Subject: Re: Dereferencing a Pointer Array
Posted by Liam E. Gumley on Tue, 01 May 2001 18:30:35 GMT
View Forum Message <> Reply to Message

Art Burden wrote:

```
>  I have a structure that contains a pointer array that points to twelve
>  512-by-512 images.  I would like to find the mean image from the twelve
>  images in a simple and fast way.  I understand that I can dereference
>  the pointer to an image or an individual element in an image by using,
>  for example
>
>  img = (*info.images[0])
>  and
>  img_element = (*info.images[0])[240,240]
>
>  but I can't figure out a way to dereference the pointers to all 12 pixel
>  values from a given coordinate in one step.  At this stage, I pass the
>  structure into my averaging subroutine and  I create a new array to
>  store the 12 images. I then fill the array by dereferencing the pointer
>  to each image in a loop.  Finally, I  loop through the rows and columns
>  to get each mean pixel value, as shown below.  Can anyone think of a
>  better (mainly faster) way to do this?
>
>  ;retrieve array of images
>  ffim = lonarr(12,512,512)
>  for num=0,11 do ffim[num,*,*] = *info.images[num]
>
>  ;calculate mean of images
>  mean_ff = fltarr(512,512)
>  for ir = 0,511 do for ic = 0,511 do mean_im[ic,ir] = mean(ffim[*,ic,ir])
```

Perhaps something like this (untested):

```
sum = fltarr(512, 512)
n = 12
for i = 0L, n - 1L do sum = sum + *info.images[i]
avg = sum / float(n)
```

Cheers,
Liam.
http://cimss.ssec.wisc.edu/~gumley/

Subject: Re: Dereferencing a Pointer Array
Posted by Art Burden on Tue, 01 May 2001 18:58:38 GMT
View Forum Message <> Reply to Message

Thanks Liam, that worked beautifully. I knew I was making this complicated..

Sometimes it's easy to use the routines blindly without thinking about how
they are defined (such as, mean = sum/num)

"Liam E. Gumley" wrote:

```
>  Art Burden wrote:
>>  I have a structure that contains a pointer array that points to twelve
>>  512-by-512 images.  I would like to find the mean image from the twelve
>>  images in a simple and fast way.  I understand that I can dereference
>>  the pointer to an image or an individual element in an image by using,
>>  for example
>>
>>  img = (*info.images[0])
>>  and
>>  img_element = (*info.images[0])[240,240]
>>
>>  but I can't figure out a way to dereference the pointers to all 12 pixel
>>  values from a given coordinate in one step.  At this stage, I pass the
>>  structure into my averaging subroutine and  I create a new array to
>>  store the 12 images. I then fill the array by dereferencing the pointer
>>  to each image in a loop.  Finally, I  loop through the rows and columns
>>  to get each mean pixel value, as shown below.  Can anyone think of a
>>  better (mainly faster) way to do this?
>>
>>  ;retrieve array of images
>>  ffim = lonarr(12,512,512)
>>  for num=0,11 do ffim[num,*,*] = *info.images[num]
>>
>>  ;calculate mean of images
>>  mean_ff = fltarr(512,512)
>>  for ir = 0,511 do for ic = 0,511 do mean_im[ic,ir] = mean(ffim[*,ic,ir])
>
>  Perhaps something like this (untested):
>
>  sum = fltarr(512, 512)
>  n = 12
>  for i = 0L, n - 1L do sum = sum + *info.images[i]
>  avg = sum / float(n)
>
>  Cheers,
>  Liam.
>  http://cimss.ssec.wisc.edu/~gumley/
```

Subject: Re: Dereferencing a Pointer Array
Posted by John-David T. Smith on Tue, 01 May 2001 19:28:14 GMT
View Forum Message <> Reply to Message

Art Burden wrote:

> I can't figure out a way to dereference the pointers to all 12 pixel
> values from a given coordinate in one step.

As you can imagine, it would be a sordid business indeed if you could dereference entire pointer arrays all at once:

```
parr=[ptr_new([1,2,3]),ptr_new([[5,6],[7,8]])]

print,total(*parr,1)
```

hmm.. what would we do with that? Since pointers can point to anything, pointer arrays don't necessarily point to similarly dimensioned blocks of data, or even similarly typed:

```
 parr=[ptr_new([1,2,3]),ptr_new([[5,6],[7,8]]),ptr_new('Octop us')]
```

If you really know you will have N images of the same size, you can have your cake and eat it too. Instead of storing them as a pointer array, store them in a single pointer as an image cube:

```
pcube=ptr_new(randomu(sd,20,20))
*pcube=[[[*pcube],[[randomu(sd,20,20)]]]
```

now you can easily get the mean along the "depth" of the cube:

```
meanim=total(*pcube,3)/n_ims
```

This will be much faster than separately dereferencing/adding a long list of pointed-to arrays. It is less flexible though (what if you wanted to remove an array from the middle of the list?).

JD