GRAYSCALE MORPHOLOGY

If you are not familiar with *binary* morphology, you may want to read about it first.

As mentioned in the section on binary morphology, grayscale morphology is simply a generalization from 1 bpp (binary) images to images with multiple bits/pixel, where the *Max* and *Min* operations are used in place of the OR and AND operations, respectively, of binary morphology. The following should be noted:

- These operations, as with binary morphology, are nonlinear. The significance of nonlinear operations, as opposed to linear operations, is that *nonlinear operations can be used directly for making decisions about regions of the image*. Nonlinear operations are therefore of particular use in *image analysis*.
- The generalization can be seen in reverse: binary morphology is a specialization of grayscale morphology. The Max of a set of values {0, 1} is equivalent to an OR, and the Min is equivalent to an AND.
- The generalization only applies to *flat* structuring elements. With grayscale images, dilation and erosion can be defined with non-uniform structuring elements. These are rarely used, and will not be discussed further.

- With flat structuring elements, grayscale morphology is itself a specialization of *rank-order filters*. When a rank-order filter acts on a source image, it generates a dest image where each pixel is computed as follows: place the structuring element with its origin on the corresponding source pixel, and choose the rank (ordered) pixel in the set of source pixels that is covered by the structuring element. Erosion is thus a rank-order filter with rank = 0.0; dilation has rank = 1.0; and the median filter, which is useful for removing noise, has rank 0.5. (To be completely accurate, when doing a dilation as described here, invert the structuring element about its origin.)
- Because the standard photometry of binary and grayscale images is opposite (white is min val in binary and max val in grayscale), grayscale dilation lightens a grayscale image and erosion darkens it, visually.
- Without special hardware registers that do Min and Max comparisons on 4 or 8 bytes simultaneously, grayscale morphology must be done by comparing integers, one pixel at a time. It is thus much more than 8 times slower than implementations of binary morphology that pack 32 pixels into each 32-bit word.
- This inefficiency is somewhat offset by the existence of a simple algorithm that computes grayscale morphological operations in a time *independent of the size of the structuring element*.

For the general case of an arbitrary SE of size *A*, where *A*, the "area" of the SE, is the number of hits, the computation of the Max or Min requires *A* pixel comparisons on the src image for each pixel of the dest. However, things get more efficient when the SE is a *brick*, by which we mean it is a solid rectangle of hits. For a brick SE, dilation (or erosion) can be decomposed into a sequence of 2 dilations (or erosions) on 1-D horizontal and vertical SEs. Each 1-D grayscale morphological operation can be done with less comparisons than the "brute-force" method. Luc Vincent described one such method, that uses an ordered queue of pixels, given by the set of pixels currently under the SE. The queue is ordered so that the Min or Max can be read off from the end. When the SE is moved one position, one pixel is removed from the queue, and a new pixel is placed into it, in the proper location. Most of the work goes into sorting the queue to maintain the order.

More recently, van Herk, Gil and Werman have described an algorithm that uses a maximum of 3 comparisons for a linear SE, regardless of the size of the SE. This method is described below, and it is the method we have implemented here.

Some useful image transforms are composed of a sequence of grayscale operations. For example, the *tophat* operator is often used to find local maxima as seeds for some filling operation.

It is defined as the *difference between an image and the opening of the image by a structuring element of given size*. Another local maximum finder is called the *h-dome* operator, and it uses a grayscale reconstruction method for seed filling. (Yes, with h-domes, you do a seed-filling operation to find seeds for another operation!).

Source: http://www.leptonica.com/grayscale-morphology.html