Subject: Fractional Pixels Origin?
Posted by CJCrockett on Wed, 15 Feb 2006 20:53:14 GMT
View Forum Message <> Reply to Message

A quick question. Does anyone know, definitively, what origin IDL uses when defining fractional pixels? Is (0.0,0.0) the center, bottom left, or other, of the pixel?

Thanks!

Regards,
Christopher Crockett
University of Maryland

Subject: Re: Fractional Pixels Origin?
Posted by David Fanning on Fri, 17 Feb 2006 14:19:53 GMT
View Forum Message <> Reply to Message

Michael A. Miller writes:

> Now my main problem is that every time I see a discussion like
> this, I have an anxiety attack about whether or not my code
> consistently does what I think it does!

Man, I hear you on this! The only thing that could possibly introduce more errors into your program than changing a counting convention is allowing the user to move the origin from the lower-left corner to the upper-left corner, ala !Order. :-(

I guess I use fractional pixels on some of my images, too. At least I remember *weeks* of debugging. A nice convention that made sense would be nice. I guess I'll be voting for JD, too. :-)

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/

Subject: Re: Fractional Pixels Origin?

Posted by CJCrockett on Fri, 17 Feb 2006 15:24:09 GMT
View Forum Message <> Reply to Message

Wow....thank you so much for all your replies, though I'm not quite sure how to digest all of it.
The fractional pixels are the output of locating the centroid of a comet in an image. The centroid then defines my center from which I calculate aperture photometry. In the interest of getting the most accurate results, I am using fractional pixel areas and hence need to know where to start counting from "inside" the pixel.
Our images are in the FITS format, as a previous poster indicated. But the centroiding algorithm knows nothing of FITS, it just receives a 2-d image array and goes to town. If anyone has any familiarity with astronomy libraries, I am using the CNTRD procedure in the Goddard library which in turn uses DAOPHOT.
I understand now that there are several different conventions in use (and my data set may actually be using two different conventions as they are from two different sources)! I know that one source puts 0,0 at the center of the pixel. My data, using CNTRD, I don't know about. Is there a test of some sort I can do to clear up the confusion?

Thanks again for the responses!

Regards,
Christopher Crockett
University of Maryland

Subject: Re: Fractional Pixels Origin?
Posted by David Fanning on Fri, 17 Feb 2006 15:37:52 GMT
View Forum Message <> Reply to Message

CJCrockett writes:

> Is there a test of some sort I can do to clear up the confusion?

It's probably not too late to switch to MatLab. :-)

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/

Subject: Re: Fractional Pixels Origin?
Posted by CJCrockett on Fri, 17 Feb 2006 15:40:38 GMT
View Forum Message <> Reply to Message

Okay, may be this bad form, but I'm going to respond to my own question. :-)

I tried a very simple test:
a = fltarr(100,100)
a[49:51,49:51] = 1.
cntrd,a,50,50,xcen,ycen,1.5

print,xcen,ycen
 50.0000 50.0000

So, I made a 3x3 box centered on [50,50]. The centroid algorithm reports that the centroid is located at 50.000,50.000. This tells me that CNTRD sets the pixel origin at the center of the pixel. Does this seem like a reasonable test?

Regards,
Christopher Crockett
University of Maryland

Subject: Re: Fractional Pixels Origin?
Posted by David Fanning on Fri, 17 Feb 2006 22:28:26 GMT
View Forum Message <> Reply to Message

JD Smith writes:

> Sorry, I had it backwards, FITS centers the first pixel at [1,1], and
> the Nasa library uses [0,0] (which is called "IDL convention"). If
> you have a choice, don't choose the FITS standard ([1,1]), or the "IDL
> convention" ([0,0]), but the natural "I'm a tiny ant living on the
> surface of your detector and measuring pixel positions from the edge
> with my tiny little ruler": [0.5,0.5]. The only disadvantage is all
> pixel centers are now fractional.
>
> Wayne is very careful to document the convention in all of the NasaLib
> routines, so if confused be sure to read the useful doc headers (as I
> just did to remedy my confusion!).

Do you think there is such a thing as knowing too much about a subject? Two days ago as was as happy as a clam. Now I'm paranoid as hell. :-(

Cheers,

David
--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/

Subject: Re: Fractional Pixels Origin?
Posted by Greg Hennessy on Sat, 18 Feb 2006 01:14:44 GMT
View Forum Message <> Reply to Message

On 2006-02-17, David Fanning <davidf@dfanning.com> wrote:
> Do you think there is such a thing as knowing too much about
> a subject? Two days ago as was as happy as a clam. Now I'm
> paranoid as hell. :-(

Want to do some astrometry? Or discuss the merits of aperture
photometry verses annular photometry?

Subject: Re: Fractional Pixels Origin?
Posted by Mark Hadfield on Sun, 19 Feb 2006 20:50:35 GMT
View Forum Message <> Reply to Message

JD Smith wrote:
> ... the natural "I'm a tiny ant living on the
> surface of your detector and measuring pixel positions from the edge
> with my tiny little ruler": [0.5,0.5].

Suffice it to say that's what natural to one person can seem very
unnatural to another.

Well perhaps it doesn't suffice, so I'll spell it out. I am a tiny
scientist crawling over a chess board measuring stuff. I go the first
square and make a measurement in its centre: I'll store that at index
[0,0] and record the location as [x0,y0]. Then I move through the
squares on the chess board stepping dx in the x direction by in the y
direction until I get to [x0+63*dx,y0+63*dy]. I now have a set of
measurements, my_data, dimensioned [64,64], and a pair of location
vectors, x=x0+dxfindgen(64) and y=y0+dy*findgen(64). The measurements
are *at* the locations, as is only natural.

I can contour this:

```
contour, my_data, x, y
```

I can do a surface

```
surface, my_data, x, y
```

or I can plot a slice

```
plot, x, my_data[*,3]
```

Then I think, "wouldn't it be cool to show this data with one of those new-fangled false-colour images!" Obviously, I want the command to be

```
my_image_plot, my_data, x, y
```

Now the image will fill the space -0.5*dx<x<63.5*dx, -0.5*dy<y<63.5*dy, but that's a detail for the plotting program. Then I think, "Hang on. I don't like the little square, uniform areas around each data point, because this variable varies smoothly" so I change to an image in which the colour is interpolated between data points. The data points are still at the same locations, so now I want my image to fill the space 0<x<63*dx, 0*dy<y<63*dy.

So my contention is that if you think of images as just another way to show data, not privileged above contours, surfaces, then the natural convention is to have the lower-left data point at [0,0].

--
Mark Hadfield "Kei puwaha te tai nei, Hoea tahi tatou"
m.hadfield@niwa.co.nz
National Institute for Water and Atmospheric Research (NIWA)

Subject: Re: Fractional Pixels Origin?
Posted by Mark Hadfield on Sun, 19 Feb 2006 21:00:03 GMT
View Forum Message <> Reply to Message

Mark Hadfield wrote:
>
> my_image_plot, my_data, x, y

Oops, sorry, JD. I missed an opportunity to gratuitously offend your sensibilities. The correct command would be

```
mgh_image_plot, my_data, x, y
```

--
Mark Hadfield "Kei puwaha te tai nei, Hoea tahi tatou"
m.hadfield@niwa.co.nz

Subject: Re: Fractional Pixels Origin?
Posted by JD Smith on Mon, 20 Feb 2006 18:56:18 GMT
View Forum Message <> Reply to Message

On Mon, 20 Feb 2006 10:00:03 +1300, Mark Hadfield wrote:

> Mark Hadfield wrote:
>>
>> my_image_plot, my_data, x, y
>
> Oops, sorry, JD. I missed an opportunity to gratuitously offend your
> sensibilities. The correct command would be

Thanks for that, Mark.

I don't see your argument. Fractional pixels are useful only when calculating things which relate pixel coordinates to some other coordinate (like celestial coordinates on the sky, etc), or when computing other derived fractional pixel positions (e.g. clip two polygons). Obviously, the computer has no understanding of a fractional pixel, but only the memory indexed offset [0,0]. But the latter does not have to drive the former. In fact I'd say it's rather strange to let the layout in memory dictate a physical coordinate system. You don't need to make this distinction.

An example: if you have an array of size 8x8, and it corresponds to, I don't know, electron mobility vs. viscosity, of course it doesn't bother you that the computer's understanding of the "coordinate" is the dumb and fixed integer set [0-7,0-7], and not your higher-level coordinate system of, say [0.001-0.1,100-200], in logarithmic bins. My recommendation puts pixel coordinate distance on the same footing as any other physical measure. How the underlying data is held in memory is immaterial.

So, perhaps if you consider two separate coordinate systems, things are made simpler (really!):

1. The "memory" coordinate system. This is the basis for mapping to any other coordinate system. You'll never get rid of it, since it's how the computer understands arrays.

2. The physical "pixel distance" coordinate system. This is among the physical systems which can be mapped to by #1 (trivially, by adding 0.5).

The first is a necessity, given how computers arrange data. The second is a particular choice, for computing distance/position based transformations.

My last argument is this: when you have a widget readout of fractional pixel position, do you really want [-0.5,-0.5] to be a valid position?

JD

P.S. Feel free to rename it JDTS_HIST_ND(). Much easier to remember.

Subject: Re: Fractional Pixels Origin?
Posted by David Fanning on Tue, 21 Feb 2006 13:29:09 GMT
View Forum Message <> Reply to Message

Greg Hennessy writes:

> Want to do some astrometry? Or discuss the merits of aperture
> photometry verses annular photometry?

I do, as a matter of fact. I spent the weekend writing some new image processing functions and beefing up some of my old ones to handle these gnarly FITS images. I have in mind to write some astronomy image software for the Astronomy Club at the local high school. I think I can do better than what I have found on the web, if I can just get my mind around these fractional pixels. :-)

Cheers,

David

P.S. I find myself pouring over astronomy texts in my free time these days. If your astronomy group would like to work collaboratively for a month or so on a project, I am available for immediate travel. This is particularly true if you are in, say, England and would like me to visit on or about July 9th. :-)

--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/

Subject: Re: Fractional Pixels Origin?
Posted by Mark Hadfield on Tue, 21 Feb 2006 21:44:25 GMT

JD Smith wrote:
> [snip]
>
> I don't see your argument. Fractional pixels are useful only when
> calculating things which relate pixel coordinates to some other
> coordinate (like celestial coordinates on the sky, etc), or when
> computing other derived fractional pixel positions (e.g. clip two
> polygons). Obviously, the computer has no understanding of a
> fractional pixel, but only the memory indexed offset [0,0]. But the
> latter does not have to drive the former. In fact I'd say it's rather
> strange to let the layout in memory dictate a physical coordinate
> system. You don't need to make this distinction.

Hang on. I'm not letting layout in memory dictate a physical coordinate system. I'm just adopting a convention for "image plots" that is the same as for other plot types.

I have a data array, mydata, dimensioned [m,n]

When I type

```
contour, mydata
```

the contour routine implicitly locates the data points at x = [0,...,m-1], y = [0,...,n-1]. Ditto when I type

```
surface, mydata
```

Similarly for plots of 1D arrays.

When I plot this data via a false-colour image using my own image-plotting routine

```
mgh_image_plot, mydata
```

then I adopt the same convention. If the image is "blocky" (eg an IDLgrImage with INTERPOLATE=0) then the cell centres represent the data points; the image has to fill the space -0.5 <= x <= m-0.5 (similarly for y) to get them in the right location. If the image is "smooth" (eg an IDLgrImage with INTERPOLATE=1) then the outer row & column of data points lie on the edge of the image and it fills the space 0 <= x <= m-1, etc.

That's the convention I prefer, for reasons which seem good to me. If that's not the subject under discussion in this thread (which was never entirely clear to me) then please disregard it.

--
Mark Hadfield "Kei puwaha te tai nei, Hoea tahi tatou"
m.hadfield@niwa.co.nz
National Institute for Water and Atmospheric Research (NIWA)

Subject: Re: Fractional Pixels Origin?
Posted by JD Smith on Wed, 22 Feb 2006 01:25:18 GMT
View Forum Message <> Reply to Message

On Wed, 22 Feb 2006 10:44:25 +1300, Mark Hadfield wrote:

> JD Smith wrote:
>> [snip]
>>
>> I don't see your argument. Fractional pixels are useful only when
>> calculating things which relate pixel coordinates to some other
>> coordinate (like celestial coordinates on the sky, etc), or when
>> computing other derived fractional pixel positions (e.g. clip two
>> polygons). Obviously, the computer has no understanding of a
>> fractional pixel, but only the memory indexed offset [0,0]. But the
>> latter does not have to drive the former. In fact I'd say it's rather
>> strange to let the layout in memory dictate a physical coordinate
>> system. You don't need to make this distinction.
>
> Hang on. I'm not letting layout in memory dictate a physical coordinate
> system. I'm just adopting a convention for "image plots" that is the
> same as for other plot types.
>
> I have a data array, mydata, dimensioned [m,n]
>
> When I type
>
> contour, mydata
>
> the contour routine implicitly locates the data points at x =
> [0,...,m-1], y = [0,...,n-1]. Ditto when I type
>
> surface, mydata
>
> Similarly for plots of 1D arrays.
>
> When I plot this data via a false-colour image using my own
> image-plotting routine
>
> mgh_image_plot, mydata
>
> then I adopt the same convention. If the image is "blocky" (eg an

> IDLgrImage with INTERPOLATE=0) then the cell centres represent the data
> points; the image has to fill the space -0.5 <= x <= m-0.5 (similarly
> for y) to get them in the right location. If the image is "smooth" (eg
> an IDLgrImage with INTERPOLATE=1) then the outer row & column of data
> points lie on the edge of the image and it fills the space 0 <= x <=
> m-1, etc.

My concept of fractional pixels treats them as a physical unit, like inches. If you had a plot of sound speed vs. linear density, and had an image with the same physical axes, you'd have to make special arrangements to have the occupy the same region of your screen, and it wouldn't concern you. Why are "pixels" different? The fact that choosing [0,0] as the origin to make things line up is an artifact of how plot/contour/surface/etc. work.

A good example is PSYM=10. Anyone who plots histograms will quickly realize the origin choice of PLOT,PSYM=10 is off by one-half bin, and codes around it. A histogram at position q with bin width w should be a line from q-w/2 to q+w/2, not q to q+w (or whatever it uses). My point is that there are reasons for adopting a given pixel origin which are entirely separate from the convenience of falling in line with what other tools or data systems have adopted (e.g. [1,1] in FITS, [0,0] in IDL, etc.). For some uses (such as making images and plot line up), perhaps that choice is a good one.

Anyway, given that so many standards abound, I think it's an issue we'll all have to continue to deal with.

JD

Subject: Re: Fractional Pixels Origin?
Posted by mmiller3 on Wed, 22 Feb 2006 14:30:36 GMT
View Forum Message <> Reply to Message

>>>> > "JD" == JD Smith <jdsmith@as.arizona.edu> writes:

> My concept of fractional pixels treats them as a physical
> unit, like inches.
[...]
> My point is that there are reasons for adopting a given
> pixel origin which are entirely separate from the
> convenience of falling in line with what other tools or
> data systems have adopted (e.g. [1,1] in FITS, [0,0] in
> IDL, etc.).

That is the way I think of them as well. Generally, my code handles three coordinate systems: the data indices, spatial

coordinates that represent where the data was measured, (or when, or ...), and device coordinates that correspond to physical pixels on the screen. The mapping between these coordinates is just what IDL's T3D and CONVERT_COORD are intended to handle. In fact, if I correctly handle these transformations, I don't care at all about how IDL or anyother software deals with pixels on the screen.

When I first started trying to sort this all out, a lot of my confusion came from refering to the data elements in my measurements (pixels in medical images) as "pixels" and then getting confused when I didn't want a 1 to 1 mapping between physical pixels on a display device and image pixels in my data. It helps me a lot to reserve the word "pixel" to mean "physical pixels on a display device."

For those of you who are not too bored by all this (yet), here's how I handle the coordinate transformations. I promise to mention fractional pixels, so it is still somewhat on topic!

If points in space are represented by homogeneous space coordinates r = [x,y,z,1], and homogeneous data coordinates by p = [i,j,k,1], then I create a matrix, using T3D, to transform between them: p = V # r and r = inverse(V) # p. For the simple 2d case, where I've measured at Nx by Ny points on a uniform grid with spacings dx and dy, V is just

```
   [ 1/dx    0      (Nx-1)/2 ]
V = [ 0       1/dy   (Ny-1)/2 ]
   [ 0       0      1        ]
```

In IDL, that is

```
T3D, /RESET
T3D, SCALE=[1.0/dx, 1.0/dy, 0.0]
T3D, TRANSLATE=[(Nx-1)/2.0, (Ny-1)/2.0, 0]
```

To go between device and space coordinates, I use CONVERT_COORD after the appropriate PLOT command (usually with /NODATA) to set up the needed system variables. To store the plot system variables, I use the very handy savesysvar and restsysvar from ICG_LIB, Forschungzentrum Juelich GmbH, http://www.fz-juelich.de/icg/icg-i/idl_icglib/idl_lib_intro. html.

So, if I want to draw some images, I follow these steps

1 - set up device coordinates with PLOT and save them with savesysvar

2 - find space coordinates of device pixels using CONVERT_COORD,
restoring plot system variables with restsysvar

3 - for each data set that I'm dealing with, calculate
(fractional) data array indices with p = V # r, using the
appropriate V for each data set. (these are what I used to
think of as fractional pixels, but I'm reserving the p word
for device pixels)

4 - INTERPOLATE the data set at the (fractional) data array
indices p. This gives me an array of data values that map 1
to 1 to the (integer) device pixels on the screen.

5 - combine multiple data sets as wanted (alpha blend, color
wash, whatever useful or glitzy method I like)

6 - TV the data to the screen

7 - if I want to overlay a plot, say using contour, I CONTOUR
after calling restsysvar

Mike

--
Michael A. Miller mmiller3@iupui.edu
Imaging Sciences, Department of Radiology, IU School of Medicine

---

Subject: Re: Fractional Pixels Origin?
Posted by William C. Keel on Mon, 06 Mar 2006 19:21:32 GMT
View Forum Message <> Reply to Message

David Fanning <davidf@dfanning.com> wrote:
> CJCrockett writes:

>> Is there a test of some sort I can do to clear up the confusion?

> It's probably not too late to switch to MatLab. :-)

Or failing that, you could follow us mere mortals and hand it a data
set in which the peak is clearly near the center of a pixel and
see whether it gives a results nearer 0 or 0.5 in the fractional
pixel. (Alternately, a data set with the peak between pixels and
thus having four equal pixels in the middle).

Bill Keel
