Subject: Re: ROT is ROTTEN (a solution)
Posted by Martin Downing on Wed, 21 Nov 2001 11:55:49 GMT
View Forum Message <> Reply to Message

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 coefficients. 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
>

Subject: Re: ROT is ROTTEN (a solution)
Posted by Heike Koch-Beuttenmue on Wed, 21 Nov 2001 13:02:16 GMT
View Forum Message <> Reply to Message

Martin Downing wrote:
>
> 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 coefficients. 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
>
Though there are some differences between the rot of pvwave and idl the same correction helps pvwave to get the right result for rot (with interp):

uncorrected: print, total(e1)
     282.000
corrected:
print, total(e2)
     300.000

Best regards

Heike Koch-Beuttenmï¿½ller

Subject: Re: ROT is ROTTEN (a solution)
Posted by Paul van Delst on Wed, 21 Nov 2001 15:19:56 GMT
View Forum Message <> Reply to Message

Martin Downing wrote:
>
> 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 coefficients. 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:

Great job!

paulv

--
Paul van Delst Religious and cultural
CIMSS @ NOAA/NCEP purity is a fundamentalist
Ph: (301)763-8000 x7274 fantasy
Fax:(301)763-8545 V.S.Naipaul

Subject: Re: ROT is ROTTEN (a solution)
Posted by thompson on Wed, 21 Nov 2001 17:34:44 GMT
View Forum Message <> Reply to Message

"Martin Downing" <martin.downing@ntlworld.com> writes:

> 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 coefficients. 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)

As others have said, great job! Can I make one small suggestion, though.
Instead of acos(0.0d)/90, can I suggest !dpi/180?

 theta = (-angle MOD 360) * !dpi/180

William Thompson

> 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
>>
```

Subject: Re: ROT is ROTTEN (a solution)
Posted by Wayne Landsman on Wed, 21 Nov 2001 18:41:39 GMT

View Forum Message <> Reply to Message

Martin Downing wrote:

> 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)
>

That's neat how the double precision improves things.   But I 'd still
emphasize that if you are rotating by a multiple of 90 degrees then you should
be using ROTATE() and not ROT() for two reasons:

(1) ROTATE() is much faster (almost a factor of 4 on my Solaris machine)
(2) Using ROTATE() will ensure that you have the exactly correct numbers in the
output array (since it simply moves elements within the array and performs no
arithmetic operations).   The improved ROT() is much better but it is not
perfect.   For example

```
{ sparc sunos unix 5.3 Nov 11 1999}
IDL>a = dist(2048)
IDL>print,total(a)
  3.28828e+09
IDL>print,total(rot(a,90))     ;use improved ROT with double precision !RADEG
```

  3.28830e+09
IDL>print,total(rotate(a,1))
  3.28828e+09

So possibly one could add to the beginning of ROT() something like:

theta = angle mod 90
if theta EQ 0 then return, rotate(a, theta/90)

although one needs to also worry if the user has also set the magnification or
pivot keywords

--Wayne Landsman landsman@mpb.gsfc.nasa.gov

Subject: Re: ROT is ROTTEN (a solution)
Posted by Martin Downing on Wed, 21 Nov 2001 20:59:05 GMT
View Forum Message <> Reply to Message

"William Thompson" <thompson@orpheus.nascom.nasa.gov> wrote in message
news:9tgojk$g7t$1@skates.gsfc.nasa.gov...
> "Martin Downing" <martin.downing@ntlworld.com> writes:
>
>> 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 coefficients. 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)
>
> As others have said, great job! Can I make one small suggestion, though.
> Instead of acos(0.0d)/90, can I suggest !dpi/180?
>
> theta = (-angle MOD 360) * !dpi/180
>

Sure, it just seemed kind of fun the other way!

Martin

Subject: Re: ROT is ROTTEN (a solution)
Posted by Martin Downing on Wed, 21 Nov 2001 21:06:52 GMT
View Forum Message <> Reply to Message

Wayne,
Sure, if you are doing 90-degree rotations, call ROTATE. Whether there should be a wrapper to do it for you I don't know - I think you are better being aware of what is going on. This was an issue of general rotations which was just well illustrated by 90 degree examples.

Regarding the accuracy issue, I can not reproduce your result:
IDL> print, !version

{ x86 Win32 Windows Microsoft Windows 5.5 Beta Jun 20 2001 32 64}

IDL> a = dist(2048)

IDL> print,total(rot(a,90)) - total(a)

0.000000

Martin

"Wayne Landsman" <landsman@mpb.gsfc.nasa.gov> wrote in message
news:3BFBF563.2E32668E@mpb.gsfc.nasa.gov...
> Martin Downing wrote:
>
>> 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)
>>
>
> That's neat how the double precision improves things.   But I 'd still
> emphasize that if you are rotating by a multiple of 90 degrees then you
should

> be using ROTATE() and not ROT() for two reasons:
>
> (1) ROTATE() is much faster (almost a factor of 4 on my Solaris machine)
> (2) Using ROTATE() will ensure that you have the exactly correct numbers
in the
> output array (since it simply moves elements within the array and performs
no
> arithmetic operations). The improved ROT() is much better but it is not
> perfect. For example
>
> { sparc sunos unix 5.3 Nov 11 1999}
> IDL>a = dist(2048)
> IDL>print,total(a)
> 3.28828e+09
> IDL>print,total(rot(a,90)) ;use improved ROT with double precision
!RADEG
> 3.28830e+09
> IDL>print,total(rotate(a,1))
> 3.28828e+09
>
> So possibly one could add to the beginning of ROT() something like:
>
> theta = angle mod 90
> if theta EQ 0 then return, rotate(a, theta/90)
>
> although one needs to also worry if the user has also set the
magnification or
> pivot keywords
>
>
> --Wayne Landsman landsman@mpb.gsfc.nasa.gov
>

Subject: Re: ROT is ROTTEN (a solution)
Posted by Wayne Landsman on Thu, 22 Nov 2001 04:55:17 GMT

View Forum Message <> Reply to Message

Martin Downing wrote:

> Regarding the accuracy issue, I can not reproduce your result:
> IDL> a = dist(2048)
> IDL> print,total(rot(a,90)) - total(a)

0.000000

I find that when I add the /DOUBLE keyword to TOTAL()I reproduce your results,
so it does seem like the new rot.pro does produce exact numerical results.

Alternatively,

```
{ sparc sunos unix 5.4.1 Jan 16 2001      64      64}
IDL> a = dist(2048)
IDL> print,array_equal( rotate(a,1), rot(a,-90) )
1
```

(I had also forgotten that ROTATE() is positive counterclockwise and ROT() is positive clockwise.

> This was an issue of general rotations which was just well illustrated by 90 degree examples.

I could be wrong again ;-), but I think that the original ROT had no problems except at exact multiples of 90 degrees. It doesn't matter if a rotation angle of 23 degrees loses precision and becomes 23.00001 degrees, but it does matter if 90 degrees becomes 90.00001 degrees. At 90 degrees the new image exactly overlaps the old image, and there is no need to extrapolate, but at 90.000001 degrees there will be subpixels on the edge of the new image that do not overlay the old (and which will be flagged as missing data).

Wayne Landsman landsman@mpb.gsfc.nasa.gov

```
b= findgen(3,3)
IDL>print,rot(b,90)
     2.00000      5.00000      8.00000
     1.00000      4.00000      7.00000
     0.00000      3.00000      6.00000
IDL> print,rot(b,90.0001)
     2.00000      5.00000      7.00000
     1.00000      4.00000      7.00000
     0.00000      0.00000      3.00000
```