
Subject: Re: Rotate volumes

Posted by [Martin Downing](#) on Mon, 17 Sep 2001 00:09:20 GMT

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

Hi Bob,

Is this code any help or have I missed the point?

```
=====
function transform_image3d, im, rotation = rot,
scale=scale,translate=translate, centre_rot=centre_rot
; translate an image volume using interplote
s = size(im)
; for clarity:
sx=s(1)
sy=s(2)
sz=s(3)
if undefined(rot) then rot =[0,0,0]
if undefined(centre_rot) then centre_rot =[(sx-1)/2.0,(sy-1)/2.0,(sz-1)/2.0]
if undefined(translate) then translate =[0,0,0]
if undefined(scale) then scale =[1,1,1]
;generate image coordinates
i = lindgen(sx*sy*sz) ; temp array = vector indices
coords = [ [i mod sx],[i / sx mod (sy)],[i /
(sx*sy)],[replicate(1,sx*sy*sz)]]
; generate transform (or add your own)
t3d, /reset
t3d,trans= -centre_rot
t3d, rot=rot
t3d, trans= centre_rot + translate
t3d, scale=scale
; calc new sample positions of voxels
coords = coords#!p.t
; use these to interpolate voxels (note this is only SAMPLED)
imageT = reform( interpolate(im, coords(*,0), coords(*,1), coords(*,2),
missing=0),sx,sy,sz)
return, imageT
end

pro test,s, rot=rot
; tests the above
if undefined(s) then s = 32
if undefined(rot) then rot = [0,10,45]
vol = rebin(bytscl(dist(s,s)),s,s,s, /sample)
vol = transform_image3d(vol, rot = rot)
; Display volume:
XVOLUME, vol
end
=====
```

you could easily add your own transform as an input arg
cheers

Martin

--

Martin Downing,
Clinical Research Physicist,
Orthopaedic RSA Research Centre,
Woodend Hospital, Aberdeen, AB15 6LS.
Tel. 01224 556055 / 07903901612
Fax. 01224 556662

m.downing@abdn.ac.uk

"B.C. Hamans" <s448443@stud.tue.nl> wrote in message
news:9o2j63\$44e\$1@news.tue.nl...

> Hi,

>

> I'm still working on my volumes (see previous posting) and trying to
rotate

> and translate them to match each other. It would be very nice if I could
use

> something like XVOLUME_ROTATE, /T3D or /MATRIX=!P.T. (Of course this isn't
> possible). I also thought about using CONVERT_COORD but this is no
solution

> either (i think). The 2 volumes are described by a matrix of dimension
> 256x256x256 containing gray values between 0 and 255. I obtain a
translation

> matrix to fit the 2 images from an external program. In the future i hope
to

> do this by using MIM or MIM2 (<http://www.nuclear.uhrad.com/mim2.htm>). The
> translation matrix is of the form !P.T (4x4).

>

> I already made some nice projections of the volumes using PROJECT_VOL in 3
> directions and would like to add some sliders to define rotation,
> translation and skew factors. To align the volumes before further
processing

> them.

>

> Anybody?

>

> Bob

>

>

Subject: Re: Rotate volumes

Posted by [Martin Downing](#) on Mon, 17 Sep 2001 09:28:25 GMT

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

Ouch - that code caused rather a large amount of memory to be used for your 256 cubes - try this which allows you to break the job into feasible chunks
A 64cube takes 0.6sec, but 256cube takes 50sec per transform on my laptop though (which is not good as the inner function of a registration!!!! :(

good luck anyhow

Martin

```
=====
function transform_image3d, image, rotation = rot, $
    scale=scale,translate=translate, centre_rot=centre_rot, chunks=chunks,$
    t3dmat=t3dmat
; transform a 3d image volume
s = size(image)
sx=s(1) & sy=s(2) & sz=s(3)
st = sx*sy*sz
imageT = image
; get transform matrix
if undefined(t3dmat) then begin
    if undefined(rot) then rot = [0,0,0]
    if undefined(centre_rot) then $
        centre_rot=[(sx-1)/2.0,(sy-1)/2.0,(sz-1)/2.0]
    if undefined(translate) then translate = [0,0,0]
    if undefined(scale) then scale = [1,1,1]
    t3d, /reset,trans= -centre_rot
    t3d, rot=rot, trans= centre_rot + translate, scale=scale
    t3dmat = !p.t
endif
; do transformation
if undefined(chunks) then chunks = 1
iter = st/chunks
for i0 = 0L, (st-1), iter do begin
    ; account for possible odd last chunk
    bufsize = iter < (st-i0)
    ;generate image coordinates
    i = i0+lindgen(bufsize) ; temp array = vector indices
    coords = [ [(i mod sx)],[((i / sx) mod (sy))],$
                [(i /(sx*sy))],[replicate(1b, bufsize)]]
    coords = (coords#t3dmat)
    imageT[i0:i0+bufsize-1] = interpolate(image, coords(*,0), coords(*,1),$
        coords(*,2), missing=0)
endfor
return, imageT
end
```

```

pro test,s, rot=rot, chunks=chunks

if undefined(s) then s = 256
if undefined(chunks) then chunks = 32
if undefined(rot) then rot = [20,10,45]
vol = rebin(bytescl(dist(s,s)),s,s,s, /sample)
t0= systime(1)
vol = transform_image3d(vol, rot = rot, chunks=chunks)
t1= systime(1)
print, "done in ", t1-t0, " sec"
; Display volume:
XVOLUME, vol

end
=====

```

Martin Downing,
Clinical Research Physicist,
Orthopaedic RSA Research Centre,
Woodend Hospital, Aberdeen, AB15 6LS.

"Martin Downing" <martin.downing@ntlworld.com> wrote in message
news:Aabp7.24911\$Pm5.5585206@news2-win.server.ntlworld.com.. .

```

> Hi Bob,
> Is this code any help or have I missed the point?
>
> =====
> function transform_image3d, im, rotation = rot,
> scale=scale,translate=translate, centre_rot=centre_rot
> ; translate an image volume using interplote
> s = size(im)
> ; for clarity:
> sx=s(1)
> sy=s(2)
> sz=s(3)
> if undefined(rot) then rot =[0,0,0]
> if undefined(centre_rot) then centre_rot
> =[(sx-1)/2.0,(sy-1)/2.0,(sz-1)/2.0]
[cut]

```

Subject: Re: Rotate volumes

Posted by [Martin Downing](#) on Mon, 17 Sep 2001 12:02:12 GMT

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

Further experimentation shows that performance (on my w2000 machine at least) is optimal for a loop size which gives a buffer of 128, ie

chunks = st/128

Why 128 is the magic number I haven't a clue, but it makes performance of order(n^3) for cube sizes 64(0.37sec) through to 512(195sec)! I normally avoid optimisation

issues like the plague, but I wonder how much other code would benefit from this kind of approach?

Martin

[Hm must stop answering my own mailings]

ps undefined() is just:

```
FUNCTION UnDefined , x
  s = size(x)
  RETURN , (s(0)+1) EQ 0 )
END
```

"Martin Downing" <martin.downing@ntlworld.com> wrote in message news:lmjp7.26942\$Pm5.5877321@news2-win.server.ntlworld.com..

> Ouch - that code caused rather a large amount of memory to be used for your

> 256 cubes - try this which allows you to break the job into feasible chunks

> A 64cube takes 0.6sec, but 256cube takes 50sec per transform on my laptop > though (which is not good as the inner function of a registration!!!! :(

>

> good luck anyhow

>

> Martin

>

> =====

> function transform_image3d, image, rotation = rot, \$
> scale=scale,translate=translate, centre_rot=centre_rot,
chunks=chunks,\$

> t3dmat=t3dmat

> ; transform a 3d image volume

> s = size(image)

> sx=s(1) & sy=s(2) & sz=s(3)

> st = sx*sy*sz

> imageT = image

> ; get transform matrix

> if undefined(t3dmat) then begin

> if undefined(rot) then rot =[0,0,0]

> if undefined(centre_rot) then \$

> centre_rot=[(sx-1)/2.0,(sy-1)/2.0,(sz-1)/2.0]

> if undefined(translate) then translate =[0,0,0]

> if undefined(scale) then scale =[1,1,1]

```

> t3d, /reset,trans= -centre_rot
> t3d, rot=rot, trans= centre_rot + translate, scale=scale
> t3dmat = !p.t
> endif
> ; do transformation
> if undefined(chunks) then chunks =1
> iter = st/chunks
> for i0 = 0L, (st-1), iter do begin
> ; account for possible odd last chunk
> bufsize = iter < (st-i0)
> ;generate image coordinates
> i = i0+lindgen(bufsize) ; temp array = vector indices
> coords = [ [(i mod sx)],[(i / sx) mod (sy))],$,
>             [(i / (sx*sy))],[replicate(1b, bufsize)]]
> coords = (coords#t3dmat)
> imageT[i0:i0+bufsize-1] = interpolate(image, coords(*,0), coords(*,1),$
>             coords(*,2), missing=0)
> endfor
> return, imageT
> end
>
> pro test,s, rot=rot, chunks=chunks
>
> if undefined(s) then s = 256
> if undefined(chunks) then chunks = 32
> if undefined(rot) then rot = [20,10,45]
> vol = rebin(bytescl(dist(s,s)),s,s,s, /sample)
> t0= systime(1)
> vol = transform_image3d(vol, rot = rot, chunks=chunks)
> t1= systime(1)
> print, "done in ", t1-t0, " sec"
> ; Display volume:
> XVOLUME, vol
>
> end
> =====
>
> -----
> Martin Downing,
> Clinical Research Physicist,
> Orthopaedic RSA Research Centre,
> Woodend Hospital, Aberdeen, AB15 6LS.
>
>
> "Martin Downing" <martin.downing@ntlworld.com> wrote in message
> news:Aabp7.24911$Pm5.5585206@news2-win.server.ntlworld.com..
>> Hi Bob,
>> Is this code any help or have I missed the point?

```

```

>>
>> =====
>> function transform_image3d, im, rotation = rot,
>> scale=scale,translate=translate, centre_rot=centre_rot
>> ; translate an image volume using interplote
>> s = size(im)
>> ; for clarity:
>> sx=s(1)
>> sy=s(2)
>> sz=s(3)
>> if undefined(rot) then rot =[0,0,0]
>> if undefined(centre_rot) then centre_rot
> =[ (sx-1)/2.0,(sy-1)/2.0,(sz-1)/2.0]
> [cut]
>
>
>
>
>
>
>

```

Subject: Re: Rotate volumes

Posted by [B.C. Hamans](#) on Mon, 17 Sep 2001 18:36:18 GMT

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

Hi Martin,

thanks for all the support. Your code works just fine. I was in the hospital all day gathering some more data and did not get a chance to implement the code until this evening. Tomorrow I will have some more time to code a bit more and hopefully complete the first phase of the project. I can't check the performance of the different code bits on my own computer because it's an oldy. I will check it tomorrow at my work on a faster workstation.

Thanks again,

Bob

P.S. I will post a link to the full program and project when i'm done.

"Martin Downing" <martin.downing@ntlworld.com> wrote in message

news:NC1p7.27174\$Pm5.5984194@news2-win.server.ntlworld.com.. .

> Further experimentation shows that performance (on my w2000 machine at
> least) is optimal for a loop size which gives a buffer of 128, ie

> chunks = st/128

> Why 128 is the magic number I havent a clue, but it makes performance of

> order(n^3) for cube sizes 64(0.37sec) through to 512(195sec)! I normally

```

> avoid optimisation
> issues like the plague, but I wonder how much other code would benefit
from
> this kind of approach?
>
> Martin
> [Hm must stop answering my own mailings]
> ps undefined() is just:
> FUNCTION UnDefined , x
>   s = size(x)
>   RETURN , (s( s(0)+1) EQ 0 )
> END
>
>
> "Martin Downing" <martin.downing@ntlworld.com> wrote in message
> news:lmjp7.26942$Pm5.5877321@news2-win.server.ntlworld.com..
>> Ouch - that code caused rather a large amount of memory to be used for
> your
>> 256 cubes - try this which allows you to break the job into feasible
> chunks
>> A 64cube takes 0.6sec, but 256cube takes 50sec per transform on my
laptop
>> though (which is not good as the inner function of a registration!!!! :(
>>
>> good luck anyhow
>>
>> Martin
>>
>> =====
>> function transform_image3d, image, rotation = rot, $
>>   scale=scale,translate=translate, centre_rot=centre_rot,
> chunks=chunks,$
>>   t3dmat=t3dmat
>> ; transform a 3d image volume
>> s = size(image)
>> sx=s(1) & sy=s(2) & sz=s(3)
>> st = sx*sy*sz
>> imageT = image
>> ; get transform matrix
>> if undefined(t3dmat) then begin
>>   if undefined(rot) then rot = [0,0,0]
>>   if undefined(centre_rot) then $
>>     centre_rot=[(sx-1)/2.0,(sy-1)/2.0,(sz-1)/2.0]
>>   if undefined(translate) then translate = [0,0,0]
>>   if undefined(scale) then scale = [1,1,1]
>>   t3d, /reset,trans= -centre_rot
>>   t3d, rot=rot, trans= centre_rot + translate, scale=scale
>>   t3dmat = !p.t

```

```

>> endif
>> ; do transformation
>> if undefined(chunks) then chunks =1
>> iter = st/chunks
>> for i0 = 0L, (st-1), iter do begin
>>   ; account for possible odd last chunk
>>   bufsize = iter < (st-i0)
>>   ;generate image coordinates
>>   i = i0+lindgen(bufsize) ; temp array = vector indices
>>   coords = [ [(i mod sx)],[(i / sx) mod (sy)],$
>>               [(i /(sx*sy))],[replicate(1b, bufsize)]]
>>   coords = (coords#t3dmat)
>>   imageT[i0:i0+bufsize-1] = interpolate(image, coords(*,0),
coords(*,1),$
>>       coords(*,2), missing=0)
>>   endfor
>>   return, imageT
>> end
>>
>> pro test,s, rot=rot, chunks=chunks
>>
>> if undefined(s) then s = 256
>> if undefined(chunks) then chunks = 32
>> if undefined(rot) then rot = [20,10,45]
>> vol = rebin(bytescl(dist(s,s)),s,s,s, /sample)
>> t0= systime(1)
>> vol = transform_image3d(vol, rot = rot, chunks=chunks)
>> t1= systime(1)
>> print, "done in ", t1-t0, " sec"
>> ; Display volume:
>> XVOLUME, vol
>>
>> end
>> =====
>>
>> -----
>> Martin Downing,
>> Clinical Research Physicist,
>> Orthopaedic RSA Research Centre,
>> Woodend Hospital, Aberdeen, AB15 6LS.
>>
>>
>> "Martin Downing" <martin.downing@ntlworld.com> wrote in message
>> news:Aabp7.24911$Pm5.5585206@news2-win.server.ntlworld.com..
>>> Hi Bob,
>>> Is this code any help or have I missed the point?
>>>
>>> =====

```

```

>>> function transform_image3d, im, rotation = rot,
>>> scale=scale,translate=translate, centre_rot=centre_rot
>>> ; translate an image volume using interplote
>>> s = size(im)
>>> ; for clarity:
>>> sx=s(1)
>>> sy=s(2)
>>> sz=s(3)
>>> if undefined(rot) then rot =[0,0,0]
>>> if undefined(centre_rot) then centre_rot
>> =[ (sx-1)/2.0,(sy-1)/2.0,(sz-1)/2.0]
>> [cut]
>>
>>
>>
>>
>>
>>
>
>
>
>
>
>
>
>
>
>

```

Subject: Re: Rotate volumes

Posted by [marc schellens\[1\]](#) on Tue, 18 Sep 2001 08:04:47 GMT

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

"B.C. Hamans" wrote:

```

>
> Hi,
>
> I'm still working on my volumes (see previous posting) and trying to rotate
> and translate them to match each other. It would be very nice if I could use
> something like XVOLUME_ROTATE, /T3D or /MATRIX=!P.T. (Of course this isn't
> possible). I also thought about using CONVERT_COORD but this is no solution
> either (i think). The 2 volumes are described by a matrix of dimension
> 256x256x256 containing gray values between 0 and 255. I obtain a translation
> matrix to fit the 2 images from an external program. In the future i hope to
> do this by using MIM or MIM2 (http://www.nuclear.uhrad.com/mim2.htm). The
> translation matrix is of the form !P.T (4x4).
>
> I already made some nice projections of the volumes using PROJECT_VOL in 3
> directions and would like to add some sliders to define rotation,

```

> translation and skew factors. To align the volumes before further processing
> them.
>
> Anybody?
>
> Bob

If Martin's function does what you want:

I had a similar problem some time ago,
this solutions seemed to be faster (even with loops), less
resource-hungry
and you get an intepolated result.

hope it helps,
:-) marc

MRI2 it the bytarr(256,256,256)
phi fltarr(3) the three angles to rotate (+/- is convention)
trans intarr(3) the translation (in voxels)

```
print,'X...'
if phi[0] ne 0.0 then begin
  for x=0,255 do begin

    MRI2[x,*,*]=rot(/INTERP,reform(MRI2[x,*,*],256,256),-phi[0], MISSING=0)
  endfor
endif
print,'Y...'
if phi[1] ne 0.0 then begin
  for y=0,255 do begin

    MRI2[* ,y,*]=rot(/INTERP,reform(MRI2[* ,y,*],256,256),phi[1],MISSING=0)
  endfor
endif
print,'Z...'
if phi[2] ne 0.0 then begin
  for z=0,255 do begin
    MRI2[* ,*,Z]=rot(/INTERP,MRI2[* ,*,Z],-phi[2],MISSING=0)
  endfor
endif

print,'shift...'
MRI2=shift(MRI2,trans[0],trans[1],trans[2])
```

Subject: Re: Rotate volumes

Posted by [Richard Tyc](#) on Tue, 18 Sep 2001 14:09:18 GMT

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

Just curious, are you using IDLgrVolume objects to render your data ? If you were, I would think any type of interactive manipulation would be PAINFULLY slow for a 256x256x256 object. I regularly manipulate MR generated objects of 60x60x40 (original MR image slices reduced in size!) size on a dual processor machine (HINTS set to take advantage of multi-processor) and 3D rotations (I used a trackball object) are rendered every few seconds.

Rich

B.C. Hamans <s448443@stud.tue.nl> wrote in message
news:9o2j63\$44e\$1@news.tue.nl...

> Hi,

>

> I'm still working on my volumes (see previous posting) and trying to rotate

> and translate them to match each other. It would be very nice if I could use

> something like XVOLUME_ROTATE, /T3D or /MATRIX=!P.T. (Of course this isn't possible). I also thought about using CONVERT_COORD but this is no solution

> either (i think). The 2 volumes are described by a matrix of dimension 256x256x256 containing gray values between 0 and 255. I obtain a translation

> matrix to fit the 2 images from an external program. In the future i hope to

> do this by using MIM or MIM2 (<http://www.nuclear.uhrad.com/mim2.htm>). The translation matrix is of the form !P.T (4x4).

>

> I already made some nice projections of the volumes using PROJECT_VOL in 3 directions and would like to add some sliders to define rotation, translation and skew factors. To align the volumes before further processing

> them.

>

> Anybody?

>

> Bob

>

>

Subject: Re: Rotate volumes

Marc,

Thats a neat way of doing it - just watch out for the effect of wrap-around if you use translate (caused by the use of SHIFT) on matching two images. The method I posted actually uses linear interpolation (I misread the help!) and is a little quicker (20sec c.f. 29sec) for a general rotation on a PIII 800). The rotation orders are not the same but could easily be made so.

cheers

Martin

"Marc Schellens" <m_schellens@hotmail.com> wrote in message
news:3BA7001F.BF7A8DF8@hotmail.com...

> "B.C. Hamans" wrote:

>>

>> Hi,

>>

>> I'm still working on my volumes (see previous posting) and trying to rotate

>> and translate them to match each other. It would be very nice if I could use

>> something like XVOLUME_ROTATE, /T3D or /MATRIX=!P.T. (Of course this isn't

>> possible). I also thought about using CONVERT_COORD but this is no solution

>> either (i think). The 2 volumes are described by a matrix of dimension

>> 256x256x256 containing gray values between 0 and 255. I obtain a translation

>> matrix to fit the 2 images from an external program. In the future i hope to

>> do this by using MIM or MIM2 (<http://www.nuclear.uhrad.com/mim2.htm>).

The

>> translation matrix is of the form !P.T (4x4).

>>

>> I already made some nice projections of the volumes using PROJECT_VOL in 3

>> directions and would like to add some sliders to define rotation,

>> translation and skew factors. To align the volumes before further processing

>> them.

>>

>> Anybody?

>>

>> Bob

>

```
> If Martin's function does what you want:
>
> I had a similar problem some time ago,
> this solutions seemed to be faster (even with loops), less
> resource-hungry
> and you get an intepolated result.
>
> hope it helps,
> :-) marc
>
>
> MRI2 it the bytarr(256,256,256)
> phi fltarr(3) the three angles to rotate (+/- is convention)
> trans intarr(3) the translation (in voxels)
>
> print,'X...'
> if phi[0] ne 0.0 then begin
>   for x=0,255 do begin
>
>     MRI2[x,*,*]=rot(/INTERP,reform(MRI2[x,*,*],256,256),-phi[0], MISSING=0)
>   endfor
> endif
> print,'Y...'
> if phi[1] ne 0.0 then begin
>   for y=0,255 do begin
>
>     MRI2[*,y,*]=rot(/INTERP,reform(MRI2[*,y,*],256,256),phi[1],MISSING=0)
>   endfor
> endif
> print,'Z...'
> if phi[2] ne 0.0 then begin
>   for z=0,255 do begin
>     MRI2[*,*,z]=rot(/INTERP,MRI2[*,*,z],-phi[2],MISSING=0)
>   endfor
> endif
>
> print,'shift...'
> MRI2=shift(MRI2,trans[0],trans[1],trans[2])
```

Subject: Re: Rotate volumes

Posted by [marc schellens\[1\]](#) on Wed, 19 Sep 2001 11:19:16 GMT

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

Martin Downing wrote:

```
>
> Marc,
```

>
> Thats a neat way of doing it - just watch out for the effect of wrap-around
> if you use translate (caused by the use of SHIFT) on matching two images.
> The method I posted actually uses linear interpolation (I misread the help!)
> and is a little quicker (20sec c.f. 29sec) for a general rotation on a PIII
> 800). The rotation orders are not the same but could easily be made so.
>
> cheers
>
> Martin
>

I used the attached test routine (just quickly put together for this test)
because I remembered, that then plane-wise method was faster for me
(IDL 5.4.1, Linux mandrake 8.0):

156^3 bytarr:
8s vs. 11s

256^3 bytarr:
35s vs. 184s

As I mentioned, the bigger the array, the larger your index list.
In the 256^3 case, my machine started swapping (with over 600MB free memory!)
which explains the huge time-difference in the second case.
With the shift you are right. For MRI images its most often ok to use
nevertheless,
as the head is surrounded by a nonused black frame.
I assume you use windows. Interesting, that the performance of the two
methods
is so much OS-dependent.

cheers,
:-) marc

```
function undefined,i  
return,n_elements(i) eq 0  
end
```

```
function transform2, im, rotation = phi,$  
scale=scale,translate=translate, centre_rot=centre_rot  
; translate an image volume using interplote  
s = size(im)  
; for clarity:
```

```

sx=s(1)
sy=s(2)
sz=s(3)
if undefined(phi) then phi =[0,0,0]
if undefined(centre_rot) then centre_rot
=[(sx-1)/2.0,(sy-1)/2.0,(sz-1)/2.0]
if undefined(translate) then translate =[0,0,0]
if undefined(scale) then scale =[1,1,1]

print,'X...'
if phi[0] ne 0.0 then begin
    for x=0,sx-1 do begin

        im[x,*,*]=rot(/INTERP,reform(im[x,*,*],sy,sz),-phi[0],MISSING G=0)
    endfor
endif
print,'Y...'
if phi[1] ne 0.0 then begin
    for y=0,sy-1 do begin

        im[*,y,*]=rot(/INTERP,reform(im[*,y,*],sx,sz),phi[1],MISSING =0)
    endfor
endif
print,'Z...'
if phi[2] ne 0.0 then begin
    for z=0,sz-1 do begin
        im[*,*,z]=rot(/INTERP,im[*,*,z],-phi[2],MISSING=0)
    endfor
endif

return,im
end

function transform1, im, rotation = rot,$
scale=scale,translate=translate, centre_rot=centre_rot
; translate an image volume using interplote
s = size(im)
; for clarity:
sx=s(1)
sy=s(2)
sz=s(3)
if undefined(rot) then rot =[0,0,0]
if undefined(centre_rot) then centre_rot
=[(sx-1)/2.0,(sy-1)/2.0,(sz-1)/2.0]
if undefined(translate) then translate =[0,0,0]
if undefined(scale) then scale =[1,1,1]
;generate image coordinates
i = lindgen(sx*sy*sz) ; temp array = vector indices

```

```

coords = [ [i mod sx],[i / sx) mod (sy)],[i
/(sx*sy)],[replicate(1,sx*sy*sz)]]
; generate transform (or add your own)
t3d, /reset
t3d,trans= -centre_rot
t3d, rot=rot
t3d, trans= centre_rot + translate
t3d, scale=scale
; calc new sample positions of voxels
coords = coords#!p.t
; use these to interpolate voxels (note this is only SAMPLED)
im = reform( interpolate(im, coords(*,0), coords(*,1), coords(*,2),$
missing=0),sx,sy,sz)
return, im
end

```

pro test

d=156

a=1.0

b=byte(indgen(d,d,d))

s=systime(1)

print,'ix'

r=transform1(b,rot=[a,a,a])

print,systime(1)-s

b=byte(indgen(d,d,d))

s=systime(1)

print,'planes'

r=transform2(b,rot=[a,a,a])

print,systime(1)-s

end