
Subject: Re: Cannot understand a part of the IDL routine!! pls help!!

Posted by [Jeremy Bailin](#) on Tue, 25 May 2010 13:23:48 GMT

[View Forum Message](#) <> [Reply to Message](#)

On May 24, 1:26 pm, Brian Daniel <Daniels...@yahoo.com> wrote:

> On 24 May, 08:16, Jeremy Bailin <astroco...@gmail.com> wrote:

>

>

>

>

>

>> On May 23, 11:07 am, David Fanning <n...@dfanning.com> wrote:

>

>>> bala murugan writes:

>>>> The following is a part of the IDL routine for region grow. The

>>>> following three lines of code is used to define the pixels that is the

>>>> ROI pixels.

>>>> x = FINDGEN(16*16) MOD 16 + 276

>>>> y = LINDGEN(16*16) / 16 + 254

>>>> roiPixels = x + y * imgDims[0]

>

>>>> The question is how does it define the ROI pixels?

>>>> I dont see how it does..... Somebody please help me by giving a

>>>> simple and clear description.

>

>>> What is happening here is the IDL is turning one-dimensional

>>> image indices into two-dimensional image indices. Before

>>> the advent of the function Array_Indices, we always had

>>> to do this by hand. This code was obviously written in

>>> those long-ago dark days.

>

>>> Here is an article that explains this process in some

>>> detail:

>

>>> http://www.dfanning.com/tips/where_to_2d.html

>

>>> Cheers,

>

>>> David

>

>>> --

>>> David Fanning, Ph.D.

>>> Fanning Software Consulting, Inc.

>>> Coyote's Guide to IDL Programming:<http://www.dfanning.com/>

>>> Sepore ma de ni thui. ("Perhaps thou speakest truth.")

>

>> Incidentally, is there an in-built routine that I've missed that does

>> the reverse mapping (multi-D to 1D)? I know I've written my own and I

```
>> suspect others have too, but it seems like there ought to be a built-
>> in version.
>
>> -Jeremy.- Hide quoted text -
>
>> - Show quoted text -
>
> reform does the trick. For example:
> image = indgen(20,30,3)
> help, image
> IMAGE      INT      = Array[20, 30, 3]
> image_vector = reform(image,20*30*3)
> help, image_vector
> IMAGE_VECTOR  INT      = Array[1800]
>
> -Brian
```

Yes, you can of course use `l/indgen` to give you a multi-D array that you can use for the mapping (I'm not sure what the point of the reform is - I would just use `image[4,3,0]` to find out the 1D index of the point (4,3,0)). But it requires generating an auxiliary array with the required dimensions, which can be very wasteful of memory in the applications where I tend to use it!

-Jeremy.
