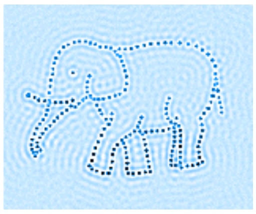
EMBO Practical Course on Image Processing for Cryo EM
2 – 12 September 2019



Practical 1: Introduction to basics of image processing with EMAN2; Scipion pipeline for image preprocessing

Log into a PC with the username and password provided and then on to Nomachine and start a terminal window according to the instruction sheet for each user. If you are not sure what directory you are in you can always type "pwd" and this will display the directory you are currently in.

Part 1: Image processing basics with EMAN2

Move into your prac-1 directory (cd prac-1).

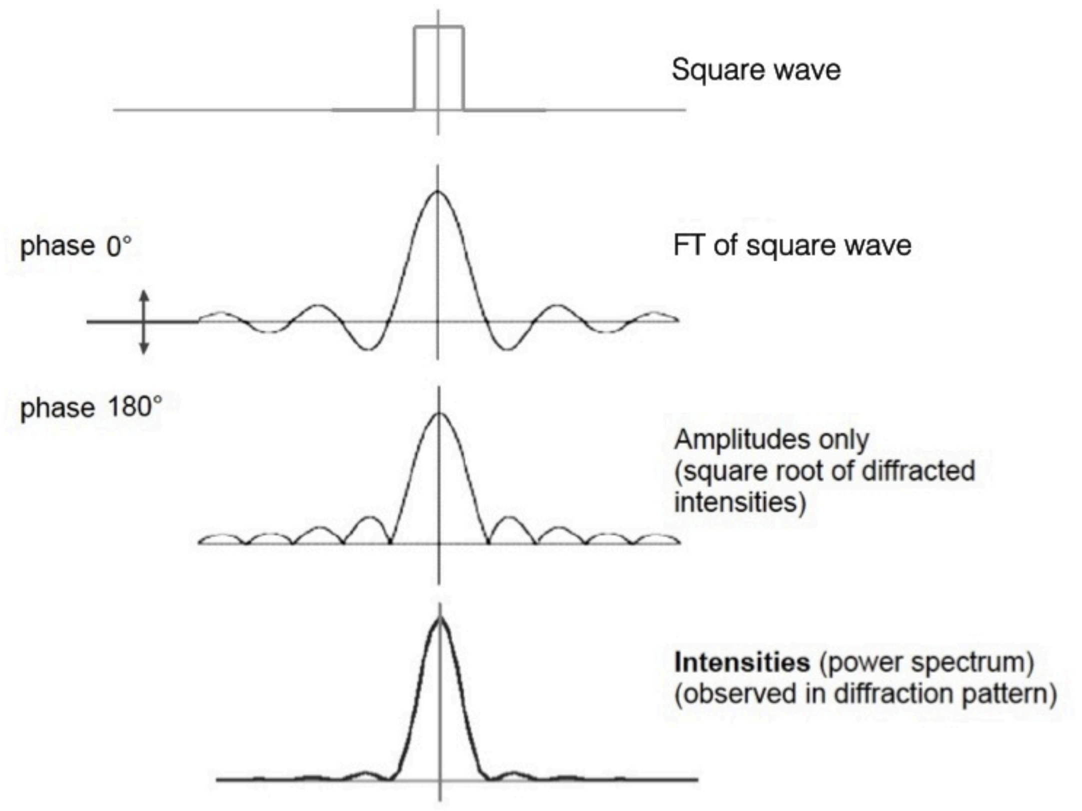
In prac-1, the files circle.hdf, rectangle.hdf, hhrot.hdf, circ3.hdf, 2circ3.hdf, circ3shift.hdf, rect1.hdf, 2rect1.hdf, micrograph1.mrc and micrograph2.mrc are the ones you will use. Check by listing them (**ls**). (Reminder: basic UNIX commands are listed on page 3 of the Getting Started document).

The purpose of this practical is to introduce you to images, their transform amplitudes and phases, the effects of filtering and measurement in transform units. Then you will do manual defocus determination from micrographs, and finally some particle picking. You will use EMAN2.3, for which there is extensive, although not complete, documentation:
http://blake.bcm.edu/emanwiki/EMAN2

Images, Fourier transforms and filtering

The purpose of this introductory part is to show you how to look at the Fourier transforms (FTs) of images. FTs can't be displayed like simple images, since they contain two components at every pixel, an amplitude and a phase angle. The amplitudes can be represented as grey scale values, as normally used to represent the transform of a micrograph, in order to check for drift and determine defocus. The phases are not normally viewed, but can be represented as colours on a cyclic scale so that 0 and 360° are the same colour. If you want to understand image processing, it is important to understand FTs and diffraction. FTs present the image as a function of spatial frequency (1/distance) instead of distance. Since many operations need to be done on specified ranges of spatial frequency, they are done in the frequency domain. Here you will view two simple 2D images and their FTs.

In one dimension, the object and its FT can be plotted as graphs:


Square wave
FT of square wave
phase 0°
phase 180°
Amplitudes only (square root of diffracted intensities)
Intensities (power spectrum) (observed in diffraction pattern)

The position of a feature in the image determines the phases of its diffraction – that is why the phases are so important in crystallography – if you want to know WHERE the atoms are in the structure, you must determine the phases of the diffraction spots.

To display images and their FTs, start by typing the command:
e2filtertool.py circle.hdf
This will give you a display window showing a white disk and a control window with drop down function menus. You can center the image in the display window by dragging with the right mouse button or by resizing the window.

Images, amplitudes and phases

Click on the image with the middle mouse button to bring up an image control window. (Note that you may need to move, expand or adjust the various windows or magnification slider to see all the options and to display data correctly). The image control window has options to display the real image, its Fourier transform (*FFT*) shown as amplitudes with colour-coded phases, or the amplitudes (*Amp*) or phases (*Pha*) separately. Look at the *amplitudes*. You may need to choose *auto contrast* and increase the view magnification (*Mag* slider or mouse wheel). You should see a set of concentric rings around a central maximum, a 2D version of the graph above, similar to the Thon rings that are used to judge the quality of micrographs. Note the effects of the limited number of pixels used to represent the circle. How does that affect the transform?

Then try *FFT*. If you magnify the display more, you will see that the phases are alternating between red and green (0 and 180°) in adjacent pixels. This is because the phase origin is at the center of the image, to make the real image display correctly. To see how the phases vary with the phase origin in the bottom corner, choose *Xform* from the function menu on the e2filtertool window and choose *phaseorigin.tocorner* on the right menu. Then check the box to the left of the *xform* menu bar. The FFT should now show alternating rings of phase reversal, like the FT of the square wave. Returning to *Real* on the image display will show you how this affects the image appearance. You may need to reduce the image display magnification to see something. To restore the image, uncheck the tick box at the *xform* menu. Close the e2filtertool window and repeat these operations with the image **rectangle.hdf**. To see the transform amplitudes, you may need to reduce the *max* value after choosing *autocontrast*. Note the inverse relationship between the repeating bands in the FT and the dimensions of the rectangle. To see the effect of shifts in the image on the phases, display **circ3shift.hdf** with **e2filtertool.py** as above and *Xform* - *phaseorigin.tocorner* selected. How would this change if the circle shift was larger or smaller?

Filtering

Close the previous windows and display the circle image with
e2filtertool.py circle.hdf.
Choose *filter* from the dropdown menu and check the box to the left of it. The right hand drop down menu will contain a list of available filters. The ones you should try out are lowpass, highpass and bandpass filters using using Gaussian (smooth) or tophat (sharp) fall off. Start with *lowpass.gauss*. If you hover the mouse over the titles or parameter boxes of the different sliders, while the window is selected, you will see a short text explanation of what they do. The best one to use is *cutoff_abs*, which gives the filter radius as a fraction of the maximum radius, which is set to 0.5. Move the slider over the range, allowing some time for the graphics window to refresh.
What does the low pass filter do to the image?
Bring up the image control window by clicking on the image with the middle button. Using display as *real* or *Amp*, look at the effects of filtering on both the image and the transform. Can you explain the relationship between the changes in the image and the effects of the filter on the FT? Repeat with high pass and band pass, and then try the series with the tophat filters.

Now close the filtertool window and start a new one with
e2filtertool.py rectangle.hdf
With this image, you can again look at the transform amplitudes (if *autocontrast* doesn't give good results, try reducing the *max* slider on the image control window). Then look at the FFT and try out the various filters - what happens with high pass and band pass, and how do the effects differ if you use the Gaussian or tophat versions?
You may need to restart e2filtertool.py if you do too many different operations. And note that the graphics window may not refresh until the filter is reapplied.

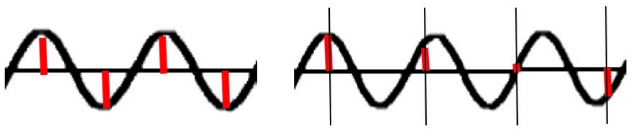
Measurement in transform units (tu)

In digital images, both the image and its transform are represented by an array of pixel values. In the image, the units are the familiar measure of distance, on a scale determined by the experimental setup, some number of Å/pixel. In this example, the box size is 512x512 pixels and we will keep the scale at 1 Å/pixel, so the image box corresponds to a square 512 Å in width. But what about the transform units? There will also be 512x512 of them. What does each one measure? Can you say what filter radius you would need to eliminate or retain features of a particular size?

Remember that the transform is in reciprocal space, so the units must be related to 1/distance. It depends on the box size – if the whole box is 512 pixels across, then 1 tu = 1/512, or, in this case, (1/512) Å^{-1}, so that:

1 tu = 1/(box size)

But the origin of the transform is in the centre of the box, making the maximum tu value only 256, half the box size – corresponding to the **Nyquist frequency**, which is the maximum frequency that can be represented by a box of 512 pixels. This limit of the frequency corresponding to half the box size can be understood graphically: the pixels must sample the highest frequency wave component twice per wave repeat, in order to resolve the positive and negative lobes of the sinusoid.



In other words, the Nyquist frequency is the sampling needed to reconstruct a particular sine wave. For example, the reconstruction of a 3 Å map in theory requires a sampling of 1.5 Å/pixel although in practice it is advisable to sample at 1/3 the required resolution, 1 Å, to allow for interpolation errors.

The tu is often given as a fractional value, a radius of 0 (origin of the transform) to a maximum of either 0.5 or 1 at Nyquist, depending on which convention is used.

How do we choose the filter radius? Suppose we want to resolve objects that are 114 Å apart, and filter out all the finer details than that. What tu value corresponds to this separation distance?

(1/114) ÷ (1/512) = 512/114 = 4.5, i.e. (box size)/(distance on image)

What does a filter cutoff correspond to in Å?

Real space distance = Box size/tu

The files circ3.hdf and 2circ3.hdf give you a chance to see the quantitative effect of the filter cut off value. They are images of one or two circles. Comparing their transform amplitudes will show you the effect of interference. Look at the phases of 2circ3.hdf, by using xform to set the phase origin to the corner as before, and then displaying the FFT. What does the interference do to the phases?

To test the effect of the filter radius, continue to display 2circ3.hdf with e2filtertool.py, but reset the phase origin by un-checking the xform box, so that the objects are back in the center of the image. Apply a low pass tophat filter and choose the slider "cutoff_pixels". This will allow you to set the filter radius in tu. The default maximum value set by the slider is 128, or half Nyquist. You won't see much difference on the image if you set it manually to 256. Look at the image and transform amplitudes at filter cutoff settings 128, 64, 32, 16, 8 and 4.5 (the tu value corresponding to the circle center to center separation of 114 pixels (512/114)). You can type the values into the window instead of using the slider. What are the critical tu (and corresponding real space) values at which the two objects start to distort, fuse and finally merge? This should give you an idea of how your structure may be distorted if you are interpreting features close to the resolution limit. You can also do this comparison with the rectangular objects in rect1.hdf and 2rect1.hdf, and/or use a low pass gaussian filter instead of the tophat, to see if that gives fewer artifacts.

A real picture

When you have investigated the behaviour of images, FTs and filters with the circle and rectangle, and understand tu, look at the effects of filtering a slightly more realistic image.
e2filtertool.py hhrot.hdf
(depending on your system, you may need to use the unrotated version hh.hdf)

How are different features in the image affected by low pass and high pass filtering?

Now you can try the filters on a real micrograph:
e2filtertool.py micrograph1.mrc

In the filtertool control window, choose filter – *lowpass.gauss*, and set the pixel size (check *apix*) to 2.44 Å. Select *cutoff_freq* (and deselect *cutoff_abs* or any other options if selected). Now you can check the effects of filtering on the micrograph, which is of GroEL on a carbon film, recorded with a DE20 detector on a Polara microscope. Try the other Gaussian filters to see which makes the particles easiest to discern. Now try **micrograph2.mrc** – this one was taken closer to focus and it is harder to see the particles.

Defocus measurement

In this part you will use EMAN2 to manually measure the defocus of the 2 micrographs. This is usually automated with a script but doing it manually will show you how the matching of parameters is done and how the parameter choices affect the outcome.

The **contrast transfer function** (CTF) is an approximation, valid for thin and weakly scattering objects such as biological samples, that describes the electron-optical distortions introduced by the microscope. You will hear more about it in tomorrow's lectures but we introduce the practical side of it here. Images need to be corrected for these optical distortions, and the key piece of experimental information needed is the actual defocus used, which is not identical to the value that was set on the microscope. The defocus depends on the exact height of the specimen area being imaged, something that varies with the local, microscopic details of the grid or support film.

The approximation to the CTF provides us with a formula, which requires us to know the electron voltage used (that determines the wavelength), the scale in Å/pixel (determined by magnification setting and detector), and the spherical aberration coefficient of the microscope Cs, which is a constant. The only unknown is the defocus, and the measured scattering (Thon rings) has oscillations whose positions depend on the defocus value. Therefore, trial and error calculations of possible CTF curves are compared to the experimental data to find the best match. The expression for the CTF is shown here:

$$\text{Phase CTF} = -2 \sin [\pi(\Delta z \lambda q^2 - C_s \lambda^3 q^4/2)]$$

where Cs is spherical aberration coefficient, ΔZ, defocus, q, spatial frequency and λ, electron wavelength.

e2evalimage.py micrograph1.mrc

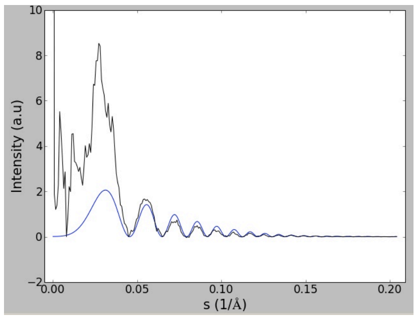
This brings up 4 windows, though some may be covered by others. Find the control window, the plot window, image and FFT windows. Set the 4 data parameters at the bottom of the control window: *A/pix* 2.44, *Voltage* 300, *Cs* 2.3 and *Box size* 256.
You will probably need to adjust the scaling on the various displays. For each display window, middle click will bring up its own display control panel.
Starting with the plot window, use its control panel to set *Y Min* to -2 and *Y Max* to 10 (entries near the bottom of the control panel). This will put the graph on a better scale. You can then close this control window. On the *evalimage* control panel, choose *refit*. On the plot, the blue curve is the fitted CTF and the black curve is the radially averaged FT of the micrograph. The radial curve is fine if there is no astigmatism in your image, but in general that is not perfect, so you also need to check the 2D presentation of the fit, on the FFT window. Resize the window or use the right mouse button to center the diffraction pattern, and bring up its control panel (middle click) to increase the mag and optimise the contrast. The fitted zeros are marked by circles – these would be elliptical to match astigmatism.
The micrograph window shows the size of the tiling boxes used to calculate the FT – the results of all boxes are averaged to give a better signal:noise ratio on the resulting curve. You can see how this affects the plot by trying 512 (*evalimage* control panel box size), instead of 256.

Now examine the effects of the other key parameters (*evalimage* control window) on the outcome. Select refit each time you do an adjustment and see if the fit improves or gets worse.

B factor
% Amplitude contrast (normally 7%)

The main problem is that there are strong features in the first part of the data (black plot) before the first zero, which are not fitted by the CTF model. Can you explain what these arise from? You can estimate what spacings the 2 or 3 sharp peaks correspond to, by estimating their position in 1/Å units, and then take the reciprocal to give the spacing in Å. (hint – you can measure features on the micrograph with the measure (*Meas*) tab on the image control window (middle click)). You will need to set 2.44 Å /pixel and magnify the display so that you can see the measurement line, which is black. Use the mouse to draw a line between the features you want to measure. The result will be shown as length in Å (*Len*).

Example of a fitted CTF:



When done, close all the display windows and repeat the defocus determination for **micrograph2.mrc**.

Part 2 Image preprocessing pipeline with Scipion

The purpose of this section is to introduce you to the first steps in processing single particle cryo EM images. For the rest of this practical and the next one, we will use the Scipion system, which is a framework for image processing that allows the user to set up an automated pipeline for running a series of programs to carry out a sequence of steps in image processing. Here we will start from the raw micrographs.

For the following Scipion and other practical, you will need to be in groups of 2 or 3. Before we start Part 2, we will cluster you into groups.