Hi All,

This was an interesting problem - I certainly hadn't noticed it before. The reason for the behaviour is precision error in the arithmatic which works out the poly2d coefficents. It can be corrected effectively by modifying line 128 of rot.pro:

from:

 theta = -angle/!radeg ;angle in degrees CLOCKWISE.

to:

 theta = (-angle MOD 360) *acos(0.0d)/90 ;angle in degrees CLOCKWISE. (mod MRD 21/11/2001 to correct for precision error)

This does two things, firstly (-angle MOD 360) ensures that a precision error does not propagate due to large angles which contain multiple 360 degree rotations,
for instance that 390.45 degree rotation is treated exactly the same as 30.45 degrees [i.e. n*360+theta = = theta].

Secondly, substituting (acos(0.0d)/90) for !radeg gives a full DOUBLE precision representation of theta in radians.

This fixes it completely as far as I can see:
IDL> a = findgen(5,5)
IDL> for deg = -720, 720,90 do print, deg, total(rot(a, deg))

-720 300.000
-630 300.000
-540 300.000
-450 300.000
-360 300.000
-270 300.000
-180 300.000
-90 300.000
0 300.000
90 300.000
180 300.000
270 300.000
360 300.000
450 300.000
540 300.000

630 300.000
720 300.000

compared this to previous output:
IDL> for deg = -720, 720,90 do print, deg, total(rot(a, deg))

-720 252.000
-630 250.000
-540 300.000
-450 273.000
-360 237.000
-270 290.000
-180 216.000
-90 244.000
0 300.000
90 222.000
180 221.000
270 300.000
360 247.000
450 249.000
540 300.000
630 251.000
720 242.000

Quite how RSI had left the code like that for so long who knows.....(but if
they want to send me a copy of David's 2nd Ed. that would be nice!)

cheers

Martin

Martin Downing,
Clinical Research Physicist,
Grampian Orthopaedic RSA Research Centre,
Woodend Hospital, Aberdeen, AB15 6LS.
Tel. 01224 556055 / 07903901612
Fax. 01224 556662

m.downing@abdn.ac.uk

"Bhautik Jitendra Joshi" <bjoshi@cse.unsw.EDU.AU> wrote in message
news:Pine.GSO.4.21.0111211537260.24363-100000@haydn.orchestra.cse.unsw.EDU.A
U...
> Hi all,
>
> The question I put to you all today is this: is ROT completely and utterly
> broken?

>
> Lets take a nice and normal 5x5 float array:
>
> MOO>a=findgen(5,5) & print, a
>   0.00000   1.00000   2.00000   3.00000   4.00000
>   5.00000   6.00000   7.00000   8.00000   9.00000
>   10.0000   11.0000   12.0000   13.0000   14.0000
>   15.0000   16.0000   17.0000   18.0000   19.0000
>   20.0000   21.0000   22.0000   23.0000   24.0000
>
> Now, lets do a quick checksum:
>
> MOO>print, total(a)
>   300.000
>
> So any 90 degree rotations we perform should maintain this. Lets try it
> out:
>
> MOO>print, total(rot(a,90))
>   296.000
>
> OMG! *world in crisis* How to fix? Use interpolation.
>
> MOO>print, total(rot(a,90,/INTERP))
>   300.000
>
> *phew* Lets do a clockwise rotation instead.
>
> MOO>print, total(rot(a,-90,/interp))
>   300.000
>
> So, for those who can remember their high school math, -90 degrees is the
> same as a 270 degree rotation.
>
> MOO>print, total(rot(a,270,/interp))
>   290.000
>
> argh! 360 degrees - a complete rotation, no difference, right?
>
> MOO>print, total(rot(a,360,/interp))
>   290.000
>
> Perhaps its the interpolation thats stuffing it up. Lets leave it out.
>
> MOO>print, total(rot(a,360))
>   262.000
>
> *brain melts*

>
> It doesn't make a difference whether you use the interp or cubic keywords,
> nor if you shift it so that the centre of rotation is set to be the corner
> of the pixel rather than the centre of the pixel. If it doesn't work for
> multiples of 90 it certainly is going to have issues with arbitrary
> angles.
>
> ROT is bad. Can it be fixed? Is there a (fast) alternative?
>
> Cheers,
> Bhautik
>
>
>   _____
>   /   \
> |bjoshi@geocities.com | phone: 0404032617 |
> |ICQ #: 2464537 | http://cow.mooh.org |
>   \~~~~~~(__)~~~~~~~/
>   (__) |..|--\
>   /--|^^| moo | |--|
> |--| | \OO/||^
>   ^||\oo/ moo
>
```
```