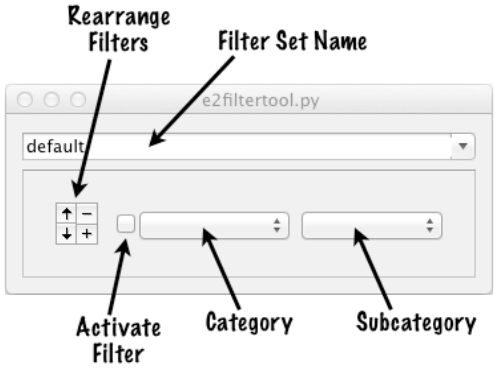
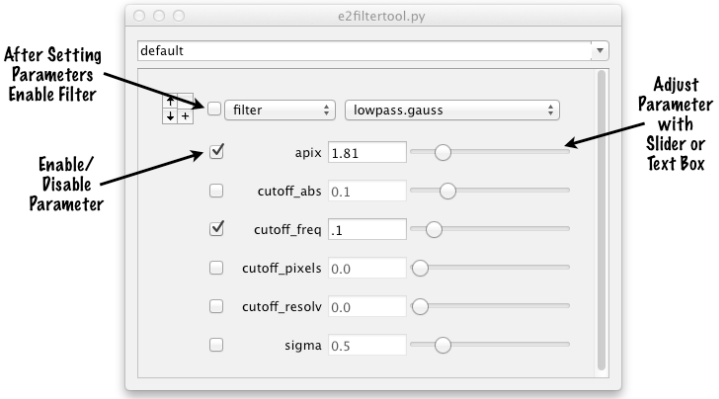
Basic Image Processing Tools

- EMAN2 uses a modular system of "processors" to perform basic image processing tasks such as normalization, masking, filtration, geometric transformations, etc.
- For a basic list of all available processors (>200 of them)
  - e2help.py processors
  - add -v 2 for a more detailed description of each one
  - add part of a name to filter the list, eg - *e2help.py processors xform -v2*



- The best tool to experiment with and learn about processors is e2filtertool.py.
- This is available from the command line as *e2filtertool.py <imagename>*
- or from *e2display.py* with the 'Filtertool' button (with selected image)
- When executed, two windows open, one showing a subset of the images and one for specifying image processing operations.
- Start by selecting a category, such as "filter"
- Then select a specific processor (subcategory), such as "filter.lowpass.gauss"
- Fill in desired parameters (check box for each used parameter)
- Finally check the box next to the Category popup to enable the filter. The image should be updated immediately, and continue to update in real-time as you adjust parameters with the sliders or text boxes.
- Press '+' to add a second processor to the sequence



- For details, http://blake.bcm.edu/emanwiki/EMAN2/Programs/e2filtertool

Image Processing from the Command Line

- To process 2-D images or stacks:
  - *e2proc2d.py <infile> <outfile> --process ...*
- To process 3-D images or stacks:
  - *e2proc3d.py <infile> <outfile> --process ...*

- *e2filtertool.py* stores all user-adjusted processor parameters in filtertool_default.txt, in a form suitable for use with *e2proc2d.py* or *e2proc3d.py*
- <infile> and <outfile> may be in any file format. See: http://blake.bcm.edu/emanwiki/EMAN2ImageFormats
- Remember to use --help to find the many other options supported by these programs

- Examples:
- Apply a mild high-pass filter to a stack of particles in-place to help remove ice gradients:
  - *e2proc2d.py particles.hdf particles.hdf --inplace --process filter.highpass.gauss:cutoff_pixels=2*
- Low-pass filter a high resolution 3-D map to 10 Å resolvability, and convert from MRC to HDF format:
  - *e2proc3d.py hiresmap.mrc filteredmap.hdf --process filter.lowpass.gauss:cutoff_freq=0.1*
- Apply a soft spherical mask with a radius of 32 pixels to a 3-D volume, convert from Spider to MRC:
  - *e2proc3d.py map.spi newmap.mrc --process mask.soft:outer_radius=32:width=4*
- If proj.hdf contains a stack of projections, and classes.hdf contains corresponding classes, merge into a single interleaved file for side-by-side comparison:
  - e2proc2d.py proj.hdf combined.hdf --interlv classes.hdf

Normalization

- Normalization, in our context, generally refers to the process of adjusting the mean value and standard deviation in each image in a set to make them meet some criteria. This is normally done by applying a linear transform to the pixel values, ie - multiply each pixel by a constant then add a different constant. There are many reasons for doing this, and different software packages have different standard normalizations. Here are the most widely used normalizations in EMAN2, and the uses for each:
  - normalize.edgemean - used ubiquitously in EMAN2. Sets the average value around the edge of the image to zero (by adding), and adjusts the standard deviation to 1.0. The average value around the edge of the image is assumed to be solvent in single particle analysis, so this adjustment makes solvent flattening and masking much easier
  - normalize - sets the mean value of the image to zero, and the standard deviation to one. Another common normalization, but this generally leaves the solvent with a slightly negative value.
  - normalize.circlemean - same as normalize.edgemean, but the solvent density is computed on a circular ring rather than from the edge of the box
  - normalize.unitlen - If the N pixels in the image are imagined to form a vector in N dimensional space, this normalization makes the length of that vector 1.0. This is useful in MSA/PCA and related processes.
  - normalize.toimage - this will adjust the density distribution of one image to optimally match a second reference image. For this to work, the second image must ostensibly contain the same 'shape' as the first.
  - normalize.bymass - normally used on 3-D volumes. Scales the image densities such that at an isosurface threshold of 1.0, the isosurface will contain a volume corresponding to a specified mass in kDa, assuming a density of 1.35 (typical for proteins). This can be used to roughly scale map densities so they can easily be compared, but is highly sensitive to resolution.

- Example:
  - e2proc2d.py particles.mrcs particles.mrcs --inplace --process normalize.edgemean

Thresholds

- Thresholding is the process of modifying pixel values according to some numerical limit. For example, in a CCD frame, you may occasionally get X-ray pixels with very high values. These values are meaningless, but can severely distort image processing. You can remove such pixels by selecting a threshold value representing the largest reasonable value in the image, then set any values larger than this to zero, the mean value of the image, or some other reasonable value. Here are the commonly used thresholds:
  - threshold.abovetozero - If the pixel has a value larger than the specified value, set it to zero
  - threshold.belowtozero - the same, but for very small rather than very large values
  - threshold.binary - sets any values larger than the threshold to 1.0 and any smaller to 0.0. Useful for generating binary masks from images.
  - threshold.clampminmax - if a value falls outside the specified range, it is shifted to the edge of the range
  - threshold.clampminmax.nsigma - the same, but specified in terms of standard deviations from the mean
  - threshold.binaryrange - one if the value falls within a range, zero otherwise
  - threshold.outlier.localmean - if a pixel is more than N standard deviations from the mean, it will be set to the average value of the surrounding pixels. This is a good choice for removing X-ray pixels.

- Example - a pretty effective filter for x-ray pixels
  - e2proc2d.py micrograph.hdf micrograph_noxray.hdf --process threshold.outlier.localmean:sigma=4

Mathematical Operations

- It is often useful to be able to perform some mathematical operation on an entire image. These are just a few of the diverse operations available in this category:
  - math.linear - apply a constant scaling and shift to each pixel
  - math.absvalue - each pixel -> absolute value
  - math.squared - each pixel -> each pixel squared
  - math.sqrt - each pixel -> square root of each pixel
  - math.laplacian - discrete approximation to the Laplacian (edge detection effect)
  - math.meanshrink - downsamples an image by taking the mean value of NxN or NxNxN pixel blocks
  - math.medianshrink - similar, but takes the median value of the NxN blocks
  - math.rotationalaverage - replaces the image with its circularly symmetric rotational average
  - math.rotationalsubtract - subtracts the rotational average from the image. This enhances azimuthal contrast
  - math.addsignoise - add Gaussian noise to the image, specified in terms of image standard deviation

- Example - downsample image data by a factor of 2 by local averaging
  - e2proc2d.py particles.hdf shrunk.hdf --process math.meanshrink:n=2

- Example - It is generally a good idea to low-pass filter images when downsampling to avoid an effect called aliasing. Of course, doing this often also causes other artifacts...
  - e2proc2d.py particles.hdf shrunk.hdf --process filter.lowpass.tophat:cutoff_abs=0.5 --process math.meanshrink:n=2

Filtration

- Filtration is a complicated topic, and requires a good understanding of Fourier transforms. There are many different types of filters. Most are 'linear filters' involving multiplication of pixel values in Fourier space, but there are also a variety of specialized filters, such as the bilateral filter, which perform much more complicated operations.

- Standard filter parameters
  - Virtually all of the highpass and lowpass filters accept a common set of options for specifying the cutoff resolution of the filter, allowing you to experiment with different filter shapes with the same cutoff easily. Some filters will also require a number of additional parameters:
    - cutoff_freq - specify the cutoff resolution in terms of the real physical dimensions of the object. This is expressed as 1/Å. ie - a 10 Å filter would be specified as 0.1 (1/Å).
    - cutoff_abs - specify the cutoff in terms of Nyquist frequency (the maximum frequency represented in an image) as 0.5. ie - to filter at 1/2 Nyquist, you would specify 0.25
    - cutoff_pixels - specify the cutoff in terms of pixels in Fourier space. ie - if you have a 64x64 image, Nyquist corresponds to 32 pixels. This can be very useful because 1 pixel basically corresponds to the number of oscillations within the box. ie a high-pass filter with a cutoff of 1 pixel will only remove frequencies equivalent to the size of the box.
    - apix - normally the A/pix value is provided in the image header, but if that is inaccurate for some reason, you can override the image A/pix by specifying it directly to the filter.

Low-pass Filters

- Lowpass filters - A lowpass filter passes low resolution information and blocks high resolution information. If you have a reconstruction containing too much detail/noise, you would normally apply a lowpass filter of some sort to filter it to a level of detail appropriate for the noise in the map. Here are some common filters:
  - filter.lowpass.gauss - A Gaussian (bell-shaped curve) low-pass filter. This sort of filter has many useful properties, such as having the same shape in both real space and Fourier space. It produces very smooth effects with little 'ringing' or other artifacts, but many people consider it too smooth features TOO much.
  - filter.lowpass.tophat - This is the sharpest possible filter, transitioning from 1.0 to 0.0 in one pixel. Unfortunately it also produces very strong 'ringing'. It does insure that no information survives beyond the cutoff frequency, unlike most other filters. This can be required for many validation methods.
  - filter.lowpass.butterworth - This is an adjustable width filter. The sharper the filter the stronger the real-space artifacts. This filter can smoothly vary between the two extremes.
  - filter.bilateral - while it lacks 'lowpass' in its name, this is a nonlinear lowpass filter, which is very similar to a diffusion filter. Its parameters allow the amount of filtration to vary with the local features of the map. Takes some experimentation to use, but can produce very nice results.
  - filter.lowpass.randomphase - This filter is largely used for validation. It randomized all of the Fourier phases past some cutoff resolution, effectively eliminating all information at high resolution. If an initial model is phase-randomized, and reconstruction is able to recover high resolution signal, this can be used as an argument that the features are not due to model bias.

- An important trick - one task people often need to perform is sharpening maps which have been overfiltered during reconstruction. This is often called 'B-factor correction'. This is normally handled differently in EMAN, but if you want to perform a simple B-factor correction, you can use a lowpass Gaussian filter and simply specify a negative value rather than positive value, and it will sharpen rather than blurring. There is also a filter called filter.gaussinverse which can be used for this purpose.

Highpass Filters

- Highpass filters retain high resolution information and eliminate low resolution information. When you perform an edge detection or enhancement in photoshop, this is effectively a type of highpass filter. The first maximum of the CTF acts like a high-pass filter, and is responsible for the strong dark ring appearing around the outside of objects in electron micrographs. While highpass filters can do a number of useful things like reducing ice gradients and other artifacts, they can also make solid object appear hollow and cause other issues if applied too aggressively.
  - filter.highpass.gauss - Instead of multiplying by a Gaussian, exp(-B x^2) as is done for a lowpass filter. This is a multiplication by 1-exp(-B x^2). Note that this is NOT the same as performing B-factor sharpening, which involves multiplying by exp(B x^2).
  - filter.highpass.tophat - As with lowpass, this is a transition from 0 to 1 at the cutoff frequency. This is a very aggressive highpass filter, and will tend to produce strong ringing artifacts.
  - filter.highpass.butterworth - again, similar to the lowpass version, but with the opposite frequency behavior.
  - filter.highpass.autopeak - this will attempt to filter out the sharp variable height peak generally caused by ice gradients in almost all electron micrographs. Unlike most filters, this filter does not completely eliminate low resolution information, to tends to leave fewer artifacts. However, in some cases it extends to higher resolutions than it should, so always double check the results.

Specialized Filters

- filter. bilateral - This filter performs a nonlinear filtration with both a typical spatial cutoff but also with an intensity 'sharpness' threshold. This filter tries to filter out noise in 'flat' regions of the image, but preserve edges. It is similar in many respects to a diffusion filter.
- filter.ampweight - multiplies each pixel in Fourier space by its own amplitude (preserving phase). This can produce an effect similar to denoising.
- filter.flattenbackground - subtracts the local mean over some distance from each pixel to help flatten the background (remove gradients). Unlike highpass filters which can produce a lot of artifacts, this can produce very useful background subtraction with minimal artifacts.
- filter.lowpass.autob - filters/sharpens a map based on the concept that the power spectrum should be fairly flat over intermediate resolutions (as proposed by Richard Henderson). This can work well for high resolution maps, but is not EMAN's normal solution to this problem.
- filter.matchto - This will filter one image to match the radial power spectrum of a second reference image. If you want to make a difference map between two maps, applying this filter will insure that they are identically filtered, so they can be subtracted without too many artifacts.
- filter.wiener.byfsc - You must provide an FSC curve to this filter. The FSC will be converted to an SSNR, which in turn is used to compute a Wiener filter. For a 'perfectly unfiltered' map, applying this filter optimizes the visible level of detail based on the real noise present in the image. This method is used automatically in EMAN to filter reconstructions to exactly filter to match the computed FSC.