**Methods and tools for I/O of images**

**Bitmap and vector types of image representation: Bitmap images**

[Bitmap](http://graphicssoft.about.com/library/glossary/bldefbitmap.htm) images (also known as raster images) are made up of [pixels](http://graphicssoft.about.com/library/glossary/bldefpixel.htm) in a grid. Bitmap images are *resolution dependent*. [Resolution](http://graphicssoft.about.com/library/glossary/bldefresolution.htm) refers to the number of pixels in an image and is usually stated as [dpi (dots per inch)](http://graphicssoft.about.com/library/glossary/bldefdpi.htm) or [ppi (pixels per inch)](http://graphicssoft.about.com/library/glossary/bldefppi.htm).

Because bitmaps are resolution dependent, it's difficult to increase or decrease their size without sacrificing a degree of image quality. When you reduce the size of a bitmap image through your software's [resample](http://graphicssoft.about.com/library/glossary/bldefresample.htm) or resize command, you must throw away pixels. When you increase the size of a bitmap image through your software's resample or resize command, the software has to create new pixels. When creating pixels, the software must estimate the color values of the new pixels based on the surrounding pixels. This process is called [*interpolation*](http://graphicssoft.about.com/library/glossary/bldefinterpolation.htm).

**Common** [bitmap formats](http://graphicssoft.about.com/od/graphicformats/index.htm) **include:**
• BMP
• GIF
• JPEG, JPG
• PNG
• PICT (Macintosh)
• PCX
• TIFF
• PSD (Adobe Photoshop)

Bitmap images in general *do not* inherently support transparency.

**Key Points About Bitmap Images:**
• pixels in a grid
• resolution dependent
• resizing reduces quality
• easily converted
• restricted to rectangle
• minimal support for transparency

**Vector images:** are made up of many individual, scalable objects. These objects are defined by mathematical equations rather than pixels, so they always render at the highest quality. Objects may consist of lines, curves, and shapes with editable attributes such as color, fill, and outline. Changing the attributes of a vector object does not affect the object itself.

Because they're scalable, vector-based images are [resolution](http://graphicssoft.about.com/library/glossary/bldefresolution.htm) independent. You can increase and decrease the size of vector images to any degree and your lines will remain crisp and sharp, both on screen and in print. Fonts are a type of vector object.

Another advantage of vector images is that they're not restricted to a rectangular shape like bitmaps. Vector objects can be placed over other objects, and the object below will show through.

Vector images have many advantages, but the primary disadvantage is that they're unsuitable for producing photo-realistic imagery. Vector images are usually made up of solid areas of color or gradients, but they cannot depict the continuous subtle tones of a photograph.

Vector images primarily originate from software. You can't scan an image and save it as a vector file without using special [conversion software](http://graphicssoft.about.com/od/bitmaptovector/index.htm). On the other hand, vector images can, quite easily, be converted to bitmaps. This process is called [*rasterizing*](http://graphicssoft.about.com/library/glossary/bldefrasterize.htm). When you convert a vector image to a bitmap, you can specify the output resolution of the final bitmap for whatever size you need.

**Common** [vector formats](http://graphicssoft.about.com/cs/graphicformats/index.htm) **include:**
• AI (Adobe Illustrator)
• CDR (CorelDRAW)
• CMX (Corel Exchange)
• CGM Computer Graphics Metafile
• DXF AutoCAD
• WMF Windows Metafile

**Key Points About Vector Images**
• scalable
• resolution independent
• no background
• cartoon-like
• inappropriate for photo-realistic images
• metafiles contain both raster and vector data

**Halftone image processing**

**Halftone** is the [reprographic](http://en.wikipedia.org/wiki/Reprographic) technique that simulates [continuous tone](http://en.wikipedia.org/wiki/Continuous_tone) imagery through the use of dots, varying either in size or in spacing.[[1]](http://en.wikipedia.org/wiki/Halftone#cite_note-campbell-0) 'Halftone' can also be used to refer specifically to the image that is produced by this process.

Where continuous tone imagery contains an infinite range of [colors](http://en.wikipedia.org/wiki/Color) or [greys](http://en.wikipedia.org/wiki/Grey), the halftone process reduces visual reproductions to a [binary](http://en.wikipedia.org/wiki/Binary_numeral_system) image that is printed with only one color of ink. This binary reproduction relies on a basic [optical illusion](http://en.wikipedia.org/wiki/Optical_illusion)—that these tiny halftone dots are blended into smooth tones by the human eye.

The resolution of a halftone screen is measured in [lines per inch](http://en.wikipedia.org/wiki/Lines_per_inch) **(lpi)**. This is the number of lines of dots in one inch, measured parallel with the screen's angle. Known as the **screen ruling**, the resolution of a screen is written either with the suffix lpi or a hash mark. E.g. *150lpi* or *150#*.

The higher the pixel resolution of a source file, the greater the detail that can be reproduced. However, such increase also requires a corresponding increase in screen ruling or the output will suffer from [posterization](http://en.wikipedia.org/wiki/Posterization). Therefore file resolution is matched to the output resolution.

When different screens are combined, a number of distracting visual effects can occur, including the edges being overly emphasized, as well as a [moiré pattern](http://en.wikipedia.org/wiki/Moir%C3%A9_pattern). This problem can be reduced by rotating the screens in relation to each other. This **screen angle** is another common measurement used in printing, measured in degrees clockwise from a line running to the left (9 o'clock is zero degrees).

Digital half toning uses a [raster](http://en.wikipedia.org/wiki/Raster_graphics) image or bitmap within which each monochrome picture element or [pixel](http://en.wikipedia.org/wiki/Pixel) may be on or off, ink or no ink. Consequently, to emulate the photographic halftone cell, the digital halftone cell must contain groups of monochrome pixels within the same-sized cell area. The fixed location and size of these monochrome pixels compromises the high frequency/low frequency dichotomy of the photographic halftone method. Clustered multi-pixel dots cannot "grow" incrementally but in jumps of one whole pixel. In addition, the placement of that pixel is slightly off-center. To minimize this compromise, the digital halftone monochrome pixels must be quite small, numbering from 600 to 2,540, or more, pixels per inch.

**Image quality enhancement**

The aim of image enhancement is to improve the interpretability or perception of information in images for human viewers, or to provide `better' input for other automated image processing techniques. Image enhancement techniques can be divided into two broad categories:

* Spatial domain methods, which operate directly on pixels, and
* Frequency domain methods, which operate on the Fourier transform of an image.

**Spatial domain methods:** The value of a pixel with coordinates (*x*,*y*) in the enhanced image is the result of performing some operation on the pixels in the neighborhood of (*x*,*y*) in the input image, *F*.

**Gray Scale manipulation:** The simplest form of operation is when the operator *T* only acts on a 1x1 pixel neighborhood in the input image, that is only depends on the value of *F* at (*x*,*y*). This is a grey scale transformation or mapping. The simplest case is thresholding where the intensity profile is replaced by a step function, active at a chosen threshold value. In this case any pixel with a grey level below the threshold in the input image gets mapped to 0 in the output image. Other pixels are mapped to 255.

**Histogram equalization:** Histogram equalization is a common technique for enhancing the appearance of images. Suppose we have an image which is predominantly dark. Then its histogram would be skewed towards the lower end of the grey scale and all the image detail is compressed into the dark end of the histogram. If we could `stretch out' the grey levels at the dark end to produce a more uniformly distributed histogram then the image would become much clearer. Histogram equalization involves finding a grey scale transformation function that creates an output image with a uniform histogram (or nearly so). We must find a transformation *T* that maps grey values *r* in the input image *F* to grey values *s* = *T*(*r*) in the transformed image.

It is assumed that T is single valued and monotonically increasing, and

for .

The inverse transformation from *s* to *r* is given by

*r* = *T-1*(*s*).

From probability theory it turns out that



where *r* = *T-1*(*s*).

Consider the transformation



This is the cumulative distribution function of *r*. Using this definition of *T* we see that the derivative of *s* with respect to *r* is



Substituting this back into the expression for *Ps*, we get



for all .

**Discrete Formulation:** We first need to determine the probability distribution of grey levels in the input image. Now



where *nk* is the number of pixels having grey level k, and *N* is the total number of pixels in the image.

The transformation now becomes



Note that , the index ,and .

The values of *sk* will have to be scaled up by 255 and rounded to the nearest integer so that the output values of this transformation will range from 0 to 255. Thus the discretization and rounding of *sk* to the nearest integer will mean that the transformed image will not have a perfectly uniform histogram.

**Image Smoothing:** The aim of image smoothing is to diminish the effects of camera noise, spurious pixel values, missing pixel values etc.

**Neighborhood averaging:** Each point in the smoothed image, is obtained from the average pixel value in a neighborhood of (*x*,*y*) in the input image.

For example, if we use a 3x3 neighborhood around each pixel we would use the mask:



Each pixel value is multiplied by 1/9, summed, and then the result placed in the output image. This mask is successively moved across the image until every pixel has been covered. That is, the image is convolved with this smoothing mask (also known as a spatial filter or kernel).

Some common weighting functions include the rectangular weighting function above (which just takes the average over the window), a triangular weighting function, or a Gaussian.

There is no big difference between different weighting functions, although Gaussian smoothing is the most commonly used. Gaussian smoothing has the attribute that the frequency components of the image are modified in a smooth manner.

Smoothing reduces or attenuates the higher frequencies in the image. Mask shapes other than the Gaussian can do odd things to the frequency spectrum, but as far as the appearance of the image is concerned we usually don't notice much.

**Edge preserving smoothing:** An alternative approach is to use median filtering. Here we set the grey level to be the median of the pixel values in the neighbourhood of that pixel. The median *m* of a set of values is such that half the values in the set are less than *m* and half are greater The outcome of median filtering is that pixels with outlying values are forced to become more like their neighbours, but at the same time edges are preserved. Of course, median filters are non-linear. Median filtering is in fact a morphological operation. When we erode an image, pixel values are replaced with the smallest value in the neighbourhood. Dilating an image corresponds to replacing pixel values with the largest value in the neighbourhood. Median filtering replaces pixels with the median value in the neighbourhood. It is the rank of the value of the pixel used in the neighbourhood that determines the type of morphological operation.

**Frequency domain methods:** Image enhancement in the frequency domain is straightforward. We simply compute the Fourier transform of the image to be enhanced, multiply the result by a filter (rather than convolve in the spatial domain), and take the inverse transform to produce the enhanced image.

The idea of blurring an image by reducing its high frequency components or sharpening an image by increasing the magnitude of its high frequency components is intuitively easy to understand. However, computationally, it is often more efficient to implement these operations as convolutions by small spatial filters in the spatial domain. Understanding frequency domain concepts is important, and leads to enhancement techniques that might not have been thought of by restricting attention to the spatial domain.

**Filtering:** Low pass filtering involves the elimination of the high frequency components in the image. It results in blurring of the image (and thus a reduction in sharp transitions associated with noise). An ideal low pass filter would retain all the low frequency components, and eliminate all the high frequency components. However, ideal filters suffer from two problems: blurring and ringing. These problems are caused by the shape of the associated spatial domain filter, which has a large number of undulations. Smoother transitions in the frequency domain filter, such as the Butterworth filter, achieve much better results.

**Homomorphic filtering:** Images normally consist of light reflected from objects. The basic nature of the image *F*(*x*,*y*) may be characterized by two components: (1) the amount of source light incident on the scene being viewed, and (2) the amount of light reflected by the objects in the scene. These portions of light are called the illumination and reflectance components, and are denoted *i*(*x*,*y*) and *r*(*x*,*y*) respectively. The functions *i* and *r* combine multiplicatively to give the image function *F*: ***F*(*x*,*y*) = *i*(*x*,*y*)*r*(*x*,*y*),** where and 0 < *r*(*x*,*y*) < 1.

We cannot easily use the above product to operate separately on the frequency components of illumination and reflection because the Fourier transform of the product of two functions is not separable:

****

Suppose, however, that we define



Then



or



where *Z*, *I* and *R* are the Fourier transforms of **z, ln*i*** and respectively. The function *Z* represents the Fourier transform of the *sum* of two images: a low frequency illumination image and a high frequency reflectance image.

If we now apply a filter with a transfer function that suppresses low frequency components and enhances high frequency components, then we can suppress the illumination component and enhance the reflectance component. Thus



where *S* is the Fourier transform of the result. In the spatial domain



By letting



and



we get

*s*(*x*,*y*) = *i*'(*x*,*y*) + *r*'(*x*,*y*).

Finally, as *z* was obtained by taking the logarithm of the original image *F*, the inverse yields the desired enhanced image : that is



**Noise Suppression:** The techniques available to suppress noise can be divided into those techniques that are based on temporal information and those that are based on spatial information. By temporal information we mean that a sequence of images {*ap*[*m*,*n*] | *p*=1,2,...,*P*} are available that contain *exactly* the same objects and that differ only in the sense of independent noise realizations. If this is the case and if the noise is additive, then simple averaging of the sequence:

*Temporal averaging* - 

will produce a result where the mean value of each pixel will be unchanged. For each pixel, however, the standard deviation will decrease from to .

If temporal averaging is not possible, then spatial averaging can be used to decrease the noise. This generally occurs, however, at a cost to image sharpness.

Within the class of linear filters, the optimal filter for restoration in the presence of noise is given by the *Wiener* *filter* . The word "optimal" is used here in the sense of minimum mean-square error (*mse*). Because the square root operation is monotonic increasing, the optimal filter also minimizes the root mean-square error (*rms*). The Wiener filter is characterized in the Fourier domain and for additive noise that is independent of the signal it is given by:



where *Saa*(*u*,*v*) is the power spectral density of an ensemble of random images {*a*[*m*,*n*]} and *Snn*(*u*,*v*) is the power spectral density of the random noise. If we have a single image then *Saa*(*u*,*v*) = |*A*(*u*,*v*)|2. In practice it is unlikely that the power spectral density of the uncontaminated image will be available. Because many images have a similar power spectral density that can be modeled by Table 4-T.8, that model can be used as an estimate of *Saa*(*u*,*v*).