Subject: Re: IDL ROI Objects
Posted by David B on Wed, 28 Jan 2015 10:00:11 GMT
View Forum Message <> Reply to Message

On Tuesday, 27 January 2015 23:10:07 UTC, Fabien wrote:
> On 27.01.2015 19:53, David Fanning wrote:
>> The ROI code has a left-bottom bias.
>
> Disclaimer: I did not read the original poster code, because it is not
> close enough to a "minimal working example" for my taste, but IDLanROI
> has a pixel centered convention, and a "pixel area convention" (a pixel
> touched is a masked pixel):
>
> IDL> x = [0., 0., 0., 0., 0.]
> IDL> y = x
> IDL> roi = OBJ_NEW('IDLanROI', x, y)
> IDL> print, roi->ComputeMask(DIMENSIONS=[3,3])
> 255 0 0
> 0 0 0
> 0 0 0
> IDL> roi = OBJ_NEW('IDLanROI', x-0.49, y-0.49)
> IDL> print, roi->ComputeMask(DIMENSIONS=[3,3])
> 255 0 0
> 0 0 0
> 0 0 0
> IDL> roi = OBJ_NEW('IDLanROI', x+0.49, y+0.49)
> IDL> print, roi->ComputeMask(DIMENSIONS=[3,3])
> 255 0 0
> 0 0 0
> 0 0 0
> IDL> roi = OBJ_NEW('IDLanROI', x+0.51, y+0.51)
> IDL> print, roi->ComputeMask(DIMENSIONS=[3,3])
> 0 0 0
> 0 255 0
> 0 0 0
>
> If you discovered a bug, can you try to reproduce it in a smaller
> example? Since you seem to mix pixel and polygon a little bit, could
> your problem be related to some floating point precision issues?
>
> Just my 2c, maybe I missed the point.
>
> Fabien

Well it is quite an odd problem. The following script works at the command line for ease:
==========
box = [[18, 22, 18, 22], $
[1, 9, 3, 11], $

```
    [10,      20,       5,      11], $
    [15,      19,       6,      12], $
    [8,      12,      12,      18], $
    [11,      15,       9,      15]]


cgdisplay, 900, 900, /FREE, title = 'Polyimage'
cgplot, box, xrange=[0, 30], yrange=[0,30], /NODATA, aspect = 1.0
cgcolorfill, [box[0,0], box[1,0], box[1,0], box[0,0] ], [box[2,0], box[2,0], box[3,0], box[3,0]], $
color = 'steel blue'
back = cgtransparentimage(MISSING_VALUE=0, TRANS = 50)

TVELLIPSE, 2, 2, 20, 20, 10.0, /DATA

image = DBLARR(30, 30)
szim = Size(image, /Dimensions)

;Okay, this doesnt look interesting, but it will do
image = NOISE_SCATTER(image)

;Open a master mask
m_mask = DBLARR(szim[0], szim[1])
roibox = box

i = 0

mask = OBJ_NEW('IDLanROI', [roibox[0, i], roibox[1, i], roibox[1, i], roibox[0,i], roibox[0, i]], $
     [roibox[2, i], roibox[2, i], roibox[3, i], roibox[3, i], roibox[2, i]]) & $

;Mask must match the image dimensions, I only want the interior too
t_mask = mask -> ComputeMask( DIMENSIONS = [szim[0], szim[1]], $
    PIXEL_CENTER = [0.5, 0.5], MASK_RULE = 1)

m_mask = m_mask + t_mask

cgloadct, 32
cgdisplay, 900, 900, /FREE, title = 'ROIimage'
cgplot, box, xrange = [0, 30], yrange =[0,30], /NODATA, aspect = 1.0
cgimage, BYTSCL(t_mask), /keep, /overplot
cgplot, box, xrange = [0, 30], yrange =[0,30], $
 axiscolor='red', /NOERASE , /NODATA, aspect = 1.0

cgimage, back
 ============================================================
```

So now I have an alpha blended image, and it is obvious that the ROI (pink) is one pixel smaller than the original image. Lets move the centre pixel.

Using the settings of PIXEL_CENTER = [0.0, 0.0] OR [0.5, 0.5] causes no change in where the pink box is relative to the black one, it should remain in the top right. [0.51, 0.51] DOES change the position of the ROI box down to the bottom left, which is great. Except that we now have another blue area towards the top right.

The problem then is, polygon defined in CGCOLORFILL is not the same as the same polygon, defined in the same manner as IDLanROI.

They key:
cgcolorfill, [box[0,0], box[1,0], box[1,0], box[0,0]], [box[2,0], box[2,0], box[3,0], box[3,0]], color = 'steel blue'

does not seem to 'create', for lack of a better word, the same polygon as:
==========

mask = OBJ_NEW('IDLanROI', [roibox[0, i], roibox[1, i], roibox[1, i], roibox[0,i], roibox[0, i]],[roibox[2, i], roibox[2, i], roibox[3, i], roibox[3, i], roibox[2, i]])

==========
So, the reason I created roibox as a duplicate to box, was that to get the same 'mask' created by cgcolorfill, I must not only subtract one pixel using PIXEL_CENTER = [0.51, 0.51]; I must ALSO add one pixel to the X2 and X2 positions.

Perhaps another option would be to forget the pixel centre and modify the MASK_RULE from 1 to 2. This apparently includes the border pixels:
==========

t_mask = mask -> ComputeMask(DIMENSIONS = [szim[0], szim[1]], $
    PIXEL_CENTER = [0.0, 0.0], MASK_RULE = 2)

==========
Sadly this causes the mask not to increase by one pixel, but by two. We can check this by running:

TVELLIPSE, 2, 2, 20, 20, 10.0, /DATA

Which I know matches the underlying blue box, as the blue box is the bounding box and strictly mathematically correct. So now the pink mask created by IDLanROI is too big. My final solution is then to do:

roibox = box
roibox[1, *]++
roibox[3, *]++

So the final code becomes:

==========
box = [[18,   22,   18,   22], $

```
    [1,       9,       3,      11], $
    [10,      20,       5,      11], $
    [15,      19,       6,      12], $
    [8,      12,      12,      18], $
    [11,      15,       9,      15]]


cgdisplay, 900, 900, /FREE, title = 'Polyimage'
cgplot, box, xrange=[0, 30], yrange=[0,30], /NODATA, aspect = 1.0
cgcolorfill, [box[0,0], box[1,0], box[1,0], box[0,0] ], [box[2,0], box[2,0], box[3,0], box[3,0]], $
color = 'steel blue'
back = cgtransparentimage(MISSING_VALUE=0, TRANS = 50)

TVELLIPSE, 2, 2, 20, 20, 10.0, /DATA

image = DBLARR(30, 30)
szim = Size(image, /Dimensions)

;Okay, this doesnt look interesting, but it will do
image = NOISE_SCATTER(image)

;Open a master mask
m_mask = DBLARR(szim[0], szim[1])
roibox = box
roibox[1, *]++
roibox[3, *]++


i = 0

mask = OBJ_NEW('IDLanROI', [roibox[0, i], roibox[1, i], roibox[1, i], roibox[0,i], roibox[0, i]], $
     [roibox[2, i], roibox[2, i], roibox[3, i], roibox[3, i], roibox[2, i]]) & $

;Mask must match the image dimensions, I only want the interior too
t_mask = mask -> ComputeMask( DIMENSIONS = [szim[0], szim[1]], $
    PIXEL_CENTER = [0.51, 0.51], MASK_RULE = 1)

m_mask = m_mask + t_mask

cgloadct, 32
cgdisplay, 900, 900, /FREE, title = 'ROIimage'
cgplot, box, xrange = [0, 30], yrange =[0,30], /NODATA, aspect = 1.0
cgimage, BYTSCL(t_mask), /keep, /overplot
cgplot, box, xrange = [0, 30], yrange =[0,30], $
 axiscolor='red', /NOERASE , /NODATA, aspect = 1.0

cgimage, back
=====================================================
```

So I must not only shift the centre of the pixels in the IDLanROI options, but I must add too my corner points manually in order to reproduce the same 'mask' that is found using CGCOLORFILL.

I do hope that helps a little more. The script works inline, but I am no IDL expert. I was therefore wondering if my solution is valid, or if I am as I mentioned earlier, being very stupid.

So the overall problem is I cannot use the same set of vertices provided in the BOX array to recover the mask that I know is correct without modifying the array.

One may be thinking that my bounding box routine may be in error, but I 'inflate' the bounding box and then use the FIX function to round off to the nearest whole vertex, as half a pixel holds no meaning to my data.

Thanks for taking a look anyhow! I seem to have a talent for breaking IDL functions lately.

David
