Subject: Re: POLY_2D inconsitent interpolation
Posted by Tom S. on Tue, 08 Aug 2006 15:07:57 GMT
View Forum Message <> Reply to Message

Very odd. However, I think the problem is due to the algorithm
accessing image indices that are out of bounds. Ordinarily the
algorithm extrapolates values for the out-of-bounds pixels, but perhaps
this extrapolation is causing the undesired results.

One can remedy the problem (at least with your example) by specifying
MISSING=0. This means that missing array values will all have a value
of zero. This ends up removing the discontinuities.

Regards,
Tom

Randolf Klein wrote:
> Hi,
>
> I found a strange behavior of POLY_2D. The resulting images are shifted
> by 1/2 pixel of the original image when using nearest neighbor or some
> interpolation method. Searching the web for this issue, I found the
> following old post, but it had no replies. The code from this post
> demonstrates this strange behavior very good still in IDL version 6.3
> (except that I do not see any difference any more between the bilinear
> and the cubic spline). Please, comment if this is a feature or a bug and
> may be someone can suggest workarounds especially for hastrom (from the
> astro library) where poly_2d is used.
>
> Thanks
> RK
>
>
> ----------here the mentioned old post's url----------
> http://groups.google.com/group/comp.lang.idl-pvwave/browse_t
hread/thread/c780ba42980c6a04/6dc30561bbaeb17b?lnk=gst&q
=poly_2d&rnum=1#6dc30561bbaeb17b
> --------and here is the post itself----------
>
> From: Craig DeForest
> Date: Fri, Aug 21 1998 12:00 am
> Email: Craig DeForest <junkmail-...@urania.nascom.nasa.gov>
> Groups: comp.lang.idl-pvwave
>
>
> 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 "

---