The Image Processing Cookbook 6.0

John C. Russ

0. Introduction

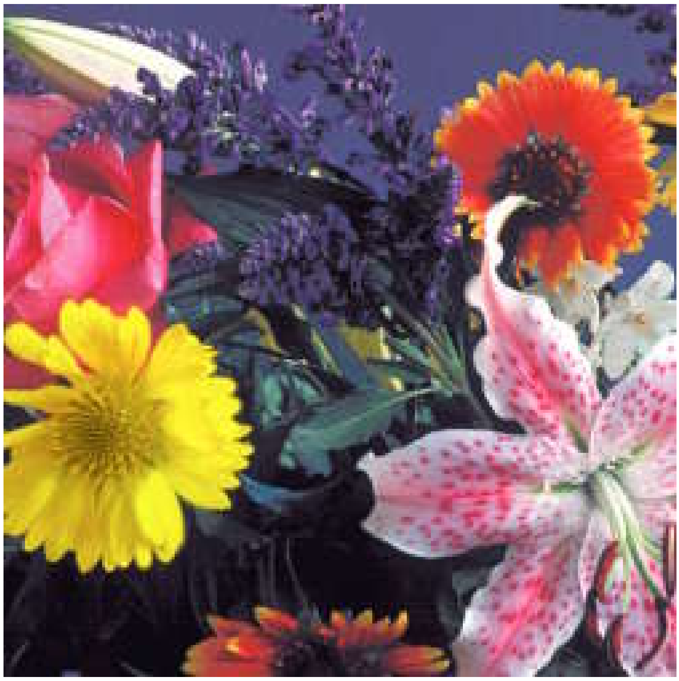
The process of image analysis - obtaining meaningful qualitative and quantitative information from images - typically requires several steps performed in sequence. Many software programs contain the basic tools, but Fovea Pro and The Image Processing Tool Kit have been designed to provide a comprehensive set of algorithms in the form of plug-ins that can be used with a variety of programs, ranging from the inexpensive (Paint Shop Pro) to professional (Image Pro Plus). They are particularly effective when used with Adobe Photoshop, because of its ability to handle 8 and 16 bit per channel images, display images in layers, and manage color spaces. Also, Photoshop provides a well documented and familiar environment for many users, supports a variety of file formats, acquires images from many diverse sources (cameras, scanners, etc.) and manages color printers. This tutorial will assume a basic level of familiarity with Photoshop. Books such as David Blatner and Bruce Fraser “Real World Photoshop” (Peachpit Press, isbn 0-321-24578-4) provide excellent guides to the effective use of the program, supplementing the program’s own documentation and help files. For a deeper explanation of how the various plug-in algorithms function, refer to John Russ “The Image Processing Handbook,” 4th edition, CRC Press, Boca Raton FL, 2002 (isbn 0-8493-1142-X). A more detailed guide to stereological measurements can be found in Russ and Dehoff, “Practical Stereology,” 2nd edition, Plenum Press, New York NY (isbn 0-306-46476-4).

The following roadmap offers a condensed guide to the most common procedures used in processing and measuring images. In the great majority of typical situations, this tutorial will cover the required steps. Follow this guide in order, selecting only those steps that apply to your particular images. For instance, for a grey scale image you would skip the steps that pertain to color images, but procedures to reduce image noise would be performed before edge enhancement; for images with good, uniform contrast you can skip steps that correct nonuniform illumination, adjust contrast, or increase local contrast, etc. For each topic, examples are shown that will help explain the procedures and show results on a wide variety of images. These examples do not exhaustively cover the various plug-ins or their options. The second part of the documentation is organized by menu and plug-in, and is a more traditional manual on the use of the software. The emphasis in this tutorial is on when and why to apply various techniques to images, to teach how to perform the various steps involved. Actual hands-on practice is essential for learning to effectively utilize the software tools. The possibilities illustrated for processing and measuring images will help you to plan procedures for your own work.

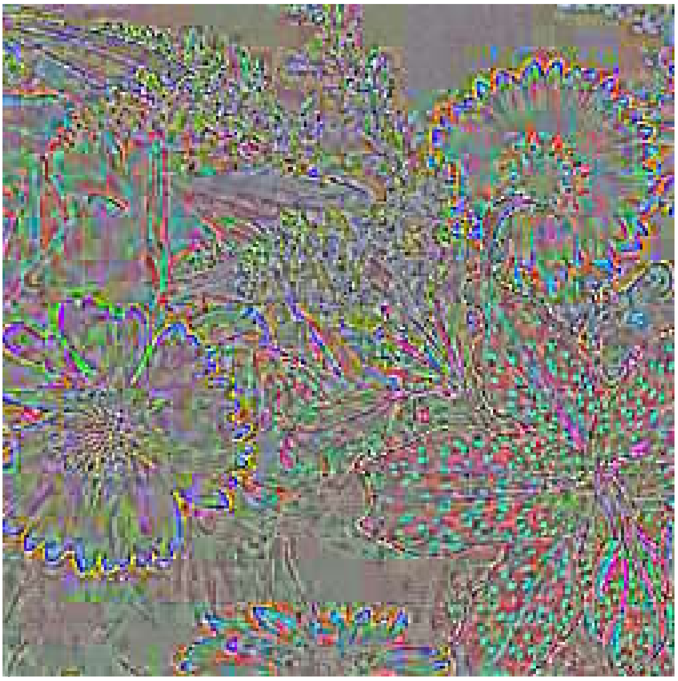
1. The program environment

This is not intended as a tutorial on Photoshop, nor a comprehensive guide to cameras and scanners, but a few reminders may be helpful. Obviously, it is important to acquire the highest quality original images. The selection of the type of camera (color, monochrome, video, digital, tube-type, CCD, CMOS, etc.) or scanner (single pass, three pass, bit depth, reflective or transparency, etc.) influences the need for processing the resulting images, because of the characteristic types of artefacts that each produces (noise - both periodic and random, real resolution as opposed to the number of stored pixels, color response, linearity, shading, etc.). Once acquired, images are stored in files, and many different formats are available. TIFF files are usually a good choice on most computer platforms and are read by most software packages. PSD is also cross-platform and accommodates text and drawing layers, as well as Photoshop’s adjustment layers. Avoid absolutely (!) any type of lossy compression (JPEG, wavelet, fractal) since you cannot be sure what information will be lost or changed that might be important in subsequent analysis.

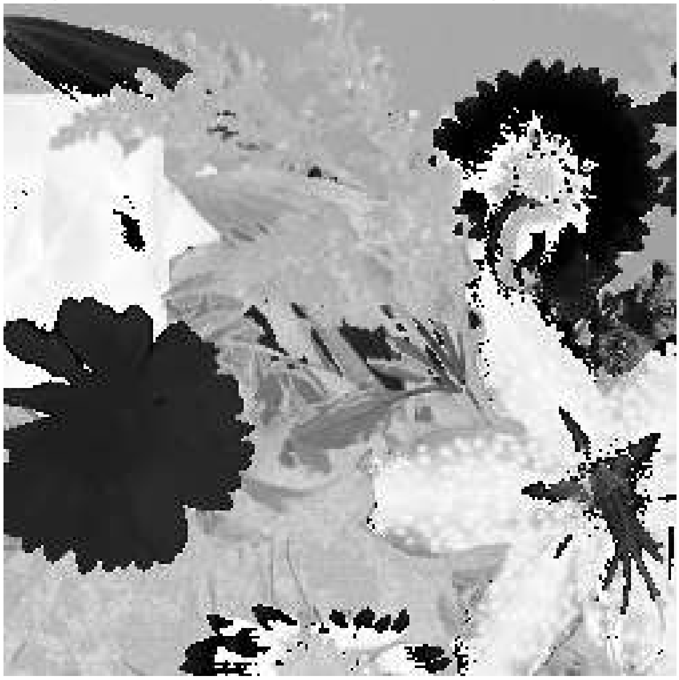
If you are not aware of the damage done by lossy compression, try the following: subtract an original image from the compressed version to see the change in colors, size, edge locations, and other details. Then examine the hue channel of the compressed image to see how much resolution has been lost.



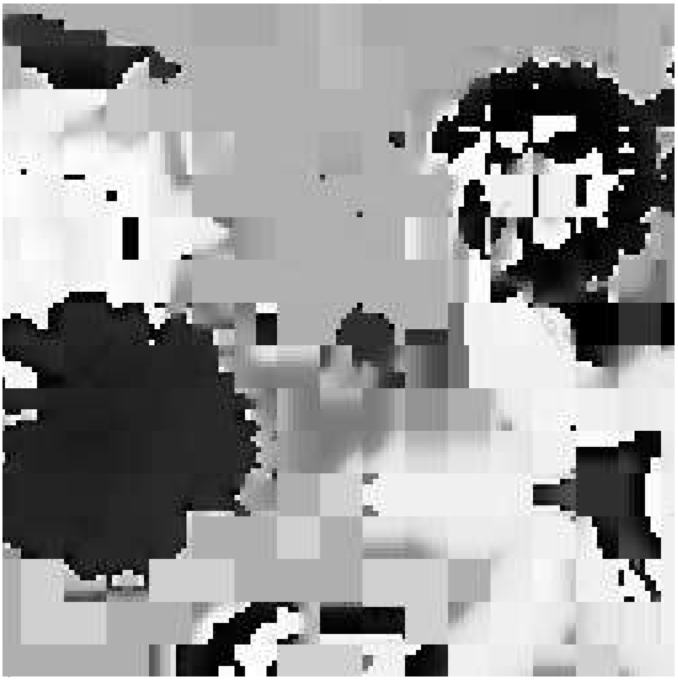
Fragment of the original *Flowers* image"



Difference between the original and JPEG version



Hue channel of the original image"



Hue channel after JPEG compression

There are many acceptable choices for printing with dye-sub, ink jet, and laser printers. Printer type and selection of ink and paper affect the image quality and must be balanced against the uses for the prints (draft, final report, publication, poster, etc.). For correct color rendition it is essential to use ICC profiles for all output devices (the screen as well as the printer, but this is much more difficult for LCD displays than for CRTs). Calibration devices and software are available, such as the GretagMacbeth Eye-One system, that can control all of these devices. The dynamic range of prints is less than that of the screen display, which in turn is less than that of film, and of human vision. The Photoshop browser shows thumbnail images and provides a basic capability to organize folders of images, enter keywords, etc. For large image sets, the use of a database to manage the collection is important, and this is still an evolving field.

The usual image modes for processing and measurement are 8 or 16 bit per channel greyscale or RGB color. If an image has been acquired in another mode (e.g., indexed color, CMYK, etc.) you can convert the mode (**Image–>Mode**) but it may not have the full range of brightness or color information present. Imaging devices that capture more than 8 but less than 16 bits of dynamic range per channel, such as cooled cameras, produce images that are generally stored and processed as 16 bit images.

The “frontmost” or selected image window is always the one being processed or edited. Some operations require two images (e.g., subtraction). This is always done by selecting the second image (which will be used but never modified), and then choosing **Filter –> IP•2nd Image –> Setup**. That copies the image into memory where it will remain (unaltered) for any two-image operations until replaced. Photoshop also allows images to be layered on top of each other, and combined in various ways, but while very useful for visual comparisons, that capability is not used in the following examples.

Photoshop offers a conventional “undo” capability, but also has a history palette that provides direct access to previous states for each open image. Mastering this important tool is strongly recommended. It is also possible (in Preferences) to have the entire history saved to a text file or with the image as a way to document the procedures used.

Selection of regions within images can be used to restrict processing or measurement. Several selection tools are available. The marquee tool draws rectangular or elliptical regions, the lasso and polygon tool draw irregular shapes, and the wand tool selects regions within a brightness tolerance of the point clicked on. The shift and alt or option keys may be used to add or subtract regions. It is often useful to display the selected region as a “quickmask,” which can then be processed or edited as a greyscale or binary image. The quickmask shows the selection as an alpha channel, with 8 bit values. Some Photoshop tools (e.g., the “fuzziness” slider in the Color Selection procedure, and feathering a selection) make use of this proportional capability. The marquee outlines and plug-in routines consider a pixel “selected” if the alpha channel value is at least 128 (out of 255). It is generally safest to turn anti-aliasing off when region outlines are used to select areas of an image for processing or measurement, so that pixels are either fully selected or not.

Tools for adding text, lines and other labels are often very useful, with selection of colors, fonts, etc., as required. Keeping the text, etc., separate from the original image pixels is usually a good idea, preserving image details and also allowing for better quality graphics. The foreground and background colors can be instantly reset to black and white, respectively, by clicking on the icon in the tools palette or by pressing the “D” (default colors) key. There are many other keyboard shortcuts, and the user can define additional ones, but with a few exceptions they are not used here for clarity.

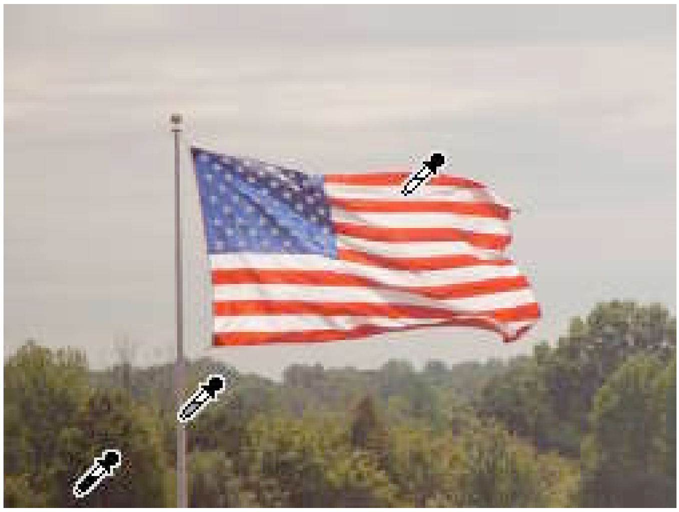
Some additional notes related to the use of Photoshop Actions for automation are collected in Section 7.

2. Correction of image defects
2.A. Color images

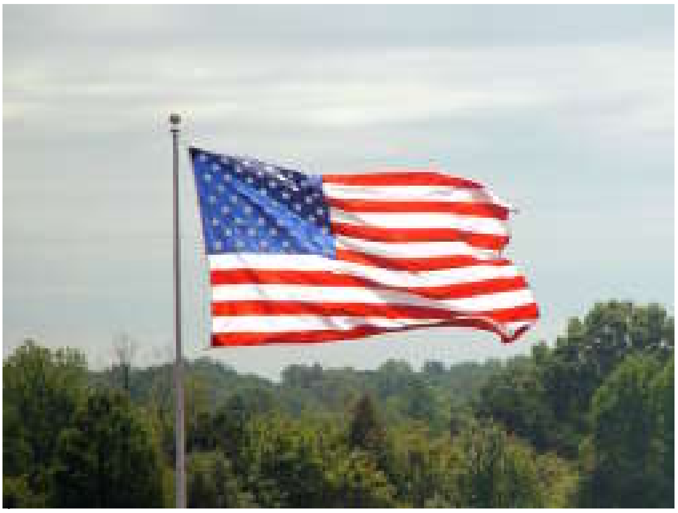
Many of the adjustments to color in images are performed using the built-in Photoshop tools, but a few require the more advanced algorithms of the plug-ins. It should be noted that in many cases the actual colors in images do not matter except as a way to distinguish one structure from another, and matching the appearance of the original scene, the on-screen presentation, and hardcopy printout is not required.

2.A.1. Is color correction required?

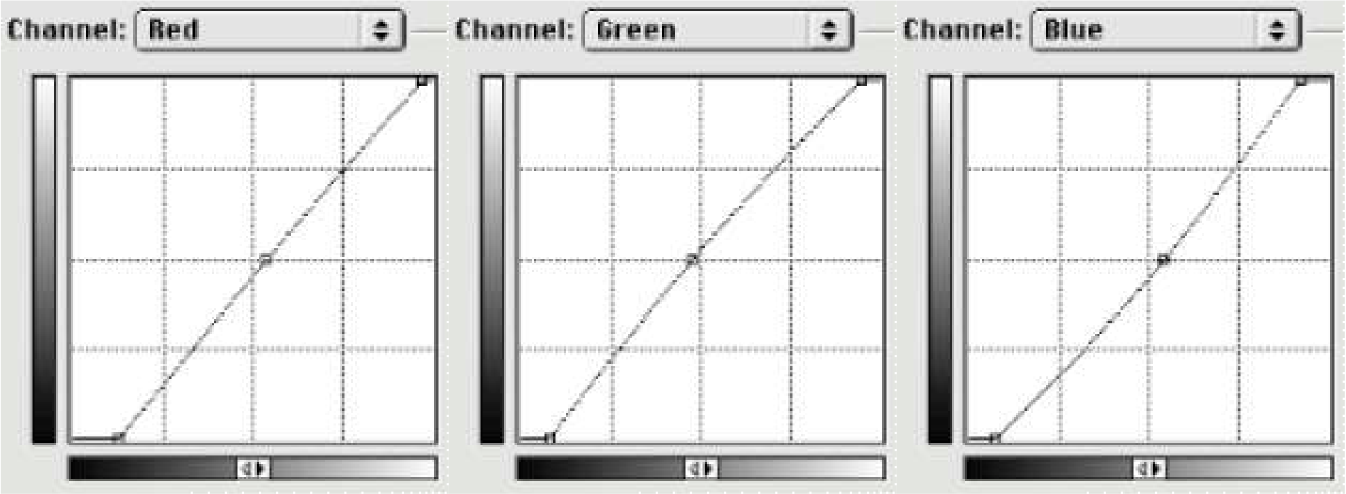
The **Image–>Adjust–>Levels** controls can be used to alter the contrast and brightness of the individual RGB channels, but for the greatest control it is preferable to use the **Image–>Adjust–>Curves** routine. This displays the curves that map the original to the processed brightness in each channel, each of which can be manipulated in detail. However, the most efficient method uses the eyedroppers by which the user can manually pick out white, black and neutral grey points in the image (assuming that such locations exist). By adjusting the proportions of red, green and blue to be equal (hence neutral greys) at those three points, the other colors in the image are usually corrected as well. In the example the black, white and grey points that were used are marked. The resulting curves are shown as well.



Original *Flag* image"

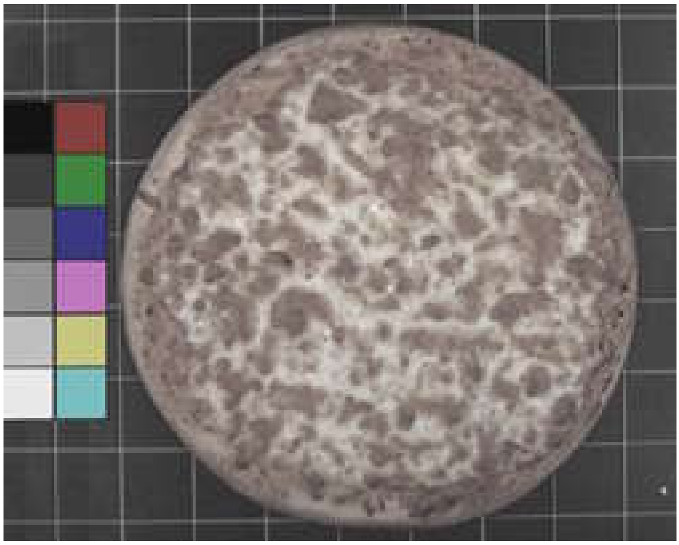


Color adjustment

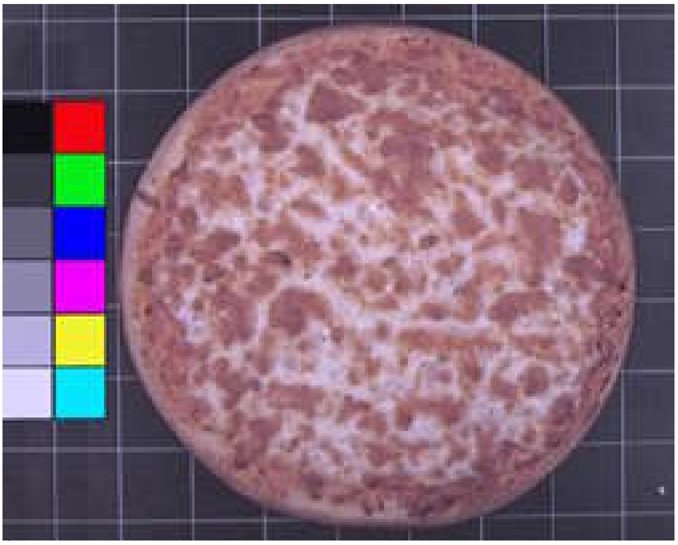


Adjustment curves for each channel.

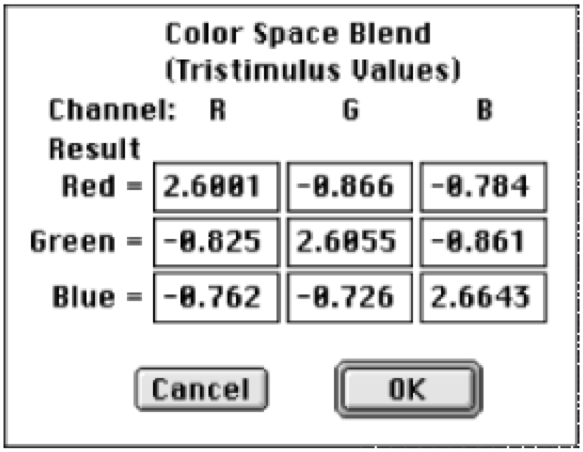
Tristimulus corrections offer a more powerful and complete way to correct colors in images by combining the values from the different channels. This generally requires measuring the RGB components in an image that contains known color panels, for example by including a color chart in the scene. The **IP•Color–>Normalize Color** plug-in will try to find the reddest, greenest and bluest patches in the scene (as in the example image shown), but in some cases it is necessary to place a selection on the image to locate them. The program measures the RGB components and calculates the tristimulus correction matrix, which is stored in memory. This can then be applied to the entire image, or to an entire series of images acquired with the same illumination conditions (select **IP•Color–>Tristimulus Correction** to apply the stored values)



Original *Pizza1* image"



After tristimulus correction



Color Space Blend
(Tristimulus Values)

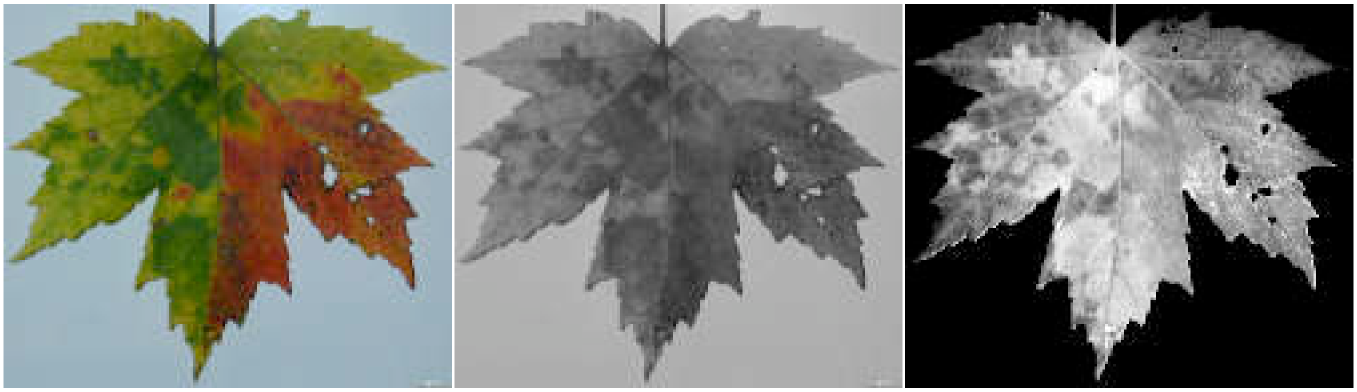
Channel:	R	G	B
Result			
Red =	2.6001	-0.866	-0.784
Green =	-0.825	2.6055	-0.861
Blue =	-0.762	-0.726	2.6643

Calculated tristimulus matrix

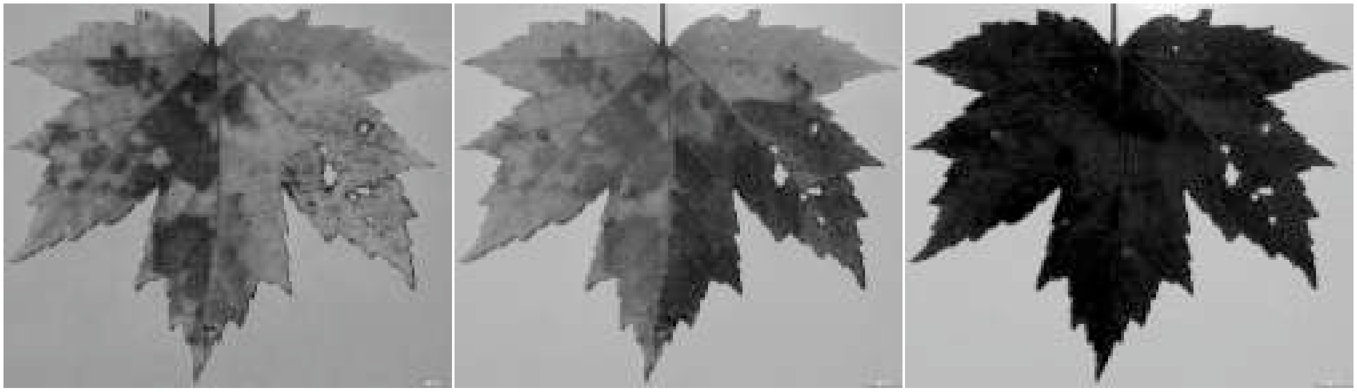
2.A.2. Color filtering and channel separation to improve contrast

When color information is used to distinguish different structures, the application of color filters or use of individual color channels or a combination of them may improve the contrast and simplify selection. It is rarely appropriate to simply convert the image mode to greyscale. In the Photoshop Channels window, clicking on the R, G or B channel will display just that channel, and allow it to be processed as a grey scale image. You can also select **Split Channels** from the Channels window to replace the RGB Color image with the three separate greyscale images containing the red, green and blue channels.

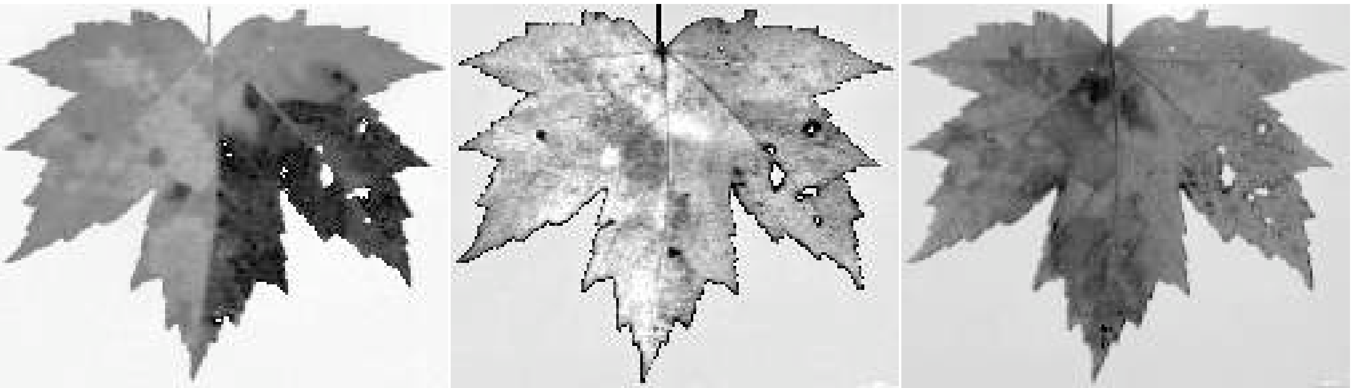
The image can also be converted to Lab mode and the luminance, a and b channels accessed in the same way. The **IP•Color–>Color Filter** routine can be used to isolate the Hue, Saturation or Intensity channels, or to apply any color filter to the image (functionally the same as placing a filter of that color in front of the camera). Finally, the **IP•Multichannel** plug-ins use principal components regression to find an axis in color space along which the greatest contrast in the image results, so that the first channel shows the optimal grey scale contrast. After the color filter or PCA routines, the image is still in RGB mode and should be converted to greyscale mode to simplify and speed up further processing. The example image illustrates these color channels.



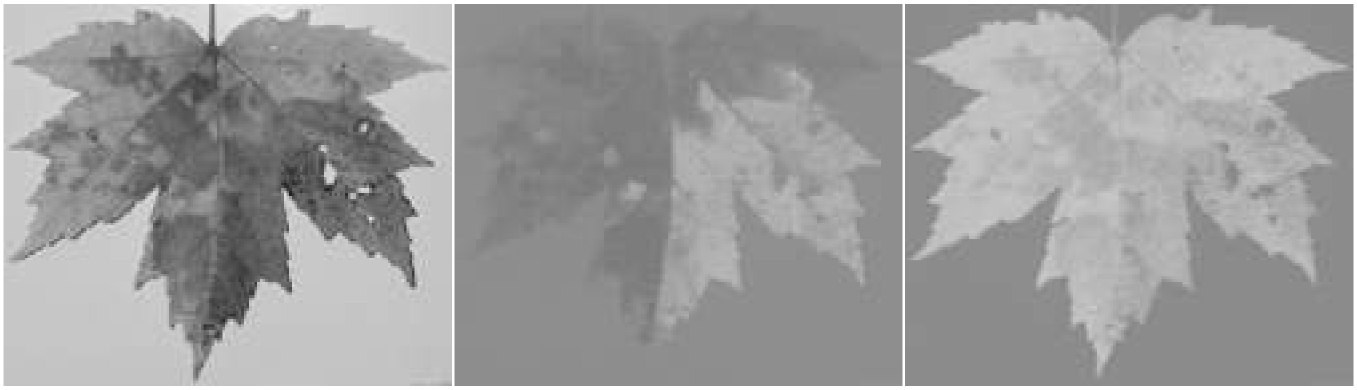
Original Leaf image, conversion from RGB to Greyscale, and Optimal Grey result



Red, Green and Blue channels

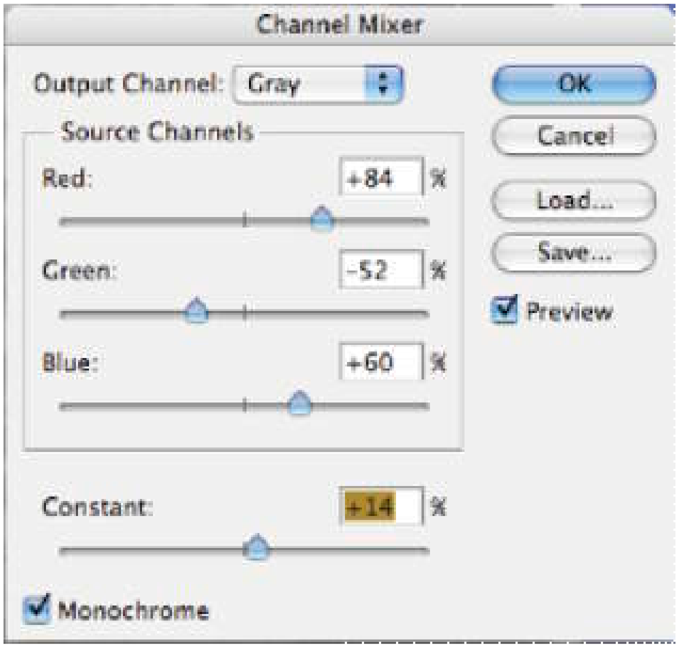


Hue, Saturation and Intensity channels



Luminance, a and b channels

Another way to interactively combine the RGB channels to produce a grey scale result is the **Layer–>New Adjustment Layer–>Channel Mixer** dialog. This allows mixing together arbitrary amounts of each channel to produce a desired result.



Channel Mixer

Output Channel: Gray

Source Channels

Red: +84 %

Green: -52 %

Blue: +60 %

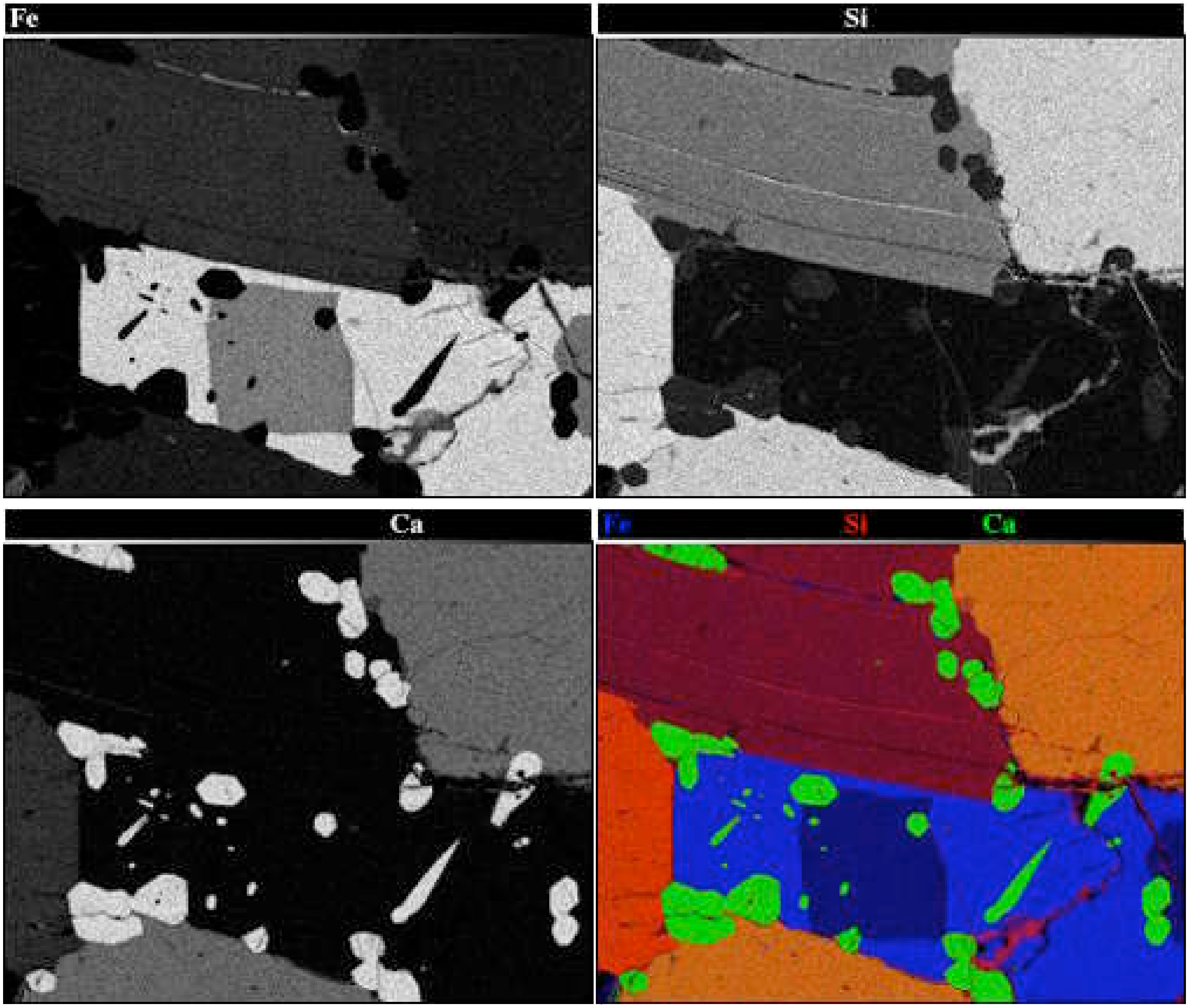
Constant: +14 %

☑ Monochrome

☑ Preview

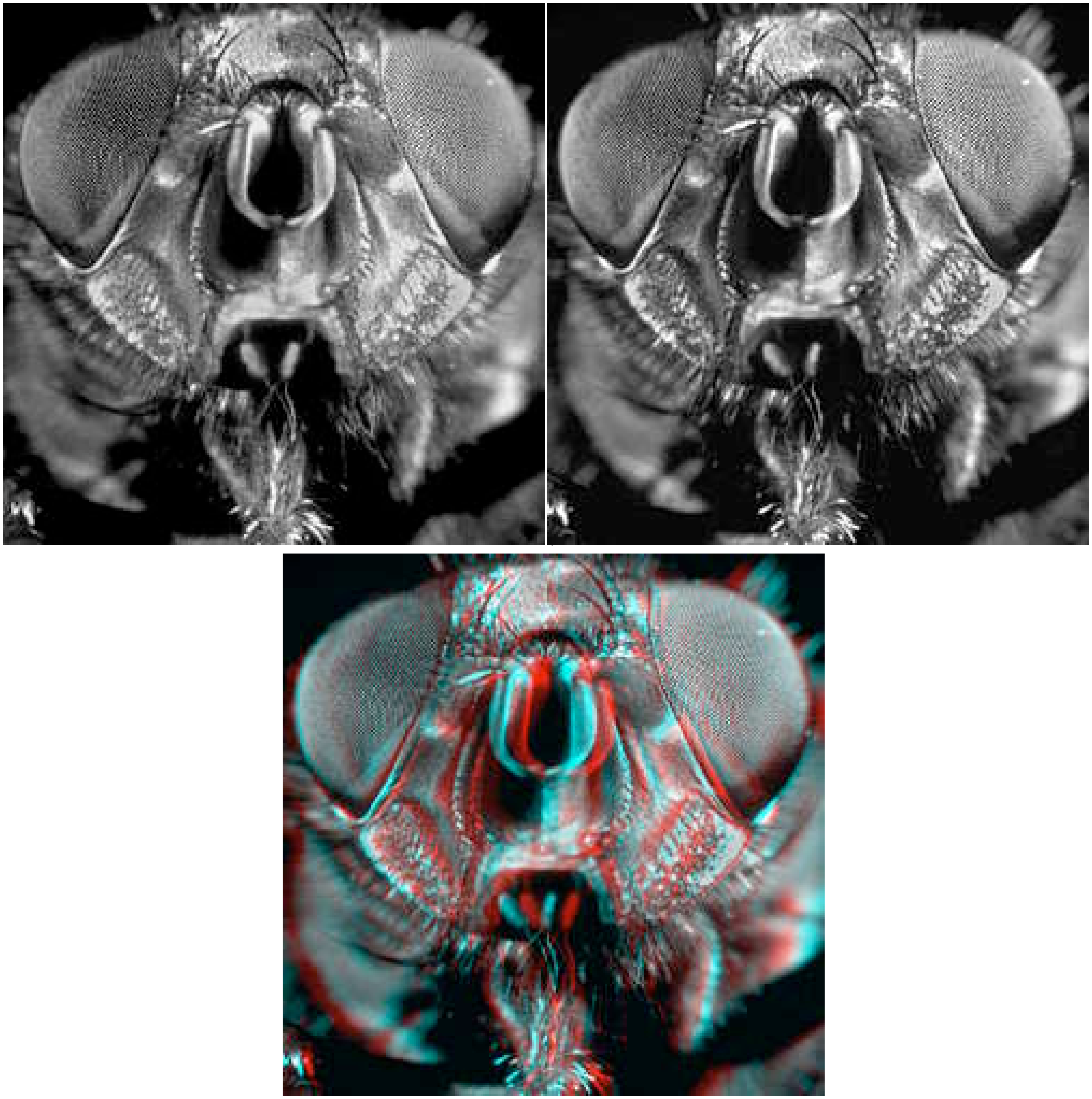
2.A.3. Merging color channels

Grey scale images can also be combined to form a color image. This is often done when separate images of the same scene are captured with different color filters, and of course the channels may also represent non-visible colors (infrared or ultraviolet) or other signals (such as X-ray maps from the SEM). Only three images can be combined to produce a viewable image, because color space is three dimensional (ultimately this is because human vision has three kinds of color-sensitive cones in the retina), and any displayable color can be produced by combining various proportions of red, green and blue. When three grey scale images with the same dimensions are open, the **Merge Channels** function in the Channels window will combine them into an RGB image (you can specify which goes into which channel). When there are more than three channels, finding the most effective combination of colors is largely a trial-and-error process. Sometimes sums or ratios of individual images can be combined and displayed in a single channel. Note that Photoshop allows merging more than three channels into a “multichannel” image, which may be convenient for keeping multiple images together and is essential for principal components analysis (described in section 2.A.4), but that these cannot be displayed in meaningful colors.



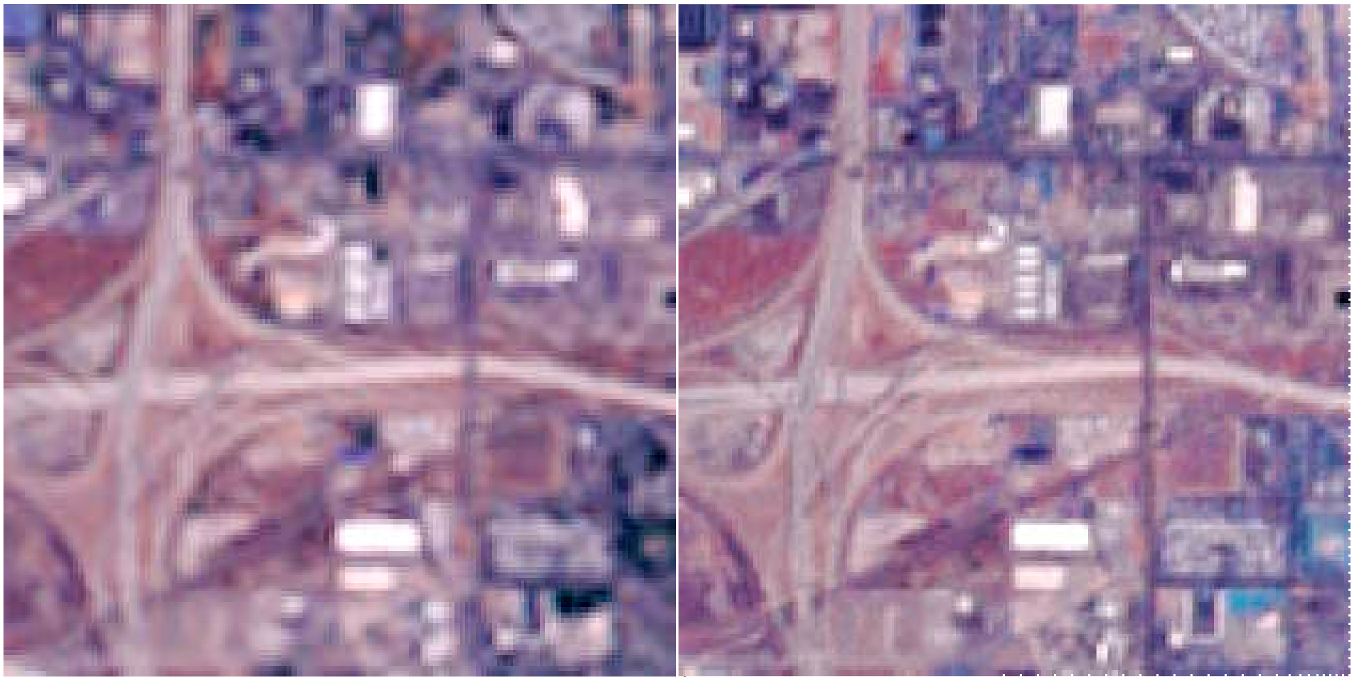
Three elemental X-ray maps and the merged RGB result (*Mica* images)

Another way to efficiently combine images to produce a color composite is to insert a greyscale image into a single color channel of another image. The example shows this being done to create a stereo pair that can be viewed by using glasses with a red left eye filter and a blue or green right eye filter. Place the left eye image into the second image memory (**IP•2nd Image–>Setup**), then select the right eye image and convert it to RGB color. This is necessary in order to have color channels in which to place the other image, but initially the appearance will be unchanged because the same information will be present in all three channels. Then select **IP•Color–>Transfer Channel** and choose the red channel.



Right and left eye views and the combined stereo pair image (*FlyEye*)

To illustrate fact that a color image can be interchangeably thought of as consisting of either RGB or HSI color channels, the next example uses a set of satellite images of Salt Lake City. The first three are from a moderate resolution satellite that captures separate images in visible blue, green and red. Combining them as shown above with image *SaltLake_1* assigned to the blue channel, *SaltLake_2* to green, and *SaltLake_3* to red, produces a color image as shown (the contrast has been maximized using the **IP•Adjust –> Contrast** plug-in with its Auto setting; this is discussed in a following section). Another image of the same scene (*SaltLake_8*) was acquired with a different, high resolution satellite. This image is panchromatic, covering the entire visual range of colors. Inserting it into the intensity channel of the image formed by merging the three RGB images produces a result in which the overall image sharpness is improved. Human vision judges image quality primarily by the intensity variations, and tolerates lower resolution for the color information (a fact that underlies television broadcasting and image compression). Note that the original color image was created by merging RGB channels but is then treated as an HSI image to replace the intensity channel. The conversion between these color spaces is handled internally and automatically by the software.

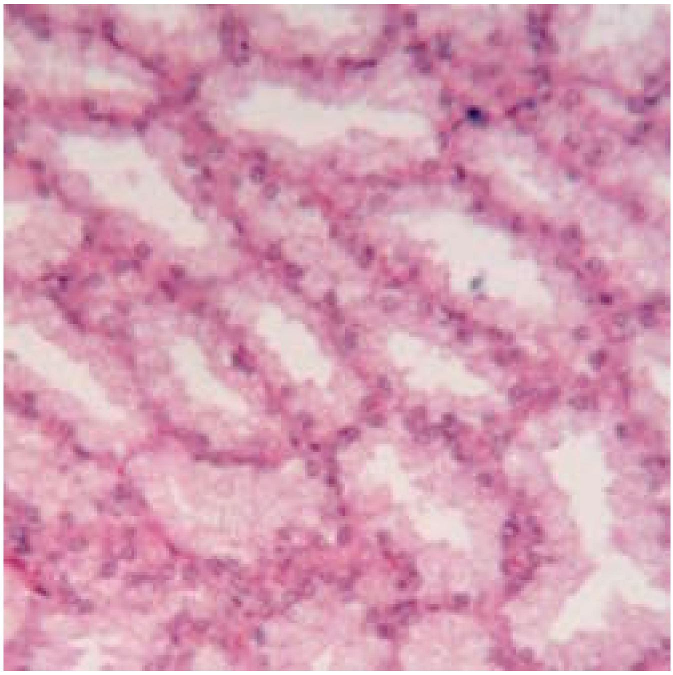


Detail of images formed with RGB channels (left) and inserting high resolution image into I channel (right)

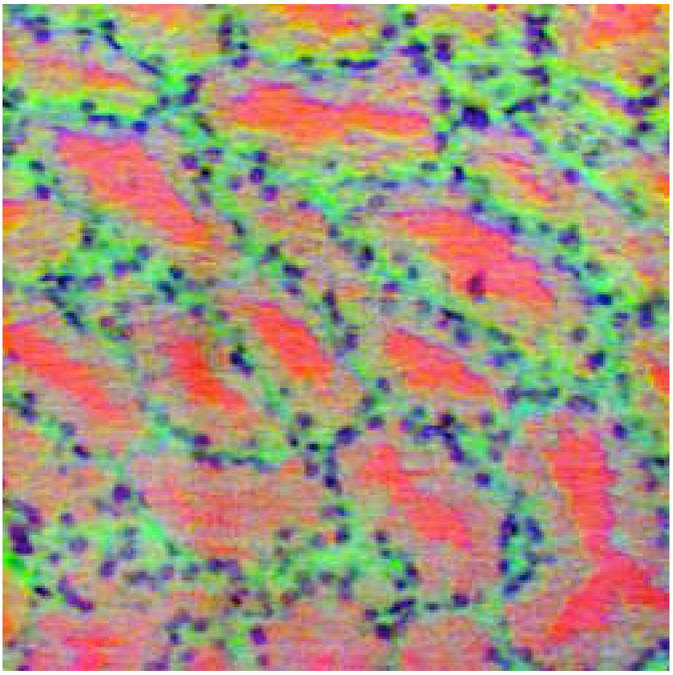
2.A.4. Principal Components Analysis

Color images, or in fact any images containing multiple channels of data, can be analyzed using a technique known as Principle Components Analysis (PCA). This statistical method is easiest to visualize for an RGB image. Instead of the three original red, green and blue axes along which the pixel values can be plotted, PCA finds a new set of orthogonal axes rotated to align with the principal axes of the actual data. The first axis is that along which the data have the greatest scatter or dispersion, and so forth.

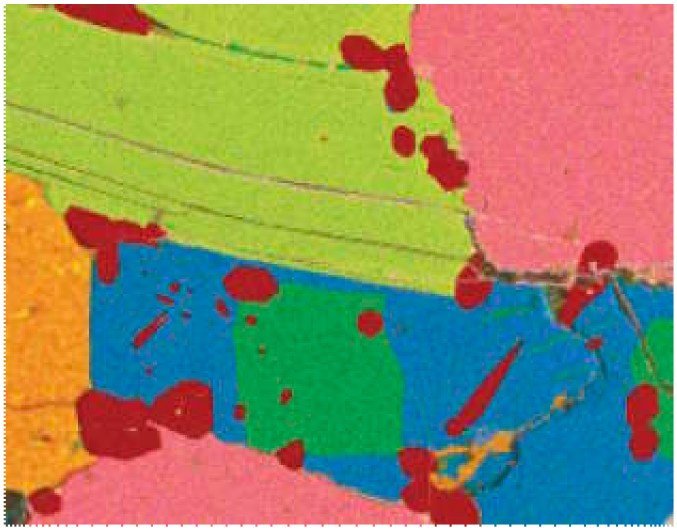
One use of this tool is to find the maximum contrast available in color images. As shown in the example, the staining of the tissue provides only weak visual distinction of the various cells and nuclei. The PCA technique (**IP•Multichannel –> Compute PCA Transform**) finds the most significant axes for the information and **IP•Multichannel -> Apply PCA (Forward)** displays those combinations of values using the red, green and blue channels to effectively separate the structures.



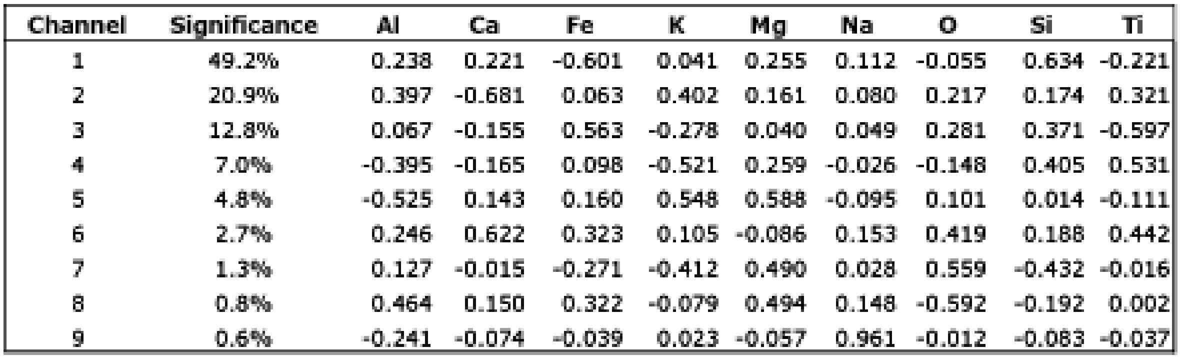
Original *Turbinate* image"



Principal components

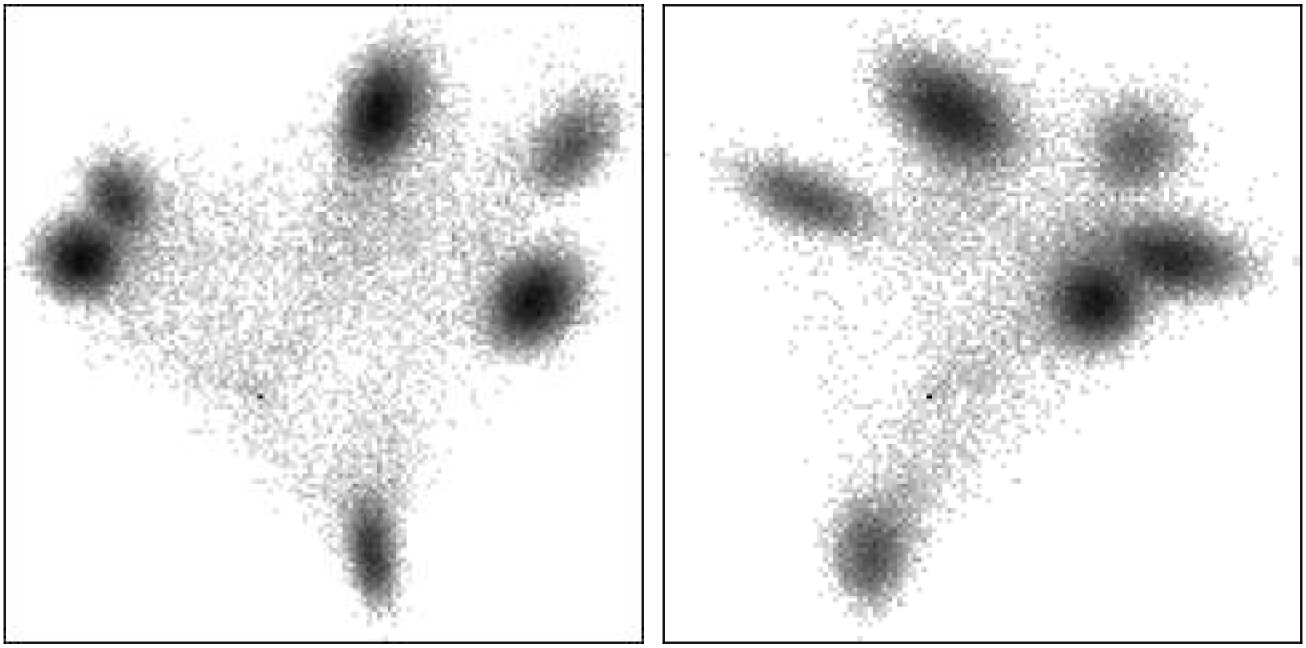


The nine SEM X-ray maps shown in 2.A.3 (Mica_Al, Ca, Fe, K, Na, O, P, Si, Ti) can only be presented in color three-at-a-time (ultimately as noted above because humans have three kinds of cones that respond to different wavelengths, and so our computer displays use three colors). With PCA it is possible to find the most significant axes in this 9-dimensional space to show the distinct compositional phases present. They appear in the display of the three most significant channels as as red, orange, yellow, green, blue and purple regions. The covariance data matrix saved to disk shows the contribution of each elemental map to the composites, and the colocalization plots show the distribution of data points on the principal planes in the PCA axes, in which the presence of the six phase clusters can be easily distinguished.



Channel	Significance	Al	Ca	Fe	K	Mg	Na	O	Si	Ti
1	49.2%	0.238	0.221	-0.601	0.041	0.255	0.112	-0.055	0.634	-0.221
2	20.9%	0.397	-0.681	0.063	0.402	0.161	0.080	0.217	0.174	0.321
3	12.8%	0.067	-0.155	0.563	-0.278	0.040	0.049	0.281	0.371	-0.597
4	7.0%	-0.395	-0.165	0.098	-0.521	0.259	-0.026	-0.148	0.405	0.531
5	4.8%	-0.525	0.143	0.160	0.548	0.588	-0.095	0.101	0.014	-0.111
6	2.7%	0.246	0.622	0.323	0.105	-0.086	0.153	0.419	0.188	0.442
7	1.3%	0.127	-0.015	-0.271	-0.412	0.490	0.028	0.559	-0.432	-0.016
8	0.8%	0.464	0.150	0.322	-0.079	0.494	0.148	-0.592	-0.192	0.002
9	0.6%	-0.241	-0.074	-0.039	0.023	-0.057	0.961	-0.012	-0.083	-0.037

Covariance matrix for the *Mica* dataset



Colocalization plots of channel 1 vs 2 and 3 vs 2 for the *Mica* data
(**IP•Multichannel->Colocalization Plots**)

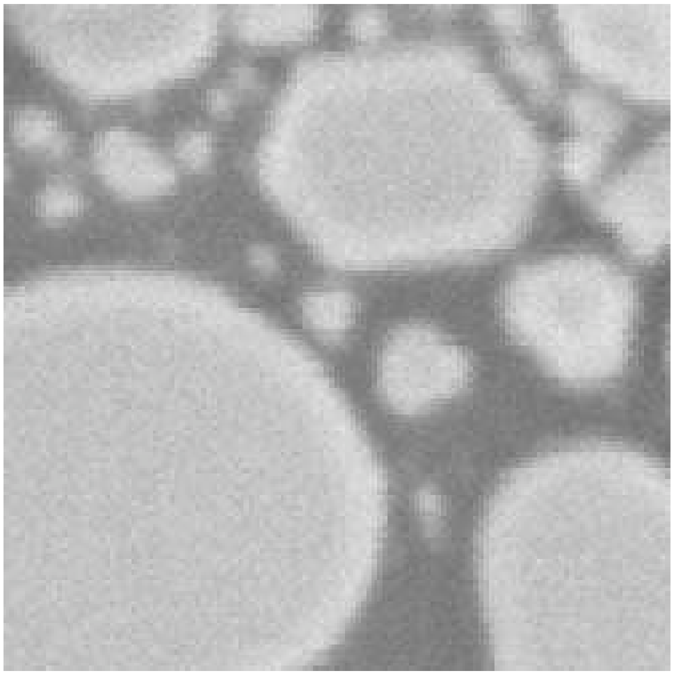
It is also possible to select intensity ranges in the individual channels to display the areas in the original image that they represent (**IP•Multichannel->Channel Thresholding**), or to process some of the channels in the principal components data before reconstructing the original image (**IP•Multichannel->Apply PCA (Inverse)).** This latter technique can sometimes be used effectively to remove artefacts from an image, since that information is often found in the low significance channels after PCA.

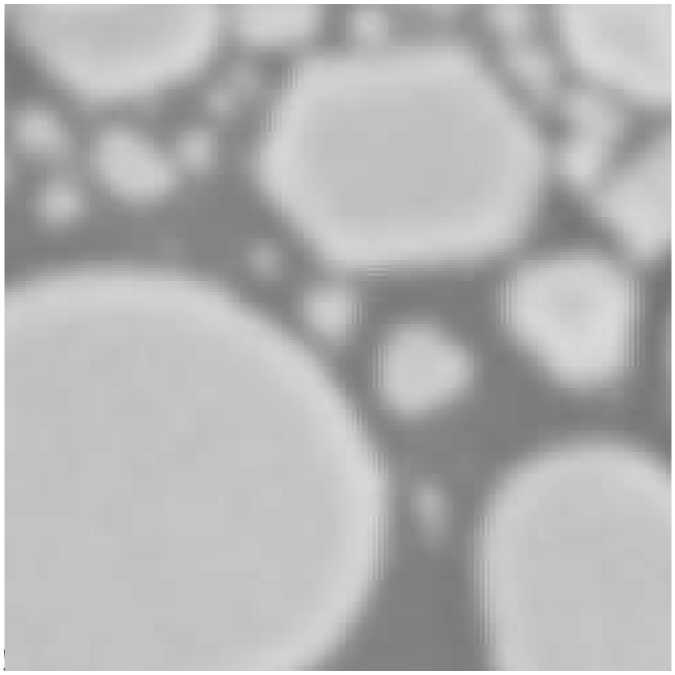
2.B. Noisy images

2.B.1. Random speckle noise

Random (or mostly random) variations in signal values arise from statistical, thermal, electronic and other effects as an image is acquired. It generally appears as a speckle variation in brightness in regions that should be uniform. Random noise that originates from low signal strength (SEM, fluorescence microscopy, etc.) can usually be reduced by collecting more signal (temporal averaging) but this may not be practical. In an existing image, it is commonly removed by spatial averaging with a Gaussian smoothing (an optimum low pass filter) or by median filtering. The latter is almost always preferred, as it retains edge sharpness and position. The neighborhood size (ideally a circular region) controls the size of noise removed (and the processing time). Specialized routines such as the hybrid and conditional median preserve corners and fine lines as well as edges. In color images, the median has a somewhat different meaning but a similar effect.

The example is first processed with a simple averaging filter using an approximately circular neighborhood. This is created using the Photoshop **Filter–>Other–>Custom** function, by entering weights of 1 in the neighborhood of pixels to be averaged (this also serves as an introduction to the creation of kernels of weights, which will be used in other situations that follow). The averaging of the 21 pixels in the neighborhood reduces the speckle noise but blurs the edges.

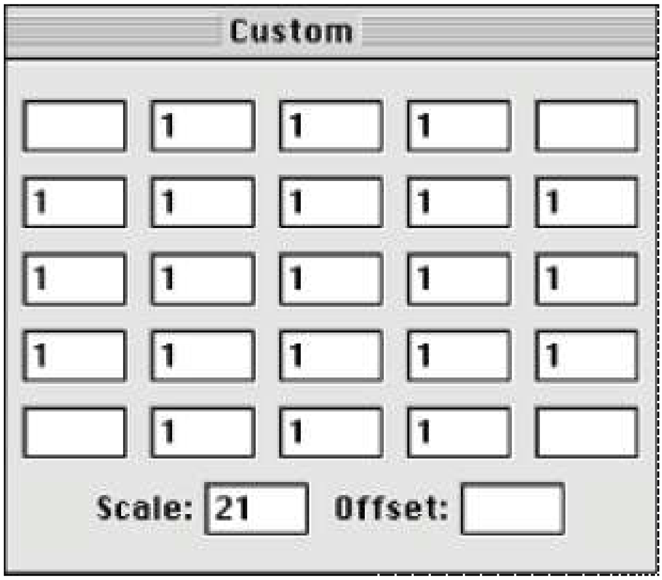




Original *Au_Resn* image (enlarged fragment)"

Averaging filter

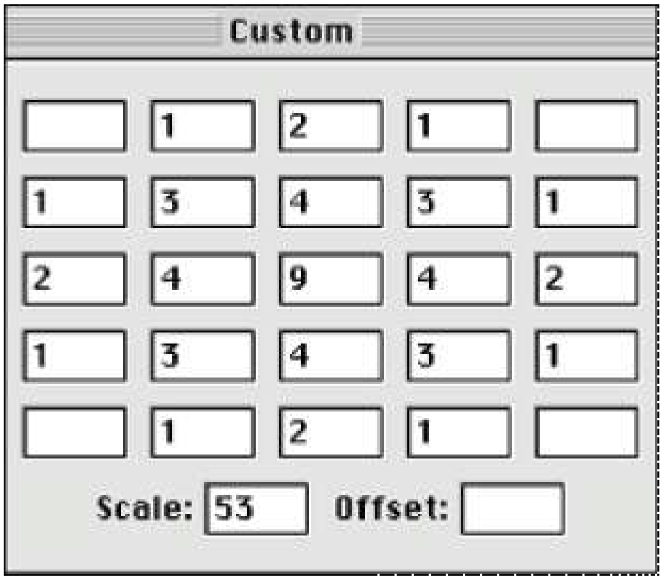
Increasing the value of the weights for the central pixels, as shown in the example, creates a better result, with less blurring of edges for a given amount of noise reduction. The optimum set of weights is Gaussian, meaning that plotting the values would correspond to a Gaussian or bell curve. The integers shown in the example approximate a Gaussian filter with a standard deviation (a measure of the width of the Gaussian peak) of about 0.55 pixels. Larger standard deviations offer greater noise reduction but more blurring of edges. Photoshop offers a Gaussian filter (**Filter–>Blur–>Gaussian Blur**) whose radius can be adjusted and does not require entering the weight factors (and is also faster).



Custom

	1	1	1	
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
	1	1	1	

Scale: 21 Offset:



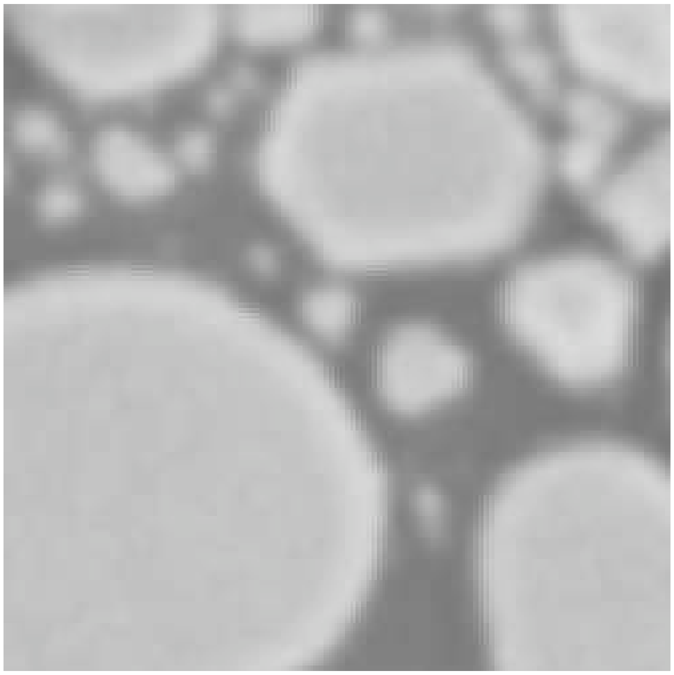
Custom

	1	2	1	
1	3	4	3	1
2	4	9	4	2
1	3	4	3	1
	1	2	1	

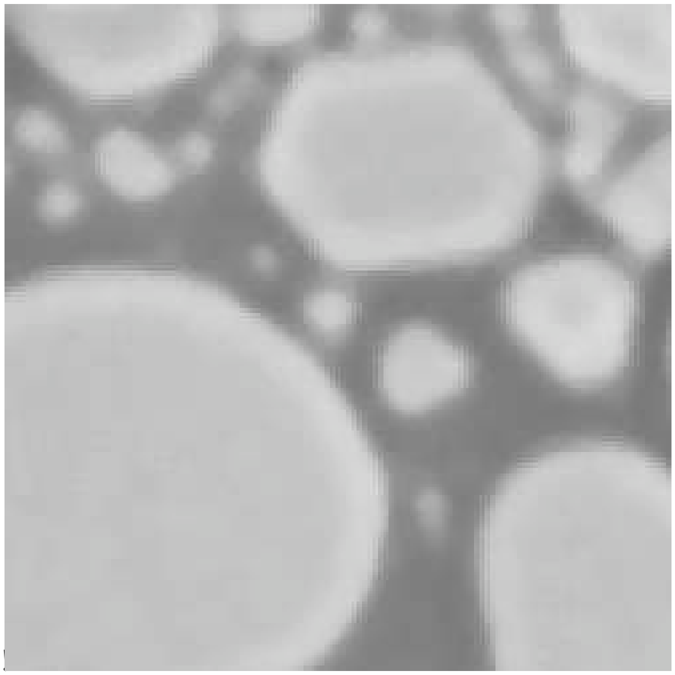
Scale: 53 Offset:

Averaging filter (5-pixel-wide circle)"

Gaussian filter (std. dev. = 0.55 pixels)

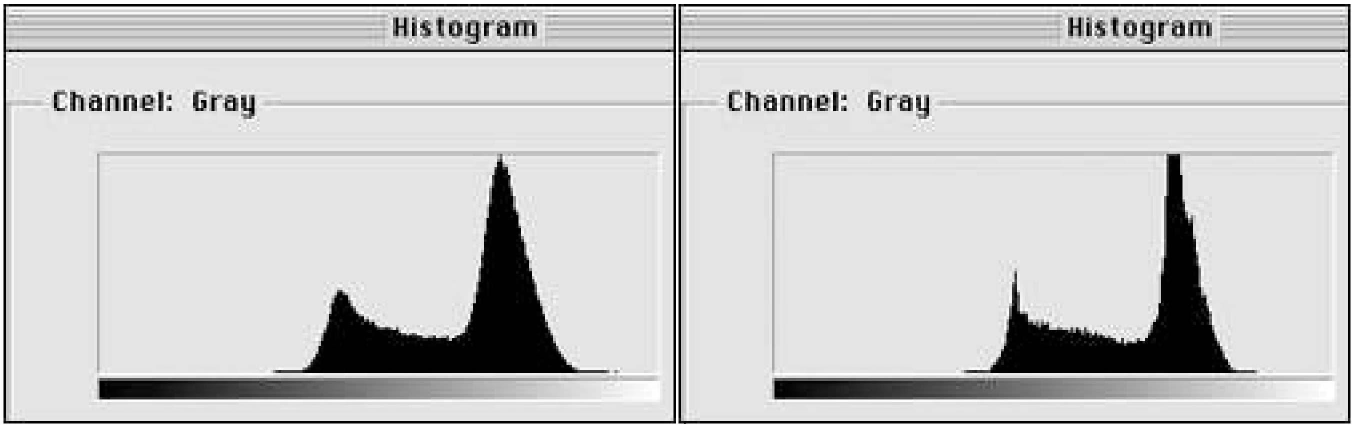


Gaussian filter"



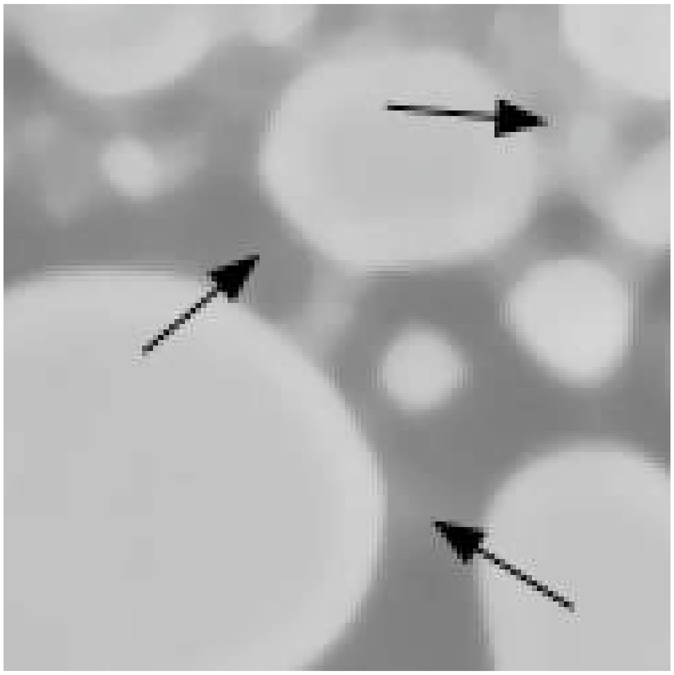
Median filter (radius = 2)

The averaging and Gaussian filters are “low pass” filters that keep the low frequency (gradual brightness variation) components of the image while reducing the high frequencies (abrupt brightness variations). We will see below that they can also be implemented in Fourier space. All low-pass filters produce some degree of blurring of edges, because high frequencies are needed to make them sharp. A different type of filter that can reduce speckle noise with out blurring edges is the median filter (**Filter –> Noise –> Median**). In the example shown, the radius of 2 pixels produces the same 5 pixel width for the neighborhood as used for the low pass filters, but instead of multiplying the pixel values by weights and adding them, they are ranked into order and the middle value in the list (the median) replaces the original central pixel value. Repeating this for every pixel reduces the speckle noise, but does not blur the edge sharpness. All methods of reducing the speckle in the image produce a histogram whose peaks are narrower, which can simplify the process of thresholding (histograms and thresholding are discussed in subsequent sections).

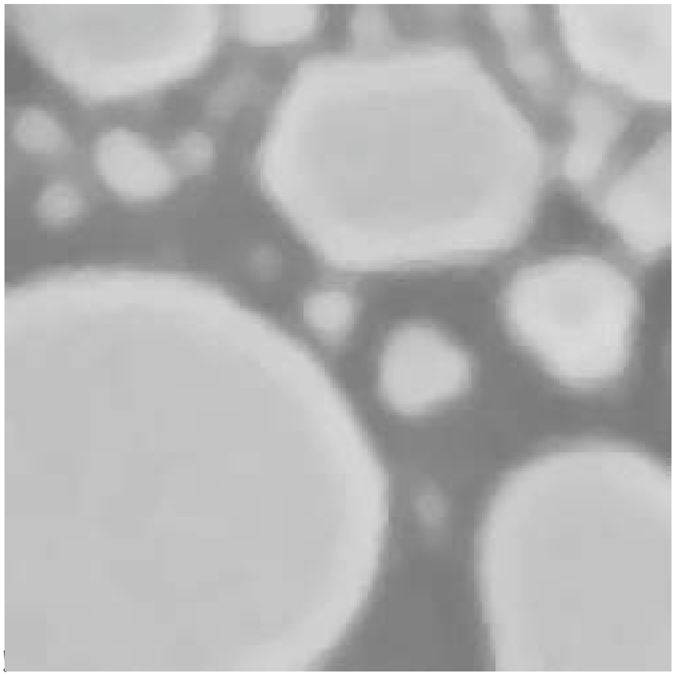


Histogram of original *Au_Resn* image and after a median filter.

The radius of the neighborhood used for the median defines the size of features that are kept in the image and those that are treated as noise and removed. Increasing the radius value to 5, as shown in the example, further reduces the noise but removes any features whose width is less than that radius. Since the median filter does not blur or shift edges, it is possible to repeat the median multiple times with a small neighborhood, which preserves small features while offering additional noise rejection.

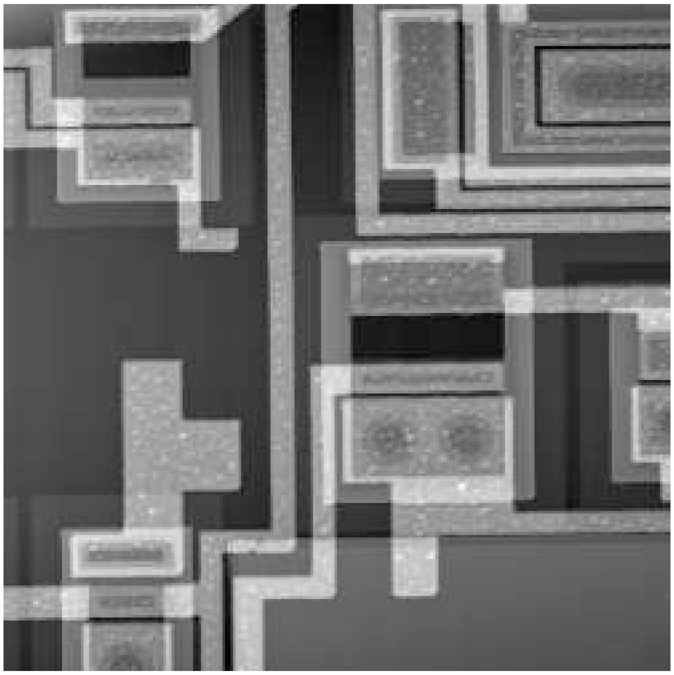


Median (radius 5); arrows indicate features removed."

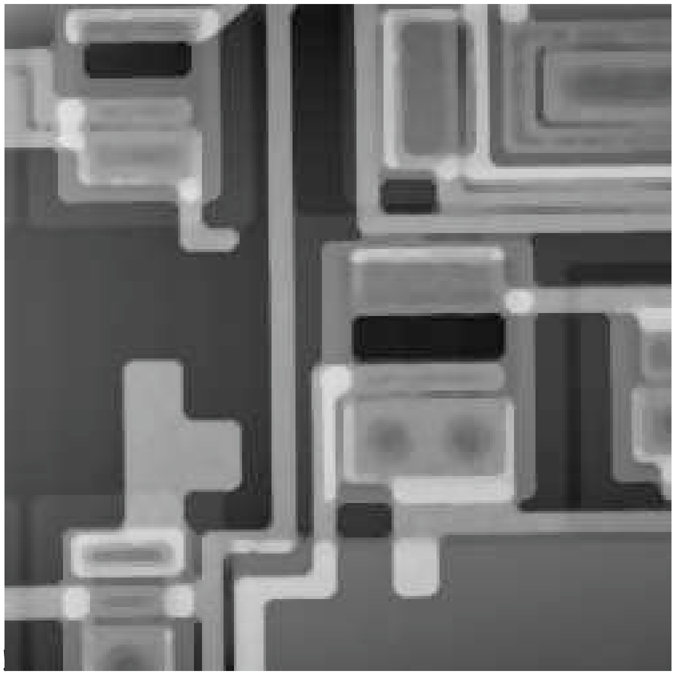


Six repetitions of radius 2 median

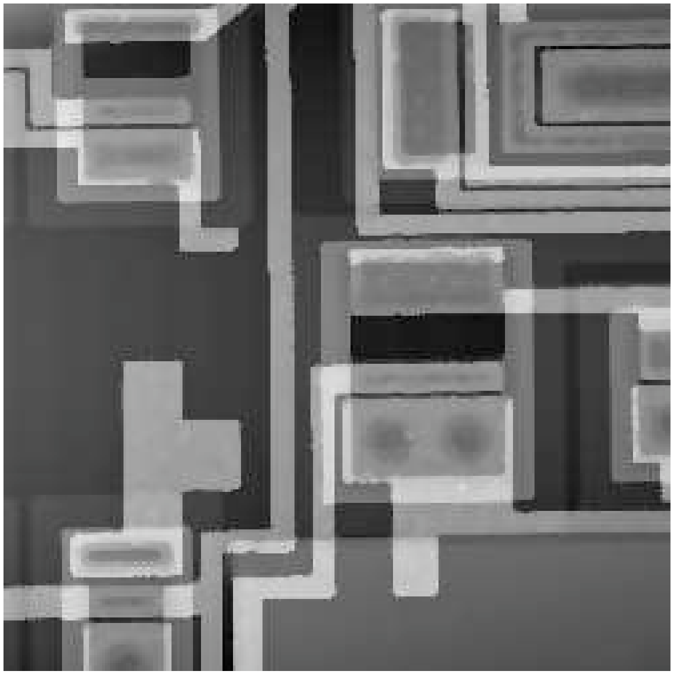
There are several variants of the median filter that preserve fine lines and sharp corners better than the basic method. The hybrid median performs the ranking operation with pixels from different subsets of the neighborhood. For example, in a 3x3 (radius = 1) neighborhood, the pixels located in an "x" pattern are ranked separately from those in a "+" pattern, and then the median results of those rankings and the original central pixel are again combined and ranked to select a median value to replace the original pixel. As shown in the example, these methods can reduce speckle (whether due to electronic effects or small details actually present in the original image) without removing lines or rounding corners (as the conventional median does). In the conditional median, pixels whose value is more than a user-set threshold (50 brightness steps in the example) different from the central one are not included in the ranking, even if they lie within the radius specified for the neighborhood.



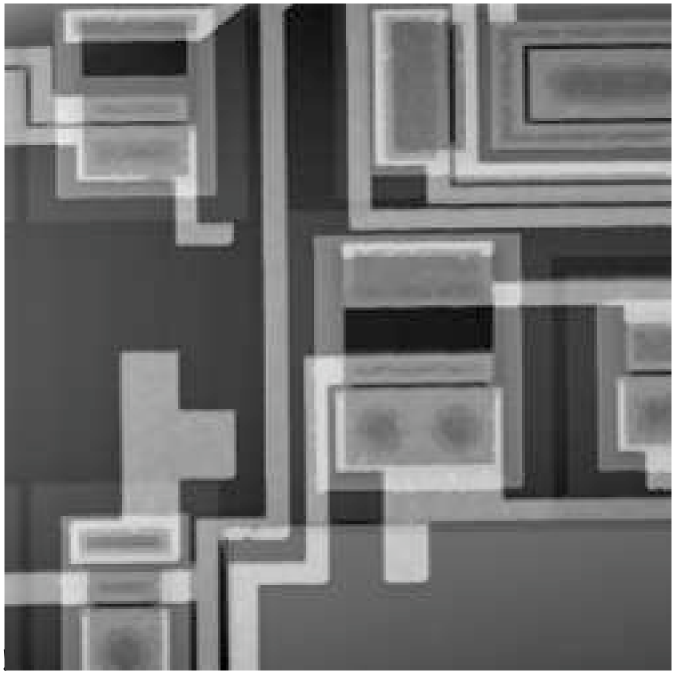
Original *Chip* image"



Conventional median, radius = 2 pixels

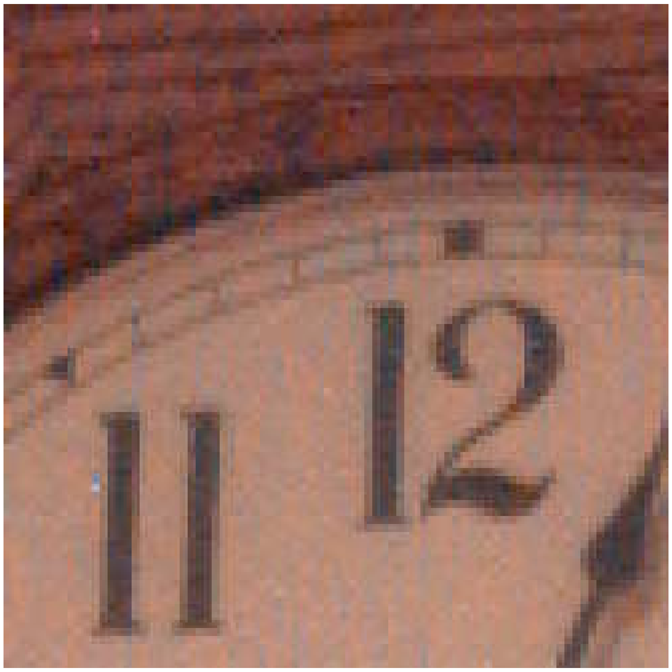


Conditional median (threshold = 50)"



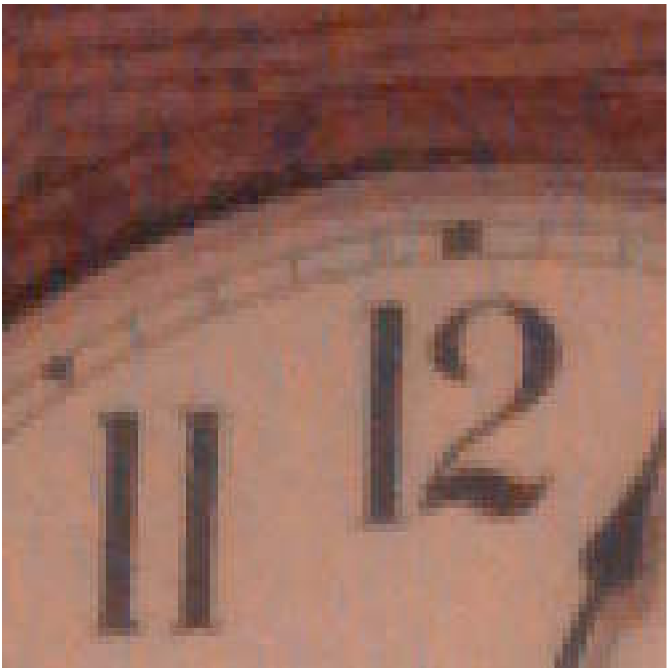
Hybrid median

The median filter ranks the pixels in a neighborhood of a grey scale image by the brightness values. For a color image, some programs (including the built-in Photoshop median) also use the brightness value (the average of the red, green and blue channels). But a true color median is also possible, selecting the colors from the neighborhood pixel that is most central, in a vector sense, to the coordinates of all of the local pixels in color space. The plug-in filters use this more advanced logic.



!

Original *C_503* image (enlarged fragment)"

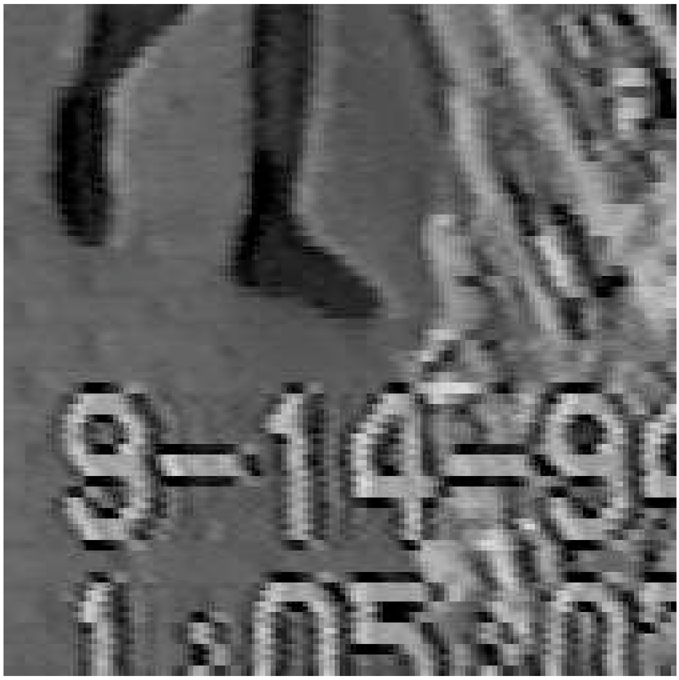


Application of a color median filter

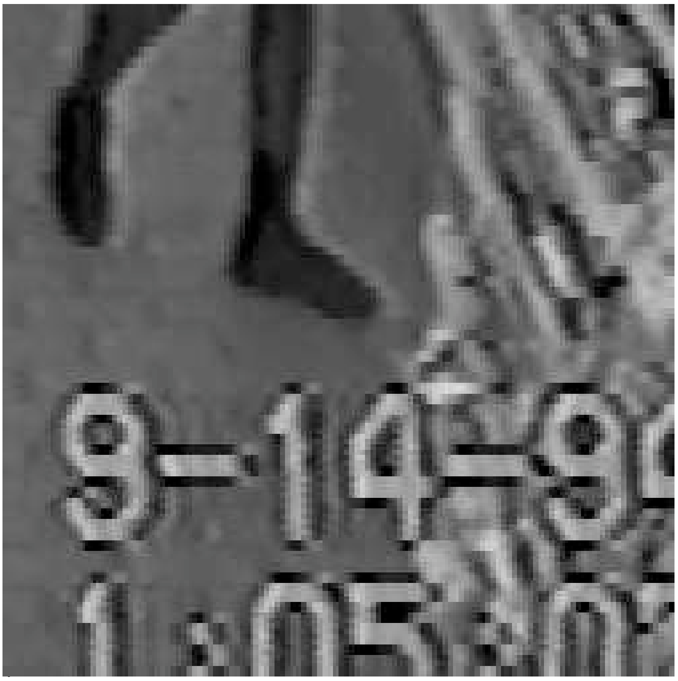
2.B.2. Shot noise and scan line noise removal

In the preceding example, several single-pixel colored dots are visible in the original image. These typically arise in digital cameras because of dead or locked transistors, which because of the color filter arrays produce a dark or bright pixel in one of the color channels. Similar dropout pixels can arise in interference microscopes due to surface slope, or in a photography because of dust on scanned negatives. This type of “shot” noise can be effectively removed with a median filter, which replaces the extreme pixel values with values from the immediate neighborhood.

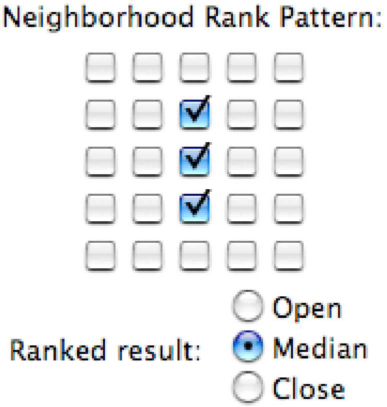
Scratches on negatives, even-odd scan line noise from interlaced video cameras, and some scan line artefacts from AFM, are also image defects that can be effectively corrected using a median filter. For a scratch or line defect, a median filter with a radius greater than the line width will replace the bad values. If the orientation of the linear defect is known (for instance, corresponding to the horizontal direction for video scan lines, or the vertical direction for scratched movie film), then instead of the circular neighborhood used for general median filtering, a custom neighborhood can be tailored to fit using the **IP•Rank –> HitOrMiss** plugin. In the example shown, a neighborhood consisting of each pixel and its neighbors above and below was used to reduce the scan line noise in a surveillance video.



Original *Surveil1* image (enlarged fragment)"



Application of a hit-or-miss median filter



Neighborhood Rank Pattern:

□ □ □ □ □
□ □ ☑ □ □
□ □ ☑ □ □
□ □ ☑ □ □
□ □ □ □ □

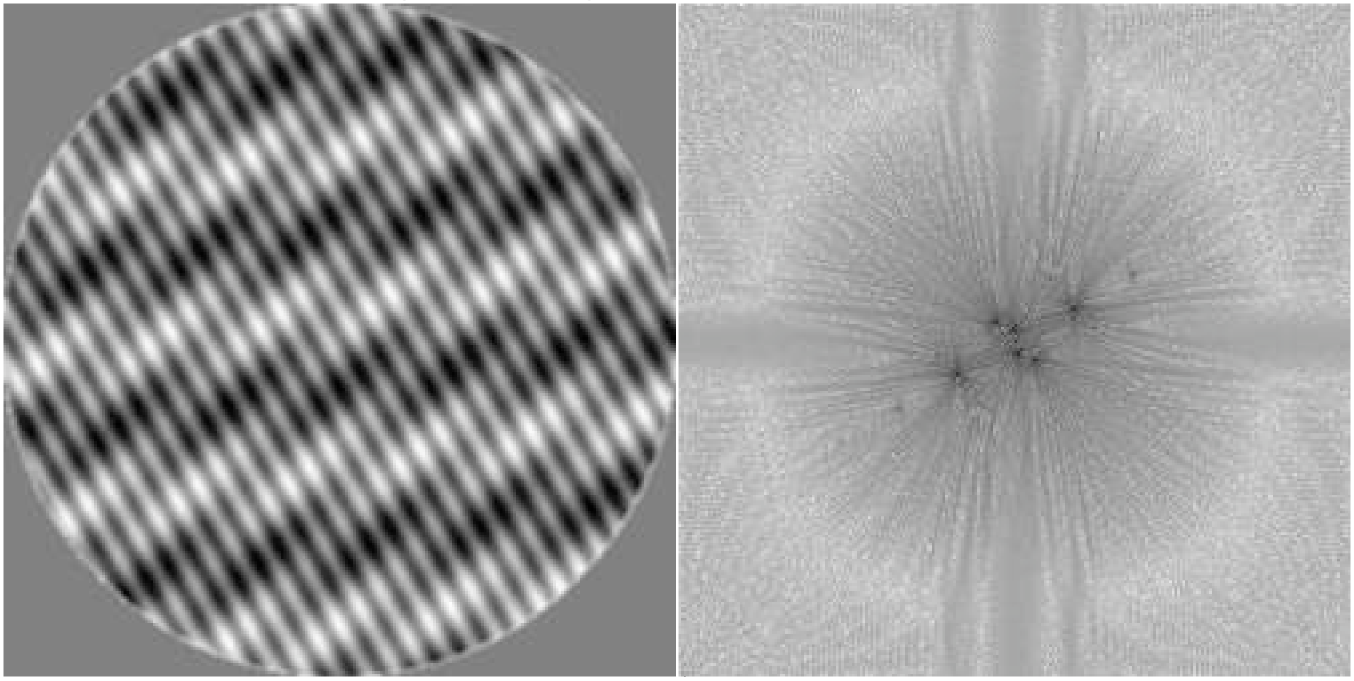
Ranked result: ○ Open ● Median ○ Close

Defining the hit-or-miss neighborhood for the median filter

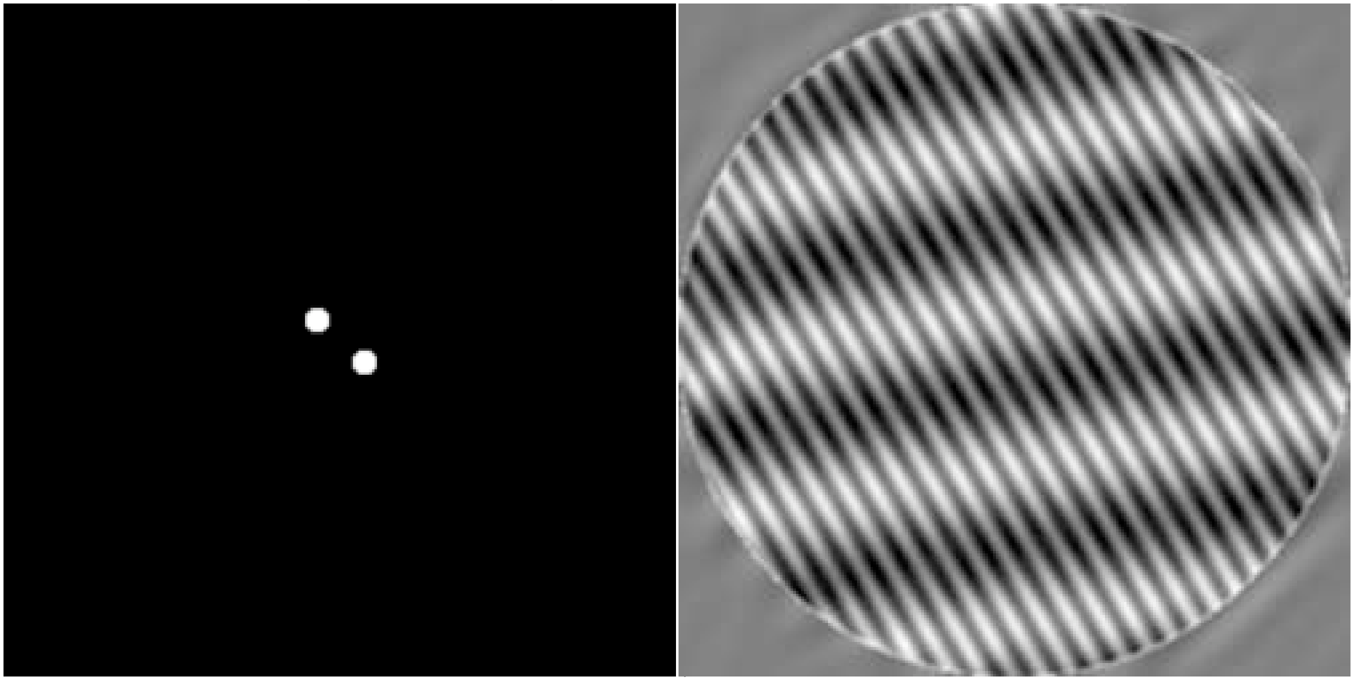
2.B.3. Periodic noise removal

Electronic interference, vibration, halftone and moiré patterns superimposed on images are another form of noise. This non-random or periodic noise is most efficiently removed in Fourier transform space, where the periodic information spread throughout the image is represented by a single point for each frequency and orientation. The relationship between the spatial or pixel domain and the Fourier transform power spectrum is shown in the illustration below. The original image was created by superimposing three sets of sinusoidal lines. The power spectrum shows three corresponding points (each plotted twice, symmetrically). Removing one point or "spike" with a filter removes just the corresponding lines, leaving the others intact. In the displayed power spectrum, the location of a spike represents the orientation angle and frequency of the lines, and the darkness of the point is proportional to the log of the amplitude.

The procedure shown is to transform the image with **IP•Fourier–>FFT (Forward)**, manually mark with the pencil tool a black spot that covers the dark spike in the power spectrum display (it is only necessary to mark one of the two corresponding points, and then select **IP•Fourier–>Generate Symmetric Filter** to mark the second). Next, threshold this to create a filter or mask (the words are often used interchangeably for this purpose), invert the mask (**Image–>Adjustments–>Invert**) so that it will erase the spikes and preserve all of the other values, and then use **IP•Fourier–>Apply Filter and FFT (Inverse)** to transform the data without the selected frequency and orientation back to the spatial domain. Automatic methods for locating and removing the spikes will be introduced below.



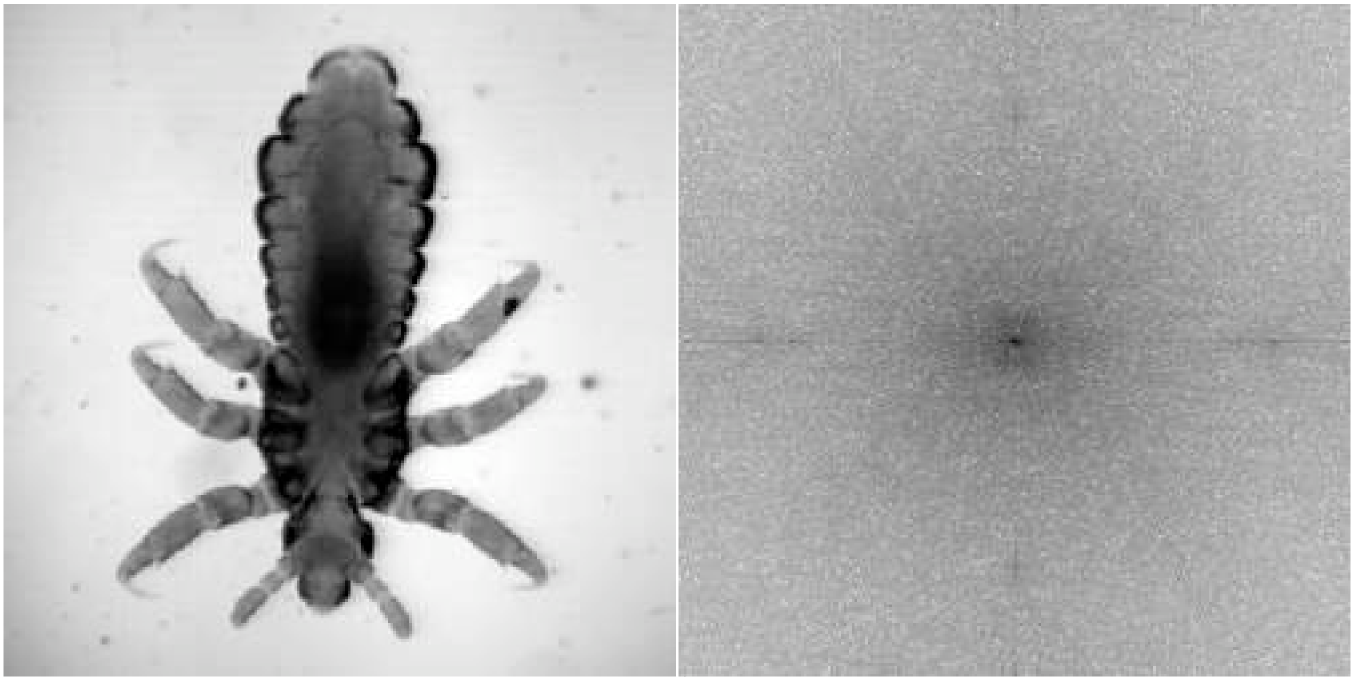
Original *ThreeSin* image and its Fourier transform power spectrum



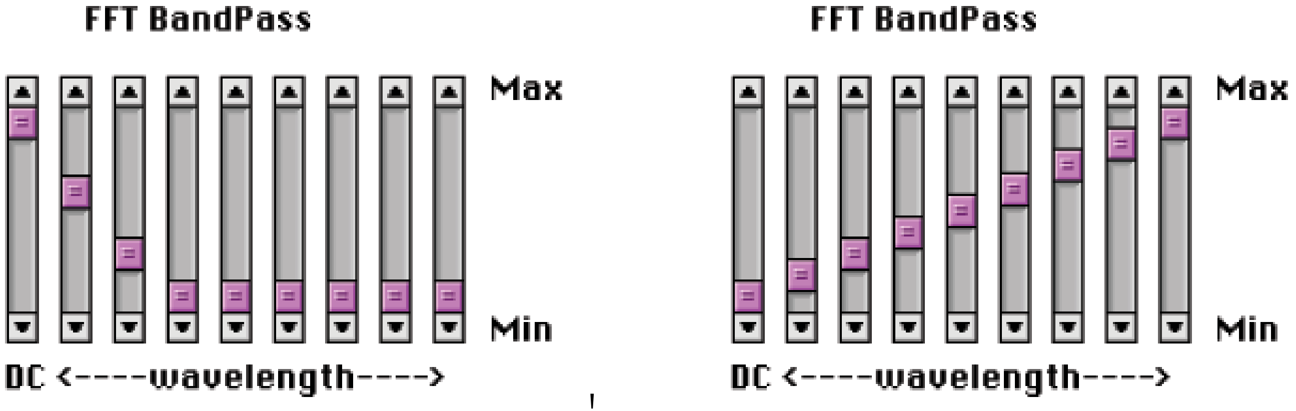
The mask used to remove one of the three sets of lines, and the result of applying it and performing the inverse FFT transform.

Note that the Fourier transform implementation in the plug-ins requires that the image have dimensions that are an exact power of 2 (64, 128, 256, 512, 1024...). If your image is not that size, you may either use a rectangular marquee with dimensions set to those values to process a portion of the image, or use the **Image–>Canvas Size** tool to surround the image with padding to the next larger size.

Fourier transforms are not limited in usefulness to images with periodic noise. Processing in Fourier space can be used to implement high- and low-pass filters (the low-pass filter was introduced above in the discussion of noise removal) by removing or emphasizing selected frequency ranges. In the example shown, reducing the high frequencies produces a blurred result identical to a Gaussian filter, while emphasizing them produces a result like the Laplacian and sharpening filters that will be introduced below. The **IP•Fourier–>Generate Bandpass Filter** plug-in was used to create filters which were then applied just as in the example above (the sliders may be interpreted in much the same way as the equalizer in a music system). In many cases it is more efficient to implement such filters with Fourier transforms even though they may be described in terms of kernels of weights applied to the pixel image.

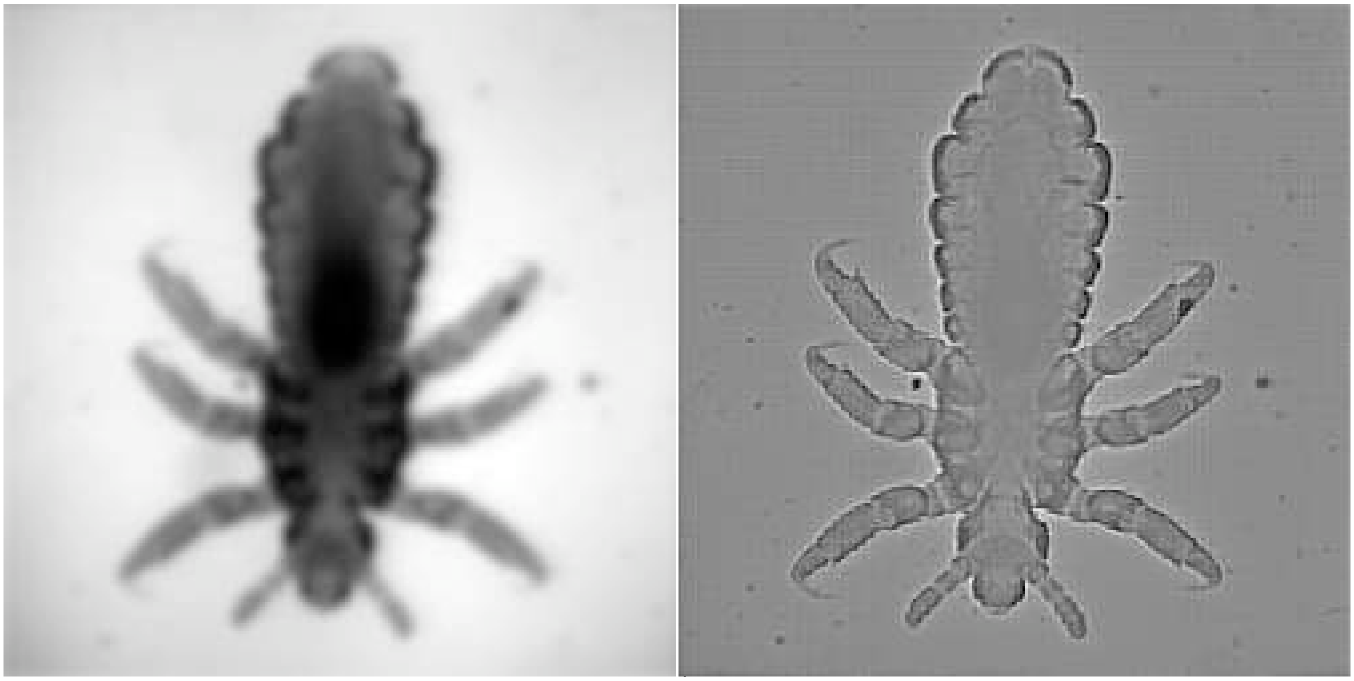


Original *Bug* image and its Fourier transform.



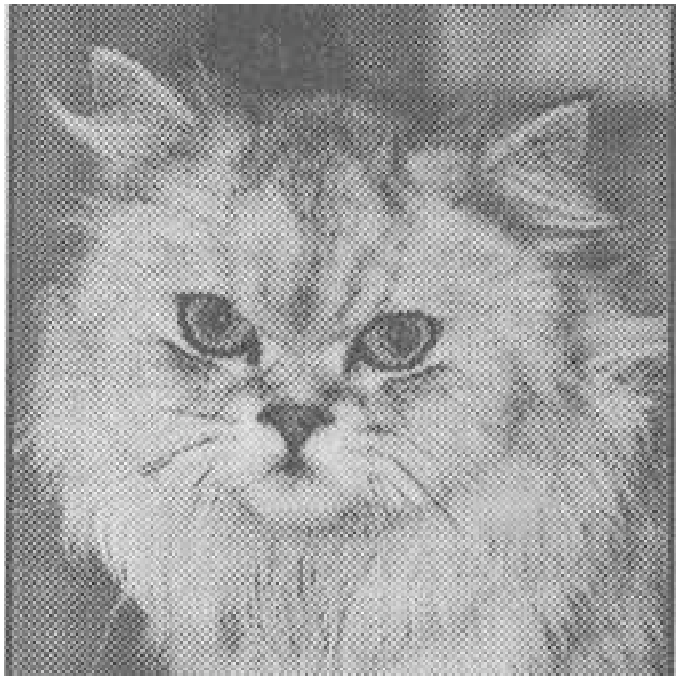
!

Using the **IP•Fourier–>Generate Bandpass** plug-in to create low and high pass filters, respectively.

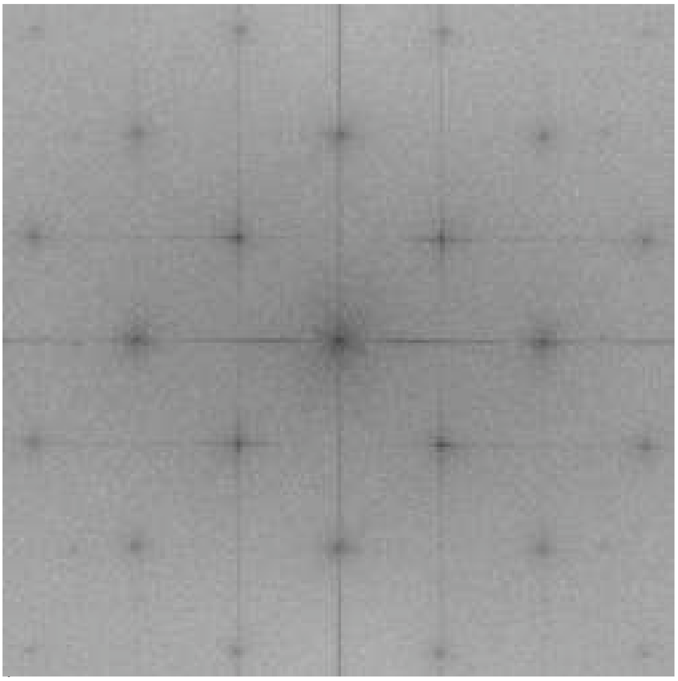


Results of applying the low- and high-pass filters to the image.

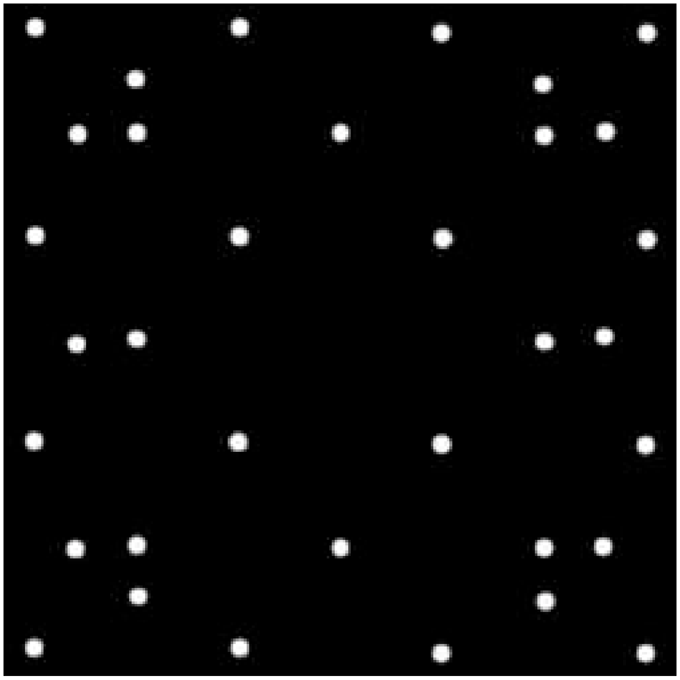
The next example combines both of these types of filters to remove the halftone pattern. The original image (scanned from a newspaper) consists of discrete black dots, which create a visual impression of greyscale because their size varies. The Fourier transform power spectrum shows a regular array of spikes that represent the orientation and spacing of the sinusoids that combine to create the dot array. It would be possible to manually mark these spots as was done above, or to mark just two of them and use the **IP•Fourier–>Generate Harmonics** function, but in this case an automatic method using the top hat filter (described in a following section) was used to locate the spikes and create the mask. Applying the mask to the stored Fourier transform (**IP•Fourier–>Apply Filter**) removes the spikes. In order to fill in the spaces between the original halftone dots, a low pass filter is needed. In this example a Butterworth filter was generated (**IP•Fourier–>Generate Filters**) and applied to the stored transform. Performing the inverse transform with the **IP•Fourier–>FFT (Inverse)** function produces a result that reveals fine details that are present in the original but visually obscured by the halftone printing process.



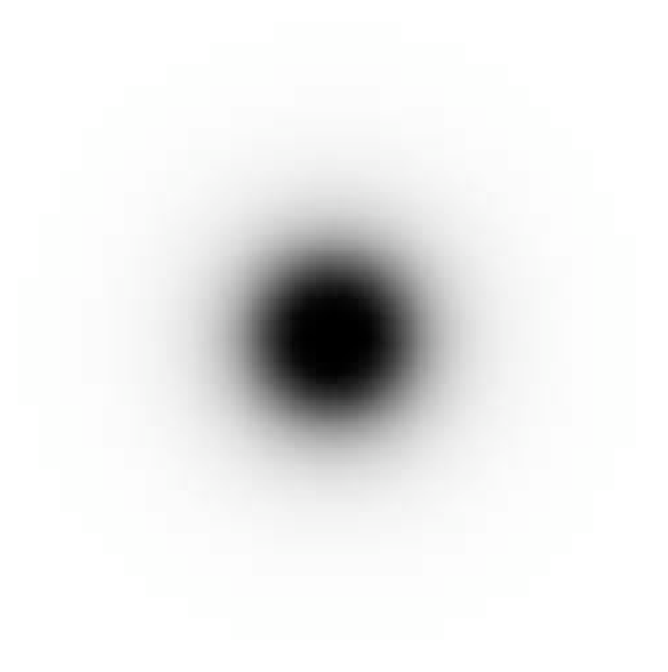
Original *Halftone* image"



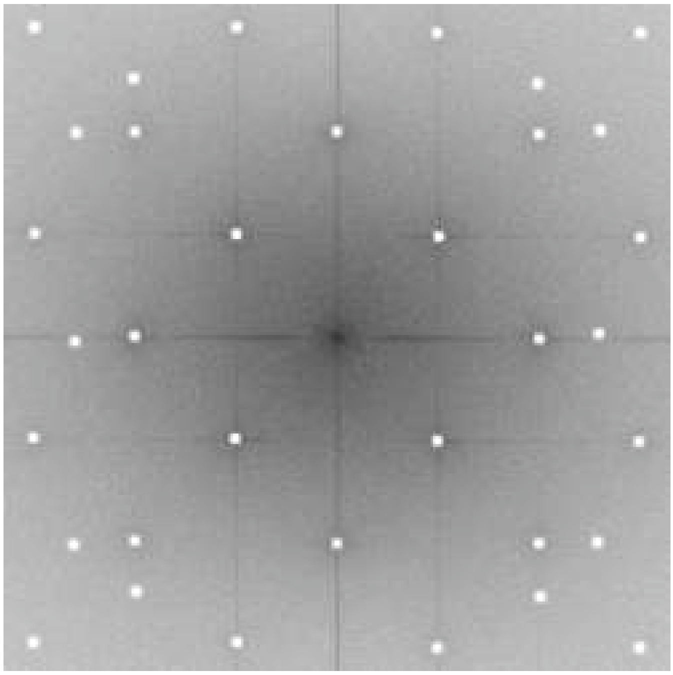
Fourier transform showing array of spikes

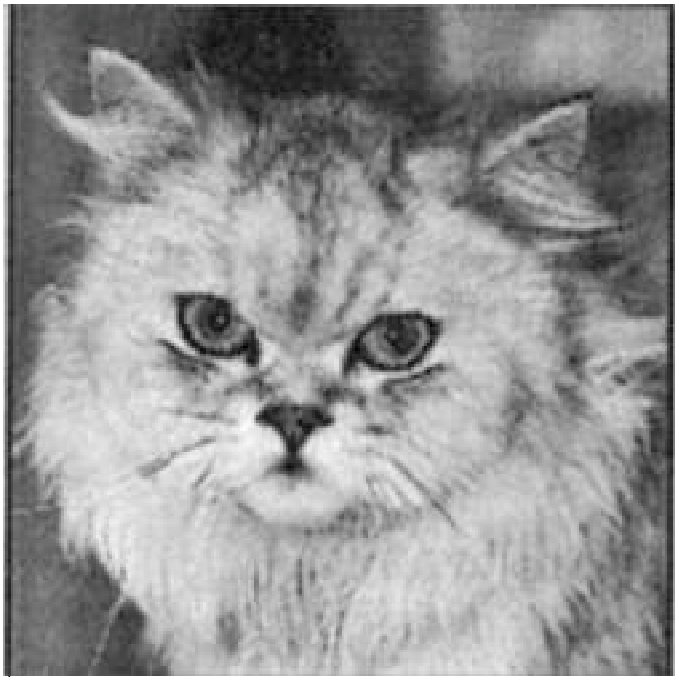


Mask to eliminate spikes (*Halft_1* image)"



Butterworth low-pass filter (cutoff = 0.4, *Halft_2* image)

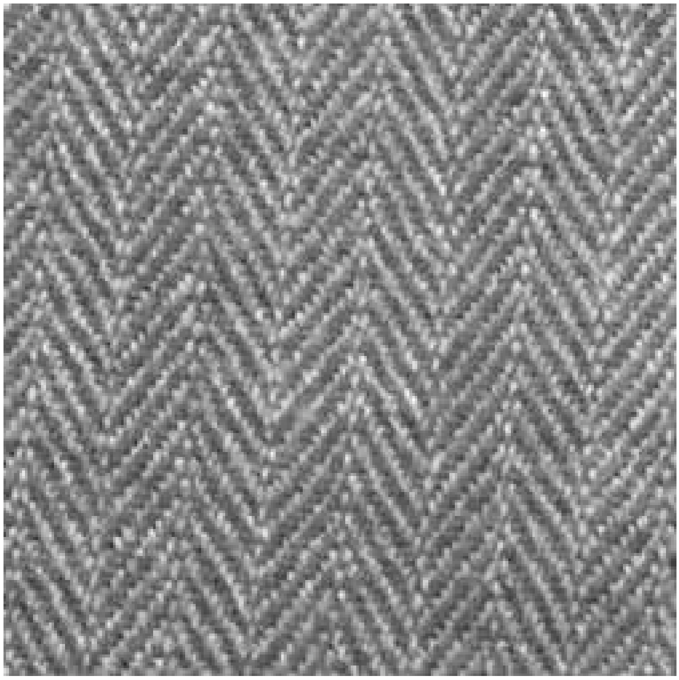


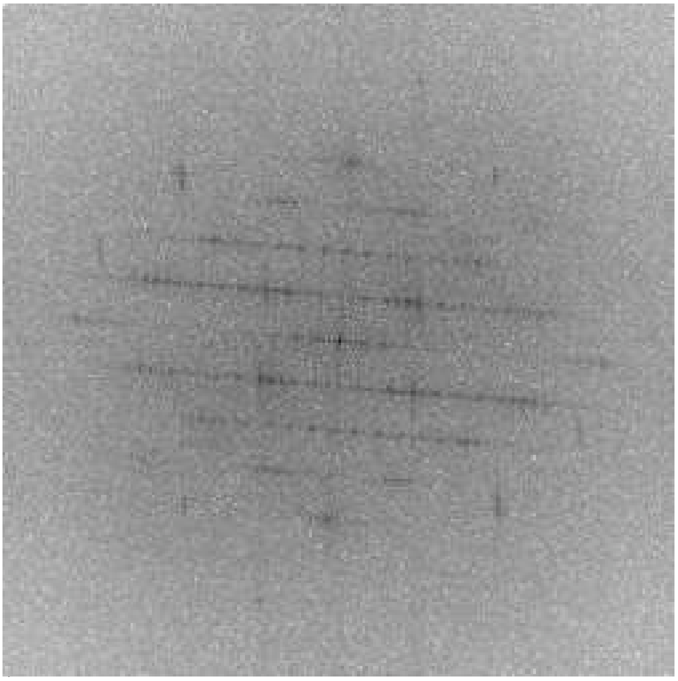


FFT power spectrum after filters are applied" Result after inverse transform

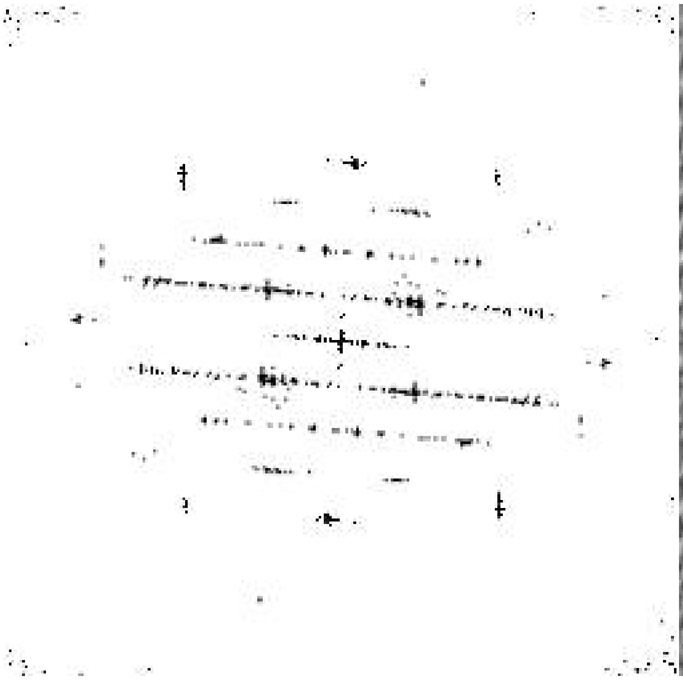
If this technique is applied to a color halftone image, it is necessary to process each color channel separately (usually in CMYK mode, since those are the inks used for color printing) because the halftone patterns used for each channel have different orientations.

For images in which the structure is periodic, with superimposed random noise, the same methods can be used except that the filter or mask is inverted to remove everything except the periodic information. In the example, the lines of spikes in the Fourier transform are used to make a mask by leveling the image of the power spectrum (discussed below), followed by thresholding. Then **IP•Fourier –> Apply Filter and FFT (Inverse)** clarifies the periodic structure of the woven cloth and suppresses the random variations.





Original *Herringbone* image" Fourier transform



Filter produced by leveling and thresholding"

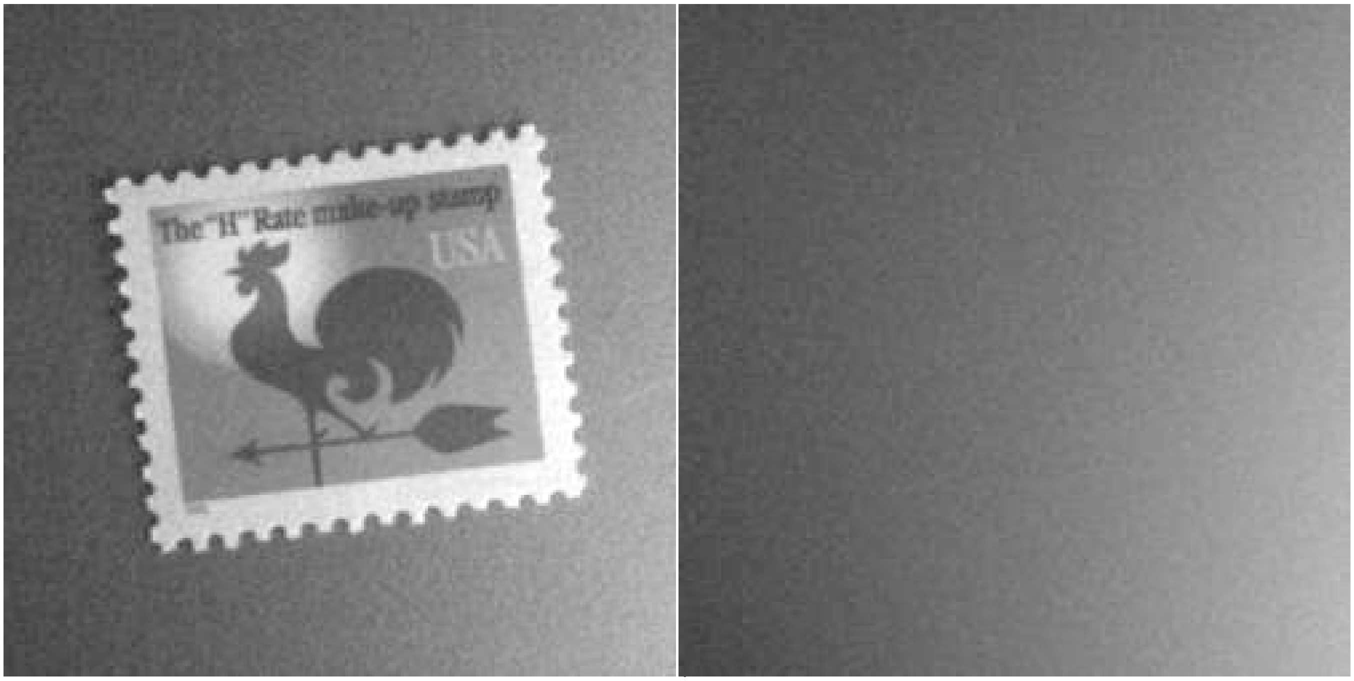
Inverse transform showing periodic structure

2.C. Nonuniform image illumination

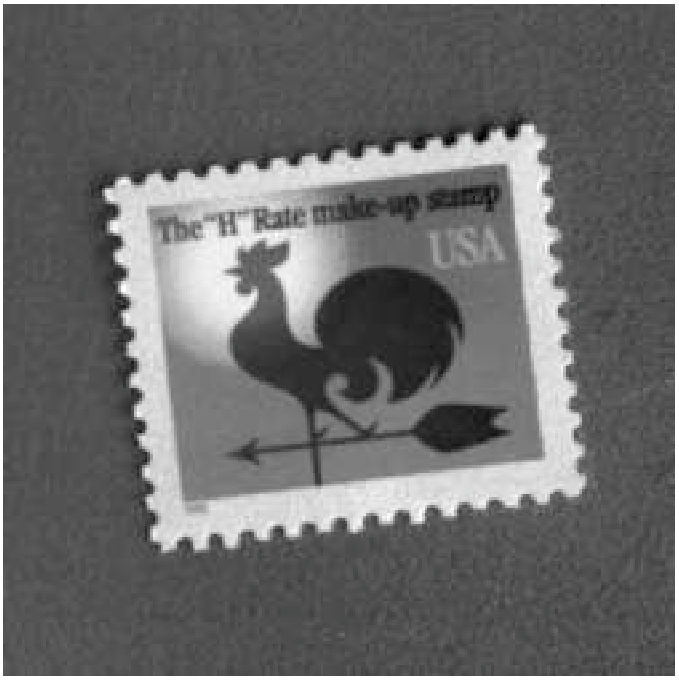
2.C.1. Is a separate background image available?

Nonuniform lighting, optical vignetting, fixed patterns in the camera response, or other factors can cause image brightness or contrast to vary from side to side or center to edge, so that the same feature would appear to have different brightness (or color) depending on where it was located. If these factors remain constant over time, one practical solution is to capture a separate background image using identical settings but with no sample present (such as a photograph of a neutral grey card on a copy stand, a blank slide in a light microscope, or a clean stub in the SEM). This image can then be used to remove the nonuniformities from acquired images by either subtraction or division. The choice of subtraction or division depends on whether the camera response in logarithmic (subtraction) or linear (division). Photographic film, video cameras, and some digital cameras are logarithmic in output vs. light intensity, while scanners and CCD detectors are inherently linear (division). Sometimes the only good way is to try both and see which produces a flat response.

In the example shown, a background image was acquired with the same lighting conditions as when the sample was present, and then subtracted to remove the nonuniformity due to placement of the lamps. The subtraction procedure is to place the background image into the second image buffer (**IP•2nd Image –> Setup**), select the image of interest, and then choose **IP•Math –> Subtract**.



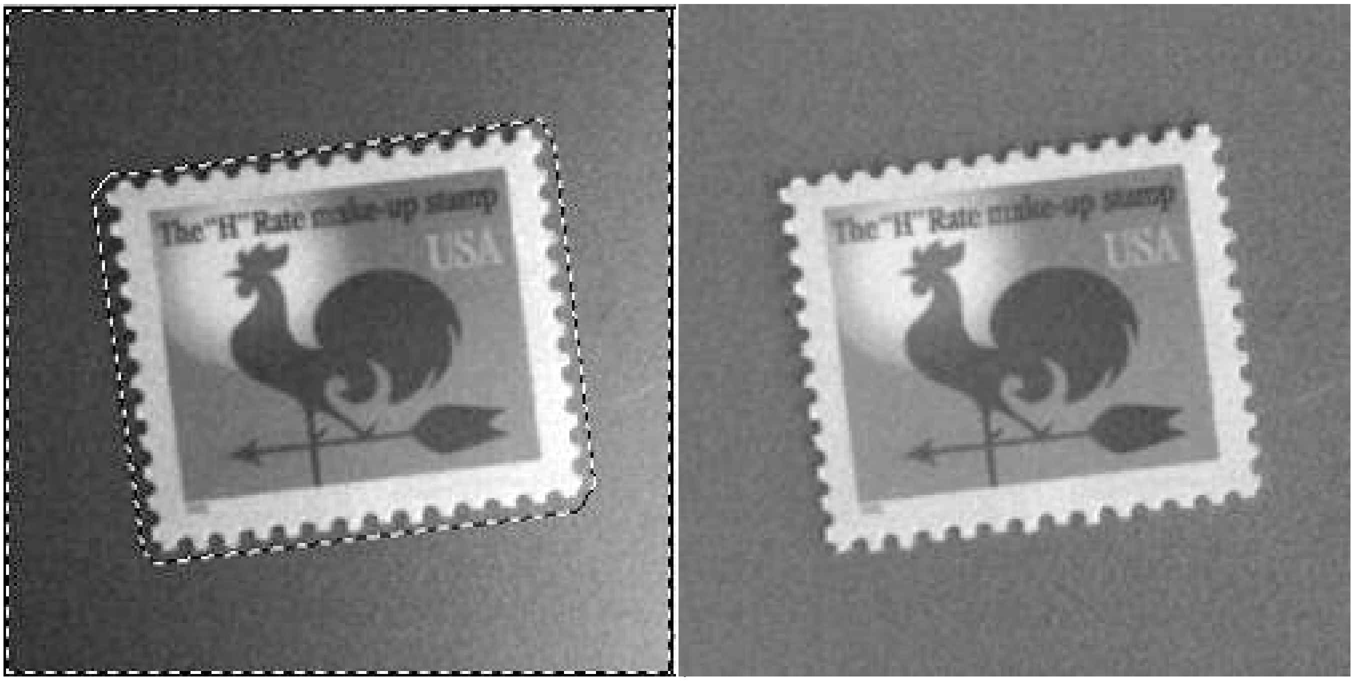
Original image (*Lighting1*) and the corresponding background with the feature removed (*Lighting2*)



Subtracted (leveled) result.

Unfortunately, in many cases a suitable background image can not be (or was not) acquired, or there are effects due to the specimen itself (variations in thickness or density, surface curvature, etc.) that cause nonuniformities in the image. In these cases, several other methods are available. Sometimes there are enough areas of a uniform background available in the image to construct a background mathematically. One method for doing this is to manually select the background areas (for instance with the Photoshop marquee or lasso tools, or the wand tool) and then choose **IP•Adjust–>Background Fitting**. That uses all of the points in the selected area (which should be distributed across the image, not all in one corner!) to fit a polynomial function. Then removing the selection or selecting the entire image and choosing

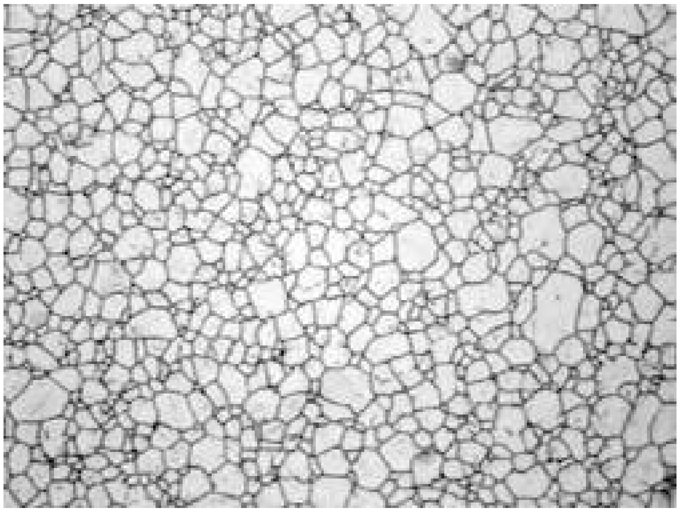
IP•Adjust–>Background Removal subtracts the constructed polynomial from the image. The polynomial is remembered and will be used again until a new one is established.



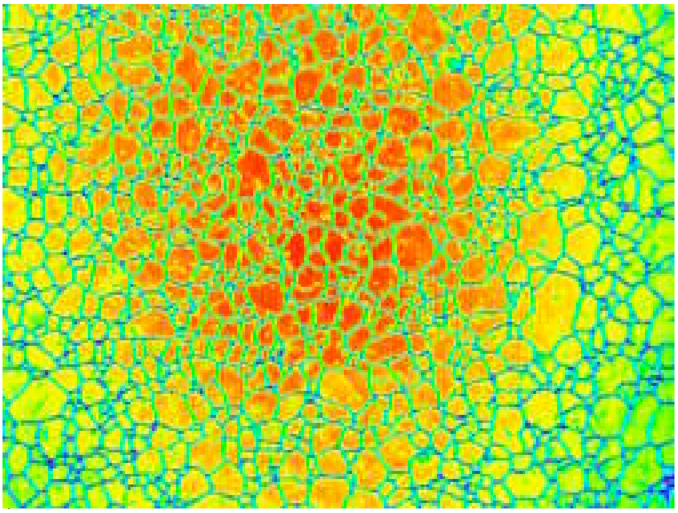
Selecting the background areas to construct a polynomial background, and the result of removing it.

2.C.2. Is background visible throughout the image?

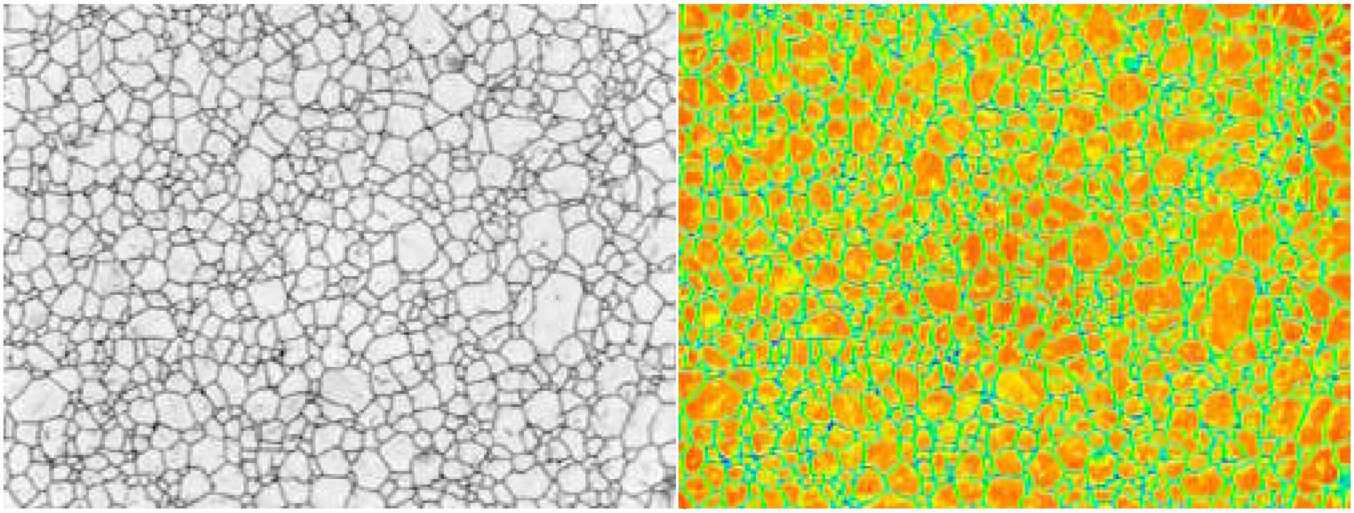
Leveling based on fitting a polynomial can be applied automatically if the background regions are either lighter or darker than the features (and of course provided that representative background patches are well distributed across the image). The **IP•Adjust–>AutoLevel** functions divide the image up into a grid and find the brightest (or darkest) pixel values in each segment, and then construct the polynomial and subtract it. This is a very rapid and effective tool in many cases. In the example shown, the presence of the nonuniform lighting (due to optical vignetting) is made more visually evident by using false color or pseudo color to assign a rainbow of colors to the grey scale values (convert the image mode to RGB color and use the **IP•Color–>Apply Color Table** plug-in to load the desired color scale).



Original *Gr_Steel* image with vignetting"



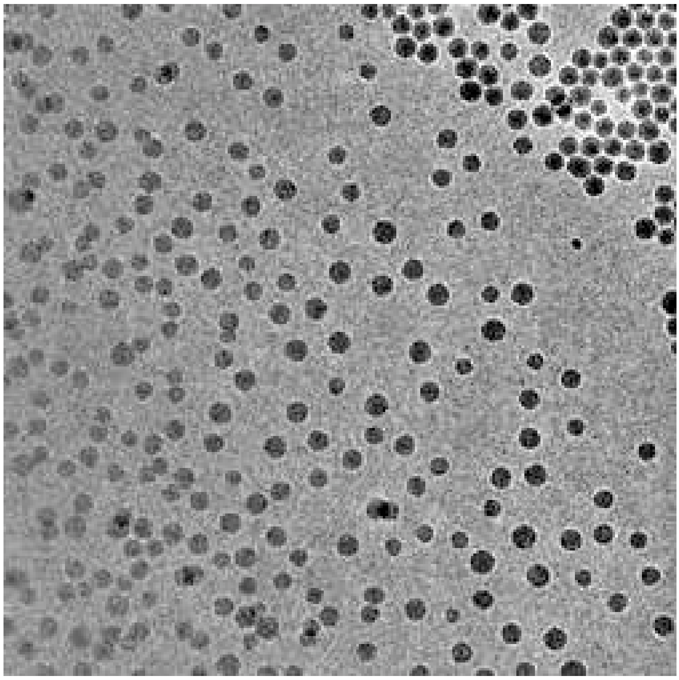
False color used to show brightness variation



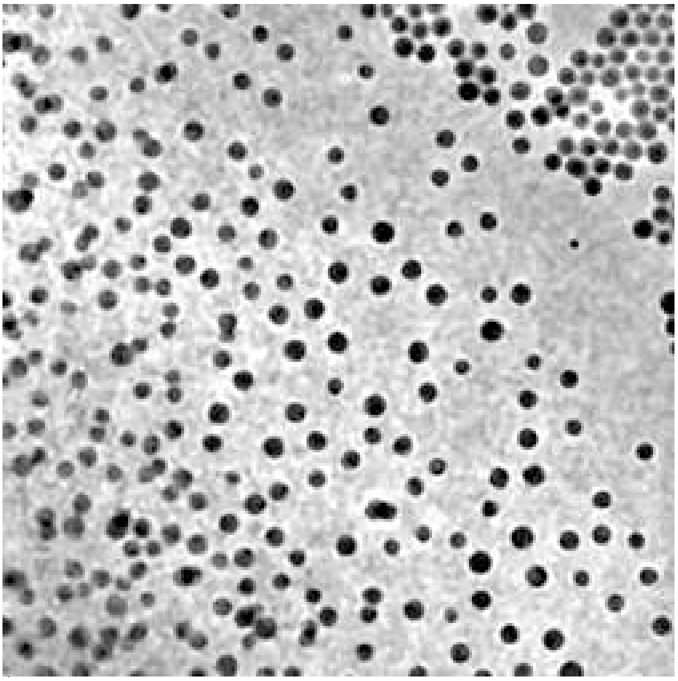
The same image after automatic leveling to make all of the bright values uniform

2.C.3. Correcting varying contrast across an image

The polynomial method can be extended to fit both the brightest and darkest values across the image in order to compensate for variations in contrast, for example due to thickness changes in samples viewed in transmission. In the example, it was necessary to first reduce the speckle noise using a median filter before using **IP•Adjust–>AutoLevel** plug-in to level the contrast in the image. The local contrast is stretched linearly between the brightest and darkest curves.



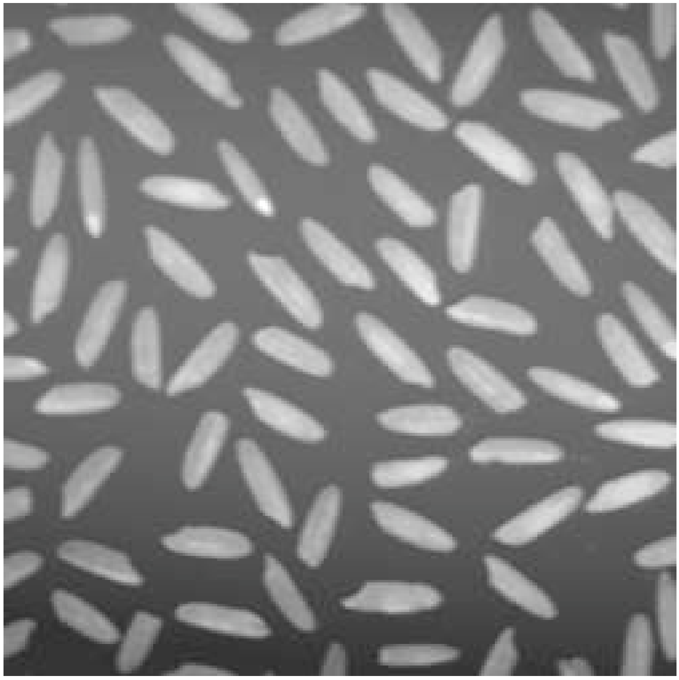
Original *Shading1* image"



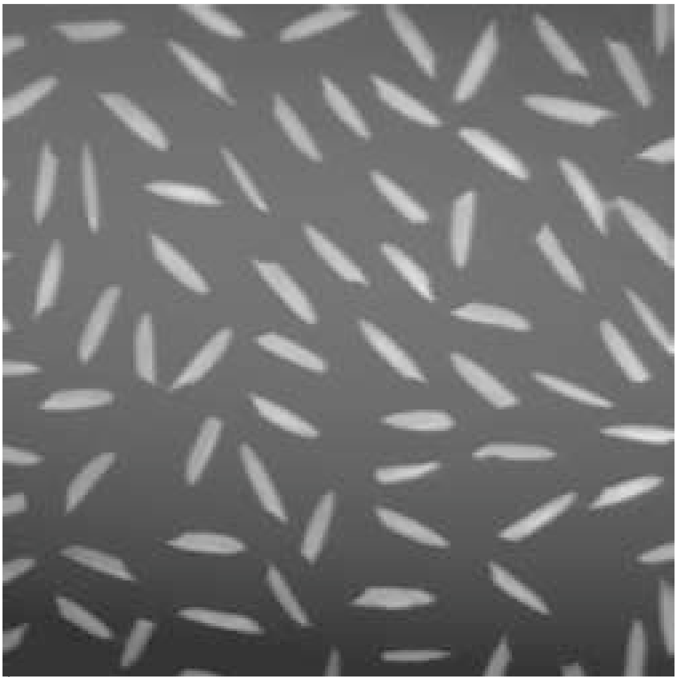
After median filter and autoleveling contrast

2.C.4. Are the features small in one dimension?

The polynomial leveling method works well when lighting or vignetting cause a gradual variation of brightness with position. An irregular pattern of variation, for instance due to surface geometry or local density variations, can be more effectively corrected using rank-based leveling to remove the features and leave just the background, for subsequent subtraction or division. The median filter introduced above as a noise removal technique is one example of rank filtering. Instead of replacing each pixel with the median of the values in a small neighborhood, it is also possible to choose the brightest or darkest value. Section 5.A. on morphology illustrates several ways that erosion, dilation, openings and closings are applied to images (a more general discussion of morphological processing is presented in a later section). One of those methods operates on grey scale images. In the first example shown, replacing each pixel with its darkest neighbor (grey scale dilation) removes the features to produce a background which is then subtracted to level the overall contrast. If the features had been dark on a light background, they would have been removed with a grey scale erosion. These operations are selected in the **IP•Rank–>Grey Scale Morphology** dialog.

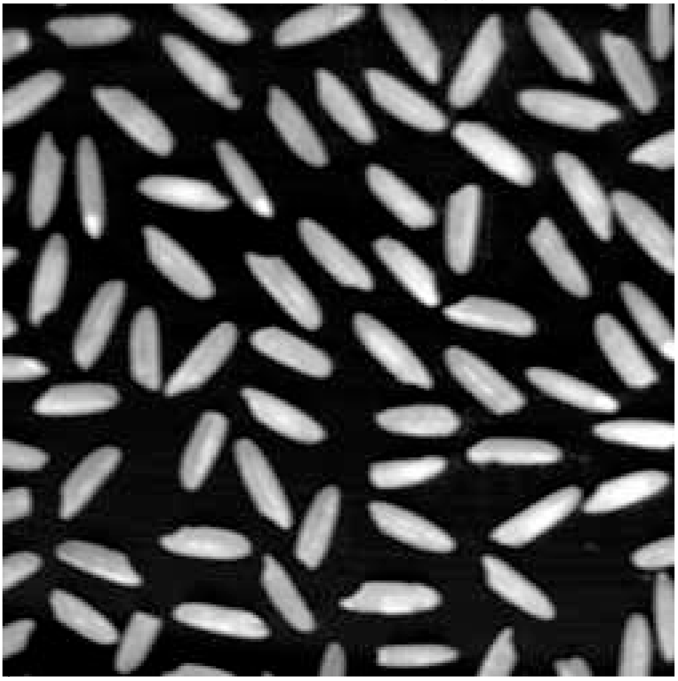


Original *Rice* image"



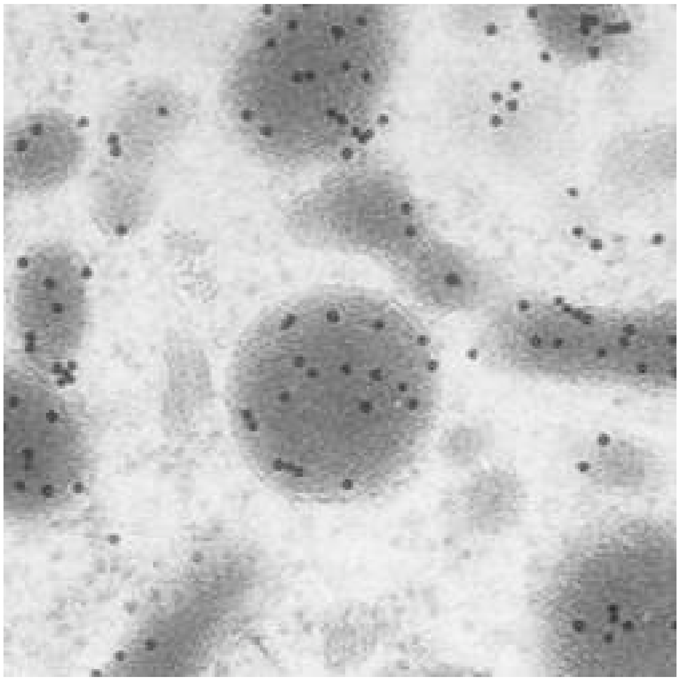
One iteration of grey scale dilation

After four iterations the rice grains are removed"

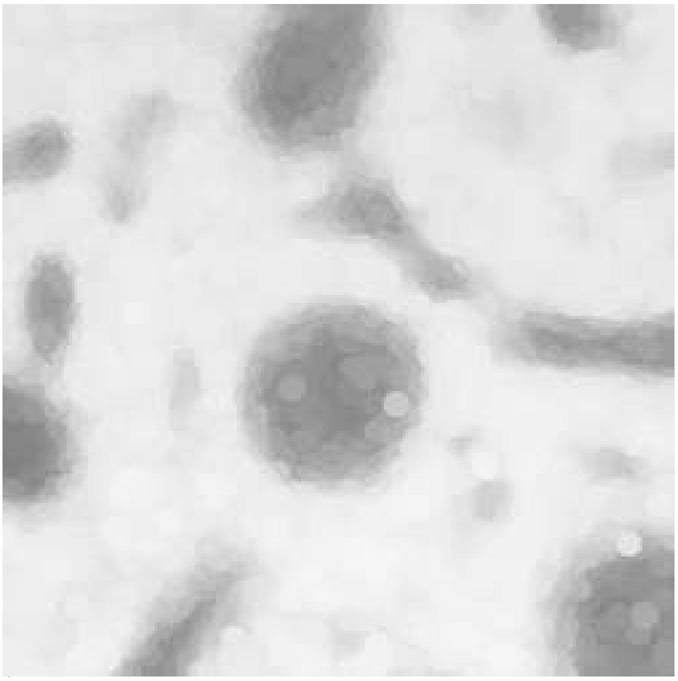


Subtracting the background from the original

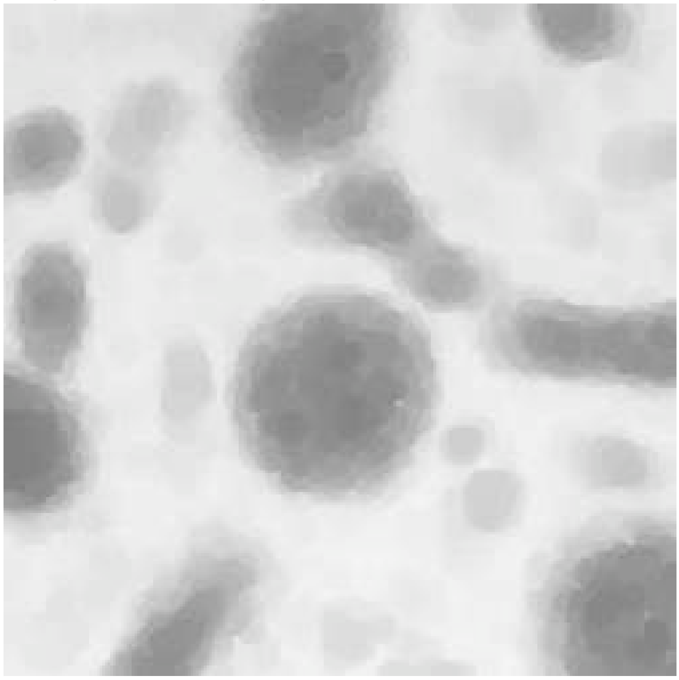
This method is made more general by combining erosions and dilations to preserve the sizes of the background structures. In the next example, in which the background density varies because of stained organelles, the small dark gold particles are removed by grey scale erosion but this also shrinks the organelles. Following two iterations of erosion, two iterations of dilation are used to restore the organelle size, and then the resulting background is divided into the original to leave just an image of the gold particles. The sequence of erosion followed by dilation is called an opening. The opposite sequence of dilation followed by erosion is called a closing. Either can be selected as single step operations in the **IP•Rank–>Grey Scale Morphology** dialog.



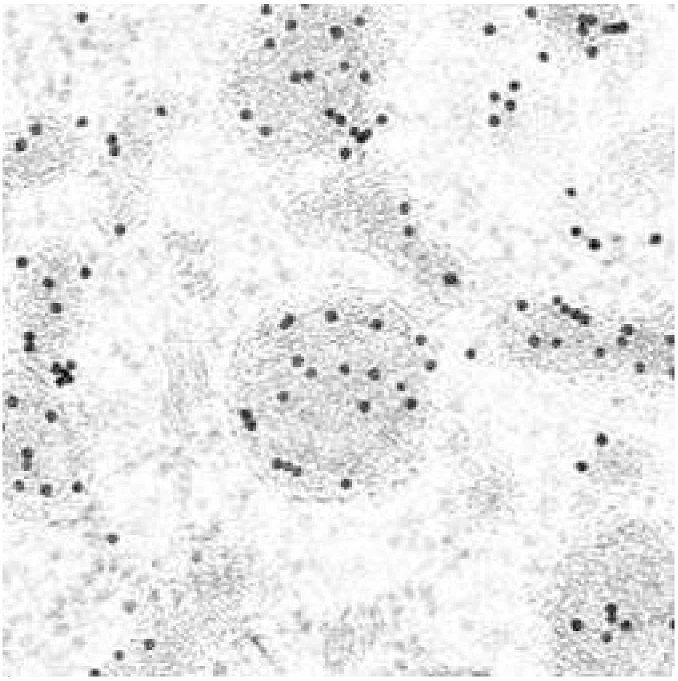
Original *Gold2* image"



After two iterations of erosion

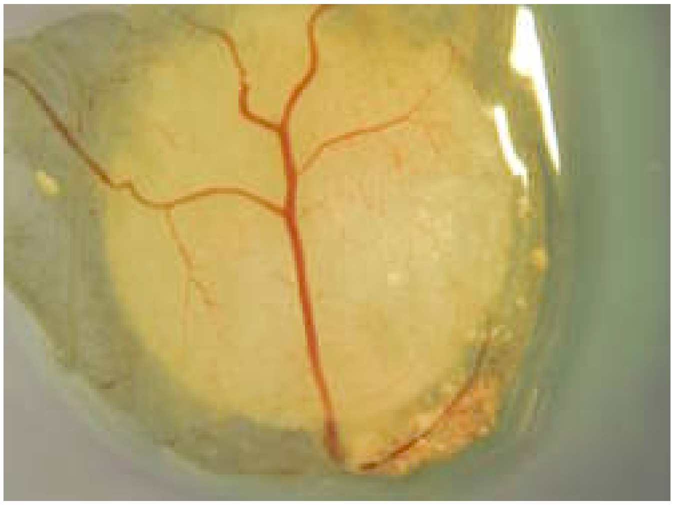


After two iterations of dilation (opening)"

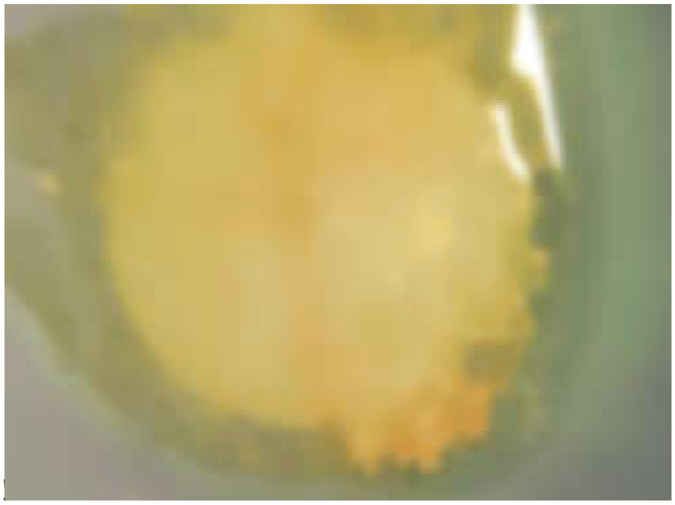


Dividing the background into the original

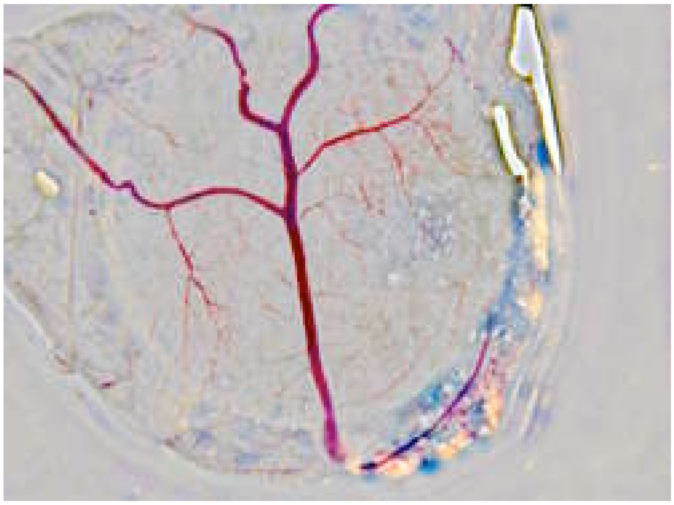
This technique can also be applied to color images. The **IP•Rank–>Color Morphology** routine performs erosion, dilations, openings and closings based on color. Select the color (in the example shown, the red of the blood vessels) with the eyedropper tool, so that it becomes the foreground color in the tool palette. Then select the plug-in. In the example, an opening (erosion followed by dilation) of the red color was used to remove the blood vessels, and the resulting background was then subtracted from the original to remove the variation in brightness and color leaving just the blood vessels. It is sometimes useful to blur the background produced by the morphology routines (e.g., using a Gaussian blur) before subtracting or dividing, to eliminate contours (this was done in the example shown).



Original *Yolk* image"



After opening and Gaussian blur

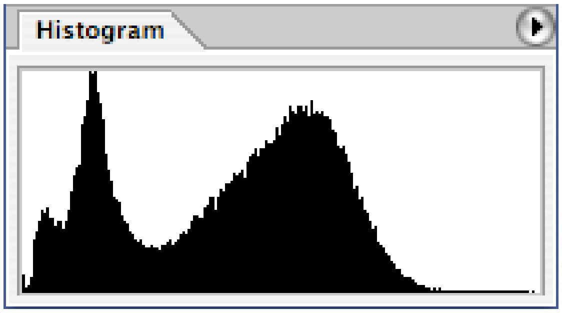


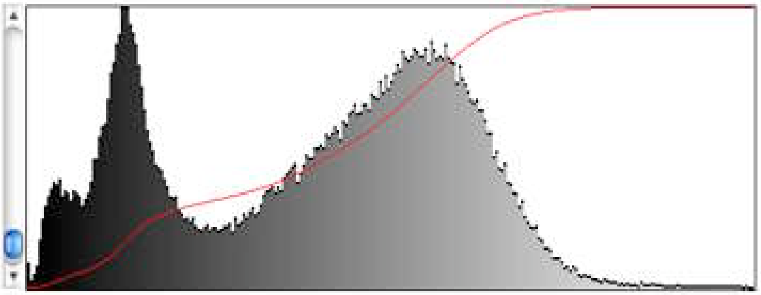
After subtraction of the background

2.D. Expanding image contrast

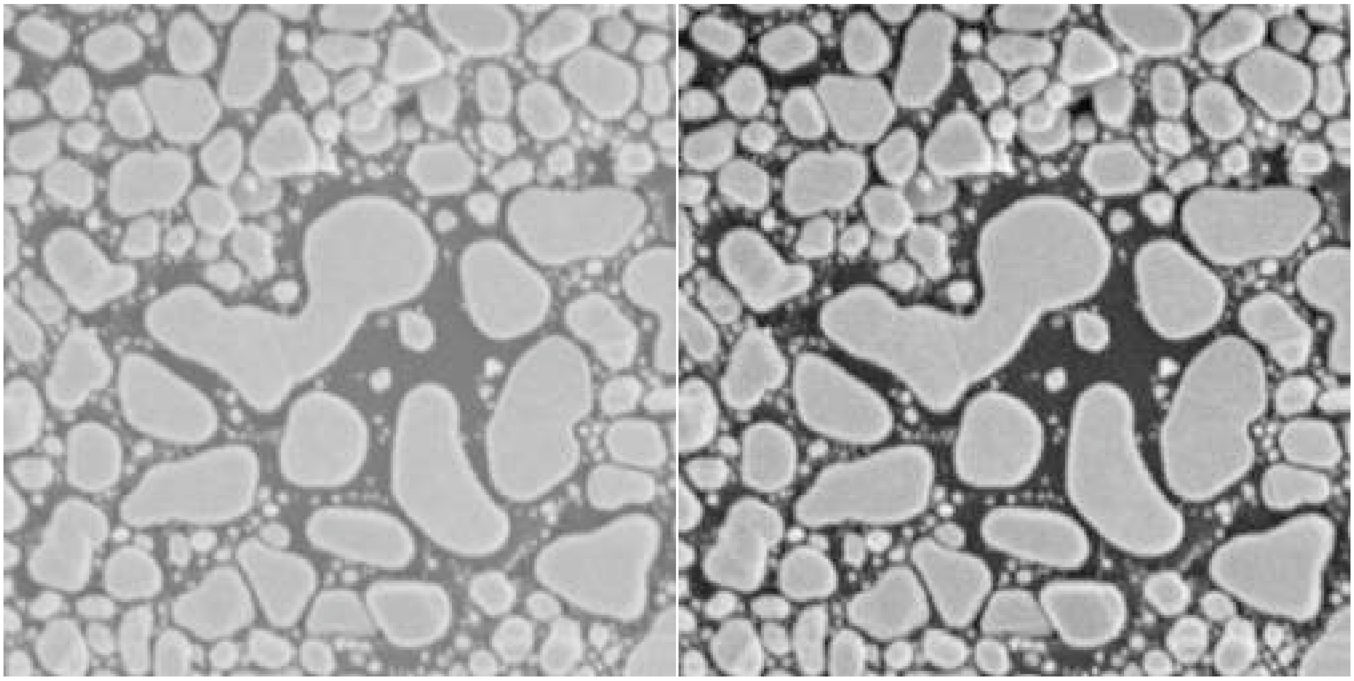
2.D.1. Linear expansion for grey scale and color images

The image histogram is a plot showing the number of pixels as a function of the pixel brightness value. It is displayed by Photoshop in a palette, and can be saved to disk (as well as displayed in both the usual form and as a cumulative or integrated plot) by **IP•Measure Global–>Histogram**. The examples show the histogram of the *Spheres2* image from the Photoshop Histogram palette and the plug-in (the red line shows the cumulative histogram). This histogram indicates good contrast, covering the full range of brightness values (the few very bright pixels are the specular reflections from the spheres).

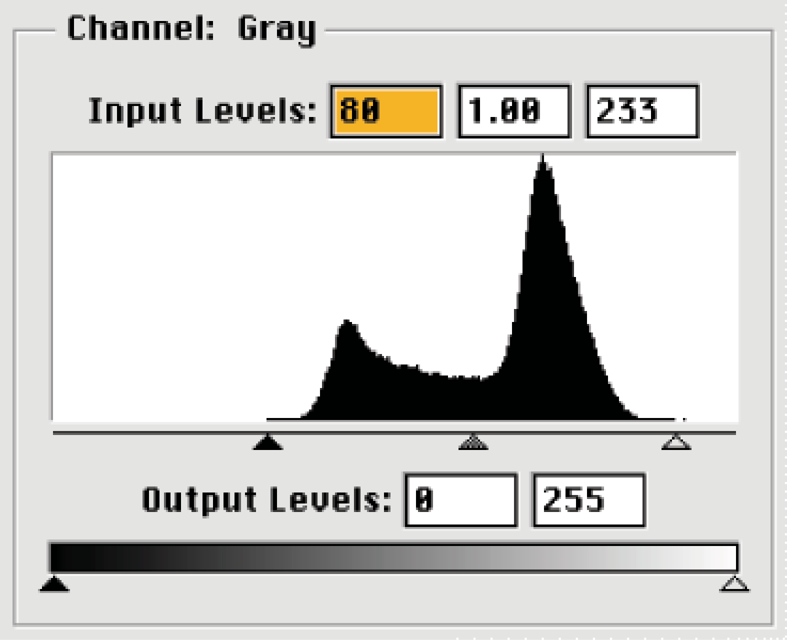




Examine the image histogram to determine whether it covers the full dynamic range without clipping. Many image acquisition problems are revealed in the histogram (broad peaks indicate nonuniformity or noise, clipping indicates improper lighting or poor brightness/contrast adjustment and the loss of data, comb patterns with missing values indicate ADC problems or limited bit depth for the image, etc.). Maximizing (stretching) low contrast by setting dark and bright limits using the image histogram is very fast (select **Image–>Adjustments–>Levels**).



The original *Au_Resn* image and the result of linear stretching using the Levels adjustment



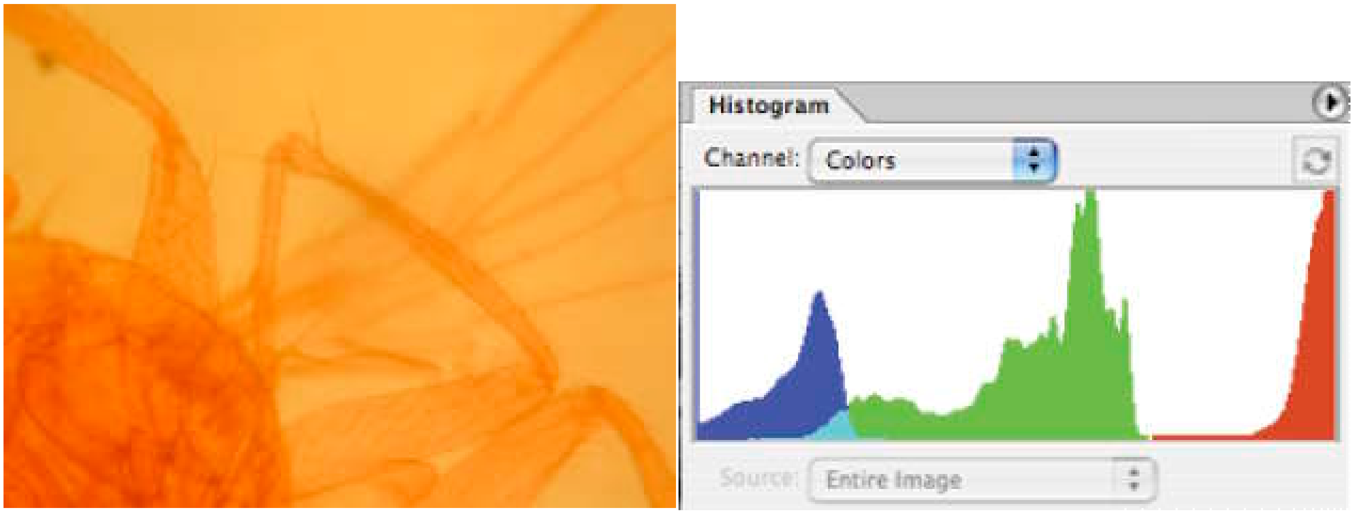
Channel: Gray

Input Levels: 80 1.00 233

Output Levels: 0 255

Setting the white and black points on the histogram for the *Au_Resn* image

For color images, the stretching must be done to the intensity information while preserving hue and saturation, not on the individual RGB channels, to avoid color shifts. This can be done by converting the image to Lab mode and processing just the L channel, but it is generally easier to use the **IP•Adjust –> Contrast** function which allows setting the limits manually or automatically. It is not recommended to use the built-in **Image–>Adjustments–>Levels–>Auto** adjustment on the individual RGB channels, since that can cause color shifts as shown in the example.

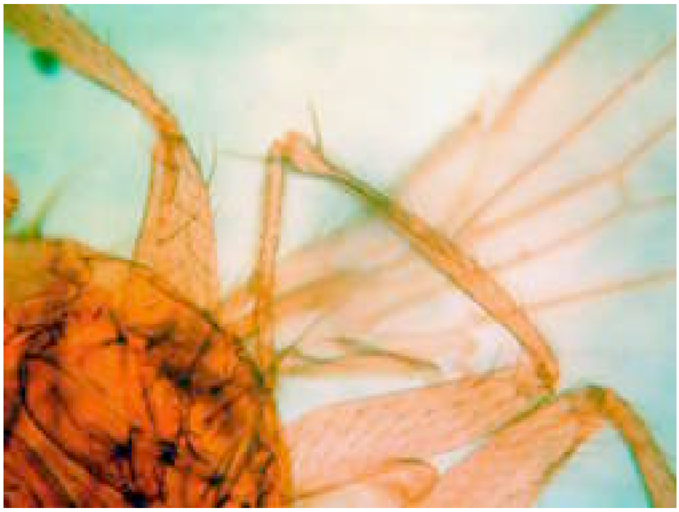


Histogram

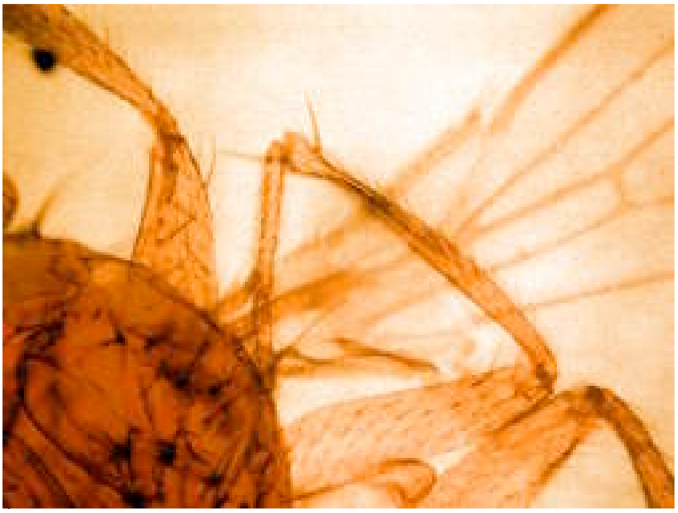
Channel: Colors

Source: Entire Image

Original *Fruitfly* image and its histogram



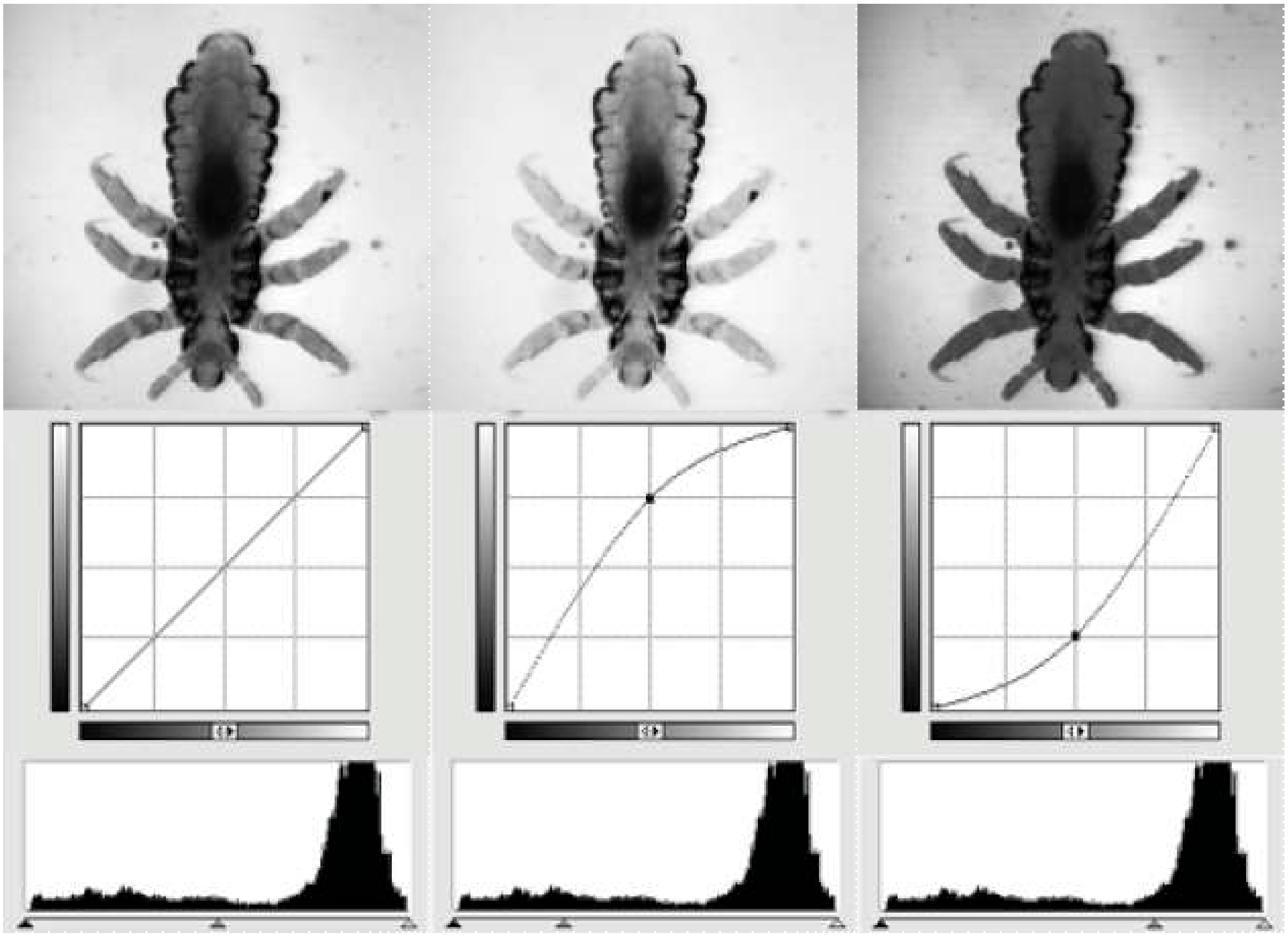
Photoshop Auto Levels function!



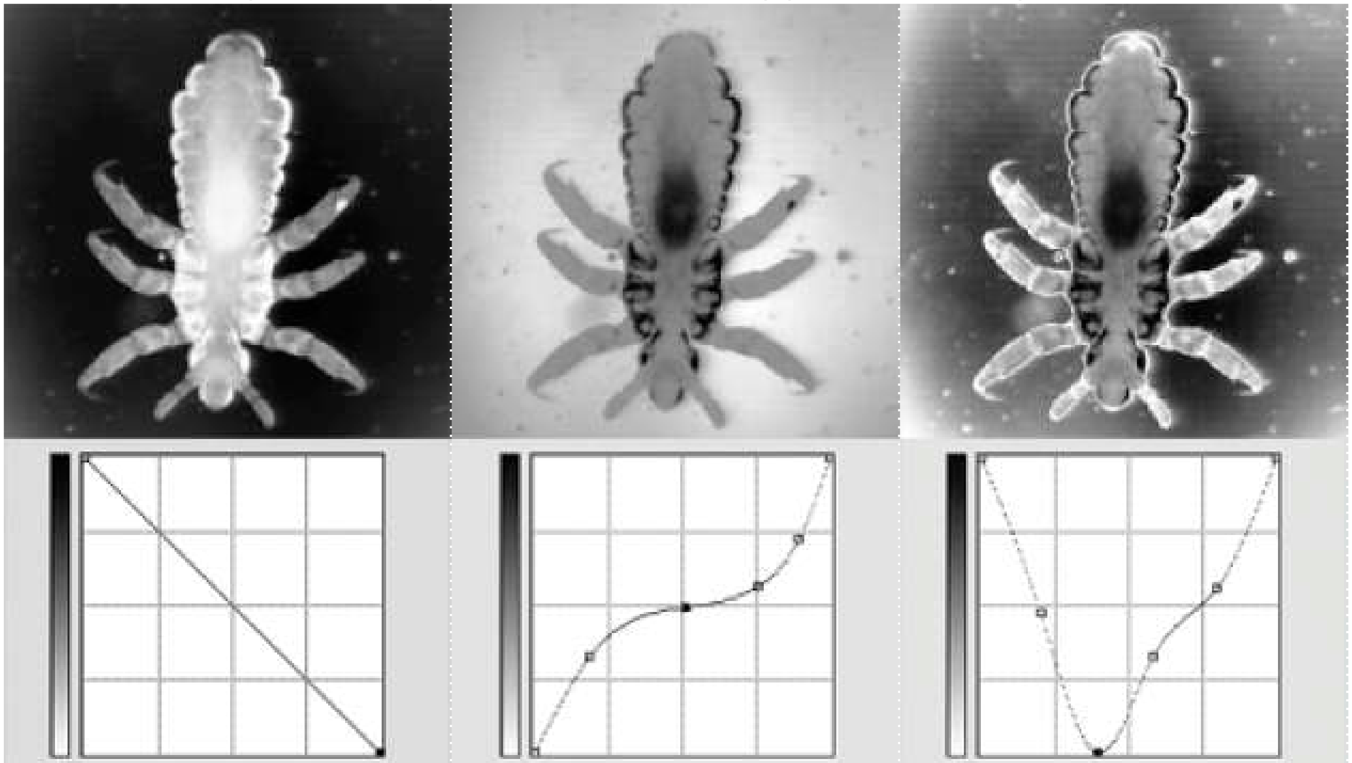
IP•Adjust–>Contrast–>Auto plug-in

2.D.2. Non-linear adjustments (gamma, equalization)

Setting the black and white points (either manually or automatically) stretches the brightness values linearly. Nonlinear functions selectively expand contrast in one part of the grey scale range by contracting contrast elsewhere. This can also be used to compensate for the characteristics of the acquisition device. Simple adjustments to gamma (which has the same meaning as in traditional photographic darkroom processes) using the **Image–>Adjustments–>Levels** dialog, or complete control over the shape of the transfer function using the **Image–>Adjustments–>Curves** dialog can be used as appropriate. Also, since human vision is logarithmic, viewing the negative image (**Image–>Adjustments–>Invert**) sometimes allows details to be seen more readily. In the examples that follow, first the use of levels to adjust the image gamma is shown, with the equivalent setting in the curves dialog that produces the same result. The second set of examples shows other manipulations that are only possible with curves.

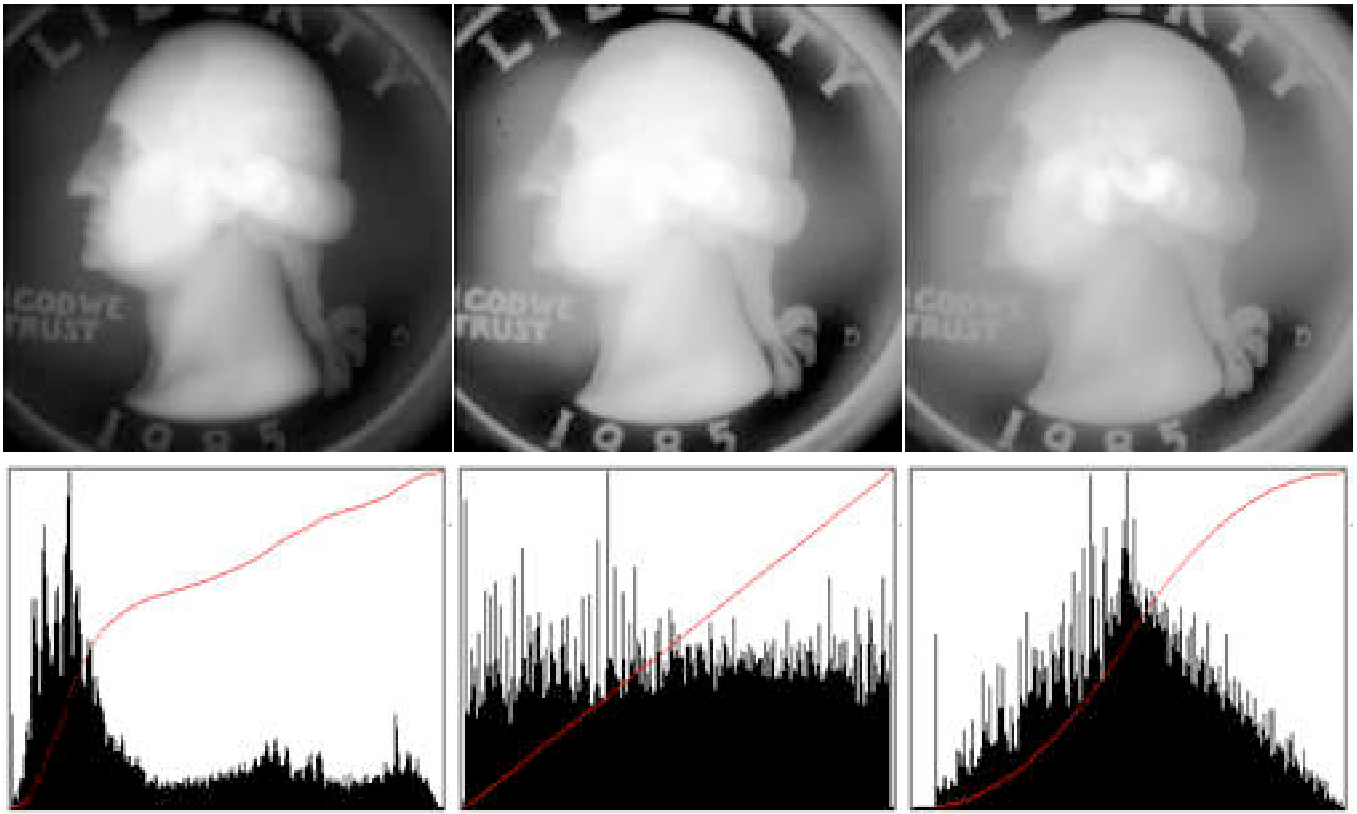


Original *Bug* image and the results of setting gamma > 1.0 and gamma <1.0



Inverting the image, expanding contrast arbitrarily, and reversing part of the contrast range (solarization)

The graph shown in the preceding examples is called the transfer function, relating the original brightness values to the resulting ones. If this transfer function is assigned the shape of the cumulative histogram, it produces a result called histogram equalization in which regions that have similar brightness values or subtle gradients are spread out in grey scale to enhance the visibility of the differences. The result of the equalization produces an image in which the cumulative histogram is a straight line, as shown in the example. The name “equalization” comes from the fact that equal areas of the image are assigned to each possible brightness value. The **IP•Adjust–>Histogram Shaping** plug-in provides the same function as the Photoshop **Image–>Adjustments–>Histogram Equalization** routine for grey scale images, but processes the intensity channel leaving colors unchanged for color images, and also allows selecting curves that emphasize the dark, light, extreme or central brightness values as well as linear.



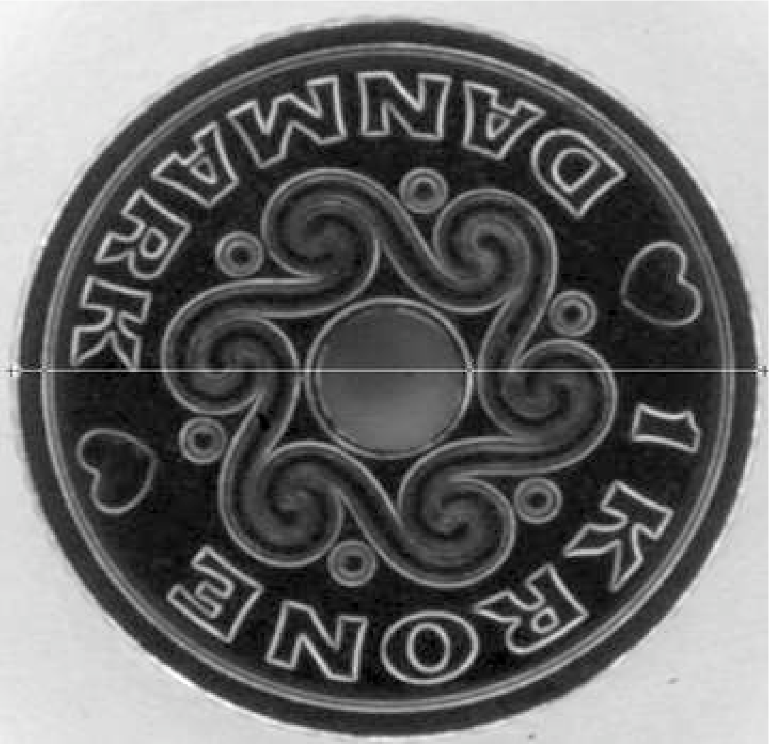
Original *Quarter* image with its histogram, and the results of linear and central emphasis equalization

It is important to be aware that manipulation of the image contrast can be very useful to assist in visual examination of structure, and as a precursor to delineation and thresholding of features. However, any calibration based on pixel brightness (e.g., for densitometry) is destroyed in the process, so it is often advisable to keep a copy of the original image so that measurements of color or brightness can be performed.

2.E. Distorted or foreshortened images

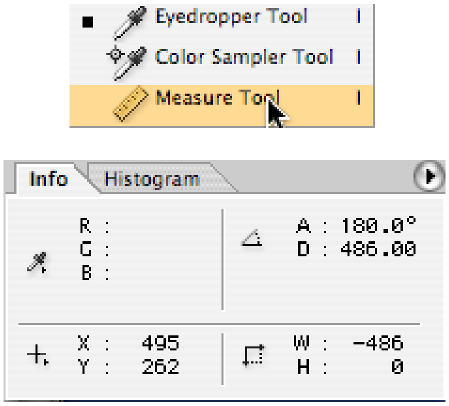
2.E.1. Making pixels square

Particularly with video cameras and analog to digital converters used with scanning microscope instruments, adjustment is needed to make dimensions the same vertically and horizontally. Non-square pixels create a variety of problems for image processing and measurement. Acquiring images of a stage micrometer or other known calibration device that is oriented vertically and horizontally can be used but it is even simpler to image a single known structure that has the same vertical and horizontal dimensions, such as a grid, or the coin shown in the example. Using the Photoshop ruler tool to measure the height and width of the coin (469 and 485 pixels, respectively) indicates that the width of the image should be reduced to 96.7% (=469/485) of its current value to make the dimensions equal. Selecting the entire image (**Select–>All**) and choosing **Edit–>Transform–>Scale** allows entering this value into the width field, as shown, resulting in an image with square pixels that can be used for further work. It is generally better to compress either the vertical or horizontal axis, as needed, rather than expanding one of them.



!

Measuring the coin width in the Kron_vid image."



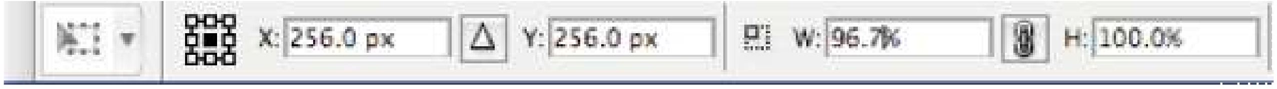
R :
G :
B :

A : 180.0°
D : 486.00

X : 495
Y : 262

W : -486
H : 0

Selecting the ruler and reading the length.

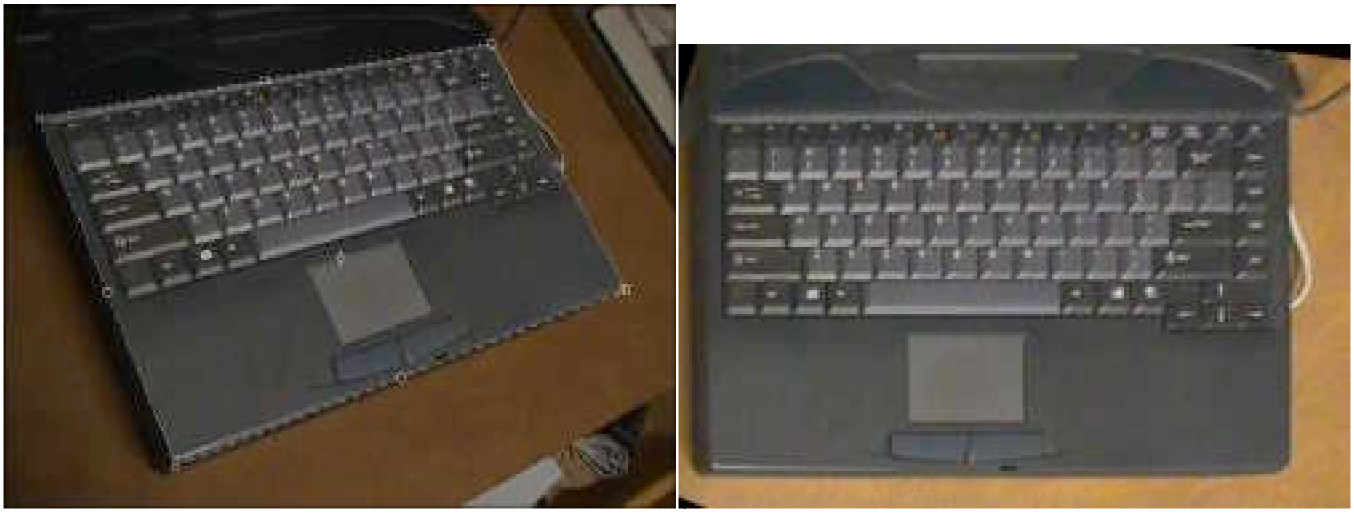


X: 256.0 px Y: 256.0 px W: 96.7% H: 100.0%

Entering the width into the Transform–>Scale function

2.E.2. Perspective distortion (non perpendicular viewpoint)

When surfaces are viewed at an angle, the distortion is much greater than simply non-square pixels. The same feature would appear to have a different size depending on where it lies in the image. The light microscope has a shallow depth of field and usually the viewpoint is perpendicular to the sample, but in the electron microscope it is common to have tilted specimens, which results in trapezoidal distortion. This must be corrected to permit meaningful measurements and even to facilitate proper image processing. The built-in Photoshop crop tool can correct for perspective distortion of planar surfaces as shown in the example. Use the tool to draw a rectangle around the surface to be corrected, be sure that the “perspective” box is checked in the tool bar, and position each corner of the selection to a corner of a rectangular region on the surface. The region can be proportionately enlarged by holding down the Alt/ Option key while dragging one of the handles on the selection. Then click on the check mark to produce a corrected image.

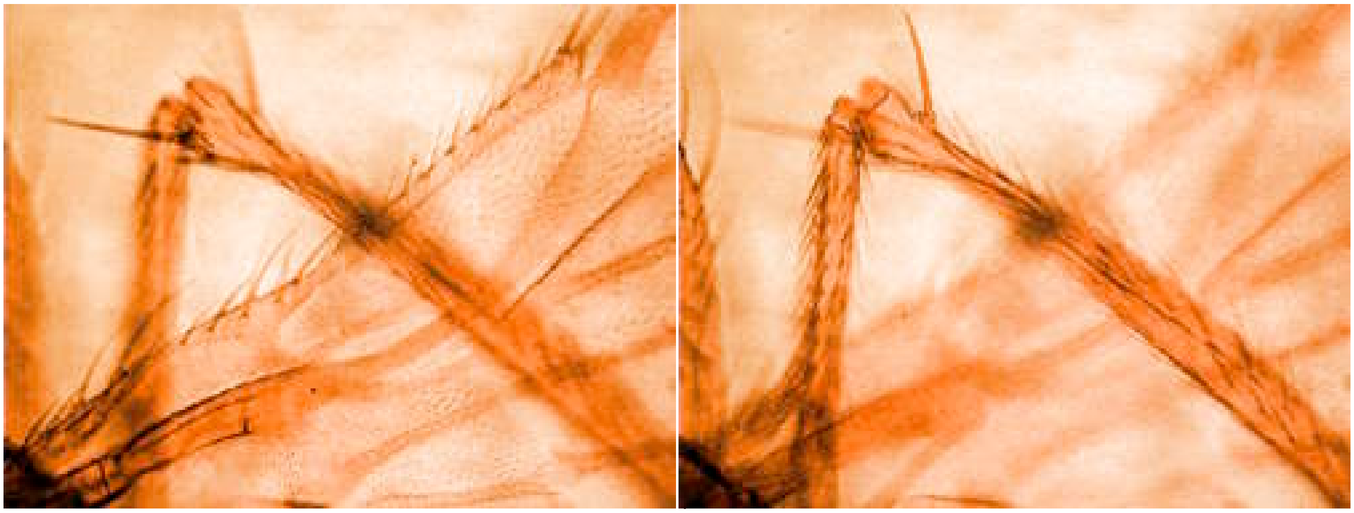


Original *Perspect* image with superimposed crop region, and the corrected result

2.F. Focus problems

2.F.1. Shallow depth of field

When the optical depth of field is insufficient to produce an image in which everything is in focus, it may be practical to capture a series of images and combine them. This method requires that the images be aligned and at the same magnification scale. That is difficult to do when the focus is adjusted by altering the lens (as on a typical camera). Multiple images obtained by moving the lens relative to the sample (as is done in a typical light microscope) can be combined by keeping the in-focus pixels from each. Place one image into memory (**IP•2nd Image–>Setup**), select the next image and choose **IP•Adjust–>Best Focus**. Repeat this for each of the images in the series.

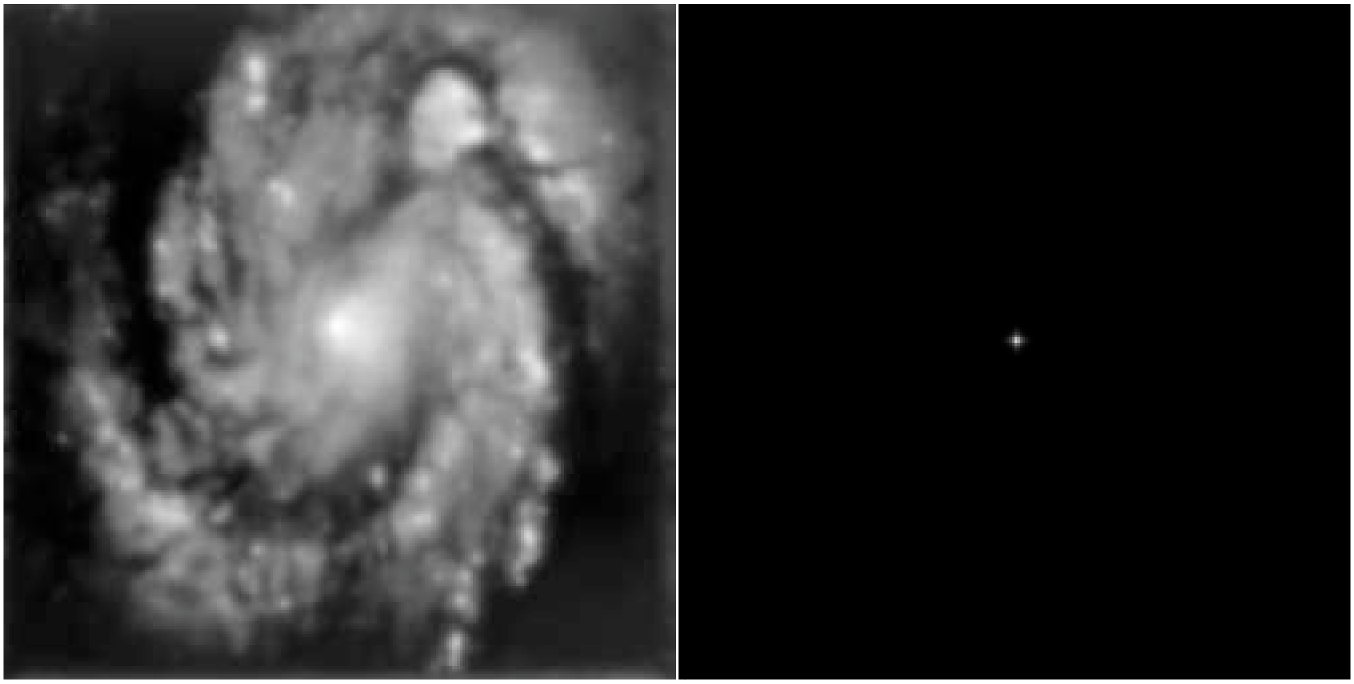


Three light microscope images (*Fly_1,2,3*) taken by raising the stage to bring different regions into focus, and the composite in which all regions are in focus.

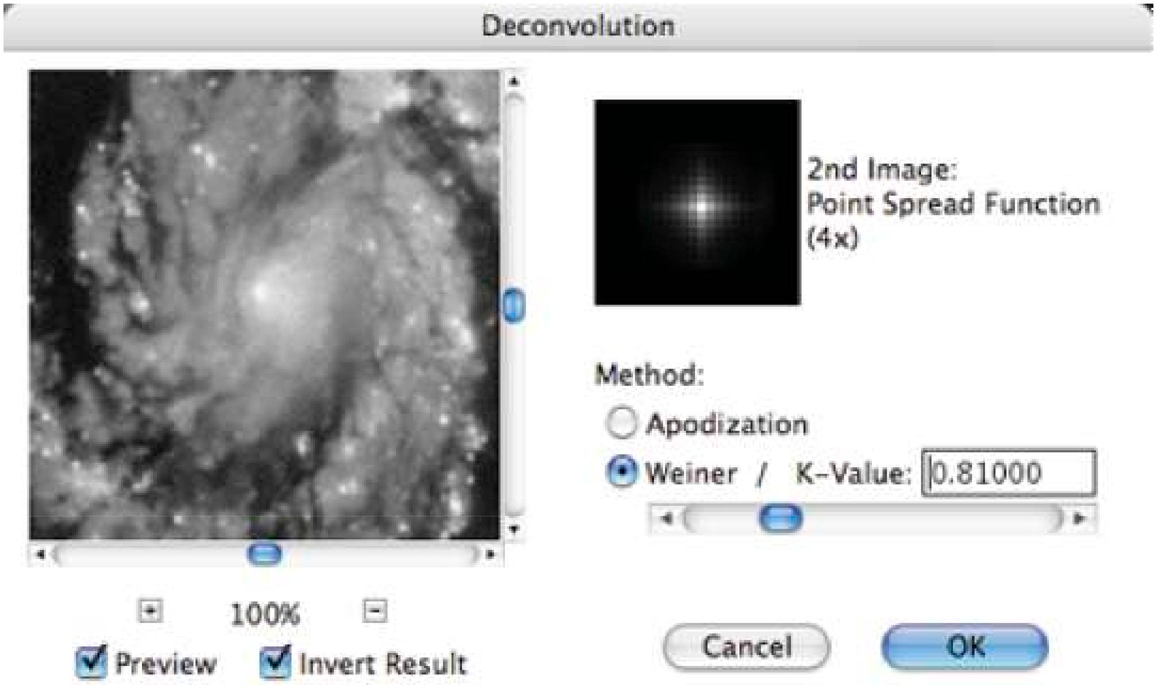
2.F.2. Deconvolution of blurred focus

The usual causes of blurred images are out-of-focus optical settings or camera motion. It is possible to correct a significant portion of these sources of blur provided that the same cause of blur affects the entire scene. The point-spread function (PSF) is an image of the blur applied to a single point. If the PSF can be directly recorded, for instance in astronomy as the image of a single bright star, then in the ideal case, dividing the Fourier transform of the PSF into the transform of the blurred image, and performing an inverse FFT back to the pixel domain, recovers the unblurred image. In the typical real case, the presence of noise in either image limits the amount of deconvolution that is possible, and the Wiener constant is introduced to limit the effect of the noise.

In the case in which a point spread function has been measured, place the PSF image into memory (**IP•2nd Image –>Setup**), select the blurred image, and choose **IP•FFT–>Deconvolution**. Either apodization (ignoring terms in the transform where the denominator in the division gets too small) or an adjustable Wiener constant (controlling the tradeoff between sharpness and noise) can be selected. The Fourier transforms and other computations are performed automatically, and the images are not restricted to power-of-two dimensions (they are padded automatically to the next larger size).



Original *Hub_blur* image and the measured point spread function (*Hub_psf*)



Deconvolution

2nd Image: Point Spread Function (4x)

Method:

○ Apodization

● Weiner / K-Value: 0.81000

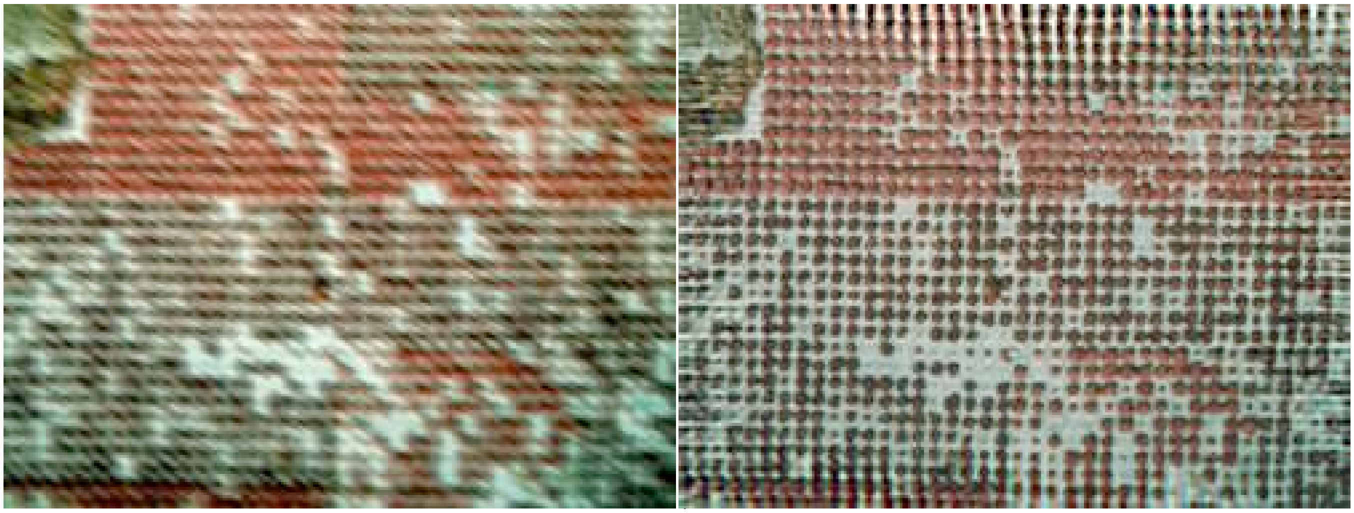
100%

☑ Preview ☑ Invert Result

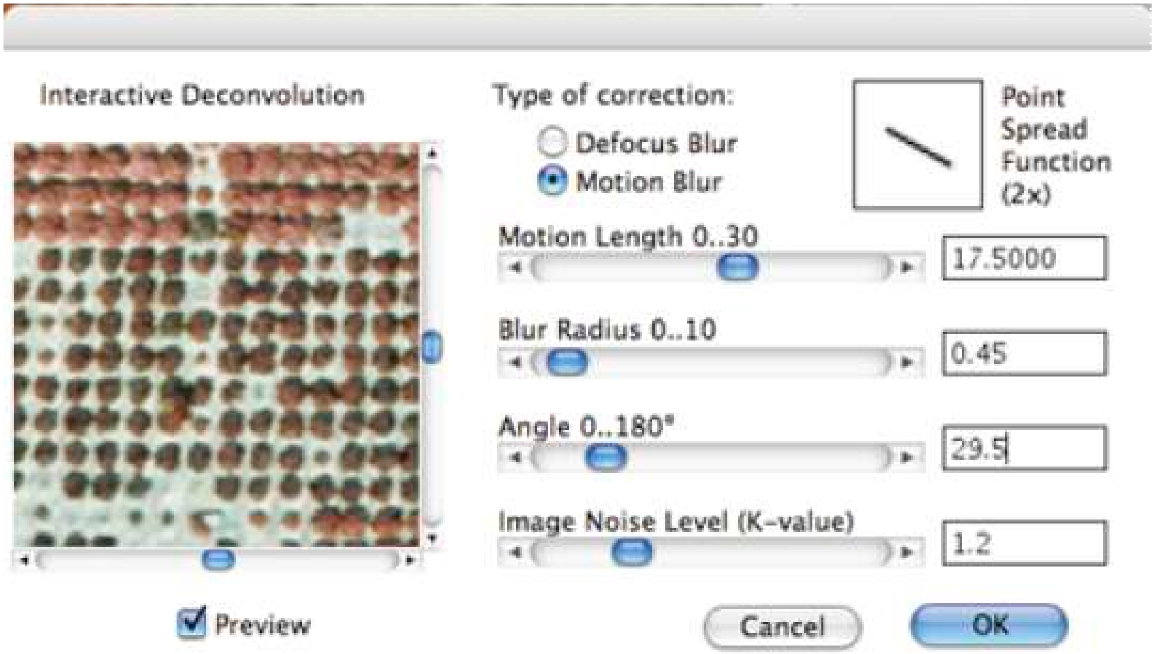
Dialog for the deconvolution plug-in

When no measured PSF is available, it may still be possible to perform a useful deconvolution to remove blur by interactively modeling a plausible point-spread function. Typically a Gaussian shape provides a good approximation to out-of-focus optics, while a straight line segment models blur due to motion. Adjusting the length and angle of the line (and perhaps its blur), or the standard deviation of the Gaussian (and perhaps the amount and orientation of any astigmatism) can often be done efficiently while watching the results in a preview. The example shows removal of motion blur from an aerial photograph (the line shown for the PSF corresponds to the motion of the airplane during the exposure) using the **IP•FFT –>**

Interactive Deconvolution plug-in. Notice that the deconvolution is imperfect at the top and bottom of the image, because the motion included a slight component of rotation as well as translation.



The original *Orch_Blur* image and the result of interactive deconvolution



Interactive Deconvolution

Type of correction:

○ Defocus Blur

● Motion Blur

Point Spread Function (2x)

Motion Length 0..30: 17.5000

Blur Radius 0..10: 0.45

Angle 0..180°: 29.5

Image Noise Level (K-value): 1.2

☑ Preview

Dialog for the interactive deconvolution plug-in

2.G. Tiling large images

2.G.1. Shift and align multiple fields of view

One way to obtain a high resolution image of a large field is to capture a series of images and “stitch” or “tile” them together to make a single large picture. When this is done for panoramic imaging, with a camera mounted on a tripod, it is usually necessary to overlap the images and often to make local scale adjustments so that they fit together properly. But for situations such as a microscope in which image distortions are minimal and stage motion can be controlled, it is often practical to combine multiple tiles simply by creating a large enough space (choose one image and use the **Image–>Canvas Size** adjustment), and then copy and paste each image in place. Since the images are pasted into separate layers initially, they can be shifted for proper alignment (the arrow keys are useful for single-pixel shifts, the

IP•Adjust–>Nudge function for fractional pixel adjustments). Layers can be shown or hidden on the display by clicking on the “eye” icon for each, or the layer opacity adjusted, to facilitate viewing and comparison. When all of the tiles are positioned correctly, the **Layer–>Flatten Layers** selection combines them into a single image. The **Photoshop File–>Automate–> Photomerge** function can assist in this process. There are also third-party plug-ins and stand-alone programs that perform this function.

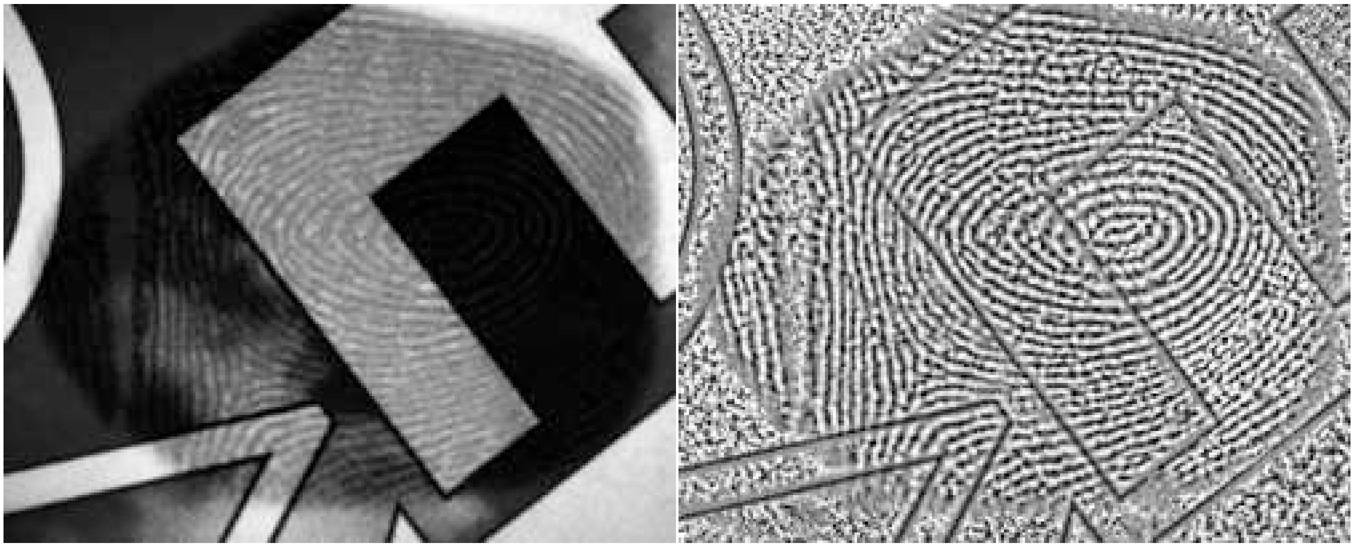
3. Enhancement of image detail

Many of the same classes of tools described above can also be used to enhance the visibility of some of the information present in the image, usually by suppressing other types of information (such as increasing local contrast by suppressing global intensity variations). The purpose may either be to improve visual interpretation of images (including better pictures for publication), to facilitate printing of images, or to allow subsequent thresholding of the image for measurement purposes.

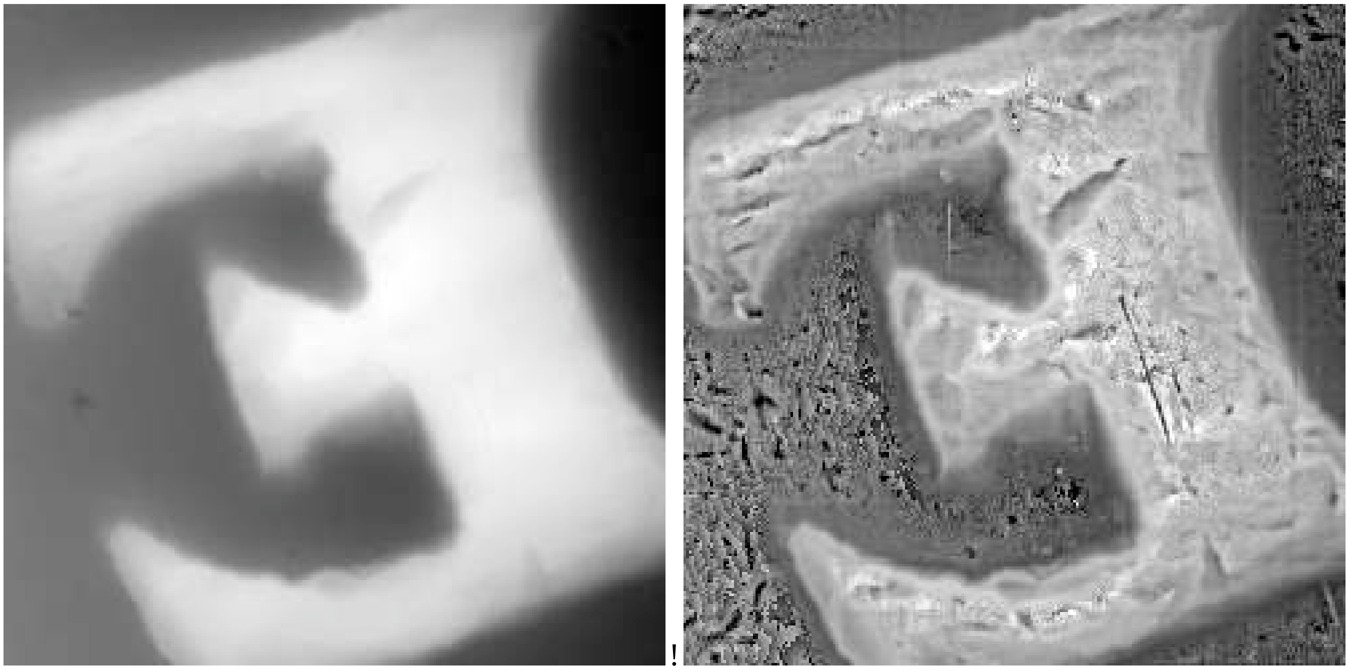
3.A. Poor local contrast and faint boundaries or detail

3.A.1. Local equalization

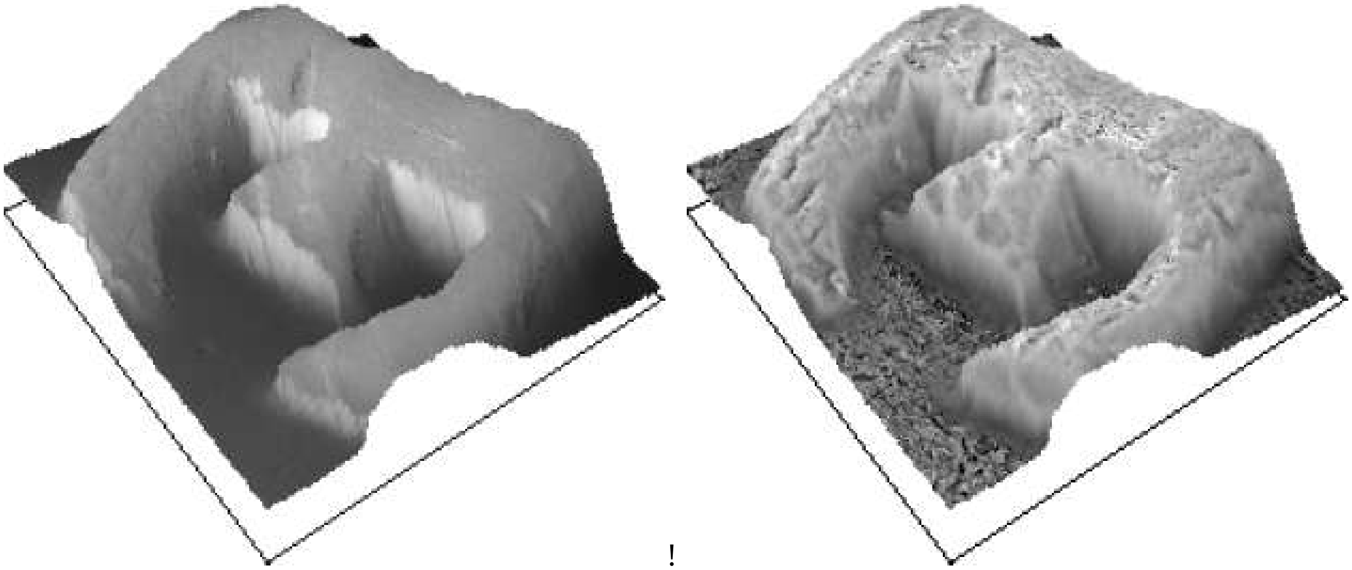
Increasing the local contrast within a moving neighborhood is a powerful non-linear tool for improving the visibility of detail. Local contrast equalization and variance equalization (**IP•Process–>Local Equalization**) suppress overall contrast while revealing local detail. Variance equalization is somewhat better at rejecting noise, but all local enhancement methods will also increase noise visibility which is why the processing steps in Section 2 should be used before those in Section 3. Adaptive equalization (**IP•Process–>Adaptive Equalization**) provides additional flexibility for making detail visible, as shown in the examples. Note that combining the results of this enhancement with surface rendering (discussed below) is particularly effective. For color images it is important to process the data in hue-saturation-intensity space (all of the plug-ins do that automatically, or you can convert to Lab space and process just the Luminance channel).



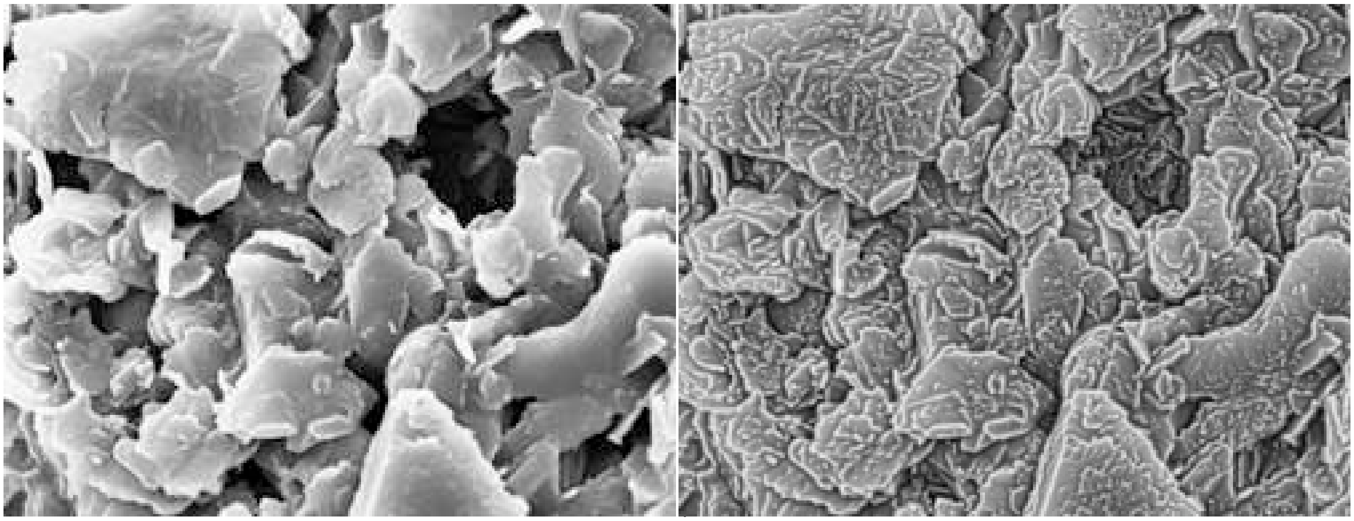
Original *FingerP2* image and the result of local contrast equalization



!
Original *CoinSurf* image and the result of local variance equalization

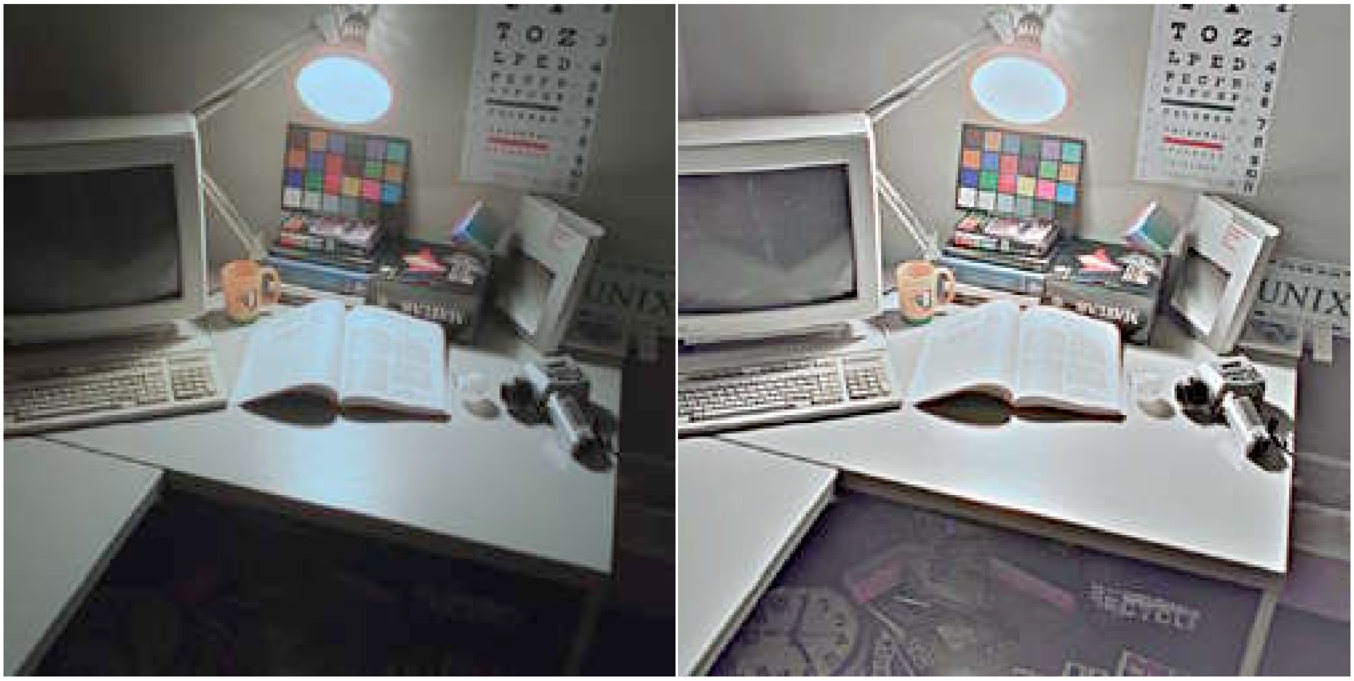


!
Rendered *CoinSurf* data as an surface display, and with the equalized image applied to the surface.
The surface rendering plug-ins (**IP•Surface Processing –> Plot 2nd as Surface** and
–> Reconstruct Surface with Overlay) are discussed in section 3.A.6



Original SEM image (*Contrast2*) and the adaptive equalization result
Showing enhanced detail on the surface and in the cavity

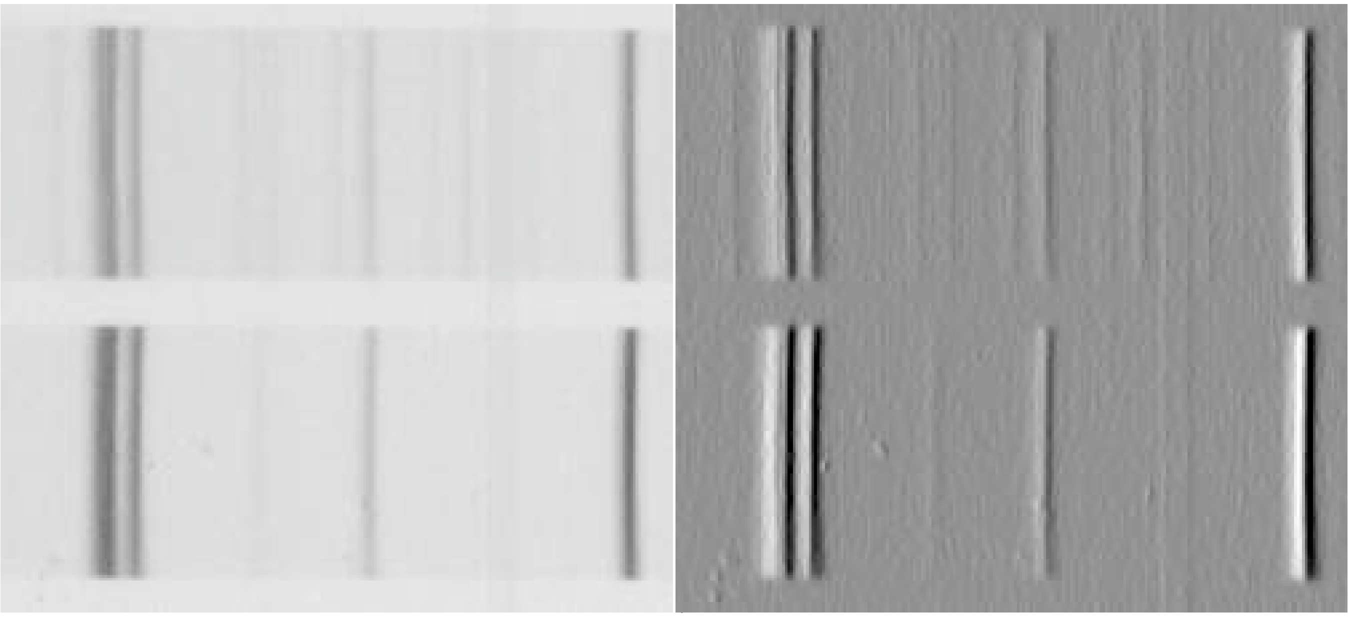
Images that have a high dynamic range, such as medical X-rays, and scenes that include brightly lit areas and deep shadow, typically require processing to compress the overall contrast range while preserving or enhancing the local detail. A combination of high pass filtering (performed in either the spatial or Fourier domains) and histogram shaping are often used as shown in the example.



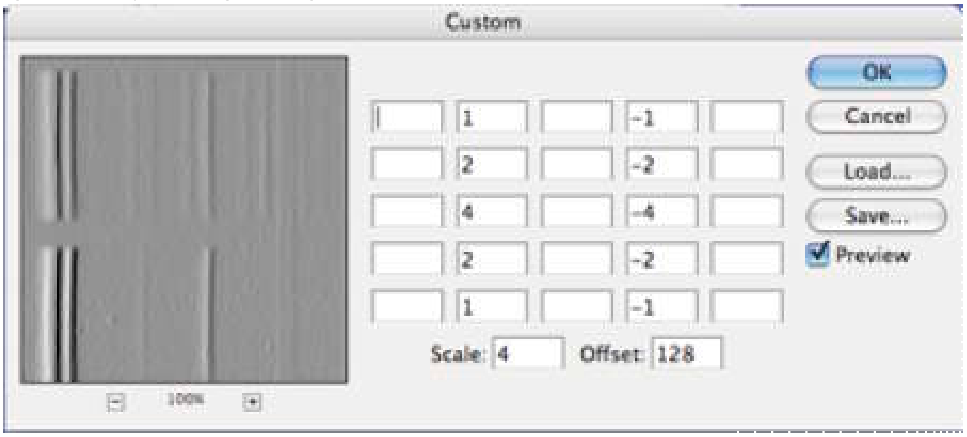
Original *Desk* image and the result of filtering and histogram shaping.

3.A.2. Sharpening (high pass filters)

Fine detail with a known orientation can be enhanced by a derivative, an example of a convolution kernel that is neither symmetrical nor positive like the Gaussian smooth shown in Section 2.B.1. The example shows the construction of the kernel with the **Filter–>Other–>Custom** function. Notice that the kernel takes the difference in the horizontal direction but performs averaging in the vertical direction to reduce noise. Also, the offset factor produces a medium grey where there is a zero difference, so that positive and negative values can be seen. The scale value is typically set equal to the largest factor in the kernel or to the sum of positive values, to keep the values within range. The **IP•Process–>Custom** plug-in is similar but allows entry of real numbers (vs. integers), a larger array, and selection of autoscaling.



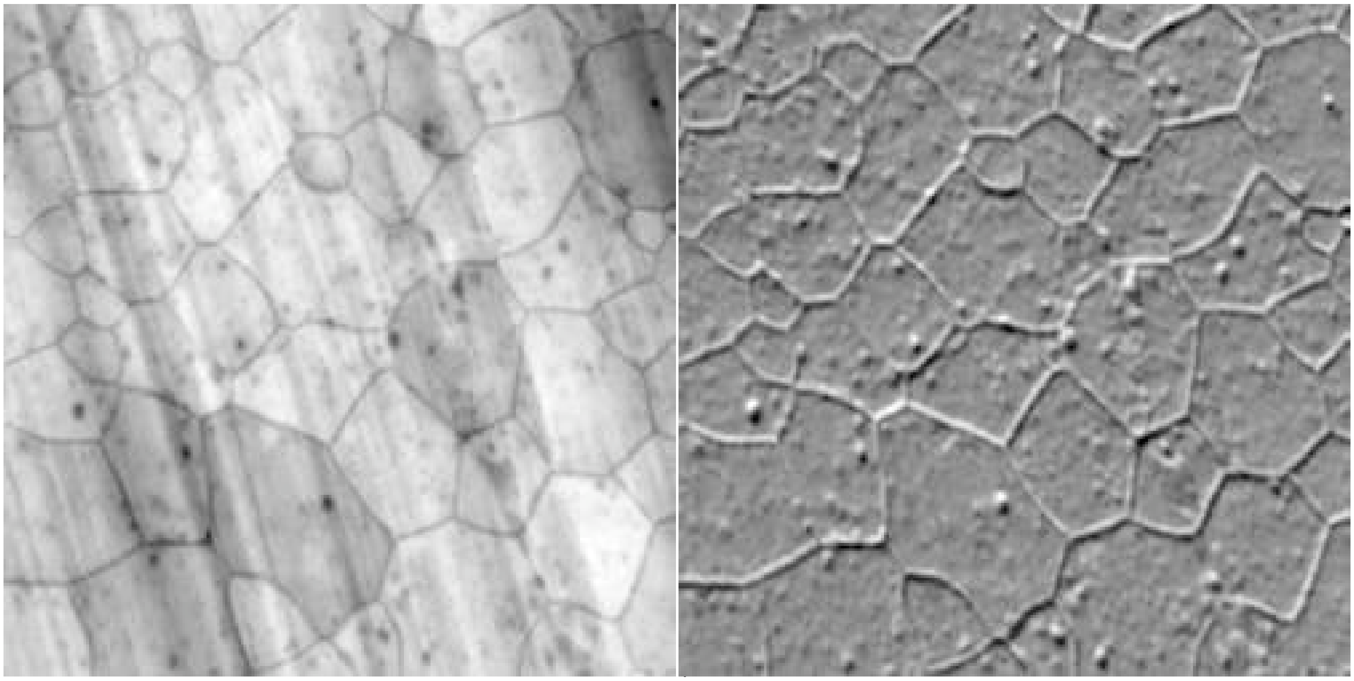
Original *Debye* image (fragment) and the result of the horizontal derivative filter



	1		-1	
	2		-2	
	4		-4	
	2		-2	
	1		-1	

Scale: 4 Offset: 128

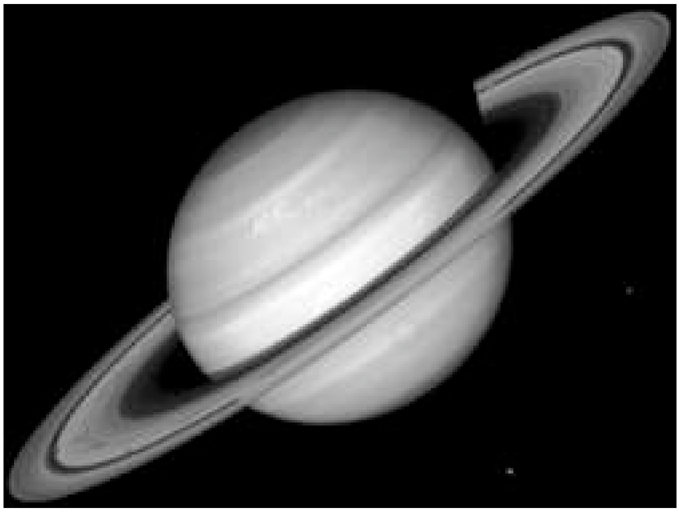
The directional derivative is sometimes called an “embossing” filter because it gives the visual impression of shadowed surface relief. One use of this filter is to eliminate directional noise or shadows in an image by orienting the derivative parallel to the orientation of the marks. In the examples, the markings on the surface of rolled metal interfere with the ability to see the grain structure, and the shadows interfere with the ability to see footprints in the snow. Using the Photoshop **Filter–>Stylize–>Emboss** routine allows setting the direction to match the marks and thus eliminate them.



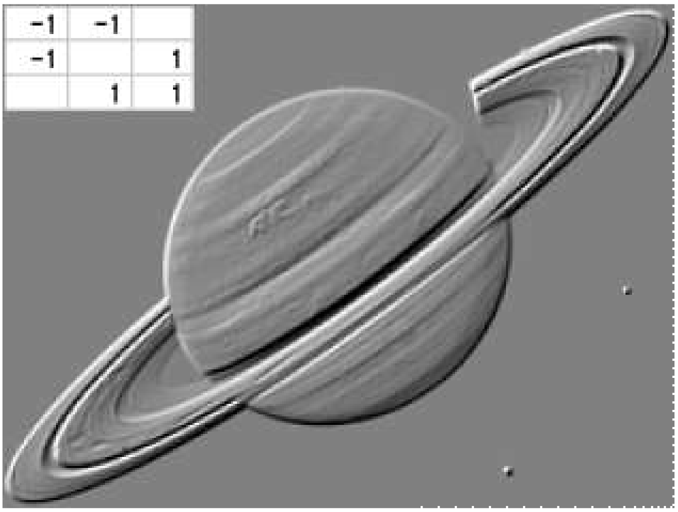
Original *Extrusn* image, and the result of setting the embossing filter to a direction of 110 degrees

Original *Footprints* image (fragment), and the result of embossing at an angle of 167 degrees

Changing the orientation of the derivative affects detail with the corresponding directionality. In order to enhance detail that may have any orientation in the image, a second derivative (Laplacian) can be used instead of a first derivative. The Laplacian result is often added back to the original image (which can be done by increasing the weight for the central pixel by 1) to provide a more readily interpretable image. This is often called a “sharpening” filter; the offset value used in the directional derivative and Laplacian is omitted for the sharpening filter. The 3x3 kernels of weights are shown in the examples. Classical sharpening operations increase the local contrast at fine detail and are equivalent to FFT-based high-pass filters as illustrated in Section 2.B.3.

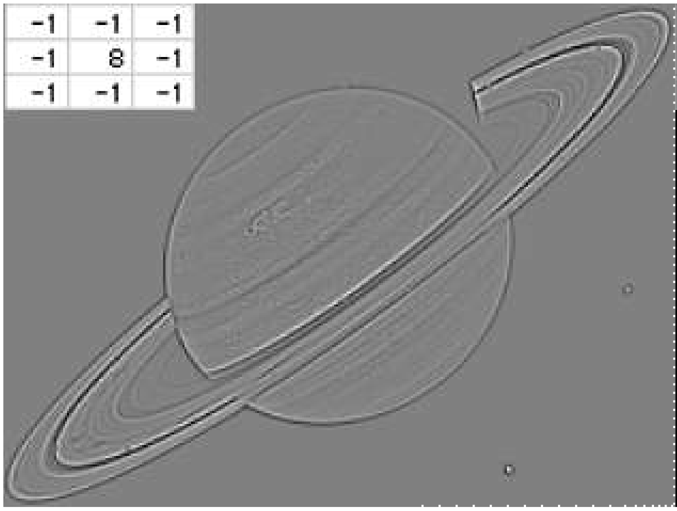


Original *Saturn* image"



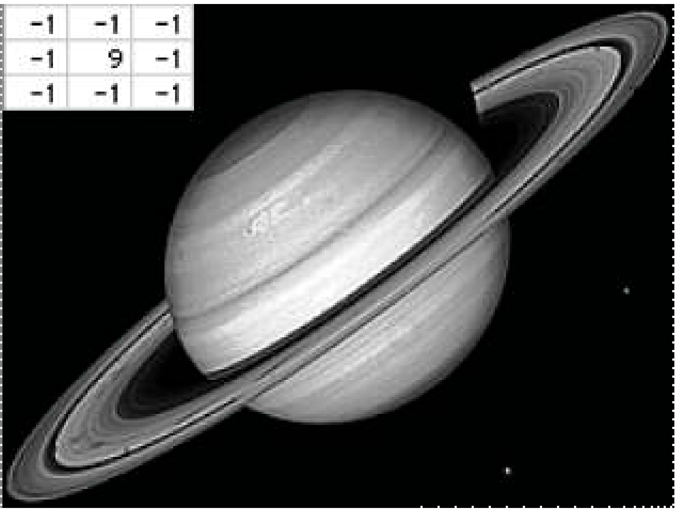
-1	-1	
-1		1
	1	1

Directional Derivative



-1	-1	-1
-1	8	-1
-1	-1	-1

Laplacian"

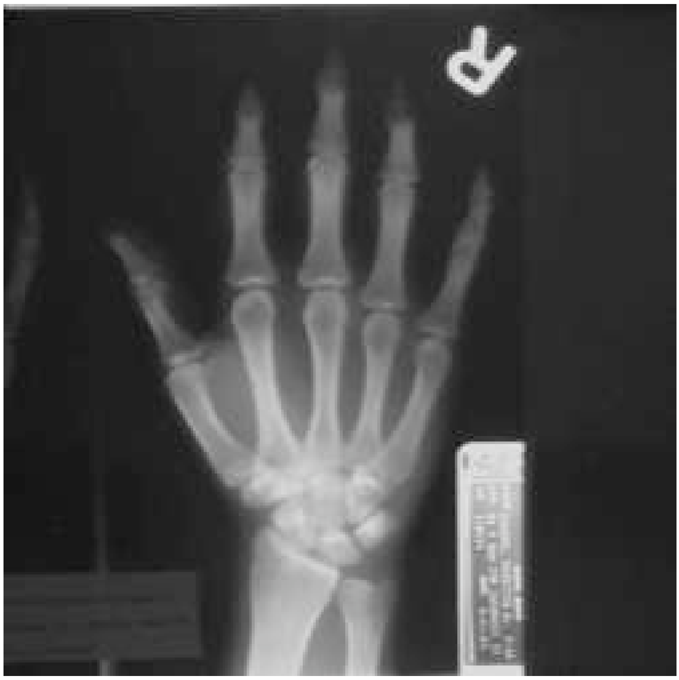


-1	-1	-1
-1	9	-1
-1	-1	-1

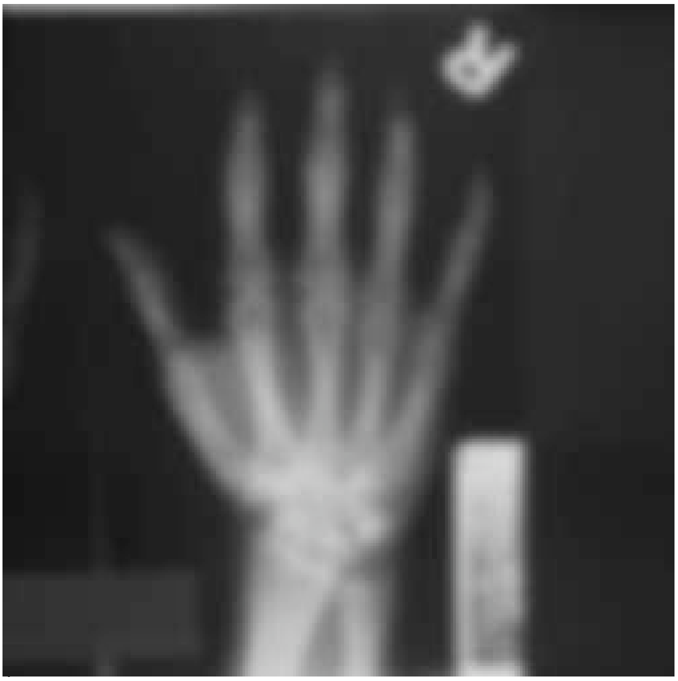
Sharpening filter

3.A.3. Unsharp mask and difference of Gaussians

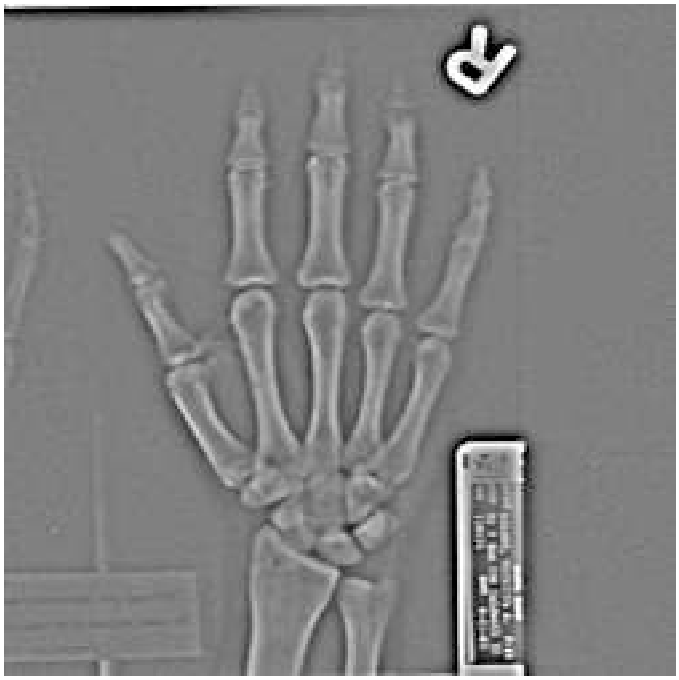
The sharpening filter shown above compares each pixel to its immediate neighboring pixels. The unsharp mask is a more general version of the same idea, and is a powerful tool derived from long-established photographic darkroom technique. A copy of the image is blurred, typically with a Gaussian filter large enough to remove whatever important detail is present. This is then subtracted from the original, since the difference is that same detail. In the built-in Photoshop unsharp mask (**Filter–>Sharpen–>Unsharp Mask**), an adjustable percentage of this difference (the detail) is added back to the original.



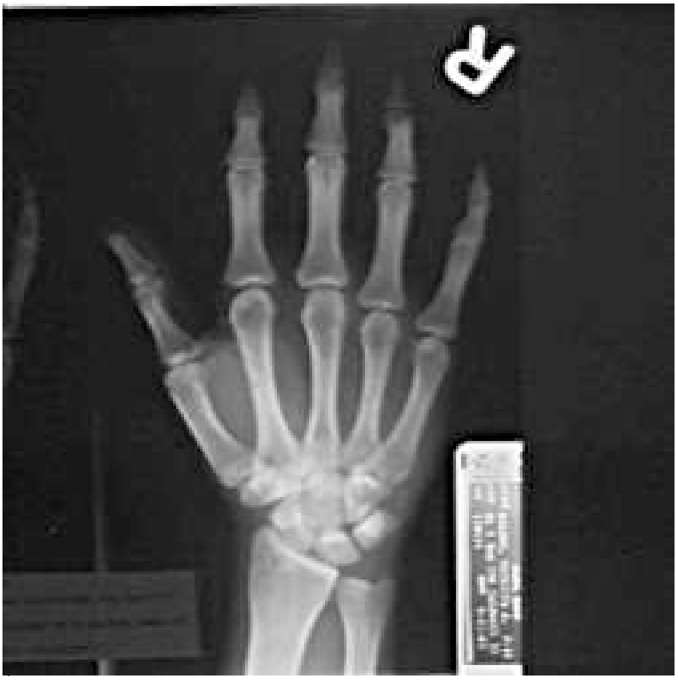
Original *Hand* image"



Gaussian blur (2.5 pixel radius)

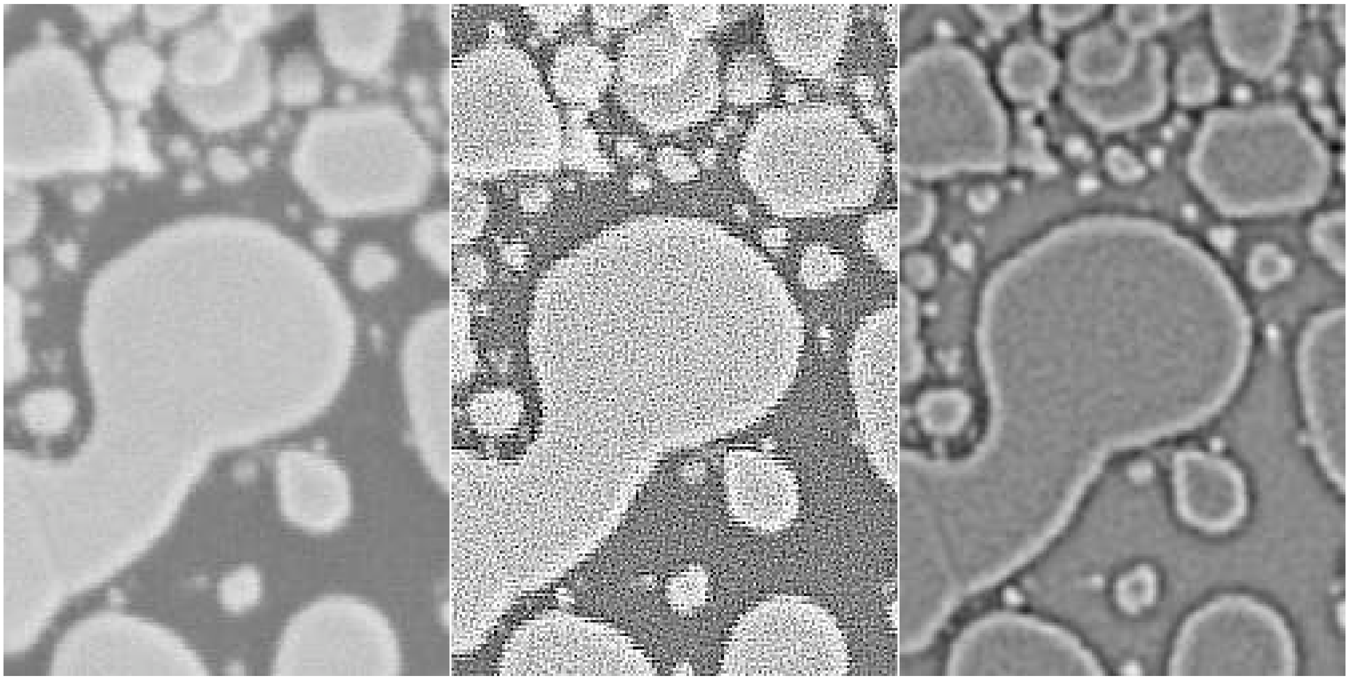


Subtracting blur from original"



Added back to original

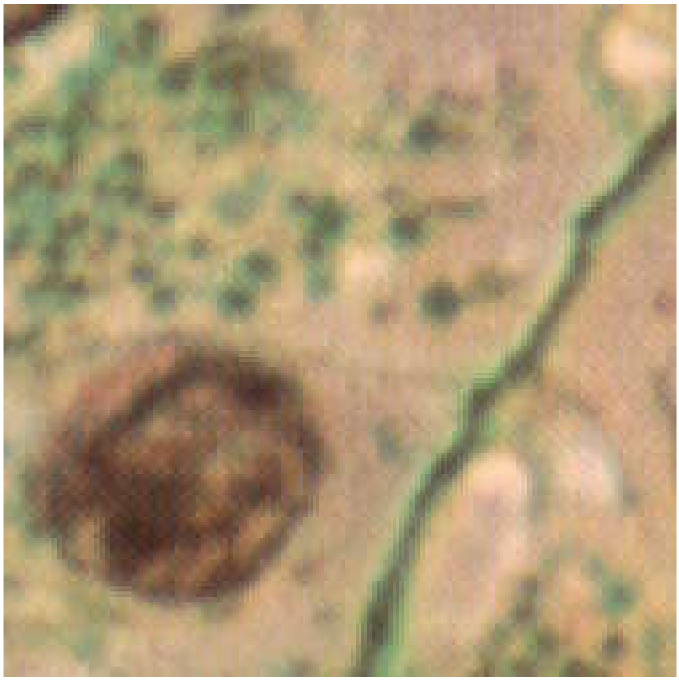
These sharpening methods increase the visibility of noise, so the Difference of Gaussians (DoG) technique, equivalent to a band-pass filter in Fourier space, is a better choice for noisy images. This uses two blurred copies of the original image, one with a small standard deviation that removes the high frequency noise, a second that removes the detail (as well as the noise). The difference shows edges while suppressing the effects of the noise. The **IP•Process–>Difference of Gaussians** plug-in allows setting the two smoothing values (one is typically about 3-7 times larger than the other).



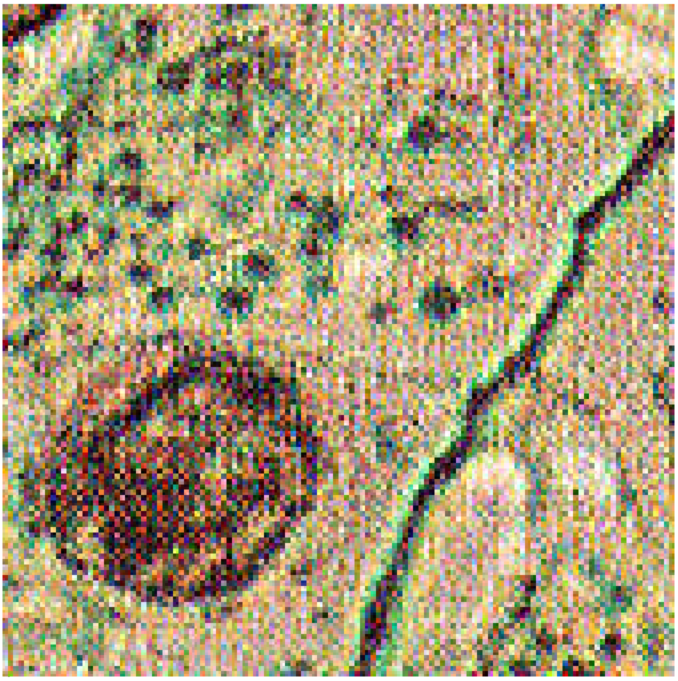
Original *Au_Resn* image, sharpening filter and difference of Gaussians

3.A.4. Color images should be processed in HSI space

Working on the separate RGB channels produces different ratios of values that are perceived as extreme color noise. The plug-ins automatically convert to HSI space and operate on just the intensity channel. In Photoshop, it is equivalent to change the image mode to Lab, and select just the L channel for processing.



Original *Tissue1* image (fragment)"

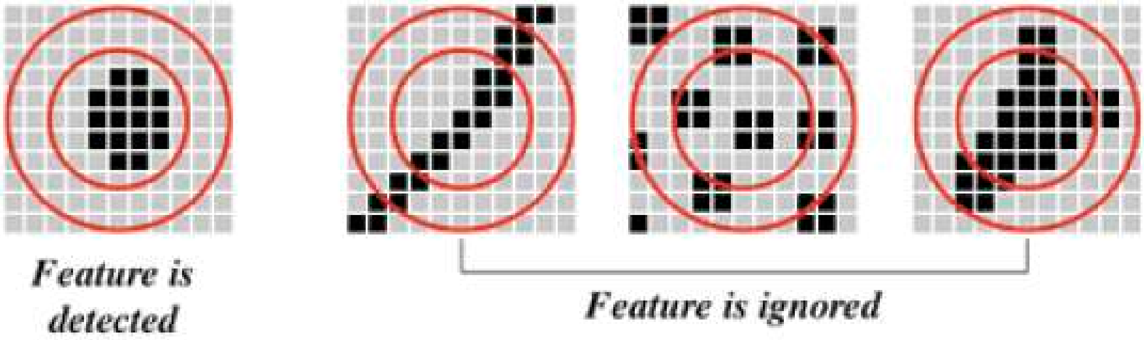


3x3 sharpening filter applied to RGB channels

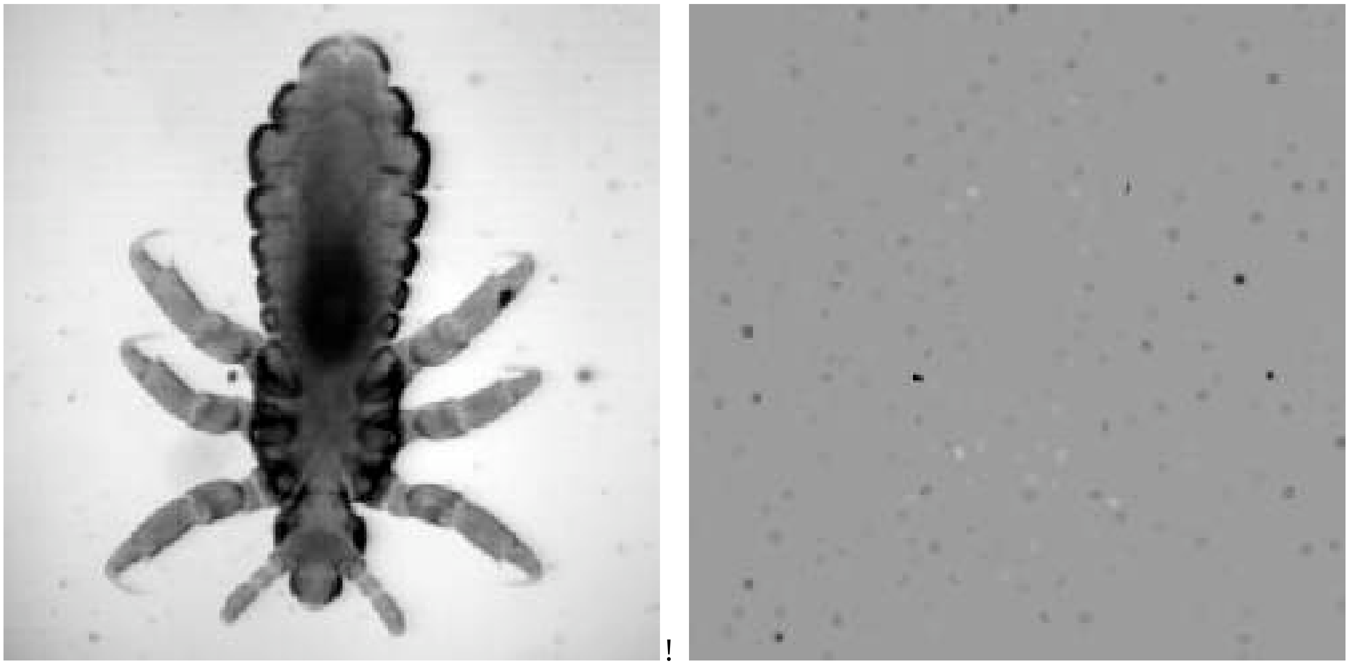
Difference of Gaussians filter applied to intensity channel only

3.A.5. Feature selection based on size

The difference of Gaussians compares the value of a small central neighborhood to the surrounding neighborhood. A different approach uses two different neighborhood sizes but compares the ranked maximum or minimum values. The top hat filter (**IP•Rank–>Top Hat**) can be used to select bright or dark objects for retention or removal (in the latter case it is called a rolling ball filter). As shown in the diagram, features that are too large for the central region are ignored by the filter. Features that fit into the smaller interior neighborhood (the crown of the hat) and are separated by the difference between the two neighborhood sizes (the width of the brim of the hat) are selected. The result of the top hat filter is a medium grey value except where features are found, where the difference between the brightest (or darkest) pixel value in the interior region and the brim is suitable for thresholding.

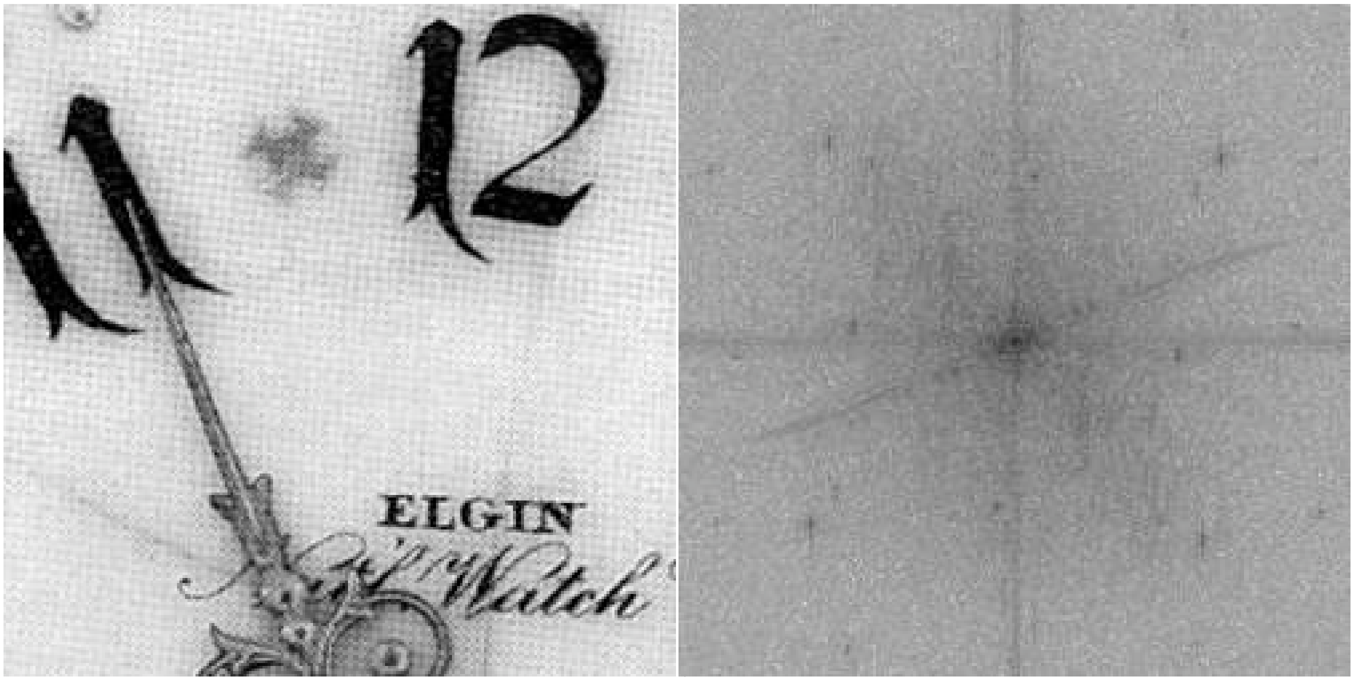


Principle of the top-hat filter: the darkest (or lightest) value in the interior region is compared to that in the surrounding annulus and the difference retained.

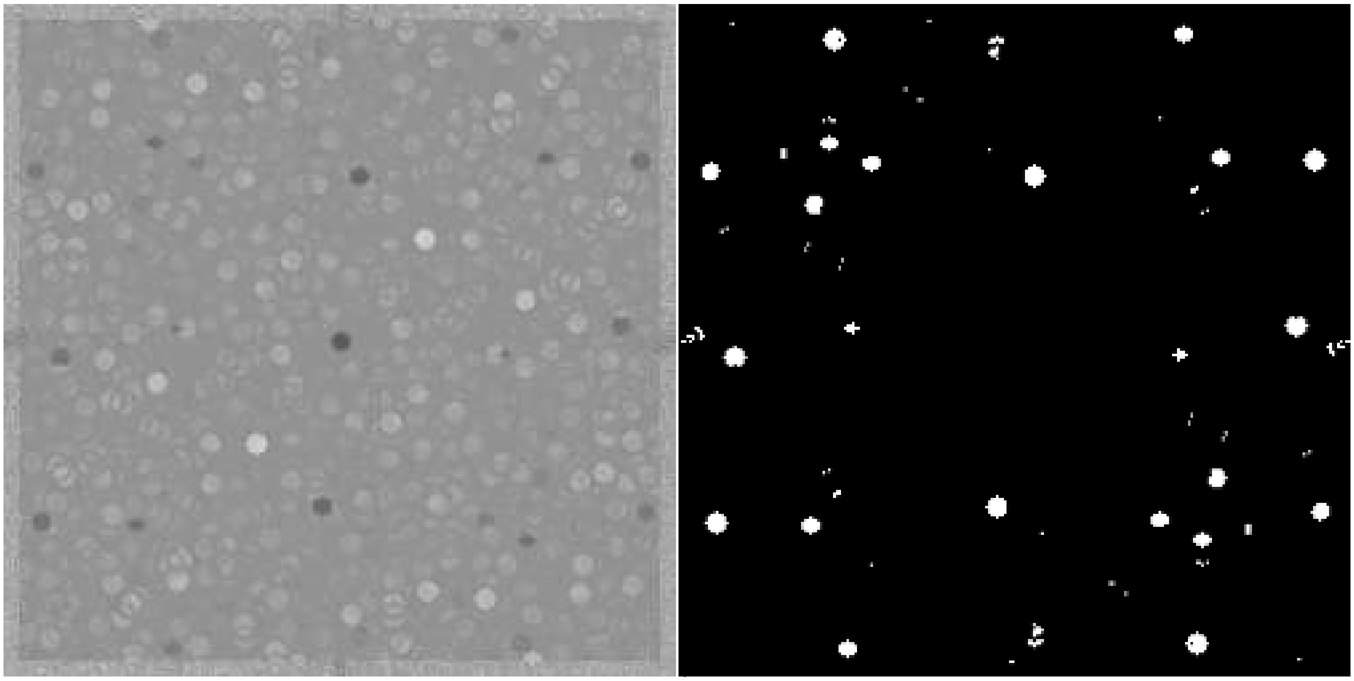


Original *Bug* image and the result of the top hat filter, which finds the dust particles on the slide

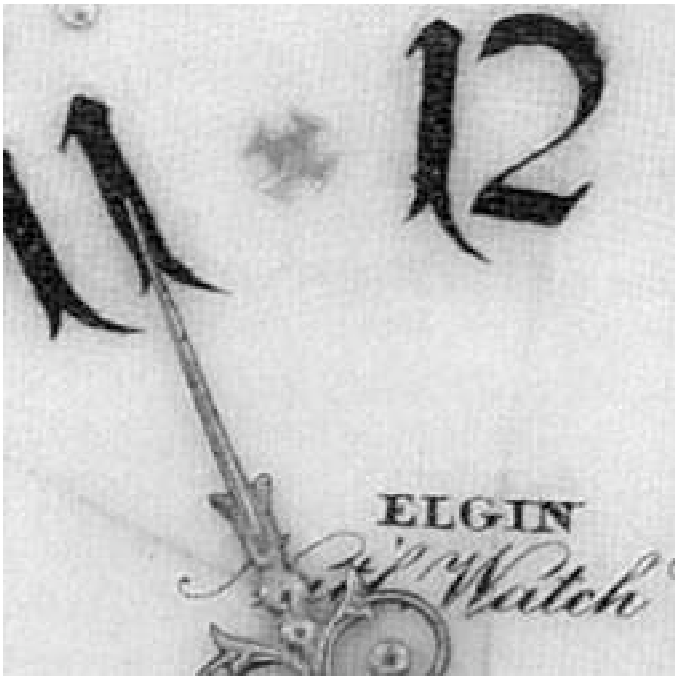
The top hat filter is particularly useful for locating peaks (“spikes”) in FFT power spectra. In the example, this is used to remove periodic noise automatically instead of the manual marking of spikes shown in Section 2.B.3. Thresholding the top hat for the dark spots, inverting the resulting binary image to remove the selected frequencies, and filling the central spot (which represents the mean brightness of the image) produces a mask that can be used with the inverse Fourier transform to produce the image without the periodic noise.



Original *Clock* image and its Fourier transform power spectrum



Application of the top hat filter to the power spectrum, and the mask that eliminates the noise frequencies

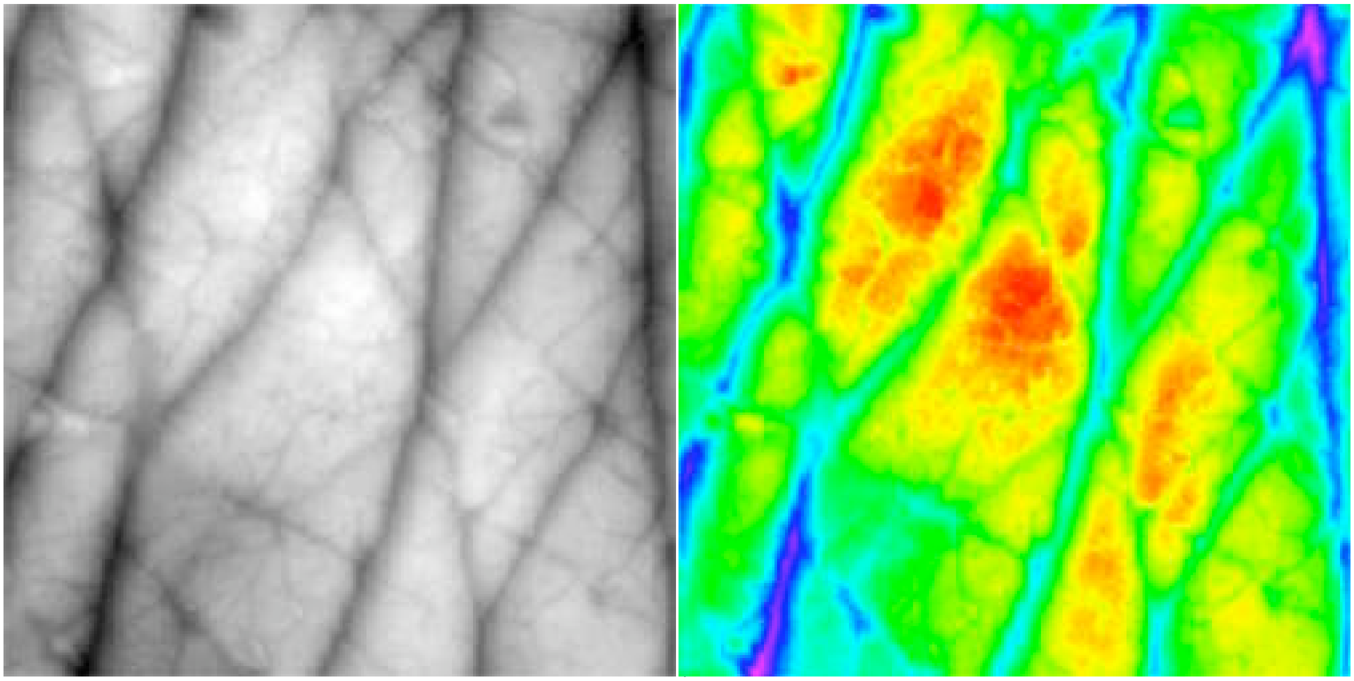


Inverse FFT using the mask removes the periodic noise

The same procedure can be used to keep periodic structure and eliminate random noise, as shown in the example in Section 3.D.1, below. This is an effective way of averaging together all of the repetitions of the same structure in the original image.

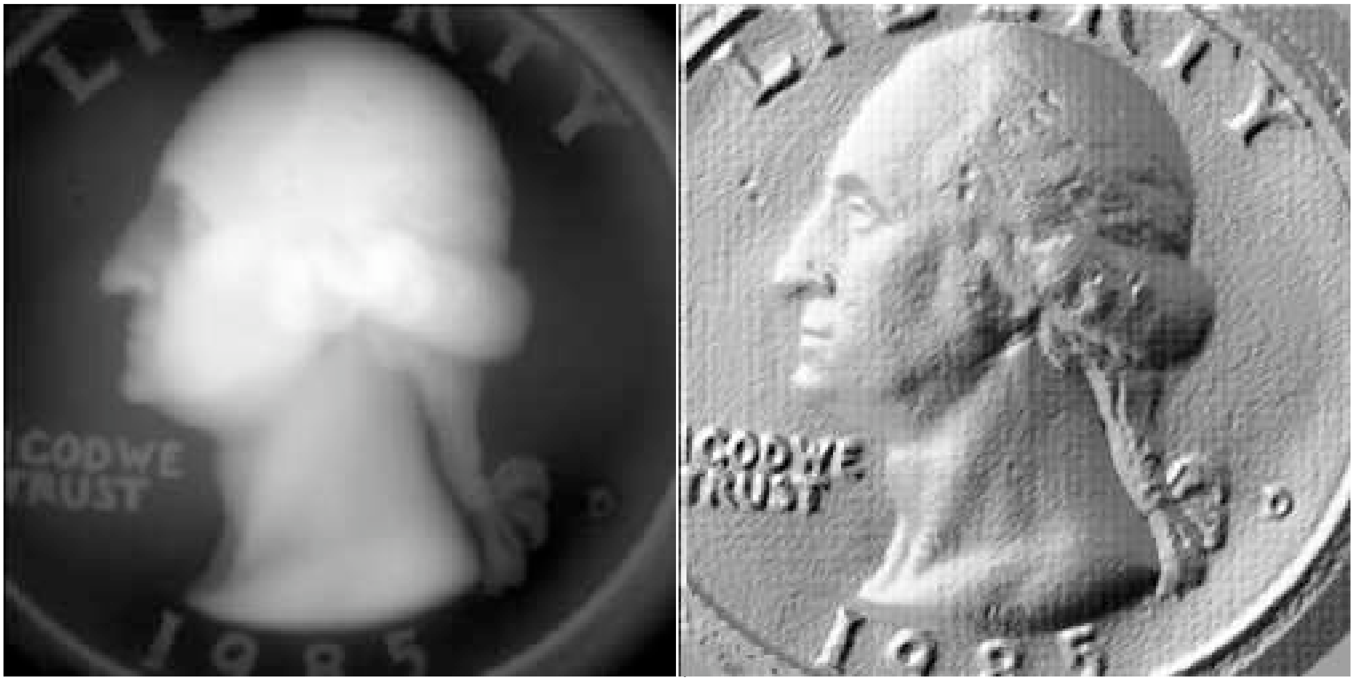
3.A.6. Pseudo-color, pseudo-3D, and other display tools

Human vision can distinguish hundreds of colors, but only 20-30 grey levels. The use of false- or pseudo-color look-up tables (CLUTs) makes small brightness differences visually evident, but can also break up the Gestalt of the image and should be used with care. Color tables can be applied by converting the image to indexed color mode and creating or loading a color table, or by converting the image to RGB color mode and using the **IP•Color–>Apply Color Table** to load the CLUT file from disk. These CLUT files are identical to the Photoshop indexed color tables (*.act) and can be created using the Photoshop color selection tools.

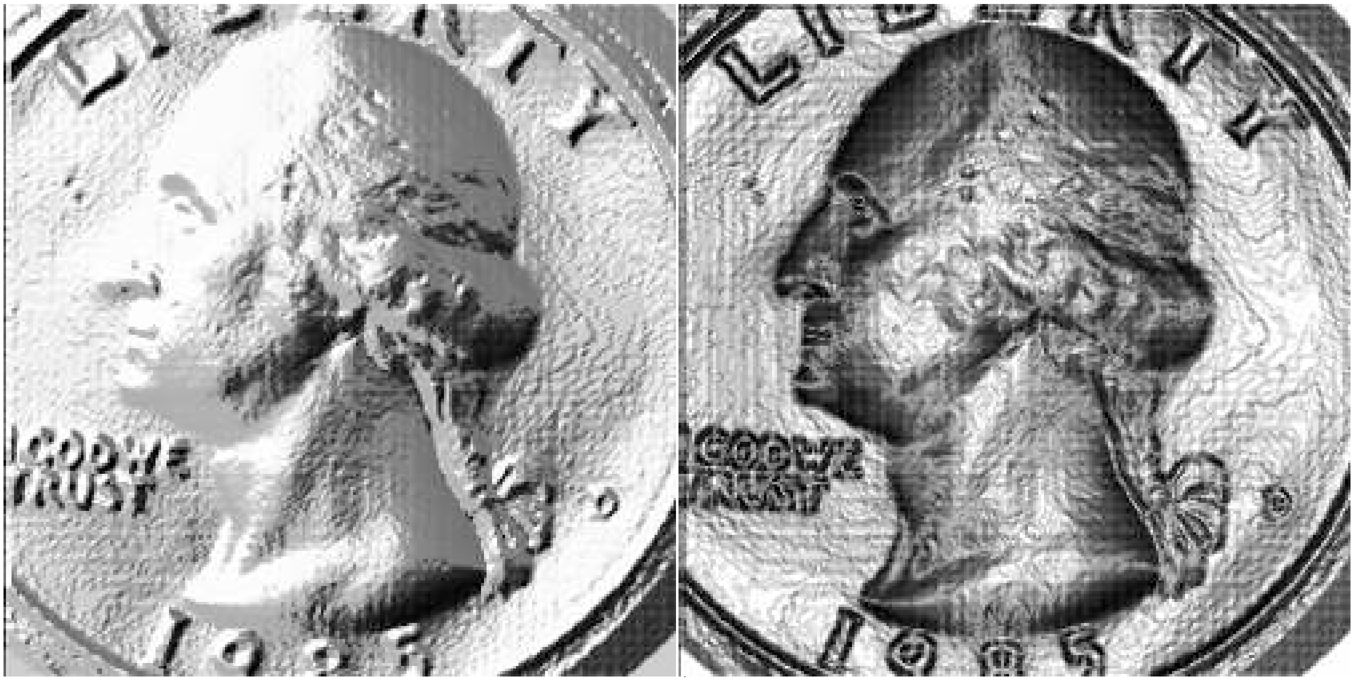


Original *Skin* image and the use of a CLUT with a spectrum of saturated colors

Because human vision is very experienced with surfaces, it is often useful to render an image as though it is a physical surface. Phong rendering with **IP•Graphics–>Render (Phong)** allows the position of the light source and the specularity of the “surface” to be adjusted to produce a wide range of results.

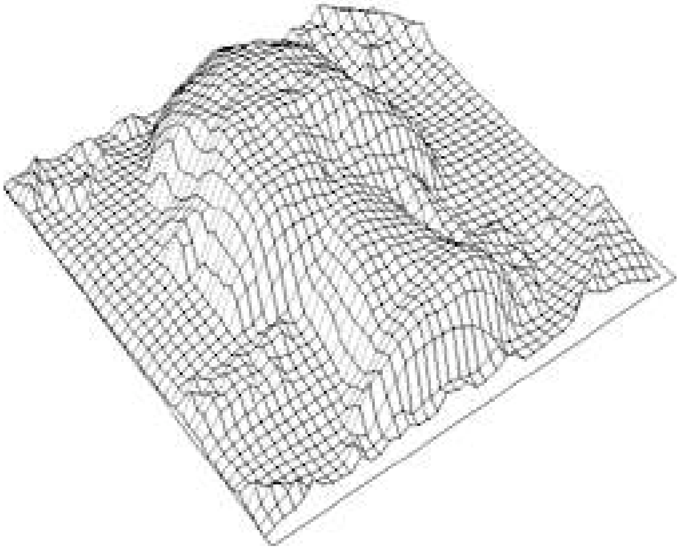


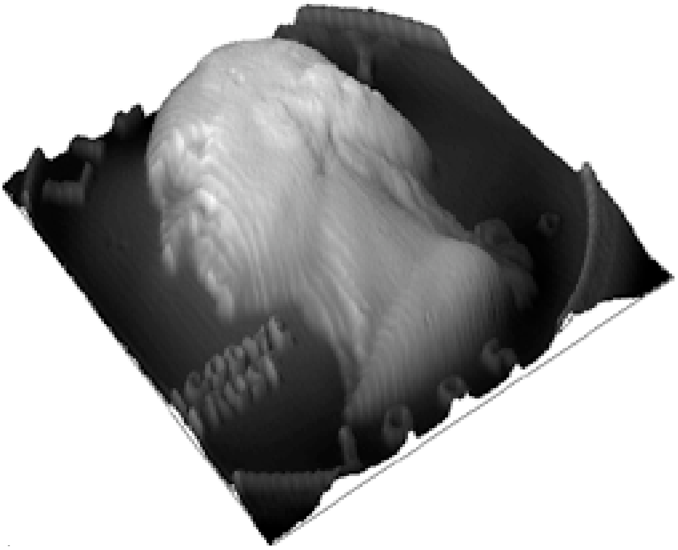
Original *Quarter* image, and Phong rendered with the light at 60 degrees elevation
in a compass direction of 300 degrees (northwest) and surface specularity = 0.8

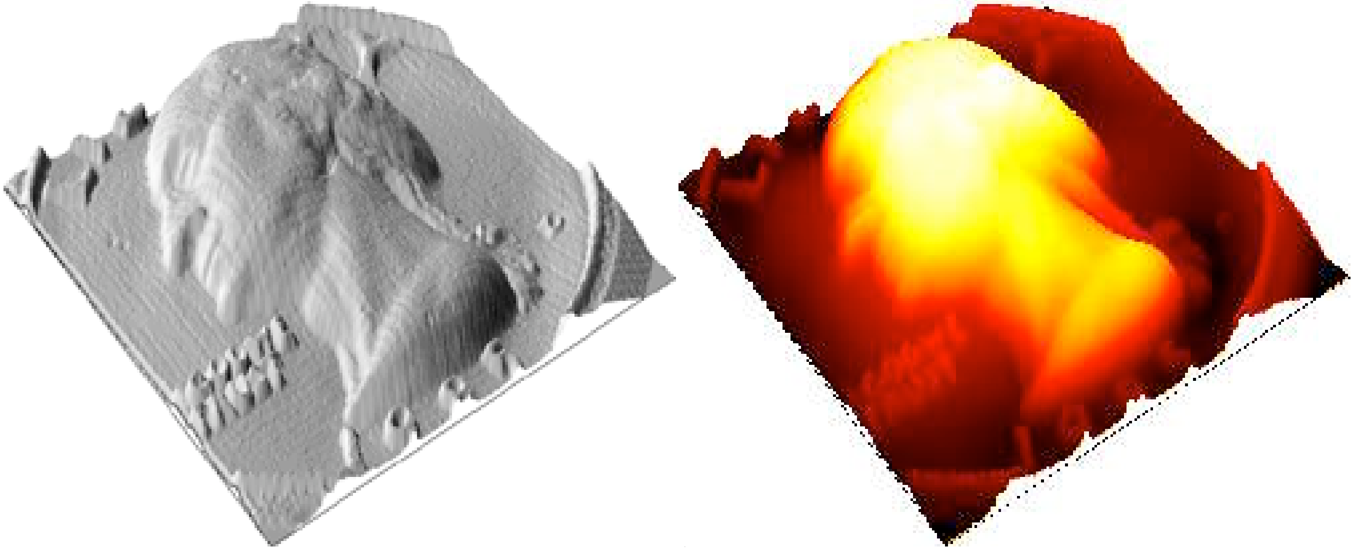


Phong rendering with the specularity = 1.0 and the light source at 87 and 90 degrees elevation

It is also possible to render the data as a perspective-corrected surface with an optional grid superimposed, or with the surface shown in photorealistic mode, or with the Phong rendered appearance superimposed. Place the image into the second image memory (**IP•2nd Image –> Setup**) and then use **IP•Surface Processing–> Plot 2nd as Surface** or –>**Reconstruct Surface with Overlay** to create a variety of representations.





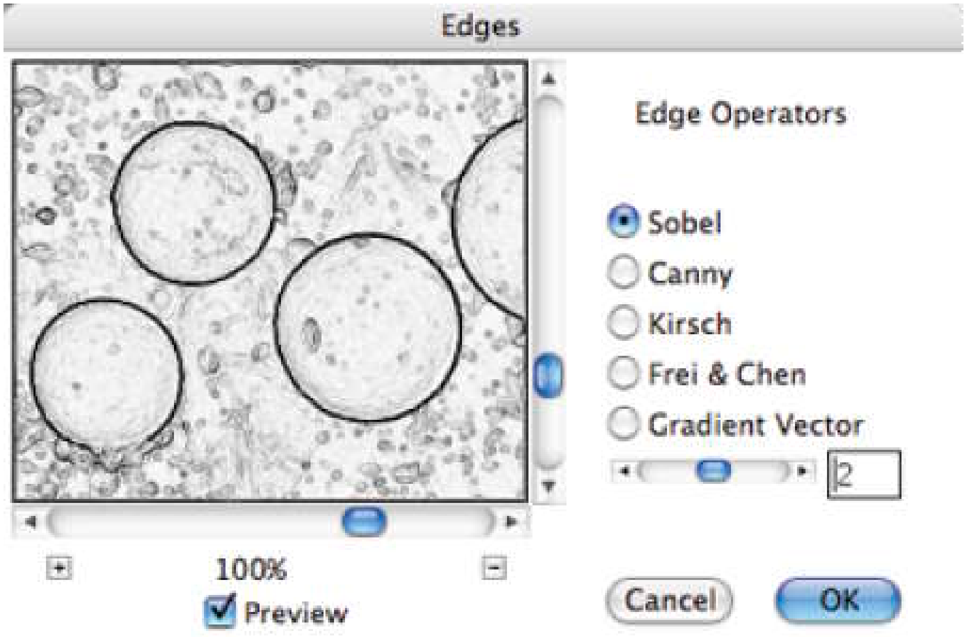


Examples of surface rendering with a grid, photorealistic surface modeling, superimposed Phong rendered surface, and a pseudo-color table.

3.B. Are feature edges important?

3.B.1. Edge enhancement with derivative operators

Locating edges in images has been a major area of algorithm development in image processing. Edges are used for many purposes. Thresholding the edges and filling provides a way to create an image of features whose centers are similar in brightness (such as typical SEM images of particles or pores). Thresholding and skeletonizing, described in sections below, can delineate boundaries between structures (measuring the length of these lines provides a stereological tool to determine the surface area in the 3D structure). This is particularly important when the boundaries are characterized by a change in brightness rather than any particular brightness value.



Edges

Edge Operators

● Sobel
○ Canny
○ Kirsch
○ Frei & Chen
○ Gradient Vector

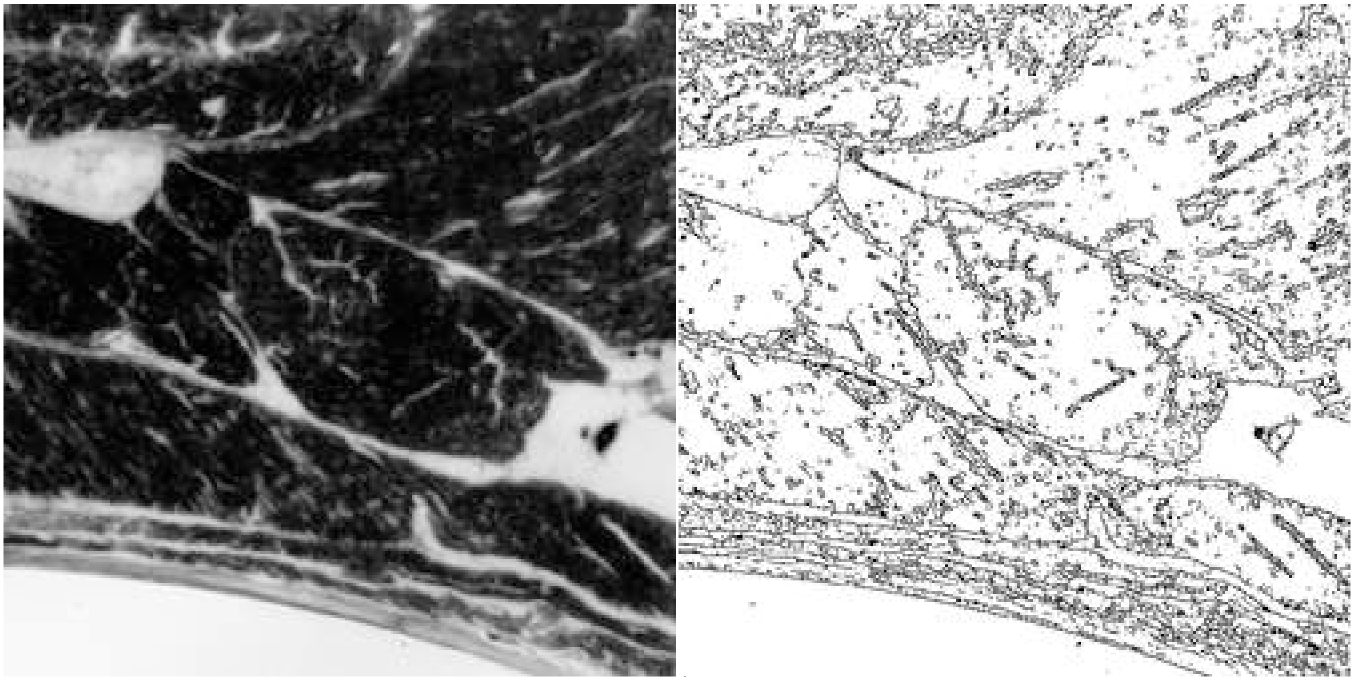
2

100%

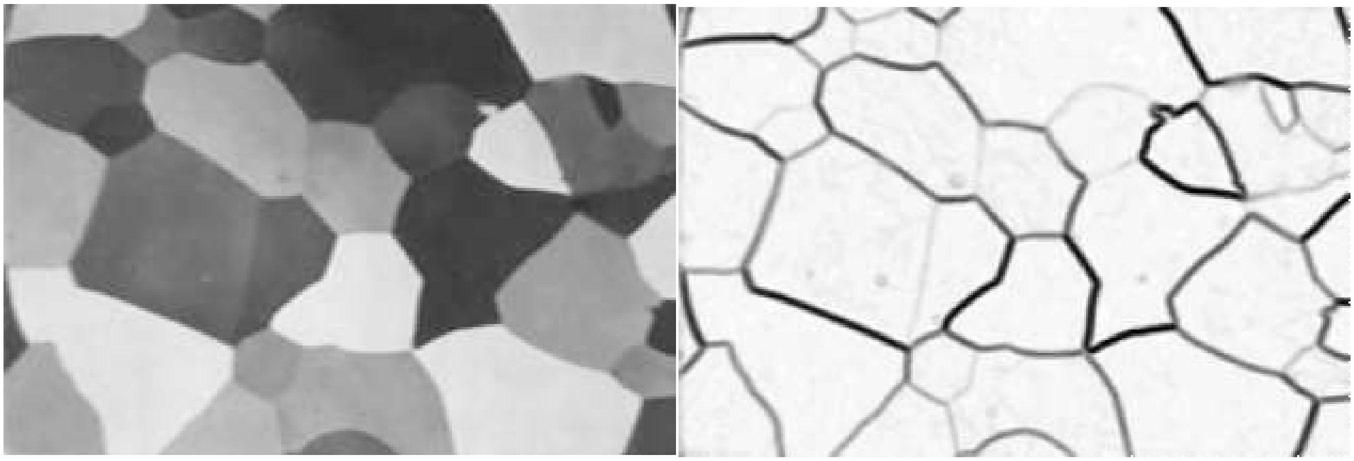
☑ Preview

The **IP•Process–>Find Edges** dialog with the *SEM_Part* image.

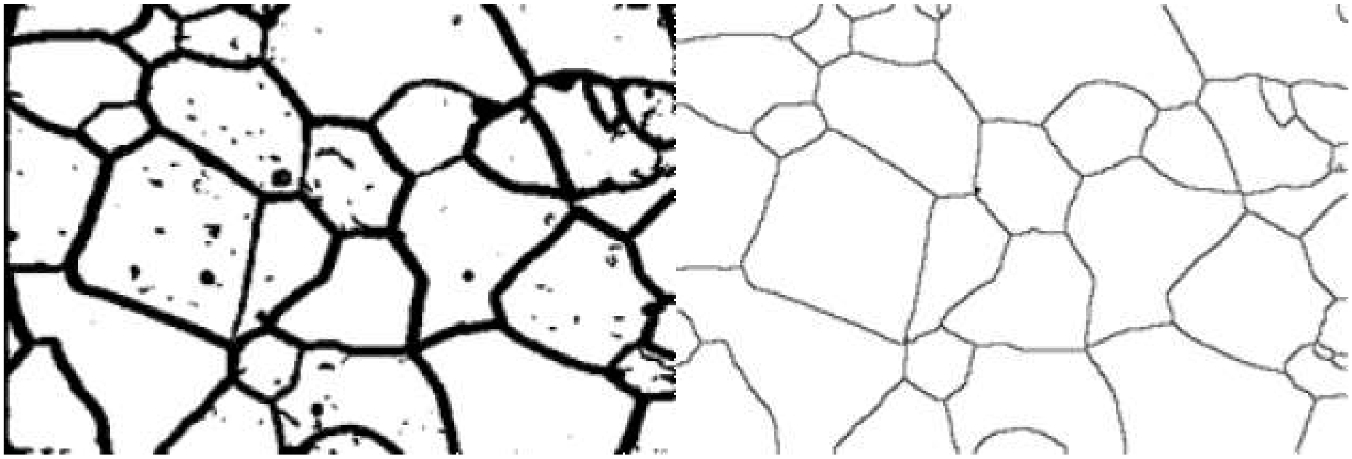
A variety of methods ranging from the simple Sobel to more advanced techniques like the Frei and Chen use multiple convolution operators. Thinning down the edge response to single-pixel width (the Canny filter) marks the most probable location of the edge. These routines are selected in the **IP•Process –> Find Edges** plug-in. Nonlinear operators (e.g., **IP•Rank–>Range** is the difference between brightest and darkest values in a neighborhood) and statistical operators (e.g., **IP•Process–>Variance**) are also useful.



Original *Beef* image (fragment) and Canny edge filter



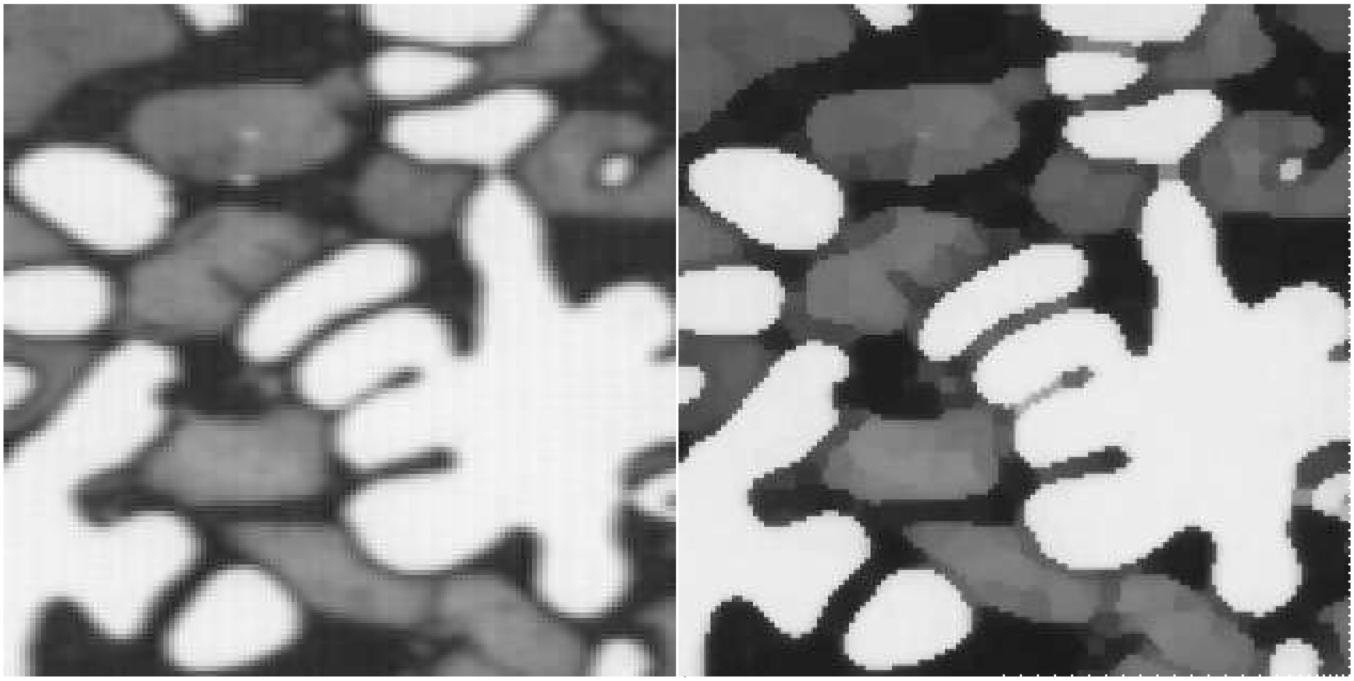
Original *Gr_Alum* image and the result of the variance operator



Thresholding and skeletonizing produces lines that mark the grain boundaries

3.B.2. Increasing edge sharpness and region uniformity

Boundaries between structures or regions may be marked by either lines or by steps, localized changes in color or brightness. However, in real images these steps are often blurred, either by sample preparation, optical resolution, or the finite size of pixels. Statistical procedures such as maximum likelihood techniques (**IP•Process–>Sharpen Steps**) can reproducibly assign doubtful pixels to regions and create abrupt transitions that facilitate thresholding and segmentation.

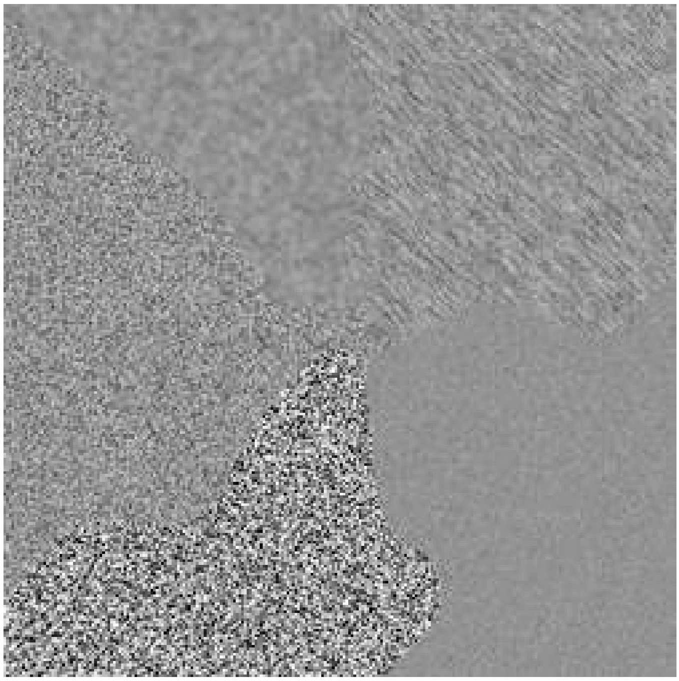


Dendrite image (enlarged fragment) and the result of applying a maximum likelihood operator

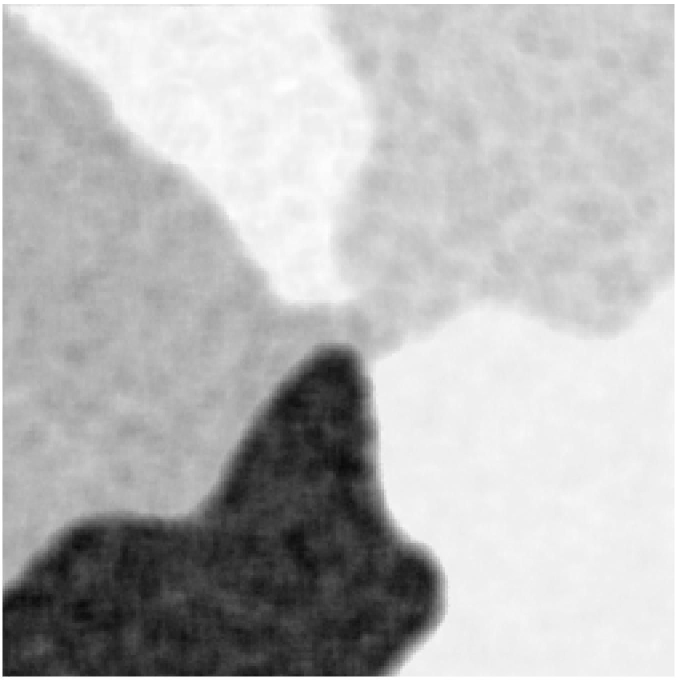
3.C. Converting texture and directionality to grey scale or color differences

In many images, structures are discernible visually based on textural rather than brightness or color differences. Processing tools such as the Sobel orientation {**IP•Process –> Orientation (Sobel)**} operator or a wide variety of local texture measurements based on entropy, fractal dimensions, statistical properties, etc., can convert these variations to brightness differences for measurement or thresholding.

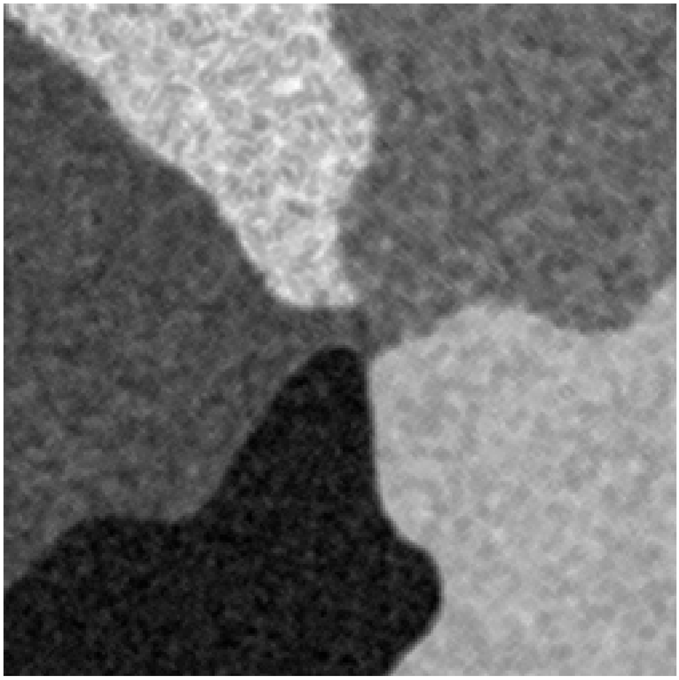
Both the range and variance operators (**IP•Rank–>Range** and **IP•Process–>Variance**) can be used for texture recognition purposes by using a suitably large neighborhood. In addition, the **IP•Process–>Texture (Spatial)** and **–>Texture (Fractal)** plug-ins implement several useful algorithms. The spatial algorithms are based on co-occurrence matrices and entropy calculations, while the fractal measurement uses the slope and intercept of the (log difference) vs. (log distance) data. Because the word "texture" covers such a wide range of human visual experiences, it is often necessary to try several different tools to find one appropriate to a specific application.



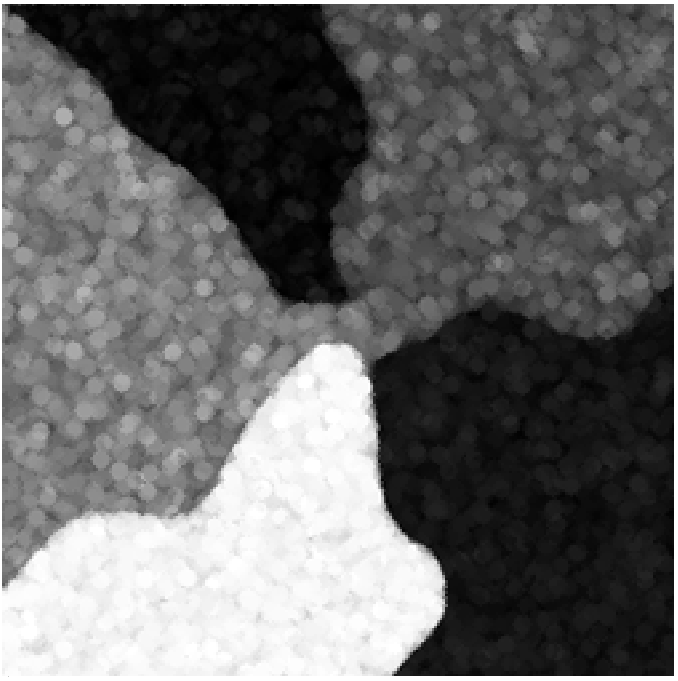
Original *Texture1* image"



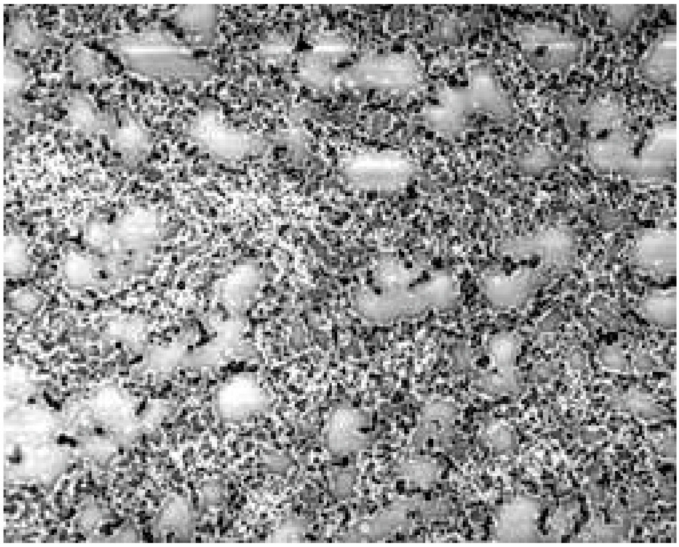
Variance (radius = 5 pixels)



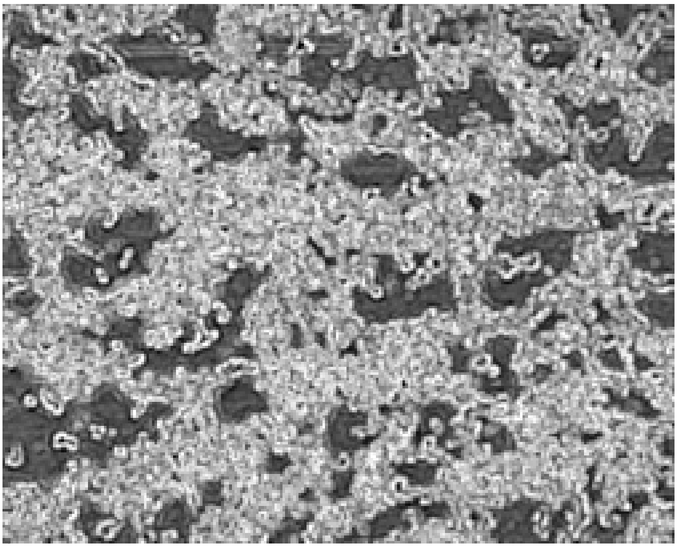
Entropy (radius = 3.5 pixels)"



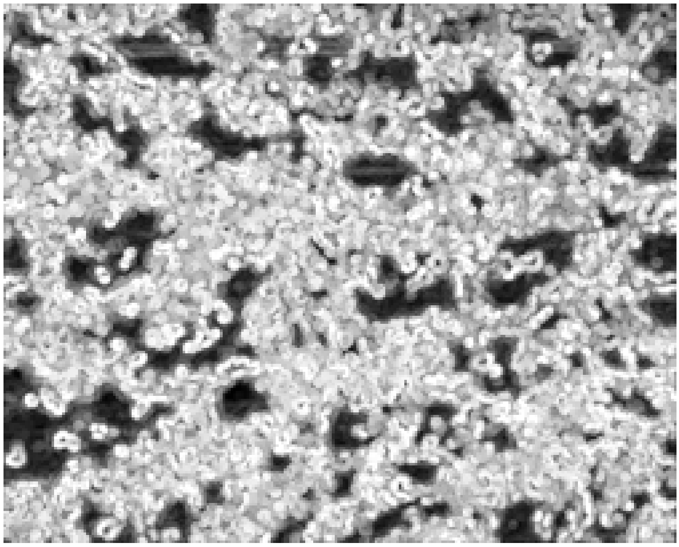
Range (radius = 4 pixels)



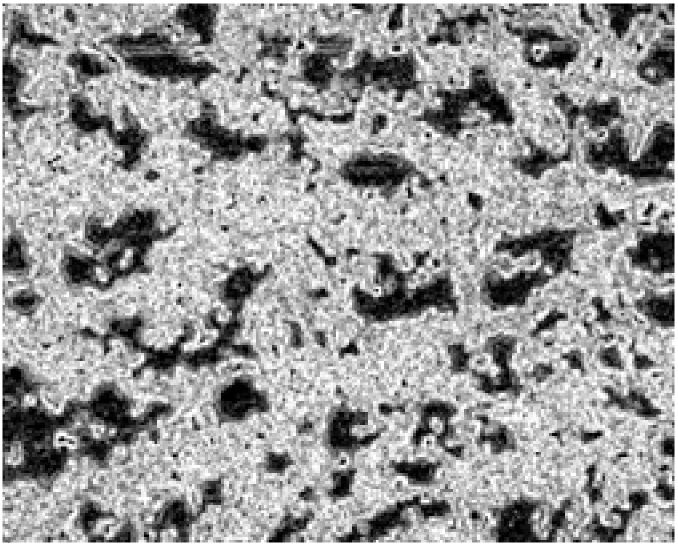
Original *CurdWhey* image"



Fractal texture intercept (radius = 5 pixels)



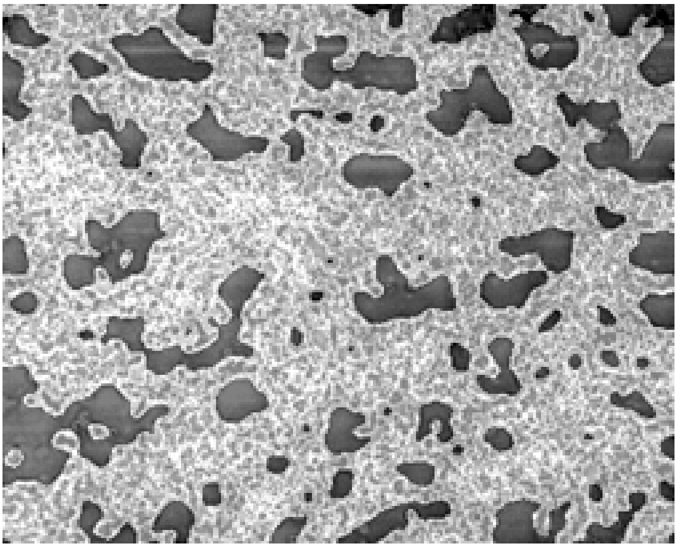
Range (radius = 2.5 pixels)"



Spatial contrast texture operator

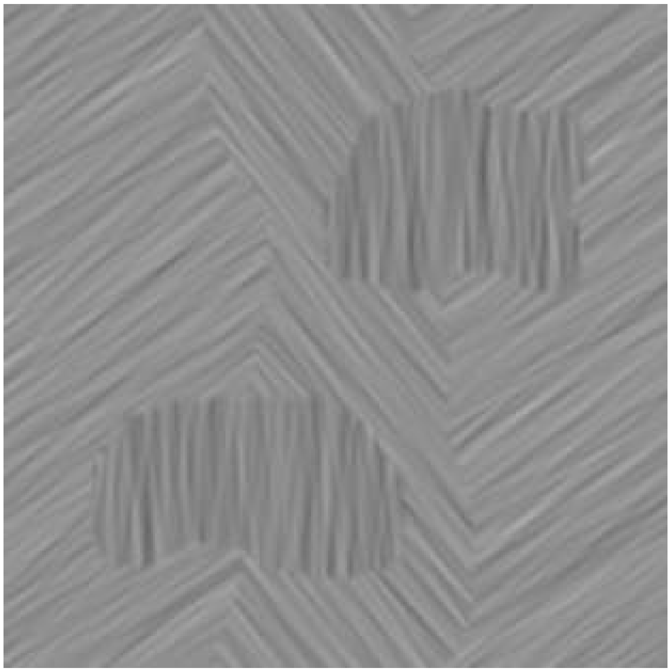
The result from using the texture operators can then be thresholded to delineate the structure of interest. In the *CurdWhey* example, the result of the range operator shown above was smoothed (Gaussian blur, radius = 2.5 pixels), automatically thresholded, and the result superimposed on the original image to verify the result. This was done by using the Layers capability of Photoshop, and setting the transparency of the binary image to 50%. An equivalent result can be obtained by adding the two images together.

Thresholded range image"

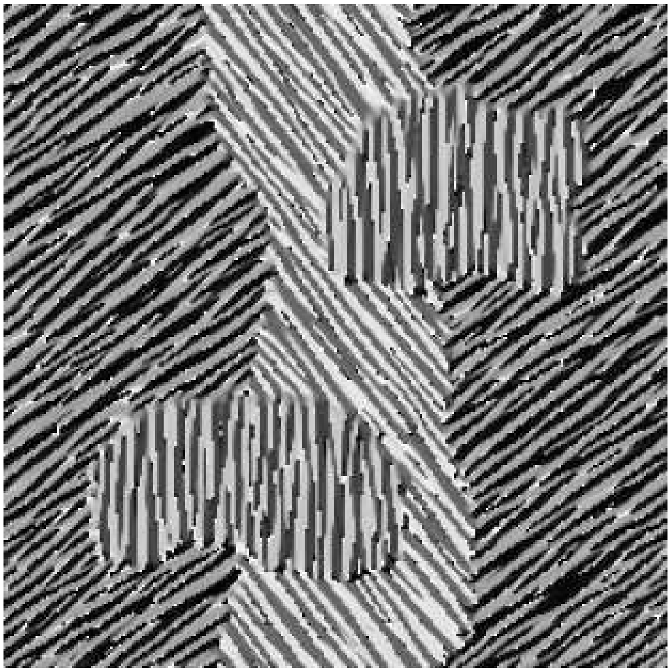


Superimposed on original *CurdWhey* image

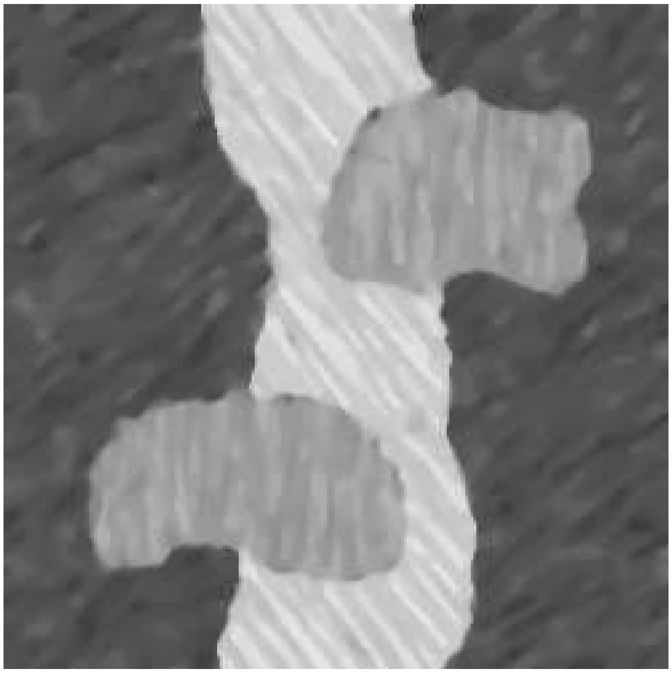
The most widely used and generally successful orientation measurement tool is based on the Sobel brightness gradient. The magnitude of this gradient was introduced above as an edge delineation tool. The angle of the vector can also be used, by assigning grey scale or color values to the angle. In the first example, the visually discernible regions in the image have the same brightness and texture but different orientations. The result of the **IP•Process–>Orientation (Sobel)** plug-in shows unique pairs of grey scale values in each region, which represent 180 degree differences in vector orientation (and thus are 128 grey scale values apart). This image can be thresholded most easily after applying a color table (FoldGrey.act) that duplicates the grey values assigned to the 0-180 and 180-360 degree range, as shown. In the example, the three well-separated peaks in the histogram correspond to the three region orientations in the original image. The horizontal axes covers the range from 0 to 180 degrees.



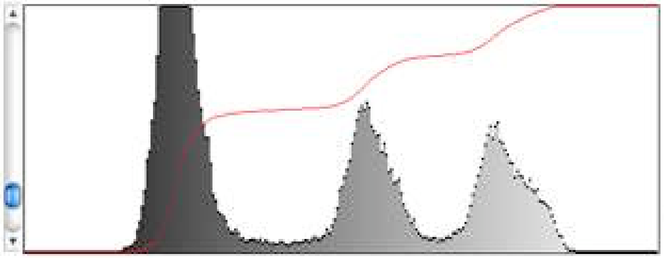
Original *Texture2* image"



Orientation operator applied



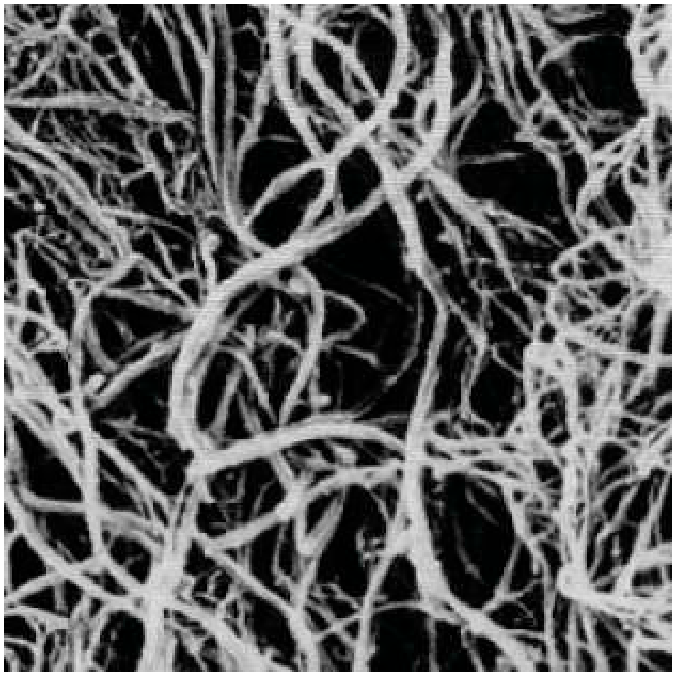
!

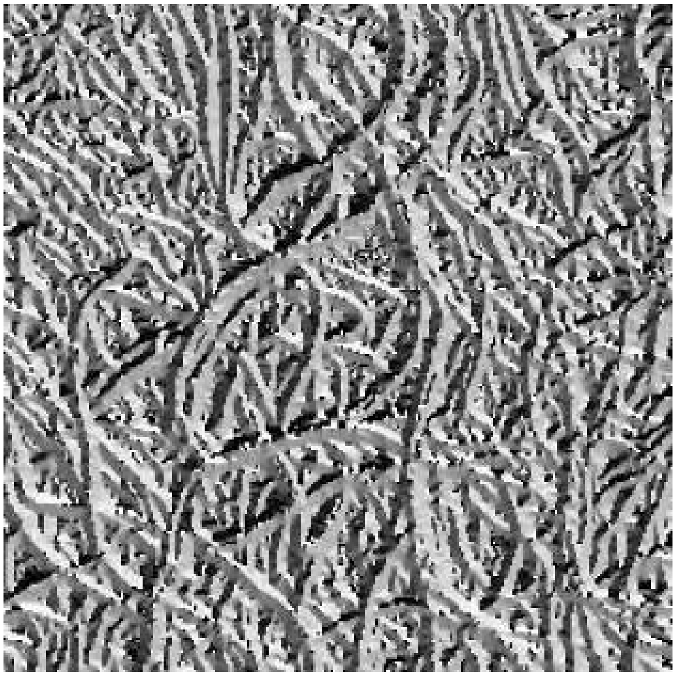


FoldGrey CLUT applied and median filtered"

Histogram showing three distinct peaks

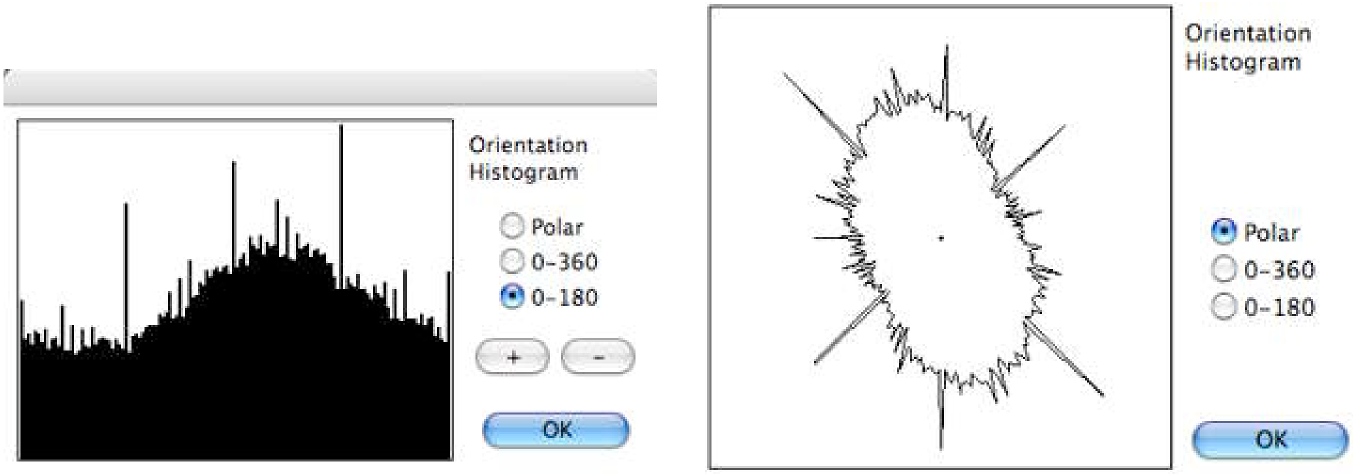
The orientation of fibers, scratches or other similar structures can be directly measured by using the histogram of the image that results from the application of the orientation filter. In the example, the relative length of curved fibers having each orientation is revealed by the corresponding number of pixels recorded in the histogram, which for convenience of interpretation is plotted as a rose plot. The preferred orientation of the fibers is immediately apparent. (The spikes at multiples of 45° are an artefact of the procedure that may be useful for inspection of the plots.)





Original *Collagen* image (fragment)"

Orientation operator applied


Orientation Histogram

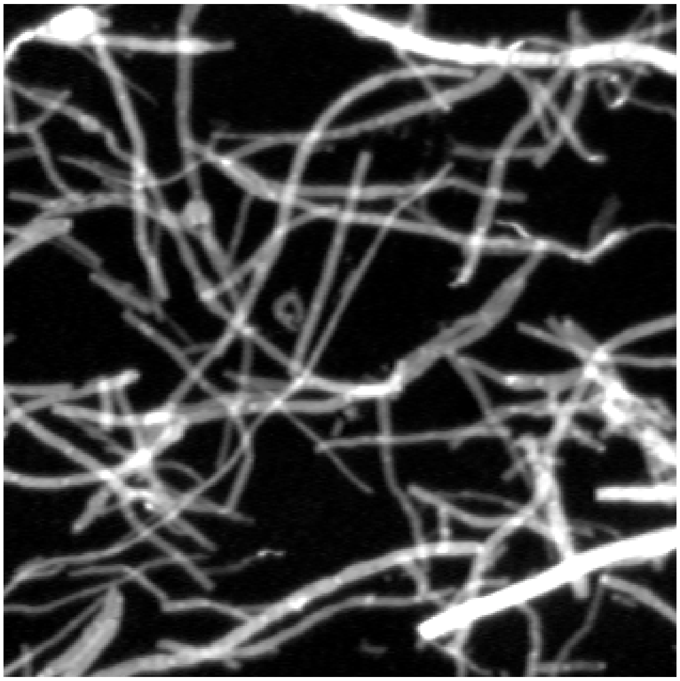
○ Polar
○ 0-360
● 0-180

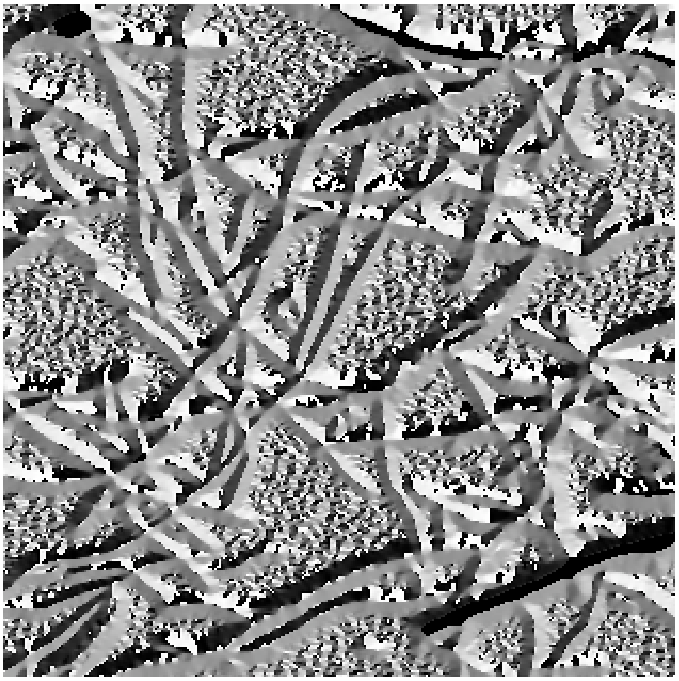
Orientation Histogram

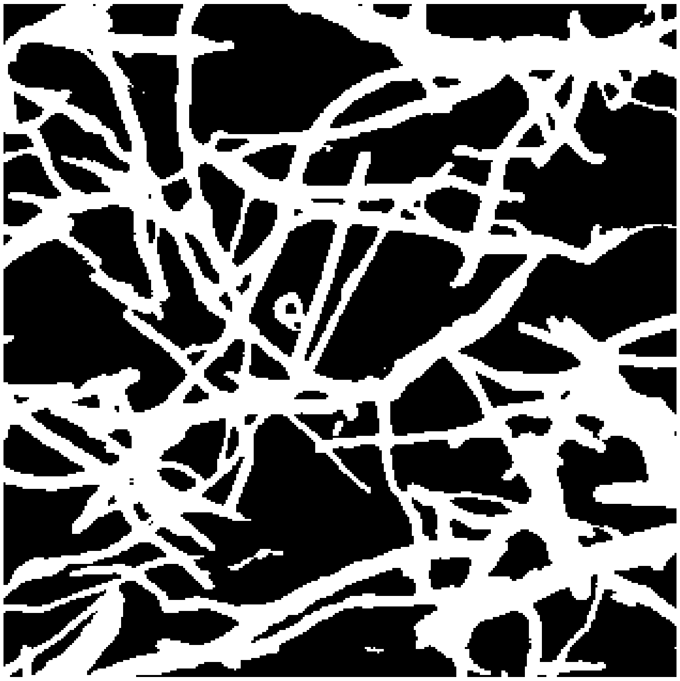
● Polar
○ 0-360
○ 0-180

Histograms of the orientation image: left - 0-180 degree linear plot, right - rose plot

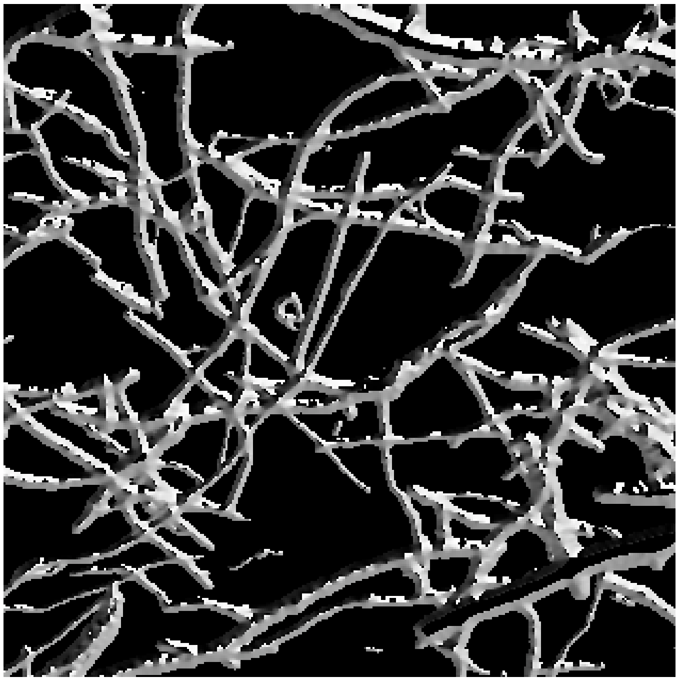
If the fibers do not cover the entire image, it is necessary to restrict the measurement to just the fibers. One way to do this is to duplicate the original image and threshold (as discussed in Section 4) the fibers. This image can then be applied as a mask to the orientation image as shown in the example, setting all of the pixels in the background to black. The procedure is to place the binary image into the second image memory (**IP•2nd Image–>Setup**), select the orientation image, and use the **IP•Math–>Keep Darker Values** routine.


Original *Fibers* image (fragment)"

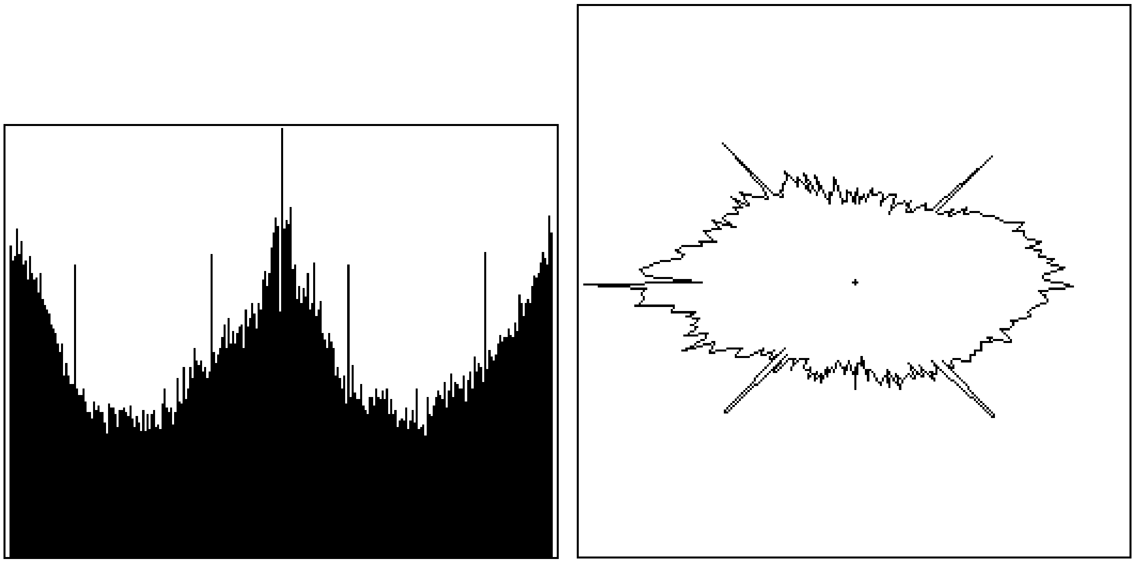

Orientation result



Thresholded binary image of background"

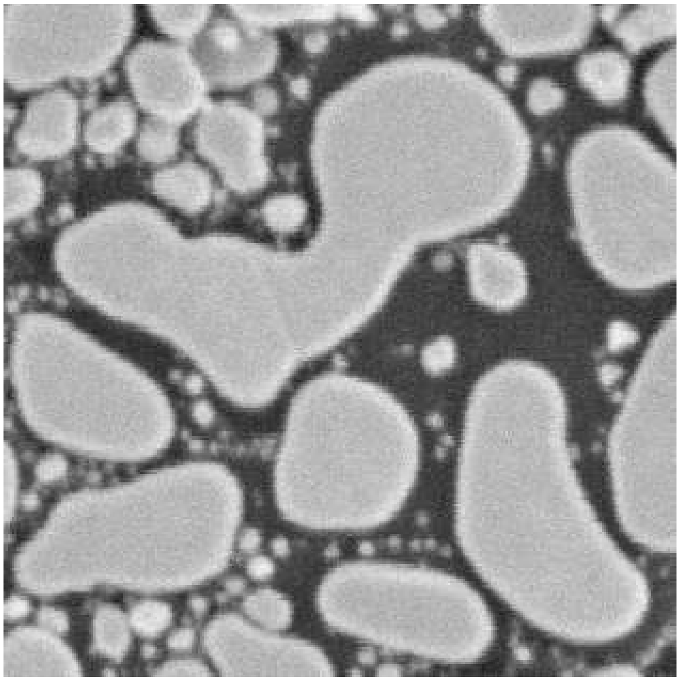


Combination of orientation and binary images

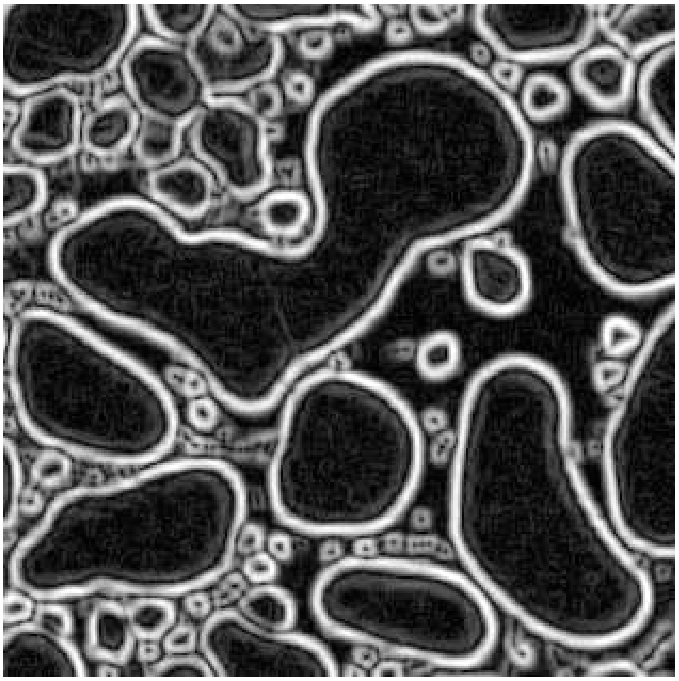


Histograms showing the fiber orientation: left - 0-360 degree linear plot, right - rose plot

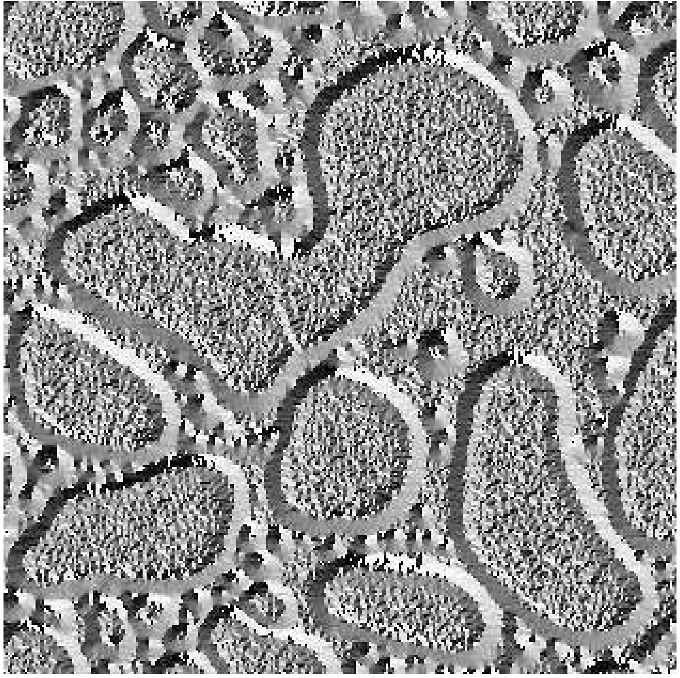
Another useful way to represent the orientation information is to assign colors to the grey scale values, since the hue values around the color wheel can effectively represent angles. In the example shown, the edges in the original image are first outlined (**IP•Process–>Find Edges–>Sobel**, followed by inverting the image with **Image–>Adjustments–>Invert** to make the edges bright). This image is then converted to RGB mode so that it can display color (**Image–>Mode–>RGB Color**). Then the orientation operator (**IP•Process–>Orientation**) is applied to a second copy of the image. These angle values are then transferred to the hue channel of the edge image, by first placing the orientation image into the second image memory (**IP•2nd Image–>Setup**) and then selecting the edge image and choosing **IP•Color–>Transfer Channel–>Hue**. The colors associated with each orientation angle of the edges (or any other structure) are made visually apparent.



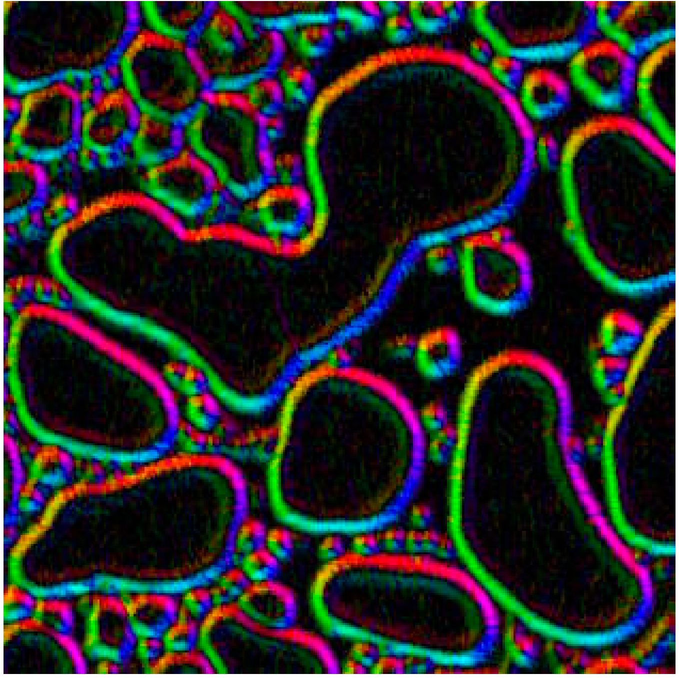
Original *Au_Resn* image (fragment)"



Edges delineated



Angle values calculated"



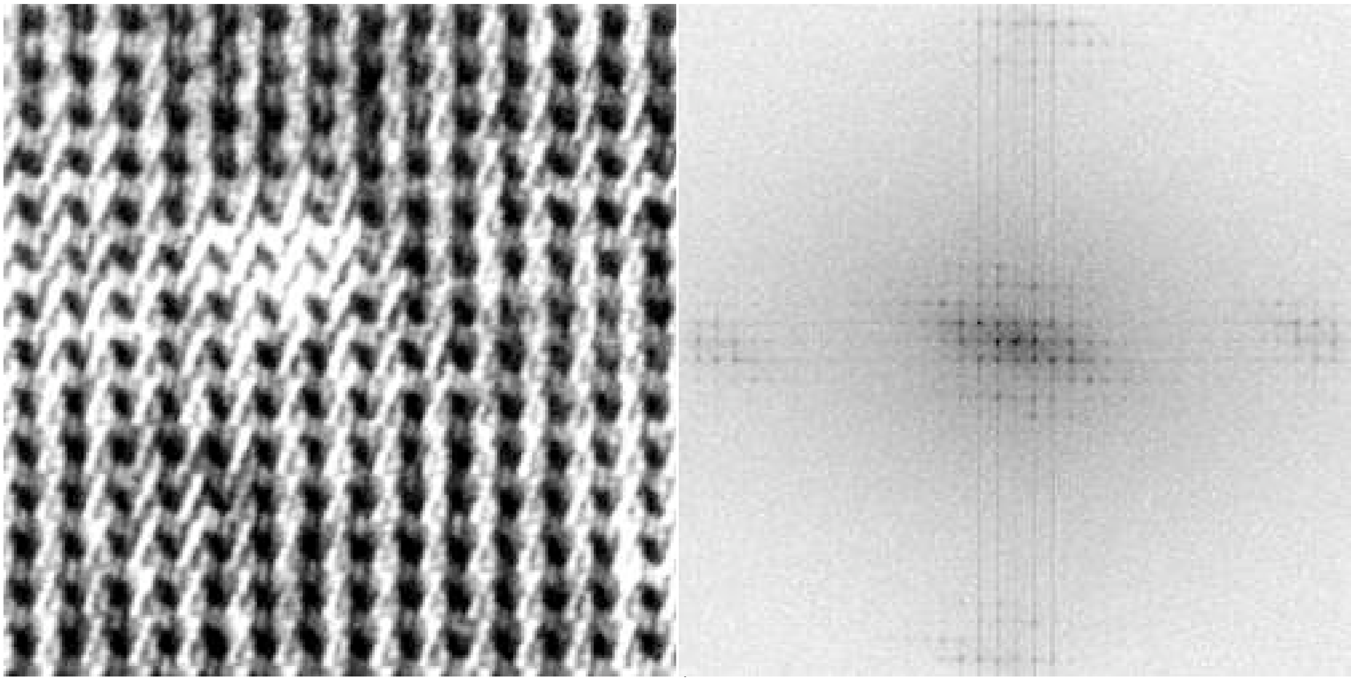
Angle values assigned to the hue channel

3.D. Fourier-space processing

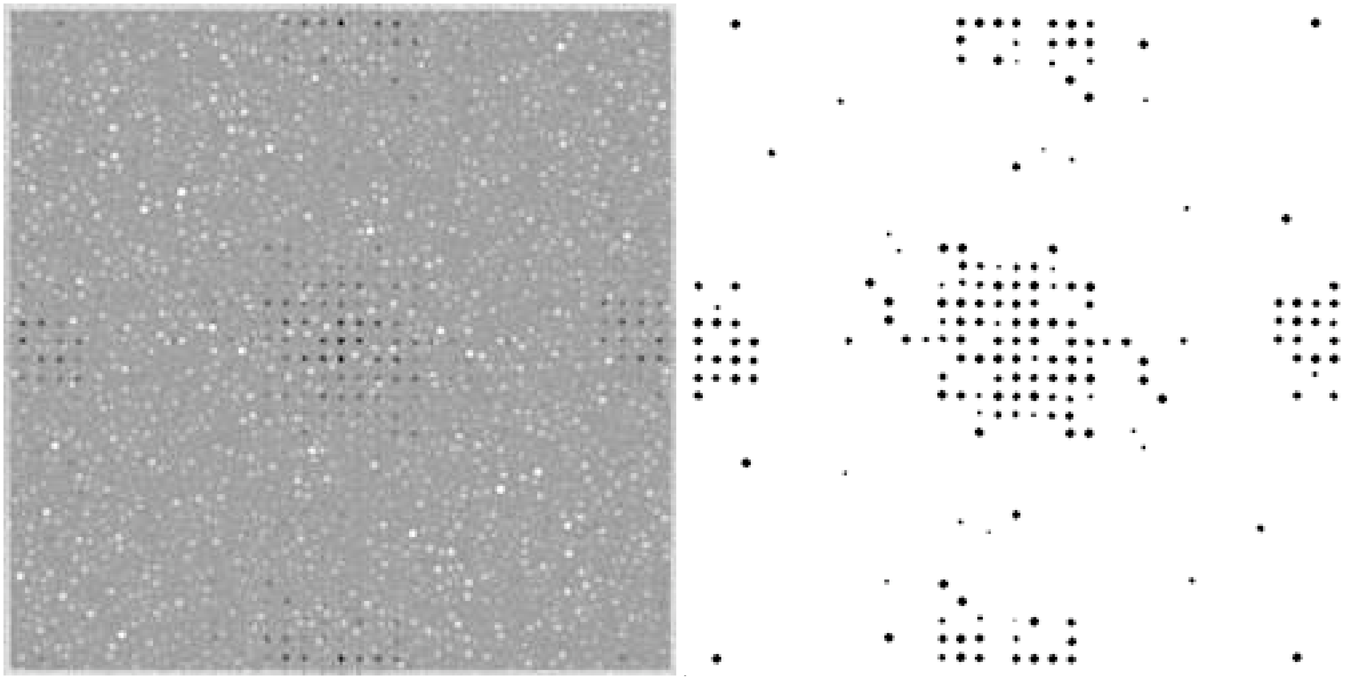
3.D.1. Isolating periodic structures or signals

The FFT power spectrum can facilitate the selection of regular structures in the image and their measurement. Besides its use for removing periodic noise from images (e.g., electronic interference, vibration, or halftone printing patterns), this facilitates averaging and measuring repetitive structures (e.g., TEM images of atomic lattices). Note that the FFT routine used in the plug-ins requires images to have dimensions that are a power of 2 (256, 512, ...). For images of other sizes, either create a square ROI of these dimensions within the image (set Style = Fixed Size), or enlarge the canvas size (**Image–>Canvas Size**)

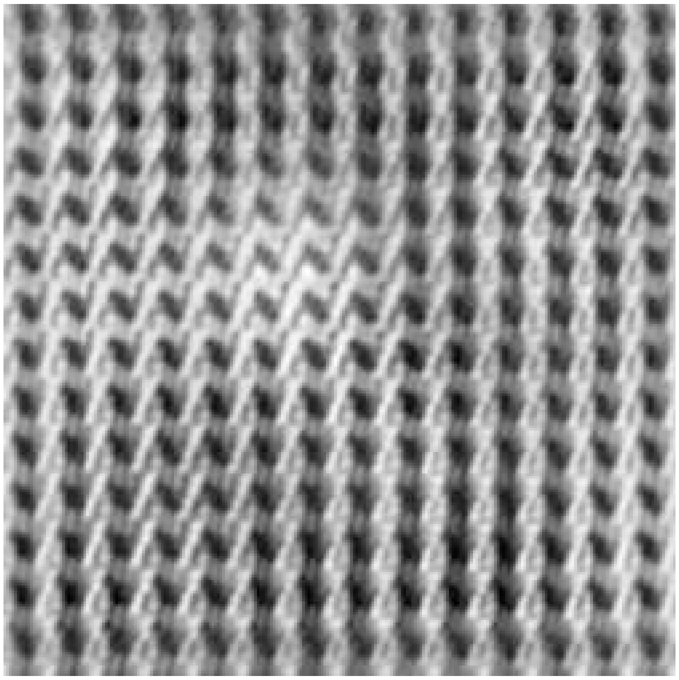
The same procedure illustrated in Section 3.A.5 above can be used to keep periodic structure and eliminate random noise, as shown in the following example. This is an effective way of averaging together all of the repetitions of the same structure in the original image. The top hat filter was used to automatically locate all of the spikes in the FFT power spectrum of the original image, and thresholded to create a mask or filter that keeps those frequencies and eliminates others.



Original *Mullite* image and its FFT power spectrum

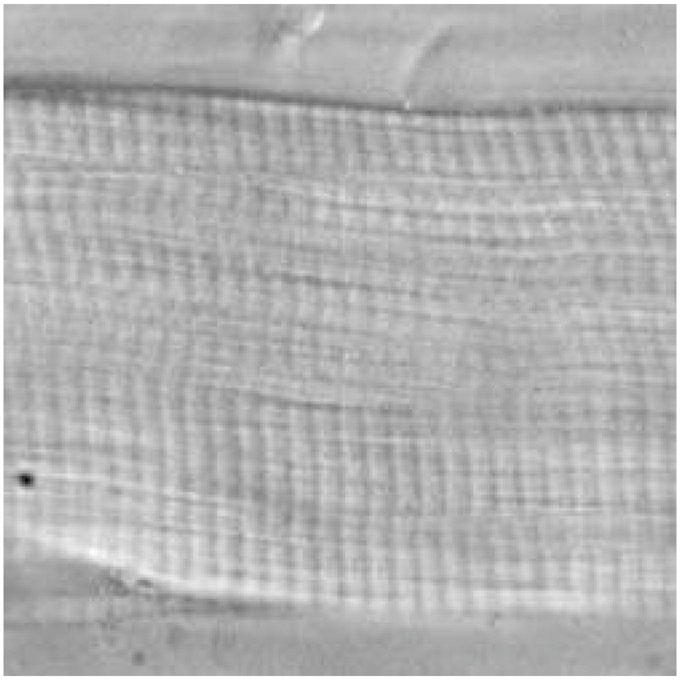


Application of a top hat filter to the power spectrum, and a mask that keeps the periodic frequencies

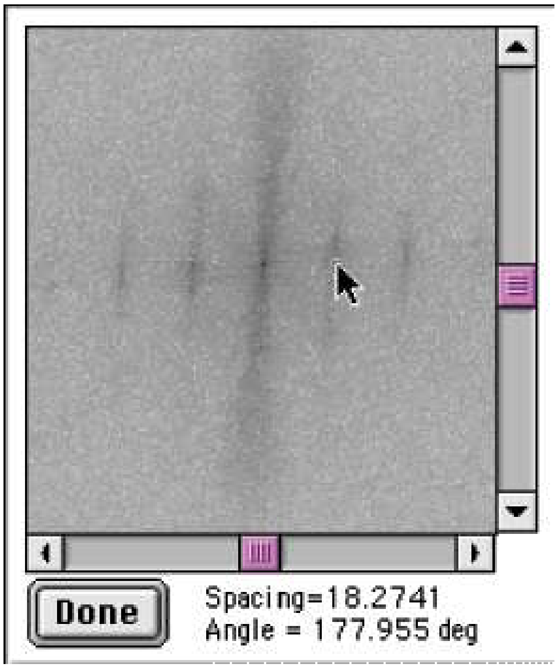


Result of using the mask with an inverse FFT

Measurement of spacings can also be performed directly with the power spectrum. As shown in the example, the lowest frequency (smallest radius) spike in the Fourier transform of the muscle image corresponds to the spacing of the z bands (spikes at multiples of this frequency are related to the shape of the bands' intensity profiles). This radius can be converted directly to the spacing (equal to the image width divided by the radius) using the **IP•FFT–>Show Fourier Values** function. If the original image had been calibrated in µm, as shown in Section 6.A.1, the value would be converted rather than being shown in pixels.



!



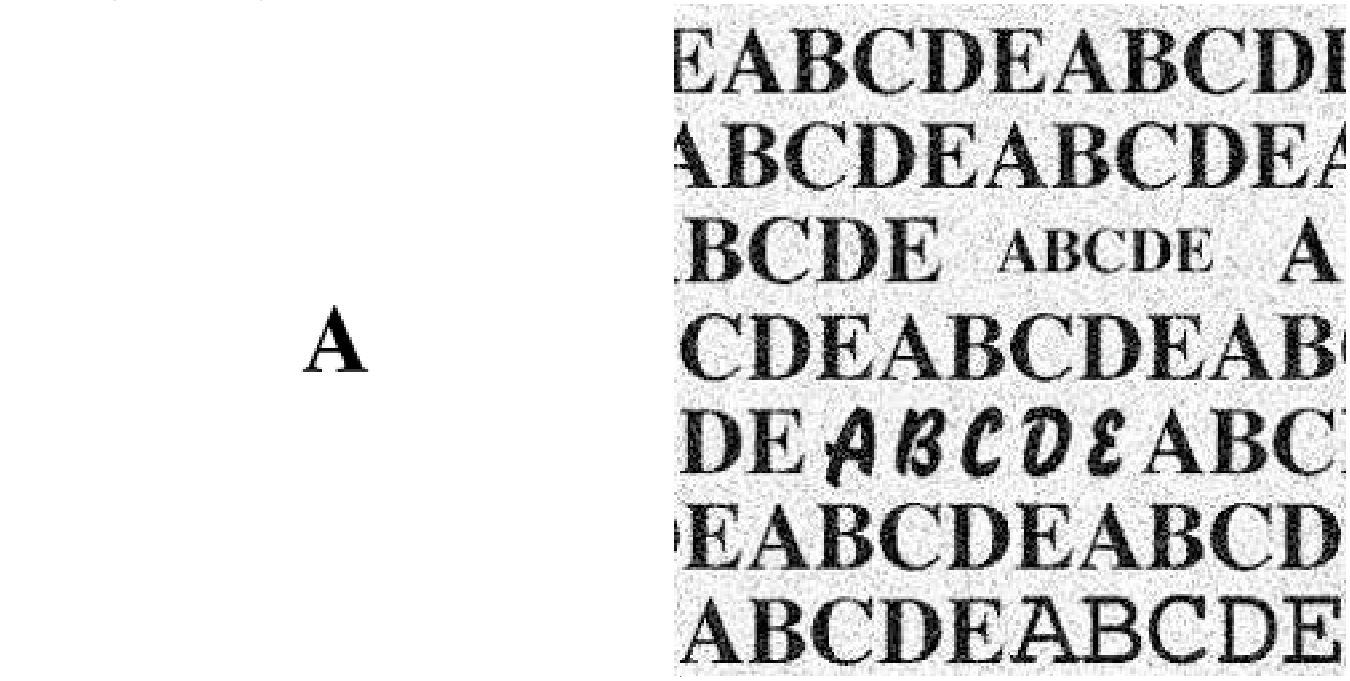
```
Spacing=18.2741
Angle = 177.955 deg
```

Original *Muscle2* image showing z-bands"

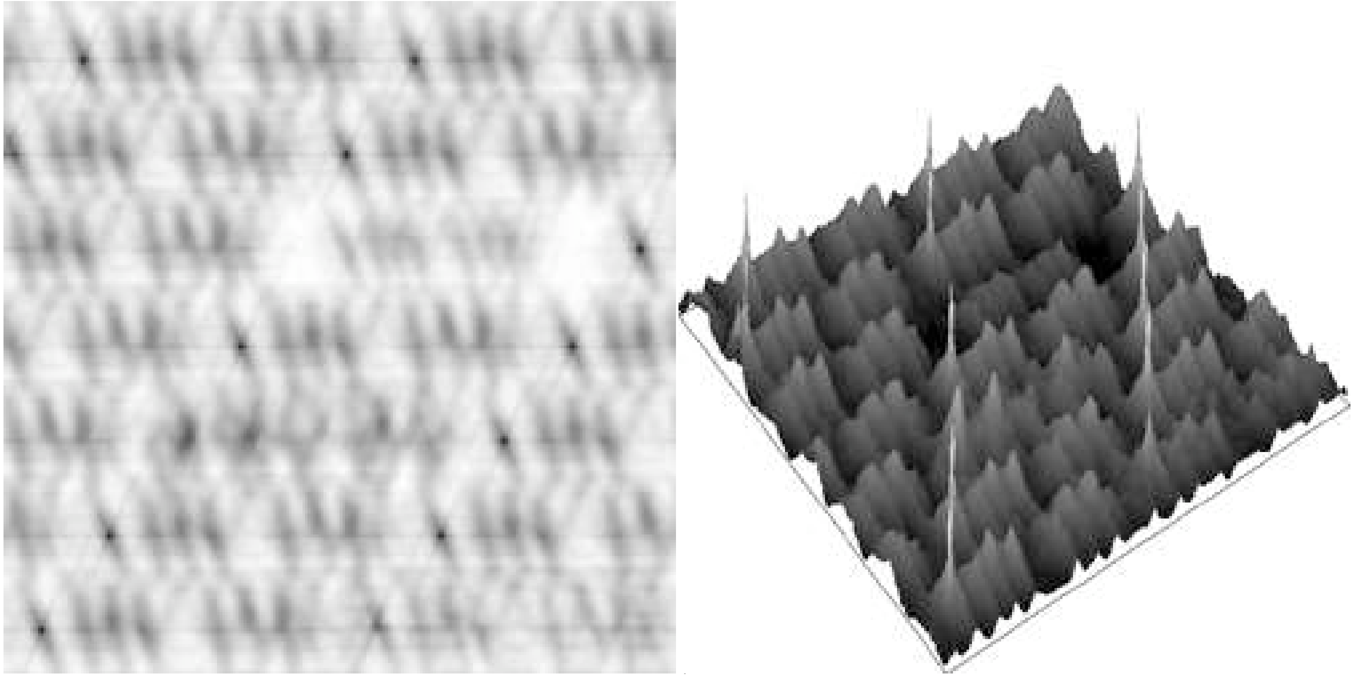
Measurement of the Fourier power spectrum

3.D.2. Location of specific features

Cross-correlation with a target image is equivalent to sliding the target across the image while measuring the degree to which the pixel patterns match, and is a powerful object finder. The example illustrates the procedure. A target image is placed in the second image memory (**IP•2nd Image–>Setup**), and then the image of interest is selected and the **IP•Fourier–>Cross Correlation** function chosen. The two images do not need to be the same size (and are not restricted to the power-of-two dimensions required by the forward and inverse FFT plug-ins). The resulting image has spikes that locate the features similar in shape and contrast to the target, and this can be thresholded, or a top hat filter applied as required. The surface rendering shows why these are called “spikes.”

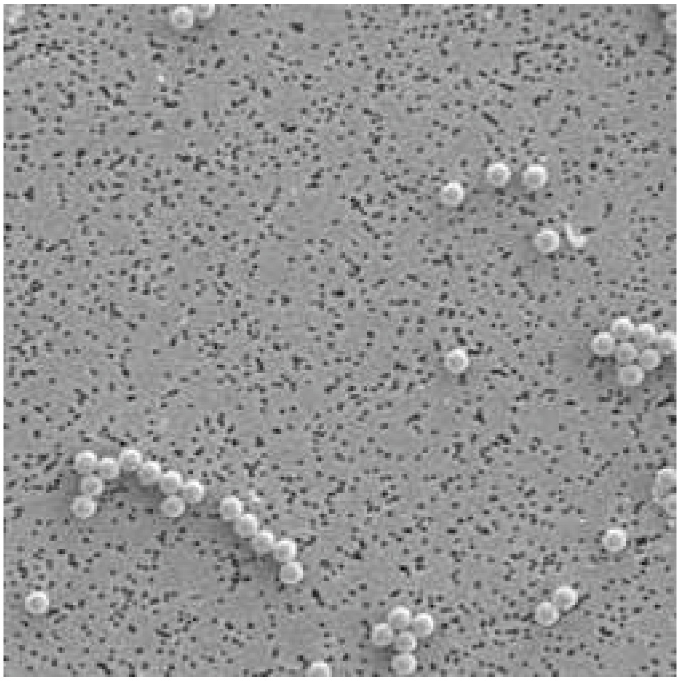


The target image (*Text_2*) and the image to be searched for the same shape (*Text_1*)

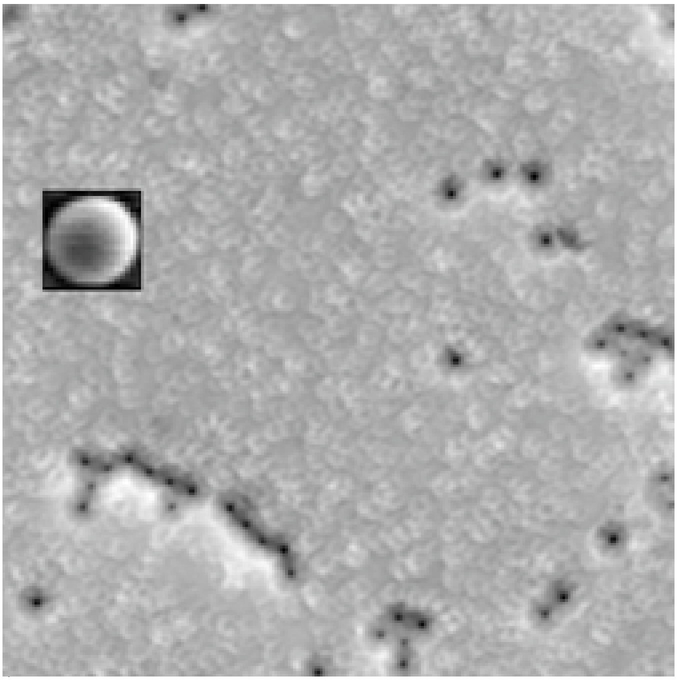


Cross-correlation result marking the location of the target shape

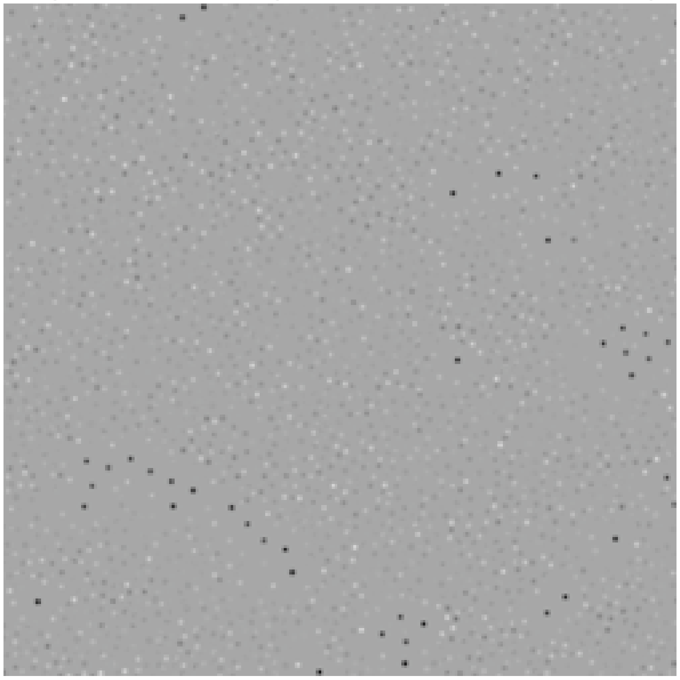
The next example shows a more realistic application, the counting of particles that are partially dispersed on a noisy background (a Nuclepore filter). There is also debris present, and particles adjacent to others are different in brightness than isolated ones. Manual selection of several representative particles and averaging their images together produces a suitable target. Cross-correlation and a top hat filter locate each of the particles, so that thresholding and counting (**IP•Measure Features–>Count**) can be performed. If the thresholded image of the particle locations is then convolved with the original target (**IP•Fourier–>Convolution**), an image showing the particles without the background or debris is obtained.



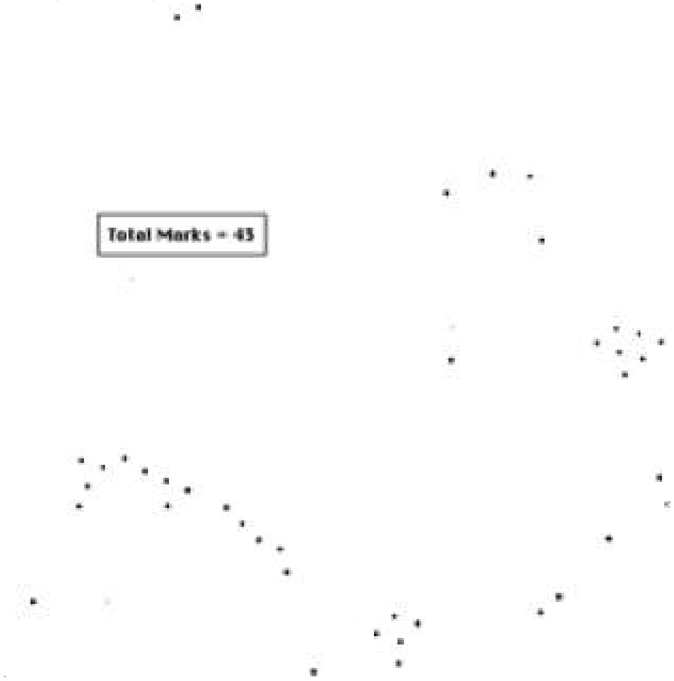
Original *Count1* image"



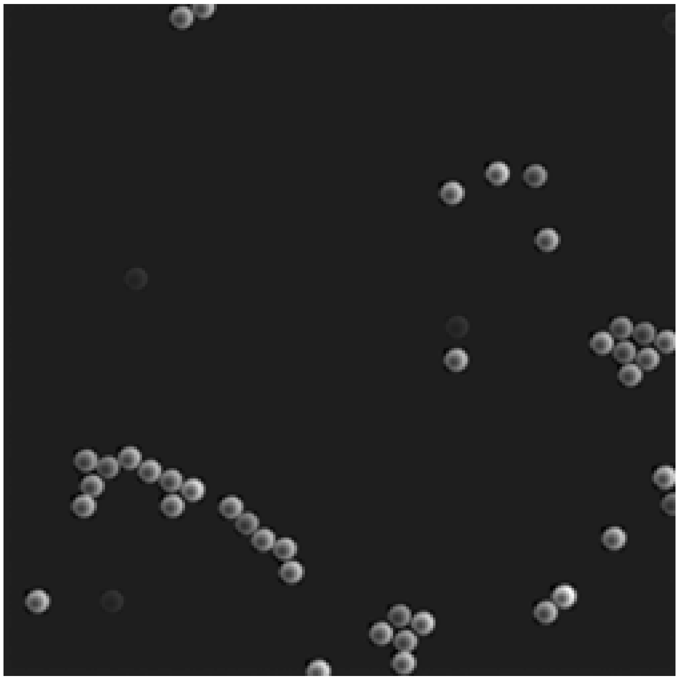
The averaged target image (inset) and the cross-correlation result



Application of a top hat filter"

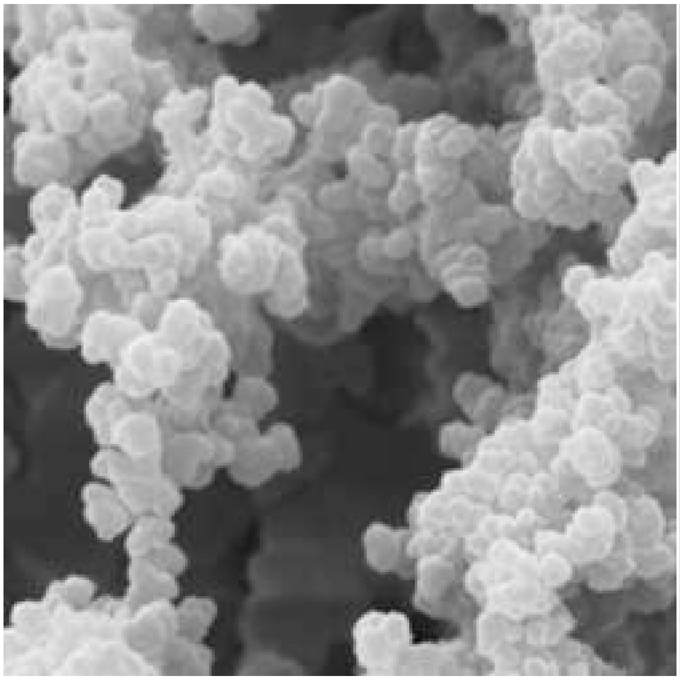


Thresholding and counting

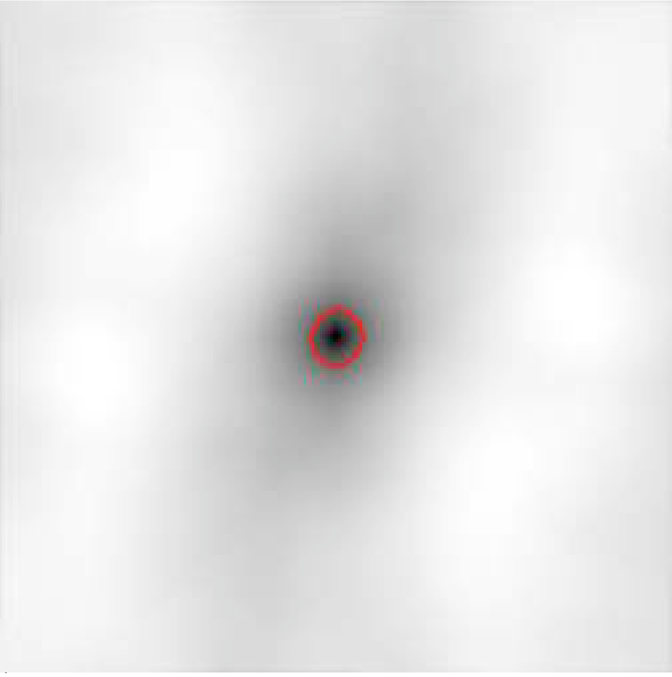


Convolution

If the same image is used for the target image and the cross-correlation image, the result is auto-correlation. This result can be best understood as imagining sliding an image across itself and seeing how far it must be displaced before features no longer lie on themselves. Hence, auto-correlation provides dimension information on the structuring element(s) present. In the example, the small features that make up the structure are not separated and even partially hidden by others, but the auto-correlation result provides an average shape and dimension. The **IP•Threshold–>Contours** plug-in was used to delineate a boundary on the correlation image.



Original *Cheese2* image"



Auto-correlation result with superimposed contour line

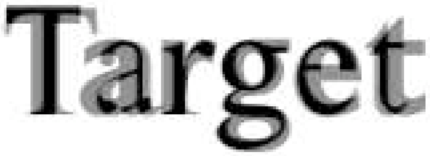
3.E. Other uses of correlation

3.E.1. Alignment

Cross-correlation is also used to match points in multiple images for alignment. The alignment procedure (which does not include rotation) requires placing one image into the second image memory, selecting the image to be shifted, and choosing **IP•Adjust–>Align with 2nd**. As shown in the example, the images do not have to match exactly. This procedure is useful before looking at image differences, lining up stereo pairs before merging or measuring them, and aligning images before combining them to form an extended focus composite.

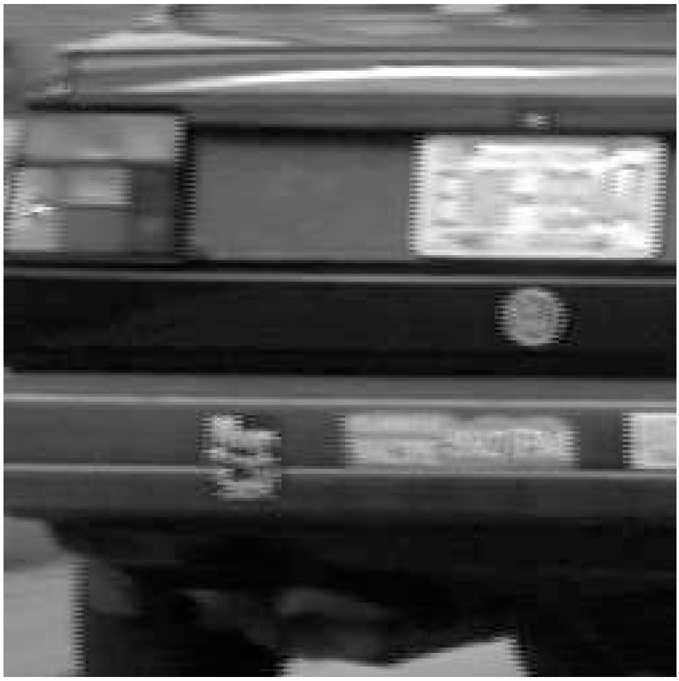
Target
Target

Two images to be aligned

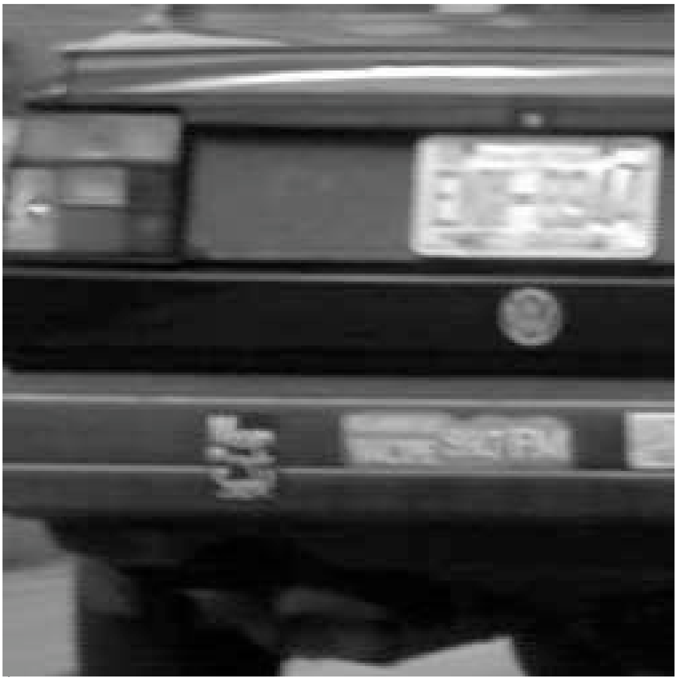


After alignment, a best-fit result is shown using layer transparency

A special case of alignment by shifting using cross-correlation occurs in video images. The two fields that make up one video frame consist of interlaced even and odd numbered scan lines, which are scanned 1/60th of a second apart in time. Motion of either the subject or camera can cause an offset between the two fields. The **IP•Adjust–>De-Interlace** plug-in finds the offset and shifts the fields for best alignment. Note that this approach is based on the assumption that the motion is uniform throughout the image or selected region.

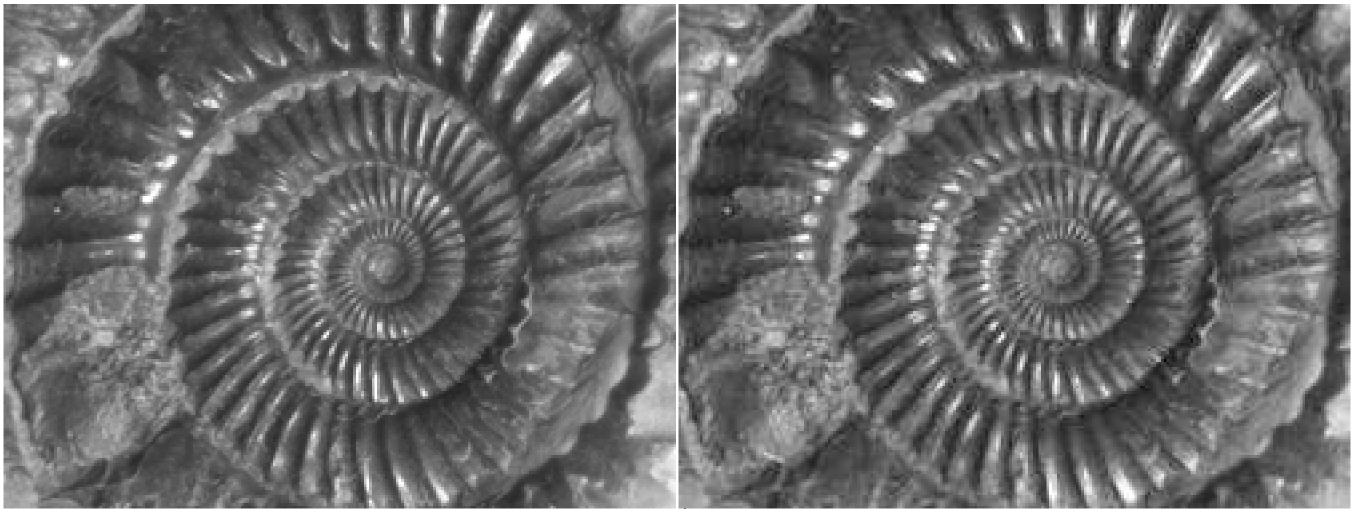


Original *Interlac* image"

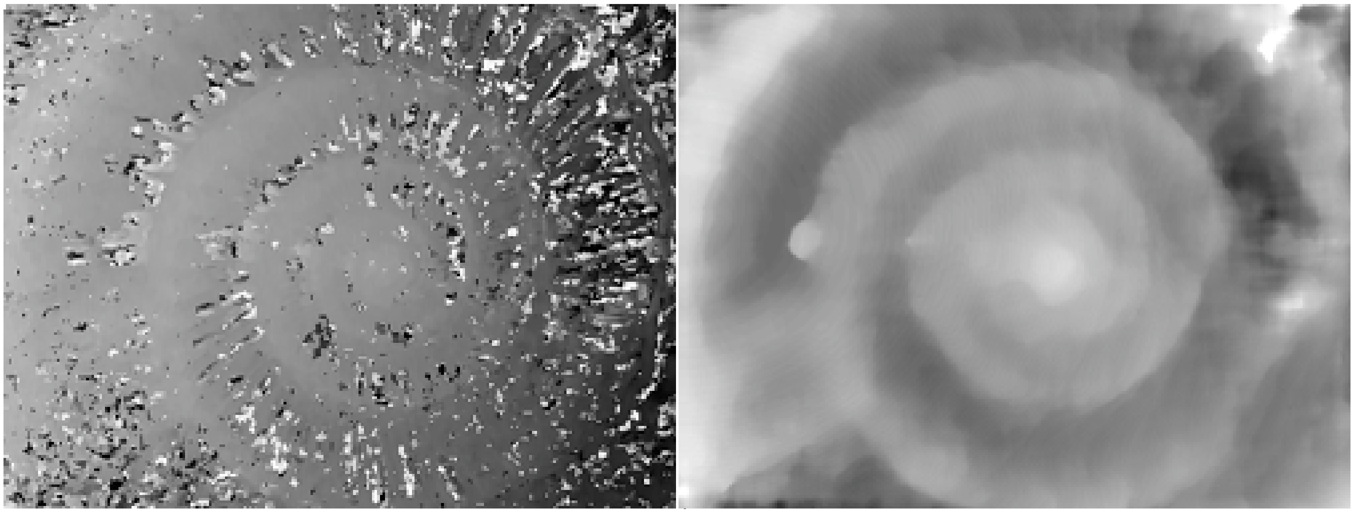


Fields aligned

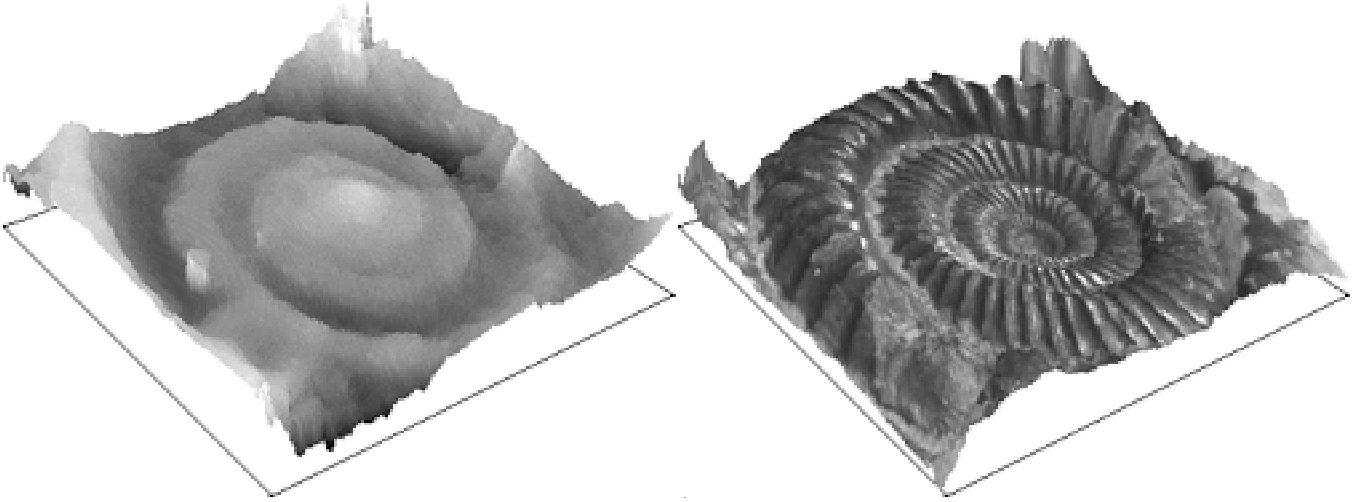
The **IP•Surface Measurement–>Stereo Pair Measurement** routine uses cross-correlation to match regions around each pixel in the stored second image to those in the current image, and shows the horizontal displacement (proportional to the vertical elevation) as a grey scale range image.



Stereo views (*Ammon_L, R*)



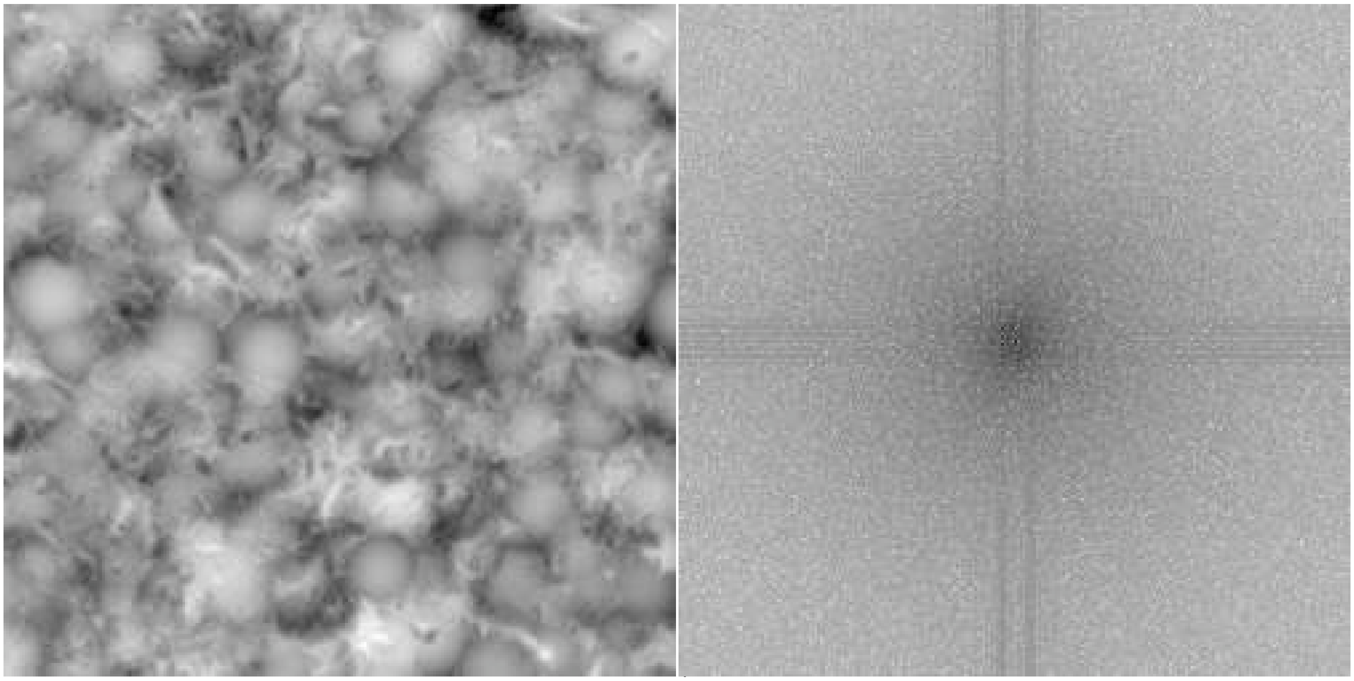
Result of stereo measurement, and the use of a median filter to remove dropouts, leveling (**IP•Surface Processing–>Flatten**) to remove the overall tilt, and contrast expansion.



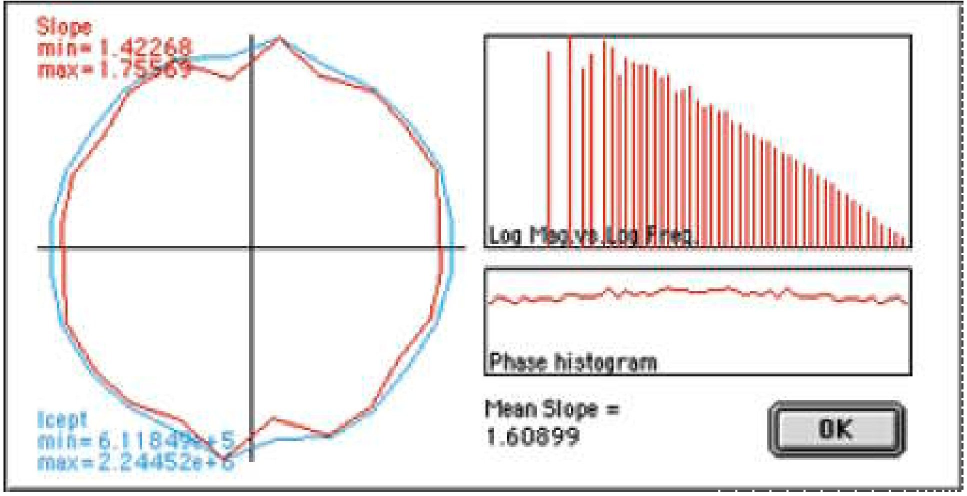
The rendered elevation measurements (**IP•Surface Processing–>Plot 2nd as Surface**), and with the original contrast superimposed (**IP•Surface Processing–>Reconstruct 2nd with Overlay**)

3.E.2. Measurement of fractal dimension

The FFT power spectrum provides a direct measurement of the fractal dimension of surfaces and structures. The slope and intercept of the amplitude vs. frequency curve (on log-log axes) give the dimension and topothesy (lacunarity) as a function of direction, provided the phases are randomized.



Original *ShotBlas* image and its Fourier transform power spectrum



Slope
min= 1.42268
max= 1.75569

Icept
min= 6.11849e+5
max= 2.24452e+6

Mean Slope =
1.60899

Analysis of the power spectrum showing a linear log-amplitude vs. log-frequency trend, random phases, and isotropic intercept and slope values for the fractal.

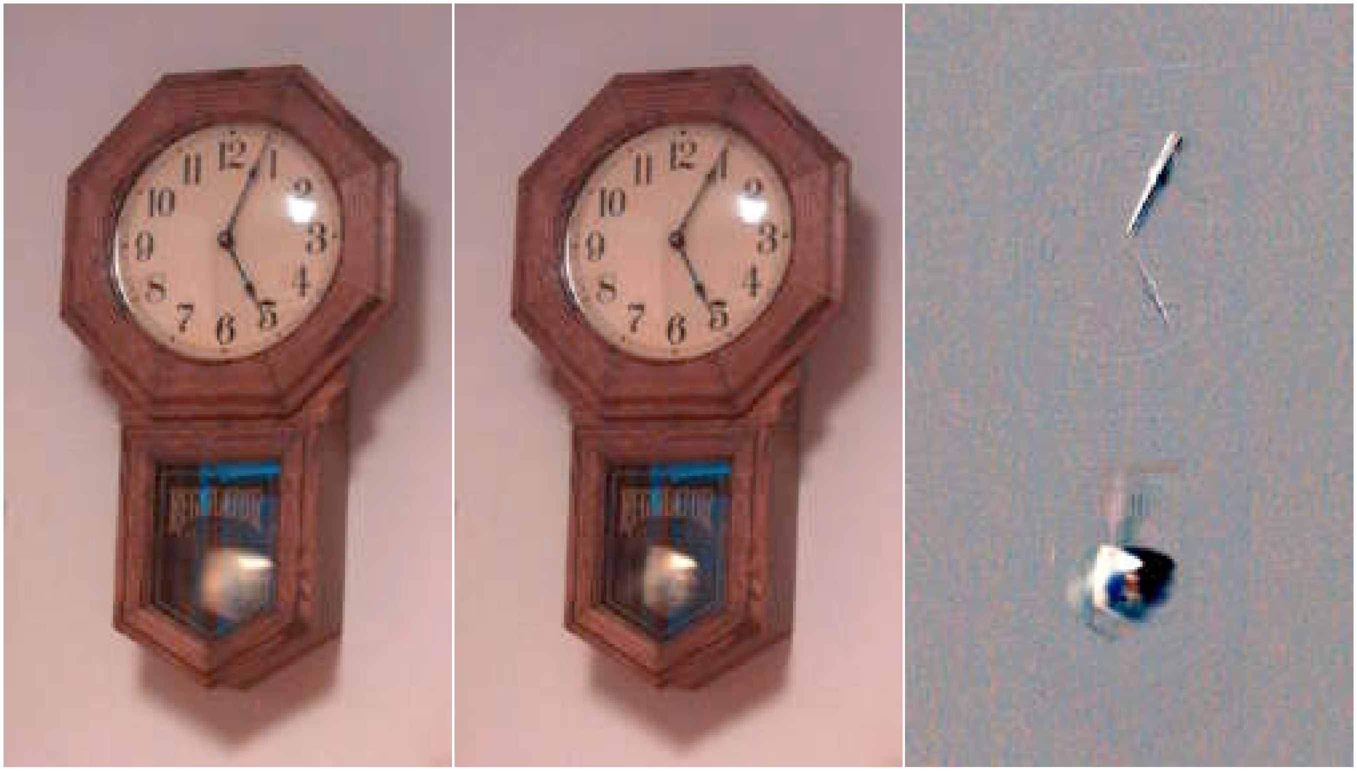
3.F. Detecting image differences

3.F.1. Alignment

The automatic shifting of one image to best align with a second, shown above in section 3.D.2, does not deal with rotation or changes in scale between images. Comprehensive alignment and registration tools for images consisting of either different color channels, serial sections, etc., may be performed manually or automatically, either based on the image contents or on discrete fiducial marks. Proper alignment is required for all procedures that combine multiple images. The use of the Photoshop layers capability for manual alignment is recommended.

3.F.2. Subtraction and ratioing

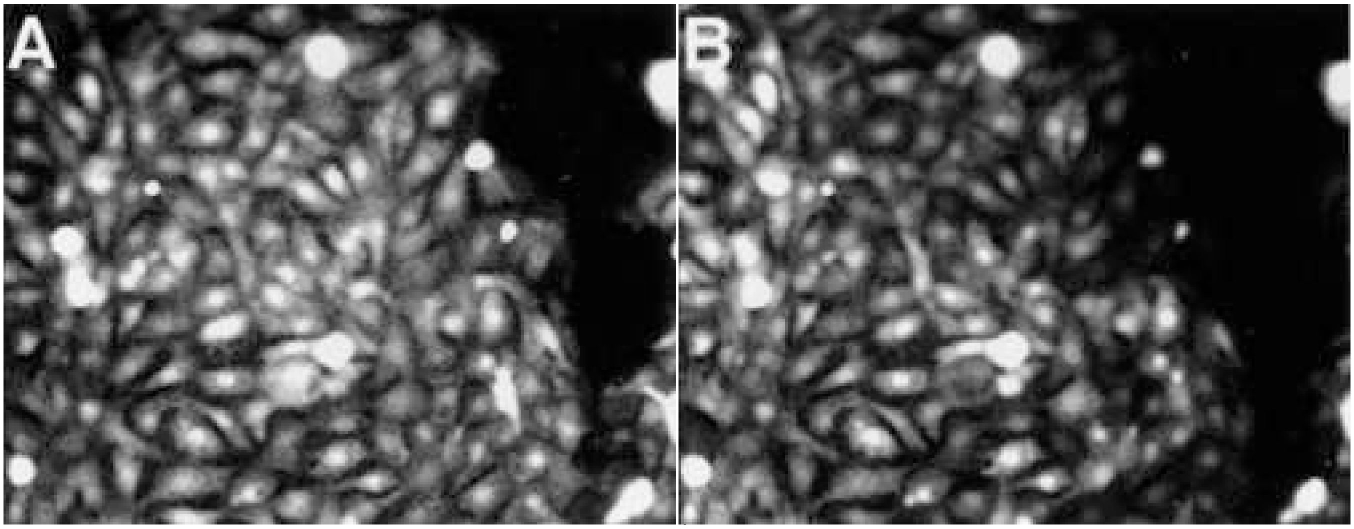
Combining multiple images by subtraction or ratioing, and displaying the results in RGB or HSI color channels, is an effective way to eliminate lighting variations on curved surfaces, thickness variations in sections, variations in stain concentration, etc. It also makes it easy to display differences that might otherwise be overlooked. Of all these techniques, subtraction is the most commonly used.



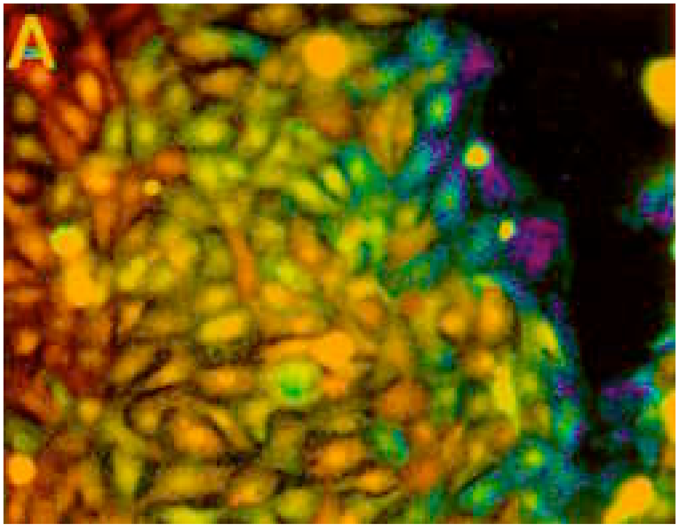
Original *C_503* and *C_505* images, and their difference

The example shows two images taken at different times, and the result of subtracting one from the other. Small details such as the movement of the hour hand are easily seen in the difference image that are not detectable from side-by-side comparison. To perform subtraction, place either image in the second image memory, click on the other image to make it front-most, and select **IP•Math–>Subtract**. Other functions, such as ratio (**–>Divide**) or absolute difference (**–>Abs.Diff**) are handled in the same way.

Examples of combining images in different color channels were shown in Sections 2.A.3 and 3.E.1. The following example combines the use of arithmetic combination with color channels for display. The two original images show fura-stained tissue illuminated by 340 and 385 nm wavelength light (which straddle the absorption edge of the stain). The ratio of the two images reveals the extent of calcium activity in the tissue. The sum of the two images is used to represent the amount of stain in each location. Finally, the ratio image is inserted into the hue channel of a composite to produce a color representation in which intensity shows the amount of stain and color represents activity.

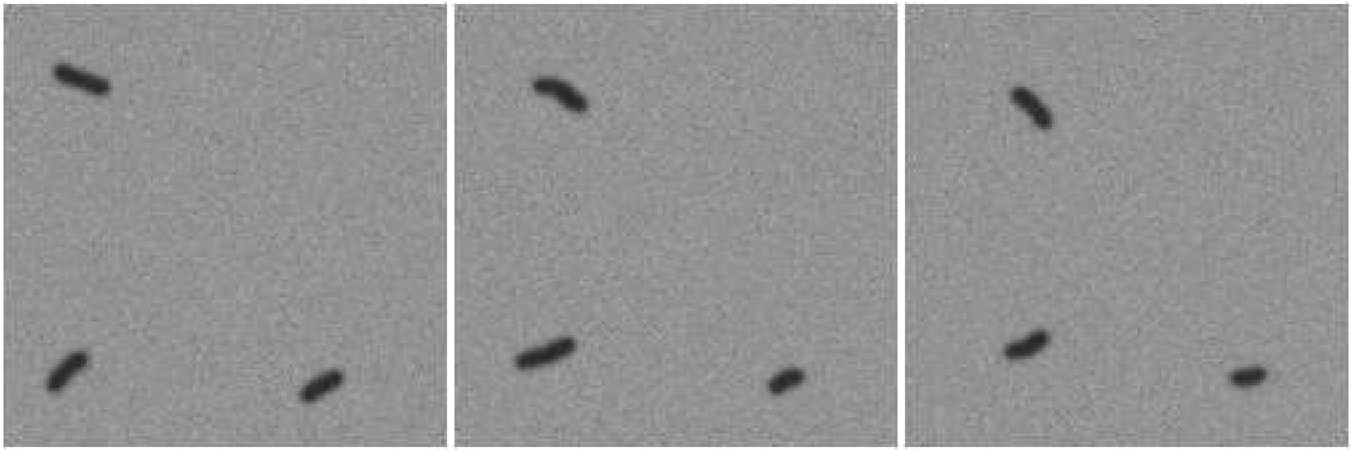


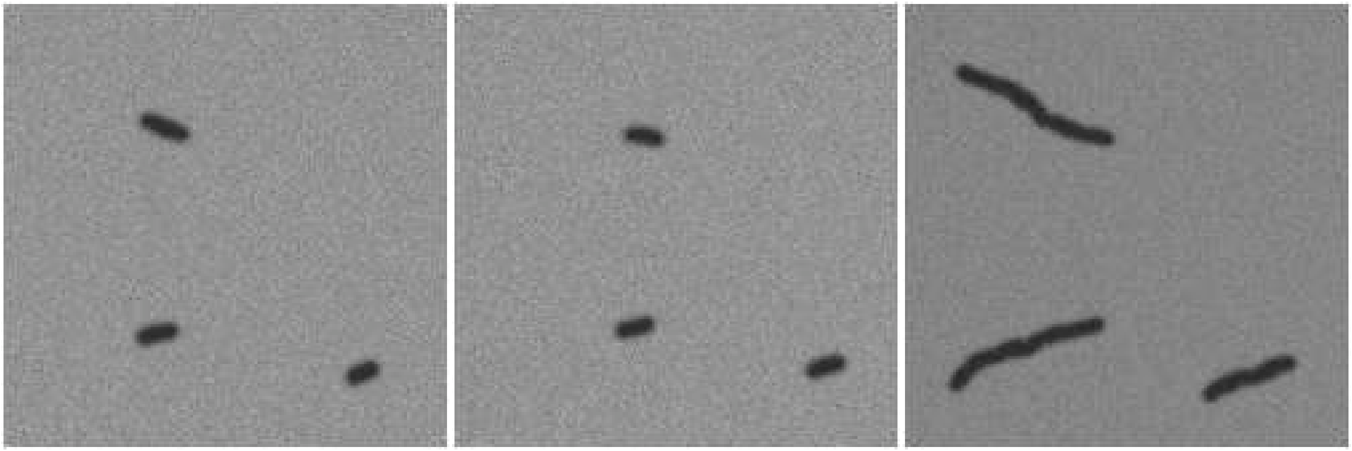
Original *Fura_340* and *Fura_385* images



Resulting color composite

Keeping the brightest or darkest pixel values in a sequence can be useful to track moving features as shown in the example. In the final composite image, the track of each feature may be thresholded and measured. This type of image combination is also useful to combine multiple images with different light source positions to fully illuminate irregular surfaces.





The five images *Track_1, 2, 3, 4, 5* and the composite produced by keeping the darker value at each point

3.G. Summary of image processing procedures

Image processing may be done for several reasons: improving the visual appearance of images to aid in detecting and observing important details; preparing images for publication, to effectively communicate to others who are less familiar with the subject; producing some measurement information directly, often by analysis of the histogram; and preparing images for thresholding so that other quantitative measures can be obtained. It may be helpful to summarize the classes of operations as follows:

Global functions (treat every pixel independent of its surroundings)
"   Histogram modification (stretching, gamma adjustment, equalization, inverting)
"   Math (add, subtract, multiply, divide, brighter, darker, absolute difference)
"   Leveling (remove background obtained by measurement, fitting, processing)
"   Color plane manipulation (merging, separating, converting, transferring)

Neighborhood operations (treat each pixel in the context of its immediate surroundings)
"   Convolution kernels (smoothing, sharpening, derivatives)
"   Ranking operations (median, brightest, darkest, range)
"   Statistical operations (variance, entropy, maximum likelihood)
"   Local equalization (compare pixel values and adjust local contrast)

Fourier-space functions (treat the image as being composed of multiple frequencies and orientations)
"   Removal of specific frequencies (high-pass, low-pass, periodic noise removal)
"   Averaging of periodic structures
"   Deconvolution for defocus correction
"   Cross-correlation (locates features)

Understanding this relatively short list of function classes, and developing enough experience with them to be able to visualize and predict what each will do to a given image, prepares the user to efficiently handle a wide variety of image analysis tasks.

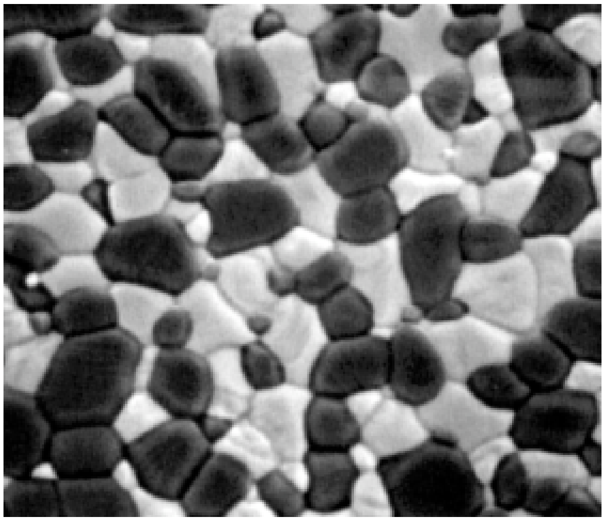
4. Thresholding of image features

4.A. Thresholding using the histogram

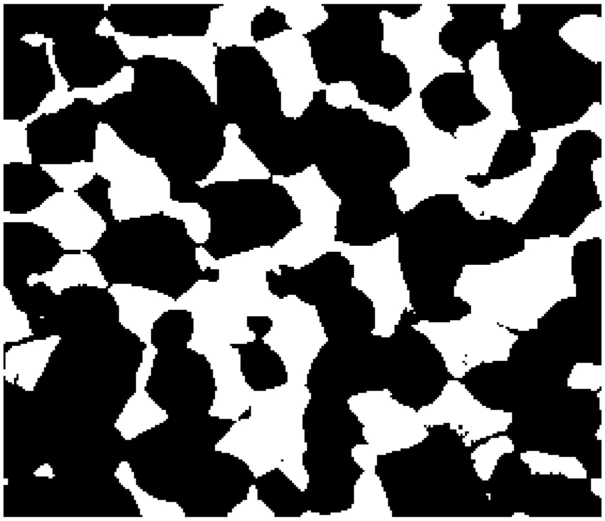
4.A.1. Manual settings

It is assumed throughout this section that whatever processing is needed, e.g. to level contrast across an image or to convert texture to brightness, has already been performed. Manual interactive setting of thresholds by moving a slider on the histogram can be used to produce binary and contour images, which define the pixels that represent the features to be measured. This is the step where most image measurement errors arise, because of inconsistent human judgment. Adjusting the slider until the result "looks right" is a quick procedure but may be difficult to replicate on another day, or another image, or by another person.

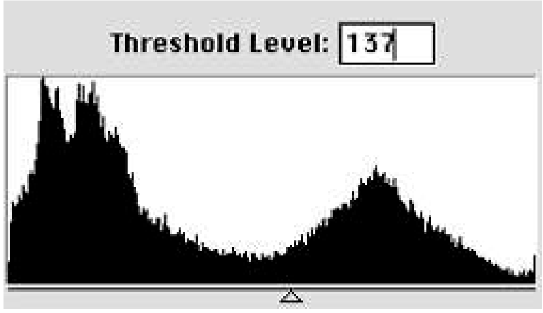
In the example, threshold settings over a considerable range produce visually plausible results, but the area fraction (which measures the volume fraction) of the dark phase varies by about 10%, and simple (but ultimately meaningless) rules such as "pick the lowest point" or "halfway between the peaks" do not produce a correct result, or even one that is consistent for a sequence of images.



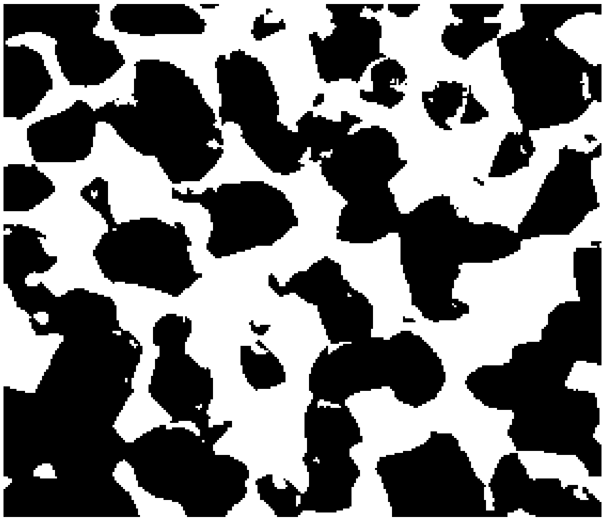
Original *Zirconia* image. Setting the manual threshold value (in **Image–>Adjustments–>Threshold**) to two extreme values as shown below produces area fractions of 65% and 55% for the dark phase, respectively.

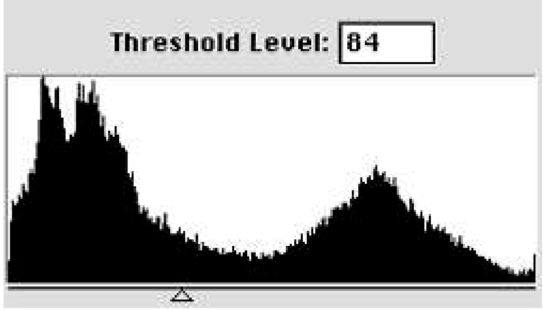


!



Threshold Level: 137

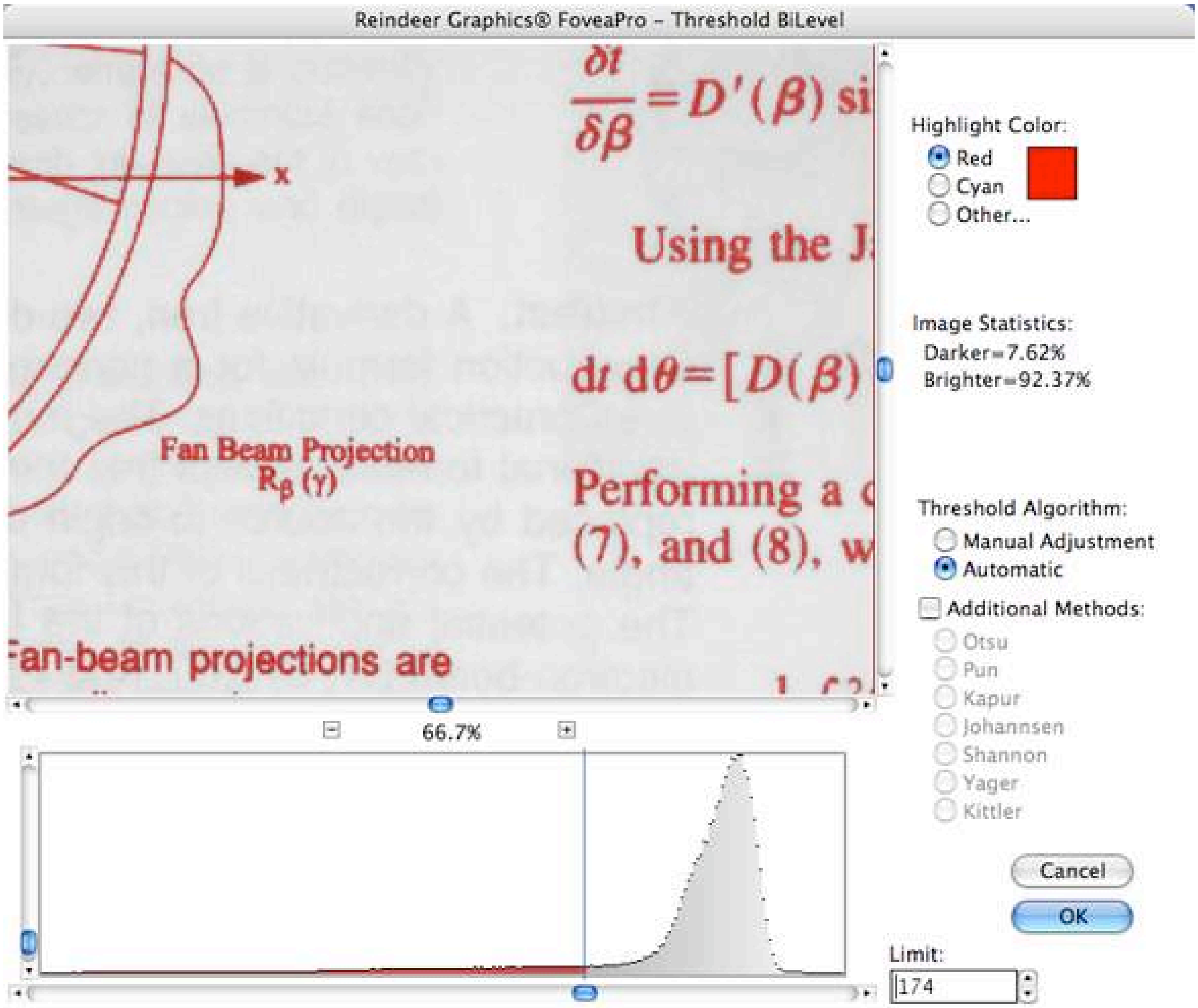

!


Threshold Level: 84

4.A.2. Automatic methods

A wide range of automatic methods for thresholding are available. Selecting the appropriate technique is based on prior knowledge about the sample and image (e.g., that there are just two phases present, or that boundaries should be smooth). In many cases thresholding does not produce a precise delineation of the desired structure and further post processing will be needed (discussed below in Section 5).

The most common class of automatic tools work with the histogram, and do not consider the values of neighboring pixels. These are primarily statistical in nature and were originally developed for thresholding print on paper as a preliminary to optical character recognition. A typical assumption is that there are two classes of pixels (white paper and black ink) and that the threshold that best separates them divides the histogram into two groups in such a way that a statistical test such as the student's t-test returns the highest probability that they are different groups. As shown in the example, that does a pretty good job for ink on paper (**IP•Threshold–>Bilevel Thresholding–>Auto**). Notice in the dialog that the histogram in this case does not have two peaks, but instead there is one large peak for the white paper and a long irregularly shaped tail for the dark ink.



Reindeer Graphics® FoveaPro – Threshold BiLevel

Highlight Color:
● Red
○ Cyan
○ Other...

Image Statistics:
Darker=7.62%
Brighter=92.37%

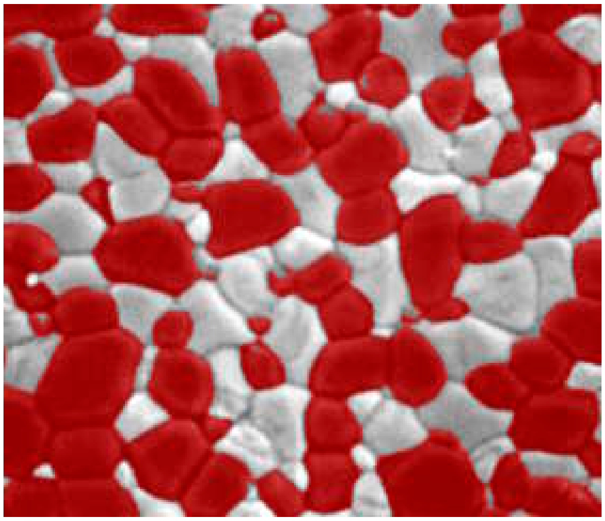
Threshold Algorithm:
○ Manual Adjustment
● Automatic

□ Additional Methods:
○ Otsu
○ Pun
○ Kapur
○ Johannsen
○ Shannon
○ Yager
○ Kittler

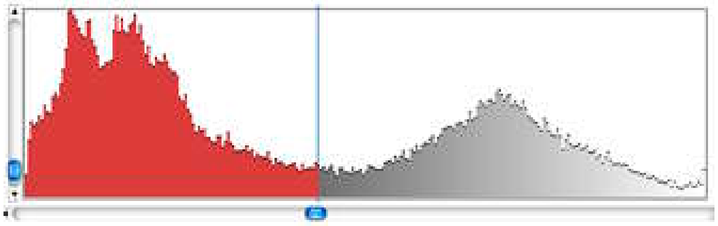
Limit: 174

Automatic threshold selection in the bilevel thresholding dialog applied to *Document* image.

The plug-in offers additional algorithms for automatic threshold selection, each of which makes slightly different assumptions about the nature of the distribution of brightness values and applies different statistical tests to the histogram.

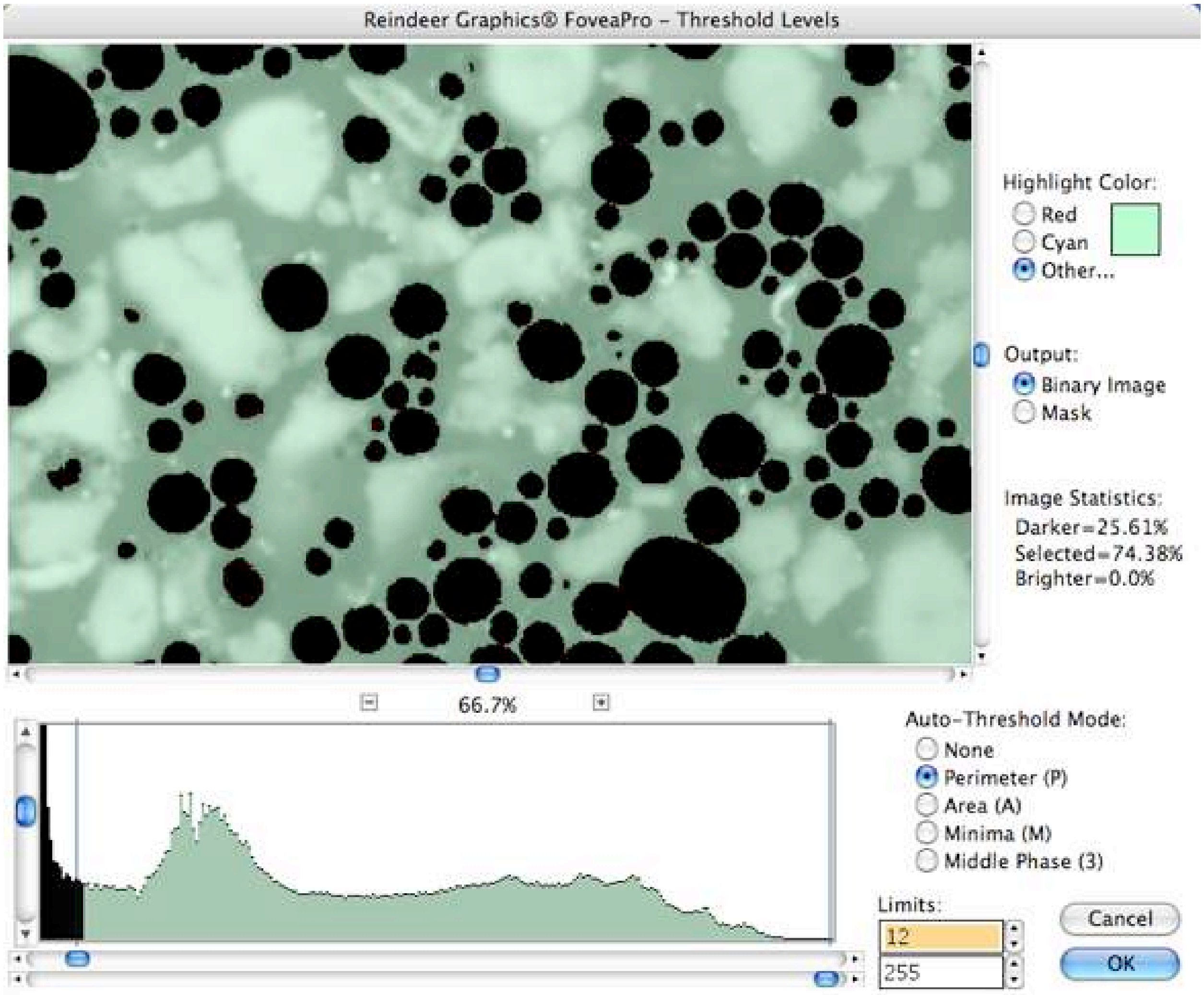


If the automatic procedure is applied to the *Zirconia* image shown above, it selects a threshold that correctly splits the area into 60%-40%. Note that the threshold point is not an obvious one from the visual appearance of the histogram (not the lowest point, or midway between the peaks, etc.)



A different approach to automatic thresholding uses information about the spatial distribution of pixels as well as their brightness values. One typical criterion is that the borders of the thresholded regions should be smooth, which corresponds to samples in which membranes, surface tension or other physical effects are expected to produce boundaries that are smooth (it would not apply, for example, to fractured concrete particles which are fractal and rough, but is very appropriate for the fat droplets in mayonnaise shown in the example). The **IP•Threshold–>Threshold Levels** plug-in offers this algorithm to refine an initial setting by up to ±16 grey levels from the user selection (click on the "Perimeter" button). Again, there are several other algorithms that can be selected but tests indicate that the "smoothest perimeter" criterion often corresponds to the setting chosen manually by experienced users.

"



Reindeer Graphics® FoveaPro – Threshold Levels

Highlight Color:
○ Red
○ Cyan
● Other...

Output:
● Binary Image
○ Mask

Image Statistics:
Darker=25.61%
Selected=74.38%
Brighter=0.0%

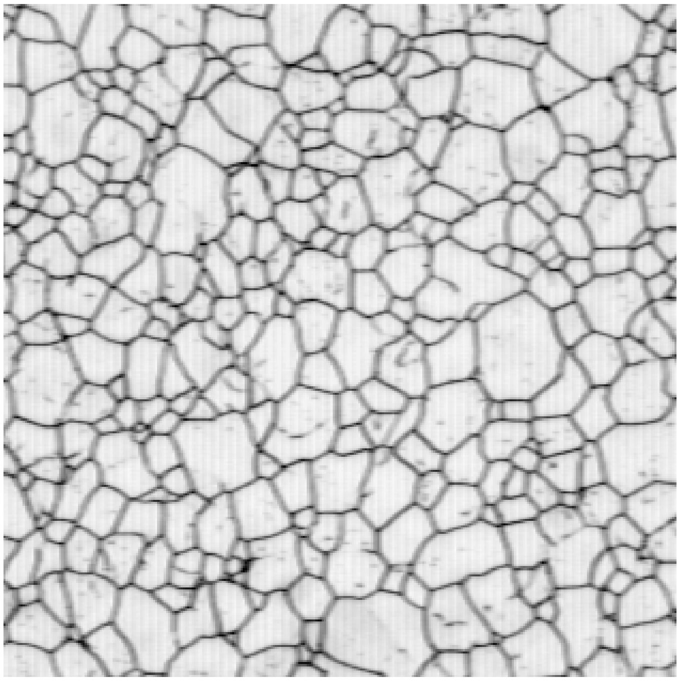
66.7%

Auto-Threshold Mode:
○ None
● Perimeter (P)
○ Area (A)
○ Minima (M)
○ Middle Phase (3)

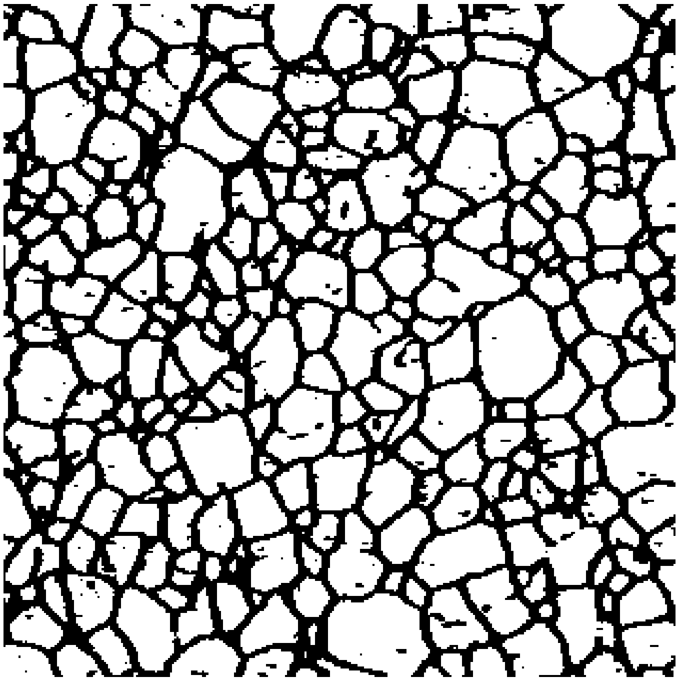
Limits:
12
255

The Threshold Levels dialog applied to the fat droplets in the *Mayo* image

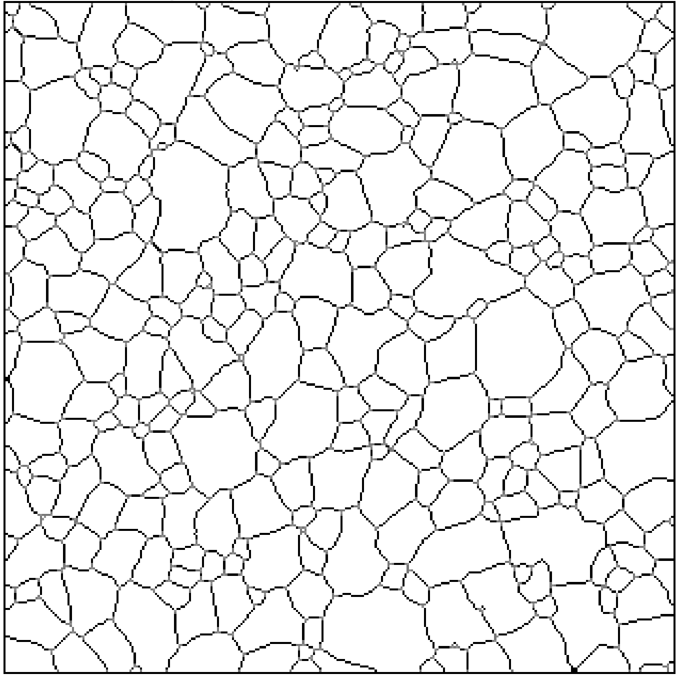
It is often useful to keep in mind that further processing after thresholding may be required, and that the requirements of the thresholding operation can sometimes be relaxed. In the following example, the boundaries between the grains in the metal are atomically narrow, but appear broad because of the chemical etching used in sample preparation. Thinning down the boundaries to single-pixel lines will be performed using skeletonization (discussed in the next section). Consequently, the threshold value can be set over a relatively broad range (which changes the width of the as-thresholded boundary lines) without affecting the final result.



Gr_Steel image (fragment) after leveling brightness"



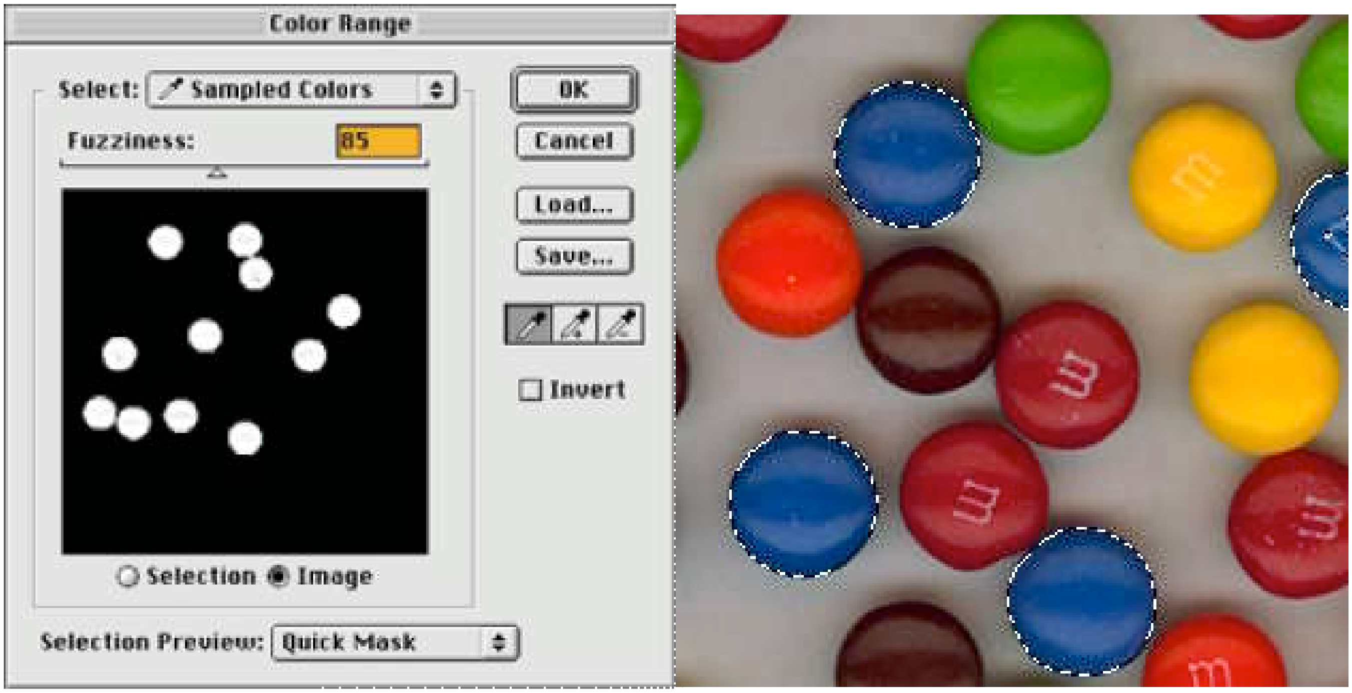
Thresholded binary image



After skeletonization

4.A.3. Selecting a color range

Thresholding of color images can be performed using the same routines as for grey scale, but then only the intensity of each pixel is considered. Usually the color information is important for selection and should be used. In either RGB or HSI space, a range of colors can be specified to select structures for linearization. The built-in Photoshop color selection routine (**Select–>Color Range**) allows clicking on the image to specify the color of interest and shows the selected pixels. The “fuzziness” slider operates equally on the R, G and B channels and controls the degree of selection. After accepting the result, the image shows selection lines (“marching ants”) that can be used to create a binary image by filling the selection with black, inverting the selection, and filling the background with white. (The Photoshop alpha channel allows pixels to be fractionally selected; the selection marquee corresponds to the 50% selection boundary.)



Select: Sampled Colors

Fuzziness: 85

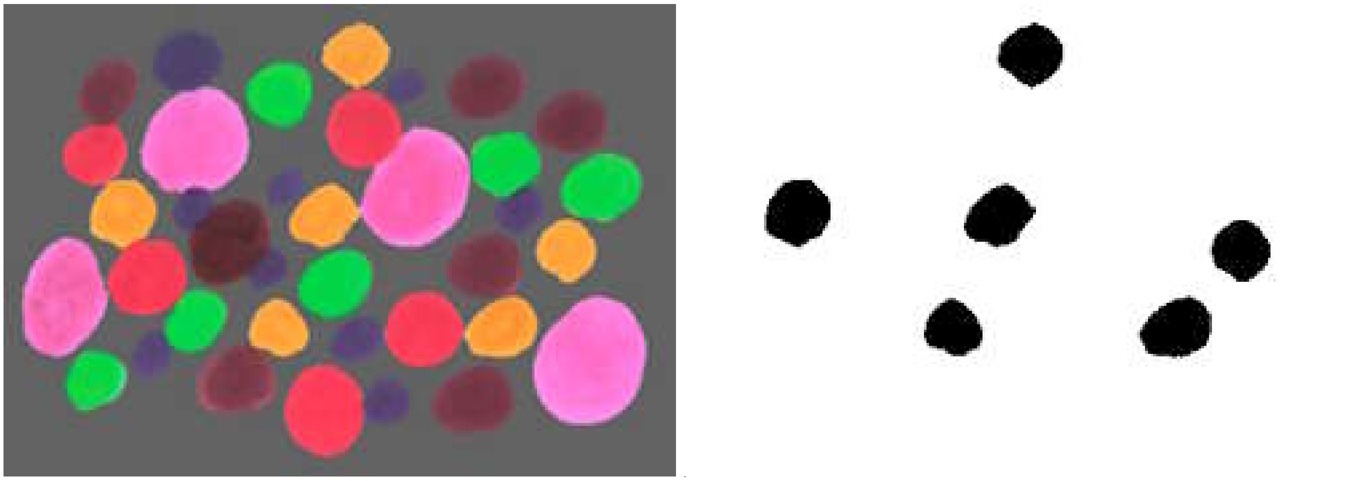
□ Invert

○ Selection ● Image

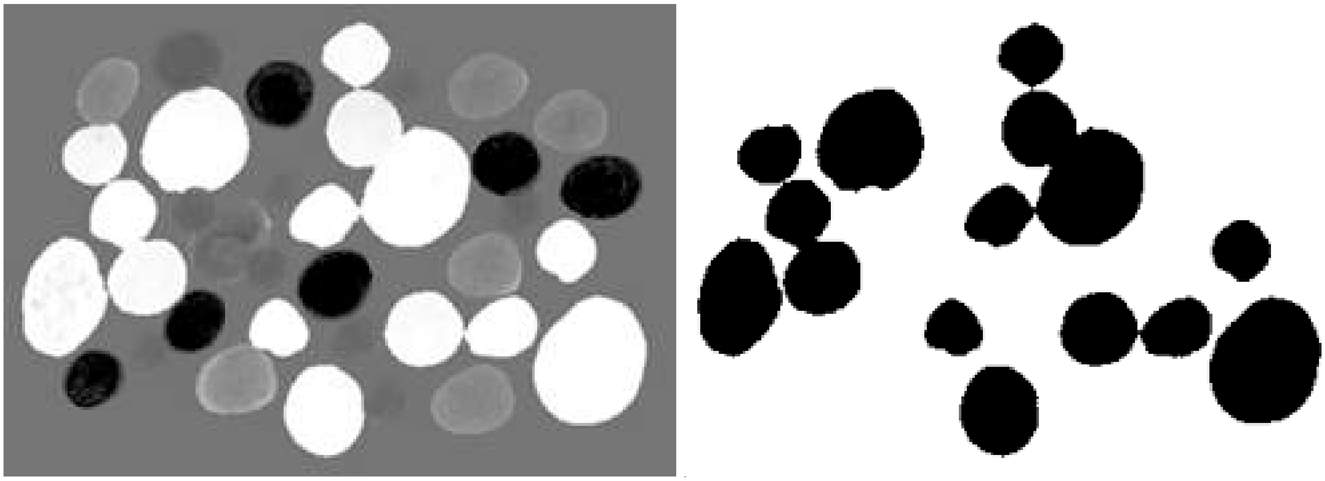
Selection Preview: Quick Mask

The Color Range selection dialog applied to the *MandM* image, and the resulting selection

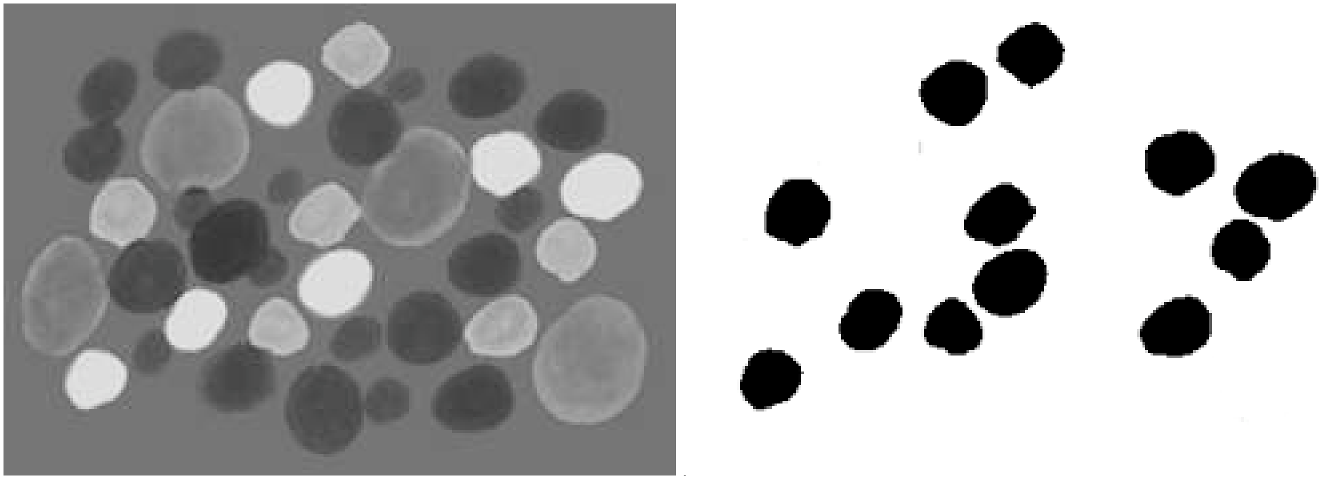
More individual control over the red, green and blue color ranges is possible by thresholding the channels individually. The individual results are combined with a Boolean AND to select pixels whose red, green and blue values all lie within the selected range. To illustrate the logic involved, it is useful to examine the channels individually. In the example, the yellow spots are selected by thresholding the red and green channels for pixels that have high intensities in both.



Original *ColrDots* image, and the yellow spots defined by Boolean AND of thresholded red and green

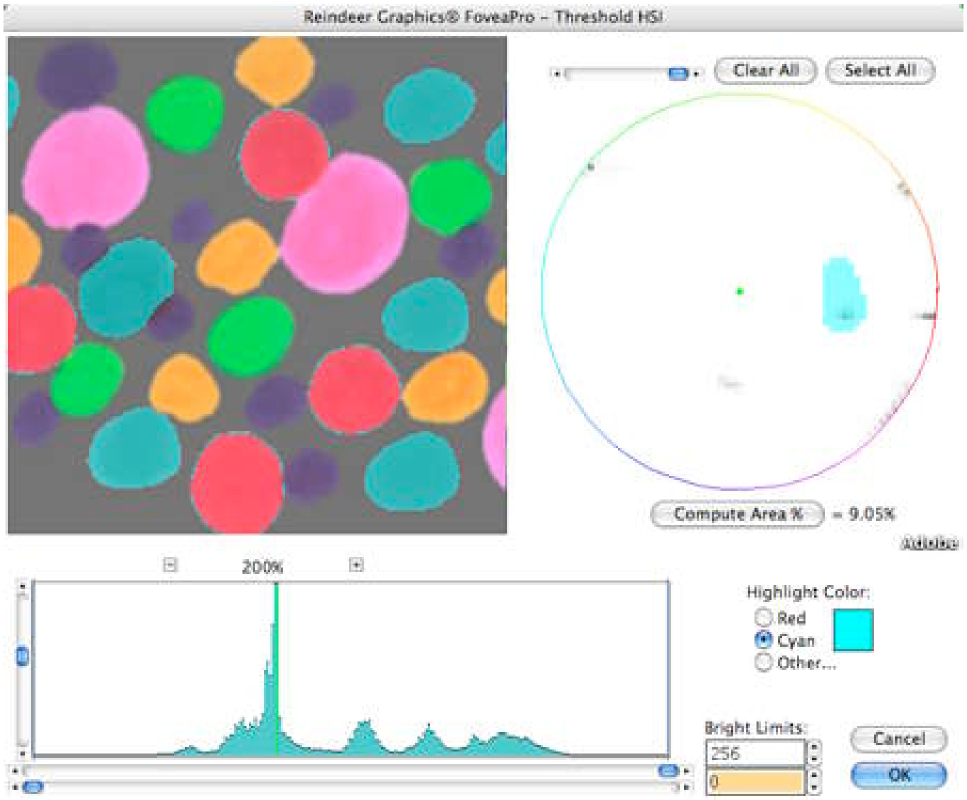


Red channel and thresholded spots with high red intensity



Green channel and thresholded spots with high green intensity

In most cases, the combination of red, green and blue intensities that define colors is not obvious (for example, consider the brown spots in the previous image). Hue-Saturation-Intensity space provides a much more user-friendly way to perceive and define color. The **IP•Threshold–>Threshold HSI** plug-in represents this color space graphically as shown in the example. Marking out a region on the Hue-Saturation circle (the angle corresponds to the hue, the radius to the saturation; the darkness of locations represents the number of pixels with those values), and setting limits on the intensity axis, select pixels shown in the preview (clicking on a point in the preview marks the corresponding color coordinates as well). Notice in the example that brown is characterized not as a combination of red, green and blue, but rather as a dark, low-saturation red.



Reindeer Graphics® FoveaPro – Threshold HSI

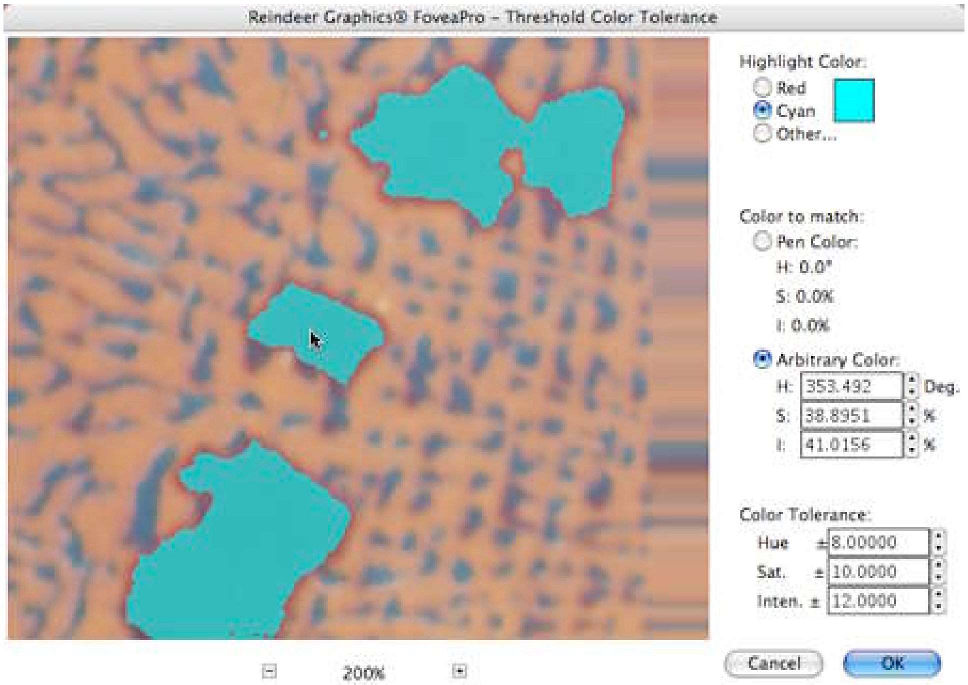
Compute Area % = 9.05%

Highlight Color:
○ Red
● Cyan
○ Other...

Bright Limits:
256
0

Selecting the brown spots in the *ColrDots* image

Once the user is familiar with thresholding in HSI coordinates, it is efficient to select a color range using the **IP•Threshold–>Color Tolerance** routine. Clicking on the image with the eyedropper tool selects a color of interest. Then the tolerances in hue, saturation and intensity can be independently set to define the regions of interest.



Reindeer Graphics® FoveaPro – Threshold Color Tolerance

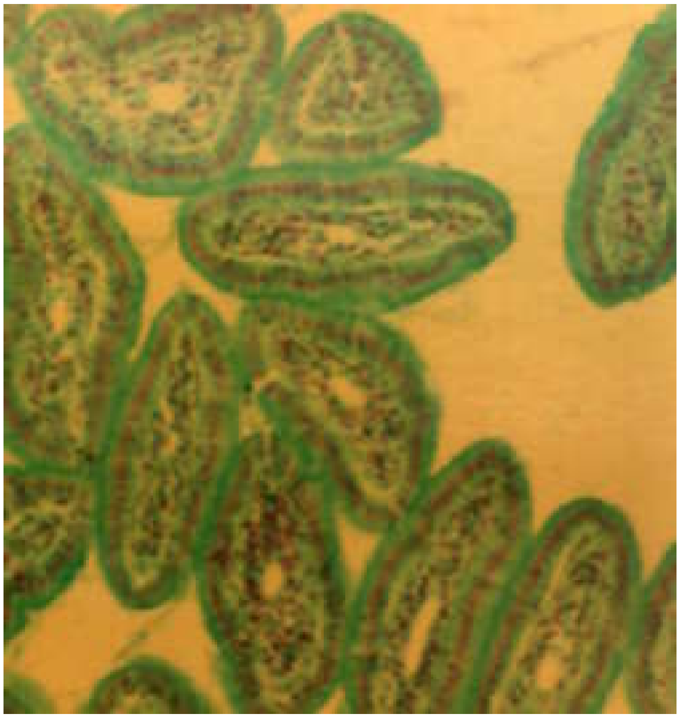
Highlight Color:
○ Red
● Cyan
○ Other...

Color to match:
○ Pen Color:
H: 0.0°
S: 0.0%
I: 0.0%
● Arbitrary Color:
H: 353.492 Deg.
S: 38.8951 %
I: 41.0156 %

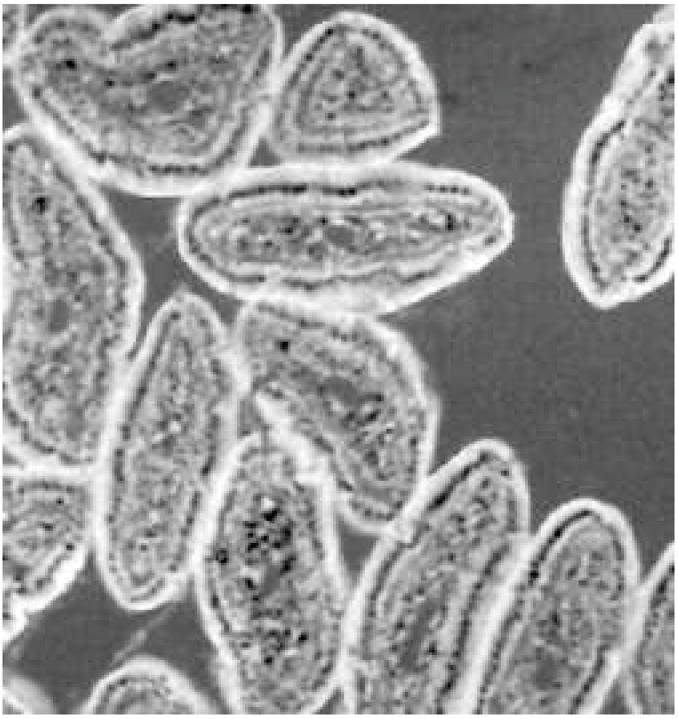
Color Tolerance:
Hue ± 8.00000
Sat. ± 10.0000
Inten. ± 12.0000

Selecting the color of interest and setting a color tolerance range (*Alloy* image)

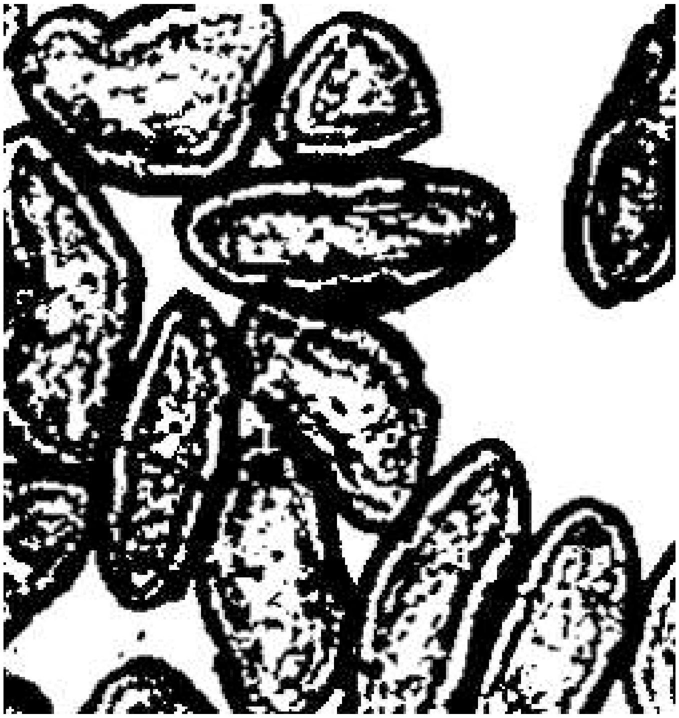
Sometimes only a single axis in the HSI coordinate space is needed, usually the hue information because most stains and fluorescing dyes are selected because of their characteristic color (hue). Extracting just the hue channel from the image (**IP•Color–>Color Filter**) and applying the Bi-Level Thresholding or Threshold Levels plug-in to the resulting greyscale image can then provide an efficient and automatic procedure for obtaining the desired binary image. In the example, the automatic bi-level thresholding method is used because there are just two colors of stain added to the tissue.



Original *Intestine* image"



Hue channel (contrast expanded)

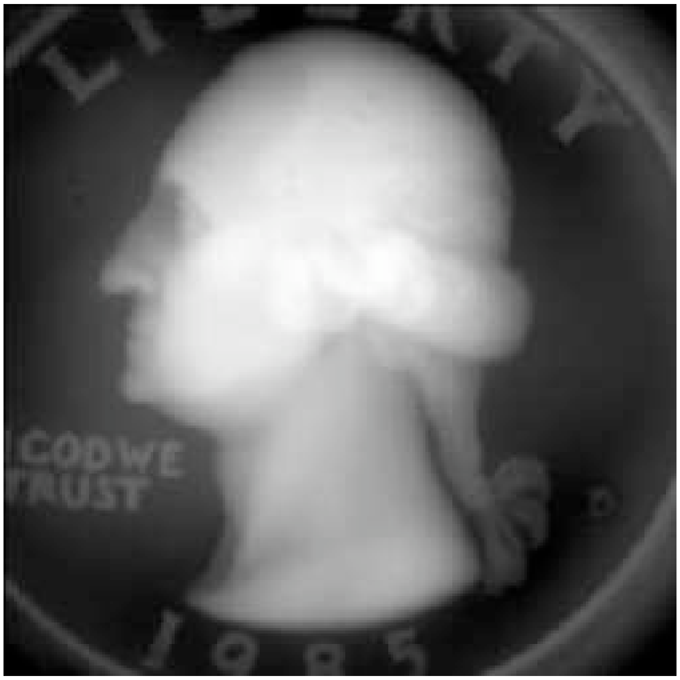


Automatic bi-level thresholding result

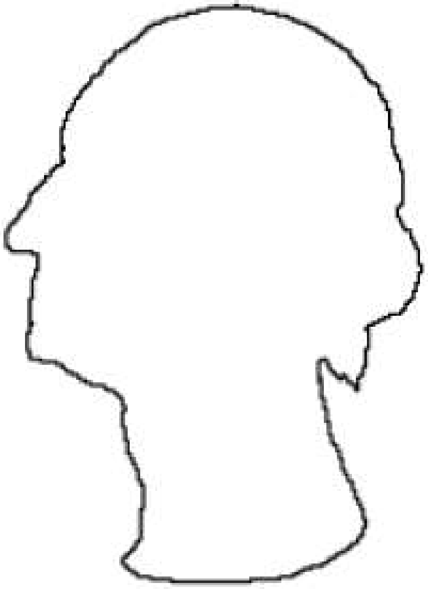
For color or other multi-channel images, more powerful techniques for separating the various structures or phases to allow thresholding are available using the Principal Components Analysis routines (under the **Filter–>Prism** menu). After transforming to principal components space, the channels can be thresholded just the same as the “regular” color channels.

4.A.4. Contour lines (iso-brightness lines)

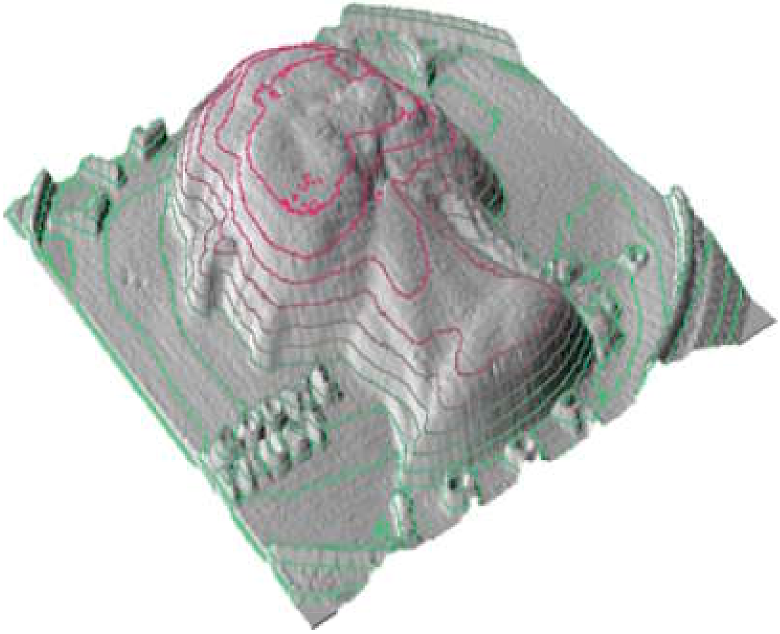
Contour lines provide an alternative to conventional thresholding. Rather than selecting a range of pixel values, they mark boundaries between such ranges. Contours are continuous lines that often correspond to physical boundaries, and can be used to measure them. For a range image in which brightness represents elevation, the contour line is an is elevation line and can be used to generate a topographic contour map.



Original *Quarter* image"

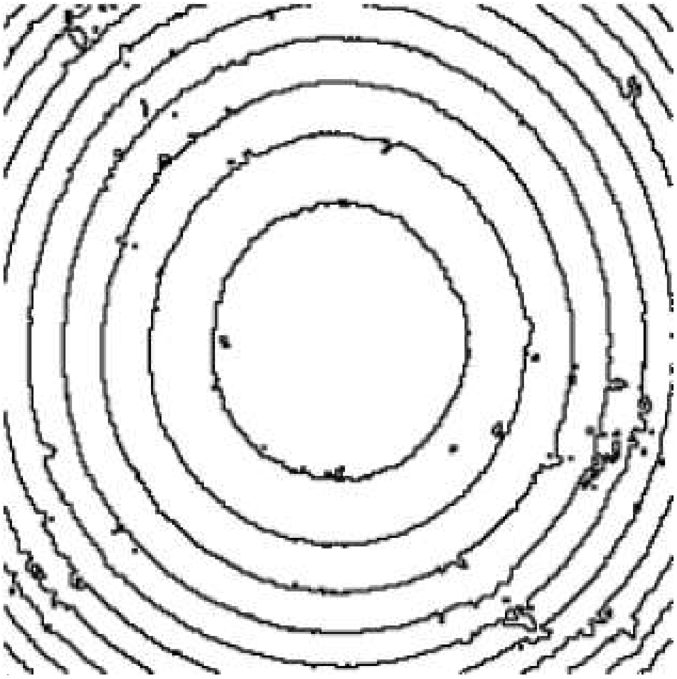


A single contour line



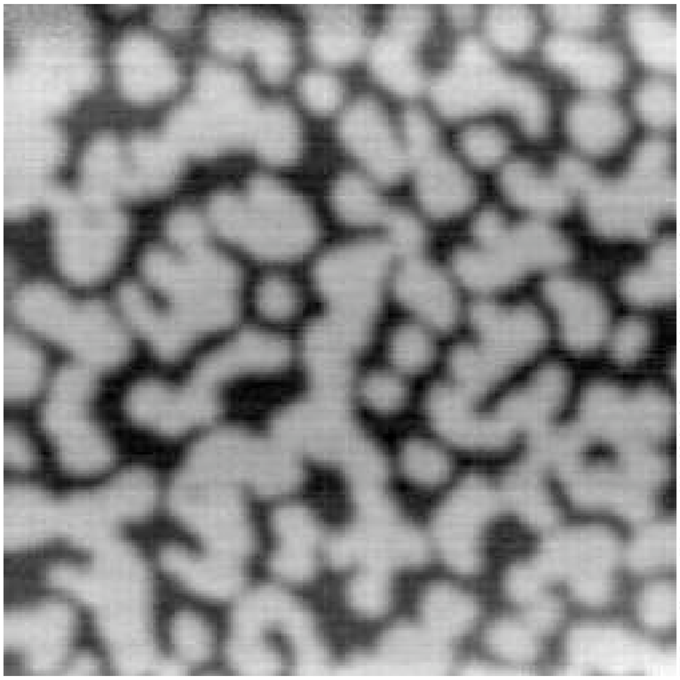
Multiple colored contour lines superimposed on a rendered, Phong shaded surface representation. (The sequence used to produce this graphic was recorded in an action; details are available on the DrJohnRuss.com website)

Original *Ball bear* image (range image)"

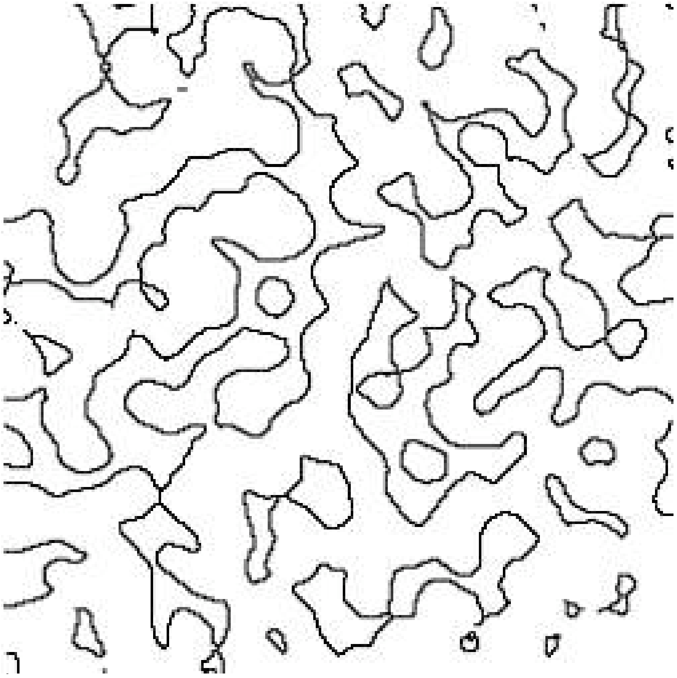


Multiple contour lines showing roughness and out-of-round

As will be covered in Section 6.B.2. on stereology, the surface area of contact between two structures can be determined by measuring the length of the line that separates them. The contour line often provides a direct representation of this boundary, and can be measured with the **IP•Measure Global–>Total Line Length** routine. The example shows the contour line separating the two phases in a metal.



Original *MonoLyr2* image"

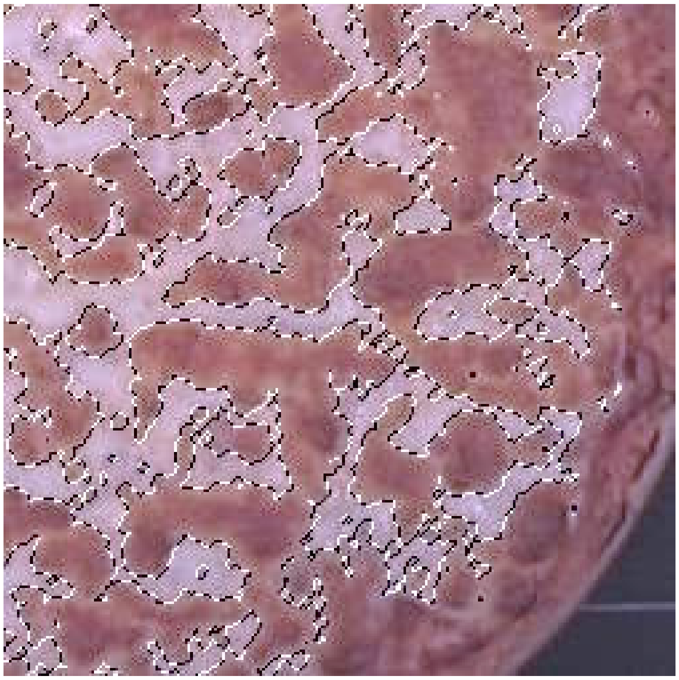


Contour line separating the phases

4.B. Marking features manually

4.B.1. *Region growing*

In some situations, a “wand” approach allows human interaction to select features for measurement, or background (which is often more uniform than the features). Clicking on one point selects all pixels within the chosen tolerance (the range of red, green and blue intensities for a color image) that are connected by a continuous selected path to the original point. Then **Select–>Similar** can be used to include other locations with the same range of colors.

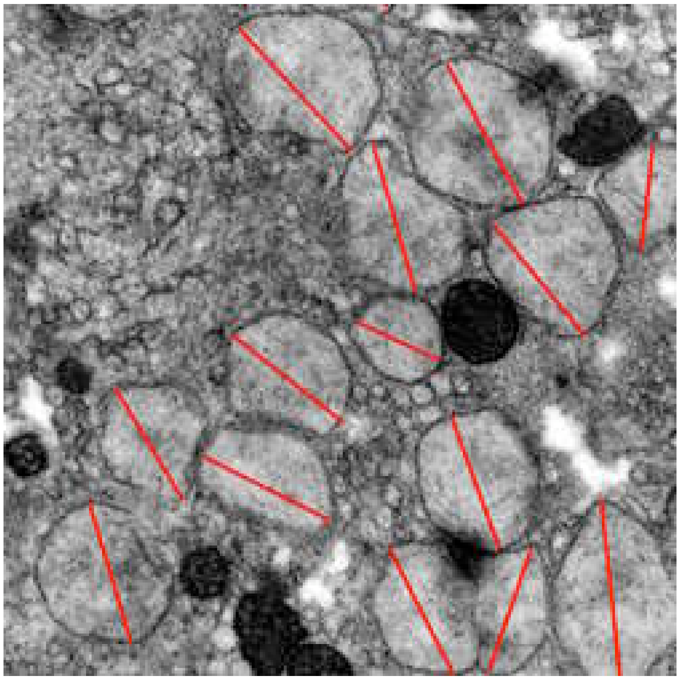


A portion of the *Pizza1* image with the uncooked cheese selected using the wand

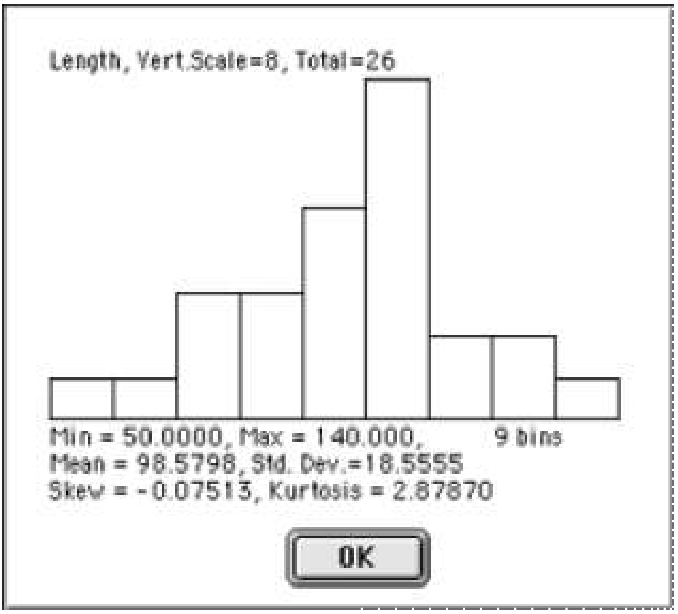
The wand is also useful for selecting regions to be fit as background (as discussed in Section 2.C.1). Depressing the shift key allows clicking on additional areas to be added to the selection, which need not be continuous. Turn off “anti-aliased” in the tool preferences bar so that pixels are not partially selected.

4.B.2. Manual lines, points, etc

It is also possible for the user to mark lines, points, circles, or other features onto the image which are counted and/or measured, when it is impractical to process the image to permit thresholding and automatic measurements. It is generally wise to select a bright color (for grey scale images) or black or white (for color images) so that the marks are easily visible, and to keep the marked image as a record of what was done. In the example, lines were drawn on the image to mark the size of organelles that are not distinct in stain density nor completely delineated in the section. Once these lines were drawn, the **IP•Threshold–>Isolate Pen Color** function erases everything except the lines, which can then be measured as shown in Section 6. Similarly, marking features of interest with a colored pencil (with a size large enough for easy visibility) permits counting of features that can be visually recognized even if they cannot be easily processed for thresholding.



TEM3 image (fragment) with manual lines superimposed!



```
Length, Vert.Scale=8, Total=26
Min = 50.0000, Max = 140.000,       9 bins
Mean = 98.5798, Std. Dev.=18.5555
Skew = -0.07513, Kurtosis = 2.87870
```

Length distribution of the lines

5. Binary image processing

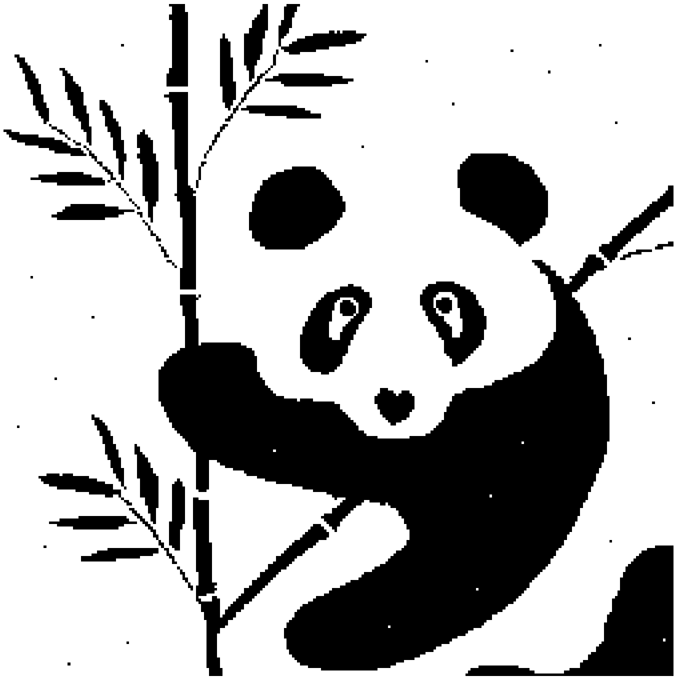
5. A. Removing extraneous lines, points or other features

5.A.1. Erosion/dilation with appropriate coefficients to remove lines or points

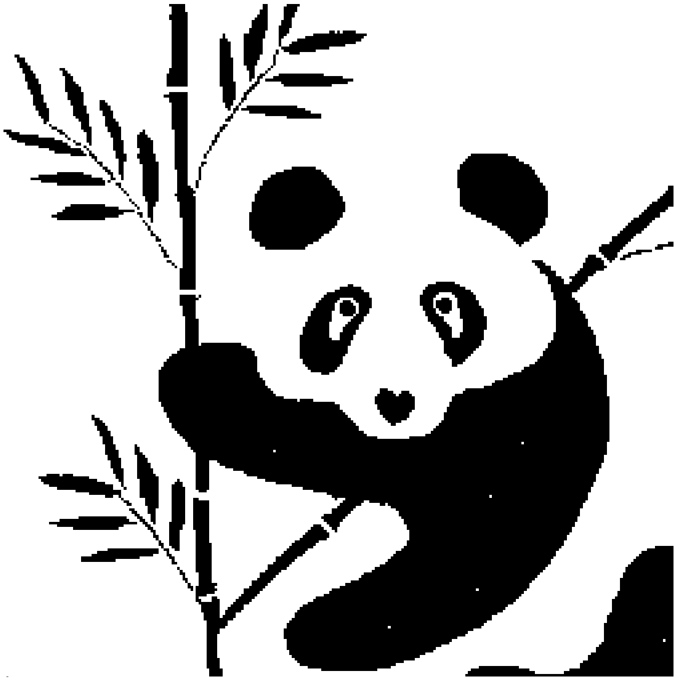
The use of combinations of erosion, dilation, opening and closing permit selective correction of binary image detail when thresholding leaves artefacts behind. Classical erosion and dilation are morphological operators that compare pixels to their immediate neighbors to determine whether to change black to white or vise versa. An opening is the sequence of erosion followed by dilation, while a closing is the opposite.

The **IP•Morphology–>Classic Morphology** plug-in provides two parameters to control the processes of erosion, dilation, opening and closing: The number of iterations of adding and/or removing pixels is related to the dimension of the modification. The coefficient is a test parameter that the number of adjacent neighbor pixels (out of the 8 possible) that are opposite in color (black or white) to the central pixel must exceed for it to change.

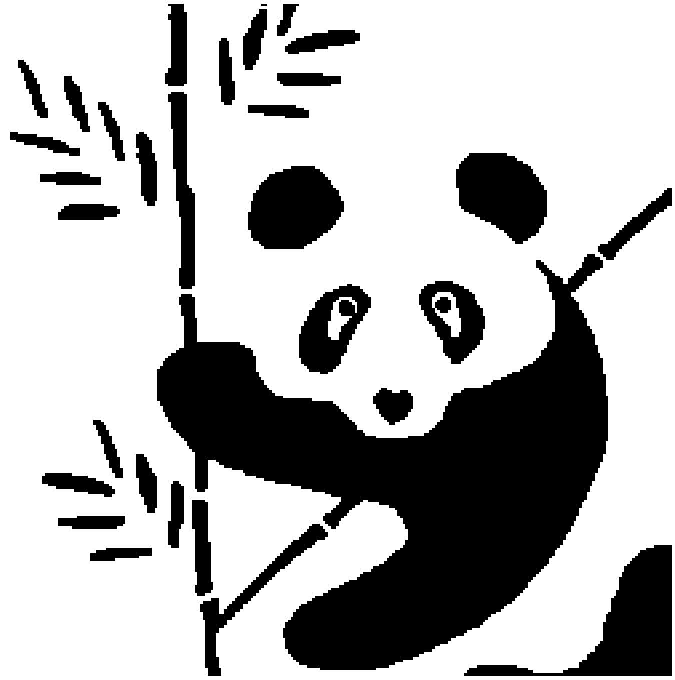
The *Panda* test image is a useful example to explore the consequences and uses of the various settings. For example, an erosion with a coefficient of 7 will remove only isolated single pixels without affecting anything else, while an opening with a coefficient of 1 will remove the fine lines and a closing will fill small gaps to connect the leaves to the branches. A coefficient of 0 (zero) corresponds to the traditional meaning of erosion and dilation, changing the color of any pixels that touch any neighbors of the opposite color. That is, in traditional erosion any white pixel adjacent to a black one becomes black, while in traditional erosion any black pixel adjacent to a white one becomes white.



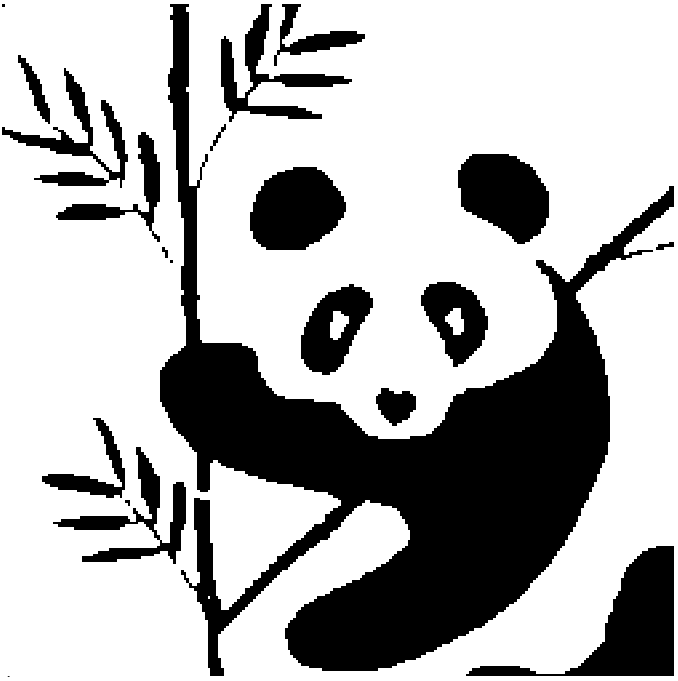
Original *Panda* image"



Erosion, Coefficient = 7



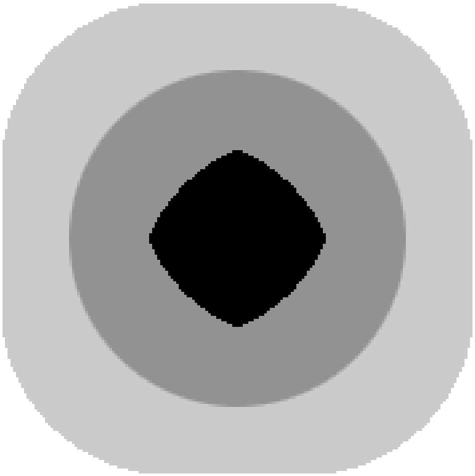
Opening, Coefficient = 1"

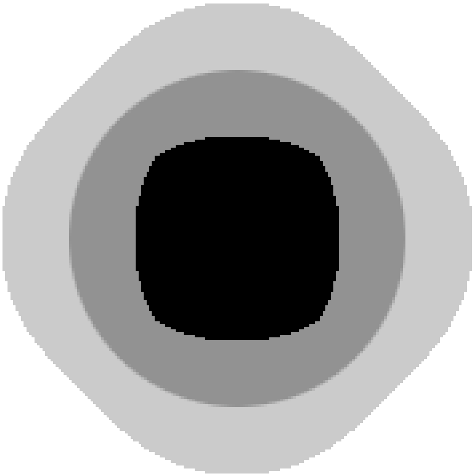


Closing, Coefficient = 1

5.A.2. EDM based morphology to remove small features or protrusions, and fill in gaps

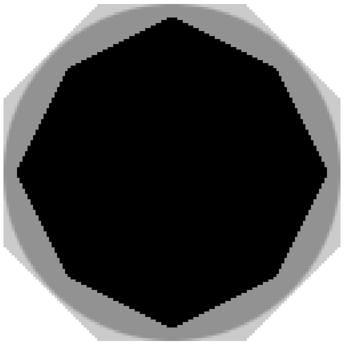
Classical erosion and dilation alter shapes when the number of iterations is large. Erosion and dilation based on the Euclidean distance map are much faster operations, and more isotropic than classical pixel-based techniques. The distance map assigns values to pixels within features that measure the distance to the nearest background point and thresholds the result. Eroding or dilating the example circle with **IP• Morphology –>EDM Morphology** produces circles of larger or smaller size without the distortions produced by the classical neighboring-pixel technique. In the illustrations below, Photoshop layers have been used with partial transparency to show the results of 25 iterations or erosion and dilation of the original circle, comparing the classical iterative method with the EDM technique. Notice also that the classical technique produces results of different size with the same number of iterations, depending on the neighbor test coefficient.

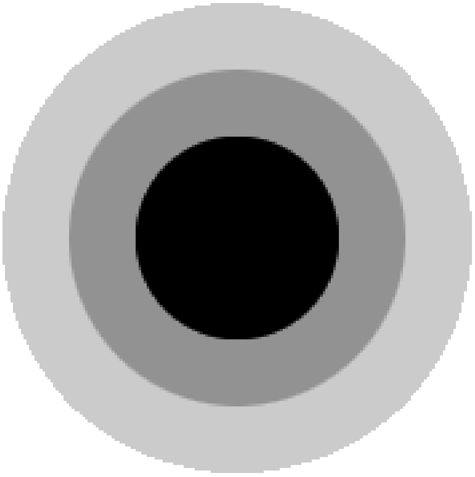




Coefficient = 0 "

Coefficient = 1

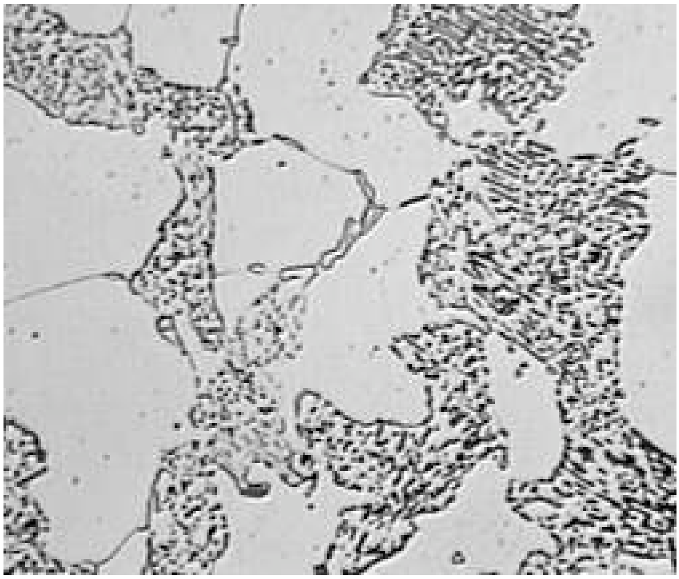




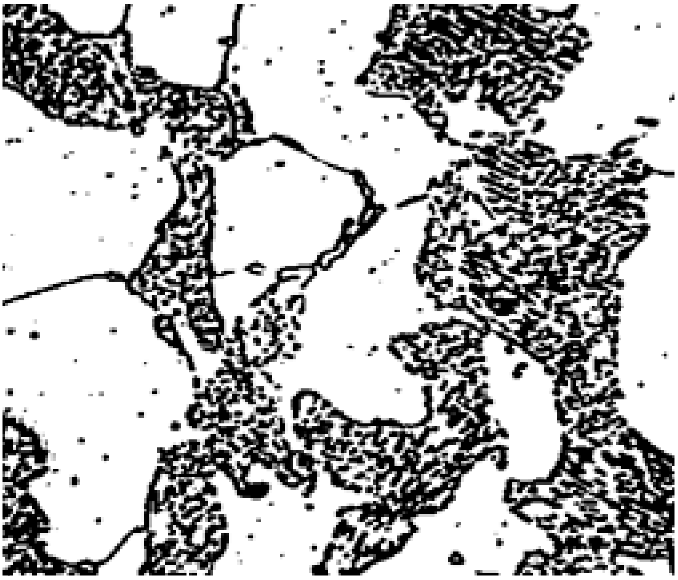
Coefficient = 3"

EDM-based results

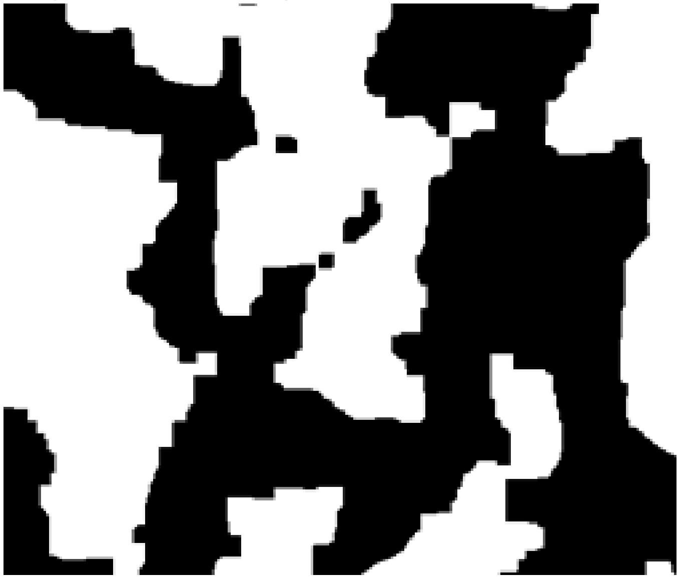
Closings and openings can be used in combination to clean up images after thresholding, as shown in the example below. The structures of interest are the region with the lamellar plates and the single-phase region without the lamellae, but simple thresholding only delineates the lamellae. Filling the spaces between the lamellae with a closing, and then removing the isolated dark spots with an opening, produces the desired image of the regions. Doing this with classical morphology (coefficient = 0, closing with 6 iterations and opening with 4) produces boundaries that have preferred directions, while EDM-based morphology (distance of 5.5 pixels) produces a more realistic result. Note that while classical erosion and dilation is limited to an integer for the number of iterations, the EDM method accepts real numbers since the distance of each pixel from the boundary is measured exactly.



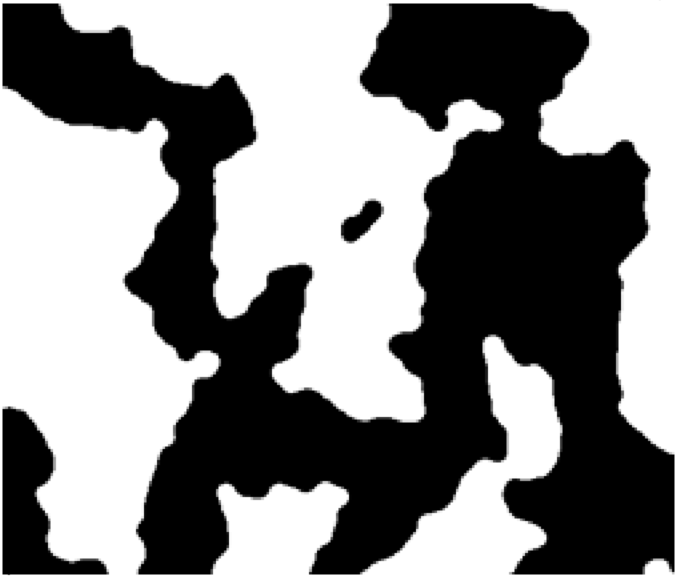
Original *Pearlite* image"



Thresholded binary



Classical closing and opening"



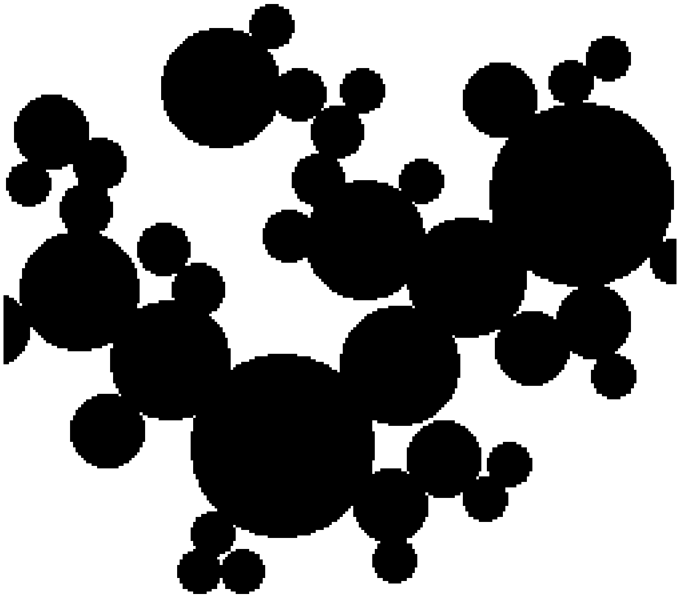
EDM-based closing and opening

To minimize confusion, be aware that there are four “morphology” plug-ins provided, each useful for different situations. **IP•Morphology–>Classic Morphology** is used only for binary (black and white) images and counts the immediate neighbor pixels that are different from the central pixel to determine its fate. It is most useful for removing lines and points by controlling the coefficient, and is best used with a small number of iterations. **IP• Morphology –>EDM Morphology** is also applied to black and white images, and uses the Euclidean distance map to determine distances. It produces the most isotropic results when large distances are needed. **IP•Rank–>Grey Scale Morphology** replaces the central pixel by its brightest or darkest neighbor, and is used for a variety of operations on grey scale images. If applied to a binary image, it is equivalent to classic morphology with a coefficient of 0. **IP•Rank–>Color**

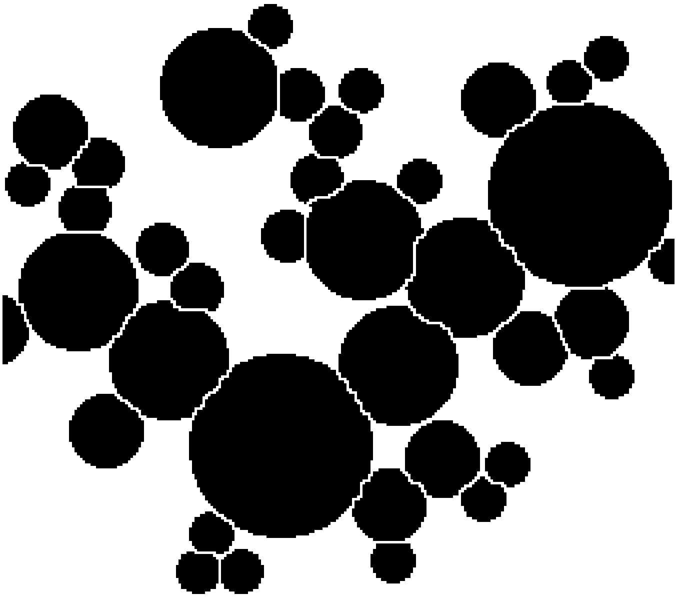
Morphology replaces the central pixel by the neighbor that is most like or unlike the current foreground color, and is used on color images.

5.B. Separating features that touch

The watershed, based on the EDM, is a powerful tool for separating touching convex shapes. The method is useful for section images and surface images in which features are adjacent. If there is slight overlap, the feature shapes may be altered due to the overlapping portion being cut off. If the overlap is too great the method will fail, and in three-dimensional samples, small features may be hidden by large ones.

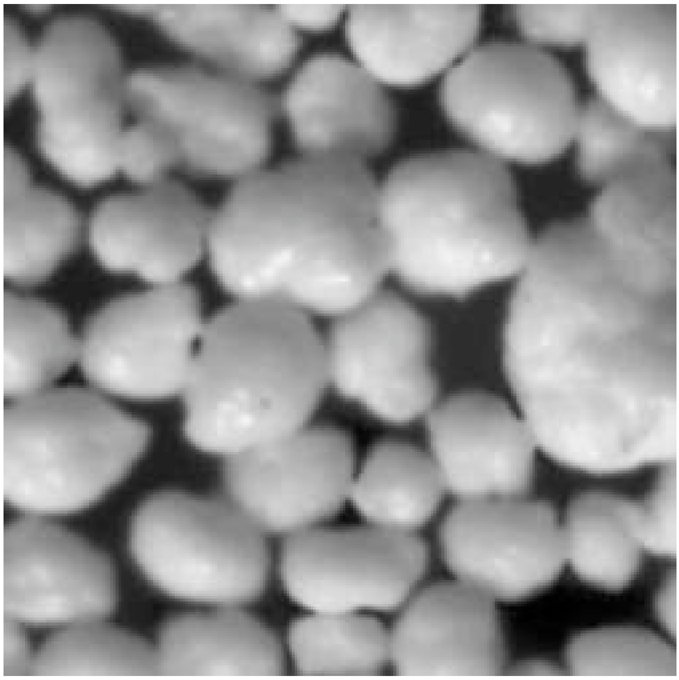


Original *Circ_Mix* image (fragment)!

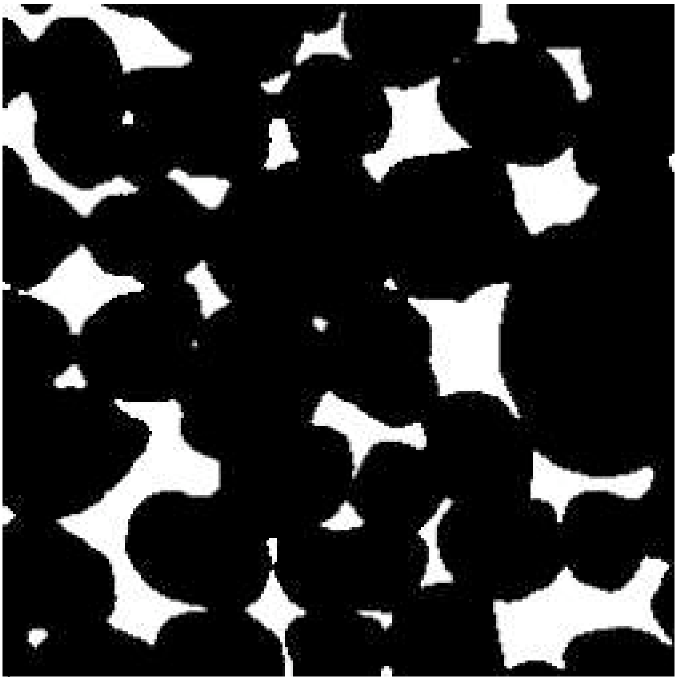


IP• Morphology –>Watershed Segmentation result

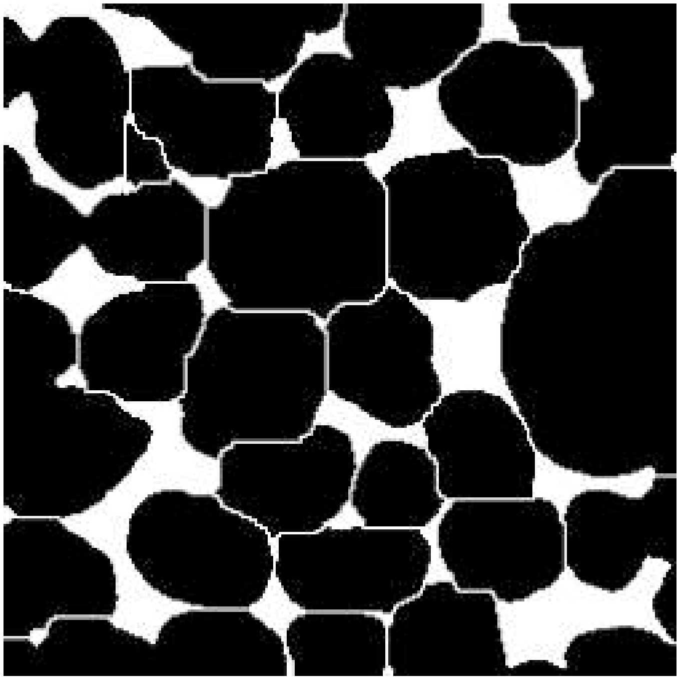
It is often useful to verify the results of thresholding and segmentation by overlaying the result on the original grey scale or color image. This is easily done using Photoshop layers. In the example the outlines of the watershed-segmented particles are superimposed on the original.



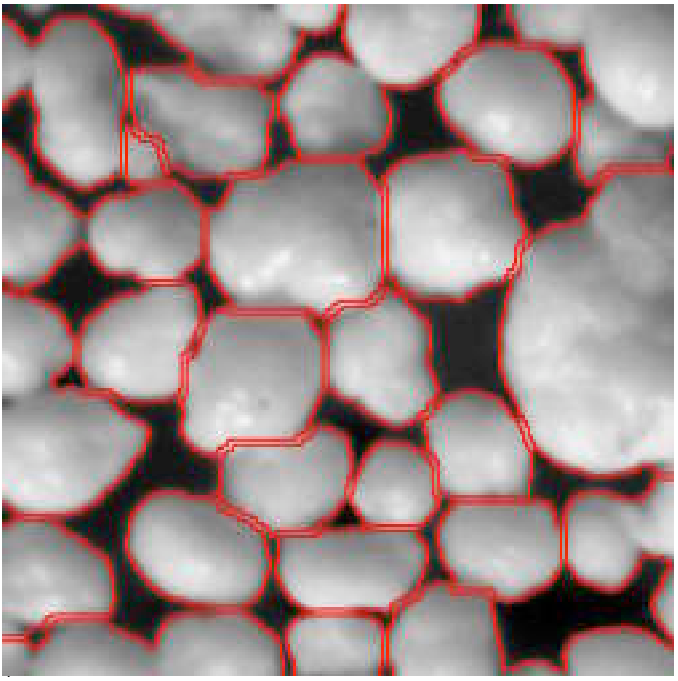
Original *Sand* image"



Thresholded

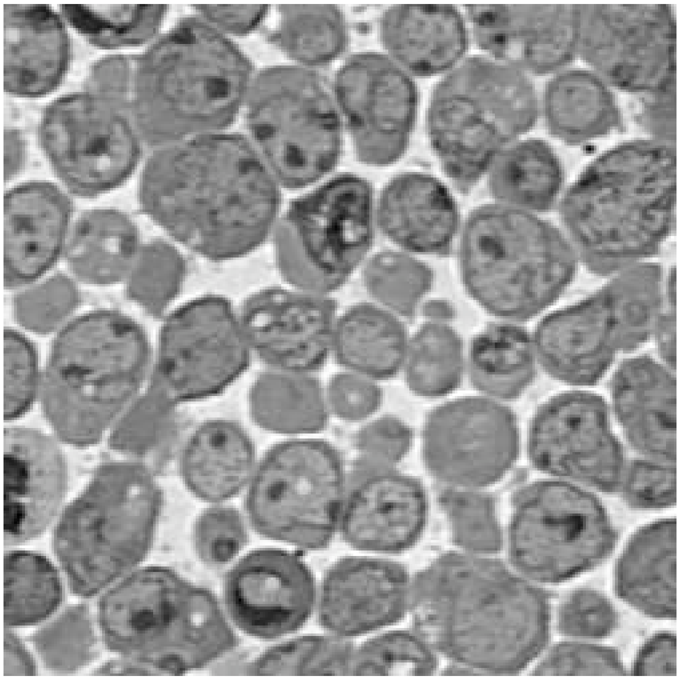


Watershed"

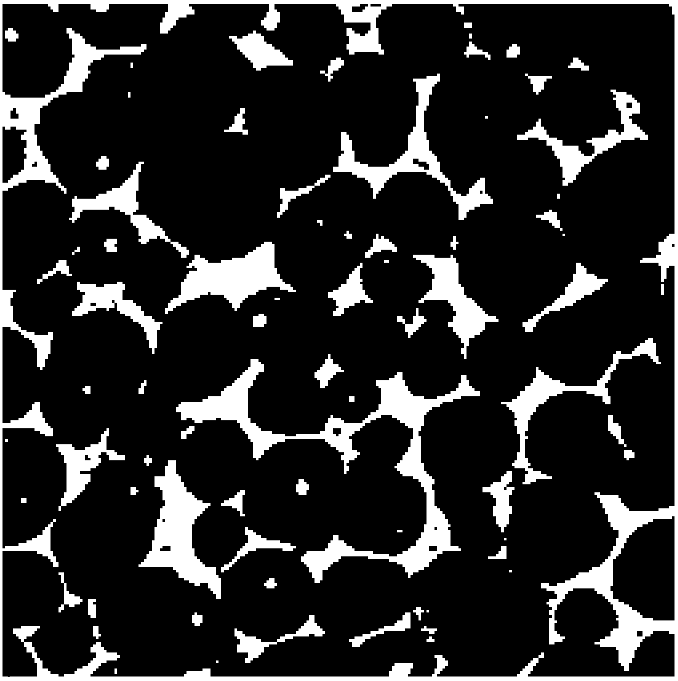


Outlines superimposed on original

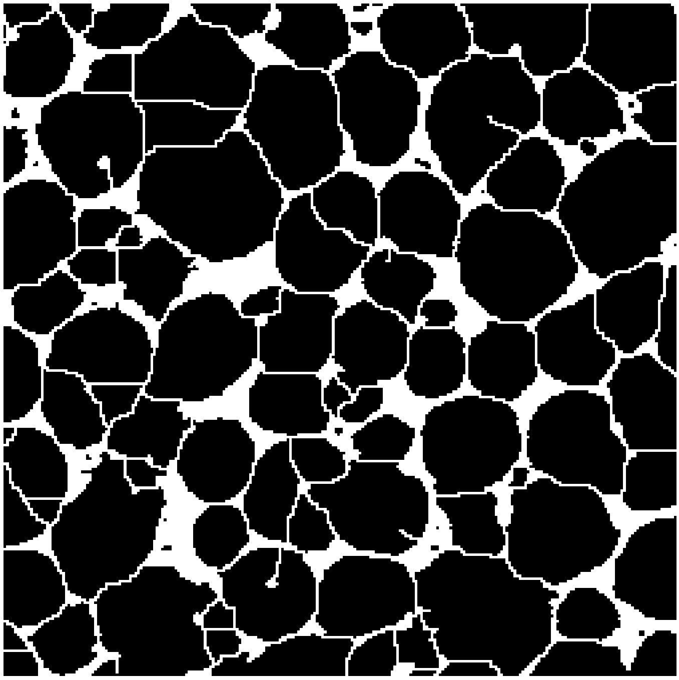
The watershed method fails when the touching features contain holes, as shown in the example below. However, in this case it is possible to distinguish the holes within the features (which are fairly round) from those between the features (which are not) by measurement. The procedure is to measure the holes as features (Section 6.E), select the round ones (Section 6.F), combine (Boolean OR, Section 5.C) those with the original thresholded image, and then perform a successful watershed segmentation of the cells.



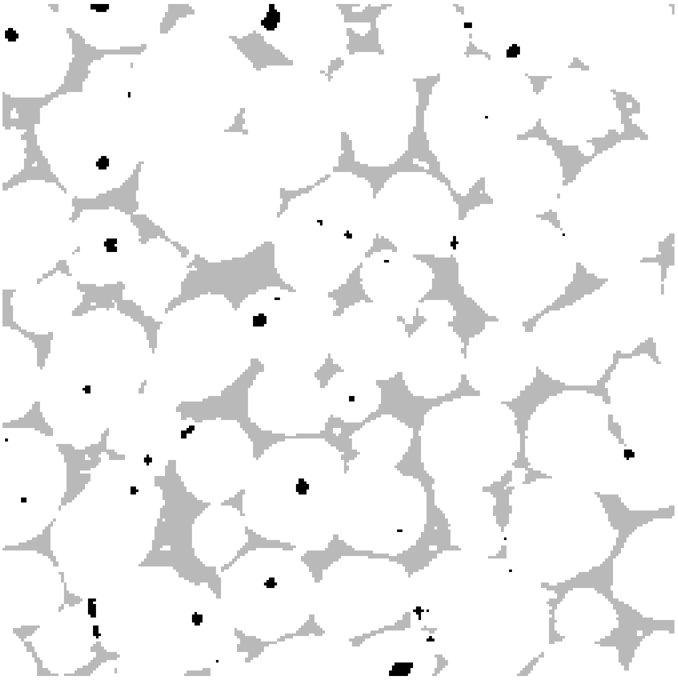
Original *Cells* image"



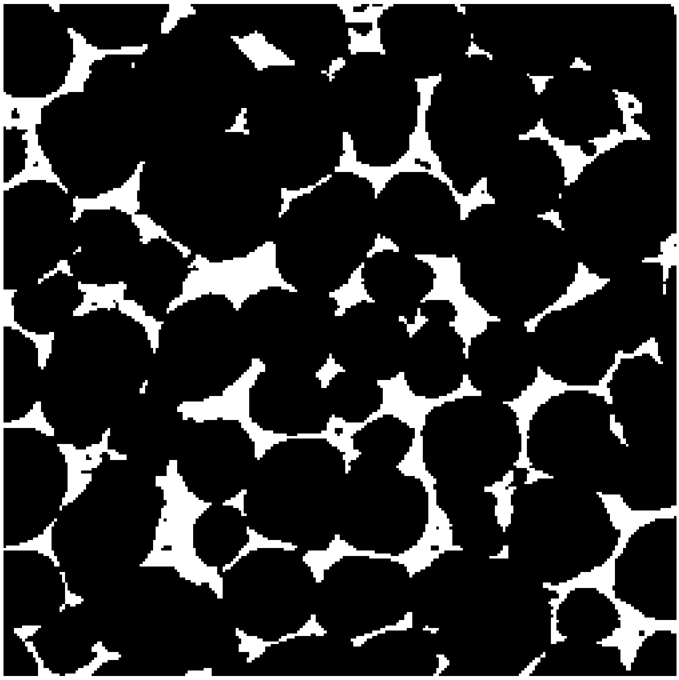
Thresholded



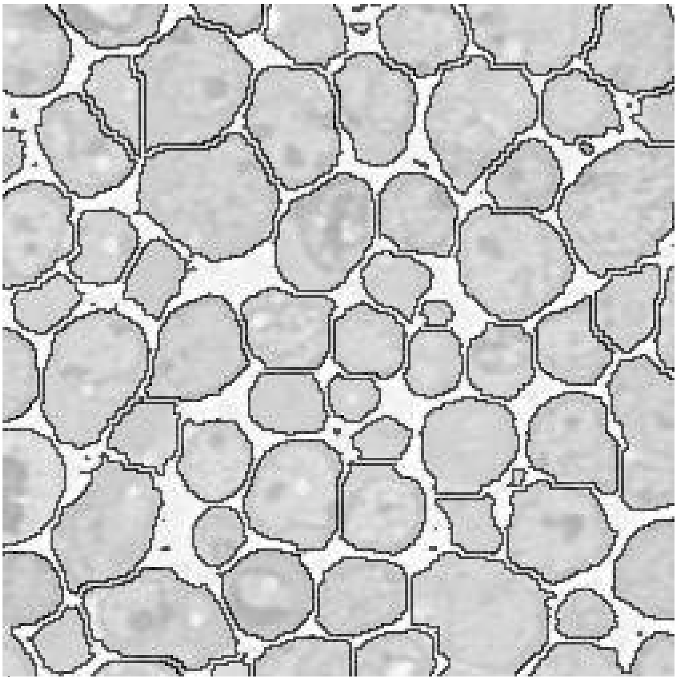
Unsuccessful watershed attempt"



Measuring the holes (dark ones are kept based on shape)

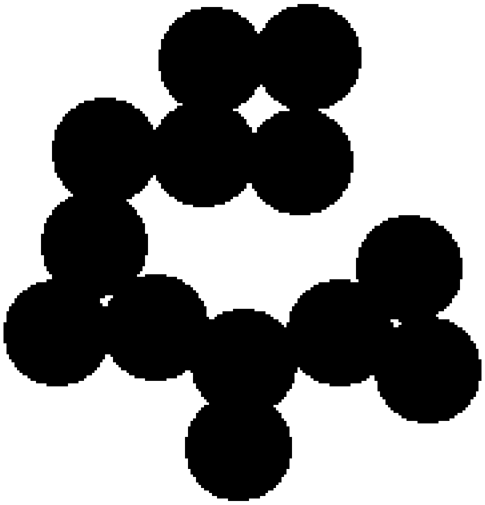


Combining the round holes with the original binary"

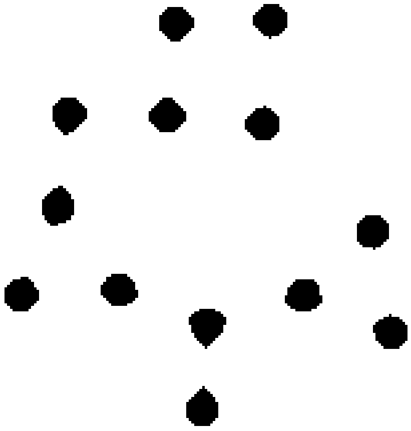


Watershed segmentation (shown as outlines)

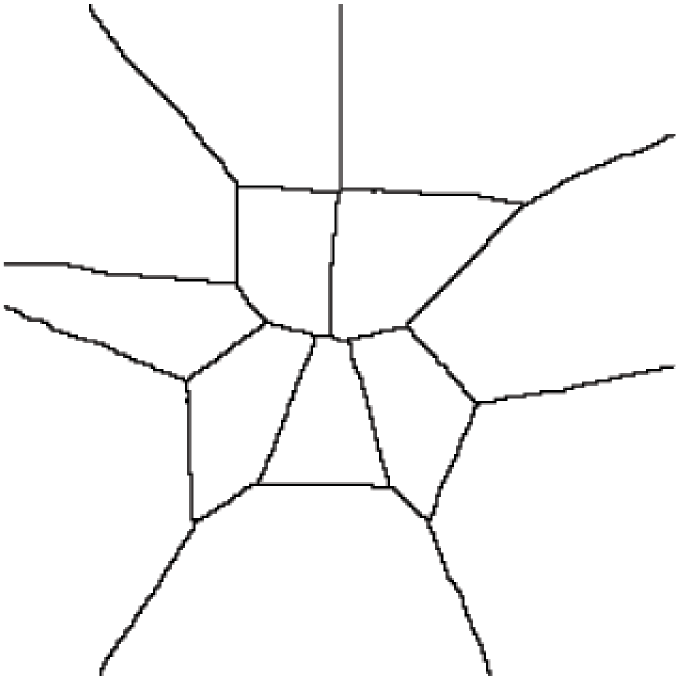
Alternative but less commonly used methods rely on successive erosions to separate features. For example, eroding the circles in the example shown until they separate, then skeletonizing (discussed in Section 5.D.1 below) the background and erasing those lines from the original separates the features. However, this approach would not work for the *Circ_Mix* image above because the range of sizes would result in the removal of some features before others separate.



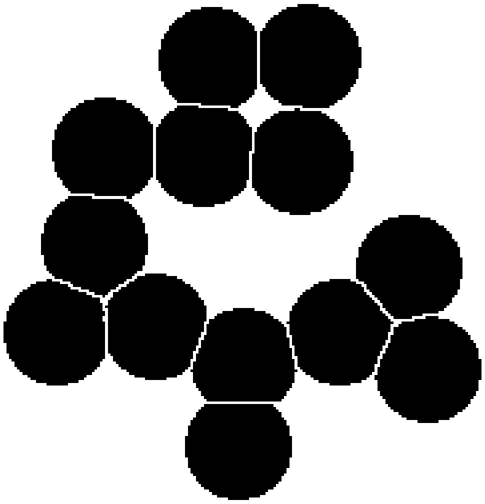
Original *Circles* image"



Eroded until features separate

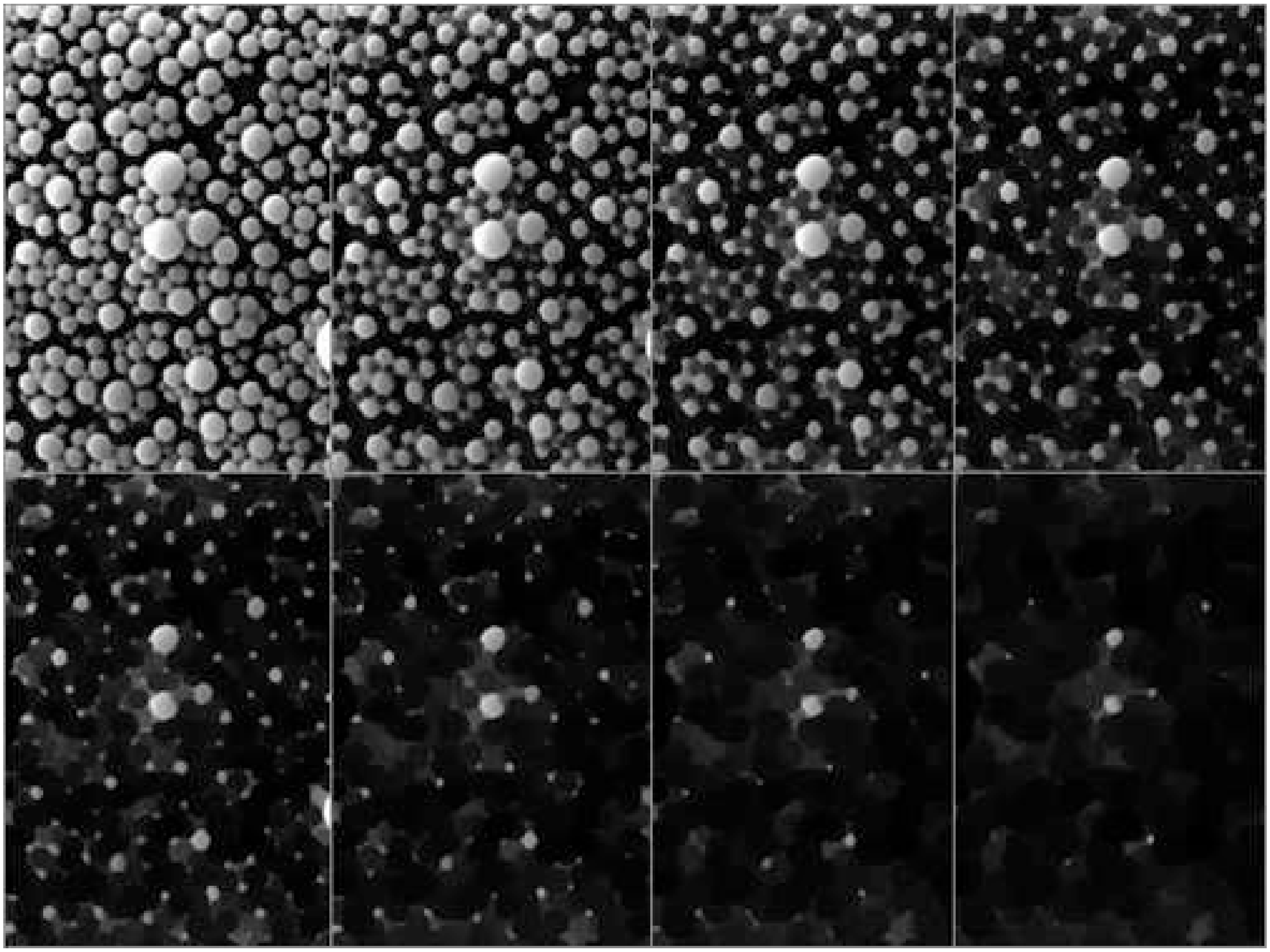


Skeleton of the background"

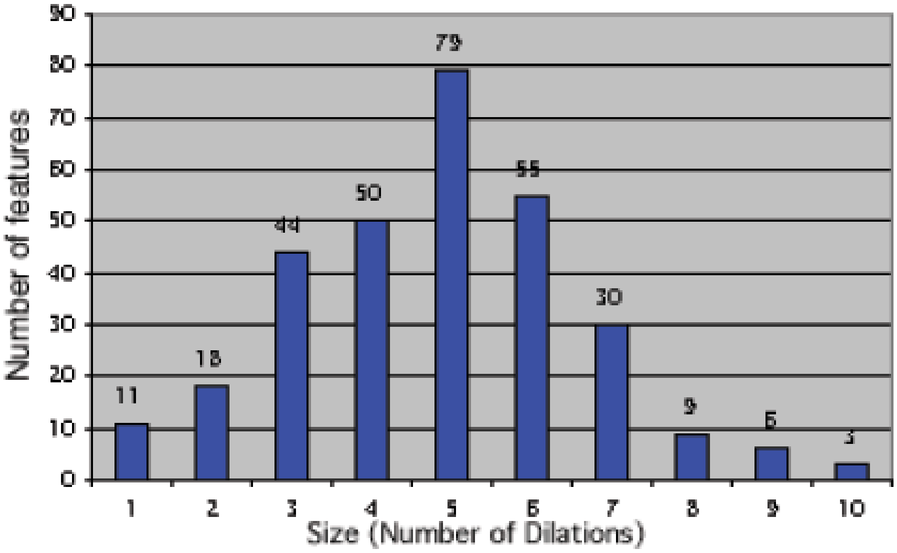


Removal of the skeleton lines from the original

Another technique that is sometimes used to determine a distribution of particle sizes employs grey scale erosion or dilation (see Section 2.C.4). In the example shown, replacing each pixel by its darkest neighbor (grey-scale dilation) reduces the size of features and eventually causes them to disappear. Thresholding the image at each stage and counting the number that disappear (which requires some Boolean logic that is described below) provides a measure of the number of particles with a radius equal to the corresponding number of erosions, and allows a size distribution to be constructed. The assumptions in this method are that the particles separate before they disappear, and that they are sufficiently round that the radius measurement obtained from the number of iterations adequately describes their size.



Sequence of grey scale dilations applied to the *Lipid* image (fragment)

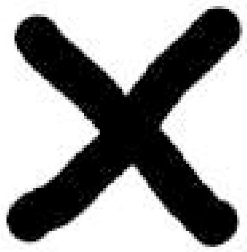


The resulting size distribution plot for the particles

5.C. Combining multiple images of the same area to select features

5.C.1. Boolean logic to apply multiple criteria

Several of the preceding sections have made incidental use of Boolean combinations of images. For different color channels, or images thresholded and processed differently, Boolean logic can combine the information and isolate the structures of interest. The Boolean functions are AND, OR, Exclusive-OR and NOT (equivalent to inverting an image so that black becomes white, and vice versa), which can be combined in many ways. The examples shown illustrate a few of the ways that these operations can be combined. Note that the **IP•Math–>General Boolean** plug-in treats white (background) pixels as “off” and any non-white value as “on.”



"   Image A"

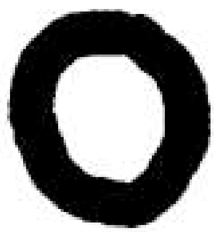
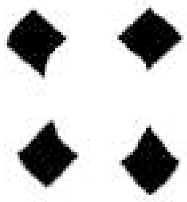
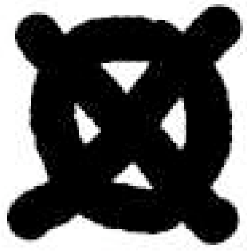


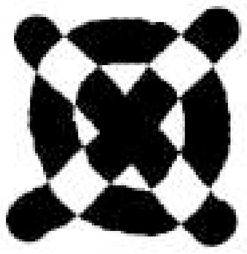
Image B"



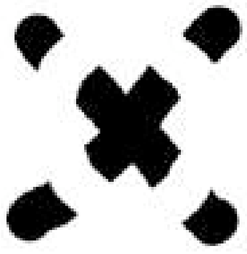
A AND B"



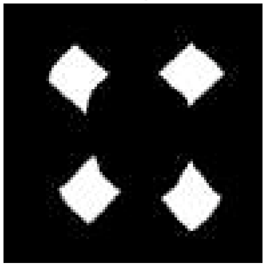
A OR B"



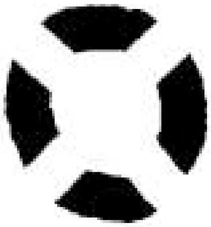
A Ex-OR B



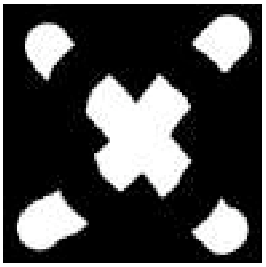
" A AND (NOT B)"



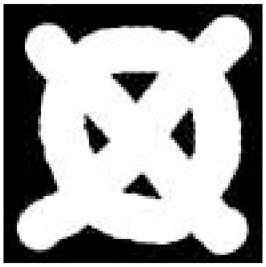
NOT (A AND B)"



(NOT A) AND B"

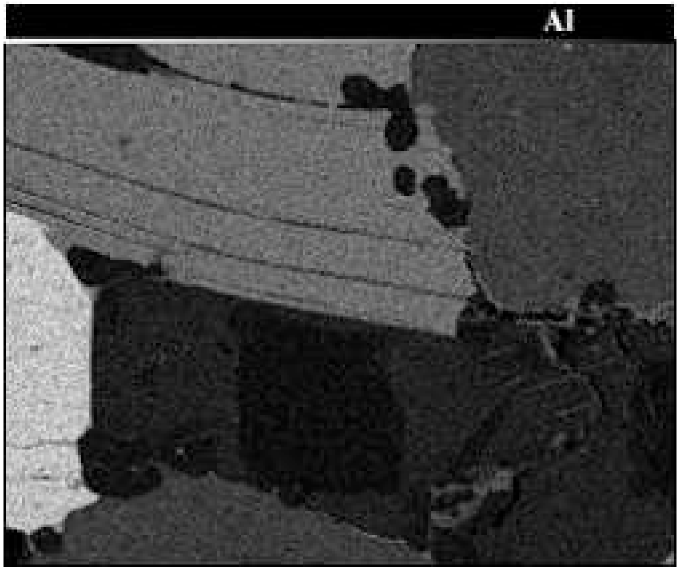


(NOT A) OR B"

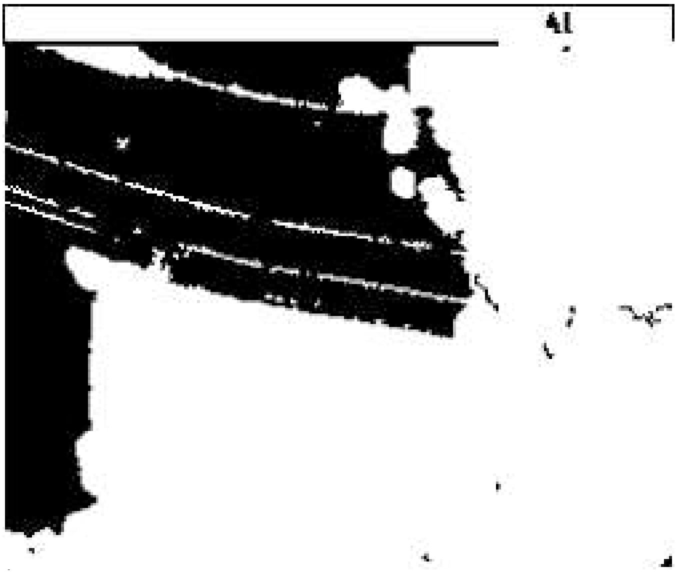


NOT (A OR B)

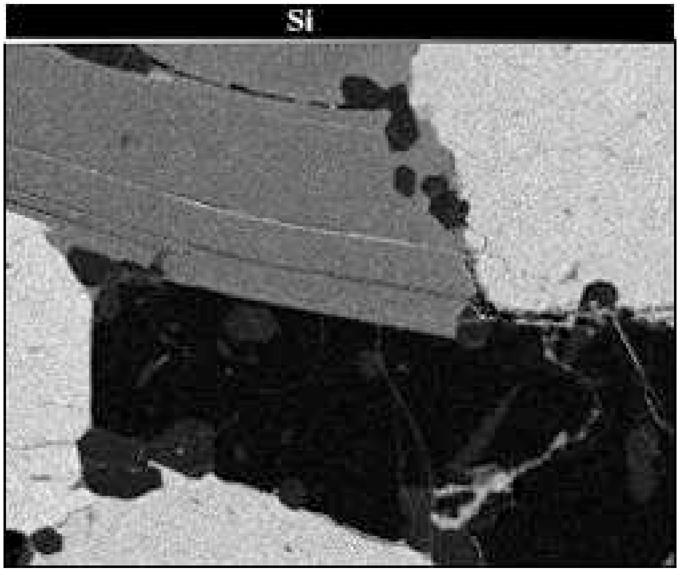
A typical use for these operations is to combine images thresholded from different color channels to select specific regions. The example shows the similar combination of different elemental maps from SEM X-ray images. The thresholded images were cleaned with a morphological closing using a coefficient of 6.



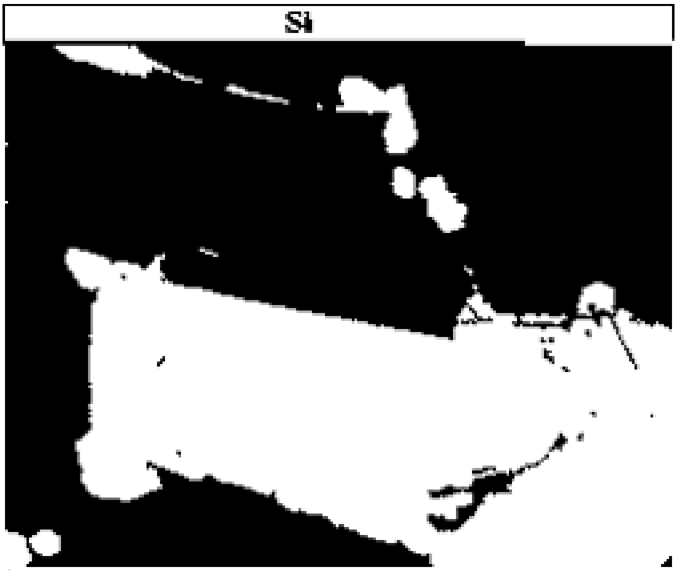
Original *Mica_Al* image"



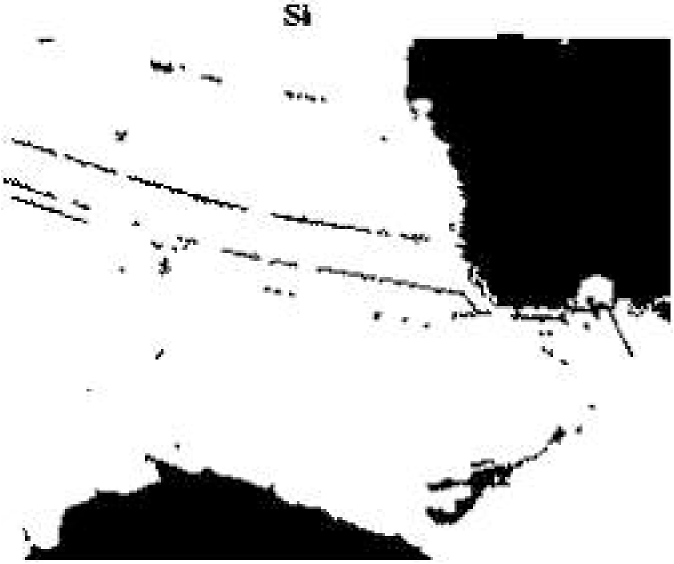
Thresholded for high Al content



Original *Mica_Si* image"

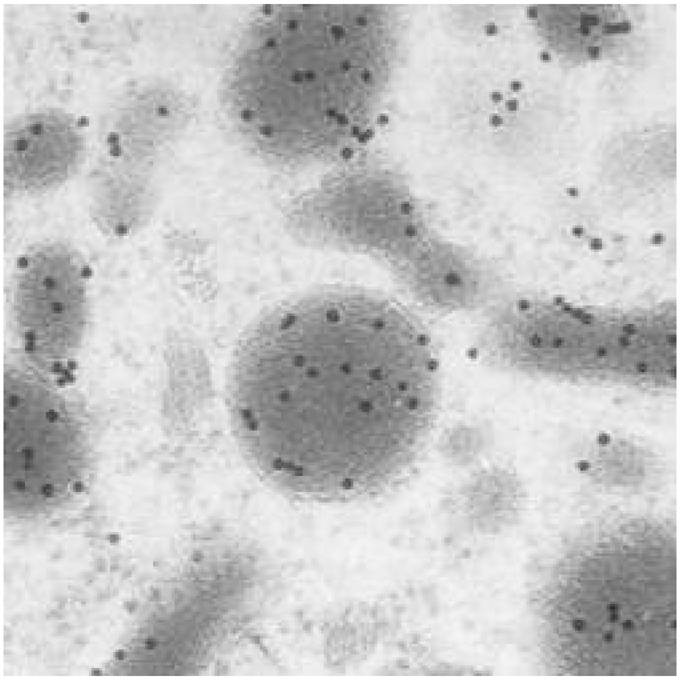


Thresholded for high Si content

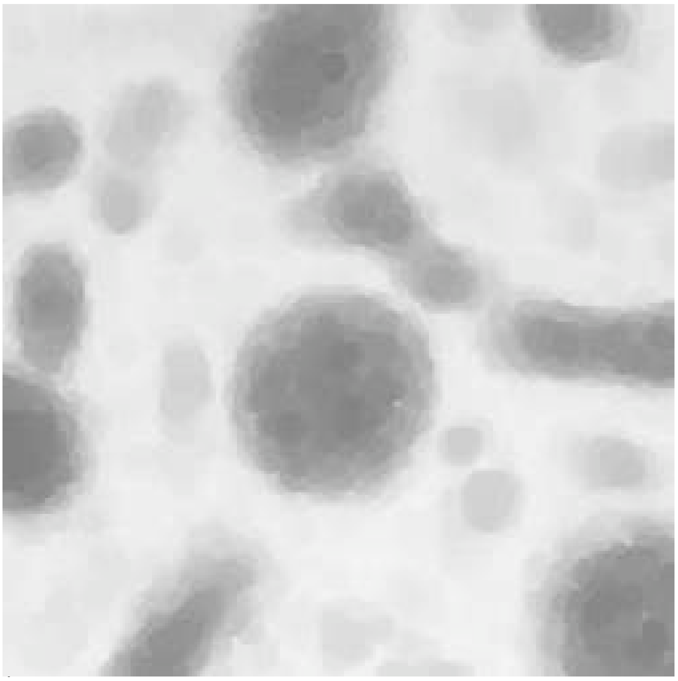


Silicon AND (NOT Aluminum)

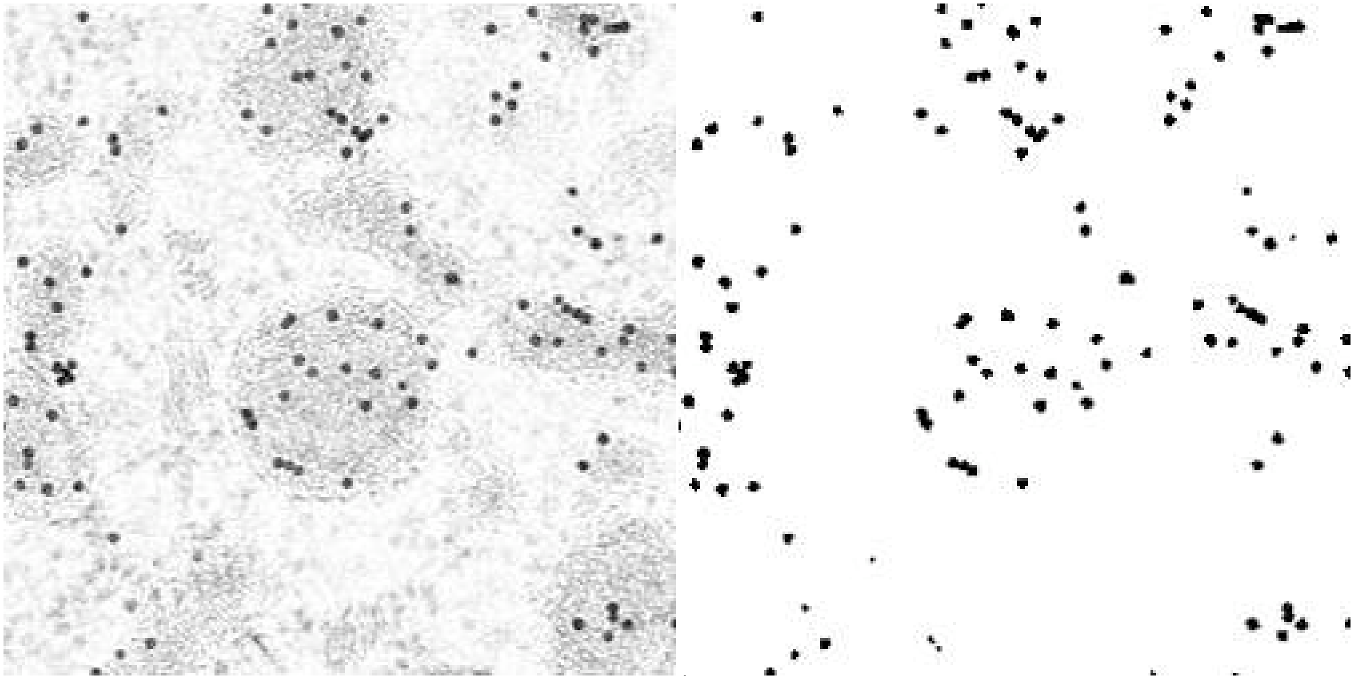
Boolean operations are also useful when the same original image has been processed in multiple ways to isolate different types of information, which are then combined. In the example, two structures (the organelles and the gold particles) are isolated separately, but only those gold particles that lie on organelles are of interest, and are selected with a Boolean AND.



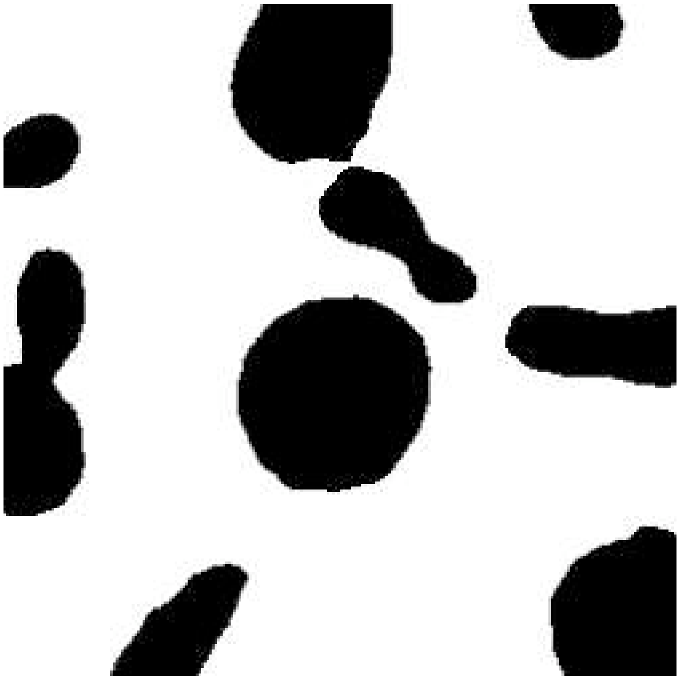
Original *Gold2* image!



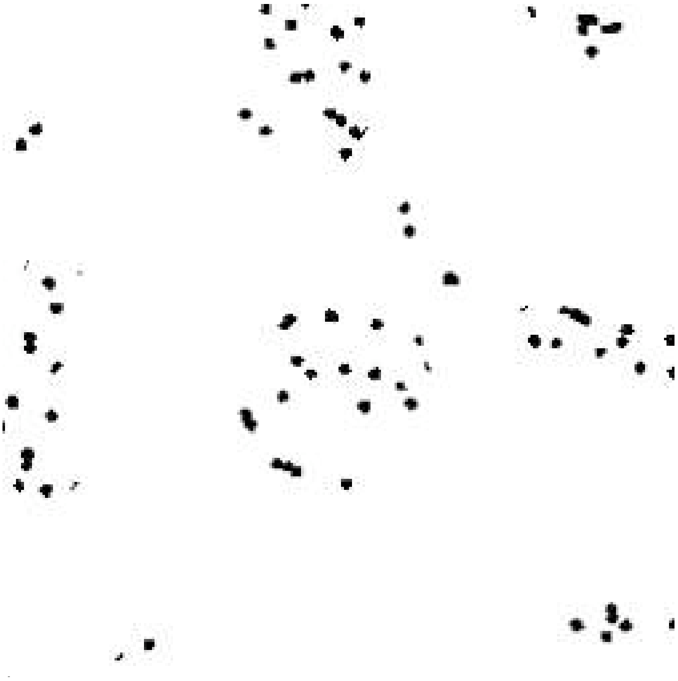
Grey scale opening removes gold particles



Original divided by background isolates gold particles, which are then thresholded

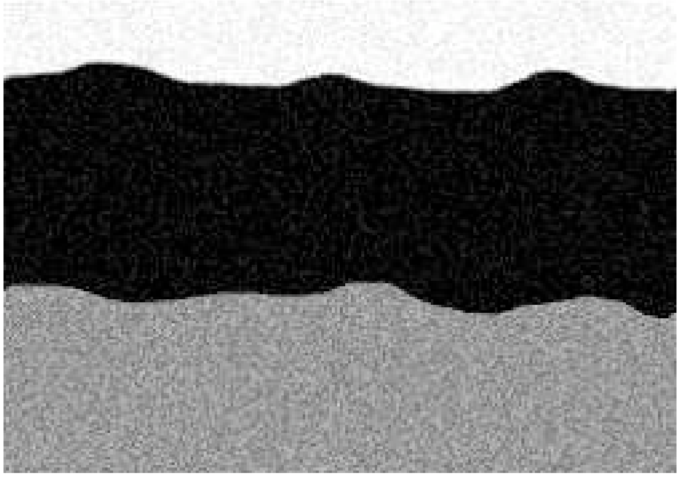


Thresholded organelles"

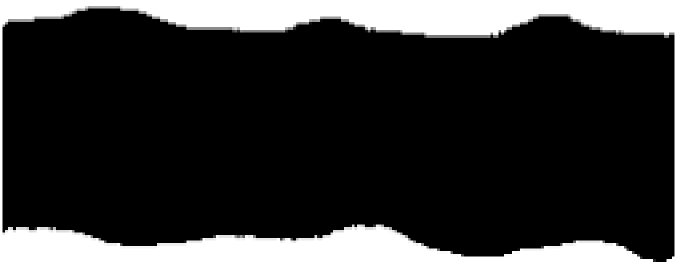


AND selects gold particles on organelles

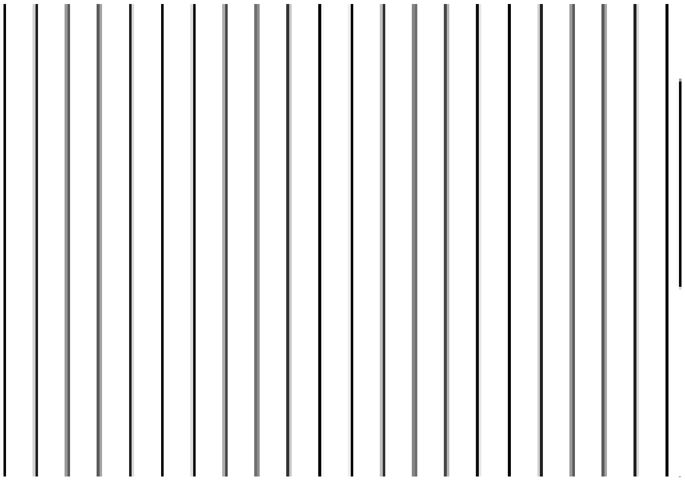
The Boolean AND operation is frequently used to combine various types of grids with thresholded binary images as a precursor to measurement. Many of the stereological techniques in Section 6.B use grids as a way to obtain concise and meaningful information about structure. But the use of grids is even more general than that. In the example shown, a vertical section through a coating is prepared for measurement. The most straightforward and efficient way to perform this measurement is to create a grid of vertical lines (**IP•Lines and Points–>Parallel Lines–>Vertical**) with an spacing appropriate to the desired density of measurements, AND it with the coating image, and measure the length of the lines.

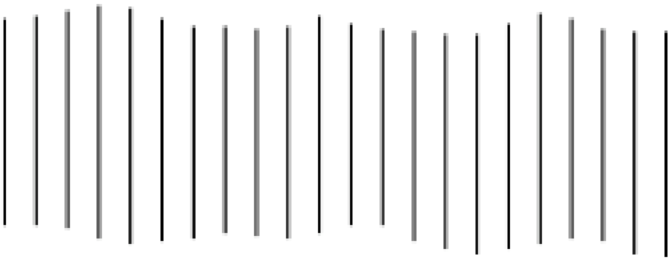


Original *Coating* image"

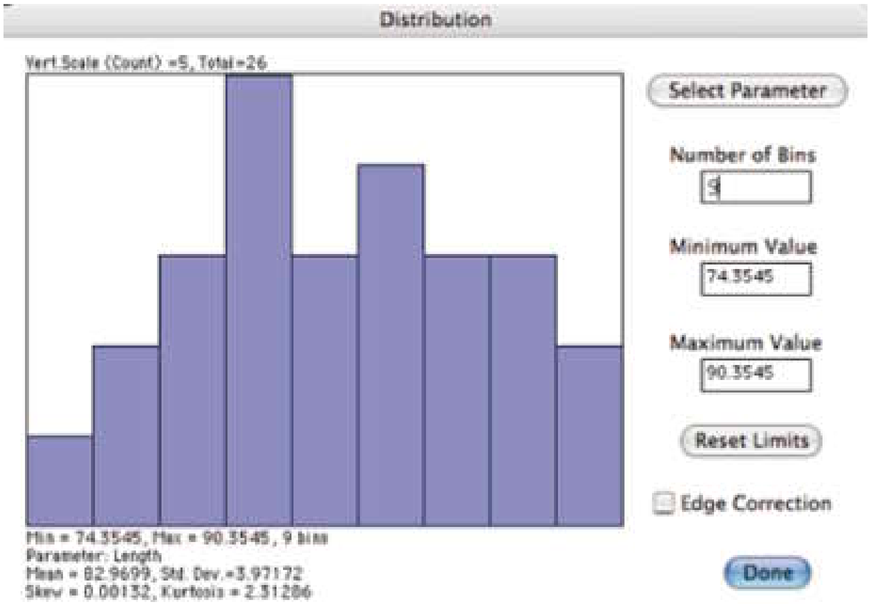


Thresholded layer cross-section





Vertical line grid" ANDed with layer



Vert.Scale (Count) =5, Total=26

Number of Bins: 9

Minimum Value: 74.3545

Maximum Value: 90.3545

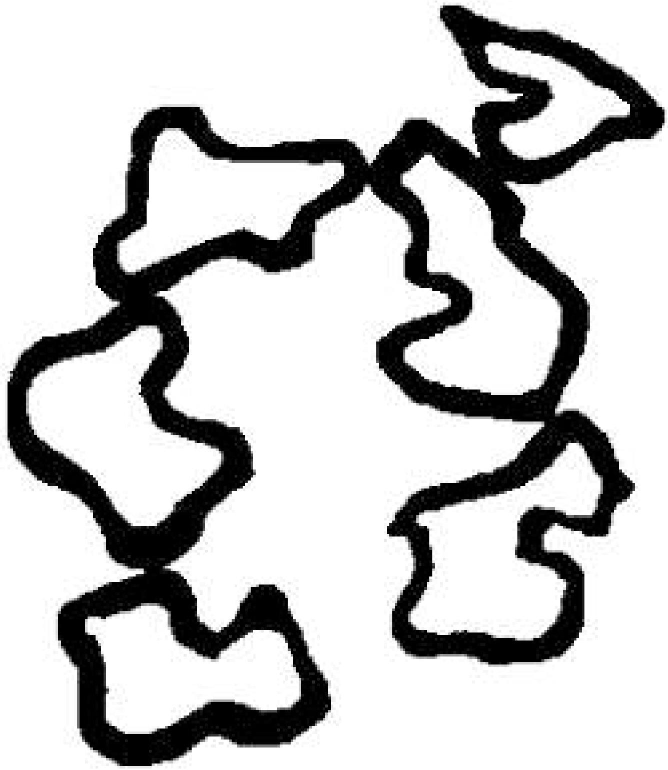
□ Edge Correction

```
Min = 74.3545, Max = 90.3545 , 9 bins
Parameter: Length
Mean = 82.9699, Std. Dev.=3.97172
Skew = 0.00132, Kurtosis = 2.31286
```

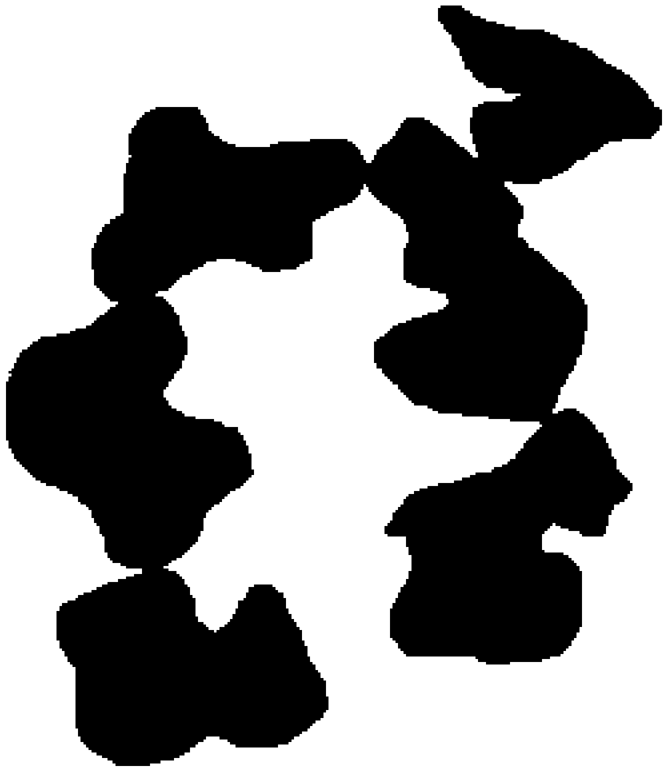
Measurement results

As an illustration of the various Boolean operations used in combination, and the way they can be used in conjunction with other binary processes, the multi-step example below addresses a situation in which the watershed segmentation method does not work: features that are hollow and irregular (non-convex) in shape. The image was thresholded to show the sheaths around nerve fibers. The goal is to separate them for individual measurement. One step (F) in the procedure uses a Feature-based AND rather than the Pixel-based AND used in the other examples. The **IP•Math–>Select Features by 2nd** routine is discussed below. The selection of the feature-crossing lines in step F can also be accomplished by measurement, keeping the longer lines.

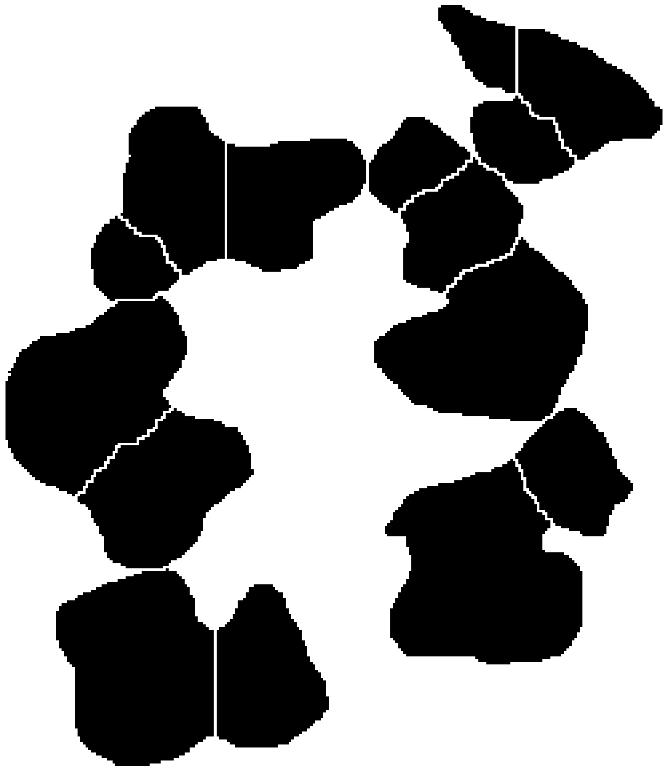
This sequence may at first appear to be long and complex, but these Boolean operations are very fast and the entire process is easily automated using an Action. Understanding this sequence is a good test for having an understanding of Boolean operations that lie at the heart of many binary image processing sequences.



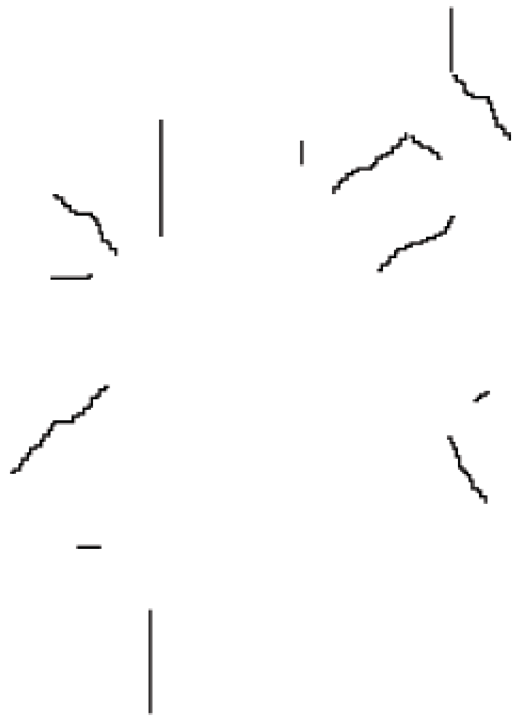
A-Original *Separate* image"



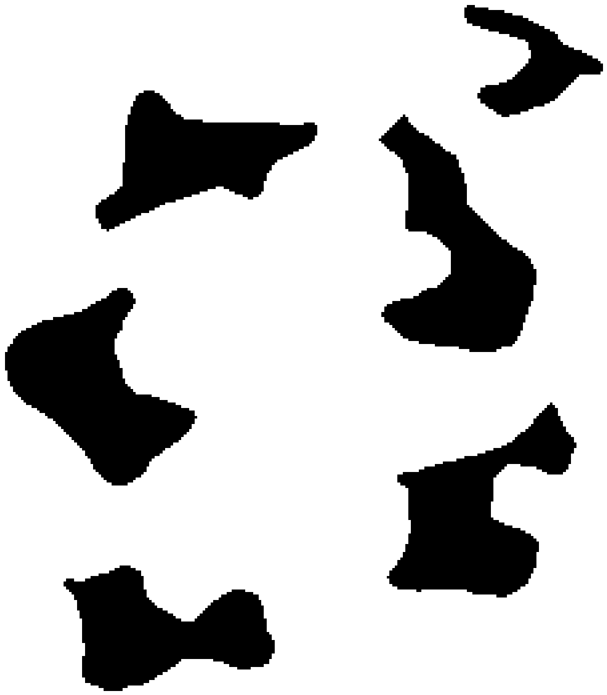
B-Holes filled



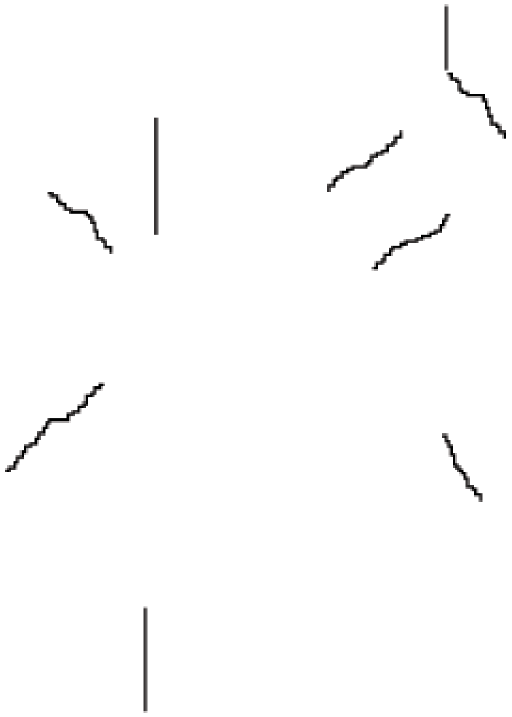
C-Watershed"



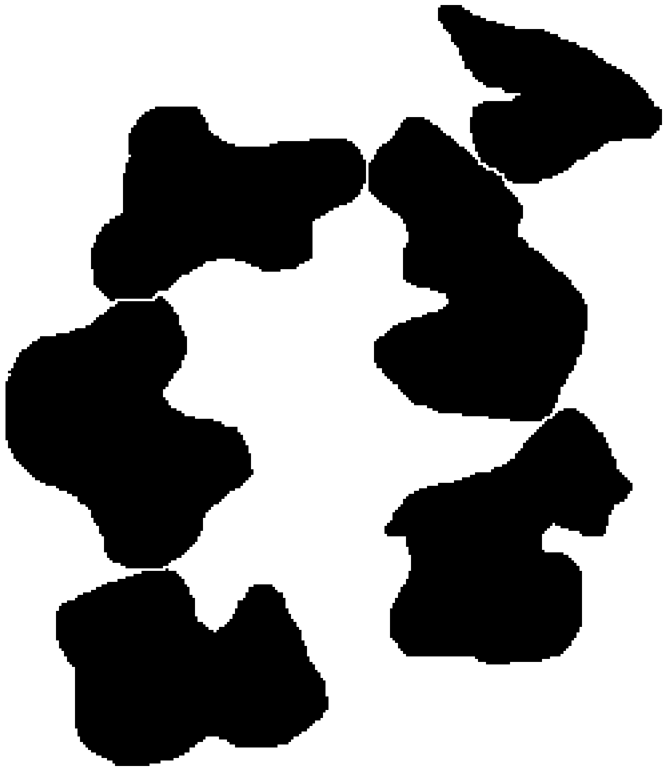
D-Lines (B Ex-OR C)



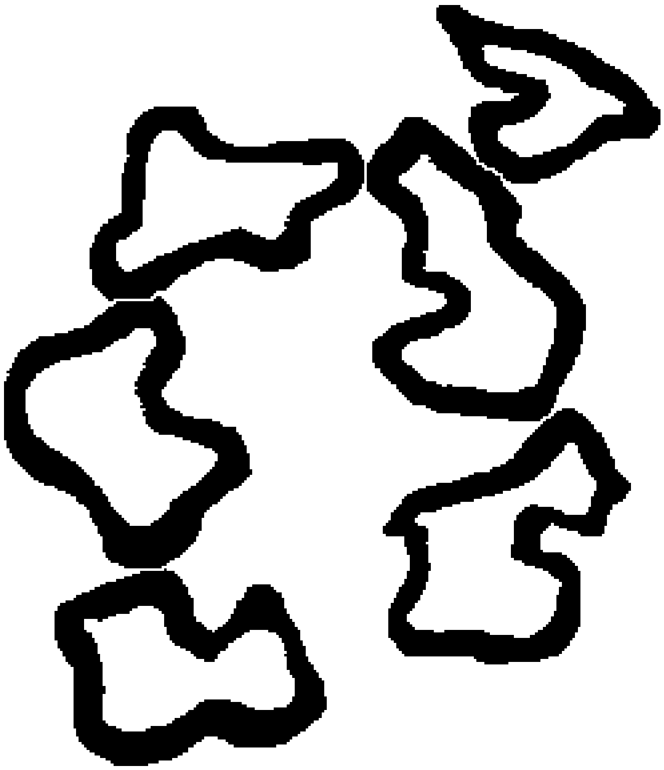
E- Holes (A Ex-OR B)"



F- Lines in D selected by features in E



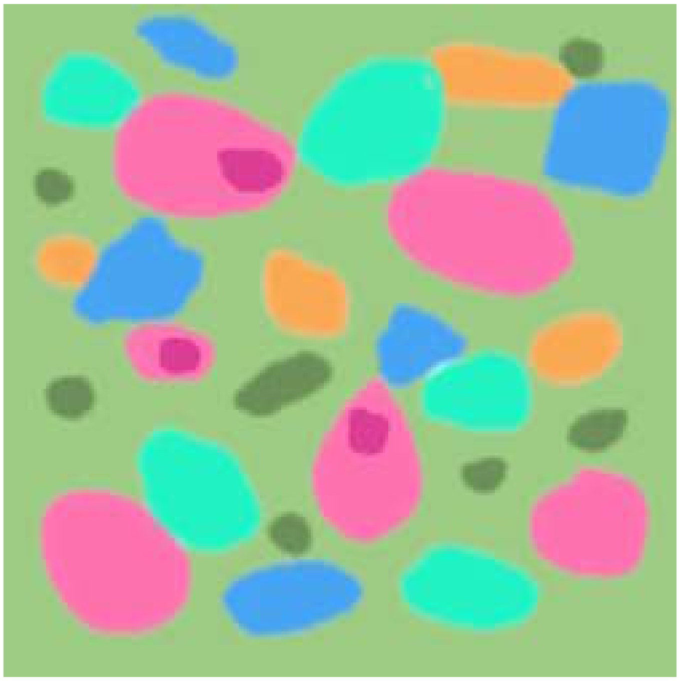
G- Breaks restored (C OR F)"



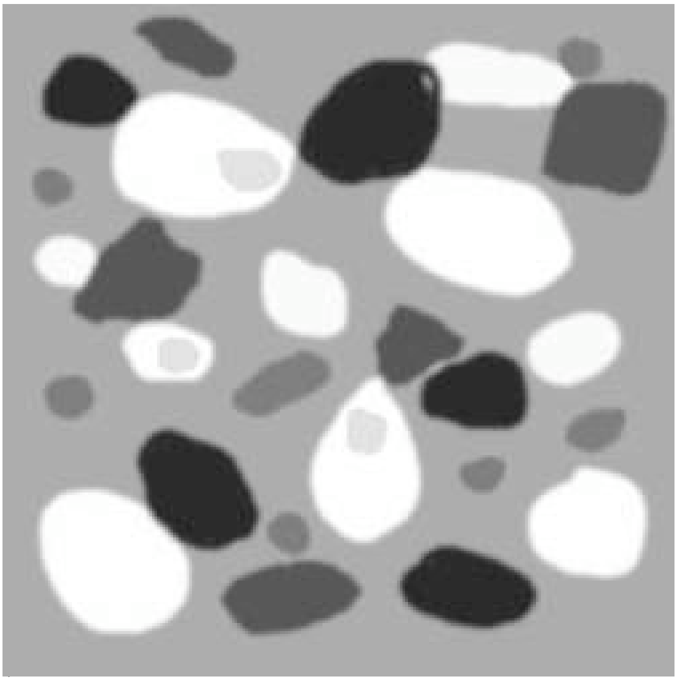
H- Features separated (G AND A)

5.C.2. Using markers to select objects

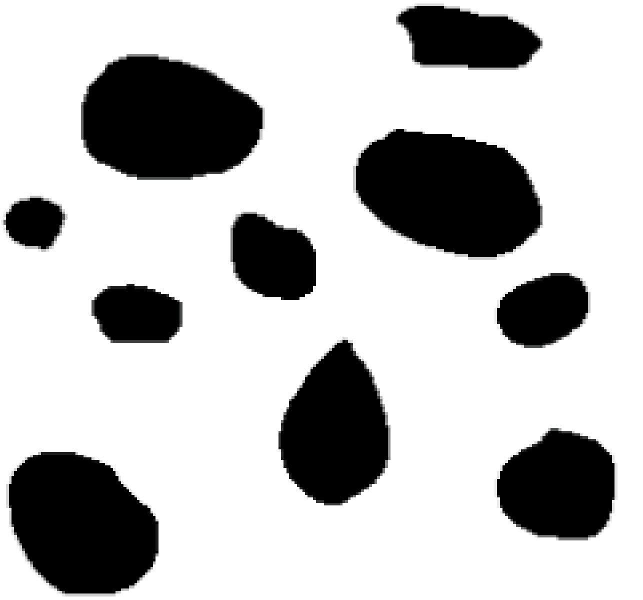
The conventional Boolean AND operation works on individual pixels. Unlike the conventional pixel-based logic, the method used in step F of the preceding example selects entire features in one image based on markers in a second image. In this procedure, using the **IP•Math–>Select Features by 2nd** plug-in, the order of images is important, unlike the conventional pixel-based Boolean operators, which commute (A AND B = B AND A). The use of markers in one image to select features in another is a very powerful and general tool. In the example, only those red features that contain a dark marker are selected. Note the difference between the marker selection result (red features that contain dark markers) and the conventional AND (dark markers that lie within red features) result.



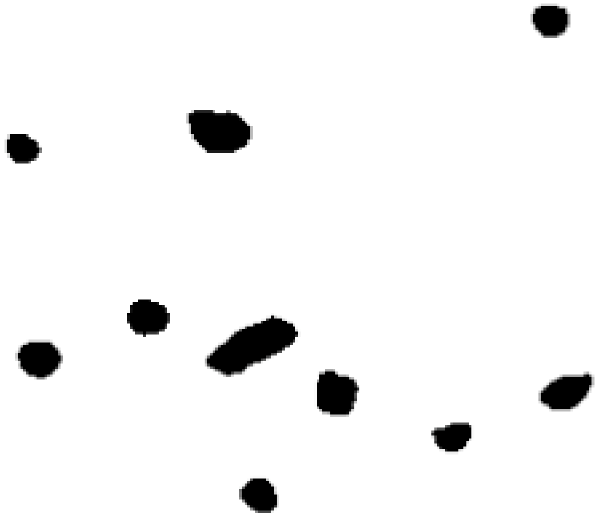
Original *ColrMark* image"



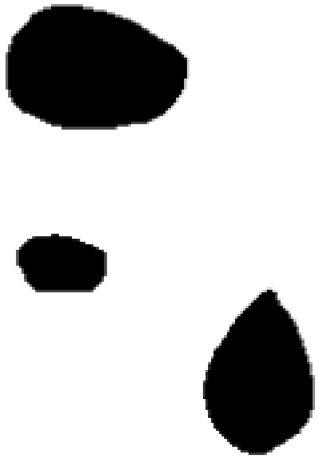
Red Channel

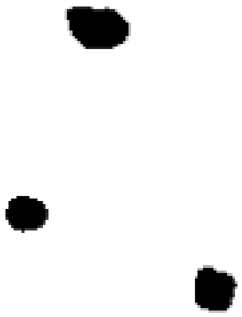


Red features"



Dark marks

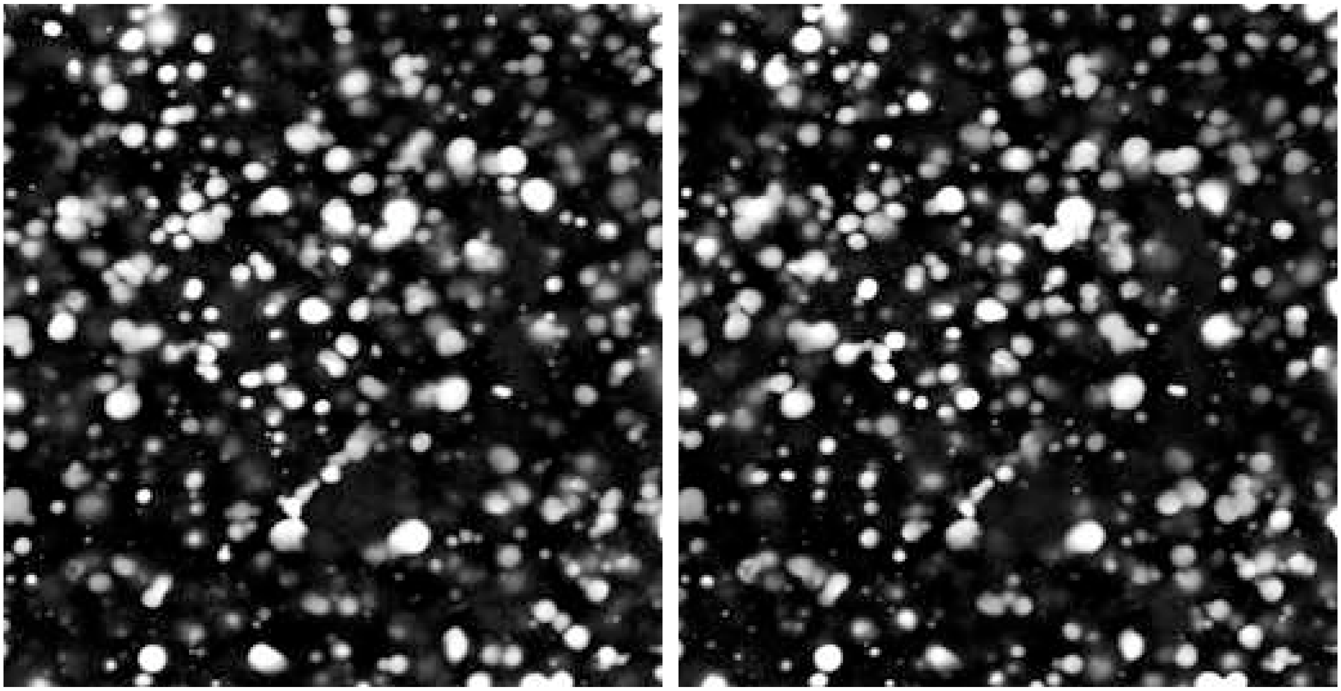




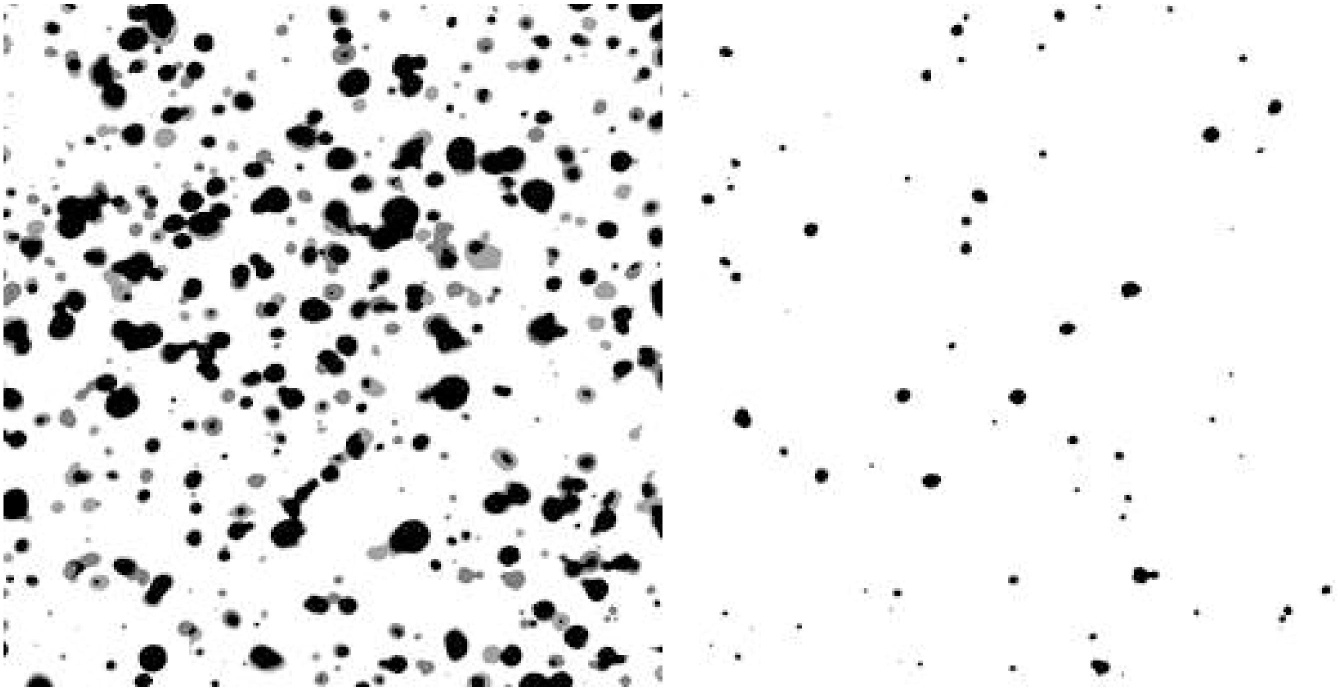
Feature selection result''

Pixel-AND result

Another important application for marker selection logic is implementing the stereological disector. This technique compares serial section images to count the features that appear in one image but are absent from the second. Those “events” represent bottoms of features, by which they can be counted. The number of events (unmatched features) divided by the product of the image area and the spacing between the sections (the volume examined) gives the number of features per unit volume directly.



Layers 1 and 3 from *Confocal*



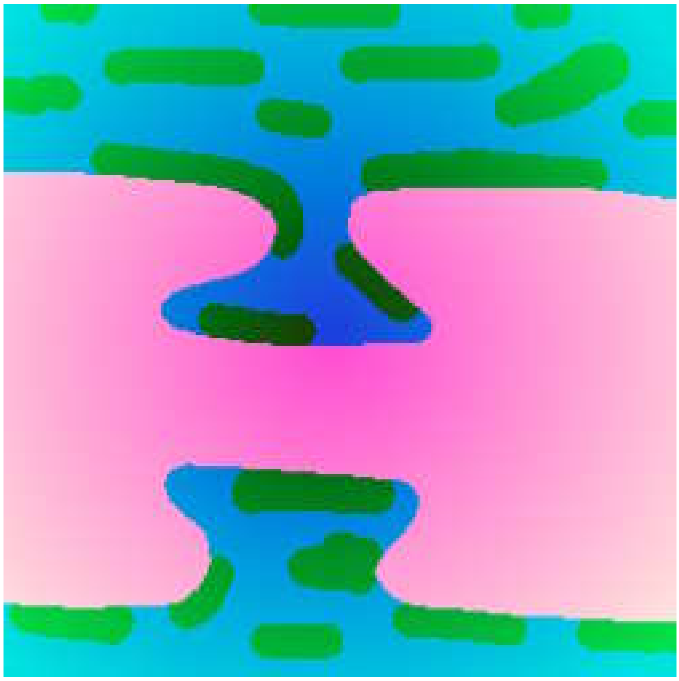
Overlay of the thresholded images with transparency showing the matched and unmatched features, and the result of using Layer 1 as a marker to select unmatched features in Layer 3 (Count = 74).

This comparison cannot be done using pixel-based Boolean logic, because the sections will not be the same size and shape in the two layer images. Marker-based selection has been used in the previous examples to find the features that are matched in two images, but now we must find those that are unmatched. This is easily done by swapping the foreground/background colors. The **IP•Math –> Select Features by 2nd** routine colors matched features in the foreground color and unmatched ones in the background color set in the Photoshop tools palette. If the foreground color is set to white, and the background to black, the matched features are erased and the unmatched ones kept so that they can be counted, as shown in the example. (Note - counting the features requires re-setting the foreground color to black, since the **IP•Measure Features->Count** routine counts features in the current pencolor. There are Photoshop shortcuts to set colors: press D for black and white, X to reverse them.)

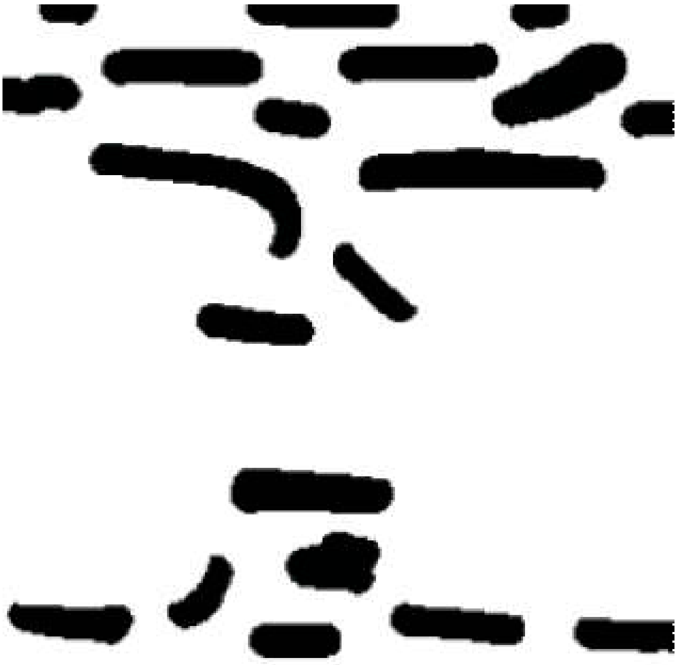
5.C.3. Region outlines as selection criteria

Outlines and boundaries are important markers to measure adjacency (common boundaries between features) or to select adjacent features. The outlines are those pixels within features that touch a white (background) pixel. They are generated by the **IP•Morphology–>Outlines** function and are equivalent to performing an Ex-OR between the original image and the result of an erosion.

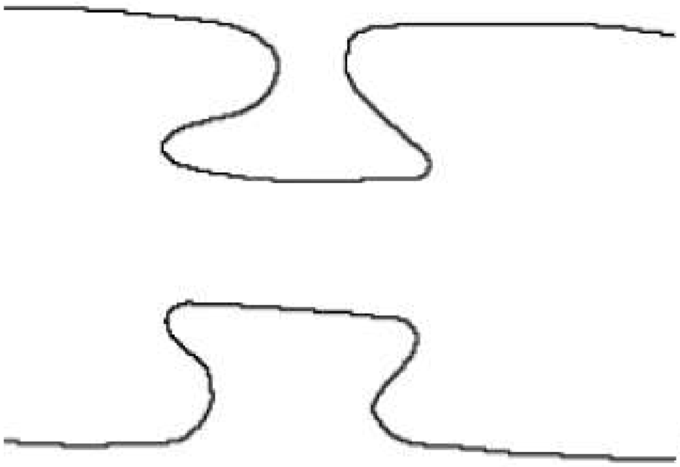
In the first example, the line of pixels immediately outside the pink substrate region can be produced either by dilation and Ex-OR or by inverting and outlining. Combining this test line or “probe” with the thresholded green features using a Boolean AND produces line segments. The ratio of the total length of the line segments to the total length of the original outline measures the fraction of the surface of the substrate that is in contact with the green features. In a multi-component structure, the adjacency of each component or phase to each of the others (many of which may be zero for regions that are not in contact) can be measured by ANDing the (dilated) boundaries around each phase and measuring the lengths to calculate the fraction for each pair.



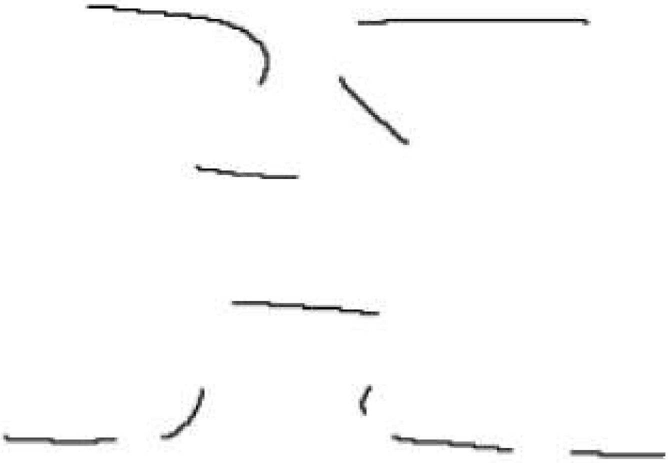
Original *Adjacent* image"



Thresholded green features

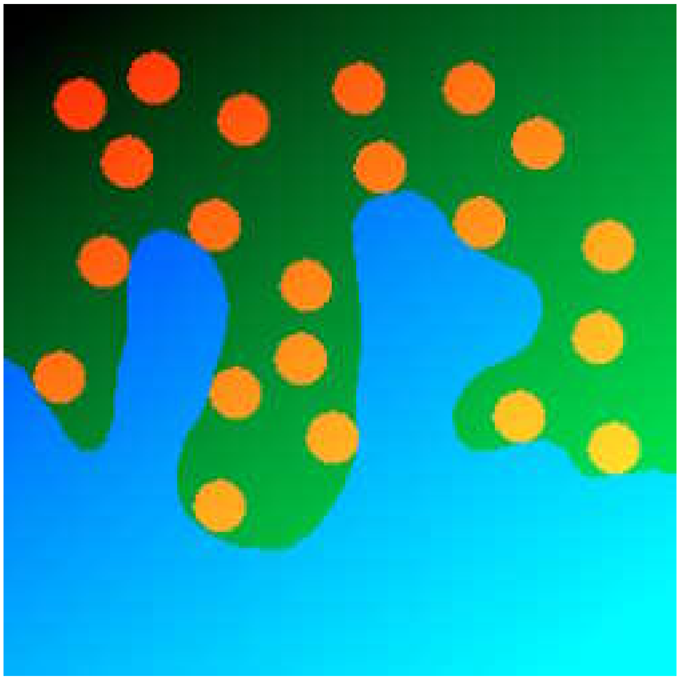


Line of pixels adjacent to pink substrate"

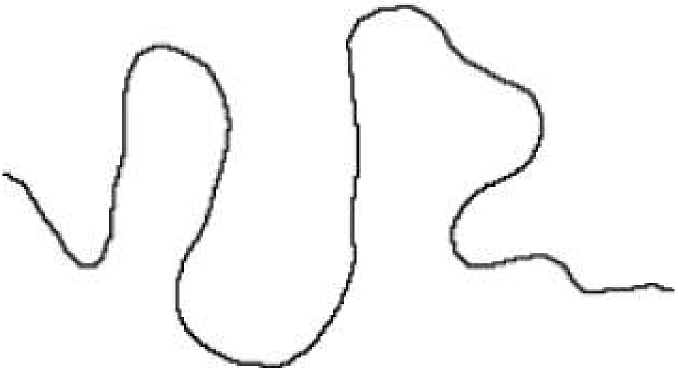


AND produces contact line segments

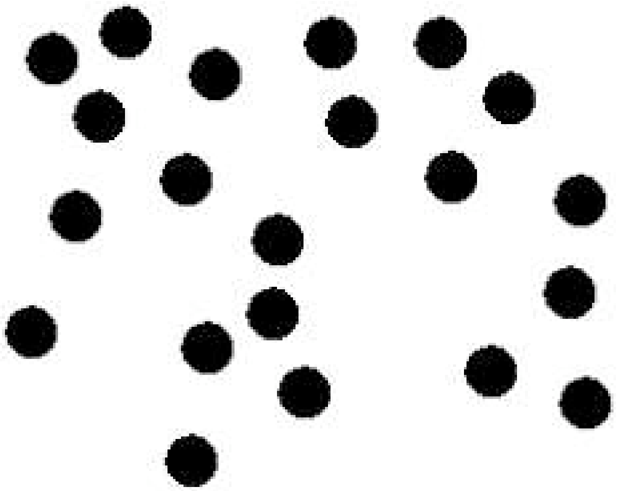
The next example initially looks like the same problem, but instead of a pixel-based AND, the marker selection routine is used to select the entire features that are in contact with the substrate so that they can be measured. The outline is placed in the 2nd image and used to select features that touch. Varying the amount of dilation applied to the outline of the substrate controls the definition of “touching.” The outlines of the final selected features were overlaid on the original image as a visual check on the result.



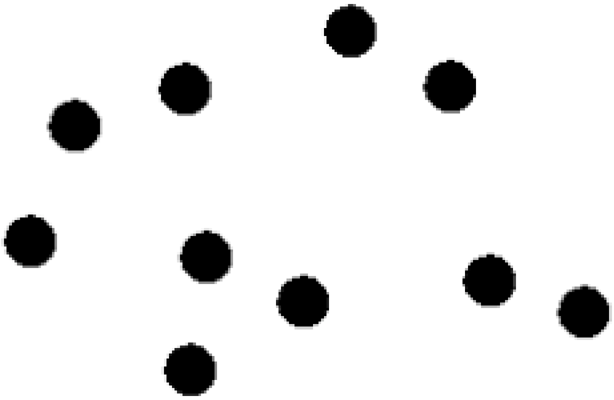
Original *Touching* image"



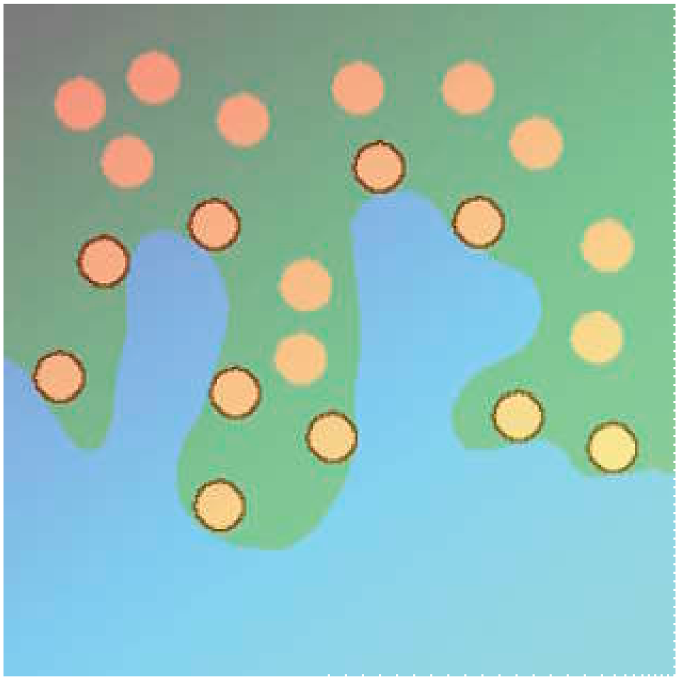
Line of pixels adjacent to substrate



Thresholded features"



Feature selection result



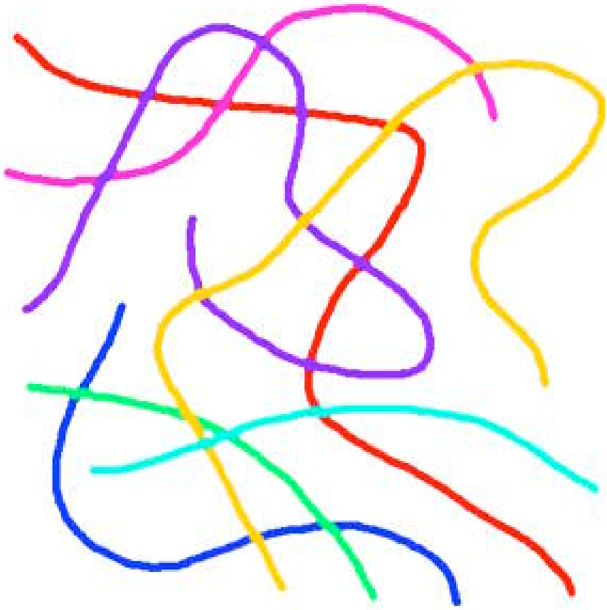
Visual check on the selected touching features

5.D. Feature skeletons provide important shape characterization

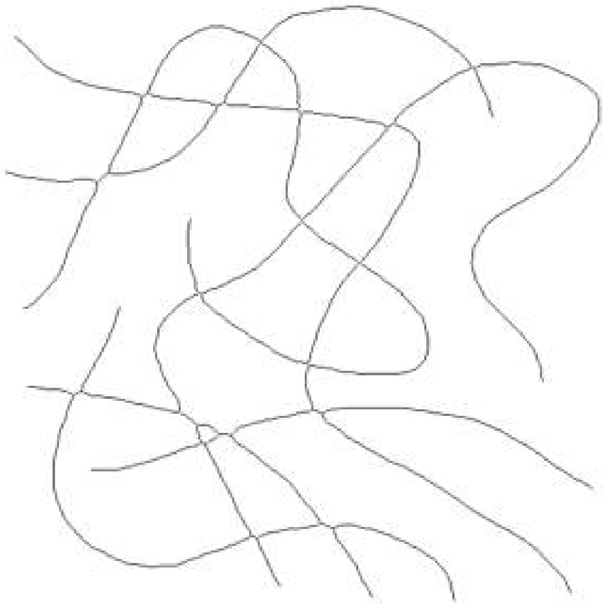
5.D.1. Grain boundary, cell wall, and fiber images

Broad thresholded lines can be thinned to single pixel width as was shown in an example in Section 4.A.2. This skeletonization is accomplished by erosion with rules that prevent removing the final line of pixels (**IP•Morphology–>Skeletonize**). Note that the skeleton is "8-connected" meaning that a pixel is assumed to touch any of its eight possible neighbors. But by this same criterion the cells or features that it divides would also be continuous, so it is often necessary to convert the skeleton to a "4-connected" line in which pixels touch only their four edge-sharing neighbors. These lines can separate the regions on either side.

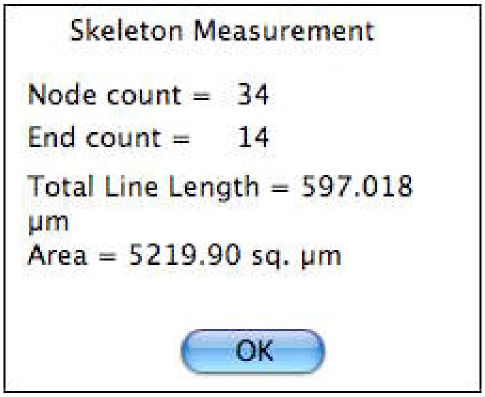
End pixels in skeletons have only a single neighbor, whereas most skeleton pixels have two neighbors and those at branch points or nodes have 3 or 4. For overlapped fiber images, the number of fibers can be determined as half the number of end points, and the mean length as the total length divided by number as shown in the example. This method is also correct for structures that extend beyond the image boundaries. A single end point is interpreted as one-half of a fiber in determining an average (the other end would be counted in another field of view).



Original *XFibers* image"



Skeleton



Skeleton Measurement

Node count = 34

End count = 14

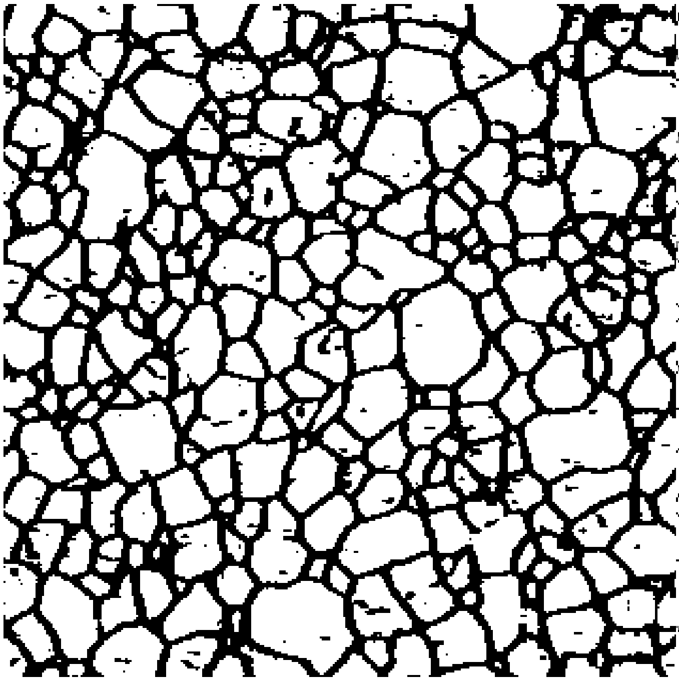
Total Line Length = 597.018 µm

Area = 5219.90 sq. µm

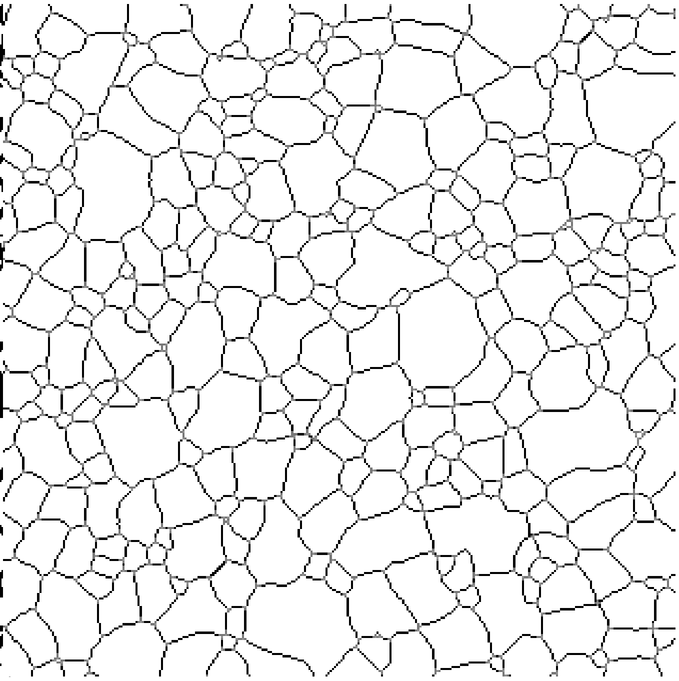
Counting the ends and measuring the length

Grain boundaries and cell walls form tessellations without end points, so pruning of branches with ends is an appropriate clean-up method. In the first example shown, the grain boundary tesselation is produced by **IP•Morphology–>Pruned Skeleton**.

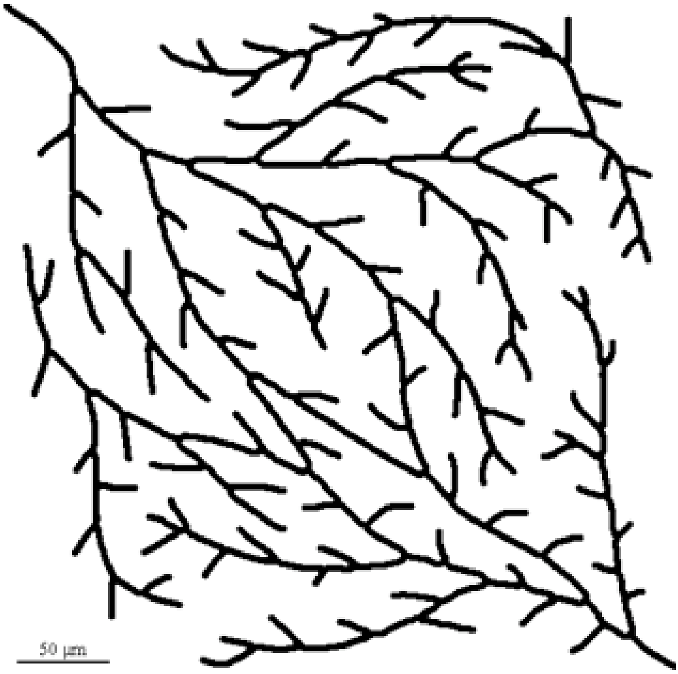
Removal or measurement of short branches based on length also provides a powerful tool. **IP•Morphology–>Skeletonize and Trim Branches** was applied in the second example. Removal of branch points leaves the separate segments for measurement. The **IP•Threshold–>Select Skeleton Components** allows selection of any of these components. In the third example this was applied after skeletonizing to leave just the individual segments for measurement.



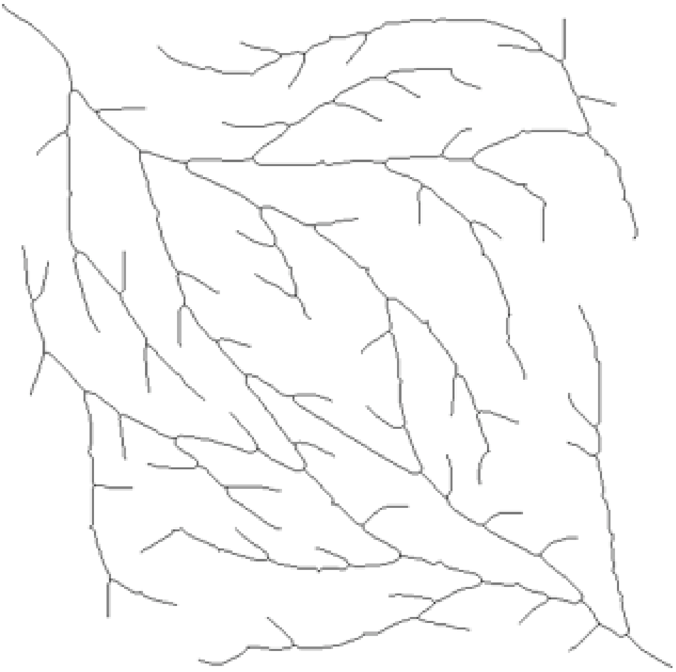
Thresholded *Gr_Steel* image"



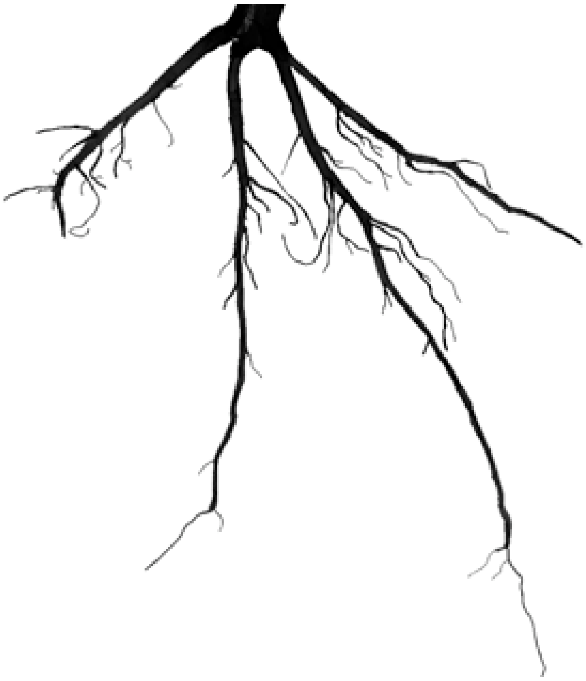
Pruned skeleton



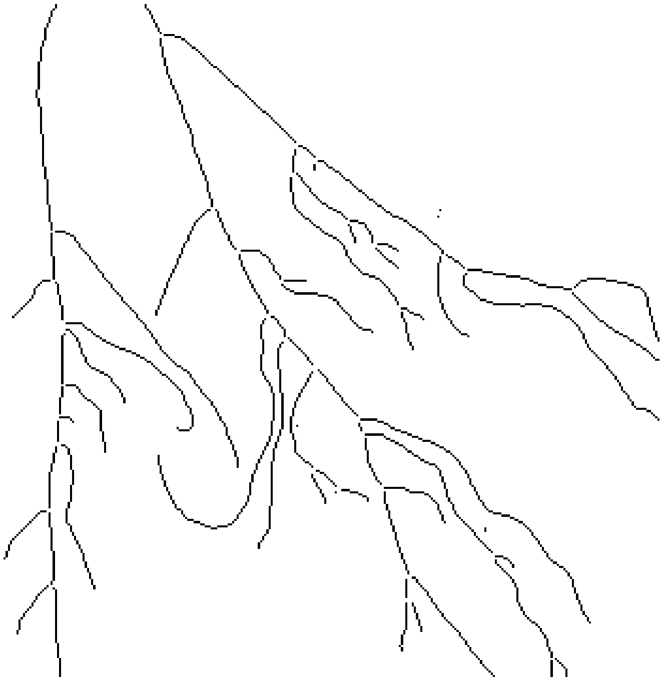
Original *Branches2* image"



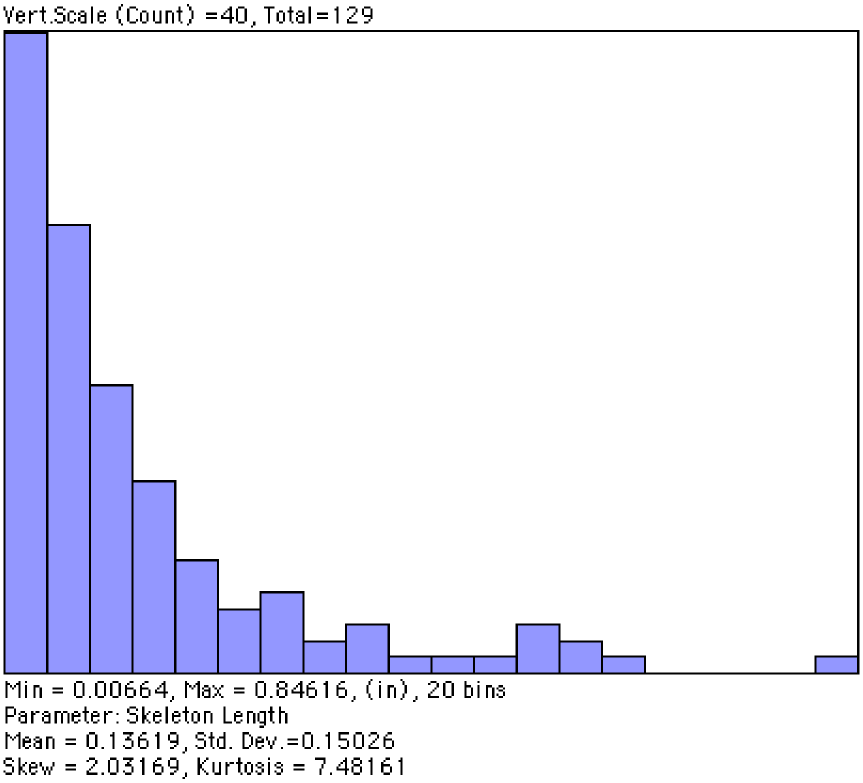
Skeleton with branches < 25 pixels removed



Original *Root2* image"



Skeleton with nodes removed (fragment)



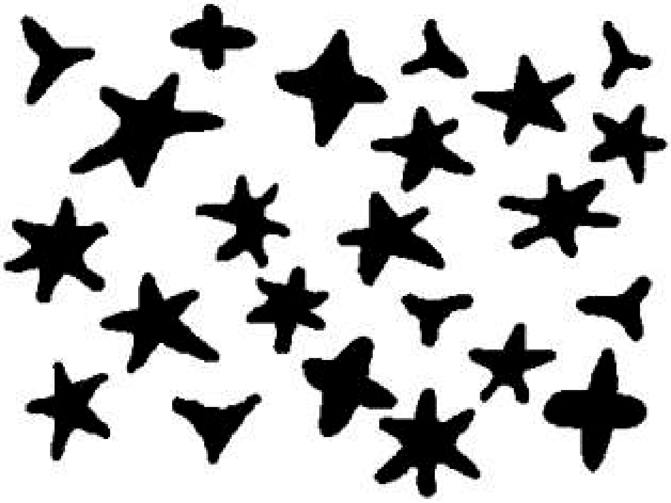
```
Vert.Scale (Count) =40, Total=129
```

```
Min = 0.00664, Max = 0.84616, (in), 20 bins
Parameter: Skeleton Length
Mean = 0.13619, Std. Dev.=0.15026
Skew = 2.03169, Kurtosis = 7.48161
```

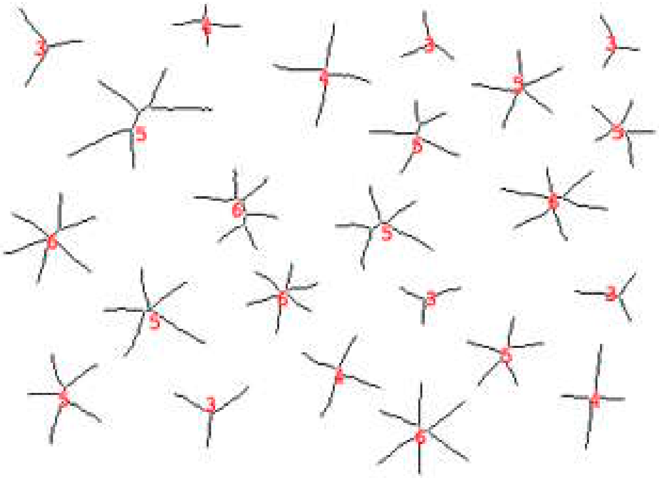
Measurement of the segment lengths

5.D.2. Measuring skeleton length, number of ends, number of branches for features

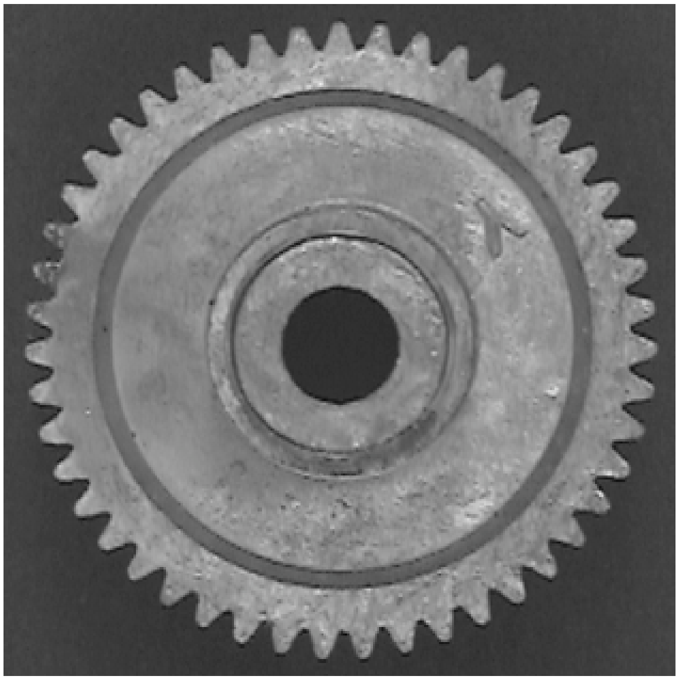
Values from skeletonized features provide basic topological (shape) information about structures, which will also be used in the discussion of feature measurement parameters. In the first example, the features are labeled with the number of skeleton end points, which measures the number of points in the original stars. In the second, the total number (43) can be counted to count the gear’s teeth. The dilated end points are superimposed on the original image using Photoshop Layers. Note: the thresholded gear image was cleaned up with an EDM-based morphological closing.



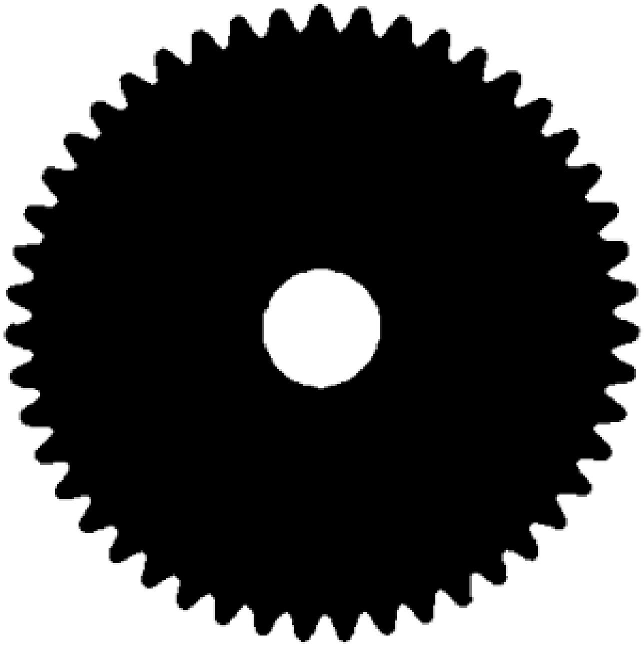
Original *Stars* image"



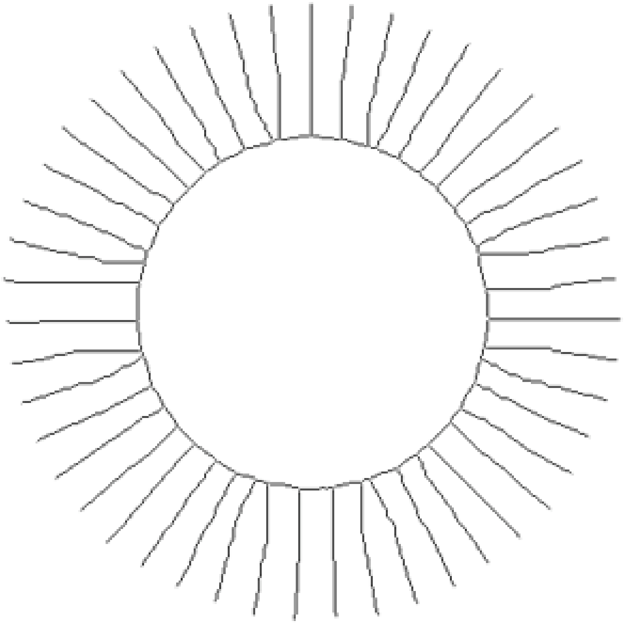
Skeleton and label (number of end points)



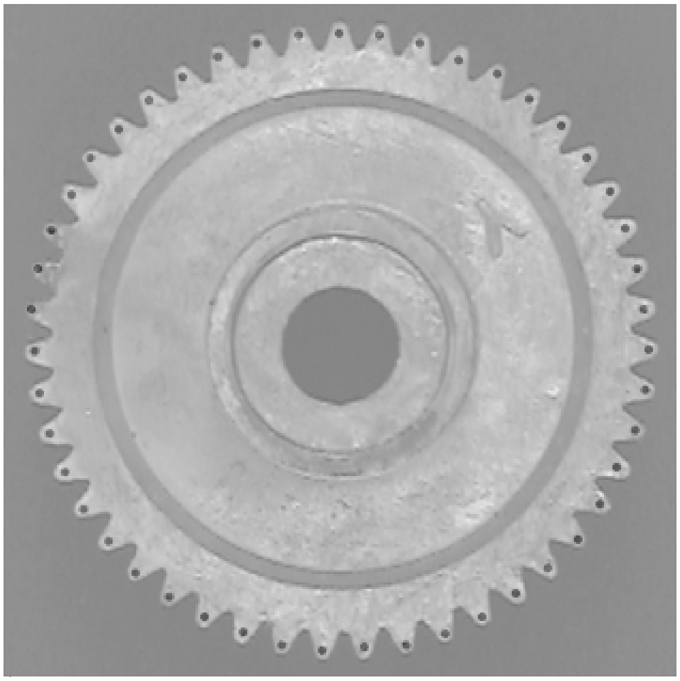
Original *Gear* image"



Binary image



Skeleton"

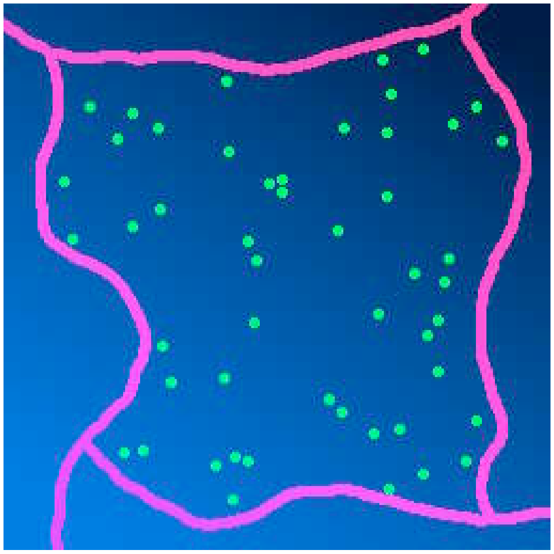


End points superimposed on original

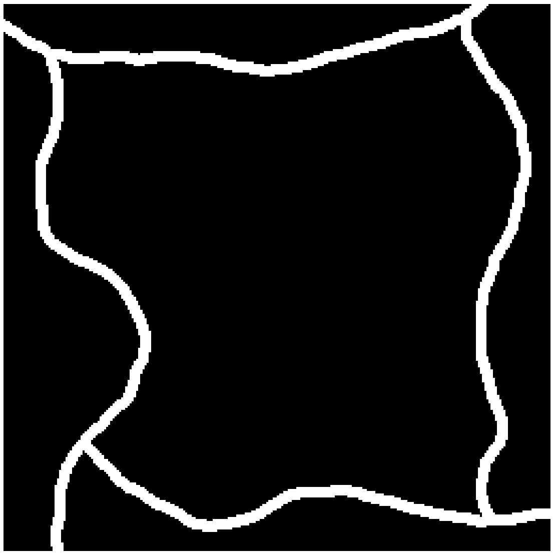
5.E. Using the Euclidean distance map for feature measurements

5.E.1. Distances from boundaries or objects

In Section 5.C.3 above, features adjacent to boundaries were selected. By assigning values from the EDM to features it is possible to select or measure features based on their distance from irregular boundaries. The assignment can be done by combining the images keeping whichever pixel is darker, or by using the binary image of the features as a mask placed on the EDM image. The EDM values can be calibrated as distance values using the density calibration routine, or just invert the EDM to measure the distance in pixels.

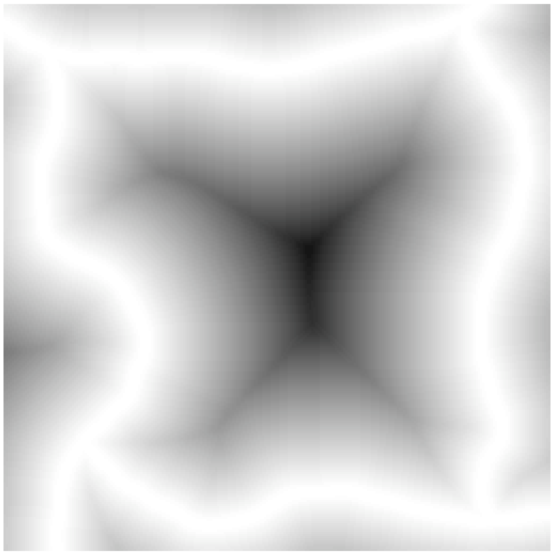


!



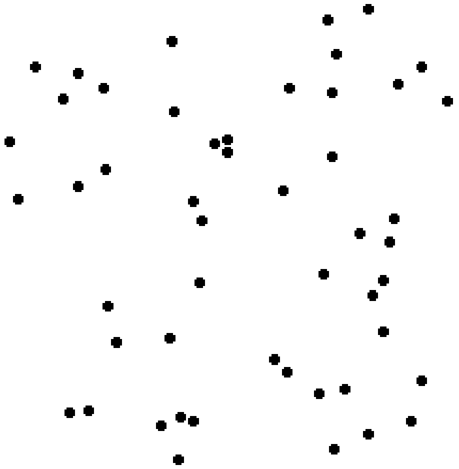
Original *Distanc2* image"

Thresholded cell interior

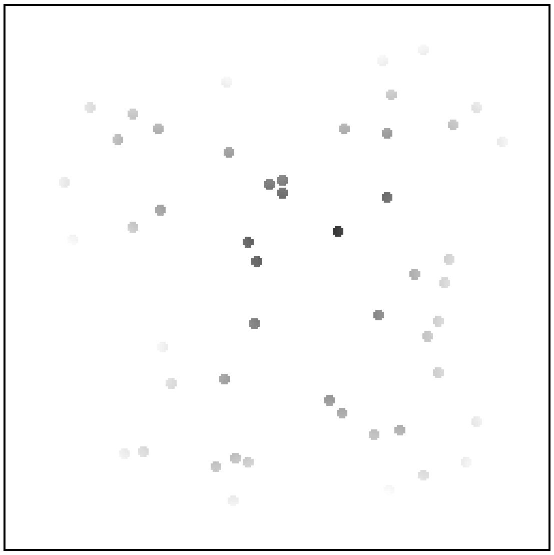


!

EDM of cell interior (inverted)"

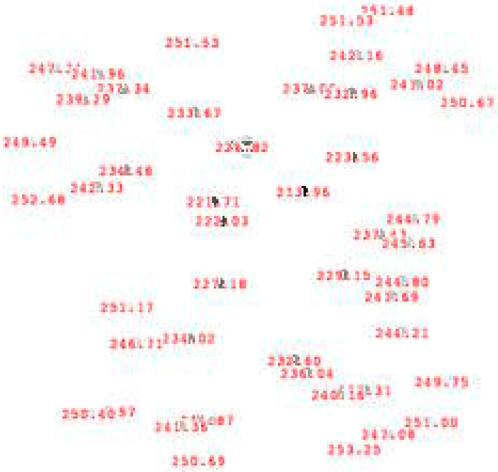


Thresholded features



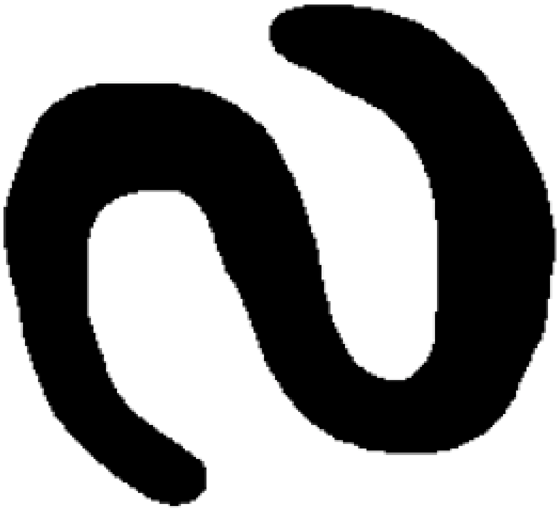
!

EDM values assigned to features"

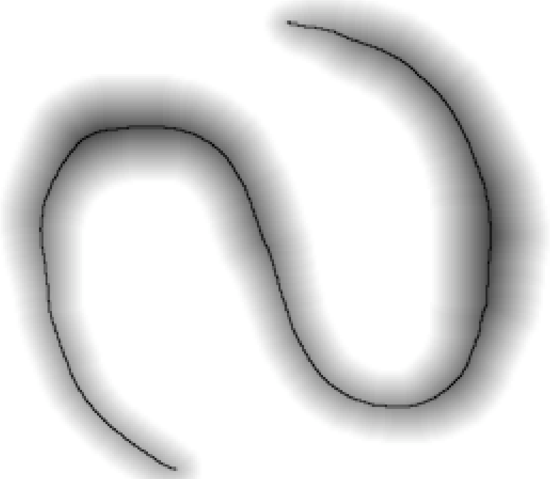


Measured distances from boundary

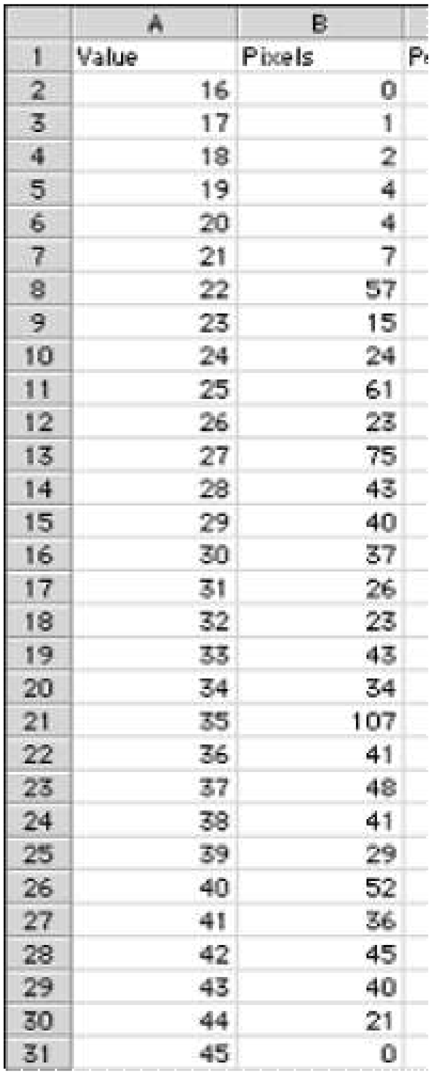
Sampling the EDM with the skeleton provides width measurement (mean, max, min, standard deviation) for irregular shapes. This method is used automatically by the feature measurement routines but can be applied manually when more complete statistics are required. Notice that the skeleton follows the central “ridge” in the EDM where the pixel values represent the radius of an inscribed circle. A histogram of the values along that ridge provides a comprehensive measurement of the feature’s width. The data are saved to disk by the **IP•Measure Global–>Histogram** routine and can be analyzed using Excel.



Original *Width2* image"



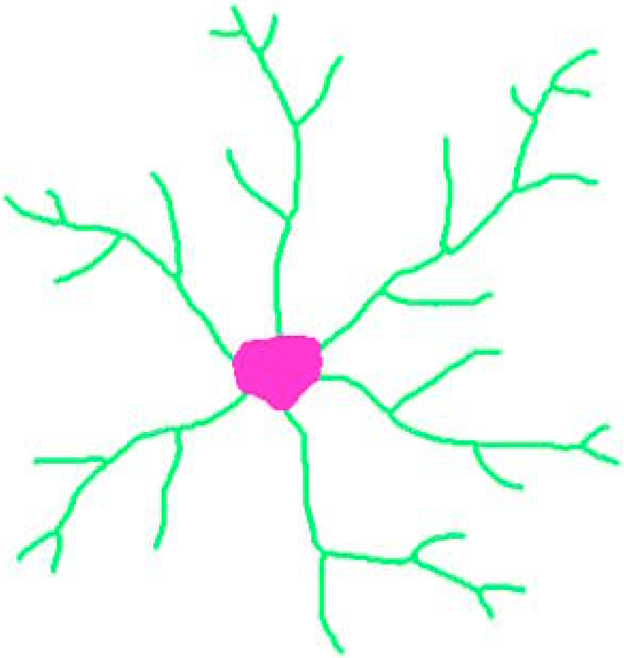
Skeleton superimposed on the EDM



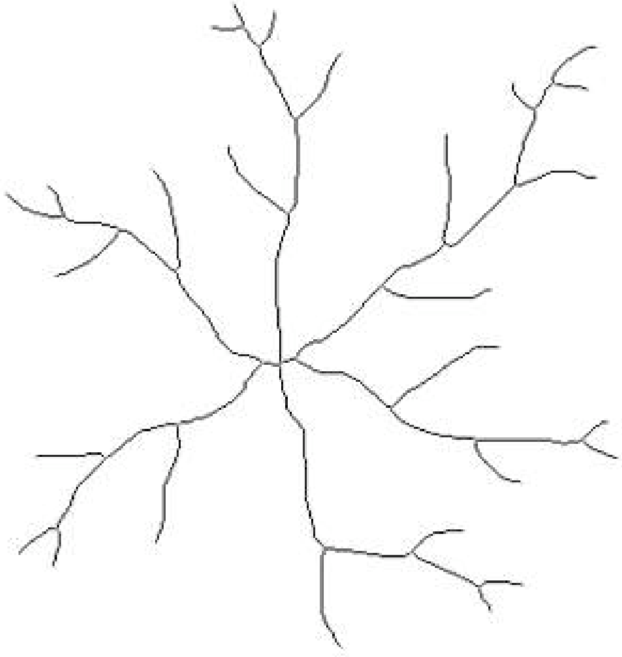
	A	B	
1	Value	Pixels	Pe
2	16	0	
3	17	1	
4	18	2	
5	19	4	
6	20	4	
7	21	7	
8	22	57	
9	23	15	
10	24	24	
11	25	61	
12	26	23	
13	27	75	
14	28	43	
15	29	40	
16	30	37	
17	31	26	
18	32	23	
19	33	43	
20	34	34	
21	35	107	
22	36	41	
23	37	48	
24	38	41	
25	39	29	
26	40	52	
27	41	36	
28	42	45	
29	43	40	
30	44	21	
31	45	0	

Histogram data in a spreadsheet

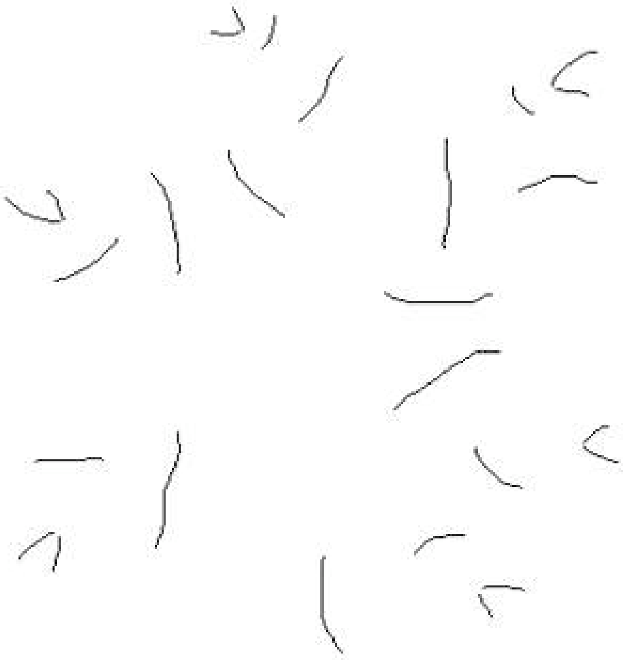
The example below illustrates the use of the EDM and skeleton together to show a complex relationship between the length of each branch and how distal it is from the cell body. The nodes are removed from the skeleton to separate the segments. Then the ends are used as markers in a feature selection operation to keep just those segments that are terminal branches. Next the EDM of the region around the cell body is generated and the values assigned to the terminal branches. Finally a plot of the length of each branch vs. the minimum brightness value (the distance of the point nearest the cell body) is constructed to show the trend. Understanding this sequence (each individual step is not shown in the illustrations) is a good indicator of mastery of these tools.



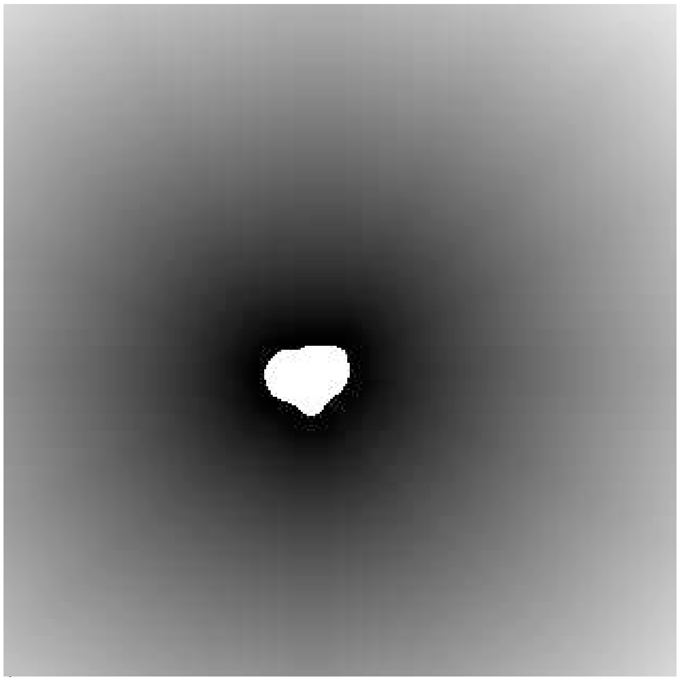
Original *Neurons* image"



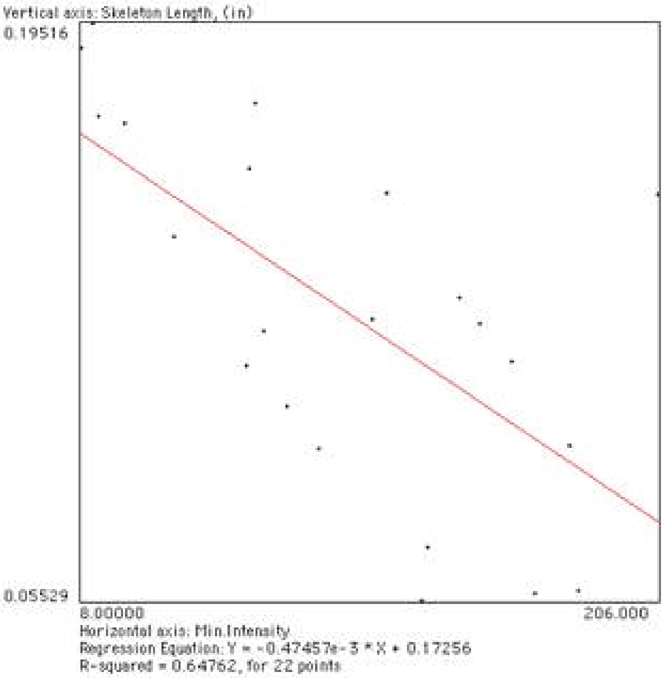
Skeleton



Terminal branches"



EDM of the region outside the cell body


Vertical axis: Skeleton Length, (in)
0.19516
0.05529
8.00000
206.000
Horizontal axis: Min.Intensity
Regression Equation: Y = -0.47457e-3 * X + 0.17256
R-squared = 0.64762, for 22 points

EDM values assigned to branches"

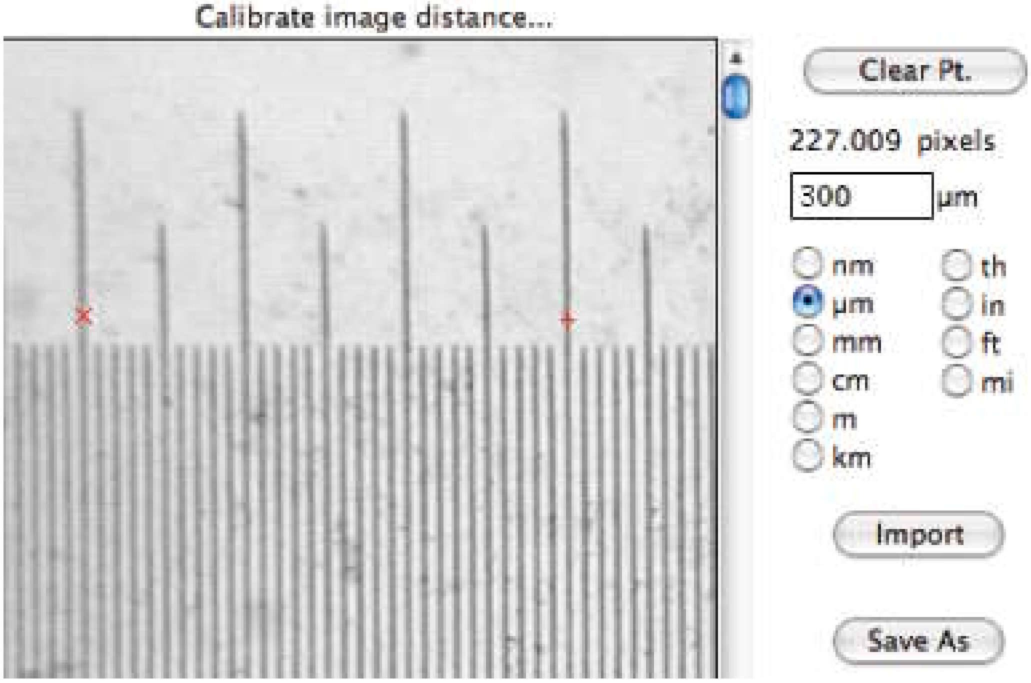
Plot showing shorter lengths for more distal terminal branches

6. Measurements

6.A. Calibration

6.A.1. Calibrating image dimensions

Establishing a calibration that can be used by the various measurement routines is typically done by acquiring an image of a stage micrometer or some standard object. A micron marker on an scanned image can also be used. The procedure in the **IP•Measure Global–>Calibrate Magnification** routine is to mark two points on the image and enter the distance between them (and select the units). Calibrations can be saved to disk by name (e.g., one for each objective lens on a light microscope) and recalled as needed.


Calibrate image distance...

227.009 pixels

300 µm

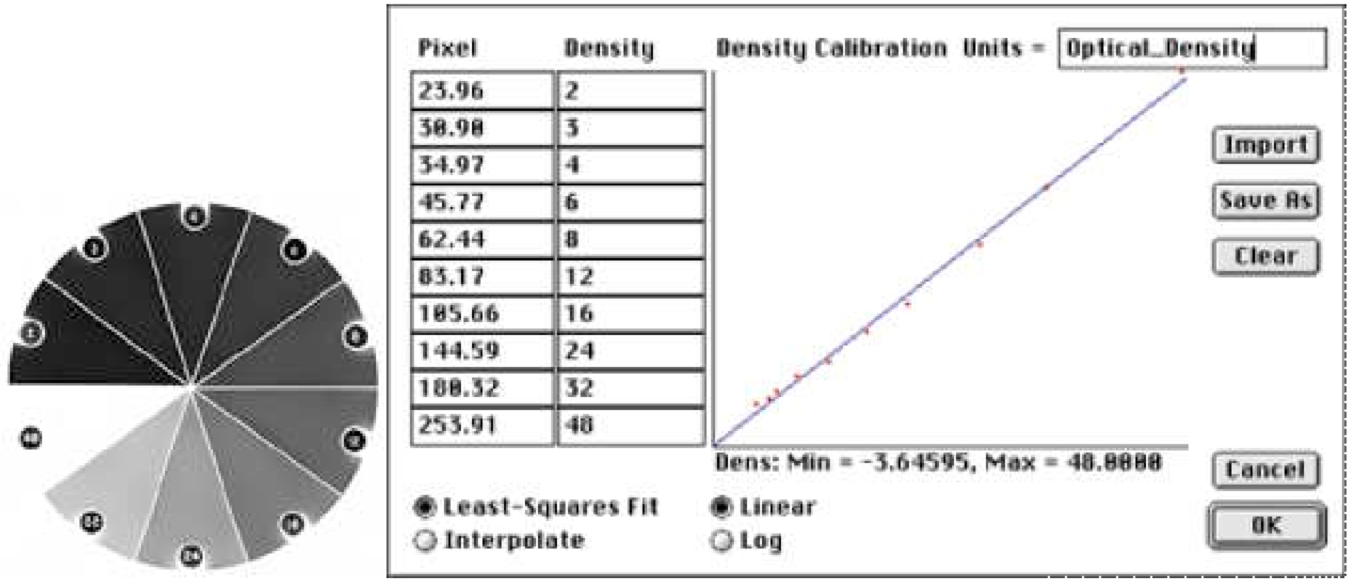
○ nm ○ th
● µm ○ in
○ mm ○ ft
○ cm ○ mi
○ m
○ km

The **IP•Measure Global–>Calibrate Magnification** dialog

Note that the magnification calibration is not tied to a specific image, but rather is a system constant that stays in effect for all measurements until it is changed.

6.A.2. Calibrating density/greyscale values

Constructing a curve relating pixel value (0..255) to density, or any other parameter, requires standards. These are readily available at macro scales but harder to make or obtain for microscopes. Recalibration is needed often because of variation in line voltages, lamp aging, optical variations, camera instabilities, etc. It is also important to turn off any automatic gain adjustments in the camera or scanner. The procedure is to measure the average brightness of uniform known regions and then enter the values into the **IP•Measure Global–>Calibrate Density** routine. The resulting curve can be stored on disk and recalled as appropriate. It is also possible to construct calibration curves of pixel value vs. distance to use with EDM-based measurements.



Density Calibration Units = Optical_Density

Pixel	Density
23.96	2
30.90	3
34.97	4
45.77	6
62.44	8
83.17	12
105.66	16
144.59	24
180.32	32
253.91	48

Dens: Min = -3.64595, Max = 48.0000

● Least-Squares Fit
○ Interpolate

● Linear
○ Log

The *DenWedge* image and its calibration curve

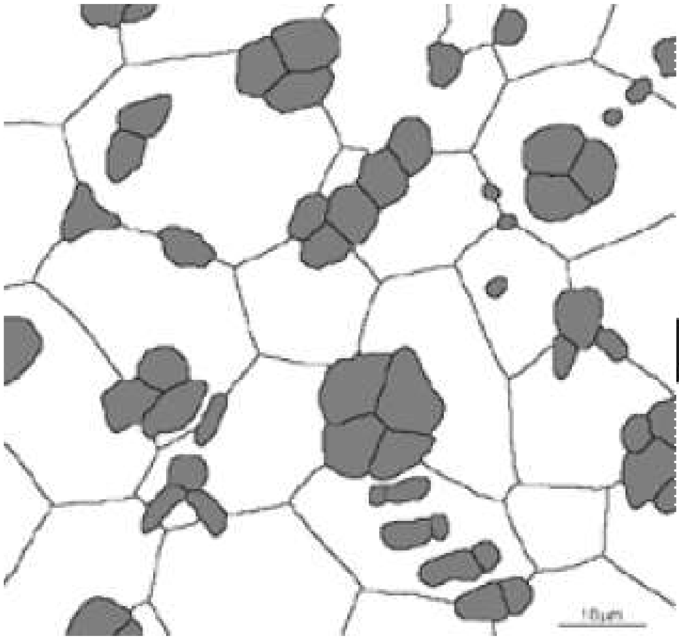
6.B. Measurement of principal stereological parameters

Relationships between three-dimensional structure and two-dimensional images provide methods to correctly measure metric properties such as volume fraction, surface area, length, etc., as well as topological properties such as number and connectivity. Modern stereology emphasizes the best ways to perform sampling and sectioning to avoid bias.

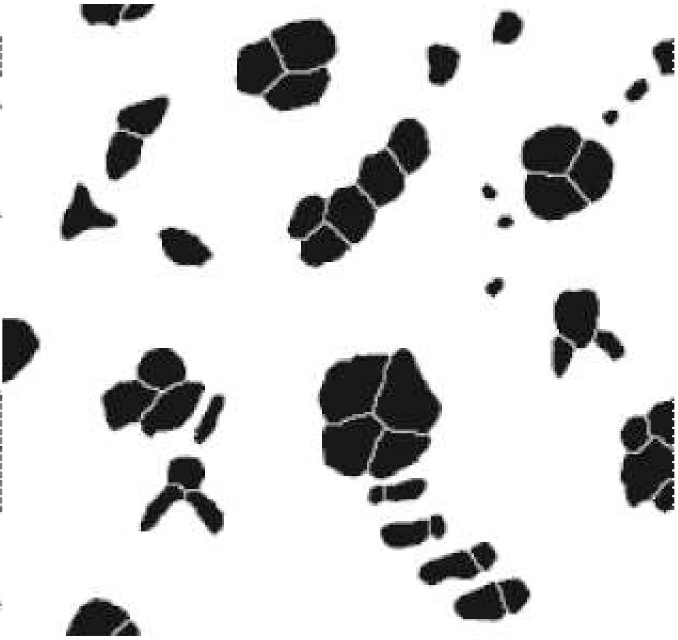
6.B.1. Volume Fraction

The volume fraction of a region, phase or structure in a 3D volume is measured by the area fraction shown on a random plane section (or better, averaged over many such sections). The **Measure Global –> Area Fraction** plug-in reports (and writes to the log file if logging is turned on) the percent of the image that is not white background. If an area of the image has been selected with the marquee, lasso or wand selection tools, the reported value is the fraction of the selection area that is not white. In most cases the image being measured will be a binary (black and white) image as a consequence of thresholding and perhaps the application of Boolean logic, and the measured area is defined by the black pixels, but the plug-in is equally applicable to images in which features of any grey scale or color value are present, on a white background. The area fraction value measured with this plug-in is identical to the result that could be obtained by summing the contents of the histogram for all values other than white (255).

As an example, in the image *Drawing* (shown), thresholding the grey cells produces a binary image in which the area fraction is reported as 16.93%



Original *Drawing* image"

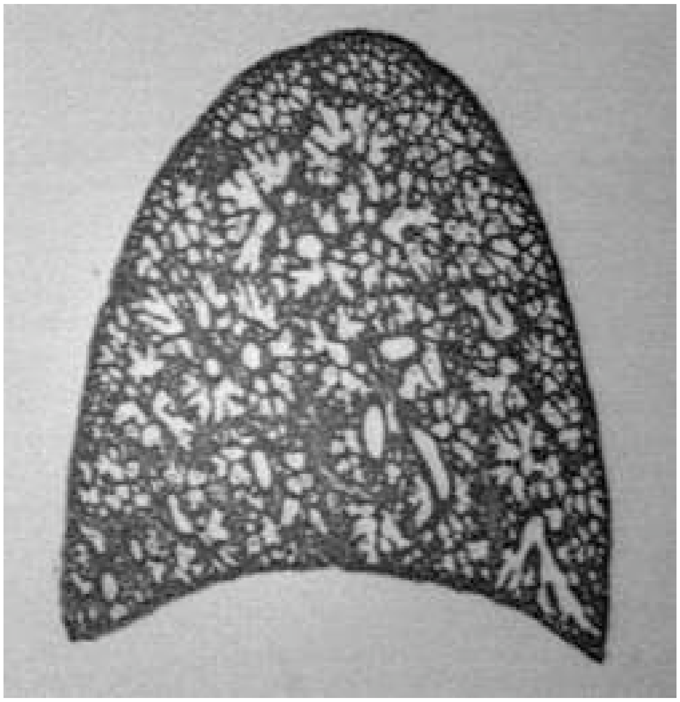


Thresholded grey cells

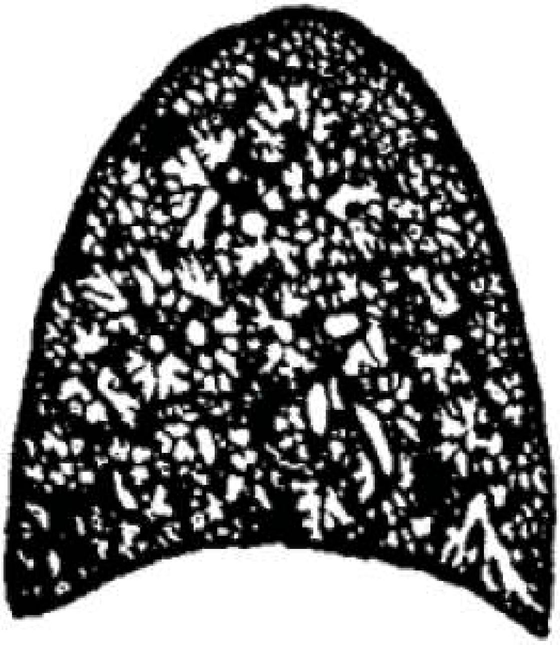
Another way to measure volume fraction (or area fraction) is to use a grid of widely spaced points (ideally separated far enough that rarely do two points in the grid fall on top of the same feature in the image) and count the fraction of the points that lie on the structure of interest. The advantage of the technique is that it allows a direct estimate of the precision of the volume fraction measurement, because the statistics of counting are such that the standard deviation in the number counted is just the square root of the number. In other words, if a grid with 100 points (10 x 10 array) is used and one third of them (33) fall on features that constitute the structure of interest, then the result is 33 ± 6 (approximately the square root of 33). If it is desired to obtain an overall measurement precision of 5%, then additional fields of view should be similarly measured, and it can be expected that about 11 more fields will be needed, because a total of 12 fields times about 33 hits per field would be 400 points counted, and the square root of 400 is 20, or 5%. Designing an experiment to collect the required number of fields of view so that they adequately represent a real specimen is an important consideration, and usually requires sampling a few images to get a rough idea of the magnitude of the values of interest.

To apply a grid to an image, an efficient procedure is to duplicate the image (**Image–>Duplicate**) and then erase it (**Select–>All** and **Edit–>Clear**, assuming the background color has been set to white). Layers can also be used. A Photoshop action implementing this sequence and assigning it to a function key is easily set up. If many images of the same size are to be processed, the image of the grid can simply be saved to disk for re-use. Generating a grid of points or lines is selected under the Points and Lines menu. In the examples shown below, the points have been dilated for better visibility.

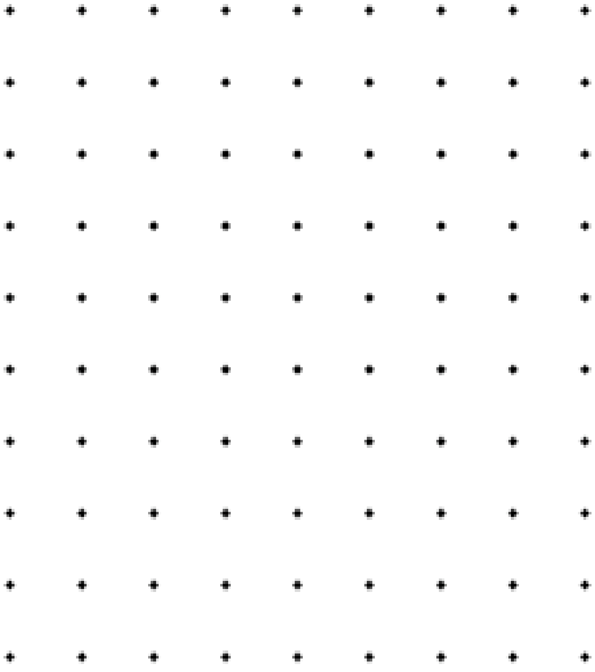
In this example, the volume fraction of the lung that is air space is wanted. But the lung tissue does not cover the entire field of view, so two measurements are needed, to measure the total lung area and the tissue area. The original image was leveled and thresholded to produce a binary representation of the tissue, and then a grid with 90 points was generated. Applying a Boolean AND to combine the grid with the binary image of the tissue and counting the number of points reports 29 hits.



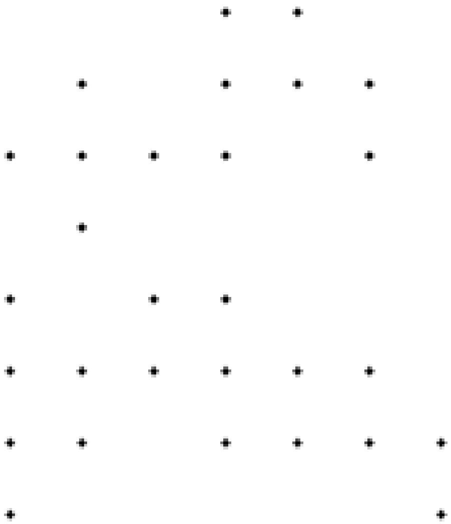
Original *RatLung* image"



Binary representation

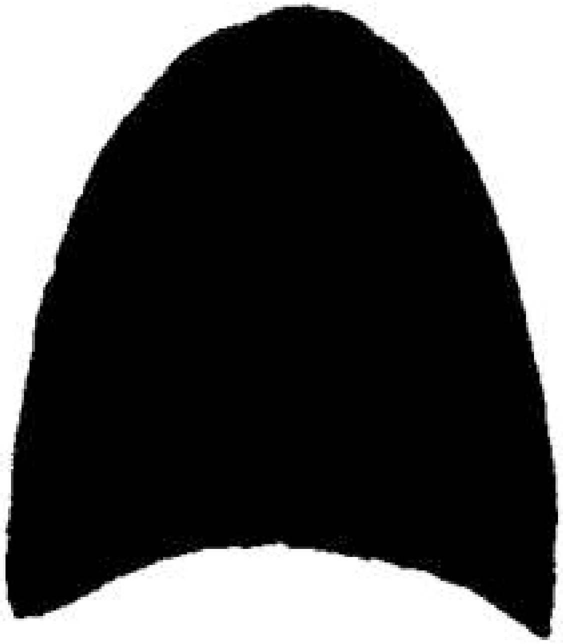


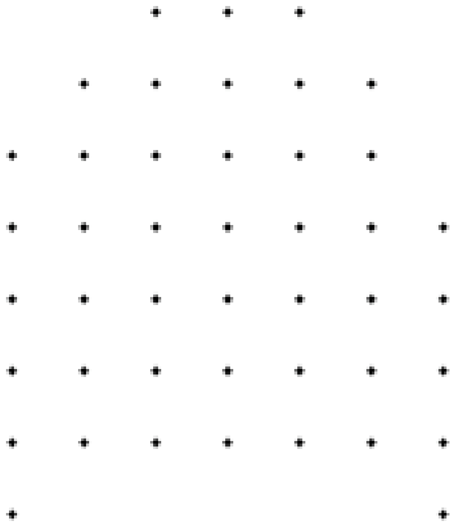
Generated 90 point grid!



Grid ANDed with binary

Next, the binary image of the tissue has its holes filled in (**Morphology–>Fill Holes**) to produce an image of the total lung cross section. This is also combined with the grid using a Boolean AND and the hits counted (44). The area fraction of the cross section that was originally air space is thus (44 – 29) / 44, or 34 percent. To properly estimate the volume fraction of the lung, multiple sections should be taken at different positions in the organ, and the counts for net tissue area and full cross section area summed individually before being used in the calculation of the percentage.





Binary with holes filled"

Grid ANDed with filled image

A manual procedure using grids with visual recognition of the structure is to superimpose the grid directly onto the image (usually with a contrasting or colored pen color), select a pencil of convenient size and unique color, and manually mark the points that are to be counted. The **IP•Measure Features–>Count** plug-in reports the number of separate marks in the current pen color, and the resulting image can be saved as a record of the procedure if desired.

6.B.2. Surface Area

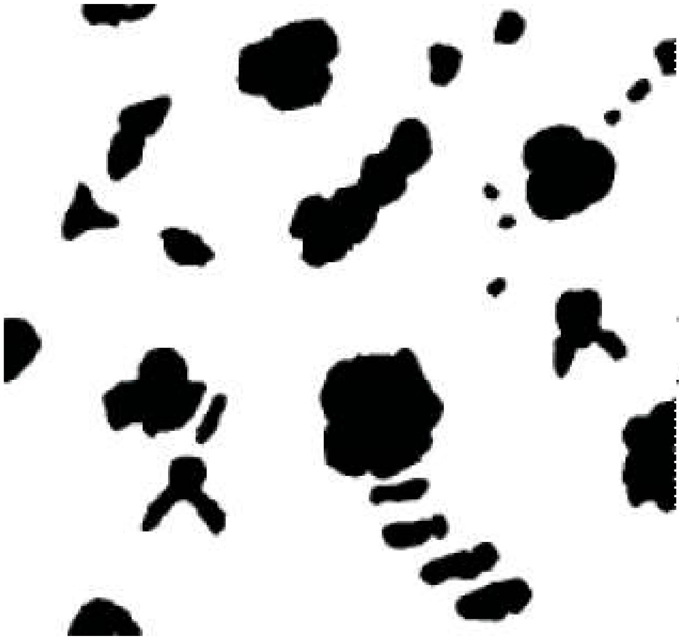
The surface area per unit volume of the boundaries between regions in the 3D volume (which may either be identical phase regions, such as the grains in a metal separated by grain boundaries, or dissimilar regions, such as the surface of membranes that surround and separate organelles from the rest of the cytoplasm of a cell) can be measured stereologically. These 3D surfaces are revealed on two-dimensional section planes as lines. If the lines are broad, as for example would be produced by etching grain boundaries for visibility, or sectioning through membranes at a shallow angle, the resulting image can be skeletonized to reduce the lines to ideal single-pixel width. Thresholding of original grey scale or color images, and perhaps other processing, is typically used to obtain an image in which the boundaries whose surface area per unit volume is to be measured are delineated by a line of black pixels on a white background.

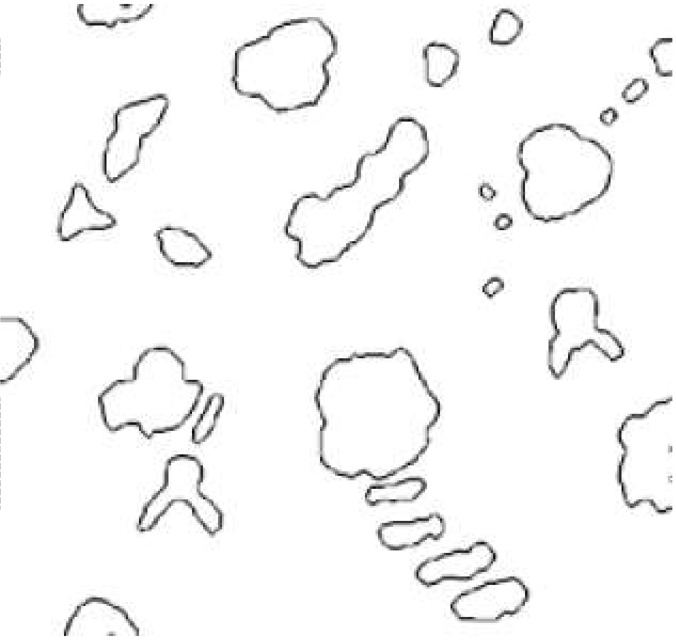
There are two principal relationships used to measure the surface area per unit volume. One is based on the length of the lines of intersection between the surfaces of interest and the sampling plane that is imaged. The other draws line grids on the image and counts the intersections between the grid lines and the lines that represent the trace of the surface on the plane of examination. The latter method may be preferred because it is a counting rather than a measurement operation. Also, it is possible to generate specific kinds of grids (e.g., circles, cycloids, random lines) that can be used with specific kinds of section orientations to obtain isotropic sampling of structures that have a preferred orientation. It must be assumed for purposes of making this measurement that either the surfaces of interest are isotropically oriented in 3D space, or that the section planes that have been imaged are oriented in such a way that isotropic sampling is obtained. One such technique is to cut so-called “vertical” sections and then draw cycloids on the surface.

Given a suitably obtained and processed image (or set of images) the surface area per unit volume can be measured by Sv = (4/#) • Length of line / Area of image. The **IP•Measure Global –> Total Line Length** plug-in reports the total length of lines present in the image, as well as the total area of the image or selection. This method can be used for either a full image area or for any rectangular region defined by the marquee selection tool. However, if applied to a non-rectangular selection region it actually measures the area and contents of the bounding rectangle that fits around the arbitrary-shaped selection. Inverting the selection and erasing exterior lines can be used to deal with any problems this might cause.

For the method of counting intersections with a line grid, the relationship is Sv = 2 • Number / Grid Length where Grid Length is the total length of the lines of the grid, which is reported and logged by the various plug-ins that generate them (e.g., **IP•Lines and Points–>Cycloid**, or **Line Grids**, or **Parallel Lines**, etc.). The number of intersection points is often obtained by applying the **IP•Measure Features –> Count** plug-in after the Boolean AND operation used to combine the image of the boundary line with an image containing the grid lines. It can also be used to count the number of points marked with the pencil tool on an image. Typically a black pencil mark several pixels in diameter can be used to mark color images, while grey scale images can be converted to RGB mode and marked with a colored pencil. In either case, the grid line serves as a reference and the crossing points can be marked and then counted with the **IP•Measure Features –> Count** plug-in, which counts separate marks in the current pencolor.

To illustrate this procedure, the test image (*Drawing*) shown above represents a structure with two type of grains or cells, grey and white. By application of morphological and Boolean logic any of the three types of surfaces can be isolated (the grey-grey cell boundaries, the grey-white cell boundaries, and the white-white cell boundaries). As an example, by thresholding the grey cells, performing a closing, selecting the outlines, and skeletonizing, the boundaries between the grey and white cells are delineated as shown.

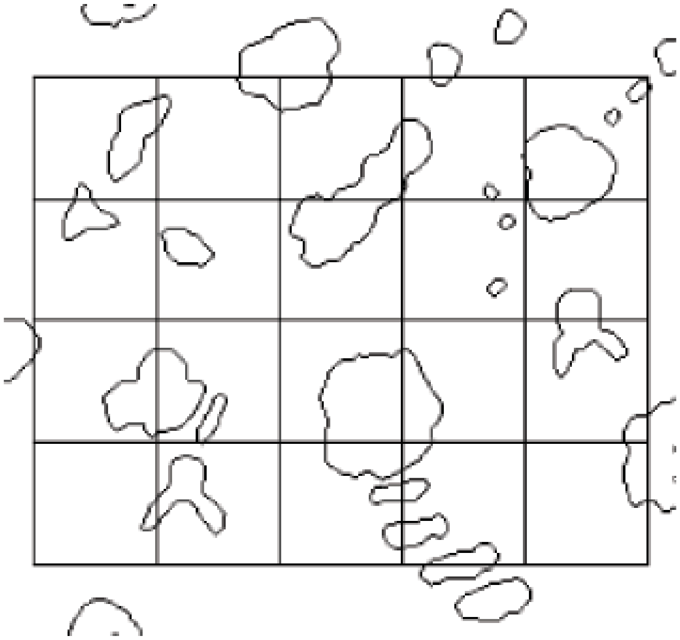




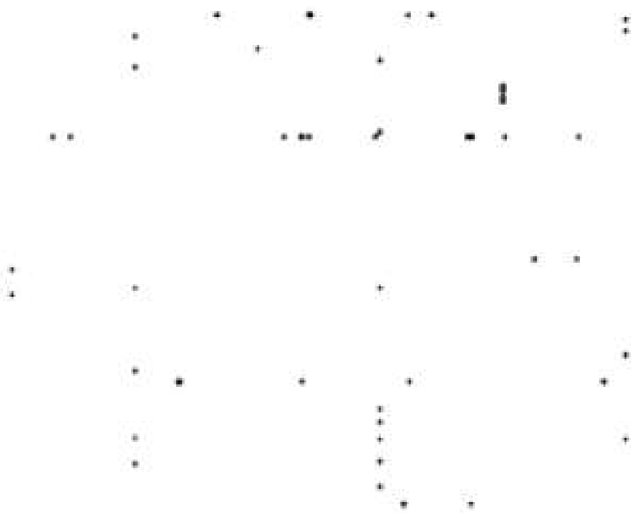
Closing to eliminate grey-grey boundaries"

Skeletonized outlines

Measuring the total length of the skeleton lines reports values of 547.08 µm for the line length and 5453.6 µm^2 for the image area, which by the equation above produces a value of 0.1277 µm^{-1} for the surface area per unit volume. Note that the measurement of volume fraction above did not depend on the calibration of the image magnification, since the units cancel out, but the measurement of surface area does require that the calibration be performed beforehand.



Grid lines superimposed on the outlines"

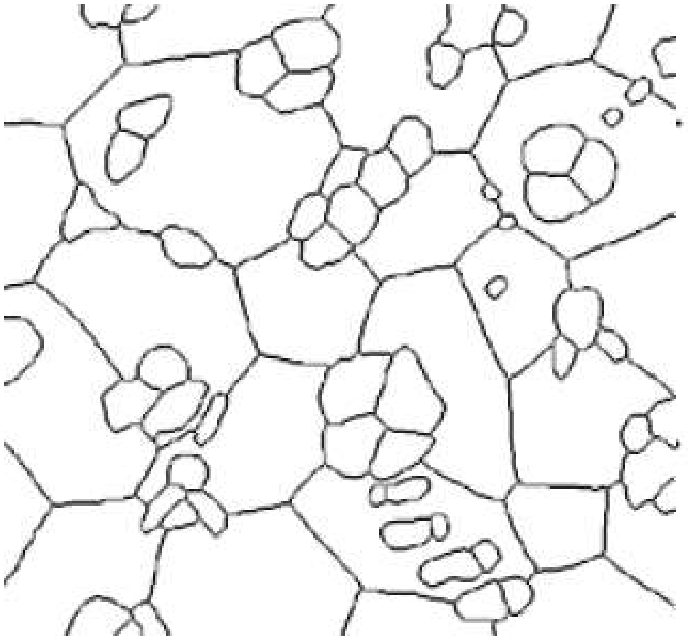


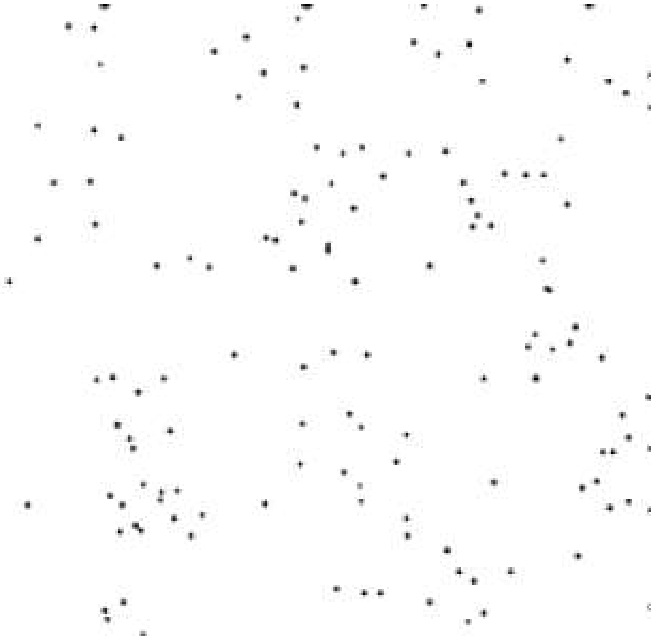
Intersection points produced by ANDing

For comparison, the second measurement procedure using a grid of lines is also shown. A rectangular line grid was used, but the same method would apply to circles, cycloids, or random lines. ANDing the grid with the outlines and counting the number of intersections (which are shown dilated in the example for visibility) produced 45(±6.7) counts with a grid line length of 680.49 µm. Using the equation above, the resulting value for surface area per unit volume is 0.1323(±0.0197) µm^{-1}, in good agreement with the line length method above.

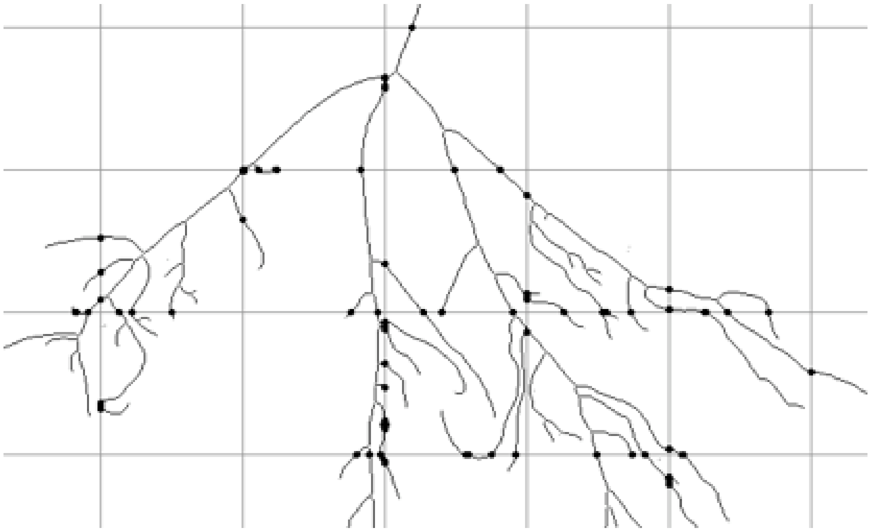
6.B.3. Line length

The length of lines in a 3D volume is calculated from the number of intersections that they make with the section plane. In the structure shown in the example image, the triple lines where three cells or grains meet are one such linear structure. These can be isolated by thresholding all of the cell boundaries in the image, skeletonizing, and selecting just the nodes (points in the skeleton with more than two neighboring pixels - these are shown dilated for visibility in the illustration). Counting these reports 145 intersections of the triple lines with the image, whose area are noted above is 5453.6 µm^2. The line length per unit volume is 2 • Number of Intersections / Area, and thus is calculated as 0.0532 µm^{-2}. Obviously, this measurement also requires the prior calibration of the image. Boolean logic combined with morphological dilation of the phase regions can be used to select triple lines that lie at the junctions of the various phases present.





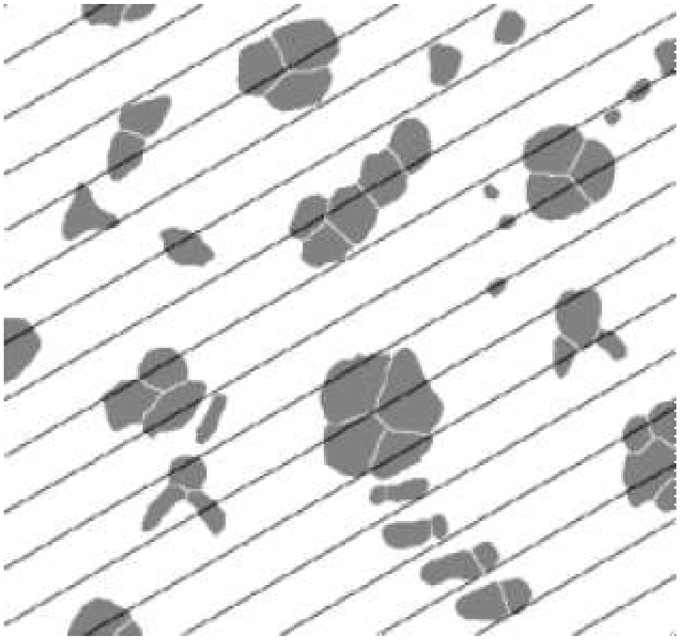
Skeletonized boundaries" Nodes (dilated for visibility)



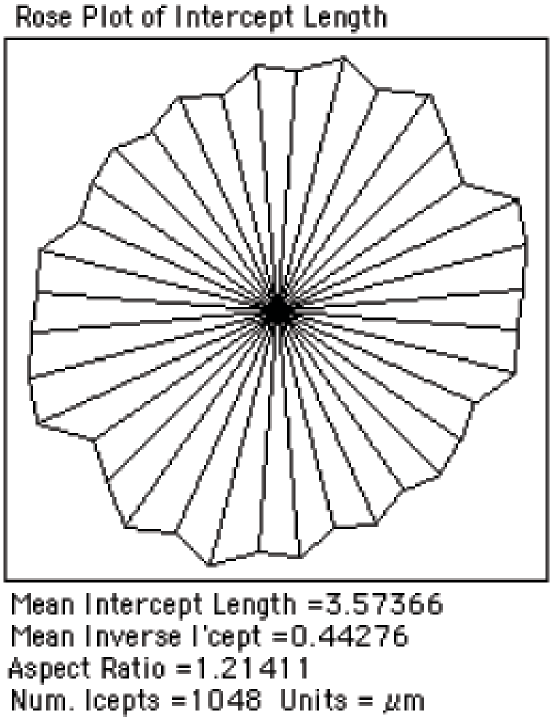
Another procedure often used to measure line lengths in volumes applies to transmission images through thin sections. The linear structures appear in such images as lines, which can be thresholded and skeletonized. Drawing a grid of lines on the image represents a set of surfaces extending down through the thickness of the section (whose thickness must be independently known). As usual, the grid lines must be made isotropic in 3D by appropriate selection of section orientation and choice of grid lines, if the sample is not itself isotropic in structure; this is sometimes done by using vertical sectioning and cycloids, or random sectioning and circles. In either case, ANDing the grid lines with the skeleton of the linear structure and counting the intersections produces a number of counts. This is used in the same calculation as above, namely length per unit volume equals 2 • Number of Intersections / Area, but now the area is the area of the surfaces that extend from the grid lines through the imaged section. This is just the thickness of the section times the length of the grid lines.

6.B.4. Intercept lengths

The mean intercept length through a structure can be measured by drawing lines and ANDing them with the binary image of the structure, followed by measuring the length. This is most efficiently done by the **IP•Measure Global–>Intercepts** plug-in, which automatically generates a grid of parallel lines with orientations that vary systematically in ten degree steps, and reports the mean value (3.574 µm in the example shown) as well as showing the rose plot to reveal any anisotropy in the structure.



!

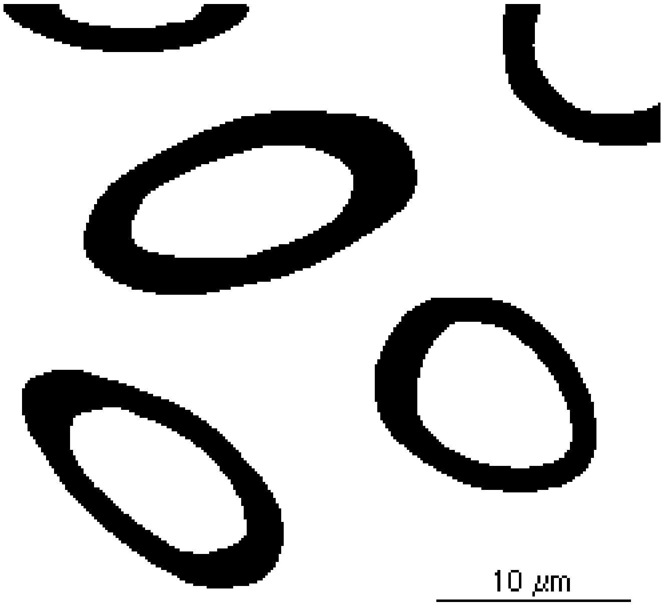


```
Mean Intercept Length =3.57366
Mean Inverse I'cept =0.44276
Aspect Ratio =1.21411
Num. Icepts =1048  Units = µm
```

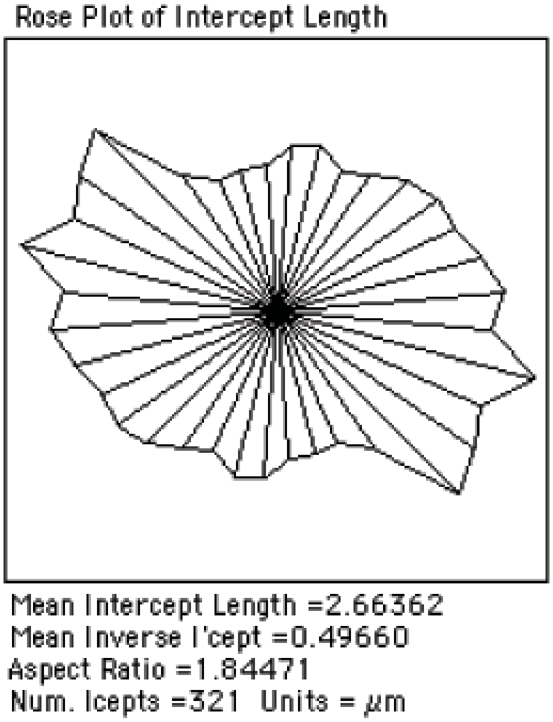
Parallel line grid overlaid on thresholded cells"

Intercept length plot as a function of orientation

This routine also reports the mean inverse intercept length, which is important for measuring the true three-dimensional thickness of a layer structure cut at random angles by the section plane. The true 3D thickness is (3/2) • 1/Mean Inverse Intercept Length. For example, in the image shown (*Thickness*), the mean inverse intercept is reported as 0.4966 µm^{-1}. That produces a calculated true thickness value for the sheaths of 3.02 µm.



!



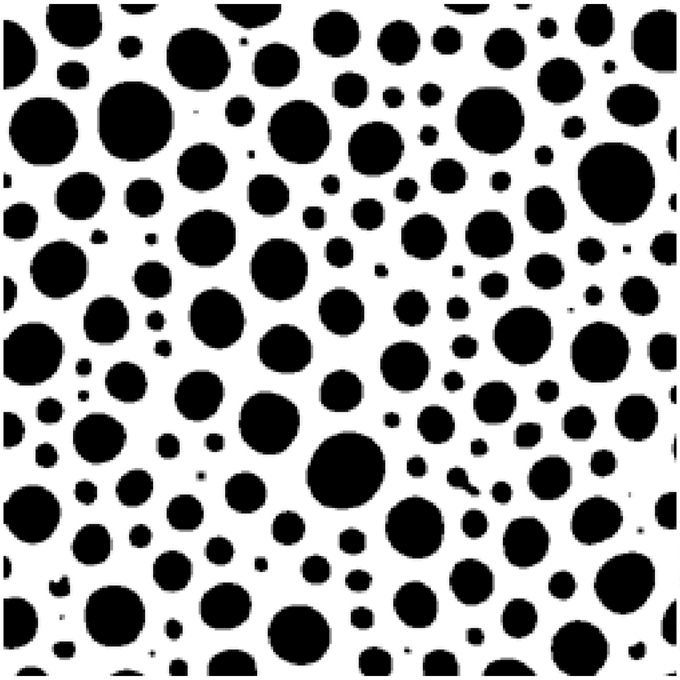
```
Mean Intercept Length =2.66362
Mean Inverse I'cept =0.49660
Aspect Ratio =1.84471
Num. Icepts =321  Units = µm
```

Thresholded image of sections through layers"

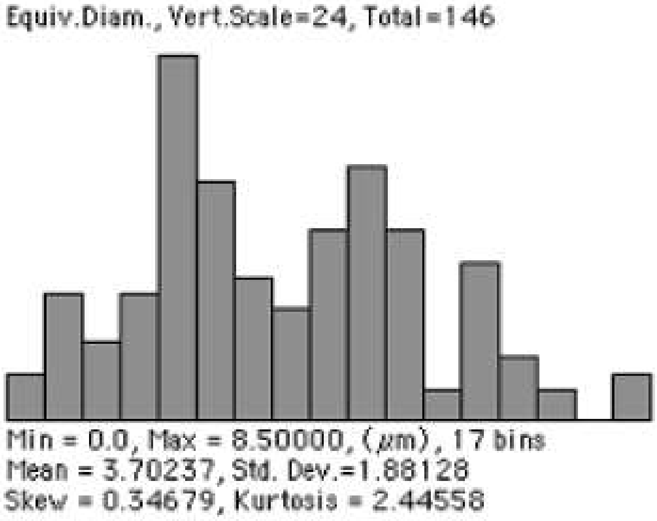
Measured intercept data

6.B.5. Number per unit volume

A single image does not directly reveal the number of objects per unit volume. A section plane is more likely to intersect large features than small ones. Counting the number of features seen per unit area is only indirectly related to the number per unit volume, but in some instances the desired information can be obtained. In the example shown (*Whip*), an image of bubbles in a whipped food product, it is reasonable to assume that all of the bubbles are the same size (created by the same gas pressure), and the different sizes of the circular intersections represent cutting through a sphere at different latitudes. It is easy to measure the size distribution, as shown, but not very useful. There are procedures for “sphere unfolding” that can use the size distribution of the circles with the assumption of a spherical 3D shape to calculate the size distribution of spheres that must have been present for random sectioning to generate the observed result, but they are notoriously sensitive to counting statistics and to any slight variations in shape of the structure.



Thresholded bubbles"

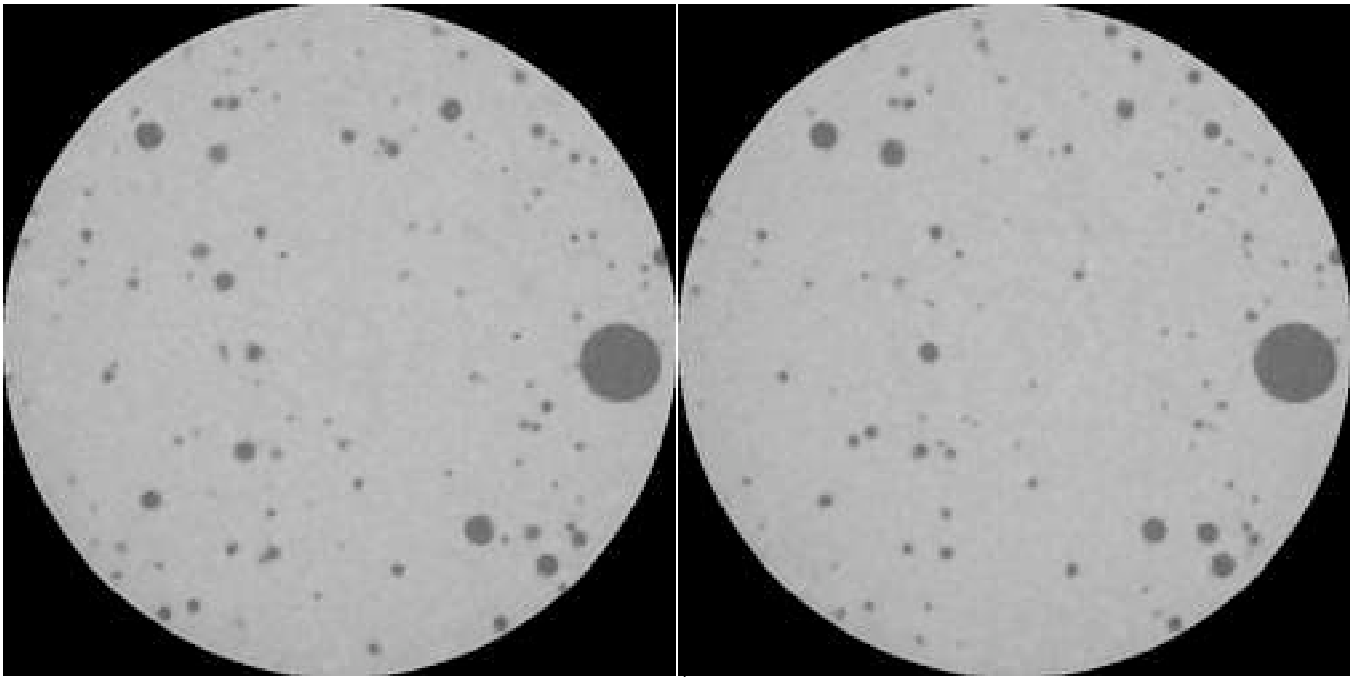


Measured 2D feature sizes

But for random sections through a unit sphere, the mean intersection circle has a diameter 2/3 that of the sphere. So for the measured mean circular diameter of 3.702 µm, the corresponding mean sphere diameter is estimated to be 5.56 µm. There are 146 features (corrected for edge-touching as discussed in Section 6.D) in an image area of 5053 µm^2. The relationship between number per unit volume (N_V) and number per unit area (N_A) is $N_V = N_A / D_{mean}$, so the calculated number per unit volume is 5.2 • 10^{-3} per µm^3. Since the volume fraction (measured by the area fraction as described above) is 40.22%, the mean volume of each sphere can be calculated (77.32 µm^3, which corresponds to a diameter of 5.29 µm and is in reasonable agreement with the original assumption). This method is applicable to any shape of particle, provided that a meaningful estimate of the mean diameter can be obtained.

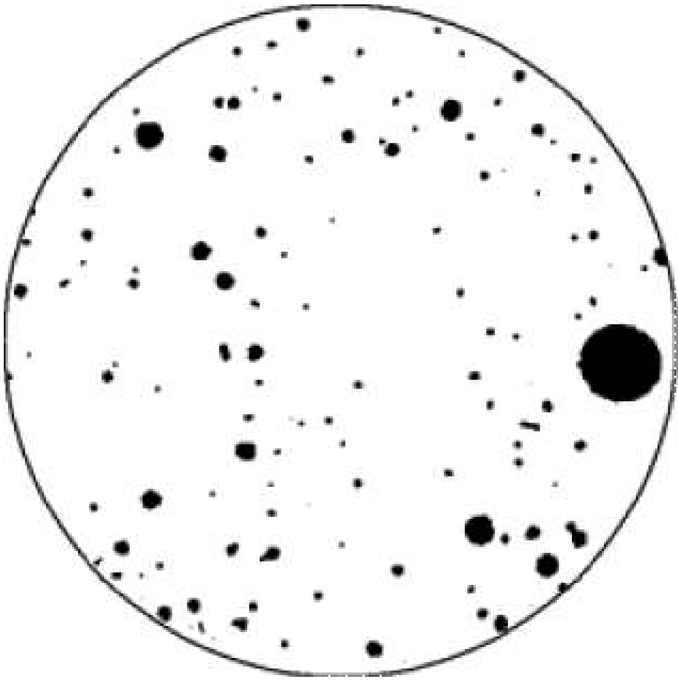
6.B.6. The Disector

A modern stereological technique for directly counting the number of features per unit volume requires imaging two section planes a known distance apart. This distance must be smaller than the size of any feature of interest, so that by comparing the two images it is possible to know what happens in the volume between the two planes. The images are aligned (this is usually conveniently done by placing them in layers) and features are counted that intersect either one of the two section planes (and thus appear in either one of the two images) but not in both. In the example shown (*Musketrs*), the two images are sequential CT slices, and confocal microscope optical sections are also convenient because they are automatically aligned. Each image is thresholded to show the features of interest (the holes). To automatically count the holes that are distinct between the two, the **IP•Math–>Select Features by 2nd** plug-in is used to combine copies of image 1 with image 2 as shown in section 5.C.2 and vice versa. ORing these two images together marks all of the features that cross one or the other of the image planes, but not both. These represent the tops and bottoms of the holes, so the number per unit volume is just half of the count divided by the volume (the product of the area times the distance between image planes).

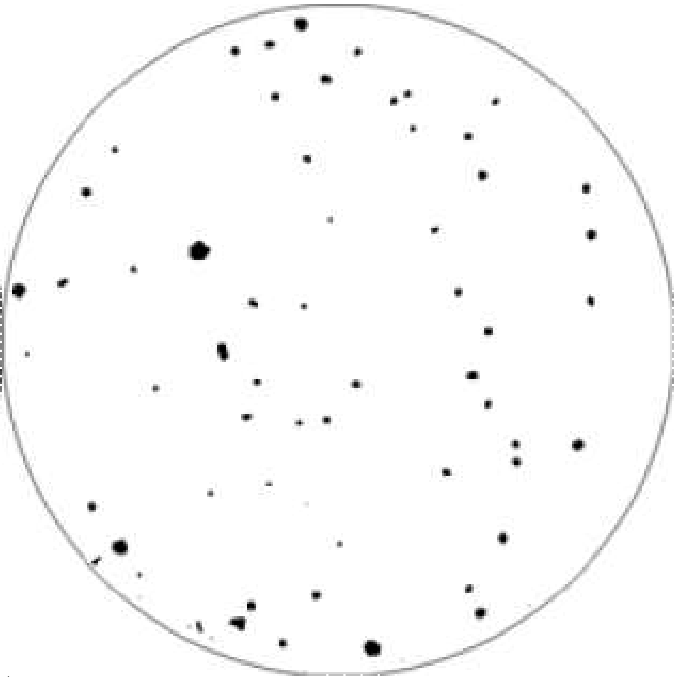


Two sequential sections through a candy bar (CT slice images)

There are 65 features that are present in either image and not matched in the other image. The area of the section is 1.54 cm^2, and the distance between them is 6.3µm. The number per unit volume is then calculated as (Number of Counts / 2) / Volume, or 32.5 / 9.7 mm^3, giving a net result of 3.35 holes per mm^3. Combined with the total volume fraction of holes (determined from the area fraction as described above), the average volume per hole is 0.0018 mm^3. This result is completely independent of any shape assumption, and is valid for features of varying shape, even if the shape varies with size. Furthermore, since the disector is a volume probe of the sample, it is not necessary to use special sectioning techniques to obtain isotopic orientations or special alignments.



Thresholded binary image of one slice"



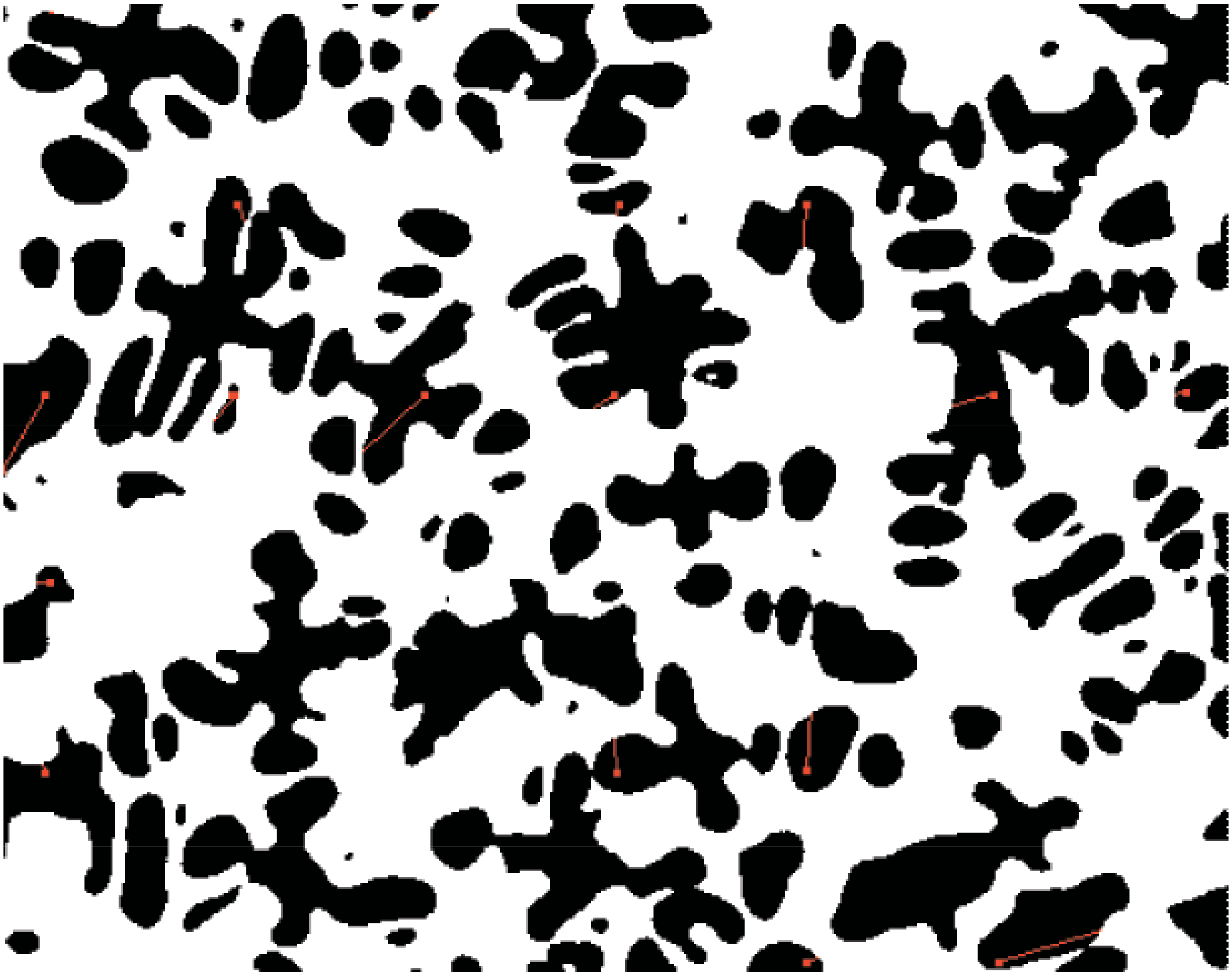
Features in one but not both sections, for counting

The disector can also be used to calculate the connectivity of network structures. In that case, three types of events must be counted by comparing the features in the two sections. As before, features that simply extend through both section planes without any change in topology (although of course they may change in shape or size) are ignored. Features that end between the two sections (and thus appear in one image but not the other) are counted as T++ events. Ones that contain holes in one image that have filled in in the other are counted as T— events. Features that split or branch between the two planes, so that a single feature in one image becomes two in the other, are counted as T+– events. The calculation is then Net Tangent Count equals the sum of the T++ and T— counts minus the T+– counts. From this, the connectivity of the structure (the number of cuts per unit volume that can be made without disconnecting it, also known as the Euler number for the network) is given by the relationship $N_V – C_V$ = (1/2) • Net Tangent Count / Volume. In most cases for a complex network structure, N_V, the number of objects per unit volume, is 1 and the connectivity C_V is determined primarily by the number of branching events in the volume between the two images. If appropriate, viewing the images in layers with partial transparency can be used with manual marking of the various types of events with different colors for counting.

6.B.7. Point sampled intercepts

By combining the disector method of determining the number of features per unit volume, which is not sensitive to shape, with a grid-based measurement method, it is possible to determine even more about the sizes of features present in a structure. The method of point-sampled intercepts places a grid of points onto the binary image. Each point that hits a feature in the image is used to generate a line whose length to the edge of the feature is measured. By using the length of each line to estimate the volume of the feature as (4#/3) • r^3, a mean value of the volumes can be obtained. However, this is the volume-weighted mean volume, not the more familiar number-weighted mean. That means that large features are more likely to be measured, because the points in the grid are more likely to fall on a large feature than on a small one.

But the usual number-weighted mean value of volume can also be determined by combining the volume fraction (measured in one of the ways described above) and the number per unit volume (determined using the disector). And the difference between the volume-weighted mean and the number-weighted mean is just the variance of the distribution of volumes of the particles, so the standard deviation of that distribution (the square root of the variance) can be determined, again without any sensitivity to feature shape.



Dendrite image, thresholded with a superimposed grid of point-sampled lines (colored for visibility)

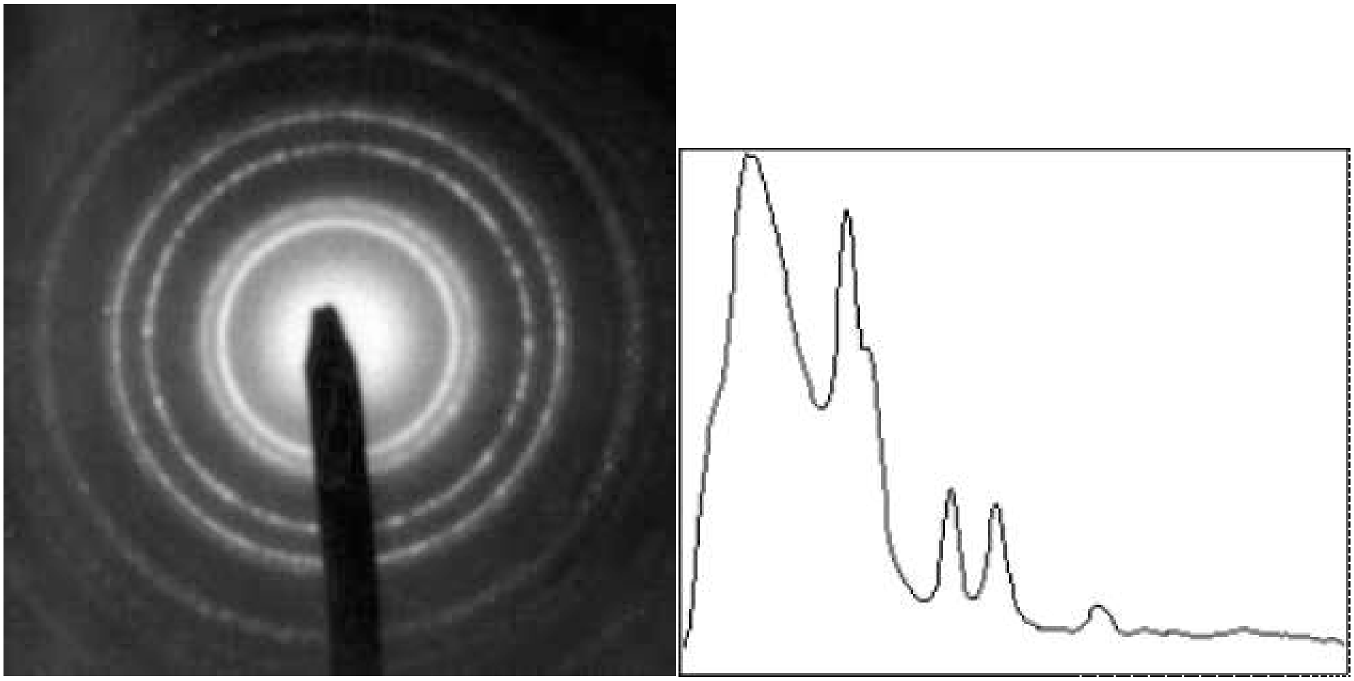
The procedure for applying this method is executed by the **IP•Lines and Points–>Point Sampled Intercept** plug-in, which generates a regular grid of points with the user-specified spacing. The angles of the generated line segments may be either uniformly distributed or sine-weighted (for use with vertical section images). In the example shown, this reports a mean cubed radius of 7.965 µm^3. Multiplying by (4#/3) gives a volume-weighted mean volume of 33.36 µm^3.

6.B.8. Other stereological methods

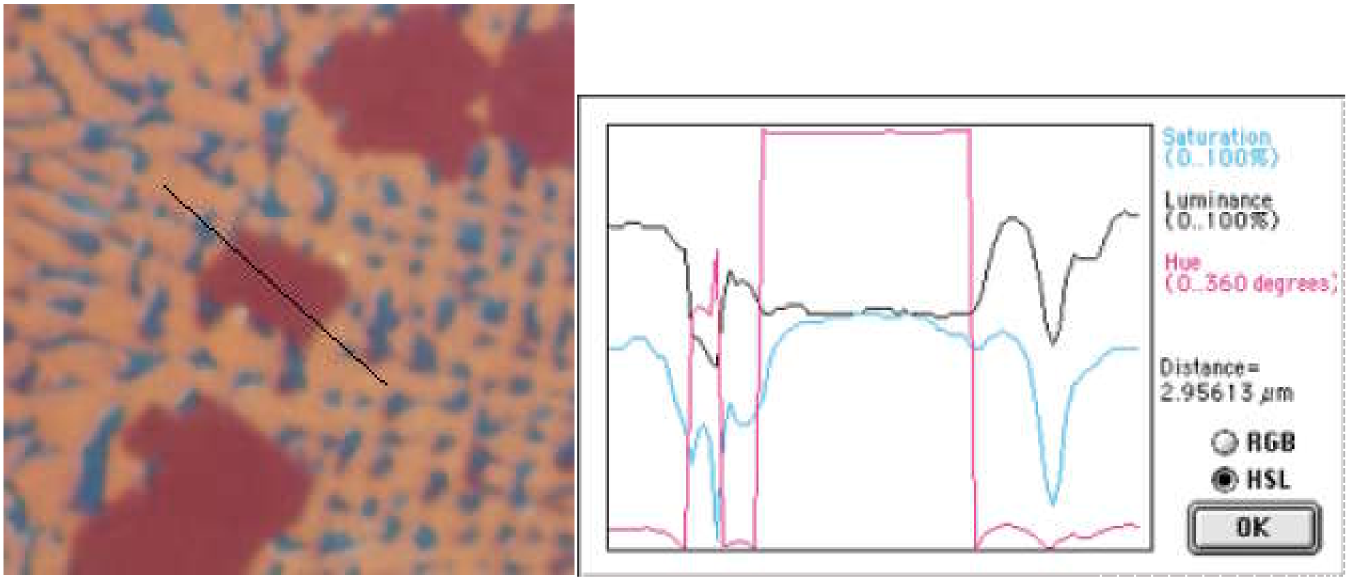
The plug-ins include routines that implement specific stereological techniques such as tangent counts, measurement of ASTM grain size in images of metal grains, and characterization of feature clustering or self-avoidance. These techniques and the examples shown do not exhaust the variety of measurements that stereology can provide as a meaningful description of the three-dimensional structure of solids based on the two-dimensional images that are captured for examination. Books on stereology, such as Russ & Dehoff, Practical Stereology, 2nd edition, 2002, Plenum Press, provide descriptions and examples of both the classical methods and the so-called “new” or “unbiased” techniques that depend heavily on proper sectioning procedures and are not dependent upon any assumptions about shape or isotropy. However, most of these methods use the same plug-in procedures already described, relying on the generation of an appropriate grid, its Boolean combination with a binary representation of the structure, and counting (or in some cases measuring) the result.

6.C. Intensity and Color

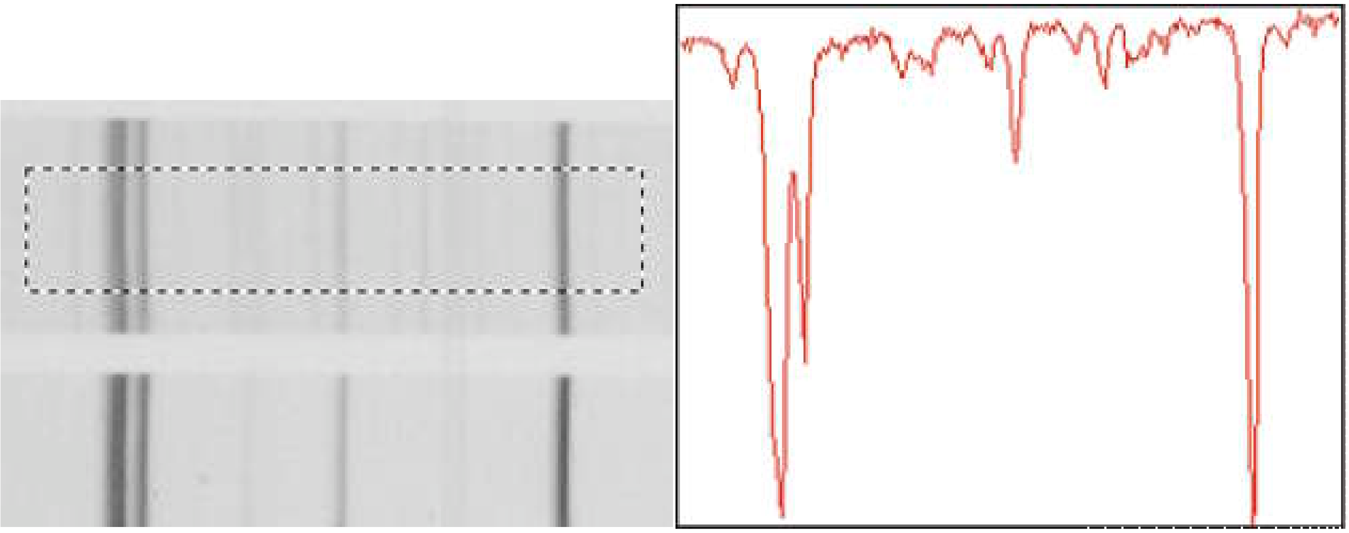
Profiles of intensity, calibrated density, or color information often provide efficient ways to extract data for analysis and can simplify measurement of dimension. Radial plots from the center of an image or region, plots along arbitrary lines, or averaged intensity values across horizontal regions are all saved in disk files for analysis in a spreadsheet. If an intensity calibration curve has been created or loaded, the calibrated values are shown. Spatial calibration is used to convert the pixel distances to real units.



Original *E_Diff* image and a plot showing the averaged radial intensity

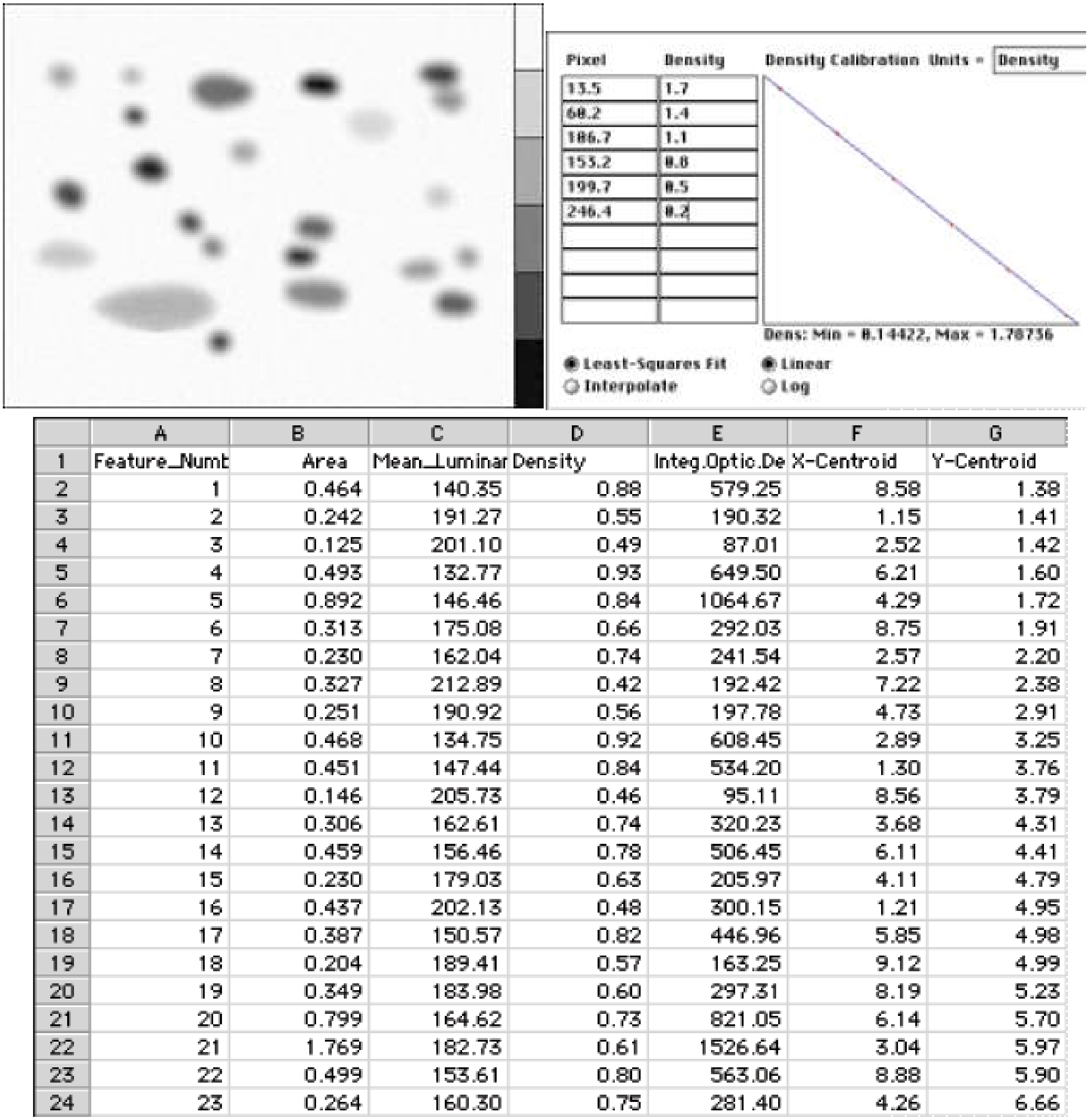


Original *Alloy* image and a line profile showing the color information



Debye image (fragment) with the averaged intensity profile of the region marked

Feature measurement parameters include brightness, calibrated density, and color information for the pixels within them. These data are saved in disk files, or can be used to label the display. Note that in the example shown, a calibration standard is scanned along with the gel and used to calibrate the density measurements.



Pixel	Density	Density Calibration Units = Density

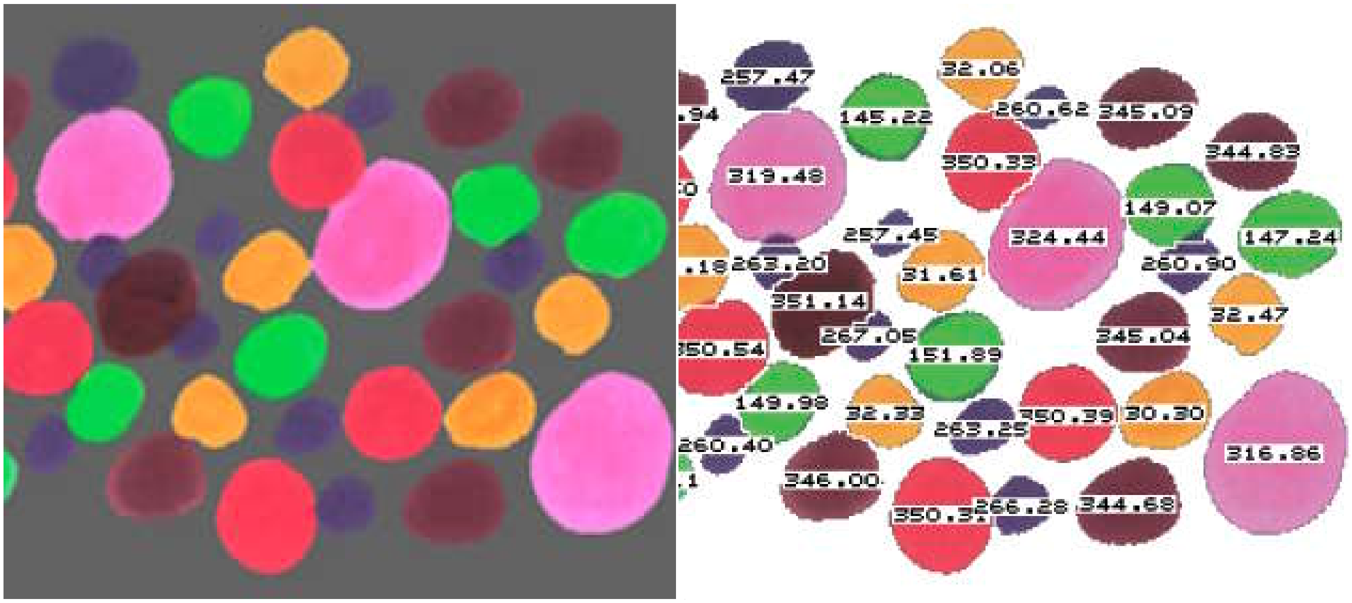
Pixel	Density
13.5	1.7
60.2	1.4
106.7	1.1
153.2	0.8
199.7	0.5
246.4	0.2

Dens: Min = 0.14422, Max = 1.78736

● Least-Squares Fit
○ Interpolate
● Linear
○ Log

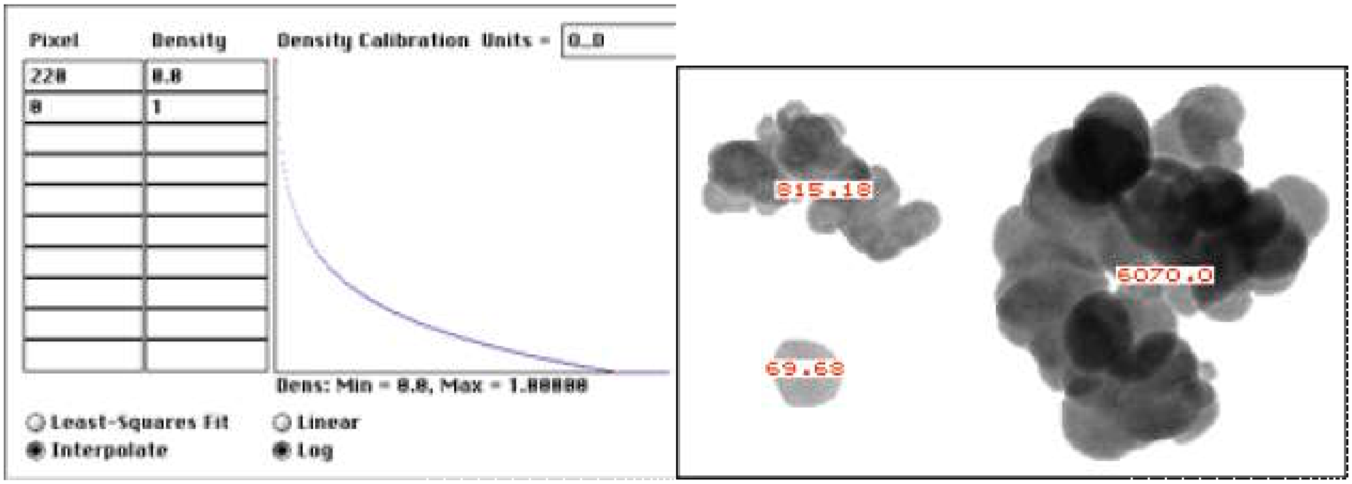
	A	B	C	D	E	F	G
1	Feature_Numb	Area	Mean_Luminar	Density	Integ.Optic.De	X-Centroid	Y-Centroid
2	1	0.464	140.35	0.88	579.25	8.58	1.38
3	2	0.242	191.27	0.55	190.32	1.15	1.41
4	3	0.125	201.10	0.49	87.01	2.52	1.42
5	4	0.493	132.77	0.93	649.50	6.21	1.60
6	5	0.892	146.46	0.84	1064.67	4.29	1.72
7	6	0.313	175.08	0.66	292.03	8.75	1.91
8	7	0.230	162.04	0.74	241.54	2.57	2.20
9	8	0.327	212.89	0.42	192.42	7.22	2.38
10	9	0.251	190.92	0.56	197.78	4.73	2.91
11	10	0.468	134.75	0.92	608.45	2.89	3.25
12	11	0.451	147.44	0.84	534.20	1.30	3.76
13	12	0.146	205.73	0.46	95.11	8.56	3.79
14	13	0.306	162.61	0.74	320.23	3.68	4.31
15	14	0.459	156.46	0.78	506.45	6.11	4.41
16	15	0.230	179.03	0.63	205.97	4.11	4.79
17	16	0.437	202.13	0.48	300.15	1.21	4.95
18	17	0.387	150.57	0.82	446.96	5.85	4.98
19	18	0.204	189.41	0.57	163.25	9.12	4.99
20	19	0.349	183.98	0.60	297.31	8.19	5.23
21	20	0.799	164.62	0.73	821.05	6.14	5.70
22	21	1.769	182.73	0.61	1526.64	3.04	5.97
23	22	0.499	153.61	0.80	563.06	8.88	5.90
24	23	0.264	160.30	0.75	281.40	4.26	6.66

Measurement of the optical density in the *GelSep* image, calibrated using the built-in standards



Original *ColrDots* image (fragment) and the features labeled with their hue, which is an angle from 0 (red) through green (120), blue (240) and so on to 360 degrees. The image was thresholded and watershed segmented to create a mask used to isolate the features for measurement.

The integrated optical density (IOD) provides a tool for counting features in three-dimensional clusters. Create a logarithmic (Beer's Law) calibration curve using the background intensity (measured as 220 on a representative area of the image shown) for zero optical density, and measure the IOD of each cluster. The ratio of the IOD for a cluster to that for a single particle estimates the number of particles in the cluster.



Density Calibration Units = O.D

Pixel	Density
220	0.0
0	1

Dens: Min = 0.0, Max = 1.00000

○ Least-Squares Fit ○ Linear
● Interpolate ● Log

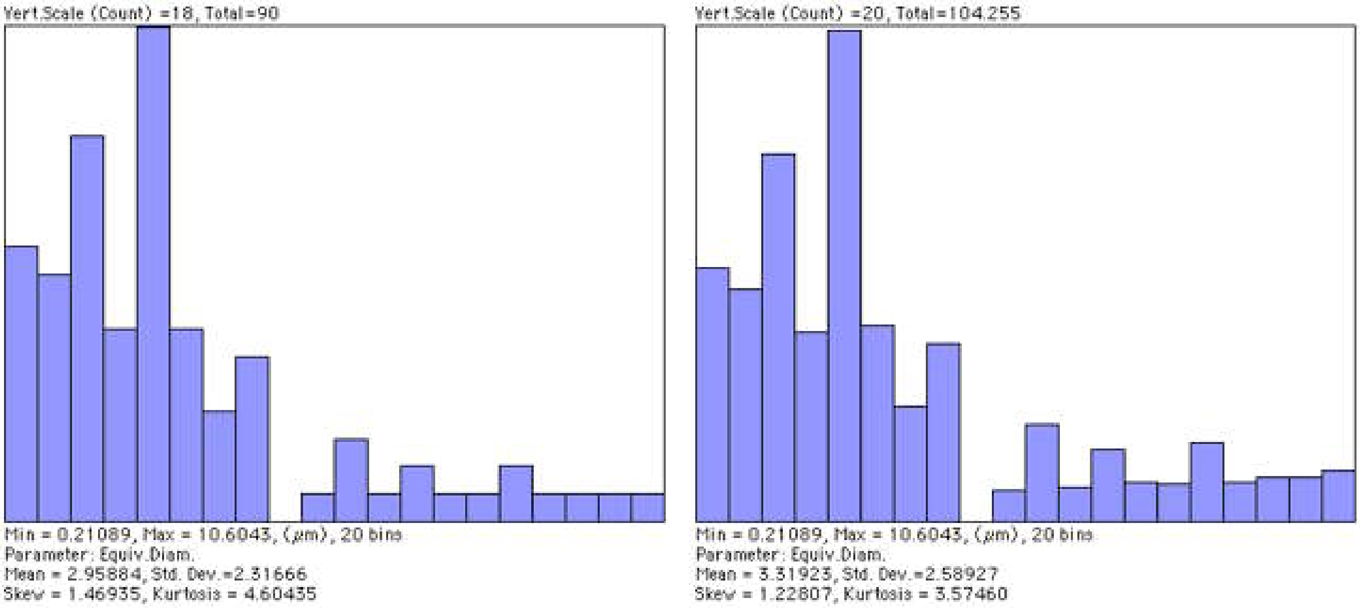
Calibration curve and IOD measurements for a few features in the *CarBlac2* image. The procedure estimates 12 and 87 particles in the two clusters, respectively.

6.D. Counting features

Counting the features present is not as simple as it may seem. Some features in the binary image may be eliminated based on size or shape (e.g., dirt). Unless the entire specimen is encompassed within the image borders, it is also necessary to correct for edge-touching features. For unbiased counting of the number per unit area, it is correct to count those which intersect two edges and not count those that touch the other two. In the example, counting all of the marks present (117) is incorrect. After eliminating those that touch the top and left edge (**IP•Measure Features–>Reject**), the unbiased count is 107 for the image area.

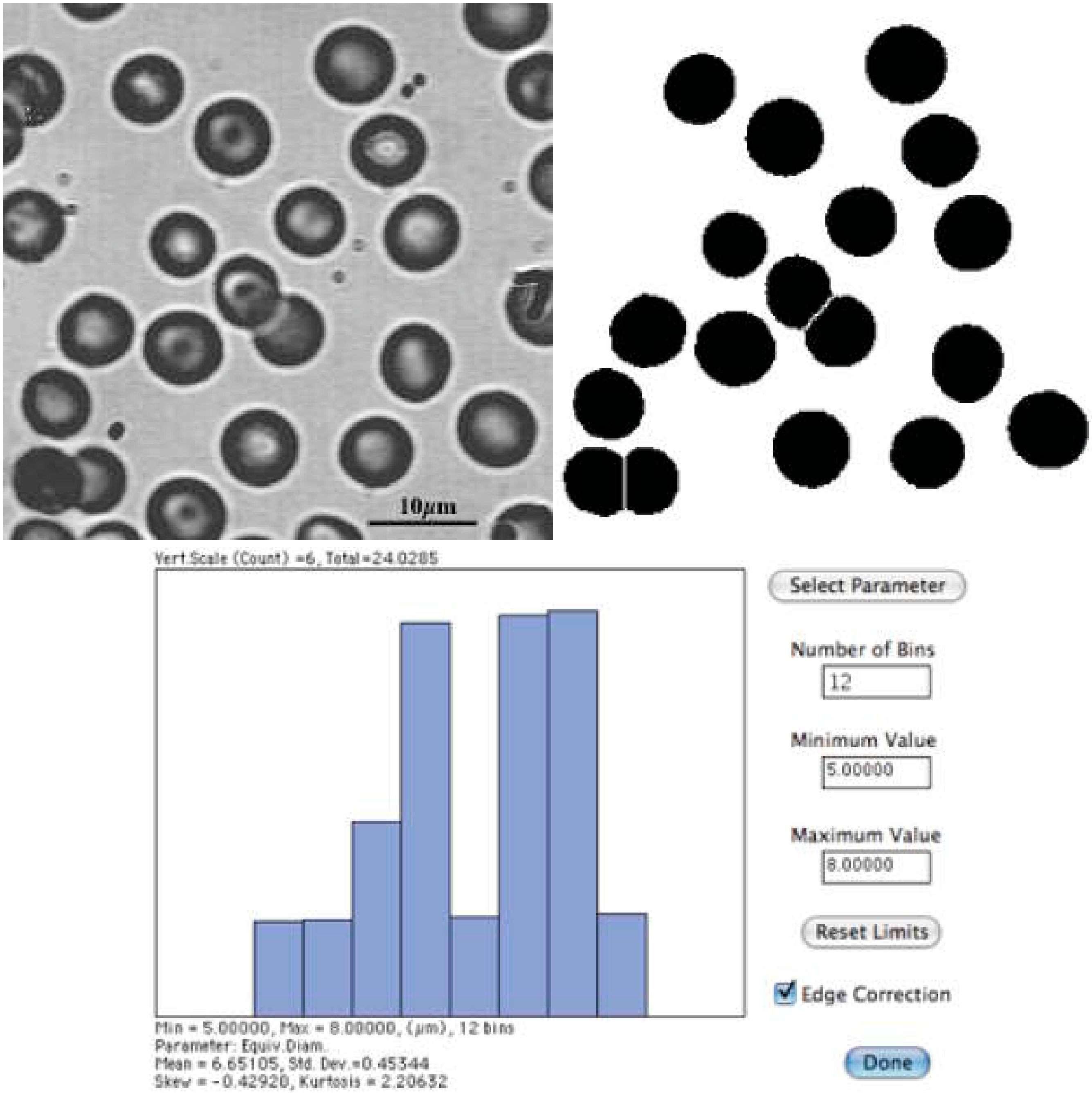
The thresholded white features in the *Dendrite* image

However, if the features are being measured as well as counted, features that touch any edge cannot be included (because they cannot be measured). Large features are more likely than small ones to be affected. The adjusted or effective count takes into consideration the probability that other features of the same size and shape would touch an edge and have to be rejected. The adjusted count for each feature is calculated from the feature dimensions, listed in the comprehensive output of feature measurements, and used when “Edge Correction” is selected in **IP•Measure Features–>Plot (Distribution)**. Note in particular the different total counts, mean values, and shapes of the distribution curves.



Measurements on the *Dendrites* image without and with correction for edge-touching features.

The next example shows a typical sequence in which some processing is required prior to measurement. The image is calibrated using the micron marker (which was then erased so that it is not included in the measurement), thresholded automatically (**IP•Threshold–>BiLevel Thresholding–>Auto**), the holes filled (**IP•Morphology–>Fill Holes**), small bits of dirt and features touching two edges removed (**IP•Measure Features–>Reject Features**), watershed segmentation applied to separate the touching cells (**IP•EDM–>Watershed Segmentation**), and finally the number per unit area counted and average size determined.



Vert.Scale (Count) =6, Total=24.0285

Number of Bins: 12

Minimum Value: 5.00000

Maximum Value: 8.00000

☑ Edge Correction

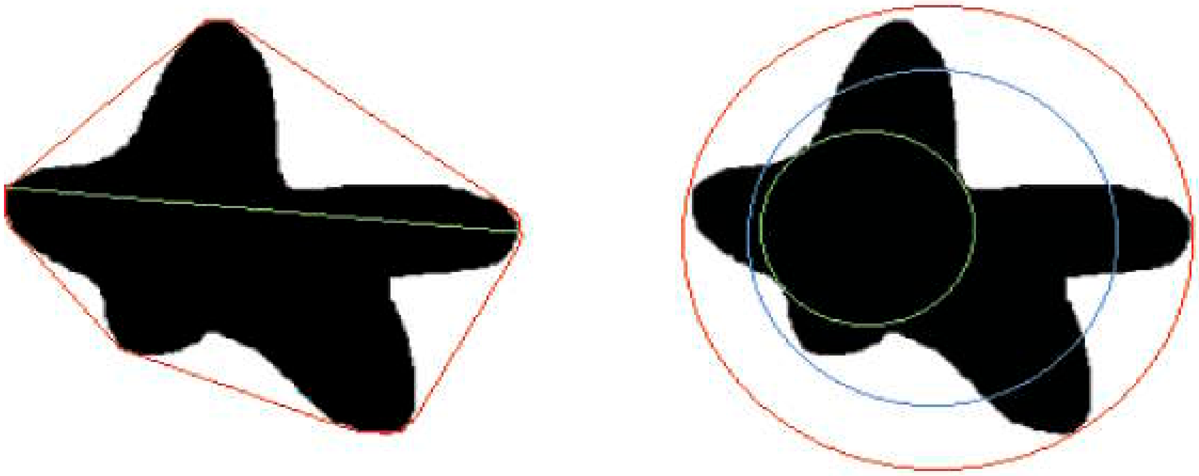
```
Min = 5.00000, Max = 8.00000, (µm), 12 bins
Parameter: Equiv.Diam.
Mean = 6.65105, Std. Dev.=0.45344
Skew = -0.42920, Kurtosis = 2.20632
```

The original *BloodCel* image, binary result after thresholding and processing, and size measurement.

6.E. Measuring features

Choosing the appropriate measurement is important. Knowing as much as possible about the application is important in choosing the measurements that are meaningful. Feature measurements can be grouped into four categories:

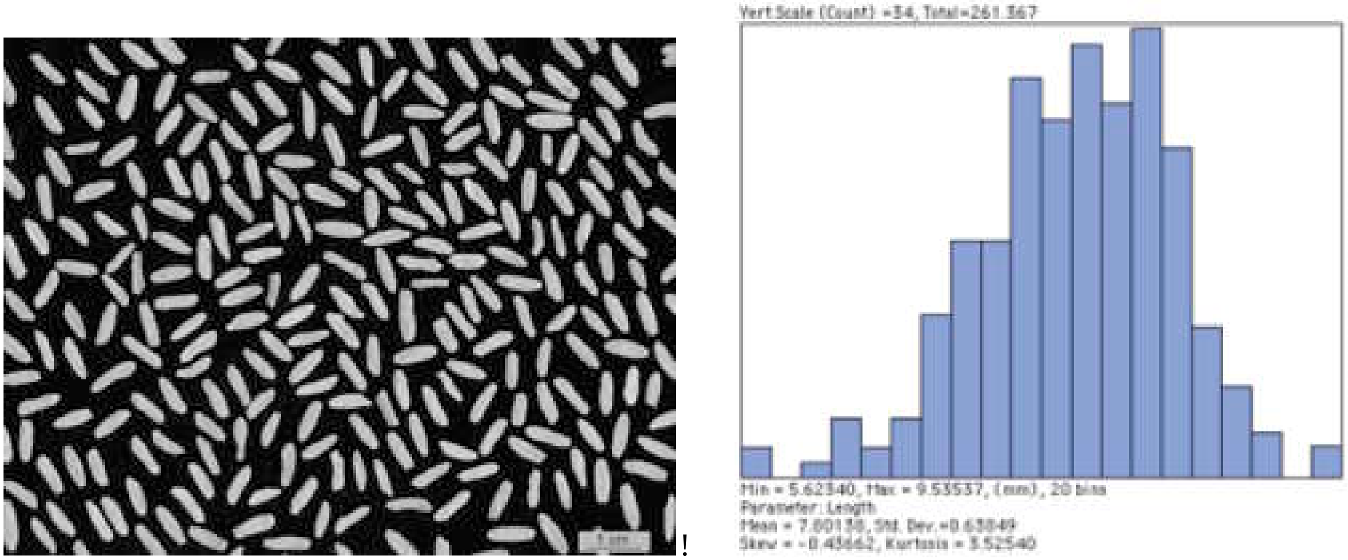
Size *(area, length, breadth, perimeter, etc.)*: Most of these measures are easily understood. The perimeter is measured using an exceptionally accurate super-resolution method. The convex area and perimeter are based on a convex hull (also called a rubber-band or taut string bounds).



!

An irregular feature with the maximum distance (“length”), convex bounds (used for convex area and convex perimeter), circumscribed circle, inscribed circle and equivalent area circle shown.

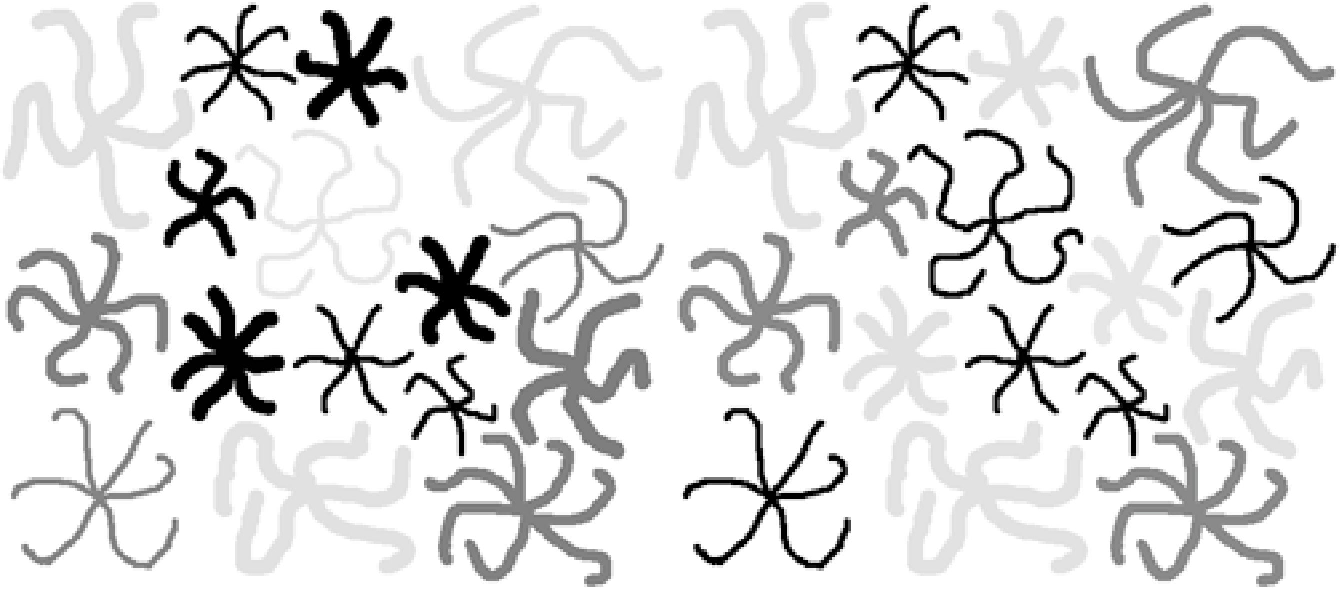
In many cases, the size distribution of features is based on the area, the equivalent circular diameter, or the maximum caliper dimension (length). The example shows the length of rice grains.



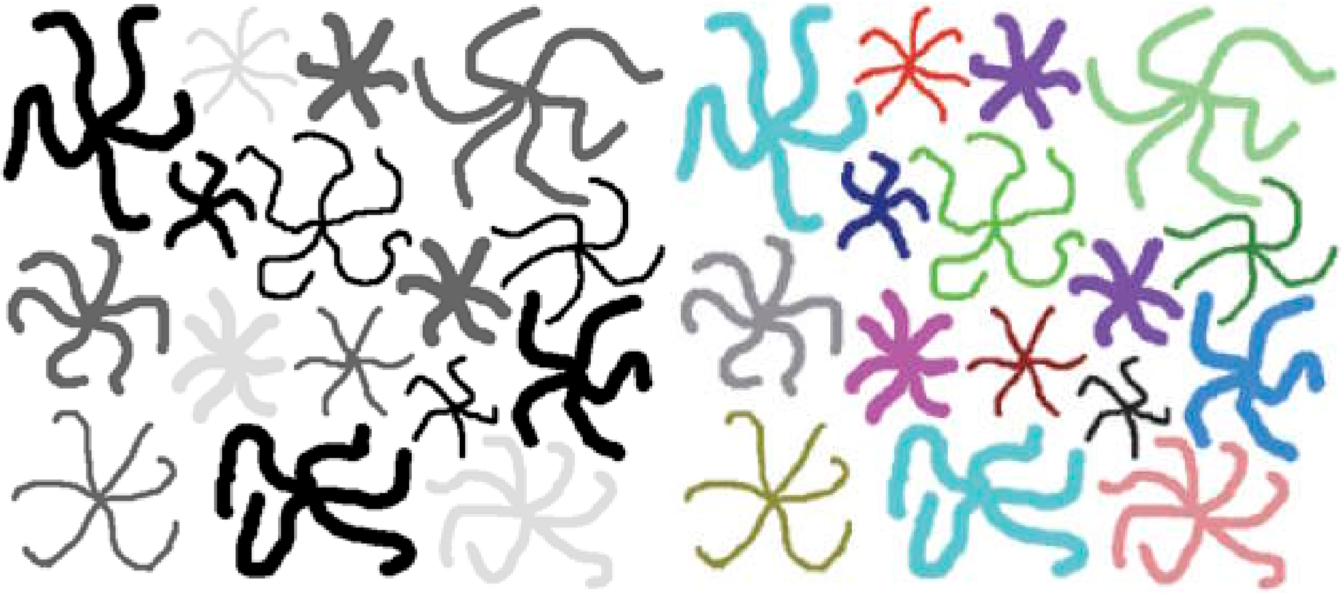
!

Original *Rice2* image and distribution of lengths of grains

The length or breadth of a curved feature is determined as shown in Section 5.E.1. using the skeleton and the Euclidean distance map. The example shows several irregular star-shaped features that are classified and color-coded by the number of arms, the width of the arms, and the length of the arms. By assigning values proportional to each of these measurements in the R, G and B channels, the pairs of features that share all three properties are given the same color and visually identified.



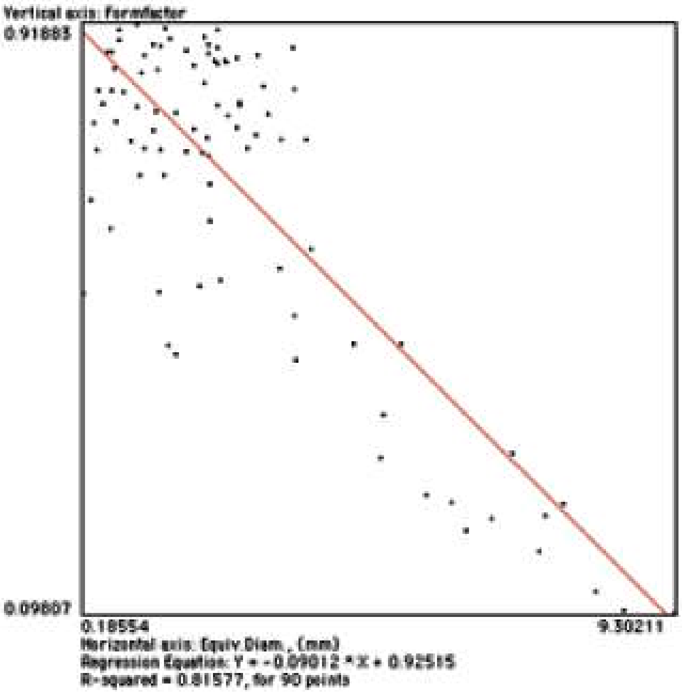
The *Stars2* image with features shaded according to the length and breadth of external branches.



Features coded by the number of branches, and with values for all three parameters in RGB channels.

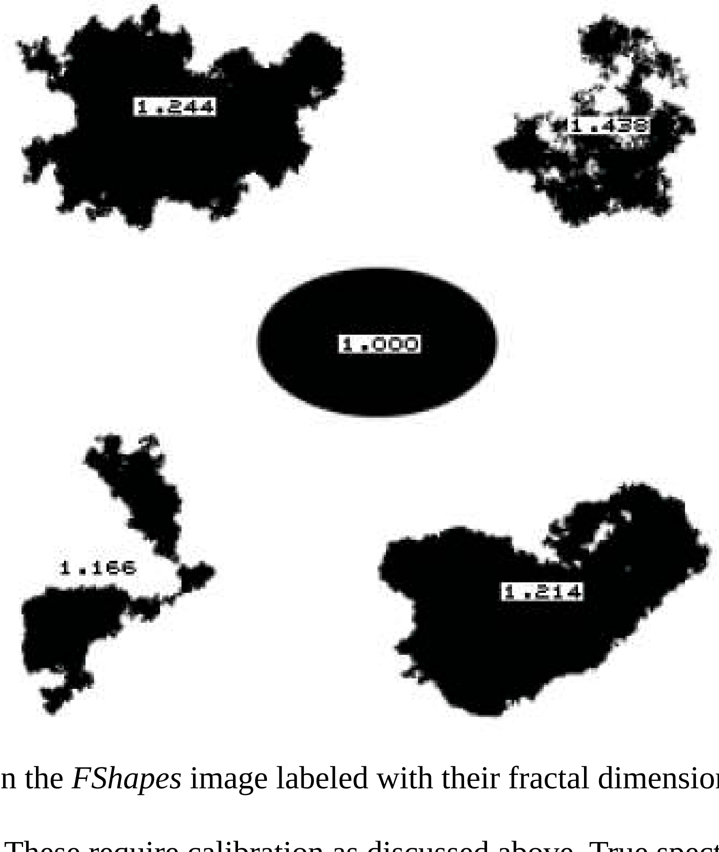
Shape *(topology, dimensionless ratios, fractal dimension)*: Shape is one of the most difficult properties to describe either verbally or numerically. Ratios of size parameters are convenient but not specific, and are given arbitrary names that are neither unique nor consistent. Topology, using the feature skeleton, captures one important aspect of shape as shown above. Fractal dimension corresponds to the boundary irregularity or “roughness”.

The example shows a regression plot of feature shape (form factor, defined as 4#Area / Perimeter^2) vs. size (equivalent diameter) for the *Dendrites* image shown above. The strong correlation results because the small features are intersections with the branches of the dendrites (which tend to be round) while the large features are intersections with the trees (which are more irregular). Fracture, agglomeration, etc. all tend to produce variations of size with shape.



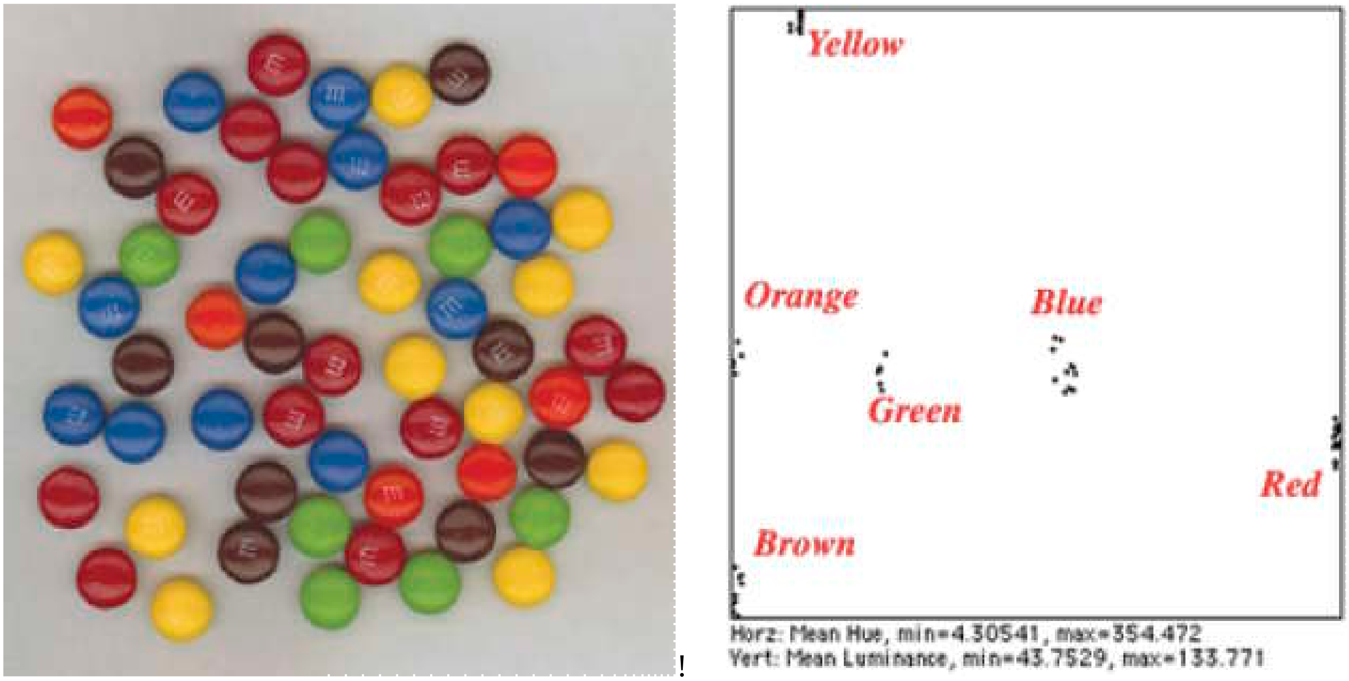
Scatterplot of size vs. shape for the features in the Dendrite image

The fractal dimension of Euclidean shapes (circles, squares, etc.) is 1. As the irregularity ("roughness") of the feature boundary increases, so does the dimension.



Features in the *FShapes* image labeled with their fractal dimension values.

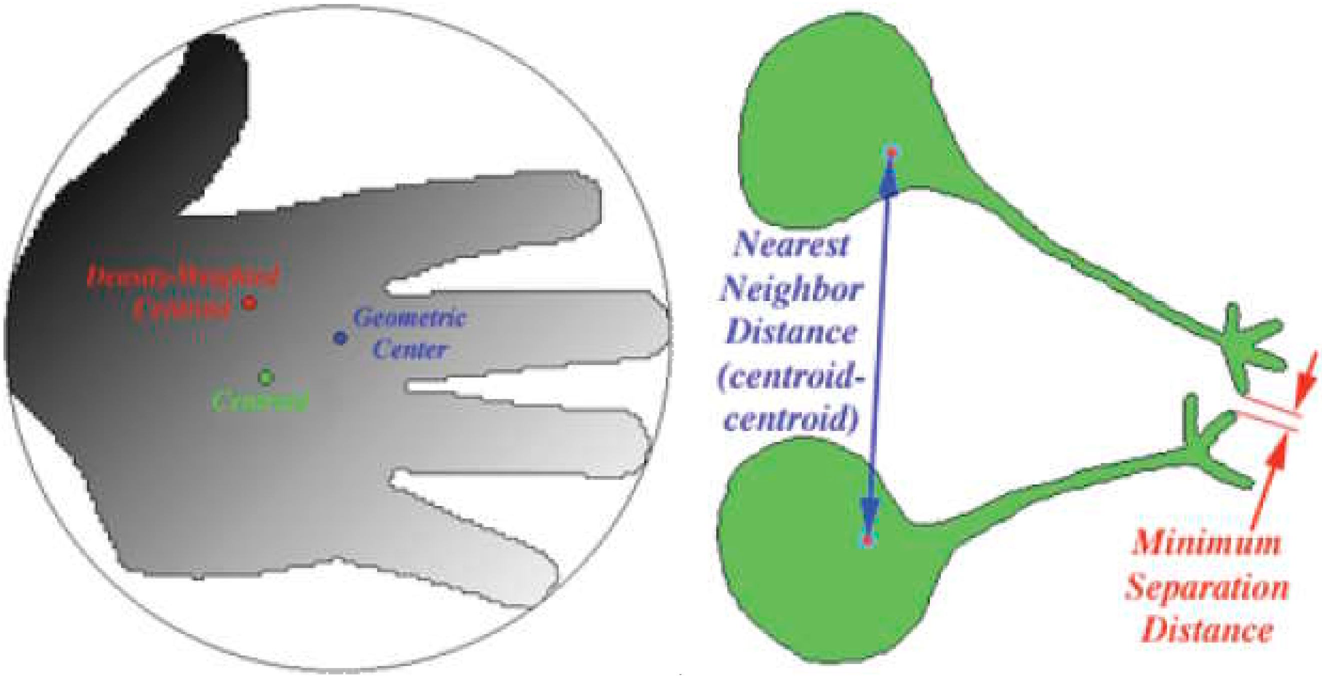
***Intensity** (density, color)*: These require calibration as discussed above. True spectral information cannot be obtained from a tristimulus camera or scanner, but the hue values for features often correspond to their visual "color." The example shows the measurement of features (after thresholding, watershed segmentation, and masking of the original image) and a scatterplot of hue vs. intensity that groups the various candies.



Original *MandM* image, and a labeled plot showing the hue and intensity values of the various candies

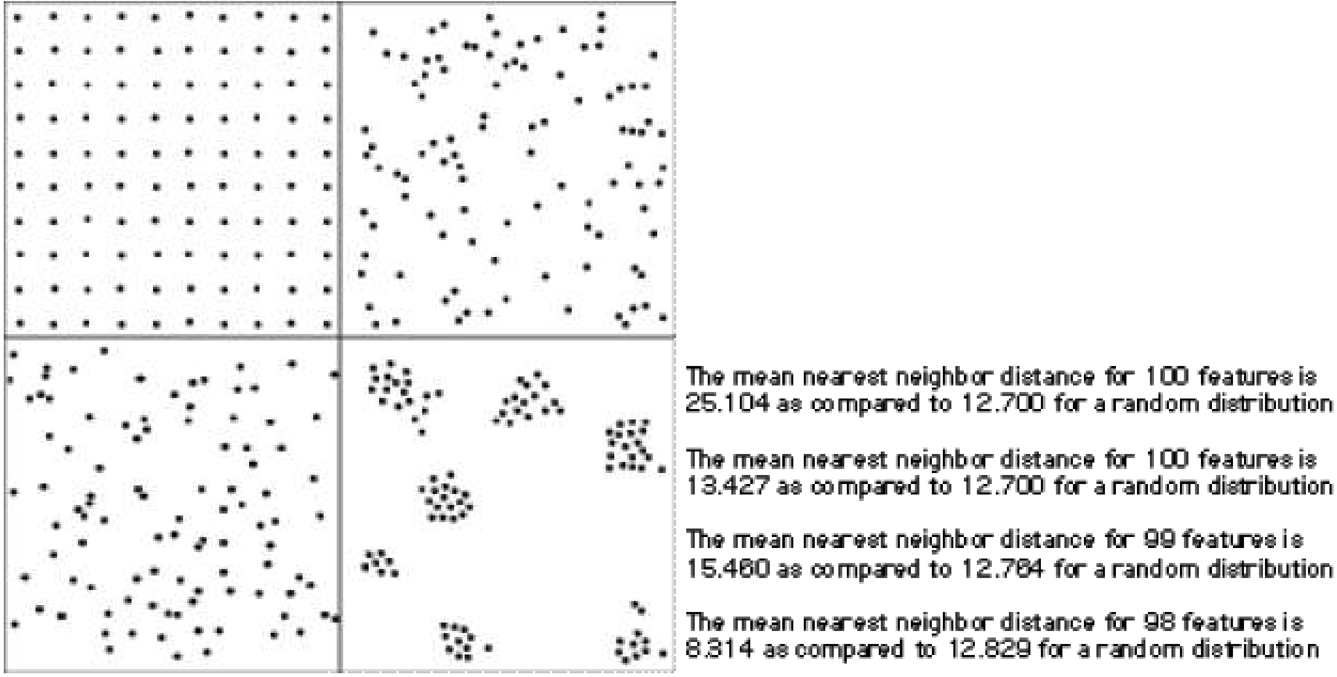
***Location** (absolute coordinates or distances from other objects)*: The coordinate position of features is measured from the upper left corner of the image, in whatever calibrated units have been established. This can be important for specific cases such as locating spots on scanned gels, but in most cases it is the position of features relative to other features present in the image that is most interesting. Neighbor relationships can be important for understanding structure.

There are several different points that can be chosen for the location of a feature, and distances can be measured from centroid to centroid or edge to edge. Spatial clustering or self avoidance can be measured by comparing the mean nearest neighbor distance to 0.5 • SQRT(Area / Number). Adjacent neighbors can be counted. Note that the definition of "adjacent" in this routine is features that are separated by a single-pixel-wide four-connected line. In most cases the easiest way to produce such an image is to skeletonize the background between the features (or the cell walls or grain boundaries in a structure). The skeleton is an 8-connected line (pixels touch any of their eight neighbors), which can be converted to a 4-connected line (pixels touch only their four edge-sharing neighbors) that separates features by selecting **Morphology –>Thicken Skeleton**. Then invert the image so that the lines are white and the features black, and count adjacent neighbors.


Density-Weighted Centroid
Geometric Center
Centroid
Nearest Neighbor Distance (centroid-centroid)
Minimum Separation Distance

Definitions for location and neighbor distances

The examples show the measurement of clustering and adjacent feature counts. Section 5.E.1 showed the measurement of individual feature distances from a boundary.



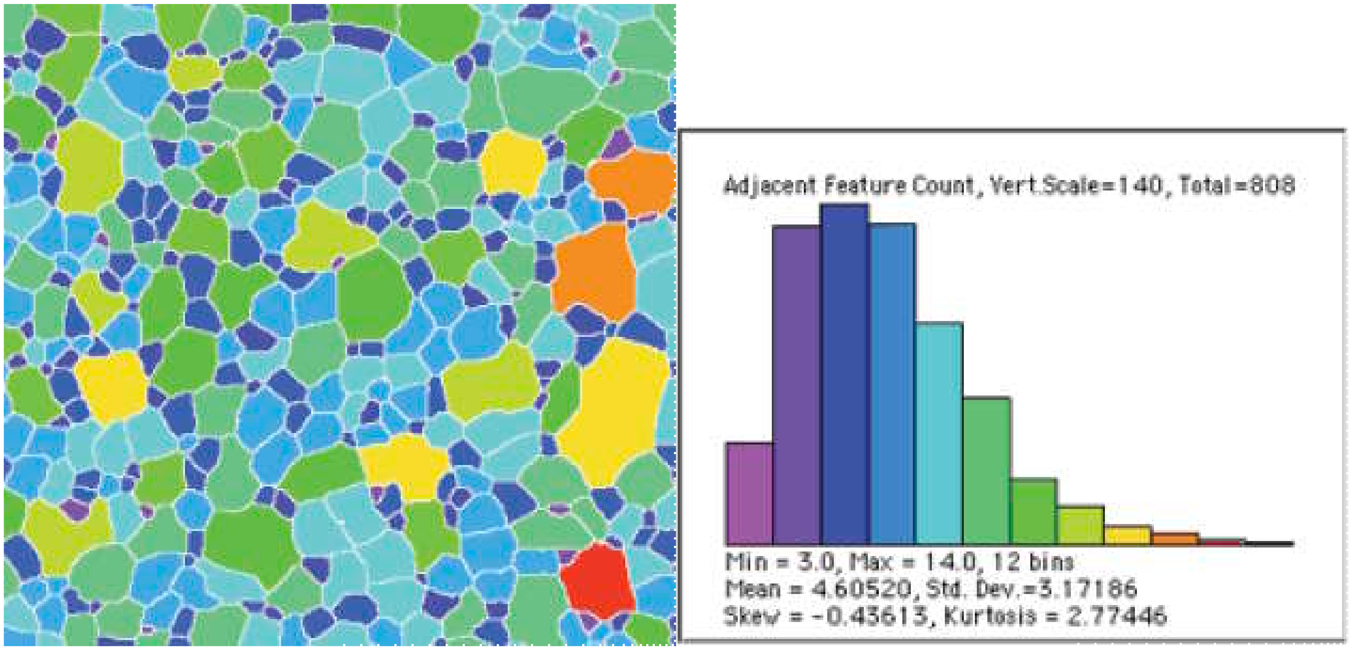
The mean nearest neighbor distance for 100 features is 25.104 as compared to 12.700 for a random distribution

The mean nearest neighbor distance for 100 features is 13.427 as compared to 12.700 for a random distribution

The mean nearest neighbor distance for 99 features is 15.460 as compared to 12.784 for a random distribution

The mean nearest neighbor distance for 98 features is 8.314 as compared to 12.829 for a random distribution

Clusters image and the measurement of clustering results for the four regions

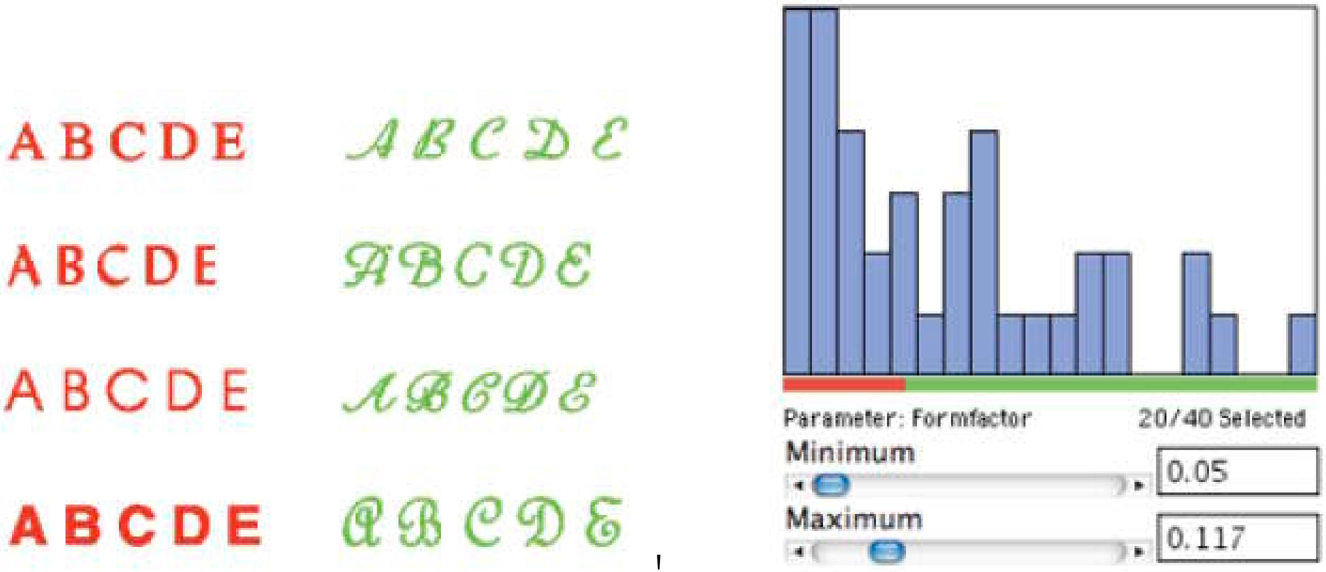


Gr_Steel image with the grains color-coded by number of adjacent neighbors and a distribution plot

!

6.F. Data analysis and feature recognition

Measurement data can be used for feature classification by establishing a recipe with limits on various measurement parameters. This is equivalent to carrying out a series of feature selection measurements to isolate each set of objects. In the first example, the **IP•Measure Features–>Select Features** routine is used to select the characters in the script fonts based on their formfactor. The selected features are given the foreground (pen) color and the others are given the background (eraser) color.



Parameter: Formfactor 20/40 Selected

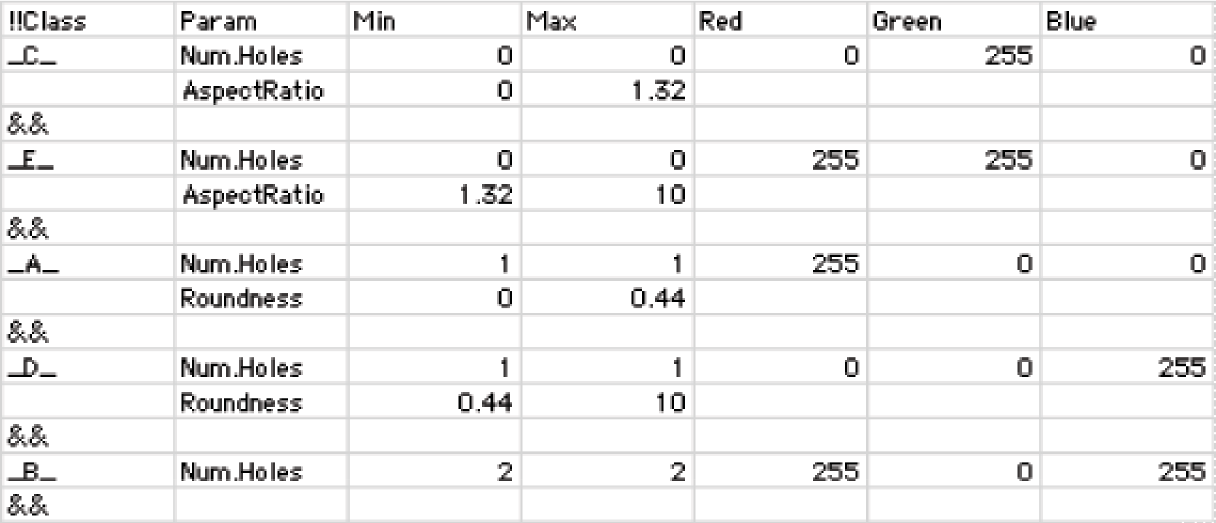
Minimum 0.05

Maximum 0.117

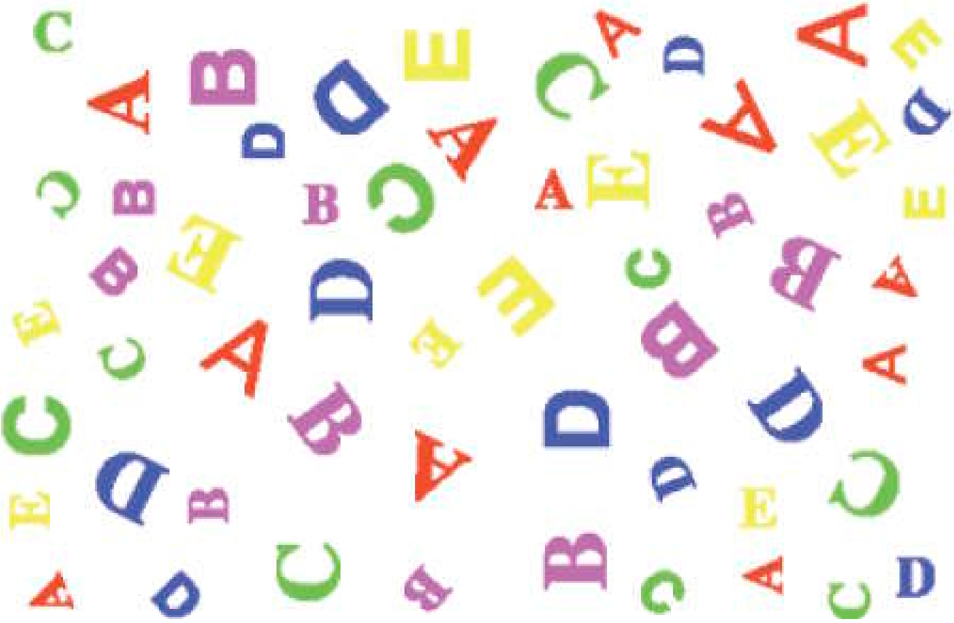
!

The *Fonts* image after selecting features based on formfactor

A recipe file contains the parameter names and limits and optionally the color values used for labeling. The example shown is used to identify (and color code) the letters A through E, in different fonts and sizes. **IP•Measure Features–>Color by Value** and **–>Label Features** allow selection of a class parameter when a recipe file has been defined (**IP•Utilities–>Select Recipe File**), and the class is included in the output from the **IP•Measure Features–>Measure All Features** routine.



!!Class	Param	Min	Max	Red	Green	Blue
_C_	Num.Holes	0	0	0	255	0
	AspectRatio	0	1.32			
&&						
_E_	Num.Holes	0	0	255	255	0
	AspectRatio	1.32	10			
&&						
_A_	Num.Holes	1	1	255	0	0
	Roundness	0	0.44			
&&						
_D_	Num.Holes	1	1	0	0	255
	Roundness	0.44	10			
&&						
_B_	Num.Holes	2	2	255	0	255
&&						



A recipe file used to automatically color code the letters in the A2E image

7. Automation

7.A. Photoshop Actions and Fovea Pro Logging

Photoshop provides capabilities for recording and playing back Actions, sequences of operations that can automate complex operations. All of the Fovea Pro plug-ins are compatible with actions: the operations are recorded along with any user selections or parameter settings, and those settings are used again when the action is played back. When used with other programs that support the use of Photoshop-compatible plug-ins, this capability is ignored but manual, interactive use is unaffected.

Actions are used to record a series of operations, so that you can document or repeat the process by which an image was manipulated, processed, thresholded, and/or measured. You begin the action recording by selecting New Action in the drop-down menu in the Actions window. You can name this action, and it will record all operations, even ones involving multiple images, until you select Stop Recording from the same drop-down menu. Each step is added to the action being recorded, along with all of the details (click on the small triangle next to the step name to reveal the various parameter settings).

Photoshop functions which can be included in actions either have no parameters at all (e.g., changing image modes, select all, etc.) or have input parameters (e.g., setting the size of a Gaussian smoothing neighborhood or the location of a selection region in the image). Those plug-in functions which fall into these categories behave similarly (e.g., performing an FFT requires no input parameters, generating a surface display requires entering several numbers and selecting radio button options in the dialog). When a step in an action has a dialog box associated with it in which the user can enter parameters, the action can either bypass the display and proceed using the parameters stored in the action, or display the dialog (with the input parameters previously selected set as the defaults) and wait for new input or for the user to proceed. This is identical to the use of Photoshop's shortcuts to repeat the last function: press Control/Command - F to repeat an action with the previous settings, or Alt-Control/Option-Command – F to repeat an action but put up the dialog for user input.

To illustrate creating a simple but very useful action, consider setting up a function key to execute the very frequently used "Setup 2nd Image" function. The full step-by-step procedure is as follows:

1. There must be an image open in the program.
2. Be sure the actions palette is visible (if not, select **Window –> Show Actions**)
3. Click on the small triangle at the upper right corner of the Actions palette and select **New Action** from the drop-down menu.
4. Give the new action a descriptive name, and optionally assign it to a function key, such as F2.
5. Click record.
6. Perform each step of the action by selecting the functions in the usual way. For this specific case, which has only a single step, select **IP•2nd image –> Setup 2nd**.
7. Stop recording. Either click the icon (shown in red) at the bottom of the Actions palette or select **Stop Recording** from the drop down menu opened by clicking on the triangle at the top right corner of the palette.
8. The action is now ready for use. Pressing the F2 function key will execute it and place whatever image you have open into the second image storage location. You can also execute an action by highlighting it in the list of actions (click on the name) and then click on the triangle icon at the bottom of the action palette.

Actions can, of course, have many steps in them. Sometimes in the process of recording an action you may need to stop recording (press the round "stop" icon at the bottom of the palette), perform some other operations, and then resume recording (press the square "start" icon at the bottom of the palette). In other cases, you may find that the recorded action has extra, unnecessary steps included, or is missing steps (a common mistake is to assume that the foreground and background colors will be set correctly; it is always wise to set them appropriately in the action). If you need to edit an action, you can do so. To remove a step from an action, highlight the step and select Delete from the drop-down menu in the Actions palette. To add a step, click on the preceding step and then record the one(s) to be added, or click on the desired step in another action and drag it to the desired location (you can also use click and drag to move steps around within an action). Sometimes it is useful to first create a dummy action with the missing steps so they can be dragged into your action; the dummy action can be deleted later.

It takes a little bit of practice and planning to create a lengthy action and get it right, but it is definitely worth the effort because once created it can be used over and over to perform the same operations flawlessly and quickly. Recorded actions can be saved to disk (select the entire set of actions and use the

Save Actions selection in the drop-down menu), loaded from disk (select Load Actions), edited (either by changing parameters for a step, or by dragging steps around in the action to change the order of operations, insert or remove steps, etc.), assigned to function keys, and called as steps by other actions. In fact, this last capability is particularly useful as it allows building up a “library” of subroutines that can be quickly assembled into a higher level process as needed. Actions are a basic tool for Photoshop users, and it is not the intent of this tutorial to duplicate the information on them provided in the Photoshop manual. One tip that is not widely documented is that depressing Command-Option (Mac) or Control-Alt (Windows) while saving an action set creates a text file listing all of the steps in your actions, which can be useful for documentation.

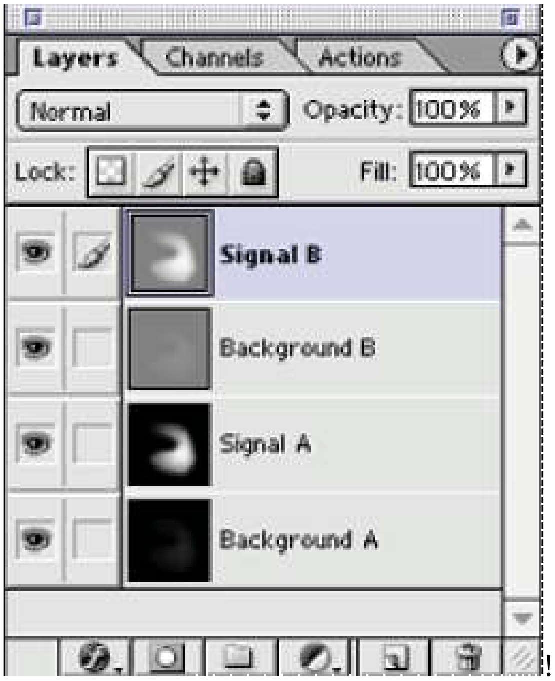
When a Fovea Pro plug-in has output data, it may either be in the form of a stored data file (e.g., **IP•Measure Features–>Measure All Features**), data and/or graphics shown in a dialog (e.g., **IP•Measure Global –> Clustering**), or both (e.g., **IP•Measure Global –> Histogram**). If you set the action to pause at the dialog for the step, the dialog will be shown. If not, it will be bypassed. Data saved to a file will be saved to the same location as previously, which may be desired (e.g., appending measurements to an existing file) or may not (e.g., overwriting an existing file). Settings from dialogs are recorded in the action, and data and graphics shown in the dialog will be recorded if you are using logging in Fovea Pro (logging is not provided by The Image Processing Tool Kit).

Dialog data and graphics can be preserved using the **IP•Utilities –> Setup Log File** function. This is used to select a location for the files that are created and appended to. A tab-delimited text file is written into which each output value (along with a short text label) is written. If there are many such output values from a single action, they are all written unto a single line in the text file, with tab separators. The **IP•Utilities –> Log New Line** function should be placed at the end of the action. It writes the date and time to the file, and then a carriage return so that the next image processed by the action will have its data written to the next line. These text files can be easily read by any word processor and also opened by Excel, which will place all of the similar data values conveniently into columns. This is particularly suitable when performing batch operations. An html file containing the dialog graphics and data output is also created, and can be accessed with any html compatible reader such as Internet Explorer, Safari or Firefox. The graphics are linked to this index file but are saved as individual, sequentially numbered *.tif files and can be used in reports as desired.

As an example of an action that uses another action as a subroutine and logs the data to an output file for analysis, consider the task of measuring the brightness of a region in each of a series of images (loaded as layers) so that they can be combined in a ratio. The logging capability of Fovea Pro makes it relatively easy to accomplish this with actions. The steps of creating and using the action are:

1.! In the Actions palette, click on the arrow for the drop-down menu, select New Action and give it a meaningful name. There will actually be two actions in this procedure so this one should be called something like “Setup”.
2.! Since the action accesses the various layers by deleting them after they have been measured, the first step in your action should be to duplicate the entire image (**Image –> Duplicate**).
3.! Select the log file for saving the data (**IP•Utilities –> Setup Log File –> Create New Log File**)
4." Put in a Stop (select **Insert Stop** from the Actions palette menu, with the message to draw in the region of interest to be measured.
5.! Draw the region of interest if you are doing this manually, or establish it by processing and thresholding to create a binary image. Note: If you do it the latter way, it is useful to have a separate action that will convert the binary image to a selection by **Select –> All**, **Image –> Invert**, **Edit –> Copy**, switch the selection display mode to Quickmask (by clicking on the quickmask icon in the

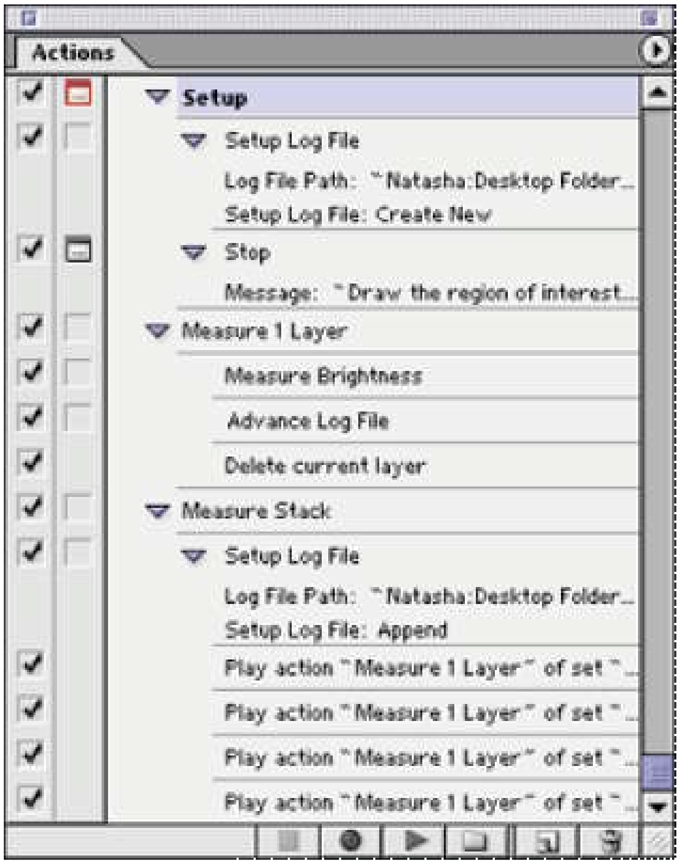
tools palette or using the Q key as a shortcut), **Edit –> Paste**, and switch back to the normal selection mode (click on the standard mode icon in the tools palette or use the Q key as a shortcut). Since one action can be called as a subroutine by another action, you could insert this into the action at this point, but in this example I'm assuming you have probably used the manual outlining method to designate the region to be measured. Be sure that the topmost layer in the image is selected (the name should be highlighted in the actions palette, as shown in the example).



Normal
Opacity: 100%
Lock:
Fill: 100%

!

The Layers palette showing the images"



Setup
Setup Log File
Log File Path: "Natasha:Desktop Folder...
Setup Log File: Create New
Stop
Message: "Draw the region of interest...
Measure 1 Layer
Measure Brightness
Advance Log File
Delete current layer
Measure Stack
Setup Log File
Log File Path: "Natasha:Desktop Folder...
Setup Log File: Append
Play action "Measure 1 Layer" of set "...
Play action "Measure 1 Layer" of set "...
Play action "Measure 1 Layer" of set "...
Play action "Measure 1 Layer" of set "...

The completed actions

6.! Stop recording the first action, and start a second, which we can call "Measure Stack". This one does the actual work.

7.! Tell the system to write data to the log file (**IP•Utilities –> Setup Log File –> Append to Log File**).

8.! Use **IP•Measure Global –> Brightness** to get the mean luminance from the region of interest.

9.! Place a carriage return in the log file (**IP•Utilities –> Log New Line**).

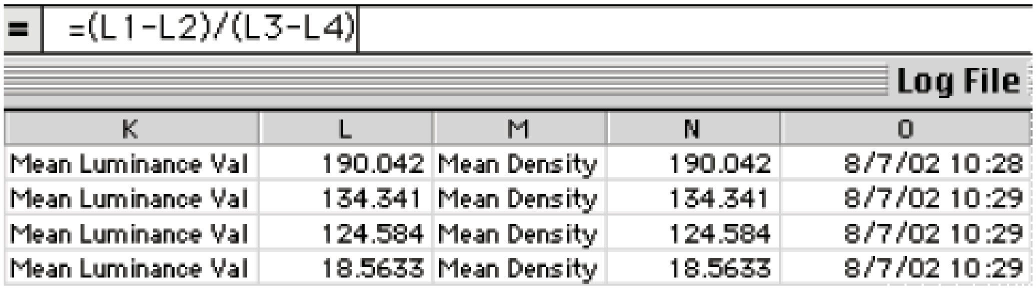
10. Delete the current layer in the image (**Layer –> Delete Layer**). This brings the next layer in the stack to the top and selects it as the active layer.

11. Steps 8, 9 and 10 must be repeated for as many images as there are in the stack. One way to simplify this (as shown in the example) is to make steps 8, 9 and 10 into an action, and then call this action as a subroutine as many times as there are layers. When you try to do this for the background layer you will be advised that you can't delete the background, but this happens at the very end of the process and does not affect the data.

12. Stop recording the action.

13. When the action is run on an image it will create a text log file that looks like the one below. It includes the red, green and blue values as well as the mean luminance and, if you have calibrated the density scale, the mean density.

14. This file can be opened in Excel (or another spreadsheet program). The ratio calculation might then have the form shown, which is (A–Background)/(B–Background) to calculate an intensity ratio with background subtraction. Obviously you can modify this according to the specific experiment you are performing.



= =(L1-L2)/(L3-L4)

K	L	M	N	O
Mean Luminance Val	190.042	Mean Density	190.042	8/7/02 10:28
Mean Luminance Val	134.341	Mean Density	134.341	8/7/02 10:29
Mean Luminance Val	124.584	Mean Density	124.584	8/7/02 10:29
Mean Luminance Val	18.5633	Mean Density	18.5633	8/7/02 10:29

Several example actions are provided on the Fovea Pro disk, and others are available on the ReindeerGraphics.com website.

7.B. Batch Processing

Actions can also be used to batch process images. In most cases these images will have been collected beforehand and saved in disk files, all placed within a single folder. It is possible in principle to include the acquisition in the batch processing but most acquisition plug-ins (e.g. for cameras or slide scanners) do not at this time support fully automatic operation, and Photoshop is not a convenient platform from which to automate hardware such as microscope stage controllers. However, since it is possible to completely control Photoshop, including actions and plug-ins, from Visual Basic or from Java programming, full automation of instrumentation is certainly possible (see Section 7.C).

Select **File –> Automate –> Batch** to bring up the control dialog. The Set and Action refer to the action script that you have created. There can be multiple sets of actions loaded from disk, and multiple actions within each set. Since an action can call other actions as subroutines, as shown above, you can construct the final processing action by combining a number of operations that have been created previously. A typical action will perform whatever image processing is required, and then append feature measurement data to a disk file for later analysis. It is important to remember to append the data, or each measurement will overwrite the existing file. The action will usually be set to not display any dialogs for either input or output so that the batch process can run without attention.

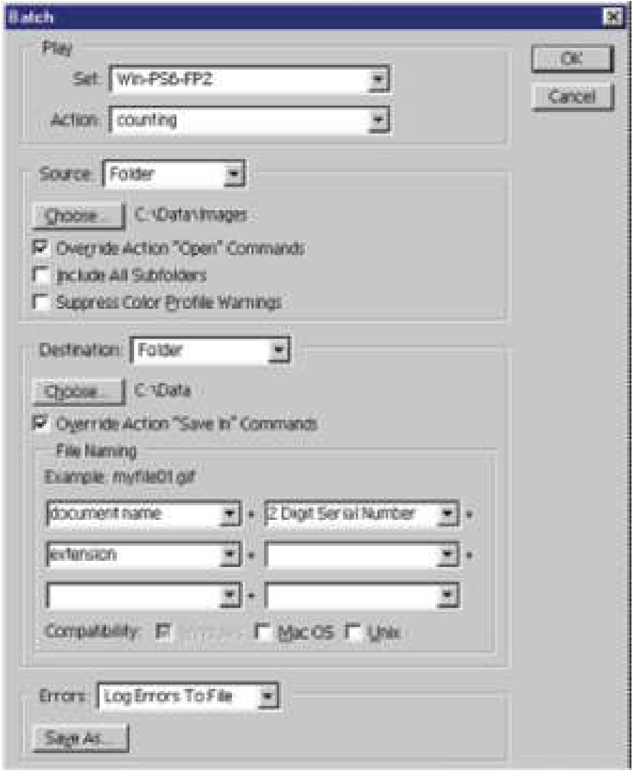
The Source folder should contain all of the images to be processed. Checking “Override Action ‘Open’ Commands” tells the action to ignore any Open commands recorded in the action, so that all of the images are loaded from the selected folder (in alphabetical order). The same logic applies to the Destination selection. If the processed versions of the images are to be saved, you can create a folder to receive them and direct the batch process to save them there. You can also design the format for the names of the saved image files.

If your action does not save the image results (for example if you are performing measurements and writing the measurement data to a disk file, and then closing the images) you do not need to specify any destination information. Saving data (usually appending it) to files from the Fovea plug-ins uses the file name destination set in the action itself. Logging, as described above, will accumulate data from various measurement routines. You can choose either to have the batch process halt if an error occurs or to log the errors to a file and continue if possible.

In creating actions to be applied to batches of images, a few simple guidelines should be kept in mind. First, it is essential that the end point of an action leaves the same conditions as those expected at the beginning. This may include pen and background color selections, for example, and any open images. If an action is sequentially comparing each image in a folder to the next one (e.g., to implement a disector count through a stack of serial sections), then typically it will begin with one image open, and will open the next one. That means that at the end of the action the original image must be closed and the newly opened one must be reverted to its original condition.

Second, it can be very difficult to write a long and complex action that performs multiple analyses (some of these may contain a hundred or more steps) as a single entity. Since one master action can call other actions as subroutines, it is wise to create each of the individual components of processing and analysis separately, and debug them first. Then the master action can call them to produce the final batch result.

Third, although Photoshop actions will faithfully record your selection of individual images and layers, when you have many images open and switch back and forth between them, it is very easy to become confused and create a sequence while writing the action that cannot be correctly executed later when running it. It is best to close images whenever possible, and instead of keeping multiple intermediate images open, to use the ten image memories provided by Fovea Pro to explicitly store and recall images by number. The **IP•Utilities–>Memory Store N** and **–>Memory Recall N** functions in Fovea Pro (not provided in The Image Processing Tool Kit) are intended for use in constructing actions. They do not have anything to do with the **2nd Image–>Setup** function used for various two-image operations, but simply provide explicit storage locations for images used in actions. (In running batch operations, you cannot store intermediate images or auxiliary images such as grids in disk files because the open commands will be overwritten by the batch procedure to access the designated folder.)



Play
Set: Win-PS6-FP2
Action: counting

Source: Folder
C:\Data\Images
☑ Override Action "Open" Commands
☐ Include All Subfolders
☐ Suppress Color Profile Warnings

Destination: Folder
C:\Data
☑ Override Action "Save In" Commands
File Naming
Example: myfile01.gif
document name + 2 Digit Serial Number +
extension + +
+
Compatibility: ☑ ☐ Mac OS ☐ Unix

Errors: Log Errors To File

7.C. Scripting

Beginning with Photoshop version 8 (Photoshop CS), it is possible to completely control the host program and the plugins using a script written in Java. This is particularly useful if it is desired to write a routine that uses looping or branching (e.g., a decision based on the image contents might be used to select between different processing operations). The Photoshop documentation describes scripting, which is beyond the intent of this tutorial, and also provides a “Listener” plugin that constructs scripts while the program is used normally, which is a good way to learn about them. One script, provided on the Fovea Pro CD, is intended to be used in conjunction with the PCA routines. It is necessary to select all of the image channels before using these plugins. The manual procedure is to shift-click on the name of each channel in the channels dialog. The “IP_selectallchannels.js” script performs this operation automatically.

```
// make all channels visible and selected
if (documents.length > 0)
{
!   var docRef = activeDocument;! !         !         !         //current document
!   var channelCount = docRef.channels.length;!         //number of channels in it
!   var aChannelArray = new Array();!       !         !         //will build a list of channels
!   for (var i = 0; i < channelCount; i++)
!   {
!   !         docRef.channels[i].visible = true; !       !         //make it visible
!   !         aChannelArray[i] = docRef.channels[i];!        //add it to the list
!   }
!   //activate the whole list of channels
!   docRef.activeChannels = aChannelArray;
}
else
{
!   alert("There must be at least one open document to run this script!");
}
```

The IP_SelectAllChannels javascript

Additional scripts provided on the Fovea Pro CD facilitate workflow for multichannel images. One script imports a folder of images as layers, and another will convert layers to channels. Both retain the original image names. There is also a script to automatically pad an image to the next larger power-of-two dimensions to facilitate performing Fourier transform operations.

There are two ways to access a script. One is to use the File->Scripts->Browse selection to locate the file and execute it. The second is to place the *.js file in the Photoshop “Scripts” folder (on the PC this would be located at C:\Program Files\Adobe\Photoshop CS2\Presets\Scripts following a normal installation; on the Mac it would be located at Applications\Adobe Photoshop CS2\Presets\Scripts by default), so that the script is listed by name in the File->Scripts submenu. It is also possible for an Action to call a script.