Subject: Coding for speed help needed
Posted by pford on Wed, 28 May 1997 07:00:00 GMT
View Forum Message <> Reply to Message

I have been requested to explain what I am try to do and why.

I am using a genetic algorithm to do a tomographic reconstruction of radionuclide data using a front or forward projection instead of the usual backprojection method. I am comparing the sinograms generated by the genetic algorithms "object image" compared to the "target's" sinogram to determine the 'fitness.' I am doing this as a proof of concept to calculated
unknown parameters, i.e. attenuation. The B array of bytes is the genetic code that needs to be decoded. Its size is:

pop_size = 36; this can vary greatly
dim = 64

B = bytarr(dim * dim * 5 * pop_size)

There are pop_size individuals
The first 64*64 (dim*dim) bytes code for the activity or intensity in a 64 by 64 image. The next 4*64*64 bytes code for the attenuation coefficients which are
between 0.0 and 1.0, where 1.0 is no attenuation and 0.0 is 100%.

I need to strip off the appropriate bytes and convert them.

For example let:

activity_code = bytarr(dim*dim*4)

activity_code(*) = B(dim:dim*dim*4-1); should give me the bytes I want to
;decode to float.

I need something that is fast and consistent. Since it is a genetic algorithm, the sequencing of the bytes, i.e, which byte is high order vs lower order, or even if they are not contigious is not important, as long as it is consistent.

The number of bytes that code for a float is arbitrary, it just affects the precision. (I am sure I won't need this much precision, but I want to debug it before fiddling with other parameters.)

In case you are curious, the forward-projection without attenuation does work.

Thanks for the help so far. I will give it a try.

Patrick Ford, MD
Department of Radiology
Baylor College of Medicine

Subject: Re: Coding for speed help needed
Posted by jackel[1] on Wed, 28 May 1997 07:00:00 GMT
View Forum Message <> Reply to Message

In article <5mhn9v$r3r@gazette.bcm.tmc.edu> pford@bcm.tmc.edu (Patrick V. Ford) writes:

> What I would like to do, is do this in IDL where the result would go into A,
> where
> A = fltarr(64,64)
> and the byte array is
> B = bytarr(64*64*4)

Not sure if I understand properly, but if your array "B" was of the form B=bytarr(4,64,64), then doing A= long(b,0,64,64) would turn each 4 byte chunk into a long word. Check the documentation for the "offset" argument to "long" (and float and fix etc). If that's what you want, then try using "transpose" to get "B" into the proper shape.

Brian Jackel
