LAMPIRAN A
LISTING PROGRAM

Configuration File

```
# Program : Lattice Generator
# Version : 4.0814
# Objective :  Creates images simulating HREM images of
               atomic lattice

# 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');

%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
```

Configuration File

```
# Program : Lattice Parameter Analyser

# Version : 4.0816
# Objective :  Extract information on the atomic lattice
               from HREM Image


# 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 theposition 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.

%---------------------------------------------------------
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
```

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
```

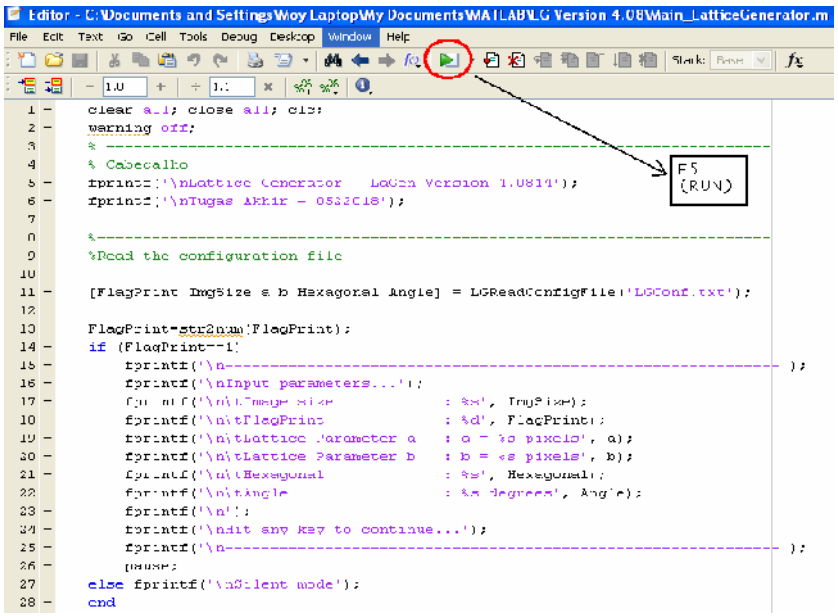
LAMPIRAN B
CARA PENGGUNAAN

Lattice Generator

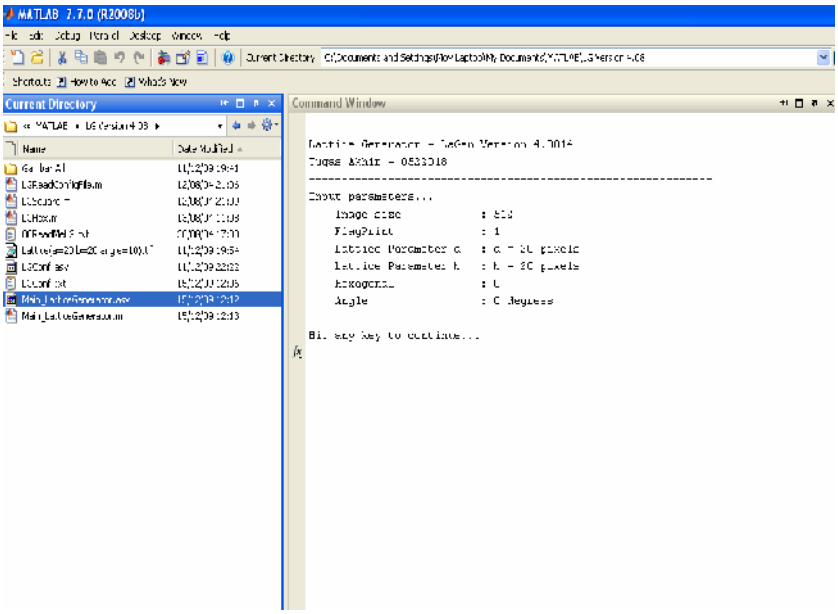
Pada penggunaan tools Lattice Generator ini dimulai dari mengatur parameter-parameter yang terdapat pada text file LGConf.txt. Terdapat 6 parameter dalam Lattice Generator seperti yang sudah dijabarkan sebelumnya. Dari keenam parameter tersebut akan diset nilainya sebagai berikut :

- Flagprint = 1
- ImgSize = 512
- a = 20
- b = 20
- Hexagonal = 0
- Angle = 0

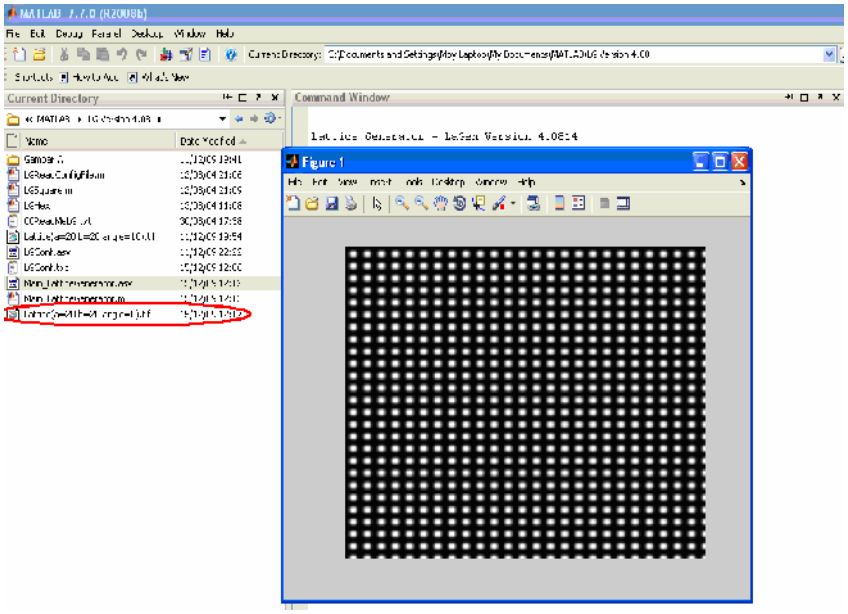
Setelah itu buka program Matlab dan cari m-file Main_LatticeGeneratorm. Dari m-file tersebut tekan F5 untuk menjalankan tools ini.



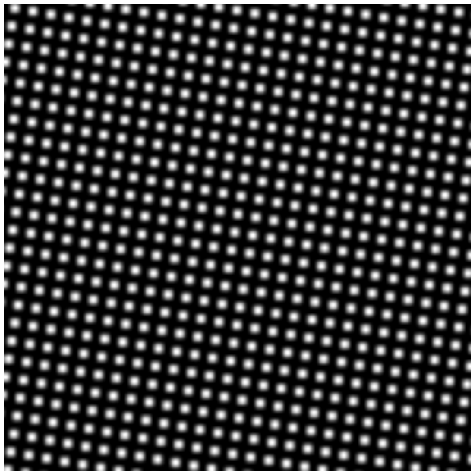
Dalam command window Matlab akan muncul konfigurasi parameter yang telah diset di LGConf.txt



Sebuah citra/gambar yang tersegementasi akan terbuat dengan nama Lattice(a=20 b=20 angle=0).tif



Gambar kemudian diolah/diproses dengan menggunakan berbagai teknik pemprosesan gambar seperti proses blurring(agar terlihat seperti gambar atanomic aslinya), proses rotasi, proses pembesaran gambar dll. Setelah input diproses kemudian gambar/citra akan langsung disave/simpan dengan format .tif dan parameter2 input akan langsung disimpan sesuai dengan nama gambar tersebut. Berikut ini merupakan contoh dari square Lattice dengan pixel 512 x 512 parameter 20 pixel dan sudut kemiringan 10 derajat

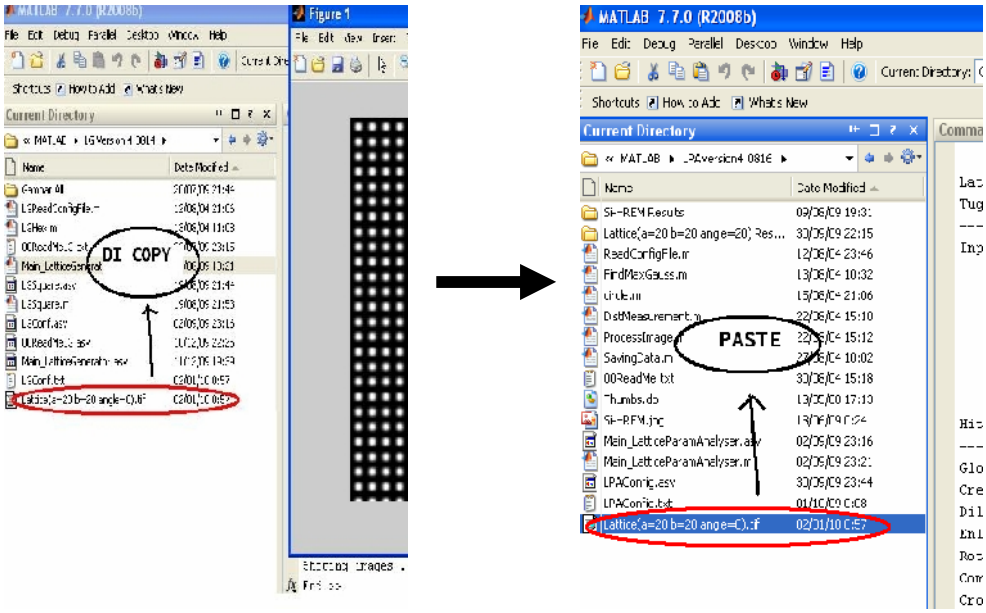


Lattice Parameter Analyzer

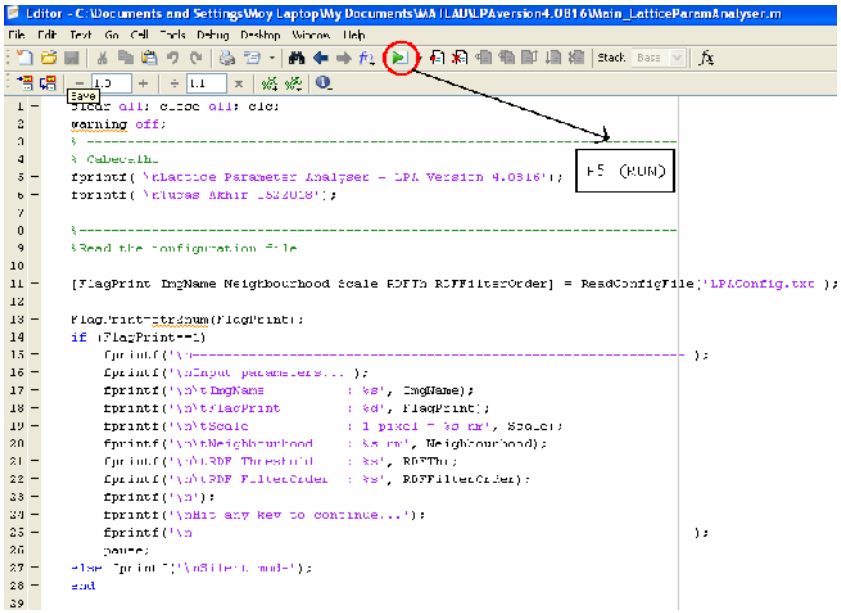
Cara penggunaan dari Lattice Parameter Analyzer, bermula dari menentukan parameter-parameter dalam LPAConfig.txt. Terdapat 5 parameter dalam text file tersebut seperti yang sudah dijelaskan sebelumnya. Kelima parameter tersebut akan diset nilai parameternya sebagai berikut :

- ImgName = Lattice(a=20 b=20 angle=0).tif
- Flagprint = 1
- Scale = 0.5
- Neighbourhood = 25
- RDFTh = 0.05
- RDFFilterOrder = 10

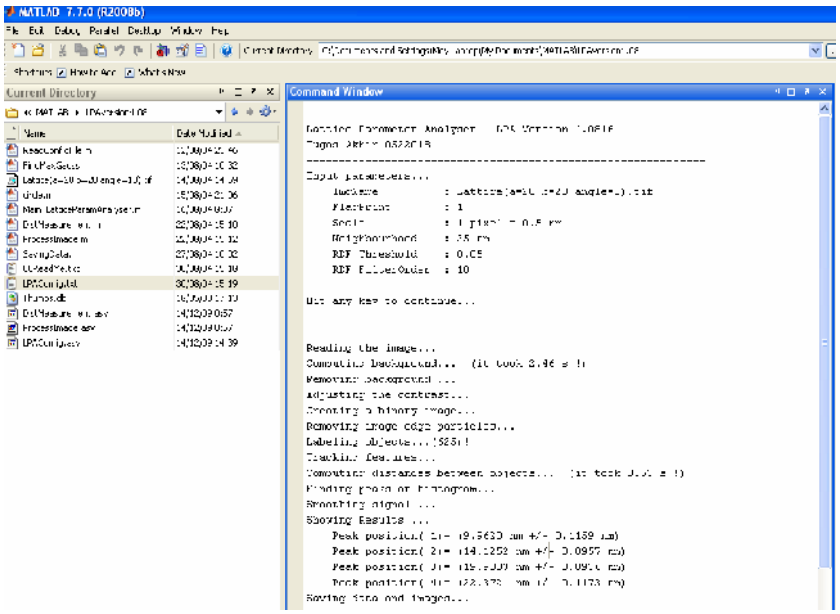
Untuk memulai tools ini, tentunya harus menentukan gambar/citra yang akan diteliti. Sample gambar yang akan diambil, berasal dari output Lattice Generator yang telah disimpan dengan nama file Lattice(a=20 b=20 angle=0).tif. Data gambar dari output Lattice Generator ini kemudian dipindahkan ke dalam folder Lattice Parameter Analyzer.



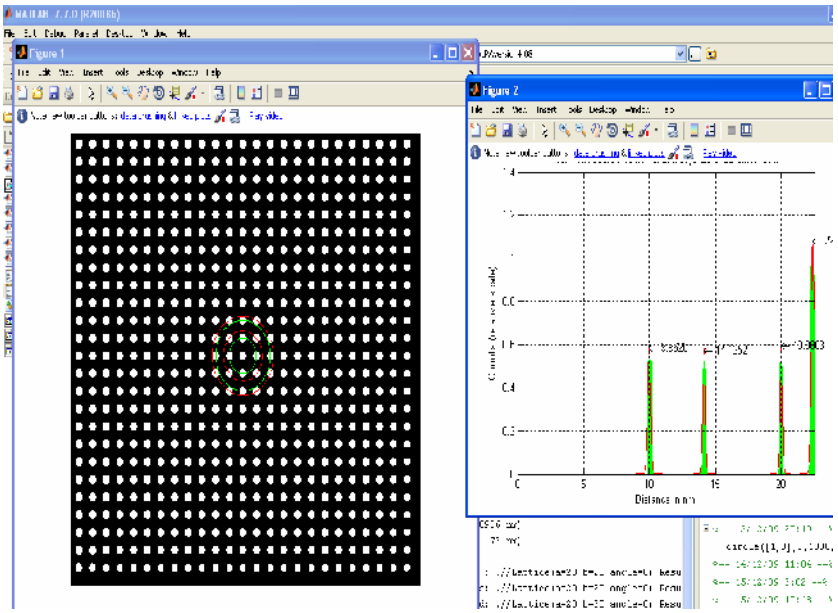
Setelah itu buka program Matlab dan cari m-file Main_LatticeParam Analyzer.m. Dari m-file tersebut tekan F5 untuk menjalankan tools ini.



Pada command window Matlab akan menampilkan proses pengukuran jarak antara nanopartikel dengan konfigurasi parameter yang diset pada LPAConfig.txt.

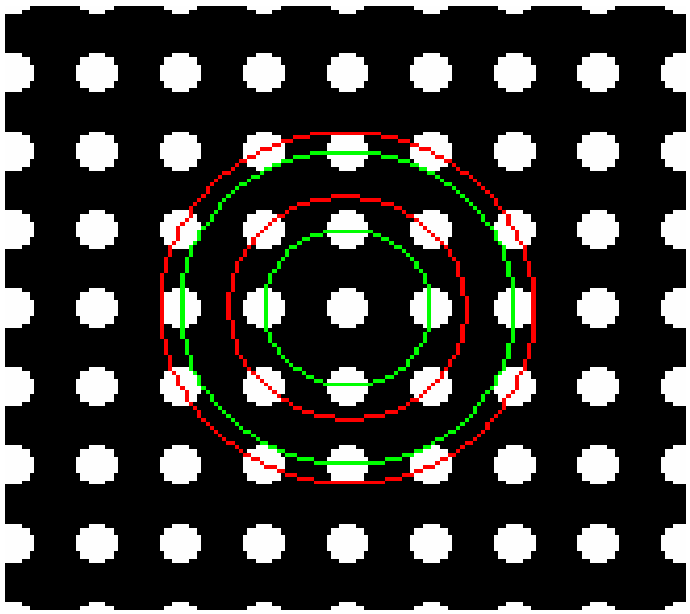


Terdapat 2 Figure Matlab yang berupa citra yang tersegmentasi dan sebuah table histogram.

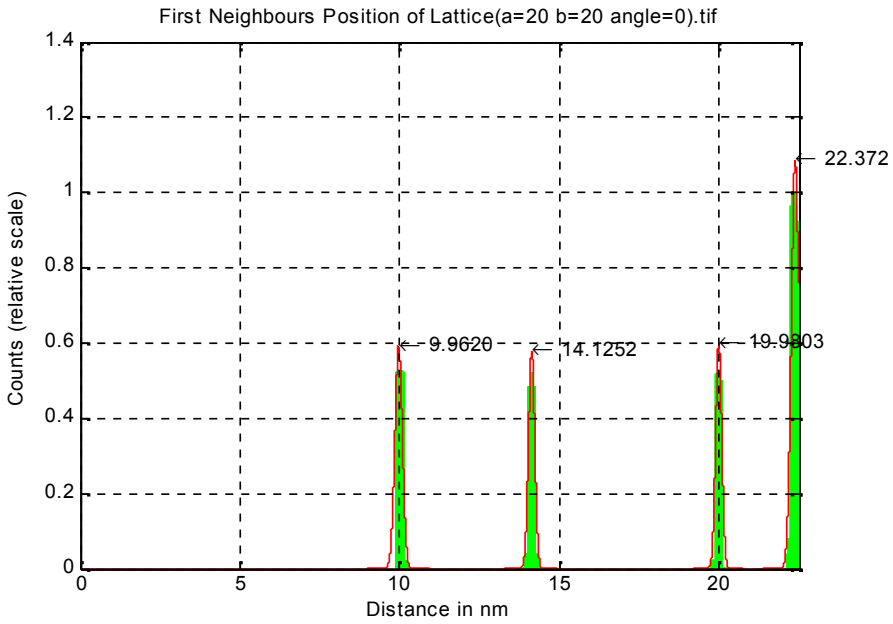


Tampilan hasil/output Lattice Parameter Analyzer

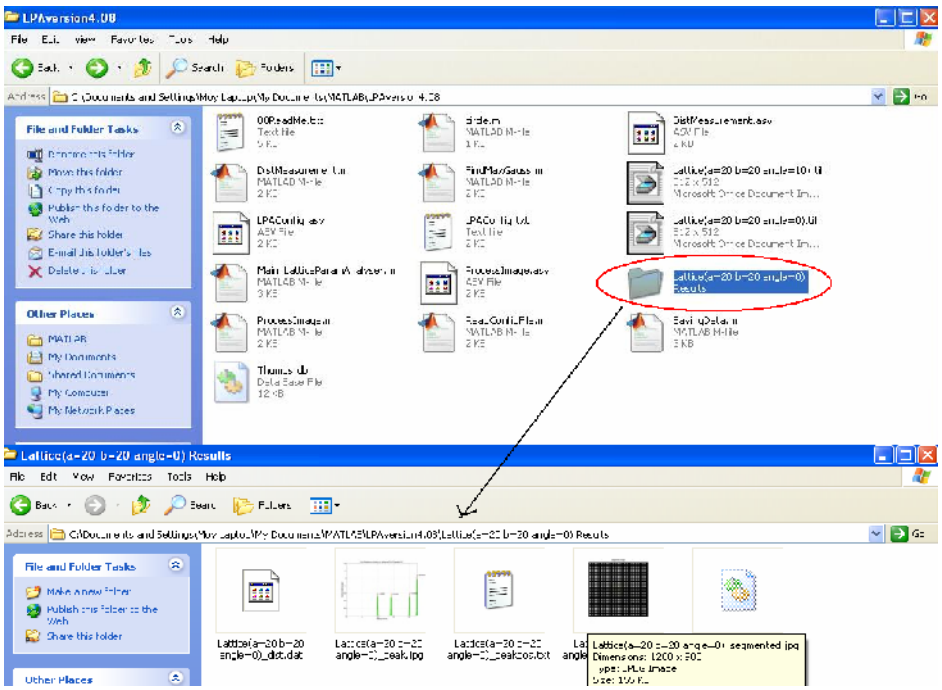
Dari Figure 1, terdapat citra dengan 4 buah lingkaran ddidalamnya. Pengukuran jarak antara nanopartikel dimulai atom yang berada dititik tengah dari gambar. Jarak yang diukur dari atom tersebut sampai ke salah satu titik tengah atom yang dilingkaran hijau paling dalam adalah jarak pengukuran yang pertama. Kemudian jarak titik tengah atom yang pertama ke salah satu titik tengah atom yang dilingkaran merah paling dalam adalah jarak pengukuran yang kedua. Jarak pengukuran ketiga dan keempat sama halnya seperti jarak pengukuran yang pertama dan kedua yaitu bermula dari titik tengah atom yang pertama.



Tabel histogram (Figure 2) merupakan hasil dari pengukuran jarak dari citra/gambar figure 1 yang tersegmentasi. Kurva X dalam tabel histogram ini menyatakan jarak antara atom pertama dengan salah satu titik tengah atom berada didalam lingkaran, sedangkan kurva Y menyatakan skala relative yang berarti jumlah maksimal atom yang berada dalam suatu lingkaran dengan skala maksimal 1.



Setelah Figure 1 dan Figure 2 ditampilkan langsung dari Matlab, kemudian tools LPA dengan otomatis akan menyimpan data hasil dari pengukuran tersebut. Data akan disimpan dalam satu folder dengan nama dari citra/gambar yang telah diolah. Letak data hasil pengukuran itu akan berada di dalam folder Lattice Parameter Analyzer. Isi dari data tersebut terdapat 4 buah yaitu 2 buah gambar dari figure 1 dan figure 2, 1 buah data dari program Matlab dengan extension .dat dan sebuah text file yang berisi jarak antara titik tengah atom satu dengan yang lainnya.


LPAversion4.08
File and Folder Tasks
Other Places
Lattice(a=20 b=20 angle=0) Results
File and Folder Tasks
Other Places