MatImage library user manual

D. Legland

January 17, 2023

Abstract *The MatImage library provides a collection of functions for image processing and analysis within the Matab environment.*

Contents

1 Introduction

1.1 Presentation

MatImage is a library for image processing and analysis, with unified interface for 2D/3D images, or color/gray-scale images. It is build as a complement of the Image Processing Toolbox from Matlab.

The library was develop to facilitate the manipulation and the analysis of images with various dimensionalities (2D/3D) and various contents (grayscale, color, binary, label maps). Some key features of the library are:

- interactive graphical visualization tools for 3D images
- implementation of several functions for quantitative image analysis
- possibility to take into account the spatial calibration of the images for many functions
- generic operators, avoiding when possible the use of specific functions depending on image type or image dimensionality

The official homepage for the project is hosted on GitHub[1]. A starting help is provided in the MatImage wiki.

1.2 Installation

The latest version of the library can be downloaded from the GitHub project page[2].

1.2.1 Installation from zip-file

To install the library from the zip archive, proceed as follow:

1. Download the zip archive corresponding to the latest version
2. Extract the file
3. Start Matlan, and run the “installMatImage.m” script located in the “matImage” folder. This will automatically add the required directories to the “path” variable of the Matlab workspace.
4. To keep the installation for the future, you can save the configuration from the “Set Path” dialog available in the toolbar.

[1] http://github.com/mattools/matImage
[2] https://github.com/mattools/matImage/releases/latest

1.2.2 For developers

It is also possible to clone the project, to facilitate the synchronization with the latest developments, and / or to propose enhancement to the library.

1.2.3 Dependencies

Several functions of the library require the Image Processing Toolbox from Matlab.

Other external libraries are required to run some of the functions:

- the MatGeom library[3] is used for several image analysis functions (chapters 4 and 5) and for the generation of synthetic shapes (chapter 8).
- the GUI Layout Toolbox[4] is required to run the "Slicer" application, that facilitates the interactive exploration of 3D images.

1.3 Library organization

The library is organised into several modules. They are presented in a "from global to technical" order: first the modules for classical image processing and for exploration/visualization, then the modules for analysis/quantification, and finally some more specialized modules.

imFilters generic filters and utilities for image processing.

imStacks functions and graphical user interfaces for the manipulation and the representation of 3D images.

imMeasures functions for quantitative measurements on binary and label images.

imMinkowski estimation of intrinsic volumes (volume, surface area, perimeter, Euler number...) from 2D/3D binary or label images.

imGranulometry computation of granulometric curves based on mathematical morphology.

imGeodesics computation of geodesic distances and of geodesic diameters from 2D/3D images.

imShapes generation of 2D and 3D binary images containing geometric shapes (ellipse, cube, cylinder...).

[3] https://github.com/mattools/matGeom
[4] https://fr.mathworks.com/matlabcentral/fileexchange/27758-gui-layout-toolbox

2 Module imFilters

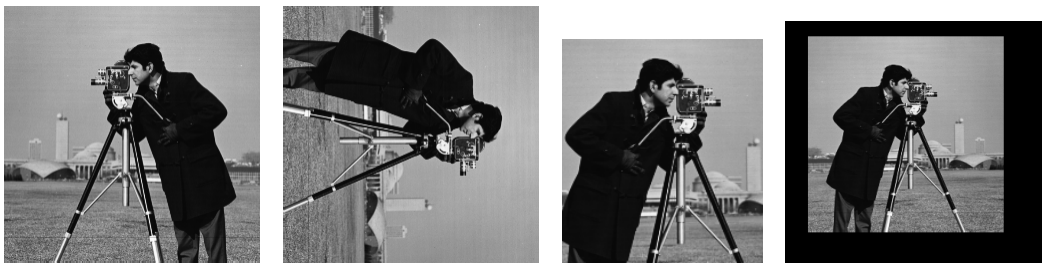
Generic filters for image processing, build as a complement to the Image Processing Toolbox. The aim is to provide a unified interface for the processing or 2D/3D images, and grayscale/color images.

Contents

2.1 Image shape operators

Several operators that change the shape of the image: flip, rotate, resize... They are often used as first pre-processing before more advanced operations. For most functions, axes are indexed in the x, y, z order (contrary to native Matlab functions, indexed in i,j order).

```
% read data
img = imread('cameraman.tif');
% flip in horizontal direction (here, x=1)
imgFlip = imFlip(img, 1);
% rotate by 90 degrees
imgRot90 = imRotate90(img);
% crop borders with various widths
imgCropBorder = imCropBorder(img, [20 40 30 50]);
% add border
imgAddBorder = imAddBorder(img, [20 40 30 50]);
```



(a) flip. **(b)** rotate90. **(c)** cropBorders. **(d)** addBorders.

Figure 2.1: *Several shape operations on images.*

imFlip

Flips an image along one of its dimensions (Fig. 2.1).

imRotate90

Rotates an image by 90 degrees around one of the main axes (Fig. 2.1). Can be applied on 3D images as well.

imAddBorder

Adds a border around a 2D or 3D image (Fig. 2.1).

imCropBorder

Crops the border around a 2D or 3D image (Fig. 2.1).

imTranspose

Transposes an image (grayscale or RGB).

imCropBox

Crops an image with a box.

imResize
Resizes 2D or 3D image.

imDownSample
Subsamples an array by applying operation on blocs.

subsamplergb
Returns a sub-sampled version of a color RGB image.

imPrincipalAxesAlign
Aligns a binary 3D image along the principal axes of the matrix of second-order moments.

2.2 Image filtering

Operators described in this section aim at removing the noise or at enhancing specific structures within the images.

2.2.1 Image enhancement and noise reduction

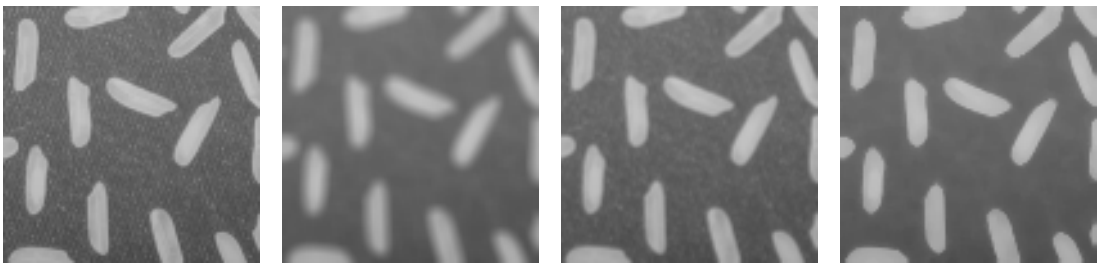
Some function mostly used for removing imaging artifacts.

imAdjustDynamic
Rescales gray levels of image to get better dynamic.

imNormalizeBackground
Normalizes image by removing background estimate.

2.2.2 Noise reduction

Several functions for spatial filtering of images, with the objective of reducing the noise within image.



(a) Original **(b)** Box 5x5. **(c)** Gaussian 2x2 **(d)** Median 5x5

Figure 2.2: *Noise reduction algorithms applied on a portion of a grayscale image.*

imBoxFilter
Box filter on 2D/3D image (Fig. 2.2-b).

imMeanFilter
Computes mean value in the neighborhood of each pixel.

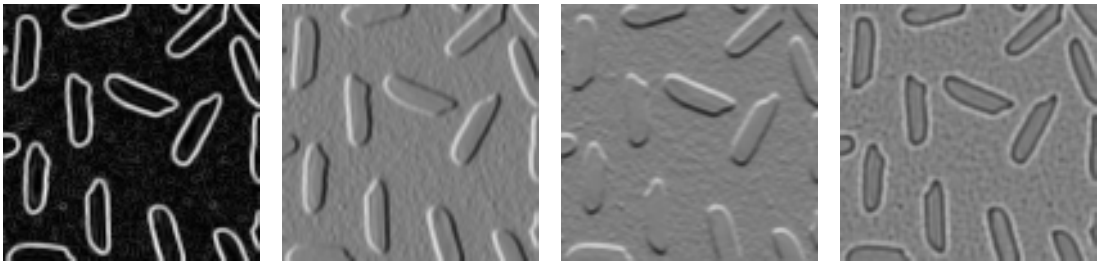
imMedianFilter
Computes median value in the neighborhood of each pixel (Fig. 2.2-d).

imGaussianFilter
Applies gaussian filter to 2D/3D image (Fig. 2.2-c). The functions uses separability of the kernel to accelerate the processing.

imDirectionalFilter
Applies and combines several directional filters.

2.2.3 Gradient and Laplacian filters

Extraction of various gradients from the images, that can be used for edge detection.



(a) Gradient Norm. **(b)** Gradient X. **(c)** Gradient Y. **(d)** Laplacian.

Figure 2.3: *Edge detection filters.*

imGradientFilter
Computes gradient components of a grayscale image. The result can be provided as a single image (corresponding to the the gradient norm, see Fig. 2.3-a), or as two (or three) images, corresponding to the gradient in each elementary direction (Fig. 2.3-b,c).

imLaplacian
Discrete Laplacian of an image (Fig. 2.3-d).

imRobinsonFilter
Extracts image edges using Robinson directional filters.

imKirschFilter
Extracts image edges using Kirsch directional filters.

imHessian
Computes the coefficients of the Hessian matrix (containing all second derivatives) for each pixel.

imEigenValues
Image eigen values from second derivatives.

imEigenValues3d
Image eigen values from second derivatives.

imVesselness2d
Computes the local vesselness of curvilinear structures, from Frangi's paper (Frangi et al., 1998).

2.2.4 Morphological filters

Additional morphological filters, as a complement to the ones implemented in the Image Processing Toolbox. A good reference for morphological filters is the book of Soille (2003).

imMorphoGradient
Morphological gradient of an image.

imMorphoLaplacian
Morphological laplacian of an image.

imrecerode
Performs a morphological reconstruction by erosion.

imHConcave
H-concave transformation of an image.

imHConvex
H-convex transformation of an image.

2.3 Segmentation

Several functions for facilitating segmentation of images.

imOtsuThreshold
Thresholds an image using Otsu method.

imMaxEntropyThreshold
Computes image threshold using maximisation of entropies.

imMultiOtsuThreshold
Computes a segmentation of the input grayscale image into an arbitrary number of classes, using minimisation of intra-class variances.

imImposedWatershed
Computes watershed after imposition of extended minima.

imCannyEdgeDetector
Edge detection using Canny-Deriche method.

2.4 Color images and gray-scale conversions

Several functions allows for converting images into color images, and extracting specific information from color images.

imOverlay
Generates a new color image by adding a binary overlay over another image (2D or 3D, grayscale or color).

imSplitChannels
Splits the 3 channels of a 2D or 3D image.

imMergeChannels
Merges 3 channels to create a 2D or 3D color image.

double2rgb
Creates a RGB image from double values.

angle2rgb
Converts an image of angles to color image.

imGetHue
Extracts hue of a color image, using the rgb2hsv function.

imGray12ToGray8
Converts a 12 bits gray scale image to 8 bits gray scale.

2.5 Binary and label image operators

Binary and label images are a convenient way of representing the result of segmentation algorithms. The MatImage library provides several functions for processing them.

2.5.1 Filters for binary images

Several functions are devoted to the processing of binary images. Fig. 2.4 are obtained from the following commands:

```
img = imread('circles.png');
bnd = imBoundary(img);
imgFH = imFillHoles(img);
skel = imSkeleton(img);
cvx = imConvexImage(img);
```

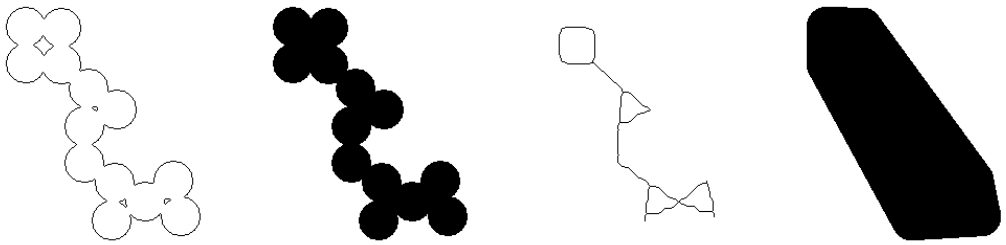


Figure 2.4: *Several operators applied on the "circles.png" binary image: boundary, fill holes, skeleton, convex hull. Images are displayed with inverted LUT.*

imBoundary

Computes the boundary of a binary image. The result is given as another binary image (Fig. 2.4-a).

imFillHoles

Fills the holes in a binary image (Fig. 2.4-b).

imSkeleton

Homothopic skeleton of a binary image (Fig. 2.4-c).

imLabelSkeleton

Labels skeleton pixels according to local topology.

imChainPixels

Chains neighbor pixels in an image to form a contour.

imConvexImage

Computes smallest convex image containing the original pixels (Fig. 2.4-d).

2.5.2 Distance maps

When analyzing images, it is often necessary to compute distances to a particular structure or position. A convenient operator for binary images is the **distance transform**. Its principle is to compute, for each foreground pixel, the distance to the nearest background pixel. The result is commonly referred to as **Distance Map** (Fig. 2.5).

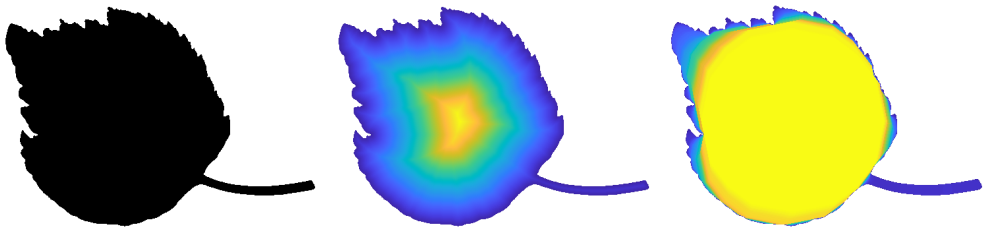


Figure 2.5: *A binary image, and the results of distance map and local thickness map.*

The image processing toolbox in Matlab provides the bwist function. As a complement, the MatImage toolbox provides the imDistanceMap function, that is based on Chamfer distances (Borgefors, 1984, 1986). It uses the same metric as geodesic distances (Chapter 7).

imDistanceMap
Computes distance from a binary image using chamfer distances (Fig. 2.5). Note that the imDistanceMap function computes the distance for each pixel within the region to the nearest background pixel or voxel, whereas the bwdist function computes the distance of each background pixel or voxel to the nearest pixel or voxel within the region.

imDistanceClasses
Converts a distance map into a label image of regions, by quantifying the distance values.

imThicknessMap
Computes thickness map of a binary image. Thickness corresponds to the size of the largest disk contained in the region and that contains the considered pixel (Fig. 2.5).

imDistance
Distance map computed from a set of points.

imDistance3d
Creates a 3D distance map from a set of 3D points.

2.5.3 Filters for binary/label images

Several functions are devoted to the processing of label images representing a collection of regions.

imFindLabels
Finds the unique labels in a label image. This function is used by several functions from the imMeasures module (chapter 4).

label2rgb3d
Converts a 3D label image to a 3D RGB image.

imLabelToValue
Converts a label image to a parametric map using an array of feature values computed for each label.

imKillBorderRegions
Removes regions on the border of an image.

imAreaOpening
Removes all regions smaller than a given area.

imAttributeOpening
Filters regions on a size or shape criterium.

imSeparateParticles
Separates touching particles using watershed algorithm applied on the inverse of the distance function.

imLargestRegion
Keeps the largest region in a binary or label image.

imCropLabel
Extracts the portion of image that contains the specified label.

imMergeLabels
Merges regions in a labeled image.

mergeRegions
Merges regions of labeled image, using inclusion criteria.

imBoundaryIndices
Finds the indices of the boundary between 2 regions.

imLabelEdges
Labels edges between adjacent regions of labeled image.

2.6 Utility functions

2.6.1 Utilities and drawing

The functions in this section aim at facilitating the representation and comparison of images.

imCreate
Creates a new image with given the size and type.

imCheckerBoard
Creates a checkerboard image from 2 images.

imCheckerboardLabels
Creates a checkerboard label image.

imThumbnail
Resizes an image to bound the size in each direction.

imDrawLine
Draws a line between two points in the image.

bresenhamLine
Integer coordinates of a bresenham line.

imDrawText
Draws some text in an image.

imTpsWarp
Warps an image using Thin-Plate Splines transform.

2.6.2 Kernels and structuring elements

Utility functions that are either used by other functions, or that can be used as argument of other functions.

ball
Generates a ball in a matrix in 2 or 3 dimensions.

gaussianKernel3d
Creates a 3D Gaussian kernel for image filtering.

orientedGaussianKernel
Oriented Gaussian kernel for directional filtering.

orientedLaplacianKernel
Oriented Laplacian kernel for directional filtering.

cross3d
Returns a 3D structuring element with cross shape.

intline
Integer-coordinate line drawing algorithm.

strelDisk
Discrete disk structuring element.

imNeighborhood
Returns the neighborhood of a given pixel.

2.6.3 Configuration map images

Most functions in this section work with **configuration maps**. Configuration maps are obtained from binary images, by replacing each binary 2×2 or $2 \times 2 \times 2$ configuration by the index of the configuration. The number of configuration indices is $16 = 2^4$ for 2D binary images, and $256 = 2^8$ for 3D binary images. More information about configuration maps can be found in Ohser & Schladitz (2009).

grayFilter
Computes configuration map of a binary image.

grayHist
Computes frequencies of configurations in binary images.

imLUT
Applies a look-up table (LUT) to a gray-scale image.

createTile
Creates a binary tile (2x2) from its index.

tileIndex
Returns the index of a 2x2 binary tile.

createTile3d
Creates a 3D binary tile (2x2x2) from its index.

tileIndex3d
Returns the index of a 2x2x2 binary tile.

3 Module imStacks

The imStacks modules provides several functions and graphical user interfaces for the manipulation and the representation of 3D images.

Contents

3.1 Slicer application

"Slicer" is a graphical application that allows for quick interactive exploration of 3D grayscale or color images, including the computation of histograms, isosurfaces, or cross-sections (Fig. 3.1).

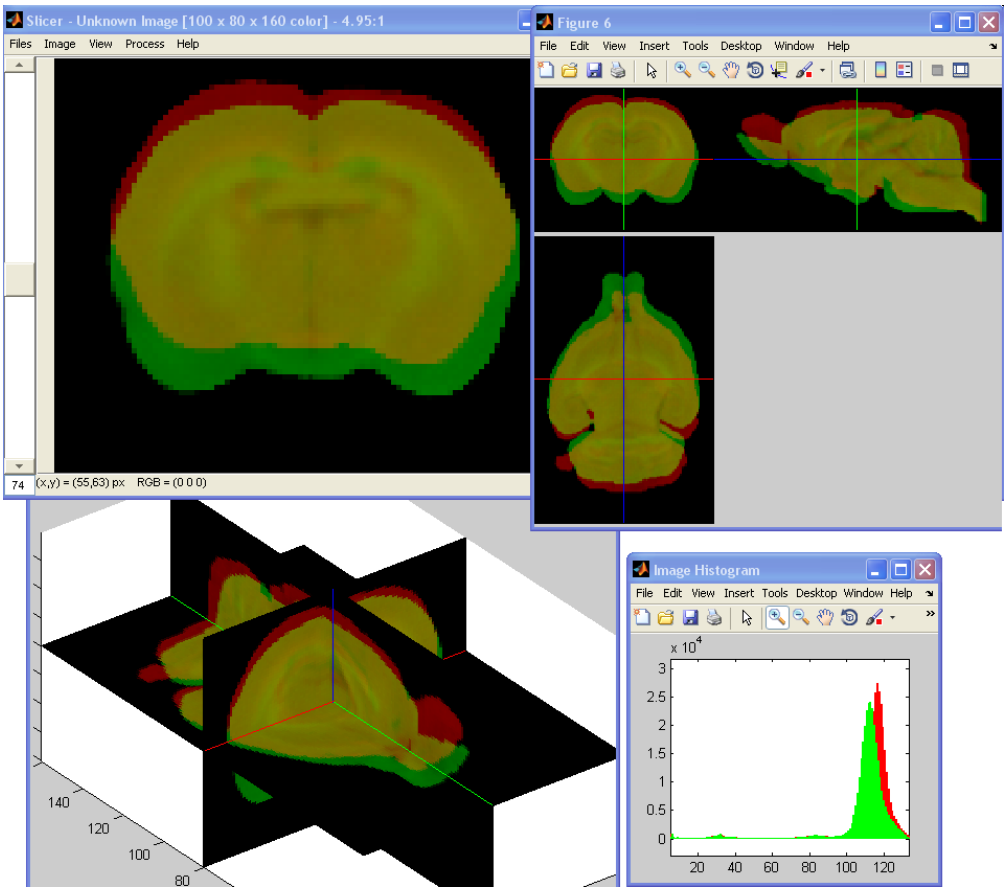


Figure 3.1: *Several views of the Slicer application on a sample 3D color image.*

The main interface is a slice-by-slice viewer that integrates slice changing slider, status bar, and an action menu. The menu provides several actions to import/export 3D images, to apply various operations on the 3D image (crop, rotation...), and to generate new visualizations of the 3D data: isosurfaces, 3D orhoslices, histograms... It is possible to specifiy the spatial calibration, facilitating the visualisation of 3D structures with non-cubic voxels.

All the features of the Slicer application are available as Matlab functions of the "stacks" module.

3.1.1 Menu description

3.1.1.1 File menu

Contains several options to import/export 3D images. See also the section 3.4.

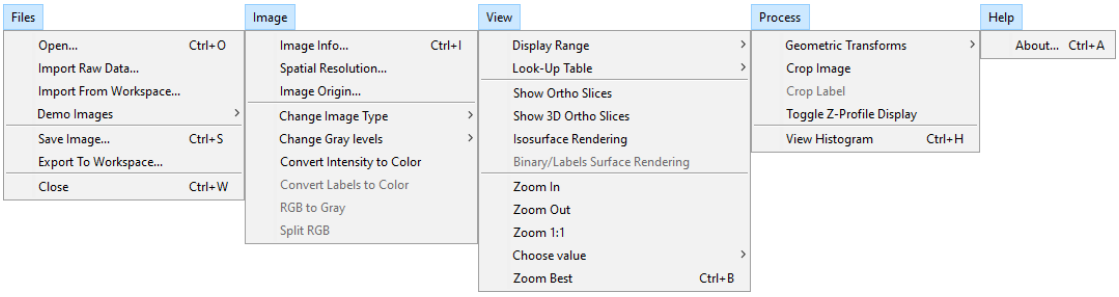


Figure 3.2: *The menu bar of the Slicer application.*

Open...
Opens a 3D image. Several common file formats are supported (TIFF, Analyze, MHD, VGI...).

Import Raw Data....
Imports image data from a binary data file.

Import From Workspace
Displays a 3D array from Matlab workspace into a new Slicer frame.

Demo images
Opens a selection of demonstration images.

Save image
Saves current 3D image.

Export To Workspace
Exports current image into workspace as a 3D or 4D numeric array.

Close
Closes the current viewer.

3.1.1.2 Image Menu

Allows for calibrating the image, and converting data types.

Image Info
Display a short description of image properties.

Spatial Calibration
Updates sptial calibration (voxel size and unit name) of image.

Image Origin
Updates the coordinates of first voxel in user coordinates.

Change Image Type
Changes the type associated to the current image. The “Type” can be one of “Grayscale”, “Intensity”, “Binary”, “Label”.

Change Gray Levels
Changes the number of levels used to represent grayscale image data.

Convert Intensity to Color
Converts a scalar image into a color image by choosing a colormap. See also section 2.4 (function double2rgb).

Convert Labels to Color
Converts a label image into a color image by choosing a colormap. See also section 2.5.3 (function label2rgb3d).

RGB To Gray
Converts a RGB color image into a grayscale image.

Split RGB
Splits the color components of a 3D color image into three distinct images. Each component is displayed into a new Slicer frame.

3.1.1.3 View Menu

Allows for tuning display settings, and computing 3D representations.

Display Range
Chooses the range of values used to display grayscale or intensity images. Can be chosen automatically from image data values, inferred from image data type, or manually selected.

Look-Up Table
Chooses the look-up table to use for representing intensity or label images.

Show Ortho Slices
Displays the 3D image as three 2D images representing XY, YZ and XZ slices (See Fig. 3.3-a). Each slice is displayed as a 2D image. The slice position can be updated by clicking on the images.

Show 3D Ortho Slices
Displays the 3D image as three 2D images mutually intersecting in 3D (See Fig. 3.3-b). Each slice is represented in 3D. The slice position can be updated by dragging the 3D images.

Isosurface rendering
Computes and renders a 3D isosurface from image data and user-defined value. Can be slow for large images.

Binary/Labels surface rendering
Computes and renders a 3D isosurface from binary and label images. Similar to isosurface rendering, but a specific processing is added to iterate over the set of distinct labels within image.

Zoom In/Out/1:1/Best...
Changes the zoom level of the current image view.

3.1.1.4 Process Menu

Provides basic image processing tools.

Geometric Transforms
Performs elementary shape transformation on the 3D image: flip along one of the main axes, rotation by 90 degrees around one of the main axes.

Crop Image
Extracts a 3D rectangular selection from the image.

Crop Label
Extracts the 3D region containing the specified label. Returns a new label image with only one label.

View Histogram
Displays the histogram of the grayscale or color values within the image. See also the imHistogram function (p. 30).

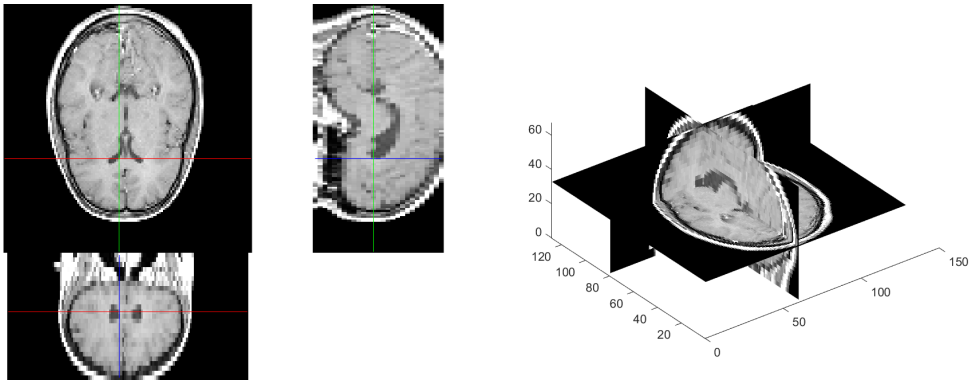
3.2 Image exploration

The imStacks modules provides several tools for interactive exploration of 3D images. Most functions work independently for both grayscale or color images, and allow to specifiy the spatial calibration of images. Several histograms functions are also provided in the measure module (see section 4.1.1).

3.2.1 3D Images

Several functions are specifically dedicated to the exploration of 3D images, or more generally to 3D arrays.

```
% read data and adjust dynamic
img = imAdjustDynamic(analyze75read('brainMRI.hdr'));
% Display orthogonal slices
figure(1); clf; axis equal; hold on;
orthoSlices(img, [60 80 13], 'Spacing', [1 1 2.5]);
% Display 3D orthogonal slices
figure(2); clf; axis equal; hold on;
orthoSlices3d(img, [60 80 13], [1 1 2.5]);
% setup axis limits
axis(imPhysicalExtent(img, [1 1 2.5])); view(3);
```



(a) showOrthoPlanes **(b)** showOrthoSlices3d

Figure 3.3: *Different representations of a 3D grayscale image. Results of the showOrthoPlanes and showOrthoSlices3d functions.*

orthoSlices

Displays three orthogonal slices in the same figure (Fig. 3.3-a).

OrthoSlicer3d

Displays 3D interactive orthoslicer.

orthoSlices3d
Shows three orthogonal 3D slices of a 3D image (Fig. 3.3-b). The slices can be dragged interactively.

3.2.2 Interactive tools

Functions for semi-interactive image exploration, such as computation of line profiles.

imLineProfile
Evaluates image value along a line segment.

imEvaluate
Evaluates image value at given position(s).

3.3 Get information on images

Some functions to query the physical size or type of the image.

imSize
Computes the size of an image in [x y z] order. For color images, returns only the spatial components.

is3DImage
Checks if an image is 3D.

isColorImage
Checks if an image is a color image.

imPhysicalExtent
Computes the physical extent of an image. Returns a 1-by-4 row array for 2D images, and a 1-by-6 row array for 3D images.

imGrayscaleExtent
Grayscale extent of an image.

3.3.1 Stacks

stackSize
Computes the size of a 3D stack in [x y z] form.

stackExtent
Computes the physical extent of a 3D image.

isColorStack
Checks if a 3D stack is color or gray-scale.

3.4 Read/Write 3D images

Several functions are provided for importing and exporting 3D images in various file formats, or for reading only a specific portion of a large 3D image.

3.4.1 General use functions

readstack
Reads a 3D image stored either in a list of 2D images (slices), or in a 3D image file. The function can parse some of the meta-data stored by ImageJ.

savestack
Saves an image stack to a file or a series of files.

savebinstack
Saves an binary stack to a file, as RGB Image.

imReadRawData
Reads image data from raw data file.

imFileInfo
Generalization of the imfinfo function that manages 3D image file formats.

3.4.2 Read image portion

When the size of the image is very large, it may be useful to be able to read only a specific rectangular region within the image, or to read a down-sampled version of the image. Note that the `tiffreadVolume` from the Image Processing Toolbox also provides similar functionnalities.

imReadRegion3d
Reads a specific 3D region of a 3D image.

imReadDownSampled3d
Reads a down-sampled version of a 3D image.

3.4.3 Management of specific file formats

In addition to the image file formats available with Matlab, MatImage also provides partial support of additional image file formats.

metaImageInfo
Reads information header of meta image data.

metaImageRead
Reads an image in MetaImage file format[1].

[1] https://itk.org/Wiki/ITK/MetaIO/Documentation

metaImageWrite
Writes header and data files of an image in MetaImage format. This results in a header file with extension ".mhd", and a binary data file with extension ".raw".

readVgiStack
Reads a 3D stack stored in VGI format.

vgiStackInfo
Reads information necessary to load a 3D stack in VGI format.

readVoxelMatrix
Reads a 3D image in VoxelMatrix (.vm) format, used by the Free-D library (Andrey & Maurin, 2005).

3.5 3D images processing

Some specific utility functions for processing 3D images.

createRGBStack
Concatenates 2 or 3 grayscale stacks to form a color stack.

stackSlice
Extracts a planar slice from a 3D image.

stackRotate90
Rotates a 3D image by 90 degrees around one image axis.

rotateStack90
Rotates a 3D image by 90 degrees around one image axis.

flipStack
Flips a 3D image along specified X, Y, or Z dimension.

cropStack
Crops a 3D image with the specified box limits.

imMiddleSlice
Extracts the middle slice of a 3D stack.

3.5.1 Visualisation routines

These low-level functions are used by other visualization functions, but can be used independently.

slice3d
Shows a moving 3D slice of an image.

showXSlice
Shows YZ slice of a 3D image.

showYSlice
Shows ZX slice of a 3D image.

showZSlice
Shows XY slice of a 3D image.

4 Module imMeasures

Provides several functions for measurements on digital images. Some functions are simple wrappers that manage 3D and/or color images, as well as eventual type conversion. Some geometrical measurements are also provided, for the quantitative analysis of regions in binary or label images.

Additional quantifications are provided in the imMinkowski module (area, volume, perimeter, surface area, mean breadth, Euler number for 2D and 3D binary/label images) and in the imGranulometry module.

Contents

4.1 Statistics for pixel values

4.1.1 Image histograms

Several functions allows for computing histogram of grayscale or color images. They can be applied to 2D/3D images. The following example provides results of Figure 4.1:

```
img = imread('peppers.png');
% Histogram of each channel
figure; imHistogram(img);
ylim([0 8000])
% Joint histogram of red and green channels
h = imJointHistogram(img(:,:,1), img(:,:,2));
figure; imshow(log(h+1), []); colormap([1 1 1; parula(256)]);
```

imHistogram

Computes the histogram of the input image. In case of a color images, returns or display the histogram of each channel.

imHistogramDialog

Opens a dialog to setup image histogram display options.

imJointHistogram

Computes the joint histogram of two images.

imColorHistogram

Plots 3D histogram of a color image.

imWeightedHistogram

Computes the weighted histogram of the values within the image, using an additional array/image as weights.

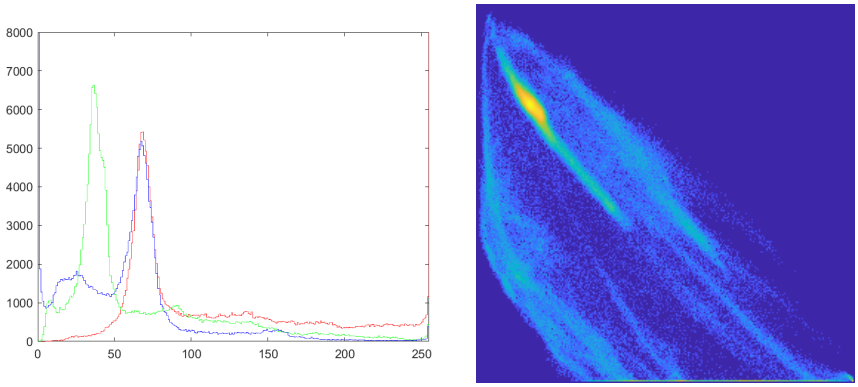


Figure 4.1: *Various histograms of a color image.*

4.1.2 Summary statistics on pixel values

These functions compute summary statistics of the gray values within an image. Some functions also accept color images as input, and compute the desired summary statistic along each channel.

imSum
Sum of a grayscale image, or sum of each color component.

imMean
Mean of a grayscale image, or mean of each color component.

imStd
Standard deviation of pixel values.

imVar
Variance of a grayscale image, or of each color component.

imMin
Minimum value of a grayscale image, or of each color component.

imMax
Maximum value of a grayscale image, or of each color component.

imMedian
Median value of a grayscale image, or of each color component.

imQuantile
Computes value that threshold a given proportion of pixels.

imMode
Mode of pixel values in an image.

imRegionFun
Applies a summary function to each region of a label image.

4.1.3 Entropy and mutual information

Entropy can be used for image segmentation (2.3). Mutual information is based on joint histogram (4.1.1).

imEntropy
Computes entropy of an image.

imJointEntropy
Joint entropy between two images.

imMutualInformation
Computes the mutual information between two images.

4.2 Analysis of regions

Region analysis usually refers to the quantification of features related to the **size** and the **shape** of regions identified within images.

The functions in this section usually accept as input either binary images (one result is returned), or label images (an array of results is returned).

4.2.1 Intrinsic volumes

The **intrinsic volumes** are a set of features with interesting mathematical properties that are commonly used for describing individual regions as well as binary microstructures. In 2D, they correspond to the **area**, the **perimeter** and the **Euler number**. For 3D regions, intrinsic volumes correspond to the **volume**, the **surface area**, the **mean breadth** and the **3D Euler number**.

The functions for computing intrinsic volumes are located in the imMinkowski module (chap 5). The functions used for describing regions are recalled here for convience. It is possible to specify options (connectivity for Euler Number, number of directions for perimeter or surface area), as well as image spatial calibration.

4.2.1.1 Examples of use

Example for a 2D image:

```
>> img = imread('circles.png');
>> [imArea(img) imPerimeter(img) imEuler2d(img)]
ans =
   1.0e+04 *
    1.4134    0.1046   -0.0003
```

Example for a 3D image:

```
>> % Surface area measured in 3D binary image (result in pixel^2)
>> img = analyze75read(analyze75info('brainMRI.hdr'));
>> bin = imclose(img>0, ones([5 5 3]));
>> S = imSurface(bin, [1 1 2.5])      % specify resolution
 ans =
     2.7291e+004
```

4.2.1.2 List of functions

imArea
Computes area of binary 2D image.

imPerimeter
Perimeter of a 2D image using Crofton formula.

imEuler2d
Euler number of a binary 2D image.

imVolume
Volume measure of a 3D binary structure.

imSurfaceArea
Surface area of a 3D binary structure.

imMeanBreadth
Mean breadth of a 3D binary or label image.

imEuler3d
Euler number of a binary 3D image.

4.2.2 Centroid and bounding shapes

The centroid and the bounding box are convenient tools to quickly describe the location of 2D or 3D regions.

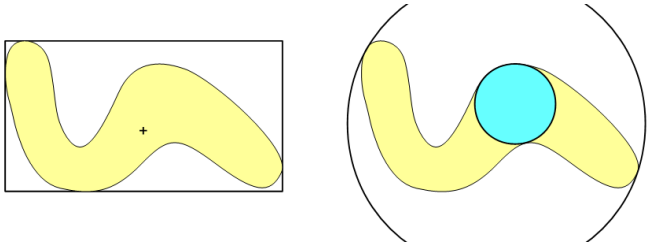


Figure 4.2: *Representative geometries obtained from a binary region. Left: Bounding box, centroid (black cross). Right: smallest enclosing circle (black) and largest inscribed circle (blue).*

imCentroid
Centroid of regions in a label image.

imBoundingBox
Computes the minimum and maximum of x, y and if necessary z coordinates of each region within a binary or label image (see Fig. 4.2).

imInscribedCircle
Maximal circle inscribed in a particle (see Fig. 4.2).

imInscribedBall
Maximal ball inscribed in a 3D particle (see Fig. 4.2).

imEnclosingCircle
Computes the smallest circle that completely encloses (see Fig. 4.2).

4.2.3 Geometric moments

An image moment is a certain particular weighted average of the intensities within image. Image moments are particularly useful to describe binary regions, as they can be related to their centroid, their size and their orientation.

4.2.3.1 Planar case

A binary region X may be described mathematically by its geometric moments m_{pq} of order (p,q), which correspond to an integral of its indicator function I_X, with various degrees along the directions:

$$m_{pq}(X) = \int\int I_X(x,y) x^p y^q \cdot dx \cdot dy \tag{4.1}$$

The moment of order $(0,0)$ simply corresponds to the area of X:

$$m_{00} = \int\int I_X(x,y) \cdot dx \cdot dy = \text{Area}(X) \tag{4.2}$$

The coordinates of the centroid of X can be expressed from the first-order moments:

$$x_c = \frac{m_{10}}{m_{00}} = \frac{1}{\text{Area}(X)} \int\int I_X(x,y) \cdot x \cdot dx \cdot dy$$
$$y_c = \frac{m_{01}}{m_{00}} = \frac{1}{\text{Area}(X)} \int\int I_X(x,y) \cdot y \cdot dx \cdot dy$$

It is often more convenient to work with **centered moments**, given by:

$$\mu_{pq} = \int\int I_X(x,y) (x - x_c)^p (y - yc)^q \cdot dx \cdot dy \tag{4.3}$$

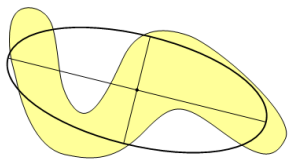


Figure 4.3: *Equivalent ellipse for a region.*

The orientation of the region can be retrieved from its (centered) second-order moments μ_{20}, μ_{11} and μ_{02}. The moments are often used to compute an equivalent ellipse with the same centroid and second order moments as the original region (Fig. 4.3).

4.2.3.2 3D moments

The mathematical definition of the geometric moments for 3D regions is similar to the 2D case:

$$m_{pqr}(X) = \int\int\int I_X(x,y,z)x^p y^q z^r \cdot dx \cdot dy \cdot dz \tag{4.4}$$

The first moment $m_{000}(X)$ corresponds to the volume of the particle. The normalization of the first-order moments by the volume leads to the **3D centroid** of the particle. The second-order moments can be used to compute an **equivalent ellipsoid**, defined as the ellipsoid with the same moments up to the second order as the region of interest.

4.2.3.3 Convention for 3D angles

In the MatImage library, 3D ellipsoids are represented by nine parameters: three for the coordinates of the centroid, three for size along each main direction, and three angles depicting the orientation. The three angles correspond to a succession of three rotations (see Figure 4.4):

1. a rotation $R_x(\psi)$ about the x-axis by ψ degrees (positive when the y-axis moves towards the z-axis)
2. a rotation $R_y(\theta)$ about the y-axis by θ degrees (positive when the z-axis moves towards the x-axis)
3. a rotation $R_z(\varphi)$ about the z-axis by φ degrees (positive when the x-axis moves towards the y-axis)

The global rotation matrix is assimilated to the product $R_z(\varphi) \cdot R_y(\theta) \cdot R_x(\psi)$.

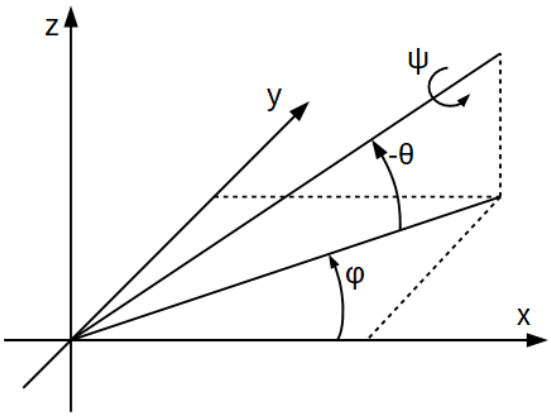


Figure 4.4: *Definition of angles for representing the orientation of equivalent ellipsoid.*

4.2.3.4 Associated functions

imEquivalentEllipse
Equivalent ellipse of a binary or label image.

imEquivalentEllipsoid
Equivalent ellipsoid of a 3D binary image.

imMoment
Computes simple moment(s) of an image.

imCMoment
Computes centered moment of an image.

imCSMoment
Computes centered and scaled moment of an image.

imHuInvariants
Hu's invariants are a common method used to summarize moments computed on a region (Hu, 1962). They can be used for indexing region based on morphology.

4.2.4 Feret diameter and oriented box

A popular way to assess the size of a region is to measure its **largest Feret diameter**. The maximum Feret diameter of a region is simply the maximum distance computed over all the pairs of points belonging to the region:

$$F_{\max} = \max_{\boldsymbol{x},\boldsymbol{y}\in X} d(\boldsymbol{x},\boldsymbol{y}) \tag{4.5}$$

It is easy to realize that the search can be performed on the set of boundary points (Fig. 4.5). In practice, the computation of maximum Feret diameter can be accelerated by first computing the convex hull of the region.

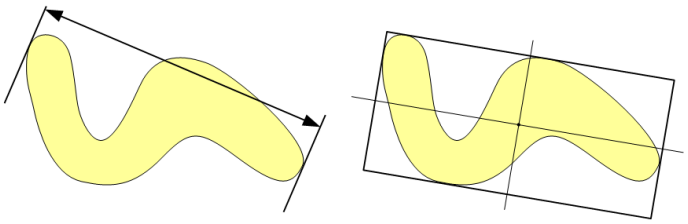


Figure 4.5: *Examples of measurements on a planar region: maximum Feret diameter and oriented box.*

Associated functions

imOrientedBox
Minimum-area oriented bounding box of particles in image (Fig. 4.5).

imMaxFeretDiameter
Maximum Feret diameter of a binary or label image (Fig. 4.5).

imFeretDiameter
Feret diameter of a particle(s) for a given direction(s).

4.2.5 Quantification by shape indices

It is often convenient to describe and **quantify the shape** of regions independently of their location, orientation, or relative scaling. Several indices are commonly used to describe the shape of the particles, independently of their size.

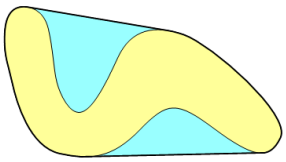


Figure 4.6: *Convexity of a planar region. The convex area is the union of the blue and yellow regions.*

imConvexity
Convexity of particles in label image (Fig. 4.6).

4.3 Extraction of geometric primitives

This sections gathers additional functions for converting binary or label images to geometric primitives.

imFind

Returns the coordinates of non-zero pixels in an image.

imRAG

Region adjacency graph of a labeled image. The result is provided as a graph, with vertices corresponding to region centroids, and edges corresponding to neighbor regions (Fig. 4.7)

imContours

Extracts polygonal contours of a binary image. The result is provided as a cell array, with one polygon per cell.

imBinaryToGraph

Transforms a binary image into a graph structure. The result is provided as a graph, with vertices corresponding to pixels, and edges corresponding to neighbor pixels.

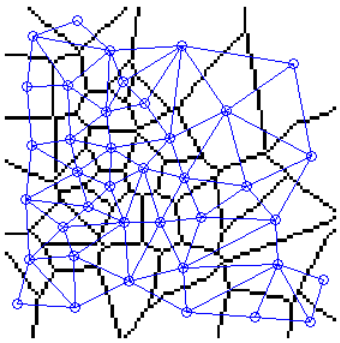


Figure 4.7: *Computation of the Region Adjacency Graph on a label image. Plotting the graph network requires the MatGeom toolbox.*

5 Module imMinkowski

Contains various functions for measuring or estimating geometric quantities from 2D or 3D images.

For 2D images, parameters are the area, the perimeter and the (2D) Euler Number. For 3D images, parameters are the volume, the surface area, the mean breadth (also known as integral of mean curvature), and the (3D) Euler Number. For the sake of completeness, parameters for 1D images are also included: length and number (1D Euler Number).

Contents

5.1 Introduction

The methods for computing the parameters are described in Lang et al. (2001); Legland et al. (2007); Ohser & Schladitz (2009). Implementation notes and examples are given in Lehmann & Legland (2012).

For each quantity, several functions are provided:

im<Quantity> evaluates the quantity from a binary or label image. If the structure touches the border of the image, it is considered as a structure border. Such functions should fit most needs.

im<Quantity>Density evaluates the quantity within the image, by considering that the image is a representative window of a larger structure. The intersection of the structure with image border is not taken into account for measurements.

im<Quantity>Estimate same as im<Quantity>Estimate, but the result is normalised by the area or the volume of the observed window.

im<Quantity>Lut returns a look-up-table of values that can be used to estimate the parameter from the histogram of binary configuration in original image, as computed by the function "imBinaryConfigHisto" (page 5.5).

Most functions work both for binary and label images. It is possible to specify options (connectivity for Euler Number, number of directions for perimeter or surface area), as well as image resolution in each direction. Examples of use:

```
>> % compute perimeter of several coins
>> lbl = bwlabel(imread('coins.png') > 100);
>> imPerimeter(lbl)
 ans =
   184.8668
   154.9495
   185.1921
   267.1690
   187.3183
   179.5038
   182.7406
   180.8445
   155.5049
   155.5049

>> % Surface area measured in 3D binary image (result in pixel^2)
>> img = analyze75read(analyze75info('brainMRI.hdr'));
>> bin = imclose(img>0, ones([5 5 3]));
>> S = imSurfaceArea(bin, [1 1 2.5])      % specify resolution
 ans =
     2.7291e+004
```

5.2 Analysis of 2D images

For 2D images, parameters are the area, the perimeter and the (2D) Euler Number. The **area** corresponds to the number of pixels belonging to the structure, multiplied by the spatial calibration if it is given. The **perimeter** corresponds to an integral over the boundary of the structure. The (2D) **Euler number** corresponds to the number of connected components minus the number of holes within the structure.

5.2.1 Definitions

The **area** and the **perimeter** of a set X with smooth surface ∂X can be defined using integrals over the set, or over its boundary:

$$A(X) = \int_X dx, P(X) = \int_{\partial X} dx \tag{5.1}$$

In image analysis, the **measurement of area** of 2D particles simply consists in counting the number of pixels that constitute it, weighted by the area of an individual pixel.

The **Euler number** is a feature that describes the **topology** of a region. It corresponds to the number of connected components, minus the number of holes (Fig. 5.1-a).

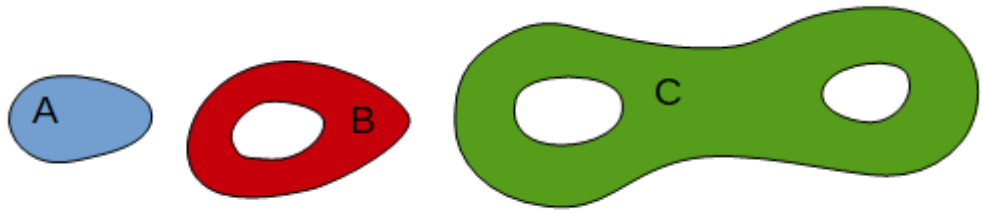


Figure 5.1: *Euler Number for 2D regions. Particle A is composed of a single component, its Euler number is 1. Particles B and C present one and two holes respectively. Their corresponding Euler numbers are equal to* $0 = 1 - 1$ *and* $-1 = 1 - 2$.

In 2D, the Euler number of a region with smooth boundary also equals the **integral of the curvature** over the boundary of the set:

$$\chi(X) = \frac{1}{2\pi} \int_{\partial X} \kappa(x) dx \tag{5.2}$$

5.2.2 Analysis of regions

The functions described here allow to quantify the morphology of either a single region given as a binary image (a logical array), or of several regions given as a label image (an array of integer values corresponding to the region labels).

imArea
Computes area of binary 2D image.

imPerimeter
Perimeter of a 2D image using Crofton formula.

perimeter
Estimates perimeter of a structure.

imEuler2d
Euler number of a binary 2D image.

5.2.3 Analysis of microstructures

imAreaDensity
Computes area density in a 2D image.

imAreaEstimate
Estimates area of binary 2D structure with edge correction.

imPerimeterDensity
Perimeter density of a 2D binary structure, using Crofton formula.

imPerimeterEstimate
Perimeter estimate of a 2D binary structure.

imEuler2dDensity
Euler density in a 2D image.

imEuler2dEstimate
Estimates Euler number in a 2D image.

5.2.4 Utility functions (2D)

imPerimeterLut
Look-Up Table for measuring perimeter in a binary image.

5.3 Analysis of 3D Images

For 3D regions, intrinsic volumes correspond to the **volume**, the **surface area** (sometimes simply called surface) the **mean breadth** (also known as the integral of mean curvature) and the **(3D) Euler number**. While the volume and the surface area are rather common, the latter two are less intuitive. Both the mean breadth and the 3D Euler number can be related to the curvatures that can be measured on smooth surfaces.

5.3.1 Definitions

The **volume** and the **surface area** of a set X with smooth surface ∂X can be defined using integrals over the set, or over its boundary:

$$V(X) = \int_X dx \tag{5.3}$$

$$S(X) = \int_{\partial X} dx \tag{5.4}$$

The **mean breadth** $\bar{b}$ of a convex set can be seen as the average of the caliper diameter over all directions. For a set X with smooth boundary ∂X, the mean breadth is proportionnal to the **integral of the mean curvature** (**?**Ohser & Schladitz, 2009):

$$\bar{b}(X) = \frac{1}{2\pi} \int_{\partial X} \frac{\kappa_1(x) + \kappa_2(x)}{2} dx \tag{5.5}$$

As in 2D, the **3D Euler number** also quantifies the topology of a set. It corresponds to the number of connected components, minus the number of "handles" or "tunnels" through the structure, plus the number of bubbles within the particles(**?**Ohser & Schladitz, 2009), see Figure 5.2.

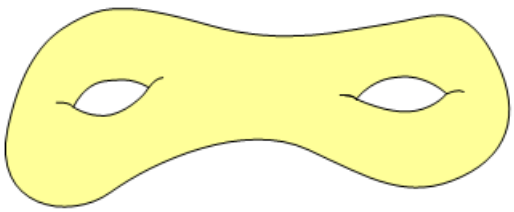


Figure 5.2: *Euler Number of a 3D particle. The Euler number equals -1, corresponding to the subtraction of 1 connected components minus two handles.*

For a set X with smooth boundary ∂X, the 3D Euler number is proportionnal to the integral of the **gaussian curvature**, corresponding to the product of the curvatures:

$$\chi(X) = \frac{1}{4\pi} \int_{\partial X} \kappa_1(x) \cdot \kappa_2(x) dx \tag{5.6}$$

5.3.2 Analysis of regions

The functions described here allow to quantify the morphology of either a single region given as a binary image (a logical array), or of several regions given as a label image (an array of integer values corresponding to the region labels).

imVolume
Volume measure of a 3D binary structure.

imSurfaceArea
Surface area of a 3D binary structure.

imJointSurfaceArea
Surface area of the interface between two regions.

imMeanBreadth
Mean breadth of a 3D binary or label image.

imEuler3d
Euler number of a binary 3D image.

5.3.3 Analysis of microstrucures

Functions described here allow to quantify the morphology of binary microstructures, not necessarily defined as a collection of disjoint particles. Examples are porous media such as soil, sponges, bread... The quantification relies on intrinsic volumes normalized by the size of the observation window.

imVolumeDensity
Computes volume density of a 3D image.

imVolumeEstimate
Estimates volume of a 3D binary structure with edge correction.

imSurfaceAreaDensity
Surface area density of a 3D binary structure.

imSurfaceAreaEstimate
Estimates surface area of a binary 3D structure.

specificSurface
Implementation of Ohser's algorithm for surface area computation.

imMeanBreadthDensity
Mean breadth density of a 3D binary structure.

imMeanBreadthEstimate
Estimates mean breadth of a binary structure.

imEuler3dDensity
Computes Euler density in a 3D image.

imEuler3dEstimate
Estimates Euler number in a 3D image.

5.3.4 Utility functions (3D)

imSurfaceAreaLut
Look-Up Table for computing surface area of a 3D binary image.

imMeanBreadthLut
Look-Up Table for computing mean breadth of a 3D image.

specificIntMeanCurv
Ohser's Integral of Mean Curvature, from Ohser & Mücklich (2000).

specificIntMeanCurvDetails
Ohser's Integral of Mean Curvature, with details.

5.4 Analysis of 1D images

For the sake of completeness, parameters for 1D images are also included: the length and the number (1D Euler Number).

5.4.1 Analysis of regions

imLength
Computes the total length of a binary 1D structure.

imEuler1d
Computes Euler number of a binary 1D image.

5.4.2 Analysis of densities

imLengthDensity
Estimates length density of a binary 1D structure using edge correction.

imLengthEstimate
Estimates total length of a binary 1D structure using edge correction.

imEuler1dEstimate
Computes Euler number of a binary 1D image.

5.5 Utility functions

imBinaryConfigHisto
Computes the histogram of the binary configurations within a 2D or 3D image. Other utilities for configuration maps are described in Section 2.6.3.

imProjectedDiameter
Projected diameter in a given direction.

imProjectedArea
Total projected area in a given direction.

epc
Computes Euler-Poincare Characteristic (EPC) of a structure.

tpl
Computes total projection length.

6 Module imGranulometry

Gray-level granulometry is an image texture analysis approach based on mathematical morphology (Soille, 2003). It consists in applying morphological filters (typically opening or closing) using structuring elements of increasing size (Jean-Louis-Chermant & Coster, 1991; Soille, 2003).

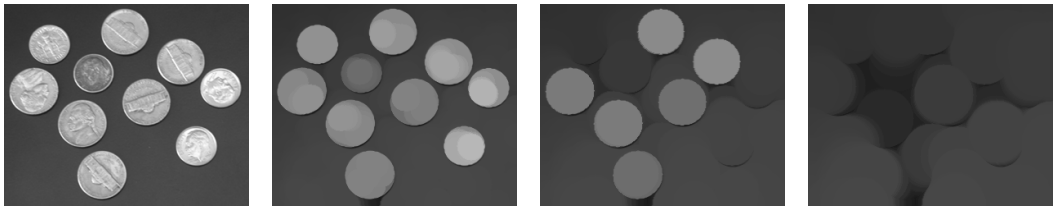
Gray-level granulometry results in **granulometry curves** that can be interpreted in terms of size distribution, making it easier to relate to the physical properties of the studied structures. Another advantage of morphological granulometries is that it is possible to focus on **either bright or dark structures** in the image, or to consider both of them.

Contents

6.1 Principle of gray-level granulometry

6.1.1 Morphological sieving

The principle of gray-level granulometry is to apply morphological filters using structuring elements ("strel") of increasing size. By applying morphological openings of increasing size, bright structures of increasing sizes are removed. A **digital sieving** is therefore obtained. Similarily, applying morphological closings of increasing sizes makes dark structures progressively disappear.



(a) Original image. **(b)** Opening R=15. **(c)** Opening R=27. **(d)** Opening R=35.

Figure 6.1: *Exemple of gray-level granulometry by opening. (a) Original gray-level image of rice grains. (b) Sum of gray levels for differents sizes of square structuring element. (c) Granulometry curve, corresponding to size distribution.*

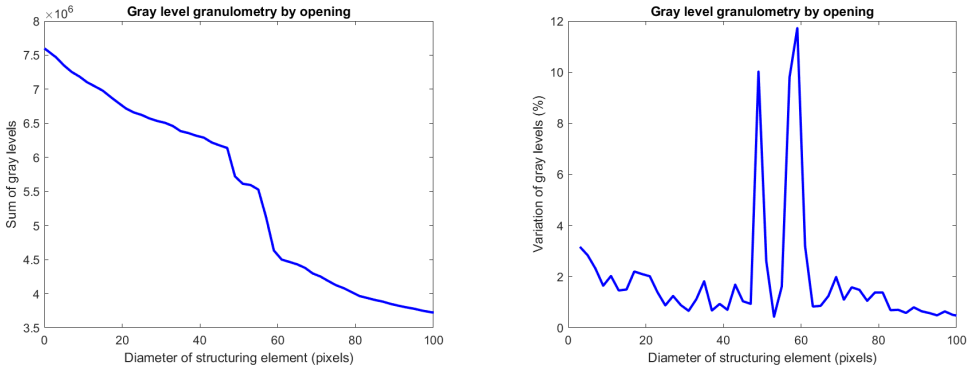
An example is provided on Figure 6.1 for a grayscale image of coins. For small values of structuring element radius, only small details vanish. When the value is equal to around 27, the morphological opening removes small coins (with diameter around 50 pixels), but retains large coins (with diameter around 60 pixels). For large values of radius, all the coins disappear.

6.1.2 Computation of granulometry curves

The difference between two successive opening or closing steps is obtained by computing the sum of gray levels within each image, called the volume of the image. Using openings [resp. closings], the volume curve decreases [resp. increases] monotonously, and reaches a plateau. The derivative of this curve, normalised by the initial and final values, can be interpreted as a **size distribution** of the bright [resp. dark] structures within the image, taking into account the gray levels. More formally, if V_i is the sum of the gray levels of the image at the i-th iteration, the granulometry curve is given by:

$$g_i = \frac{V_{i+1} - V_i}{V_{\text{final}} - V_{\text{initial}}}$$

An example is provided on Figure. 6.2, using the coins image as original image, and using granulometry by opening with disk structuring elements. Two peaks an be noticed on the granulometry curve: they correspond to the size of the small and large coins. Then, the two population of coins can be determined from the granulometry curve. In practice, the



(a) Sum of gray levels. **(b)** Granulometry curve (pattern spectrum).

Figure 6.2: *Exemple of gray-level granulometry by opening. (b) Sum of gray levels for differents sizes of square structuring element. (c) Granulometry curve, corresponding to size distribution.*

granulometry curve presents larger peaks, and the populations are difficult to distinguish. However, the typical size of the structures may be assessed without requiring segmentation.

The granulometry curve on sample image can be computed with the following script:

```
% read image
img = imread('coins.png');
% compute granulometry curve by opening using square strel
xi = 1:50;
[gr, diams, vols] = imGranulo(img, 'opening', 'disk', xi);
% display volume curve
figure; plot([0 diams], vols); xlim([0 100]);
xlabel('Diameter of structuring element (pixels)');
ylabel('Sum of gray levels');
title('Variation of volume curve');
% display granulo
figure; plot(diams, gr); xlim([0 100]);
xlabel('Diameter of structuring element (pixels)');
ylabel('Variation of gray levels (%)');
title('Gray level granulometry by opening');
```

More details can be found in Devaux et al. (2005, 2008); Devaux & Legland (2014).

6.2 Summary statistics

It may be useful to compute **summary statistics** from granulometry curves, to facilitate their comparison. The simplest one is the arithmetic average, that can be obtained from:

$$m = \sum_i g_i \cdot x_i \tag{6.1}$$

where g_i is the value of the granulometric curve (between 0 and 1), and x_i is the size of the corresponding step, expressed either in pixels, or in physical unit. The standard deviation can also be obtained:

$$s = \left[\sum_i g_i \cdot (x_i - m)^2 \right]^{1/2} \tag{6.2}$$

Granulometry curves often present a log-normal shape, making the arithmetic average sometimes different from the mode of the distribution. An alternative is to compute the average size by using the **geometrical mean**:

$$m_G = \exp\left(\sum_i g_i \cdot \log x_i \right) \tag{6.3}$$

6.3 Granulometry mapping

Instead of considering the whole image for computing the granulometry curves, it is possible to restrict the process to a specific region of interest. The region of interest is typically defined by a binary mask with the same size as the grayscale image.

By applying the process on several region of interest, and computing a summary statistic for each region, it is possible to build a synthetic result image where the value is given by the summary value of the regions. Such a method was used to build parametric mapping of cellular morphology from images of plant tissues (Legland et al., 2020).

6.4 Oriented granulometry

Granulometry curves strongly depends on the shape of the structuring elements. Using disks, the size can be related to the thickness of the structures, as the disks can erode from any direction.

When using **linear structuring elements**, it is possible to consider also thin elongated structures. The principle can be either to consider horizontal structuring elements, and applying rotations of the image (Gallos et al., 2017), or to consider a family of orientated structuring elements. The latter strategy was used in (Gager et al., 2020; Melelli et al., 2020).

6.5 List of functions

6.5.1 Computation of granulometry curves

imGranulo

Computes granulometry curve for the whole image. The syntax is as follow:

```
GR = imGranulo(IMG, TYPE, SHAPE, SIZES);[GR, DIAMS, VOLS] = imGranulo(IMG, TYPE, SHAPE,
    SIZES);
```

imGranuloByRegion

Computes granulometry curve for each region of a label image. The result is an array with as many rows as the number of regions. The regions can also be specified by a cell array containing pixel indices for each regions. The latter possibility makes it possible to specify overlapping regions.

6.5.2 Summary statistics

granuloMeanSize

Computes the geometric mean of a granulometric curve, or of a series of granulometric curves.

granuloMean

Computes the arithmetic mean of granulometric curve(s).

granuloStd

Computes the standard deviation of granulometric curve(s).

6.5.3 Oriented granulometry

imDirectionalGranulo

Directional granulometries for several orientations.

imGranuloOrientationMap

Orientation map of directional granulometry.

orientedLineStrel

Creates an oriented line structuring element.

imOrientedGranulo

Computes gray level granulometry mean size for various orientations, by rotating the image before computing granulometry with a line structuring element. More details can be found in Gallos et al. (2017).

7 Module imGeodesics

The functions in this module are devoted to the computation of geodesic distances within images. The geodesic distances can be ued for the extraction of specific features such as geodesic diameters, geodesic centers of geodesic extremities.

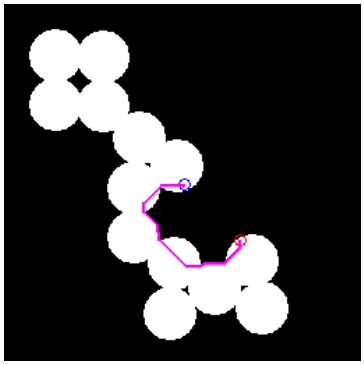
More information about the concept of geodesic distances and geodesic propagation can be found in Lantuéjoul & Beucher (1981), and in the book of Soille (2003). These functions were developed in the context of the study presented in Legland & Beaugrand (2013).

7.1 Distance propagation

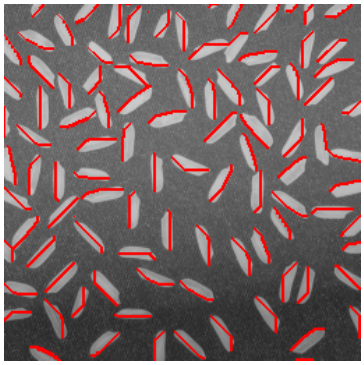
The functions in this sections compute geodesic distances, from a marker image constrained to a mask images. The computation of geodesic distance maps is based on chamfer distances, that are an approximation of the Euclidean distance (see also Section 2.5.2).

From the geodesic distance maps, it is possible to extract geodesic paths and/or maximal length geodesic path.

(a) Geodesic Distance Map.



(b) Geodesic Path.



(c) Geodesic Diameters.

Figure 7.1: *Geodesic paths and diameters. (a) Computation of the geodesic distance map from a marker located within the binary region. (b) Computation of a geodesic path between two points with a binary region. The geodesic path correspond to the shortest path with the chosen metric. (c) Geodesic diameter computed for each binarised grains, and superimposed on the original image.*

imGeodesicDistanceMap

Geodesic distance transform for binary or label images.

imGeodesicDistanceMap3d
Geodesic distance transform for 3D binary or label images.

imGeodesicPath
Compute a geodesic path between two markers in an image.

imMaxGeodesicPath
Find a path in a region with maximal geodesic length.

imGeodesicDistance
Compute geodesic distance between 2 markers.

7.2 Geodesic parameters

Based on the geodesic distance distance map, it is possible to define several features describing the morphology of the particles.

imGeodesicDiameter
Computes geodesic diameter of particles.

imGeodesicDiameter3d
Computes geodesic diameter of 3D particles.

imGeodesicCenter
Computes geodesic center of a binary particle.

imGeodesicExtremities
Computes geodesic extremities of a binary particle.

imGeodesicRadius
Computes the geodesic radius of a binary particle.

imGeodesicPropagation
Computes geodesic propagation for each foreground pixel.

7.3 Validation

chamferDistanceError
Computes relative error of chamfer distance with Euclidean distance.

8 Module imShapes

The functions from this module aim at generating 2D or 3D phantom images of geometric primitives such as rectangles, ellipsoids, cylinders... Generated images can be used for validation of image quantification algorithms (such as in chapters 4 or 5).

Most functions from this module requires the MatGeom library[1].

8.1 2D images

Typical script for creating images is as follow:

```
% generate coordinate system for square images
lx = 1:100;
ly = 1:100;
% define ellipse by center, two radius and one orientation
ellipse = [50 50  40 20  theta];
% generation of binary image
img = discreteEllipse(lx, ly, ellipse);
% display
figure; imshow(img);
```

Some examples are given in Fig. 8.1 and 8.2.

8.1.1 Planar domains

discreteDisc
Discretize a disc, defined by its center and radius (Fig. 8.1).

discreteEllipse
Discretize a planar ellipse, defiend by its center, two radius, and an orientation (Fig. 8.1).

discreteSquare
Discretize a planar square (Fig. 8.1).

discreteRectangle
Discretize a planar rectangle (Fig. 8.1).

discreteCapsule
Discretize a planar capsule (Fig. 8.2).

discretePolygon
Discretize a planar polygon, defined by a series of vertex coordinates.

[1] https://github.com/mattools/matGeom

Figure 8.1: *Examples of simple discrete shapes in 2D.*

discreteHalfPlane
Discretize a half plane.

discreteParabola
Discretize a planar parabola.

discreteEgg
Discretize a planar egg (Fig. 8.2).

discreteTrefoil
Discretize a trefoil curve (Fig. 8.2).

discreteStarfish
Discretize a starfish curve (Fig. 8.2).

Figure 8.2: *More examples of discrete shapes in 2D.*

8.1.2 Curves and points

discretePoints
Discretize a set of points, defined by a series of coordinates.

discretePolyline
Discretize a planar polyline, defined by a series of vertex coordinates.

discreteCurve
Discretize a planar curve, defined by a series of vertex coordinates.

8.2 3D images

Typical script for creating 3D binary shapes is as follow:

```
elli = [50 50 50  30 20 10  40 30 20];
img = discreteEllipsoid(1:100, 1:100, 1:100, elli);
figure;
isosurface(img, .5);
hold on; axis square; view(3); light;
```

Some examples are given in Fig. 8.3 and 8.4.

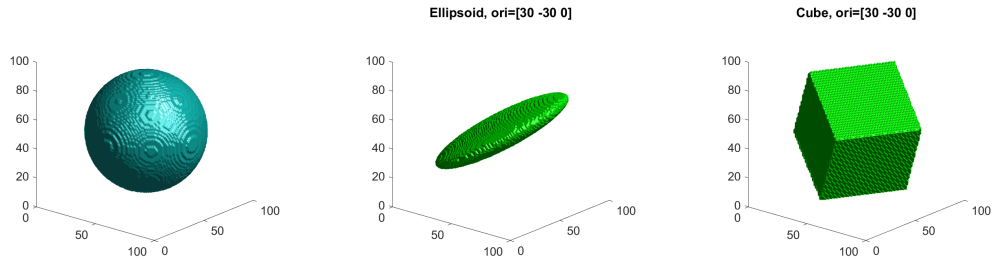


Figure 8.3: *Examples of simple discrete shapes in 3D*

discreteBall
Discretize a 3D Ball, defined by a center and a radius.

discreteHalfBall
discretize a 3D half-ball, defined by a center, a radius and a normal orientation.

discreteEllipsoid
discretize a 3D ellipsoid.

discreteCube
discretize a 3D cube.

discreteCuboid
discretize a 3D cuboid.

discreteTorus
Discretize a 3D Torus.

discreteCylinder
Discretize a 3D cylinder, defined by two extreimty points and a radius.

discreteCapsule3d
Create a binary image of a 3D capsule.

discreteReuleauxRevol
Discretize the revolution of a Reuleaux triangle.

discreteSphereEighth
Discretize a 3D sphere eighth.

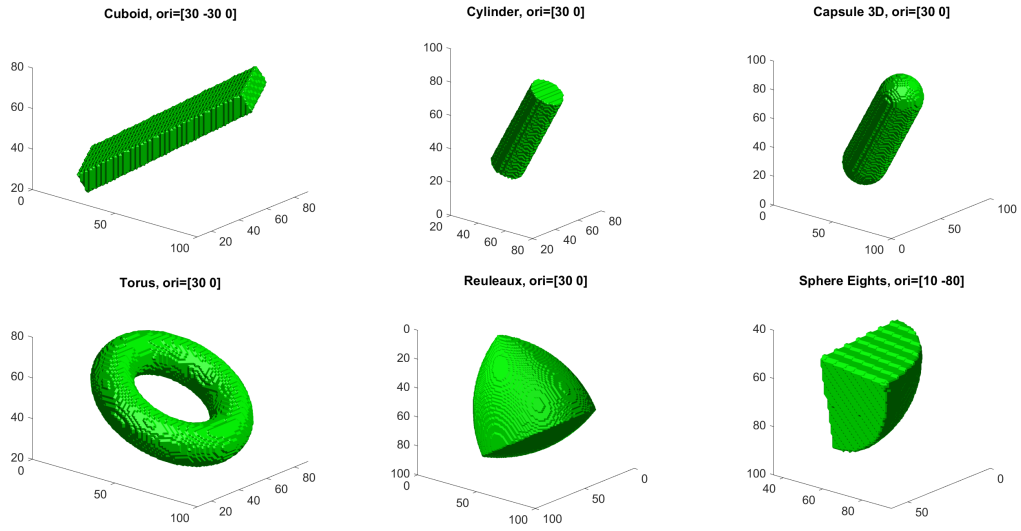


Figure 8.4: *More examples of discrete shapes in 3D.*

8.3 Tessellations

The functions in this section allows to quickly generate images of cellular patterns. Outputs are usually label images.

imPointsInfluenceZones
Maps influence zones of a set of 2D/3D points.

imvoronoi2d
Generate a 2D voronoi image from a set of points.

imvoronoi3d
generate a 3D voronoi image from a set of points.

dilatedVoronoi
Simulate a 'thick' voronoi tesselation.

imAWVoronoi
generate Additively Weighted Voronoi Diagram image.

imPowerDiagram
Power diagram of a set of points.

Bibliography

Andrey, P. & Maurin, Y. (2005). Free-d: an integrated environment for three-dimensional reconstruction from serial sections. *Journal of Neuroscience Methods*, 145(1–2), 233–244.

Borgefors, G. (1984). Distance transformation in arbitrary dimensions. *Computer Vision, Graphics, and Image Processing*, 27, 321–145.

Borgefors, G. (1986). Distance transformations in digital images. *Computer Vision, Graphics, and Image Processing*, 34, 344–371.

Devaux, M.-F., Barakat, A., Robert, P., Bouchet, B., Guillon, F., Navez, B., & Lahaye, M. (2005). Mechanical breakdown and cell wall structure of mealy tomato pericarp tissue. *Postharvest Biology and Technology*, 37(3), 209–221.

Devaux, M.-F., Bouchet, B., Legland, D., Guillon, F., & Lahaye, M. (2008). Macro-vision and grey level granulometry for quantification of tomato pericarp structure. *Postharvest Biology and Technology*, 47(2), 199–209.

Devaux, M.-F. & Legland, D. (2014). *Microscopy: advances in scientific research and education*, chapter Grey level granulometry for histological image analysis of plant tissues, (pp. 681–688). Formatex Research Center.

Frangi, A. F., Niessen, W. J., Vinc, K. L., & Viergever, M. A. (1998). Multiscale Vessel Enhancement Filtering. In *Medical Image Computing and Computer-Assisted Interventation*, Lecture Notes in Computer Science (pp. 130).

Gager, V., Legland, D., Bourmaud, A., Le Duigou, A., Pierre, F., Behlouli, K., & Baley, C. (2020). Oriented granulometry to quantify fibre orientation distributions in synthetic and plant fibre composite preforms. *Industrial Crops and Products*, 152, 112548.

Gallos, A., Paës, G., Legland, D., Allais, F., & Beaugrand, J. (2017). Exploring the microstructure of natural fibre composites by confocal Raman imaging and image analysis. *Composites Part A: Applied Science and Manufacturing*, 94, 32 – 40.

Hu, M. K. (1962). Visual pattern recognition by moment invariants. *IRE Trans. Inf. Theory*, 12, 179–187.

Jean-Louis-Chermant & Coster, M. (1991). Granulometry and granulomorphy by image analysis. *Acta Stereologica*, 10(1), 7–23.

Lang, C., Ohser, J., & Hilfer, R. (2001). On the analysis of spatial binary images. *Journal of Microscopy*, 203(3), 303–313.

Lantuéjoul, C. & Beucher, S. (1981). On the use of geodesic metric in image analysis. *Journal of Microscopy*, 121(1), 39–49.

Legland, D. & Beaugrand, J. (2013). Automated clustering of lignocellulosic fibres based on morphometric features and using clustering of variables. *Industrial Crops and Products*, 45(0), 253 – 261.

Legland, D., Guillon, F., & Devaux, M.-F. (2020). Parametric mapping of cellular morphology in plant tissue sections by gray level granulometry. *Plant Methods*, 16(63).

Legland, D., Kiêu, K., & Devaux, M.-F. (2007). Computation of Minkowski measures on 2D and 3D binary images. *Image Analysis and Stereology*, 26(6), 83–92.

Lehmann, G. & Legland, D. (2012). *Efficient N-Dimensional surface estimation using Crofton formula and run-length encoding*. Technical report.

Melelli, A., Jamme, F., Legland, D., Beaugrand, J., & Bourmaud, A. (2020). Microfibril angle of elementary flax fibres investigated with polarised second harmonic generation microscopy. *Industrial Crops and Products*, 156, 112847.

Ohser, J. & Mücklich, F. (2000). *Statistical Analysis of Microstructures in Materials Sciences*. Chichester: J. Wiley & Sons.

Ohser, J. & Schladitz, K. (2009). *3D Images of Materials Structures*. WILEY-VCH Verlag GmbH & Co. KGaA, Weinheim.

Soille, P. (2003). *Morphological Image Analysis*. Springer, 2nd edition.