Subject: Convol with Kernel Dependency Of the Radius to the Middle
Posted by bente on Tue, 23 Oct 2001 14:18:43 GMT
View Forum Message <> Reply to Message

Hi,

have again a nice Problem ;-)

I need a Convolution with a Kernel that depends on how far the Voxel
is away from the Middle Of the Array.

The Idee behind that is, that i have to simulate PET Pictures out of
MRI Datasets (im working my diploma in the Medical Imaging).

In the moment i have solved this Problem with convoluting on several
rings. But this isnï¿½t so smooth and i have to convolute the same
picture up to 10 times (with a duration of round about 1minute for
each (thanks to Jaco who made 1minute out of 45minutes i had before
;-).

So the question is, if someone has the listing of the convol routines
from IDL or any other idea???

I believe that if I try this alone i will result in many many FOR
loops.

with regards
Kay

Subject: Re: Convol with Kernel Dependency Of the Radius to the Middle
Posted by Martin Downing on Tue, 30 Oct 2001 13:40:37 GMT
View Forum Message <> Reply to Message

"Kay" <bente@uni-wuppertal.de> wrote in message
news:e143e8bc.0110230618.6339be6e@posting.google.com...
> Hi,
>
> have again a nice Problem ;-)
>
> I need a Convolution with a Kernel that depends on how far the Voxel
> is away from the Middle Of the Array.

I assume you mean that the function that describes the 3d kernel depends on
the radial (cartesian) distance of the image voxel from a point in the
image:
pseudo formula alert!

Image'(x,y,z) = Image [convolved_with] Kernel(R(x,y,z))
where R(x,y,z) = sqrt((X-Xo)^2+(Y-Yo)^2 + (Z-Zo)^2))

If so, I guess the question is, what is the dependency of the kernel on R?
If linear then maybe the radial aspect of the kernel is separable

Martin

>
> The Idee behind that is, that i have to simulate PET Pictures out of
> MRI Datasets (im working my diploma in the Medical Imaging).
>
> In the moment i have solved this Problem with convoluting on several

> picture up to 10 times (with a duration of round about 1minute for
> each (thanks to Jaco who made 1minute out of 45minutes i had before
> ;-).
>
> So the question is, if someone has the listing of the convol routines
> from IDL or any other idea???
>
>
> I believe that if I try this alone i will result in many many FOR
> loops.
>
> with regards
> Kay

Subject: Re: Convol with Kernel Dependency Of the Radius to the Middle
Posted by bente on Wed, 31 Oct 2001 09:14:10 GMT
View Forum Message <> Reply to Message

Hi,
> I assume you mean that the function that describes the 3d kernel depends on
> the radial (cartesian) distance of the image voxel from a point in the
> image:
> *pseudo formula alert!*
> Image'(x,y,z) = Image [convolved_with] Kernel(R(x,y,z))
> where R(x,y,z) = sqrt((X-Xo)^2+(Y-Yo)^2 + (Z-Zo)^2))

Thats right.

> If so, I guess the question is, what is the dependency of the kernel on R?
> If linear then maybe the radial aspect of the kernel is separable

My Proffessor had the "nice" idea, that a PET image, has a better resolution in the middle of the picture than on the edge for each slice. To simulate this he wants that the Kernel has a dependency of x & y (not z!) in that form that the FWHM (Full Width Half Max) of a 3D Gaussian Kernel increases linear with the distance from the middle.

Thats a bit too much for my "weak" knowledge of IDL, I solved it in that form, that i convol the whole stuff 10 times and then copy several "barrels" together to the whole picture. It looks ok, but itï¿½s not as smooth as it should be.

I think I have to write a complete new convol function :-((
smile

thanks for the answer.

Kay

Subject: Re: Convol with Kernel Dependency Of the Radius to the Middle
Posted by Martin Downing on Wed, 31 Oct 2001 15:09:47 GMT
View Forum Message <> Reply to Message

Hi Kay,

I was wondering about a radial-proportioned linear combination of two gaussian smoothings, one for the edge and one for the middle - however the intermediate values are not a good approximation :(
I'm afraid I think you will have to code it from scratch, probably with 6 for loops(!).Convolutions are easy, just for loops. - I think I'd go straight to coding it as a callable C routine, as then you wont have to worry about the many loops, I would also not bother computing a kernel as it will change too often, instead create a lookup table for your gaussian's exponents.
e.g. : gauss_exp(sig = s, x = x) = GAUSS_LUT_100[100*(x*x)/(2*s*s)]

a simple IDL-convolve algorithm is given below, youll have to modify it to include the radial bit
good luck,
Martin

```
function eg_convol3d, array, kern, debug=debug
; a simple convolution routine

if keyword_set(debug) then t0 = systime(1)
b = array*0.
sa = size(array)
sk = size(kern)
```

```
; note: ignoring edges
for k = sk(3)-1, sa(3)-1 do begin
 for j = sk(2)-1, sa(2)-1 do begin
  for i = sk(1)-1, sa(1)-1 do begin

   v = 0
   for n = 0, sk(3)-1 do begin
    for m = 0, sk(2)-1 do begin
     for l = 0, sk(1)-1 do begin
       v = v + array[i-l, j-m, k-n]*kern[l, m, n]
     endfor
    endfor
   endfor

   b[i-sk(1)/2,j-sk(2)/2,k-sk(3)/2] = v

  endfor
 endfor
endfor

if keyword_set(debug) then print, "convol completed in ", systime(1) -  t0,
" sec
return, b

end
```

"Kay" <bente@uni-wuppertal.de> wrote in message
news:e143e8bc.0110310114.3cbb5c53@posting.google.com...
> Hi,
>> I assume you mean that the function that describes the 3d kernel depends
on
>> the radial (cartesian) distance of the image voxel from a point in the
>> image:
>> *pseudo formula alert!*
>> Image'(x,y,z) = Image [convolved_with] Kernel(R(x,y,z))
>> where R(x,y,z) = sqrt((X-Xo)^2+(Y-Yo)^2 + (Z-Zo)^2))
>
> Thats right.
>
>> If so, I guess the question is, what is the dependency of the kernel on
R?
>> If linear then maybe the radial aspect of the kernel is separable
>
> My Proffessor had the "nice" idea, that a PET image, has a better
> resolution in the middle of the picture than on the edge for each
> slice. To simulate this he wants that the Kernel has a dependency of x

> & y (not z!) in that form that the FWHM (Full Width Half Max) of a 3D
> Gaussian Kernel increases linear with the distance from the middle.
>
> Thats a bit too much for my "weak" knowledge of IDL, I solved it in
> that form, that i convol the whole stuff 10 times and then copy

> not as smooth as it should be.
>
> I think I have to write a complete new convol function :-((
> smile
>
> thanks for the answer.
>
> Kay

Subject: Re: Convol with Kernel Dependency Of the Radius to the Middle
Posted by bente on Fri, 02 Nov 2001 08:31:18 GMT
View Forum Message <> Reply to Message

Thanks,
I tried it in 2 dimensions at the moment (for the 3rd i can put this
little proggy in another for-loop.

Hereï¿½s my solution.
The version with 1D-Kernel is really fast, but i have to use a very
large Kernel (10times larger than the FWHM) otherwise i get ugly
stripes in the picture.

```
FUNCTION Gauss, s, sig, scale=scale, fwhm=fwhm
;Create 3D Gaussian
;Kay Bente, Wuppertal: 06.09.01 - 11.09.01

;s -> [nx,ny,nz]
;sigma -> Sigma for Gauss
;scale -> [sx,sy,sz] for example for MRI images e.g. : [0.9,0.9,1.25]
;fwhm=1 -> Caluculate with FWHM instead of sigma,
FWHM=2*sigma*Sqrt(2*ALog(2)) => sigma~FWHM/2.35
;       sigma input in FWHM

;Determine Dimensions
ss=Size(s, /Dimensions)


CASE ss(0) OF
        0 : GOTO, label_1D
        1 : GOTO, label_1D
```

```
       2 : GOTO, label_2D
       3 : GOTO, label_3D
    ELSE : BEGIN
              Print, 'Wrong Dimensions!'
              Print, 'Size(s, /Dimensions) Has To Be Less Or Equal
3'
              Print, 'You Entered :',s
              Print, String('That has ',StrTrim(String(ss),1), '
Dimensions!')
              Print, '"0" Returned!'
              Return, 0
              GOTO, label_END
           ENDELSE
ENDCASE

;1D
label_1D:

;Check Keywords
IF Keyword_Set(scale) EQ 0 THEN scale = [1]
IF Keyword_Set(fwhm) THEN sigma = sig/2.35 ELSE sigma = sig

;Create Indizes
arr  = (FindGen(s(0)) - s(0)/2)*scale(0)
;Create Gaussian on Indizes
arr2 = Exp(-(Temporary(arr))^2/(2.*sigma^2))/(Sqrt(2.*!PI)*sigma)
Return,arr2
GOTO, label_END

;2D Adapted from 3D
label_2D:

;Check Keywords
IF Keyword_Set(scale) EQ 0 THEN scale = [1,1]
IF Keyword_Set(fwhm) THEN sigma = sig/2.35 ELSE sigma = sig

;Faktor 1 for odd size
f=[s(0) mod 2, s(1) mod 2]

mid=[s(0)/2 + f(0),s(1)/2 + f(1)]
arr=FltArr(mid(0),mid(1))

fak=2.*sigma^2
fak2=2.*!pi*sigma^2

;Create 1/4  Circle
FOR i=0,mid(0)-1 DO BEGIN
 FOR j=0,mid(1)-1 DO BEGIN
```

```
  arr(i,j) = Exp(-((i*scale(0))^2+(j*scale(1))^2)/fak)/fak2
 ENDFOR
ENDFOR

arr2=FltArr(s(0),s(1))

;Copy & Mirror To Fill The Rest Of Circle
arr2(mid(0)-f(0):s(0)-1,mid(1)-f(1):s(1)-1)=arr
arr2(mid(0)-f(0):s(0)-1,0:mid(1)-1)=Reverse(arr,2)
 arr2(0:mid(0)-1,0:s(1)-1)=Reverse(arr2(mid(0)-f(0):s(0)-1,0: s(1)-1),1)

Return, arr2
GOTO, label_END

;3D
label_3D:

;Check Keywords
IF Keyword_Set(scale) EQ 0 THEN scale = [1,1,1]
IF Keyword_Set(fwhm) THEN sigma = sig/2.35 ELSE sigma = sig

;Faktor 1 for odd size
f=[s(0) mod 2, s(1) mod 2, s(2) mod 2]

mid=[s(0)/2 + f(0),s(1)/2 + f(1),s(2)/2 + f(2)]
arr=FltArr(mid(0),mid(1),mid(2))

fak=2.*sigma^2
fak2=(2.*!pi)^(3./2.)*sigma^3

;Create 1/8  Sphere
FOR i=0,mid(0)-1 DO BEGIN
 FOR j=0,mid(1)-1 DO BEGIN
  FOR k=0,mid(2)-1 DO BEGIN
   arr(i,j,k)=  Exp(-((i*scale(0))^2+(j*scale(1))^2+(k*scale(2))^2)/fak)/fak 2
  ENDFOR
 ENDFOR
ENDFOR

arr2=FltArr(s(0),s(1),s(2))

;Mirror Subarrays to create whole Sphere
;   3 2 1                1 2 3                0 0 3
;a= 0 0 2 -> Reverse(a,1)= 2 0 0     Reverse(a,2)= 0 0 2
;   0 0 3                3 0 0                3 2 1

 arr2(mid(0)-f(0):s(0)-1,mid(1)-f(1):s(1)-1,mid(2)-f(2):s(2)- 1)=arr
 arr2(mid(0)-f(0):s(0)-1,0:mid(1)-1,mid(2)-f(2):s(2)-1)=Rever se(arr,2)
```

```
 arr2(0:mid(0)-1,0:s(1)-1,mid(2)-f(1):s(2)-1)=Reverse(arr2(mi
d(0)-f(0):s(0)-1,0:s(1)-1,mid(2)-f(1):s(2)-1),1)
 arr2(0:s(0)-1,0:s(1)-1,0:mid(2)-1)=Reverse(arr2(0:s(0)-1,0:s (1)-1,mid(2)-f(2):s(2)-1),3)
Return, arr2

label_END:
END


FUNCTION Kernel_Size, scale, fac, fw

;Determines the Size of The Kernel from FWHM and a Faktor and the
Scaling of the Dataset (PIXEL <-> mm)
;e.g. scale = [0.9,0.9,1.25]
;      FWHM = 6mm
;       fac = 3  (Kernel should be 3times larger then the FWHM
;Kernel_Size= [21,21,15]


; written by Kay Bente
; 11.9.01

 s=Ceil(fac*fw/scale)

 ;Round Up to next largest Odd integer
 s=s+1-(s MOD 2)

Return, s
END



FUNCTION R_Convol, array, minn, maxx, scale=scale, one=one, two=two
;Convolutes an 2D array with a 1D-Gaussian with minn in the middle and
maxx as maximal FWHM
;one -> convol in one dimension  (sinogram)
;two -> convol in two dimensions (slice)
;needs Kernel_Size
;    Gauss

IF Keyword_Set(scale) EQ 0 THEN scale=[1.,1.]
s=Size(array, /Dimensions)
xx=s(0)
yy=s(1)

image=FltArr(xx,yy)
;-----------------OneDimensional----------------------
IF KeyWord_Set(one) THEN BEGIN
```

```
l_arr=FltArr(3*xx,yy)
l_arr(xx,0)=array

FOR t=0,xx-1 DO BEGIN
 r=Abs(xx/2.-t)       ;Get Radius
 fwhmm=minn+((maxx-minn)*r/(xx/2.))  ;Calculate FWHM
 ss=Kernel_Size(scale(0),10,fwhmm)  ;Calculate KernelSize
 kernel=Gauss(ss,fwhmm,/FWHM)   ;Build Kernel (1D)
 large_kernel=Rebin(kernel,ss(0),yy)  ;Duplicate Kernel in y-Dimension
  image(t,*)=Rebin(large_kernel*l_arr(t+xx-1-ss/2:t+xx-1+ss/2, *),1,yy)*ss
ENDFOR

ENDIF


;------------------TwoDimensional-----------------------
IF KeyWord_Set(two) THEN BEGIN

l_arr=FltArr(3*xx,3*yy)
l_arr(xx,yy)=array

FOR x=0,xx-1 DO BEGIN
FOR y=0,yy-1 DO BEGIN
 r=Sqrt((xx/2.-x)^2+(yy/2.-y)^2)   ;Get Radius
 fwhmm=minn+((maxx-minn)*r/(xx/2.))  ;Calculate FWHM
 ss=Kernel_Size(scale,3,fwhmm)      ;Calculate KernelSize
 kernel=Gauss(ss,fwhmm,/FWHM)   ;Build Kernel (1D)
  image(x,y)=Total(kernel*l_arr(x+xx-1-ss(0)/2:x+xx-1+ss(0)/2, y+yy-1-ss(1)/2:y+yy-1+ss(1)/2))
ENDFOR
ENDFOR

ENDIF


Return, image
END
```