Subject: Re: arbitrary rotation of 3-d arrays
Posted by D. Mattes on Tue, 15 Jun 1999 07:00:00 GMT
View Forum Message <> Reply to Message

IDL users: thanks for the lively discussion regarding this topic. i never did find a canned procedure for me to use, but i implemented a 3D volume rotation class method using VERT_T3D and INTERPOLATE. i submit it here for anybody who would like it, and for suggestions for optimization and improvement.

cheers,
david mattes

```
------------------CUT HERE------------------------------
pro ImageClass::ApplyRigidTransformation,U,t
    ;U is a 3x3 unitary rotation matrix
    ;t is a 3x1 vector of x,y,z translations

    ;self is the local object reference with members:
    ;   self.xdim, self.ydim, self.zdim  (size of volume)
    ;   self.volume  (pointer to 3D volume data)

    ;build 4x4 homogeneous transformation matrix
   localT=fltarr(4,4)
   localT(0:2,0:2)=U
   localT(3,0:2)=t
   localT(3,3)=1.

    ;build x,y,z interpolation points
   vert_size=LONG(self.xdim)*$
      LONG(self.ydim)*$
      LONG(self.zdim)
   verts=fltarr(3,vert_size)
   count=0L
   for i=0,self.zdim-1 do begin
      for j=0,self.ydim-1 do begin
         for k=0,self.xdim-1 do begin
            ;notice index swapping here!!!!!!!!!!!!
            ;for our image, the x coordinate is the most
            ;quickly varying, so it must also be this way for the
            ;interpolate locations
           verts(*,count)=[k,j,i]
           count=count+1
         endfor
      endfor
   endfor

    ;verts are the location points for which we want
```

```
    ;interpolates...really just the indices in the volume array.
    ;we transform these indices using the localT transformation
    ;matrix, and pass them to the interpolate function.
    ;notice: verts is 3 x (n*m*l) 2D matrix for a n x m x l array!!!
   verts=VERT_T3D(verts,MATRIX=localT,/NO_COPY,/NO_DIVIDE)

    ;right now, use missing=-1 for values outside input data range.
    ;cubic interpolation is not supported for 3-d case
   *self.volume=INTERPOLATE(TEMPORARY(*self.volume),$
      verts(0,*),verts(1,*),verts(2,*),$
      MISSING=-1)

    ;the interpolate function returns a 1-d array of interpolated
    ;points, which must be resized into the original array shape.
   *self.volume=REFORM(*self.volume,$
      self.xdim,$
      self.ydim,$
      self.zdim,/OVERWRITE)

end
```