
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]
>>
>>
>>
>>
>>
>>
>
>
>
>
>

```

>
>
>
>
