Image Processing Tutorials
with PYTHON®

Yann GAVET
Johan DEBAYLE

Credits
Cover image: pixexid, pixabay.

Contributors:

- Léo Théodon
- Victor Rabiet
- Séverine Rivollier
- Valentin Penaud–Polge
- Place your name here and get special credit if you want to participate to this book.

Table of Contents

11 Part I
Day 1

1 Introduction to image processing 13

19 Part II
Day 2

2 Histogram-based image segmentation 21

25 Part III
Day 3: activity of your choice

3 Image Enhancement 27
4 2D Fourier Transform 31
5 Image restoration: denoising 35
6 Binary Mathematical Morphology 39
7 Hough transform and line detection 43
8 Shape Diagrams 45
9 Image Characterization 47
10 Machine Learning 51
11 Harris corner detector 55

About the authors

Yann GAVET

received his "Ingénieur Civil des Mines de Saint-Etienne" diploma in 2001. He then obtained a Master of Science and a PhD thesis on the segmentation of human corneal endothelial cells (in 2004 and 2008). He is now an assistant professor at the Saint-Etienne School of Mines, where he teaches computer science and image processing to engineering and master students. He is a member of the PMDM Department of the LGF Laboratory, UMR CNRS 5307, dedicated to granular media analysis and modelisation.

He is particularly interested in the world of free (LIBRE) software in computer science. His research interests include image processing and analysis, stochastic geometry and numerical simulations. He published more than 70 papers in international journals and conference proceedings. He is a member of the Institute of Electrical and Electronics Engineers (IEEE), the International Association for Pattern Recognition (IAPR), International Society for Stereology and Image Analysis (ISSIA). He has worked for CS-SI (Toulouse, France) as an IT engineer, and for Thalès-Angénieux (Saint-Héand, France) as an image processing expert.

Johan DEBAYLE

received his M.Sc., Ph.D. and Habilitation degrees in the field of image processing and analysis, in 2002, 2005 and 2012 respectively. Currently, he is a Full Professor at the Ecole Nationale Supérieure des Mines de Saint-Etienne (ENSM-SE) in France, within the SPIN Center and the LGF Laboratory, UMR CNRS 5307, where he leads the PMDM Department interested in image analysis of granular media. In 2015, he was a Visiting Researcher for 3 months at the ITWM Fraunhofer / University of Kaisersleutern in Germany. In 2017 and 2019, he was invited as Guest Lecturer at the University Gadjah Mada, Yogyakarta, Indonesia. He was also Invited Professor at the University of Puebla in Mexico in 2018 and 2019. He is the Head of the Master of Science in Mathematical Imaging and Spatial Pattern Analysis (MISPA) at the ENSM-SE.

His research interests include image processing and analysis, pattern recognition and stochastic geometry. He published more than 120 international papers in international journals and conference proceedings and served as Program committee member in several international conferences (IEEE ICIP, MICCAI, ICIAR…). He has been invited to give a keynote talk in several international conferences (SPIE ICMV, IEEE ISIVC, SPIE-IS&T EI, SPIE DCS…) He is Associate Editor for 3 international journals: Pattern Analysis and Applications (Springer), Journal of Electronic Imaging (SPIE) and Image Analysis and Stereology (ISSIA).

He is a member of the International Society for Optics and Photonics (SPIE), International Association for Pattern Recognition (IAPR), International Society for Stereology and Image Analysis (ISSIA) and Senior Member of the Institute of Electrical and Electronics Engineers (IEEE).

ÉCOLE NATIONALE SUPÉRIEURE DES MINES DE SAINT-ÉTIENNE

One of the missions of École des Mines de Saint-Étienne, France, is scientific research at the highest level and contributions to companies' competitiveness. It aims at conveying the economical politics of the country and speeding up the sustainable industrial development by innovation and efficient contributions.

This high level scientific research leads to publications recognized by the international scientific community. Research and teaching are very closely interwoven and the consequence of this is the attractiveness of our Master's Degree courses and our renowned Doctoral School.

MINES
Saint-Étienne

Une école de l'IMT

Liminar considerations

This book is presented as a collection of tutorials. Each chapter presents different objectives, usually with practical applications, in order to develop the image processing and analysis skills by its own. For each tutorial, you will first find in the statement different questions that will guide you to the complete results. Our suggestion is to start by the first tutorials, that acts as a round-up stage. Then, choose the topic that interests and motivates you, and start to work on your code.

The statements does not always contain the necessary theoretical background that you need to answer to the questions. Find resources on the net. Wikipedia is usually a very good start. Go to the university library!

The correction gives you a proposition of solution: it is not unique and it might not be perfect (or it might unfortunately contain errors). This correction is also available in a dedicated website (links will be provided along with the tutorials), we suggest you have a look at it only when you encounter blocking problems or fastidious lines of code to program. Moreover, you can evaluate the processing time of your version of the code and compare it to our proposed version, which usually uses fast and optimal functions, except for readibility purposes.

With Python, you can use the following code:

```
import time

time_start = time.timeit()
# run your code
time_elapsed = (time.timeit() - time_start)
```

CC-By license?

If you manage to code a drastically faster method, if you notice errors or mistakes, if you want to add precisions, remember that this book as well as the code is published under a FREE CC-BY license (as in FREE speech). Contact us and we will be really happy to introduce modifications and insert you in the credits, or more...

The book is available in a high quality printed format, obviously not for free, but the pdf format is free (as in FREE beer). This is due to the fact that we (Johan Debayle and Yann Gavet) are employed by the French government, and we are paid to teach and disseminate informations to students. Our work is thus belongs to the society, and it seems a normal process to give the results back to the society. Feel free to use, modify and distribute these tutorials as you wish. Just remember to keep the appropriate credits (our names and MINES Saint-Etienne). If you want to express your gratitude, send us a postcard at:

```
Yann GAVET or Johan Debayle
MINES Saint-Etienne
CS 62362
42023 SAINT-ETIENNE cedex 2 - FRANCE
```

You can also buy the printed version of the book, which will make our editor really happy.

Images

The images belong to their authors, unless otherwise stated. If by mistake we used images without giving the appropriate credit, please contact us.

Softwares

This book is available for two programming languages, MATLAB® and Python. MATLAB® is a proprietary software dedicated to scientific computing. Functions are grouped into so-called toolboxes. The documentation

and the compatibility of the functions are very good, but the price is a consequence of this. Python is built upon open-source and Free softwares. Scientific modules are numerous, but other general purposes functions can also be found. Documentation and code may be of inequal qualities, but major scientific modules (numpy, scipy, opencv...) are really well presented and optimized.

Portions of code will be highlighted by the use of special boxes, like:

```
% this is matlab code
```

```
# this is python code
```

Time to spend on a tutorial

The tutorials are almost independent. To evaluate the difficulty of the tutorial, stars are present at the header of each one. Depending on your knowledge, you will have to spend between 2 hours and 10 hours per tutorial.

- One star tutorial should be a relatively easy tutorial,
- two stars tutorials introduce some difficulties, either in programming or in the theoretical concepts,
- three stars tutorials are difficult both in theory and in programming.

They are divided into 6 parts, namely:

- Enhancement and Restoration: dedicated to image processing methods employed to eliminate noise and improve vision of data.
- Mathematical Morphology: introduction to basic operations of mathematical morphology.
- Registration and Segmentation: methods to segment, i.e. detect objects in images.
- Stochastic Analysis: method based on random processes.
- Characterization and Pattern Analysis: measures performed on objects in order to perform analysis and recognition.
- Exams: uncorrected problems and questions.

Enjoy!

You will find online corrections on the editor website and at page: http://iptutorials.science. You will also find qrcodes for each correction that points to code and images.

Part I Day 1

1 Introduction to image processing

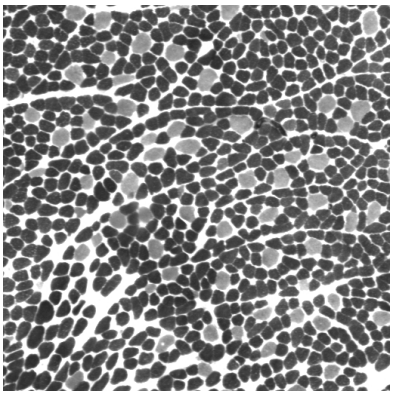
In this tutorial, you will discover the basic functions in order to load, manipulate and display images. The main informations of the images will be retrieved, like size, number of channels, storage class, etc. Afterwards, you will be able to perform your first classic filters.
Do not forget that one of the main objectives is finding by yourself (in the documentation of the different modules) the usage of the appropriate functions.

The different processes will be realized on the following images:

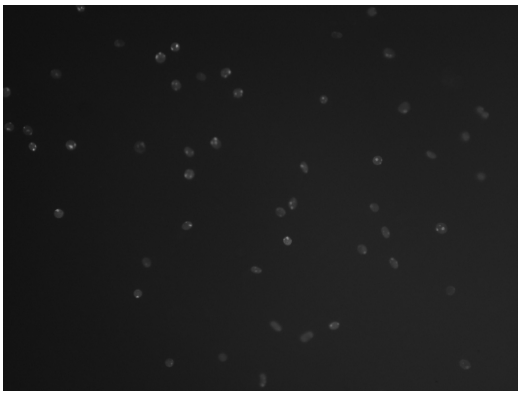
Figure 1.1: Image examples.

(a) Retinal vessels.

(b) Muscle cells.



(c) Cornea cells (BIIGC, Univ. Jean Monnet, Saint-Etienne, France).



1.1. First manipulations

- Image loading can be made by the use of the python function skimage.io.imread.
- The visualization of the image in the screen is realized by using the module matplotlib.pyplot.
- Also import the numpy module with `import numpy as np`. All images with be manipulated as a numpy array.

- Load and visualize the first image as below. Notice the differences.
- Look at the data structure of the image I such as its size, type....
- Visualize the green component of the image. Is it different from the red one? What is the most contrasted color component? Why?
- Enumerate some digital image file formats. What are their main differences? Try to write images with the JPEG file format with different compression ratios (0, 50 and 100), as while as the lossless compression, and compare.

See skimage.io.imsave

1.2. Color quantization

Color quantization is a process that reduces the number of distinct colors used in an image, usually with the intention that the resulting image should be as visually similar as possible to the original image. In principle, a color image is usually quantized with 8 bits (i.e. 256 gray levels) for each color component.

- By using the gray level image 'muscle', reduce the number of gray levels to 128, 64, 32, and visualize the different resulting images.
- Compute the different image histograms and compare.

You can take advantage of the floor division in order to get integer value of the division, for example:

```
>>> print (255//4)
63
```

1.3. Image histogram

An image histogram represents the gray level distribution in a digital image. The histogram corresponds to the number of pixels for each gray level. It is equivalent to a probability density function.

With numpy, you can use a the following code to plot the histogram:

```
# I is the 2D grayscale image
h, edges = np.histogram(I, bins=256)
plt.plot(edges[:-1], h)
plt.show()
```

You have to code a python function with the following prototype:

```
def histogram(I):
```

- Test the numpy version of the histogram.
- Code your own version of this function and visualize the histogram of the image 'muscle.jpg'.

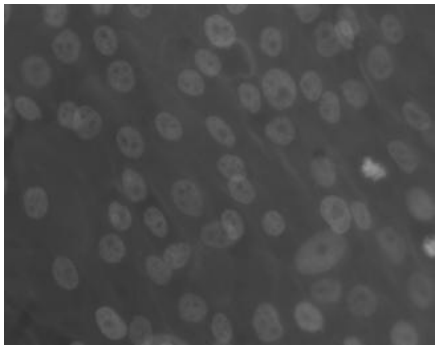
1.4. Linear mapping of the image intensities

The gray level range of the image 'cellules_cornee.jpg' can be enhanced by a linear mapping such that the minimum (resp. maximum) gray level value of the resulting image is 0 (resp. 255). Mathematically, it consists in finding a function $f(x) = ax + b$ such that $f(min) = 0$ and $f(max) = 255$.

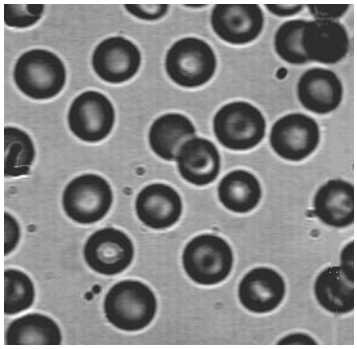
- Load the image and find its extremal gray level values.
- Adjust the intensities by a linear mapping into $[0, 255]$.
- Visualize the resulting image and its histogram.

1.5. Low-pass filtering

The different processes will be realized on the following images:



(a) osteoblast cells



(b) blood cells

Low-pass filtering aims to smooth the fast intensity variations of the image to be processed.

Test the low-pass filters 'mean', 'median', 'min', 'max' and 'gaussian' on the noisy image 'blood cells'.

The modules skimage. filters and skimage. filters .rank contain a lot of classical filters.

Which filter is suitable for the restoration of this image?

1.6. High-pass filtering

High-pass filtering aims to smooth the low intensity variations of the image to be processed.

- Test the high-pass filters HP on the two initial images in the following way: $HP(f) = f - LP(f)$ where LP is a low-pass filtering (see the previous exercise).
- Test the Laplacian (high-pass) filter on the two initial images.

1.7. Convolution algorithm

The general expression of the convolution, denoted by the operator $*$, is given by

$$g(x,y) = h * f(x,y) = \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} h(i,j) f(x-i, y-j)$$

with f the original image, g the filtered image and h the convolution mask (a.k.a. kernel).
The `scipy.signal.convolve2d` is used for 2D convolution.

Apply the convolution operation with the following convolution kernels, that define some classical filters:

- Mean filter
$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$
- Laplacian filter
$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & +8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$
- Gaussian filter:
$$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

1.8. Derivative filters

Derivative filtering aims to detect the edges (contours) of the image to be processed.

- Test the Prewitt and Sobel derivative filters (corresponding to first order derivatives) on the image 'blood cells' with the use of the following convolution masks:

$$\begin{bmatrix} -1 & 0 & +1 \\ -1 & 0 & +1 \\ -1 & 0 & +1 \end{bmatrix} \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ +1 & +1 & +1 \end{bmatrix}$$

$$\begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix}$$

- Look at the results for the different gradient directions.
- Define an operator taking into account the horizontal and vertical directions.

Remark : the edges could be also detected with the zero-crossings of the Laplacian filtering (corresponding to second order derivatives).

1.9. Enhancement filtering

Enhancement filtering aims to enhance the contrast or accentuate some specific image characteristics.

- Test the enhancement filter E on the image 'osteoblast cells' defined as: $E(f) = f + HP(f)$ where HP is a Laplacian filter (see also tutorial 3 for enhancement methods.).
- Parameterize the previous filter as: $E(f) = \alpha f + HP(f)$, where $\alpha \in \mathbb{R}$.

1.10. Aliasing effect

The following image is generated via a function g defined by: $g(x, y) = \sin(2\pi f_0 \sqrt{x^2 + y^2})$.

Informations

To generate a 2D realization of this function, the np.meshgrid function is strongly recommended. Its usage is as follows, with f_s being the (spatial) sampling frequency:

```
t = np.arange(0, 1, 1./fs)
xx, yy = np.meshgrid(t, t)
realization = g(xx, yy) # compute 2D values
```

- Create an image (as right) that contains rings, defined above. The function takes two input parameters: the sampling frequency f_s and the signal frequency f_0.
- Look at the influence of the two varying frequencies. What do you observe? Explain the phenomenon from a theoretical point of view.

1.11. Open question

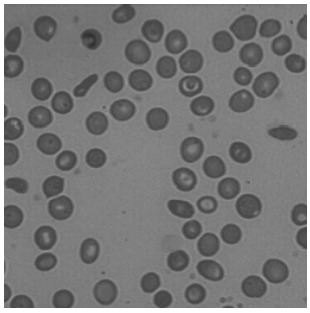
Find an image filter for enhancing the gray level range of the image 'osteoblast cells'.

Part II Day 2

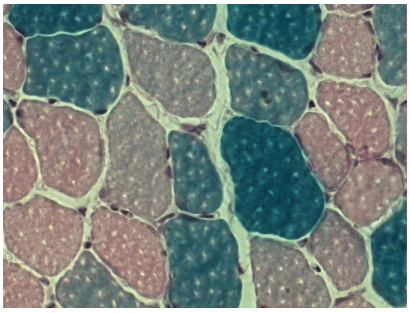
2 Histogram-based image segmentation

This tutorial aims to implement some image segmentation methods based on histograms (thresholding and "k-means" clustering).

The different processes will be applied on the following images:



(a) Cells.



(b) Muscle cells, author: Damien Freyssenet, University Jean Monnet, Saint-Etienne, France.

2.1. Manual thresholding

The most simple segmentation method is thresholding.

- Visualize the histogram of the grayscale image 'cells'.
- Make the segmentation with a threshold value determined from the image histogram.

2.2. k-means clustering

Let $X = \{x_i\}_{i\in[1;n],n\in\mathbb{N}}$ be a set of observations (the points) in $\mathbb{R}^d$. The k-means clustering consists in partitioning X in k ($k < n$) disjoint subsets $\tilde{S}$ such that:

$$\tilde{S} = \arg\min_{S=\{S_i\}_{i\leq k}} \sum_{i=1}^{k} \sum_{x_j\in S_i} \|x_j - \mu_i\|^2 \tag{2.1}$$

where μ_i is the mean value of the elements in S_i. The k-means algorithm is iterative. From a set of k initial elements $\{m_i^{(1)}\}_{i\in[1;k]}$ (randomly selected), the algorithm iterates the following (t) steps:

- Each element of X is associated to an element m_i according to a distance criterion (computation of a Voronoi partition):

$$S_i^{(t)} = \left\{\mathbf{x}_j : \|\mathbf{x}_j - \mathbf{m}_i^{(t)}\| \leq \|\mathbf{x}_j - \mathbf{m}_{i^*}^{(t)}\|, \forall i^* \in [1;k]\right\} \tag{2.2}$$

- Computation of the new mean values for each class:

$$\mathbf{m}_i^{(t+1)} = \frac{1}{|S_i^{(t)}|} \sum_{\mathbf{x}_j\in S_i^{(t)}} \mathbf{x}_j \tag{2.3}$$

where $|S_i^{(t)}|$ is the number of elements of $S_i^{(t)}$.

Although interesting, the goal of this tutorial is not to code the k-means algorithm in a general case. We will code it for grayscale images, and use builtin functions for other cases with higher dimensions.

2.3. Grayscale image, $k = 2$ in one dimension

The objective is to binarize image image 'cells', which is a grayscale image. The set X is defined by $X = \{I(p)\}$, for p being the pixels of the image I.

- Implement the algorithm proposed below (Alg. 1).
- Test this operator on the image 'cells'.
- Test another method of automatic thresholding (defined by Otsu in [3]).
- Compare the values of the thresholds (manual and automatic).

Data: Original image A
Data: Stop condition ε
Result: thresholded image
Initialize T_0, for example at $\frac{1}{2}(\max(A) + \min(A))$;
$done \leftarrow False$;
while *NOT done* **do**
  Segment the image A with the threshold value T;
  it generates two classes G_1 (intensities $\geq T$) and G_2 (intensities $< T$);
  Compute the mean values, denoted μ_1, μ_2, of the two classes G_1, G_2, respectively;
  Compute the new threshold value $T_i = \frac{1}{2}(\mu_1 + \mu_2)$;
  if $|T_i - T_{i-1}| < \varepsilon$ **then**
    $done \leftarrow True$
  end
end
Segment the image with the estimated threshold value.

Algorithm 1: K-means algorithm for automatic threshold computation of grayscale images.

For use with python, use the module skimage. filter and the function threshold_otsu.

2.4. Simulation example, $k = 3$ in two dimensions

The objective is to generate a set of 2-D random points (within $k = 3$ distinct classes) and to apply the k-means clustering for separating the points and evaluating the method (the classes are known!).

Write a function for generating a set of n random points (x_i, y_i) around a point with coordinates (x, y).

This function is described with the following equation:

$$Points = \begin{bmatrix} x_1 & \dots & x_n \\ y_1 & \dots & y_n \end{bmatrix} + \begin{bmatrix} x \\ y \end{bmatrix}$$

We will use the python function randn from the module numpy.random. The following function generates such a point cloud:

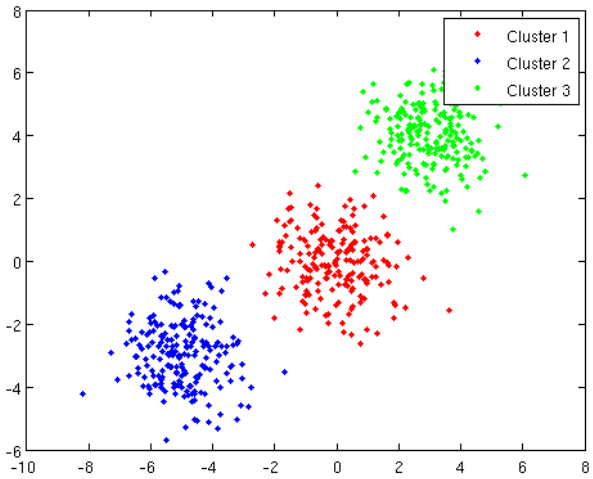
```
def generation(n, x, y):
    Y = np.random.randn(n, 2) + np.array([[x, y]])
    return Y
```

The concatenation of numpy arrays can be done by using np.concatenate((A1, A2, A3)).

- Use this function to generate 3 set of points (in a unique matrix) around the points $(0, 0)$, $(3, 4)$ and $(-5, -3)$.
- Use the builtin kmeans function for separating the points. The result is presented in Fig. 2.1.

Use the python function sklearn.cluster.KMeans. Verify the utility of the option n_init=10.

Figure 2.1: Resulting clustering of random points.



2.5. Color image segmentation using K-means: $k = 3$ in 3D

The k-means clustering is now used for segmenting the color image representing the muscle cells 'Tv16.png'.

Which points have to be separated? Transform the original image into a vector of size $N \times 3$ (where N is the number of pixels) which represent the 3 components R, G and B of each image pixel.

Informations

An image is an array of shape $(nLines, nCols, nChannels)$. The sklearn.cluster.KMeans functions requires data of shape $(nPoints, nDimensions)$. Thus, the following lines will *numpy.reshape* the arrays within the right shape:

```
# image is the color image
[nLines, nCols, nChannels] = image.shape
data = np.reshape(image, (nLines*nCols, nChannels))
data = data.astype(np.float32) # convert in float32 data type
```

After applying the KMeans function on data, the result should be converted into the image space, i.e. a 2D array of shape $(nLines, nCols)$:

```
# k_means_labels are the results of the Kmeans algorithm, see doc
result = np.reshape(k_means_labels, (nLines, nCols))
```

- Visualize the 3-D map (histogram) of all these color intensities.
- Make the clustering of this 3-D map by using the K-means method.
- Visualize the corresponding segmented image.

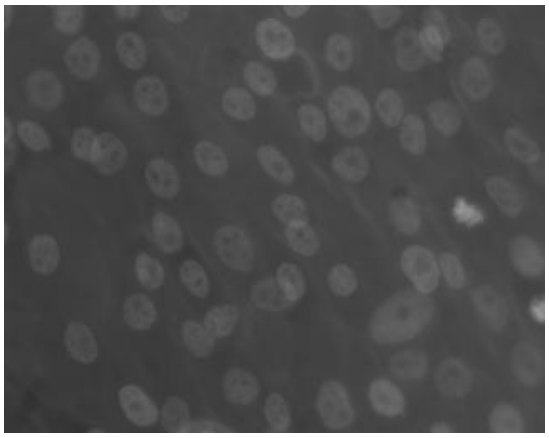
Part III Day 3: activity of your choice

3 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 3.1: The different processes of this tutorial will be applied on these images.

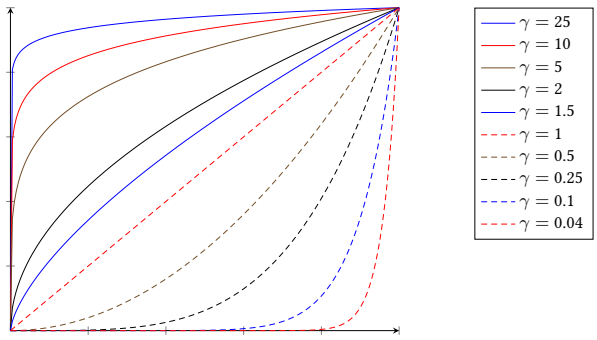
(a) Osteoblasts.



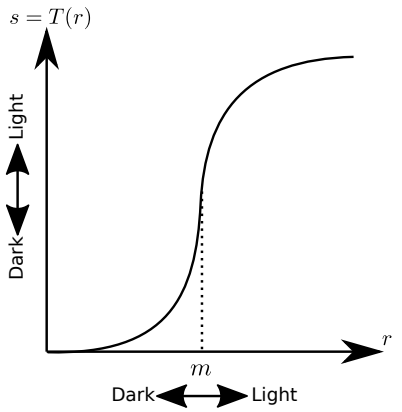
(b) Phobos (ESA/DLR/FU Berlin, CC-By-SA).

3.1. 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'.

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

3.2. 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 numpy.histogram and skimage.exposure functions.

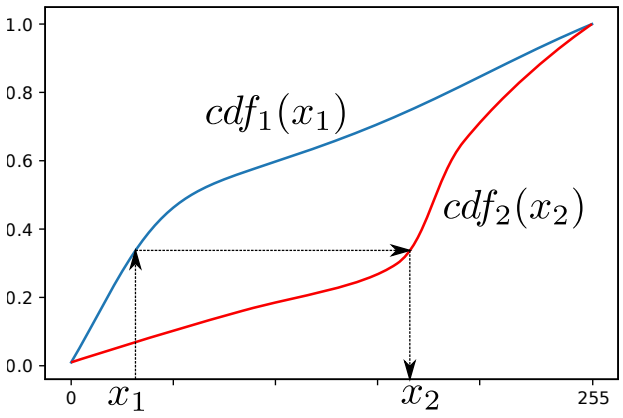
3.3. Histogram matching

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.3.2). 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 3.2: 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 numpy.interp for interpolation and LUT application.

4 2D Fourier Transform

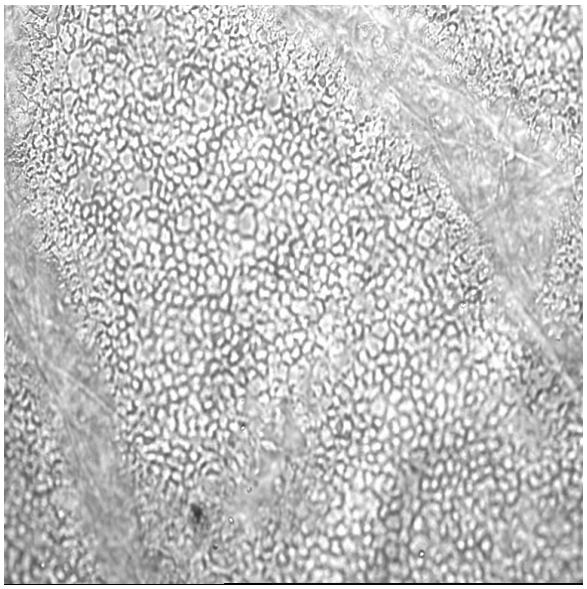
The main objective of this tutorial is to study image filters applied in the frequencial or spatial domain with the Fourier transform.

Use module np. fft for Fourier Transform functions (function fft2), angle and abs for phase and amplitude, fftshift for changing the representation).

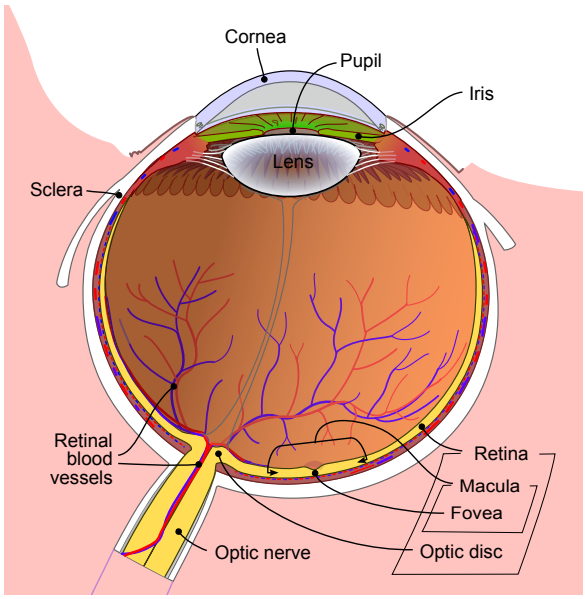
The image to be used comes from a human cornea endothelium observed ex vivo by optical microscopy.

Figure 4.1: Image of human cornea endothelium observed by optical microscopy. The ophthalmologists would like to know the cell density, without manually counting all the cells.

(a) Human corneal endothelium, extracted from a donor and observed here before grafting.



(b) Human eye (from Wikipedia, authors: Rhcastilhos and Jmarchn, CC-By-SA).



4.1. Fourier transform

1. Load an image and visualise it.
2. Compute the Fourier Transform by the fft algorithm.
3. Visualise the images of the phase and amplitude of the Fourier Transform.

4.2. Inverse Fourier transform

In this exercise, it can be interesting to consider different images, like the cameraman picture for example.

```python
from skimage.data import camera
I = camera()
```

1. Apply the inverse Fourier transform on the Fourier transform to find the original image.
2. Now, apply the inverse Fourier transform on the phase information only (without using the frequency informations).
3. In a same spirit, apply the inverse Fourier transform on the frequency informations only (without the phase).

4.3. Low-pass and high-pass filtering

The objective is to apply an ideal filter, that "perfectly" cuts the frequencies. You have to code this filter H in the Fourier domain so that $G = F \times H$ is the Fourier transform of the filtered image g.

1. Modify the Fourier transform of the image to
   - keep only low frequencies,
   - keep only high frequencies.
2. Apply the inverse Fourier transform on both and comment.

4.4. Application: evaluation of cellular density

The ophthalmologists would like to evaluate the cell density of the cornea endothelium observed in Fig. 4.1.

1. Compute the Fourier transform of the image. If one considers that the cells constitute a repeated pattern on the whole image, locate the repetition frequency on the amplitude image.
2. Can this frequency be linked to the cell density ?

The answer to this last question is obviously yes. Here follows a simple method to evaluate the cell density (or the mean cell radius), if these cells a considered as circular [7].

1. The amplitude information is very noisy. First of all, a gaussian filter should be applied.
2. Find a simple way to evaluate the mean radius of the cells.

Informations

See skimage. filters .gaussian.

5 Image restoration: denoising

This tutorial aims to study some random noises and to test different image restoration methods (image denoising).

The different processes will be applied on the following MR image.

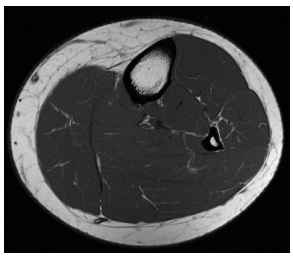


Figure 5.1: Leg.

5.1. Generation of random noises

Some random noises are defined with the given functions (corresponding to specific distributions):

- uniform noise: $R = a + (b - a) * U(0, 1)$.
- Gaussian noise: $R = a + b * N(0, 1)$.
- Salt and pepper noise: $R : \begin{cases} 0 \leq U(0,1) \leq a & \mapsto & 0 \\ a < U(0,1) \leq b & \mapsto & 0.5 \\ b < U(0,1) \leq 1 & \mapsto & 1 \end{cases}$
- Exponential noise : $R = -\frac{1}{a} * \ln(1 - U(0, 1))$

Generate sample images with the four kinds of random noise and visualize their histograms with the built-in functions. Pay attention to the intensity range of the resulting images when calculating the histograms

5.2. Noise estimation

The objective is to evaluate the characteristics of the noise in a damaged/noisy image (in a synthetic manner in this tutorial).

1. Visualize the histogram of a Region Of Interest (ROI) of the image of Fig.5.1. The ROI should be extracted from a uniform (intensity) region.
2. Add an exponential noise to the original image and visualize the histogram of the selected ROI.

3. Add a Gaussian noise to the original image and visualize the histogram of the selected ROI.

5.3. Image restoration by spatial filtering

1. Add a salt-and-pepper noise to the image of Fig.5.1.
2. Test the 'min', 'max', 'mean' and 'median' image filters.

See the module scipy.ndimage

The median filter is efficient in the case of salt-and-pepper noise. However, it replaces every pixel value (first problem) by the median value determined at a given scale (second problem). In order to avoid these two problems, Gonzalez and Woods proposed an algorithm [2].

Data: Input (noisy) image I
Data: Maximal scale S_{max}
Result: Filtered image F

Main Function `amf`(I, S_{max})**:**
  forall *pixels* (i, j) **do**
    $S \leftarrow 1$
    while `isMedImpulseNoise`(I, i, j, S) *AND* $S \leq S_{max}$ **do**
      $S \leftarrow S + 1$
      $med \leftarrow$ `Med`(I, i, j, S)
    end
    if $I(i, j) =$`Min`(I, i, j, S) OR $I(i, j) =$`Max`(I, i, j, S) OR $S = S_{max}$ **then**
      $F(i, j) \leftarrow$ `Med`(I, i, j, S)
    else
      $F(i, j) \leftarrow I(i, j)$
    end
  end

Algorithm 2: Adaptive median filter [2]. The main function takes two arguments: the image I to be filtered and the maximal size of neighborhood S_{max}. For each pixel (i, j), the good scale is the scale where the median value is different from the min and the max (median value is thus not an impulse noise). Then, if the pixel value is an impulse noise or the maximal scale has been reached, it must be filtered. Otherwise, it is kept untouched. Min, Max and Med functions compute the minimum, maximum and the median value in a neighborhood of size S centered at a pixel (i, j) in image I.

Function `isMedImpulseNoise`(I, i, j, S)**:**
  $med \leftarrow$ `Med`(I, i, j, S)
  if *med*=`Max`(I, i, j, S) OR *med*=`Min`(I, i, j, S) **then**
    return True
  else
    return False
  end

Algorithm 1: End.

Implement the adaptive median filter from the following algorithm (Alg.2) based on two steps (the operator acts on an operational window of size $k \times k$ where different statistics are calculated: median, min or max).

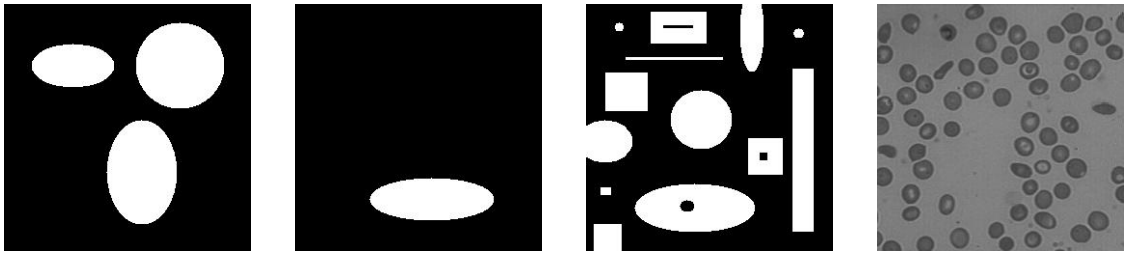
6 Binary Mathematical Morphology

The objective of this tutorial is to process binary images with the elementary operators of mathematical morphology. More particularly, different image transformations, based on the morphological reconstruction, will be studied (closing holes, removing small objects…).

The different transformations will be applied on the following images Fig. 6.1:

Figure 6.1: Images to use for this tutorial.

(a) I_1. (b) M. (c) I_2. (d) Cells.



6.1. Introduction to mathematical morphology

Mathematical morphology started in the 1960s with Serra and Matheron [8]. It is based on Minkowski addition of sets. The main operators are erosion and dilation, and by composition, opening and closing. More informations can be found in [10].

The erosion of a binary set A by the structuring element B is defined by:

$$\varepsilon_B(A) = A \ominus B = \{z \in A | B_z \subseteq A\} \tag{6.1}$$

where, $B_z = \{b + z | b \in B\}$.

The dilation can by obtained by:

$$\delta_B(A) = A \oplus B = \{z \in A | (B^s)_z \cap A \neq \varnothing\} \tag{6.2}$$

where $B^s = \{x \in E | -x \in B\}$ is the symmetric of B.

By composition, the opening is defined as: $A \circ B = (A \ominus B) \oplus B$. The closing is defined as: $A \bullet B = (A \oplus B) \ominus B$.

6.2. Elementary operators

- Create a square structuring element by simply generating a square matrix of ones.
- Create a circular structuring element.
- Test the functions of dilation, erosion, opening and closing on the image I_2 by varying:
  1. the shape of the structuring element,
  2. the size of the structuring element.

The python functions come from the python module scipy.ndimage.morphology. Useful functions are binary_dilation, binary_erosion, binary_opening and binary_closing.
There are multiple modules and functions for reading images; among them, you could use imagio.imread or skimage.io.imread.

For generating a circular structuring element, the following function makes use of the numpy.meshgrid functions.

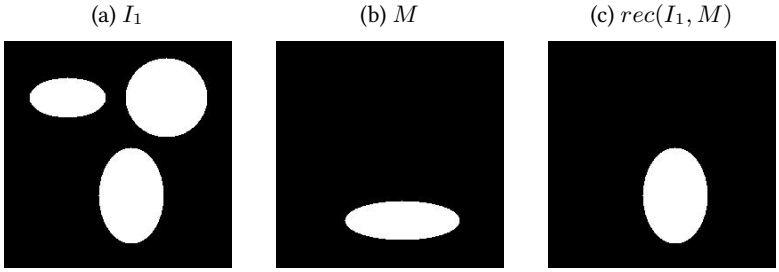
```
def disk(radius):
    # circular structuring element with a given radius
    x = np.arange(-radius, radius+1, 1)
    xx, yy = np.meshgrid(x, x)
    d = np.sqrt(xx**2 + yy**2)
    return d <= radius
```

6.3. Morphological reconstruction

The operator of morphological reconstruction ρ is very powerful and largely used for practical applications. The principle is very simple. We consider two binary images: I_1 (the studied binary image) and M (the marker image). The objective is to reconstruct the elements of I_1 marked by M as illustrated in the Fig. 6.2.

To do this, we iteratively dilate the marker M while being included in I_1 (see Eq.6.3). In order to guarantee this inclusion in A, we keep from each dilated set its intersection with I_1. The algorithm is stopped when the process dilation-intersection is equal to the identity transformation (convergence).

Figure 6.2: Illustration of morphological reconstruction of I_1 by M.



Let $\delta_I^c(M) = \delta_{B_1}(M) \cap I$ be the dilation of the marker set M constrained to the set I. Then, the morphological reconstruction is defined as:

$$\rho_I(M) = \lim_{n \to \infty} \underbrace{\delta_I \circ \ldots \circ \delta_I(M)}_{n \text{ times}} \tag{6.3}$$

Data: image I and marker M
Result: reconstructed image $rec(I, M)$
$r = area(M)$;
$s = 0$;
while $r \neq s$ **do**
  $s = r$;
  $M = I \cap (M \oplus B_1)$;
  $r = area(M)$;
end
$rec(I, M) = M$;

Algorithm 2: The algorithm of this morphological reconstruction

1. Implement the algorithm.
2. Test this operator with the images I_1 and M.

Evaluate the area of a set may be done by counting the number of its pixels.

6.4. Operators by reconstruction

Using the reconstruction operator, implement the 3 following transformations:

1. removing the border objects,
2. removing the small objects,
3. closing the object holes.

Test these operators on the image I_2.

The morphological reconstruction function to use is binary_propagation in the module ndimage.morphology.

6.5. Cleaning of the image of cells

1. Threshold the image of cells (Fig. 6.1d).
2. Process the resulting binary image with the 3 cleaning processes of the previous question.

7 Hough transform and line detection

This tutorial introduces the Hough transform. Line detection operators are implemented.

7.1. Introduction

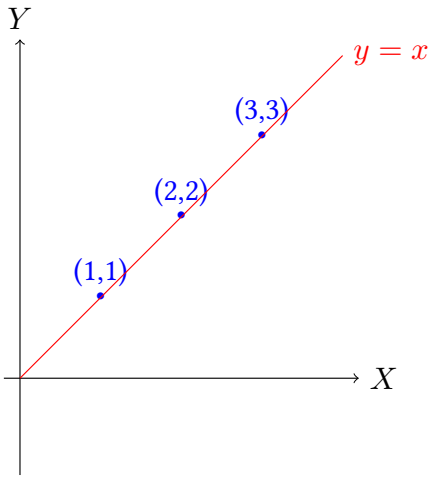
This tutorial deals with line detection in an image. For a given point of coordinates (x, y) in $\mathbb{R}^2$, there exists an infinite number of lines going by this point, with different angles θ. These lines are represented by the following equation:

$$\rho = x \cdot \cos(\theta) + y \cdot \sin(\theta).$$

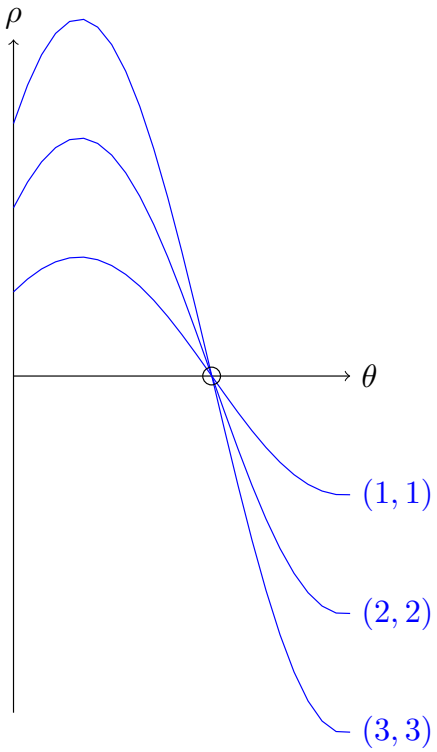
Thus, for each point (x, y) (Fig. 7.1a) corresponds a curve parametered by $[\theta, \rho]$, where $\theta \in [0;2\pi]$ (Fig. 7.1b). The intersection of these curves represents a line (in this case, $y = x$).

Figure 7.1: Representation of the Hough transform.

(a) Different points in $\mathbb{R}^2$.



(b) Hough transform of the 3 points.



7.2. Algorithm

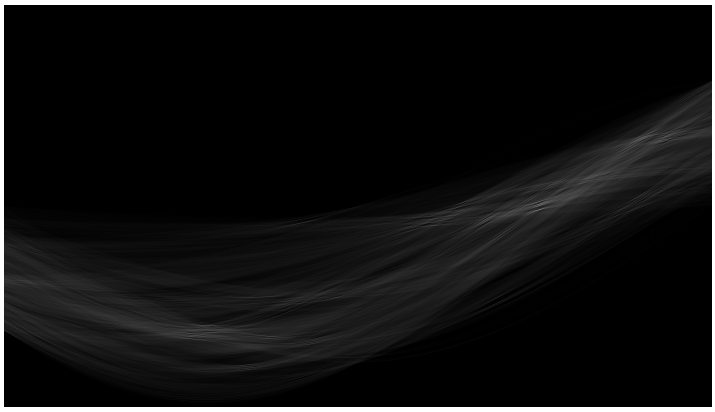
The (general and simple) method for line detection is then:

1. Compute contours detections (get a binary image BW).
2. Apply the Hough transform on the contours BW.
3. Detect the maxima of the Hough transform.
4. Get back in the Euclidean space and draw the lines on the image.

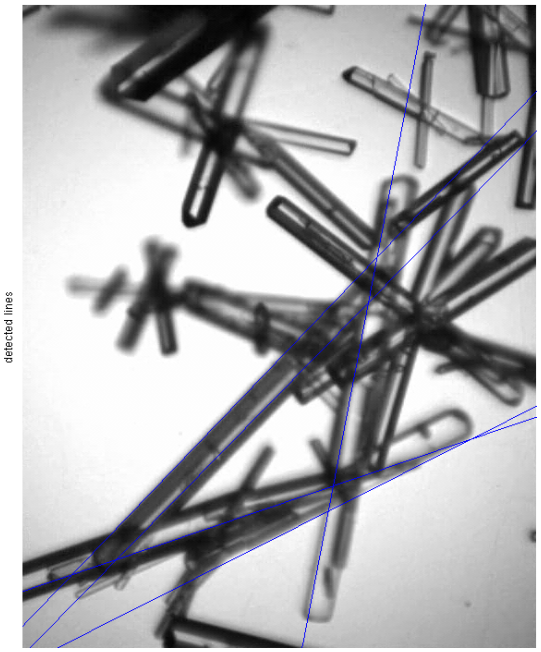
Results should look like in Fig. 7.2.

Figure 7.2: Lines detection via Hough transform.

(a) Hough transform and maxima detection. Angles θ are represented in abscissa, pixels ρ are represented in ordinates. The detection of the absolute maxima of this images will lead to the lines.



(b) Line detection.



7.3. Hough transform

Code a function that will transform each point of a binary image into a curve in the Hough space. For each curve, increment each pixel by one in the Hough space.

7.4. Maxima detection

Use or code a function to detect maxima (regional maxima). For each maximum, keep only one point.

7.5. Display lines

For each maximum, display the corresponding line above the original image.

8 Shape Diagrams

The objective is to study some shape diagrams and the possibility to define properties that may be useful in order to distinguish the different objects.

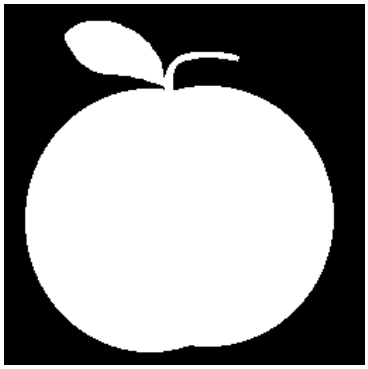
Shape diagrams are representations of single shapes (connected compact sets, see [4, 5, 6]) as points in the 2-D unit square plane. They are based on inequalities between 6 geometrical measurements: area A, perimeter P, radius of the inscribed circle r, radius of the circumscribed circle R, minimum Feret diameter ω and maximum Feret diameter d. In this way, the morphometrical functionals used in the different shape diagrams are normalized ratios of such geometrical functionals. The following table shows the morphometrical functionals for non-convex sets:

Geometrical functionals	Inequalities	Morphological functionals
r, R	$r \leq R$	r/R
ω, R	$\omega \leq 2R$	$\omega/2R$
A, R	$A \leq \pi R^2$	$A/\pi R^2$
d, R	$d \leq 2R$	$d/2R$
r, d	$2r \leq d$	$2r/d$
ω, d	$\omega \leq d$	ω/d
A, d	$4A \leq \pi d^2$	$4A/\pi d^2$
R, d	$\sqrt{3}R \leq d$	$\sqrt{3}R/d$
r, P	$2\pi r \leq P$	$2\pi r/P$
ω, P	$\pi\omega \leq P$	$\pi\omega/P$
A, P	$4\pi A \leq P^2$	$4\pi A/P^2$
d, P	$2d \leq P$	$2d/P$
R, P	$4R \leq P$	$4R/P$
r, A	$\pi r^2 \leq A$	$\pi r^2/A$
r, ω	$2r \leq \omega$	$2r/\omega$

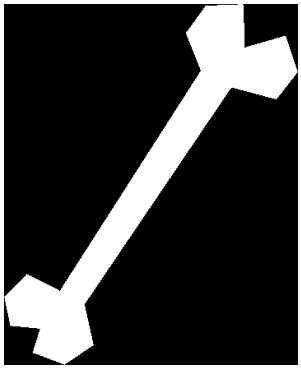
Table 8.1: Morphometrical functionals.

Figure 8.1: The different processes will be applied on images from the Kimia database [1, 9].

(a) Apple.



(b) Bone.



(c) Camel.

8.1. Geometrical functionals

Code functions in order to evaluate the different parameters:

- the area, Crofton perimeter and Feret diameters have been already presented in the tutorial about integral geometry;
- the radius of the inscribed circle can be defined from the ultimate erosion of a set.

The function scipy.ndimage.morphology.distance_transform_cdt computes the distance map of a binary image (chamfer distance transform).

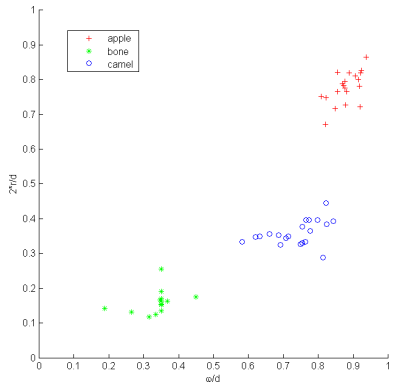
8.2. Morphometrical functionals

Code and evaluate some of the morphometrical functionals listed in the table 8.1. Note that each of them has a physical meaning, e.g. $\frac{4\pi A}{P^2}$ (circularity), $\frac{4A}{\pi d^2}$ (roundness), $2\omega/P$ (thinness).

8.3. Shape diagrams

- Visualize the different shape diagrams for all the images (from the Kimia database) within the three classes 'apple', 'bone' and 'camel'. The Fig.8.2 illustrates the result for the shape diagram $(x = 2r/d, y = P/\pi d)$.
- Which shape diagram is the most appropriate for the discrimination of such objects?

Figure 8.2: Example of a shape diagram.



8.4. Shape classification

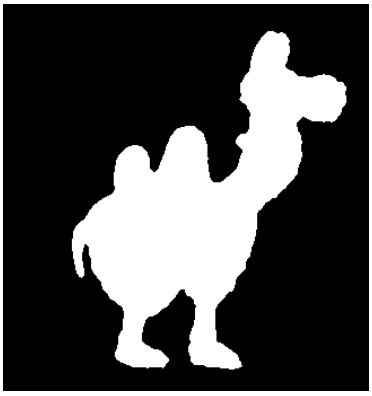
- Use a K-means clustering method for automatic classification of such shapes.
- Propose a method to quantify the classification accuracy for each shape diagram.

9 Image Characterization

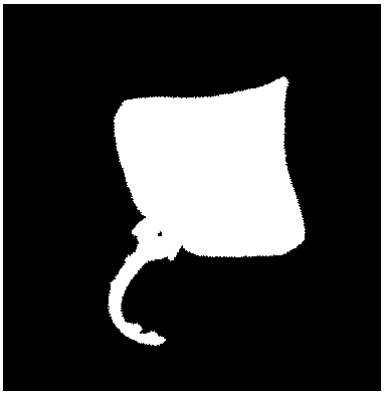
This tutorial aims to characterize objects by geometrical and morphometrical measurements.

The different processes will be applied on synthetic images as well as images from the Kimia database [1, 9]:

(a) Bat.



(b) Camel.



(c) Ray.

9.1. Perimeters

We are going to calculate the perimeter using the Crofton formula. This formula consists in integrating the intercept number of the object with lines of various orientation and positions. Its expression in the 2-D planar case is given by:

$$P(X) = \pi \int \chi(X \cap L) dL$$

where the Euler-poincaré characteristic χ is equal to the number of connected components of the intersection of X with a line L.
In the discrete case, the Crofton formula can be estimated by considering the intercept numbers for the horizontal i_0, vertical $i_{\pi/2}$ and diagonal orientations $i_{\pi/4}$ and $i_{3\pi/4}$ as:

$$P(X) = \pi \times \frac{1}{4}\left(i_0 + \frac{1}{\sqrt{2}} i_{\pi/4} + i_{\pi/2} + \frac{1}{\sqrt{2}} i_{3\pi/4}\right)$$

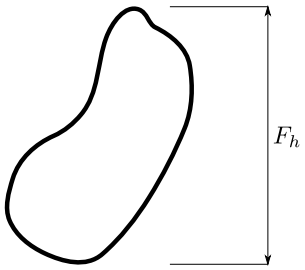
1. Calculate the intercept number of a binary object from the Kimia database with lines oriented in the following four directions: $0, \pi/4, \pi/2, 3\pi/4$.
2. Deduce the value of the Crofton perimeter.
3. Compare the result with the classical perimeter functions.

See perimeter from skimage.measure.

9.2. Feret Diameter

The Feret diameter (a.k.a. the caliper diameter) is the length of the projection of an object in one specified direction Fig.9.1.

Figure 9.1: Feret diameter of the object in horizontal direction.



1. Calculate the projections in different directions of a binary object from the Kimia database.
2. Deduce the minimum, maximum and mean value of the Feret diameters.

9.3. Circularity

We want to know if the object X is similar to a disk. For that, we define the following measurement (circularity criterion):

$$circ(X) = \frac{4\pi A(X)}{P(X)^2}$$

where $A(X)$ and $P(X)$ denote the area and perimeter of the object X.

1. Show that the circularity of a disc is equal to 1.
2. Generate an array representing an object as a discrete disc.
3. Calculate its circularity and comment the results.

Use numpy.meshgrid.

9.4. Convexity

We want to know if the object X is convex. For that, we define the following measurement:

$$conv(X) = \frac{A(X)}{A(CH(X))}$$

where $CH(X)$ denotes the filled convex hull of the object X.

1. Compute the convex hull of a pattern from the Kimia database.
2. Evaluate the area of the filled convex hull.
3. Deduce the convexity of the pattern.

See ConvexHull from scipy . spatial .

machine learning

10 Machine Learning

The objective of this tutorial is to classify images by using machine learning techniques. Some images, as illustrated in Figure 10.1 of the Kimia database will be used [1, 9].

Figure 10.1: The different processes will be applied on images from the Kimia database. Here are some examples.

(a) bird

(b) camel

(c) ray

(d) turtle

10.1. Feature extraction

The image database used in this tutorial is composed of 18 classes. Each class contains 12 images. All these 216 images come from the Kimia database. In order to classify these images, we first have to extract some features of each binary image.

1. For each image in the database, extract a number of different features, denoted *nbFeatures*.
2. Organize these features in an array of *nbFeatures* lines and 216 columns. In this way, each column i represents the different features of the image i.

You can use the Python function skimage.measure.regionprops to extract the same geometrical features. In order to load all image, from the directory, you can use this for loop and use the glob module in order to load all files.

```
import glob
rep = 'images_Kimia216/'
classes = ['bird', 'bone', 'brick', 'camel', 'car', 'children',
           'classic', 'elephant', 'face', 'fork', 'fountain',
           'glass', 'hammer', 'heart', 'key', 'misk', 'ray', 'turtle']
nbClasses = len(classes)
nbImages = 12

# The features are manually computed
properties = np.zeros((nbClasses*nbImages, 9))
target = np.zeros(nbClasses * nbImages)
index = 0
for ind_c, c in enumerate(classes):
    filelist = glob.glob(rep+c+'*')
    for filename in filelist :
        print(filename)
```

10.2. Image classification

In order to classify the images, we are going to use neural networks. Pattern recognition networks are feedforward networks that can be trained to classify inputs according to target classes. The inputs are the features of each image. The target data are the labels, indicating the class of each image.

10.2.1. Construction of the array of properties

- Build the target data, representing the class of each image. The target data for pattern recognition networks should consist of vectors of all zero values except for a 1 in element i, where i is the class they are to represent. In our example, the target data will be an array of 18 lines and 216 columns.
- The database will be divided into a training set (75%) and a test set (25%).

Use `from sklearn.model_selection import train_test_split` to perform this partition into training/test sets.

10.2.2. Training and classification

- Run the training task and classify the test images.
- Show the classification confusion matrix as well as the overall performance.

Look at the Python module `sklearn.neural_network.MLPClassifier` to make the classification. You may also use SVM classifier. Notice that your data may be normalized (use `from sklearn.preprocessing import StandardScaler`) before performing the classification.

- Try to run the same process again (starting from the traint/test split). What can you conclude?
- Try to change the parameters of the network to improve the classification performance.

⋆ 11 Harris corner detector

The aim of this tutorial is to develop a simple Harris corner detector. This is the first step in pattern matching, generally followed by a feature descriptor construction, and a matching process.

11.1. Corner detector and cornerness measure

Use the sobel and gaussian_filter from the scipy.ndimage module, with a scale parameter σ.

11.1.1. Gradient evaluation

The Harris corner detector is based on the gradients of the image, I_x and I_y in x and y directions, respectively.

Apply a Sobel gradient in both directions in order to compute I_x and I_y.

11.1.2. Structure tensor

The structure tensor is defined (for each point of coordinates (u, v)) by the following matrix. The coefficients ω follow a gaussian law, and each summation represents a gaussian filtering process. W is an operating window.

$$M(u,v) = \begin{bmatrix} I_x(u,v)^2 & I_x(u,v)I_y(u,v) \\ I_x(u,v)I_y(u,v) & I_y(u,v)^2 \end{bmatrix}$$

This defines 4 matrices M_1, M_2, M_3, M_4.

- Using the Sobel gradients in horizontal and vertical direction, compute M_1 to M_4.
- Use a gaussian filter (with parameter σ) in order to smooth the matrices M_1 to M_4.

11.1.3. Cornerness measure

The cornerness measure C, as proposed by Harris and Stephens, is defined as follows for every pixel of coordinates (u, v):

$$C(u,v) = \det(M(u,v)) - K\text{trace}(M(u,v))^2$$

with K between 0.04 and 0.15.

By using the capacity of numpy to make operations on all values of arrays, compute C for all pixels and display it for several scales σ (parameter used in the gaussian filter).

11.2. Corners detection

A so-called Harris corner is the result of keeping only local maxima above a certain threshold value. You can use the checkerboard image for testing, or load the sweden road sign image Fig.11.1.

Use the following function to generate a checkerboard pattern.

```python
def checkerboard(nb_x=2, nb_y=2, s=10):
    """
    checkerboard generation
    a grid of size 2*nb_x by 2*nb_y is generated
    each square has s pixels.
    """
    C = 255*np.kron([[1, 0] * nb_x, [0, 1] * nb_x] * nb_y, np.ones((s, s)))
    return C
```

Figure 11.1: Sweden road sign to be used for corner detection.

- Evaluate the extended maxima of the image.
- Only the strongest values of the cornerness measure should be kept. Two strategies can be employed in conjonction:
  - Use a threshold value t on C: the choice of this value is not trivial, and it strongly depends on the considered image. An adaptive method would be preferred.
  - Keep only the n strongest values.
- The previous operations are affected by the borders of the image. Thus, eliminate the corner points near the borders.
- The detected corners may contain several pixels. Keep only the centroid of each cluster.

Informations

There are numerous methods for computing local maxima. One can choose between skimage.morphology.local_maxima, skimage.morphology.h_maxima, skimage.feature.peak_local_max.

Bibliography

[1] http://vision.lems.brown.edu/content/available-software-and-databases. 45, 47, 51

[2] R. C. Gonzalez and R. E. Woods. *Digital Image Processing*. Pearson, 2007. 36

[3] N. Otsu. A Threshold Selection Method from Gray-Level Histograms. *Systems, Man and Cybernetics, IEEE Transactions on*, 9(1):62–66, Jan 1979. 22

[4] S. Rivollier, J. Debayle, and J.-C. Pinoli. Shape diagrams for 2d compact sets-part i: analytic convex sets. australian journal of. *The Australian Journal of Mathematical Analysis and applications*, 7(2), 2010. 45

[5] S. Rivollier, J. Debayle, and J.-C. Pinoli. Shape diagrams for 2d compact sets-part ii: analytic simply connected sets. *The Australian Journal of Mathematical Analysis and applications*, 7(2), 2010. 45

[6] S. Rivollier, J. Debayle, and J.-C. Pinoli. Shape diagrams for 2d compact sets-part iii: convexity discrimination for analytic and discretized simply connected sets. *The Australian Journal of Mathematical Analysis and applications*, 7(2), 2010. 45

[7] B. Selig, K. A. Vermeer, B. Rieger, T. Hillenaar, and C. L. Luengo Hendriks. Fully automatic evaluation of the corneal endothelium from in vivo confocal microscopy. *Biomed central*, 2015. 32

[8] J. Serra. *Image Analysis and Mathematical Morphology.* London: academic press, 1982. 39

[9] D. Sharvit, J. Chan, H. Tek, and B. B. Kimia. Symmetry-based indexing of image databases. In *Content-Based Access of Image and Video Libraries, 1998. Proceedings. IEEE Workshop on*, pages 56–62. IEEE, 1998. 45, 47, 51

[10] P. Soille. *Morphological Image Analysis: Principles and Applications.* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2nd edition, 2003. 39