Image enhancement - Point and local operators

1. Generation and display of intensity histograms

The main IPT function for generating intensity histograms is the *imhist* function, whose syntax is

```
h=imhist(f, b)
```

where *h* is the histogram of image *f*, and *b* is the number of intervals (bins) used in the representation (the default value for *b* is 256). The width of each interval (bin) corresponds to a uniform subdivision of the intensity scale.

The simplest way to display an image histogram is to use the command:

```
>> imhist(f);
```

or the command sequence:

```
>> figure, imhist(f));
```

However, histograms can be visualized using other graphic representation through the use of the *plot* (plot with solid line), *bar* (bar chart) or *stem* ("stamens" charts) functions (see MATLAB Help to check the parameters and functionality of the different options).

2. Point operators

Image enhancement can be achieved using different types of operators, with particular emphasis on point operators. Point operators that are used for image enhancement are often referred to as “intensity transformation functions”.
We will study also another type of point operators, generally known as “histogram specification transformations” which aim to obtain a final image whose histogram has a specific shape.
Many intensity transformation functions are based on information taken from the image intensity histogram, so we will start by analyzing the IPT routines that allow obtaining and visualizing this type of data.

The basic tool provided by the IPT for the intensity transformation implementation is the *imadjust* function. The syntax of this function is:

```
G = imadjust(f, [low_in high_in], [low_out high_out], gamma)
```

This function maps the intensity values in the range defined by *low_in* and *high_in* to values in the range *low_out* and *high_out*. The values below *low_in* and above *high_in* are truncated to *low_out* and *high_out*, respectively. The use of empty matrix, [], instead of [*low_in high_in*] or [*low_out high_out*] is equivalent to the use of default values [0 1].

The output image has the same class of the input image; regardless of the image class, all the values for the parameters *low_in, high_in, low_out* and *high_out* should be represented in the range [0 1].

The parameter *gamma* determines the transformation function (gamma = 1: linear; gamma < 1: lower intensities expansion; gamma > 1: higher intensities expansion).

The histogram specification point operator most used in image processing applications is the histogram equalization transformation, which aims at obtaining an output image with an uniform histogram. The IPT function that performs this kind of transformation is the *histeq* routine:

```
g=histeq(f, nlev)
```

where *g* is the resulting equalized image, *f* is input image and *nlev* is the number of intensity levels specified for the output image (the *nlev* default value is 64).

The *histeq* function can also be used for generating an output image whose histogram is not uniform but in opposite has a form specified by the user. In this case, the syntax of the function is:

```
g=histeq(f, hspec)
```

where *hspec* is a row matrix containing the desired histogram and *g* is the output image whose histogram approximates *hspec*.

The *adapthisteq* function is very similar to the *histeq* function. This function also allows performing histogram equalization, thereby improving the contrast of the image; however, it operates on small data regions (tiles), rather than the entire image. Subsequently, the small regions are combined using bilinear interpolation. Given the default values for the parameters of this function, the syntax of this function is:

```
g=adapthisteq(f)
```

where *g* is the image result and *f* is the input image.

3. Local operators

Spatial filtering is the usual designation for the application of local operators defined in the spatial domain. Local operators can be divided into two groups: linear operators and nonlinear operators.

Spatial filtering operations are implemented using the *imfilter* linear function, whose syntax is:

```
g=imfilter(f, w, boundary_options, size_options, filtering_mode)
```

where g is the filtered image, f is the original image and w is the filter mask.

This function allows the implementation of a convolution (linear filtering) or correlation operator, depending on the value assigned to the filtering_mode parameter (see details in MATLAB Help); if this parameter is omitted, a correlation operator is implemented. The two other parameters specify how to fill the border of the images and the size of the final filtered image (see details in MATLAB help).

The weights of the filter mask (w) can be set directly by the user. However, IPT already includes a set of predefined masks which may be obtained using the *fspecial* function, whose syntax is:

```
w=fspecial('type', parameters)
```

where 'type' specifies the filter type. Parameter values depend on the filter type selected (see details in MATLAB Help).

The IPT provides two functions for implementing generic nonlinear filters. However, the nonlinear filters most commonly used in image processing are the order filters, implemented in the IPT by the *ordfilt2* function. The syntax of this function is:

g=ordfilt2(f, order, domain)

The response of this filter is based on the ordering of pixel values in the neighborhood defined in parameter “domain”, followed by the selection of the value contained in the “order” position of the sorted list. The implementation of a minimum filter is accomplished setting order to 1; the implementation of a maximum filter order is achieved by setting order to the number of elements in the neighborhood.

The most frequently used order filter is the median filter; the response of this filter is the median value of the sorted list. Given its importance, the IPT provides a specific implementation for this filter in its two-dimensional version through the *medfilt2* function:

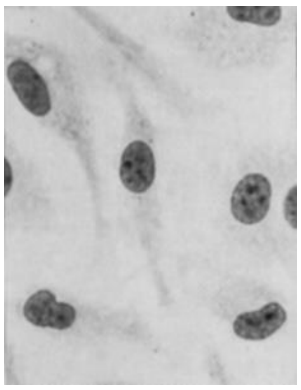
g=medfilt2(f, [m n], padopt)

(See details in MATLAB Help).

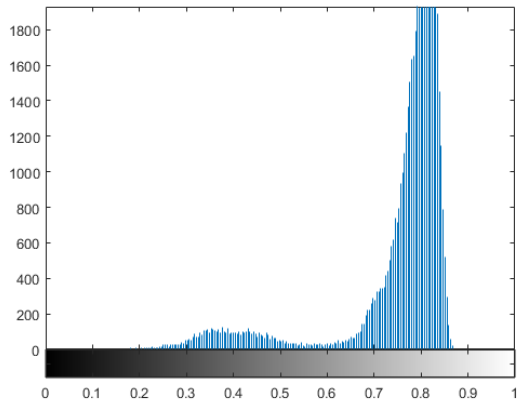
Work proposal

1. Write a program in Matlab that allows performing the operations listed below.

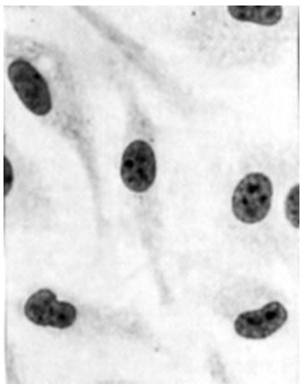
A. (pen and paper) Draw and determine the parameters of the linear transformation expression that allows to obtain the following histogram:



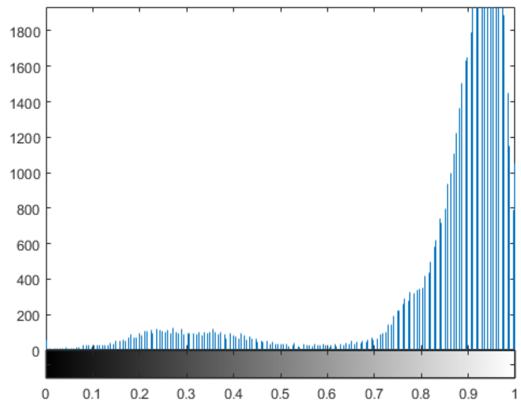
Original image



Original histogram



Transformed image



Transformed histogram

B. Read the image “celulas_1.tif” and display the image histogram. Then, implement and apply the linear transformation from A.

C. Perform a linear expansion of intensities using 256 output levels (function *imadjust* with parameter *gamma*=1). Compare the result with the implementation from B.

D. Perform a non-linear expansion of the image low level intensity range (function *imadjust* with parameter *gamma*<1).

E. Perform a non-linear expansion of the image high level intensity range (function *imadjust* with parameter *gamma*>1).

F. Perform a histogram equalization operation. Analyze and discuss the results, indicating which of the proposed solutions (B, C, D or E) is most appropriate to the image.

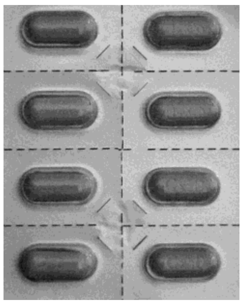
G. Obtain the intensity transformation function that implements the histogram equalization from E, display this intensity transformation function, the image resulting from the equalization and its histogram.

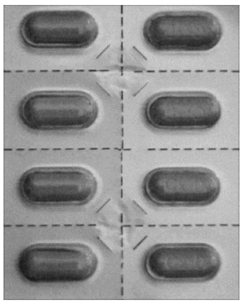
H. Use the *adapthisteq* function to improve the image contrast considering the default values for the parameters (use the Matlab help for more information about this function). Display and compare the result to the original image and the equalized image (from E).

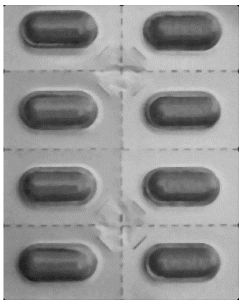
I. For the *adapthisteq*, test the influence of the *“NumTiles”* parameter ∈ {2,5,10,20}. Compare the contrast enhancement results and the runtimes (*tic, toc* functions) and comment on the results.

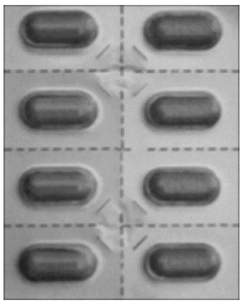
J. Similarly, assess the influence of different values for the “ClipLimit” parameter ∈ {0,0.001,0.01,0.02,0.1}. Compare the contrast enhancement results and comment on the influence of this parameter value.

2. The original image of figure 1.a was filtered using 3 smoothing filters: an average filter, a Gaussian filter and a median filter, all defined using a square 9×9 window. The resulting images are shown in figures 1.b, 1.c e 1.d. What filter was used for processing which one of the images?


1.a


1.b


1.c


1.d

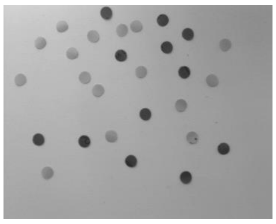
3. The image of the previous problem is available in the file “blister.tif”. Use this image for evaluating and comparing smoothing filters (average, gaussian) and rank filters (minimum, maximum, median).

A. Apply the filters of problem 3 to the image. Display the images and the corresponding histograms to compare the changes;

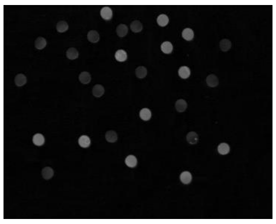
B. Repeat A. after changing the size of the filters.;

C. Evaluate the results of minimum, maximum and median filters using filter masks with different sizes and shapes (square, rectangle, cross, diagonal).

4. Suppose that we want to segment the circular objects of figure 2.a. However, this image shows a slow variation in the background. In order to do this, we need to estimate the background (figure 2.b) and then subtract this estimation to the original image (figure 2.c).


2.a

2.b


2.c

The image of figure 2.a is available in file "smarties.tif". Write MatLab code to generate the images in Figures 2.b and 2.c.

5. The image "hand.tif" needs to be processed for enhancing the edges. Because a direct application of a Laplacian filter gives raise to noise, a Gaussian filter was applied first.

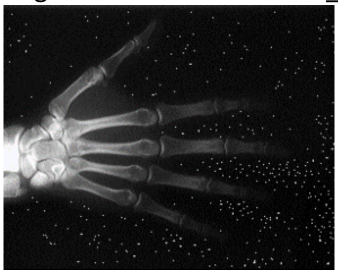
A. Compute the images that results from 1) the application of a Laplacian filter; 2) the application of a Gaussian filter before the Laplacian filter and 3) the LoG filter (Laplacian of Gaussian). Use the *imfilter* and *fspecial* functions. Use sigma=1.4.

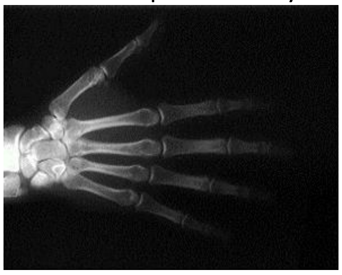
B. Determine the image with the edges (*i.e.* the zero crossings) that result from the application of the LoG filter. Evaluate the influence of the smoothing operation by changing the dimension and standard deviation of the Gaussian. Use the *edge* function and sigma $\in$ {1, 1.4, 1.8, 2.2, 2.6, 3, 5}.

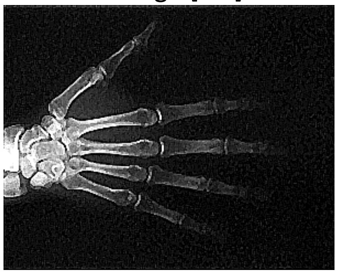
6. For enhancing Image 1, a sequence of two local operators was used for transforming figure 3.a into figure 3.b (operation #1), and this last image into figure 3.c (operation #2).

A. Characterize both operations and suggest an operator to implement each one of these functions.

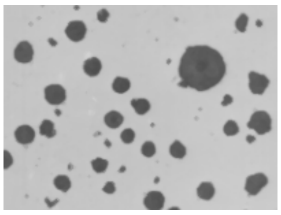
B. Implement the suggested operators using the MATLAB functions using as input the image available in file "hand_noise.tif". Represent only values in the range [0 1].

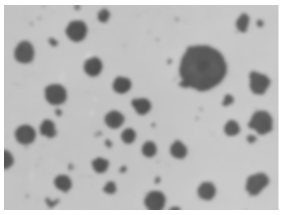

3.a

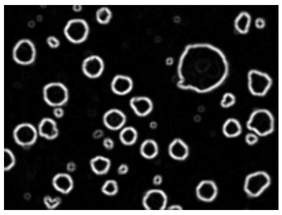

3.b


3.c

7. Figures below present an original image and the images that resulted from smoothing and edge enhancement operations.


Original


Smoothed


Edges

A. Characterize the two above mentioned operations and select convolution operators for their implementation. The original image is available in the "blobs2.tif" file. Write a set of MATLAB instructions to obtain the results shown in the figures.

B. The smoothed image was submitted to a sequence of operations aiming the segmentation of the dark objects with larger size. The image shown on the left resulted from the binarization of the smoothed image, and was then filtered to obtain the image shown on the right. The binary image presented on the left is available in the "blobs2_bin.tif" file. Write a set of MATLAB instructions for implementing a sequence of local operations that is able to achieve the final image (right) from the binarized image.