ROTATION BY SAMPLING AND SHEAR

The simplest implementation of rotation, rotation by sampling (RSamp), chooses the dest pixel to have the value of the closest corresponding source pixel. The implementation marches through the dest -- if it marched through the src, pixels in the dest would be missed. For 1 bpp images, where RAM is not an option, one must choose between RSamp and RShear. The result of these two operations is similar, but for large rotation angles (say, greater than 10 degrees) the result from RSamp is preferable. For depth greater than 1 bpp, RSamp gives inferior results to RAM, but RSamp or RShear should be used if you don't need the interpolation quality of RAM because they are much faster.

Rotation by Sher

A good reference to RShear is "A Fast Algorithm for General Raster Rotation" by Alan Paeth in *Graphics Gems*, p. 179, edited by Andrew Glassner, published by Academic Press, 1990.

The mathematics of the rotation of a continuous image (i.e., an image with infinitesimally small pixels) using horizontal and vertical shears is simple.

A horizontal shear moves a row of image pixels a horizontal distance that is proportional to its vertical distance from some reference point. A succession of 3 alternating horizontal and vertical shears can be used to rotate an image perfectly by any angle. For rotation by small angles of 0.05 radians (about 3 degrees) or less, rotation can be performed using only 2 shears. Expressing the rotation angle in radians, this 2-shear rotation induces a distortion in the rotated image that is a change in the length to width ratio of the amount:

$$\text{delta(length/width)} = 0.5 * \text{angle}^2$$

For an angle of 0.05 radians, this is about 1 part in a thousand, which is typically not something you'd worry about. For larger angles, 3 shears are required. The details of the shear angles required for 2 and 3 shear rotation are given in rotateshear.c.

In-place rotation is performed by in-place shears. For a horizontal in-place shear, each pixel row is translated to the left or right by some number of pixels. Typically, several rows are translated together, and *the pixels that are not over-written are filled in*. A similar up or down translation of pixel columns is used for vertical in-place shears. For shear, we give two choices for the color of the filling pixels: white and black. (Henry Ford only offered black.)

The "jaggies" at sharp boundaries are visible because images are composed of pixels of finite size, arranged on a square lattice. They arise because the two and three shear rotations are nearest-pixel approximations to continuous rotations.

An interesting result is obtained using a sequence of in-place rotations with the same angle. The distortions introduced due to the finite pixel size accumulate, resulting in a nearly random pixel diffusion if repeated many times. For example, suppose you rotate an image six full revolutions (2160 degrees) using 180 rotations of 12 degrees each. The result is an image that is "encrypted" because of the pixel diffusion; everything is blurred and a text image is not readable. However, if you then *unwind* the image with 180 rotations of 12 degrees in the opposite direction, you obtain the original image. Well, not quite. You get a part of the image -- a circular subimage that is equal in size to the largest inscribed circle, with the circle center at the center of rotation, that fits within the original boundaries of the image. You can try this yourself, using prog/rotatetest1.c with the above parameters on the image test24.jpg. See what you get. You can then recover the central part of the original image by unscrambling the "encrypted" image using an angle of -12.0 degrees. If you want to regain the entire image, it is necessary before "encrypting" to first embed the original in a larger rectangle that would contain the full rotated image at every angle it is rotated through.

Because the underlying operation is rasterops, RShear works on images of all depths, with or without color mapping. Speed plus flexibility, plus the ability to rotate in-place, make RShear of ubiquitous importance. The number of pixels/sec that can be rotated by RShear is inversely proportional to the pixel depth. RShear rotates bits at a rate of about 1 bit in 3 machine cycles. On a 3 GHz machine, a 1 bpp image is thus rotated at about 1 billion pixels/sec, and an 8 bpp image is rotated at about 120 million pixels/sec.

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