Subject: Re: gaussian blurring for images
Posted by asad on Mon, 22 May 2000 07:00:00 GMT
View Forum Message <> Reply to Message

I haven't actually tried this...but can't you just convolve a 3-d
gaussian kernel with your image?

Asad

> "D. Mattes" wrote:
>
> now i'm looking for an image processing routine so i don't have to write
> it myself. i want to apply a 3-d gaussian blurring filter to a 3-d image,
> so that i can decimate the image without introducing aliasing.
>
> does anybody out there have such a beast?

remove "!"s to mail

Subject: Re: gaussian blurring for images
Posted by Amara Graps on Mon, 22 May 2000 07:00:00 GMT
View Forum Message <> Reply to Message

"D. Mattes" wrote:

> now i'm looking for an image processing routine so i don't have to write
> it myself. i want to apply a 3-d gaussian blurring filter to a 3-d image,
> so that i can decimate the image without introducing aliasing.
>
> does anybody out there have such a beast?

Hi,

This isn't three-d, and it is not a Gaussian blurring, but perhaps you
can start with this, and modify it.

Here, I used Frank Varosi's psf_gaussian filter to design a high-pass
gaussian filter for images, calling it in the following way.

```
len = 256 ;256 by 256.. same as image
;Using Frank Varosi's Point-Spread Function, but want 0 in the middle, 1
outside
;(will be removing center-lowest frequecies)
filter_real = 1.0 - PSF_GAUSSIAN(Npixel=256, FWHM=25)
```

Attached below are the necessary functions. Hope that this helps.

Amara

```
 ============================================================
==================

function gaussian, xi, parms, pderiv
;+
; NAME:
; GAUSSIAN
;
; PURPOSE:
; Compute the 1-D Gaussian function and optionally the derivative
; at an array of points.
;
; CALLING:
; y = gaussian( xi, parms,[ pderiv ])
;
; INPUTS:
; xi = array, independent variable of Gaussian function.
;
; parms = parameters of Gaussian, 2 or 3 element array:
;  parms(0) = maximum value (factor) of Gaussian,
;  parms(1) = mean value (center) of Gaussian,
;  parms(2) = standard deviation (sigma) of Gaussian.
;  (if parms has only 2 elements then sigma taken from common).
;
; OPTIONAL OUTPUT:
; pderiv = optional output of partial derivatives,
;  computed only if parameter is present in call.
;
;  pderiv(*,i) = partial derivative at all xi absisca values
;  with respect to parms(i), i=0,1,2.
;
; Function returns array of Gaussian evaluated at xi.
;
; EXAMPLE:
; Evaulate a Gaussian centered at x=0, with sigma=1, and a peak value
; of 10 at the points 0.5 and 1.5.   Also compute the derivative
;
; IDL> f = gaussian( [0.5,1.5], [10,0,1], DERIV )
; ==> f= [8.825,3.25].   DERIV will be a 2 x 3 array containing the
; numerical derivative at the two point with respect to the 3
parameters.
;
; COMMON BLOCKS:
```

```
; common gaussian, sigma
; HISTORY:
; Written, Frank Varosi NASA/GSFC 1992.
;-
  common gaussian, sigma

 Nparmg = N_elements( parms )
 parms = float( parms )
 if (Nparmg GE 3) then sigma = parms(2)

 z = ( xi - parms(1) )/sigma
 zz = z*z
 gauss = fltarr( N_elements( zz ) )
 w = where( zz LT 172, nw )
 if (nw GT 0) then gauss(w) = exp( -zz(w) / 2 )

 if N_params() GE 3 then begin

  pderiv = fltarr( N_elements( xi ), Nparmg )
  fsig = parms(0) / sigma

  pderiv(0,0) = gauss
  pderiv(0,1) = gauss * z * fsig

  if (Nparmg GE 3) then  pderiv(0,2) = gauss * zz * fsig
    endif

return, parms(0) * gauss
end

function psf_gaussian, parameters, NPIXEL=npix, NDIMENSION=ndim,
FWHM=fwhm,  $
     CENTROID=cntrd, ST_DEV=st_dev,  $
     XY_CORREL=xy_corr, NORMALIZE=normalize
;+
; NAME:
; psf_Gaussian
;
; PURPOSE:
; Return a point spread function having Gaussian profiles,
; as either a 1D vector, a 2D image, or 3D volumetric-data.
;
; CALLING:
; psf = psf_Gaussian( NPIXEL=, FWHM= , [/NORMALIZE, /ST_DEV,  )
; or:
; psf = psf_Gaussian( parameters, NPIXEL =  )
;
; REQUIRED KEYWORDS:
```

```
; NPIXEL = number pixels for each dimension, specify as an array,
;  or just one number to make all sizes equal.
;
; OPTIONAL KEYWORDS:
;
; NDIMEN = dimension of result: 1 (vector), 2 (image), or 3 (volume),
;  default = 2 (an image result).
;
; FWHM = the desired Full-Width Half-Max (pixels) in each dimension,
;  specify as an array, or single number to make all the same.
;
; CENTROID = pixels numbers of PSF maximum ( 0.5 is center of a pixel ),
;  default is exact center of requested vector/image/volume.
;
; STDEV = optional way to specify width by standard deviation param.
;
; XY_CORREL = scalar between 0 and 1 specifying correlation coefficient
;  Use this keyword, for example, to specify an elliptical
;  gaussian oriented at an angle to the X,Y axis
;
; /NORMALIZE causes resulting PSF to be normalized so Total( psf ) = 1.
;
; INPUTS (optional):
;
; parameters = an NDIMEN by 3 array giving for each dimension:
;   [ maxval, center, stdev ],  overrides other keywords.
;
; EXAMPLE:
; Create a 31 x 31 array containing a normalized centered gaussian
; with an X FWHM = 4.3 and a Y FWHM = 3.6
;
; IDL> array = PSF_GAUSSIAN( Npixel=31, FWHM=[4.3,3.6], /NORMAL
;
; EXTERNAL CALLS:
; function Gaussian
;
; HISTORY:
; Written, Frank Varosi NASA/GSFC 1991.
;-
 On_error,2

 if (N_params() LT 1 ) and not keyword_set( FWHM) then begin
  print,'Syntax - psf = PSF_GAUSSIAN( parameters, NPIXEL = )
  print, $
 'or       psf = PSF_GAUSSIAN( FWHM = ,STDEV = ,NPIXEL = ,[CENTROID =
])'
  return, -1
 endif
```

```
sp = size( parameters )

if (sp(0) GE 1) then begin
 ndim = sp(0)
 factor = total( parameters(*,0) )/float( ndim )
 cntrd = parameters(*,1)
 st_dev = parameters(*,2)
   endif

if N_elements( ndim ) NE 1 then ndim=2
ndim = ndim>1

if N_elements( npix ) LE 0 then begin
 message,"must specify size of result with NPIX=",/INFO
 return,(-1)
  endif else if N_elements( npix ) LT ndim then $
  npix = replicate( npix(0), ndim )

if (N_elements( cntrd ) LT ndim) AND (N_elements( cntrd ) GT 0) then $
  cntrd = replicate( cntrd(0), ndim )

if N_elements( cntrd ) LE 0 then cntrd=(npix-1)/2. else cntrd=cntrd-0.5
if N_elements( fwhm ) GT 0 then st_dev = fwhm/( 2* sqrt( 2* aLog(2) ) )

if N_elements( st_dev ) LE 0 then begin
 message,"must specify ST_DEV= or FWHM=",/INFO
 return,(-1)
  endif

if N_elements( st_dev ) LT ndim then $
  st_dev = replicate( st_dev(0), ndim )
sigfac = 1 / (2. * st_dev^2 )

CASE ndim OF

1: BEGIN
 x = findgen( npix(0) ) - cntrd(0)
 psf = gaussian( x, [1,0,st_dev] )
     END

2: BEGIN
 psf = make_array( DIM=npix(0:ndim-1), /FLOAT )
 x = findgen( npix(0) ) - cntrd(0)
 y = findgen( npix(1) ) - cntrd(1)

 if N_elements( xy_corr ) EQ 1 then begin
  y2 = sigfac(1) * y^2
```

```
   x1 = sigfac(0) * x
   yc = y * ( xy_corr/(st_dev(0)*st_dev(1)) )
   for j=0,npix(1)-1 do begin
    zz = x * (yc(j) + x1) + y2(j)
    w = where( zz LT 86, nw )
    if (nw GT 0) then psf(w,j) = exp( -zz(w) )
     endfor
    endif else begin
   psfx = gaussian( x, [ 1, 0, st_dev(0) ] )
   psfy = gaussian( y, [ 1, 0, st_dev(1) ] )
   for j=0,npix(1)-1 do psf(0,j) = psfx * psfy(j)
     endelse
      END

 3: BEGIN
  psf = make_array( DIM=npix(0:ndim-1), /FLOAT )
  x = findgen( npix(0) ) - cntrd(0)
  y = findgen( npix(1) ) - cntrd(1)
  z = findgen( npix(2) ) - cntrd(2)
  psfx = gaussian( x, [ 1, 0, st_dev(0) ] )
  psfy = gaussian( y, [ 1, 0, st_dev(1) ] )
  psfz = gaussian( z, [ 1, 0, st_dev(2) ] )
  for k=0,npix(2)-1 do begin
      for j=0,npix(1)-1 do psf(0,j,k) = psfx * psfy(j) * psfz(k)
   endfor
      END

 ENDCASE

 if keyword_set( normalize ) then return, psf/total( psf )

 if N_elements( factor ) EQ 1 then begin
  if (factor NE 1) then return,factor*psf else return,psf
    endif else return, psf
end
```

--


 ************************************************************* ***
Amara Graps               | Max-Planck-Institut fuer Kernphysik
Interplanetary Dust Group | Saupfercheckweg 1
+49-6221-516-543          | 69117 Heidelberg, GERMANY
                      * http://galileo.mpi-hd.mpg.de/~graps
 ************************************************************* ***

      "Never fight an inanimate object." - P. J. O'Rourke