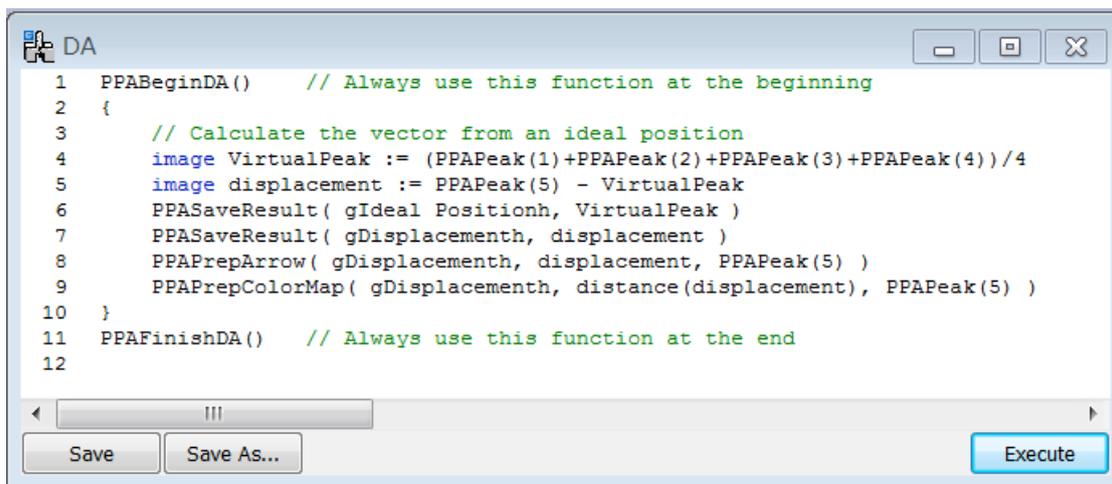


Distortion Analysis (from v4.0)

From information on the peak positions of a complex structure, you can analyse a peak distance and angle (a length and angle of the peak-pair vector). Furthermore, in principle, you can calculate a rotation of a structural unit (e.g. tetrahedron or octahedron), or even a distance from the ideal/virtual position to the observed peak. However, such an advanced analysis requires some skill in programming. The Distortion Analysis of the PPA will provide an easy way to perform such an analysis using the DM script.

For the Distortion Analysis we first perform Motif detection, and then on the original image select the peaks to be used in the analysis using the HREM mouse. Next, write a script to measure distortions in the script window as shown below:



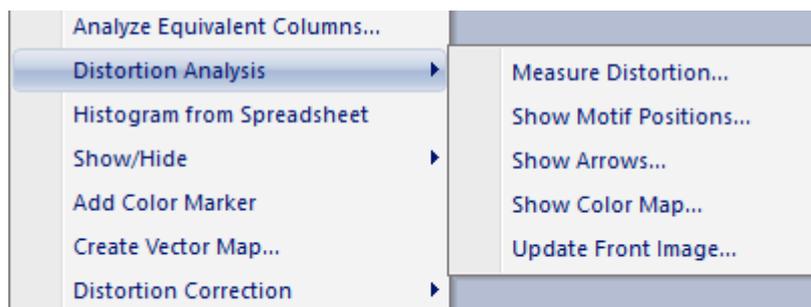
```
1 PPABeginDA() // Always use this function at the beginning
2 {
3 // Calculate the vector from an ideal position
4 image VirtualPeak := (PPAPeak(1)+PPAPeak(2)+PPAPeak(3)+PPAPeak(4))/4
5 image displacement := PPAPeak(5) - VirtualPeak
6 PPASaveResult( gIdeal Positionh, VirtualPeak )
7 PPA displacement ( gDisplacementh, displacement )
8 PPA Prep Arrow( gDisplacementh, displacement, PPAPeak(5) )
9 PPA Prep Color Map( gDisplacementh, distance(displacement), PPAPeak(5) )
10 }
11 PPAFinishDA() // Always use this function at the end
12
```

Then, you can run the script by pushing “Execute” button. Before executing the script, place a rectangle ROI on the original image to show the area to be analysed.

TIPS: If you have no error messages, you may want to save the script for a future use by clicking “Save As...” button. As you see the example script, you have to respectively use PPABeginDA() and PPAFinishDA() at the beginning and the end of the script in addition to your analysing code.

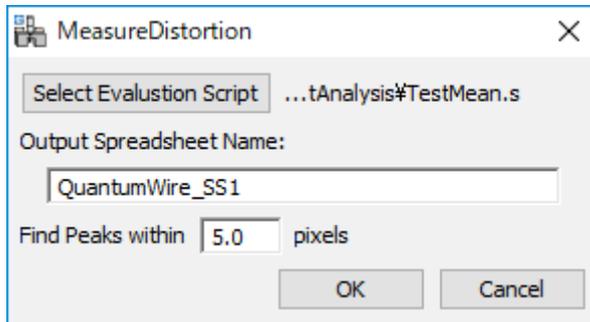
NOTE: This script will measure a distance from the virtual position (the center of the four peaks from peak #1 to peak #4) to the observed peak #5, and save some information for a later use. Here, the functions with the prefix of PPA are provided with the PPA plug-in v4.0 or later and explained at the end of this section.

The following shows the menu for the Distortion Analysis.



The first step of the Distortion Analysis is to measure distortions by launching the command “Measure Distortion...” to run the saved script, or by directly executing the script from the

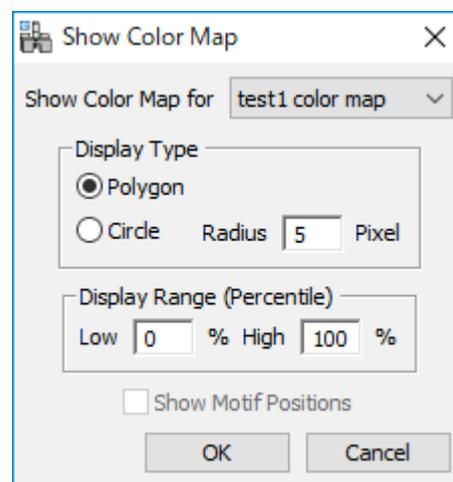
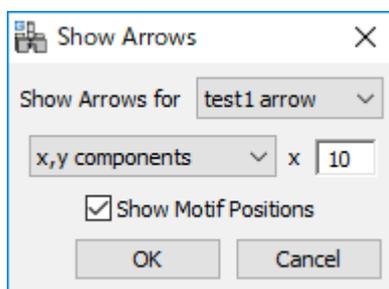
script window as described above. Before launching the command, place a rectangle ROI on the original image to show the area to be analysed. Furthermore, on the original image select the peaks to be used in the analysis using the HREM mouse. This command will open the dialog shown below:



Here, you can select your saved script file to evaluate distortions, and specify the output spreadsheet name. If you directly execute the script from the script window, you will have a dialog without the file selection capability.

Peaks will be searched in the area specified by “Find Peak within” value from the start point that corresponds to a Motif peak. If the peak is not found within the specified region, you will find NaN in the spreadsheet.

The measured distortion can be visualized using the commands “Show Arrows...” or “Show Color map...” when the spreadsheet obtained by the analysing script or the project image is at the front most. When the project image is at the front most, the dialog will be popped up to select the spreadsheet.



The commands “Show Arrows...” and “Show Color Map...” will work for the spreadsheet that includes the information saved by PPAprepArrow and PPAprepColorMap, respectively.

TIPS: You can modify the appearance of the map using “Update Front Image...” command, if the front-most image is an arrow or color map.

You can check the positions of the approximate motif centres using “Show Motif Positions...” command.

Description of the functions*PPABeginDA()*

should be called at the beginning of the script

PPAFinishDA()

should be called at the end of the script

image PPAPeak(number n)

generates a list of column positions for the peak #n

image PPALength(image point1, image point2)

calculates the length of the vector (point1 to point2)

image PPAAngle(image point1, image point2)

calculates the angle of the vector (point1 to point2), anticlockwise from the horizontal axis

image PPAAngle(image point1, image point2, image point3)

calculates the angle between two vectors (point1 to point2, and point1 to point3), anticlockwise measured from point2 to point3.

void PPASaveResult(string "name", image result_data)

saves result_data with an identifier "name" to the spreadsheet.

if result_data is vector, its coordinates (x, y), length and angle (measured anti-clockwise from the horizontal axis) will be saved to the spreadsheet.

void PPAPrepArrow(string "name", image vector_data, image start_point)

saves the vector_data and the start_point to the spreadsheet for an arrow map.

void PPAPrepColorMap(string "name", image scalar_data, image display_point)

saves the scalar_data and the display_point to the spreadsheet for a color map.

Sample code*PPABeginDA()*

// calculate the vector from an ideal position

image VirtualPeak := (PPAPeak(1)+PPAPeak(2)+PPAPeak(3)+PPAPeak(4))/4

// Here, we define the image "VirtualPeak" to which the center of the four peaks saved.

*image displacement := PPAPeak(5) - VirtualPeak**PPASaveResult("Ideal Position", VirtualPeak)**PPASaveResult("Displacement", displacement)**PPAPrepArrow("Displacement", displacement, PPAPeak(5))*

//calculate angle

*image angle := PPAAngle(PPAPeak(5), PPAPeak(2), PPAPeak(1))**PPASaveResult("Angle", angle)**PPAPrepColorMap("Angle", angle, PPAPeak(5))**PPAFinishDA()***Another sample code***PPABeginDA()*

// Here, we define the image "peakn" to which the coordinates of the peak #n is saved.

*image peak1 := PPAPeak(1)**image peak2 := PPAPeak(2)**image peak3 := PPAPeak(3)**image peak4 := PPAPeak(4)**image peak5 := PPAPeak(5)*

// calculate the vector from an ideal position

image VirtualPeak := (peak1+peak2+peak3+peak4)/4

```
// Here, we define the image "VirtualPeak" to which the center of the four peaks saved.
image displacement := peak5 - VirtualPeak
PPASaveResult( "Ideal Position", VirtualPeak )
PPASaveResult( "Displacement", displacement )
PPAPrepArrow( "Displacement", displacement, peak5 )

//calculate angle
image angle := PPAAngle( peak5, peak2, peak1 )
PPASaveResult( "Angle", angle )
PPAPrepColorMap( "Angle", angle, peak5 )
PPAFinishDA()
```