StEM
Stuttgart Center for Electron Microscopy

Lectures in the theory of TEM
III. Advanced Data Analysis in TEM:
How to do what cannot be done using Image processing software

Christoph T. Koch
Stuttgart Center for Electron Microscopy
Max Planck Institute for Intelligent Systems
Heisenbergstr. 3, 70569 Stuttgart, Germany
Email: koch@is.mpg.de

MPI for Metals Research

StEM
Stuttgart Center for Electron Microscopy

Make a Choice !

You want to win the race? ...

... Spend some time considering what equipment you want to use.

2 MPI for Intelligent Systems

Some Options for TEM Data Analysis

- DigitalMicrograph
- Matlab
- IDL
- Mathematica
- MathCad
- Semper
- Python
- ImageJ
- …

Questions to ask (yourself):
- What are my needs ?
- Availability ?
- Price ?
- Flexibility ?
- What am I used to ?
- Who else is using it ?
- How easily can I get help ?
- …

A few Possibilities

DigitalMicrograph
- Installed on most TEMs
- Can acquire CCD images and control the TEM
- Scripting language similar to C++ but specialized for image processing

Matlab
- Available at most universities, or cheap student/teaching licenses
- Contains a lot of general math functions not available within DM
- With some effort you can make it acquire images though DigitalMicrograph (see, e.g. TOM Toolbox for tomography [http://www.biochem.mpg.de/baumeister/tom_e/index.html])

Python
- Platform independent (Unix, Linux, Windows, Max OSX, Java VM, …)
- Large user base
- Slim standalone applications

StEM
Stuttgart Center for Electron Microscopy

Resources

DigitalMicrograph

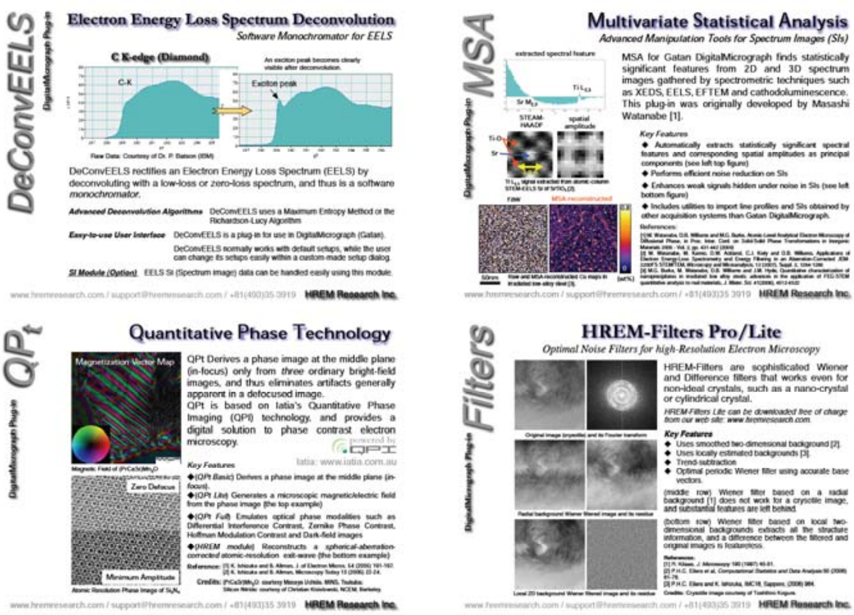
- Introduction to DM scripting: http://www.felmi-zfe.tugraz.at/dm_scripts
- A scripting handbook: http://dm-scripting.tavernmaker.de/
- The scripting database: http://www.felmi-zfe.tugraz.at/dm_scripts/
- Free Plug-Ins: http://www.hremresearch.com/
  (for commercial ones – see next slide
- Dave Mitchell's scripting website: http://www.dmscripting.com/
- Scripting lectures: http://www.christophtkoch.com/Vorlesung/index_SS07.html

Matlab

- Large database of Matlab programs: www.matlabcentral.org
- Matlab routine for reading DM3 files:
  http://www.mathworks.com/matlabcentral/fileexchange/27021-imagic-mrc-and-dm3-file-io

StEM
Stuttgart Center for Electron Microscopy

Coomercial 3rd Party add-ons to DM



What these programs can't do …

- Does not record macros (i.e. it does not automatically generate a script from command that you enter)

  - But: all the menu items can be accessed from scripts

- DM-scripts become slow, when excessive use of loops is being made (interpreted language: code is being compiled "on the fly")

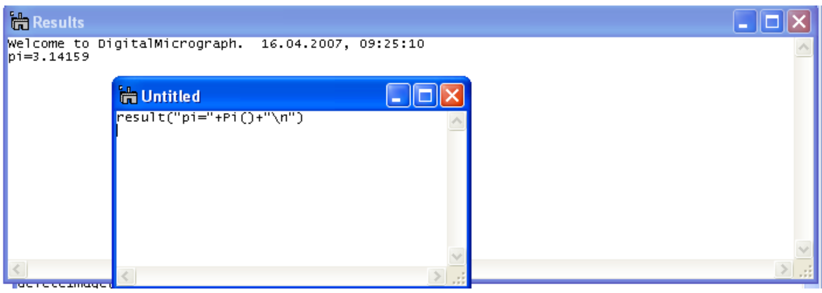
  - But: compiled C/C++-code may be included as PLUG-IN

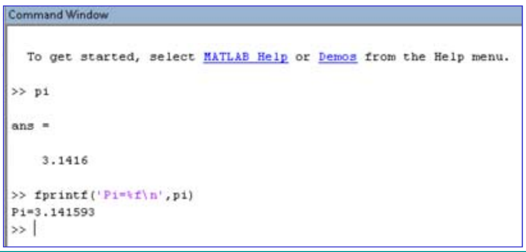
the same is true for Matlab!

MPI for Intelligent Systems

Displaying Text

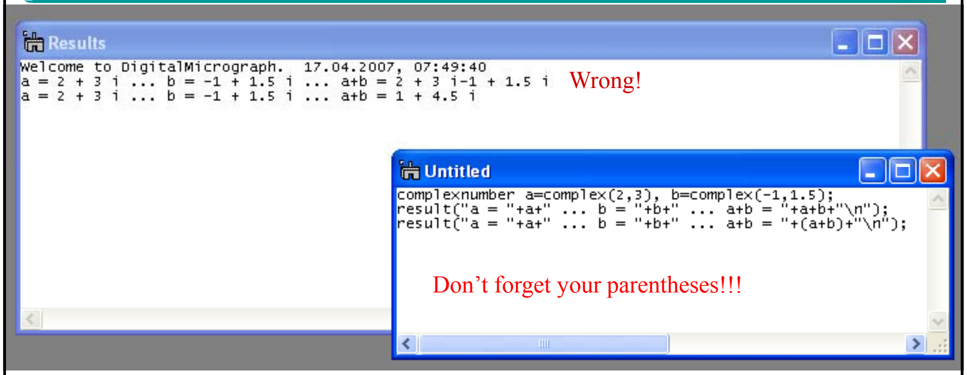
```
result("pi="+Pi()+"\n")
```





MPI for Intelligent Systems

Work with complex numbers



Command Window

```
>> a=2+3*i;
>> b=-1+1.5*i;
>> a+b
ans =
   1.0000 + 4.5000i
>>
```

MPI for Intelligent Systems

Built-in operations on complex numbers

- abs
- cis
- complex
- conjugate
- cos
- cosh
- exp
- imaginary
- log
- modulus
- norm
- Phase
- Polar
- real
- Rect
- sin
- sin
- sqrt
- tan
- tanh

There are similar commands in Matlab (and many more)!

MPI for Intelligent Systems

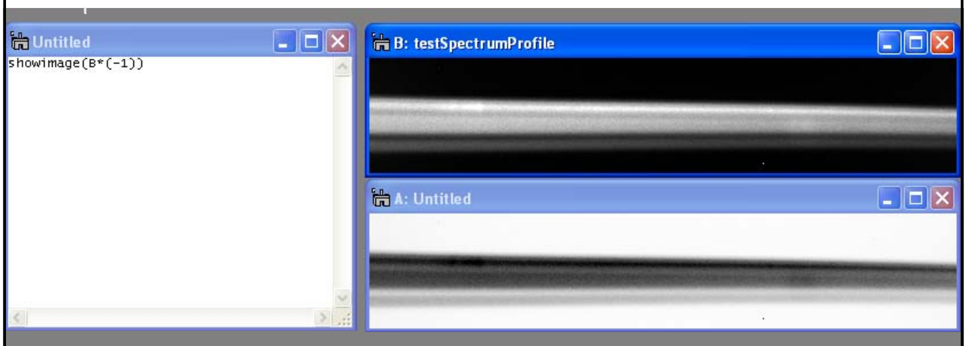
A selection of real-number functions

- sin, asin, sinh
- cos, acos, cosh
- tan, tanh, atan, atan2, atanh
- exp, exp2, exp10
- log, log2, log10
- exp1(x) = exp(x)-1
- log1(x) = log(x+1)
- AiryAi, AiryBi
- BesselI, (also J, K, Y)
- SphericalBesselJ (also Y)
- Beta
- erf, erfc
- Factorial
- Gamma, GammaP, GammaQ
- LegendrePolynomial
- PoissonRandom (also Binomial, Gaussian, Gamma, Uniform)
- BinomialCoefficient
- …

There are similar commands in Matlab (and many more)!

On the fly data manipulation

Makes sense if script is only used once !

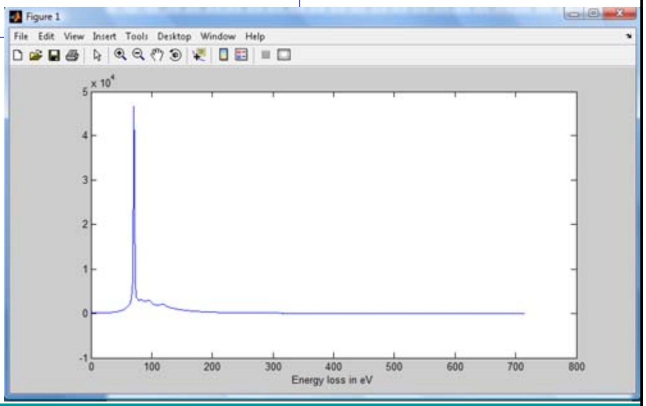


- Images may be addressed in a script by the letter assigned to the window they are shown in.
- If a new image is generated (e.g. by performing some operation on the image), this image will receive a new letter (image variable)

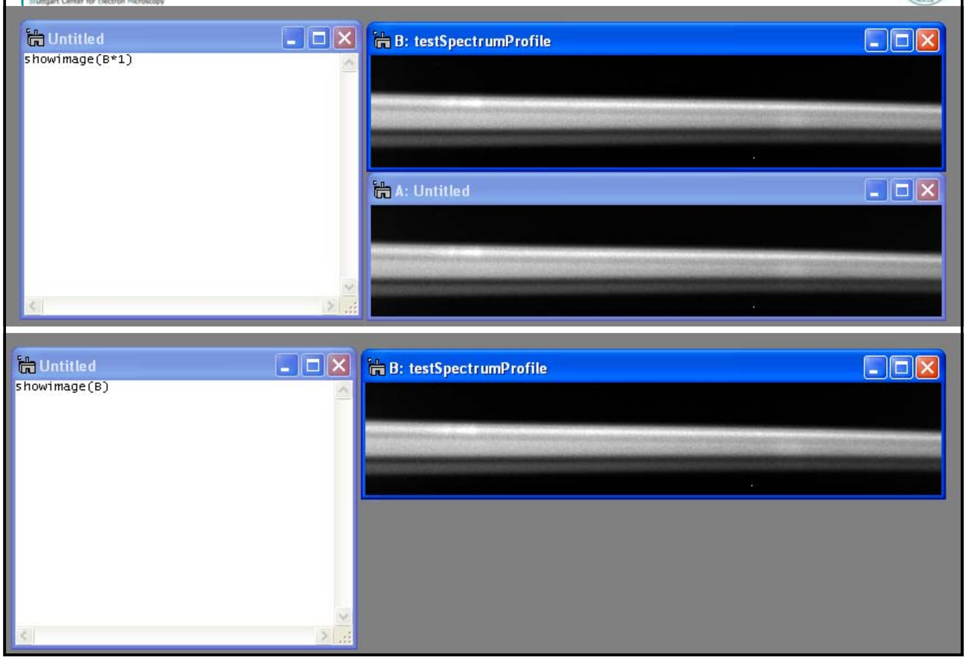
Matlab: Reading and Displaying DM3 Data

Command Window

```
>> [mSize,dx,dy] = ReadDM3_size('Spectra\\01 ZL.dm3')
mSize =
        1022           1           1
dx =
    0.7000
dy =
     1
>> spec=ReadDM3_slice('Spectra\\01 ZL.dm3',mSize,1).';
>> plot(dx*[1:length(spec)],spec)
>> xlabel('Energy loss in eV')
>>
```

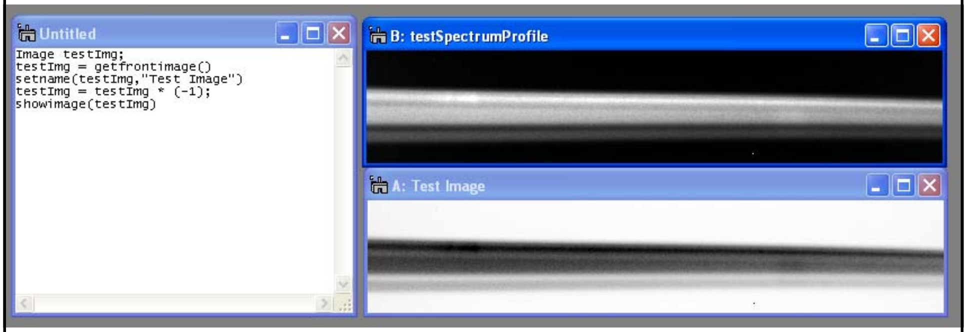


Automatic generation of new images



Make script independent of image name

This makes scripts more general !



Img = GetFrontImage(): obtain the image which is currently active.

Note: if no name is assigned to an image variable, the name of the image remains "Untitled". It helps to assign names to image windows using SetName

DM Built-in commands

Script Reference 2.5

See list of script functions in alphabetical order.

Images

Image Processing
Image Data Types
Real Images
Complex Images
RGB Images
Image Management
Image Display
Image Scrap

Numbers and Strings

Real Number Functions
Complex Number Functions
RGB Number Functions
Number Conversion
Strings

Annotations, Selections, Tags, I/O

Annotations
Selections
Tags (aka Notes)
Dialogs
Input/Output

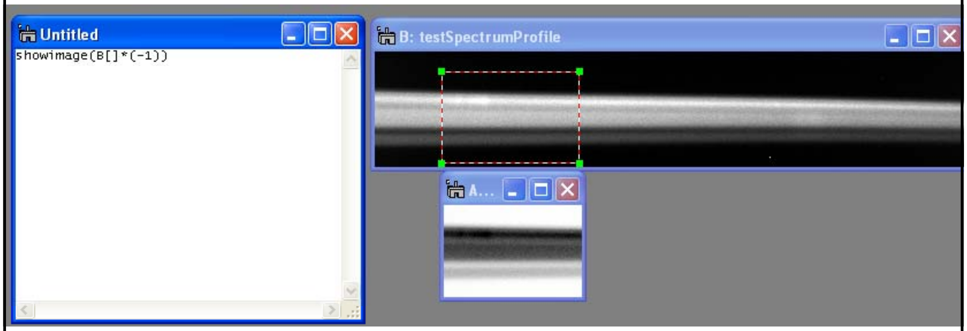
Other

Miscellaneous

This list of commands used to be available online but has been discontinued by Gatan.

A copy of this is available at http://www.felmi-zfe.tugraz.at/dm_scripts

Addressing regions of interest (ROI)



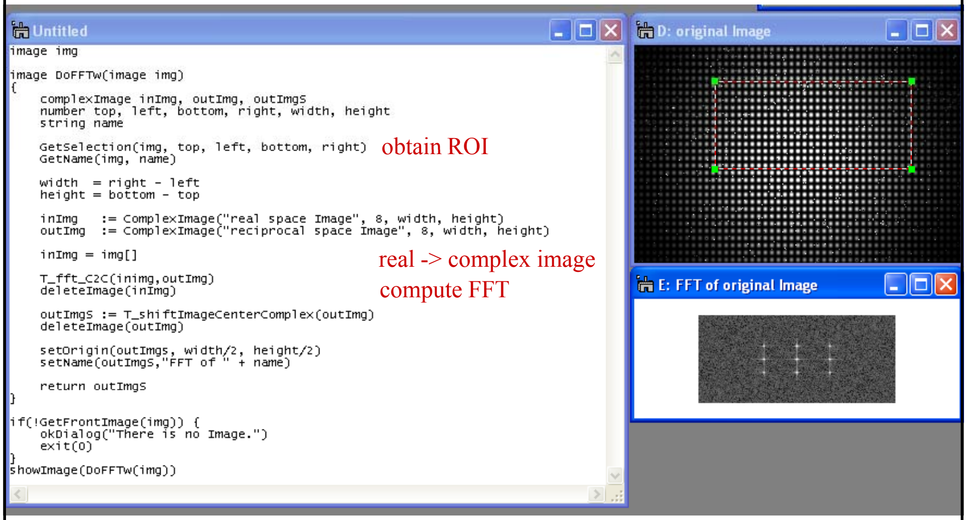
On the fly: simply use square brackets behind the image letter

FFTW Fastest Fourier Transform in the West

- FFTW is a software library that also allows every 3rd, 4th, fifth, etc. element to be used. It can therefore handle any size of array.
- Implemented for DM in plugin Transforms.dll (http://hrem.mpi-stuttgart.mpg.de/koch/DM-Plugin/index.html)
- Implemented functions (do not allocate memory for new image):
  - T_fft_c2c(compl_ImgIn,compl_ImgOut)
  - T_ifft_c2c(compl_ImgIn,compl_ImgOut)
  - compl_Img =T_shiftImageCenterComplex(compl_Img) (shifts k=(0,0) of FFT to center of image for complex images – DM function does not work)
  - Additional functions for computing different correlations

Matlab comes with FFTW, but DM can only do radix 2 FFTs

Usage of FFTW within DM



Such scripts (written by Bernd Kraus) are available for FFT and IFFT, for both real- and complex input.

Difference between '==', '=', and ':='

A==B comparison (1, if left=right, 0 otherwise)

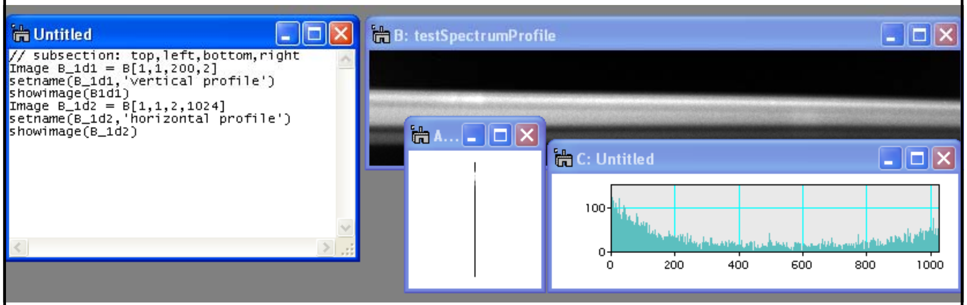
A = B copies variable content

A:= B assigns variable A to the image given by B (should always be used when creating images, although it also works with '=')

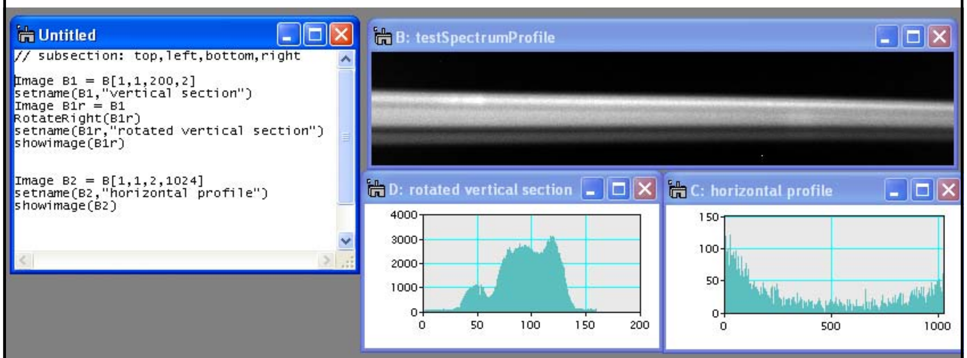
(see more examples in B. Schaffer's tutorial, slides 33 & 34)

In Matlab there is no ':='

Spectra: 1-dimensional images



Vertical 1D-image -> spectrum



Allowed values for selection of sub-region:			
top: 0	…(bottom-1)	bottom: (top+1)	… height
left: 0	… (right-1)	right: (left+1)	… width

In Matlab, indices start at 1 (Fortran), in DM they start at 0 (C/C++)

icol, irow, iradius, …

• There are several *intrinsic* variables which can be used in calculations of images. Their value depends on the position within the image.

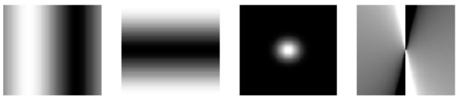
(e.g.: `icol` becomes 5 for all points in an image, which have x=5 as coordinate. It becomes 6 for x=6 and so on..)

Name	Description
icol	column of the image
iheight	height of the image
ipoints	number of points in the image
iradius	distance from the center of the image
irow	row of the image
itheta	angle with respect to the center of the image
iwidth	width of the image
iplane	plane of the image (3D images)

• The following script creates some examples:

(The function `Pi()` returns the value of Pi.)

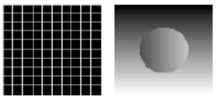
```
image  TestImage
TestImage := RealImage("Test",4,100,100)
ShowImage(TestImage)

TestImage = sin(2*Pi()/iwidth*icol)
TestImage = cos(2*Pi()/iheight*irow)
TestImage = exp(-iradius**2/(iheight/10)**2)
TestImage = tan(itheta)
```



• Often, the intrinsic variables are used in the `tert()` command:

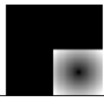
(The function `mod(a,b)` returns the modulo, e.g. `mod(14,3)=2` as 14 = 4*3 + 2)

```
TestImage = tert( mod(icol,10)==0 || mod(irow,10)==0,1,0)
TestImage = tert( iradius<iwidth/4 , icol , irow)
```



• Note that the variables check the actual image expression, not the image itself. If an area of an image is used, the top-left pixel of this area is (0/0):

```
TestImage = 0
TestImage[50,50,100,100] = iradius  // the center is now at 75/75!
```



Not such built-in variables in Matlab

Slide: Bernhard Schaffer, TU Graz

Applications of Convolution in TEM

- Smoothing of data
- Differentiating of data (e.g. Laplacian, …)
- Simulate the effect of microscope instabilities (e.g. sample vibrations for images, energy fluctuations in spectra)
- Simulate the effect of detector point spread functions (PSF)
- Simulate the effect of microscope aberrations in HAADF-STEM (based on an oversimplifying approximation)

MPI for Intelligent Systems

Applications of Deonvolution in TEM

- Inversion of gradient and Laplacian operations
- Removal of microscope instabilities (e.g. sample vibrations in images, energy instabilities in spectra)
- Removal of microscope aberrations in HAADF-STEM (assuming that the image is the convolution of probe and object function)
- Removal of plural scattering in EELS
- Removal of source energy spread in EELS

Definition of Convolution

1D $$f \otimes g = \int_{-\infty}^{\infty} f(r') \cdot g(r-r')dr'$$

2D $$f \otimes g = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x',y') \cdot g(x-x',y-y')dx'dy'$$

In general, the value of F(**r**)=f(**r**)⊗g(**r**) depends on the values of f(**r**) and g(**r**) for all **r**, i.e. across the whole image

Convolution Theorem: $$f \otimes g = FT^{-1}\left[FT(f)FT(g)\right]$$

Computing the gradient by convolution

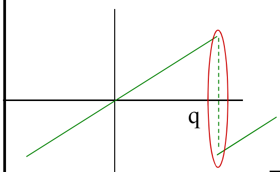
Real space:

$$\frac{df}{dx}=\frac{f_{i+1}-f_i}{\Delta x}=f\otimes D_x^1 \qquad D_x^1=\begin{pmatrix}1 & -1\end{pmatrix}$$

or

$$\frac{df}{dx}=\frac{f_{i+1}-f_{i-1}}{2\cdot\Delta x}=f\otimes D_x^1 \qquad D_x^1=\begin{pmatrix}1 & 0 & -1\end{pmatrix}$$

Reciprocal space:



Discontinuous edges!

$$\frac{df}{dx}=\frac{d}{dx}\sum F_q\cdot e^{2\pi iqx}$$

$$=\sum F_q\cdot 2\pi iq\cdot e^{2\pi iqx}$$

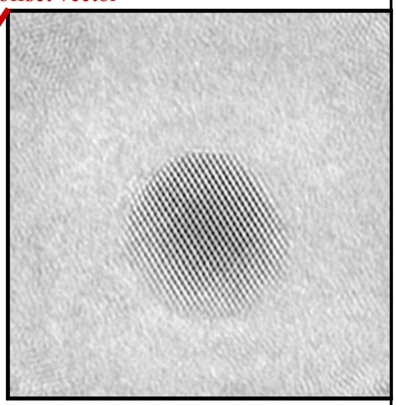
$$\Rightarrow\frac{df}{dx}=FT^{-1}\{2\pi i\cdot FT[f]\cdot q\}$$

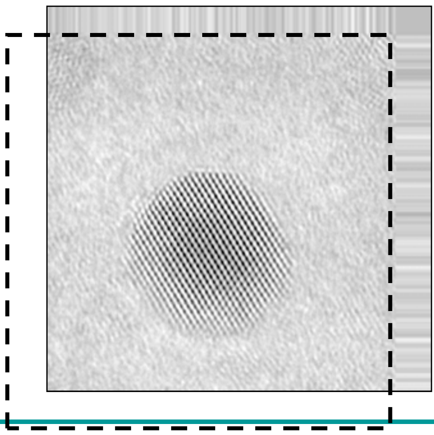
[F_q = Fourier components of f(x)]

MPI for Intelligent Systems

The DM function ‘offset(img,dx,dy)’

offset vector





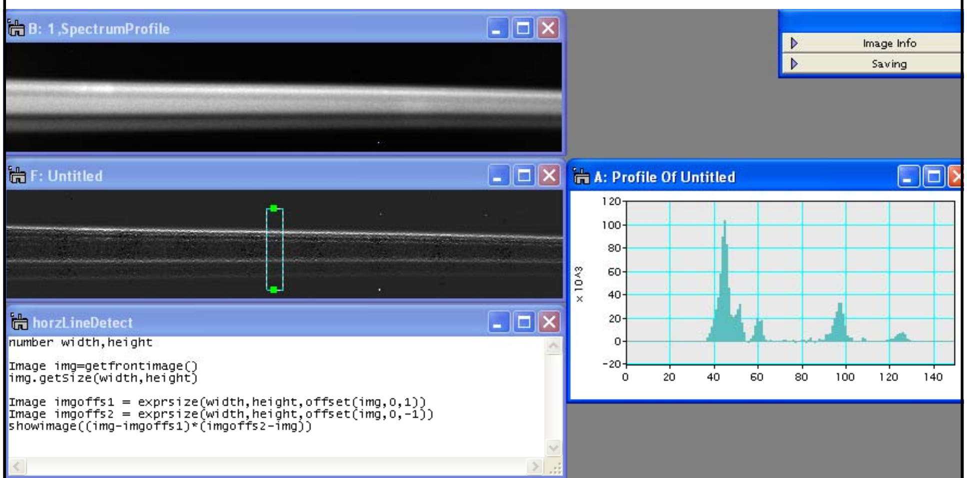
One must provide the command ‘offset’ with the size of the target image bei either:

- create the target image beforehand, or
- use ‘exprsize’

MPI for Intelligent Systems

Expand images using 'offset'

One dimensional derivative: Edge detection



Computing the Laplacian by convolution

Real space:

$$\Delta f(x,y) = \frac{\partial^2 f(x,y)}{\partial x^2} + \frac{\partial^2 f(x,y)}{\partial y^2}$$

$$\Delta f(x,y) = f \otimes D^2_{xy} \qquad \mathbf{D}^2_{xy} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Reciprocal space:

$$\Delta f = \left(\frac{d^2}{dx^2} + \frac{d^2}{dy^2}\right) \sum F_{q_x,q_y} \cdot e^{2\pi i (q_x x + q_y y)}$$

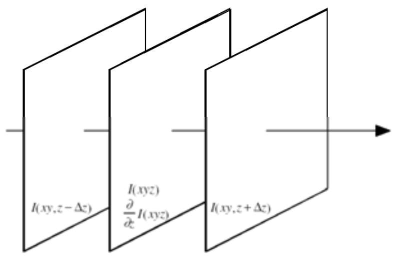
$$= -4\pi^2 \sum \left(q_x^2 + q_y^2\right) \cdot F_{q_x,q_y} \cdot e^{2\pi i (q_x x + q_y y)}$$

$$\Rightarrow \Delta f = FT^{-1}\left\{-4\pi^2 \cdot FT[f] \cdot \left(q_x^2 + q_y^2\right)\right\}$$

Transport of Intensity Equation (TIE)

The wave function satisfies the Schrödinger eqn in free space (Fresnel propagation):

$$\left(2ik\frac{\partial}{\partial z} + \nabla^2_{xy} + 2k^2\right)\psi(xyz) = 0$$



The phase of the electron wave is then given by:

$$\phi(xyz) = -\frac{2\pi}{\lambda} \nabla^{-2}_{xy} \nabla_{xy} \bullet \left(\frac{1}{I(xyz)} \nabla_{xy} \nabla^{-2}_{xy} \frac{\partial}{\partial z} I(xyz)\right)$$

Inverse Laplace operator

where

$$\psi(xyz) \equiv \sqrt{I(xyz)}\, \exp\{i\phi(xyz)\} \exp\{i\mathbf{kr}\}$$

The Modulation Transfer Function (MTF)

A sharp image produced by the electron wave on the detector will be smeared by "cross-talk" between the pixels of the detector.

If an electron hits the scintillator (or phosphor screen) above a certain CCD pixel, then the neighboring pixels may also receive a few photons (this is also true for film and imaging plates).

The resulting image is the convolution of the original image with the MTF:

$$I_{\exp}(\vec{r}) = I_{ideal}(\vec{r}) \otimes FT^{-1}\left[MTF(\vec{q})\right]$$
$$= FT^{-1}\left\{FT\left[I_{ideal}(\vec{r})\right] \cdot MTF(\vec{q})\right\}$$

MPI for Intelligent Systems

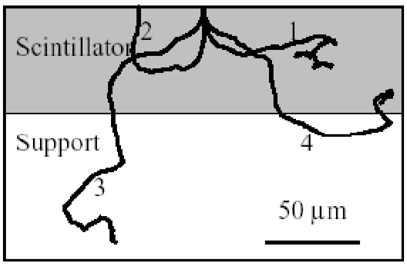
Detector Point Spread Function

1.) Stopped in the scintillator.

2.) Back-scattered from the scintillator.

3.) Stopped in the support.

4.) Back-scattered from the support.



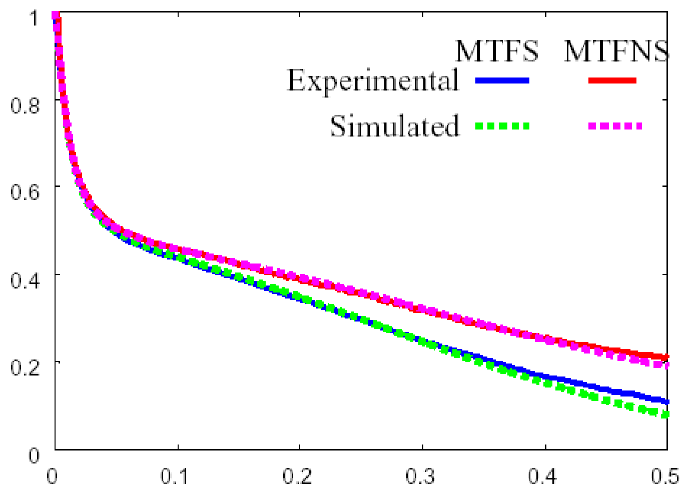
Voltage	100 kV	200 kV	300 kV	400 kV
Type 1	82 %	16 %	0.6 %	0 %
Type 2	18 %	20 %	10 %	5 %
Type 3	0 %	48 %	66 %	70 %
Type 4	0 %	16 %	22 %	25 %

Slide: A. Kirkland

MPI for Intelligent Systems

StEM

MTFS of a YAG scintillator at 100kV

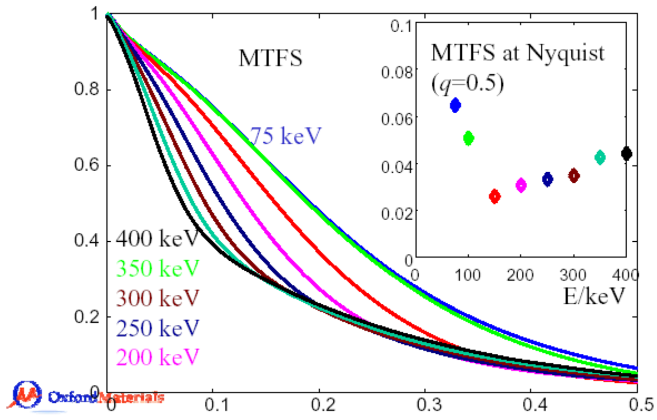


Slide: A. Kirkland

MPI for Intelligent Systems

StEM

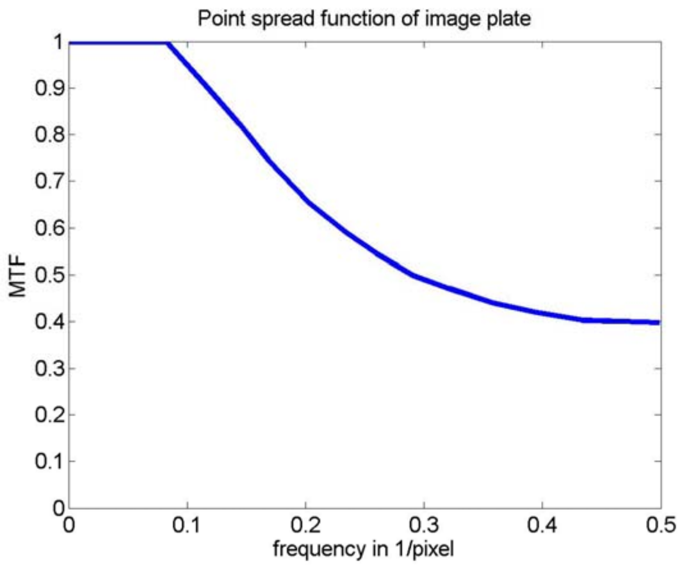
MTFS of a phosphor powder scintillator



Slide: A. Kirkland

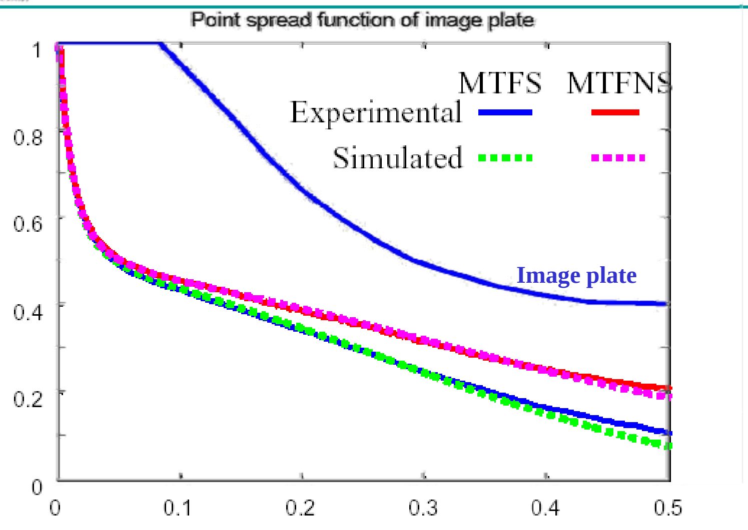
MPI for Intelligent Systems

MTF of Image Plate



MPI for Intelligent Systems

Comparison Image Plate - CCD



MPI for Intelligent Systems

Convolution -> Deconvolution

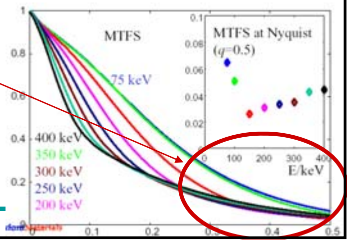
Convolution of an image with the detector MTF (also called point spread function [PSF]):

$$I_{\exp}(\vec{r}) = I_{ideal}(\vec{r}) \otimes FT^{-1}\left[MTF(\vec{q})\right]$$
$$= FT^{-1}\left\{FT\left[I_{ideal}(\vec{r})\right] \cdot MTF(\vec{q})\right\}$$

De-Convolution of an image with the detector MTF:

$$I_{ideal}(\vec{r}) = FT^{-1}\left\{FT\left[I_{\exp}(\vec{r})\right] / MTF(\vec{q})\right\}$$

Problem: At high frequencies the MTF(q) is very small (division by small numbers!) and $I_{exp}(r)$ may be dominated by noise.
=> Noise Amplification!



Avoiding Noise Amplification

Possible solutions to avoid noise amplification are:

1. Impose an upper limit on 1/MTF(q)
2. Lower the upper limit on 1/MTF(q) with increasing q (S/N ratio usually decreases)
3. Let 1/MTF(q) go to zero above a certain resolution q (ideally q should match the resolution present in the image data)
4. Make sure that the deconvolution kernel (e.g. MTF(q)) is smooth. Otherwise this makes errors even worse.
5. Use Richardson-Lucy deconvolution
6. Use Maximum Likelihood deconvolution

MPI for Intelligent Systems

1. Upper limit on Inverse of Convolution Kernel

Use the DM command ‘tert’:

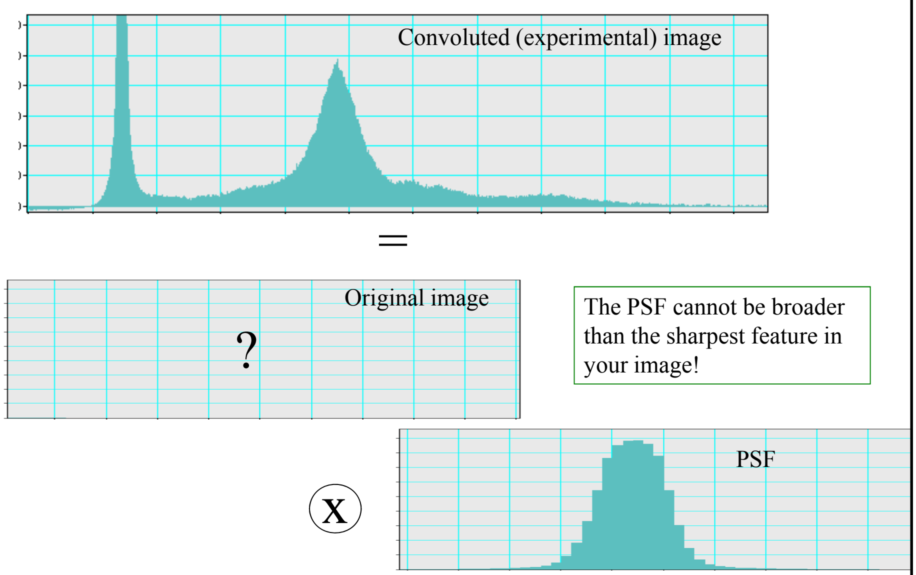
```
Image img = getFrontImage()
Image iMTF_lim = tert(1/MTF>thresh, thresh, 1/MTF)
```

condition (1/MTF>thresh) — condition true (thresh) — condition false (1/MTF)

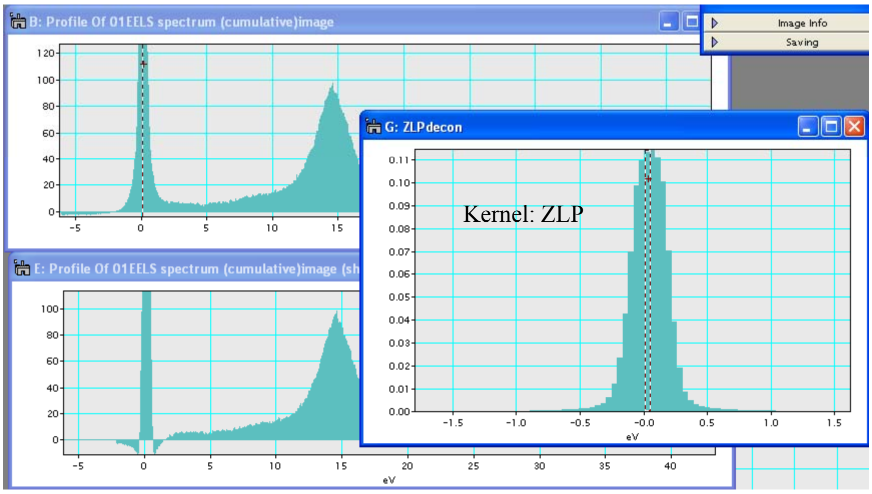
```
Image deconv = realifft(realfft(img)*iMTF_lim)
```

for options 2 & 3: replace thresh with a Gaussian image (see script “GaussEdgeSmoothingInterp.s”)

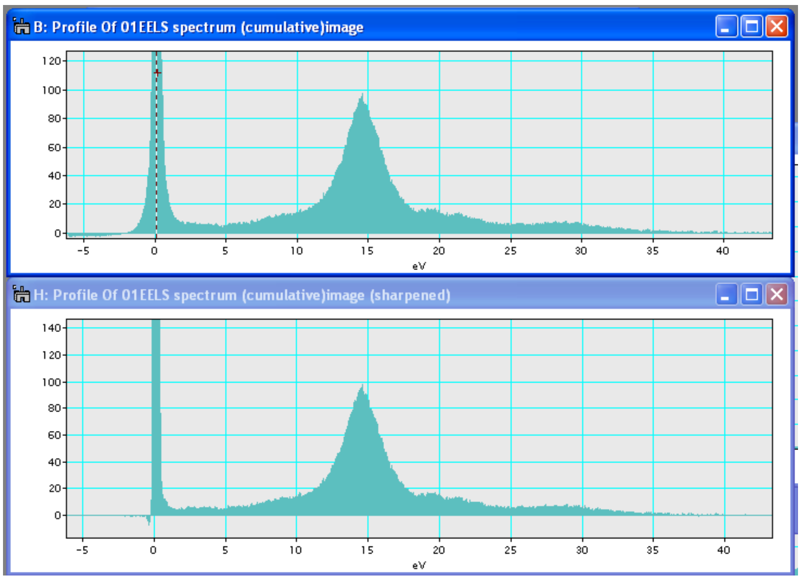
Convolution Kernel must be sensible



DM-EELS Function: “Sharpen Spectrum”



‘Sharpen Spectrum’ with ZLP from spectrum



StEM

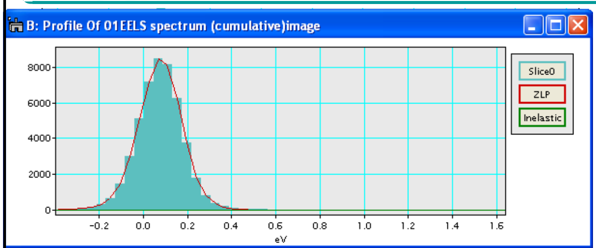
How does ‘Sharpen Spectrum’ work?

1. Fit a Gaussian to the narrow central portion (within 90% of the maximum, mimimum of 3 channels) of the zero-loss peak [ZLP] (=Convolution kernel)
2. Replace that portion of the ZLP with a δ-function of equal area.
3. Subtract the fitted Gaussian from the original data and set negative pixels of the resulting “Gaussian-subtracted ZLP” to zero.
4. Place the total Intensity difference between the ZLP and the “Gaussian-subtracted ZLP” in a single pixel at the position of the ZLP maximum (=> delta-function).
5. Normalize this modified ZLP to 1 and makesure the delta-function is in pixel(0,0).

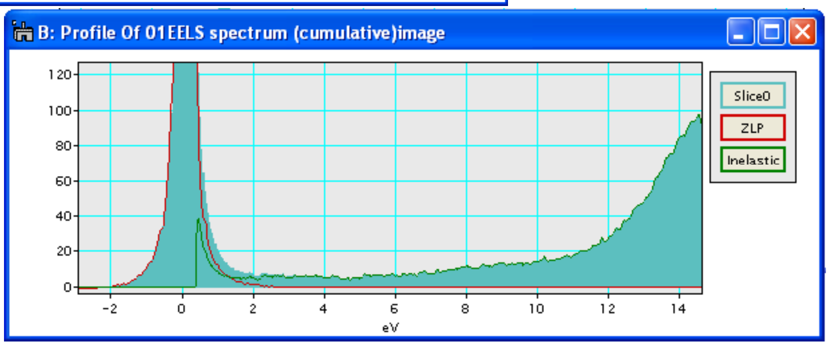
MPI for Intelligent Systems

StEM

How does the ZLP-extraction work



1. A symmetric ZLP is assumed. The right hand side of the ZLP below ¼ of its height is replaced by its left hand side.
2. Negative counts in the “inelastic spectrum” (original – ZLP) are set to zero.



MPI for Intelligent Systems

EELS Multiple Scattering Deconvolution

Assuming independent scattering events the intensity in an experimental EELS spectrum can be simulated by the expression

$$I_{\exp}(E) = ZLP(E) \otimes \left[\frac{t}{\lambda} I_{theor}(E) + \frac{1}{2!}\left(\frac{t}{\lambda} I_{theor}(E)\right) \otimes \left(\frac{t}{\lambda} I_{theor}(E)\right) + \dots \right]$$

$$= ZLP(E) \otimes FT^{-1}\left[\exp\left\{ \frac{t}{\lambda} FT\left[I_{theor}(E)\right] \right\} \right]$$

$$= FT^{-1}\left\{ FT\left[ZLP(E)\right] \cdot \exp\left(\frac{t}{\lambda} FT\left[I_{theor}(E)\right] \right) \right\}$$

This means, in order to extract the true spectrum $I_{theor}(E)$ from an experimental spectrum one must first deconvolute by the ZLP as precisely as possible.

MPI for Intelligent Systems

EELS Multiple Scattering Deconvolution (2)

Inverting the previous expression, one can extract the single scattered spectrum from the experimental data according to

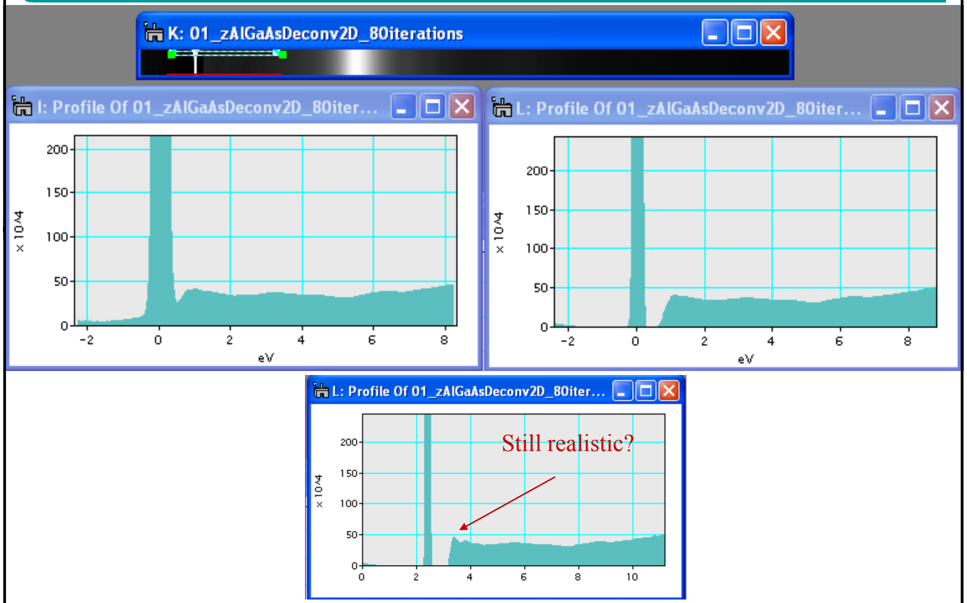
$$\frac{t}{\lambda} I_{ss}(E) = FT^{-1}\left\{ \ln\left(\frac{FT\left[I_{theor}(E)\right]}{FT\left[ZLP(E)\right]} \right) \right\}$$

How does an incomplete deconvolution of the ZLP (or deconvolution by a smoothed ZLP) affect $I_{ss}(E)$?

It mainly affects the resolution of $I_{ss}(E)$. Peak positions and –heights will hardly be affected.

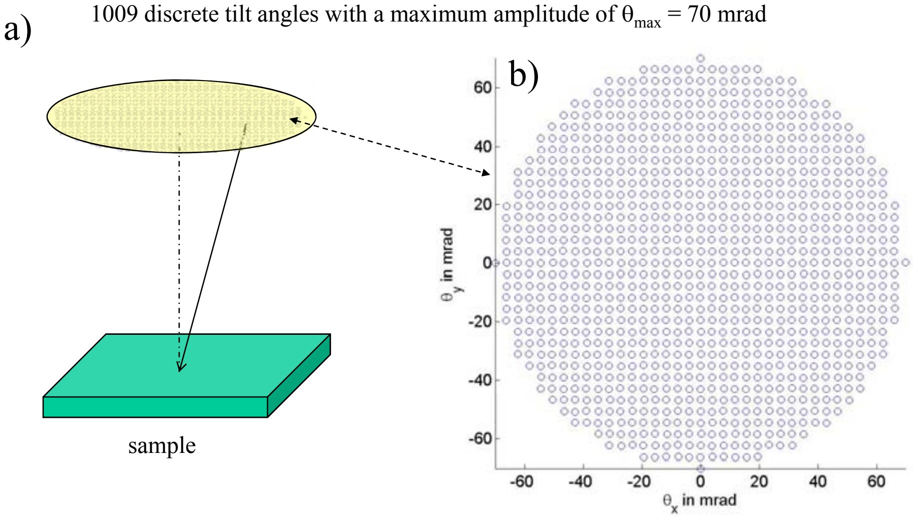
MPI for Intelligent Systems

Result of Richardson-Lucy Deconvolution



LARBED Acquisition Using Discrete Tilts

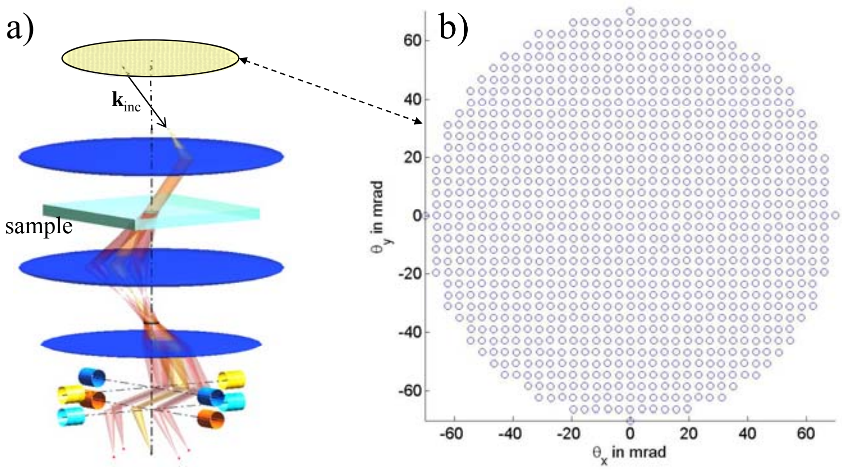
1009 discrete tilt angles with a maximum amplitude of θ_{max} = 70 mrad



Aberration-compensated large-angle rocking-beam electron diffraction
C.T. Koch, Ultramicroscopy (2011) in press, doi:10.1016/j.ultramic.2010.12.014

LARBED Acquisition Using Discrete Tilts

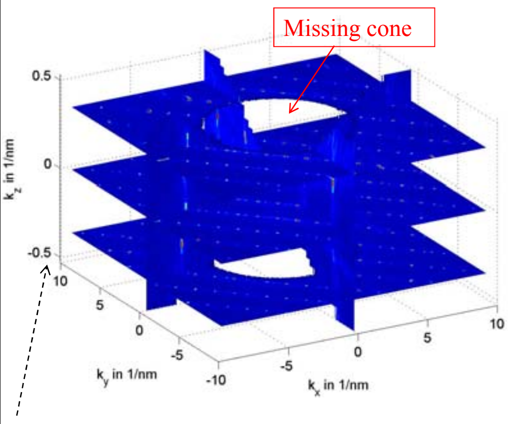
1009 discrete tilt angles with a maximum amplitude of θ_{max} = 70 mrad



Aberration-compensated large-angle rocking-beam electron diffraction
C.T. Koch, Ultramicroscopy (2011) in press, doi:10.1016/j.ultramic.2010.12.014

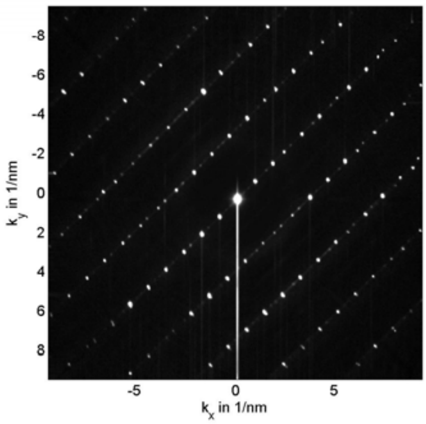
Reconstruction of 3D k-Space

Each voxel in this 3D k-space is the intensity averaged along the corresponding Bragg line

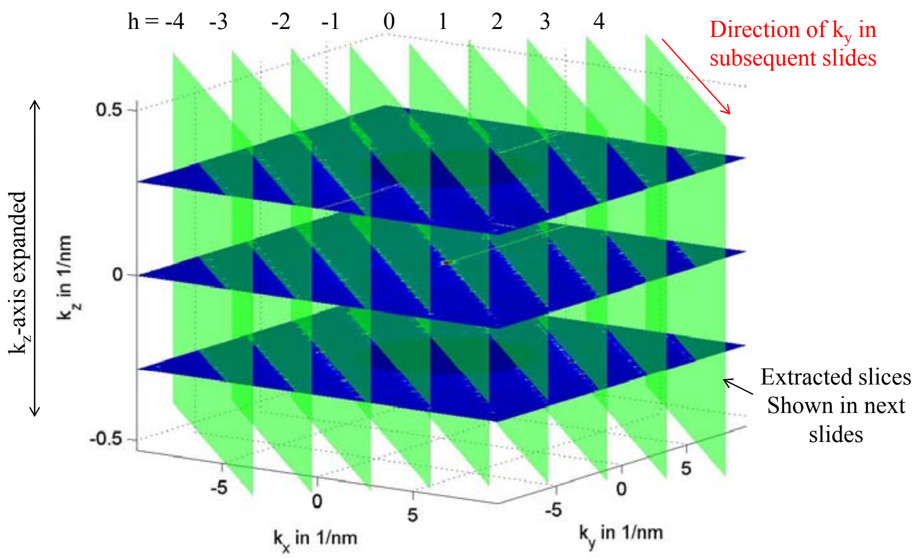


k_z-axis expanded

Central slice of reconstructed volume

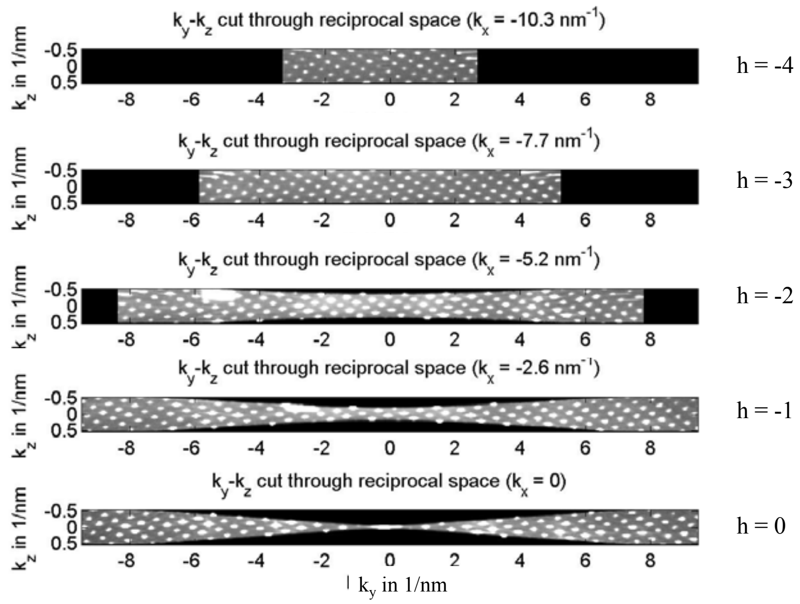


The Reconstructed k-Space Volume



53 MPI for Intelligent Systems

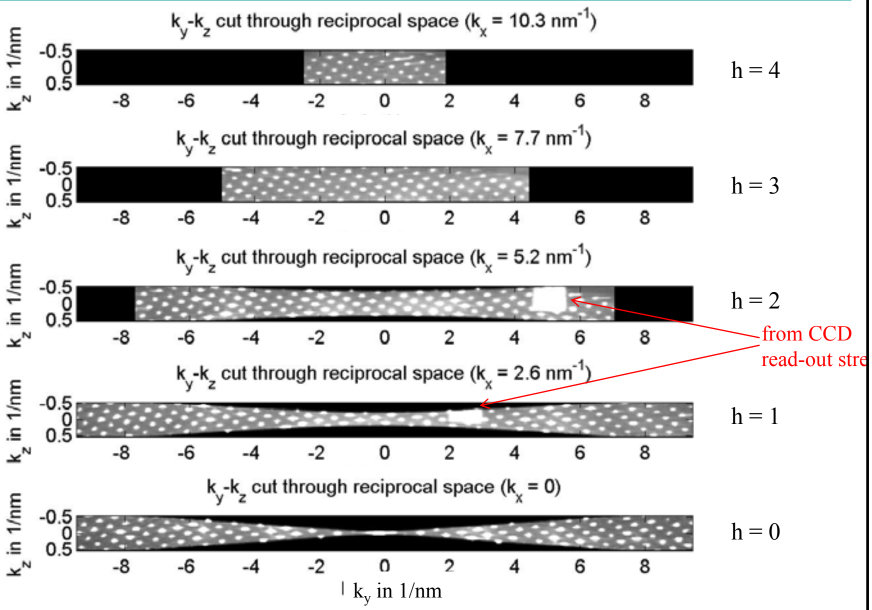
Vertical Slices through k-Space ($k_x \leq 0$)



54 MPI for Intelligent Systems

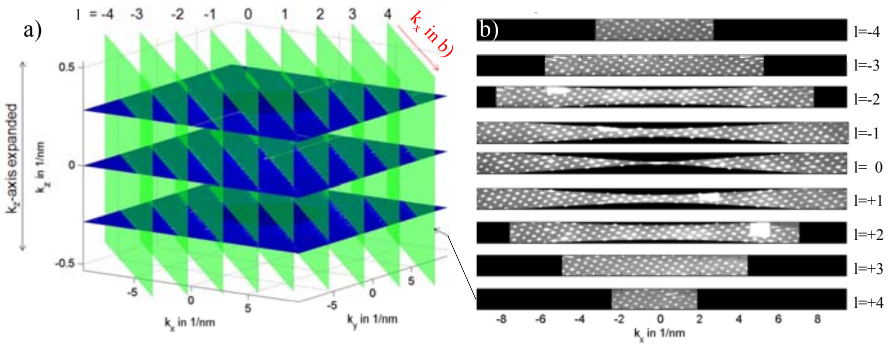
StEM Stuttgart Center for Electron Microscopy

Vertical Slices through k-Space ($k_x \geq 0$)



55 MPI for Intelligent Systems

StEM Stuttgart Center for Electron Microscopy



56 MPI for Intelligent Systems