Subject: Re: ROI stack
Posted by Chris[5] on Fri, 13 Jun 2008 00:01:33 GMT
View Forum Message <> Reply to Message

On Jun 12, 11:45 am, mega...@phas.ubc.ca wrote:
> Hello,
> Can anyone see where I'm going wrong here? I'm trying to generate a
> binary mask of a multislice MRI image. When I run this at the
> command line it works, but when I put it in a loop, it fails to return
> anything. i.e. all variables of interest are "undefined"
> Any help/tips would be appreciated!
> Thanks,
> Megan
> AT COMMAND LINE:
> xroi, img[*,*,i], regions_out = roi, /Block
> mask = roi -> ComputeMask(Dimensions = dims, Mask_Rule=2)
> masks[*,*,i] = mask
> LOOP:
> pro twodmask
> img = read_nifti(dialog_pickfile())
> dims = size(img[*,*,3], /dimensions)
> s = size(img, /dimensions)
> n_slices = 2 ;s[2]
> masks = fltarr(dims[0], dims[1], s[2])
> for i = 0,n_slices-1 do begin
> xroi, img[*,*,i], regions_out = roi, /Block
> mask = roi -> ComputeMask(Dimensions = dims, Mask_Rule=2)
> masks[*,*,i] = mask
> endfor
>
> end

When you run a procedure (something that starts with pro), all of the variables defined within the main body are local in scope. That is, when you get to the "end" statement, all of the variables within the program are discarded. Try this:

Change the first line from "PRO twodmask" to "function twodmask." Add the line "return, mask" on the line before "end." Call the function by typing in "mask=twodmask()." This now feeds the variable mask to you upon completion.

chris

Subject: Re: ROI stack

megan17@phas.ubc.ca writes:

> Can anyone see where I'm going wrong here?

I'm reminded on one of my favorite movies, Gettysburg, when
Lee's forces inexplicably fail to press forward to take the high ground
before the Union forces can bring up their main army. General Hill,
totally distraught at his leadership that day, is trying to resign his
commission. "Why a *blind* man could have seen the need to take
that hill!"

> I'm trying to generate a
> binary mask of a multislice MRI image. When I run this at the
> command line it works, but when I put it in a loop, it fails to return
> anything. i.e. all variables of interest are "undefined"

I guess, since you are providing no way to get the local
variables out of your procedure. Are you sure you *want*
a procedure? Why not just leave the procedure definition
statement off of this code, add another END at the end of
the code, and make it a main-level program you could compile
and run with the .RUN command? Then you don't have to worry
about passing data into and out of your procedure.

All procedures and function create local variables inside them.
And thank God they do, or we would have to give a unique name
to every IDL variable we use. Most of us don't have that kind
of vocabulary. :-(

> Any help/tips would be appreciated!

I would do this until you get a bit more familiar with
programming in general. Put the following code in a file
named twodmask.pro:

```
img = read_nifti(dialog_pickfile())
dims = size(img[*,*,3], /dimensions)
s = size(img, /dimensions)
n_slices = 2   ;s[2]
masks = fltarr(dims[0], dims[1], s[2])
for I = 0,n_slices-1  do begin
    xroi, img[*,*,I], regions_out = roi, /Block
    mask = roi -> ComputeMask(Dimensions = dims, Mask_Rule=2)
    masks[*,*,I] = mask
    endfor
```

```
end
END
```

Then, to run it, type this at the IDL command line:

```
IDL> .RUN twodmask
```

If you want to see how to get data into or out of a procedure or function, you could examine just about any IDL program you can download from the Internet. Most of them do it. XROI, for example.

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.")
