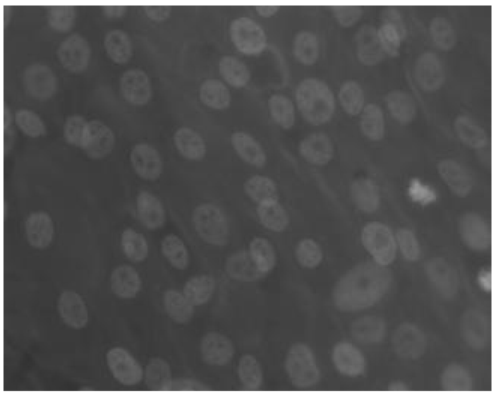
Tutorial: Image Enhancement

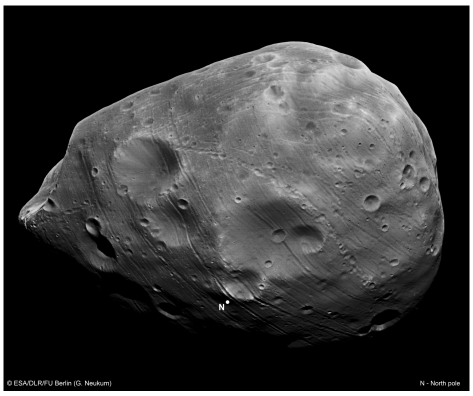
The objective of this tutorial is to implement some image enhancement methods, based on intensity transformations or histogram modifications. It will make use of statistical notions like probability density functions or cumulative distribution functions.

Figure 1: The different processes of this tutorial will be applied on these images.

(a) Osteoblasts.

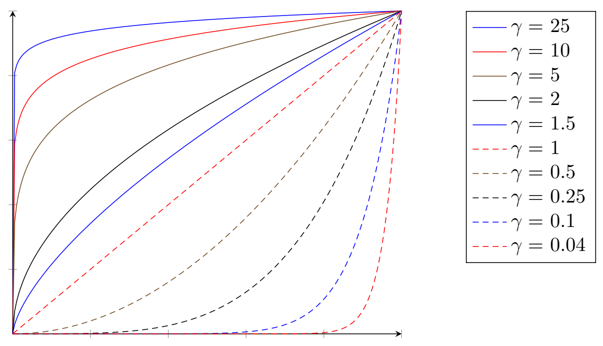


(b) Phobos (ESA/DLR/FU Berlin, CC-By-SA).

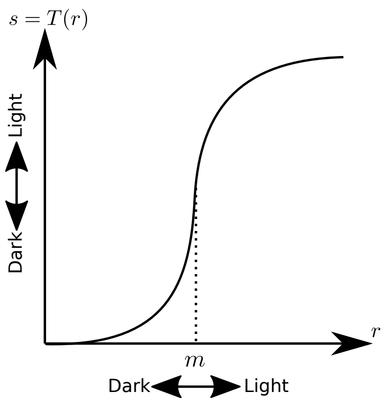


1 Intensity transformations (LUT)

Two transformations will be studied: the γ correction and the contrast stretching. They enable the intensity dynamics of the gray tone image to be changed. These two operators are based on the following Look Up Tables (LUT):



(c) γ correction LUT.



(d) Contrast stretching LUT.

?

1. Test the transformation 'γ correction' on the image 'osteoblast'.
2. Implement the operator 'contrast stretching' with the following LUT (also called cumulative distribution function cdf), with m being the mean gray value of the image, and r being a given gray value:

$$s = T(r) = \frac{1}{1 + (m/r)^E}$$

3. Test this transformation with different values of E on the image 'osteoblast'.

You can have a look at imadjust for γ correction.

The module skimage.exposure contains different methods for contrast enhancement, among them adjust_gamma.

2 Histogram equalization

The objective is to transform the image so that its histogram would be constant (and its cumulative distribution function would be linear). The notations are:

- I is the image of n pixels, with intensities between 0 and L (for 8-bits images, $L = 255$).
- h is the histogram, defined by:

$$h_I(k) = p(x = k) = \frac{n_k}{n}, \quad 0 \leq k \leq L$$

The following transformation $T(I)$ is called histogram equalization.

$$T(x_k) = L \cdot \mathrm{cdf}_I(k)$$

where

$$\mathrm{cdf}_I(k) = \sum_{j=0}^{k} p(x_j)$$

is the cumulative distribution function (cumulative histogram).

?

1. Compute and visualize the histogram of the image 'osteoblast'.
2. Test this histogram equalization transformation on the image 'osteoblast' (with builtin functions) and visualize the resulting histogram.

3. Code your own function.
4. The corresponding LUT to this transformation is the cumulative sum of the normalized histogram. Evaluate and visualize this intensity transformation.

See the imhist, histcounts and histeq functions.

See the numpy.histogram and skimage.exposure functions.

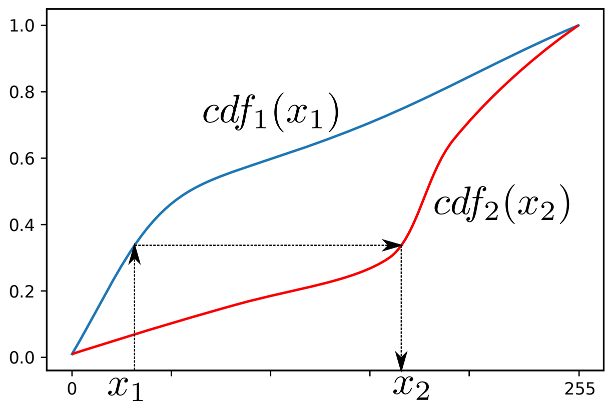
3 Histogram matching

The objective is to enhance the original image by matching its histogram with a modeled one. The principle is to transform the gray value x_1 of first image into x_2: $T(x_1) = x_2$ (see Fig.2). Based on the fact that $\mathrm{cdf}_1(x_1) = \mathrm{cdf}_2(x_2)$, the formula to find x_2 is:

$$x_2 = \mathrm{cdf}_2^{-1}\left(\mathrm{cdf}_1(x_1)\right).$$

As x_1 and x_2 are discrete values, this requires an interpolation.

Figure 2: Histogram matching principle.



?

1. Visualize the histogram of the image 'phobos'.
2. Make the histogram equalization and visualize the resulting image.
3. Construct a bi-modal histogram (for example) and code your own function for histogram matching.

See interp1 for interpolation and LUT application.

See numpy.interp for interpolation and LUT application.