
Subject: Re: Convolve 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).

Here's 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 -> Calculate 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)=Reverse(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

I_arr=FltArr(3*xx,3*yy)
I_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*I_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

```
