
Subject: Building Voxel arrays from png files (compression, analysis, and visualization)

Posted by [Greener](#) on Mon, 22 Aug 2005 17:51:47 GMT

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

Hello all,

I have no previous programming experience, prior to this I was a biology student with some computing background. I am working on speeding up the below script and trying to reduce the amount of data lost during compression. Currently I am relying on the IDL Interpolation to create a new voxel array from a series of 2-dimensional images. I am constrained to 1.5 GB of RAM and I am using the two for loops in an attempt to reduce the amount of data in memory at any one time.

The program usually takes 4 days or more hours to run a set of 10,000 images compressed 4x, resulting in a voxel array of dimensions [2000,2000,2500]. The images are pictures of small animal organs and are being used to image the arterial networks.

Any suggestions would be greatly appreciated.

Sincerely,

David Errington

```
;
; Copyright (c) 2002-2003, Barlow Scientific Inc. All rights reserved.
; Unauthorized reproduction prohibited.
;+BETA
; NAME:
;   BSI_VolFile::buildVol
;
; PURPOSE:
;   To build a volFile from a stack of images
;
; CALLING SEQUENCE:
;   Typically this method should be called from the objects init
method,
;   but it could also be called later.
;   result = volFile->buildVol()
;
; OPTIONAL INPUTS:
;   FIRSTFILE: The path to the first image in the sequence
;   ex: 'c:\data\VasMap\030303\gr\0001.png'
;
;   PARAMETERS: The sampling parameters. This is a named structure
{sampling} that
;   can be found in BSI_VolFile__define.pro.
;   The structure has 3 tags, pixel_space, slice_space, and
```

```

voxel_space
;   ex:  {sampling, 8.3, 9.6, 9.6}

;   FILENAME: The path where the new .vol file will be written
;   ex: 'c:\data\VasMap\030303\anal\one.vol'
;
;
; KEYWORDS:
; /NOPROGRESS: Supress the standard progress bar.
;
; /NOCONFIRM:  Supress the user conformation that shows final vol
file size before
;   it is written to disk
;
;
; OUTPUTS:
;   This function returns 0 or 1 indicating success or failure.
;
;
; RESTRICTIONS:
;   If any of the optional arguments are not provided then a widget
will be presented
;   to the user to collect those arguments.
;
;
; EXAMPLE:
;   example1, calling buildVol from the init method
;   volFile = OBJ_NEW('BSI_VolFile', /BUILD)
;
;   In example 2 no widgets will appear for this method since all the
parameters
;   are being passed in
;
;   example2:
;   sampling = {sampling, 12.3, 12.5, 12.5}
;   volFile = OBJ_NEW('BSI_VolFile', /BUILD, FIRSTFILE='c:\001.png',
SETTINGS=sampling, FILENAME=c:\one.vol', /NOCONFIRM, /NOPROGRESS)
;
;   example3:
;   volFile = OBJ_NEW('BSI_VolFile', /BLANK)
;   result = volFile->buildVol()
;
;   example4:
;   sampling = {sampling, 12.3, 12.5, 12.5}
;   volFile = OBJ_NEW('BSI_VolFile', /BLANK)
;   result = volFile->buildVol(FIRSTFILE='c:\001.png',
SETTINGS=sampling, FILENAME=c:\one.vol', /NOCONFIRM, /NOPROGRESS))
;
;
; MODIFICATION HISTORY:
;   Written by: Michael Finch, August 2003
; -BETA , David Errington

```

```

    out_fp = STRMID(file, 0, STRPOS(file, '\', /REVERSE_SEARCH)) +
'\
EndElse

WIDGET_CONTROL, /HOURGLASS

file_type = StrLowerCase(STRMID(file,2, /REVERSE_OFFSET))

files = FINDFILE(out_fp+'*.'+file_type)
n_files = N_ELEMENTS(files)

IF (files[0] EQ "") THEN RETURN, 0

IF (n_files LE 2L) THEN BEGIN
    ans = DIALOG_MESSAGE(['At least three files must be selected for
loading.'], /ERROR)
    RETURN, 0
ENDIF

; Put in alphanumeric order.
files = files[sort(files)]

; if no output filename was provided, ask for one
If n_elements(vol_file) EQ 0 Then Begin
    sys = OBJ_NEW('BSI_System')
    vol_file = sys->pickFile(FILTER='*.vol', TITLE='Select output
volume file', /WRITE)
    OBJ_DESTROY, sys

    WIDGET_CONTROL, /HOURGLASS

    IF (vol_file[0] EQ "") THEN RETURN, 0

;STRLOWCASE function returns a copy of String converted to lowercase
characters. $
;This avoids errors resulting from upper and lower case file name
conflicts

    ext_pos = STRPOS(STRLOWCASE(vol_file), '.vol', /REVERSE_SEARCH)
    IF (ext_pos LT 0L) THEN vol_file = vol_file + '.vol'

EndIf

; if no sampling parameters were provided, ask for them

If n_elements(sampling) EQ 0 Then Begin

```

; Get sampling parameters from user.

```
sampling = BSI_SAMPLE_VOL()
IF n_tags(sampling) EQ 0 THEN RETURN, 0
ENDIF
WIDGET_CONTROL, /HOURGLASS
```

```
dx = sampling.pixel_space
dy = sampling.pixel_space
dz = sampling.slice_space
dv = sampling.voxel_space
```

; Read first image.
; CASE selects one, and only one, statement for execution, \$
; depending on the value of an expression(i.e file name)
; SIZE returns size and type information for its argument if no \$
; keywords are set. If a keyword is set, SIZE returns the specified
quantity.

```
CASE file_type OF
  'jpg': READ_JPEG, files[0], img
  'png': img = READ_PNG(files[0])
ENDCASE
sz_img = SIZE(img)
xdim = sz_img[1]
ydim = sz_img[2]
zdim = n_files
```

```
x_samples = LONG(FLOAT(xdim) * dx / dv)
y_samples = LONG(FLOAT(ydim) * dy / dv)
z_samples = LONG(FLOAT(zdim) * dz / dv)
```

```
x_ramp = dv * FINDGEN(x_samples) / dx
y_ramp = dv * FINDGEN(y_samples) / dy
z_ramp = dv * FINDGEN(z_samples) / dz
```

; confirm the file size with the user unless explicitly told not to.

```
If n_elements(noconfirm) EQ 0 Then Begin
```

; Confirm file size.

```
dm_text = 'Output volume will be ' + $
```

```

        STRTRIM(x_samples, 2) + ' by ' + $
        STRTRIM(y_samples, 2) + ' by ' + $
        STRTRIM(z_samples, 2) + ' voxels ' + $
        '(' + STRTRIM(x_samples*y_samples*z_samples, 2) + '
bytes)'

```

```

        stat = DIALOG_MESSAGE([dm_text, 'Is this OK ?'], /QUESTION)
        WIDGET_CONTROL, /HOURGLASS
        IF (stat NE 'Yes') THEN RETURN, 0
    EndIf

```

; Write .hdr file.

```

    GET_LUN, lun1

    vol_file_hdr = vol_file + '.hdr'
    OPENW, lun1, vol_file_hdr
    PRINTF, lun1, x_samples
    PRINTF, lun1, y_samples
    PRINTF, lun1, z_samples
    PRINTF, lun1, dv
    CLOSE, lun1

```

; Set up the sub_vol array txt file, it will be saved in the folder \$
; where the original images were stored
CD, out_fp

```

    GET_LUN, lun2
    SubVol= 'SubVol'+ '.txt'
    OPENW, lun2, SubVol
    Close, lun2

```

; Start output.
 GET_LUN, lun3
 OPENW, lun3, vol_file

; start up the progress bar unless told not to.

```

    If n_elements(noprogress) EQ 0 Then Begin
        progressBar = Obj_New("PROGRESSBAR", COLOR='ORANGE',$
        /NOCANCEL, TITLE='Volume Creation Progress')
        progressBar -> Start
    EndIf

```

; Q is a defined variable that dictates the size of the file storage
array \$
; based on the slice space (dz)and the voxel output size(dv).

$Q = (dv/dz)$

```
; Establish the parameters that define the last file in the stack to  
access $  
; this avoids an I/O error
```

```
last = (n_files/Q)
```

```
; Create a loop that that creates an array (sub_vol) and files $  
; it with images
```

```
FOR i = 0L, last-1 DO BEGIN  
  z_int = z_ramp[i] - FLOAT(LONG(z_ramp[i]))  
; Create an empty array that can hold the original images  
; BYTARR sets every element of the result to zero. NOZERO keyword is  
set, $  
; this zeroing is not performed (array elements contain random values)  
$  
; and BYTARR executes faster.  
  sub_vol = BYTARR(xdim, ydim, Q)
```

```
FOR j = 0L, Q-1 DO BEGIN  
  a = (i*Q+j)  
  sub_vol [0,0,j] = READ_PNG(files [a])
```

```
;Append the subvol.txt file  
S_VS= SIZE (sub_vol [*,*,*])  
S_VM= MEAN (sub_vol [*,*,*])  
OPENU, lun2, SubVol, /APPEND  
PRINTF, lun2,
```

```
_____  
PRINTF, lun2, 'Time=', + systime()  
PRINTF, lun2, 'File name=', +files[a]  
PRINTF, lun2, 'a=', +a  
PRINTF, lun2, 'sub_vol Read size =', +S_VS  
PRINTF, lun2, 'sub_vol Mean=', +S_VM  
PRINTF, lun2, 'z_int=', +z_int  
PRINTF, lun2,
```

```
'===== '  
CLOSE, lun2
```

```
ENDFOR
```

```
new_img = INTERPOLATE  
(sub_vol[*,*,*],x_ramp,y_ramp,z_int,/GRID)  
WRITEU, lun3, new_img
```

```

        If n_elements(noprogress) EQ 0 Then Begin
            progressBar -> Update, (100*(i+1))/z_samples,
TEXT=STRING(i+1, FORMAT='(i0)') + ' of ' + $
            STRING(z_samples, FORMAT='(i0)') + ' slices
processed...'
        EndIf

```

```

NI_S= SIZE (new_img)
NI_M= MEAN (new_img)
NI_T= TOTAL (new_img)

```

```

ENDFOR

```

```

; destroy the progress bar if required

```

```

    If n_elements(noprogress) EQ 0 Then Begin
        progressBar -> Destroy
    EndIf

```

```

CLOSE, lun3
FREE_LUN, lun3

```

```

;ASSOC associates an array structure with a file

```

```

    GET_LUN, lunVol
    OPENU, lunVol, vol_file
    image = ASSOC(lunVol, BYTARR(xdim,ydim))

```

```

IS= size (image)
; Write to the subvol txt file
OPENU, lun2, SubVol, /APPEND
PRINTF, lun2, 'Associated image size=', +IS
PRINTF, lun2,
'+++++'
CLOSE, lun2
FREE_LUN, lun2

```

```

    self.image = PTR_NEW(image)
    self.xdim = xdim
    self.ydim = ydim
    self.zdim = zdim
    self.dv = dv
    self.filename = vol_file
    self.isInMemory = 'FALSE'

```

```
; end file writing  
  Return, 1
```

```
END
```
