# BINARY RANK-ORDER AND MEDIAN FILTER USING AN ACCUMULATOR

A median filter is a nonlinear filter that selects, for every position of the filter, the median value of the source pixels under the filter. When applied to grayscale images, it is necessary to order the selected pixels by intensity to find the median. The rank-order filter selects the pixel of specified *rank* in the set. A median filter is a rank-order filter where the rank is *0.5* -- the 50th percentile. A morphological dilation is a rank-order filter where the *Max* is selected, and a morphological erosion is a rank-order filter where the *Min* is selected. For grayscale images, dilations tend to lighten the image and erosions tend to darken it, just the opposite of operations on binary images. (The difference results from the opposing conventions of light and dark pixels in binary and grayscale images.)

When applied to binary images, the rank-order filter takes a sum of pixel values followed by a threshold. It is asking if the number of ON pixels under the filter exceeds a given fraction of the total pixels under the filter. A median filter is a special case that gives an ON pixel if at least half of the pixels are ON. Median pixels are very useful for eliminating some types of noise.

The rank-order filter is much less expensive on binary images because it is not necessary to order the pixels by value -- you just take a sum and apply a threshold. The rank threshold *r* is the fraction, between *0.0* and *1.0*, of the pixels that are required to be ON.

Grayscale convolution with a flat filter also takes a sum, and we have seen that the accumulator allows you to take a sum over any rectangle of pixels very quickly. So the same accumulator can be used *to apply a rank-order filter to binary images*! For a rectangular filter of width *w* and height *h*, and using a rank *r*, the sum of ON pixels is thresholded by *rwh*.

We provide a bit more than this. The function pixBlocksum() takes a 1 bpp image and generates an 8 bpp image, using a rectangular convolution filter. It sums the ON pixels under the rectangular filter at each location, and normalizes to between 0 (all pixels OFF) and 255 (all pixels ON), taking into account the number of pixels under the filter at each location.

The rank-order filter uses this function. It first calls pixBlocksum() to compute the intermediate block sum image, and then thresholds it to generate the binary rank-order dest image.

For the high-level interface, we provide a function that generates the accumulator image, and both the convolution and rank-order functions can (re)use the accumulator. If you are just running the convolution or rank-order filter once, you can have the accumulator generated, used and destroyed by using NULL as input for the accumulator.

Source: http://www.leptonica.com/convolution.html