
Subject: Filtering based on Hessian Matrix
Posted by [stef](#) on Thu, 15 Jul 2004 19:34:40 GMT
[View Forum Message](#) <> [Reply to Message](#)

Hi there,

I'm posting here a wonderful filter (my humble opinion) to detect bright or dark lines, blobs etc..., mainly in a try to give back a little bit to the community. It's a great place here!

It is based on the eigenvalues of the Hessian Matrix (a matrix of partial secondary derivatives). It's currently standard procedure to use a filter similar to this prior to segmentation or vessel tracking (there are 20+ papers out there, google for 'vessel enhancement hessian matrix').

There's a demo included in the source code. I use the filter to detect the coronary vessels. I do have a 3D version of the filter, write me a mail if you're interested. However, the 3D filter is much slower, since the analytical solution for the eigenvalues of a 3 by 3 Matrix is not trivial. So I was forced to use a triple FOR loop (shame on me, other solutions highly appreciated) and then use LA_EIGENQL on each voxel. If anyone has a C-Coded version of this filter, I'd be glad to write an IDL Wrapper for it.

Have a great day,
Stefan

```
;+
; Copyright (c) Dept. Radiology, University of California San
; Francisco (UCSF)
; written by: Stefan Tuchschrnid, stefan.tuchschrnid@netpictures.ch
;
; NAME:
; CV_Vessel_Filter_2D
;
; PURPOSE:
; This function is a 2D version of the vesselness enhancement filter
; [A. Frangi, W. Niessen, K.L. Vincken, and M.A. Viergever
; Multiscale vessel enhancement filtering.
; Proc. MICCAI'98, pp.130-137, 1998. ]
; It is based on the eigenvalues of a Hessian matrix (a matrix of
; partial
; secondary derivatives)
;
; This version detects bright lines, but could easily be adapted to
```

```

; detect dark lines, or dark and brigh blobs, etc..
;
; CATEGORY:
; Filtering
;
;
; EXAMPLE:
;
; ;Read the image
; path=FILEPATH('md5290fc1.jpg',SUBDIR=['examples','data'])
; READ_JPEG, path, img
;
; ;Create window:
; WINDOW, 0, XSIZE=600, YSIZE=600
;
; ;Show original image
; TVSCL, img, 0
;
; ;Filter image
; filtered=CV_Vessel_Filter_2D(img)
;
; ;Show filtered image
; TVSCL, filtered, 1
;
;
; MODIFICATION HISTORY:
; Written by: Stefan Tuchschrnid, stefan.tuchschrnid@netpictures.ch

```

FUNCTION CV_Vessel_Filter_2D, image

```

image=FLOAT(image)

```

```

x_size=(SIZE(image, /DIMENSIONS))[0]
y_size=(SIZE(image, /DIMENSIONS))[1]

```

```

; Smooth image, kernel can be changed
; a larger kernel will detect bigger vessels
smooth_kernel=[[1,2,1],$
               [2,4,2],$
               [1,2,1]]

```

```

smooth_image=CONVOL(image, smooth_kernel, 16)

```

```

; Calculate the Hessian Matrix

```

```

x_deriv_kernel=[-1,0,1]
y_deriv_kernel=TRANPOSE([-1,0,1])

Lx=CONVOL(smooth_image, x_deriv_kernel)
Ly=CONVOL(smooth_image, y_deriv_kernel)

Lxy=CONVOL(Lx, y_deriv_kernel)

Lxx=CONVOL(Lx, x_deriv_kernel)
Lyy=CONVOL(Ly, y_deriv_kernel)

; Analytically solve for the eigenvalues
t1=(Lxx+Lyy)/2
t2=SQRT((Lxx+Lyy)^2- 4*(Lxx*Lyy-Lxy^2))/2

lambda_1 = REFORM(t1+t2,x_size*y_size)
lambda_2 = REFORM(t1-t2,x_size*y_size)

lambda_array=[[lambda_1],[lambda_2]]

; Sort the eigenvalues
dummy=MAX(ABS(lambda_array), max_subscript, DIMENSION=2,
SUBSCRIPT_MIN=min_subscript)
lambda_max=lambda_array[max_subscript]
lambda_min=lambda_array[min_subscript]

; Filter response:
; bright blob : both eigenvalues large AND negative
; dark blob : both eigenvalues large AND positive
; bright line: one eigenvalue small, other large AND negative
; dark line: one eigenvalue small, other large AND positive
; flat structure: both eigenvalues small

; Here, we want to detect bright lines
lambda_max = TEMPORARY(lambda_max) < 0.0
lambda_min = TEMPORARY(lambda_min) < 0.0

filtered_image = REFORM(-lambda_max, x_size, y_size)

; Alternative Filter
; filtered_image = REFORM(-lambda_max+lambda_min, x_size, y_size)

RETURN, filtered_image
END

```

PRO CV_Vessel_Filter_2D_Demo, image

```
;Read the image
path=FILEPATH('md5290fc1.jpg',SUBDIR=['examples','data'])
READ_JPEG, path, img

;Create window:
WINDOW, 0, XSIZE=600, YSIZE=600

;Show original image
TVSCL, img, 0

;Filter image
filtered=CV_Vessel_Filter_2D(img)

;Show filtered image
TVSCL, filtered, 1
END
```
