Subject: IDL ROI Objects
Posted by David B on Tue, 27 Jan 2015 17:28:13 GMT
View Forum Message <> Reply to Message

SO I am currently creating ROI objects so I can group sets of points.

Suppose I can have a set of box values, with corner locations,

X1	X2	Y1	y2
18	22	18	22
1	9	3	11
10	20	5	11
15	19	6	12
8	12	12	18
11	15	9	15

Now these boxes are simply bounding boxes from some ellipses.

I then enter the following commands:

```
==========================================
cgdisplay, 1000, 800
cgplot, box, xrange=[-10, 25], yrange=[0,25], /NODATA
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'

cgcolorfill, [box[0,1], box[1,1], box[1,1], box[0,1] ], [box[2,1], box[2,1], box[3,1], box[3,1]], $
  color = 'orange'

cgcolorfill, [box[0,2], box[1,2], box[1,2], box[0,2] ], [box[2,2], box[2,2], box[3,2], box[3,2]], $
  color = 'red'

cgcolorfill, [box[0,3], box[1,3], box[1,3], box[0,3] ], [box[2,3], box[2,3], box[3,3], box[3,3]], $
  color = 'purple'
cgcolorfill, [box[0,4], box[1,4], box[1,4], box[0,4] ], [box[2,4], box[2,4], box[3,4], box[3,4]], $
  color = 'sea green'
cgcolorfill, [box[0,5], box[1,5], box[1,5], box[0,5] ], [box[2,5], box[2,5], box[3,5], box[3,5]], $
  color = 'red 2'
```

Followed by:

```
TVELLIPSE, 2, 2, 20, 20, 10.0, /DATA
TVELLIPSE, 3, 4, 5, 7, 35.0, /DATA
TVELLIPSE, 5, 2.5, 15, 8, 10.0, /DATA
TVELLIPSE, 3, 1.5, 17, 9, 85.0, /DATA
TVELLIPSE, 3, 1.5, 10, 15, 85.0, /DATA
TVELLIPSE, 3, 1.5, 13, 12, 67.0, /DATA
cgtext, 20, 20, '0', charsize = 2.0, color = 'white'
```

```
cgtext, 5, 7, '1', charsize = 2.0, color = 'white'
cgtext, 15, 8, '2', charsize = 2.0, color = 'white'
cgtext, 17, 9, '3', charsize = 2.0, color = 'white'
cgtext, 10, 15, '4', charsize = 2.0, color = 'white'
cgtext, 13, 12, '5', charsize = 2.0, color = 'white'
==============================================
```

This is merely an example, I didn't loop yet as I was conducting tests.

And you get a nice display in multiple colours. Some of these boxes overlap, so I tried to create an 'intersection matrix' if you like, of which boxes collide with others. But this is not very efficient.

I then approached the problem using ROIs. I think I can flatted multiple masks into one and then use region labelling to separate the ellipses into groups, according to their bounding box locations.

I discovered that if I ROId the same groups, I get a different graph of the rectangles using the commands:

```
==============================================
image = DBLARR(30, 30)

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

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

Then using:

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

The above commands are crucial apparently, I must add 1 to my pixel values to recover the ROI mask correctly...

```
==============================================
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 -> Setproperty, interior = 0 ;This does bugger and all :(

 ;Mask must match the image dimensions
 t_mask = mask -> ComputeMask( DIMENSIONS = [szim[0], szim[1]], $
     PIXEL_CENTER = [1, 1], MASK_RULE = 1) & $
;Add this onto the mask
```

```
 m_mask = m_mask + t_mask

 cgdisplay, 900, 900
 cgplot, box, xrange = [0, 30], yrange =[0,30], /NODATA, aspect = 1.0
 cgimage, m_mask, /keep, /overplot
 cgplot, box, xrange = [0, 30], yrange =[0,30], $
  axiscolor='red', /NOERASE , /NODATA, aspect = 1.0
==============================================
```

If I do not add the single pixel and set the PIXEL_CENTER to subtract a pixel off, then my mask seems to be a pixel down on the Y2 and X2 lines.

My question is:

Does this seem a reasonable solution? Or am I being very stupid? I can understand what the ROI is doing, but a little unsure exactly what the issue is?

Is taking the PIXEL_CENTER keyword out and adding/subtracting one from the box like:

x1-- & x2++
y1-- & x3++

strictly the same result?

David

Subject: Re: IDL ROI Objects
Posted by David Fanning on Tue, 27 Jan 2015 18:53:13 GMT
View Forum Message <> Reply to Message

David B writes:

> My question is:
>
> Does this seem a reasonable solution? Or am I being very stupid? I can understand what the ROI is doing, but a little unsure exactly what the issue is?

The ROI code has a left-bottom bias. That is, it treats the lower-left corner of a pixel as the [0,0] location of the pixel. This is sort of standard IDL behavior. But, you don't have to add *anything* to the roibox in your code. All you really have to do is make sure the code uses the center of the pixel for doing its calculations. You do this by making the PIXEL_CENTER=[0.5, 0.5].

Here is slightly better code that will allow you to see what is happening a bit easier.

http://www.idlcoyote.com/misc/box_test.pro

Cheers,

David
--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.idlcoyote.com/
Sepore ma de ni thue. ("Perhaps thou speakest truth.")

Subject: Re: IDL ROI Objects
Posted by David Fanning on Tue, 27 Jan 2015 19:00:34 GMT
View Forum Message <> Reply to Message

David Fanning writes:

>
> David B writes:
>
>> My question is:
>>
>> Does this seem a reasonable solution? Or am I being very stupid? I can understand what the ROI is doing, but a little unsure exactly what the issue is?
>
> The ROI code has a left-bottom bias. That is, it treats the lower-left
> corner of a pixel as the [0,0] location of the pixel. This is sort of
> standard IDL behavior. But, you don't have to add *anything* to the
> roibox in your code. All you really have to do is make sure the code
> uses the center of the pixel for doing its calculations. You do this by
> making the PIXEL_CENTER=[0.5, 0.5].
>
> Here is slightly better code that will allow you to see what is
> happening a bit easier.
>
>   http://www.idlcoyote.com/misc/box_test.pro
>

Whoops! I was wrong about that. (I shouldn't use main-level programs for testing purposes. Something I know well, yet often seem to ignore!) Clearly, something screwy is going on here. Someone else will have to explain it to us. :-)

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.idlcoyote.com/
Sepore ma de ni thue. ("Perhaps thou speakest truth.")

Subject: Re: IDL ROI Objects
Posted by Fabzi on Tue, 27 Jan 2015 23:10:02 GMT
View Forum Message <> Reply to Message

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

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

Subject: Re: IDL ROI Objects
Posted by David B on Wed, 28 Jan 2015 10:06:43 GMT
View Forum Message <> Reply to Message

I also forgot to mention that in terms of mixing polygon and pixel I am sorry.

I mean that the 'polygon' created by IDLanROI is merely the set of pixels that make up the same area in the same shape and location as CGCOLORFILL.

I know it is strictly a mask of pixels.

David

Subject: Re: IDL ROI Objects
Posted by David Fanning on Wed, 28 Jan 2015 14:12:32 GMT
View Forum Message <> Reply to Message

David B writes:

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

A useful talent. :-)

Here is a piece of code that I think dramatically shows the problem:

```
;************************************************************ *******
PRO ROI_BUG
poly = [[5, 10, 10, 5, 5], [5, 5, 10, 10, 5]]
cgDisplay, 600, 600, WID=0, Title = 'Polygon Fill'
cgPlot, [1], XRange=[0,20], YRange=[0,20], /NoData, ASPECT=1.0, $
   YTickLen=1.0, XTickLen=1.0
cgPolygon, poly[*,0], poly[*,1], COLOR='dodger blue', /Fill

cgDisplay, 600, 600, WID=1, Title='Polygon Pixel Fill - PolyFillV'
testImage = BytArr(20,20)+1B
pixels = PolyfillV(poly[*,0], poly[*,1], 20, 20)
testImage[pixels]=255
TVLCT, cgColor('red6', /Triple), 255
TVLCT, cgColor('white', /Triple), 1
cgPlot, [1], XRange=[0,20], YRange=[0,20], /NoData, ASPECT=1.0
cgImage, testImage, XRange=[0,20], YRange=[0,20], /Overplot
cgPlot, [1], XRange=[0,20], YRange=[0,20], /NoData, ASPECT=1.0, $
   YTickLen=1.0, XTickLen=1.0, /NoErase

cgDisplay, 600, 600, WID=2, Title='Polygon Interior Pixel Fill -
IDLanROI'
p = OBJ_NEW('IDLanROI', poly[*,0], poly[*,1])
mask = p -> ComputeMask(DIMENSIONS=[20,20], MASK_RULE=1)
pixels = Where(mask EQ 255)
anImage = BytArr(20,20)+1B
anImage[pixels] = 255
cgPlot, [1], XRange=[0,20], YRange=[0,20], /NoData, ASPECT=1.0
cgImage, anImage, XRange=[0,20], YRange=[0,20], /Overplot
cgPlot, [1], XRange=[0,20], YRange=[0,20], /NoData, ASPECT=1.0, $
   YTickLen=1.0, XTickLen=1.0, /NoErase

cgDisplay, 600, 600, WID=3, Title='Polygon All Pixel Fill - IDLanROI'
p = OBJ_NEW('IDLanROI', poly[*,0], poly[*,1])
mask = p -> ComputeMask(DIMENSIONS=[20,20], MASK_RULE=2)
pixels = Where(mask EQ 255)
anImage = BytArr(20,20)+1B
anImage[pixels] = 255
cgPlot, [1], XRange=[0,20], YRange=[0,20], /NoData, ASPECT=1.0
cgImage, anImage, XRange=[0,20], YRange=[0,20], /Overplot
cgPlot, [1], XRange=[0,20], YRange=[0,20], /NoData, ASPECT=1.0, $
    YTickLen=1.0, XTickLen=1.0, /NoErase
END
;************************************************************ *******
```

With IDLanROI, if you try to take interior pixels, the box is one pixel

too short. If you try to take *all* the pixels, the box is one pixel too
long. It seems impossible to get the box "just right", given a
particular polygon description.

Cheers,

David
--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.idlcoyote.com/
Sepore ma de ni thue. ("Perhaps thou speakest truth.")

Subject: Re: IDL ROI Objects
Posted by David Fanning on Wed, 28 Jan 2015 14:19:12 GMT
View Forum Message <> Reply to Message

David Fanning writes:

> With IDLanROI, if you try to take interior pixels, the box is one pixel
> too short. If you try to take *all* the pixels, the box is one pixel too
> long. It seems impossible to get the box "just right", given a
> particular polygon description.

By the way, if you change the PIXEL_CENTER in the IDLanROI calculations,
you just move the problem one pixel away from where it was. The problem
itself doesn't go away.

Cheers,

David
--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.idlcoyote.com/
Sepore ma de ni thue. ("Perhaps thou speakest truth.")

Subject: Re: IDL ROI Objects
Posted by David Fanning on Wed, 28 Jan 2015 17:17:07 GMT
View Forum Message <> Reply to Message

David Fanning writes:

> With IDLanROI, if you try to take interior pixels, the box is one pixel
> too short. If you try to take *all* the pixels, the box is one pixel too

> long. It seems impossible to get the box "just right", given a
> particular polygon description.

Here is an article that describes the problem for those who prefer not
to run the code:

http://www.idlcoyote.com/code_tips/roipolygon.php

Cheers,

David
--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.idlcoyote.com/
Sepore ma de ni thue. ("Perhaps thou speakest truth.")

Subject: Re: IDL ROI Objects
Posted by Fabzi on Wed, 28 Jan 2015 17:49:07 GMT
View Forum Message <> Reply to Message

David,

the problem is, I believe, not in IDLanROI. It is in how you DRAW the mask.

Consider your example:

```
poly = [[5, 10, 10, 5, 5], [5, 5, 10, 10, 5]]

cgDisplay, 400, 400, WID=2, Title='Polygon Interior Pixel Fill - IDLanROI'
p = OBJ_NEW('IDLanROI', poly[*,0], poly[*,1])
mask = p -> ComputeMask(DIMENSIONS=[20,20])
pixels = Where(mask EQ 255)
anImage = BytArr(20,20)+1B
anImage[pixels] = 255
cgPlot, [1], XRange=[0,19], YRange=[0,19], /NoData, ASPECT=1.0
cgImage, anImage, XRange=[0,19], YRange=[0,19], /Overplot
cgPlot, [1], XRange=[0,19], YRange=[0,19], /NoData, ASPECT=1.0, $
    YTickLen=1.0, XTickLen=1.0, /NoErase
```

Note that I changed the range of the plot to [0, 19].

IDLanROI's follows a PIXEL CENTER view. so a pixel is touched from -0.5
to +0.5. IDL's graphics pixels are following a lower left pixel convention.
PIXEL_CENTER does not change anything to this fact, it just shifts the
whole grid.

See those three examples which are perfectly coherent with each other:

```
pro ROIbug

poly = [[5, 10, 10, 5, 5], [5, 5, 10, 10, 5]]

cgDisplay, 600, 600, WID=1, Title='Polygon rule 0 Pixel Fill - IDLanROI'
p = OBJ_NEW('IDLanROI', poly[*,0], poly[*,1])
mask = p -> ComputeMask(DIMENSIONS=[20,20], MASK_RULE=0)
pixels = Where(mask EQ 255)
anImage = BytArr(20,20)+1B
anImage[pixels] = 255
cgImage, anImage, /AXES, /SAVE
centersX = transpose(fltarr(20)+1) ## (INDGEN(20)+ 0.5)
centersY = transpose(INDGEN(20)+ 0.5) ## (fltarr(20)+1)
cgPlotS, centersX, centersY, PSYM=16, /DATA
cgPlotS, poly[*, 0]+ 0.5, poly[*, 1]+ 0.5, COLOR='blue', /DATA

cgDisplay, 600, 600, WID=2, Title='Polygon rule 1 Pixel Fill - IDLanROI'
p = OBJ_NEW('IDLanROI', poly[*,0], poly[*,1])
mask = p -> ComputeMask(DIMENSIONS=[20,20], MASK_RULE=1)
pixels = Where(mask EQ 255)
anImage = BytArr(20,20)+1B
anImage[pixels] = 255
cgImage, anImage, /AXES, /SAVE
centersX = transpose(fltarr(20)+1) ## (INDGEN(20)+ 0.5)
centersY = transpose(INDGEN(20)+ 0.5) ## (fltarr(20)+1)
cgPlotS, centersX, centersY, PSYM=16, /DATA
cgPlotS, poly[*, 0]+ 0.5, poly[*, 1]+ 0.5, COLOR='blue', /DATA

cgDisplay, 600, 600, WID=3, Title='Polygon  rule 2  Pixel Fill - IDLanROI'
p = OBJ_NEW('IDLanROI', poly[*,0], poly[*,1])
mask = p -> ComputeMask(DIMENSIONS=[20,20], MASK_RULE=2)
pixels = Where(mask EQ 255)
anImage = BytArr(20,20)+1B
anImage[pixels] = 255
cgImage, anImage, /AXES, /SAVE
centersX = transpose(fltarr(20)+1) ## (INDGEN(20)+ 0.5)
centersY = transpose(INDGEN(20)+ 0.5) ## (fltarr(20)+1)
cgPlotS, centersX, centersY, PSYM=16, /DATA
cgPlotS, poly[*, 0]+ 0.5, poly[*, 1]+ 0.5, COLOR='blue', /DATA

end
```

Cheers,

Fabien

On 28.01.2015 18:17, David Fanning wrote:
> David Fanning writes:
>
>> With IDLanROI, if you try to take interior pixels, the box is one pixel
>> too short. If you try to take *all* the pixels, the box is one pixel too
>> long. It seems impossible to get the box "just right", given a
>> particular polygon description.
>
> Here is an article that describes the problem for those who prefer not
> to run the code:
>
> http://www.idlcoyote.com/code_tips/roipolygon.php
>
> Cheers,
>
> David
>

Subject: Re: IDL ROI Objects
Posted by David Fanning on Wed, 28 Jan 2015 18:17:10 GMT
View Forum Message <> Reply to Message

Fabien writes:

> See those three examples which are perfectly coherent with each other:

Humm. I guess maybe I should look up "perfectly coherent" in my
dictionary. ;-)

I do note that even Function Graphics gets the notion of a polygon
correct, a result I find especially remarkable:

```
poly = [[5, 10, 10, 5, 5], [5, 5, 10, 10, 5]]
aplot = Plot([1], XRange=[0,20], YRange=[0,20], /NoData, $
  YTickLen=1.0, XTickLen=1.0)
apoly = Polygon(poly[*,0], poly[*,1], /FILL_BACKGROUND, $
  FILL_COLOR='Steel Blue', /DATA)
```

I think if you have to jump through elaborate hoops and rationalizations
to justify results you intuitively can't possibly expect, there is
probably something wrong somewhere.

This would go a long way, too, in explaining why IDLanROI areas don't
jive with areas calculated from, say, chain-code algorithms that also
rely on pixel centers to determine if a pixel is in or out of a polygon.

Cheers,

David
--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.idlcoyote.com/
Sepore ma de ni thue. ("Perhaps thou speakest truth.")

Subject: Re: IDL ROI Objects
Posted by Fabzi on Wed, 28 Jan 2015 18:30:24 GMT
View Forum Message <> Reply to Message

On 28.01.2015 19:17, David Fanning wrote:
> Fabien writes:
>
>>> See those three examples which are perfectly coherent with each other:
> Humm. I guess maybe I should look up "perfectly coherent" in my
> dictionary.;-)

I am sorry David, but I don't understand your point ;-) Did you find a
bug in IDLanROI, or do you find it's behavior non-intuitive?

Subject: Re: IDL ROI Objects
Posted by David Fanning on Wed, 28 Jan 2015 18:42:53 GMT
View Forum Message <> Reply to Message

Fabien writes:

> I am sorry David, but I don't understand your point ;-) Did you find a
> bug in IDLanROI, or do you find it's behavior non-intuitive?

I find it's non-intuitive behavior a de facto bug. :-)

Cheers,

David
--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.idlcoyote.com/
Sepore ma de ni thue. ("Perhaps thou speakest truth.")

Subject: Re: IDL ROI Objects
Posted by Fabzi on Wed, 28 Jan 2015 18:56:24 GMT
View Forum Message <> Reply to Message

On 28.01.2015 19:42, David Fanning wrote:
> Fabien writes:
>
>> I am sorry David, but I don't understand your point ;-) Did you find a
>> bug in IDLanROI, or do you find it's behavior non-intuitive?
>
> I find it's non-intuitive behavior a de facto bug. :-)

I disagree, and I think that the examples you put online are more
misleading than helping.

IDLanROI's behavior is not going to change (I hope not, since a big part
of my code is relying on this pixel centered behavior), so I believe
that it would be more productive to show how it works rather than
showing how it's not working the way you expect, don't you think?

You quoted only a part of my sentence: "perfectly coherent", but I said:
"perfectly coherent with each other" ;-)

Subject: Re: IDL ROI Objects
Posted by David Fanning on Wed, 28 Jan 2015 19:20:36 GMT
View Forum Message <> Reply to Message

Fabien writes:

> I disagree, and I think that the examples you put online are more
> misleading than helping.
>
> IDLanROI's behavior is not going to change (I hope not, since a big part
> of my code is relying on this pixel centered behavior), so I believe
> that it would be more productive to show how it works rather than
> showing how it's not working the way you expect, don't you think?
>
> You quoted only a part of my sentence: "perfectly coherent", but I said:

> "perfectly coherent with each other" ;-)

I had already updated the article with your example before I read this.
See what you think:

http://www.idlcoyote.com/code_tips/roipolygon.php

Cheers,

David
--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.idlcoyote.com/
Sepore ma de ni thue. ("Perhaps thou speakest truth.")

Subject: Re: IDL ROI Objects
Posted by Fabzi on Wed, 28 Jan 2015 19:24:50 GMT
View Forum Message <> Reply to Message

On 28.01.2015 20:20, David Fanning wrote:
> I had already updated the article with your example before I read this.
> See what you think:

perfect, thanks! I agree to your disagreement, I see that something is
different between graphics and ROI.

But for the climate data world, where a lon lat pixel coordinate
extracted from NetCDF is located at the center of a grid point (not
called "pixel" anymore), the behavior of IDLanROI is quite nice.

Thanks for the discussion David!

Subject: Re: IDL ROI Objects
Posted by David Fanning on Wed, 28 Jan 2015 19:38:51 GMT
View Forum Message <> Reply to Message

Fabien writes:

> But for the climate data world, where a lon lat pixel coordinate
> extracted from NetCDF is located at the center of a grid point (not
> called "pixel" anymore), the behavior of IDLanROI is quite nice.

Yes, but, of course, when you go to put the result on a map, everything
gets confused again! When Matt Savoie and I worked together at NSIDC we

would start out in the morning in complete disagreement and end up in
the afternoon in complete disagreement, but on the opposite side of the
argument from where we started! It's confusing. ;-)

Cheers,

David
--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.idlcoyote.com/
Sepore ma de ni thue. ("Perhaps thou speakest truth.")

Subject: Re: IDL ROI Objects
Posted by David B on Thu, 29 Jan 2015 10:05:17 GMT
View Forum Message <> Reply to Message

On Wednesday, January 28, 2015 at 7:38:57 PM UTC, David Fanning wrote:
> Fabien writes:
>
>> But for the climate data world, where a lon lat pixel coordinate
>> extracted from NetCDF is located at the center of a grid point (not
>> called "pixel" anymore), the behavior of IDLanROI is quite nice.
>
> Yes, but, of course, when you go to put the result on a map, everything
> gets confused again! When Matt Savoie and I worked together at NSIDC we
> would start out in the morning in complete disagreement and end up in
> the afternoon in complete disagreement, but on the opposite side of the
> argument from where we started! It's confusing. ;-)
>
> Cheers,
>
> David
> --
> David Fanning, Ph.D.
> Fanning Software Consulting, Inc.
> Coyote's Guide to IDL Programming: http://www.idlcoyote.com/
> Sepore ma de ni thue. ("Perhaps thou speakest truth.")

So what may be required is IDL to ask a question on opening:

IDl>Are you looking at the floor or the sky?

This is a most confusing issue and indeed non-intuitive. I think my solution may be satisfactory for the time being. I do apologise for causing such a ruckus. I think I have gained valuable insights into how IDL 'thinks' things should work.

So perhaps a more formal tutorial could be done by the IDL software engineers, as it would clearly be inconvenient for the people looking at the floor or the sky if the ROI objects went one way or the other based on the discussion here. I think I will just have to put up with the pixel shifted FITS convention.

David

Subject: Re: IDL ROI Objects
Posted by Fabzi on Thu, 29 Jan 2015 10:50:10 GMT
View Forum Message <> Reply to Message

On 29.01.2015 11:05, David B wrote:
> IDl>Are you looking at the floor or the sky?

For the record, a better "tutorial" here:

```
PRO ROIPolygonAlternativeView

poly = [[0.8, 1.7, 2.7, 2.7, 0.8], [0.2, 2.2, 2.2, 0.2, 0.2]]

nn = 4
centersX = transpose(fltarr(nn)+1) ## (INDGEN(nn)+ 0.5)
centersY = transpose(INDGEN(nn)+ 0.5) ## (fltarr(nn)+1)

TVLCT, cgColor('red6', /Triple), 255
TVLCT, cgColor('white', /Triple), 1

cgDisplay, 700, 800
pos = cgLayout([2,2], OXMARGIN=[2, 2], OYMARGIN=[3, 4], XGAP=2, YGAP=6)


; Rule 0
p = OBJ_NEW('IDLanROI', poly[*,0], poly[*,1])
mask = p->ComputeMask(DIMENSIONS=[nn, nn], MASK_RULE=0)
anImage = BytArr(nn,nn)+1B
anImage[Where(mask EQ 255)] = 255
po = pos[*,0]
cgImage, anImage, /AXES, /SAVE, POS=po, TITLE='RULE 0', /NOERASE
cgPlotS, centersX, centersY, PSYM=16, /DATA
cgPlotS, poly[*, 0]+ 0.5, poly[*, 1]+ 0.5, COLOR='green', /DATA, THICK=3
cgPlot, [1], XRange=[0,4], YRange=[0,4], /NoData, $
    YTickLen=1.0, XTickLen=1.0, POS=po, /NOERASE

; Rule 1
p = OBJ_NEW('IDLanROI', poly[*,0], poly[*,1])
mask = p->ComputeMask(DIMENSIONS=[nn, nn], MASK_RULE=1)
anImage = BytArr(nn,nn)+1B
```

```
anImage[Where(mask EQ 255)] = 255
po = pos[*,1]
cgImage, anImage, /AXES, /SAVE, POS=po, TITLE='RULE 1', /NOERASE
cgPlotS, centersX, centersY, PSYM=16, /DATA
cgPlotS, poly[*, 0]+ 0.5, poly[*, 1]+ 0.5, COLOR='green', /DATA, THICK=3
cgPlot, [1], XRange=[0,4], YRange=[0,4], /NoData, $
     YTickLen=1.0, XTickLen=1.0, POS=po, /NOERASE

; Rule 2
p = OBJ_NEW('IDLanROI', poly[*,0], poly[*,1])
mask = p->ComputeMask(DIMENSIONS=[nn, nn], MASK_RULE=2)
anImage = BytArr(nn,nn)+1B
anImage[Where(mask EQ 255)] = 255
po = pos[*,2]
cgImage, anImage, /AXES, /SAVE, POS=po, TITLE='RULE 2', /NOERASE
cgPlotS, centersX, centersY, PSYM=16, /DATA
cgPlotS, poly[*, 0]+ 0.5, poly[*, 1]+ 0.5, COLOR='green', /DATA, THICK=3
cgPlot, [1], XRange=[0,4], YRange=[0,4], /NoData, $
     YTickLen=1.0, XTickLen=1.0, POS=po, /NOERASE

; Contains points
p = OBJ_NEW('IDLanROI', poly[*,0], poly[*,1])
mask = p->ContainsPoints(centersX[*]-0.5, centersY[*]-0.5)
anImage = BytArr(nn,nn)+1B
p = where(mask eq 1)
po = pos[*,3]
cgImage, anImage, /AXES, /SAVE, POS=po, TITLE='CONTAINS POINTS', /NOERASE
cgPlotS, centersX, centersY, PSYM=16, /DATA
cgPlotS, centersX[p], centersY[p], PSYM=16, /DATA, COLOR='red'
cgPlotS, poly[*, 0]+ 0.5, poly[*, 1]+ 0.5, COLOR='green', /DATA, THICK=3

end
```