Subject: POLY_2D proken. Film at 11.
Posted by Craig DeForest on Fri, 21 Aug 1998 07:00:00 GMT
View Forum Message <> Reply to Message

I found a rather interesting bug in poly_2d, the IDL built-in to do scaling of image data. The bilinear and spline interpolation features are designed inconsistently with the sampling feature. The bug is both in 4.x and 5.x versions of IDL.

Sampling works correctly: when scaling an original image by an integer factor, each pixel is scaled an integer number of times. But bilinear and cubic interpolation do not work the same way -- there is a 1/2-pixel offset in the output compared to linear sampling. Apparently, the interpolation algorithms wrongly regard each (old) pixel's value as resident at the *corner* of the (old) pixel, and not at the *center* of the (old) pixel.

Here's some example code:

```
 ------------------------------------------------------------ ----------
pro break_poly_2d

; Generate a symmetrical image of a crosshairs
a = bytarr(9,9)
a(4,*) = 255
a(*,4) = 255
window,0,xsiz=9,ysiz=9
tv,a

; Scale it up by a factor of 10 using the sampling algorithm
; The output looks nice so far...
b = poly_2d(a,[0,0.1,0,0],[0,0,0.1,0],0,90,90)
window,1,xsiz=90,ysiz=90
tv,b

; Scale it up by a factor of 10 using the bilinear interpolation
; algorithm.  Shudder at the lack of consistency.
c = poly_2d(a,[0,0.1,0,0],[0,0,0.1,0],1,90,90)
window,2,xsiz=90,ysiz=90
tv,c

; Scale it up by a factor of 10 using the bilinear interpolation
; algorithm, but offset to account for the pixel-corner bug.
; Recoil in horror at the sloppy treatment of the boundary condition.
d = poly_2d(a,[-0.5,0.1,0,0],[-0.5,0,0.1,0],1,90,90)
window,3,xsiz=90,ysiz=90
tv,d

; Scale it up by a factor of 10 using the cubic spline.
; Laugh that at least it's broken consistently with the
; bilinear case.
e = poly_2d(a,[-0.5,0.1,0,0],[-0.5,0,0.1,0],2,90,90)
window,4,xsiz=90,ysiz=90
tv,d

end
```
-- ----------

The best one can do is to say something inane like:

```
 P1=P
 P1(0) = P1(0)-0.5*keyword_set(method)
 Q1=Q
 Q1(0) = Q1(0)-0.5(keyword_set(method)
 out = poly_2d(in,P1,Q1,method,xsize,ysize)
```

instead of

```
 out = poly_2d(in,P,Q,method,xsize,ysize)
```

but even then you get wacky results near the lower and left hand boundaries of <out>.

--
I work for Stanford, *NOT* the government. My opinions are my own.

If you're a robot, please reply to the address in the header.
If you're human, try " zowie (at) urania . nascom . nasa . gov "

Subject: Re: poly_2d
Posted by davidf on Wed, 04 Apr 2001 13:11:56 GMT
View Forum Message <> Reply to Message

pfis@mytec.com (pfis@mytec.com) writes:

> I am using poly_2d to warp an image. I am using 3 parameters for each
> coordinate (e.g. xnew=a+bx+cy and similar for ynew). My problem is I am
> having trouble choosing 'a' such that the center of the image (actually pixel
> [64,64] of a 128x128 image) does not move. Using a=fix(-64.*(b+c-1.)) keeps
> the center stationary to about 1 pixel which is not good enough. I wrote my
> own version of the warping program which does what I want but is slower than
> poly_2d. Any help would be appreciated.

I'm wondering if ROUNDing the final result wouldn't give you better results. (Or CEILing, FLOORing, etc.)

Cheers,

David
--
David Fanning, Ph.D.
Fanning Software Consulting
Phone: 970-221-0438 E-Mail: davidf@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Toll-Free IDL Book Orders: 1-888-461-0155