

---

# **LaueTools Documentation**

**JS Micha, O. Robach. S. Tardif**

**Apr 02, 2020**



---

## Contents

---

<b>1</b>	<b>Introduction</b>	<b>3</b>
1.1	Tools . . . . .	3
1.2	Developers . . . . .	3
1.3	Browse Modules and Functions . . . . .	3
<b>2</b>	<b>Installation</b>	<b>5</b>
2.1	How to Get LaueTools code . . . . .	5
2.2	Build LaueTools Documentation . . . . .	5
<b>3</b>	<b>Getting started</b>	<b>7</b>
3.1	Launch Graphical User Interfaces of LaueTools . . . . .	7
<b>4</b>	<b>Conventions</b>	<b>11</b>
4.1	Mathematics and Conventions . . . . .	11
<b>5</b>	<b>Graphical User Interfaces</b>	<b>13</b>
5.1	Interactive Graphical User Interfaces . . . . .	13
5.2	Batch Processing Graphical User Interfaces . . . . .	24
<b>6</b>	<b>Tutorials</b>	<b>25</b>
6.1	Basics of Laue Pattern peak search and Unit cell Refinement . . . . .	25
6.2	Indexation of spots data set. . . . .	43
<b>7</b>	<b>LaueTools Modules</b>	<b>61</b>
7.1	Browse Modules and Functions . . . . .	61
7.2	Modules for Laue Pattern Simulation . . . . .	61
7.3	Modules for Digital Image processing, Peak Search & Fitting . . . . .	85
7.4	Modules for Laue Pattern Indexation . . . . .	96
7.5	Modules for Crystal unit cell refinement . . . . .	96
7.6	Modules for batch processing . . . . .	96
<b>8</b>	<b>Indices and tables</b>	<b>97</b>
	<b>Python Module Index</b>	<b>99</b>
	<b>Index</b>	<b>101</b>



You have data to analyse? You are at the right place! You can also contact us for more assistance.



# CHAPTER 1

---

## Introduction

---

LaueTools is a **python** written package aiming at analysing from 1 to few 10000s microdiffraction Laue patterns coming from synchrotron CRG-IF BM32 beamline at ESRF.

With pip installer, it is now relatively easy to install LaueTools and put hands immediately on data. See [Installation](#) page.

### 1.1 Tools

LaueTools has got :

- Modules to be imported in your own scripts like any **scientific library**.
- LaueTools has got several *Graphical User Interfaces* to interact graphically with data so that to accelerate the design of future scripts and to set the most finely of parameters for batch processing.
- **Notebooks** to guide the typical way you can handle the data (visualisation, selection, analysis, post processing)

### 1.2 Developers

Developers are welcome to improve the code readability and to add new functionalities. Please contact us.

### 1.3 Browse Modules and Functions

- `genindex`
- `modindex`





Some dependencies are rather usual (numpy, scipy, matplotlib) while others are more uncommon but very useful (fabio, networkx).

GUIs are based on wxpython graphical libraries which can be tricky to install (Sorry. We are working on it).

## 2.1 How to Get LaueTools code

- Download the very last version of the code at **gitlab.esrf.fr** (but you are also welcome to fork this project):

<https://gitlab.esrf.fr/micha/lauetools>

- or Download last (or older releases) on **pypi** by means of pip

<https://pypi.org/project/LaueTools/>

if pip is installed:

```
pip install lauetools
```

## 2.2 Build LaueTools Documentation

Documentation can be generated, by installing sphinx and a cool html theme:

```
pip install sphinx
```

```
pip install sphinx-rtd-theme
```

You may need rinohtype:

```
pip install Rinohtype
```

Then from `/LaueTools/Documentation` folder which contains *Makefile* and 2 folders *build* and *source*, build the documentation

```
make html
```

Files in html format will be browsed in `/build/html` folder with any web navigator. You can start with `index.html`.

### 3.1 Launch Graphical User Interfaces of LaueTools

- start Lauetools GUIs from command line :

To deal with relative import, the package name 'LaueTools' must be specified to the python interpreter as following

Examples:

- `python -m LaueTools.LaueToolsGUI`
- `python -m LaueTools.GUI.LaueSimulatorGUI`
- `python -m LaueTools.GUI.PeaksearchGUI`

The first main GUI, LaueToolsGUI can open also the two last GUIs (LaueSimulatorGUI, PeaksearchGUI)

There are additional basic GUIs for batch processing located in FileSeries folder:

- `python -m LaueTools.FileSeries.Peak_Search`
- `python -m LaueTools.FileSeries.Index_Refine`
- `python -m LaueTools.FileSeries.Build_summary`
- `python -m LaueTools.FileSeries.Plot_Maps2`
- within interactive python (say, `ipython -i`), GUI can be started thanks to a `start()` function:
  - In [1] : `import LaueTools.LaueToolsGUI as LTGUI`
  - In [2] : `LTGUI.start()`

---

**Note:** in the LaueTools folder :

- neither `> python LaueToolsGUI`
  - nor in `>ipython -i : > run LaueToolsGUI` will work...
-

### 3.1.1 Use LaueTools module as a library

With pip installation, LaueTools package will be included to python packages. Therefore any module will be callable as the following:

```
-In [1] : import LaueTools.readmccd as rmccd
```

```
-In [2] : rmccd.readCCDimage('myimage.tif')
```

In jupyter-notebook, it is also simple in the same manner:





## CHAPTER 4

---

### Conventions

---

#### 4.1 Mathematics and Conventions





---

## Graphical User Interfaces

---

LaueTools provides two types of GUI:

- GUI with graphical interaction when handling data
- GUI used as input file parameters for batch processing

### 5.1 Interactive Graphical User Interfaces

The main steps of analysis are Laue peaks search, Laue Pattern indexation and unit Cell Refinement. Detector geometry calibration (DetectorCalibrationBoard) and Laue Pattern of Polycrystals (LaueSimulatorGUI) are also available.

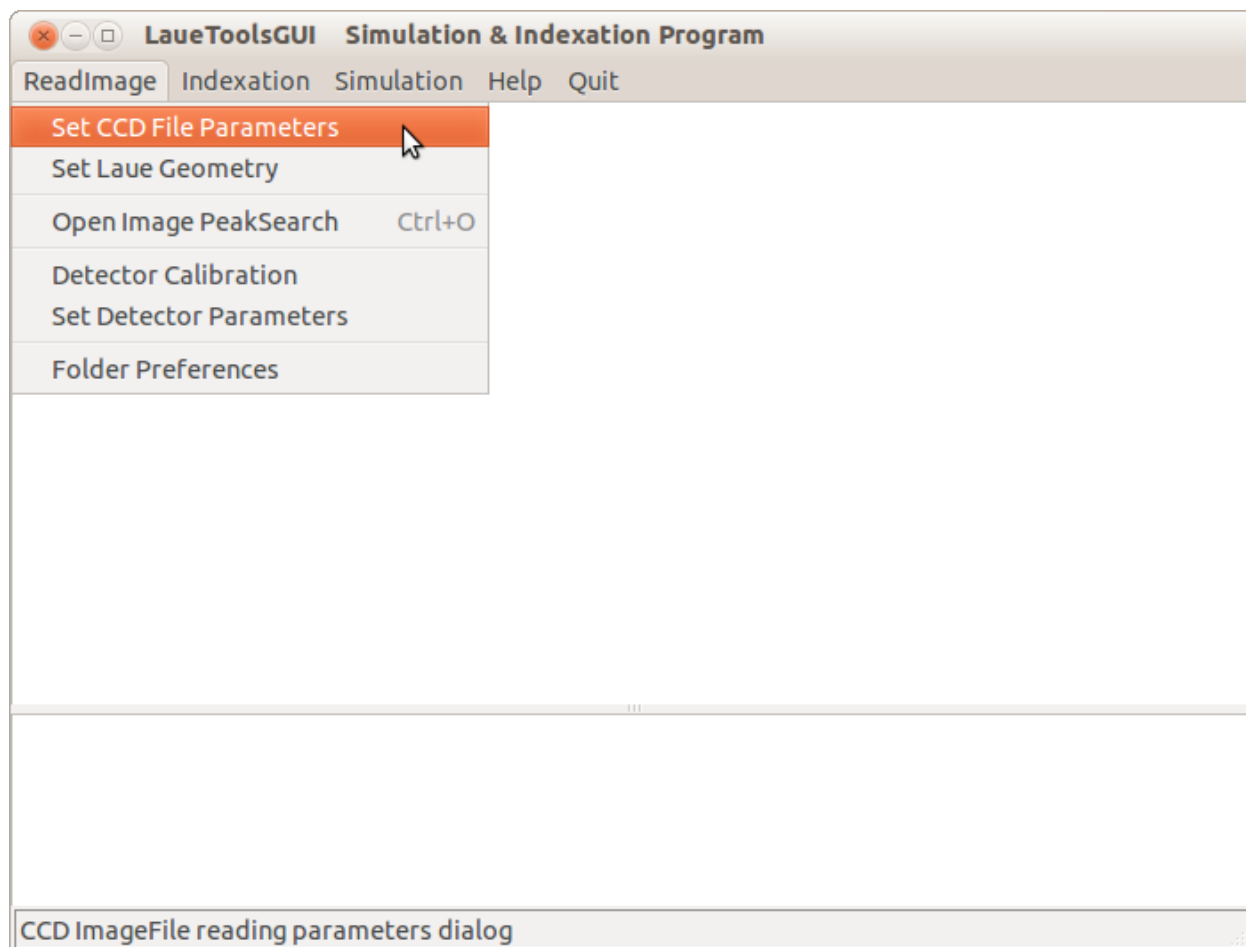
#### 5.1.1 Peak Search (PeaksearchGUI)

##### PeakSearchGUI

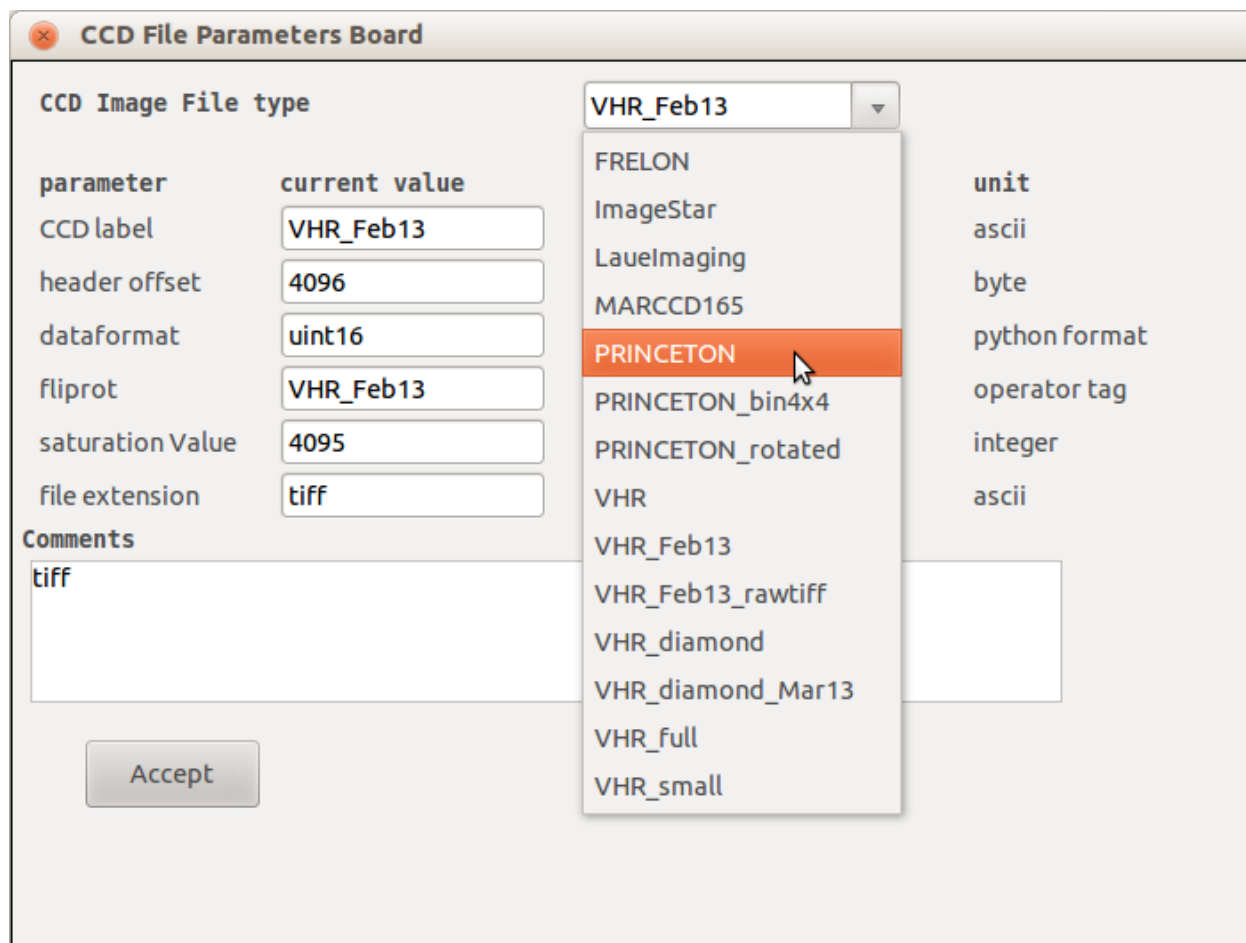
The module PeakSearchGUI.py is made to provide graphical tools to perform peak search and fitting. It enables also tools to get mosaic from a set of images.

##### Read Images and Binary files

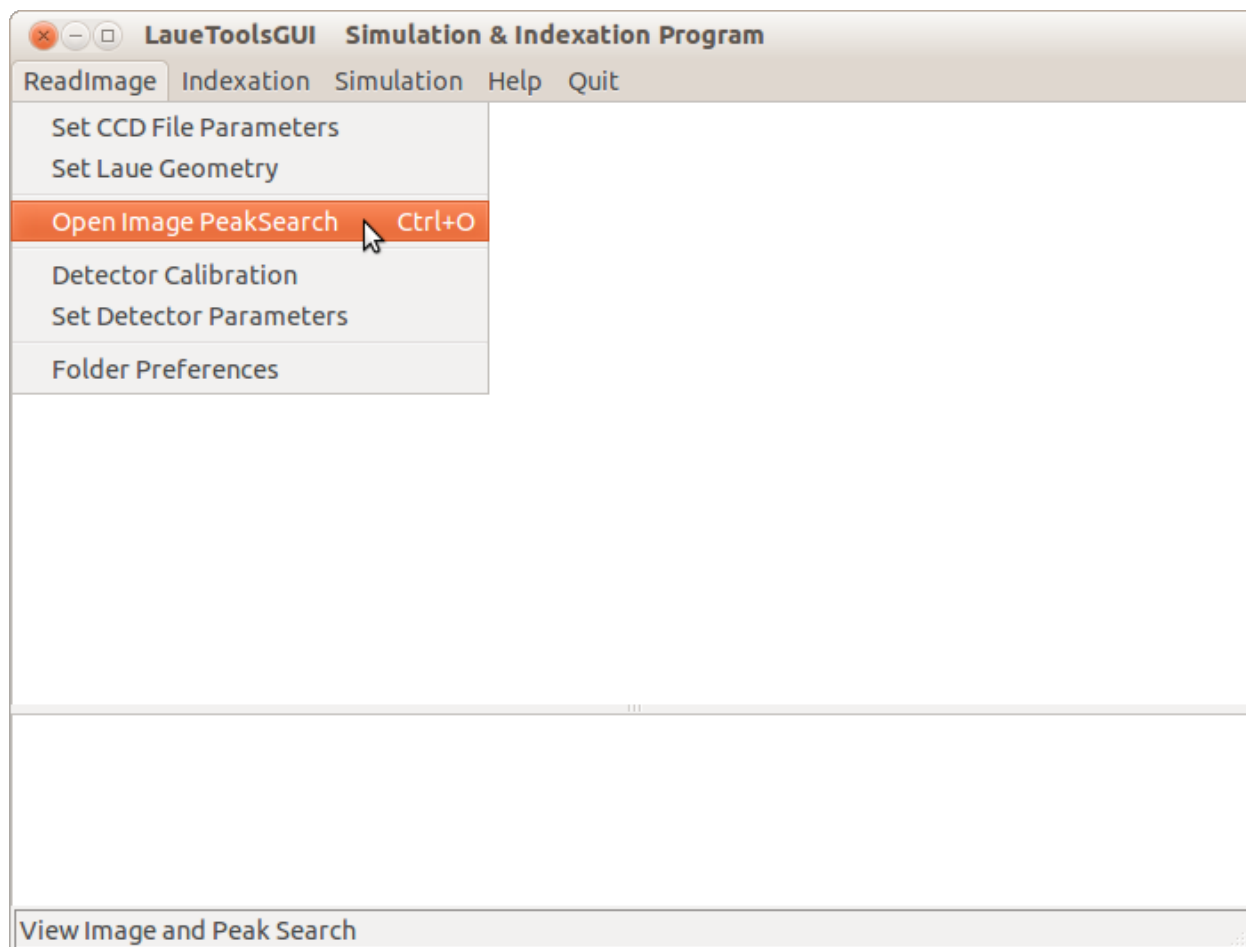
First select the detector you have used for the data collection in the menu Calibration



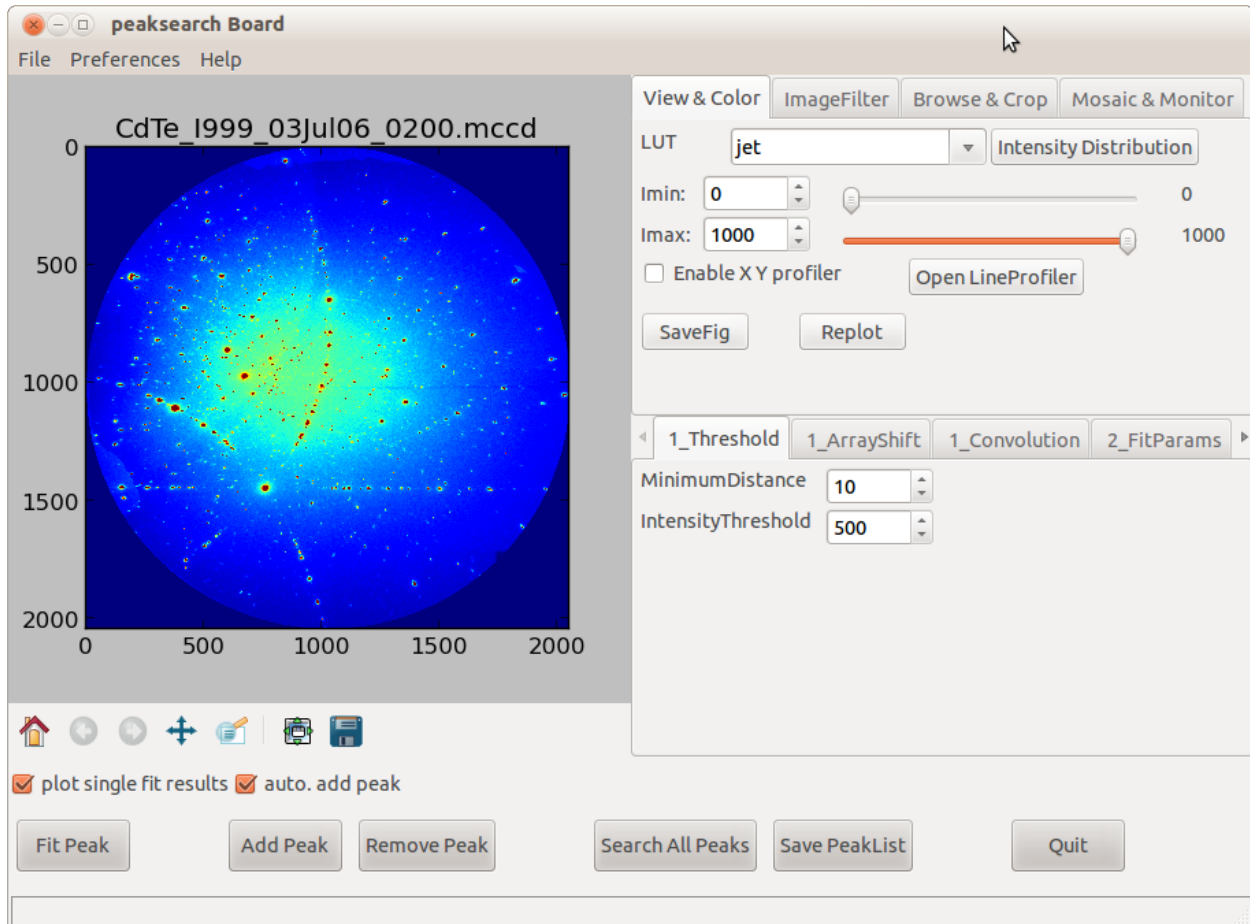
Choose the correct camera to define the binary image file parameters



View an image by selecting in the Menu File/Open Image and Peak Search.



Then you obtain a board enabling to browse your set of images and to search for peaks



This board is composed by composed by 4 TOP tabs:

- parameters of display (View & color)
- Digital image processing (Image & Filter)
- Browse a set of images and select a ROI (Browse & crop)
- Mosaic or image-related monitor from a ROI over a set of images

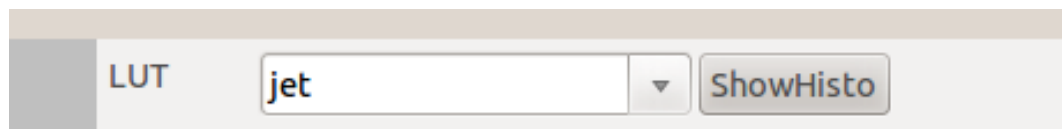
And 5 tabs at the middle

- 1 Selection of local maxima search Method
- 2 Fitting procedure parameters
- 3 Peaks List Manager

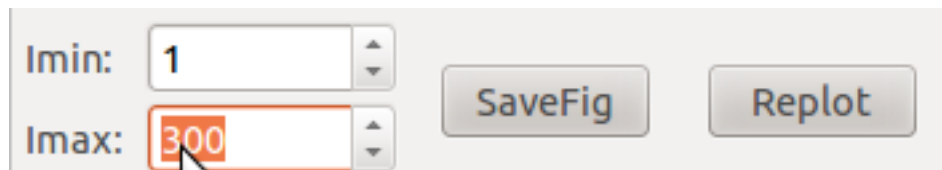
### View & Color

This panel helps for viewing images with an appropriate color LUT, getting some infos on pixels intensity on the whole image (distribution) or along line (line profilers)

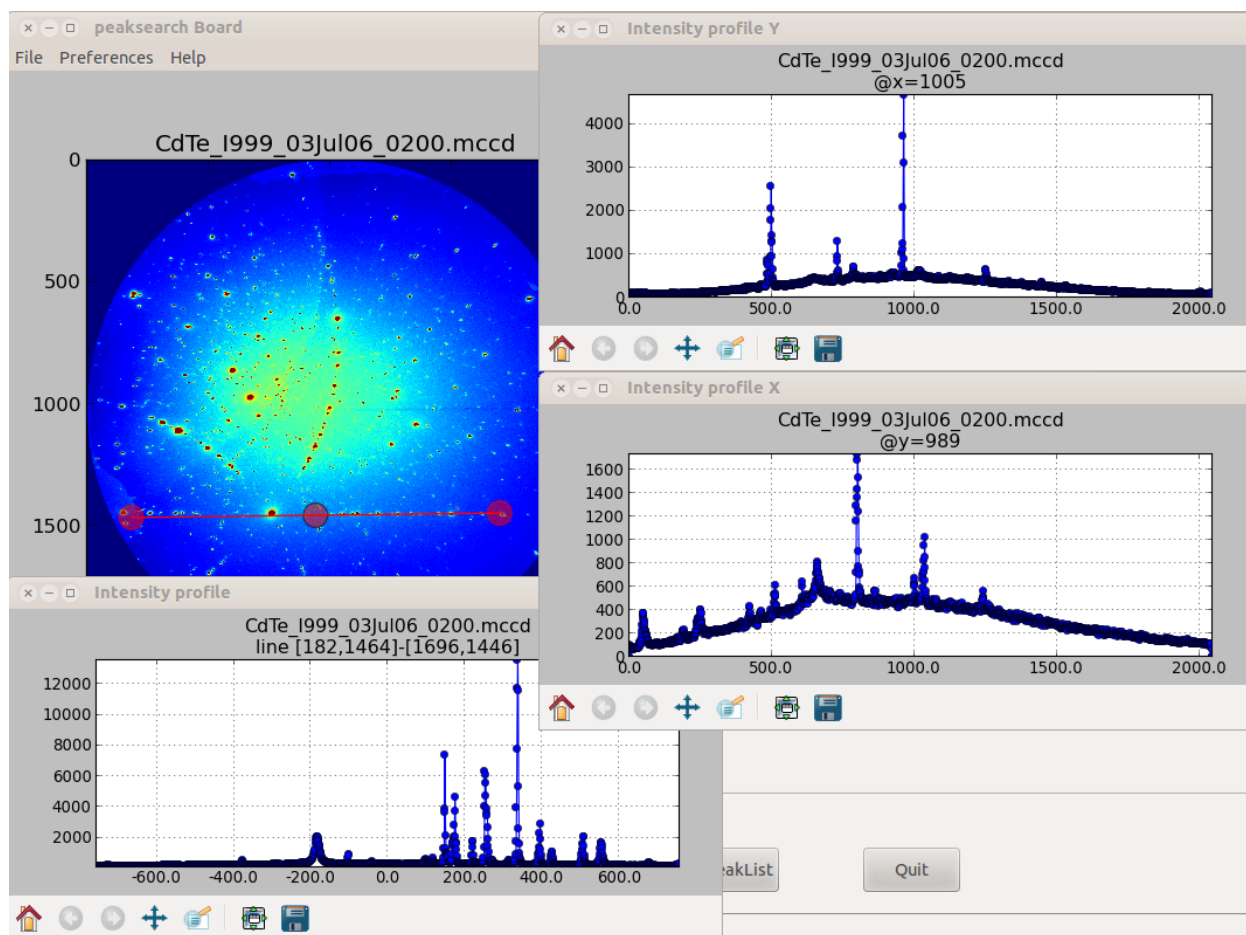
- Color Mapping (LUT) of the displayed image. `ShowHisto` displays the histogram of intensity distribution (nb of pixel as a function of pixel intensity)



- intensities limits taken into account by the LUT



- Open LineProfiler: 1D plot of pixel intensities along a movable-by-user line. And Eneable X Y Profiler: 1D plot of pixel intensities along X and Y from a clicked pixel position



## Image & Filter

This panel supplies digital image processing tools to filter current image and particularly remove background.

Blur Image computes the image filtered by a gaussian kernel (similar to a low pass filter). By checking Substract blur as background the raw image minus the filtered image can be displayed and used for the local maxima (blob) search.

Calculate image with A and B allows with an arithmetical formula with A (current image) and B (additional image to be open) the computation of a new image A' on which will be performed the local maxima (blob) searchChecking. By default this image will not be used to refine the position and shape of local maxima but the former and initial A image. Check use also for fit to apply the fit procedures on pixel intensities A'.

Save results saves on hard disk the A' image with the header contains and format of A.

### Browse & Crop

One can navigate on a set of images provided the images file name contains a numerical index with a constant prefix (e.g. myimage\_0032.ccd). Navigation with button with small step  $\text{index}-1$  and  $\text{index}+1$  corresponds to consecutive images collected with time or along a line on sample. Navigation with button with larger step (Nb of images per line (step index))  $\text{index}-10$  and  $\text{index}+10$  (for instance with larger step equals to 10) permits to look at the images collected along the direction perpendicular to the direction corresponding to the small step.

The Go To index button allows to read directly an image with an other in the same dataset. Auto  $\text{index}+1$  button will display the next image and wait for it if it is not already in the folder.

To navigate and display faster the image when browsing on a particular region of interest (ROI) of the images, you can crop the data (CropData) by specifying the half sizes (boxsize) of the cropping box in the two directions.

### Mosaic & Monitor

Several counters can be defined from pixel intensities in the same ROI (centered a clicked pixel with half box sizes given by user) over a set of images.

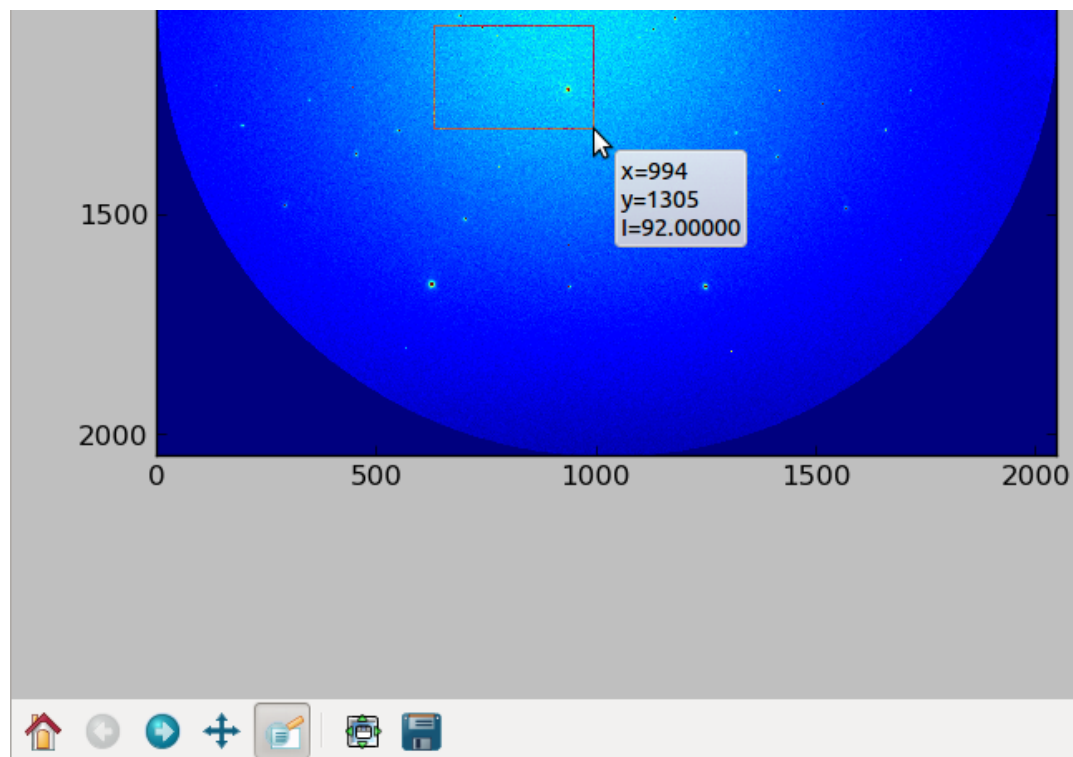
- Mosaic: Recompose a 2D raster scan from the selected ROI of each images as a function of image index

- Mean Value: plot a 1D graph or a 2D raster scan from the mean pixel value in the selected ROI of each images as a function of image index
- Max Value: plot a 1D graph or a 2D raster scan from the maximum pixel value in the selected ROI of each images as a function of image index
- Peak-to-peak Value (or 'peak to valley'): plot a 1D graph or a 2D raster scan from the largest pixel amplitude value in the selected ROI of each images as a function of image index
- Peak position: plot two 1D graphes of X and Y peak position of the peak in the selected ROI of each images as a function of image index

### Plot Tools

The standard toolbar is provided by the graphical 'matplotlib' library with the following features:

- A ROI can be set to zoom in the data.

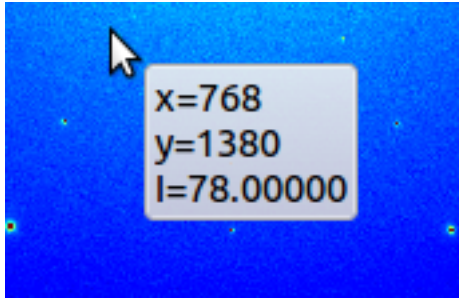


- This ROI can be moved easily with the pan button



- When hovering the mouse on the image pixel position and corresponding intensity are displayed

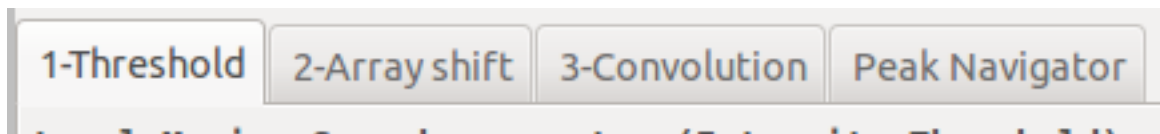




- Previous selected ROIs are stored and can be recalled by arrows (Home icon recalls the initial full image)

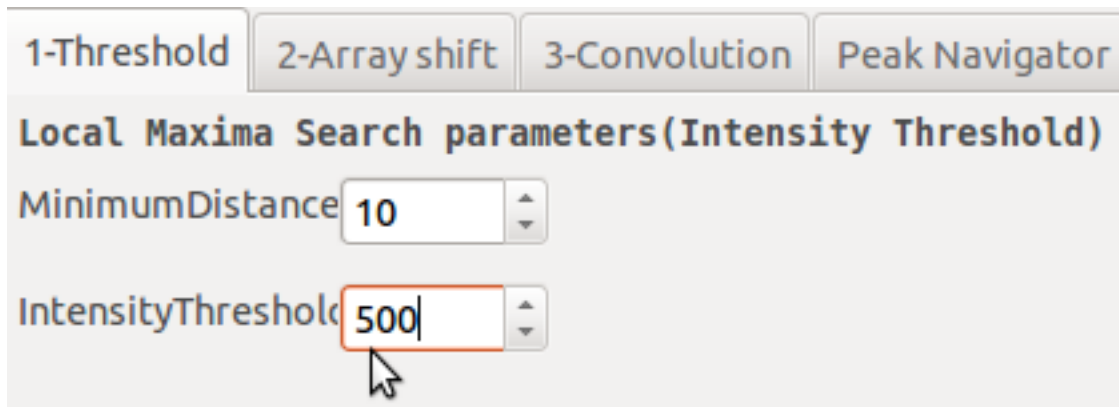
### Local Maxima or Blob Search

To guess the initial parameters fitting, 3 methods leads to a list of local maxima (or blobs)



#### Method 1: pixels above a threshold on raw image

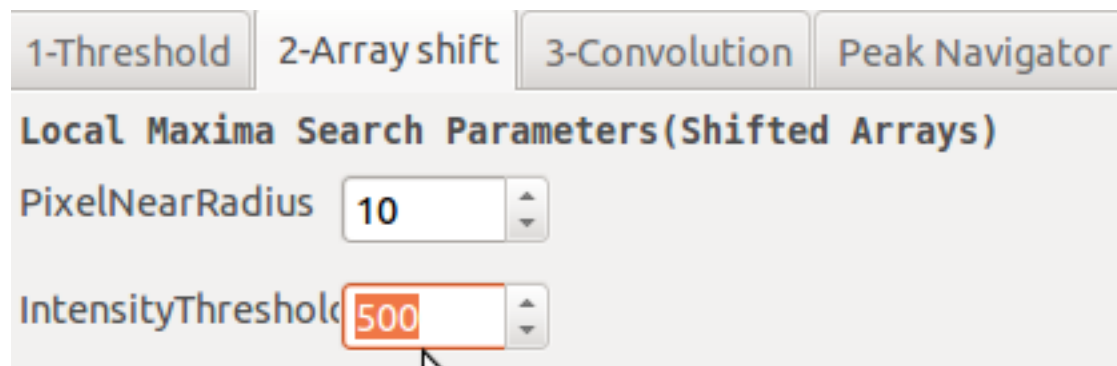
The basic method consists in considering every pixel higher than `intensityThreshold` as a local maxima pixel. If `intensityThreshold` is too high you will get only few pixels at the submit of laue spots. If `intensityThreshold` is too small you will too much pixels that you may stuck the software. `MinimumDistance` is the smallest distance separating two local maxima. A good habit is too check the highest background level (e.g close to the image centre) and set `intensityThreshold` to a larger value. But even in this case if (fluorescence) background varies a lot, you will miss peaks whose maximum intensities are below the threshold... This is why removing the background is mandatory.



#### Method 2: hottest pixel in a box (by array shift)

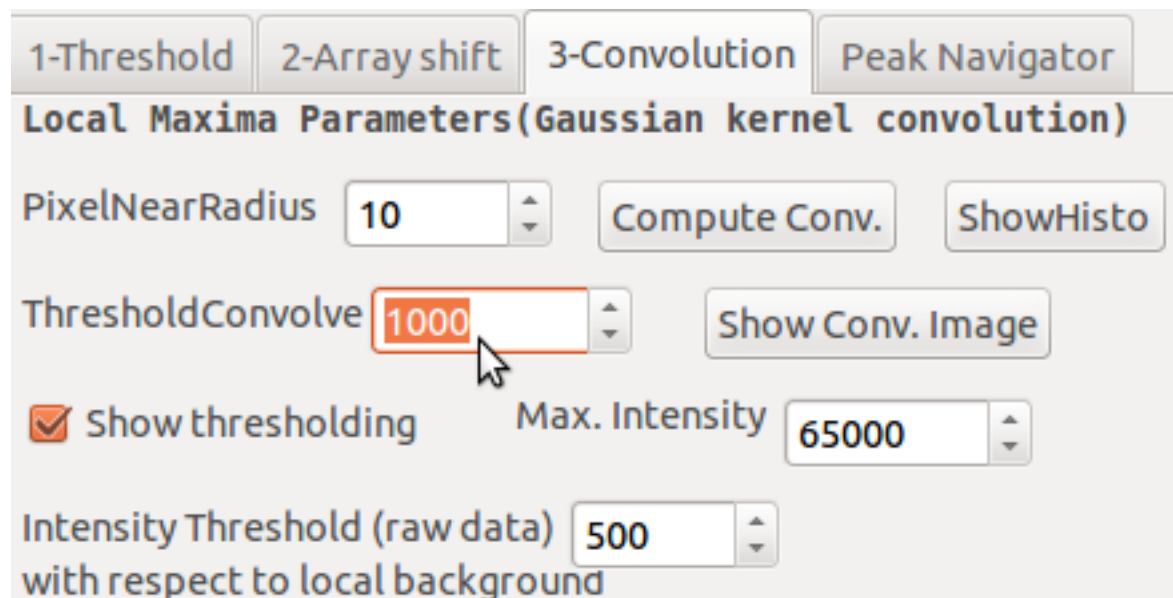
Second Method finds the hottest pixel in a small box given by `PixelNearRadius`. It shifts the whole data array in 8 directions and determines every pixel hotter than the others lying in these 8 directions. The **thresholding** with the `IntensityThreshold` level is performed on the intensity of these hot pixels **with respect to local background level** (set to the lowest pixel intensity in the box around the hot pixel). One drawback of this method is that 2 hot pixels

at the top of the peak but with strictly the same intensity are not detected (coincidence or more likely when peak is saturated).



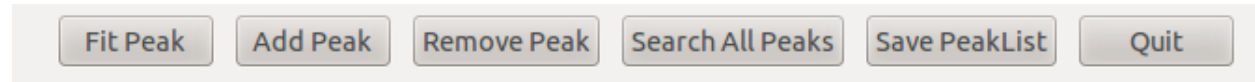
### Method 3: Peak enhancement by convolution on raw image

This is the fastest method as soon as you have found the few parameters value (for batch). The *Raw image* data (unsigned 16 bits integers) are convolved by a 2D gaussian (mexican-hat-like) kernel. The resulting *convolved Image* (floats) have intense region (called blobs) where pixel intensity 2D profile on raw data is similar to the kernel intensity profile. A first threshold with the level `ThresholdConvolve` (float) allows to select enhanced blob above a background. An good estimate of `ThresholdConvolve` value can be found by means of the pixel intensity histogram (`ShowHisto`): it corresponds to the (float) intensity that separates the hottest pixel population that belong to the peak (large abscissa value) from the weakest one that belong to background. With this blobs list, a second thresholding at `Intensity Threshold` (raw data) level is performed in the raw data pixel intensity with respect to local background level (like in method 2). `PixelNearRadius` value enables the user to reject too much closed spots. `Max. Intensity` value is used for the display of the convolved Image by clicking on `Show Conv. Image`. Thresholding can be visualize by checking `Show thresholding`.



## Peak Position Fit

The goal of the peak search is to have systematic values of Laue peaks and intensities. You can decide to fit or not the blob found. The Button `Search All Peaks` launches the local maxima chosen method and apply a fitting procedure (or not) to each found blob. A general peak list is built. By clicking close to a Laue spot in the image and clicking on the `Fit Peak` button, a gaussian fit is performed. This result can be added to the current general peak list with the button `Add Peak`. Clicking close to a Laue spot that belongs to the peak list (blue circle marker on top of the image) and pressing `Remove Peak` removes the laue spot from the list.



Parameters to choose the fitting model or no fit and to reject or not the fitting result of each peak according to deviation from the initial guessed position (`FitPixelDeviation`).

The dialog contains the following controls:

- Use peak position:** Radio buttons for `Hottest Pixel` (selected) and `Centroid`.
- FitFunction:** A dropdown menu with options `Gaussian`, `Lorentzian`, and `NoFit`. A mouse cursor is shown pointing at `Gaussian` in the top screenshot.
- xtol:** A text input field containing `0.001`.
- Boxsize:** A text input field containing `15` (with up/down arrows).
- FitPixelDev:** A text input field containing `2.0`.
- Max. Nb of Fits:** A text input field containing `2000`.
- Apply on whole Image:** A checked checkbox.

## Module functions

The next documentation comes from the docstring in the header of function or class definition.

PeakSearchGUI.py (PeakSearchBoard)

### 5.1.2 Indexation (LaueToolsGUI)

#### 5.1.3 Crystal unit cell refinement (LaueToolsGUI)

#### 5.1.4 Detector Geometry Calibration (DetectorCalibrationBoard)

#### 5.1.5 Laue pattern simulation of assembly of crystals (LaueSimulatorGUI)

## 5.2 Batch Processing Graphical User Interfaces

### 5.2.1 Peak Search (FileSeries.Peak\_Search)

### 5.2.2 Indexing and Unit Cell Refinement (FileSeries.Index\_Refine)

### 5.2.3 Summary of all refined data (FileSeries.Build\_Summary)

### 5.2.4 Plot of Quantities in Summary file (FileSeries.Plot\_Map2)

## 6.1 Basics of Laue Pattern peak search and Unit cell Refinement

### 6.1.1 This Notebook is a part of Tutorials on LaueTools Suite.

Author: J.-S. Micha

Last Revision: August 2019

tested with python3

#### Objectives

- Load and display Laue pattern images
- Perform a Peak Search
- Perform the indexation of a Laue spots list
- Perform the crystal orientation and unit cell refinement

Setting absolute path to LaueTools Modules if Lauetools has not been installed with pip. It is assumed that this notebook is located in a subfolder (normally Notebooks)

```
abspathLaueTools /home/micha/LaueToolsPy3/LaueTools
```

```
'/home/micha/LaueToolsPy3/LaueTools/__init__.py'
```

```
LaueToolsProjectFolder /home/micha/LaueToolsPy3/LaueTools
```

```
/home/micha/anaconda3/lib/python3.6/site-packages/h5py/__init__.py:36:
↳FutureWarning: Conversion of the second argument of issubdtype from float
↳to np.floating is deprecated. In future, it will be treated as np.float64
↳== np.dtype(float).type.
  from ._conv import register_converters as _register_converters
```

```
module Image / PIL is not installed
Cython compiled module 'gaussian2D' for fast computation is not installed!
module Image / PIL is not installed
```

Considering single image analysis (that belong to the LaueTools distribution)

Considering analysis of one image in dataset

**For information:** select image file of interest, in case of set of images with index. Then, splitting imagefilename allows to loop over images: prefix+index.extension

### Read image file, get data and display it

Function `readCCDimage()` returns dataimage as a 2D numpy array with the proper dimensions and orientation given by `framedim` and the geometrical transformations labelled by `fliprot`

```
Displaying Ge_blanc_0000.mccd

nb elements 4194304
framedim (2048, 2048)
framedim nb of elements 4194304
```

```
<IPython.core.display.Javascript object>
```

```
Text(0.5,1,'Ge_blanc_0000.mccd')
```

**\*peaksearch\*** is performed in two main steps: - 1) blobs or local maxima finder - 2) for blob, refinement starting from blob average center.

For the first step, `readCCDimage()` is called to obtain raw data if no different data array is provided with the argument `Data_for_localMaxima` (set to `None` by default). After second step, Peaksearch results can be purged from peaks already present in a file as an optional argument `Remove_BlackListedPeaks_fromfile`.

```
CCDLabel: MARCCD165
nb of pixels (4194304,)
nb elements 4194304
framedim (2048, 2048)
framedim nb of elements 4194304
image from filename ../LaueImages/Ge_blanc_0000.mccd read!
Read Image. Execution time : 0.006 seconds
Data.shape for local maxima (2048, 2048)
Using simple intensity thresholding to detect local maxima (method 1/3)
len(peaklist) 82
Local maxima search. Execution time : 0.336 seconds
Keep 82 from 82 initial peaks (ready for peak positions and shape fitting)
```

```
*****
```

```
82 local maxima found
```

```
    Fitting of each local maxima
```

```
addImax False
nb elements 4194304
framedim (2048, 2048)
framedim nb of elements 4194304
framedim in readoneimage_mancrops (2048, 2048)
fitting time for 82 peaks is : 0.2039
```

```

nb of results: 82
After fitting, 0/82 peaks have been rejected
  due to (final - initial position)> FitPixelDev = 10
0 spots have been rejected
  due to negative baseline
0 spots have been rejected
  due to much intensity
0 spots have been rejected
  due to weak intensity
0 spots have been rejected
  due to small peak size
0 spots have been rejected
  due to large peak size
ToTake {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19,
→20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38,
→39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57,
→58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76,
→77, 78, 79, 80, 81}
len(ToTake) 82

82 fitted peak(s)

Removing duplicates from fit

82 peaks found after removing duplicates (minimum intermaxima distance = 5)
peak search time 0.5514366626739502

```

### Spots properties:

peak\_X, peak\_Y, peak\_I, peak\_fwaxmaj, peak\_fwaxmin, peak\_inclination, Xdev, Ydev, peak\_bkg, Ipixmax,

Spots are sorted by intensity (according to the 2D gaussian fit)

```

Digital Spots properties for the 5 most intense spots
[[ 6.231334887002143e+02  1.657728161614024e+03  2.979937967247360e+04
   8.515577956136006e-01  7.511212178165599e-01  1.820409415704489e+01
   1.334887002143432e-01 -2.718383859764799e-01  2.966046444454810e+02
   2.000000000000000e+02]
 [ 1.244326205473488e+03  1.662150473603958e+03  2.242563070589073e+04
   6.977180389301006e-01  6.390759549880105e-01  1.293529253564775e+02
   3.262054734875619e-01  1.504736039580621e-01  1.998717405717028e+02
   2.000000000000000e+02]
 [ 9.330365915823824e+02  1.215440948340315e+03  2.219753607623998e+04
   7.846021988134166e-01  7.862341648303387e-01  3.246533552343026e+02
   3.659158238235705e-02  4.409483403153445e-01  2.110241433113940e+02
   2.000000000000000e+02]
 [ 5.852254505694141e+02  5.887990375606668e+02  9.528825561251553e+03
   7.791458064142315e-01  7.399755695034808e-01  1.068330480796285e+02
   2.254505694140789e-01 -2.00962439332388e-01  1.501265419627265e+02
   2.000000000000000e+02]
 [ 1.276607259246803e+03  6.002998208781320e+02  9.153971543092421e+03
   8.153699693677648e-01  8.035243472697837e-01  8.044575533084571e+01
  -3.927407531973586e-01  2.998208781319818e-01  1.324821469609317e+02
   2.000000000000000e+02]
 [ 9.326643784727646e+02  7.500763813513167e+02  6.940078500922347e+03
   7.065049204848781e-01  7.482895847319173e-01  8.063512928783894e+00
  -3.356215272353893e-01  7.638135131674062e-02  1.317606341369902e+02

```

(continues on next page)

(continued from previous page)

```
2.0000000000000000e+02]]
```

X, Y pixel refinement positions **for** the first 5 spots

```
array([[ 623.1334887002143, 1657.7281616140235],
       [1244.3262054734876, 1662.150473603958 ],
       [ 933.0365915823824, 1215.4409483403153],
       [ 585.2254505694141,  588.7990375606668],
       [1276.6072592468026,  600.299820878132 ]])
```

*add markers to image*

```
<IPython.core.display.Javascript object>
```

**List of peaks props is written in a file with extension .dat, here the variable is “datfilename”**

```
peaklist.shape (82, 10)
fullpathimagefile ../LaueImages/Ge_blanc_0000.mccd
imagefolder ../LaueImages
table of 82 peak(s) with 10 columns has been written in
/home/micha/LaueToolsPy3/LaueTools/LaueImages/Ge_blanc_0000Notebook.dat
```

## Now indexing

### geometry calibration parameters

Either you fill manually the dict of parameters or you read a file .det

```
reading geometry calibration file
calib = [ 7.07760e+01  9.41760e+02  1.08244e+03  6.29000e-01 -6.85000e-01
          7.91400e-02  2.04800e+03  2.04800e+03]
matrix = [ 0.995829 -0.071471 -0.056709  0.012247  0.720654 -0.693187  0.09041
          0.689602  0.718523]
```

### creation of a .cor file containing accurate scattering angles thanks to detector geometry parameters

Only list of spots with scattering angles can be indexed. In LaueTools .dat file contains only X, Y pixel positions, .cor file contains in addition 2theta and chi scattering angles, and .fit file in addition indexed results properties (such as h, k, l, energy, grain index ...)

Entering CrystalParameters **\*\*---\*\*\*\*\***

```
nb of spots and columns in .dat file (82, 3)
file ../LaueImages/Ge_blanc_0000Notebook.dat
containing 82 peaks
(2theta chi X Y I) written in ../LaueImages/Ge_blanc_0000Notebook.cor
```

### create instance of an objet spotsset class



```

Cython compiled module for fast computation of Laue spots is not installed!
Cython compiled 'angulardist' module for fast computation of angular distance is not
↳ installed!
Using default module
Cython compiled module for fast computation of angular distance is not installed!
module Image / PIL is not installed
CCDcalib in readfile_cor {'dd': 70.776, 'xcen': 941.76, 'ycen': 1082.44, 'xbet': 0.
↳ 629, 'xgam': -0.685, 'xpixelsize': 0.07914, 'ypixelsize': 0.07914, 'CCDLabel':
↳ 'MARCCD165', 'framedim': [2048.0, 2048.0], 'detectordiameter': 162.07872, 'kf_
↳ direction': 'Z>0', 'pixelsize': 0.07914}
CCD Detector parameters read from .cor file
CCDcalibdict {'dd': 70.776, 'xcen': 941.76, 'ycen': 1082.44, 'xbet': 0.629, 'xgam': -
↳ 0.685, 'xpixelsize': 0.07914, 'ypixelsize': 0.07914, 'CCDLabel': 'MARCCD165',
↳ 'framedim': [2048.0, 2048.0], 'detectordiameter': 162.07872, 'kf_direction': 'Z>0',
↳ 'pixelsize': 0.07914}

```

True

```

array([[ 0.0000000e+00,  5.8426915e+01,  2.0130035e+01,  6.2313000e+02,
         1.6577300e+03,  2.9799380e+04],
       [ 1.0000000e+00,  5.7634672e+01, -1.8415523e+01,  1.2443300e+03,
         1.6621500e+03,  2.2425630e+04],
       [ 2.0000000e+00,  8.0919846e+01,  6.6158100e-01,  9.3304000e+02,
         1.2154400e+03,  2.2197540e+04]])

```

#### \*Set parameters for indexation: Ge, maximum energy\*

All materials are listed in dict\_LaueTools.py in dict\_Materials. You can edit/modify the module (then a restart of the kernel is necessary)

**Before launching the indexation procedure you may want to check a solution found elsewhere or sometimes ago. Then fill “previousResults” as shown below**

**Then launch indexation by specifying some arguments of the method “IndexSpotsSet”:**

```

- nbGrainstoFind: nb of grains of this material you want to find
- set_central_spots_hkl: imposed miller indices [h,k,l] of central spots (set A of
↳ spots) else : None
...

```

```

CCDcalib in readfile_cor {'dd': 70.776, 'xcen': 941.76, 'ycen': 1082.44,
↳ 'xbet': 0.629, 'xgam': -0.685, 'xpixelsize': 0.07914, 'ypixelsize': 0.07914,
↳ 'CCDLabel': 'MARCCD165', 'framedim': [2048.0, 2048.0], 'detectordiameter':
↳ 162.07872, 'kf_direction': 'Z>0', 'pixelsize': 0.07914}
CCD Detector parameters read from .cor file
CCDcalibdict {'dd': 70.776, 'xcen': 941.76, 'ycen': 1082.44, 'xbet': 0.629,
↳ 'xgam': -0.685, 'xpixelsize': 0.07914, 'ypixelsize': 0.07914, 'CCDLabel':
↳ 'MARCCD165', 'framedim': [2048.0, 2048.0], 'detectordiameter': 162.07872,
↳ 'kf_direction': 'Z>0', 'pixelsize': 0.07914}
self.pixelsize in IndexSpotsSet 0.07914
ResolutionAngstromLUT in IndexSpotsSet False

```

Remaining nb of spots to index for grain #0 : 82

★★

start to index grain #0 of Material: Ge

★★

```

providing new set of matrices Using Angles LUT template matching
nbspot 82
NBMAXPROBED 10
nbspot 82
set_central_spots_hkl [0, 1, 1]
Central set of exp. spotDistances from spot_index_central_list probed
self.absolute_index [ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18_
→19 20 21 22 23
 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47
 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71
 72 73 74 75 76 77 78 79 80 81]
spot_index_central_list [0, 1, 2, 3, 4]
[0 1 2 3 4]
LUT is None when entering getOrientMatrices()
set_central_spots_hkl [0, 1, 1]
set_central_spots_hkl is not None in getOrientMatrices()
set_central_spots_hkl [0 1 1]
set_central_spots_hkl.shape (3,)
case: 1a
set_central_spots_hkl_list [[0 1 1]
 [0 1 1]
 [0 1 1]
 [0 1 1]
 [0 1 1]]
cubicSymmetry True
LUT_tol_angle 0.5
---***-----*
Calculating all possible matrices from exp spot #0 and the 9 other(s)
hkl in getOrientMatrices [0 1 1] <class 'numpy.ndarray'>
using LUTspecific
LUTspecific is None for k_centspot_index 0 in getOrientMatrices()
hkl1 in matrices_from_onespot_hkl() [0 1 1]
Computing hkl2 list for specific or cubic LUT in matrices_from_onespot_hkl()
Calculating LUT in PlanePairs_from2sets()
Looking up planes pairs in LUT from exp. spots (0, 2):
Looking up planes pairs in LUT from exp. spots (0, 6):
Looking up planes pairs in LUT from exp. spots (0, 9):
calculating matching rates of solutions for exp. spots [0, 2]
calculating matching rates of solutions for exp. spots [0, 6]
calculating matching rates of solutions for exp. spots [0, 9]

---***-----*
Calculating all possible matrices from exp spot #1 and the 9 other(s)
hkl in getOrientMatrices [0 1 1] <class 'numpy.ndarray'>
using LUTspecific
LUTspecific is not None for k_centspot_index 1 in getOrientMatrices()
hkl1 in matrices_from_onespot_hkl() [0 1 1]
Using specific LUT in matrices_from_onespot_hkl()
Looking up planes pairs in LUT from exp. spots (1, 2):
Looking up planes pairs in LUT from exp. spots (1, 7):

```

```

Looking up planes pairs in LUT from exp. spots (1, 9):
calculating matching rates of solutions for exp. spots [1, 2]
calculating matching rates of solutions for exp. spots [1, 7]
calculating matching rates of solutions for exp. spots [1, 9]

---***-----*
Calculating all possible matrices from exp spot #2 and the 9 other(s)
hkl1 in getOrientMatrices [0 1 1] <class 'numpy.ndarray'>
using LUTspecific
LUTspecific is not None for k_centspot_index 2 in getOrientMatrices()
hkl11 in matrices_from_onespot_hkl() [0 1 1]
Using specific LUT in matrices_from_onespot_hkl()
Looking up planes pairs in LUT from exp. spots (2, 0):
Looking up planes pairs in LUT from exp. spots (2, 1):
Looking up planes pairs in LUT from exp. spots (2, 9):
calculating matching rates of solutions for exp. spots [2, 0]
calculating matching rates of solutions for exp. spots [2, 1]
calculating matching rates of solutions for exp. spots [2, 9]

---***-----*
Calculating all possible matrices from exp spot #3 and the 9 other(s)
hkl1 in getOrientMatrices [0 1 1] <class 'numpy.ndarray'>
using LUTspecific
LUTspecific is not None for k_centspot_index 3 in getOrientMatrices()
hkl11 in matrices_from_onespot_hkl() [0 1 1]
Using specific LUT in matrices_from_onespot_hkl()
Looking up planes pairs in LUT from exp. spots (3, 7):
Looking up planes pairs in LUT from exp. spots (3, 8):
Looking up planes pairs in LUT from exp. spots (3, 9):
calculating matching rates of solutions for exp. spots [3, 7]
calculating matching rates of solutions for exp. spots [3, 8]
calculating matching rates of solutions for exp. spots [3, 9]

---***-----*
Calculating all possible matrices from exp spot #4 and the 9 other(s)
hkl1 in getOrientMatrices [0 1 1] <class 'numpy.ndarray'>
using LUTspecific
LUTspecific is not None for k_centspot_index 4 in getOrientMatrices()
hkl11 in matrices_from_onespot_hkl() [0 1 1]
Using specific LUT in matrices_from_onespot_hkl()
Looking up planes pairs in LUT from exp. spots (4, 6):
Looking up planes pairs in LUT from exp. spots (4, 8):
Looking up planes pairs in LUT from exp. spots (4, 9):
calculating matching rates of solutions for exp. spots [4, 6]
calculating matching rates of solutions for exp. spots [4, 8]
calculating matching rates of solutions for exp. spots [4, 9]

-----
results:

```

```
matrix:                                matching results
[ 0.071442298339536 -0.056791122889477 -0.995826674863109]    res: [94.0, ↪
↪135.0] 0.005 69.63
[-0.720597705663885 -0.693247631822884 -0.012110638459969]    spot ↪
↪indices [0 9]
[-0.689608632382347  0.718518569419943 -0.090393581312322]    planes [[-2.
↪0, 1.0, 1.0], [0.0, 1.0, 1.0]]

[ 0.071284911945937 -0.056791915954001 -0.995837908301915]    res: [93.0, ↪
↪134.0] 0.006 69.40
[-0.720581964957976 -0.693265307713712 -0.012035152592104]    spot ↪
↪indices [1 9]
[-0.68968586701208  0.718453369664079 -0.09032253573791 ]    planes [[-1.
↪0, 2.0, 1.0], [0.0, 1.0, 1.0]]

[-0.071331310042019 -0.995834915200135 -0.056786141584281]    res: [93.0, ↪
↪134.0] 0.005 69.40
[ 0.720561755804718 -0.012058289359476 -0.693285910522741]    spot ↪
↪indices [2 9]
[ 0.689652960030445 -0.090345399841628  0.718482082900264]    planes [[1.
↪0, 1.0, 1.0], [0.0, 1.0, 1.0]]

[-0.071391558888759 -0.995830258190095 -0.056792096214888]    res: [94.0, ↪
↪135.0] 0.006 69.63
[ 0.720671448854468 -0.012140732995851 -0.693170444701969]    spot ↪
↪indices [3 9]
[ 0.689588625514858 -0.090412962995124  0.718535332243983]    planes [[2.
↪0, 3.0, 1.0], [0.0, 1.0, 1.0]]

[ 0.071340651481789 -0.056823338834159 -0.995832124210648]    res: [94.0, ↪
↪135.0] 0.005 69.63
[-0.720605825766681 -0.693235936569283 -0.012295532523216]    spot ↪
↪indices [4 9]
[-0.689563524109094  0.718557247109645 -0.090430242974657]    planes [[-1.
↪0, 2.0, 3.0], [0.0, 1.0, 1.0]]

Number of matrices found (nb_sol): 5
set_central_spots_hkl in FindOrientMatrices [0, 1, 1]
Merging matrices
keep_only_equivalent = False
sorting according to rank
rank [0 4 3 2 1]
```

```
-----
results:
matrix:                                matching results
[ 0.071442298339536 -0.056791122889477 -0.995826674863109]    res: [ 94. ↪
↪135.] 0.005 69.63
[-0.720597705663885 -0.693247631822884 -0.012110638459969]
[-0.689608632382347  0.718518569419943 -0.090393581312322]

[ 0.071340651481789 -0.056823338834159 -0.995832124210648]    res: [ 94. ↪
↪135.] 0.005 69.63
[-0.720605825766681 -0.693235936569283 -0.012295532523216]
```

```

[-0.689563524109094  0.718557247109645 -0.090430242974657]

[-0.071391558888759 -0.995830258190095 -0.056792096214888]      res: [ 94.
↪135.] 0.006 69.63
[ 0.720671448854468 -0.012140732995851 -0.693170444701969]
[ 0.689588625514858 -0.090412962995124  0.718535332243983]

[-0.071331310042019 -0.995834915200135 -0.056786141584281]      res: [ 93.
↪134.] 0.005 69.40
[ 0.720561755804718 -0.012058289359476 -0.693285910522741]
[ 0.689652960030445 -0.090345399841628  0.718482082900264]

[ 0.071284911945937 -0.056791915954001 -0.995837908301915]      res: [ 93.
↪134.] 0.006 69.40
[-0.720581964957976 -0.693265307713712 -0.012035152592104]
[-0.68968586701208  0.718453369664079 -0.09032253573791 ]

Nb of potential orientation matrice(s) UB found: 5
[[[ 0.071442298339536 -0.056791122889477 -0.995826674863109]
  [-0.720597705663885 -0.693247631822884 -0.012110638459969]
  [-0.689608632382347  0.718518569419943 -0.090393581312322]]]

[[[ 0.071340651481789 -0.056823338834159 -0.995832124210648]
  [-0.720605825766681 -0.693235936569283 -0.012295532523216]
  [-0.689563524109094  0.718557247109645 -0.090430242974657]]]

[[[-0.071391558888759 -0.995830258190095 -0.056792096214888]
  [ 0.720671448854468 -0.012140732995851 -0.693170444701969]
  [ 0.689588625514858 -0.090412962995124  0.718535332243983]]]

[[[-0.071331310042019 -0.995834915200135 -0.056786141584281]
  [ 0.720561755804718 -0.012058289359476 -0.693285910522741]
  [ 0.689652960030445 -0.090345399841628  0.718482082900264]]]

[[[ 0.071284911945937 -0.056791915954001 -0.995837908301915]
  [-0.720581964957976 -0.693265307713712 -0.012035152592104]
  [-0.68968586701208  0.718453369664079 -0.09032253573791 ]]]]
Nb of potential UBs 5

Working with a new stack of orientation matrices
MATCHINGRATE_THRESHOLD_IAL= 100.0
has not been reached! All potential solutions have been calculated
taking the first one only.
bestUB object <LaueTools.indexingSpotsSet.OrientMatrix object at
↪0x7f7c954f5e80>

```

```
-----refining grain orientation and strain #0-----
```

```
refining grain #0 step -----0
```

```
bestUB <LaueTools.indexingSpotsSet.OrientMatrix object at 0x7f7c954f5e80>
True it is an OrientMatrix object
```

```
Orientation <LaueTools.indexingSpotsSet.OrientMatrix object at 0x7f7c954f5e80>
matrix [[-0.071391558888759 -0.995830258190095 -0.056792096214888]
 [ 0.720671448854468 -0.012140732995851 -0.693170444701969]
 [ 0.689588625514858 -0.090412962995124 0.718535332243983]]
*nb of selected spots in AssignHKL** 82
UBOrientMatrix [[-0.071391558888759 -0.995830258190095 -0.056792096214888]
 [ 0.720671448854468 -0.012140732995851 -0.693170444701969]
 [ 0.689588625514858 -0.090412962995124 0.718535332243983]]
For angular tolerance 0.50 deg
Nb of pairs found / nb total of expected spots: 81/147
Matching Rate : 55.10
Nb missing reflections: 66
```

grain #0 : 81 links to simulated spots have been found

```
*****mean pixel deviation 0.2522039400422887 *****
Initial residues [0.191356096913678 0.158479888122211 0.125984922997516 0.
↪007464331088751
0.220920883122328 0.065197410024489 0.405464259132319 0.079769518806149
0.309835172580193 0.024815180122634 0.146635771529913 0.197926454567734
0.253815574123594 0.241517985294699 0.301875442673439 0.217498144625921
0.186026257638361 0.152430964466482 0.022468745875909 0.372498387808433
0.225884815274198 0.155682523936061 0.308213213363587 0.354423361607117
0.237793437184287 0.344146246502948 0.117700835663451 0.22732103372742
0.263538267437741 0.133994037769124 0.091015982918167 0.367309714380722
0.359174426753832 0.281533512444384 0.191021625928391 0.219461033259323
0.371983339466526 0.3512796731268 0.298580209240117 0.447936020775024
0.160438308161376 0.631208433750478 0.420060120050684 0.195238104695171
0.051118832992816 0.159003375870547 0.354123955360538 0.049380652521924
0.301744705672337 0.127112320459672 0.082920786835417 0.19281838475986
0.130182524243209 0.332360152782496 0.533160923855596 0.276782907418236
0.125265672564509 0.184320173657227 0.238789408490181 0.149666955002009
0.473603697641706 0.235878500572685 0.425250385266374 0.445829965130009
0.255853120078437 0.271987274130697 0.298711159306184 0.310382741777609
0.228936459666657 0.374245425300233 0.100039587285176 0.096572087547537
0.196098380129156 0.140612883827292 0.338936919946618 0.526244208385607
0.190627233723994 0.629487817359844 0.233333060530866 0.316852495355672
0.61336433904477 ]
```

```
*****
first error with initial values of: ['b/a', 'c/a', 'a12', 'a13', 'a23', ↪
↪'theta1', 'theta2', 'theta3']
```

```
*****
```

```
*****mean pixel deviation 0.2522039400422887 *****
```

```
*****
Fitting parameters: ['b/a', 'c/a', 'a12', 'a13', 'a23', 'theta1', 'theta2', ↪
↪'theta3']
```

\*\*\*\*\*

```
With initial values [1. 1. 0. 0. 0. 0. 0. 0.]
code results 1
nb iterations 189
mesg Both actual and predicted relative reductions in the sum of squares
are at most 0.000000
strain_sol [ 9.999862096544356e-01  9.999941276097474e-01 -7.144700897145182e-
↪06
  5.728542645493929e-05  1.242229382006964e-05  2.977984275077042e-05
-1.953472667313407e-03  6.919002586889341e-03]
```

\*\*\*\*\* End of Fitting - Final errors \*\*\*\*\*

```
*****mean pixel deviation      0.18215488925149526      *****
devstrain, lattice_parameter_direct_strain [[-4.815998025634964e-06  5.
↪250997004746657e-06 -2.870284279887629e-05]
 [ 5.250997004746657e-06  7.352907498721824e-06 -9.199263307200638e-06]
 [-2.870284279887629e-05 -9.199263307200638e-06 -2.536909473086861e-06]] [ 5.
↪657509728355469  5.657578574250457  5.657522951317707
 90.0010541881893  90.00328909016345  89.99939828842365 ]
For comparison: a,b,c are rescaled with respect to the reference value of a =_
↪5.657500 Angstroms
lattice_parameter_direct_strain [ 5.6575 5.657568845776604  5.
↪6575132229395
 90.0010541881893  90.00328909016345  89.99939828842365 ]
devstrain1, lattice_parameter_direct_strain1 [[-4.815998025634964e-06  5.
↪250997004746657e-06 -2.870284279887629e-05]
 [ 5.250997004746657e-06  7.352907498721824e-06 -9.199263307200638e-06]
 [-2.870284279887629e-05 -9.199263307200638e-06 -2.536909473086861e-06]] [ 5.
↪6575 5.657568845776604  5.6575132229395
 90.0010541881893  90.00328909016345  89.99939828842365 ]
new UBs matrix in q= UBs G (s for strain)
strain_direct [[ 1.719550237533340e-06  5.250997004746657e-06 -2.
↪870284279887629e-05]
 [ 5.250997004746657e-06  1.388845576189013e-05 -9.199263307200638e-06]
 [-2.870284279887629e-05 -9.199263307200638e-06  3.998638790081444e-06]]
deviatoric strain [[-4.815998025634964e-06  5.250997004746657e-06 -2.
↪870284279887629e-05]
 [ 5.250997004746657e-06  7.352907498721824e-06 -9.199263307200638e-06]
 [-2.870284279887629e-05 -9.199263307200638e-06 -2.536909473086861e-06]]
new UBs matrix in q= UBs G (s for strain)
strain_direct [[ 1.719550237533340e-06  5.250997004746657e-06 -2.
↪870284279887629e-05]
 [ 5.250997004746657e-06  1.388845576189013e-05 -9.199263307200638e-06]
 [-2.870284279887629e-05 -9.199263307200638e-06  3.998638790081444e-06]]
deviatoric strain [[-4.815998025634964e-06  5.250997004746657e-06 -2.
↪870284279887629e-05]
 [ 5.250997004746657e-06  7.352907498721824e-06 -9.199263307200638e-06]
 [-2.870284279887629e-05 -9.199263307200638e-06 -2.536909473086861e-06]]
For comparison: a,b,c are rescaled with respect to the reference value of a =_
↪5.657500 Angstroms
```

```
lattice_parameter_direct_strain [ 5.6575          5.657568845776604  5.
↪6575132229395
  90.0010541881893  90.00328909016345  89.99939828842365 ]
final lattice_parameters [ 5.6575          5.657568845776604  5.
↪6575132229395
  90.0010541881893  90.00328909016345  89.99939828842365 ]
UB and strain refinement completed
True it is an OrientMatrix object
Orientation <LaueTools.indexingSpotsSet.OrientMatrix object at 0x7f7c851edeb8>
matrix [[-0.071478224945242 -0.995814588672313 -0.056724144676328]
 [ 0.720639310822985 -0.012262885888099 -0.693156594892675]
 [ 0.689613233174427 -0.090416539541802  0.718545890295648]]
*nb of selected spots in AssignHKL*** 82
UBOrientMatrix [[-0.071478224945242 -0.995814588672313 -0.056724144676328]
 [ 0.720639310822985 -0.012262885888099 -0.693156594892675]
 [ 0.689613233174427 -0.090416539541802  0.718545890295648]]
For angular tolerance 0.50 deg
Nb of pairs found / nb total of expected spots: 81/147
Matching Rate : 55.10
Nb missing reflections: 66

grain #0 : 81 links to simulated spots have been found
GoodRefinement condition is True
nb_updates 81 compared to 6

refining grain #0 step ----1

bestUB <LaueTools.indexingSpotsSet.OrientMatrix object at 0x7f7c954f5e80>
True it is an OrientMatrix object
Orientation <LaueTools.indexingSpotsSet.OrientMatrix object at 0x7f7c954f5e80>
matrix [[-0.071391558888759 -0.995830258190095 -0.056792096214888]
 [ 0.720671448854468 -0.012140732995851 -0.693170444701969]
 [ 0.689588625514858 -0.090412962995124  0.718535332243983]]
*nb of selected spots in AssignHKL*** 82
UBOrientMatrix [[-0.071391558888759 -0.995830258190095 -0.056792096214888]
 [ 0.720671448854468 -0.012140732995851 -0.693170444701969]
 [ 0.689588625514858 -0.090412962995124  0.718535332243983]]
For angular tolerance 0.20 deg
Nb of pairs found / nb total of expected spots: 81/147
Matching Rate : 55.10
Nb missing reflections: 66

grain #0 : 81 links to simulated spots have been found
*****mean pixel deviation 0.2522039400422887 *****
Initial residues [0.191356096913678 0.158479888122211 0.125984922997516 0.
↪007464331088751
 0.220920883122328 0.065197410024489 0.405464259132319 0.079769518806149
 0.309835172580193 0.024815180122634 0.146635771529913 0.197926454567734
 0.253815574123594 0.241517985294699 0.301875442673439 0.217498144625921
 0.186026257638361 0.152430964466482 0.022468745875909 0.372498387808433
 0.225884815274198 0.155682523936061 0.308213213363587 0.354423361607117
 0.237793437184287 0.344146246502948 0.117700835663451 0.22732103372742
 0.263538267437741 0.133994037769124 0.091015982918167 0.367309714380722
```



```

0.359174426753832 0.281533512444384 0.191021625928391 0.219461033259323
0.371983339466526 0.3512796731268 0.298580209240117 0.447936020775024
0.160438308161376 0.631208433750478 0.420060120050684 0.195238104695171
0.051118832992816 0.159003375870547 0.354123955360538 0.049380652521924
0.301744705672337 0.127112320459672 0.082920786835417 0.19281838475986
0.130182524243209 0.332360152782496 0.533160923855596 0.276782907418236
0.125265672564509 0.184320173657227 0.238789408490181 0.149666955002009
0.473603697641706 0.235878500572685 0.425250385266374 0.445829965130009
0.255853120078437 0.271987274130697 0.298711159306184 0.310382741777609
0.228936459666657 0.374245425300233 0.100039587285176 0.096572087547537
0.196098380129156 0.140612883827292 0.338936919946618 0.526244208385607
0.190627233723994 0.629487817359844 0.233333060530866 0.316852495355672
0.61336433904477 ]
-----

```

\*\*\*\*\*

first error with initial values of: ['b/a', 'c/a', 'a12', 'a13', 'a23',  
→ 'theta1', 'theta2', 'theta3']

\*\*\*\*\*

\*\*\*\*\*mean pixel deviation      0.2522039400422887      \*\*\*\*\*

\*\*\*\*\*

Fitting parameters: ['b/a', 'c/a', 'a12', 'a13', 'a23', 'theta1', 'theta2',  
→ 'theta3']

\*\*\*\*\*

With initial values [1. 1. 0. 0. 0. 0. 0. 0.]

code results 1

nb iterations 189

mesg Both actual and predicted relative reductions in the sum of squares  
are at most 0.000000

strain\_sol [ 9.999862096544356e-01 9.999941276097474e-01 -7.144700897145182e-  
→ 06

5.728542645493929e-05 1.242229382006964e-05 2.977984275077042e-05

-1.953472667313407e-03 6.919002586889341e-03]

\*\*\*\*\* End of Fitting - Final errors \*\*\*\*\*

\*\*\*\*\*mean pixel deviation      0.18215488925149526      \*\*\*\*\*

devstrain, lattice\_parameter\_direct\_strain [[-4.815998025634964e-06 5.  
→ 250997004746657e-06 -2.870284279887629e-05]

[ 5.250997004746657e-06 7.352907498721824e-06 -9.199263307200638e-06]

[-2.870284279887629e-05 -9.199263307200638e-06 -2.536909473086861e-06]] [ 5.

→ 657509728355469 5.657578574250457 5.657522951317707

90.0010541881893 90.00328909016345 89.99939828842365 ]

For comparison: a,b,c are rescaled with respect to the reference value of a =

```

→5.657500 Angstroms
lattice_parameter_direct_strain [ 5.6575                    5.657568845776604  5.
→6575132229395
  90.0010541881893   90.00328909016345   89.99939828842365 ]
devstrain1, lattice_parameter_direct_strain1 [[-4.815998025634964e-06  5.
→250997004746657e-06 -2.870284279887629e-05]
 [ 5.250997004746657e-06  7.352907498721824e-06 -9.199263307200638e-06]
 [-2.870284279887629e-05 -9.199263307200638e-06 -2.536909473086861e-06]] [ 5.
→6575                    5.657568845776604  5.6575132229395
  90.0010541881893   90.00328909016345   89.99939828842365 ]
new UBs matrix in q= UBs G (s for strain)
strain_direct [[ 1.719550237533340e-06  5.250997004746657e-06 -2.
→870284279887629e-05]
 [ 5.250997004746657e-06  1.388845576189013e-05 -9.199263307200638e-06]
 [-2.870284279887629e-05 -9.199263307200638e-06  3.998638790081444e-06]]
deviatoric strain [[-4.815998025634964e-06  5.250997004746657e-06 -2.
→870284279887629e-05]
 [ 5.250997004746657e-06  7.352907498721824e-06 -9.199263307200638e-06]
 [-2.870284279887629e-05 -9.199263307200638e-06 -2.536909473086861e-06]]
new UBs matrix in q= UBs G (s for strain)
strain_direct [[ 1.719550237533340e-06  5.250997004746657e-06 -2.
→870284279887629e-05]
 [ 5.250997004746657e-06  1.388845576189013e-05 -9.199263307200638e-06]
 [-2.870284279887629e-05 -9.199263307200638e-06  3.998638790081444e-06]]
deviatoric strain [[-4.815998025634964e-06  5.250997004746657e-06 -2.
→870284279887629e-05]
 [ 5.250997004746657e-06  7.352907498721824e-06 -9.199263307200638e-06]
 [-2.870284279887629e-05 -9.199263307200638e-06 -2.536909473086861e-06]]
For comparison: a,b,c are rescaled with respect to the reference value of a =_
→5.657500 Angstroms
lattice_parameter_direct_strain [ 5.6575                    5.657568845776604  5.
→6575132229395
  90.0010541881893   90.00328909016345   89.99939828842365 ]
final lattice_parameters [ 5.6575                    5.657568845776604  5.
→6575132229395
  90.0010541881893   90.00328909016345   89.99939828842365 ]
UB and strain refinement completed
True it is an OrientMatrix object
Orientation <LaueTools.indexingSpotsSet.OrientMatrix object at 0x7f7c84338898>
matrix [[-0.071478224945242 -0.995814588672313 -0.056724144676328]
 [ 0.720639310822985 -0.012262885888099 -0.693156594892675]
 [ 0.689613233174427 -0.090416539541802  0.718545890295648]]
*nb of selected spots in AssignHKL*** 82
UBOrientMatrix [[-0.071478224945242 -0.995814588672313 -0.056724144676328]
 [ 0.720639310822985 -0.012262885888099 -0.693156594892675]
 [ 0.689613233174427 -0.090416539541802  0.718545890295648]]
For angular tolerance 0.20 deg
Nb of pairs found / nb total of expected spots: 81/147
Matching Rate : 55.10
Nb missing reflections: 66

grain #0 : 81 links to simulated spots have been found
GoodRefinement condition is True
nb_updates 81 compared to 6

```

```

refining grain #0 step ----2

bestUB <LaueTools.indexingSpotsSet.OrientMatrix object at 0x7f7c954f5e80>
True it is an OrientMatrix object
Orientation <LaueTools.indexingSpotsSet.OrientMatrix object at 0x7f7c954f5e80>
matrix [[-0.071391558888759 -0.995830258190095 -0.056792096214888]
 [ 0.720671448854468 -0.012140732995851 -0.693170444701969]
 [ 0.689588625514858 -0.090412962995124 0.718535332243983]]
*nb of selected spots in AssignHKL*** 82
UBOrientMatrix [[-0.071391558888759 -0.995830258190095 -0.056792096214888]
 [ 0.720671448854468 -0.012140732995851 -0.693170444701969]
 [ 0.689588625514858 -0.090412962995124 0.718535332243983]]
For angular tolerance 0.10 deg
Nb of pairs found / nb total of expected spots: 81/147
Matching Rate : 55.10
Nb missing reflections: 66

```

```

grain #0 : 81 links to simulated spots have been found
*****mean pixel deviation      0.2522039400422887      *****
Initial residues [0.191356096913678 0.158479888122211 0.125984922997516 0.
↪007464331088751
0.220920883122328 0.065197410024489 0.405464259132319 0.079769518806149
0.309835172580193 0.024815180122634 0.146635771529913 0.197926454567734
0.253815574123594 0.241517985294699 0.301875442673439 0.217498144625921
0.186026257638361 0.152430964466482 0.022468745875909 0.372498387808433
0.225884815274198 0.155682523936061 0.308213213363587 0.354423361607117
0.237793437184287 0.344146246502948 0.117700835663451 0.22732103372742
0.263538267437741 0.133994037769124 0.091015982918167 0.367309714380722
0.359174426753832 0.281533512444384 0.191021625928391 0.219461033259323
0.371983339466526 0.3512796731268 0.298580209240117 0.447936020775024
0.160438308161376 0.631208433750478 0.420060120050684 0.195238104695171
0.051118832992816 0.159003375870547 0.354123955360538 0.049380652521924
0.301744705672337 0.127112320459672 0.082920786835417 0.19281838475986
0.130182524243209 0.332360152782496 0.533160923855596 0.276782907418236
0.125265672564509 0.184320173657227 0.238789408490181 0.149666955002009
0.473603697641706 0.235878500572685 0.425250385266374 0.445829965130009
0.255853120078437 0.271987274130697 0.298711159306184 0.310382741777609
0.228936459666657 0.374245425300233 0.100039587285176 0.096572087547537
0.196098380129156 0.140612883827292 0.338936919946618 0.526244208385607
0.190627233723994 0.629487817359844 0.233333060530866 0.316852495355672
0.61336433904477 ]

```

```

*****
first error with initial values of: ['b/a', 'c/a', 'a12', 'a13', 'a23', ↪
↪'thetal', 'theta2', 'theta3']

```

```

*****
*****mean pixel deviation      0.2522039400422887      *****

```

\*\*\*\*\*

Fitting parameters: ['b/a', 'c/a', 'a12', 'a13', 'a23', 'theta1', 'theta2',  
↳ 'theta3']

\*\*\*\*\*

With initial values [1. 1. 0. 0. 0. 0. 0. 0.]

code results 1

nb iterations 189

mesg Both actual and predicted relative reductions in the sum of squares  
are at most 0.000000

strain\_sol [ 9.999862096544356e-01 9.999941276097474e-01 -7.144700897145182e-  
↳ 06  
5.728542645493929e-05 1.242229382006964e-05 2.977984275077042e-05  
-1.953472667313407e-03 6.919002586889341e-03]

\*\*\*\*\* End of Fitting - Final errors \*\*\*\*\*

\*\*\*\*\*mean pixel deviation 0.18215488925149526 \*\*\*\*\*

devstrain, lattice\_parameter\_direct\_strain [[-4.815998025634964e-06 5.  
↳ 250997004746657e-06 -2.870284279887629e-05]  
[ 5.250997004746657e-06 7.352907498721824e-06 -9.199263307200638e-06]  
[-2.870284279887629e-05 -9.199263307200638e-06 -2.536909473086861e-06]] [ 5.  
↳ 657509728355469 5.657578574250457 5.657522951317707  
90.0010541881893 90.00328909016345 89.99939828842365 ]

For comparison: a,b,c are rescaled with respect to the reference value of a =  
↳ 5.657500 Angstroms

lattice\_parameter\_direct\_strain [ 5.6575 5.657568845776604 5.  
↳ 6575132229395  
90.0010541881893 90.00328909016345 89.99939828842365 ]

devstrain1, lattice\_parameter\_direct\_strain1 [[-4.815998025634964e-06 5.  
↳ 250997004746657e-06 -2.870284279887629e-05]  
[ 5.250997004746657e-06 7.352907498721824e-06 -9.199263307200638e-06]  
[-2.870284279887629e-05 -9.199263307200638e-06 -2.536909473086861e-06]] [ 5.  
↳ 6575 5.657568845776604 5.6575132229395  
90.0010541881893 90.00328909016345 89.99939828842365 ]

new UBs matrix in q= UBs G (s for strain)

strain\_direct [[ 1.719550237533340e-06 5.250997004746657e-06 -2.  
↳ 870284279887629e-05]  
[ 5.250997004746657e-06 1.388845576189013e-05 -9.199263307200638e-06]  
[-2.870284279887629e-05 -9.199263307200638e-06 3.998638790081444e-06]]  
deviatoric strain [[-4.815998025634964e-06 5.250997004746657e-06 -2.  
↳ 870284279887629e-05]  
[ 5.250997004746657e-06 7.352907498721824e-06 -9.199263307200638e-06]  
[-2.870284279887629e-05 -9.199263307200638e-06 -2.536909473086861e-06]]

new UBs matrix in q= UBs G (s for strain)

strain\_direct [[ 1.719550237533340e-06 5.250997004746657e-06 -2.  
↳ 870284279887629e-05]  
[ 5.250997004746657e-06 1.388845576189013e-05 -9.199263307200638e-06]  
[-2.870284279887629e-05 -9.199263307200638e-06 3.998638790081444e-06]]

```

deviatoric strain [[-4.815998025634964e-06  5.250997004746657e-06 -2.
↳870284279887629e-05]
 [ 5.250997004746657e-06  7.352907498721824e-06 -9.199263307200638e-06]
 [-2.870284279887629e-05 -9.199263307200638e-06 -2.536909473086861e-06]]
For comparison: a,b,c are rescaled with respect to the reference value of a =_
↳5.657500 Angstroms
lattice_parameter_direct_strain [ 5.6575                                5.657568845776604  5.
↳6575132229395
 90.0010541881893  90.00328909016345  89.99939828842365 ]
final lattice_parameters [ 5.6575                                5.657568845776604  5.
↳6575132229395
 90.0010541881893  90.00328909016345  89.99939828842365 ]
UB and strain refinement completed
True it is an OrientMatrix object
Orientation <LaueTools.indexingSpotsSet.OrientMatrix object at 0x7f7c94485da0>
matrix [[-0.071478224945242 -0.995814588672313 -0.056724144676328]
 [ 0.720639310822985 -0.012262885888099 -0.693156594892675]
 [ 0.689613233174427 -0.090416539541802  0.718545890295648]]
*nb of selected spots in AssignHKL*** 81
UBOrientMatrix [[-0.071478224945242 -0.995814588672313 -0.056724144676328]
 [ 0.720639310822985 -0.012262885888099 -0.693156594892675]
 [ 0.689613233174427 -0.090416539541802  0.718545890295648]]
For angular tolerance 0.10 deg
Nb of pairs found / nb total of expected spots: 81/147
Matching Rate : 55.10
Nb missing reflections: 66

grain #0 : 81 links to simulated spots have been found
GoodRefinement condition is True
nb_updates 81 compared to 6

-----
indexing completed for grain #0 with matching rate 55.10
-----

writing fit file -----
for grainindex= 0
self.dict_grain_matrix[grain_index] [[-0.071478224945242 -0.995814588672313 -
↳0.056724144676328]
 [ 0.720639310822985 -0.012262885888099 -0.693156594892675]
 [ 0.689613233174427 -0.090416539541802  0.718545890295648]]
self.refinedUBmatrix [[-0.071478224945242 -0.995814588672313 -0.
↳056724144676328]
 [ 0.720639310822985 -0.012262885888099 -0.693156594892675]
 [ 0.689613233174427 -0.090416539541802  0.718545890295648]]
new UBs matrix in q= UBs G (s for strain)
strain_direct [[ 1.719550237533340e-06  5.250997004746657e-06 -2.
↳870284279887629e-05]
 [ 5.250997004746657e-06  1.388845576189013e-05 -9.199263307200638e-06]
 [-2.870284279887629e-05 -9.199263307200638e-06  3.998638790081444e-06]]
deviatoric strain [[-4.815998025634964e-06  5.250997004746657e-06 -2.
↳870284279887629e-05]
 [ 5.250997004746657e-06  7.352907498721824e-06 -9.199263307200638e-06]
 [-2.870284279887629e-05 -9.199263307200638e-06 -2.536909473086861e-06]]

```

```

new UBs matrix in q= UBs G (s for strain)
strain_direct [[ 1.719550237533340e-06  5.250997004746657e-06 -2.
↪870284279887629e-05]
[ 5.250997004746657e-06  1.388845576189013e-05 -9.199263307200638e-06]
[-2.870284279887629e-05 -9.199263307200638e-06  3.998638790081444e-06]]
deviatoric strain [[-4.815998025634964e-06  5.250997004746657e-06 -2.
↪870284279887629e-05]
[ 5.250997004746657e-06  7.352907498721824e-06 -9.199263307200638e-06]
[-2.870284279887629e-05 -9.199263307200638e-06 -2.536909473086861e-06]]
For comparison: a,b,c are rescaled with respect to the reference value of a =_
↪5.657500 Angstroms
lattice_parameter_direct_strain [ 5.6575                                5.657568845776604  5.
↪6575132229395
90.0010541881893  90.00328909016345  89.99939828842365 ]
final lattice_parameters [ 5.6575                                5.657568845776604  5.
↪6575132229395
90.0010541881893  90.00328909016345  89.99939828842365 ]
File : Ge_blanc_0000Notebook_g0.fit written in /home/micha/LaueToolsPy3/
↪LaueTools/notebooks
Experimental experimental spots indices which are not indexed []
Missing reflections grainindex is -100 for indexed grainindex 0
within angular tolerance 0.500

```

Remaining nb of spots to index for grain #1 : 1

```

81 spots have been indexed over 82
indexing rate is --- : 98.8 percents
indexation of ../LaueImages/Ge_blanc_0000Notebook.cor is completed
for the 1 grain(s) that has(ve) been indexed as requested
Leaving Index and Refine procedures...
Saving unindexed fit file: ../LaueImages/Ge_blanc_0000Notebook_unindexed.cor
File : ../LaueImages/Ge_blanc_0000Notebook_g0.fit written in /home/micha/
↪LaueToolsPy3/LaueTools/notebooks

```

Indexation time 2.487 second(s)

Spots properties of the 10 first spots that have been indexed (sorted by intensity)

```

#spot 2theta chi X, Y intensity h k l energy
[[ 0.000000000000000e+00  5.842691500000000e+01  2.013003500000000e+01
  6.231300000000000e+02  1.657730000000000e+03  2.979938000000000e+04
  4.000000000000000e+00  2.000000000000000e+00  2.000000000000000e+00
  1.099874517758171e+01]
[ 1.000000000000000e+00  5.763467200000000e+01 -1.841552300000000e+01
  1.244330000000000e+03  1.662150000000000e+03  2.242563000000000e+04
  2.000000000000000e+00  2.000000000000000e+00  4.000000000000000e+00
  1.113626245226060e+01]
[ 2.000000000000000e+00  8.091984600000000e+01  6.615810000000000e-01
  9.330400000000000e+02  1.215440000000000e+03  2.219754000000000e+04
  3.000000000000000e+00  3.000000000000000e+00  3.000000000000000e+00
  8.773642495456929e+00]
[ 3.000000000000000e+00  1.168117220000000e+02  2.128975200000000e+01
  5.852300000000000e+02  5.888000000000000e+02  9.528830000000000e+03
  4.000000000000000e+00  6.000000000000000e+00  2.000000000000000e+00
  9.626187155122841e+00]

```

(continues on next page)

(continued from previous page)

```
[ 4.000000000000000e+00  1.159585970000000e+02 -2.073868400000000e+01
 1.276610000000000e+03  6.003000000000000e+02  9.153969999999999e+03
 2.000000000000000e+00  6.000000000000000e+00  4.000000000000000e+00
 9.671066779127555e+00]
[ 5.000000000000000e+00  1.097624120000000e+02  3.270820000000000e-01
 9.326600000000000e+02  7.500800000000000e+02  6.940080000000000e+03
 3.000000000000000e+00  5.000000000000000e+00  3.000000000000000e+00
 8.784305505382406e+00]
[ 6.000000000000000e+00  9.664626300000000e+01 -3.623969200000000e+01
 1.596620000000000e+03  9.353600000000000e+02  6.136730000000000e+03
 1.000000000000000e+00  5.000000000000000e+00  5.000000000000000e+00
 1.047621498150968e+01]
[ 7.000000000000000e+00  9.807526100000000e+01  3.749168800000000e+01
 2.523600000000000e+02  9.206000000000000e+02  5.635550000000000e+03
 5.000000000000000e+00  5.000000000000000e+00  1.000000000000000e+00
 1.036136430057910e+01]
[ 8.000000000000000e+00  6.421410100000000e+01 -1.397936600000000e+01
 1.168360000000000e+03  1.512820000000000e+03  5.570320000000000e+03
 3.000000000000000e+00  3.000000000000000e+00  5.000000000000000e+00
 1.351813316448898e+01]
[ 9.000000000000000e+00  9.620103100000000e+01 -4.831242900000000e+01
 1.945880000000000e+03  9.142200000000000e+02  5.317320000000000e+03
 0.000000000000000e+00  4.000000000000000e+00  4.000000000000000e+00
 8.328162135695869e+00]]
```

DataSet is an object with many attributes and methods related to spots properties (indexed or not, belonging to grains counted from zero). By press Tab key after having typed DataSet. can show you infos about spots

```
array([[ 1.767565178965975e-01, -2.842461599074922e-17,
        -2.842461599074922e-17],
       [ 0.000000000000000e+00,  1.767565178965975e-01,
        -1.082321519352500e-17],
       [ 0.000000000000000e+00,  0.000000000000000e+00,
        1.767565178965975e-01]])
```

## 6.2 Indexation of spots data set.

Using Class spotsSet and play with spots considered for indexation and refinement

### 6.2.1 This Notebook is a part of Tutorials on LaueTools Suite. Author:J.-S. Micha Date: July 2019

```
/home/micha/anaconda3/lib/python3.6/site-packages/h5py/___init___py:36:
↳FutureWarning: Conversion of the second argument of issubdtype from float
↳to np.floating is deprecated. In future, it will be treated as np.float64
↳== np.dtype(float).type.
from ._conv import register_converters as _register_converters

module Image / PIL is not installed
LaueToolsProjectFolder /home/micha/LaueToolsPy3/LaueTools
Cython compiled module 'gaussian2D' for fast computation is not installed!
module Image / PIL is not installed
Entering CrystalParameters **--*****
```

Cython compiled module for fast computation of Laue spots is not installed!  
 Cython compiled 'angulardist' module for fast computation of angular distance\_  
 ↳ is not installed!  
 Using default module  
 Cython compiled module for fast computation of angular distance is not\_  
 ↳ installed!  
 module Image / PIL is not installed  
 refinement from guessed solutions with two materials (see script IndexingTwinsSeries)

**Let's take a simple example of a single Laue Pattern. From the peak search we get 83 spots**

```
reading geometry calibration file
calib = [ 6.91790e+01  1.05081e+03  1.11559e+03  1.04000e-01 -2.73000e-01
         8.05700e-02  2.04800e+03  2.04800e+03]
matrix = [-0.211596  0.092178 -0.973001 -0.77574    0.589743  0.224568  0.594521
         0.802313 -0.053281]
```

Compute scattering angles from spots pixel positions and detector geometry. Write a .cor file from .dat including these new infos

```
nb of spots and columns in .dat file (83, 3)
file ../Examples/Ge/Ge0001.dat
containing 83 peaks
(2theta chi X Y I) written in ../Examples/Ge/Ge0001.cor
```

Create an instance of the class spotset. Initialize spots properties to data contained in .cor file

```
CCDcalib in readfile_cor {'dd': 69.179, 'xcen': 1050.81, 'ycen': 1115.59, 'xbet': 0.
↳ 104, 'xgam': -0.273, 'xpixelsize': 0.08057, 'ypixelsize': 0.08057, 'CCDLabel':
↳ 'sCMOS', 'framedim': [2048.0, 2048.0], 'detectordiameter': 165.00736, 'kf_direction
↳ ': 'Z>0', 'pixelsize': 0.08057}
CCD Detector parameters read from .cor file
CCDcalibdict {'dd': 69.179, 'xcen': 1050.81, 'ycen': 1115.59, 'xbet': 0.104, 'xgam': -
↳ 0.273, 'xpixelsize': 0.08057, 'ypixelsize': 0.08057, 'CCDLabel': 'sCMOS', 'framedim
↳ ': [2048.0, 2048.0], 'detectordiameter': 165.00736, 'kf_direction': 'Z>0',
↳ 'pixelsize': 0.08057}
```

**True**

Class methods and attributes rely on a dictionary of spots properties. key = experimental spot index, val = spots properties

```
[[0, 78.215821, 1.638153, 1027.11, 1293.28, 70931.27, 0],
 [1, 64.329767, -20.824155, 1379.17, 1553.58, 51933.84, 0],
 [2, 68.680451, -15.358122, 1288.11, 1460.16, 22795.07, 0],
 [3, 105.61498, 8.176187, 926.22, 872.06, 19489.69, 0],
 [4, 103.859791, 27.866566, 595.46, 876.44, 19058.79, 0],
 [5, 120.59561, -8.92066, 1183.27, 598.92, 17182.88, 0],
 [6, 60.359458, 26.483191, 626.12, 1661.28, 15825.39, 0],
 [7, 56.269853, 12.967153, 856.14, 1702.52, 15486.2, 0],
 [8, 82.072076, -35.89243, 1672.67, 1258.62, 13318.81, 0],
 [9, 83.349535, -27.458061, 1497.4, 1224.7, 13145.99, 0]]
```



```
array([[ 0.0000000e+00,  7.8215821e+01,  1.6381530e+00,  1.0271100e+03,
         1.2932800e+03,  7.0931270e+04],
       [ 1.0000000e+00,  6.4329767e+01, -2.0824155e+01,  1.3791700e+03,
         1.5535800e+03,  5.1933840e+04],
       [ 2.0000000e+00,  6.8680451e+01, -1.5358122e+01,  1.2881100e+03,
         1.4601600e+03,  2.2795070e+04]])
```

Normally we read all spots data from a **.cor** file

```
CCDcalib in readfile_cor {'dd': 69.179, 'xcen': 1050.81, 'ycen': 1115.59, 'xbet': 0.
↪104, 'xgam': -0.273, 'xpixelsize': 0.08057, 'ypixelsize': 0.08057, 'CCDLabel':
↪'sCMOS', 'framedim': [2048.0, 2048.0], 'detectordiameter': 165.00736, 'kf_direction
↪': 'Z>0', 'pixelsize': 0.08057}
CCD Detector parameters read from .cor file
CCDcalibdict {'dd': 69.179, 'xcen': 1050.81, 'ycen': 1115.59, 'xbet': 0.104, 'xgam': -
↪0.273, 'xpixelsize': 0.08057, 'ypixelsize': 0.08057, 'CCDLabel': 'sCMOS', 'framedim
↪': [2048.0, 2048.0], 'detectordiameter': 165.00736, 'kf_direction': 'Z>0',
↪'pixelsize': 0.08057}
```

5

but we can import a custom list of spots. For example, starting from spots a the previous **.cor** file

```
CCDcalib in readfile_cor {'dd': 69.179, 'xcen': 1050.81, 'ycen': 1115.59, 'xbet': 0.
↪104, 'xgam': -0.273, 'xpixelsize': 0.08057, 'ypixelsize': 0.08057, 'CCDLabel':
↪'sCMOS', 'framedim': [2048.0, 2048.0], 'detectordiameter': 165.00736, 'kf_direction
↪': 'Z>0', 'pixelsize': 0.08057}
CCD Detector parameters read from .cor file
```

```
array([[ 7.8215821e+01,  1.6381530e+00,  1.0271100e+03,
         1.2932800e+03,  7.0931270e+04],
       [ 6.4329767e+01, -2.0824155e+01,  1.3791700e+03,
         1.5535800e+03,  5.1933840e+04],
       [ 6.8680451e+01, -1.5358122e+01,  1.2881100e+03,
         1.4601600e+03,  2.2795070e+04],
       [ 1.0561498e+02,  8.1761870e+00,  9.2622000e+02,
         8.7206000e+02,  1.9489690e+04],
       [ 1.03859791e+02,  2.7866566e+01,  5.9546000e+02,
         8.7644000e+02,  1.9058790e+04],
       [ 1.2059561e+02, -8.9206600e+00,  1.1832700e+03,
         5.9892000e+02,  1.7182880e+04],
       [ 6.0359458e+01,  2.6483191e+01,  6.2612000e+02,
         1.6612800e+03,  1.5825390e+04]])
```

(5, 83)

spots data must be imported as an array of 5 elements: 2theta, chi, Intensity, pixelX, pixelY

```
(array([[ 6.0359458e+01,  7.8215821e+01,  6.8680451e+01,
         1.08452917e+02,  8.3349535e+01,  8.2072076e+01,
```

(continues on next page)

(continued from previous page)

```

      8.17712570e+01,  9.17982210e+01,  1.20595610e+02,
      6.43297670e+01,  5.62698530e+01,  1.14942090e+02],
[ 2.64831910e+01,  1.63815300e+00, -1.53581220e+01,
  3.77494610e+01, -2.74580610e+01, -3.58924300e+01,
  3.03824700e+01, -8.30994100e+00, -8.92066000e+00,
 -2.08241550e+01,  1.29671530e+01,  1.15295700e+01],
[ 1.58253900e+04,  7.09312700e+04,  2.27950700e+04,
  4.40061000e+03,  1.31459900e+04,  1.33188100e+04,
  6.13787000e+03,  1.17999300e+04,  1.71828800e+04,
  5.19338400e+04,  1.54862000e+04,  1.00105200e+04]],
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11]))

```

## 6.2.2 core function to index a set of spots

by default `DataSet.getUnIndexedSpotsallData()` is called

if `use_file = 0`, then current non indexed exp. spots will be considered for indexation

if `use_file = 1`, reimport data from file and reset also spots properties dictionary (i.e. with status unindexed)

```

self.pixelsize in IndexSpotsSet 0.08057
ResolutionAngstromLUT in IndexSpotsSet False

```

```

Remaining nb of spots to index for grain #0 : 12

```

```

**

```

```

start to index grain #0 of Material: Ge

```

```

**

```

```

providing new set of matrices Using Angles LUT template matching

```

```

nbspot 12

```

```

NBMAXPROBED 6

```

```

nbspot 12

```

```

set_central_spots_hkl None

```

```

Computing LUT from material data

```

```

Compute LUT for indexing Ge spots in LauePattern

```

```

Build angles LUT with latticeparameters

```

```

[ 5.657499999999999 5.657499999999999 5.657499999999999
 90.                90.                90.                ]

```

```

and n=3

```

```

MaxRadiusHKL False

```

```

cubicSymmetry True

```

```

Central set of exp. spotDistances from spot_index_central_list probed

```

```

self.absolute_index [ 0 1 2 3 4 5 6 7 8 9 10 11]

```

```

spot_index_central_list [0]

```

```

[0]

```

```

LUT is not None when entering getOrientMatrices()

```

```

set_central_spots_hkl None

```

```

set_central_spots_hkl_list [None]

```

```

cubicSymmetry True

```

```

LUT_tol_angle 0.5

```

```

---***-----*

```

```

Calculating all possible matrices from exp spot #0 and the 5 other(s)
hkl in getOrientMatrices None <class 'NoneType'>
using LUTcubic
LUTcubic is None for k_centspot_index 0 in getOrientMatrices()
hkl1 in matrices_from_onespot_hkl() [[1 0 0]
[1 1 0]
[1 1 1]
[2 1 0]
[2 1 1]
[2 2 1]
[3 1 0]
[3 1 1]
[3 2 1]
[3 2 2]
[3 3 1]
[3 3 2]]
Computing hkl2 list for specific or cubic LUT in matrices_from_onespot_hkl()
Calculating LUT in PlanePairs_from2sets()
Looking up planes pairs in LUT from exp. spots (0, 1):
Looking up planes pairs in LUT from exp. spots (0, 2):
Looking up planes pairs in LUT from exp. spots (0, 3):
Looking up planes pairs in LUT from exp. spots (0, 4):
Looking up planes pairs in LUT from exp. spots (0, 5):
calculating matching rates of solutions for exp. spots [0, 1]
calculating matching rates of solutions for exp. spots [0, 2]
calculating matching rates of solutions for exp. spots [0, 3]
calculating matching rates of solutions for exp. spots [0, 4]

return best matrix and matching scores for the one central_spot

-----
results:
matrix:                                matching results
[-0.211852735694566  0.092255643652867 -0.972937466948891]      res: [20.0,
↪162.0] 0.014 12.35
[-0.775856536468367  0.58951816141498   0.22475536073965 ]      spot_
↪indices [0 1]
[ 0.594300563948835  0.802473664571131 -0.053318452339475]      planes [[1.
↪0, 3.0, 2.0], [1.0, 1.0, 1.0]]

Number of matrices found (nb_sol): 1
set_central_spots_hkl in FindOrientMatrices None

-----
results:
matrix:                                matching results
[-0.211852735694566  0.092255643652867 -0.972937466948891]      res: [20.0,
↪162.0] 0.014 12.35
[-0.775856536468367  0.58951816141498   0.22475536073965 ]      spot_
↪indices [0 1]
[ 0.594300563948835  0.802473664571131 -0.053318452339475]      planes [[1.
↪0, 3.0, 2.0], [1.0, 1.0, 1.0]]

```

```

Nb of potential orientation matrice(s) UB found: 1
[array([[ -0.211852735694566,  0.092255643652867, -0.972937466948891],
        [ -0.775856536468367,  0.58951816141498 ,  0.22475536073965 ],
        [ 0.594300563948835,  0.802473664571131, -0.053318452339475]])]
Nb of potential UBs 1

Working with a new stack of orientation matrices
MATCHINGRATE_THRESHOLD_IAL= 100.0
has not been reached! All potential solutions have been calculated
taking the first one only.
bestUB object <LaueTools.indexingSpotsSet.OrientMatrix object at 0x7fb8ec7ddc50>

-----refining grain orientation and strain #0-----

refining grain #0 step ----0

bestUB <LaueTools.indexingSpotsSet.OrientMatrix object at 0x7fb8ec7ddc50>
True it is an OrientMatrix object
Orientation <LaueTools.indexingSpotsSet.OrientMatrix object at 0x7fb8ec7ddc50>
matrix [[ -0.211852735694566  0.092255643652867 -0.972937466948891]
        [ -0.775856536468367  0.58951816141498  0.22475536073965 ]
        [ 0.594300563948835  0.802473664571131 -0.053318452339475]]
*nb of selected spots in AssignHKL** 12
UBOrientMatrix [[ -0.211852735694566  0.092255643652867 -0.972937466948891]
                [ -0.775856536468367  0.58951816141498  0.22475536073965 ]
                [ 0.594300563948835  0.802473664571131 -0.053318452339475]]
For angular tolerance 0.50 deg
Nb of pairs found / nb total of expected spots: 12/176
Matching Rate : 6.82
Nb missing reflections: 164

grain #0 : 12 links to simulated spots have been found
*****mean pixel deviation      0.560750282710606      *****
Initial residues [0.053680370172309 0.013739858524874 0.921977335411896 0.
↪403270956234836
0.919825854310187 0.785969463406447 0.565019172757509 1.127873079813964
0.363514793614926 0.412635402450867 0.711008521607465 0.450488584221994]

-----

*****
first error with initial values of: ['b/a', 'c/a', 'a12', 'a13', 'a23',
↪'theta1', 'theta2', 'theta3']

*****

*****mean pixel deviation      0.560750282710606      *****

*****

```

Fitting parameters: ['b/a', 'c/a', 'a12', 'a13', 'a23', 'theta1', 'theta2',  
 ↪ 'theta3']

\*\*\*\*\*

With initial values [1. 1. 0. 0. 0. 0. 0.]

code results 1

nb iterations 1767

mesg Both actual and predicted relative reductions in the sum of squares  
 are at most 0.000000

strain\_sol [ 1.001128981010799e+00 9.993806401299155e-01 8.449040381845989e-  
 ↪ 04

-8.486913595751131e-04 3.520626401662759e-04 -2.714612741167435e-02

3.054889720130747e-02 5.311773668297186e-02]

\*\*\*\*\* End of Fitting - Final errors \*\*\*\*\*

\*\*\*\*\*mean pixel deviation 0.3158807195732847 \*\*\*\*\*

devstrain, lattice\_parameter\_direct\_strain [[ 0.000159671589578 -0.

↪ 000413843247724 0.00039782267455 ]

[-0.000413843247724 -0.00095830941256 -0.000151781897557]

[ 0.00039782267455 -0.000151781897557 0.000798637822982]] [ 5.

↪ 657424223234738 5.651101185787196 5.661104877237695

90.01741959362538 89.9544419036642 90.04747664598754 ]

For comparison: a,b,c are rescaled with respect to the reference value of a =  
 ↪ 5.657500 Angstroms

lattice\_parameter\_direct\_strain [ 5.657499999999999 5.651176877860324 5.

↪ 661180703302433

90.01741959362538 89.9544419036642 90.04747664598754 ]

devstrain1, lattice\_parameter\_direct\_strain1 [[ 0.000159671589578 -0.

↪ 000413843247724 0.00039782267455 ]

[-0.000413843247724 -0.00095830941256 -0.000151781897557]

[ 0.00039782267455 -0.000151781897557 0.000798637822982]] [ 5.

↪ 657499999999999 5.651176877860324 5.661180703302433

90.01741959362538 89.9544419036642 90.04747664598754 ]

new UBs matrix in q= UBs G (s for strain)

strain\_direct [[-1.339403716515974e-05 -4.138432477238585e-04 3.

↪ 978226745502020e-04]

[-4.138432477238585e-04 -1.131375039302607e-03 -1.517818975573709e-04]

[ 3.978226745502020e-04 -1.517818975573709e-04 6.255721962393768e-04]]

deviatoric strain [[ 0.000159671589578 -0.000413843247724 0.00039782267455 ]

[-0.000413843247724 -0.00095830941256 -0.000151781897557]

[ 0.00039782267455 -0.000151781897557 0.000798637822982]]

new UBs matrix in q= UBs G (s for strain)

strain\_direct [[-1.339403716515974e-05 -4.138432477238585e-04 3.

↪ 978226745502020e-04]

[-4.138432477238585e-04 -1.131375039302607e-03 -1.517818975573709e-04]

[ 3.978226745502020e-04 -1.517818975573709e-04 6.255721962393768e-04]]

deviatoric strain [[ 0.000159671589578 -0.000413843247724 0.00039782267455 ]

[-0.000413843247724 -0.00095830941256 -0.000151781897557]

[ 0.00039782267455 -0.000151781897557 0.000798637822982]]

For comparison: a,b,c are rescaled with respect to the reference value of a =  
 ↪

```
→5.657500 Angstroms
lattice_parameter_direct_strain [ 5.657499999999999 5.651176877860324 5.
→661180703302433
  90.01741959362538 89.9544419036642 90.04747664598754 ]
final lattice_parameters [ 5.657499999999999 5.651176877860324 5.
→661180703302433
  90.01741959362538 89.9544419036642 90.04747664598754 ]
UB and strain refinement completed
True it is an OrientMatrix object
Orientation <LaueTools.indexingSpotsSet.OrientMatrix object at 0x7fb8c98baa20>
matrix [[-0.211415207301911 0.091252946262469 -0.97230572627369 ]
 [-0.77573594328912 0.590041596352251 0.224552051799525]
 [ 0.594613709497768 0.803610914885283 -0.054088075690982]]
*nb of selected spots in AssignHKL*** 12
UBOrientMatrix [[-0.211415207301911 0.091252946262469 -0.97230572627369 ]
 [-0.77573594328912 0.590041596352251 0.224552051799525]
 [ 0.594613709497768 0.803610914885283 -0.054088075690982]]
For angular tolerance 0.50 deg
Nb of pairs found / nb total of expected spots: 12/177
Matching Rate : 6.78
Nb missing reflections: 165

grain #0 : 12 links to simulated spots have been found
GoodRefinement condition is True
nb_updates 12 compared to 6

refining grain #0 step ----1

bestUB <LaueTools.indexingSpotsSet.OrientMatrix object at 0x7fb8ec7ddc50>
True it is an OrientMatrix object
Orientation <LaueTools.indexingSpotsSet.OrientMatrix object at 0x7fb8ec7ddc50>
matrix [[-0.211852735694566 0.092255643652867 -0.972937466948891]
 [-0.775856536468367 0.58951816141498 0.22475536073965 ]
 [ 0.594300563948835 0.802473664571131 -0.053318452339475]]
*nb of selected spots in AssignHKL*** 12
UBOrientMatrix [[-0.211852735694566 0.092255643652867 -0.972937466948891]
 [-0.775856536468367 0.58951816141498 0.22475536073965 ]
 [ 0.594300563948835 0.802473664571131 -0.053318452339475]]
For angular tolerance 0.50 deg
Nb of pairs found / nb total of expected spots: 12/176
Matching Rate : 6.82
Nb missing reflections: 164

grain #0 : 12 links to simulated spots have been found
*****mean pixel deviation 0.560750282710606 *****
Initial residues [0.053680370172309 0.013739858524874 0.921977335411896 0.
→403270956234836
  0.919825854310187 0.785969463406447 0.565019172757509 1.127873079813964
  0.363514793614926 0.412635402450867 0.711008521607465 0.450488584221994]
-----
```

```

*****
first error with initial values of: ['b/a', 'c/a', 'a12', 'a13', 'a23',
↳ 'theta1', 'theta2', 'theta3']

*****

*****mean pixel deviation      0.560750282710606      *****

*****
Fitting parameters:  ['b/a', 'c/a', 'a12', 'a13', 'a23', 'theta1', 'theta2',
↳ 'theta3']

*****

With initial values [1. 1. 0. 0. 0. 0. 0. 0.]
code results 1
nb iterations 1767
mesg Both actual and predicted relative reductions in the sum of squares
are at most 0.000000
strain_sol [ 1.001128981010799e+00  9.993806401299155e-01  8.449040381845989e-
↳ 04
-8.486913595751131e-04  3.520626401662759e-04 -2.714612741167435e-02
3.054889720130747e-02  5.311773668297186e-02]

***** End of Fitting - Final errors *****

*****mean pixel deviation      0.3158807195732847      *****
devstrain, lattice_parameter_direct_strain [[ 0.000159671589578 -0.
↳ 000413843247724  0.00039782267455 ]
[-0.000413843247724 -0.00095830941256 -0.000151781897557]
[ 0.00039782267455 -0.000151781897557  0.000798637822982]] [ 5.
↳ 657424223234738  5.651101185787196  5.661104877237695
90.01741959362538  89.9544419036642  90.04747664598754 ]
For comparison: a,b,c are rescaled with respect to the reference value of a =
↳ 5.657500 Angstroms
lattice_parameter_direct_strain [ 5.657499999999999  5.651176877860324  5.
↳ 661180703302433
90.01741959362538  89.9544419036642  90.04747664598754 ]
devstrain1, lattice_parameter_direct_strain1 [[ 0.000159671589578 -0.
↳ 000413843247724  0.00039782267455 ]
[-0.000413843247724 -0.00095830941256 -0.000151781897557]
[ 0.00039782267455 -0.000151781897557  0.000798637822982]] [ 5.
↳ 657499999999999  5.651176877860324  5.661180703302433
90.01741959362538  89.9544419036642  90.04747664598754 ]
new UBs matrix in q= UBs G (s for strain)
strain_direct [[-1.339403716515974e-05 -4.138432477238585e-04  3.
↳ 978226745502020e-04]
[-4.138432477238585e-04 -1.131375039302607e-03 -1.517818975573709e-04]
[ 3.978226745502020e-04 -1.517818975573709e-04  6.255721962393768e-04]]
deviatoric strain [[ 0.000159671589578 -0.000413843247724  0.00039782267455 ]
↳ [-0.000413843247724 -0.00095830941256 -0.000151781897557]

```

```
[ 0.00039782267455 -0.000151781897557 0.000798637822982]]
new UBs matrix in q= UBs G (s for strain)
strain_direct [[-1.339403716515974e-05 -4.138432477238585e-04 3.
↪978226745502020e-04]
[-4.138432477238585e-04 -1.131375039302607e-03 -1.517818975573709e-04]
[ 3.978226745502020e-04 -1.517818975573709e-04 6.255721962393768e-04]]
deviatoric strain [[ 0.000159671589578 -0.000413843247724 0.00039782267455 ]
[-0.000413843247724 -0.00095830941256 -0.000151781897557]
[ 0.00039782267455 -0.000151781897557 0.000798637822982]]
For comparison: a,b,c are rescaled with respect to the reference value of a =_
↪5.657500 Angstroms
lattice_parameter_direct_strain [ 5.657499999999999 5.651176877860324 5.
↪661180703302433
90.01741959362538 89.9544419036642 90.04747664598754 ]
final lattice_parameters [ 5.657499999999999 5.651176877860324 5.
↪661180703302433
90.01741959362538 89.9544419036642 90.04747664598754 ]
UB and strain refinement completed
True it is an OrientMatrix object
Orientation <LaueTools.indexingSpotsSet.OrientMatrix object at 0x7fb8c98ba9b0>
matrix [[-0.211415207301911 0.091252946262469 -0.97230572627369 ]
[-0.77573594328912 0.590041596352251 0.224552051799525]
[ 0.594613709497768 0.803610914885283 -0.054088075690982]]
*nb of selected spots in AssignHKL*** 12
UBOrientMatrix [[-0.211415207301911 0.091252946262469 -0.97230572627369 ]
[-0.77573594328912 0.590041596352251 0.224552051799525]
[ 0.594613709497768 0.803610914885283 -0.054088075690982]]
For angular tolerance 0.50 deg
Nb of pairs found / nb total of expected spots: 12/177
Matching Rate : 6.78
Nb missing reflections: 165

grain #0 : 12 links to simulated spots have been found
GoodRefinement condition is True
nb_updates 12 compared to 6

refining grain #0 step ----2

bestUB <LaueTools.indexingSpotsSet.OrientMatrix object at 0x7fb8ec7ddc50>
True it is an OrientMatrix object
Orientation <LaueTools.indexingSpotsSet.OrientMatrix object at 0x7fb8ec7ddc50>
matrix [[-0.211852735694566 0.092255643652867 -0.972937466948891]
[-0.775856536468367 0.58951816141498 0.22475536073965 ]
[ 0.594300563948835 0.802473664571131 -0.053318452339475]]
*nb of selected spots in AssignHKL*** 12
UBOrientMatrix [[-0.211852735694566 0.092255643652867 -0.972937466948891]
[-0.775856536468367 0.58951816141498 0.22475536073965 ]
[ 0.594300563948835 0.802473664571131 -0.053318452339475]]
For angular tolerance 0.20 deg
Nb of pairs found / nb total of expected spots: 12/176
Matching Rate : 6.82
Nb missing reflections: 164
```



```

grain #0 : 12 links to simulated spots have been found
*****mean pixel deviation      0.560750282710606      *****
Initial residues [0.053680370172309 0.013739858524874 0.921977335411896 0.
↳403270956234836
0.919825854310187 0.785969463406447 0.565019172757509 1.127873079813964
0.363514793614926 0.412635402450867 0.711008521607465 0.450488584221994]
-----

*****
first error with initial values of: ['b/a', 'c/a', 'a12', 'a13', 'a23', '
↳'theta1', 'theta2', 'theta3']

*****

*****mean pixel deviation      0.560750282710606      *****

*****
Fitting parameters:  ['b/a', 'c/a', 'a12', 'a13', 'a23', 'theta1', 'theta2', '
↳'theta3']

*****

With initial values [1. 1. 0. 0. 0. 0. 0. 0.]
code results 1
nb iterations 1767
mesg Both actual and predicted relative reductions in the sum of squares
are at most 0.000000
strain_sol [ 1.001128981010799e+00  9.993806401299155e-01  8.449040381845989e-
↳04
-8.486913595751131e-04  3.520626401662759e-04 -2.714612741167435e-02
3.054889720130747e-02  5.311773668297186e-02]

***** End of Fitting - Final errors *****

*****mean pixel deviation      0.3158807195732847      *****
devstrain, lattice_parameter_direct_strain [[ 0.000159671589578 -0.
↳000413843247724 0.00039782267455 ]
[-0.000413843247724 -0.00095830941256 -0.000151781897557]
[ 0.00039782267455 -0.000151781897557 0.000798637822982]] [ 5.
↳657424223234738 5.651101185787196 5.661104877237695
90.01741959362538 89.9544419036642 90.04747664598754 ]
For comparison: a,b,c are rescaled with respect to the reference value of a =
↳5.657500 Angstroms
lattice_parameter_direct_strain [ 5.657499999999999 5.651176877860324 5.
↳661180703302433
90.01741959362538 89.9544419036642 90.04747664598754 ]
devstrain1, lattice_parameter_direct_strain1 [[ 0.000159671589578 -0.
↳000413843247724 0.00039782267455 ]
[-0.000413843247724 -0.00095830941256 -0.000151781897557]

```

```
[ 0.00039782267455 -0.000151781897557 0.000798637822982]] [ 5.
↪6574999999999999 5.651176877860324 5.661180703302433
90.01741959362538 89.9544419036642 90.04747664598754 ]
new UBs matrix in q= UBs G (s for strain)
strain_direct [[-1.339403716515974e-05 -4.138432477238585e-04 3.
↪978226745502020e-04]
[-4.138432477238585e-04 -1.131375039302607e-03 -1.517818975573709e-04]
[ 3.978226745502020e-04 -1.517818975573709e-04 6.255721962393768e-04]]
deviatoric strain [[ 0.000159671589578 -0.000413843247724 0.00039782267455 ]
[-0.000413843247724 -0.00095830941256 -0.000151781897557]
[ 0.00039782267455 -0.000151781897557 0.000798637822982]]
new UBs matrix in q= UBs G (s for strain)
strain_direct [[-1.339403716515974e-05 -4.138432477238585e-04 3.
↪978226745502020e-04]
[-4.138432477238585e-04 -1.131375039302607e-03 -1.517818975573709e-04]
[ 3.978226745502020e-04 -1.517818975573709e-04 6.255721962393768e-04]]
deviatoric strain [[ 0.000159671589578 -0.000413843247724 0.00039782267455 ]
[-0.000413843247724 -0.00095830941256 -0.000151781897557]
[ 0.00039782267455 -0.000151781897557 0.000798637822982]]
For comparison: a,b,c are rescaled with respect to the reference value of a = ↪
↪5.657500 Angstroms
lattice_parameter_direct_strain [ 5.657499999999999 5.651176877860324 5.
↪661180703302433
90.01741959362538 89.9544419036642 90.04747664598754 ]
final lattice_parameters [ 5.657499999999999 5.651176877860324 5.
↪661180703302433
90.01741959362538 89.9544419036642 90.04747664598754 ]
UB and strain refinement completed
True it is an OrientMatrix object
Orientation <LaueTools.indexingSpotsSet.OrientMatrix object at 0x7fb8cf573a90>
matrix [[-0.211415207301911 0.091252946262469 -0.97230572627369 ]
[-0.77573594328912 0.590041596352251 0.224552051799525]
[ 0.594613709497768 0.803610914885283 -0.054088075690982]]
*nb of selected spots in AssignHKL*** 12
UBOrientMatrix [[-0.211415207301911 0.091252946262469 -0.97230572627369 ]
[-0.77573594328912 0.590041596352251 0.224552051799525]
[ 0.594613709497768 0.803610914885283 -0.054088075690982]]
For angular tolerance 0.20 deg
Nb of pairs found / nb total of expected spots: 12/177
Matching Rate : 6.78
Nb missing reflections: 165

grain #0 : 12 links to simulated spots have been found
GoodRefinement condition is True
nb_updates 12 compared to 6
```

```
refining grain #0 step -----3
```

```
bestUB <LaueTools.indexingSpotsSet.OrientMatrix object at 0x7fb8ec7ddc50>
True it is an OrientMatrix object
Orientation <LaueTools.indexingSpotsSet.OrientMatrix object at 0x7fb8ec7ddc50>
matrix [[-0.211852735694566 0.092255643652867 -0.972937466948891]
[-0.775856536468367 0.58951816141498 0.22475536073965 ]
```

```

[ 0.594300563948835  0.802473664571131 -0.053318452339475]]
*nb of selected spots in AssignHKL*** 12
UBOrientMatrix [[-0.211852735694566  0.092255643652867 -0.972937466948891]
[-0.775856536468367  0.58951816141498  0.22475536073965 ]
[ 0.594300563948835  0.802473664571131 -0.053318452339475]]
For angular tolerance 0.20 deg
Nb of pairs found / nb total of expected spots: 12/176
Matching Rate : 6.82
Nb missing reflections: 164

grain #0 : 12 links to simulated spots have been found
*****mean pixel deviation      0.560750282710606      *****
Initial residues [0.053680370172309 0.013739858524874 0.921977335411896 0.
↪403270956234836
0.919825854310187 0.785969463406447 0.565019172757509 1.127873079813964
0.363514793614926 0.412635402450867 0.711008521607465 0.450488584221994]
-----

*****
first error with initial values of: ['b/a', 'c/a', 'a12', 'a13', 'a23', ↪
↪'theta1', 'theta2', 'theta3']

*****

*****mean pixel deviation      0.560750282710606      *****

*****
Fitting parameters: ['b/a', 'c/a', 'a12', 'a13', 'a23', 'theta1', 'theta2', ↪
↪'theta3']

*****

With initial values [1. 1. 0. 0. 0. 0. 0. 0.]
code results 1
nb iterations 1767
mesg Both actual and predicted relative reductions in the sum of squares
are at most 0.000000
strain_sol [ 1.001128981010799e+00  9.993806401299155e-01  8.449040381845989e-
↪04
-8.486913595751131e-04  3.520626401662759e-04 -2.714612741167435e-02
3.054889720130747e-02  5.311773668297186e-02]

***** End of Fitting - Final errors *****

*****mean pixel deviation      0.3158807195732847      *****
devstrain, lattice_parameter_direct_strain [[ 0.000159671589578 -0.
↪000413843247724 0.00039782267455 ]
[-0.000413843247724 -0.00095830941256 -0.000151781897557]
[ 0.00039782267455 -0.000151781897557 0.000798637822982]] [ 5.

```

```

→657424223234738 5.651101185787196 5.661104877237695
90.01741959362538 89.9544419036642 90.04747664598754 ]
For comparison: a,b,c are rescaled with respect to the reference value of a =
→5.657500 Angstroms
lattice_parameter_direct_strain [ 5.657499999999999 5.651176877860324 5.
→661180703302433
90.01741959362538 89.9544419036642 90.04747664598754 ]
devstrain1, lattice_parameter_direct_strain1 [[ 0.000159671589578 -0.
→000413843247724 0.00039782267455 ]
[-0.000413843247724 -0.00095830941256 -0.000151781897557]
[ 0.00039782267455 -0.000151781897557 0.000798637822982]] [ 5.
→657499999999999 5.651176877860324 5.661180703302433
90.01741959362538 89.9544419036642 90.04747664598754 ]
new UBs matrix in q= UBs G (s for strain)
strain_direct [[-1.339403716515974e-05 -4.138432477238585e-04 3.
→978226745502020e-04]
[-4.138432477238585e-04 -1.131375039302607e-03 -1.517818975573709e-04]
[ 3.978226745502020e-04 -1.517818975573709e-04 6.255721962393768e-04]]
deviatoric strain [[ 0.000159671589578 -0.000413843247724 0.00039782267455 ]
[-0.000413843247724 -0.00095830941256 -0.000151781897557]
[ 0.00039782267455 -0.000151781897557 0.000798637822982]]
new UBs matrix in q= UBs G (s for strain)
strain_direct [[-1.339403716515974e-05 -4.138432477238585e-04 3.
→978226745502020e-04]
[-4.138432477238585e-04 -1.131375039302607e-03 -1.517818975573709e-04]
[ 3.978226745502020e-04 -1.517818975573709e-04 6.255721962393768e-04]]
deviatoric strain [[ 0.000159671589578 -0.000413843247724 0.00039782267455 ]
[-0.000413843247724 -0.00095830941256 -0.000151781897557]
[ 0.00039782267455 -0.000151781897557 0.000798637822982]]
For comparison: a,b,c are rescaled with respect to the reference value of a =
→5.657500 Angstroms
lattice_parameter_direct_strain [ 5.657499999999999 5.651176877860324 5.
→661180703302433
90.01741959362538 89.9544419036642 90.04747664598754 ]
final lattice_parameters [ 5.657499999999999 5.651176877860324 5.
→661180703302433
90.01741959362538 89.9544419036642 90.04747664598754 ]
UB and strain refinement completed
True it is an OrientMatrix object
Orientation <LaueTools.indexingSpotsSet.OrientMatrix object at 0x7fb8ec7bc128>
matrix [[-0.211415207301911 0.091252946262469 -0.97230572627369 ]
[-0.77573594328912 0.590041596352251 0.224552051799525]
[ 0.594613709497768 0.803610914885283 -0.054088075690982]]
*nb of selected spots in AssignHKL*** 12
UBOrientMatrix [[-0.211415207301911 0.091252946262469 -0.97230572627369 ]
[-0.77573594328912 0.590041596352251 0.224552051799525]
[ 0.594613709497768 0.803610914885283 -0.054088075690982]]
For angular tolerance 0.20 deg
Nb of pairs found / nb total of expected spots: 12/177
Matching Rate : 6.78
Nb missing reflections: 165

grain #0 : 12 links to simulated spots have been found
GoodRefinement condition is True

```

nb\_updates 12 compared to 6

-----  
indexing completed for grain #0 with matching rate 6.78  
-----

transform matrix to matrix with lowest Euler Angles

start

```
[[-0.211415207301911  0.091252946262469 -0.97230572627369 ]
 [-0.77573594328912  0.590041596352251  0.224552051799525]
 [ 0.594613709497768  0.803610914885283 -0.054088075690982]]
```

final

```
[ [ 0.97230572627369  0.211415207301911  0.091252946262469]
 [-0.224552051799525  0.77573594328912  0.590041596352251]
 [ 0.054088075690982 -0.594613709497768  0.803610914885283]]
```

hkl [[2. 6. 4.]

[3. 3. 3.]

[5. 3. 3.]

[1. 3. 5.]

[6. 2. 4.]

[5. 1. 3.]

[1. 3. 3.]

[6. 4. 6.]

[4. 2. 6.]

[4. 2. 2.]

[3. 5. 3.]

[2. 2. 4.]]

new hkl (min euler angles) [[-4. -2. 6.]

[-3. -3. 3.]

[-3. -5. 3.]

[-5. -1. 3.]

[-4. -6. 2.]

[-3. -5. 1.]

[-3. -1. 3.]

[-6. -6. 4.]

[-6. -4. 2.]

[-2. -4. 2.]

[-3. -3. 5.]

[-4. -2. 2.]]

UB before [[-0.211415207301911 0.091252946262469 -0.97230572627369 ]

[-0.77573594328912 0.590041596352251 0.224552051799525]

[ 0.594613709497768 0.803610914885283 -0.054088075690982]]

new UB (min euler angles) [[ 0.97230572627369 0.211415207301911 0.

→0.091252946262469]

[-0.224552051799525 0.77573594328912 0.590041596352251]

[ 0.054088075690982 -0.594613709497768 0.803610914885283]]

writing fit file -----

for grainindex= 0

self.dict\_grain\_matrix[grain\_index] [[ 0.97230572627369 0.211415207301911 0.  
→0.091252946262469]

[-0.224552051799525 0.77573594328912 0.590041596352251]

[ 0.054088075690982 -0.594613709497768 0.803610914885283]]

self.refinedUBmatrix [[-0.211415207301911 0.091252946262469 -0.

→97230572627369 ]

```

[-0.77573594328912  0.590041596352251  0.224552051799525]
[ 0.594613709497768  0.803610914885283 -0.054088075690982]]
new UBs matrix in q= UBs G (s for strain)
strain_direct [[-1.339403716515974e-05 -4.138432477238585e-04  3.
↪978226745502020e-04]
[-4.138432477238585e-04 -1.131375039302607e-03 -1.517818975573709e-04]
[ 3.978226745502020e-04 -1.517818975573709e-04  6.255721962393768e-04]]
deviatoric strain [[ 0.000159671589578 -0.000413843247724  0.00039782267455 ]
[-0.000413843247724 -0.00095830941256 -0.000151781897557]
[ 0.00039782267455 -0.000151781897557  0.000798637822982]]
new UBs matrix in q= UBs G (s for strain)
strain_direct [[-1.339403716515974e-05 -4.138432477238585e-04  3.
↪978226745502020e-04]
[-4.138432477238585e-04 -1.131375039302607e-03 -1.517818975573709e-04]
[ 3.978226745502020e-04 -1.517818975573709e-04  6.255721962393768e-04]]
deviatoric strain [[ 0.000159671589578 -0.000413843247724  0.00039782267455 ]
[-0.000413843247724 -0.00095830941256 -0.000151781897557]
[ 0.00039782267455 -0.000151781897557  0.000798637822982]]
For comparison: a,b,c are rescaled with respect to the reference value of a =_
↪5.657500 Angstroms
lattice_parameter_direct_strain [ 5.657499999999999  5.651176877860324  5.
↪661180703302433
90.01741959362538  89.9544419036642  90.04747664598754 ]
final lattice_parameters [ 5.657499999999999  5.651176877860324  5.
↪661180703302433
90.01741959362538  89.9544419036642  90.04747664598754 ]
File : Ge0001_g0.fit written in /home/micha/LaueToolsPy3/LaueTools/notebooks
Experimental experimental spots indices which are not indexed []
Missing reflections grainindex is -100 for indexed grainindex 0
within angular tolerance 0.500

```

Remaining nb of spots to index for grain #1 : 0

12 spots have been indexed over 12  
indexing rate is --- : 100.0 percents  
indexation of short\_Ge0001.cor is completed  
for the 1 grain(s) that has(ve) been indexed as requested  
Leaving Index and Refine procedures...

number of indexed spots 12

## 6.2.3 Results of indexation can be found in attributes or through methods

```

first 2 indexed spots properties

#spot #grain 2theta chi X Y I h k l Energy
[[ 0.000000000000000e+00  0.000000000000000e+00  6.035945800000000e+01
 2.648319100000000e+01  6.261200000000000e+02  1.661280000000000e+03
 1.582539000000000e+04 -4.000000000000000e+00 -2.000000000000000e+00
 6.000000000000000e+00  1.632366988464985e+01]
[ 1.000000000000000e+00  0.000000000000000e+00  7.821582100000001e+01
 1.638153000000000e+00  1.027110000000000e+03  1.293280000000000e+03
 7.093127000000000e+04 -3.000000000000000e+00 -3.000000000000000e+00

```

(continues on next page)

(continued from previous page)

```
3.0000000000000000e+00 9.031851548397370e+00]]
```

```
#grain : [Npairs = Nb pairs with tolerance angle 0.2000, 100*Npairs/Ndirections theo.
↪]
```

```
{0: [12, 6.779661016949152]}
```

```
#grain : deviatoric strain
```

```
{0: array([[ 0.000159671589578, -0.000413843247724,  0.00039782267455 ],
           [-0.000413843247724, -0.00095830941256 , -0.000151781897557],
           [ 0.00039782267455 , -0.000151781897557,  0.000798637822982]])}
```

```
#grain : refined UB matrix
```

```
{0: array([[ 0.97230572627369 ,  0.211415207301911,  0.091252946262469],
           [-0.224552051799525,  0.77573594328912 ,  0.590041596352251],
           [ 0.054088075690982, -0.594613709497768,  0.803610914885283]])}
```

```
[0, 60.359458, 26.483191, 626.12, 1661.28, 15825.39, array([-4., -2.,  6.]), 16.
↪323669884649853, 0, 1], [1, 78.215821, 1.638153, 1027.11, 1293.28, 70931.27,
↪array([-3., -3.,  3.]), 9.03185154839737, 0, 1], [2, 68.680451, -15.358122, 1288.11,
↪1460.16, 22795.07, array([-3., -5.,  3.]), 12.73794897550435, 0, 1], [3, 108.
↪452917, 37.749461, 383.77, 754.58, 4400.61, array([-5., -1.,  3.]), 7.
↪989738078276174, 0, 1], [4, 83.349535, -27.458061, 1497.4, 1224.7, 13145.99,
↪array([-4., -6.,  2.]), 12.327936233758937, 0, 1], [5, 82.072076, -35.89243, 1672.
↪67, 1258.62, 13318.81, array([-3., -5.,  1.]), 9.870774398536632, 0, 1], [6, 81.
↪771257, 30.38247, 548.25, 1260.32, 6137.87, array([-3., -1.,  3.]), 7.
↪2986772558942405, 0, 1], [7, 91.798221, -8.309941, 1176.09, 1086.19, 11799.93,
↪array([-6., -6.,  4.]), 14.309911950813975, 0, 1], [8, 120.59561, -8.92066, 1183.27,
↪598.92, 17182.88, array([-6., -4.,  2.]), 9.435284238042113, 0, 1], [9, 64.329767,
↪-20.824155, 1379.17, 1553.58, 51933.84, array([-2., -4.,  2.]), 10.087272916663885,
↪0, 1]]
```





### 7.1 Browse Modules and Functions

- `genindex`
- `modindex`

### 7.2 Modules for Laue Pattern Simulation

The following modules are used to compute Laue pattern from grain (or crystal) structural parameters and 2D plane detector geometry:

- *CrystalParameters.py* defines structural parameters describing the crystal. It includes orientation matrix and strain operators.
- *lauecore.py* contains the core procedures to compute all Laue spots properties.
- *LaueGeometry.py* handles the 2D plane geometry set by the detector position and orientation with respect to sample and incoming direction.
- *multigrainsSimulator.py* allows to simulate an assembly of grains, some of them according to a distribution of grains. This module is called by the graphical user interface (LaueSimulatorGUI) which provides all arguments in an intuitive way.

#### 7.2.1 CrystalParameters

This module belongs to LaueTools package. It gathers procedures to define crystal lattice parameters and strain calculations

Main authors are JS Micha, O. Robach, S. Tardif June 2019

```

LaueTools.CrystalParameters.GrainParameter_from_Material (key_material, dictma-
                                                            terials={'Al':      ['Al',
[4.05, 4.05, 4.05, 90, 90,
90], 'fcc'], 'Al2Cu':
['Al2Cu',      [6.063,
6.063, 4.872, 90, 90,
90], 'no'], 'Al2O3':
['Al2O3', [4.785, 4.785,
12.991, 90, 90, 120],
'Al2O3'], 'Al2O3_all':
['Al2O3_all', [4.785,
4.785, 12.991, 90, 90,
120], 'no'], 'AlN':
['AlN', [3.11, 3.11, 4.98,
90.0, 90.0, 120.0],
'wurtzite'], 'Au':
['Au', [4.078, 4.078,
4.078, 90, 90, 90],
'fcc'], 'CCDL1949':
['CCDL1949', [9.89,
17.85, 5.31, 90, 107.5,
90], 'h+k=2n'], 'Cd-
HgTe':      ['CdHgTe',
[6.46678,      6.46678,
6.46678, 90, 90,
90], 'dia'], 'Cd-
HgTe_fcc':  ['Cd-
HgTe_fcc', [6.46678,
6.46678, 6.46678, 90,
90, 90], 'fcc'], 'CdTe':
['CdTe', [6.487, 6.487,
6.487, 90, 90, 90],
'fcc'], 'CdTeDiagB':
['CdTeDiagB', [4.5721,
7.9191, 11.1993, 90,
90, 90], 'no'], 'Crocidi-
dolite':    ['Crocicidolite',
[9.811, 18.013, 5.326,
90, 103.68, 90], 'no'],
'Crocicidolite_2': ['Cro-
cidolite_2', [9.76,
17.93, 5.35, 90, 103.6,
90], 'no'], 'Crocido-
lite_2_72deg': ['Crocicido-
lite_2', [9.76, 17.93,
5.35, 90, 76.4, 90], 'no'],
'Crocicidolite_small':
['Crocicidolite_small',
[3.2533333333333334,
5.976666666666667,
1.7833333333333332,
90, 103.6, 90],
'no'], 'Crocido-
lite_whittaker_1949':
['Crocido-
lite_whittaker_1949',
[9.89, 17.85, 5.31, 90,
107.5, 90], 'no'], 'Cu':
['Cu', [3.6, 3.6, 3.6,
90, 90, 90], 'fcc'],

```

create grain parameters list for the Laue pattern simulation

Can handle material defined in dictionary by four elements instead of 6 lattice parameters

**Parameters** **key\_material** (*string*) – material or structure label

**Returns** grain (4 elements list), contains\_U (boolean)

```

LaueTools.CrystalParameters.Prepare_Grain(key_material, OrientMatrix,
force_extinction=None, dictmaterials={'Al':
['Al', [4.05, 4.05, 4.05, 90, 90, 90], 'fcc'],
'Al2Cu': ['Al2Cu', [6.063, 6.063, 4.872, 90, 90,
90], 'no'], 'Al2O3': ['Al2O3', [4.785, 4.785,
12.991, 90, 90, 120], 'Al2O3'], 'Al2O3_all':
['Al2O3_all', [4.785, 4.785, 12.991, 90, 90,
120], 'no'], 'AlN': ['AlN', [3.11, 3.11, 4.98,
90.0, 90.0, 120.0], 'wurtzite'], 'Au': ['Au',
[4.078, 4.078, 4.078, 90, 90, 90], 'fcc'],
'CCDL1949': ['CCDL1949', [9.89, 17.85,
5.31, 90, 107.5, 90], 'h+k=2n'], 'CdHgTe':
['CdHgTe', [6.46678, 6.46678, 6.46678, 90,
90, 90], 'dia'], 'CdHgTe_fcc': ['CdHgTe_fcc',
[6.46678, 6.46678, 6.46678, 90, 90, 90], 'fcc'],
'CdTe': ['CdTe', [6.487, 6.487, 6.487, 90,
90, 90], 'fcc'], 'CdTeDiagB': ['CdTeDiagB',
[4.5721, 7.9191, 11.1993, 90, 90, 90], 'no'],
'Crocidolite': ['Crocidolite', [9.811, 18.013,
5.326, 90, 103.68, 90], 'no'], 'Crocidolite_2':
['Crocidolite_2', [9.76, 17.93, 5.35, 90, 103.6,
90], 'no'], 'Crocidolite_2_72deg': ['Crocido-
lite_2', [9.76, 17.93, 5.35, 90, 76.4, 90], 'no'],
'Crocidolite_small': ['Crocidolite_small',
[3.2533333333333334, 5.976666666666667,
1.7833333333333332, 90, 103.6, 90], 'no'],
'Crocidolite_whittaker_1949': ['Crocido-
lite_whittaker_1949', [9.89, 17.85, 5.31, 90,
107.5, 90], 'no'], 'Cu': ['Cu', [3.6, 3.6, 3.6,
90, 90, 90], 'fcc'], 'Cu6Sn5_monoclinic':
['Cu6Sn5_monoclinic', [11.02, 7.28, 9.827,
90, 98.84, 90], 'no'], 'Cu6Sn5_tetra':
['Cu6Sn5_tetra', [3.608, 3.608, 5.037, 90,
90, 90], 'no'], 'DIA': ['DIA', [5.0, 5.0, 5.0,
90, 90, 90], 'dia'], 'DIAs': ['DIAs', [3.56683,
3.56683, 3.56683, 90, 90, 90], 'dia'], 'DarinaMolecule':
['DarinaMolecule', [9.4254, 13.5004, 13.8241,
61.83, 84.555, 75.231], 'no'], 'FCC': ['FCC', [5.0, 5.0, 5.0, 90, 90,
90], 'fcc'], 'Fe': ['Fe', [2.856, 2.856, 2.856,
90, 90, 90], 'bcc'], 'Fe2Ta': ['Fe2Ta', [4.83,
4.83, 0.788, 90, 90, 120], 'no'], 'FeAl': ['FeAl',
[5.871, 5.871, 5.871, 90, 90, 90], 'fcc'], 'GaAs':
['GaAs', [5.65325, 5.65325, 5.65325, 90, 90,
90], 'dia'], 'GaN': ['GaN', [3.189, 3.189,
5.185, 90, 90, 120], 'wurtzite'], 'GaN_all':
['GaN_all', [3.189, 3.189, 5.185, 90, 90, 120],
'no'], 'Ge': ['Ge', [5.6575, 5.6575, 5.6575,
90, 90, 90], 'dia'], 'Ge_compressedhydro':
['Ge_compressedhydro', [5.64, 5.64, 5.64, 90,
90, 90.0], 'dia'], 'Ge_s': ['Ge_s', [5.6575,
5.6575, 5.6575, 90, 90, 89.5], 'dia'], 'Getest':
['Getest', [5.6575, 5.6575, 5.6574, 90, 90, 90],
'dia'], 'Hematite': ['Hematite', [5.03459,
5.03459, 13.7533, 90, 90, 120], 'no'],
'In': ['In', [3.2517, 3.2517, 4.9459, 90,
90, 90], 'h+k+l=2n'], 'InGa': ['InGa',
[3.3609999999999998, 3.3609999999999998,
5.439, 90, 90, 120], 'wurtzite'], 'InN': ['InN',
[3.533, 3.533, 5.693, 90, 90, 120], 'wurtzite'],

```

Constructor of the grain (crystal) parameters for Laue pattern simulation

if in key\_material definition (see dict\_Materials) orient matrix is missing (i.e. only lattice parameter are input)

**Parameters** **key\_material** (*str*) – material label

then list parameter will consider the provided value of the optional OrientMatrix argument

**Parameters** **force\_extinction** (*str*) – None, use default extinction rules, otherwise use other extinction corresponding to the label

`LaueTools.CrystalParameters.AngleBetweenNormals` (*HKL1s, HKL2s, Gstar*)

compute pairwise angles (in degrees) between reflections or lattice plane normals of two sets according to unit cell metrics Gstar

**Parameters**

- **HKL1s** – list of [H1,K1,L1]
- **HKL2s** – list of [H2,K2,L2]
- **Gstar** – 3\*3 matrix corresponding to reciprocal metric tensor of unit cell (as provided by `Gstar_from_directlatticeparams()`)

**Returns** array of pairwise angles between reflections

`LaueTools.CrystalParameters.FilterHarmonics_2` (*hkl, return\_indices\_toremove=0*)

keep only hkl 3d vectors that are representative of direction nh,nk,nl for any h,k,l signed integers

It removes only parallel vector but KEEPs antiparallel vectors (vec , -n vec) with n>0

**Parameters**

- **hkl** – array of 3d hkl indices
- **return\_indices\_toremove** – 1, returns indices of element in hkl that have been removed

`LaueTools.CrystalParameters.calc_B_RR` (*latticeparameters, directspace=1, setvolume=False*)

- Calculate B0 matrix (columns = vectors a\*,b\*,c\*) from direct (real) space lattice parameters (directspace=1)
- Calculate a matrix (columns = vectors a,b,c) from direct (real) space lattice parameters (directspace=0)

$$\mathbf{q}_{ortho} = B_0 \mathbf{G}^* \text{ where } \mathbf{G}^* = h\mathbf{a}^* + k\mathbf{b}^* + l\mathbf{c}^*$$

**Parameters**

- **latticeparameters** –
  - [a,b,c, alpha, beta, gamma] (angles are in degrees) if directspace=1
  - [a\*,b\*,c\*, alpha\*, beta\*, gamma\*] (angles are in degrees) if directspace=0
- **directspace** –
  - **1 (default) converts (reciprocal) direct lattice parameters** to (direct) reciprocal space calculates “B” matrix in the reciprocal space of input latticeparameters
  - **0 converts (reciprocal) direct lattice parameters to (reciprocal) direct space** calculates “B” matrix in same space of input latticeparameters
- **setvolume** –
  - False, sets direct unit cell volume to the true volume from lattice parameters
  - 1, sets direct unit cell volume to 1

- 'a\*\*3', sets direct unit cell volume to a\*\*3
- 'b\*\*3', sets direct unit cell volume to b\*\*3
- 'c\*\*3', sets direct unit cell volume to c\*\*3

**Returns** B Matrix (triangular up) from crystal (reciprocal space) frame to orthonormal frame matrix

**Return type** numpy array

**B matrix is used in  $q=U B G^*$  formula or** as B0 in  $q=(UB) B0 G^*$

after Busing Levy, Acta Crysta 22 (1967), p 457

$$\begin{pmatrix} a^* & b^* \cos \gamma^* & c^* \cos \beta^* \cos \alpha^* \\ 0 & b^* \sin \gamma^* & -c^* \sin \beta^* \cos \alpha^* \\ 0 & 0 & c^* \sin \beta^* \sin \alpha^* \end{pmatrix}$$

with

$$\cos(\alpha) = (\cos \beta^* \cos \gamma^* - \cos \alpha^*) / (\sin \beta^* \sin \gamma^*)$$

and

$$c^* \sin \beta^* \sin \alpha = 1/c$$

`LaueTools.CrystalParameters.DeviatoricStrain_LatticeParams` (*newUBmat*, *latticeparams*, *constantlength='a'*)

Computes deviatoric strain and new direct (real) lattice parameters from matrix *newUBmat* (rotation and deformation) considering that one lattice length is chosen to be constant

Zero strain corresponds to reference state of input *lattice parameters*

#### Parameters

- **newUBmat** – (3x3) matrix operator including rotation and deformation
- **latticeparams** – 6 lattice parameters [a,b,c,math:alpha, beta, gamma] in Angstrom and degrees
- **constantlength** – 'a', 'b', or 'c' to set one length according to the value in *latticeparams*

#### Returns

- 3x3 deviatoric strain tensor)
- *lattice\_parameter\_direct\_strain* (direct (real) lattice parameters)

**Return type** 3x3 numpy array, 6 elements list

---

#### Note:

- $q = \text{newUBmat} \cdot B0 \cdot G^*$  where B0 (triangular up matrix) comes from lattice parameters input.
  - equivalently,  $q = UB_{\text{star}_s} \cdot G^*$
- 

`LaueTools.CrystalParameters.VolumeCell` (*latticeparameters*)

Computes unit cell volume from lattice parameters (either real or reciprocal)

**Parameters** *latticeparameters* – 6 lattice parameters

**Returns** scalar volume

`LaueTools.CrystalParameters.VolumeCell (latticeparameters)`

Computes unit cell volume from lattice parameters (either real or reciprocal)

**Parameters** `latticeparameters` – 6 lattice parameters

**Returns** scalar volume

`LaueTools.CrystalParameters.matrix_to_rlat (mat, angles_in_deg=1)`

Returns RECIPROCAL lattice parameters of the unit cell  $a^*, b^*, c^*$  in columns of `mat`

**Parameters** `mat` – matrix where columns are respectively  $a^*, b^*, c^*$  coordinates in orthonormal frame

**Returns**  $[a^*, b^*, c^*, \alpha^*, \beta^*, \gamma^*]$  (angles are in degrees)

---

**Note:** Reciprocal lattice parameters are contained in UB matrix :  $q = \text{mat } G^*$

---

## 7.2.2 Laue Pattern Simulation

Core module to compute Laue Pattern in various geometry

Main author is J. S. Micha: `micha [at] esrf [dot] fr`

version July 2019 from LaueTools package hosted in

<http://sourceforge.net/projects/lauetools/>

or

<https://gitlab.esrf.fr/micha/lauetools>

`LaueTools.lauecore.Quicklist (OrientMatrix, ReciprocalBasisVectors, listRSnorm, lambdamin, verbose=0)`

return 6 indices min and max boundary values for each Miller index  $h, k, l$  to be contained in the largest Ewald Sphere.

**Parameters**

- **OrientMatrix** – orientation matrix (3\*3 matrix)
- **ReciprocalBasisVectors** – list of the three vectors  $a^*, b^*, c^*$  in the lab frame before rotation with OrientMatrix
- **listRSnorm** – : list of the three reciprocal space lengths of  $a^*, b^*, c^*$
- **lambdamin** – : lambdamin (in Angstrom) corresponding to energy max

**Returns**  $[[hmin, hmax], [kmin, kmax], [lmin, lmax]]$

`LaueTools.lauecore.genHKL_np (listn, Extinc)`

Generate all Miller indices  $hkl$  from indices limits given by `listn` and taking into account for systematic extinctions

**Parameters**

- **listn** ( $[[hmin, hmax], [kmin, kmax], [lmin, lmax]]$ ) – Miller indices limits (warning: these lists are used in python range (last index is excluded))
- **Extinc** (*string*) – label corresponding to systematic extinction rules on  $h, k$  and  $l$  miller indices such as ('fcc', 'bcc', 'dia', ...) or 'no' for any rules

**Returns** array of [h,k,l]

---

**Note:** node [0,0,0] is excluded

---

```
LaueTools.lauecore.getLaueSpots (wavelmin, wavelmax, crystalsParams, linestowrite,  
                                kf_direction='Z>0', OpeningAngleCollection=22.0, fast-  
                                compute=0, ResolutionAngstrom=False, fileOK=1, verbose=1,  
                                dictmaterials=None)
```

Compute Qxyz vectors and corresponding HKL miller indices for nodes in reciprocal space that can be measured for the given detection geometry and energy bandpass configuration.

#### Parameters

- **wavelmin** – smallest wavelength in Angstrom
- **wavelmax** – largest wavelength in Angstrom
- **crystalsParams** – list of *SingleCrystalParams*, each of them being a list of 4 elements for crystal orientation and strain properties:
  - [0](array): is the B matrix **a\*,b\*,c\*** vectors are expressed in column in LaueTools frame in reciprocal angstrom units
  - [1](str): peak Extinction rules ('no','fcc','dia', etc...)
  - [2](array): orientation matrix
  - [3](str): key for material element
- **kf\_direction** – string defining the average geometry, mean value of exit scattered vector: 'Z>0' top spots  
'Y>0' one side spots (towards hutch door)  
'Y<0' other side spots  
'X>0' transmission spots  
'X<0' backreflection spots
- **fastcompute** –
  - 1, compute reciprocal space (RS) vector BUT NOT the Miller indices
  - 0, returns both RS vectors (normalised) and Miller indices
- **ResolutionAngstrom** –
  - scalar, smallest interplanar distance ordered in crystal in angstrom.
  - None, all reflections will be calculated that can be time-consuming for large unit cell
- **linestowrite** – list of [string] that can be write in file or display in stdout. Example: [""], ["\*\*\*\*\*"], ["lauetools"]

#### Returns

- list of [Qx,Qy,Qz]s for each grain, list of [H,K,L]s for each grain (fastcompute = 0)
- list of [Qx,Qy,Qz]s for each grain, None (fastcompute = 1)

**Caution:** This method doesn't create spot instances.

This is done in filterLaueSpots with fastcompute = 0



**Caution:** finer selection of nodes : on camera , without harmonics can be done later with filterLaueSpots()

**Note:** lauetools laboratory frame is in this case: x// ki (center of ewald sphere has neagtive x component) z perp to x and belonging to the plane defined by x and dd vectors (where dd vector is the smallest vector joining sample impact point and points on CCD plane) y is perpendicular to x and z

`LaueTools.lauecore.create_spot(pos_vec, miller, detectordistance, allattributes=False, pixel-size=0.08056640625, dim=(2048, 2048))`

From reciprocal space position and 3 miller indices create a spot instance (on top camera geometry)

**Parameters**

- **pos\_vec** (*list of 3 float*) – 3D vector
- **miller** – list of 3 miller indices
- **detectordistance** – approximate distance detector sample (to compute complementary spots attributes)
- **allattributes** – False or 0 not to compute complementary spot attributes
- **allattributes** – boolean

**Returns** spot instance

**Note:** spot.Qxyz is a vector expressed in lauetools frame

X along x-ray and Z towards CCD when CCD on top and y towards experimental hutch door

`LaueTools.lauecore.create_spot_np(Qxyz, miller, detectordistance, allattributes=False, pixel-size=0.08056640625, dim=(2048, 2048))`

From reciprocal space position and 3 miller indices create a spot instance (on top camera geometry)

**Parameters**

- **pos\_vec** (*list of 3 float*) – 3D vector
- **miller** – list of 3 miller indices
- **detectordistance** – approximate distance detector sample (to compute complementary spots attributes)
- **allattributes** – False or 0 not to compute complementary spot attributes
- **allattributes** – boolean

**Returns** spot instance

**Note:** spot.Qxyz is a vector expressed in lauetools frame

X along x-ray and Z towards CCD when CCD on top and y towards experimental hutch door

`LaueTools.lauecore.filterLaueSpots(vec_and_indices, HarmonicsRemoval=1, fastcompute=0, kf_direction='Z>0', fileOK=0, detectordistance=70.0, detectordiameter=165.0, pixelsize=0.08056640625, dim=(2048, 2048), linestowrite=[[""]])`

Calculates list of grains spots on camera and without harmonics and on CCD camera from [[spots grain 0],[spots grain 1],etc] => returns [[spots grain 0],[spots grain 1],etc] w / o harmonics and on camera CCD

**Parameters**

- **vec\_and\_indices** – list of elements corresponding to 1 grain, each element is composed by \* [0] array of vector  
– [1] array of indices
- **HarmonicsRemoval** – 1, removes harmonics according to their miller indices (only for fastcompute = 0)
- **fastcompute** –  
– 1, outputs a list for each grain of 2theta spots and a list for each grain of chi spots (HARMONICS spots are still HERE!)  
– 0, outputs list for each grain of spots with
- **kf\_direction** (*string*) – label for detection geometry (CCD plane with respect to the incoming beam and sample)

**Returns**

- list of spot instances if fastcompute=0
- 2theta, chi if fastcompute=1

---

**Note:**

- USED IMPORTANTLY in lauecore.SimulateResults lauecore.SimulateLaue
  - USED in matchingrate.AngularResidues
  - USED in ParametricLaueSimulator.dosimulation\_parametric
  - USED in AutoindexationGUI.OnSimulate\_S3, DetectorCalibration.Reckon\_2pts, and others
- 

---

**Todo:** add dim in create\_spot in various geometries

---

`LaueTools.lauecore.get2ThetaChi_geometry (oncam_vec, oncam_HKL, detectordistance=70.0, pixelsize=0.08056640625, dim=(2048, 2048), kf_direction='Z>0')`  
computes list of spots instances from oncam\_vec (q 3D vectors) and oncam\_HKL (miller indices 3D vectors)

**Parameters**

- **oncam\_vec** (*array with 3D elements (shape = (n, 3))*) – q vectors [qx,qy,qz] (corresponding to kf collected on camera)
- **dim** (*list or tuple of 2 integers*) – CCD frame dimensions (nb pixels, nb pixels)
- **detectordistance** – approximate distance detector sample
- **detectordistance** – float or integer
- **pixelsize** (*float*) – pixel size in mm

**Param** kf\_direction : label for detection geometry (CCD plane with respect to the incoming beam and sample)

**Type** kf\_direction: string

**Returns** list of spot instances

---

**Note:** USED in `lauecore.filterLaueSpots`

---

**Todo:**

- to be replaced by something else not using spot class
  - put this function in LaueGeometry module ?
- 

`LaueTools.lauecore.calcSpots_fromHKLlist (UB, B0, HKL, dictCCD)`

computes all Laue Spots properties on 2D detector from a list of hkl (given structure by B0 matrix, orientation by UB matrix, and detector geometry by dictCCD)

**Parameters**

- **UB** (*3x3 array (or list)*) – orientation matrix (rotation -and if any- strain)
- **B0** (*3x3 array (or list)*) – initial  $a^*, b^*, c^*$  reciprocal unit cell basis vector in Laue-tools frame ( $x//ki$ )
- **HKL** (*array with shape = (n, 3)*) – array of Miller indices
- **dictCCD** – dictionary of CCD properties (with key 'CCDparam', 'pixelsize', 'dim') for 'ccdparam' 5 CCD calibration parameters [dd,xcen,ycen,xbet,xgam], pixelsize in mm, and (dim1, dim2)
- **dictCCD** – dict object

**Returns** list of arrays H, K, L, Qx, Qy, Qz, X, Y, twthe, chi, Energy

Fundamental equation  $\mathbf{q} = UB * B0 * \mathbf{G}^*$  with  $\mathbf{G}^* = h\mathbf{a}^* + k\mathbf{b}^* + l\mathbf{c}^*$

---

**Note:** USED in `DetectorCalibration.OnWriteResults`, and `PlotRefineGUI.onWriteFitFile`

---

```

LaueTools.lauecore.SimulateLaue (grain, emin, emax, detectorparameters, kf_direction='Z>0',
ResolutionAngstrom=False, removeharmonics=0, pixel-
size=0.08056640625, dim=(2048, 2048), detectordiam-
eter=None, force_extinction=None, dictmaterials={'Al': ['Al',
[4.05, 4.05, 4.05, 90, 90, 90], 'fcc'], 'Al2Cu': ['Al2Cu',
[6.063, 6.063, 4.872, 90, 90, 90], 'no'], 'Al2O3': ['Al2O3',
[4.785, 4.785, 12.991, 90, 90, 120], 'Al2O3'], 'Al2O3_all':
['Al2O3_all', [4.785, 4.785, 12.991, 90, 90, 120], 'no'], 'AlN':
['AlN', [3.11, 3.11, 4.98, 90.0, 90.0, 120.0], 'wurtzite'], 'Au':
['Au', [4.078, 4.078, 4.078, 90, 90, 90], 'fcc'], 'CCDL1949':
['CCDL1949', [9.89, 17.85, 5.31, 90, 107.5, 90], 'h+k=2n'],
'CdHgTe': ['CdHgTe', [6.46678, 6.46678, 6.46678, 90, 90,
90], 'dia'], 'CdHgTe_fcc': ['CdHgTe_fcc', [6.46678, 6.46678,
6.46678, 90, 90, 90], 'fcc'], 'CdTe': ['CdTe', [6.487, 6.487,
6.487, 90, 90, 90], 'fcc'], 'CdTeDiagB': ['CdTeDiagB',
[4.5721, 7.9191, 11.1993, 90, 90, 90], 'no'], 'Crocicidolite':
['Crocicidolite', [9.811, 18.013, 5.326, 90, 103.68, 90], 'no'],
'Crocicidolite_2': ['Crocicidolite_2', [9.76, 17.93, 5.35, 90, 103.6,
90], 'no'], 'Crocicidolite_2_72deg': ['Crocicidolite_2', [9.76,
17.93, 5.35, 90, 76.4, 90], 'no'], 'Crocicidolite_small': ['Cro-
cidolite_small', [3.2533333333333334, 5.976666666666667,
1.7833333333333332, 90, 103.6, 90], 'no'], 'Crocido-
lite_whittaker_1949': ['Crocicidolite_whittaker_1949', [9.89,
17.85, 5.31, 90, 107.5, 90], 'no'], 'Cu': ['Cu', [3.6, 3.6, 3.6, 90,
90, 90], 'fcc'], 'Cu6Sn5_monoclinic': ['Cu6Sn5_monoclinic',
[11.02, 7.28, 9.827, 90, 98.84, 90], 'no'], 'Cu6Sn5_tetra':
['Cu6Sn5_tetra', [3.608, 3.608, 5.037, 90, 90, 90], 'no'],
'DIA': ['DIA', [5.0, 5.0, 5.0, 90, 90, 90], 'dia'], 'DIAs':
['DIAs', [3.56683, 3.56683, 3.56683, 90, 90, 90], 'dia'],
'DarinaMolecule': ['DarinaMolecule', [9.4254, 13.5004,
13.8241, 61.83, 84.555, 75.231], 'no'], 'FCC': ['FCC', [5.0,
5.0, 5.0, 90, 90, 90], 'fcc'], 'Fe': ['Fe', [2.856, 2.856, 2.856,
90, 90, 90], 'bcc'], 'Fe2Ta': ['Fe2Ta', [4.83, 4.83, 0.788,
90, 90, 120], 'no'], 'FeAl': ['FeAl', [5.871, 5.871, 5.871,
90, 90, 90], 'fcc'], 'GaAs': ['GaAs', [5.65325, 5.65325,
5.65325, 90, 90, 90], 'dia'], 'GaN': ['GaN', [3.189, 3.189,
5.185, 90, 90, 120], 'wurtzite'], 'GaN_all': ['GaN_all',
[3.189, 3.189, 5.185, 90, 90, 120], 'no'], 'Ge': ['Ge', [5.6575,
5.6575, 5.6575, 90, 90, 90], 'dia'], 'Ge_compressedhydro':
['Ge_compressedhydro', [5.64, 5.64, 5.64, 90, 90, 90.0], 'dia'],
'Ge_s': ['Ge_s', [5.6575, 5.6575, 5.6575, 90, 90, 89.5], 'dia'],
'Getest': ['Getest', [5.6575, 5.6575, 5.6574, 90, 90, 90],
'dia'], 'Hematite': ['Hematite', [5.03459, 5.03459, 13.7533,
90, 90, 120], 'no'], 'In': ['In', [3.2517, 3.2517, 4.9459, 90, 90,
90], 'h+k+l=2n'], 'InGaN': ['InGaN', [3.3609999999999998,
3.3609999999999998, 5.439, 90, 90, 120], 'wurtzite'], 'InN':
['InN', [3.533, 3.533, 5.693, 90, 90, 120], 'wurtzite'], 'Mag-
netite': ['Magnetite', [8.391, 8.391, 8.391, 90, 90, 90], 'dia'],
'Magnetite_fcc': ['Magnetite_fcc', [8.391, 8.391, 8.391, 90,
90, 90], 'fcc'], 'Magnetite_sc': ['Magnetite_sc', [8.391, 8.391,
8.391, 90, 90, 90], 'no'], 'Nd45': ['Nd45', [5.4884, 5.4884,
5.4884, 90, 90, 90], 'fcc'], 'Ni': ['Ni', [3.5238, 3.5238,
3.5238, 90, 90, 90], 'fcc'], 'NiO': ['NiO', [2.96, 2.96, 7.23,
90, 90, 120], 'no'], 'NiTi': ['NiTi', [3.5506, 3.5506, 3.5506,
90, 90, 90], 'fcc'], 'SC': ['SC', [1.0, 1.0, 1.0, 90, 90, 90],
'no'], 'SC5': ['SC5', [5.0, 5.0, 5.0, 90, 90, 90], 'no'], 'SC7':
['SC7', [7.0, 7.0, 7.0, 90, 90, 90], 'no'], 'Si': ['Si', [5.4309,
5.4309, 5.4309, 90, 90, 120], 'no'], 'Sn': ['Sn', [5.83, 5.83, 3.18,
90, 90, 90], 'h+k+l=2n'], 'Ti': ['Ti', [2.95, 2.95, 4.68, 90,

```

**Computes Laue Pattern spots positions, scattering angles, miller indices** for a SINGLE grain or Xtal

**Parameters**

- **grain** – crystal parameters made of a 4 elements list
- **emin** – minimum bandpass energy (keV)
- **emax** – maximum bandpass energy (keV)
- **removeharmonics** –
  - **1, removes harmonics spots and keep fondamental spots (or reciprocal direction)**  
(with lowest Miller indices)
  - **0** keep all spots (including harmonics)

**Returns** single grain data: Twicetheta, Chi, Miller\_ind, posx, posy, Energy

---

**Todo:** To update to accept kf\_direction not only in reflection geometry

---

---

**Note:** USED in detectorCalibration...simulate\_theo for non routine geometry (ie except Z>0 (reflection top)  
X>0 (transmission))

---

```

LaueTools.lauecore.SimulateLaue_full_np (grain, emin, emax, detectorparameters,
kf_direction='Z>0', ResolutionAngstrom=False,
removeharmonics=0, pixelsize=0.08056640625,
dim=(2048, 2048), detectordiameter=None,
force_extinction=None, dictmaterials={'Al': ['Al',
[4.05, 4.05, 4.05, 90, 90, 90], 'fcc'], 'Al2Cu':
['Al2Cu', [6.063, 6.063, 4.872, 90, 90, 90], 'no'],
'Al2O3': ['Al2O3', [4.785, 4.785, 12.991, 90,
90, 120], 'Al2O3'], 'Al2O3_all': ['Al2O3_all',
[4.785, 4.785, 12.991, 90, 90, 120], 'no'], 'AlN':
['AlN', [3.11, 3.11, 4.98, 90.0, 90.0, 120.0],
'wurtzite'], 'Au': ['Au', [4.078, 4.078, 4.078, 90,
90, 90], 'fcc'], 'CCDL1949': ['CCDL1949', [9.89,
17.85, 5.31, 90, 107.5, 90], 'h+k=2n'], 'CdHgTe':
['CdHgTe', [6.46678, 6.46678, 6.46678, 90,
90, 90], 'dia'], 'CdHgTe_fcc': ['CdHgTe_fcc',
[6.46678, 6.46678, 6.46678, 90, 90, 90], 'fcc'],
'CdTe': ['CdTe', [6.487, 6.487, 6.487, 90, 90,
90], 'fcc'], 'CdTeDiagB': ['CdTeDiagB', [4.5721,
7.9191, 11.1993, 90, 90, 90], 'no'], 'Crocidolite':
['Crocidolite', [9.811, 18.013, 5.326, 90, 103.68,
90], 'no'], 'Crocidolite_2': ['Crocidolite_2',
[9.76, 17.93, 5.35, 90, 103.6, 90], 'no'], 'Crocidi-
lite_2_72deg': ['Crocidolite_2', [9.76, 17.93,
5.35, 90, 76.4, 90], 'no'], 'Crocidolite_small':
['Crocidolite_small', [3.2533333333333334,
5.976666666666667, 1.7833333333333332, 90,
103.6, 90], 'no'], 'Crocidolite_whittaker_1949':
['Crocidolite_whittaker_1949', [9.89, 17.85, 5.31,
90, 107.5, 90], 'no'], 'Cu': ['Cu', [3.6, 3.6,
3.6, 90, 90, 90], 'fcc'], 'Cu6Sn5_monoclinic':
['Cu6Sn5_monoclinic', [11.02, 7.28, 9.827,
90, 98.84, 90], 'no'], 'Cu6Sn5_tetra':
['Cu6Sn5_tetra', [3.608, 3.608, 5.037, 90, 90,
90], 'no'], 'DIA': ['DIA', [5.0, 5.0, 5.0, 90, 90,
90], 'dia'], 'DIAs': ['DIAs', [3.56683, 3.56683,
3.56683, 90, 90, 90], 'dia'], 'DarinaMolecule':
['DarinaMolecule', [9.4254, 13.5004, 13.8241,
61.83, 84.555, 75.231], 'no'], 'FCC': ['FCC',
[5.0, 5.0, 5.0, 90, 90, 90], 'fcc'], 'Fe': ['Fe',
[2.856, 2.856, 2.856, 90, 90, 90], 'bcc'], 'Fe2Ta':
['Fe2Ta', [4.83, 4.83, 0.788, 90, 90, 120], 'no'],
'FeAl': ['FeAl', [5.871, 5.871, 5.871, 90, 90,
90], 'fcc'], 'GaAs': ['GaAs', [5.65325, 5.65325,
5.65325, 90, 90, 90], 'dia'], 'GaN': ['GaN',
[3.189, 3.189, 5.185, 90, 90, 120], 'wurtzite'],
'GaN_all': ['GaN_all', [3.189, 3.189, 5.185, 90,
90, 120], 'no'], 'Ge': ['Ge', [5.6575, 5.6575,
5.6575, 90, 90, 90], 'dia'], 'Ge_compressedhydro':
['Ge_compressedhydro', [5.64, 5.64, 5.64, 90,
90, 90.0], 'dia'], 'Ge_s': ['Ge_s', [5.6575,
5.6575, 5.6575, 90, 90, 89.5], 'dia'], 'Getest':
['Getest', [5.6575, 5.6575, 5.6574, 90, 90,
90], 'dia'], 'Hematite': ['Hematite', [5.03459,
5.03459, 13.7533, 90, 90, 120], 'no'], 'In': ['In',
[3.2517, 3.2517, 4.9459, 90, 90, 90], 'h+k+l=2n'],
'InGaN': ['InGaN', [3.3609999999999998, 5.439, 90, 90, 120],
'wurtzite'], 'InN': ['InN', [3.533, 3.533, 5.693,
90, 90, 120], 'wurtzite'], 'Magnetite': ['Mag-

```

**Compute Laue Pattern spots positions, scattering angles, miller indices** for a SINGLE grain or Xtal using numpy vectorization

**Parameters**

- **grain** – crystal parameters in a 4 elements list
- **emin** – minimum bandpass energy (keV)
- **emax** – maximum bandpass energy (keV)
- **removeharmonics** – 1, remove harmonics spots and keep fundamental spots (with lowest Miller indices)

**Returns** single grain data: Twicetheta, Chi, Miller\_ind, posx, posy, Energy

---

**Todo:** update to accept kf\_direction not only in reflection geometry

---

---

**Note:** USED in detectorCalibration... simulate\_theo for routine geometry Z>0 (reflection top) X>0 (transmission)

---

`LaueTools.lauecore.SimulateResult` (*grain, emin, emax, simulparameters, fastcompute=1, ResolutionAngstrom=False, dictmaterials={'Al': ['Al', [4.05, 4.05, 4.05, 90, 90, 90], 'fcc'], 'Al2Cu': ['Al2Cu', [6.063, 6.063, 4.872, 90, 90, 90], 'no'], 'Al2O3': ['Al2O3', [4.785, 4.785, 12.991, 90, 90, 120], 'Al2O3'], 'Al2O3\_all': ['Al2O3\_all', [4.785, 4.785, 12.991, 90, 90, 120], 'no'], 'AlN': ['AlN', [3.11, 3.11, 4.98, 90.0, 90.0, 120.0], 'wurtzite'], 'Au': ['Au', [4.078, 4.078, 4.078, 90, 90, 90], 'fcc'], 'CCDL1949': ['CCDL1949', [9.89, 17.85, 5.31, 90, 107.5, 90], 'h+k=2n'], 'CdHgTe': ['CdHgTe', [6.46678, 6.46678, 6.46678, 90, 90, 90], 'dia'], 'CdHgTe\_fcc': ['CdHgTe\_fcc', [6.46678, 6.46678, 6.46678, 90, 90, 90], 'fcc'], 'CdTe': ['CdTe', [6.487, 6.487, 6.487, 90, 90, 90], 'fcc'], 'CdTeDiagB': ['CdTeDiagB', [4.5721, 7.9191, 11.1993, 90, 90, 90], 'no'], 'Crocicidolite': ['Crocicidolite', [9.811, 18.013, 5.326, 90, 103.68, 90], 'no'], 'Crocicidolite\_2': ['Crocicidolite\_2', [9.76, 17.93, 5.35, 90, 103.6, 90], 'no'], 'Crocicidolite\_2\_72deg': ['Crocicidolite\_2', [9.76, 17.93, 5.35, 90, 76.4, 90], 'no'], 'Crocicidolite\_small': ['Crocicidolite\_small', [3.2533333333333334, 5.976666666666667, 1.7833333333333332, 90, 103.6, 90], 'no'], 'Crocicidolite\_whittaker\_1949': ['Crocicidolite\_whittaker\_1949', [9.89, 17.85, 5.31, 90, 107.5, 90], 'no'], 'Cu': ['Cu', [3.6, 3.6, 3.6, 90, 90, 90], 'fcc'], 'Cu6Sn5\_monoclinic': ['Cu6Sn5\_monoclinic', [11.02, 7.28, 9.827, 90, 98.84, 90], 'no'], 'Cu6Sn5\_tetra': ['Cu6Sn5\_tetra', [3.608, 3.608, 5.037, 90, 90, 90], 'no'], 'DIA': ['DIA', [5.0, 5.0, 5.0, 90, 90, 90], 'dia'], 'DIAs': ['DIAs', [3.56683, 3.56683, 3.56683, 90, 90, 90], 'dia'], 'DarinaMolecule': ['DarinaMolecule', [9.4254, 13.5004, 13.8241, 61.83, 84.555, 75.231], 'no'], 'FCC': ['FCC', [5.0, 5.0, 5.0, 90, 90, 90], 'fcc'], 'Fe': ['Fe', [2.856, 2.856, 2.856, 90, 90, 90], 'bcc'], 'Fe2Ta': ['Fe2Ta', [4.83, 4.83, 0.788, 90, 90, 120], 'no'], 'FeAl': ['FeAl', [5.871, 5.871, 5.871, 90, 90, 90], 'fcc'], 'GaAs': ['GaAs', [5.65325, 5.65325, 5.65325, 90, 90, 90], 'dia'], 'GaN': ['GaN', [3.189, 3.189, 5.185, 90, 90, 120], 'wurtzite'], 'GaN\_all': ['GaN\_all', [3.189, 3.189, 5.185, 90, 90, 120], 'no'], 'Ge': ['Ge', [5.6575, 5.6575, 5.6575, 90, 90, 90], 'dia'], 'Ge\_compressedhydro': ['Ge\_compressedhydro', [5.64, 5.64, 5.64, 90, 90, 90.0], 'dia'], 'Ge\_s': ['Ge\_s', [5.6575, 5.6575, 5.6575, 90, 90, 89.5], 'dia'], 'Getest': ['Getest', [5.6575, 5.6575, 5.6574, 90, 90, 90], 'dia'], 'Hematite': ['Hematite', [5.03459, 5.03459, 13.7533, 90, 90, 120], 'no'], 'In': ['In', [3.2517, 3.2517, 4.9459, 90, 90, 90], 'h+k+l=2n'], 'InGaN': ['InGaN', [3.3609999999999998, 3.3609999999999998, 5.439, 90, 90, 120], 'wurtzite'], 'InN': ['InN', [3.533, 3.533, 5.693, 90, 90, 120], 'wurtzite'], 'Magnetite': ['Magnetite', [8.391, 8.391, 8.391, 90, 90, 90], 'dia'], 'Magnetite\_fcc': ['Magnetite\_fcc', [8.391, 8.391, 8.391, 90, 90, 90], 'fcc'], 'Magnetite\_sc': ['Magnetite\_sc', [8.391, 8.391, 8.391, 90, 90, 90], 'no'], 'Nd45': ['Nd45', [5.4884, 5.4884, 5.4884, 90, 90, 90], 'fcc'], 'Ni': ['Ni', [3.5238, 3.5238, 3.5238, 90, 90, 90], 'fcc'], 'NiO': ['NiO', [2.96, 2.96, 7.23, 90, 90, 120], 'no'], 'NiTi': ['NiTi', [3.5506, 3.5506, 3.5506, 90, 90, 90], 'fcc'], 'SC': ['SC', [1.0, 1.0, 1.0, 90, 90, 90], 'no'], 'SC5': ['SC5', [5.0, 5.0, 5.0, 90, 90, 90], 'no'], 'SC7': ['SC7', [7.0, 7.0, 7.0, 90, 90, 90], 'no'], 'Sb': ['Sb', [4.3, 4.3, 11.3, 90, 90, 120], 'no'], 'Si': ['Si', [5.4309, 5.4309, 5.4309, 90, 90, 90], 'dia'], 'Sn': ['Sn', [5.83, 5.83, 3.18,*



Simulates 2theta chi of Laue Pattern spots for ONE SINGLE grain

#### Parameters

- **grain** – crystal parameters in a 4 elements list
- **emin** – minimum bandpass energy (keV)
- **emax** – maximum bandpass energy (keV)

**Returns** 2theta, chi

**Warning:** Need of approximate detector distance and diameter to restrict simulation to a limited solid angle

#### Note:

- USED: in AutoindexationGUI.OnStart, LaueToolsGUI.OnCheckOrientationMatrix
- USED also IndexingImageMatching, lauecore.SimulateLaue\_merge

## 7.2.3 2D Detection Geometry

Module of lauetools project to compute Laue spots position on CCD camera. It handles detection and source geometry.

**Warning:** The frame (LT2) considered in this package (with y axis parallel to the incoming beam) is not the LaueTools frame (for which x is parallel to the incoming beam)

JS Micha June 2019

#### • Vectors Definitions

- **q** momentum transfer vector from resp. incoming and outgoing wave vector **ki** and **kf**,  $q = k_f - k_i$
- When a Laue spot exists, **q** is equal to the one node of the reciprocal lattice given by **G\*** vector
- **G\* is perpendicular to atomic planes defined by the three Miller indices h,k,l** such as  $\mathbf{G}^{***} = h^{**}\mathbf{a}^{*} + k^{**}\mathbf{b}^{**} + l^{**}\mathbf{c}^{**}$  where **a\***, **b\***, and **c\*** are the unit cell lattice basis vectors.
- **kf**: scattered beam vector whose corresponding unit vector is **uf**
- **ki** incoming beam vector, **ui** corresponding unit vector

#### • Laboratory Frame LT2

- I: origin
- z vertical up perpendicular to CCD plane (top camera geometry)
- y along X-ray horizontal
- x towards wall behind horizontal
- O: origin of pixel CCD frame in detecting plane
- **j** // **ui** incoming beam unit vector
- **z axis is defined by the CCD camera position. z axis is perpendicular to CCD plane** such as IO belongs to the plane Iyz

- bet: angle between **IO** and **k**
- **i\*\*=\*\*j\*\*^\*\*k** (when entering the BM32 hutch **i** is approximately towards the wall (in CCD on top geometry and beam coming from the right)
- M: point lying in CCD plane corresponding to Laue spot
- **uf** is the unit vector relative to vector **IM**

**k<sub>f</sub>** is also a vector collinear to **IM** with a length of  $R=1/\text{wavelength}=E/12.398$  [keV] with wavelength and Energy of the corresponding bragg's reflections.

**I** is the point from which calibration parameters (CCD position) are deduced (from a perfectly known crystal structure Laue pattern) **Iprime** is an other source of emission (posI or offset in functions)

$2\theta$  is the scattering angle between **ui** and **uf**, i.e.

$$\cos(2\theta) = u_i \cdot u_f$$

$$\mathbf{k}_f = (-\sin 2\theta \sin \chi, \cos 2\theta, \sin 2\theta \cos \chi)$$

$$\mathbf{k}_i = (0, 1, 0)$$

$$\text{Energy} = 12.398 * q^{**2} / (2 * \mathbf{q} \cdot \mathbf{ui}) = 12.398 * q^{**2} / (-2 \sin \theta)$$

#### **Calibration parameters (CCD position and detection geometry)**

- **calib**: list of the 5 calibration parameters [dd,xcen,ycen,xbet,xgam]
- **dd**: norm of **IO** [mm]
- **xcen,ycen** [pixel unit]: **pixels values in CCD frame of point O with respect to Oprime where**  
Oprime is the origin of CCD pixels frame (at a corner of the CCD array)
- **xbet**: angle between **IO** and **k** [degree]
- **xgam**: **azimutal rotation angle around z axis. Angle between CCD array axes and (i,\*\*j\*\*)** after rotation by **xbet** [degree].

#### *sample frame*

Origin is **I** and unit frame vectors (**is**, **js**, **ks**) are derived from absolute frame by the rotation (axis= - **i**, angle= wo) where wo is the angle between **js** and **j**

`LaueTools.LaueGeometry.calc_uflab(xcam, ycam, detectorplaneparameters, offset=0, returnAngles=1, verbose=0, pixelsize=0.08056640625, rectpix=0, kf_direction='Z>0')`

Computes unit vector  $\mathbf{u}_f = \frac{\mathbf{k}_f}{\|\mathbf{k}_f\|}$  in laboratory frame of scattered beam  $k_f$  (angle scattering angles 2theta and chi) from X, Y pixel Laue spot position

Unit vector **uf** correspond to normalized **k<sub>f</sub>** vector:  $\mathbf{q} = \mathbf{k}_f - \mathbf{k}_i$  from lists of X and Y Laue spots pixels positions on detector

#### **Parameters**

- **xcam** (list of floats) – list of pixel X position
- **ycam** (list of floats) – list of pixel Y position
- **detectorplaneparameters** – list of 5 calibration parameters
- **offset** – float, offset in position along incoming beam of source of scattered rays if positive: offset in sample depth units: mm

#### **Returns**

- if returnAngles=1 : 2theta, chi (default)

- if returnAngles!=1 : uflab, IMlab

LaueTools.LaueGeometry.**calc\_uflab\_trans**(*xcam, ycam, calib, returnAngles=1, verbose=0, pixelsize=0.08056640625, rectpix=0*)

compute  $2\theta$  and  $\chi$  scattering angles or **uf** and **kf** vectors from lists of X and Y Laue spots positions in TRANSMISSION geometry

#### Parameters

- **xcam** (*list of floats*) – list of pixel X position
- **ycam** (*list of floats*) – list of pixel Y position
- **calib** – list of 5 calibration parameters

#### Returns

- if returnAngles=1 : *twicetheta, chi (default)*
- if returnAngles!=1 : uflab, IMlab

# TODO: add offset like in reflection geometry

LaueTools.LaueGeometry.**calc\_xycam**(*uflab, calib, energy=0, offset=None, verbose=0, returnIpM=False, pixelsize=0.08056640625, rectpix=0*)

Computes Laue spots position x and y in pixels units in CCD frame from unit scattered vector **uf** expressed in Lab. frame

computes coordinates of point M on CCD from point source and **uflab**. Point Ip (source Iprime of x-ray scattered beams) (for each Laue spot **uflab** is the unit vector of **IpM**) Point Ip is shifted by offset (if not None) from the default point I (used to calibrate the CCD camera and 2theta chi determination)

th0 (theta in degrees) Energy (energy in keV)

#### Parameters

- **uflab** (*list or array (length must > 1)*) – list or array of [qx,qy,qz] (q vector)
- **calib** (*list of floats*) – list 5 detector calibration parameters
- **offset** (*list of floats ([x,y,z])*) – offset (in mm) in the scattering source (origin of Laue spots) position with respect to the position which has been used for the calibration of the CCD detector plane. Offset is positive when in the same direction as incident beam (i.e. in sample depth) (incident beam direction remains constant)

#### Returns

- xcam: list of pixel X coordinates
- ycam: list of pixel Y coordinates
- theta: list half scattering angle “theta” (in degree)
- optionally energy=1: add in output list of spot energies (in keV)
- if returnIpM and offset not None: return list of vectors **IprimeM**

LaueTools.LaueGeometry.**calc\_xycam\_transmission**(*uflab, calib, energy=0, offset=None, verbose=0, returnIpM=False, pixelsize=0.08056640625, rectpix=0*)

Computes Laue spots position x and y in pixels units (in CCD frame) from scattering vector q

As calc\_xycam() but in TRANSMISSION geometry

LaueTools.LaueGeometry.**uflab\_from2thetachi** (*twicetheta, chi, verbose=0*)

Computes **uf** vectors coordinates in lauetools LT2 frame from **kf** scattering angles  $2\theta$  and  $2\chi$  angles

### Parameters

- **twicetheta** – (list)  $2\theta$  angle(s) ( in degree)
- **chi** – (list)  $2\chi$  angle(s) ( in degree)

**Returns** list of  $\{bfu_f\} = [uf_x, uf_y, uf_z]$

**Return type** `list`

`LaueTools.LaueGeometry.from_twchi_to_qunit (Angles)`

from kf 2theta, chi to q unit in LaueTools frame (xx// ki) q=kf-ki returns array = (all x's, all y's, all z's)

Angles in degrees !! Angles[0] 2theta deg values, Angles[1] chi values in deg

this is the inverse function of from\_qunit\_to\_twchi(), useful to check it

`LaueTools.LaueGeometry.from_twchi_to_q (Angles)`

From kf 2theta,chi to q (arbitrary lenght) in lab frame (xx// ki) q=kf-ki returns array = (all qx's, all qy's, all qz's)

Angles in degrees !! Angles[0] 2theta deg values, Angles[1] chi values in deg

`LaueTools.LaueGeometry.from_qunit_to_twchi (arrayXYZ, labXMAS=0)`

Returns 2theta chi from a q unit vector (defining a direction) expressed in LaueTools frame (xx// ki) **q=kf-ki**

$$\begin{bmatrix} -\sin \theta \\ \cos \theta \sin \chi \\ \cos \theta \cos \chi \end{bmatrix}$$

---

**Note:** in LaueTools frame

$$kf = \begin{bmatrix} \cos 2\theta \\ \sin 2\theta \sin \chi \\ \sin 2\theta \cos \chi \end{bmatrix}$$

$$q = 2 \sin \theta \begin{bmatrix} -\sin \theta \\ \cos \theta \sin \chi \\ \cos \theta \cos \chi \end{bmatrix}$$

In LT2 Frame labXMAS=1

$$kf = \begin{bmatrix} \sin 2\theta \sin \chi \\ \cos 2\theta \\ \sin 2\theta \cos \chi \end{bmatrix}$$

$$q = 2 \sin \theta \begin{bmatrix} \cos \theta \sin \chi \\ -\sin \theta \\ \cos \theta \cos \chi \end{bmatrix}$$

---

`LaueTools.LaueGeometry.qvector_from_xy_E (xcamList, ycamList, energy, detectorplaneparameters, pixelsize)`

Returns q vectors in Lauetools frame given x and y pixel positions on detector for a given Energy (keV)

### Parameters

- **xcamList** – list pixel x postions
- **ycamList** – list pixel y postions
- **energy** – list pf energies
- **detectorplaneparameters** – list of 5 calibration parameters

- **pixelsize** – pixel size in mm

`LaueTools.LaueGeometry.unit_q(ttheta, chi, frame='lauetools', anglesample=40.0)`

Returns unit q vector from 2theta,chi coordinates

#### Parameters

- **ttheta** – list of 2theta angles (in degrees)
- **chi** – list of chi angles (in degrees)
- **anglesample** – incidence angle of beam to surface plane (degrees)
- **frame** – frame to express vectors in: 'lauetools', 'XMASlab' (LT2 frame), 'XMASSample'

**Returns** list of 3D u\_f (unit vector of scattering transfer q)

`LaueTools.LaueGeometry.Compute_data2thetachi(filename, tuple_column_X_Y_I, _nblines_headertoskip, sorting_intensity='yes', param=None, kf_direction='Z>0', verbose=1, pixelsize=0.08056640625, dim=(2048, 2048), saturation=0, forceextension_lines_to_extract=None, col_isbadspot=None, alpha_xray_incidence_correction=None)`

Converts spot positions x,y to scattering angles 2theta, chi from a list of peaks

#### Parameters

- **filename** (*string*) – fullpath to peaks list ASCII file
- **tuple\_column\_X\_Y\_I** (*3 elements*) – tuple with column indices of spots X, Y (pixels on CCD) and intensity
- **\_nblines\_headertoskip** – nb of line to skip before reading an array of data in ascii file
- **param** – list of CCD calibration parameters [det, xcen, ycen, xbet, xgam]
- **pixelsize** – pixelsize in mm
- **dim** – (nb pixels x, nb pixels y)
- **kf\_direction** (*string*) – label of detection geometry (CCD position): 'Z>0', 'X>0', ...
- **sorting\_intensity** – 'yes' sort spots list by decreasing intensity

saturation = 0 : do not read Ipixmax column of DAT file from LaueTools peaksearch saturation > 0 : read Ipixmax column and create data\_sat list data\_sat[i] = 1 if Ipixmax[i] > saturation, =0 otherwise

**Note:** **\_nblines\_headertoskip = 0 for .pik file (no header at all)** **\_nblines\_headertoskip = 1** for .peaks coming from fit2d

col\_Ipixmax = 10 for .dat from LT peak search using method "Local Maxima" (TODO : bug in Ipixmax for method "convolve")

`LaueTools.LaueGeometry.convert2corfile(filename, calibparam, dirname_in=None, dirname_out=None, pixelsize=0.08056640625, CCDCalibdict=None, add_props=False)`

Convert .dat (peaks list from peaksearch procedure) to .cor (adding scattering angles 2theta chi)

From X,Y pixel positions in peak list file (x,y,I,...) and detector plane geometry computes scattering angles 2theta chi and creates a .cor file (ascii peaks list (2theta chi X Y int ...))

**Parameters**

- **calibparam** – list of 5 CCD calibration parameters (used if CCDCalibdict is None or CCDCalibdict['CCDCalibParameters'] is missing)
- **pixelsize** – CCD pixelsize (in mm) (used if CCDCalibdict is None or CCDCalibdict['pixelsize'] is missing)
- **CCDCalibdict** – dictionary of CCD file and calibration parameters
- **add\_props** – add all peaks properties to .cor file instead of the 5 columns

`LaueTools.LaueGeometry.convert2corfile_fileseries` (*fileindexrange*, *filenameprefix*, *calibparam*, *suffix*=", *nbdigits*=4, *dirname\_in*=None, *dirname\_out*=None, *pixelsize*=0.08056640625, *fliprot*='no')

convert a serie of peaks list ascii files to .cor files (adding scattering angles).

Filename is decomposed as following for incrementing file index in #####: prefix####suffix example: myimage\_0025.myccd => prefix=myimage\_ nbdigits=4 suffix=.myccd

**Parameters**

- **nbdigits** – nb of digits of file index in filename (with zero padding) (example: for myimage\_0002.ccd nbdigits = 4)
- **calibparam** – list of 5 CCD calibration parameters

`LaueTools.LaueGeometry.convert2corfile_multiprocessing` (*fileindexrange*, *filenameprefix*, *calibparam*, *dirname\_in*=None, *suffix*", *nbdigits*=4, *dirname\_out*=None, *pixelsize*=0.08056640625, *fliprot*='no', *nb\_of\_cpu*=6)

launch several processes in parallel to convert .dat file to .cor file

`LaueTools.LaueGeometry.vec_normalTosurface` (*mat\_labframe*)  
solve  $\text{Mat} * X = (0,0,1)$  for X for pure rotation  $\text{invMat} = \text{transpose}(\text{Mat})$

TODO: add option sample angle and axis

`LaueTools.LaueGeometry.vec_onsurface_alongys` (*mat\_labframe*)  
solve  $\text{Mat} * X = (0,1,0)$  for X for pure rotation  $\text{invMat} = \text{transpose}(\text{Mat})$

`LaueTools.LaueGeometry.convert_xycam_from_sourceshift` (*OMs*, *IIP*, *calib*, *verbose*=0)  
From x,y on CCD camera (OMs) and source shift (IIP) compute modified x,y values for the SAME calibration (calib)(for further analysis)

return new value of x,y

`LaueTools.LaueGeometry.lengthInSample` (*depth*, *twtheta*, *chi*, *omega*, *verbose*=False)  
compute geometrical lengths in sample from impact point (I) at the surface to a point (B) where xray are scattered (or fluorescence is emitted) and finally escape from inside at point (C) lying at the sample surface (intersection of line with unit vector u with sample surface plane tilted by omega)

**Warning:** twtheta and chi angles can be misleading. Assumption is made that angles of unit vector from B to C (or to detector frame pixel) are  $2\theta$  and  $\chi$ . For large depth D, unit vector scattered beam direction is not given by  $2\theta$  and  $\chi$  angles as they are used for describing the scattering direction from point I and a given

detector frame position (you should then compute the two angles correction , actually  $\chi$  is unchanged, and the  $2\theta$  change is approx  $d/\text{distance}$  .i.e.  $3 \cdot 10^{-4}$  for  $d=20 \mu\text{m}$  and CCD at 70 mm)

---

**Note:**

**incoming beam coming from the right positive x direction with**

- IB = (-D,0,0)
  - BC =(xc+D,yc,zc)
  - and length BC is proportional to the depth D
- 

## 7.2.4 Multiple Grains and Strain/orientation Distribution

Module to compute Laue Patterns from several crystals in various geometry

Main author is J. S. Micha: `micha [at] esrf [dot] fr`

version July 2019 from LaueTools package for python2 hosted in

<http://sourceforge.net/projects/lauetools/>

or for python3 and 2 in

<https://gitlab.esrf.fr/micha/lauetools>

`LaueTools.multigrainsSimulator.Read_GrainListparameter` (*param*)

Read dictionary of input key parameters for simulation

`LaueTools.multigrainsSimulator.Construct_GrainsParameters_parametric` (*SelectGrains\_parametric*)

return list of simulation parameters for each grain set (mother and children grains)

```

LaueTools.multigrainsSimulator.dosimulation_parametric(_list_param,      Trans-
form_params=None,
SelectGrains=None,
emax=25.0,      emin=5.0,
detectordistance=68.7,
detectordiameter=165.0,
posCEN=(1024.0, 1024.0),
cameraAngles=(0.0,
0.0),      gauge=None,
kf_direction='Z>0',      pix-
elsize=0.08056640625,
dictmaterials={'Al': ['Al',
[4.05, 4.05, 4.05, 90,
90, 90], 'fcc'], 'Al2Cu':
['Al2Cu', [6.063, 6.063,
4.872, 90, 90, 90], 'no'],
'Al2O3': ['Al2O3', [4.785,
4.785, 12.991, 90, 90, 120],
'Al2O3'],      'Al2O3_all':
['Al2O3_all', [4.785, 4.785,
12.991, 90, 90, 120], 'no'],
'AlN': ['AlN', [3.11, 3.11,
4.98, 90.0, 90.0, 120.0],
'wurtzite'], 'Au': ['Au',
[4.078, 4.078, 4.078, 90, 90,
90], 'fcc'], 'CCDL1949':
['CCDL1949', [9.89,
17.85, 5.31, 90, 107.5,
90], 'h+k=2n'], 'CdHgTe':
['CdHgTe', [6.46678,
6.46678, 6.46678, 90, 90,
90], 'dia'], 'CdHgTe_fcc':
['CdHgTe_fcc', [6.46678,
6.46678, 6.46678, 90, 90,
90], 'fcc'], 'CdTe': ['CdTe',
[6.487, 6.487, 6.487, 90,
90, 90], 'fcc'], 'CdTe-
DiagB': ['CdTeDiagB',
[4.5721, 7.9191, 11.1993,
90, 90, 90], 'no'], 'Cro-
cidolite': ['Crocidolite',
[9.811, 18.013, 5.326,
90, 103.68, 90], 'no'],
'Crocidolite_2': ['Cro-
cidolite_2', [9.76, 17.93,
5.35, 90, 103.6, 90], 'no'],
'Crocidolite_2_72deg':
['Crocidolite_2', [9.76,
17.93, 5.35, 90, 76.4, 90],
'no'], 'Crocidolite_small':
['Crocidolite_small',
[3.2533333333333334,
5.976666666666667,
1.7833333333333332, 90,
103.6, 90], 'no'], 'Croc-
idolite_whittaker_1949':
lite_whittaker_1949', [9.89,
17.85, 5.31, 90, 107.5, 90],
'no'], 'Cu': ['Cu', [3.6,

```



Simulation of orientation or deformation gradient. From parent grain simulate a list of transformations (deduced by a parametric variation)

`_list_param` : list of parameters for each grain [grain parameters, grain name]

`posCEN` =(Xcen, Ycen) `cameraAngles` =(Xbet, Xgam)

**Returns** (list\_twicetheta, list\_chi, list\_energy, list\_Miller, list\_posX, list\_posY, ParentGrain-Name\_list, list\_ParentGrain\_transforms, calib, total\_nb\_grains)

TODO:simulate for any camera position TODO: simulate spatial distribution of laue pattern origin

## 7.3 Modules for Digital Image processing, Peak Search & Fitting

### 7.3.1 Peak Search and Fit (readmccd.py)

#### Module functions

The next documentation comes from the docstring in the header of function or class definition.

#### readmccd.py

#### IOimagefile.py

IOimagefile module is made for reading data contained in binary image file fully or partially.

More tools can be found in LaueTools package at [sourceforge.net](http://sourceforge.net) and [gitlab.esrf.fr](http://gitlab.esrf.fr) March 2020

`LaueTools.IOimagefile.stringint(k, n)`

returns string of integer k with n zeros padding (by placing zeros before to have n characters)

#### Parameters

- **k** – integer to convert
- **n** – nb of digits for zero padding

**Returns** string of length n containing integer k

Example: 1 -> '0001' 15 -> '0015'

`LaueTools.IOimagefile.setfilename(imagefilename, imageindex, nbdigits=4, CCDLabel=None)`

reconstruct filename string from imagefilename and update filename index with imageindex

#### Parameters

- **imagefilename** – filename string (full path or not)
- **imageindex** (*string*) – index in filename

**Return filename** input filename with index replaced by input imageindex

**Return type** string

`LaueTools.IOimagefile.getIndex_fromfilename(imagefilename, nbdigits=4, CCDLabel=None, stackimageindex=-1)`

get integer index from imagefilename string

**Parameters** **imagefilename** – filename string (full path or not)

**Returns** file index

`LaueTools.IOimagefile.getfilename` (*dirname*, *imfileprefix*, *imfilesuffix=None*, *numim=None*,  
*nbdigits\_filename=4*)

to get the global file name (name+path) for given components of the name put %4d instead of stringint

`LaueTools.IOimagefile.getwildcardstring` (*CCDlabel*)

return smart wildcard to open binary CCD image file with priority of CCD type of CCDlabel

**Parameters** *CCDlabel* (*str*) – string label defining the CCD type

**Returns** string from concatenated strings to be used in wxpython open file dialog box

**Return type** *str*

`LaueTools.IOimagefile.getpixelValue` (*filename*, *x*, *y*, *ccdtypegeometry='edf'*)

return pixel value at x,y

**Warning:** Very old function. To be checked. Use better readpixelvalue in plotdip.py

#### Parameters

- **filename** (*str*) – path to image file
- **x** (*int*) – x pixel value
- **y** (*int*) – y pixel value
- **ccdtypegeometry** (*str*, *optional*) – CCD label, defaults to “edf”

**Returns** pixel intensity

**Return type** *int*

`LaueTools.IOimagefile.readheader` (*filename*, *offset=4096*, *CCDLabel='MARCCD165'*)

return header in a raw format

default offset for marccd image

`LaueTools.IOimagefile.read_header_marccd` (*filename*)

return string of parameters found in header in marccd image file .mccd

- print allsentences displays the header
- use allsentences.split('n') to get a list

`LaueTools.IOimagefile.read_header_marccd2` (*filename*)

return string of parameters comments and exposure time found in header in marccd image file .mccd

- print allsentences displays the header
- use allsentences.split('n') to get a list

`LaueTools.IOimagefile.read_header_scmos` (*filename*)

return string of parameters comments and exposure time found in header in scm is image file .tif

- print allsentences displays the header
- use allsentences.split('n') to get a list

`LaueTools.IOimagefile.read_motorsposition_fromheader` (*filename*, *CCDLabel='MARCCD165'*)

return xyzpositions, expo\_time from image file header available for “MARCCD165”, “sCMOS”, “sC-MOS\_fliplr”

`LaueTools.IOimagefile.readoneimage_full(filename, frametype='mccd', dirname=None)`  
 too SLOW! reads 1 entire image (marCCD format) :return: PILimage, image object of PIL module (16 bits integer) and arrayofdata: 2D array of intensity #TODO: manage framedim like readoneimage() just below

`LaueTools.IOimagefile.readCCDimage(filename, CCDLabel='MARCCD165', dirname=None, stackimageindex=-1, verbose=0)`

Read raw data binary image file.

Read raw data binary image file and return pixel intensity 2D array such as to fit the data (2theta, chi) scattering angles representation convention.

#### Parameters

- **filename** (*str*) – path to image file (fullpath if 'dirname' =None)
- **CCDLabel** (*str*, *optional*) – label, defaults to “MARCCD165”
- **dirname** (*str*, *optional*) – folder path, defaults to None
- **stackimageindex** (*int*, *optional*) – index of images bunch, defaults to -1
- **verbose** (*int*, *optional*) – 0 or 1, defaults to 0

**Raises** `ValueError` – if data format and CCD parameters from label are not compatible

#### Returns

- dataimage, 2D array image data pixel intensity properly oriented
- framedim, iterable of 2 integers shape of dataimage
- fliprot : string, key for CCD frame transform to orient image

**Return type** tuple of 3 elements

`LaueTools.IOimagefile.readoneimage(filename, framedim=(2048, 2048), dirname=None, offset=4096, formatdata='uint16')`

returns a 1d array of integers from a binary image file (full data)

#### Parameters

- **filename** (*str*) – image file name (full path if dirname=0)
- **framedim** (*tuple of 2 integers*, *optional*) – detector dimensions, defaults to (2048, 2048)
- **dirname** (*str*, *optional*) – folder path, defaults to None
- **offset** (*int*, *optional*) – file header in byte (octet), defaults to 4096
- **formatdata** (*str*, *optional*) – numpy format of raw binary image pixel value, defaults to “uint16”

**Returns** dataimage : image data pixel intensity

**Return type** 1D array

`LaueTools.IOimagefile.readoneimage_band(filename, framedim=(2048, 2048), dirname=None, offset=4096, line_startindex=0, line_finalindex=2047, formatdata='uint16')`

returns a 1d array of integers from a binary image file. Data located in band according shape of data (framedim)

#### Parameters

- **filename** – string path to image file (fullpath if 'dirname'=None)
- **offset** – integer nb of file header bytes

- **framedim** – iterable of 2 integers shape of expected 2D data
- **formatdata** – string key for numpy dtype to decode binary file

**Returns** dataimage, 1D array, image data pixel intensity

`LaueTools.IOimagefile.readoneimage_crop_fast(filename, dirname=None, CCDLabel='MARCCD165', firstElemIndex=0, lastElemIndex=2047)`

Returns a 2d array of integers from a binary image file. Data are taken only from a rectangle with respect to firstElemIndex and lastElemIndex.

**Parameters**

- **filename** – string, path to image file (fullpath if 'dirname'=None)
- **offset** – integer, nb of file header bytes
- **framedim** – iterable of 2 integers, shape of expected 2D data
- **formatdata** – string, key for numpy dtype to decode binary file

**Returns** dataimage : 1D array image data pixel intensity

`LaueTools.IOimagefile.readrectangle_in_image(filename, pixx, pixy, halfboxx, halfboxy, dirname=None, CCDLabel='MARCCD165', verbose=True)`

returns a 2d array of integers from a binary image file. Data are taken only from a rectangle centered on pixx, pixy

**Returns** dataimage : 2D array, image data pixel intensity

`LaueTools.IOimagefile.readoneimage_crop(filename, center, halfboxsize, CCDLabel='PRINCETON', dirname=None)`

return a cropped array of data read in an image file

**Parameters**

- **filename** – string, path to image file (fullpath if 'dirname'=None)
- **center** – iterable of 2 integers, (x,y) pixel coordinates
- **halfboxsize** – integer or iterable of 2 integers, ROI half size in both directions

**Returns** dataimage : 1D array, image data pixel intensity

---

**Todo:** useless?

---

`LaueTools.IOimagefile.readoneimage_manycrops(filename, centers, boxsize, stackimageindex=-1, CCDLabel='MARCCD165', addImax=False, use_data_corrected=None)`

reads 1 image and extract many regions centered on center\_pixel with xyboxsize dimensions in pixel unit

**Parameters**

- **filename** – string,fullpath to image file
- **centers** – list or array of [int,int] centers (x,y) pixel coordinates
- **use\_data\_corrected** – enter data instead of reading data from file must be a tuple of 3 elements: fulldata, framedim, fliprot where fulldata is a numpy.ndarray as output by `readCCDimage()`

- **boxsize** – iterable 2 elements or integer boxsizes [in x, in y] direction or integer to set a square ROI

**Returns** Data, list of 2D array pixel intensity or Data and Imax

`LaueTools.IOimagefile.writeimage(outputname, _header, data, dataformat=<class 'numpy.uint16'>)`  
 from data 1d array of integers with header coming from a `f.open('imagefile')`; `f.read(headersize)`; `f.close()` ..  
 warning:: header contain dimensions for subsequent data. Check before the compatibility of data with header  
 infos(nb of byte per pixel and array dimensions)

`LaueTools.IOimagefile.write_rawbinary(outputname, data, dataformat=<class 'numpy.uint16'>)`

write a binary file without header of a 2D array

used ?

`LaueTools.IOimagefile.SumImages(prefixname, suffixname, ind_start, ind_end, dirname=None, plot=0, output_filename=None, CCDLabel=None, nbdigits=0)`

sum images and write image with 32 bits per pixel format (4 bytes)

used?

`LaueTools.IOimagefile.Add_Images2(prefixname, ind_start, ind_end, plot=0, writefilename=None, CCDLabel='MARCCD165', average=True)`

in dev

`LaueTools.IOimagefile.Add_Images(prefixname, ind_start, ind_end, plot=0, writefilename=None)`  
 Add continuous sequence of images

---

**Note:** Add\_Images2 exists

---

### Parameters

- **prefixname** – string, prefix common part of name of files
- **ind\_start** – int, starting image index
- **ind\_end** – int, final image index
- **writefilename** – string, new image filename where to write datastart (with last image file header read)

**Returns** datastart, array accumulation of 2D data from each image

`LaueTools.IOimagefile.get_imagesize(framedim, nbbits_per_pixel, headersize_bytes)`  
 return size of image in byte (= 1 octet = 8 bits)

## imageprocessing.py

imageprocessing module is made to modify filter data array

More tools can be found in LaueTools package at [sourceforge.net](https://sourceforge.net) and [gitlab.esrf.fr](https://gitlab.esrf.fr) March 2020

`LaueTools.imageprocessing.getindices2cropArray(center, halfboxsizeROI, arrayshape, flipxycenter=False)`

return array indices limits to crop array data

### Parameters

- **center** – iterable of 2 elements (x,y) pixel center of the ROI
- **halfboxsizeROI** – integer or iterable of 2 elements half boxsize ROI in two dimensions
- **arrayshape** – iterable of 2 integers maximal number of pixels in both directions
- **flipxycenter** – boolean True: swap x and y of center with respect to others parameters that remain fixed

**Returns** imin, imax, jmin, jmax : 4 integers 4 indices allowing to slice a 2D np.ndarray

---

**Todo:** merge with check\_array\_indices()

---

LaueTools.imageprocessing.**check\_array\_indices** (*imin, imax, jmin, jmax, framedim=None*)

Return 4 indices for array slice compatible with framedim

**Parameters**

- **imax, jmin, jmax** (*imin,*) – 4 integers mini. and maxi. indices in both directions
- **framedim** – iterable of 2 integers shape of the array to be sliced by means of the 4 indices

**Returns** imin, imax, jmin, jmax: 4 integers mini. and maxi. indices in both directions

---

**Todo:** merge with getindices2cropArray()

---

LaueTools.imageprocessing.**to8bits** (*PILimage, normalization\_value=None*)

convert PIL image (16 bits) in 8 bits PIL image

**Returns**

- [0] 8 bits image
- [1] corresponding pixels value array

---

**Todo:** since not used, may be deleted

---

LaueTools.imageprocessing.**diff\_pix** (*pix, array\_pix, radius=1*)

returns index in array\_pix which is the closest to pix if below the tolerance radius

array\_pix: array of 2d pixel points pix: one 2elements pixel point

LaueTools.imageprocessing.**minmax** (*D\_array, center, boxsize, framedim=(2048, 2048), withmax-pos=False*)

extract min and max from a 2d array in a ROI

Obsolete? Still used in LocalMaxima\_ShiftArrays()

Parameters D\_array : 2D array

data array

**center** [iterable of 2 integers] (x,y) pixel center

**boxsize** [integer or iterable of 2 integers] full boxsize defined in both directions

**framedim** [iterable of 2 integers] shape of D\_array

Return [min, max]: minimum and maximum pixel intensity in ROI [min, max], absolute\_max\_pos : if with-maxpos is True add in output

the absolute position of the largest pixel

#TODO: replace by `scipy.ndimage.extrema` # see next functions below # framedim = from dictionary of CCDs  
D\_array shape is `flip(framedim)`

```
LaueTools.imageprocessing.getExtrema(data2d, center, boxsize, framedim, ROIcoords=0,
                                     flipxycenter=True)
return min max XYposmin, XYposmax values in ROI
```

#### Parameters

- **ROIcoords** – 1 in local array indices coordinates 0 in X,Y pixel CCD coordinates
- **flipxycenter** – boolean like swap input center coordinates
- **data2d** – 2D array data array as read by `readCCDimage()`

**Returns** min, max, XYposmin, XYposmax: - min : minimum pixel intensity - max : maximum pixel intensity - XYposmin : list of absolute pixel coordinates of lowest pixel - XYposmax : list of absolute pixel coordinates of largest pixel

```
LaueTools.imageprocessing.getIntegratedIntensity(data2d, center, boxsize, framedim,
                                                  thresholdlevel=0.2, flipxycenter=True)
return crude estimate of integrated intensity of peak above a given relative threshold
```

#### Parameters

- **ROIcoords** – 1 in local array indices coordinates 0 in X,Y pixel CCD coordinates
- **flipxycenter** – boolean like swap input center coordinates
- **data2d** – 2D array data array as read by `readCCDimage()`
- **Thresholdlevel** – relative level above which pixel intensity must be taken into account  
 $I(p) - \text{minimum} > \text{Thresholdlevel} * (\text{maximum} - \text{minimum})$

**Returns** integrated intensity, minimum absolute intensity, nbpixels used for the summation

```
LaueTools.imageprocessing.getMinMax(data2d, center, boxsize, framedim)
return min and max values in ROI
```

Parameters:

**data2d** [2D array] array as read by `readCCDimage`

```
LaueTools.imageprocessing.minmax_fast(D_array, centers, boxsize=(25, 25))
extract min (considered as background in boxsize) and intensity at center from a 2d array at different places (centers)
```

centers is tuple a two array ( array([slow indices]), array([fast indices]))

return:

[0] background values [1] intensity value

used?

```
LaueTools.imageprocessing.normalize_shape(shape)
return shape in case a scalar was given: return (shape,)
```

```
LaueTools.imageprocessing.LoG(r, sigma=None, dim=1, r0=None, peakVal=None)
```

note: returns *negative* Laplacian-of-Gaussian (aka. mexican hat) zero-point will be at  $\sqrt{\text{dim}} * \text{sigma}$  integral is `_always_ 0` if peakVal is None: uses “mathematical” “gaussian derived” norm if r0 is not None: specify radius of zero-point (IGNORE sigma !!)

`LaueTools.imageprocessing.LoGArr` (*shape*=(256, 256), *r0*=None, *sigma*=None, *peakVal*=None, *orig*=None, *wrap*=0, *dtype*=<class 'numpy.float32'>)

returns n-dim Laplacian-of-Gaussian (aka. mexican hat) if *peakVal* is not None

result max is *peakVal*

if *r0* is not None: specify radius of zero-point (IGNORE *sigma* !!)

credits: “Sebastian Haase <[haase@msg.ucsf.edu](mailto:haase@msg.ucsf.edu)>”

`LaueTools.imageprocessing.radialArr` (*shape*, *func*, *orig*=None, *wrap*=False, *dtype*=<class 'numpy.float32'>)

generates and returns radially symmetric function sampled in volume(*image*) of shape *shape* if *orig* is None the origin defaults to the center *func* is a 1D function with 1 parameter: *r*

if *shape* is a scalar uses implicitly (*shape*,) *wrap* tells if functions is continued wrapping around image boundaries *wrap* can be True or False or a tuple same length as *shape*:

then *wrap* is given for each axis sperately

`LaueTools.imageprocessing.LocalMaxima_ndimage` (*Data*, *peakVal*=4, *boxsize*=5, *central\_radius*=2, *threshold*=1000, *connectivity*=1, *returnfloatmeanpos*=0, *autothresholdpercentage*=None)

returns (float) i,j positions in array of each blob (peak, spot, assembly of hot pixels or whatever)

---

**Note:** used only in LocalMaxima\_KernelConvolution

---

inputs

**peakVal, boxsize, central\_radius:** parameters for numerical convolution with a mexican-hat-like kernel

**threshold:** intensity threshold of filtered *Data* (by convolution with the kernel) above which blob signal will be considered if = 0 : take all blobs at the expense of processing time

**connectivity :** 1 for filled square 3\*3 connectivity 0 for 3\*3 star like connectivity

**autothresholdpercentage :** threshold in filtered image with respect to the maximum intensity in filtered image

output: array (n,2): array of 2 indices

`LaueTools.imageprocessing.ConvolvebyKernel` (*Data*, *peakVal*=4, *boxsize*=5, *central\_radius*=2)

Convolve *Data* array withn mexican-hat kernel

inputs: *Data* : 2D array containing pixel intensities *peakVal* > *central\_radius* : defines pixel distance from box center where weights are positive

(in the middle) and negative farther to converge back to zero

*boxsize* : size of the box

ouput: array (same shape as *Data*)



```
LaueTools.imageprocessing.LocalMaxima_KernelConvolution (Data, framedim=(2048, 2048), peakValConvolve=4, boxsizeConvolve=5, central_radiusConvolve=2, thresholdConvolve=1000, connectivity=1, IntensityThreshold=500, boxsize_for_probing_minimal_value_background=30, return_nb_raw_blobs=0, peakposition_definition='max')
```

return local maxima (blobs) position and amplitude in Data by using convolution with a mexican hat like kernel.

**Two Thresholds are used sequentially:**

- **thresholdConvolve** : level under which intensity of kernel-convolved array is discarded
- **IntensityThreshold** : level under which blob whose local intensity amplitude in raw array is discarded

**Parameters**

- **Data** – 2D array containing pixel intensities
- **boxsizeConvolve, central\_radiusConvolve** (*peakValConvolve,*) – convolution kernel parameters
- **thresholdConvolve** – minimum threshold (expressed in unit of convolved array intensity) under which convoluted blob is rejected. It can be zero (all blobs are accepted but time consuming)
- **connectivity** – shape of connectivity pattern to consider pixels belonging to the same blob.
  - 1: filled square (1 pixel connected to 8 neighbours)
  - 0: star (4 neighbours in vertical and horizontal direction)
- **IntensityThreshold** – minimum local blob amplitude to accept
- **boxsize\_for\_probing\_minimal\_value\_background** – boxsize to evaluate the background and the blob amplitude
- **peakposition\_definition** – string ('max' or 'center') key to assign to the blob position its hottest pixel position or its center (no weight)

**Returns**

**peakslist** [array like (n,2)] list of peaks position (pixel)

**Ipixmax** [array like (n,1) of integer] list of highest pixel intensity in the vicinity of each peak

**npeaks** [integer] nb of peaks (if return\_nb\_raw\_blobs =1)

```
LaueTools.imageprocessing.LocalMaxima_ShiftArrays (Data, framedim=(2048, 2048), IntensityThreshold=500, Saturation_value=65535, boxsize_for_probing_minimal_value_background=30, nb_of_shift=25, pixeldistance_remove_duplicates=25, verbose=0)
```

blob search or local maxima search by shift array method (kind of derivative)

**Warning:** Flat peak (= two neighbouring pixel with rigourously the same intensity) is not detected

`LaueTools.imageprocessing.shiftarrays_accum` (*Data\_array, n, dimensions=1, diags=0*)  
 idem than `shiftarrays()` but with all intermediate shifted arrays 1D returns 3 arrays corresponding to shifted arrays by *n* in two directions and original one 2D returns 5 arrays corresponding to shifted arrays by *n* in two directions and original one

these arrays are ready for comparison with eg `np.greater`

*Data\_array* must have shape (slowdim,fastdim) so that  $\text{slowdim}-2*n \geq 1$  and  $\text{fastdim}-2*n \geq 1$  (ie central array with zero shift has some elements)

TODO: replace append by a pre allocated array

---

**Note:** `readmccd.localmaxima` is better

---

`LaueTools.imageprocessing.LocalMaxima_from_thresholdarray` (*Data, IntensityThreshold=400, rois=None, framedim=None, verbose=False*)

return center of mass of each blobs composes by pixels above `IntensityThreshold`

if `Centers` = list of (x,y, halfboxsize, halfboxsize) perform only blob search in theses ROIs

**Warning:** center of mass of blob where all intensities are set to 1

`LaueTools.imageprocessing.localmaxima` (*DataArray, n, diags=1, verbose=0*)  
 from `DataArray` 2D returns (`array([i1,i2,...,ip]),array([j1,j2,...,jp])`) of indices where pixels value is higher in two direction up to *n* pixels

this tuple can be easily used after in the following manner: `DataArray[tupleresult]` is an array of the intensity of the hottest pixels in array

in similar way with only four cardinal directions neighbouring (found in the web): `import numpy as N` def `local_minima(array2d):`

```
    return ((array2d <= np.roll(array2d, 1, 0)) & (array2d <= np.roll(array2d, -1, 0)) & (array2d <=
            np.roll(array2d, 1, 1)) & (array2d <= np.roll(array2d, -1, 1)))
```

WARNING: flat top peak are not detected !!

`LaueTools.imageprocessing.gauss_kern` (*size, sizey=None*)  
 Returns a normalized 2D gauss kernel array for convolutions

`LaueTools.imageprocessing.blur_image` (*im, n, ny=None*)

**blurs the image by convolving with a gaussian kernel of typical** size *n*. The optional keyword argument *ny* allows for a different size in the y direction.

`LaueTools.imageprocessing.blurCCD` (*im, n*)  
 apply a blur filter to image ndarray

`LaueTools.imageprocessing.circularMask` (*center, radius, arrayshape*)  
 return a boolean ndarray of elem in array inside a mask

`LaueTools.imageprocessing.compute_autobackground_image` (*dataimage, boxsize=10, filter=10*)  
 return 2D array of filtered data array :param dataimage: array of image data :type dataimage: 2D array

`LaueTools.imageprocessing.compute_filtered_image` (*dataimage*, *bkg\_image*, *CCDlabel*,  
 $\text{kernel\_size}=5$ ,  $\text{formula\_expression}='A-B'$ ,  
 $\text{usemask}=\text{True}$ )

return 2D array of initial image data without background given by *bkg\_image* data

**usemask** [True then subtract bkg image on masked raw data] False apply formula on all pixels (no mask)

#### Parameters

- **dataimage** (2D array) – array of image data
- **bkg\_image** (2D array) – array of filtered image data (background)
- **CCDlabel** (string) – key for CCD dictionary

`LaueTools.imageprocessing.filter_image` (*image\_array*, *framedim*, *blurredimage=None*, *kernel-*  
 $\text{size}=5$ , *mask\_parameters=None*, *clipvalues=None*,  
 $\text{imageformat}=\langle \text{class 'numpy.uint16'} \rangle$ )

compute a difference of images inside a region defined by a mask

#### Parameters

- **blurredimage** – ndarray image to subtract to *image\_array*
- **kernel\_size** – pixel size of gaussian kernel if *blurredimage* is None
- **mask\_parameters** – circular mask parameter: center=(x,y), radius, value outside mask

`LaueTools.imageprocessing.rebin2Darray` (*inputarray*, *bin\_dims*=(2, 2), *operator*='mean')

rebin 2D array by applying an operator to define the value of one element from the other

#### Parameters

- **operator** – mean, min, max, sum
- **bin\_dims** – side sizes of binning. (2,3) means 2X3

`LaueTools.imageprocessing.blurCCD_with_binning` (*im*, *n*, *binsize*=(2, 2))

blur the array by rebinning before and after applying the filter

`LaueTools.imageprocessing.filter_minimum` (*im*, *boxsize*=10)

return filtered image using minimum filter

`LaueTools.imageprocessing.remove_minimum_background` (*im*, *boxsize*=10)

remove to image array the array resulting from *minimum\_filter*

`LaueTools.imageprocessing.plot_image_markers` (*image*, *markerpos*, *position\_definition*=1)

plot 2D array (image) with markers at first two columns of (*markerpos*)

---

**Note:** used in LaueHDF5. Could be better implementation in some notebooks

---

`LaueTools.imageprocessing.apply_formula_on_images` (*A*, *B*,  $\text{formula\_expression}='A-B'$ ,  
 $\text{SaturationLevel}=\text{None}$ ,  $\text{clipintensities}=\text{True}$ )

calculate image data array from math expression

#### Parameters

- **B** (*A*,) – ndarray of the same shape
- **SaturationLevel** – saturation level of intensity
- **clipintensities** – clip resulting intensities to zero and saturation value

## **7.4 Modules for Laue Pattern Indexation**

## **7.5 Modules for Crystal unit cell refinement**

## **7.6 Modules for batch processing**

## CHAPTER 8

---

### Indices and tables

---

- `genindex`
- `modindex`
- `search`



### I

`LaueTools.CrystalParameters`, [61](#)  
`LaueTools.imageprocessing`, [89](#)  
`LaueTools.IOimagefile`, [85](#)  
`LaueTools.lauecore`, [67](#)  
`LaueTools.LaueGeometry`, [77](#)  
`LaueTools.multigrainsSimulator`, [83](#)





## A

Add\_Images() (in module *LaueTools.IOimagefile*), 89  
 Add\_Images2() (in module *LaueTools.IOimagefile*), 89  
 AngleBetweenNormals() (in module *LaueTools.CrystalParameters*), 65  
 applyformula\_on\_images() (in module *LaueTools.imageprocessing*), 95

## B

blur\_image() (in module *LaueTools.imageprocessing*), 94  
 blurCCD() (in module *LaueTools.imageprocessing*), 94  
 blurCCD\_with\_binning() (in module *LaueTools.imageprocessing*), 95

## C

calc\_B\_RR() (in module *LaueTools.CrystalParameters*), 65  
 calc\_uflab() (in module *LaueTools.LaueGeometry*), 78  
 calc\_uflab\_trans() (in module *LaueTools.LaueGeometry*), 79  
 calc\_xycam() (in module *LaueTools.LaueGeometry*), 79  
 calc\_xycam\_transmission() (in module *LaueTools.LaueGeometry*), 79  
 calcSpots\_fromHKLlist() (in module *LaueTools.lauecore*), 71  
 check\_array\_indices() (in module *LaueTools.imageprocessing*), 90  
 circularMask() (in module *LaueTools.imageprocessing*), 94  
 compute\_autobackground\_image() (in module *LaueTools.imageprocessing*), 94  
 Compute\_data2thetachi() (in module *LaueTools.LaueGeometry*), 81  
 computefilteredimage() (in module *LaueTools.imageprocessing*), 95

Construct\_GrainsParameters\_parametric() (in module *LaueTools.multigrainsSimulator*), 83  
 convert2corfile() (in module *LaueTools.LaueGeometry*), 81  
 convert2corfile\_fileseries() (in module *LaueTools.LaueGeometry*), 82  
 convert2corfile\_multiprocessing() (in module *LaueTools.LaueGeometry*), 82  
 convert\_xycam\_from\_sourceshift() (in module *LaueTools.LaueGeometry*), 82  
 ConvolvebyKernel() (in module *LaueTools.imageprocessing*), 92  
 create\_spot() (in module *LaueTools.lauecore*), 69  
 create\_spot\_np() (in module *LaueTools.lauecore*), 69

## D

DeviatoricStrain\_LatticeParams() (in module *LaueTools.CrystalParameters*), 66  
 diff\_pix() (in module *LaueTools.imageprocessing*), 90  
 dosimulation\_parametric() (in module *LaueTools.multigrainsSimulator*), 83

## F

filter\_minimum() (in module *LaueTools.imageprocessing*), 95  
 FilterHarmonics\_2() (in module *LaueTools.CrystalParameters*), 65  
 filterimage() (in module *LaueTools.imageprocessing*), 95  
 filterLaueSpots() (in module *LaueTools.lauecore*), 69  
 from\_qunit\_to\_twchi() (in module *LaueTools.LaueGeometry*), 80  
 from\_twchi\_to\_q() (in module *LaueTools.LaueGeometry*), 80  
 from\_twchi\_to\_qunit() (in module *LaueTools.LaueGeometry*), 80

## G

gauss\_kern() (in module *LaueTools.imageprocessing*), 94  
 genHKL\_np() (in module *LaueTools.lauecore*), 67  
 get2ThetaChi\_geometry() (in module *LaueTools.lauecore*), 70  
 get\_imagesize() (in module *LaueTools.IOimagefile*), 89  
 getExtrema() (in module *LaueTools.imageprocessing*), 91  
 getfilename() (in module *LaueTools.IOimagefile*), 85  
 getIndex\_fromfilename() (in module *LaueTools.IOimagefile*), 85  
 getindices2cropArray() (in module *LaueTools.imageprocessing*), 89  
 getIntegratedIntensity() (in module *LaueTools.imageprocessing*), 91  
 getLaueSpots() (in module *LaueTools.lauecore*), 68  
 getMinMax() (in module *LaueTools.imageprocessing*), 91  
 getpixelValue() (in module *LaueTools.IOimagefile*), 86  
 getwildcardstring() (in module *LaueTools.IOimagefile*), 86  
 GrainParameter\_from\_Material() (in module *LaueTools.CrystalParameters*), 61

## L

*LaueTools.CrystalParameters* (module), 61  
*LaueTools.imageprocessing* (module), 89  
*LaueTools.IOimagefile* (module), 85  
*LaueTools.lauecore* (module), 67  
*LaueTools.LaueGeometry* (module), 77  
*LaueTools.multigrainsSimulator* (module), 83  
 lengthInSample() (in module *LaueTools.LaueGeometry*), 82  
 localmaxima() (in module *LaueTools.imageprocessing*), 94  
 LocalMaxima\_from\_thresholdarray() (in module *LaueTools.imageprocessing*), 94  
 LocalMaxima\_KernelConvolution() (in module *LaueTools.imageprocessing*), 92  
 LocalMaxima\_ndimage() (in module *LaueTools.imageprocessing*), 92  
 LocalMaxima\_ShiftArrays() (in module *LaueTools.imageprocessing*), 93  
 LoG() (in module *LaueTools.imageprocessing*), 91  
 LoGArr() (in module *LaueTools.imageprocessing*), 91

## M

matrix\_to\_rlat() (in module *LaueTools.CrystalParameters*), 67

minmax() (in module *LaueTools.imageprocessing*), 90  
 minmax\_fast() (in module *LaueTools.imageprocessing*), 91

## N

normalize\_shape() (in module *LaueTools.imageprocessing*), 91

## P

plot\_image\_markers() (in module *LaueTools.imageprocessing*), 95  
 Prepare\_Grain() (in module *LaueTools.CrystalParameters*), 63

## Q

Quicklist() (in module *LaueTools.lauecore*), 67  
 qvector\_from\_xy\_E() (in module *LaueTools.LaueGeometry*), 80

## R

radialArr() (in module *LaueTools.imageprocessing*), 92  
 Read\_GrainListparameter() (in module *LaueTools.multigrainsSimulator*), 83  
 read\_header\_marccd() (in module *LaueTools.IOimagefile*), 86  
 read\_header\_marccd2() (in module *LaueTools.IOimagefile*), 86  
 read\_header\_scmos() (in module *LaueTools.IOimagefile*), 86  
 read\_motorsposition\_fromheader() (in module *LaueTools.IOimagefile*), 86  
 readCCDimage() (in module *LaueTools.IOimagefile*), 87  
 readheader() (in module *LaueTools.IOimagefile*), 86  
 readoneimage() (in module *LaueTools.IOimagefile*), 87  
 readoneimage\_band() (in module *LaueTools.IOimagefile*), 87  
 readoneimage\_crop() (in module *LaueTools.IOimagefile*), 88  
 readoneimage\_crop\_fast() (in module *LaueTools.IOimagefile*), 88  
 readoneimage\_full() (in module *LaueTools.IOimagefile*), 86  
 readoneimage\_manycrops() (in module *LaueTools.IOimagefile*), 88  
 readrectangle\_in\_image() (in module *LaueTools.IOimagefile*), 88  
 rebin2Darray() (in module *LaueTools.imageprocessing*), 95  
 remove\_minimum\_background() (in module *LaueTools.imageprocessing*), 95

## S

`setfilename()` (in module *LaueTools.IOimagefile*), [85](#)  
`shiftarrays_accum()` (in module *LaueTools.imageprocessing*), [94](#)  
`SimulateLaue()` (in module *LaueTools.lauecore*), [71](#)  
`SimulateLaue_full_np()` (in module *LaueTools.lauecore*), [73](#)  
`SimulateResult()` (in module *LaueTools.lauecore*), [75](#)  
`stringint()` (in module *LaueTools.IOimagefile*), [85](#)  
`SumImages()` (in module *LaueTools.IOimagefile*), [89](#)

## T

`to8bits()` (in module *LaueTools.imageprocessing*), [90](#)

## U

`uflab_from2thetachi()` (in module *LaueTools.LaueGeometry*), [79](#)  
`unit_q()` (in module *LaueTools.LaueGeometry*), [81](#)

## V

`vec_normalTosurface()` (in module *LaueTools.LaueGeometry*), [82](#)  
`vec_onsurface_alongys()` (in module *LaueTools.LaueGeometry*), [82](#)  
`VolumeCell()` (in module *LaueTools.CrystalParameters*), [66](#), [67](#)

## W

`write_rawbinary()` (in module *LaueTools.IOimagefile*), [89](#)  
`writeimage()` (in module *LaueTools.IOimagefile*), [89](#)