Extracting profiles in III-V [110] simulations

Mathematica code description

1. Initialization

```
NotebookDirectory []
```

```
/Users/pierrestadelmann/Desktop/III-V/
```

The section "initialization" is automatically evaluated when the notebook is loaded. This is the section that should contain the definition of the all the modules used to analyze the profiles. It hs to be made Evaluatable (menu “Cell” → menu item “Cell Properties”).

In this notebook emsImage or emsImages are images saved by jems, while mathImage or mathImages are images defined by *Mathematica*.

Some of the (numerous) Options of Plot [] and ListPlot [] are first set in the initialization section:

```
SetOptions [#, BaseStyle → {"Times", Bold, 16},
     Background → LightGray, GridLines → Automatic,
     FrameTicksStyle → {}, AxesStyle → {}, ImageSize → 600,
     PlotStyle → {AbsoluteThickness [2], AbsolutePointSize [4]},
     PlotRange → {0, 1}] & /@ {Plot, ListPlot};
```

```
plotLabel [f_] := Style [ToString@TraditionalForm [f], "Palatino", Bold, 24]
axesLabel [s_List] :=
 Style [ToString@TraditionalForm [#], "Palatino", Bold, 20] & /@ s
frameLabel [s_List] :=
 Style [ToString@TraditionalForm [#], "Palatino", Bold, 20] & /@ s
```

We can define parameters to read the simulated in any one of the AlAs, ..., ZnTe folders using replacement rules. We first define 5 symbols that will be replaced when necessary:
-IIIV : 3-5 structure to be replaced by “AlAS”, “AlP”, “AlSb”, etc.
-III : to be replaced by “Al”, “Ga”, “In”, “Zn”.
-V : to be replaced by “P”, “As”, “Sb”, “Te”.
-config : to be replaced by

```
vars = {IIIV, III, V, config, detector};
```

The images related to ZnTe, 5 configurations and bright - field detector 1 are defined by :

```
ZnTe = {"ZnTe", "Zn", "Te", "Conf5", "BF1"};
```

The List [] rules contains the replacement rules specific to analyze a given set of images. For example to access ZnTe images simulated using 5 atomic configurations and detector BF1, rules will read:

```
rules = Thread [vars -> ZnTe]
```

```
{IIIV → ZnTe, III → Zn, V → Te, config → Conf5, detector → BF1}
```

1.1 setDirectory []

```
/Users/pierrestadelmann/Dropbox/Development/
```

In order to set the default directory we define a function **setDirectory []** that uses the IIIV variables (**vars**) and the list of replacement rules (**rules**). "<>" concatenates strings.

```
setDirectory [vars_List, rules_List] := Module[{dir},
  dir =
   (IIIV /. rules) <> "/" <> (config /. rules) <> (detector /. rules) <> "/";
  dir = SetDirectory [NotebookDirectory [] <> dir];
  Print ["Data directory : ", Directory []];
 ]
```

To set the working directory that contains the ZnTe bright - field images of detector BF1 :

```
setDirectory [vars, rules]
```

1.2 listEMSImages []

listEMSImages [] sets the working directory (**setDirectory []**) and lists the images in this working directory:

```
listEMSImages [vars_, rules_] := Module [{dir},
  setDirectory [vars, rules];
  Print ["Listing .ems files of directory ", Directory []];
  FileNames ["*.ems"]
 ]
```

To list .ems images of the ZnTe directory :

```
files = listEMSImages [vars, rules]
```

1.3 readEMSImages []

Function readEMSImage [] reads a "ems" image (see below for a description of the code) :

```
readEMSImage [filename_String] := Module [{is, nr, ns, data},
  is = OpenRead [filename, BinaryFormat → True];
  ns = BinaryRead[is, "Integer32", ByteOrdering → 1];
  nr = BinaryRead[is, "Integer32", ByteOrdering → 1];
  data = BinaryRead [is, Table["Real32", {i, 1, ns nr}], ByteOrdering → 1];
  Close [is];
  data = Partition [data, ns];
  {{ns, nr}, data}
 ]
```

```
emsImage = readEMSImage [Last@files]
```

1.4 displayEMSImage []

displayEMSImage [] shows an EMS image:

```
displayEMSImage [image_] := Module [{data, min, max, scale},
  data = image [[2]];
  min = Min@data;
  max = Max@data;
  scale = 1 / (max - min);
  data = (data - min) scale;
  Image [data]
 ]
```

```
mathImage = displayEMSImage [emsImage]
```

Note that **displayEMSImage[]** creates a Mathematica image :

```
mathImage
```

1.5 displayTransposedEMSImage []

This next function **displayTransposedEMSImage []** transposes and displays an EMS image :

```
displayTransposedEMSImage[image_] := Module[{data, min, max, scale},
  data = image[[2]];
  min = Min@data;
  max = Max@data;
  scale = 1 / (max - min);
  data = ((data - min) scale) // Transpose;
  Image[data]
 ]
```

```
mathImage = displayTransposedEMSImage[emsImage]
```

Here again a Mathematica image has been created :

```
mathImage
```

1.6 shiftMathematicaImage [] and shiftTransposedMathematicaImage []

The next 2 functions shift a (Mathematica) image. The data part of the *Mathematica* image is first obtained (**ImageData []**) and then rotated (**RotateLeft []**):

```
shiftMathematicaImage[mathImage_, {w_, h_}] :=
 Module[{data = ImageData[mathImage], d},
  d = data // RotateLeft[#, h] &;
  d = RotateRight[#, w] & /@ d;
  Image[d]
 ]
```

```
shiftTransposedMathematicaImage[mathImage_, {w_, h_}] :=
 Module[{data = ImageData[mathImage], d},
  data = data // Transpose;
  d = data // RotateLeft[#, h] &;
  d = RotateRight[#, w] & /@ d;
  Image[d]
 ]
```

1.7 analyzeProfile []

This function creates .pdf profiles images.

```
analyzeProfile[vars_List, rules_List, s_Integer] :=
 Module [{data, dir, emsImage, files, profiles},
  setDirectory [vars, rules];
  files = FileNames ["*.ems"];
  emsImage = readEMSImage [Last@files];
  data = Part [emsImage, 2] // Transpose;
  profiles = takeProfiles [data, 4, 2, s, rules];
  Export [# <> ".pdf", profiles, "PDF"] & [IIIV /. rules];
  Show@Import [# <> ".pdf"] & [IIIV /. rules]
 ]
```

1.8 readAndDisplayEMSImage []

Read and display an ems image :

```
readAndDisplayEMSImage [vars_, rules_] := Module [{dir, emsImage, files},
  setDirectory [vars, rules];
  Print ["Listing .ems files of directory ", Directory []];
  files = FileNames ["*.ems"];
  emsImage = readEMSImage [Last@files];
  displayTransposedEMSImage [emsImage]
 ]
```

1.9 saveEMSProfile []

Save the profiles as ".csv" :

```
saveEMSProfile [data_List, {first_Integer, last_Integer}, name_String] :=
 Module [{range},
  range = Range [first, last];
  saveCSVProfile [d_, filename_, n_] := Module [{p, y, x},
    y = d[[n]];
    x = Range [1, Length@y];
    p = {x, y} // Transpose;
    Export [filename, p, "CSV"]
   ];
  saveCSVProfile[data, name <> ToString[#] <> ".csv", #] & /@ range
 ]
```

1.10 takeProfiles []

Extract several profiles of ems images :

```
takeProfiles[data_List, n_Integer, p_Integer, s_Integer, rules_List] :=
 Module[{plotLabel, profs, range},
  profs = Take[#, {s, s + 128}] & /@ Take[data, {1, n}];
  range = {Min@profs, Max@profs};
  label = ToString[IIIV /. rules];
  Grid[Partition[ ListPlot[#, PlotRange → range,
        ImageSize → 300, PlotLabel → label] & /@ profs, p]]
 ]
```

1.11 analyzeIIIV []

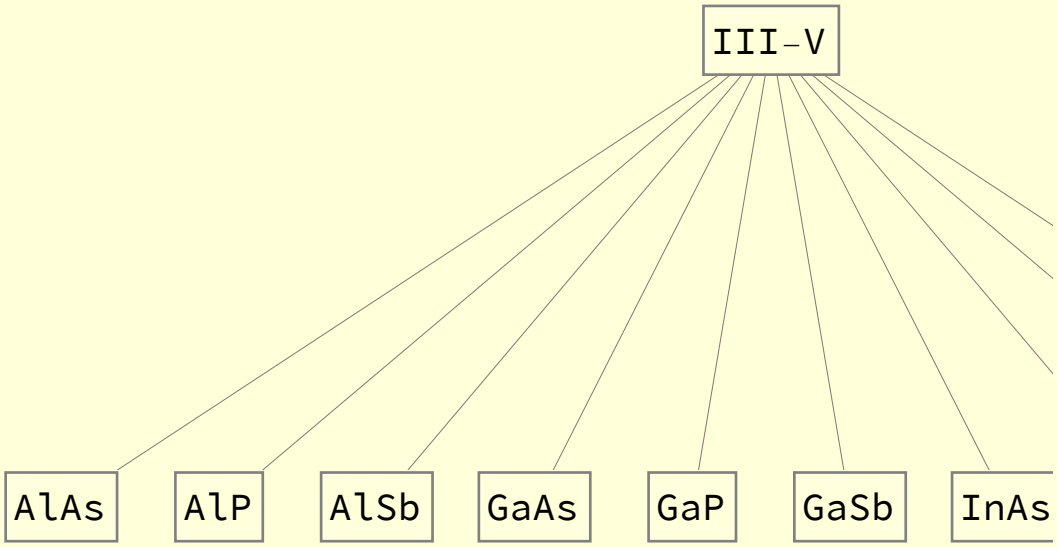
Function to perform the profile analysis on all data set at once :

```
analyzeIIIV[sys_] := Module[{image, pdf},
  rules = Thread[vars → sys];
  image = readAndDisplayEMSImage[vars, rules];
  pdf = analyzeProfile[vars, rules, 32];
  {IIIV /. rules, image, pdf}
 ]
```

---

2. Setting the path to the directory that contains the images

Use *Mathematica* commands Directory [], SetDirectory [] (and strings concatenation <> when necessary) to define the path to the directory containing the images. FileNames [] will make a list of the files ending with ".ems". A very simple way to organize your work is to put the notebook in the directory containing the images folder. After starting *Mathematica* by clicking on the notebook you can use NotebookDirectory [] to set the working directory to the image folder. We suggest to adopt for the following folder structure:



The notebook is assumed to be put in folder "III-V". For exemple, bright-field images of detector BF1 calculated for 5 different atoms configurations will be put in folder “ZnTe/Conf5BF1” when jems is using the ZnTe data files of folder “ZnTe” to calculate STEM images.

The path to the data directory is set by (for ZnTe):

```
ZnTe = {"ZnTe", "Zn", "Te", "Conf5", "BF1"};
rules = Thread [vars -> ZnTe];
setDirectory [vars, rules]
FileNames []
```

For AlAs :

```
AlAs = {"AlAs", "Al", "As", "Conf5", "BF1"}
rules = Thread [vars → AlAs]
setDirectory [vars, rules]
FileNames []
```

3. Making lists of .ems images

We now read simulated ZnTe bright - field STEM images. Since there are 4 different detectors acquisition angles we name these bf1, bf2, ..., bf4. Images are listed when the rules are defined for ZnTe:

```
ZnTe = {"ZnTe", "Zn", "Te", "Conf5", "BF1"};
rules = Thread [vars → ZnTe];
setDirectory [vars, rules];
```

When IIIV, III, V, conf and detector are the character strings “ZnTe”, “Zn”, “Te”, “Conf5” and “BF1” respectively, *Mathematica* lists images simulated for ZnTe, 5 frozen lattice configurations and

detector BF1.

```
bf1 = listEMSImages [vars, rules];
Short [bf1, 5]
```

more than 100 bright - field images are listed.

In order to load multiple sets of BF images, for example BF detector 1 to BF detector 4, one can Map [] (/@) loadImages [] on the list {“BF1”, “BF2”, “BF3”, “BF4”}:

```
files =
  {bf1, bf2, bf3, bf4} = listEMSImages [{IIIV, III, V, config, #}, rules] & /@
    {"BF1", "BF2", "BF3", "BF4"};
Short [bf1, 5]
Short [bf2, 5]
Short [bf3, 5]
Short [bf4, 5]
```

4. Reading a jems ADF bright field image

4-1. Defining a function to read an ADF image

To read an “.ems” image we have to use the following commands:

OpenRead [] open a stream,
BinaryRead [] read the image size
BinaryReadList [] read the image pixels
Close [] close the stream

Since BinaryReadList [] reads all the pixels in a **List []** ({}), it is necessary to partition the list in order to make it an image. **Partition []** just does that.

The commands are grouped into a little **Module []** that returns a list of 2 elements, the first element is the image dimensions (width, height) and the second the image data. Later we will see how to create packages of functions and how to call them in a *Mathematica* notebook.

```
readEMSImage [filename_String] := Module [{is, nr, ns, data},
  Print ["Reading : ", Directory [] <> "/" <> filename];
  is = OpenRead [filename, BinaryFormat → True];
  ns = BinaryRead[is, "Integer32", ByteOrdering → 1];
  nr = BinaryRead[is, "Integer32", ByteOrdering → 1];
  data = BinaryRead [is, Table["Real32", {i, 1, ns nr}], ByteOrdering → 1];
  Close [is];
  data = Partition [data, ns];
  {{ns, nr}, data}
 ]
```

4-2. Reading the first image listed by FileNames[]

Calling "image" the first image listed by FileNames [] (in bf1):

```
emsImage = readEMSImage [Last@bf1];
Short [emsImage, 5]
```

Where Short [emsImage, 5] tells Mathematica to display a short portion of the data.

The emsImage is made of 2 parts :
- part 1 gives the image dimensions.
-part 2 contains the image data.

This can be done also for list bf2:

```
emsImage = readEMSImage [Last@bf2]
Short [emsImage, 5]
```

The image dimensions make the first elemnet of image (Part [image, 1] or image [[1]] or better image ⟦1⟧):

```
dim = {width, height} = emsImage ⟦1⟧
dim = {width, height} = Part [emsImage, 1]
dim = {width, height} = emsImage [[1]]
```

The image data is the second part of image (Part [image, 2]):

```
data = emsImage ⟦2⟧;
```

Do not forget the ";" after the Mathematica expression. Forgetting it will display all the pixels.

5. Showing the image

The image is displayed using:

```
displayEMSImage [emsImage]
```

To take the last 4 ems images of bf1 :

```
files = Take [bf1, {-4, -1}]
```

To display the last 4 images of list bf1 :

```
displayEMSImage [readEMSImage[#]] & /@ files
```

6. How to read many images at once?

The simplest solution is to Map [] (/@) ReadEMSImage [] on the list provided by bf1 (or bf2). Reading 16 images is achieved by (in order NOT to display the *Mathematica* expressions of the images do not

forget the ";" at the end of the *Mathematica* expression!):

```
emsImages = readEMSImage [#] & /@ Take [bf1, 16];
```

Short [] only displays a small part of the expressions (here only the size of the images, i.e. first part):

```
Short [First@#, 5] & /@ emsImages
```

To have a look to the ems image data :

```
Short [Last@#, 5] & /@ emsImages
```

To display the last ems image (Last []) :

```
emsImage = Last@emsImages;
displayEMSImage [emsImage]
```

Read a few images and make a list of *Mathematica* Image:

```
mathImages = displayEMSImage [#] & /@ Take [emsImages, {-8, -1, 2}]
```

We see that the profiles that we are interested to plot are vertical. We will need to transpose the image data. This can be done using displayTransposedEMSImage []. Read the same set, but display transposed images :

```
mathImages = displayTransposedEMSImage [#] & /@ Take [emsImages, {-8, -1, 2}]
```

6.1 Image manipulation with *Mathematica* (little parenthesis)

6.1.1 Rotation, resizing

Note that often it is necessary to rotate images for comparison purposes. This easily achieved using *Mathematica* images.

```
mathImage1 =
 (data = Reverse [#] & /@ Transpose[(Last@emsImages)[[2]]]) // Image // ImageAdjust
```

```
{mathImage2, mathImage3} = {ImageRotate [mathImage1, 45 Degree],
  ImageRotate [mathImage1, 45 Degree, Background → White]}
```

The rotated image is interpolated. There several possibilities to interpolate. Using Spline interpolation :

```
spline = {"Constant", "Linear", "Quadratic", "Cubic", "Quartic", "Quintic",
   {"Spline", 6}}
```

```
mathImages4 =
 ImageRotate [mathImage1, 45 Degree, Background → White, Resampling → #] & /@
  spline
```

Using ImageSubtract [] we can see the differences due to the different interpolation :

```
ImageAdjust[ImageSubtract [mathImage3, #]] & /@ mathImages4
```

Obviously Mathematica uses "Linear interpolation" by default. There are many other interpolation schemes. Here are some classical ones:

```
classic = {"Dodgson", {"Keys", -1 / 2}, "CatmullRom",
  {"Hermite", 5} , {"Schaum", 5}, {"Meijering", 5}, {"OMOMS", 7}}
```

```
mathImages5 =
 ImageRotate [mathImage1, 45 Degree, Background → White, Resampling → #] & /@
  classic
```

```
ImageAdjust[ImageSubtract [mathImage3, #]] & /@ mathImages5
```

In principal better interpolation are obtained with these windowing methods :

```
windowing = {{"Bartlett", 4}, {"Blackman", 4},
  {"Connes", 4, 1}, {"Cosine", 3, 1}, {"Hamming", 4}, {"Hann", 4, 1 / 2},
  {"Kaiser", 4, 13 / 2}, {"Lanczos", 3}, {"Parzen", 4}, {"Welch", 4, 1}}
```

```
mathImages6 =
 ImageRotate [mathImage1, 45 Degree, Background → White, Resampling → #] & /@
  windowing
```

```
ImageAdjust[ImageSubtract [mathImage3, #]] & /@ mathImages6
```

6.1.2 Stack of images

It is also possible to stack images in 3-D:

```
Image3D [mathImages6, ColorFunction → "GrayLevelOpacity"]
```

A useful coloring scheme :

```
Image3D [mathImages6, ColorFunction → (GrayLevel[#, .005] &)]
```

6.1.3 An example from the Mathematica help

We can load all ems images at once (do not forget the colon at the end of these *Mathematica* expressions):

```
emsImages = readEMSImage [#] & /@ bf1;
```

```
mathImages = ImageAdjust@Image@Part [#, 2] & /@ emsImages;
Labeled [Image3D [mathImages, ColorFunction → #], #] & /@ {"XRay", "HighRange",
  "LowRange", "WhiteBlackOpacity", "SunsetColorsOpacity", "RainbowOpacity"}
```

```
emsImages = Take [emsImages, {1, Length@emsImages, 2}];
mathImages = ImageAdjust@Image@Part [#, 2] & /@ emsImages;
Labeled [Image3D [mathImages, ColorFunction → #], #] & /@ {"XRay", "HighRange",
  "LowRange", "WhiteBlackOpacity", "SunsetColorsOpacity", "RainbowOpacity"}
```

```
Image3DSlices [Image3D [mathImages]];
Take [%, {1, 5}]
```

7. Extracting profiles

From the images we see that, in order to get the contrast of the ratio III-V atoms, we need to transpose the images. First we do that on the last image:

```
(data = Transpose[(Last@emsImages)[[2]]]) // Image // ImageAdjust
```

To reverse the image on a line basis :

```
(data = Reverse [#] & /@ Transpose[(Last@emsImages)[[2]]]) // Image // ImageAdjust
```

7.1 Plotting the rows

We can plot then the first 4 rows of the image, i.e. profiles:

```
Grid [Partition [ListPlot [#, PlotRange → All, ImageSize → 300,
      PlotLabel → (IIIV /. rules)] & /@Take[data, {1, 4}], 2]]
```

7.2 Saving the profiles

```
saveEMSProfile[data_List, {first_Integer, last_Integer}, name_String] :=
 Module[{range},
  range = Range[first, last];
  saveCSVProfile[d_, filename_, n_] := Module[{p, y, x},
    y = d[[n]];
    x = Range[1, Length@y];
    p = {x, y} // Transpose;
    Export[filename, p, "CSV"]
   ];
  saveCSVProfile[data, name <> ToString[#] <> ".csv", #] & /@ range
 ]
```

```
savedCSV = saveEMSProfile[data, {1, 4}, IIIV /. rules]
```

```
FilePrint[savedCSV[[1]]]
```

```
ListPlot[Import[savedCSV[[1]]], PlotRange → {0.9, 1}]
```

7.3 Taking part of the profiles

We take some part of the rows in order to show the difference of contrast of the III and V columns. We can also have initialization cells in the middle of a notebook. Selecting the cell at the right of the notebook we can identify “Initialization Cell” by looking at its upper right square bracket: a vertical line segment is added to the standard “Input Cell” bracket.

```
takeProfiles[data_List, n_Integer, p_Integer, s_Integer, rules_List] :=
 Module[{plotLabel, profs, range},
  profs = Take[#, {s, s + 128}] & /@ Take[data, {1, n}];
  range = {Min@profs, Max@profs};
  plotLabel = ToString[IIIV /. rules];
  Grid[Partition[ ListPlot[#, PlotRange → range,
        ImageSize → 300, PlotLabel → plotLabel] & /@ profs, p]]
 ]
```

```
profIIIV = takeProfiles[data, 4, 2, #, rules] & /@ Range[1, 128, 32]
```

We can export the profiles as "PDF" files :

```
Export[# <> ".pdf", profIIIV, "PDF"] & [IIIV /. rules]
Show@Import[# <> ".pdf"] & [IIIV /. rules]
```

7.4 Further image manipulations

It could be interesting to shift the images before extracting the profiles. This is achieved with by the next 2 functions that takes as input a Mathematica image:

```
shiftMathematicaImage [mathImage_, {w_, h_}] :=
 Module [{data = ImageData [mathImage], d},
  d = data // RotateLeft [#, h] &;
  d = RotateRight [#, w] & /@ d;
  Image [d]
 ]
```

```
shiftTransposedMathematicaImage [mathImage_, {w_, h_}] :=
 Module [{data = ImageData [mathImage], d},
  data = data // Transpose;
  d = data // RotateLeft [#, h] &;
  d = RotateRight [#, w] & /@ d;
  Image [d]
 ]
```

We use them to look at profiles from mathImages :

```
mathImages = displayTransposedEMSImage [#] & /@ Take [emsImages, {-8, -1, 2}]
```

```
mathImage = Image[(Last@mathImages)];
dims = {width, height} = (Last@emsImages)[[1]]
```

```
shiftMathematicaImage [mathImage, {32, 32}]
```

```
GraphicsRow [
 {mathImage, shiftMathematicaImage [mathImage, {128, 128}]}, ImageSize → 256]
```

Interactive shift is performed with Manipulate [] :

```
Manipulate [GraphicsRow [{mathImage,
    shiftTransposedMathematicaImage [mathImage, {x, y}]}, ImageSize → 512],
 {{x, width, "x"}, -width, width, 1}, {{y, height, "y"}, -height, height, 1}]
```

```
data = shiftTransposedMathematicaImage [mathImage, {32, 0}] // ImageData;
ListPlot [First@data]
```

```
(Take [#, {1, 128}] & /@ Take[data, {1, 2}]) // ListPlot
```

```
Manipulate [ListPlot [Take [data [[n]], {1, 128}], PlotRange → {0, 1}],
 {{n, 1, "n"}, 1, 16, 1}]
```

8 Finding III and V minimum (and their ratio)

8.1 Finding III and V minimum using Manipulate [] :

```
profileIIIV = Take [data ⟦1⟧, {32, 32 + 127}];
```

```
range = {min, max} = {Min@profileIIIV, Max@profileIIIV}
```

```
Manipulate [ListPlot[profileIIIV, PlotRange → range, ImageSize → 600, Epilog →
    {AbsolutePointSize [10], RGBColor [1, 0, 0], Point [{i, profileIIIV⟦i⟧}]},
  PlotLabel → "(" <> ToString@i <> ", " <> ToString@profileIIIV⟦i⟧ <> ")"],
 {{i, 1, "n"}, 1, 128, 1}]
```

8.2 Storing the results :

```
makeResult[IIIV_String, profile_List,
  {ei_String, ej_String}, {i_Integer, j_Integer}] := Module [{},
  {IIIV, {{ei, i, profile ⟦i⟧}, {ej, j, profile ⟦j⟧}}}]
```

```
makeResult [IIIV /. rules, profileIIIV, {III /. rules, V /. rules}, {105, 54}]
```

9. Finding III and V minima automatically

```
range = {min, max} = {Min@profileIIIV, Max@profileIIIV}
```

```
ListPlot [profileIIIV // Reverse, PlotRange → range]
```

```
l = RotateLeft [profileIIIV, 32];
ListPlot [l, PlotRange → range, PlotStyle → Red]
m = l //.{a__, b_, c___} ⧴ {a, c} /; b > Min[a]
ListPlot [{l, m}, PlotRange → range, PlotStyle → {Red, Blue}]
{Length@m, l⟦Length@m⟧}
```

```
l
```

```
l = Take [l, {64, 80}];
m = l //.{a__, b_, c___} ⧴ {a, c} /; b > Min[a];
ListPlot [{l, m}, PlotRange → range, PlotStyle → {Red, Blue}]
{63 + Length@m, l⟦Length@m⟧}
```

```
makeResult [IIIV /. rules, profileIIIV, {III /. rules, V /. rules}, {74, 22}]
```

For AlAs :

To access AlAs images simulated using 5 atomic configurations and detector BF1, rules will read :

```
AlAs = {"AlAs", "Al", "As", "Conf5", "BF1"};
rules = Thread [vars → AlAs]
```

Now the profile analysis is simple :

```
bf1 = listEMSImages [vars /. rules];
Short [bf1, 5]
```

```
emsImage = readEMSImage [Last@bf1];
Short [emsImage, 5]
```

```
displayEMSImage [readEMSImage[#]] & [Last@bf1]
```

```
profIIIV = takeProfiles [emsImage[[2]], 4, 2, 16, rules]
```

```
Export [# <> ".pdf", profIIIV, "PDF"] & [IIIV /. rules]
Show@Import [# <> ".pdf"] & [IIIV /. rules]
```

We can indeed group the reduced set of command in a single Module [] (already put as an initialization cell):

```
analyzeProfile[vars_List, rules_List, s_Integer] :=
 Module [{data, dir, emsImage, files, profiles},
  setDirectory [vars, rules];
  files = FileNames ["*.ems"];
  emsImage = readEMSImage [Last@files];
  data = Part [emsImage, 2] // Transpose;
  profiles = takeProfiles [data, 4, 2, s, rules];
  Export [# <> ".pdf", profiles, "PDF"] & [IIIV /. rules];
  Show@Import [# <> ".pdf"] & [IIIV /. rules]
 ]
```

Analysis of the III - V images

1. Notes

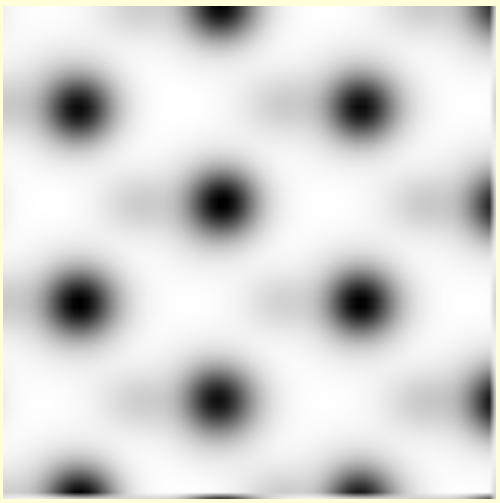
For all III - V except GaAs the BF detector sizes are: 25, 30, 35, and 40 mrad and the crystal thickness ~25 nm (120 to 128 slices of ~0.2 nm).
For GaAs bright-filed detector size are: 23, 28, 33 and 38 mrad, dark-field images 60→150, 75→165, 90→180, 105→195 mrad and the crystal thickness 51 nm (configurations 6 and 7).

2. AlAs

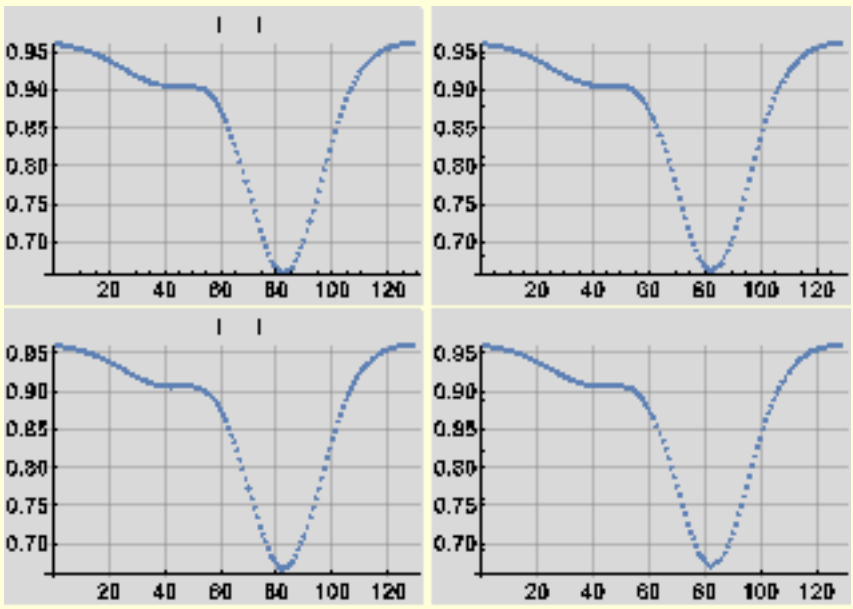
```
AlAs = {"AlAs", "Al", "As", "Conf5", "BF1"};
rules = Thread [vars → AlAs]
readAndDisplayEMSImage [vars, rules]
analyzeProfile[vars, rules, 32]
```

```
{IIIV → AlAs, III → Al, V → As, config → Conf5, detector → BF1}
```

```
Data directory : /Users/pierrestadelmann/Desktop/III-V/AlAs/Conf5BF1
Listing .ems files of directory /Users/pierrestadelmann/Desktop/III-V/AlAs/Conf5BF1
Reading : /Users/pierrestadelmann/Desktop/III-V/AlAs/Conf5BF1/BF_0119.ems
```



```
Data directory : /Users/pierrestadelmann/Desktop/III-V/AlAs/Conf5BF1
Reading : /Users/pierrestadelmann/Desktop/III-V/AlAs/Conf5BF1/BF_0119.ems
```

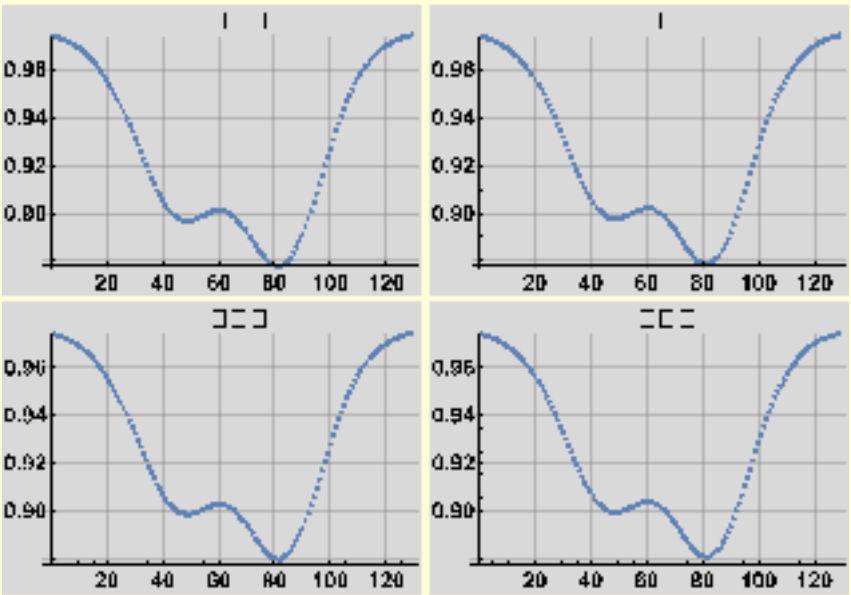


3. AlP

```
AlP = {"AlP", "Al", "P", "Conf5", "BF1"};
rules = Thread [vars → AlP];
readAndDisplayEMSImage [vars, rules]
analyzeProfile[vars, rules, 32]
```

```
Data directory : /Users/pierrestadelmann/Desktop/III-V/AlP/Conf5BF1
Listing .ems files of directory /Users/pierrestadelmann/Desktop/III-V/AlP/Conf5BF1
Reading : /Users/pierrestadelmann/Desktop/III-V/AlP/Conf5BF1/BF_0127.ems
```

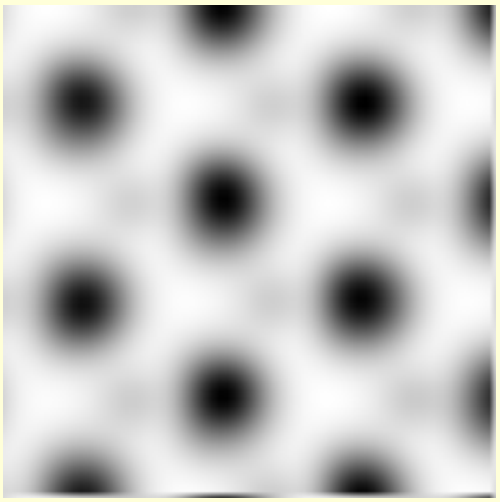
```
Data directory : /Users/pierrestadelmann/Desktop/III-V/AlP/Conf5BF1
Reading : /Users/pierrestadelmann/Desktop/III-V/AlP/Conf5BF1/BF_0127.ems
```



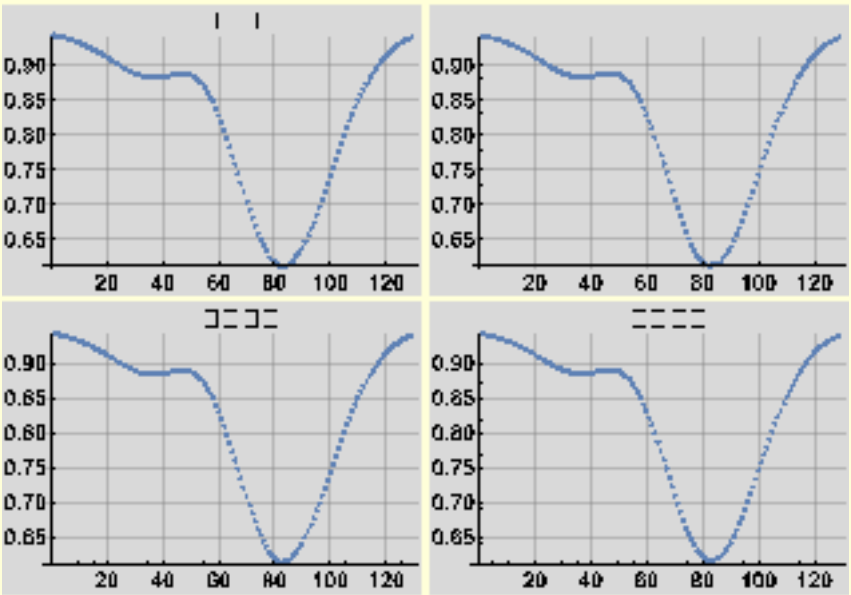
4. AlSb

```
AlSb = {"AlSb", "Al", "Sb", "Conf5", "BF1"};
rules = Thread [vars → AlSb];
readAndDisplayEMSImage [vars, rules]
analyzeProfile[vars, rules, 32]
```

```
Data directory : /Users/pierrestadelmann/Desktop/III-V/AlSb/Conf5BF1
Listing .ems files of directory /Users/pierrestadelmann/Desktop/III-V/AlSb/Conf5BF1
Reading : /Users/pierrestadelmann/Desktop/III-V/AlSb/Conf5BF1/BF_0127.ems
```



```
Data directory : /Users/pierrestadelmann/Desktop/III-V/AlSb/Conf5BF1
Reading : /Users/pierrestadelmann/Desktop/III-V/AlSb/Conf5BF1/BF_0127.ems
```



5. GaAs

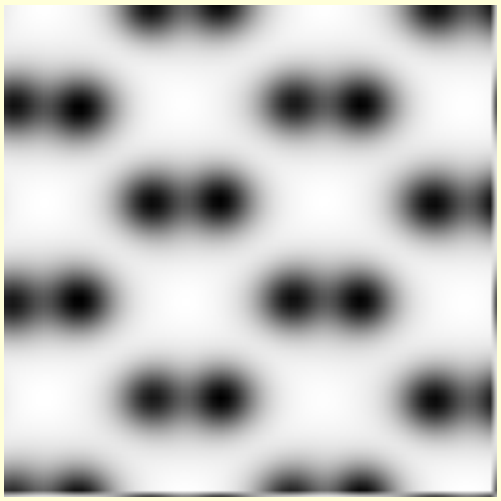
Thickness of GaAs is 256 x 0.2 nm =
51 nm (last image is BF_0255.ems it the the 256th one)

5.1 Bright - field (BF1 to BF4)

Bright-field detector sizes BF1: 23, BF2: 28, BF3: 33, BF4: 38 mrad.

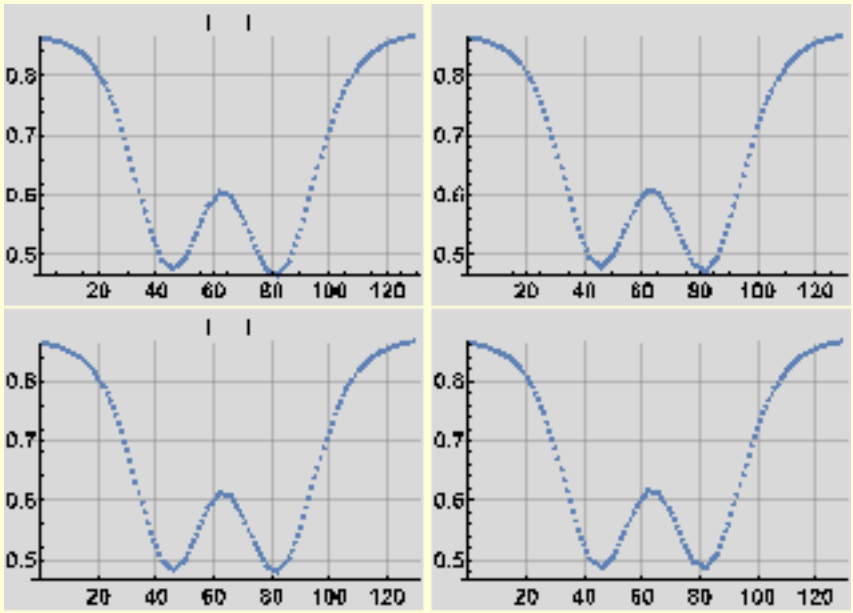
```
GaAs = {"GaAs", "Ga", "As", "Conf7", "BF1"};
rules = Thread [vars → GaAs];
readAndDisplayEMSImage [vars, rules]
analyzeProfile[vars, rules, 32]
```

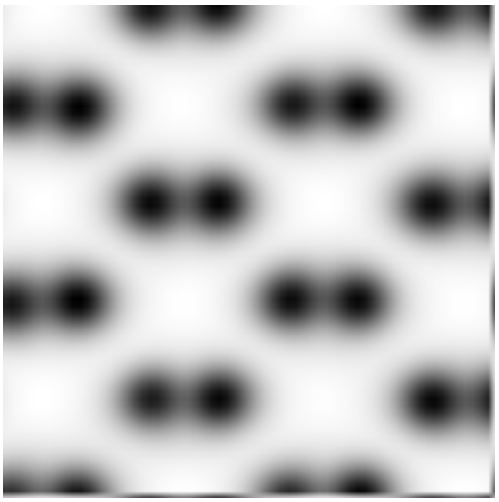
```
Data directory : /Users/pierrestadelmann/Desktop/III-V/GaAs/Conf7BF1
Listing .ems files of directory /Users/pierrestadelmann/Desktop/III-V/GaAs/Conf7BF1
Reading : /Users/pierrestadelmann/Desktop/III-V/GaAs/Conf7BF1/BF_0255.ems
```



Data directory : /Users/pierrestadelmann/Desktop/III-V/GaAs/Conf7BF1

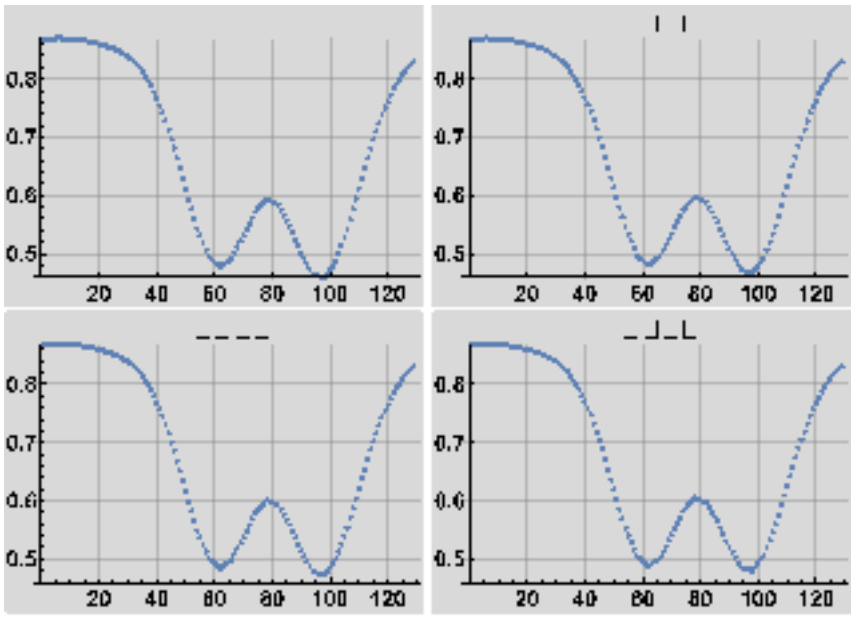
Reading : /Users/pierrestadelmann/Desktop/III-V/GaAs/Conf7BF1/BF_0255.ems





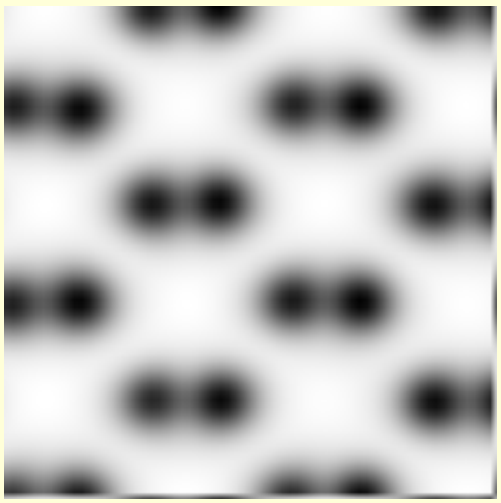
Data directory : /Users/pierrestadelmann/Desktop/III-V/GaAs/Conf6BF1

Reading : /Users/pierrestadelmann/Desktop/III-V/GaAs/Conf6BF1/BF_0255.ems

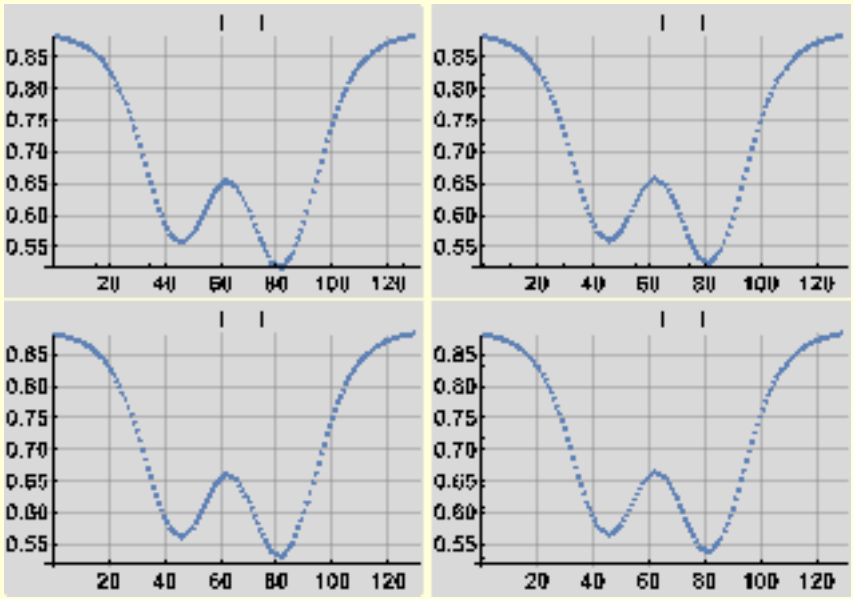



```
GaAs = {"GaAs", "Ga", "As", "Conf7", "BF2"};
rules = Thread [vars → GaAs];
readAndDisplayEMSImage [vars, rules]
analyzeProfile[vars, rules, 32]
```

```
Data directory : /Users/pierrestadelmann/Desktop/III-V/GaAs/Conf7BF2
Listing .ems files of directory /Users/pierrestadelmann/Desktop/III-V/GaAs/Conf7BF2
Reading : /Users/pierrestadelmann/Desktop/III-V/GaAs/Conf7BF2/BF_0255.ems
```

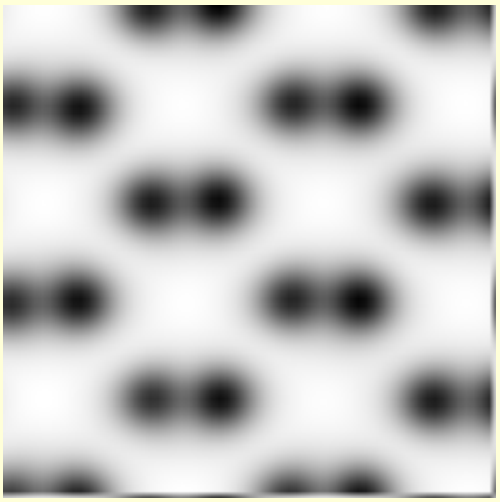


```
Data directory : /Users/pierrestadelmann/Desktop/III-V/GaAs/Conf7BF2
Reading : /Users/pierrestadelmann/Desktop/III-V/GaAs/Conf7BF2/BF_0255.ems
```

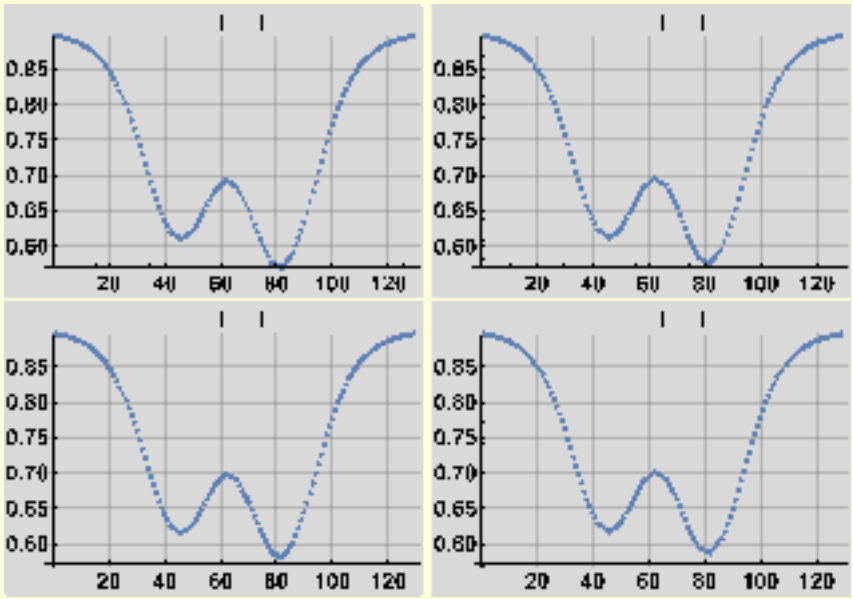



```
GaAs = {"GaAs", "Ga", "As", "Conf7", "BF3"};
rules = Thread [vars → GaAs];
readAndDisplayEMSImage [vars, rules]
analyzeProfile[vars, rules, 32]
```

```
Data directory : /Users/pierrestadelmann/Desktop/III-V/GaAs/Conf7BF3
Listing .ems files of directory /Users/pierrestadelmann/Desktop/III-V/GaAs/Conf7BF3
Reading : /Users/pierrestadelmann/Desktop/III-V/GaAs/Conf7BF3/BF_0255.ems
```

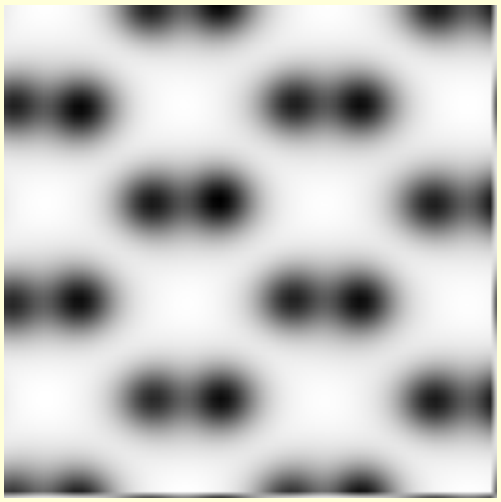


```
Data directory : /Users/pierrestadelmann/Desktop/III-V/GaAs/Conf7BF3
Reading : /Users/pierrestadelmann/Desktop/III-V/GaAs/Conf7BF3/BF_0255.ems
```

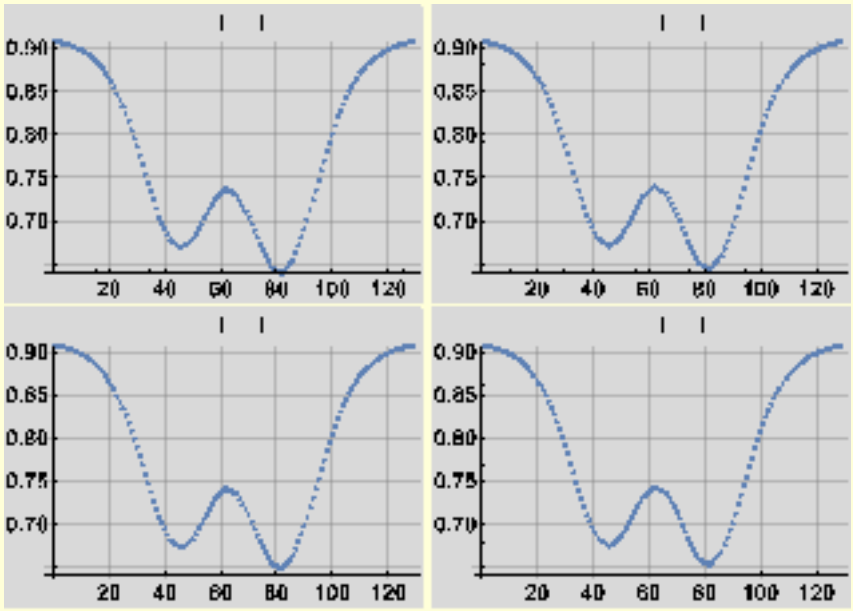



```
GaAs = {"GaAs", "Ga", "As", "Conf7", "BF4"};
rules = Thread [vars → GaAs];
readAndDisplayEMSImage [vars, rules]
analyzeProfile[vars, rules, 32]
```

```
Data directory : /Users/pierrestadelmann/Desktop/III-V/GaAs/Conf7BF4
Listing .ems files of directory /Users/pierrestadelmann/Desktop/III-V/GaAs/Conf7BF4
Reading : /Users/pierrestadelmann/Desktop/III-V/GaAs/Conf7BF4/BF_0255.ems
```



```
Data directory : /Users/pierrestadelmann/Desktop/III-V/GaAs/Conf7BF4
Reading : /Users/pierrestadelmann/Desktop/III-V/GaAs/Conf7BF4/BF_0255.ems
```

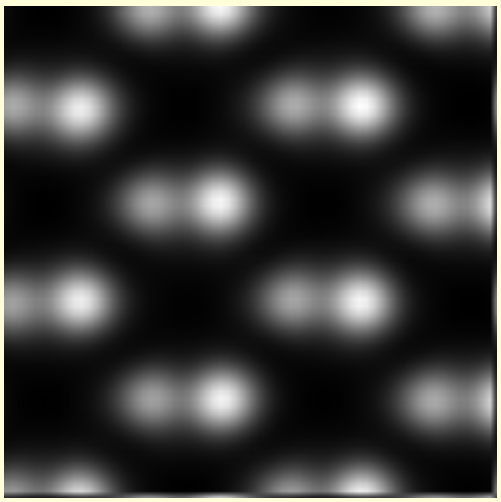


5.2 Dark - field (ADF1 to ADF4)

Dark - field detector sizes ADF1 : 60 → 150, ADF2 : 75 → 165, ADF3 : 90 → 180, ADF4 : 105 → 195 mrad.

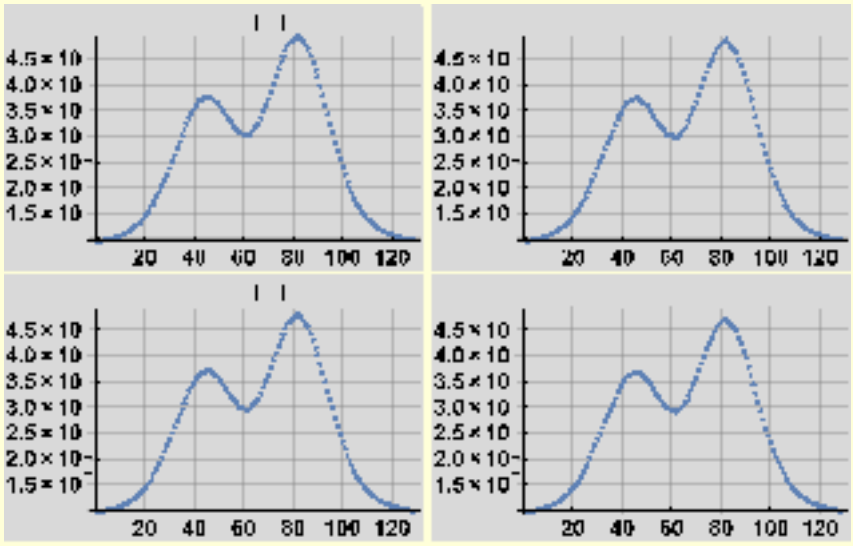
```
GaAs = {"GaAs", "Ga", "As", "Conf7", "ADF1"};
rules = Thread [vars → GaAs];
readAndDisplayEMSImage [vars, rules]
analyzeProfile[vars, rules, 32]
```

```
Data directory : /Users/pierrestadelmann/Desktop/III-V/GaAs/Conf7ADF1
Listing .ems files of directory /Users/pierrestadelmann/Desktop/III-V/GaAs/Conf7ADF1
Reading : /Users/pierrestadelmann/Desktop/III-V/GaAs/Conf7ADF1/DF_0255.ems
```



```
Data directory : /Users/pierrestadelmann/Desktop/III-V/GaAs/Conf7ADF1
```

```
Reading : /Users/pierrestadelmann/Desktop/III-V/GaAs/Conf7ADF1/DF_0255.ems
```

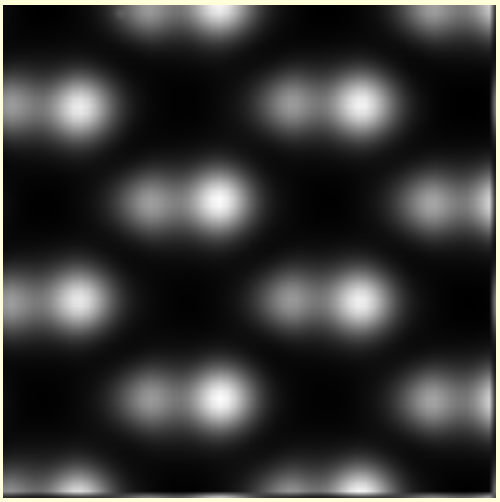



```
GaAs = {"GaAs", "Ga", "As", "Conf7", "ADF2"};
rules = Thread [vars → GaAs];
readAndDisplayEMSImage [vars, rules]
analyzeProfile[vars, rules, 32]
```

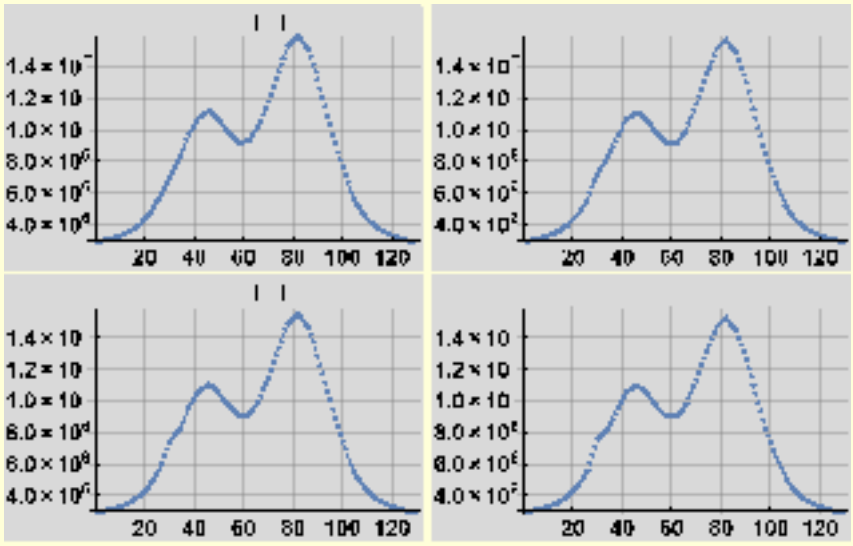
```
Data directory : /Users/pierrestadelmann/Desktop/III-V/GaAs/Conf7ADF2
```

```
Listing .ems files of directory /Users/pierrestadelmann/Desktop/III-V/GaAs/Conf7ADF2
```

```
Reading : /Users/pierrestadelmann/Desktop/III-V/GaAs/Conf7ADF2/DF_0255.ems
```

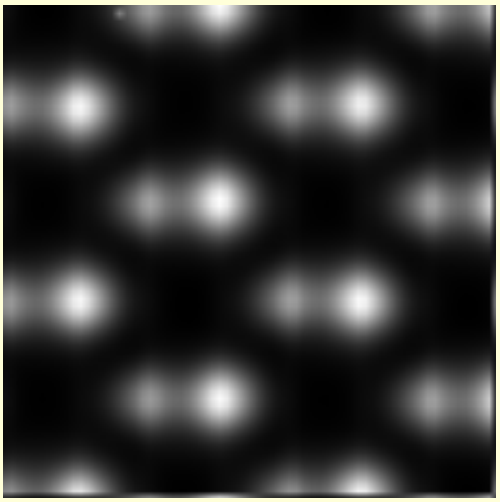


```
Data directory : /Users/pierrestadelmann/Desktop/III-V/GaAs/Conf7ADF2
Reading : /Users/pierrestadelmann/Desktop/III-V/GaAs/Conf7ADF2/DF_0255.ems
```

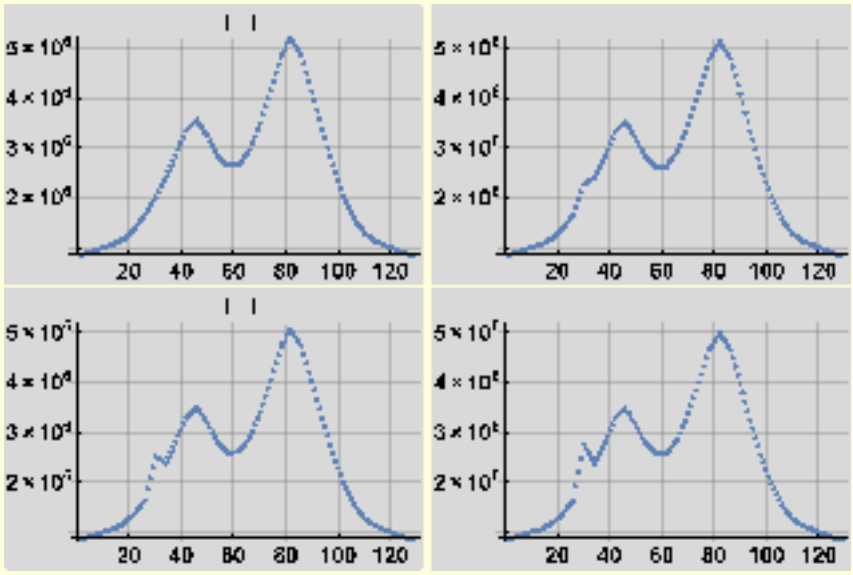


```
GaAs = {"GaAs", "Ga", "As", "Conf7", "ADF3"};
rules = Thread [vars → GaAs];
readAndDisplayEMSImage [vars, rules]
analyzeProfile[vars, rules, 32]
```

```
Data directory : /Users/pierrestadelmann/Desktop/III-V/GaAs/Conf7ADF3
Listing .ems files of directory /Users/pierrestadelmann/Desktop/III-V/GaAs/Conf7ADF3
Reading : /Users/pierrestadelmann/Desktop/III-V/GaAs/Conf7ADF3/DF_0255.ems
```

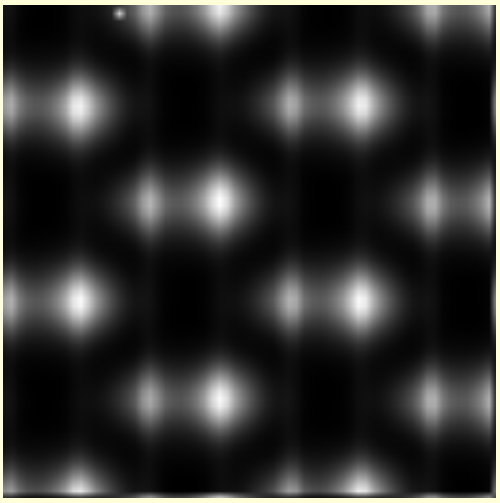


```
Data directory : /Users/pierrestadelmann/Desktop/III-V/GaAs/Conf7ADF3
Reading : /Users/pierrestadelmann/Desktop/III-V/GaAs/Conf7ADF3/DF_0255.ems
```

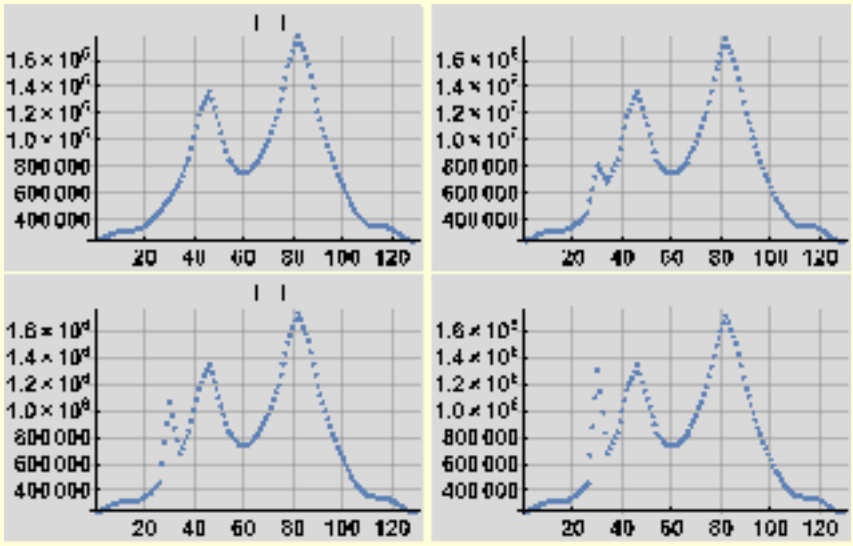



```
GaAs = {"GaAs", "Ga", "As", "Conf7", "ADF4"};
rules = Thread [vars → GaAs];
readAndDisplayEMSImage [vars, rules]
analyzeProfile[vars, rules, 32]
```

```
Data directory : /Users/pierrestadelmann/Desktop/III-V/GaAs/Conf7ADF4
Listing .ems files of directory /Users/pierrestadelmann/Desktop/III-V/GaAs/Conf7ADF4
Reading : /Users/pierrestadelmann/Desktop/III-V/GaAs/Conf7ADF4/DF_0255.ems
```



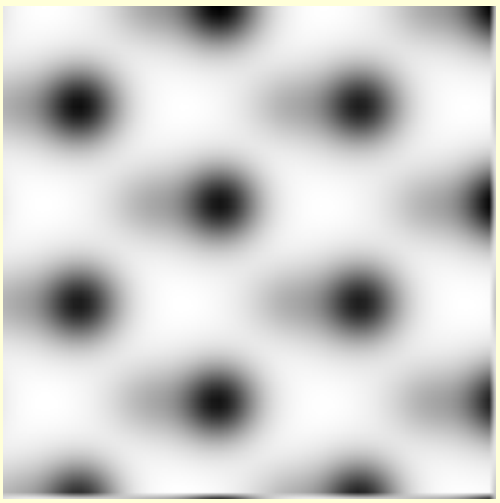
```
Data directory : /Users/pierrestadelmann/Desktop/III-V/GaAs/Conf7ADF4
Reading : /Users/pierrestadelmann/Desktop/III-V/GaAs/Conf7ADF4/DF_0255.ems
```



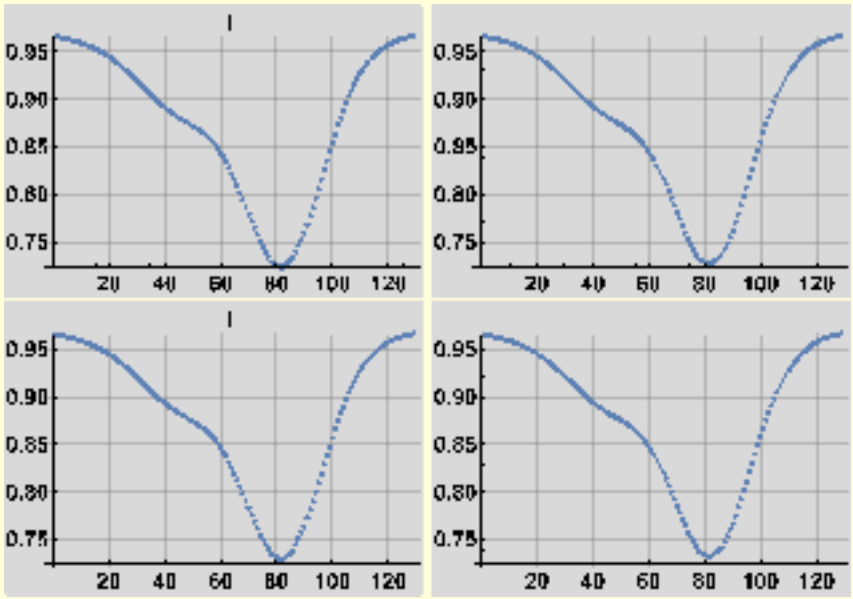
6. GaP

```
GaP = {"GaP", "Ga", "P", "Conf5", "BF1"};
rules = Thread [vars → GaP];
readAndDisplayEMSImage [vars, rules]
analyzeProfile[vars, rules, 32]
```

```
Data directory : /Users/pierrestadelmann/Desktop/III-V/GaP/Conf5BF1
Listing .ems files of directory /Users/pierrestadelmann/Desktop/III-V/GaP/Conf5BF1
Reading : /Users/pierrestadelmann/Desktop/III-V/GaP/Conf5BF1/BF_0127.ems
```



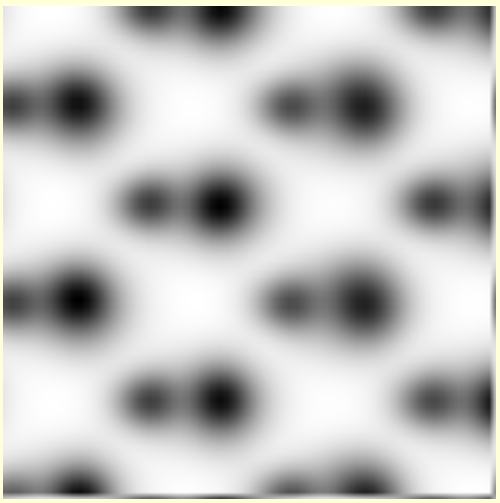
```
Data directory : /Users/pierrestadelmann/Desktop/III-V/GaP/Conf5BF1
Reading : /Users/pierrestadelmann/Desktop/III-V/GaP/Conf5BF1/BF_0127.ems
```



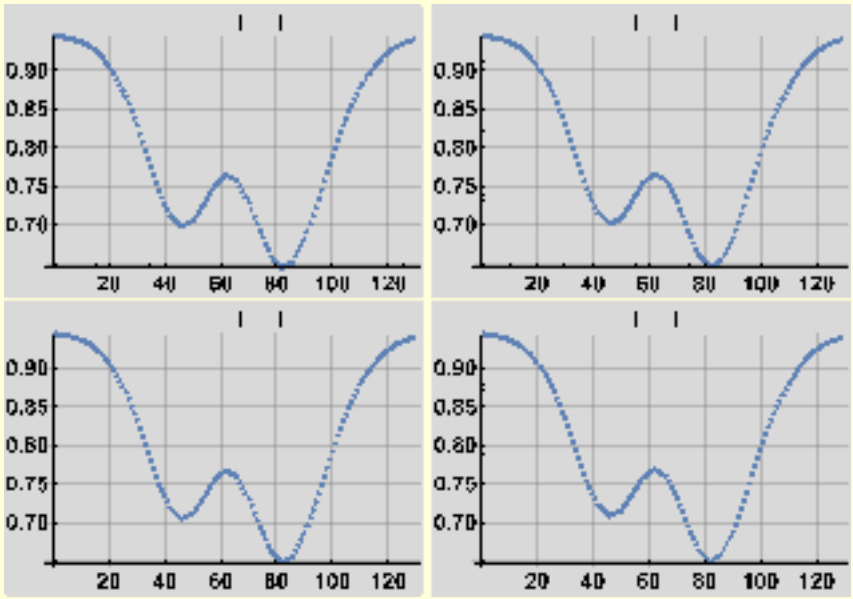
7. InAs

```
InAs = {"InAs", "In", "As", "Conf5", "BF1"};
rules = Thread [vars → InAs];
readAndDisplayEMSImage [vars, rules]
analyzeProfile[vars, rules, 32]
```

```
Data directory : /Users/pierrestadelmann/Desktop/III-V/InAs/Conf5BF1
Listing .ems files of directory /Users/pierrestadelmann/Desktop/III-V/InAs/Conf5BF1
Reading : /Users/pierrestadelmann/Desktop/III-V/InAs/Conf5BF1/BF_0111.ems
```



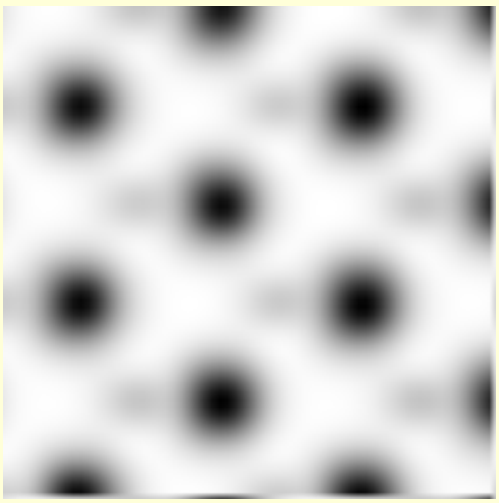
```
Data directory : /Users/pierrestadelmann/Desktop/III-V/InAs/Conf5BF1
Reading : /Users/pierrestadelmann/Desktop/III-V/InAs/Conf5BF1/BF_0111.ems
```



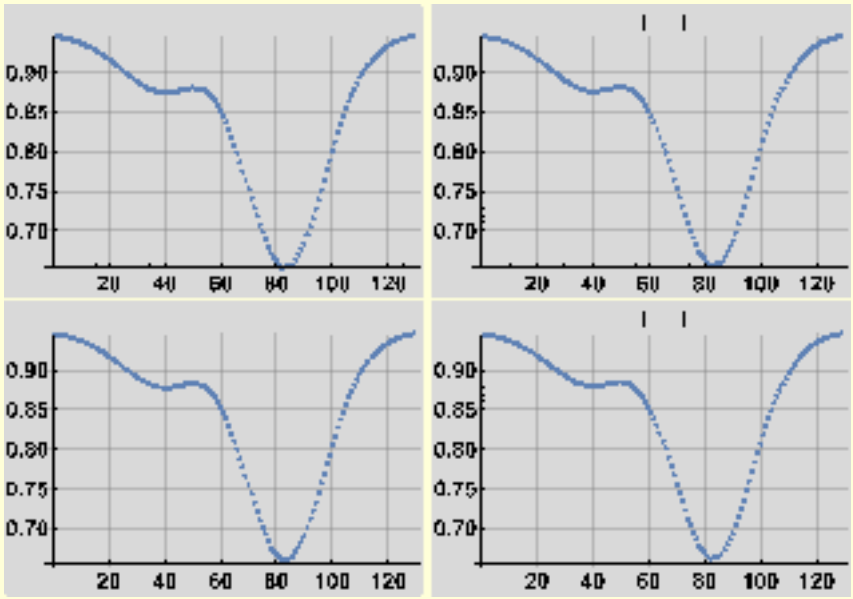
8. InP

```
InP = {"InP", "In", "P", "Conf5", "BF1"};
rules = Thread [vars → InP];
readAndDisplayEMSImage [vars, rules]
analyzeProfile[vars, rules, 32]
```

```
Data directory : /Users/pierrestadelmann/Desktop/III-V/InP/Conf5BF1
Listing .ems files of directory /Users/pierrestadelmann/Desktop/III-V/InP/Conf5BF1
Reading : /Users/pierrestadelmann/Desktop/III-V/InP/Conf5BF1/BF_0117.ems
```



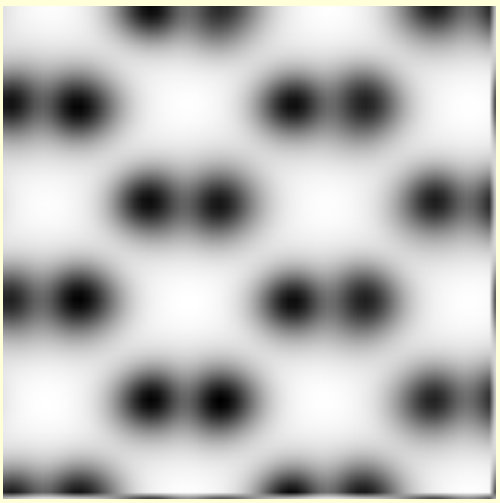
```
Data directory : /Users/pierrestadelmann/Desktop/III-V/InP/Conf5BF1
Reading : /Users/pierrestadelmann/Desktop/III-V/InP/Conf5BF1/BF_0117.ems
```



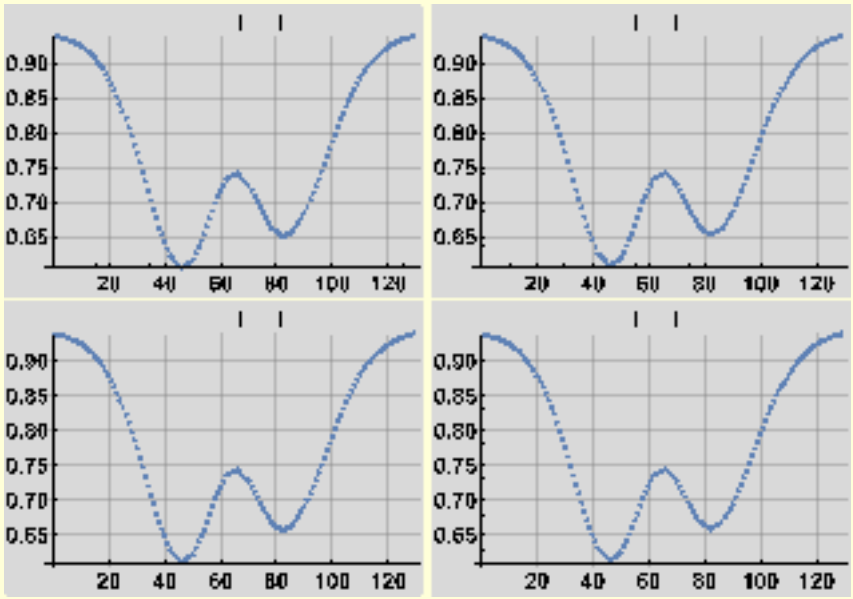
9. InSb

```
InSb = {"InSb", "In", "Sb", "Conf5", "BF1"};
rules = Thread [vars → InSb];
readAndDisplayEMSImage [vars, rules]
analyzeProfile[vars, rules, 32]
```

```
Data directory : /Users/pierrestadelmann/Desktop/III-V/InSb/Conf5BF1
Listing .ems files of directory /Users/pierrestadelmann/Desktop/III-V/InSb/Conf5BF1
Reading : /Users/pierrestadelmann/Desktop/III-V/InSb/Conf5BF1/BF_0105.ems
```



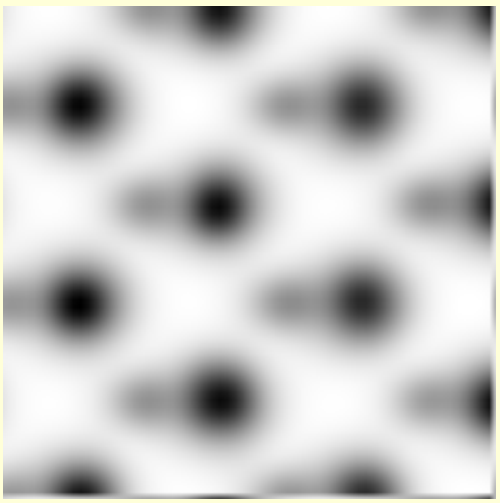
```
Data directory : /Users/pierrestadelmann/Desktop/III-V/InSb/Conf5BF1
Reading : /Users/pierrestadelmann/Desktop/III-V/InSb/Conf5BF1/BF_0105.ems
```



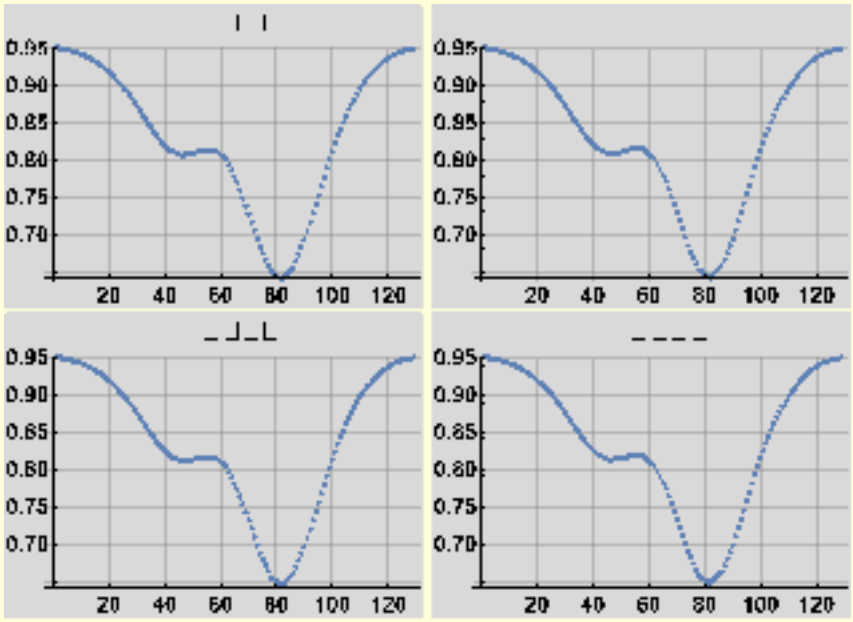
10. ZnTe

```
ZnTe = {"ZnTe", "Zn", "Te", "Conf5", "BF1"};
rules = Thread [vars → ZnTe];
readAndDisplayEMSImage [vars, rules]
analyzeProfile[vars, rules, 32]
```

```
Data directory : /Users/pierrestadelmann/Desktop/III-V/ZnTe/Conf5BF1
Listing .ems files of directory /Users/pierrestadelmann/Desktop/III-V/ZnTe/Conf5BF1
Reading : /Users/pierrestadelmann/Desktop/III-V/ZnTe/Conf5BF1/BF_0111.ems
```

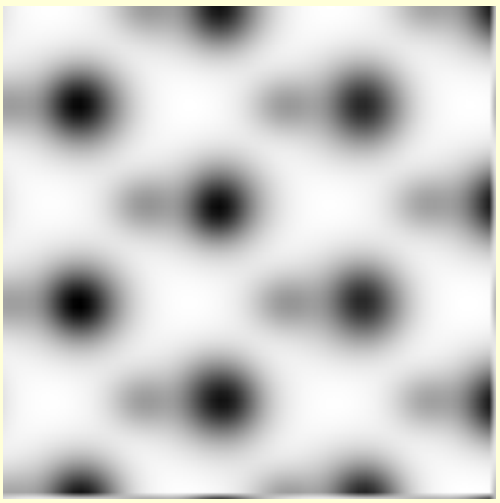


```
Data directory : /Users/pierrestadelmann/Desktop/III-V/ZnTe/Conf5BF1
Reading : /Users/pierrestadelmann/Desktop/III-V/ZnTe/Conf5BF1/BF_0111.ems
```

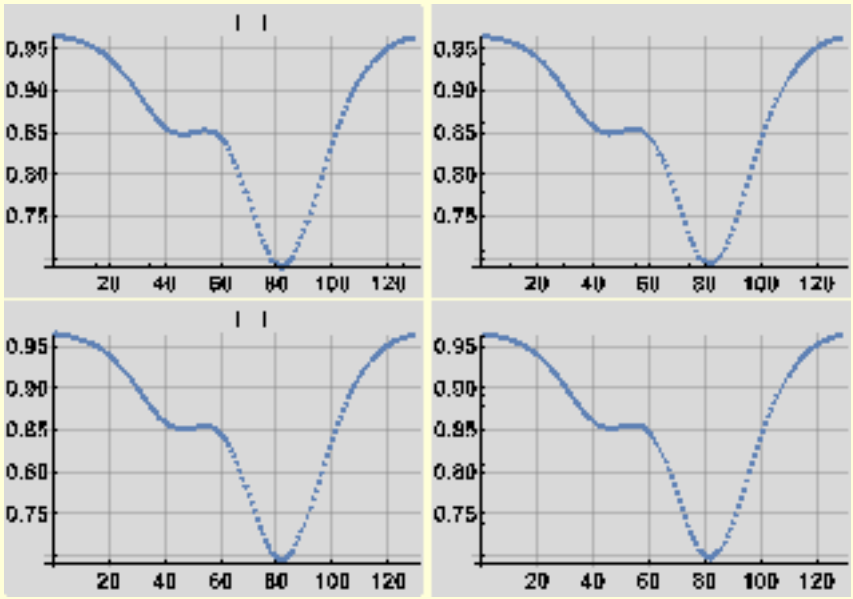



```
ZnTe = {"ZnTe", "Zn", "Te", "Conf5", "BF2"};
rules = Thread [vars → ZnTe];
readAndDisplayEMSImage [vars, rules]
analyzeProfile[vars, rules, 32]
```

```
Data directory : /Users/pierrestadelmann/Desktop/III-V/ZnTe/Conf5BF2
Listing .ems files of directory /Users/pierrestadelmann/Desktop/III-V/ZnTe/Conf5BF2
Reading : /Users/pierrestadelmann/Desktop/III-V/ZnTe/Conf5BF2/BF_0111.ems
```

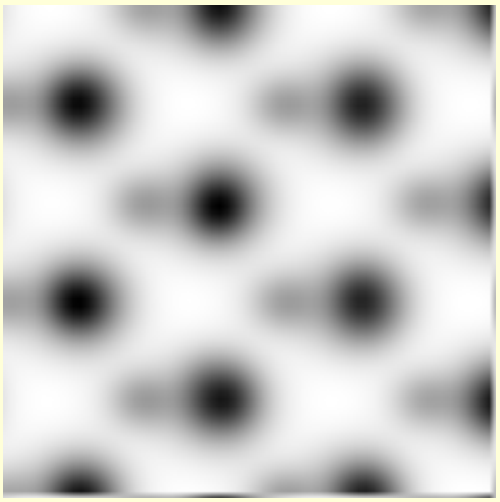


```
Data directory : /Users/pierrestadelmann/Desktop/III-V/ZnTe/Conf5BF2
Reading : /Users/pierrestadelmann/Desktop/III-V/ZnTe/Conf5BF2/BF_0111.ems
```

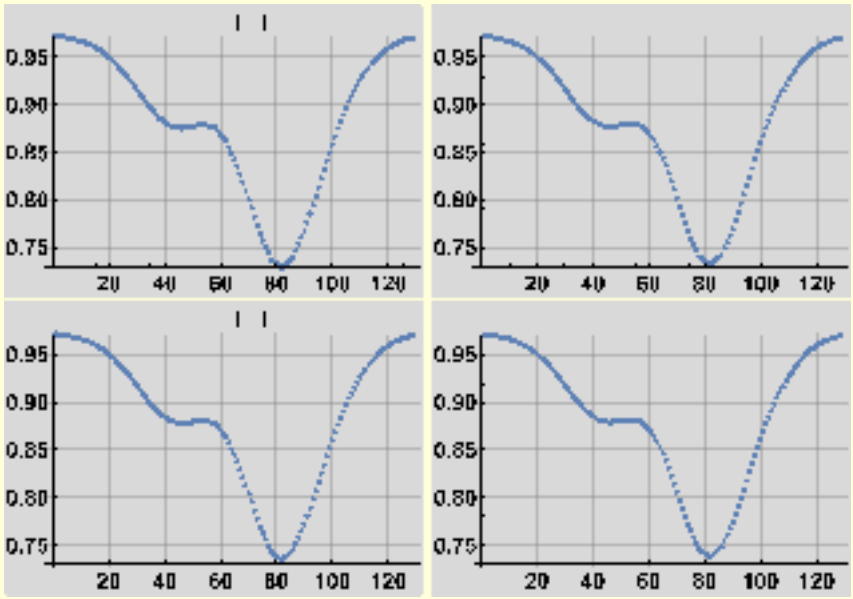



```
ZnTe = {"ZnTe", "Zn", "Te", "Conf5", "BF3"};
rules = Thread [vars → ZnTe];
readAndDisplayEMSImage [vars, rules]
analyzeProfile[vars, rules, 32]
```

```
Data directory : /Users/pierrestadelmann/Desktop/III-V/ZnTe/Conf5BF3
Listing .ems files of directory /Users/pierrestadelmann/Desktop/III-V/ZnTe/Conf5BF3
Reading : /Users/pierrestadelmann/Desktop/III-V/ZnTe/Conf5BF3/BF_0111.ems
```

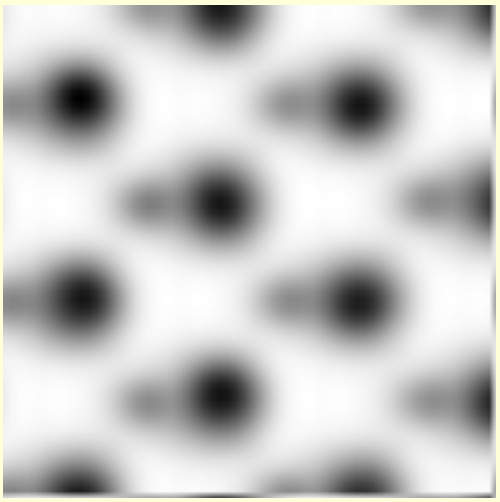


```
Data directory : /Users/pierrestadelmann/Desktop/III-V/ZnTe/Conf5BF3
Reading : /Users/pierrestadelmann/Desktop/III-V/ZnTe/Conf5BF3/BF_0111.ems
```

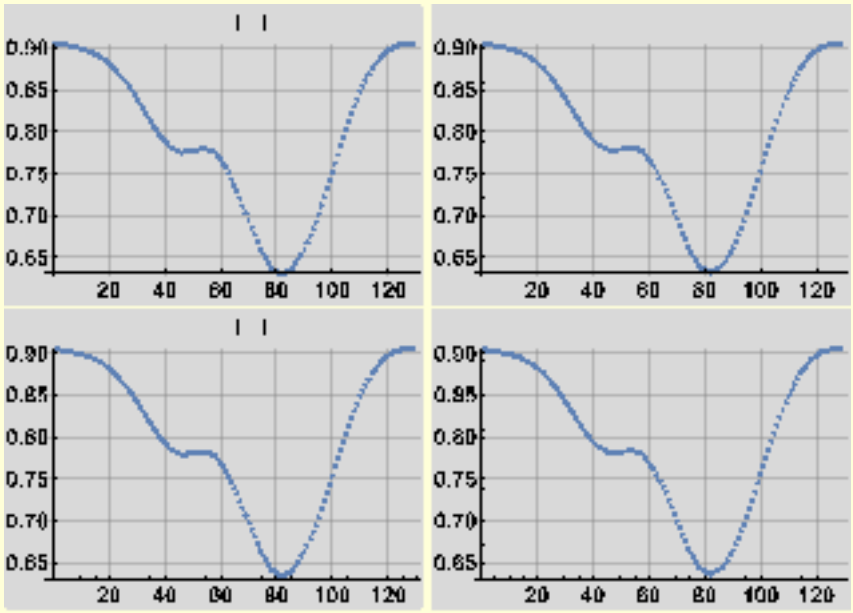



```
ZnTe = {"ZnTe", "Zn", "Te", "Conf5", "BF4"};
rules = Thread [vars → ZnTe];
readAndDisplayEMSImage [vars, rules]
analyzeProfile[vars, rules, 32]
```

```
Data directory : /Users/pierrestadelmann/Desktop/III-V/ZnTe/Conf5BF4
Listing .ems files of directory /Users/pierrestadelmann/Desktop/III-V/ZnTe/Conf5BF4
Reading : /Users/pierrestadelmann/Desktop/III-V/ZnTe/Conf5BF4/BF_0275.ems
```



```
Data directory : /Users/pierrestadelmann/Desktop/III-V/ZnTe/Conf5BF4
Reading : /Users/pierrestadelmann/Desktop/III-V/ZnTe/Conf5BF4/BF_0275.ems
```



11. Analyzing all III-V at once

We quit the kernel and do the analysis at once :

```
Exit[]
```

We make a list of 3 - 5 :

```
s35 = {{"AlAs", "Al", "As", "Conf5", "BF1"},
    {"AlP", "Al", "P", "Conf5", "BF1"},
    {"AlSb", "Al", "Sb", "Conf5", "BF1"},
    {"GaAs", "Ga", "As", "Conf5", "BF1"},
    {"GaP", "Ga", "P", "Conf5", "BF1"},
    {"InAs", "In", "As", "Conf5", "BF1"},
    {"InP", "In", "P", "Conf5", "BF1"},
    {"InSb", "In", "Sb", "Conf5", "BF1"},
    {"ZnTe", "Zn", "Te", "Conf5", "BF1"}};
```

and define a module that performs the analysis steps :

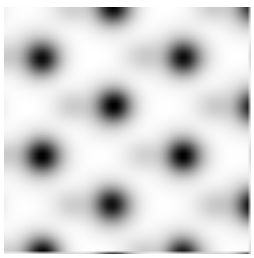
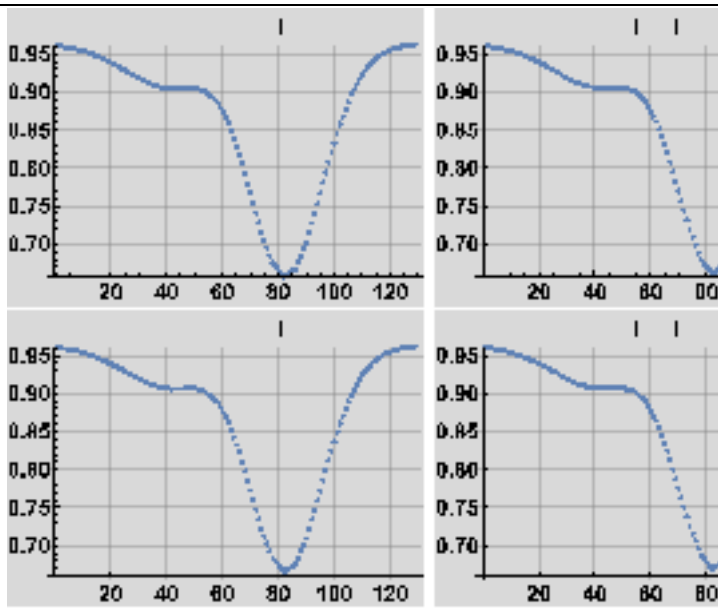
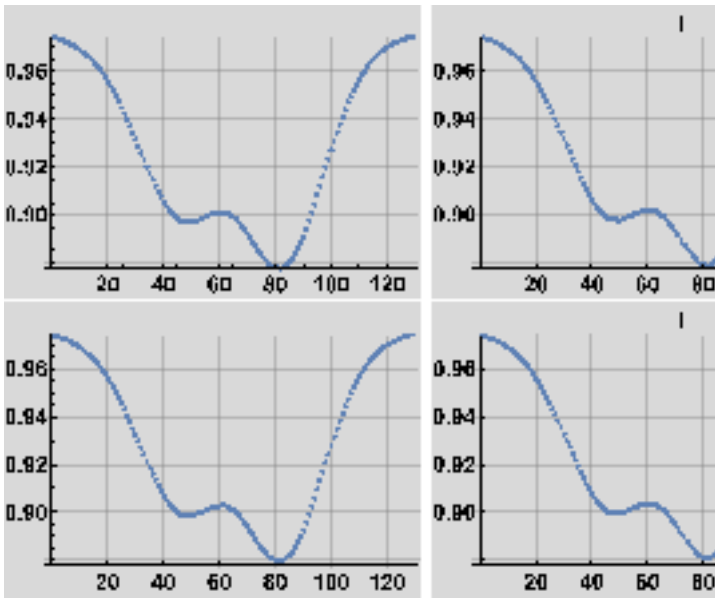
```
analyzeIIIV [sys_] := Module [{image, pdf},
   rules = Thread [vars → sys];
   image = readAndDisplayEMSImage [vars, rules];
   pdf = analyzeProfile[vars, rules, 32];
   {IIIV /. rules, image, pdf}
  ]
```

For AlAs :

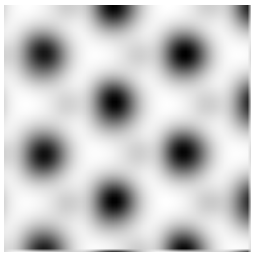
```
analyzeIIIV[s35[[1]]]
```

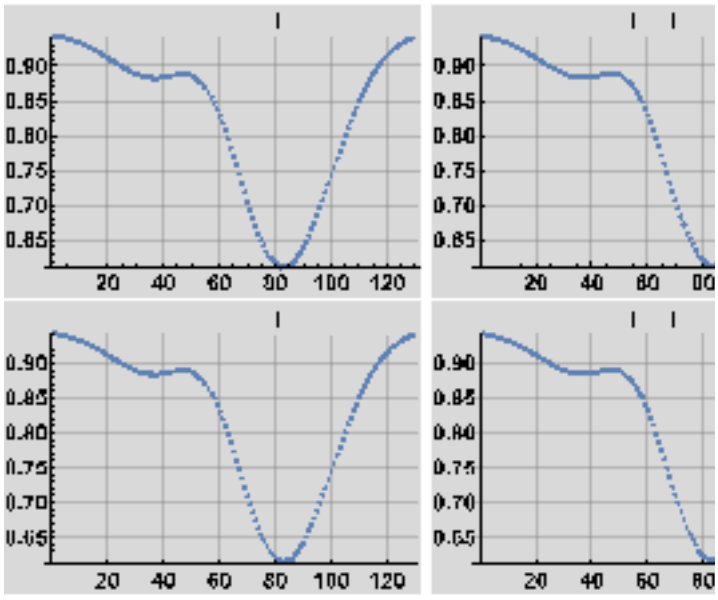
11.1 Detector 1 (30 mrad)

```
(analyzeIIIV[#] & /@ s35) //
 TableForm[#, TableHeadings → {None, {"III-V", "Bright-field", "Profile"}}] &
```

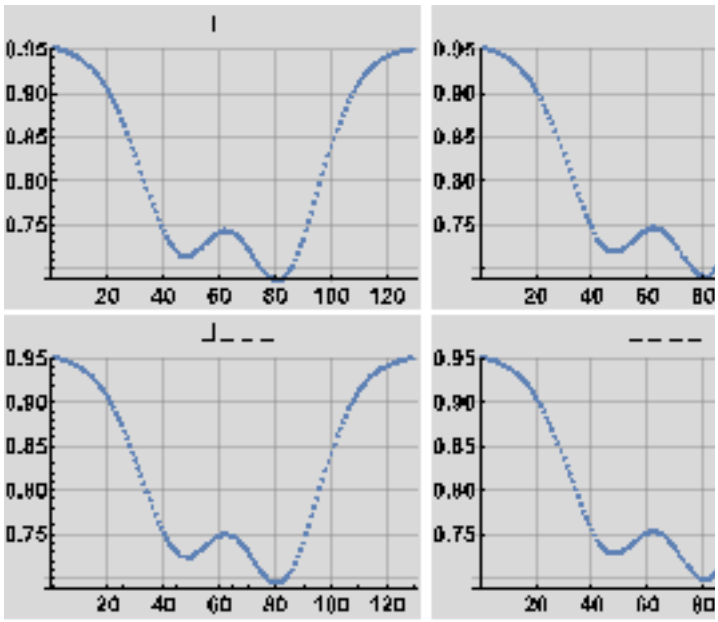
"III-V"	"Bright-field"	"Profile"
"AlAs"		
"AlP"		

"AlSb"

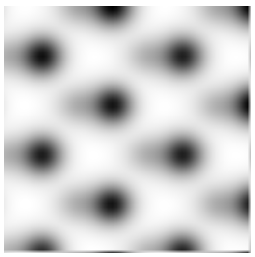


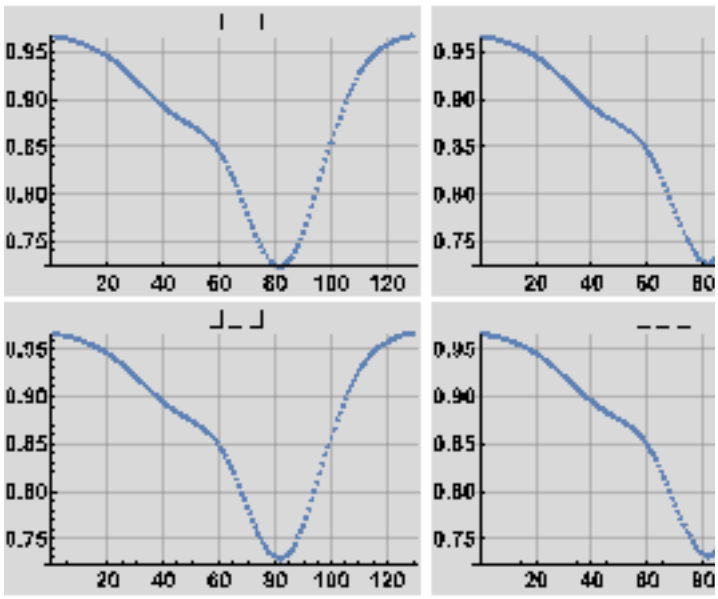


"GaAs"

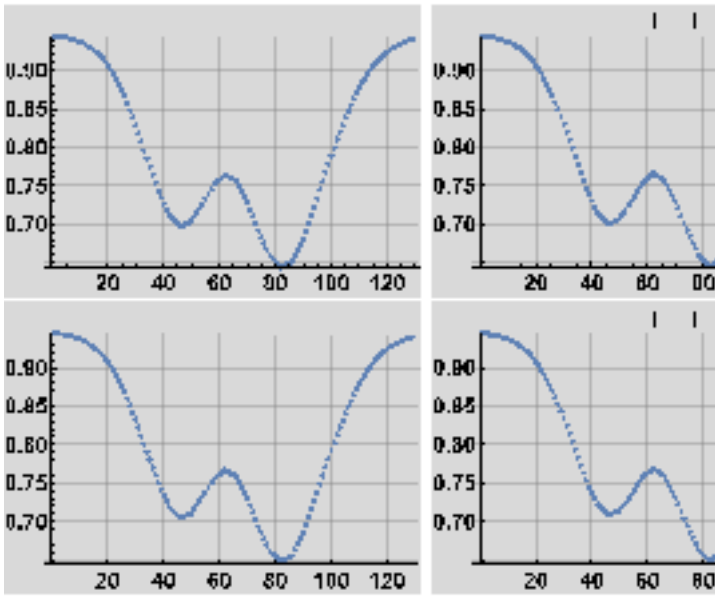


"GaP"

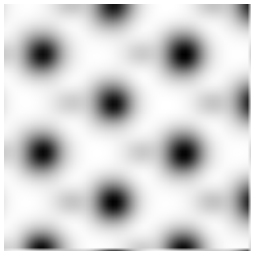


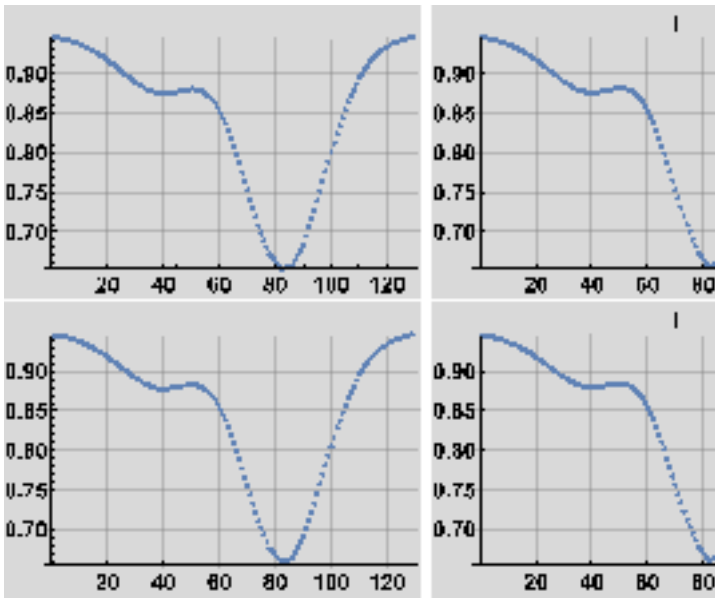


"InAs"

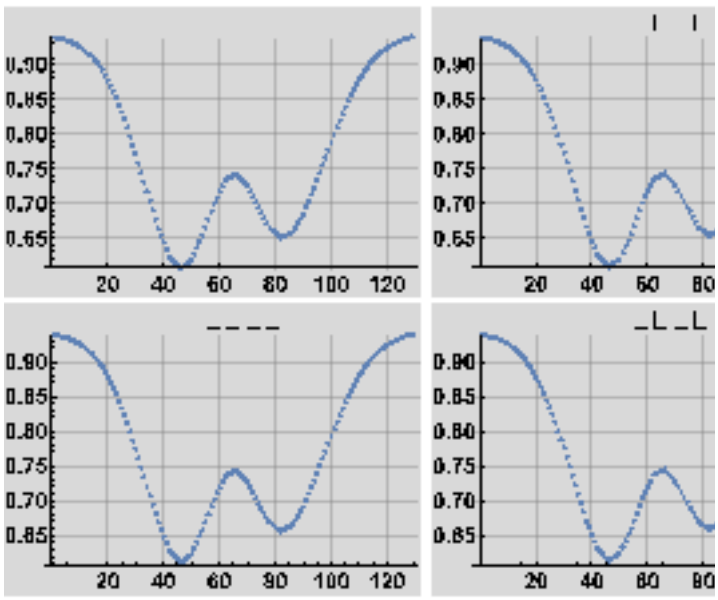


"InP"

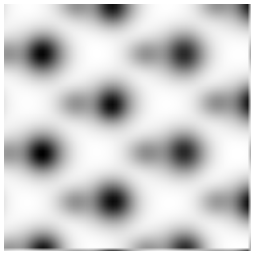


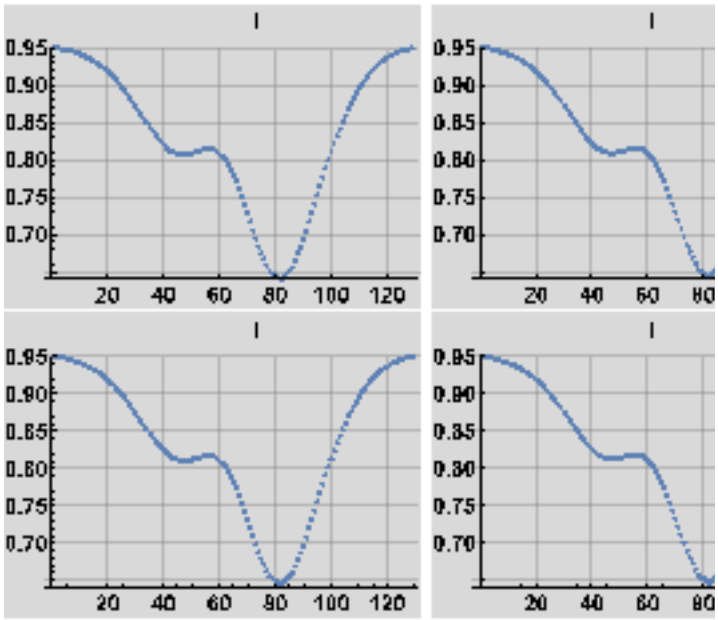


"InSb"



"ZnTe"





Setting a symbol BF in the s35 list and using (s35 /. Detector -> "BF2") one can do the same analysis for detector 2 (30 mrad) :

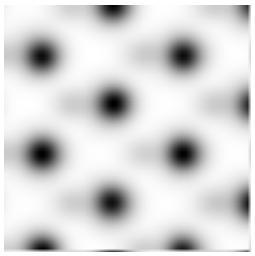
```
s35 = {{"AlAs", "Al", "As", "Conf5", Detector},
    {"AlP", "Al", "P", "Conf5", Detector},
    {"AlSb", "Al", "Sb", "Conf5", Detector},
    {"GaAs", "Ga", "As", "Conf5", BDetector},
    {"GaP", "Ga", "P", "Conf5", Detector},
    {"InAs", "In", "As", "Conf5", Detector},
    {"InP", "In", "P", "Conf5", Detector},
    {"InSb", "In", "Sb", "Conf5", Detector},
    {"ZnTe", "Zn", "Te", "Conf5", Detector}};
```

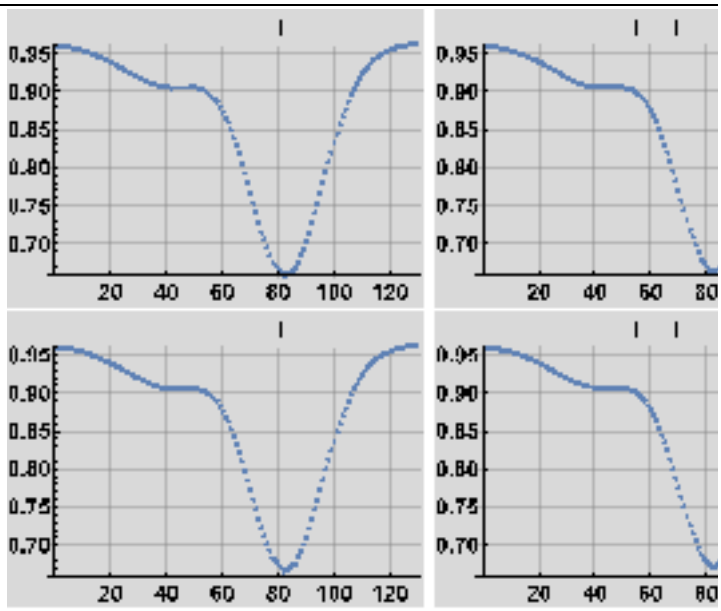
11.2 Detector 2 (30 mrad)

```
(analyzeIIIV[#] & /@ (s35 /. Detector → "BF2")) //
 TableForm[#, TableHeadings → {None, {"III-V", "Bright-field", "Profile"}}] &
```

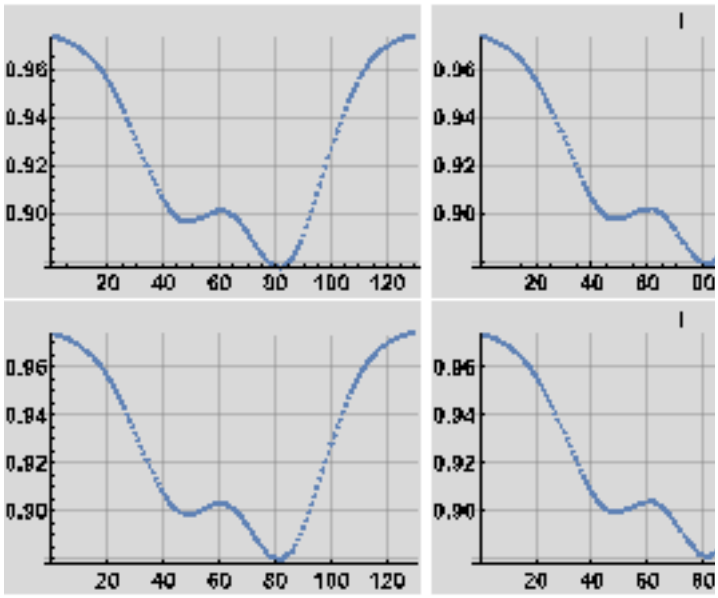
"III-V"	"Bright-field"	"Profile"

"AlAs"

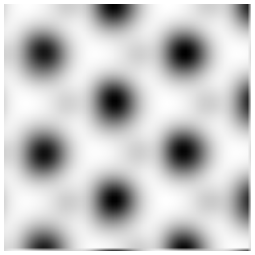


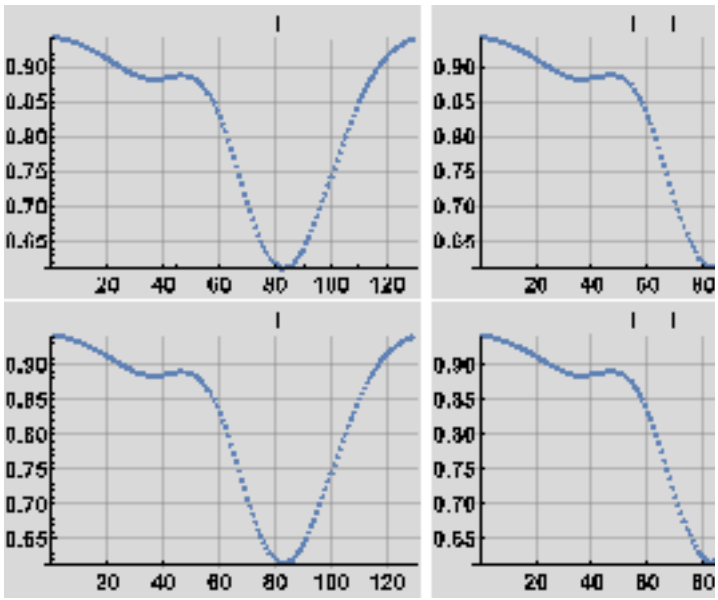


"AlP"

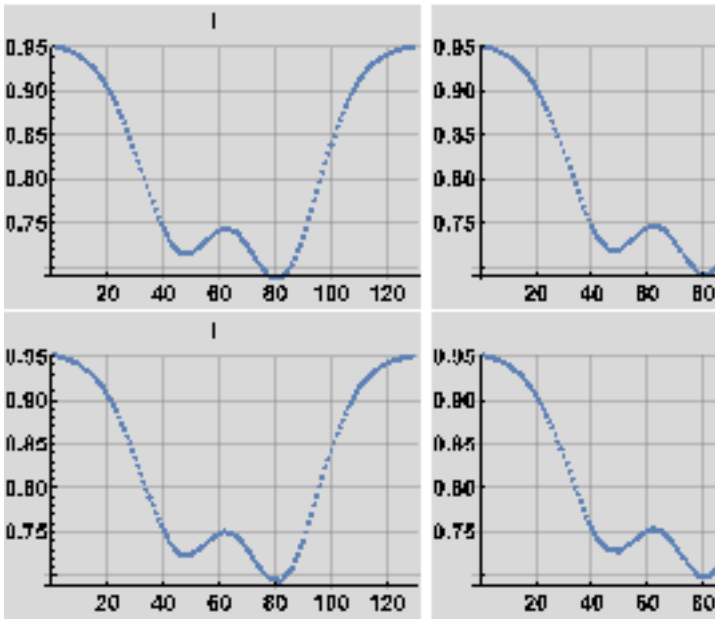


"AlSb"

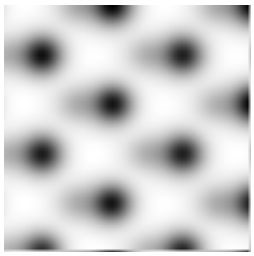


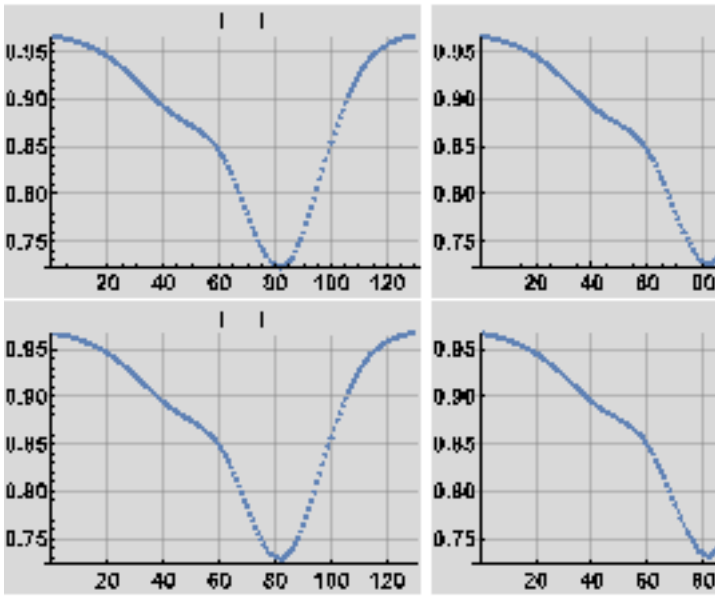


"GaAs"

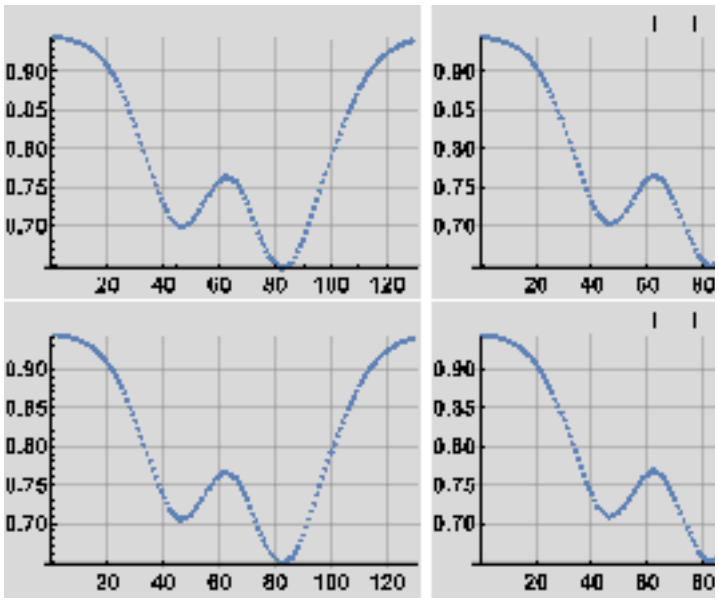


"GaP"

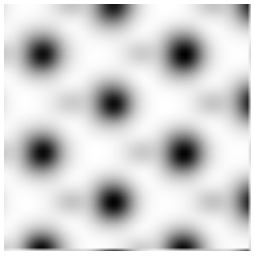


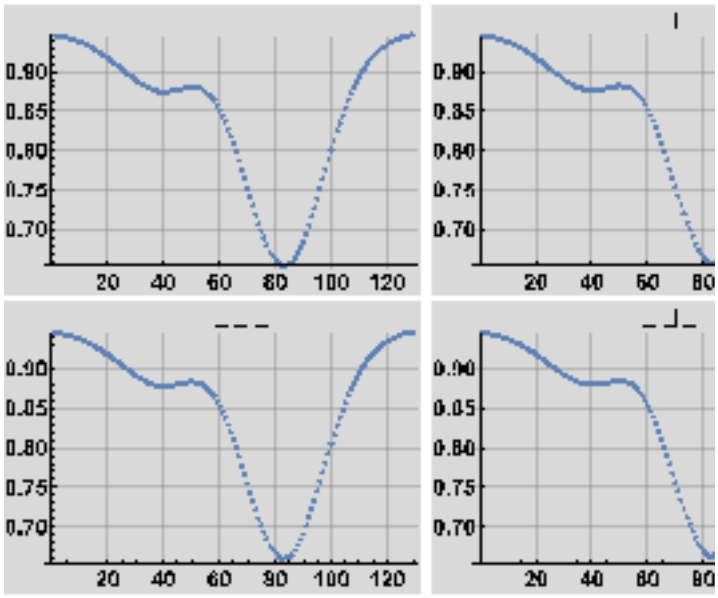


"InAs"

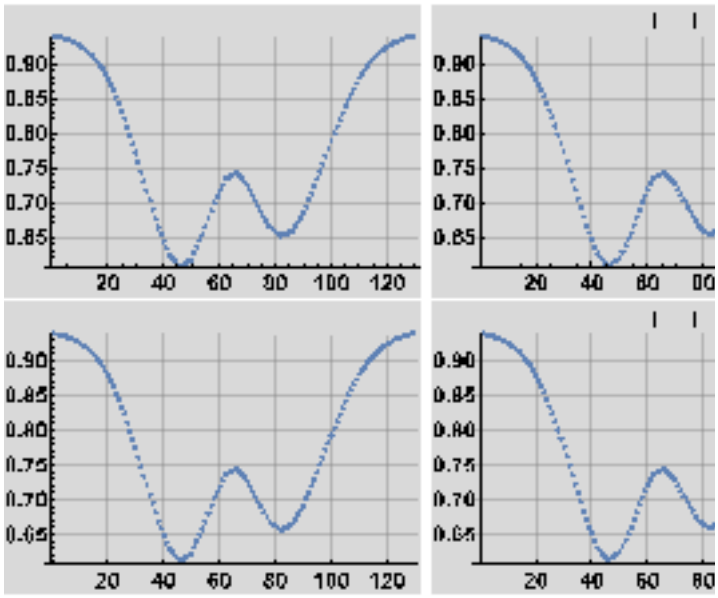


"InP"

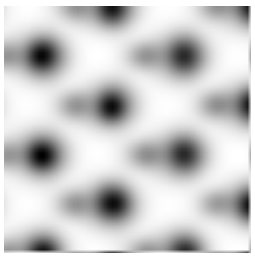


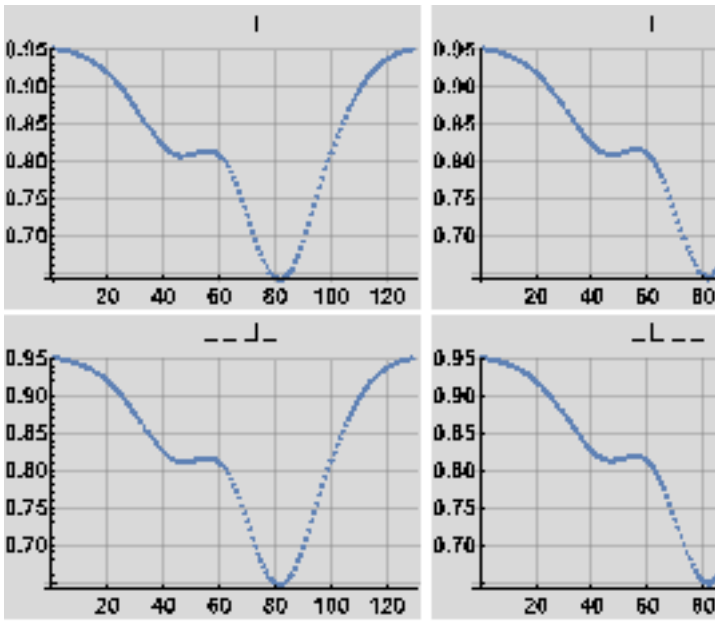


"InSb"



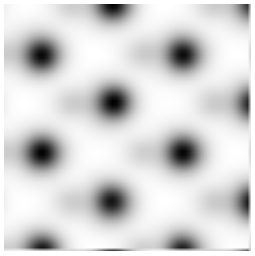
"ZnTe"

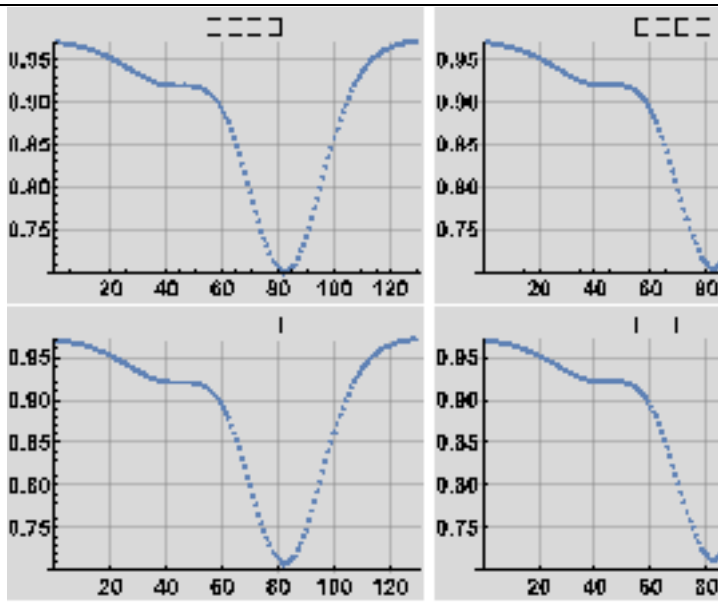




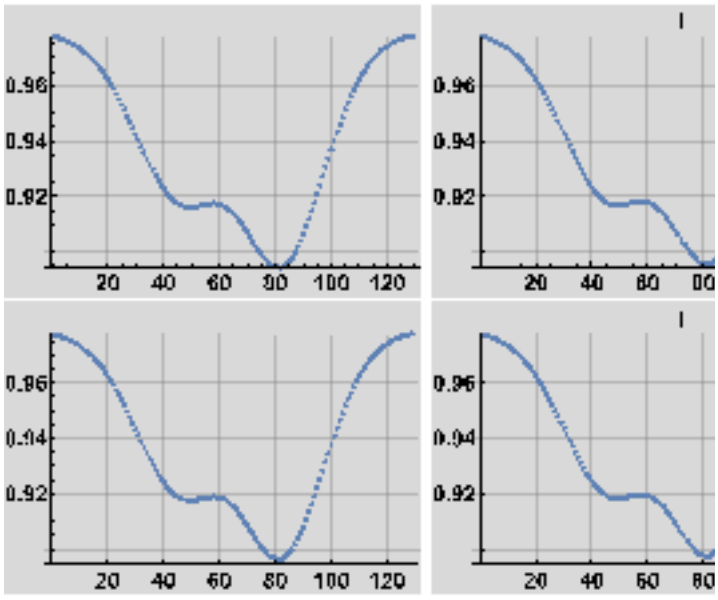
"III-V" "Bright-field" "Profile"

"AlAs"

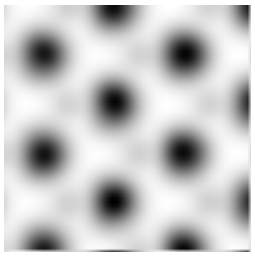


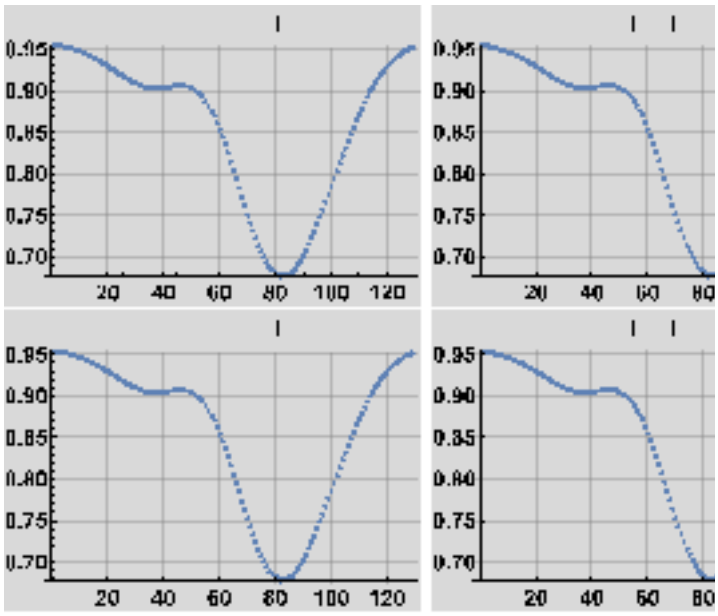


"AlP"

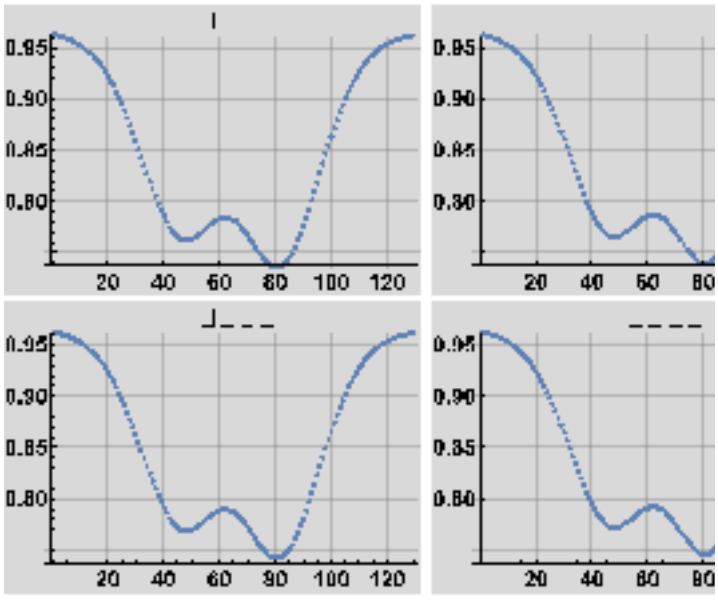


"AlSb"

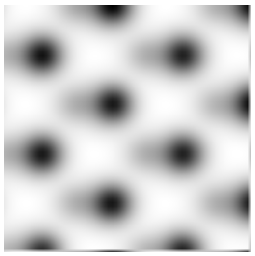


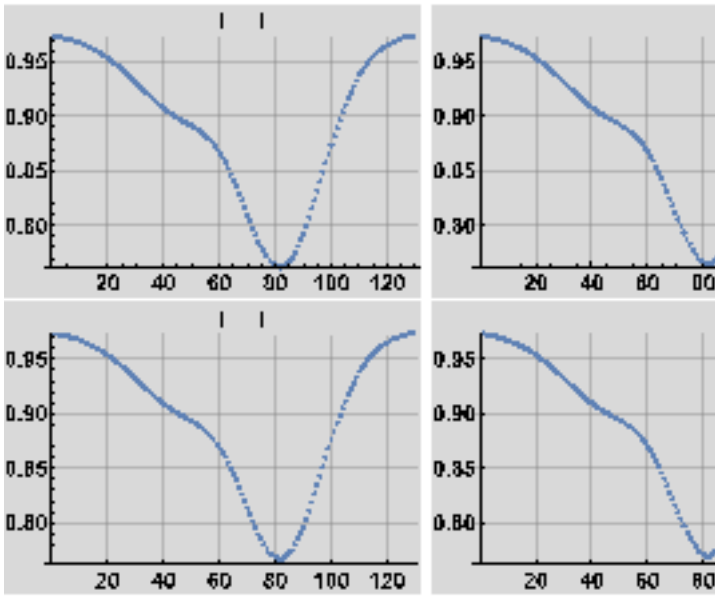


"GaAs"

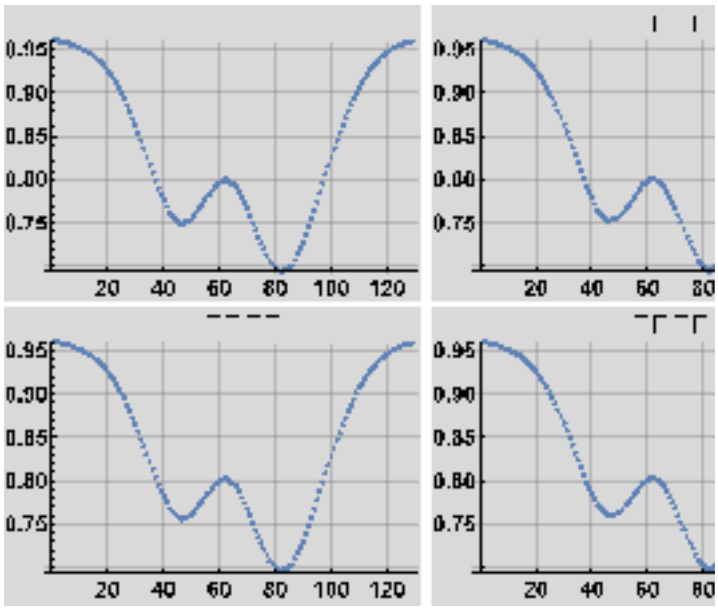


"GaP"

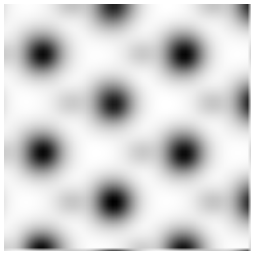


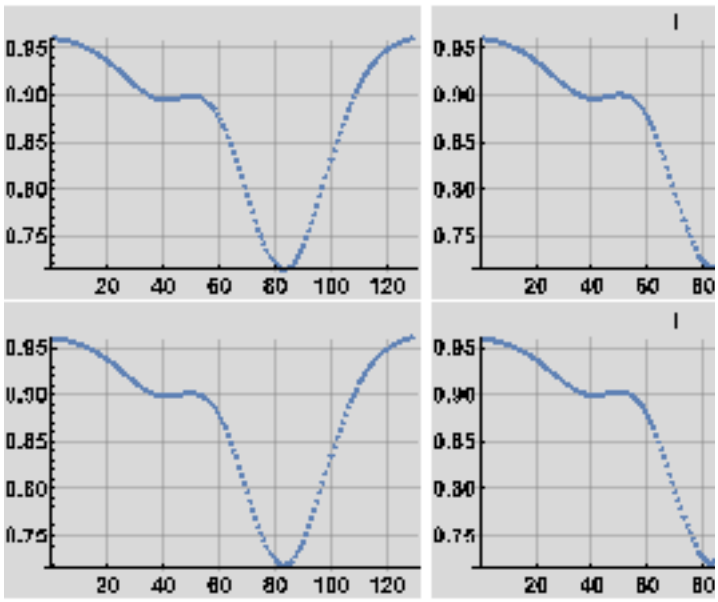


"InAs"

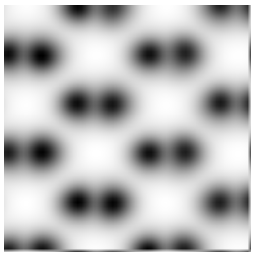


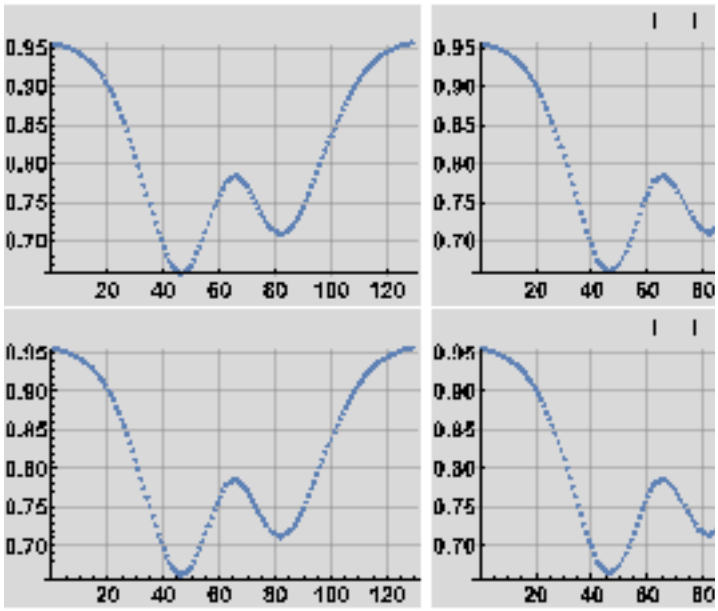
"InP"



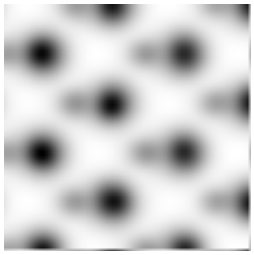


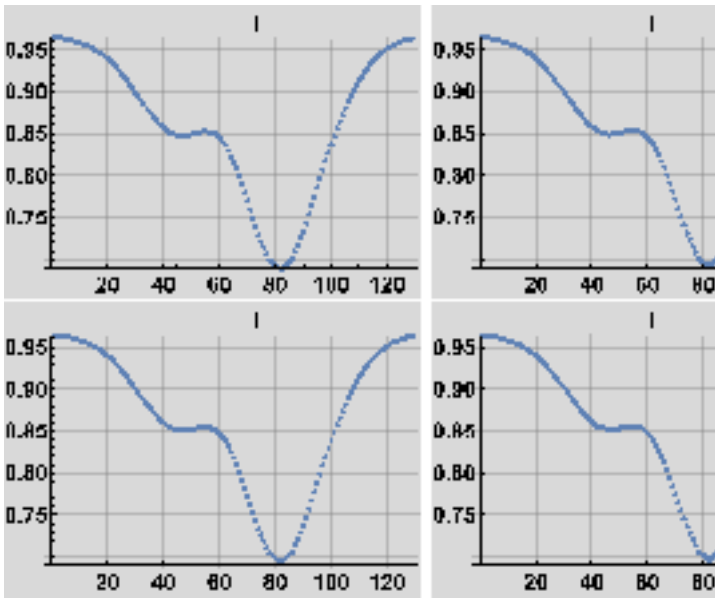
"InSb"





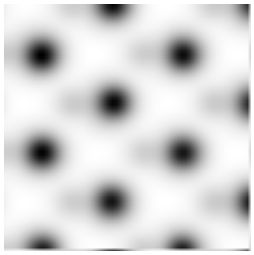
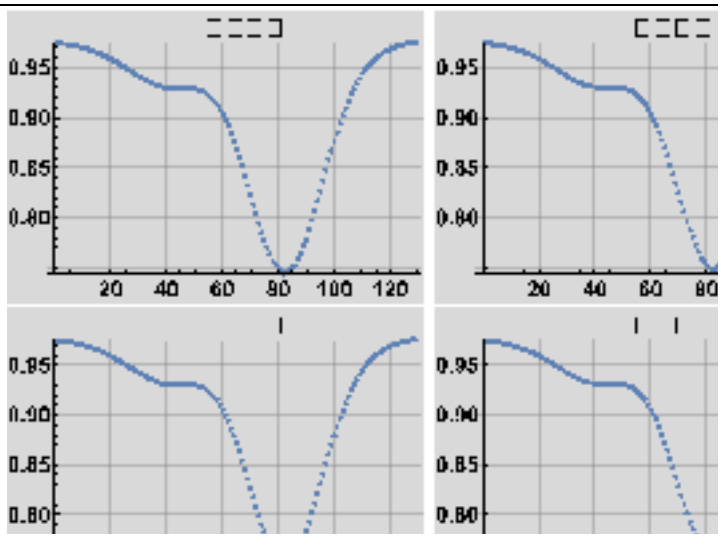
"ZnTe"



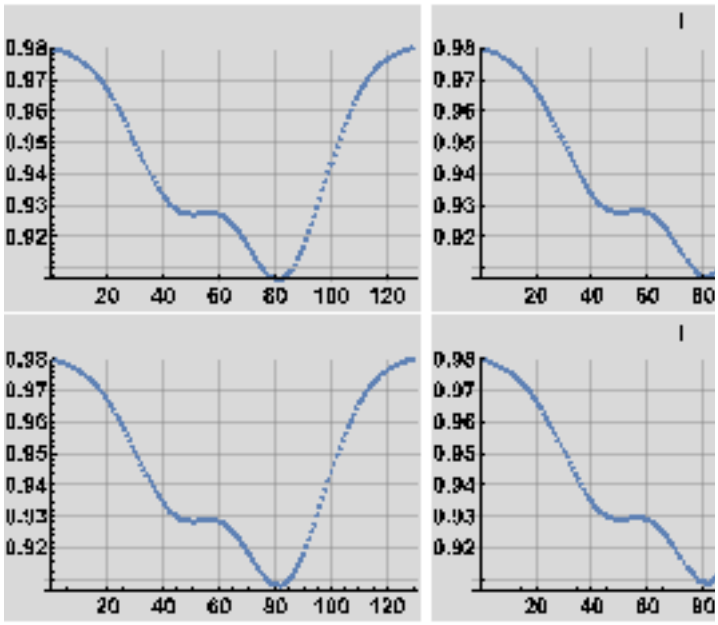


11.3 Detector 3 (35 mrad)

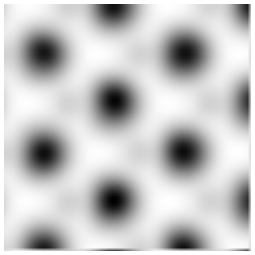
```
(analyzeIIIV[#] & /@ (s35 /. Detector → "BF3")) //
 TableForm[#, TableHeadings → {None, {"III-V", "Bright-field", "Profile"}}] &
```

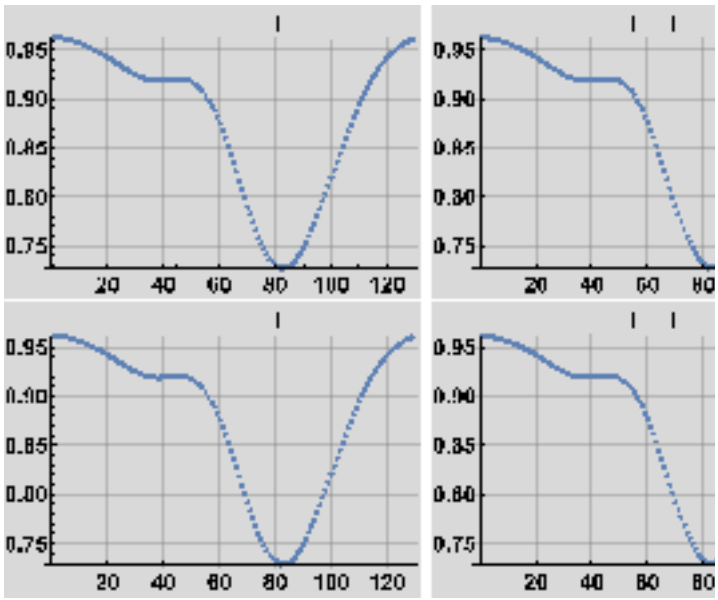
"III-V"	"Bright-field"	"Profile"
"AlAs"		

"AlP"

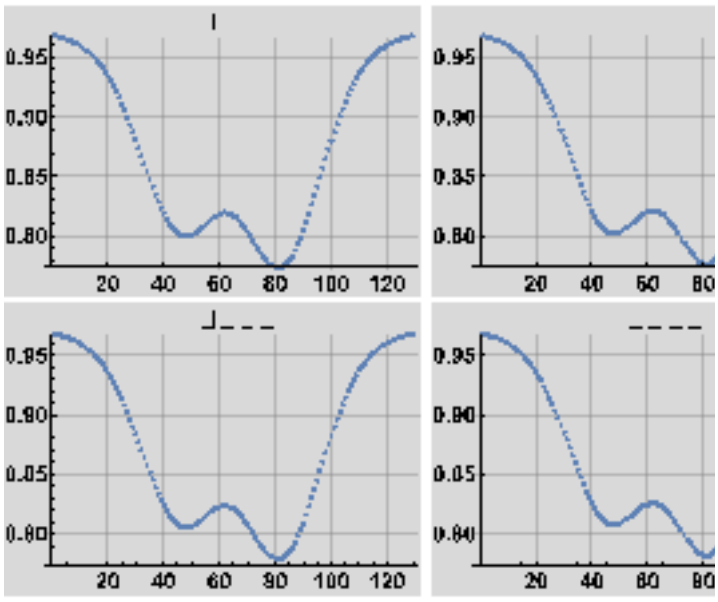


"AlSb"

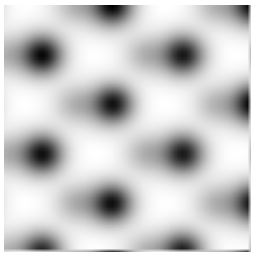


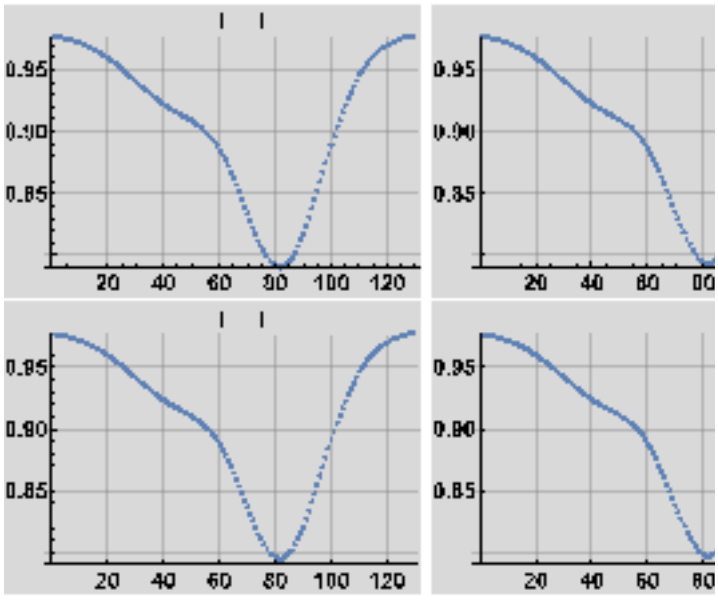


"GaAs"

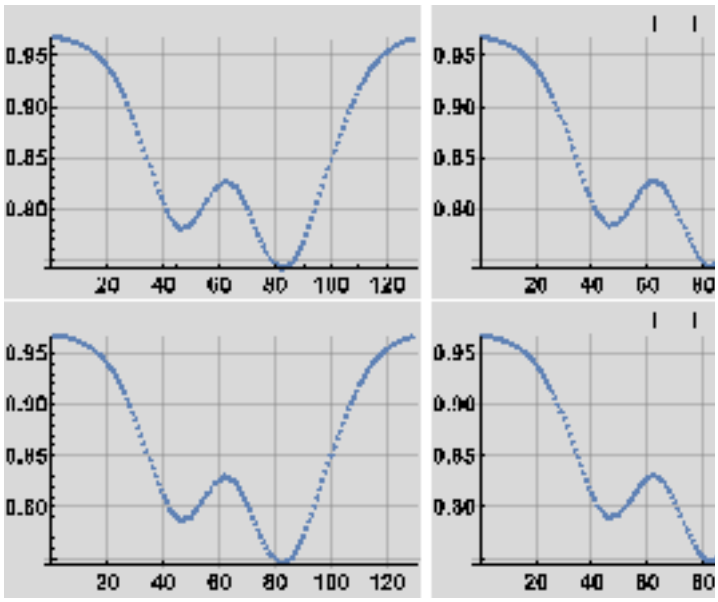


"GaP"

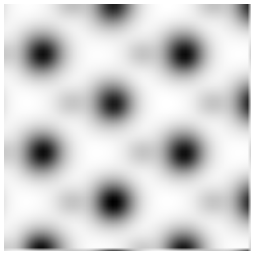


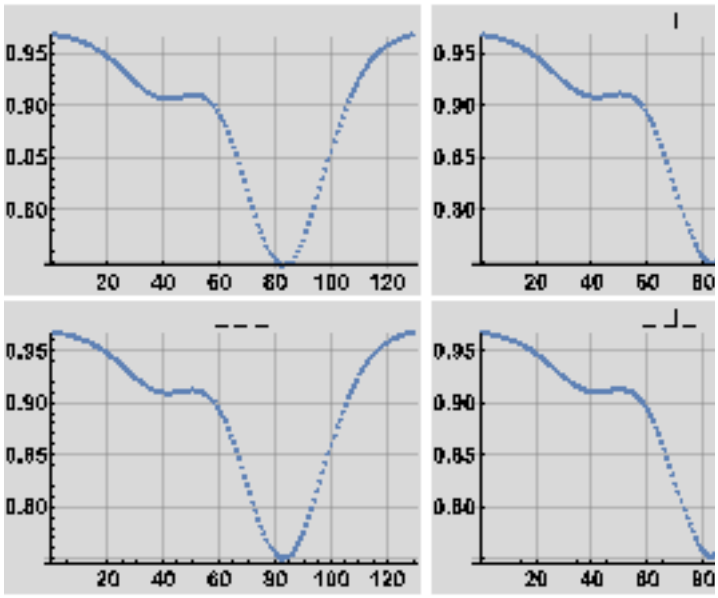


"InAs"

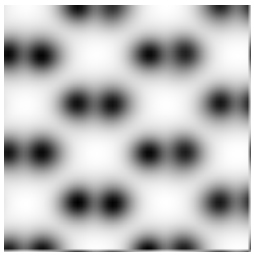


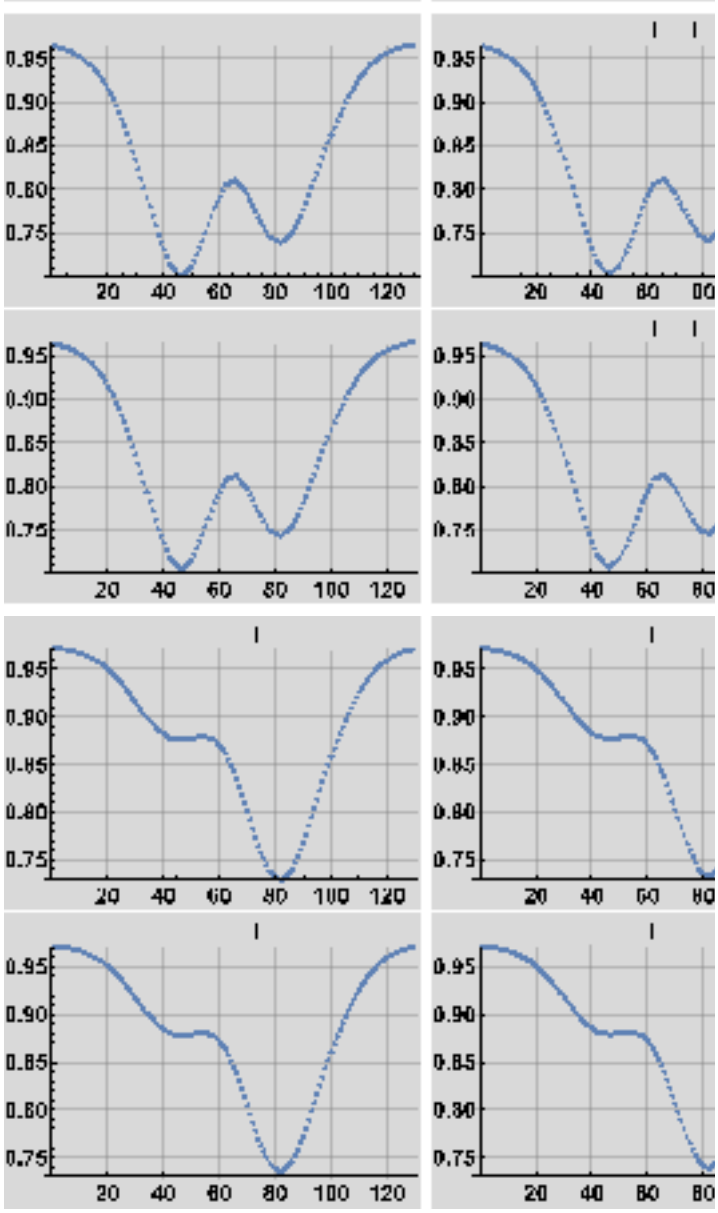
"InP"



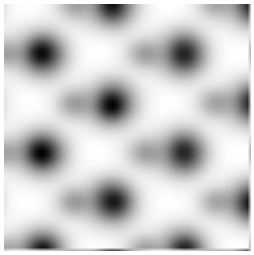


"InSb"



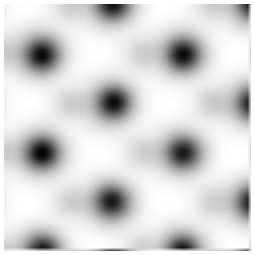
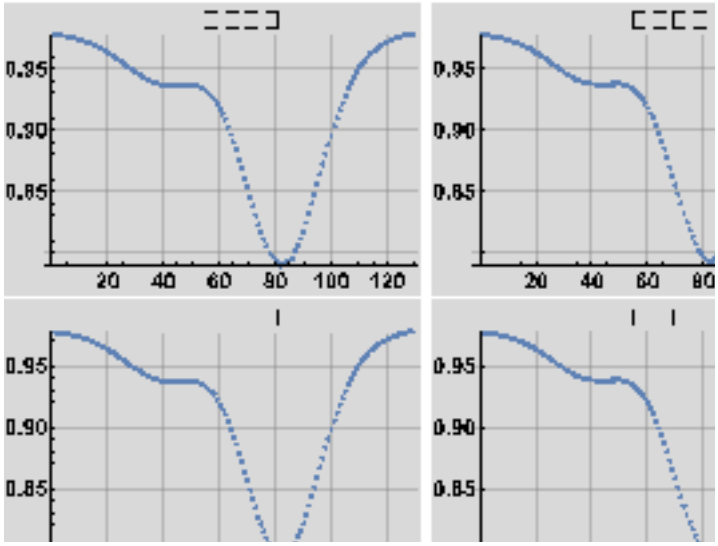


"ZnTe"

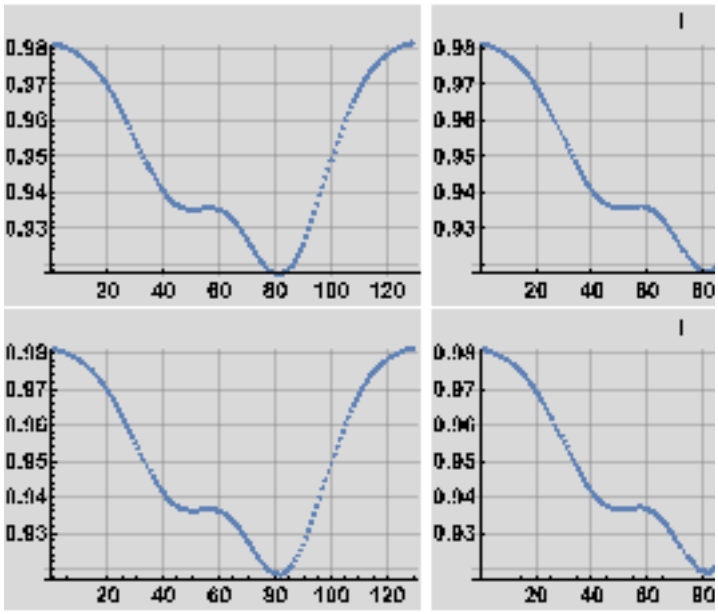


11.4 Detector 4 (40 mrad)

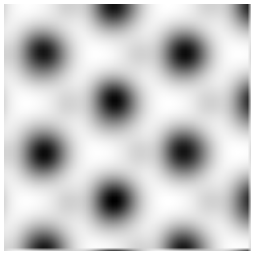
```
(analyzeIIIV[#] & /@ (s35 /. Detector → "BF4")) //
 TableForm[#, TableHeadings → {None, {"III-V", "Bright-field", "Profile"}}] &
```

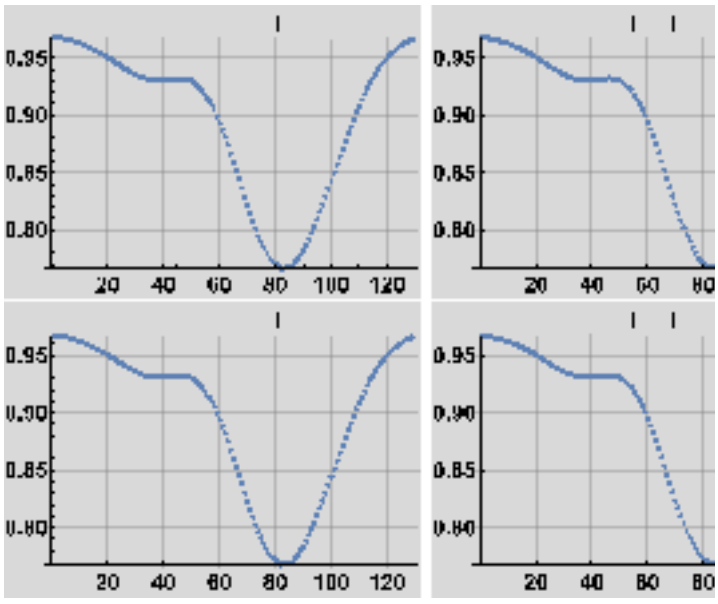
"III-V"	"Bright-field"	"Profile"
"AlAs"		

"AlP"

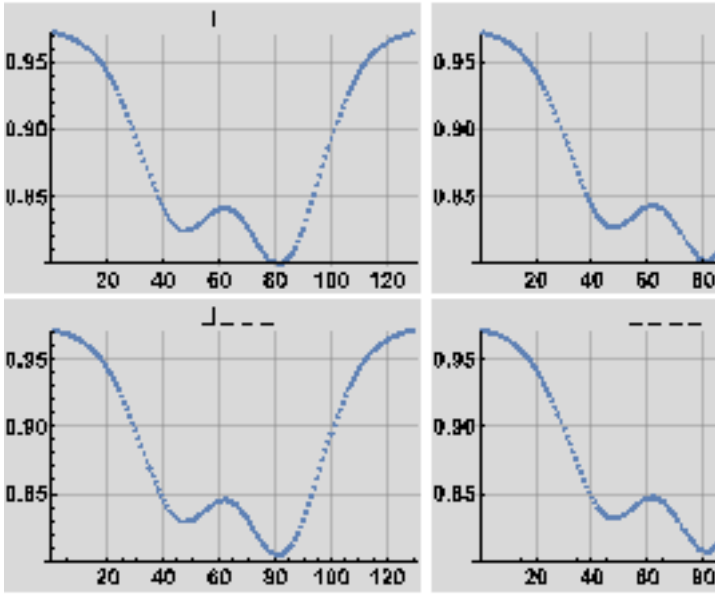


"AlSb"

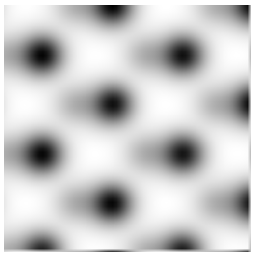


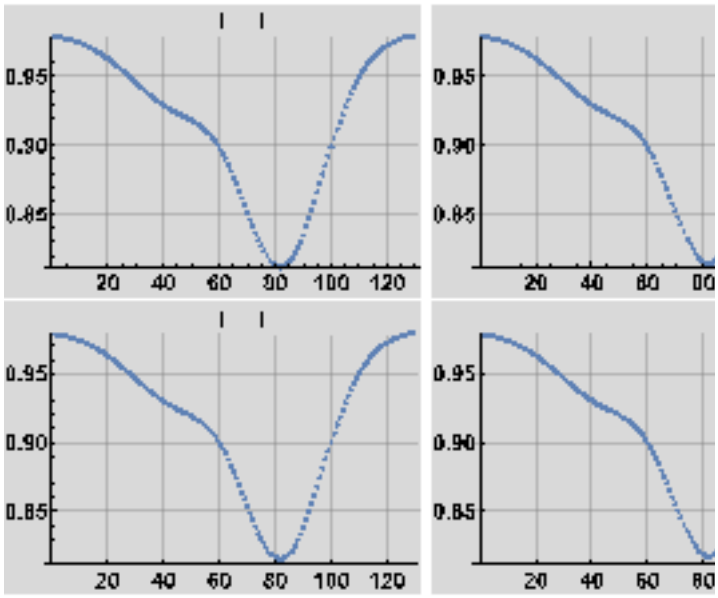


"GaAs"

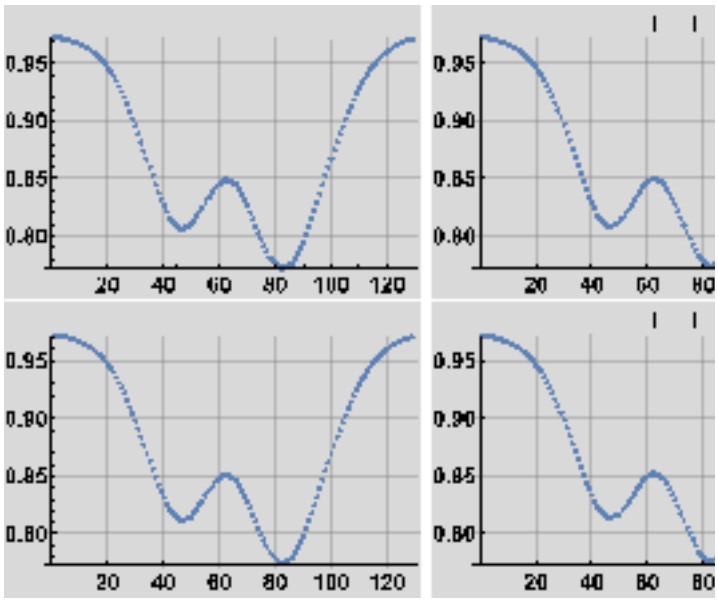


"GaP"

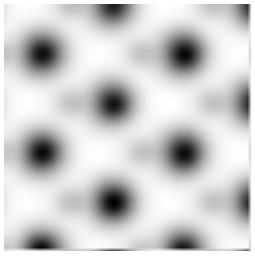


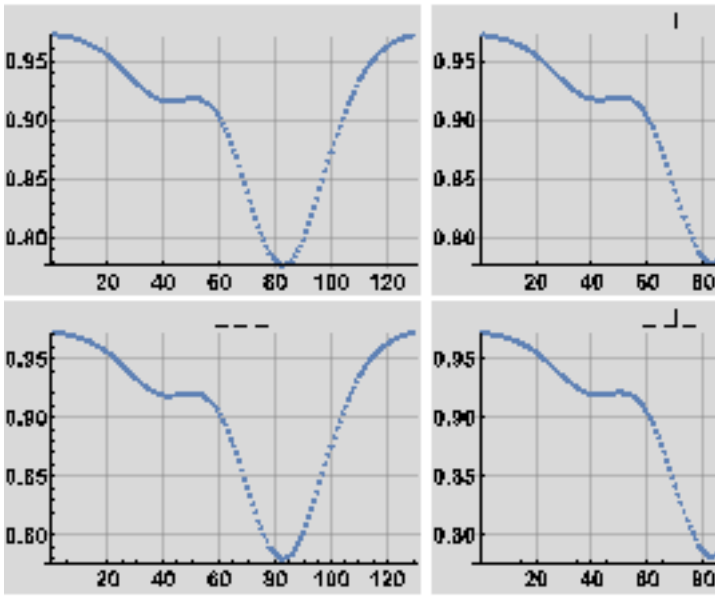


"InAs"

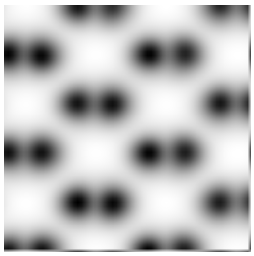


"InP"

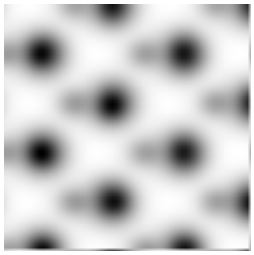


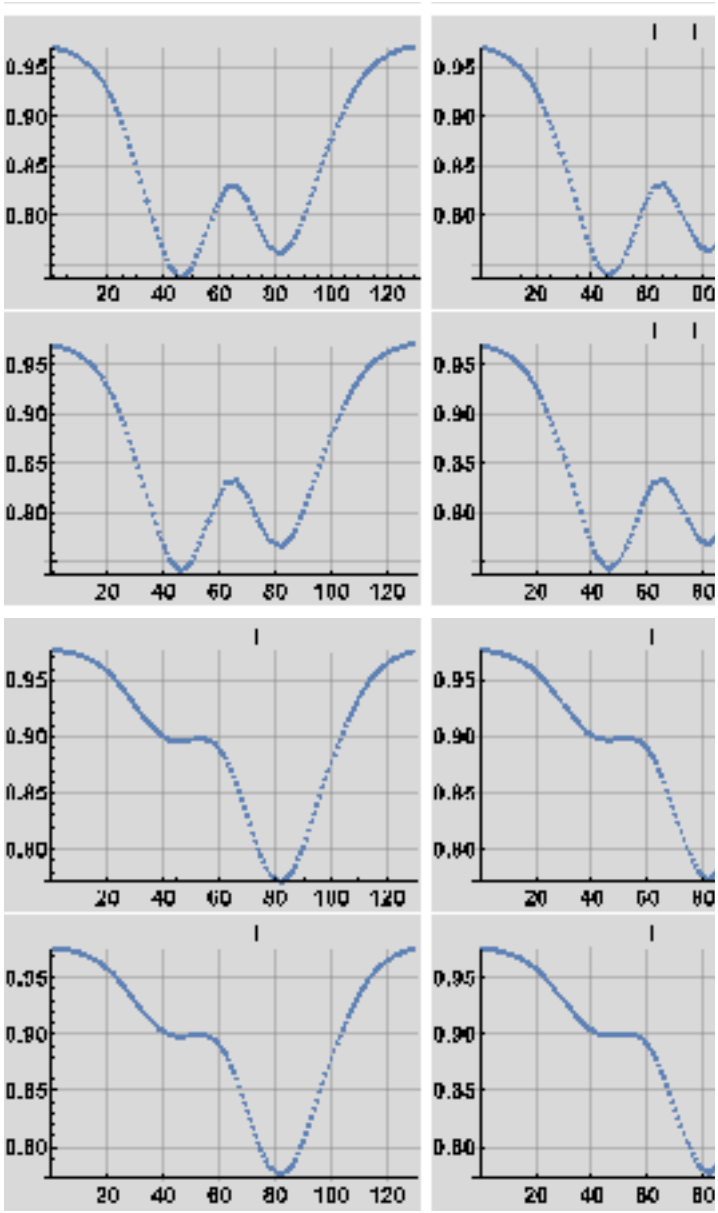


"InSb"



"ZnTe"





12. Conclusions

The simulations of BF images does not require large multislice calculations and 5 to 10 different atomic configurations of the “Frozen Lattice” model seems to suffice for obtaining good III-V dumbells profiles. The simulation of ADF images is more delicate and a few trials might be necessary in order to obtain reliable III-V profiles of large specimen thicknesses. It is also necessary for this kind of simulations to adapt the dark-field detector size to the size of the multislice calculations.