Subject: Reclicing image volumes
Posted by VU KHAC Tri on Wed, 05 May 1999 07:00:00 GMT
View Forum Message <> Reply to Message

Hi folks,
I need a function which reclicing an image volume (3D) which is not isotropic.
This function should not compute an isotropic volume before reslicing it. The function directly reslices the isotropic volume.
Does anyone write this function ?
Thank you.
Tri.

Subject: Re: Reclicing image volumes
Posted by David Foster on Tue, 11 May 1999 07:00:00 GMT
View Forum Message <> Reply to Message

Tri VU KHAC wrote:
>
> Hi folks,
> I need a function which reclicing an image volume (3D) which is not
> isotropic.
> This function should not compute an isotropic volume before reslicing
> it. The function directly reslices the isotropic volume.
> Does anyone write this function ?
> Thank you.
> Tri.

Tri -

With a little hesitation, I am attaching my RESLICE.PRO and RESLICE.DOC files. This routine is intended for medical images, so the units for parameters may seem a bit strange or limiting. Make sure you read the documentation file first before using this.

This routine is by no means "ready for prime-time". This isn't to say that it doesn't work, because it does. But we use it here for a very specific purpose, and I'm a bit embarrassed at how "non-general" it is. But it will get you off to a good start.

You can see how you have to adjust the Z coordinates in your 2D section before interpolating, to account for the anisotropic nature of the voxels. You can also see how you have to be careful to apply 90-degree rotations *first* when doing exchange-of-axis rotations.

This routine uses code from IDL's EXTRACT_SLICE.PRO, which assumes isotropic voxels and doesn't handle 90-degree rotations correctly.

I hope this helps.

Dave
--

~~~~~~ ~~~~~~~
David S. Foster Univ. of California, San Diego
Programmer/Analyst Brain Image Analysis Laboratory
foster@bial1.ucsd.edu Department of Psychiatry
(619) 622-5892 8950 Via La Jolla Drive, Suite 2240
La Jolla, CA 92037
~~~~~~ ~~~~~~~

```
;   RESLICE.PRO   3-31-98   DSFoster
;
; Routine to reslice brain into image which is perpendicular to the Z plane
; defined by the rotation angles supplied, and a specified distance "up"
; this Z plane. First computes the X,Y,Z coordinates for each pixel in
; matrix/section form by inverse rotation/translation, then reads pixels
; from volume supplied  . Must supply the original midpoints and
; angles of rotation, the desired Z level (in mm and relative to angles
; given), the image to fill.
; Rotation angles passed are in degrees or, using keyword RAD, in radians,
; and are relative to the matrix-section coordinate system.
;
; NOTE:  We have learned that images that are "256x192" are in fact
; transformed into 256x256 format somewhere before they are stored
; onto tape. This means that the voxel is isotropic in the X and Y
; direction; therefore no correction needs to be made regarding
; "stretched" pixels within each image. However, the variables
; necessary to compute this correction, should it ever be necessary,
; are still read from a parameter file using READ_PARAMETER() (FOV_CM,
; XDIM and YDIM)
;
; If you want to get slices at angles relative to the "brain-centered"
; coordinate system, just start with rotation angles for the brain relative
; to the image coordinate system, and then add your desired offsets.
;
; Use the INTERP keyword to specify which method of interpolation to use:
;   0: Nearest neighbor sampling (the default)
;       1: Trilinear interpolation (IDL's INTERPOLATE)
;       2: Trilinear interp with zero-correction (our INTERP_3D from
;             IMAGE_PROC_IDL.SO sharable module.
;
; This routine was adapted from RESLICE.FOR, a Fortran routine which
; resliced volumes on the PC.
;
```

```
; The algorithm for extracting the slice was borrowed from
; EXTRACT_SLICE.PRO, an IDL User's Library function, to speed things up.
;
; Modifications
;
;  3-22-94 DSF If OUT_VAL=-1 is flag to set out_val to zero. Flag required
;           since saying "...,OUT_VAL=0" disables keyword.
; 10-24-94 DSF Fix bug that led to strange behavior at the top and bottom
;           of the volume (z=0,z=zdim).
;  6-01-95 DSF Enable interpolation algorithm from sharable C module
;           IMAGE_PROC_IDL.SO . Calls routine INTERP_3D() using
;           call_external().
;        DSF Remove CUBIC keyword, change SAMPLE keyword to INTERP.
;  9-18-96 DSF Add Z_BASE keyword to account for a coordinate
;           system where Z begins at 1 (not 0). This was necessary
;           to correct the inconsistency in MRORIENT.PRO that the
;           X,Y coordinates were 0-based but Z, which is in section
;           units, is 1-based. This led to all reslicing being off
;           by one section; this was discovered when we noticed that
;           the resliced 3D's were 2mm "behind" the FSE's.
; 12-18-96 DSF Use updated code from EXTRACT_SLICE.PRO (IDL User Routine)
;           for defining VOL_IND array, to speed things up.
;        DSF Added AXIS_ROTATION to enable the exchange of axis
;           necessary when creating one plane from another (eg. coronal
;           from sagittal). You cannot encorporate this exchange into
;           the three offsets because once the axes are exchanged
;           (say by a Y rotation of 90 degrees) then subsequent rotations
;           are with respect to the wrong axes. This exchange of axes
;           must be done after the three offsets.
;  4-01-97 DSF Call wrapper routine INTERP_3D.PRO which calls
;           C interpolation routines using CALL_EXTERNAL() (when using
;           zero-corrected trilinear interpolation INTERP=2).
; 12-15-97 DSF Allow only one element of AXIS_ROTATION to be nonzero.
;  3-23-98 DSF MAJOR BUG FIX: didn't treat Coronal -> Sagittal transformation
;           correctly. Use of AXIS_ROTATION was for Sagittal -> Coronal
;           transform, and produced correct results only because the
;           X and Z rotation angles saved to the .pa were switched
;           when the .pa was created by MrOrient (I misunderstood how
;           Jim's routine in TRANSFORM_UTILS.PRO was working. The angles
;           computed for Sag scan are relative to the Sag orientation,
;           NOT the Cor orientation!). The AXIS_ROTATION keyword failed
;           for Cor -> Sag because the 90-degree Y rotation exchanges
;           the X and Z angles unless the Y rotation is done FIRST.
;
;           Now only a single angle of absolute value near 90 is supported.
;           When one occurs, this rotation is applied first so that the
;           remaining angles are applied with respect to the correct axes.
;
```

```
;             It is assumed that the 3D .pa's have been "corrected", by
;             having their X and Z rotation angles exchanged, before
;             using this routine.
;
;*************************************************** ****************


FUNCTION reslice, volume, x0,y0,z0, rotx,roty,rotz, zlocation, param_file, $
   RADIANS=radians, OUT_VAL=out_val, FOV=fov, $
   INTERVAL=interval, INTERP=interp, Z_BASE=z_base


; Check for correct number of required arguments

if (keyword_set(FOV) and keyword_set(INTERVAL)) then begin
 argexp = 8
endif else begin
 argexp = 9
endelse

if (n_params() ne argexp) then begin
 message, 'Incorrect number of arguments (' + strtrim(argexp,2) + $
   ' expected)', /continue
 return, -1
endif

if ( keyword_set( Z_BASE ) ) then begin          ; Is Z coord 0 => ? or 1 => ?
 if ( z_base ne 1 ) then begin
  message, 'Value for keyword Z_BASE must be 1 when defined', /continue
  return, -1
 endif
endif else begin
 z_base = 0
endelse

if (not keyword_set(INTERP)) then $
 interp = 0                              ; Default to sampling

; Get dimension and data-type information on volume, and create
;  array that will hold resliced image (unless INTERP=2).

info = size(volume)
if (info(0) ne 3) then begin
 message, "First argument must be three-dimensional array",/CONTINUE
 return, -1
endif
xdim = info(1)                           ; Get dimensions of volume
ydim = info(2)
```

```
zdim = info(3)
type = info(4)                          ; Data type

; Check data types and create image to return, unless INTERP=2.
;  Currently only supports BYTE or INT volumes. Define constants.

case (type) of
 1: begin
  if ( interp ne 2 ) then $
   slice = bytarr(xdim,ydim,/nozero)
  zero = 0B
  one = 1B
  end
 2: begin
  if ( interp ne 2 ) then $
   slice = intarr(xdim,ydim,/nozero)
  zero = 0
  one = 1
  end
 else: begin
  message, "Volume data type must be BYTE or INT", /CONTINUE
  return, -1
  end
endcase

if (keyword_set(OUT_VAL)) then begin          ; Initialize image matrix
 if (out_val(0) eq -1) then begin          ; Flag for value of 0
  sample_out_val = zero                ;  (one's will be removed
  interp_out_val = one                 ;   from image if interpolation
  if ( interp ne 2 ) then $            ;   is used)
   slice(*) = zero
 endif else begin
  case (type) of
   1: begin
    sample_out_val = byte( out_val(0) )
    interp_out_val = byte( out_val(0) )
    if ( interp ne 2 ) then $
     slice(*) = byte( out_val(0) )
    end
   2: begin
    sample_out_val = fix( out_val(0) )
    interp_out_val = fix( out_val(0) )
    if ( interp ne 2 ) then $
     slice(*) = fix( out_val(0) )
    end
  endcase
 endelse
endif else begin
```

```
 if ( interp ne 2 ) then $
  slice(*) = zero
endelse

; Convert angular measures to degrees if /RADIANS specified

if (keyword_set(RADIANS)) then begin
 rot_x = (rotx * !PI)/180.0
 rot_y = (roty * !PI)/180.0
 rot_z = (rotz * !PI)/180.0
endif else begin
 rot_x = rotx
 rot_y = roty
 rot_z = rotz
endelse

; Get required information from file containing image parameters.
; If keywords FOV and INTERVAL are set with appropriate values already
; then skip this.

if (keyword_set(FOV) and keyword_set(INTERVAL)) then begin
 fov_cm = float(fov)
 interval = float(interval)
endif else begin
 interval = READ_PARAMETER(param_file, "SECTION_INTERVAL", /FLT)
 fov_cm = READ_PARAMETER(param_file, "FIELD_OF_VIEW", /FLT)
 info1 = size(interval)
 info2 = size(fov_cm)
 if (info1(1) eq 2 or info2(1) eq 2) then begin   ; If INT then error!
  message, 'Unable to get volume parameters from file ' + param_file, $
   /continue
  print, '   Missing parameters: SECTION_INTERVAL, FIELD_OF_VIEW'
  return,-1
 endif
endelse

; We want to allow for fully anisotropic pixels, so we must standardize
; our units of measure for the X,Y,Z coordinates. To do this, we assume
; the units for the X direction to be the standard, and then adjust
; the Y and Z accordingly. For the Y direction this means simply
; multiplying by the ratio of the X dimension of the original images to
; the Y dimension. For the Z direction, we calculate the number of
; X-pixels/section = (mm/section) / (mm/pixel)  ( = INTERVAL/ZMM_PIX )

zmm_pix = (fov_cm/float(xdim)) * 10.0  ; mm/pixel (using X units)
zcorr = interval/zmm_pix               ; Convert from Z-section => X-pixel units
ycorr = float(xdim)/float(ydim)        ; Convert from Y-pixel to X-pixel units
```

```
; Since Z location is in millimeters adjust to "image" units (0,1,2,3...).
; This is the position of the desired image plane relative to the rotation
;  angles provided.

zlocsec = zlocation / interval

; Create one-dimensional matrix which will hold the X,Y,Z coordinates
; for the slice; these coordinates will be transformed using !P.T .
; (Borrowed from IDL User's Library routine EXTRACT_SLICE.PRO . Code
; commented out is from earlier version -- speed enhancement.)

im_size = long( xdim ) * ydim

; Statement replaces next 5 commented lines below, and is a speed enhancement.

vol_ind = [ [reform( (findgen(xdim) # replicate(1.0, ydim)), im_size )], $
      [reform( (replicate(1.0, xdim) # findgen(ydim)), im_size )], $
      [replicate( 0.0, im_size )], $
   [replicate( 1.0, im_size )] ]

; index = lindgen( im_size )
; vol_ind = fltarr((im_size), 4, /NOZERO)
; vol_ind(*,3) = 1.0

vol_ind(*,2) = zlocsec * zcorr               ; Z => X pixel units

; vol_ind(*,1) = float(long(index / xdim))
; vol_ind(*,0) = float(index - (long(vol_ind(*,1)) * xdim))

; Compute transformation matrix using T3D function.
; First translate coordinates to "center" of VOL_IND 2D image.
;  Now do rotation to get coordinates about matrix-section
;  axes. If the rotations involve an exchange-of-axes (angle
;  about 90 degrees) then perform this rotation *first* so
;  subsequent rotations are about the correct axes.
;  Then translate about the brain origin to give final matrix/section
;  coordinates (account for whether the Z origin coordinate is 0-based
;  or 1-based). Scale the Z dimension to account for section thickness.

save_pt = !P.T
T3D, /RESET                          ; Reset, translate to image center
T3D, TRANSLATE = [ -(float(xdim-1L)/2.0), -(float(ydim-1L)/2.0), 0.0 ]

; Allow only one angle of rotation near (+/-) 90 degrees.
;  If one exists, apply that exchange-of-axis rotation first so
;  that the remaining angles are applied to the appropriate axes.

abs_rot = abs( [rot_x, rot_y, rot_z] )
```

```
obtuse = where(abs_rot ge 75.0)
if (n_elements(obtuse) gt 1) then begin
 msg = 'Only one rotation > 90-degrees supported'
 message, msg, /continue
 return, -1
endif else begin
 case (obtuse(0)) of
  -1: begin                          ; All acute angles
   T3D, ROTATE = [0.0,  0.0, rot_z]
   T3D, ROTATE = [0.0, rot_y,  0.0]
   T3D, ROTATE = [rot_x, 0.0,  0.0]
   end
  0: begin                           ; Obtuse X rotation
   T3D, ROTATE = [rot_x, 0.0,  0.0]
   T3D, ROTATE = [0.0, rot_y,  0.0]
   T3D, ROTATE = [0.0,  0.0, rot_z]
   end
  1: begin                           ; Obtuse Y rotation
   T3D, ROTATE = [0.0, rot_y,  0.0]
   T3D, ROTATE = [0.0,  0.0, rot_z]
   T3D, ROTATE = [rot_x, 0.0,  0.0]
   end
  2: begin                           ; Obtuse Z rotation
   T3D, ROTATE = [0.0,  0.0, rot_z]
   T3D, ROTATE = [0.0, rot_y,  0.0]
   T3D, ROTATE = [rot_x, 0.0,  0.0]
   end
 endcase
endelse

T3D, SCALE =  [1.0, 1.0, 1.0/zcorr]         ; Adjust for slice thickness
case ( z_base ) of                          ; Does Z start at 0 or 1?
 0: T3D, TRANSLATE = float( [x0, y0, z0] )
 1: T3D, TRANSLATE = float( [x0, y0, (z0 - 1.0)] )
endcase

vol_ind(*,*) = vol_ind(*,*) # !P.T(*,*)    ; Transform coordinates

; Now fill the resliced image array

case (interp) of
 0: begin
  slice(*) = volume(0 > vol_ind(*,0) < (xdim-1),  $
                    0 > vol_ind(*,1) < (ydim-1),  $
            0 > vol_ind(*,2) < (zdim-1))
  if (n_elements(out_val) ne 0) then begin
   out_v = where((((vol_ind(*,0) LT 0.0) OR  $
      (vol_ind(*,0) GE xdim)) OR  $
```

```
        ((vol_ind(*,1) LT 0.0) OR  $
       (vol_ind(*,1) GE ydim))) OR  $
        ((vol_ind(*,2) LT 0.0) OR  $
       (vol_ind(*,2) GE zdim)))
   if (out_v(0) GE 0L) then $
    slice(index(out_v)) = sample_out_val
  endif
  end
 1: begin
  if (Keyword_set(out_val)) then begin
   slice(*) =  $
    interpolate(volume, vol_ind(*,0), vol_ind(*,1), vol_ind(*,2), $
      missing = interp_out_val)
   if (out_val(0) eq -1) then begin     ; Flag for 0 so remove 1's
    zeroes = where(slice eq 1)
    slice(zeroes) = zero
   endif
  endif else begin
   slice(*) = interpolate(volume, vol_ind(*,0), vol_ind(*,1), $
     vol_ind(*,2))
  endelse
  end
 2: begin

  ; Call wrapper function that uses CALL_EXTERNAL() to call
  ;  routines from C module IMAGE_PROC_IDL.SO .

  status = 1

  slice = interp_3d( volume, vol_ind(*,0), vol_ind(*,1), $
    vol_ind(*,2), /zero_corrected, status=status )

  if ( status ne 0 ) then begin
   message, 'Internal error within IMAGE_PROC_IDL.SO', /continue
   return, -1
  endif else if (slice(0) lt 0) then begin
   message, 'Error: INTERP_3D.PRO, module IMAGE_PROC_IDL.SO', $
     /CONTINUE
   print, '  Invalid argument or module not found.'
   return, -1
  endif
  end
endcase

!P.T = save_pt

return, slice
END
```

NAME:
RESLICE

PURPOSE:
This function returns a 2-D planar slice extracted from
3-D volumetric data. The slicing plane may be oriented at
any angle, and may pass through any desired location in the
volume. Nearest neighbor sampling is the default sampling
technique. The user may specify that trilinear interpolation
be used, or a specialized form of trilinear interpolation in
which zero-valued voxels are not used to compute the interpolates.
The field-of-view (FOV) and distance between images for the imaging
sequence are required, and may be specified
using keywords or contained in a configuration file
(specified as an optional parameter).

CALLING SEQUENCE:

Slice = RESLICE(Vol, X_center, Y_center, Z_center, $
X_rot, Y_rot, Z_rot, Z_location [, param_file])

INPUTS:

Vol: The three dimensional volume of data to slice.
Data type : BYTE or INT.
X_center: The X coordinate (index) of the "origin" of the volume.
Data type : Any scalar numeric value (usually Long).
Y_center: The Y coordinate (index) of the "origin" of the volume.
Data type : Any scalar numeric value (usually Long).
Z_center: The Z coordinate (index) of the "origin" of the volume.
Data type : Any scalar numeric value (usually Long).
X_rot: The orientation (X rotation) of the slicing plane.
Before transformations, the slicing plane is parallel
to the X-Y plane. The slicing plane transformations
are performed in the following order :
1. Rotate Z_rot degrees about the Z axis.
2. Rotate Y_rot degrees about the Y axis.
3. Rotate X_rot degrees about the X axis.
4. Perform any 90-degree rotations that result
in an exchange of axes.
5. Translate the center of the plane to
X_center, Y_center, Z_center.
6. Move the image along the "new" Z axis
Z_LOCATION units.
Data type : Float.
Y_rot: The orientation (Y rotation) of the slicing plane.

Data type : Float.

Z_rot: The orientation (Z rotation) of the slicing plane.
Data type : Float.

Z_location: The location along the "new" Z direction (relative to the rotation angles provided) from the origin where the center of the image is located. If zero, the image is centered at the origin; otherwise, it is centered a distance out from this origin in the direction defined by X_rot, Y_rot and Z_rot. The units are millimeters.
Data type: float.

Param_file: A file containing information regarding the field-of-view of the images of the volume and the distance between images. Data type: String. See the example below.

KEYWORD PARAMETERS:

FOV: Set this and the INTERVAL keyword to avoid the need for a parameter file. This specifies the field-of-view of the original images in the volume, which is the width of the volume in the X dimension. Units are cm.
Data type: Float.

INTERP: Specifies the type of sampling to use to compute the resliced images. Possible values are:

0: Nearest neighbor sampling (the default)
1: Trilinear interpolation using IDL's INTERPOLATE()
2: Trilinear interpolation which does not use the value zero to compute interpolate values. Avoids aliases near borders with background voxels having value zero.

Note: Method #2 Requires INTERP_3D() from sharable C library module IMAGE_PROC_IDL.SO .

INTERVAL: The distance between sections. Set this and FOV to avoid using a parameter file. Units are mm.
Data type: Float.

OUT_VAL: If OUT_VAL is set, then the portions of the returned slice that lie outside the original volume are set to the value passed to OUT_VAL. To set outlying pixels to 0 set OUT_VAL=-1 (since OUT_VAL=0 "unsets" the keyword!). Data type : Any scalar numeric value (usually the same type as Vol).

RADIANS: Set this keyword to a non-zero value to indicate that X_rot, Y_rot, and Z_rot are in radians. The default

is degrees. Data type : Int.

OUTPUTS:

This function returns the planar slice as a two dimensional array with the same data type as Vol. The dimensions of the returned array are the same as the X- and Y-dimensions of the volume.

EXAMPLE:

Display an oblique slice through volumetric data.

```
; Create a volume from an image sequence (requires MAKEVOL)
  ret = MAKEVOL('/spare/im/892413*.mr', vol)

; Extract and display a slice. Outlying pixels to zero.
  slice = RESLICE(vol, 128.0, 128.0, 17.0, 3.0, -20.5, 0.0,  $
      30.0, param_file, OUT_VAL=-1)
  TVSCL, REBIN(slice, 400, 400)

; Command when not using parameter file:
  slice = RESLICE(vol, 128.0, 128.0, 17.0, 3.0, -20.5, 0.0,  $
      30.0, OUT_VAL=-1, FOV=24.0, INTERVAL=4.0)

; Create a coronal image from a sagittal volume:
  slice = RESLICE(vol, 128.0, 128.0, 62.0, 0.0, 90.0, 0.0, $
      0.0, OUT_VAL=-1, FOV=24.0, INTERVAL=1.2
```

EXAMPLE PARAMETER FILE:

```
SECTION_INTERVAL: 4.0;
FIELD_OF_VIEW: 24.0;
```

The parameter file must be in a format which can be read using READ_PARAMETER routine.

See Also: MAKEVOL, READ_PARAMETER

File Attachments

1) reslice.pro, downloaded 95 times
2) reslice.doc, downloaded 100 times