Subject: Rotation of 3D Array
Posted by davidf on Wed, 10 Sep 1997 07:00:00 GMT
View Forum Message <> Reply to Message

Folks,

Christophe Morisset has no access to the IDL newsgroup these days, but he asks me to post the following program for him in hope that it will save someone else time and effort.

Chris had need to rotate a 3D array, but there was no built-in IDL routine for this purpose, so he wrote his own and offers it here.

If you want to reach Chris with a question or comment about the routine, or if you just want to thank him for his contribution, you can reach him at morisset@agn1.iagusp.usp.br.

Cheers,

David

---------- -----
David Fanning, Ph.D.
Fanning Software Consulting
Customizable IDL Programming Courses
Phone: 970-221-0438 E-Mail: davidf@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com

********** ******

```
Function TURN_3D, a, x_angle, y_angle, z_angle, INTERP = interp, $
 MISSING = missing, PIVOT = pivot, CUBIC = cubic, $
 RESIZE = resize, CONSERV = conserv, VERBOSE = verbose
;+
; NAME:
; Turn_3D
;
; CALLING SEQUENCE:
; result = Turn_3D(a, x_angle, y_angle, z_angle)
;
; PURPOSE:
; Rotate a 3D array. It applies the ROT IDL function to each
; 2D subarray of A. The computation is done in a 50% bigger
; cube to assure that no information is lost.
;
; INPUT PARAMETERS:
; A = The 3D array to be rotated. This array may be of any type
```

```
;     except string and structure.
;
; X_ANGLE, Y_ANGLE, Z_ANGLE = Angles of rotation in degrees
;         CLOCKWISE.
;
; KEYWORDS:
; INTERP, MISSING, PIVOT, and CUBIC will be passed to ROT:
;
; INTERP: Set this keyword for bilinear interpolation.  If this
;  keyword is set to 0 or omitted, nearest neighbor sampling is
;  used. Note that setting this keyword is the same as using the
;  ROT_INT User Library function.  This change (and others)
;  essentially makes ROT_INT obsolete.
;
; CUBIC: If specified and non-zero, "Cubic convolution"
;  interpolation is used.  This is a more
;  accurate, but more time-consuming, form of interpolation.
;  CUBIC has no effect when used with 3 dimensional arrays.
;  If this parameter is negative and non-zero, it specifies the
;  value of the cubic interpolation parameter as described
;  in the INTERPOLATE function. Valid ranges are -1 <= Cubic < 0.
;  Positive non-zero values of CUBIC (e.g. specifying /CUBIC)
;  produce the default value of the interpolation parameter
;  which is -1.0.
;
;   MISSING: The data value to substitute for pixels in the output
;  image that map outside the input image.
;
;   PIVOT: Setting this keyword causes the image to pivot around the
;  point (X0,Y0), so that this point maps into the same point
;  in the output image. If this keyword is set to 0 or omitted,
;  then the point (X0,Y0) in the input image is mapped into the
;  center of the output image.
;
; RESIZE: Setting this keyword to resize the result to the maximum
;  size (x,y or z-one) of A. The resizing is NOT a rebining,
;  it extracts a 3D sub-array of the big 3D array in which
;  the computation is done.
;
; CONSERVE: Set this keyword to assure that TOTAL(result)=TOTAL(A).
;
; VERBOSE: Setting this keyword will print the ratio of the
;  sizes of the input array and the result. Works only if
;  RESIZE not set.
;
; LIMITATIONS: They are those of ROT... For small dimensions arrays,
;       a rotation of +10 deg followed by a rotation of -10 deg
;       will NOT give you back the input data.
```

```
;
; AUTHOR:
; Christophe MORISSET, 1997. morisset@iagusp.usp.br
;-

if (size(a))(0) ne 3 then stop,' A must be 3D'

x_size = (size(a))(1)
y_size = (size(a))(2)
z_size = (size(a))(3)

max_size = x_size > y_size > z_size

; let's do a 50% larger 3D array containing the input
; 3D array at his "center"

new_size = fix(max_size*1.5) + 1
b = congrid(a*0.,new_size,new_size,new_size)

b[(new_size-x_size)/2:(new_size-x_size)/2+x_size-1,$
(new_size-y_size)/2:(new_size-y_size)/2+y_size-1,$
(new_size-z_size)/2:(new_size-z_size)/2+z_size-1] = a

; X-rotation
if x_angle ne 0. then begin
 for x = 0,new_size-1 do b[x,*,*] = rot(reform(b[x,*,*]),x_angle,$
   INTERP = interp,MISSING = missing,PIVOT = pivot,CUBIC = cubic)
endif

; Y-rotation
if y_angle ne 0. then begin
 for y = 0,new_size-1 do b[*,y,*] = rot(reform(b[*,y,*]),y_angle,$
   INTERP = interp,MISSING = missing,PIVOT = pivot,CUBIC = cubic)
endif

; Z-rotation
if z_angle ne 0. then begin
 for z = 0,new_size-1 do b[*,*,z] = rot(reform(b[*,*,z]),z_angle,$
   INTERP = interp,MISSING = missing,PIVOT = pivot,CUBIC = cubic)
endif

if keyword_set(resize) then b = $
  b[(new_size-x_size)/2:(new_size-x_size)/2+x_size-1,$
  (new_size-y_size)/2:(new_size-y_size)/2+y_size-1,$
  (new_size-z_size)/2:(new_size-z_size)/2+z_size-1] $
else if keyword_set(verbose) then $
 print,' Size changed by: ',float(new_size) / float(max_size)
```

```
if keyword_set(conserv) then b = b / total(b) * total(a)

return,b

end
```