PUC Rio



# IMAGE PROCESSING TECHNIQUES FOR NANOPARTICLES CHARACTERISATION

**Nota Técnica CBPF-NT-015/04**

**Authors:**
**Carole Jonville (ENSPG),**
**Ivan Guillermo Solórzano-Naranjo (PUC Rio),**
**Márcio Portes de Albuquerque (CBPF) and**
**Marcelo Portes de Albuquerque (CBPF)**

**August 2004**

# INTRODUCTION

In this report is presented image processing techniques for nanoparticles characterization developed as part of the *Projet de Fin d´Etudes* (Final Project) for ENSPG of Carole Jonville. This work was accomplished at CBPF in LPDSI (Laboratório de Processamento de Sinais e Imagens) of the CAT group (Coordenação de Atividades Técnicas).

First, an attempt is made to create an image processing algorithm that extracts information about size and form characteristics from images showing particles. It could be used, for instance, for Bright Field images (BF) from the Transmission Electron Microscope (TEM) that present nanoparticles.

Then, another more complete Matlab program was elaborated in the aim of analysing atomic lattice characteristics from High Resolution Electron Microscopy images (HREM, also from the TEM). The Lattice Parameter Analyser (LPA) uses the Radial Distribution Function (RDF) to obtain a distance histogram where peaks reveal the distances between first neighbours of the atomic lattice. To test the LPA, a Lattice Generator (LG) was made, it creates simulated lattices. Finally, the LPA is used to try and identify an atomic plane of a $TiO_2$ nanoparticle featured on a HREM image.

# PART I: AUTOMATIC SIZE REPARTITION PROGRAM

In order to try to calculate automatically statistical values of characteristics from images showing particles (e.g. Bright Field images from Transmission electron Microscope), an image processing algorithm is described.

To avoid the need of the particle contour determination, it was attempted to develop an automatic tool for detecting particles from the background. The objective was to create a program that detected the particles in the image and calculated their characteristics (e.g. area, size, form factor…). It also aimed at leaving a tool that other scientists could use in case of nanopowder characterisation. Such program would turn the statistic analysis of the TEM pictures fast and automatic. A Matlab algorithm was elaborated with those aims.

## *1. Program Description*

Digital image processing is divided into basic steps:

- Acquisition of the image: capturing the image and digitalisation if necessary.
- Pre processing: preparing the image before the segmentation.
- Segmentation: obtaining a black and white image separating the objects from the background.
- Attributes extraction: computing of the desired characteristics of the image.
- Interpretation.

In the following paragraphs, more details are given about the different steps. To illustrate the program an image showing rice grains will be used (Figure 1).

### 1.1. Pre Processing

#### Background illumination correction:

It can be observed on the original image (Figure 1) that the bottom of the picture is darker than the top. This may cause problem when the thresholding will be done to detect the objects. It is necessary to correct the background illumination.

First the image is divided into small squares (32 by 32 pixels) and the program computes the minimum value (the blackest pixel) in each square. Then it creates an image with the size of the original one with these values. Finally, this background representation image is subtracted to the original image. Figure 2 shows the corrected picture.

#### Contrast correction:

Because there has been a subtraction, the value of the intensity has lowered, thus the image is blacker. It is necessary to adjust the contrast. This is done with the `imadjust` Matlab function. It rescales the image intensity redistributing the pixel intensity values between the minimum and the maximum intensity value. The result can be observed in Figure 3.

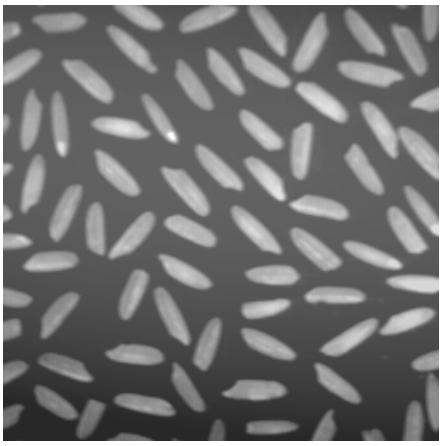


**Figure 1: Original image.**

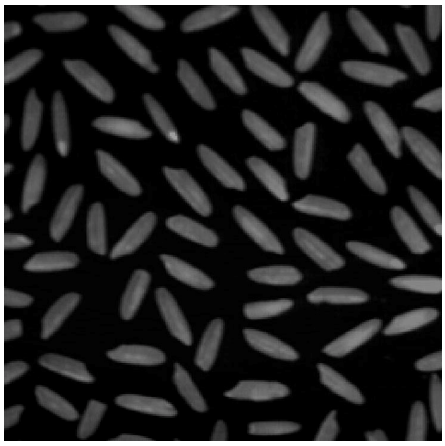


**Figure 2: Image with background illumination corrected.**

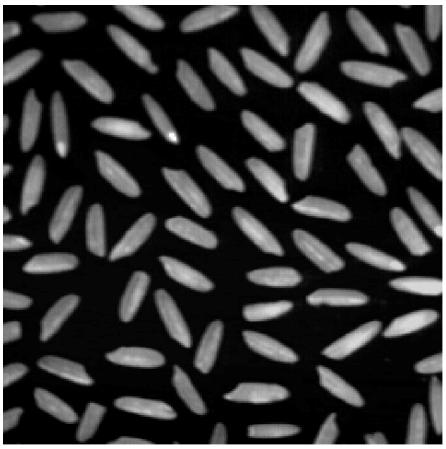


**Figure 3: Image with contrast adjusted.**

### 1.2. Segmentation

This step is done by the Matlab function `graythresh`, it detects the correct threshold level by the Otsu method [ref 1]. Then the segmentation is done with the `im2bw` function. Figure 4 shows the segmented image.

The program has correctly detected the objects from the background. However, nothing guaranties that for every picture, the level chosen by the function will be correct. That is why the segmented picture will be shown at the end to the user so that he can check that the binary image displays the particles. A possible improvement of this step would be an option that lets the user try different levels of threshold until obtaining a satisfactory result.

An important step is to remove the objects that touch the border. Indeed, they are not complete objects and can reduce the accuracy of the measurements of object characteristics. This is done by the Matlab function `imclearborder`. The result can be observed in Figure 5.

### 1.3. Attribute Extraction

First, the program detects and labels each object in the picture. Then with the `imfeature` function, the following parameters are extracted and kept in an array:

- Area.
- Equivalent diameter: computes the diameter of a disk that has the same area.
- Centroid: keeps the centre of mass of the object.
- Image: keeps the image of each object individually.

Note that all these characteristics are measured in number of pixels (or square pixels). The scale has to be put into the program to have the results in nanometres.

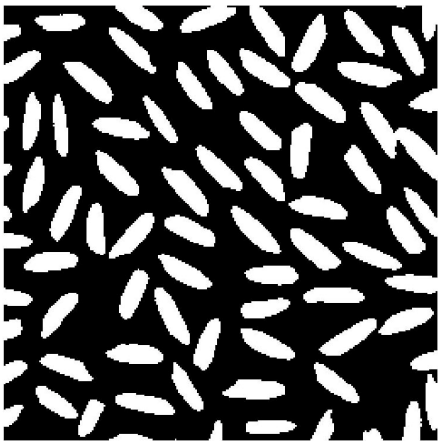


**Figure 4: Segmented image.**

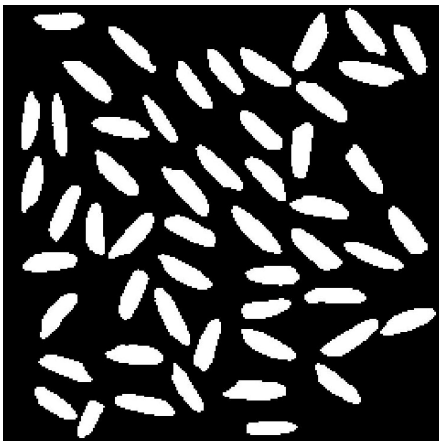


**Figure 5: Image with border elements removed.**

Because there is no automatic way to obtain the perimeter of an object, it is necessary to use an indirect way. From each individual object (obtained with "image") a black pixel is added to the borders of the box. Then the `bwperim` function transforms the image keeping only the edge of the object (if no extra pixels had been added, the function would be unable to detect the object in the box). The perimeter is obtained by the `imfeature` function computing the area (with the option of a connectivity of 8 because in this case the pixels that share corners are considered from the same object).

The mean particle's equivalent diameter and its normal deviation can be calculated with Matlab. It also computes a form factor (expression 1, where A is the object's area and P its perimeter).

$$f = \frac{4pA}{P^2} \tag{1}$$

### 1.4. Finishing

The program displays the results of the computing: average particle size and area and their normal deviations, the time taken for the computing and the relevant images (original image and segmented image).

If the user wishes to do further analysis or obtain the histogram from another program, he can use a file containing the computed characteristics saved by the program. The file is automatically named after the original image name and contains for each object: the area, the equivalent diameter and the f form factor.

## *2. Results*

This program works well with the example of the rice and other pictures where the objects to be measured are separated. However, it is difficult to analyse BF TEM pictures for two reasons.

Although it is a good quality TEM image, the contrast shown is very complex for the particle detection. First, as can be observed in Figure 6 (showing agglomerations of $TiO_2$ nanoparticles), there is little difference between the background and the light particles' intensity. Then, there is a big difference of intensity from different particles. Because some of them are

diffracting the electrons from the TEM beam, the transmitted beam is weaker and they appear blacker. This difference also turns the detection more difficult.

Nevertheless, the most important problem lies in the fact that there is a lot of particle superposition. When the program does the segmentation step, it is impossible to separate the particles. It finds objects that are several agglomerated particles. Therefore the characteristic analysis obtained with this program is irrelevant. The result of the automatic segmentation of the described algorithm is shown in Figure 7. It can be observed that the detected objects do not correspond to the particles.

Other ideas were used to try and better the critical segmentation step. One was to de double threshold segmentation; another was to use the watershed method [ref 2]. But the results were not satisfying.

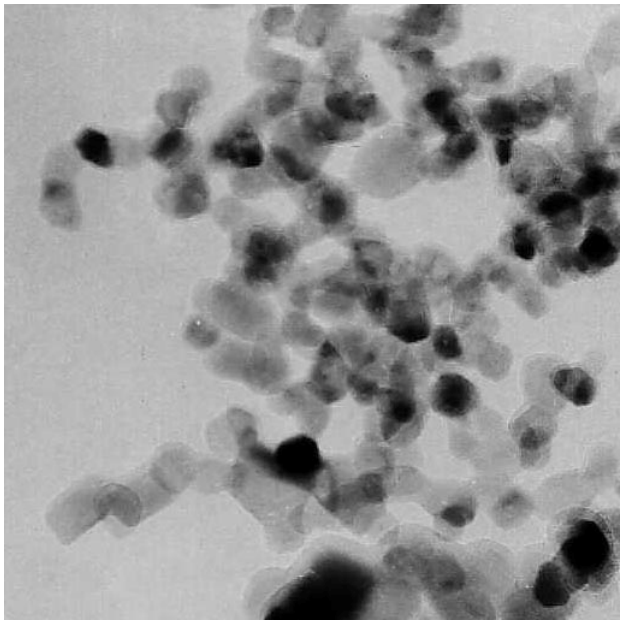


**Figure 6: Cropped image from a BF image.**

**Figure 7: Automatic segmentation of the BF image.**

## *3. Conclusion*

An algorithm was developed under Matlab, it permits to extract the desired characteristics of objects. However, the BF TEM pictures showing the nanoparticles proved to be too complex to be analysed by this program. That is why the program was left at an incomplete state.

# PART II: LATTICE CHARACTERISTICS FROM HIGH RESOLUTION ELECTRON MICROSCOPY ANALYSIS

Another important part of the project consisted in analysing the crystal structure of the $TiO_2$ particles. For this, the HREM images were studied.

From a direct image analysis method, the HREM images showed nanoparticles of size from 8 to 28 nm in diameter. It can also be observed that they are almost defect free single crystalline particles which is an expected result for such small particle size. A Matlab image processing algorithm is presented as a new technique for computing statistical values for the atomic lattice characteristics.

## *1. Direct Method*

### 1.1. Introduction

The first technique used to extract information about the $TiO_2$ nanoparticle's lattice properties was direct. It consisted in measuring plane separation distances and angles "by hand", using measurement tools with Scion Image.

The measurements were done on the mf9415 micrograph (Figure 8) because it was the picture with the best quality: the atoms were identifiable in two directions.

### 1.2. Results

The values of the atomic raw separation (d) in the two directions and angle between them are in Table 1. They result from an average made with several measurements, nevertheless they cannot be considered as statistical. That is why it is not possible to say if the difference found in the parameters in the two directions is relevant. The accuracy is at minimum of the spatial resolution (0.03 nm).

| Direction 1 | Direction 2 | Angle |
|---|---|---|
| d = 0.3509 nm | d = 0.3549 nm | 97.7º |

**Table 1: Result of the direct measurement method.**

It seems that the distances measured on the image could correspond to the first parameter of the anatase system (a = 3.7820 Å from crystallographic values of the refined Rietveld parameter analysis, see Ref 8). However, the values are 7% smaller. Moreover, the measured angle do not correspond to a simple plane (such as (0,0,1) for example).

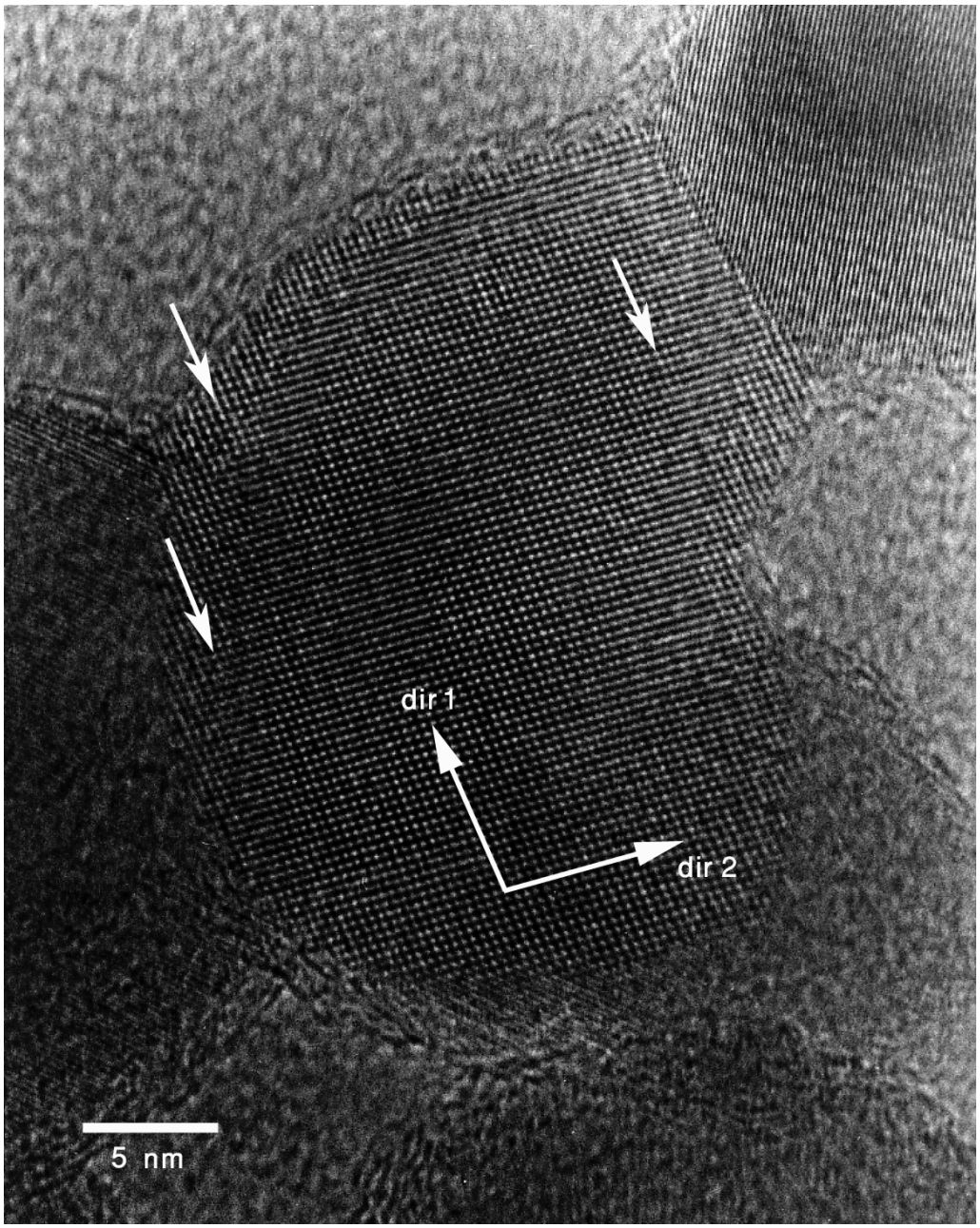



**Figure 8: The mf9415 HREM picture showing atoms rows in 2 directions. The arrows show dislocations (crystal defects).**

It was expected that the particles were defect free because of their small size, however the central particle of Figure 8 presented dislocations. The arrows indicate the positions of identified dislocations. Figure 9 is a detail of the picture where the atomic rows have been coloured in red to make the dislocation plane appear.

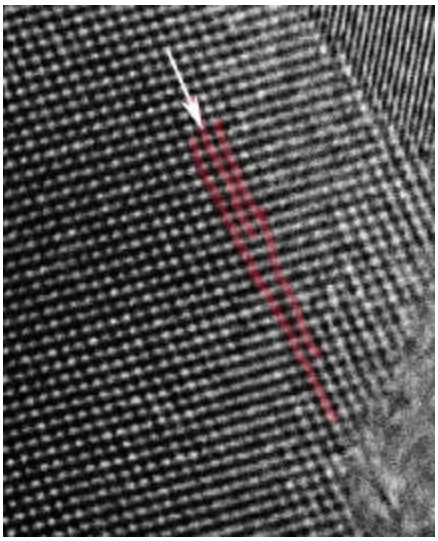


**Figure 9: Detail of mf9415 showing a dislocation.**

### 1.3. Conclusion

The particles were found to be almost defect free single crystalline particles. Nevertheless, obtaining a good estimation of the lattice's characteristics with this direct method in not obvious. It is also a tedious work to obtain statistical values of the plane separation distances. There was not enough time to explore the methods of complex atomic plane determination, that is why the atomic plane showed was not identified. More details about the attempts of lattice plane identif ication will be given in Part II.

## *2. Automation of the Analysis of HREM Images*

In this part, a Matlab program called Lattice Parameter Analyser (LPA) is described. Its aim is to give distances between first neighbours from a high resolution electron micrograph. The LPA is also characterised using simulated lattices generated by another elaborated program: Lattice Generator (LG). Finally, the LPA is used for the analysis of the HREM images from our $TiO_2$ sample (see ref 8).

### 2.1. Lattice Generator

To be able to test the Lattice Parameter Analyser, a program was elaborated. The Lattice Generator (LG) creates images with perfect lattices, thus simulating the pictures that can be obtained with the TEM in HREM conditions. Indeed, it is necessar y to check that the LPA program gives correct results with images for which the lattice parameters are perfectly known.

#### 2.1.1. Program Input

The LG program reads a configuration file (text file) containing information about the lattice the user wants to create and then generates it. The information contained are:

- The image size in pixel
- The lattice parameters a and b in pixels
- An option if an hexagonal lattice is desired
- An angle value to rotate the lattice

### 2.1.2. Program Output

The LG program creates the desired lattice and saves it as a tif format picture. The picture's name is automatically given according to the parameters (a, b and angle values).

### 2.1.3. Program Description

First, the LG reads and converts the parameters contained in the configuration file. It then creates a black picture of the specified size. It puts a white dot at a regular interval given by the parameters specified by the user. It creates "atoms" from the dot applying a dilatation [ref 2], then the resulting image is blurred so that it looks like a real atomic resolution picture. The image is enlarged before being rotated of the specified angle value. Finally the rotated image is cropped to the desired image size. If the user set the hexagonal option, a special function is called to generate the lattice.

### 2.1.4. Results

This program works correctly. Figure 10 and Figure 11 are examples of generated lattices.

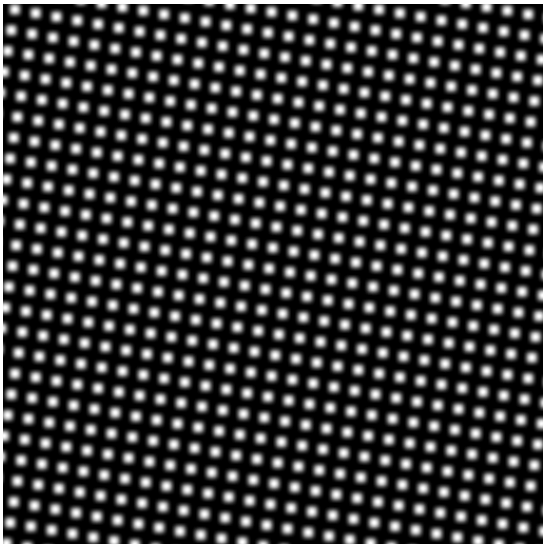


**Figure 10: Example of created image: square lattice of 512x512 pixels with a parameter of 20 pixels and an angle of 10 degrees.**

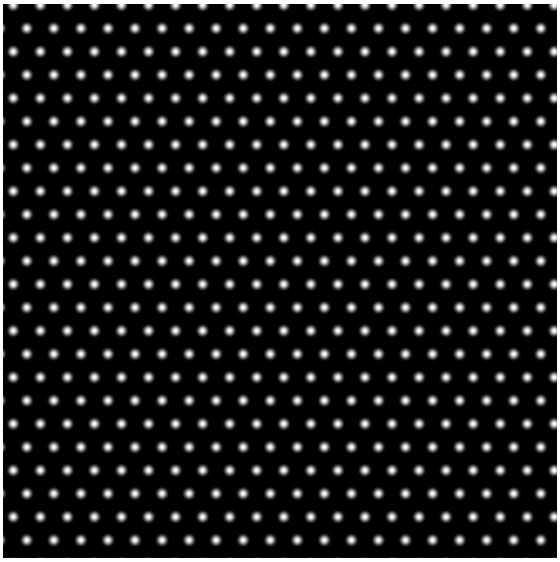


**Figure 11: Example of created image: hexagonal lattice of 512x512 pixels with a parameter of 50 pixels and no rotation.**

### 2.1.5. Conclusion

A program was developed to serve as a generator of test images to check the LPA program. The user does not need to understand Matlab coding or how it functions to use it. What is simply required is to fill the desired parameters in the configuration file and to run the main program. The generated image will be automatically created and saved. The user can use the ReadMe file to have a brief description of LG, the information about the required software to run it and indications to know how to use it.

The LG complete code can be found in Appendix A. The LG program can be downloaded with the ReadMe file and a default configuration file at http://www.cbpf.br/cat/lpdsi/lpa.

## 2.2. Lattice Parameter Analyser

The LPA computes the distances between all atoms in a defined neighbourhood area through a Radial Distribution Function (RDF). The program outputs a histogram of such distances, it will feature peaks corresponding to the first neighbour's distances. It is a powerful tool since the information extracted is statistical.

### 2.2.1. RDF Method

The RDF is a method to obtain an average notion of a structure [ref 7]. The RDF addresses the question: "Given that I have one atom at some position, how many atoms can I expect to find at a distance r away from it?" (Figure 12). An example of an application of the method is the determination of patterns in the formation of magnetic bubbles in ferrofluids [ref 3].

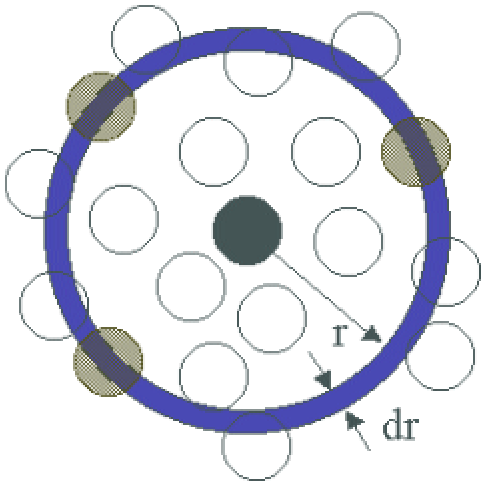



**Figure 12: Illustration of the RDF technique.**

### 2.2.2. Input

The inputs are a picture and, as with the LG program, a configuration file. It specifies relevant parameters to be set in order to obtain a correct analysis. The analysed image must be a good quality atomic resolution picture where the atoms are separated. The configuration file must specify the following parameters:

- The image name (with the extension).
- The image scale: it is the spatial resolution, in other words, the distance value of one pixel.
- The neighbourhood value: the distance in nanometres for which the distance analysis (RDF) will be done.
- FlagPrint: if set to 0, the program will run in silence, if set to 1, it will show the running status and intermediary results (graphs, data…)

It also contains additional parameters that can be modified to obtain an optimization of the analysis:

- The RDF Threshold: parameter used in the detection of the peaks in the histogram of the distances.
- The RDF Filter Order: order of the filter that smoothes the histogram to help doing a Gaussian fit of the curve.

However, the analysis works well in general with the default values.

### 2.2.3. Output

The LPA shows the results in the Matlab window and saves relevant information in different forms:

- The segmented image
- The list of all the distances computed by LPA
- The histogram graph
- The peak positions and standard deviations

The output files allows the user not only to check that the analysed image was correct and to have the distances' list to do an independent analysis but also to have all the results saved in the form of a complete graph and a text file.

### 2.2.4. Program Description

The program is divided into steps which correspond to a function calling of the main program. The flow chart of the LPA can be found in Figure 13 and the complete code of the program's functions can be found in Appendix B.

#### Reading the Configuration file: ReadConfigFile.m

This function is the part of the program that reads the parameters in the configuration file and converts the text strings into correct units.

#### Processing the Image: ProcessImage.m

The image processing step (Table 2) is basically the same as the pre processing and segmentation steps in the size distribution analysis program .

| | |
|---|---|
| Adjusting background illumination | $I_{bkgd}$ = GaussFilter($I_{init}$)<br>$I_2 = I_{init} - I_{bkgd}$ |
| Adjusting the contrast | $I_3$ = RedistributeIntensity($I_2$) |
| Segmentation | Level = GrayThresh($I_3$)<br>$I_4$ = Thresholding($I_3$, Level) |
| Removing dots/noise | $I_5$ = MorphologicalOperation($I_4$) |
| Removing edge objects | $I_{final}$ = ImgClearBorder($I_5$) |

**Table 2: Steps of the ProcessImage sub program.**

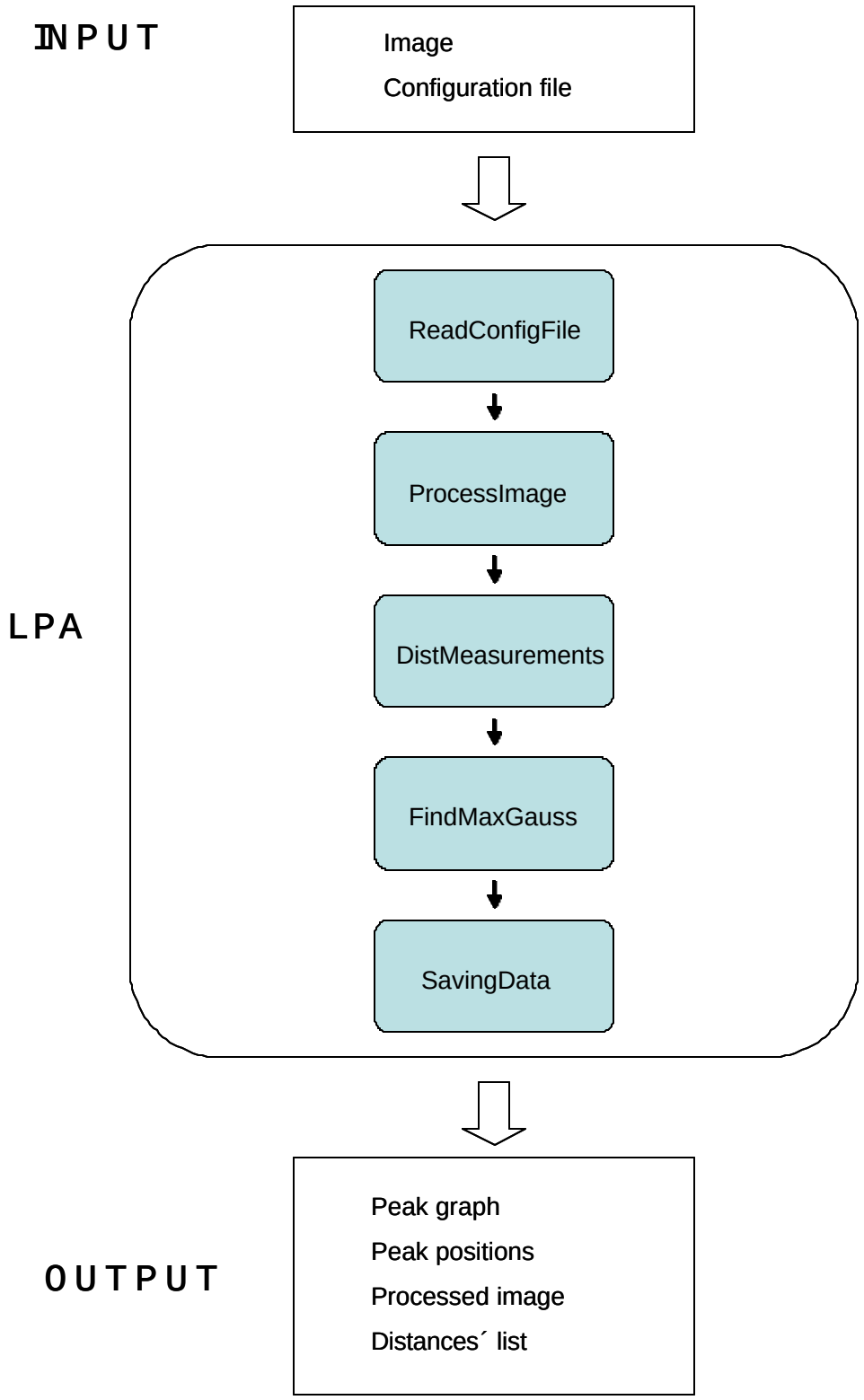



**Figure 13: Flow chart of the Lattice Parameter Analyser.**

Comments:

Adjusting illumination:

The method used here is different from the first program because it had a drawback: it was limited to images with a size in length and width dividable by 32. To solve this problem the image is filtered by Gaussian type of filter to obtain a background approximation [ref 5]. Then this calculated background is subtracted from the original image.

Morphological operation:

A basic morphologic operation is done on the segmented image. An opening [ref 2] is done to remove small dots that may appear after the segmentation that do not correspond to atoms on the original picture. An amelioration of the program could be to transform this morphological operation as a parameter in the configuration file that the user could modify to optimize the analysis.

## Computing the Distances: DistMeasurements.m

This part of the program (Table 3) first detects and labels each object of the processed image. It then extracts with the `imfeature` function the centre of mass of each atom. Finally, it computes the distances between atoms (RDF method). It uses a double loop algorithm: for each atom i in the picture, it computes for each atom j within the neighbourhood distance, the distance between the centres of mass of atoms i and j. All the distances calculated are kept in an array.

| | |
|---|---|
| Labelling object | `Objects = Label(`$I_{final}$`)` |
| Extracting centre of mass | `Centroid = Imfeature(Objects)` |
| Distance computation | `For i=1 to nb of all objects`<br>`  For j=1 to nb of objects in neighbourhood`<br>`  Compute distance between object i and j`<br>`  End`<br>`End` |

**Table 3: Steps of the DistMeasurements sub program.**

Comments:

In a first version of the LPA, there was not the neighbourhood concept for the distance computation. It calculated all the distances between all the atoms in the image. This computation took a lot if time (about 30 minutes for 1000 atoms) and it was necessary to use the Linux Cluster of the CBPF to run the program. The final version including the neighbourhood distance is much faster (about a minute for 1000 atoms) and it is relevant for our study because the objective was to extract the distances between the first neighbours in the atomic lattice.

## Finding the Peaks: FindMaxGauss.m

First, this sub program (Table 4) creates a histogram from the array of all the distances, then it smoothes the signal with a convolution technique and normalizes it. The critical step is finding the peaks in the histogram. A special Matlab function `imextendedmax` detects the peaks and their range in the histogram. A Gaussian fitting is accomplished for each detected peak. The position of the maximum of the curve and its standard deviation (width of the Gaussian curve) are calculated for each peak. The result is shown in a scaled graph (the distances are converted to nanometres with the scale given in the configuration file).

| | |
|---|---|
| Histogram | `Histogram = Hist(Distance, 500)` |
| Smooth signal | `Signal = Convolution(Histogram, Filter)` |
| Find peaks of the signal | `Peaks = Imextendemax(Signal)` |
| Gaussian fitting | `GaussPeaks = Gaussfit(Peaks)` |
| Extract information | `NeighDistance = Position(Max(GaussPeaks))`<br>`Deviation = std(GaussPeaks)` |
| Showing graph and results | `Plot(Histogram, GaussPeaks, NeighDistances)` |

**Table 4: Steps of the sub program FindMaxGauss.**

Comments:

Histogram creation:

The number of bins is fixed to 500 and Matlab estimates automatically the bin width. An optimization for the number of bins was tempted. An evaluation of the number of bins as a function of the neighbourhood distance and the scale was searched. However, there is no formula that works for all cases [ref 4] that is why the value of 500 was left as a default setting.

Convolution for signal smoothing:

The convolution is done between an array containing the histogram's counts and an array containing ones [ref 5]. The length of the second array determines the force of the smoothing. This length is a parameter that can be changed in the configuration file (RDF Filter Order), it is set to 10 as default. For example, it can be increased when the histogram is very rough; the effect will be to facilitate the peak detection and the Gaussian fitting.

Peak detection:

There is a parameter of the configuration file that the program takes into account when calling the `imextendedmax` function. The RDF Threshold is a sort of thresholding: it is the difference in height that will determine if two close peaks are separated or not. Its value is set to 0.05. If the user is not satisfied with the peak detection, he can try to raise the RDFTh value if he wants to obtain a better peak separation.

Gaussian fitting:

A trick for a correct Gaussian fitting is to enlarge the range given by the `imextendedmax` function. A good value of 2 bins on each side was found by testing various values and images.

### Saving the Outputs: SavingData.m

This function creates a directory after the original image's name. It will contain the 4 output files, also named automatically.

Peak position: The position and standard deviation of the peaks are saved in text file.

Neighbour's distances: All the distances computed in the DistMeasurement function are saved in a data file Therefore, the user has all the information needed if he wishes to plot the histogram with another program or run other analysis.

Segmented image: The processed image showing the detected objects is saved in a jpeg format. The user can then check if the image that was analysed for the distance computation was correct. Indeed, there may be cases of HREM image where the atoms are not well separated and

the computed distances will not correspond to the first neighbour's distance since the objects did not correspond to the atomic lattice. In addition to that, on one of the atoms, this image will show coloured circles that correspond to the first neighbour's distances. It is useful to visualise those distance on the atomic lattice.

Peak graph: A complete nanometre scaled and normalized graph is saved in a jpeg format. It contains:

- The bar histogram
- The Gaussian curve fitting each peak
- The peak position in nanometres on top of each one

#### 2.2.5. Conclusion

To accomplish this complete LPA program, a lot of optimization was done. Several methods were tried before obtaining the final version. It also contains various parameters that can be modified if the user wishes to reach the best results.

The distances evaluated by this program are obtained automatically and are useful for the determination of the atomic plane of HREM images. It is no longer necessary to measure "by hand" the lattice characteristics. Moreover, it is a powerful tool because its results are statistical. The user just needs to interpret the result to finally know which atomic plane is shown on the original HREM picture.

The LPA program can be downloaded with the ReadMe file for instructions and a default configuration file at http://www.cbpf.br/cat/lpdsi/lpa.

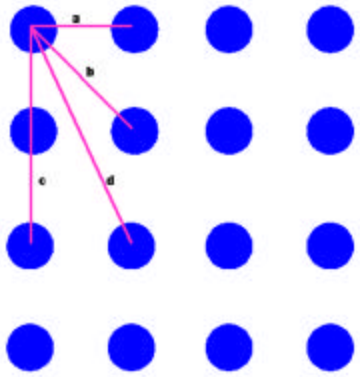
### 2.3. LPA Results with Simulated Lattices

An important step is to check that the elaborated program gives correct results before using it to analyse real HREM photos. The Lattice Generator program was used because, as the lattice characteristics are set, a simple comparison of the values of the first neighbours' distances can tell the accuracy of the LPA program.

#### 2.3.1. Square Lattices

First, the simplest lattice was tested. Square lattices with different lattice parameters and angles of rotation were made with the LG program. The scale was set to 1 pixel = 0.03 nm.

Figure 14 shows a representation of a square lattice and the values of the first neighbours' distances in function of the lattice parameter a.




- $b = a\sqrt{2}$
- $c = 2a$
- $d = a\sqrt{5}$

**Figure 14: Representation of a square lattice and the values of the first neighbours' distance in function of the lattice parameter a.**

Figure 15 shows an example of the output graph of the LPA program for a square lattice of 20 pixels created by the LG (similar to Figure 10). The bar histogram values are in green and in red are the Gaussian curves that fit the histogram's peaks. The positions written correspond to the first neighbours' distances computed by LPA. The standard deviations are not showed on the graph because it would over load it. They are given in the Matlab window and in the text file saved automatically.

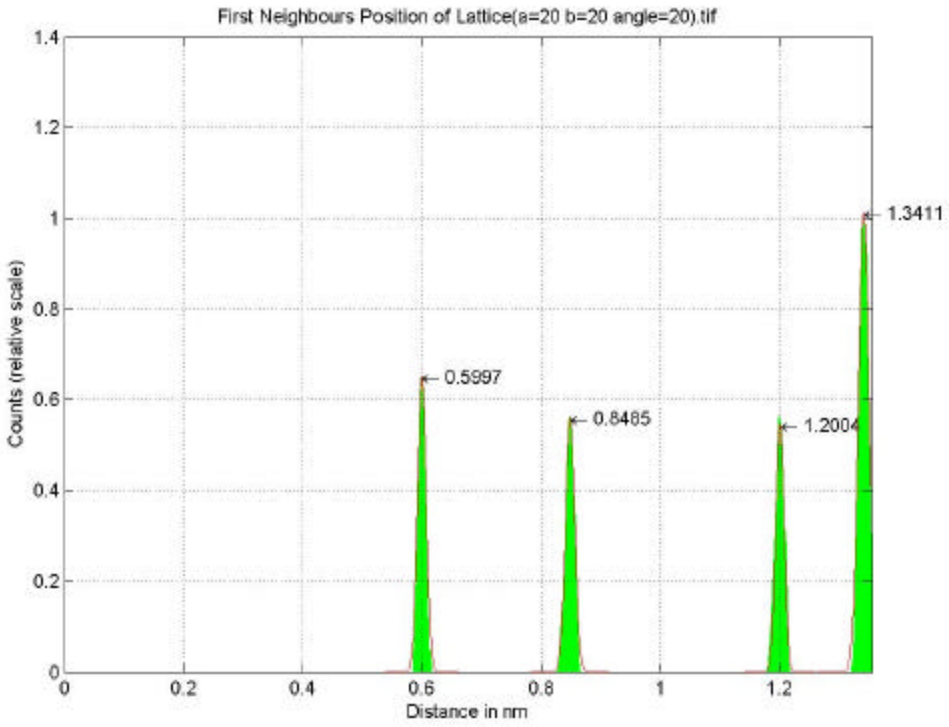



**Figure 15: Example of peak graph that is created and saved by the LPA program.**

The comparison of the theoretical values and the results given by the LPA are featured in Table 5. What can be observed is that the program always gives values very close to the theoretical values. In addition to that, the theoretical values are always included in the error range according to the standard deviations. This is proved by the value of $\div^2$ [ref 4]. Its formula is given in expression (2) where $N$ is the number of peaks taken into account, $X_i^{th}$ is the

theoretical value of the peak position, $X_i^{mes}$ is the measured peak position and *ó* is the standard deviation of the peak. When the value of ÷² is inferior to 1, the measured value is inside the error bar given by the standard deviation.

$$c^2 = \frac{1}{N}\sum_i^N \frac{\left(X_i^{mes} - X_i^{th}\right)^2}{s_i^2} \qquad (2)$$

The intensity of the peak cannot be taken into account for an extra analysis. Indeed, the convolution done to smooth the curve to facilitate the Gaussian fitting, multiplies the counts value.

The LPA program managed to find the correct neighbours' distances of all the square lattices tested and gives results with a considerable accuracy.

| | Distance in nm | | | | | |
|---|---|---|---|---|---|---|
| | a = 20 pixels = 0.6 nm | | | a = 30 pixels = 0.9 nm | | |
| Neighbour | Theory | Angle = 0 | Angle= 20 | Theory | Angle = 0 | Angle= 20 |
| 1 | 0.6000 | 0.5994 +/- 0.0060 | 0.5997 +/- 0.0073 | 0.9000 | 0.9077 +/- 0.0078 | 0.9014 +/- 0.0118 |
| 2 | 0.8485 | 0.8482 +/- 0.0057 | 0.8485 +/- 0.0078 | 1.2728 | 1.2712 +/- 0.0137 | 1.2703 +/- 0.0158 |
| 3 | 1.200 | 1.1996 +/- 0.0056 | 1.2004 +/- 0.0069 | 1.800 | 1.7966 +/- 0.0137 | 1.7976 +/- 0.0152 |
| 4 | 1.3416 | 1.3418 +/- 0.0071 | 1.3411 +/- 0.0090 | 2.0125 | 2.0121 +/- 0.0137 | 2.0097 +/- 0.0154 |
| ÷² | | 0.0047 | 0.0020 | | 0.2627 | 0.0243 |

| | Distance in nm | | | | | |
|---|---|---|---|---|---|---|
| | a = 40 pixels = 1.2 nm | | | a = 15 pixels = 0.45 nm | | |
| Neighbour | Theory | Angle = 0 | Angle= 20 | Theory | Angle = 0 | Angle= 10 |
| 1 | 1.2 | 1.2094 +/- 0.0095 | 1.2023 +/- 0.0139 | 0.4500 | 0.4510 +/- 0.0037 | 0.4516 +/- 0.0048 |
| 2 | 1.6971 | 1.6935 +/- 0.0168 | 1.6941 +/- 0.0178 | 0.6364 | 0.6360 +/- 0.0038 | 0.6351 +/- 0.0068 |
| 3 | 2.400 | 2.3977 +/- 0.0168 | 2.3976 +/- 0.0168 | 0.9000 | 0.9002 +/- 0.0037 | 0.8994 +/- 0.0099 |
| 4 | 2.6833 | 2.6780 +/- 0.0189 | 2.6802 +/- 0.0190 | 1.0062 | 1.0051 +/- 0.0043 | 1.0063 +/- 0.0047 |
| ÷² | | 0.2806 | 0.0257 | | 0.0381 | 0.0379 |

**Table 5: Table comparing the theoretical values of the first neighbours' distance and the values given by the Lattice Parameter Analyser.**

### 2.3.2. Rectangular Lattices

Some rectangular lattices were tested too. Figure 16 shows an example of the resulting graph in the case of a rectangular lattice (a=15 pixels and b=30 pixels as can be seen in the title). What can be observed is that the Gaussian fitting is correct for the peak position but over estimates the peak intensity. Indeed, because the analysed lattice is almost perfect (created by

the LG program), the histogram's peaks are very sharp, turning the Gaussian fitting difficult. However, this does not induce any error on the result since what is important are the peak positions that correspond to the first neighbour's distances of the lattice. The scale was set to 1 pixel = 0.03 nm.

Figure 17 shows a part of the image that the LPA program saves. It shows the segmented image resulting from the image processing of the original picture (rectangular lattice with a=15 pixels, b=30 pixels and an angle of 0). It can be observed that the detected objects correspond to the simulated atoms. The circles represent the distance computed by the program, they are alternatively green and red. It is a very practical way for the user to check the veracity of the program's results.

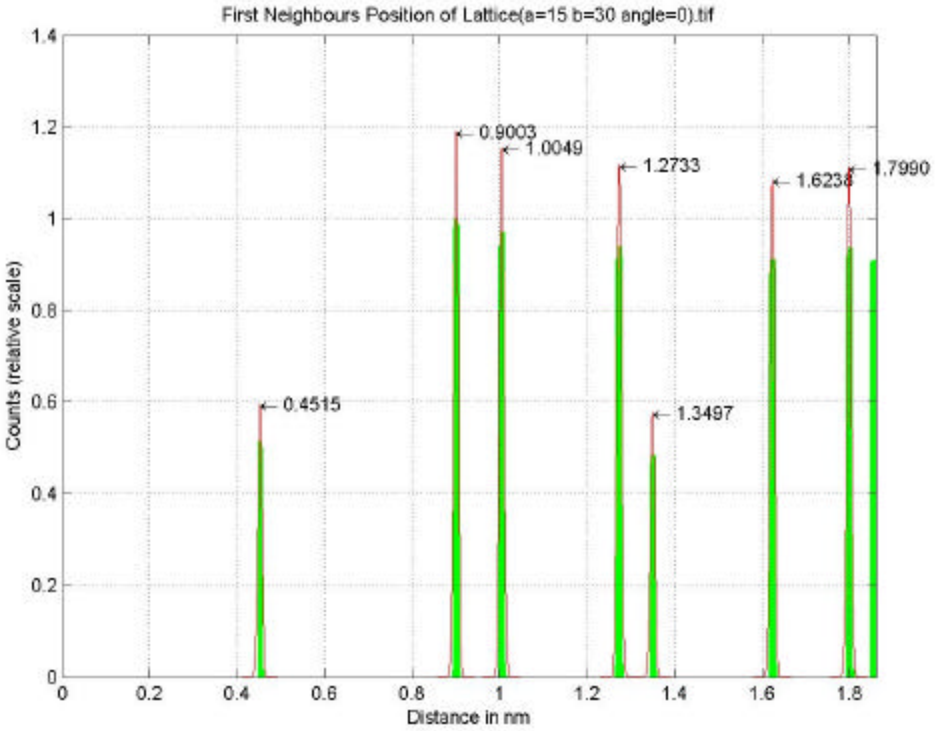



**Figure 16: An example of the pe ak graph in the case of a rectangular lattice.**

Figure 18 shows a representation of a rectangular lattice and the values of the first neighbours' distances in function of the lattice parameters a and b.

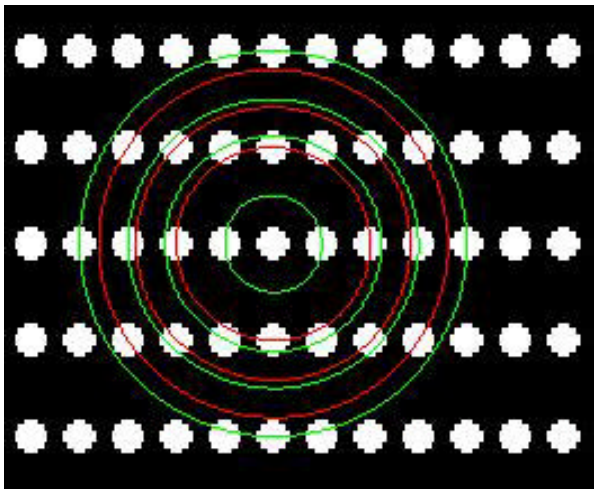


**Figure 17: A detail of the segmented image saved by LPA.**

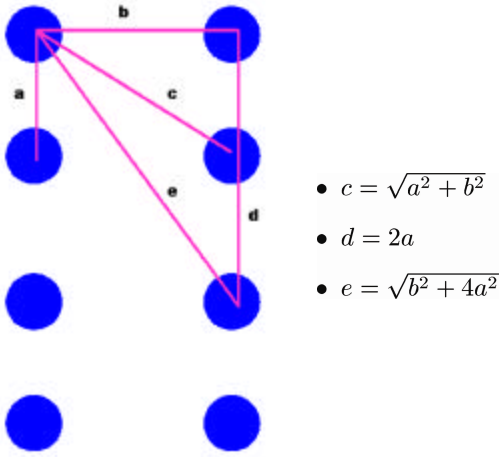



**Figure 18: Representation of a rectangular lattice and the values of the first neighbours' distance in function of the lattice parameters a and b.**

The comparison of the theoretical values and the results given by the LPA in this case are featured in Table 6. The first 5 neighbour's distances were analysed. The comments are the same as in the square lattices' case. The program always gives values very close to the theoretical values and the exact values are always within the error range (value of ÷² inferior to 1).

| | Distance in nm | | | |
|---|---|---|---|---|
| | a = 20 pixels = 0.6 nm<br>b = 30 pixels = 0.9 nm | | a = 15 pixels = 0.45 nm<br>b = 30 pixels = 0.90 nm | |
| Neighbour | Theory | Angle = 10 | Theory | Angle = 0 |
| 1 | 0.6 | 0.5986 +/- 0.0075 | 0.45 | 0.4515 +/- 0.0048 |
| 2 | 0.9 | 0.8992 +/- 0.0070 | 0.9 | 0.9003 +/- 0.0050 |
| 3 | 1.0820 | 1.0805 +/- 0.0091 | 1.0062 | 1.0049 +/- 0.0050 |
| 4 | 1.2 | 1.1994 +/- 0.0066 | 1.2728 | 1.2733 +/- 0.0050 |
| 5 | 1.5 | 1.4989 +/- 0.0077 | 1.35 | 1.3497 +/- 0.0050 |
| ÷² | | 0.0208 | | 0.0365 |

**Table 6: Table comparing the theoretical values of the first neighbours' distance and the values given by the Lattice Parameter Analyser.**

### 2.3.3. Hexagonal Lattice

Hexagonal lattices created with LG were also tested. A representation of a hexagonal lattice and the values of the distances in function of the lattice parameter a are given in Figure 19.

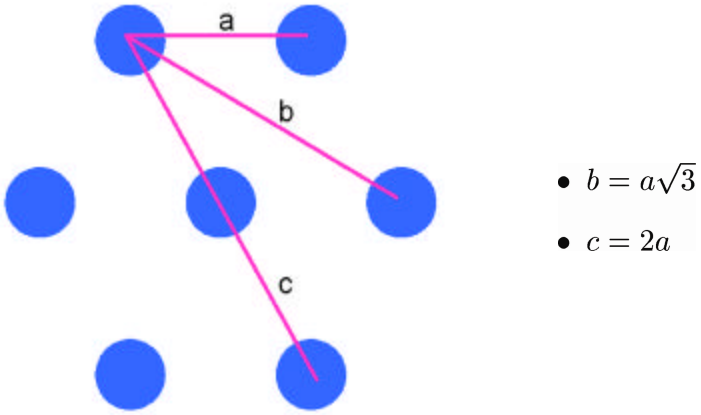



**Figure 19: Representation of a hexagonal lattice and the first neighbours' distance in function of the lattice parameter a.**

The comparison for the first 3 distances given by the LPA and the theoretical values are given in Table 7. The comments are identical: the LPA successfully finds the correct value with a good accuracy. However on the case of the simulated lattice of 50 pixels parameter, the $\div^2$ is found slightly superior to 1. This is probably due to the fact that the histogram has extremely thin peaks, turning the Gaussian fitting is more delicate.

| | Distance in nm | | | |
|---|---|---|---|---|
| | a = 30 pixels = 0.9 nm | | a = 50 pixels = 1.5 nm | |
| Neighbour | Theory | Angle = 10 | Theory | Angle = 0 |
| 1 | 0.9 | 0.8996 +/- 0.0051 | 1.5 | 1.4955 +/- 0.0093 |
| 2 | 1.5588 | 1.5595 +/- 0.0046 | 2.5981 | 2.5869 +/- 0.0104 |
| 3 | 1.8 | 1.8000 +/- 0.0051 | 3.0 | 2.9880 +/- 0.0092 |
| $\div^2$ | | 0.0098 | | 1.0317 |

**Table 7: Comparison for some hexagonal lattices.**

### 2.3.4. Verification with Real Lattices

An old HREM picture of $TiO_2$ was analysed with the LPA program. The analysed picture can be seen in Figure 20. However the scale for this picture was not found, it was set to 1 pixel = 0.03 nm.

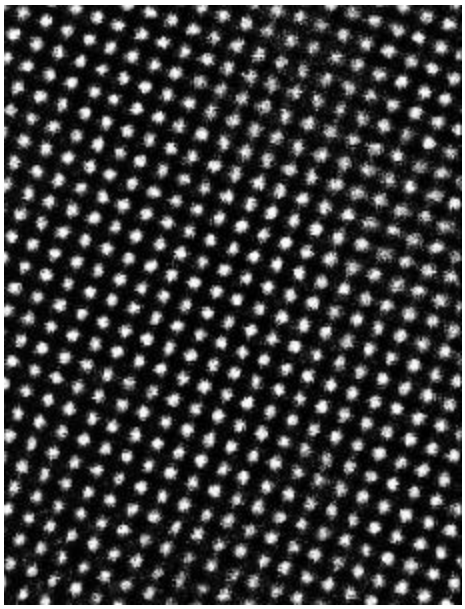


**Figure 20: Atomic resolution HREM of a rutile $TiO_2$ (0,0,1) plane.**

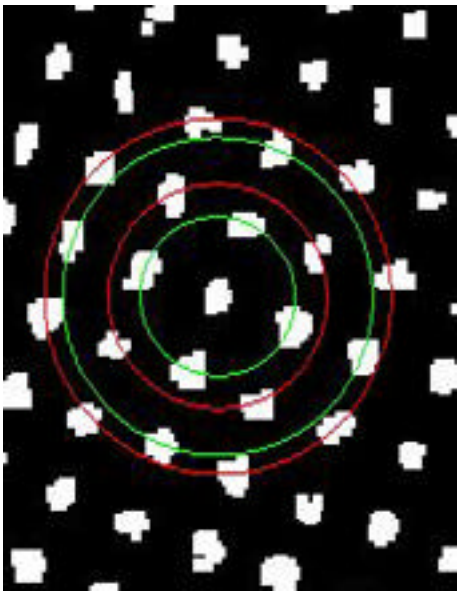


**Figure 21: Detail of the saved image showing the neighbour's distances found by LPA.**

Figure 21 shows a detail of the saved image showing the distance computed by LPA in the form of successive circles and Figure 22 shows the resulting graph. The first observation is that LPA correctly detected the distances. The peaks are wider than the ones found in the simulated images. This is due to the fact that a real, therefore not perfect, atomic lattice was analysed. But the Gaussian fitting is close to the histogram values.

Because the scale was not found, it was impossible to check the values of the distance given by the program. However, the ratios of the values respect a square lattice:

- $d2/d1 = 0.8297/0.5871 = 1.4132 \approx 2$
- $d3/d1 = 2.0015 \approx 2$
- $d4/d1 = 2.2410 \approx 5$

This result also proves that LPA correctly found the first neighbour's distances.

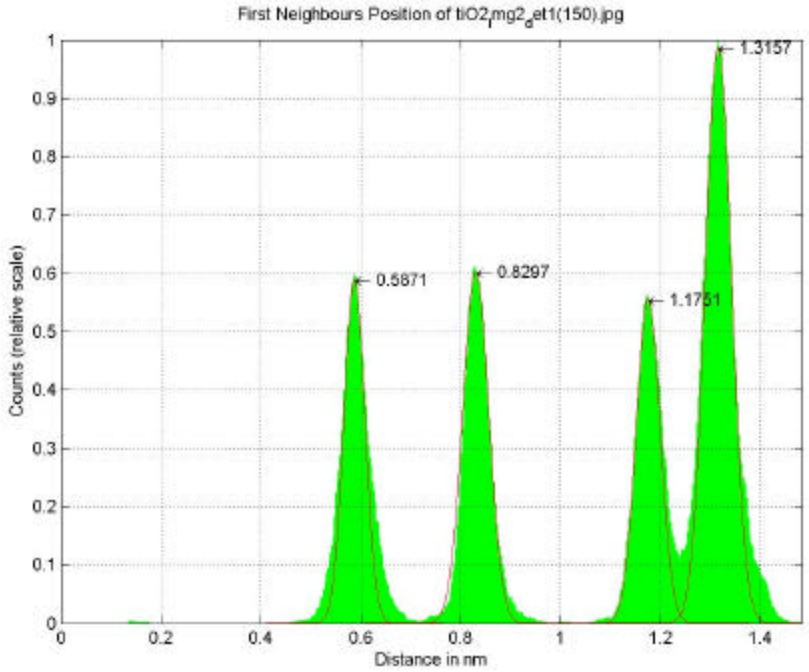



**Figure 22: Graph resulting from the LPA computation for the rutile picture.**

### 2.3.5. Conclusion

With all the generated lattices, the Lattice Parameter Analyser managed to detect the first neighbours' distances in the histogram. The values given are correct and the exact values are almost always included in the error range given by the standard deviation of the Gaussian curves that best fit the peaks (in only one extreme case it was not included, nevertheless the value was very close). The deviations found are very small due to the fact that the analysed lattices are almost perfect. Indeed, the peaks are very sharp because the atoms are positioned perfectly. It was also tested with a real HREM picture and it proved to give correct distances.

Now that the program has been checked and that it was concluded that it functioned correctly, it can be used to find atomic lattice characteristics of our $TiO_2$ HREM images.

## 2.4. LPA Results with the $TiO_2$ HREM Images

From the set of HREM images obtained with the $TiO_2$ powder sample (see ref 8), only one showed a sufficient quality to be analysed by the Lattice Parameter Analyser. It is a $TiO_2$ particle of a HREM image (Figure 8) that was analysed (Figure 23).

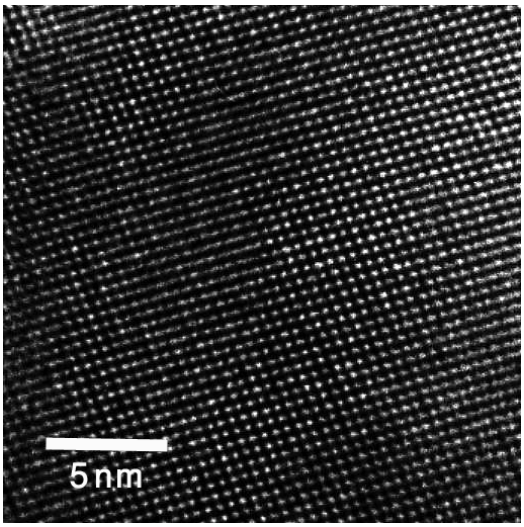



**Figure 23: Image that was analysed by the LPA. It is a part of Figure 8.**

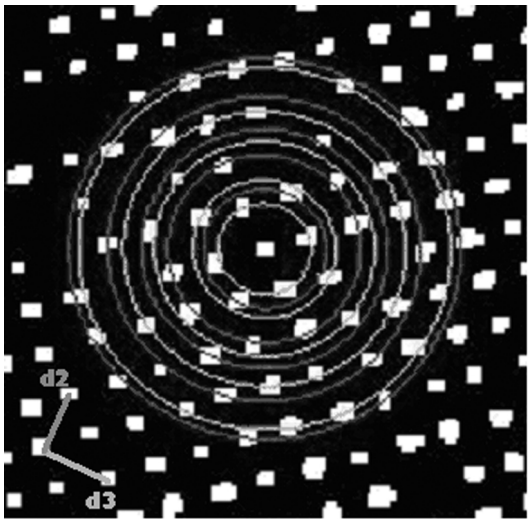



**Figure 24: Detail of the segmented image. The successive circles correspond to the distances found by the LPA.**

### 2.4.1. Comments

The scale was measured on the complete image, one pixel measures 0.03086 nm. 1247 atoms were taken into account in the distance analysis (RDF). The analysis was done for a distance of 1.5 nm (value of the Neighbourhood distance), RDF Threshold is set to 0.05 and RDF Filter Order is set to 10. The segmented image is relevant since the selected objects correspond to the atoms of the original image, as can be observed in Figure 24.

### 2.4.2. Results

Figure 25 shows the graph produced by the program with the first neighbours' distance found and Table 8 gives the distances found and their standard deviation.

As expected, the peaks are wider than with the simulated lattices. However it can be observed that the Gaussian curves from the fitting, respect roughly the histogram's shape. It is important that the fitting respects closely the top of the peaks because it means that the distances given as the position of the maximum of the Gaussian curves correspond to the first neighbours' distances.

| Peak number | Distance (in nm) | Standard Deviation (in nm) |
|---|---|---|
| 1 | 0.3536 | 0.0198 |
| 2 | 0.4671 | 0.0244 |
| 3 | 0.5395 | 0.0289 |
| 4 | 0.7313 | 0.0433 |
| 5 | 0.8369 | 0.0345 |
| 6 | 0.9314 | 0.0244 |
| 7 | 1.0712 | 0.0228 |
| 8 | 1.1873 | 0.0292 |
| 9 | 1.4118 | 0.0265 |
| 10 | 1.4831 | 0.0157 |

**Table 8: Peak positions and standard deviation.**

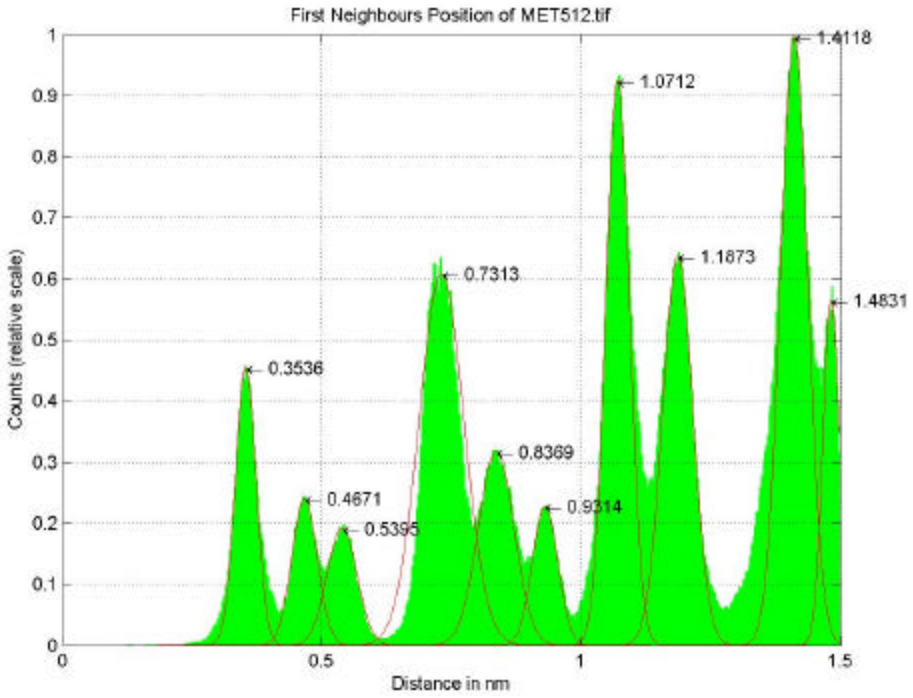



**Figure 25: Graph produced by LPA for the Figure 23.**

The value of the first neighbour's distance was found to be 0.3536 nm. It is close to the value found with the direct measuring method. But the LPA proved that the difference found with the first method does not exist. Indeed, there is only one peak at 0.3556 nm.

At first, the graph seems to show too many peaks, not that many were expected. However, the veracity of the result can be checked with the circles on the segmented image (Figure 24). It can be observed that each circle truly corresponds to a relevant distance between neighbours. For example the inner circle passes through the 4 first neighbouring atoms.

Something aroused from this analysis that was not expected: the program detected that the second neighbour's distance in one direction is different in the other direction. Indeed, the second distance is 0.4671 nm and the third is 0.5395 nm (Figure 25), there are 2 peaks on the graph. Figure 24 shows the difference in the form of the first red circle and the second green circle. In addition, the 2 distances are represented in the bottom left corner: it can be seen that d2 is slightly smaller than d3. The argument that this can be observed on the image because it is a coincidence is false. The results given by the program are statistical because it made the RDF calculation for all 1247 atoms in the picture. The program gave information about this lattice that was not observed initially.

Also, it can be noticed that the $4^{th}$ peak corresponds to the double of the $1^{st}$. The $6^{th}$ corresponds to the double of the $2^{nd}$ and the $7^{th}$ to the double of the $3^{rd}$.

### 2.4.3. Discussion

To determine the atomic plane of the image, a demo version of the Crystal Studio program has been used (http://www.crystalsoftcorp.com/index.html). It helped visualize the projections of the complex $TiO_2$ crystallographic structures.

The geometry is the same as the (2,1,1) plane of anatase phase but the distances are not respected. In this case, the first neighbour's distance is 0.8464 nm.

The atomic lattice also has the same geometry as the (1,1,1) plane of the brookite structure [ref 6]. This structure was analysed with LPA with a HREM image taken from this reference (Figure 26), the results are shown in Table 9 and the graph can be found in Figure 27. For this analysis, 635 atoms were taken into account, the RDF Threshold was set to 0.075 (because the histogram is noisier, it was necessary to lower this threshold) and RDF Filter Order was set to 15.

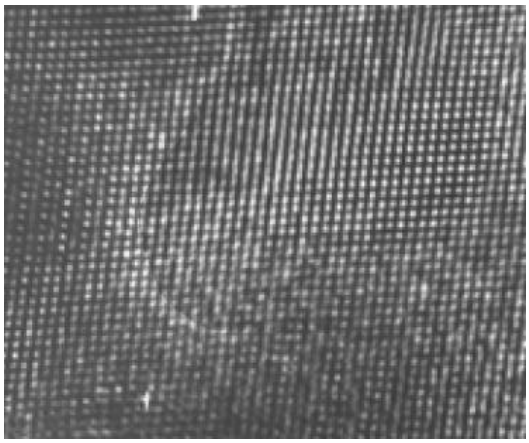


**Figure 26: Part of the HREM brookite image from ref 6.**

| Peak number | Distance (in nm) | Standard Deviation (in nm) |
|---|---|---|
| 1 | 0.3138 | 0.0164 |
| 2 | 0.3981 | 0.0275 |
| 3 | - | - |
| 4 | 0.6303 | 0.0308 |
| 5 | 0.7515 | 0.0659 |
| 6 | - | - |
| 7 | 0.9357 | 0.0237 |
| 8 | 1.0246 | 0.0329 |
| 9 | 1.2413 | 0.0341 |

**Table 9: Results for the LPA analysis of the HREM from ref 6 with the same peak indexing as Figure 25.**

The histogram shows great similarities with the one from our sample. Some of the peaks are not detected by the program but the missing ones can be guessed (e.g. the 3rd peak would be at approximately 0.4 nm). This is probably due to the fact that fewer atoms are detected because the image is smaller and of a lower quality. The values are slightly lower (e.g. 0.3138 nm for the first peak instead of 0.3536 nm in our case), however, the ratios between the distances are respected. This histogram seems to be a noisier version of ours, with the values shifted 10% to the left.

Nevertheless, it does not mean that the analysed image of our sample shows a brookite particle. Indeed, similar atomic plane geometries can be found in different structures (e.g. (1,1,1) plane of a body centred cubic structure has a hexagonal symmetry). Moreover, the X-ray analysis did not detect the brookite phase (see ref 8).

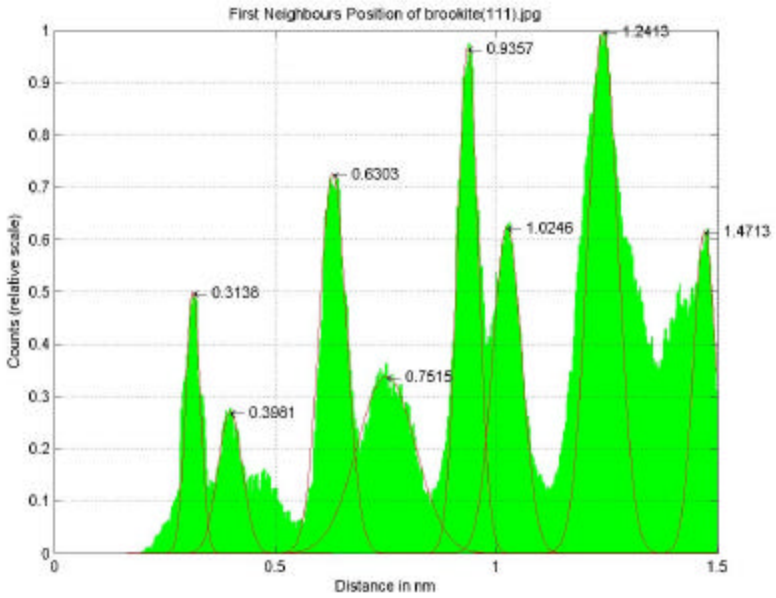



**Figure 27: Resulting graph from the Figure 26 image.**

### 2.4.4. Other Idea

Another useful information would be the value of the angle between the direction 1 and 2. Indeed, angle between lattice directions is an important characteristic and would help the atomic plane identification in HREM images.

We began to elaborate a method also using a Matlab algorithm in this objective. Nevertheless, there was no time to accomplish a complete program.

It consists in rotating the segmented image with a step of 1 degree. At each rotation, the projections on the X and Y axis are computed. From each projection, the standard deviation is calculated. In the end the standard deviations are plotted for the X and Y axis in function of the angle. Those plots present peaks that correspond to the direction of the atomic lattice where the alignment of the atoms is maximum. With the technique of peak detection presented in the LPA program, the angles could be determined.

This technique would bring extra information about the lattice. The work could be continued to integrate this tool in the LPA program.

### 2.4.5. Conclusion

Despite the precious information given by the Lattice Parameter Analyser, the atomic plane on our HREM image was not identified. However, the LPA proved to give precious information in a totally automatic way. It only takes a few seconds to obtain precise lattice distances between first neighbours from a good quality HREM image. This program can be a powerful tool for scientists that analyse lattice structures using high resolution TEM micrographs.

## 3. Conclusion

Obtaining accurate values for lattice parameters from direct HREM image analysis is not straight forward. However, this step is necessary to make basic observations and collect information about crystal defects such as dislocations.

A program called Lattice Parameter Analyser (LPA) was developed to have a different approach to the measurements of lattice parameters of the HREM pictures of the analysed $TiO_2$ particles. Its results were analysed with simulated lattice images. It also permitted to discover characteristics of the HREM image that were not detected in the direct analysis method. In the end, what was developed is a complete program that measures lattice characteristics from HREM images that can be used by other persons. No knowledge of Matlab coding is required to use LPA, however, the final interpretation of the results has to be done carefully by the scientist.

# CONCLUSION

### Summary of the work accomplished and main results:

First a Matlab program was developed with the aim of extracting automatically statistical values of characteristics of particles on images. The program works with images where the objects are separated, however, if there is superposit ion, it becomes too complex.

Then, lattice characteristics from high resolution electron microscopy have been obtained from a direct image analysis as well as from image processing techniques. The nanoparticles of the powder are single crystalline and almost defect free. A Matlab program has been developed to extract accurate and statistical values of the distance between first neighbours. The Lattice Parameter Analyser has been proved to be an efficient tool.

### Work to continue:

What could be explored to complete the LPA program is the proposed rotation method.

### Produced scientific material:

In addition to that, an internet page (http://www.cbpf.br/cat/lpdsi/lpa) has been done where basic information about the LPA can be found. The Matalb program can be downloaded together with all the information needed to use it and with an example of a simulated lattice image that can be analysed by the LPA. This internal report is available there in a pdf format as well as with the final report for ENSPG.

# REFERENCES

[1] N. Otsu, “A Threshold Selection Method from Gray-Level Histograms”, *IEEE Transactions on Systems*, Man, and Cybernetics, vol. 9, no. 1, pp. 62-66, 1979.

[2] P. Soille, “Morphological Image Analysis”, Chapter 9, Springer (1998).

[3] M. Portes de Albuquerque, “Analyse par traitement d’image de domaines magnétiques”, Thèse de doctorat de l´INPG (1995).

[4] L. Lyons, “A practical guide to data analysis for physical science student”, Cambridge university press (1993)

[5] Oppenheim, A. V. and R.W. Schafer, “Discrete-Time Signal Processing”, Englewood Cliffs, NJ: Prentice-Hall, 1989, pp. 311-312.

[6] A. Turkovi, E. Tonejc, S. Popovi, D. Cek, M. Ivanda, S. Musi and M. Goti, “Transmission Electron Microscopy, X Ray diffraction and Raman scattering studies of nanop hase $TiO_2$” *Fizika* A 6 (1997) 2, 77–88.

[7] http://www.ccr.buffalo.edu/etomica/app/modules/sites/Ljmd/Background1.html

[8] C. Jonville, “Characterisation of $TiO_2$ Nanoparticles Involving TEM and Image Processing Analysis”, Rapport de Projet de Fin d’Etudes, ENSPG.

# BIBLIOGRAPHY

D.B. Williams and C. Barry Carter, “Transmission Electron Microscopy”, Plenum Press (1996).

R.C. Gonzalez, R.E Woods and S.L. Eddins, “Digital Image Processing Using Matlab”, Pearson Prentice Hall (2004).

# APPENDIX A: LATTICE GENERATOR

## *ReadMe*

```
README FOR THE LATTICE GENERATOR (LG)
-------------------------------------------------------

This file contains basic information on how to run and test the
Lattice Generator program.

PRINCIPLE OF LG
-----------------------------------------------------------------

The LG Matlab program generates simulated images of atomic lattices
such as can be obtained with the Transmission Electron Microscope.

INSTRUCTIONS
-----------------------------------------------------------------

This Matlab program runs with the Matlab Version 6.5 Release 13.
The user´s Matlab version must feature the Image Processing Toobox
as well as the Signal Processing Toolbox.

Simply run with Matlab the "Main_LatticeGenerator.m" program.

It will read the input file "LGConf.txt" which contains set up
parameters and will generate an image named after the parameters.

INPUT FILE
-----------------------------------------------------------------

- FlagPrint: if set to 1, the program status and the final image
             will appear on the Matlab window.
- ImgSize: size in pixel of the created image
- a: first lattice parameter in number of pixels
- b: second lattice parameter in number of pixels
- Hexagonal: an hexagonal lattice can be created with
             the a as parameter.
             if you wish square or rectangular write 0
             if you wish hexagonal write 1
- Angle: the final image can be rotated if wished

The file "LGConf.txt" is set as followed:

      FlagPrint             : 1
    Image size            : 512
      Lattice Parameter a   : a = 20 pixels
      Lattice Parameter b   : b = 20 pixels
      Hexagonal             : 0
      Angle                 : 10 degrees

These parameters can be modified if desired.


OUTPUT
-----------------------------------------------------------------

The LG program will generate an image in tif format of the following
```

```
name: Lattice(a=20 b=20 angle=10).tif.

If the parameters of the "LGConf.txt" configuration file are modified,
the name will change according to the values chosen.

If the hexagonal option is chosen (Hexagonal=1), the name will be:
LatticeHex(a=20 angle=10).tif.

MORE INFO ABOUT LG
------------------------------------------------------------------

More detailed explainations about the LPA program can be found in the
webpage:
http://www.cbpf.br/cat/lpdsi/lpa

AUTHORS
------------------------------------------------------------------------
Marcelo Portes de Albuquerque
Contact: marcelo@cbpf.br

Carole Jonville
Contact: carole@cbpf.br
         jonville@dcmm.puc-rio.br
         carole.jonville@enspg.inpg.fr
```

## *Configuration File*

```
# ------------------------------------------------------------------
---
# Program     : Lattice Generator
# Written by  : Carole Jonville & Marcelo Portes
# email for   : carole@cbpf.br
#               jonville@dcmm.puc-rio.br
#               carole.jonville@enspg.inpg.fr
#               marcelo@cbpf.br
# Copyright(©): (c) CBPF - CAT
#
# Version     : 4.0814
# Objective   : Creates images simulating HREM images of atomic
lattice
# Project     : PFE ENSPG - PUC Rio - CBPF
# ------------------------------------------------------------------
---


# FlagPrint=1 shows the results on the screen
FlagPrint=1

# ImgSize: size in pixel of the created image
ImgSize=512

# a: first lattice parameter in number of pixels
a=20

# b: second lattice parameter in number of pixels
b=20

# Hexagonal: an hexagonal lattice can be created with
#            the a as parameter.
#            if you wish square or rectangular write 0
#            if you wish hexagonal write 1
Hexagonal=0

# Angle: the final image can be rotated if wished
Angle=10
```

## *Main_LatticeGenerator*

```
clear all; close all; clc;
warning off;
% ------------------------------------------------------------------
% Cabecalho
fprintf('\nLattice Generator - LaGen Version 4.0814');
fprintf('\nCentro Brasileiro de Pesquisas Fisicas - CBPF Brazil');
fprintf('\nEcole Nationale Superieure de Physique de Grenoble - ENSPG
France');
fprintf('\nPontificia Universidade Catolica do Rio de Janeiro - PUC
Rio Brazil\n');
fprintf('\n(C) MCT - All Rights Reserved - 2004');

%-------------------------------------------------------------------
%Read the configuration file

[FlagPrint ImgSize a b Hexagonal Angle] =
LGReadConfigFile('LGConf.txt');

FlagPrint=str2num(FlagPrint);
if (FlagPrint==1)
    fprintf('\n-------------------------------------------------');
      fprintf('\nInput parameters...');
    fprintf('\n\tImage size            : %s', ImgSize);
    fprintf('\n\tFlagPrint             : %d', FlagPrint);
    fprintf('\n\tLattice Parameter a   : a = %s pixels', a);
    fprintf('\n\tLattice Parameter b   : b = %s pixels', b);
    fprintf('\n\tHexagonal             : %s', Hexagonal);
    fprintf('\n\tAngle                 : %s degrees', Angle);
      fprintf('\n');
    fprintf('\nHit any key to continue...');
    fprintf('\n-------------------------------------------------');
    pause;
else fprintf('\nSilent mode');
end

ImgSize=str2num(ImgSize);
a=str2num(a);
b=str2num(b);
Hexagonal=str2num(Hexagonal);
Angle=str2num(Angle);
DeltaEnlarge=200;

%-------------------------------------------------------------------
%If Hexagonal call special function

if Hexagonal==1
   [ImgNoir Rayon SizeX SizeY ImgName] = LGHex(FlagPrint, ImgSize, a,
Angle);
end
if Hexagonal==0
   [ImgNoir Rayon SizeX SizeY ImgName] = LGSquare(FlagPrint, ImgSize,
a, b, Angle);
end

% ------------------------------------------------------------------
-
%Dilating atoms
```

```
if (FlagPrint==1) fprintf('\nDilating atoms ...'); end
se = strel('disk',Rayon);
ImgCircle = imdilate(ImgNoir,se);

% ------------------------------------------------------------------
-
%Enlarging image

if (FlagPrint==1) fprintf('\nEnlarging image ...'); end
ImgGrande=zeros(SizeY+DeltaEnlarge,SizeX+DeltaEnlarge);
ImgGrande(DeltaEnlarge/2:(DeltaEnlarge/2+SizeY-
1),DeltaEnlarge/2:(DeltaEnlarge/2+SizeX-1))=ImgCircle;

% ------------------------------------------------------------------
-
%Rotating image

if (FlagPrint==1) fprintf('\nRotating image ...'); end
ImgRotTemp = imrotate(ImgGrande,-Angle,'bilinear');

% ------------------------------------------------------------------
-
%Bluring the image

if (FlagPrint==1) fprintf('\nComputing gaussian filter...'); end
H = fspecial('gaussian', 12, 4);
ImgCircleBlur = imfilter(ImgRotTemp, H , 'replicate');

% ------------------------------------------------------------------
-
%Croping image

if (FlagPrint==1) fprintf('\nCroping image ...'); end
[ESizeY ESizeX]=size(ImgCircleBlur);
ImgFinal=ImgCircleBlur(ESizeY/2-SizeY/3:ESizeY/2+SizeY/3-1,ESizeX/2-
SizeX/3:ESizeX/2+SizeX/3-1);

% ------------------------------------------------------------------
-
%Showing image

if (FlagPrint==1) fprintf('\nShowing images ...');
image(ImgFinal); hold on; colormap(gray); axis off; hold off;
end

%------------------------------------------------------------------
---
%Saving the image

ImgSave = uint8(ImgFinal);

max = max(max(ImgSave));
max = double(max)/256;
min = min(min(ImgSave));
min = double(min)/256;
ImgSave = imadjust(ImgSave, [min max] ,[0 1] );
imwrite(ImgSave,ImgName,'tif');
% ------------------------------------------------------------------
-
fprintf('\nEnd!');
```

## *LGReadConfigFile*

```
function [FlagPrint, ImgSize, a, b, Hexagonal, Angle] =
LGReadConfigFile(ConfigFileName)

fid=fopen(ConfigFileName);
while 1
    tline = fgetl(fid);
    if (~ischar(tline)), break, end
    idx = findstr(tline,'=');

    if(strcmp(tline(1:idx-1),'FlagPrint'))
        FlagPrint=tline(idx+1:end);
    else
        if(strcmp(tline(1:idx-1),'ImgSize'))
            ImgSize=tline(idx+1:end);
        else
            if(strcmp(tline(1:idx-1),'a'))
                a=tline(idx+1:end);
             else
                   if(strcmp(tline(1:idx-1),'b'))
                   b=tline(idx+1:end);
                    else
                        if(strcmp(tline(1:idx-1),'Hexagonal'))
                          Hexagonal=tline(idx+1:end);
                         else
                             if(strcmp(tline(1:idx-1),'Angle'))
                               Angle=tline(idx+1:end);
                             else
                             end
                         end
                   end
               end
            end
        end
    end

fclose(fid);
```

## *LGSquare*

```
function [ImgNoir, Rayon, SizeX, SizeY, ImgName] = LGSquare(FlagPrint,
ImgSize, a, b, Angle)

% ------------------------------------------------------------------------
-
% Fixed Global Variables
if (FlagPrint==1) fprintf('\nGlobal variables...'); end
Rayon=round( (a + b)/2/6);
ImgName = sprintf('Lattice(a=%d b=%d angle=%d).tif', a, b, Angle);


% ------------------------------------------------------------------------
-
%Putting black points
if (FlagPrint==1) fprintf('\nCreating image...'); end
ImgSize=ImgSize*1.5;
SizeX=ImgSize; SizeY=ImgSize;
ImgNoir=zeros(SizeX,SizeY);

StepX=a;
StepY=b;
cont=0;
for y=1:StepY:SizeY
    for x=1:StepX:SizeX
        ImgNoir(y,x)=255;
    end
    cont=cont+1;
end
```

## *LGHex*

```
function [ImgNoir, Rayon, SizeX, SizeY, ImgName] = LGHex(FlagPrint,
ImgSize, a, Angle)
% ------------------------------------------------------------------
-
% Fixed Global Variables
if (FlagPrint==1) fprintf('\nGlobal variables...'); end
Rayon=round(a/6);
ImgName = sprintf('LatticeHex(a=%d angle=%d).tif', a, Angle);


% ------------------------------------------------------------------
-
if (FlagPrint==1) fprintf('\nCreating image...'); end
ImgSize=ImgSize*1.5;
SizeX=ImgSize; SizeY=ImgSize;
ImgNoir=zeros(SizeX,SizeY);

StepX=a;
StepY=round(a*sqrt(3)/2);
cont=0;
for y=1:StepY:SizeY
    for x=1:StepX:SizeX
        if(mod(cont,2)==0)
            idx=round(x+StepX/2);
            if(idx>SizeX) continue; end
            ImgNoir(y,idx)=255;
        else
            ImgNoir(y,x)=255;
        end
    end
    cont=cont+1;
end
```

# APPENDIX B: LATTICE PARAMETER ANALYSER

## *ReadMe*

```
README FOR THE LATTICE PARAMETER ANALYSER (LPA)
-----------------------------------------------------

This file contains basic information on how to run and test the
Lattice Parameter Analyser program.

PRINCIPLE OF LPA
-----------------------------------------------------------------

The LPA Matlab program analyses High Resolution Electron Microscopy
(HREM) pictures obtained with Transmission Electron Microscope (TEM).
Such images show an atomic plane of the material observed.
This program aims at extracting information on the lattice of atoms.

The LPA analyses the distance between atoms and computes the distance
between the first neighbours of the lattice. Therefore, the user
obtains
automatically precious information about the atomic lattice.

INSTRUCTIONS
-----------------------------------------------------------------

This Matlab program runs with the Matlab Version 6.5 Release 13.
The user´s Matlab version must feature the Image Processing Toobox,
the Curve Fitting Toolbox and the Signal Processing Toolbox.

Simply run with Matlab the "Main_LatticeParamAnalyser.m" program.

It will read the input file "LPAConfig.txt", analyse the
"Lattice(a=20 b=20 angle=10).tif" image and the results will be found
in the directory "Lattice(a=20 b=20 angle=10) Results" created by the
program and in the Matlab window.

INPUT FILE
-----------------------------------------------------------------

The input of this program is the text file "LPAConfig.txt" containing
the following information:

- ImgName:       Name of the image to be processed.
- FlagPrint:     if 0 the program will run in silence.
                 if 1 the program will show the program status and
                 the results on the screen.
- Scale:         Scale of the picture analysed so that the program
                 reveals results in nanometers. The scale is the
distance
                 that one pixel of the image represents (spacial
                 resolution).
- Neighbourhood: Distance in nanometers within which the program will
                 compute the distance between atoms.

Additional parameters only to use in case of program optimization:
```

```
- RDFThreshold:  value used in the function FindMax (that finds the
                 position of the peaks in the histogram of the
                 distances). It is a type of threshold used to
separate
                 peaks that are close to each other.
- Filter Order:  order of the filter that smoothes the Radial
                 Distribution Function (histogram of the distances)

The file "LPAConfig.txt" is set as followed:

- ImgName = Lattice(a=20 b=20 angle=10).tif
- FlapgPrint = 1
- Scale = 0.03
- Neighbourhood = 1,5 nm
- RDFThreshold = 0.05
- Filter Order = 10

These parameters can be modified if desired (eg if the user wants to
test another image). The images cannot be colored images.

OUTPUT FILE
-----------------------------------------------------------------------

The LPA program will create a file named after the analysed picture.
after the LPA ran, it will contain:

- Lattice(a=20 b=20 angle=10)_dist.dat      : data file containing all
                          The distances computed. If the user wishes,
                          the data can be studied to draw histograms.
- Lattice(a=20 b=20 angle=10)_peakpos.dat      : data file containing
the
                          position in nanometers of the first
neighbours
                          found by the LPA.
- Lattice(a=20 b=20 angle=10)_peak.jpg      : graph of the peaks of
the
                          histogram, showing the distance of the first
                          neighbours.
- Lattice(a=20 b=20 angle=10)_segmented.jpg : image resulting from the
image
                          processing. The user can check that the
binary
                          image used for the distance analysis
iscorrect.

IMAGE "Lattice(a=20 b=20 angle=10)"
-----------------------------------------------------------------------

The "Lattice(a=20 b=20 angle=10).tif" image simulates a typical square
lattice of atoms.
It has been generated by another Matlab program developped by the same
authors
(Lattice Generator LG).

The measurement of the first neighbour distance is 20 pixels,
so with the abitrary scale of 0.03 nm, it represents about 0,60
nanometers.

MORE INFO ABOUT LPA
-----------------------------------------------------------------------
```

More detailed explainations about the LPA and LG programs can be found in the webpage:
http://www.cbpf.br/cat/lpdsi/lpa

AUTHORS

---

Marcelo Portes de Albuquerque
Contact: marcelo@cbpf.br

Carole Jonville
Contact: carole@cbpf.br
jonville@dcmm.puc-rio.br
carole.jonville@enspg.inpg.fr

## *Configuration File*

```
# ------------------------------------------------------------------
---
# Program     : Lattice Parameter Analyser
# Written by  : Carole Jonville & Marcelo Portes
# email for   : carole@cbpf.br
#               jonville@dcmm.puc-rio.br
#               carole.jonville@enspg.inpg.fr
#               marcelo@cbpf.br
# Copyright(©): (c) CBPF - CAT
#
# Version     : 4.0816
# Objective   : Extract information on the atomic lattice from HREM
image
# Project     : PFE ENSPG - PUC Rio - CBPF
# ------------------------------------------------------------------
---

# ImgName: name of the image to be analysed
ImgName=Lattice(a=20 b=20 angle=10).tif

# FlagPrint=1 shows the results on the screen
#             set to 0 if want the program to run in silence
FlagPrint=1

# Scale: distance value of 1 pixel in nanometers (spatial resolution)
Scale=0.03

# Neighbourhood: distance in nanometer in the neighbourhood for Radial
Distribution
#            Function (the distance analysis)
Neighbourhood=1.5


#Additional parameters only to use in case of program optimization:
#------------------------------------------------------------------
---

# RDFThreshold: value used in the function FindMax (that finds the
position of the
#            peaks in the histogram of the distances). It is a type of
threshold
#            used to separate peaks that are close to each other.
RDFTh=0.05

# Filter Order: order of the filter that smoothes the Radial
Distribution Function
               (histogram of the distances)
RDFFilterOrder=10
```

## *Main_LatticeParameterAnalyser*

```
clear all; close all; clc;
warning off;
% ------------------------------------------------------------------
---
% Cabecalho
fprintf('\nLattice Parameter Analyser - LPA Version 4.0816');
fprintf('\nCentro Brasileiro de Pesquisas Fisicas - CBPF Brazil');
fprintf('\nEcole Nationale Superieure de Physique de Grenoble - ENSPG
France');
fprintf('\nPontificia Universidade Catolica do Rio de Janeiro - PUC
Rio Brazil\n')

%------------------------------------------------------------------
---
%Read the configuration file

[FlagPrint ImgName Neighbourhood Scale RDFTh RDFFilterOrder] =
ReadConfigFile('LPAConfig.txt');

FlagPrint=str2num(FlagPrint);
if (FlagPrint==1)
    fprintf('\n----------------------------------------------------
');
      fprintf('\nInput parameters...');
    fprintf('\n\tImgName          : %s', ImgName);
    fprintf('\n\tFlagPrint        : %d', FlagPrint);
    fprintf('\n\tScale            : 1 pixel = %s nm', Scale);
    fprintf('\n\tNeighbourhood    : %s nm', Neighbourhood);
    fprintf('\n\tRDF Threshold    : %s', RDFTh);
    fprintf('\n\tRDF FilterOrder  : %s', RDFFilterOrder);
      fprintf('\n');
    fprintf('\nHit any key to continue...');
    fprintf('\n----------------------------------------------------
');
    pause;
else fprintf('\nSilent mode');
end

Scale=str2num(Scale);
Voisinage=str2num(Neighbourhood);
RDFTh=str2num(RDFTh);
RDFFilterOrder=str2num(RDFFilterOrder);
Voisinage=Voisinage/Scale;


%------------------------------------------------------------------
---
%Processes the image

[ImgSEB, ImgBW] = ProcessImage(ImgName, FlagPrint);

%------------------------------------------------------------------
---
%Computes distance

[Dist, cont, CoordObjAtCenter] = DistMeasurement(ImgSEB, FlagPrint,
Voisinage);
```

```
%-------------------------------------------------------------------
---
%Finds the peaks on the histogram

[sinal2, Intensity, PeakPos, Deviation, xout, DeltaBin] =
FindMaxGauss(Dist, Scale, FlagPrint, RDFTh, RDFFilterOrder);

% ------------------------------------------------------------------
---
% Saves data and images

SavingData(FlagPrint, ImgName, ImgSEB, Dist, xout, Scale, sinal2,
Intensity, PeakPos, Deviation, cont, DeltaBin, CoordObjAtCenter);

% ------------------------------------------------------------------
---
% End

fprintf('\nEnd !!')
```

## *ReadConfigFile*

```
function [FlagPrint, ImgName, Neighbourhood, Scale, RDFTh,
RDFFilterOrder] = ReadConfigFile(ConfigFileName)

fid=fopen(ConfigFileName);
while 1
    tline = fgetl(fid);
    if (~ischar(tline)), break, end
    idx = findstr(tline,'=');

    if(strcmp(tline(1:idx-1),'FlagPrint'))
        FlagPrint=tline(idx+1:end);
    else
        if(strcmp(tline(1:idx-1),'ImgName'))
            ImgName=tline(idx+1:end);
        else
            if(strcmp(tline(1:idx-1),'Neighbourhood'))
                Neighbourhood=tline(idx+1:end);
             else
                   if(strcmp(tline(1:idx-1),'Scale'))
                   Scale=tline(idx+1:end);
                    else
                       if(strcmp(tline(1:idx-1),'RDFTh'))
                       RDFTh=tline(idx+1:end);
                        else
                           if(strcmp(tline(1:idx-1),'RDFFilterOrder'))
                             RDFFilterOrder=tline(idx+1:end);
                           end
                       end
                   end
               end
            end
        end
    end

fclose(fid);
```

## *ProcessImage*

```
function [ImgSEB, ImgBW] = ProcessImage(ImgName, FlagPrint)

% ------------------------------------------------------------------------
---
% Reading the image.
if (FlagPrint==1) fprintf('\n\nReading the image...'); end
ImgOriginal=imread(ImgName);%ImgOriginal is RGB thus 3D

if (isrgb(ImgOriginal)==1) ImgOriginal=ImgOriginal(:,:,1); end;% If
image is RGB makes it grey scale.

% ------------------------------------------------------------------------
---
% Corrects the background.
if (FlagPrint==1) fprintf('\nComputing background...'); end
tic;
H = fspecial('gaussian', 100, 30);
ImgBlurGaus = imfilter(ImgOriginal, H , 'replicate');
t1=toc;
if (FlagPrint==1) fprintf('  (it took %4.2f s !)', t1); end

if (FlagPrint==1) fprintf('\nRemoving background ...'); end
ImgCorrected = double(ImgOriginal)*.5-double(ImgBlurGaus)*.5;
ImgCorrected = uint8(ImgCorrected);

%------------------------------------------------------------------------
---
% Adjusts the contrast.
if (FlagPrint==1)  fprintf('\nAdjusting the contrast...'); end
ImgAjusted = imadjust(ImgCorrected, stretchlim(ImgCorrected), [0 1]);

% ------------------------------------------------------------------------
---
% Creating a binary image.
if (FlagPrint==1) fprintf('\nCreating a binary image...'); end
level = graythresh(ImgAjusted);
ImgBW = im2bw(ImgAjusted, level); % Makes ImgAjusted binary using a
threshold value of level

%------------------------------------------------------------------------
---
%Removing small irrelevant objects
ImgSeparated=bwmorph(ImgBW, 'open', 3);

%------------------------------------------------------------------------
---
%Removing image edge particles.
if (FlagPrint==1) fprintf('\nRemoving image edge particles...'); end
ImgSEB=imclearborder(ImgSeparated,4);
```

## *Circle*

```
function H=circle(center,radius,NOP,style)
%-------------------------------------------------------------------------
---% H=CIRCLE(CENTER,RADIUS,NOP,STYLE)
% This routine draws a circle with center defined as
% a vector CENTER, radius as a scaler RADIS. NOP is
% the number of points on the circle. As to STYLE,
% use it the same way as you use the rountine PLOT.
% Since the handle of the object is returned, you
% use routine SET to get the best result.
%
%   Usage Examples,
%
%   circle([1,3],3,1000,':');
%   circle([2,4],2,1000,'--');
%
%   Zhenhai Wang <zhenhai@ieee.org>
%   Version 1.00
%   December, 2002
%-------------------------------------------------------------------------
---

if (nargin <3),
 error('Please see help for INPUT DATA.');
elseif (nargin==3)
    style='r-';
end;
THETA=linspace(0,2*pi,NOP);
RHO=ones(1,NOP)*radius;
[X,Y] = pol2cart(THETA,RHO);
X=X+center(1);
Y=Y+center(2);
H=plot(X,Y,style);
axis square;
```

## *DistMeasurement*

```
function [Dist, cont, CoordObjAtCenter] = DistMeasurement(ImgSEB,
FlagPrint, Voisinage)

%-------------------------------------------------------------------
---
% Labeling objects.
if (FlagPrint==1) fprintf('\nLabeling objects...'); end
[ImgLabeled,numObjects] = bwlabel(ImgSEB,4);% Label components.
if (FlagPrint==1) fprintf('(%d)!',numObjects); end

% -----------------------------------------------------------------
---
% Tracking features.
if (FlagPrint==1) fprintf('\nTracking features...'); end
Objects=imfeature(ImgLabeled, 'Centroid'); %Computes only mass center.
[SizeY SizeX] = size(ImgLabeled);

%-------------------------------------------------------------------
---
%Computing distances between centroid of all objects.

numCalc=numObjects;

if (FlagPrint==1) fprintf('\nComputing distances between objects...');
end
cont=1;
tic;
DistCenter=inf;
for i=1:numCalc-1
    DistCenterTmp=sqrt( ( Objects(i).Centroid(1) - SizeX/2 )^2 + (
Objects(i).Centroid(2) - SizeY/2 )^2 );
    if (DistCenterTmp<=DistCenter)
        DistCenter=DistCenterTmp;
        CoordObjAtCenter = [Objects(i).Centroid(1),
Objects(i).Centroid(2)];
    end
    for j=i+1:numCalc
        DistTmp=sqrt( ( Objects(i).Centroid(1) -
Objects(j).Centroid(1) )^2 + ( Objects(i).Centroid(2) -
Objects(j).Centroid(2) )^2 );
        if (DistTmp<=Voisinage)
            Dist(cont)=DistTmp;
            cont=cont+1;
        end
    end
end
t2=toc;
if (FlagPrint==1) fprintf('  (it took %4.2f s !)', t2); end
```

## *FindMaxGauss*

```
function [sinal2, Intensity, PeakPos, Deviation, xout, DeltaBin] =
FindMax(Dist, Scale, FlagPrint, RDFTh, RDFFilterOrder)

if (FlagPrint==1) fprintf('\nFinding peaks on histogram...'); end

nbins = 500;
[HistCont, xout] = hist(Dist,nbins);

if (FlagPrint==1) fprintf('\nSmoothing signal ...'); end
h=ones(1,RDFFilterOrder);
sinal2 = convn(HistCont,h,'same');
sinal2 = sinal2 ./ max(sinal2);

sinal3 = imextendedmax(sinal2,RDFTh); % Find maximas, threshold to
determine

% search maximas range and peaks
FlagOnTop=0;  count=0;
for i=1:length(sinal3)
    if(sinal3(i)==1 & FlagOnTop==0)
        FlagOnTop=1;
        LeftRange=i;
    else
        if(sinal3(i)==0 & FlagOnTop==1)
            FlagOnTop=0;
            RightRange=i;

            %Enlarge Range to best fit gaussians
            LeftRange=LeftRange-2;
            if LeftRange<1 LeftRange=1; end
            RightRange=RightRange+2;
            if RightRange>length(sinal3) RightRange=length(sinal3);
end

            %Fit gaussians
            ftype = fittype('gauss1');

            Xdata = xout(LeftRange:RightRange);
            Xdata = Xdata.';
            Ydata = sinal2(LeftRange:RightRange);
            Ydata = Ydata.';

            gfit = fit(Xdata, Ydata, ftype);

            count = count + 1;
            Intensity(count) = gfit.a1;
            PeakPos(count) = gfit.b1*Scale;
            Deviation(count) = gfit.c1/sqrt(2)*Scale;

        end
    end
end

if (FlagPrint==1) fprintf('\nShowing Results ...'); end

DeltaBin=xout(2)-xout(1);
```

```
if (FlagPrint==1)
    for ind=1:count
        fprintf('\n\tPeak position(%2d)= (%4.4f nm +/- %4.4f nm)',ind,
PeakPos(ind), Deviation(ind));
    end
end
```

## *SavingData*

```
function SavingData(FlagPrint, ImgName, ImgSEB, Dist, xout, Scale,
sinal2, Intensity, PeakPos, Deviation, cont, DeltaBin,
CoordObjAtCenter)

if (FlagPrint==1) fprintf('\nSaving data and images...'); end

if (FlagPrint==1) Option='on';
else Option='off';
end

rep = sprintf('%s Results',ImgName(1:end-4));
mkdir(rep);

%-------------------------------------------------------------------
---
%Saving  Segmented Image

BWname = sprintf('.//%s//%s_segmented.jpg',rep,ImgName(1:end-4));
h=figure('Visible',Option,'Color',[1,1,1]);
imshow(ImgSEB);
hold on; colormap(gray); axis square;

[trash count] = size(PeakPos);
for ind=1:count
    if(mod(ind,2)==0) ColorStr=sprintf('r-');
    else ColorStr=sprintf('g-');
    end
    H=circle(CoordObjAtCenter,PeakPos(ind)/Scale,5000,ColorStr);
end
hold off;

title('Segmented image');
saveas(gcf,BWname,'jpg');

%-------------------------------------------------------------------
---
%Saving distances

DataOutFileName = sprintf('.//%s//%s_dist.dat',rep,ImgName(1:end-4));
fid = fopen(DataOutFileName, 'w');

for i=1:cont-1
    fprintf(fid, '\n%f',Dist(i)*Scale);
end

%-------------------------------------------------------------------
---
%Saving the peaks graph

t=(1:1:500);
DeltaX=xout(2)-xout(1);
t2=(DeltaX*t+xout(1)-DeltaX)*Scale;
h=figure('Visible',Option,'Color',[1,1,1]);
bar(t2,sinal2,'g'); set(gca,'Layer','top');  hold on;

tt=(0:0.00001:1);
hirest2=(tt*xout(end))*Scale;
```

```
[trash count] = size(PeakPos);
for ind=1:count
    EstSinal2=Intensity(ind)*exp(-((hirest2-
PeakPos(ind))./(Deviation(ind)*sqrt(2))).^2);
plot(hirest2,EstSinal2,'r-');
    TextStr=sprintf('\\leftarrow %4.4f',PeakPos(ind));
    text(PeakPos(ind), Intensity(ind), TextStr,
'HorizontalAlignment','left')
end
grid on; hold off;

title(['First Neighbours Position of ', ImgName]);
xlabel('Distance in nm');
ylabel('Counts (relative scale)');
max=((xout(2)-xout(1))*500+xout(1))*Scale;
axis([0 max 0 2000]);
axis 'auto y';

NamePeak=sprintf('.//%s//%s_peak.jpg', rep, ImgName(1:end-4));
saveas(gcf,NamePeak,'jpg');

fclose(fid);

%-------------------------------------------------------------------------
---
%Saving peak information

DataOutPeakFileName = sprintf('.//%s//%s_peakpos.txt', rep,
ImgName(1:end-4));
fid = fopen(DataOutPeakFileName, 'w');

for ind=1:count
    fprintf(fid, '\nPeak number %d at %4.4f nm +/- %4.4f nm\n', ind,
PeakPos(ind), Deviation(ind));
end

fclose(fid);

%-------------------------------------------------------------------------
---
%End

if (FlagPrint==1)
fprintf('\n\tSegmented image                        : %s', BWname);
fprintf('\n\tGraph with first neighbours´ distances: %s',NamePeak);
fprintf('\n\tAll distances within the neihgbourhood:
%s',DataOutFileName);
fprintf('\n\tFirst neighbours´ distances           : %s',
DataOutPeakFileName);
end
```