Subject: Re: IDL User's Library routine ROT
Posted by thompson on Thu, 13 Aug 1992 14:29:00 GMT
View Forum Message <> Reply to Message

In article <12AUG199218251536@stars.gsfc.nasa.gov>, fireman@stars.gsfc.nasa.gov (Gwyn Fireman) writes...
> Hello, IDL users -
>
> I've been trying to figure out the IDL User's Library routine ROT, and
> am now pretty confused. It seems to me that a 90-degree rotation should have
> the same result for both ROT and the built-in function ROTATE, at least for
> square matrices, and that interpolation shuld have no effect. This is not the
> case. Even or odd array sizes make no difference. Am I missing something
> obvious? Is there a better routine for doing array rotations in non-multiples
> of 90 degrees?

(rest of message deleted)

The problem seems to be one of integer truncation. ROT uses "nearest neighbor" interpolation. In other words, it tries to find the nearest pixel to where it calculates the rotated pixel to be. When the rotation angle is 90 degrees, that should give an exact solution. However, roundoff errors in the calculation may calculate the nearest pixel as, say, 4.0001,5.9999 when it should be 4,6. The 4.0001 is no problem since it be truncated correctly to 4. The 5.9999, however, will be truncated incorrectly to 5. Thus, there will be one pixel shifts in various parts of the image.

Maybe ROT could be rewritten to avoid these integer truncations. Another work around is to use ROT_INT instead of ROT. Since this does bilinear interpolation, it's not sensitive to this integer truncation problem. It also gives a better looking rotated picture. There may be slight changes in the values, particularly for integer arrays (a different truncation effect). There are also some funny effects at the corners, which is unavoidable.

Bill Thompson