TECHNICAL REPORT

VSG IMAGE PROCESSING AND ANALYSIS (VSG IPA) TOOLBOX

Version 3.1

VSG IPA: Application Programming Interface

May 2013

Paul F Whelan

Function Summary:

This report outlines the mechanism by which you can interface MATLAB with the VSG IPA toolbox functions. Each function must be called with the correct number and type of arguments. For the majority of the functions, their arguments are self explanatory. In general, xx_img, stands for an image, suffixed with an 'r' or an 'i', it'd be the real or imaginary part of a Fourier spectrum image. Co-ordinates are denoted as p1 or p2 in this help, and should be specified as a 2x1 vector. RGB triplets can be specified as a 3x1 vector and are called col in the help. Arrays will be described using arr or array in the variable name in the help. Input numbers' names are given descriptive names wherever possible. The list of available functions and how to call them is given below.

```
% [out_img1]=vsg('DrawBox',in_img1,p1,p2,col,thickness);
% [out_img1]=vsg('DrawCircle',in_img1,p1,rad,col,thickness);
% [out_img1]=vsg('DrawLine',in_img1,p1,p2,col,thickness);
% [out_img1]=vsg('FillBox',in_img1,p1,p2,col);
% [out_img1]=vsg('FillCircle',in_img1,p1,rad,col);
% [num1]=vsg('GetWidth',in_img1);
% [num1]=vsg('GetHeight',in_img1);
% [num1]=vsg('GetChannels',in_img1);
% [col]=vsg('GetPixel',in_img1,p1);
% [out_img1]=vsg('RemovePixel',in_img1,p1);
% [out_img1]=vsg('SetPixel',in_img1,p1,col);
% [out_img1]=vsg('DoubleSize',in_img1);
% [out_img1]=vsg('HalveSize',in_img1);
% [out_img1]=vsg('MaskImg',in_img1,thickness);
% [out_img1]=vsg('Point2Cross',in_img1);
% [out_img1]=vsg('Point2Square',in_img1);
% [out_img1]=vsg('RotateImg',in_img1,angle);
% [out_img1]=vsg('Rotatem90',in_img1);
% [out_img1]=vsg('Rotatep90',in_img1);
% [out_img1]=vsg('ScaleImg',in_img1,dimx,dimy);
% [out_img1]=vsg('Centroid',in_img1);
% [out_img1]=vsg('Centroid16',in_img1);
% [p1]=vsg('FWP',in_img1);
% [num1]=vsg('PixelSum',in_img1);
% [out_img1]=vsg('Connectivity',in_img1);
% [out_img1]=vsg('ConvexHull',in_img1);
% [out_img1]=vsg('DetectCracks',in_img1);
% [num1]=vsg('EulerNum',in_img1);
% [out_img1]=vsg('FilledConvex',in_img1);
% [num1]=vsg('WPCounter',in_img1);
% [out_img1]=vsg('BiggestBlob',in_img1);
% [out_img1]=vsg('BlobFill',in_img1);
% [out_img1]=vsg('IsolateHoles',in_img1);
% [out_img1]=vsg('IsolateBays',in_img1);
% [num1]=vsg('CountBlobs',in_img1);
% [out_img1]=vsg('SmallestBlob',in_img1);
% [out_img1]=vsg('BoundingBox',in_img1);
% [out_img1]=vsg('FillBounding',in_img1);
% [out_img1]=vsg('LabelByArea',in_img1);
% [out_img1]=vsg('Labeller',in_img1);
% [out_img1]=vsg('Add',in_img1,in_img2);
% [out_img1]=vsg('And',in_img1,in_img2);
% [out_img1]=vsg('Divide',in_img1,in_img2);
% [out_img1]=vsg('Maximum',in_img1,in_img2);
% [out_img1]=vsg('Minimum',in_img1,in_img2);
% [out_img1]=vsg('Multiply',in_img1,in_img2);
% [out_img1]=vsg('OR',in_img1,in_img2);
% [out_img1]=vsg('NOT',in_img1);
% [out_img1]=vsg('Subtract',in_img1,in_img2);
% [out_img1]=vsg('XOR',in_img1,in_img2);
% [out_img1]=vsg('AdaptSmooth',in_img1,num1,num2,num3);
% [out_img1]=vsg('Convolution',in_img1,conv_arr);
% [out_img1]=vsg('Diffusion',in_img1,num1,num2);
% [out_img1]=vsg('DOLPS',in_img1);
% [out_img1]=vsg('Gauss',in_img1,std);
```

```
% [out_img1]=vsg('LINFilter',in_img1);
% [out_img1]=vsg('LowPass',in_img1);
% [out_img1]=vsg('Median',in_img1);
% [out_img1]=vsg('Midpoint',in_img1);
% [out_img1]=vsg('RAFilter',in_img1,size);
% [out_img1]=vsg('Sharpen',in_img1);
% [out_img1]=vsg('SINFilter',in_img1);
% [out_img1]=vsg('ZeroCross',in_img1);
% [out_img1]=vsg('BinaryBorder',in_img1);
% [out_img1,out_img2]=vsg('Canny',in_img1,std,thresh1,thresh2);
% [out_img1]=vsg('CornerPoint',in_img1);
% [out_img1]=vsg('FreiChen',in_img1);
% [out_img1]=vsg('IntensGradDir',in_img1);
% [out_img1]=vsg('Laplacian',in_img1,connected);
% [out_img1]=vsg('NonMaxima',in_img1);
% [out_img1]=vsg('Prewitt',in_img1);
% [out_img1]=vsg('Roberts',in_img1);
% [out_img1]=vsg('Sobel',in_img1);
% [out_img1]=vsg('ThinImg',in_img1,iters);
% [out_img1]=vsg('FullThin',in_img1);
% [out_img1]=vsg('Junctions',in_img1);
% [out_img1]=vsg('LimbEnds',in_img1);
% [out_img1]=vsg('Susan',in_img1,num1,num2);
% [num1]=vsg('AvgIntensity',in_img1);
% [num1]=vsg('EntropyCalc',in_img1);
% [num1]=vsg('HighestGrey',in_img1);
% [num1]=vsg('KurtosisCalc',in_img1);
% [num1]=vsg('LowestGrey',in_img1);
% [num1]=vsg('MSECalc',in_img1,in_img2);
% [num1]=vsg('SkewCalc',in_img1);
% [num1]=vsg('STDCalc',in_img1);
% [num1]=vsg('VarianceCalc',in_img1);
% [out_img1]=vsg('LocalEq3x3',in_img1);
% [out_img1]=vsg('LocalEq5x5',in_img1);
% [out_img1,out_img2,out_img3]=vsg('DBLT',in_img1,num1,num2);
% [out_img1]=vsg('Dilation',in_img1,connected);
% [out_img1]=vsg('DilationNxN',in_img1,conv_arr);
% [out_img1]=vsg('Erosion',in_img1,connected);
% [out_img1]=vsg('ErosionNxN',in_img1,conv_arr);
% [out_img1]=vsg('BeucherGrad',in_img1,connected);
% [out_img1]=vsg('Close',in_img1,connected);
% [out_img1]=vsg('CloseNxN',in_img1,conv_arr);
% [out_img1]=vsg('Open',in_img1,connected);
% [out_img1]=vsg('OpenNxN',in_img1,conv_arr);
% [out_img1]=vsg('HitAndMiss',in_img1,conv_arr1,conv_arr2);
% [out_img1]=vsg('TopHat',in_img1,connected);
% [out_img1]=vsg('Valley',in_img1,connected);
% [out_img1,out_img2]=vsg('ReconByDil',in_img1,in_img2,connected);
% [out_img1,out_img2]=vsg('Watershed',in_img1);
% [out_img1]=vsg('AdditiveNoise',in_img1,num1);
% [out_img1]=vsg('GaussianNoise',in_img1,num1);
% [out_img1]=vsg('PoissonNoise',in_img1,num1);
% [out_img1]=vsg('RayleighNoise',in_img1,num1);
% [out_img1]=vsg('S+PNoise',in_img1,num1);
% [out_img1]=vsg('IntegrateC',in_img1);
% [out_img1]=vsg('IntegrateR',in_img1);
% [out_img1]=vsg('L2RSum',in_img1);
% [out_img1]=vsg('RemIso',in_img1);
% [out_img1]=vsg('B2TWash',in_img1);
% [out_img1]=vsg('T2BWash',in_img1);
% [out_img1]=vsg('L2RWash',in_img1);
% [out_img1]=vsg('R2LWash',in_img1);
% [out_img1]=vsg('HistEqualiser',in_img1);
% [out_img1]=vsg('Enhancer',in_img1);
% [out_img1]=vsg('Stretcher',in_img1,num1,num2);
% [out_img1]=vsg('Exponential',in_img1);
% [out_img1]=vsg('Inverse',in_img1);
% [out_img1]=vsg('Logarithm',in_img1);
% [out_img1]=vsg('Power',in_img1,num1);
% [out_img1]=vsg('Square',in_img1);
% [out_img1]=vsg('DualThresh',in_img1,low_th,high_th);
```

```
% [num1]=vsg('EntropicThresh',in_img1);
% [out_img1]=vsg('GSCapThresh',in_img1,num1);
% [out_img1]=vsg('MidThresh',in_img1);
% [out_img1]=vsg('Threshold',in_img1,thresh);
% [out_img1]=vsg('3x3Thresh',in_img1,thresh);
% [out_img1]=vsg('5x5Thresh',in_img1,thresh);
% [out_img1]=vsg('TripleThresh',in_img1,low_th,high_th);
% [out_img1]=vsg('BinGrey',in_img1,num1);
% [out_img1]=vsg('Hough',in_img1);
% [out_img1]=vsg('InvHough',in_img1,num1);
% [out_img1]=vsg('CircHough',in_img1,num1,num2,num3);
% [out_img1]=vsg('InvCircHough',in_img1,num1,num2,num3);
% [out_img1,out_img2]=vsg('CircDetector',in_img1);
% [out_img1]=vsg('MedialAxis',in_img1);
% [out_img1]=vsg('CoOcMatGen',in_img1);
% [num1]=vsg('CoHomog',in_img1);
% [num1]=vsg('CoEntropy',in_img1);
% [num1]=vsg('CoEnergy',in_img1);
% [num1]=vsg('CoContrast',in_img1);
% [out_img1]=vsg('L2RDistance',in_img1);
% [out_img1]=vsg('R2LDistance',in_img1);
% [out_img1]=vsg('T2BDistance',in_img1);
% [out_img1]=vsg('B2TDistance',in_img1);
% [out_img1]=vsg('3x3Distance',in_img1);
% [out_img1]=vsg('5x5Distance',in_img1);
% [out_img1]=vsg('GFTransform',in_img1);
% [out_img1]=vsg('DCT',in_img1);
% [out_img1]=vsg('IDCT',in_img1);
% [out_img1]=vsg('ViewDCT',in_img1);
% [out_img1,out_img2]=vsg('FFT',in_img1);
% [out_img1]=vsg('InvFFT',in_img1,in_img2);
% [out_img1r,out_img1i]=vsg('FFTAdaptiveLP',in_img1r,in_img1i,thresh);
% [out_img1r,out_img1i]=vsg('FFTDivide',in_img1r,in_img1i,in_img2r,in_img2i);
% [out_img1r,out_img1i]=vsg('FFTGaussian',sizex,sizey,std);
% [out_img1r,out_img1i]=vsg('FFTHighPass',in_img1r,in_img1i,rad1);
% [out_img1r,out_img1i]=vsg('FFTLowPass',in_img1r,in_img1i,rad1);
% [out_img1r,out_img1i]=vsg('FFTMultiply',in_img1r,in_img1i,in_img2r,in_img2i);
% [out_img1r,out_img1i]=vsg('FFTBandPass',in_img1r,in_img1i,rad1,rad2);
% [out_img1r,out_img1i]=vsg('FFTBandStop',in_img1r,in_img1i,rad1,rad2);]
% [out_img1r,out_img1i]=vsg('FFTSelectivePass',in_img1r,in_img1i,rad1,offsetx,offsety);
% [out_img1]=vsg('ViewFFT',in_img1r,in_img1i);
% [out_img1]=vsg('CMYToRGB',in_img1);
% [out_img1]=vsg('RGBToCMY',in_img1);
% [out_img1]=vsg('RGBToHSI',in_img1);
% [out_img1]=vsg('RGBToLAB',in_img1);
% [out_img1]=vsg('RGBToOpp',in_img1);
% [out_img1]=vsg('RGBToXYZ',in_img1);
% [out_img1]=vsg('RGBToYIQ',in_img1);
% [out_img1]=vsg('RGBToYUV',in_img1);
% [out_img1]=vsg('GreyScaler',in_img1);
% [out_img1]=vsg('HSIToRGB',in_img1);
% [out_img1]=vsg('LABToRGB',in_img1);
% [out_img1]=vsg('ViewLAB',in_img1);
% [out_img1]=vsg('ViewOpp',in_img1);
% [out_img1]=vsg('ViewXYZ',in_img1);
% [out_img1]=vsg('ViewYIQ',in_img1);
% [out_img1]=vsg('ViewYUV',in_img1);
% [out_img1]=vsg('XYZToRGB',in_img1);
% [out_img1]=vsg('YIQToRGB',in_img1);
% [out_img1]=vsg('YUVToRGB',in_img1);
% [out_img1]=vsg('Cluster',in_img1,num1);
% [out_img1,out_img2,num1]=vsg('SICluster',in_img1);
% [array1]=vsg('DICOMAvg',in_img1);
% [array1]=vsg('DICOMEnt',in_img1);
% [array1]=vsg('DICOMHi',in_img1);
% [array1]=vsg('DICOMKurt',in_img1);
% [array1]=vsg('DICOMLow',in_img1);
% [array1]=vsg('DICOMMSE',in_img1,in_img2);
% [array1]=vsg('DICOMSkew',in_img1);
% [array1]=vsg('DICOMSTD',in_img1);
% [array1]=vsg('DICOMVar',in_img1);
```

```
% [array1]=vsg('DICOMCoHomog',in_img1);
% [array1]=vsg('DICOMCoEnt',in_img1);
% [array1]=vsg('DICOMCoEng',in_img1);
% [array1]=vsg('DICOMContrast',in_img1);
% [out_img1]=vsg('DOLPS3D',in_img1);
% [out_img1]=vsg('LIN3D',in_img1);
% [out_img1]=vsg('LowPass3D',in_img1);
% [out_img1]=vsg('Midpoint3D',in_img1);
% [out_img1]=vsg('Sharpen3D',in_img1);
% [out_img1]=vsg('SIN3D',in_img1);
% [out_img1]=vsg('FreiChen3D',in_img1);
% [out_img1]=vsg('Laplace3D',in_img1);
% [out_img1]=vsg('NonMax3D',in_img1);
% [out_img1]=vsg('Prewitt3D',in_img1);
% [out_img1]=vsg('Roberts3D',in_img1);
% [out_img1]=vsg('Sobel3D',in_img1);
% [out_img1]=vsg('Gauss3D',in_img1,std);
% [out_img1]=vsg('Mask3D',in_img1,border);
% [out_img1]=vsg('Dilate3D',in_img1,connected);
% [out_img1]=vsg('Erode3D',in_img1,connected);
% [out_img1]=vsg('Close3D',in_img1,connected);
% [out_img1]=vsg('Open3D',in_img1,connected);
% [out_img1]=vsg('BeucherGrad3D',in_img1,connected);
% [out_img1,out_img2]=vsg('ReconByDil3D',in_img1,in_img2,connected);
% [out_img1]=vsg('AIP',in_img1);
% [out_img1]=vsg('MIP',in_img1);
% [out_img1]=vsg('Enhancer3D',in_img1,num1);
% [out_img1]=vsg('Window3D',in_img1,num1);
% [out_img1]=vsg('LocalMin',in_img1,win_size);
% [out_img1]=vsg('LocalMax',in_img1,win_size);
% [out_img1]=vsg('Conv3D',in_img1,conv_arr);  % array must be 9x3...
% i.e. three x,y, (3x3)'s
% [out_img1]=vsg('Labeller3D',in_img1);
% [out_img1]=vsg('LabelByVol',in_img1);
% [out_img1]=vsg('BigBlob3D',in_img1);
% [out_img1]=vsg('SmallBlob3D',in_img1);
% [out_img1]=vsg('Median3D',in_img1);
% [frac]=vsg('EjectFrac',in_img1,in_img2,p1,p2);
% [out_img1,out_img2,out_img3]=vsg('LevelSet',in_img1,x,y);
% vsg('DICOMSave',in_img2,'filename.dcm');
% [out_img1]=vsg('DICOMRead','filename.dcm');
% [out_img1]=vsg('KapurSeg',in_img1);
% [out_img1]=vsg('KittlerSeg',in_img1);
% [out_img1]=vsg('LloydSeg',in_img1);
% [out_img1]=vsg('ModNiblackSeg',in_img1);
% [out_img1]=vsg('OtsuSeg',in_img1);
% [out_img1]=vsg('PalumboSeg',in_img1);
% [out_img1]=vsg('RiddlerSeg',in_img1);
% [out_img1]=vsg('SauvolaSeg',in_img1);
% [out_img1]=vsg('TsaiSeg',in_img1);
% [out_img1]=vsg('YanniSeg',in_img1);
% [out_img1]=vsg('DICOMReadSingle','DICOMImageFile.dcm');
```

If you use this software, please cite:

Paul F Whelan (2013) “VSG Image Processing and Analysis (VSG IPA) Toolbox”, Technical Report, Centre for Image Processing & Analysis, Dublin City University.

Toolbox Installation

1. Download the VSG IPA toolbox setup file – setup_VSG.exe (http://www.cipa.dcu.ie/code.html).

2. Run the executable setup_VSG.exe

3. The VSG IPA Toolbox files and examples are installed to the newly created directory – C:\VSG_IPA_Toolbox. Do **not** delete any files from this directory.

4. The executable continues to install XMedCon which is needed for the advanced medical functions in the toolbox. (User must accept and continue).

5. If the host machine contains multiple versions of MATLAB, the VSG IPA Toolbox will install to the most current version.

6. To check that the toolbox has installed correctly,

- Start Matlab (most current version on machine)
- Enter the command: cd C:\VSG_IPA_toolbox\examples
- Enter the command: convolution_example

This runs the example file "convolution_example.m" and should generate an output as in Figure 1.

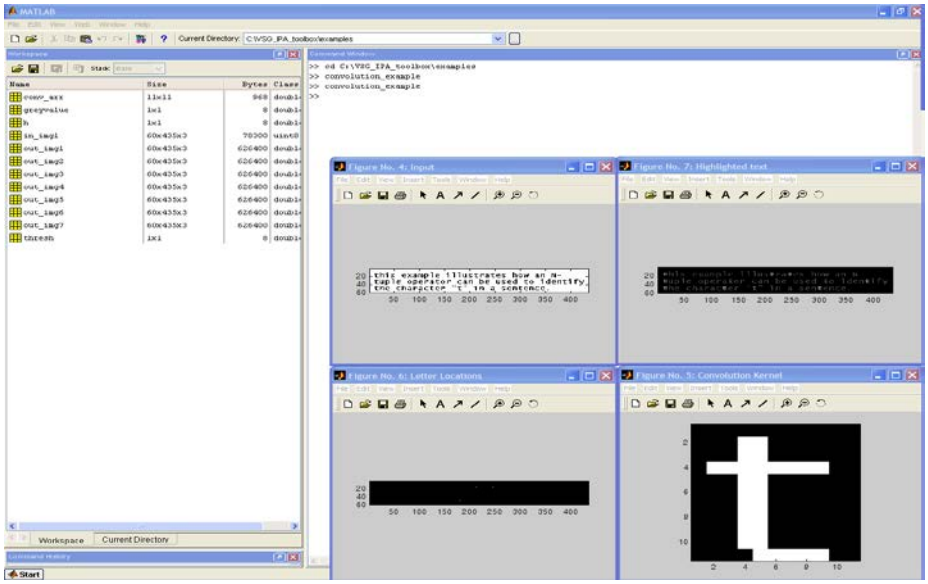


Figure 1: Output of the convolution_example file

7. The VSG IPA Toolbox is now successfully installed. The API is available online at http://www.cipa.dcu.ie/toolbox/HTML_API/api_index.html

8. To uninstall the VSG IPA Toolbox, run the "uninstall.exe" executable which is located at C:\VSG_IPA_Toolbox\uninstall.exe.

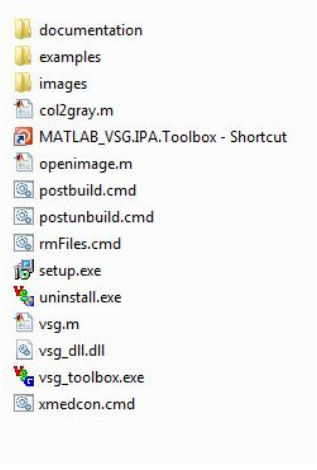
DCU Engineering Labs Use:

The toolbox is set up in S144/S143 (UG use) and S210 (PG use).

You only have write access to two directories. These are your home drive (H:) and the lab temporary drive (C:\Temp). It is strongly recommended that you work from your H drive (see Note below 2 if you experience any performance issues using this drive). The IPA toolbox is stored in Matlab's root directory, therefore, you can use the functionality of the toolbox from your home drive without the need to copy any files. However, if you wish to run the example files from the directory C:\VSG_IPA_toolbox\examples you **MUST** copy the directoriesC:\VSG_IPA_toolbox\examples and C:\VSG_IPA_toolbox\images into your home drive.

Alternatively to set up your own local copy:

- create a local directory on your H drive (let's call this *vsg_matlab*).
- copy all the files (as listed below) from *c:\VSG_IPA_toolbox* to this drive (ie *h:\vsg_matlab*)

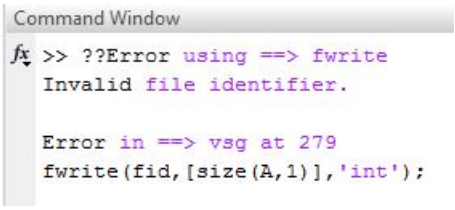


- this is the new path that your program should point to for them to run - as illustrated below:

```
%
% new_program.m
%

%set up paths

%setup the path to the toolbox files.
addpath('h:\vsg_matlab');

%setup the path to the toolbox sample programes.
addpath('h:\vsg_matlab\examples');

%setup the path to the toolbox sample images.
addpath('h:\vsg_matlab\images');

% removes all variables from the workspace. This frees up system memory.
clear;

% load input image
......
```

- you are now in a position to run your programs
- test out all is ok by running one of the example programs supplied
- remember to add any new paths if you are creating/using new subdirectories

NOTE: The toolbox example files (from the installation section) will not run from the C drive in S210/S144/S143. If you want to run the sample files, you must follow the procedure outlined above. If you are getting the following error whenever a vsg function is called: You only have write access to two directories. These are your home drive (H:) and the lab temporary drive (C:\Temp). It is strongly recommended that you work from your H drive. The IPA toolbox is stored in Matlab's root directory, therefore, you can use the functionality of the toolbox from your home drive without the need to copy any files. However, if you wish to run the example files from the directory "C:\VSG_IPA_toolbox\examples", you MUST copy the directories "C:\VSG_IPA_toolbox\examples" and "C:\VSG_IPA_toolbox\images" into your home drive.



Then your path is setup incorrectly. You can confirm this by running the *path* command in the matlab command window.

```
>> path

        MATLABPATH

    C:\Program Files\MATLAB\R2010b\toolbox\matlab\general
    C:\Program Files\MATLAB\R2010b\toolbox\matlab\ops
    C:\Program Files\MATLAB\R2010b\toolbox\matlab\lang
    C:\Program Files\MATLAB\R2010b\toolbox\matlab\elmat
    C:\Program Files\MATLAB\R2010b\toolbox\matlab\randfun
    C:\Program Files\MATLAB\R2010b\toolbox\matlab\elfun
    C:\Program Files\MATLAB\R2010b\toolbox\matlab\specfun
    C:\Program Files\MATLAB\R2010b\toolbox\matlab\matfun
    C:\Program Files\MATLAB\R2010b\toolbox\matlab\datafun
```

NOTE 2: Some people experience performance issues (random errors, slow performance) when running Matlab from the H drive or an external USB drive. If this is an issue for you, one suggestion is to run you code from "C:\Temp" directory. Remember that the contents of this directory is **cleared once you logout**, so make sure to copy all your work to the H drive (or USB stick) on completion.

Utility functions

Notes:

In any function where a colour must be specified as an argument to the function, the colour must be specified as an RGB vector. Even for single channel images, a colour must still be specified as a 3-vector. Only the first value is used to draw the box. If for example the colour [100, 200, 20] was used, the greyscale value of the output box on the image would be 100.

Note that, when examining pixels in MatLab, the coordinates have origin (1,1,1), whereas in VSG_API, the origin is at (0,0,0). This means that if you get the pixel at [100,150] using VSGAPI, this corresponds to the position (151,101,:) in MatLab. (MatLab uses row, column indexing rather than x,y).

DrawBox

Draw a hollow box in the image.

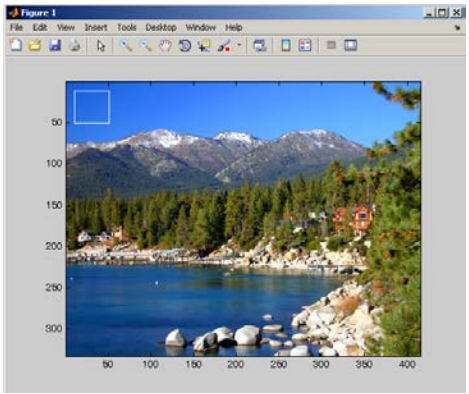
Function call:
[out_img1]=vsg('DrawBox',in_img1,p1,p2,col,thickness);

Arguments:
in_img1 input image, can be 3 channel RGB, single channel greyscale, or binary image.
p1 - top left corner of the box to be drawn, including this pixel. Should be specified as a 2 vector e.g. [100;100].
p2 - bottom right corner of the box to be drawn, including this pixel. Should be specified as a 2 vector e.g. [100;100].
col - the colour of the box to be drawn. This should be specified as a 3-vector e.g. [50;100;200].
thickness - the thickness of the line used to draw the box, specified as an integer number of pixels.

Description:
[out_img1]=vsg('DrawBox',in_img1,p1,p2,col,thickness) – *out_img1* is an image of the same dimensions and number of colour planes as the input image, with a box drawn at the specified location, with the specified colour and thickness.

Example:

```
img1 = openimage('lake.jpg');
[out_img1]=vsg('DrawBox',img1,[0,0],[50,50],[255,255,255],1);
figure, image(uint8(out_img1));
```



DrawCircle

Draw a hollow circle in the image.

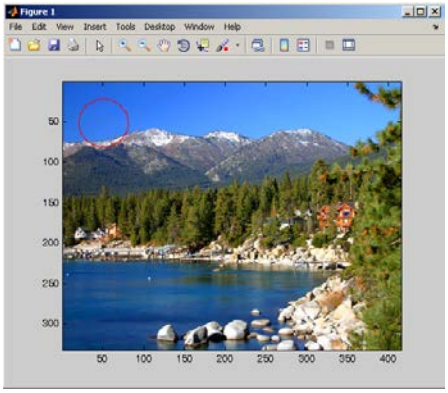
function call:
[out_img1]=vsg('DrawCircle',in_img1,p1,rad,col,thickness);

Arguments:
in_img1 input image, can be 3 channel RGB, single channel greyscale, or binary image.
p1 – centre of the circle to be drawn. Should be specified as a 2 vector e.g. [100;100].
rad – radius of the circle to be drawn. Can be specified as an integer or a double.
col - the colour of the circle to be drawn. This should be specified as a 3-vector e.g. [50;100;200].
thickness - the thickness of the line used to draw the circle, specified as an integer number of pixels.

Description:
[out_img1]=vsg('DrawCircle',in_img1,p1,rad,col,thickness) – *out_img1* is an image of the same dimensions and number of colour planes as the input image, with a circle drawn at the specified location, with the specified radius, colour and thickness.

Example:

```
img1 = openimage('lake.jpg');
[out_img1]=vsg('DrawCircle',img1,[50,50],30,[255,0,0],1);
figure, image(uint8(out_img1));
```



DrawLine

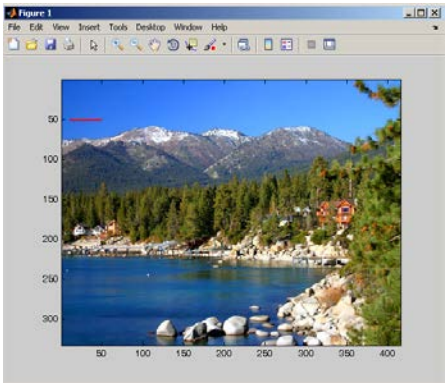
Draw a line in the image.

function call:
[out_img1]=vsg('DrawLine',in_img1,p1,p2,col,thickness);

Arguments:
in_img1 input image, can be 3 channel RGB, single channel greyscale, or binary image.
p1 – first coordinate of the line to be drawn. Should be specified as a 2 vector e.g. [100;100].
p2 – second coordinate of the line to be drawn. Should be specified as a 2 vector e.g. [100;100].
col - the colour of the box to be line. This should be specified as a 3-vector e.g. [50;100;200].
thickness - the thickness of the line used to be drawn, specified as an integer number of pixels.

Description:
[out_img1]=vsg('DrawLine',in_img1,p1,p2,col,thickness) – *out_img1* is an image of the same dimensions and number of colour planes as the input image, with a line drawn between the specified coordinates with the specified colour and thickness.
Example:

```
img1 = openimage('lake.jpg');
[out_img1]=vsg('DrawLine',img1,[10,50],[50,50],[255,0,0],2);
figure, image(uint8(out_img1));
```



FillBox

Draw a filled box in the image.

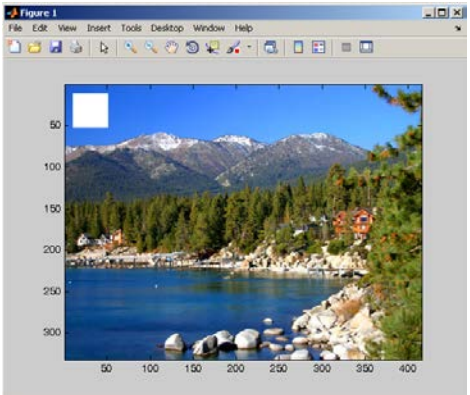
function call:
[out_img1]=vsg('FillBox',in_img1,p1,p2,col);

Arguments:
in_img1 input image, can be 3 channel RGB, single channel greyscale, or binary image.
p1 - top left corner of the box to be drawn, including this pixel. Should be specified as a 2 vector e.g. [100;100].
p2 - bottom right corner of the box to be drawn, including this pixel. Should be specified as a 2 vector e.g. [100;100].
col - the colour of the box to be drawn. This should be specified as a 3-vector e.g. [50;100;200].

Description:
[out_img1]=vsg('FillBox',in_img1,p1,p2,col) – The *out_img1* is an image of the same dimensions and number of colour planes as the input image, with a filled box drawn at the specified location, with the specified colour.

Example:

```
img1 = openimage('lake.jpg');
[out_img1]=vsg('FillBox',img1,[10,10],[50,50],[255,255,255]);
figure, image(uint8(out_img1));
```



FillCircle

Draw a filled circle in the image.

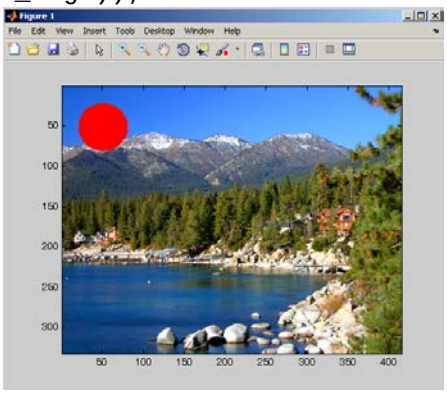
function call:
[out_img1]=vsg('FillCircle',in_img1,p1,rad,col);

Arguments:
in_img1 input image, can be 3 channel RGB, single channel greyscale, or binary image.
p1 – centre of the circle to be drawn. Should be specified as a 2 vector e.g. [100;100].
rad – radius of the circle to be drawn. Can be specified as an integer or a double.
col - the colour of the circle to be drawn. This should be specified as a 3-vector e.g. [50;100;200].

Description:
[out_img1]=vsg('FillCircle',in_img1,p1,rad,col)– *out_img1* is an image of the same dimensions and number of colour planes as the input image, with a filled circle drawn at the specified location, with the specified radius and colour.

Example:

```
img1 = openimage('lake.jpg');
[out_img1]=vsg('FillCircle',img1,[50,50],30,[255,0,0]);
figure, image(uint8(out_img1));
```



GetWidth

Get the width of the input image.

function call:
[num1]=vsg('GetWidth',in_img1);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.

Description:
[num1]=vsg('GetWidth',in_img1) – *num1* is an integer representing the width of the input image.

Example:

```
img1 = openimage('lake.jpg');
[width] = vsg('GetWidth',img1);
```

GetHeight

Get the height of the input image.

function call:
[num1]=vsg('GetHeight',in_img1);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.

Description:
[num1]=vsg('GetHeight',in_img1) – *num1* is an integer representing the height of the input image.

Example:
```
img1 = openimage('lake.jpg');
[height] = vsg('GetHeight',img1);
```

GetChannels

Get the number of colour planes in the input image.

function call:
[num1]=vsg('GetChannels',in_img1);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.

Description:
[num1]=vsg('GetChannels',in_img1) – *num1* is an integer representing the number of colour planes of the input image.

Example:
```
img1 = openimage('lake.jpg');
[ch] = vsg('GetChannels',img1);
```

GetPixel

Get the pixel intensity from an image at a certain coordinate..

function call:
[col]=vsg('GetPixel',in_img1,p1);

Arguments:
in_img1 input image, can be 3 channel RGB, single channel greyscale, or binary image.
p1 – An (x,y) coordinate of the pixel to be examined, specified as a 2-vector, e.g. [100;150].

Description:
[col]=vsg('GetPixel',in_img1,p1) – *col* is an RGB triplet stored as a 3 vector.

Example:
```
img1 = openimage('lake.jpg');
[col] = vsg('GetPixel',img1,[10,10]);
```

Notes:
Returns the colour at the specified position in the image. If the image only has a single channel, a 3 vector is still returned, but only the first value will be set.

RemovePixel

Remove a pixel at a certain coordinate from the image (removing a pixel sets that pixel to black).

function call:
[out_img1]=vsg('RemovePixel',in_img1,p1);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.
p1 – An (x,y) coordinate of the pixel to be removed, specified as a 2-vector, e.g. [100;150].

Description:
[out_img1]=vsg('RemovePixel',in_img1,p1) – *out_img1* is an image of the same dimensions and number of colour planes as the input image, with the pixel at the specified position set to black.

Example:
```
img1 = openimage('lake.jpg');
[out_img1]=vsg('RemovePixel',img1,[15,15]);
figure, image(uint8(out_img1));
```

Notes:
Regardless of the type of image data, integer, binary, or positive or negative double, the removed pixel on all planes will be sent to 0. Also, if this is used with DICOM images, the same pixel will be zeroed on each slice of the DICOM image.

SetPixel

Set the intensity of a pixel at a certain coordinate in the image.

function call:
[out_img1]=vsg('SetPixel',in_img1,p1,col);

Arguments:
in_img1 input image, can be 3 channel RGB, single channel greyscale, or binary image, can also be DICOM image.
p1 – An (x,y) coordinate of the pixel to be removed, specified as a 2-vector, e.g. [100;150].
col - the colour of the pixel to be drawn. This should be specified as a 3-vector e.g. [50;100;200].

Description:
[out_img1]=vsg('SetPixel',in_img1,p1,col) – *out_img1* is an image of the same dimensions and number of colour planes as the input image, with the pixel at the specified position set to the specified colour.

Example:
```
img1 = openimage('lake.jpg');
[out_img1]=vsg('SetPixel',img1,[100,100],[255,0,0]);
figure, image(uint8(out_img1));
```

Notes:
For single colour plane images, the first value of the RGB triplet will be used. Also, if this is used with DICOM images, the same pixel will be set on each slice of the dicom image.

DoubleSize

An image whose size is doubled.

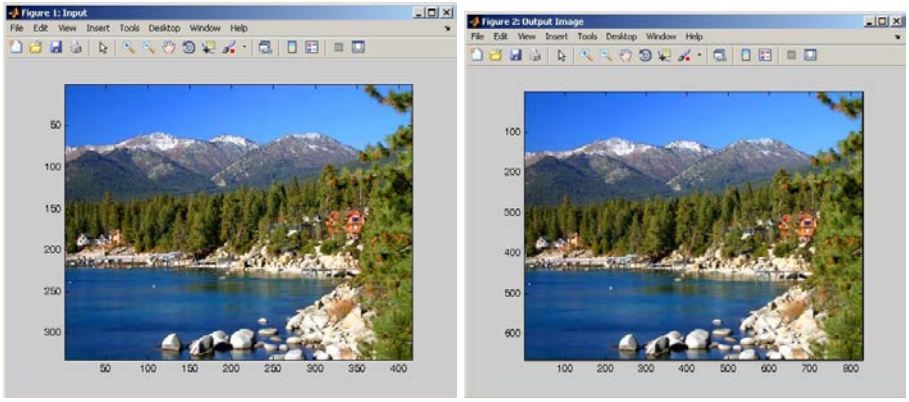
function call:
[out_img1]=vsg('DoubleSize',in_img1);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.

Description:
[out_img1]=vsg('DoubleSize',in_img1) – *out_img1* is an image which is twice the width and twice the height of the input, the number of colour channels remains the same and in the case of DICOM images, the number of slices remains the same.

Example:

```
img1 = openimage('lake.jpg');
h=figure, image(uint8(img1)); set(h,'Name','Input');
[out_img1] = vsg('DoubleSize',img1);
h=figure, image(uint8(out_img1)); set(h,'Name','Output Image');
```



HalveSize

An image whose size is halved.

function call:
[out_img1]=vsg('HalveSize',in_img1);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.

Description:
[out_img1]=vsg('HalveSize',in_img1) – *out_img1* is an image which is half the width and half the height of the input, the number of colour channels remains the same and in the case of DICOM images, the number of slices remains the same.

Example:

```
img1 = openimage('lake.jpg');
[out_img1]=vsg('HalveSize',img1);
h=figure, image(uint8(out_img1)); set(h,'Name','Output Image');
```

MaskImg

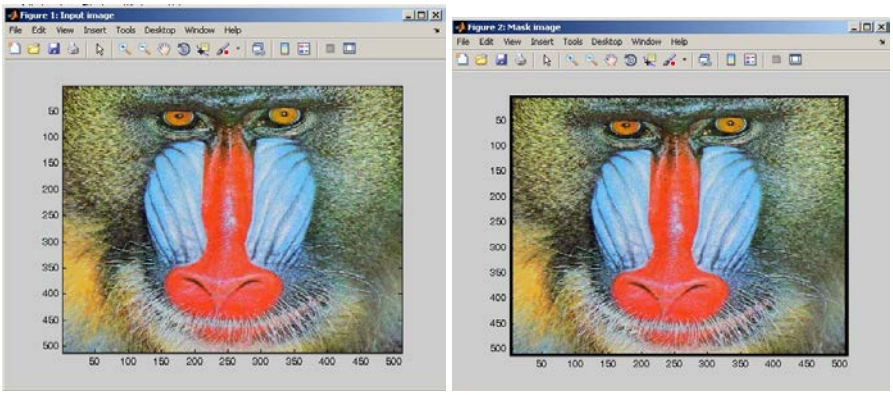
An image whose border is masked by a user specified amount.

function call:
[out_img1]=vsg('MaskImg',in_img1,thickness);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.
thickness – an integer value specify the thickness of the black border.

Description:
[out_img1]=vsg('MaskImg',in_img1,thickness) – *out_img1* is an image which is the same as the input image, except the output image has a frame of pixels at its edge, which are set to black (0). The thickness of this edge is specified as an argument.

Example:
```
img1 = openimage('baboon.jpg');
h=figure;image(uint8(img1));set(h,'Name','Input image');
[out_img1]=vsg('MaskImg',img1,5);
h=figure; image(uint8(out_img1));set(h,'Name','Mask image');
```



Notes:
For DICOM images, a mask is placed on each slice.

Point2Cross

An image whose white pixels are represented by white crosses.

function call:
[out_img1]=vsg('Point2Cross',in_img1);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.

Return values:
out_img1 – An image of the same size as the input image. For every pixel with a value of 255 in the input image, a cross 5 pixels high and 5 pixels wide is drawn on the output image, centred at the position of the input pixel. Each colour plane is treated separately, so red, green and blue crosses may be drawn if the 3 planes don't have the same "white" (255) pixels.

Example:
```
img1 = openimage('lake.jpg');
h=figure;image(uint8(img1));set(h,'Name','Input image');
```

```
[out_img1]=vsg('Point2Cross',img1);
h=figure; image(uint8(out_img1));set(h,'Name','Point2Cross image');
```

Notes:
For DICOM images, each slice is treated separately. Binary images must be scaled so that they have a range of {0,255} rather than {0,1} (only an issue if the images are logical arrays created in MatLab).

Point2Square

An image whose white pixels are represented by white squares.

function call:
[out_img1]=vsg('Point2Square',in_img1);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.

Return values:
out_img1 – An image of the same size as the input image. For every pixel with a value of 255 in the input image, a square 5 pixels high and 5 pixels wide is drawn on the output image, centred at the position of the input pixel. Each colour plane is treated separately, so red, green and blue squares may be drawn if the 3 planes don't have the same "white" (255) pixels.

Example:
```
img1 = openimage('lake.jpg');
h=figure;image(uint8(img1));set(h,'Name','Input image');
[out_img1]=vsg('Point2Square',img1);
h=figure; image(uint8(out_img1));set(h,'Name','Point2Square image');
```

Notes:
For DICOM images, each slice is treated separately. Binary images must be scaled so that they have a range of {0,255} rather than {0,1} (only an issue if the images are logical arrays created in MatLab).

RotateImg

An image is rotated in a clockwise direction by a user specified amount.

function call:
[out_img1]=vsg('RotateImg',in_img1,angle);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.
angle – double, angle in degrees by which the image should be rotated. Positive values rotate the image in an anticlockwise direction as is the convention.

Return values:
out_img1 – An image of the same size as the input image. The output image is a rotated version of the input image.

Example:
```
img1 = openimage('lake.jpg');
h=figure;image(uint8(img1));set(h,'Name','Input image');
[out_img1]=vsg('RotateImg',img1,30);
h=figure; image(uint8(out_img1));set(h,'Name', 'Rotated image');
```

Notes:
The output image values are calculated using a bilinear interpolation. Parts of the image that would be transformed to regions outside the image are cut-off to maintain the image dimensions.

Rotatem90

An image is rotated in an anticlockwise direction by 90 degrees.

function call:
[out_img1]=vsg('Rotatem90',in_img1);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.

Return values:
out_img1 – An image which is an anti-clockwise 90 degree rotated version of the input image, the dimensions of which have changed as follows. The output image width is the same as the input image height and the output image height is the same as the input image width.

Example:
```
img1 = openimage('lake.jpg');
h=figure;image(uint8(img1));set(h,'Name','Input image');
[out_img1]=vsg('Rotatem90',img1);
h=figure; image(uint8(out_img1));set(h,'Name', 'Rotated image');
```

Notes:
The output image values are taken directly from the input values in a matched pixel by pixel operation.

Rotatep90

An image is rotated in a clockwise direction by 90 degrees.

function call:
[out_img1]=vsg('Rotatep90',in_img1);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.

Return values
out_img1 – An image which is a clockwise 90 degree rotated version of the input image, the dimensions of which have changed as follows. The output image width is the same as the input image height and the output image height is the same as the input image width.

Example:
```
img1 = openimage('lake.jpg');
h=figure;image(uint8(img1));set(h,'Name','Input image');
[out_img1]=vsg('Rotatep90',img1);
h=figure; image(uint8(out_img1));set(h,'Name', 'Rotated image');
```

Notes:
The output image values are taken directly from the input values in a matched pixel by pixel operation.

ScaleImg

An image is scaled by user defined dimensions.

function call:

[out_img1]=vsg('ScaleImg',in_img1,dimx,dimy);

Arguments:

in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.
dimx – the new width of the image specified as an integer.
dimy – the new height of the image specified as an integer.

Description:

[out_img1]=vsg('ScaleImg',in_img1,dimx,dimy) - *out_img1* is an image which is a scaled version of the input image, the dimensions of which have changed as follows. The output image width is the same as the specified *dimx* and the output image height is the same as the specified *dimy*.

Example:

```
img1 = openimage('lake.jpg');
h=figure;image(uint8(img1));set(h,'Name','Input image');
[out_img1]=vsg('ScaleImg',img1,200,150);
h=figure; image(uint8(out_img1));set(h,'Name', 'Scaled image');
```

Notes:

The output image is a scaled version of the input, the scaling is performed using a bi-linear interpolation.

Analysis Functions

Centroid

Replace the greyscale shapes (Range 0-255) in the original image by their respective centroids (commonly used after the 8-bit labelling operators).

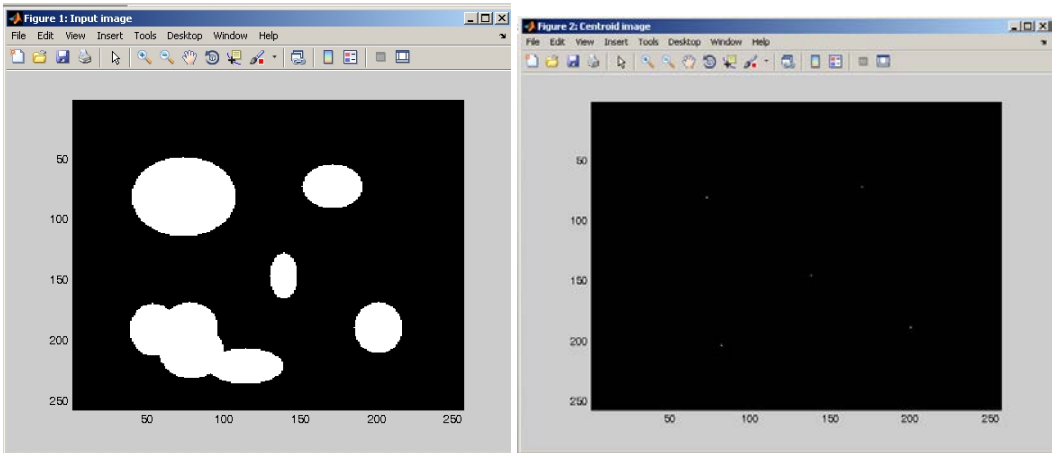
function call:
[out_img1]=vsg('Centroid',in_img1);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.

Description:
Replace grey scale shape by a single pixel at its centre of gravity. The Centroid function is limited by the fact that it can have maximum 255 blobs in a scene. It is also assumes a black background which is ignored in the centroid calculation. This approach is based on the fact that all the shapes must have a distinct grey scale. *out_img1*- An image which contains a white pixel representing the centroid of each white blob in the input image.

Example:

```
img1=openimage('blobs.jpg');
[img1] = vsg('Threshold',img1,100);
[out_img1]=vsg('Centroid',img1);
h=figure;image(uint8(img1));set(h,'Name','Input image');
h=figure; imagesc(uint8(out_img1));set(h,'Name','Centroid image');
```



Notes:
This function can handle up to 255 individual blobs. If there are more blobs in the input image, use the slower function Centroid16 which can handle more blobs.

Centroid16

Replace the greyscale shapes (Range 0-65535) in the original image by their respective centroids (commonly used after the Label_16 operators).

function call:
[out_img1]=vsg('Centroid16',in_img1);
Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.

Description:
Replace grey scale shape by a single pixel at its centre of gravity. It is also assumes a black background which is ignored in the centroid calculation. This approach is based on the fact that all the shapes must have a distinct grey scale. *out_img1* – An image which contains a white pixel representing the centroid of each white blob in the input image.

Example:
```
img1=openimage('blobs.jpg');
[img1] = vsg('Threshold',img1,100);
[out_img1]=vsg('Centroid16',img1);
h=figure;image(uint8(img1));set(h,'Name','Input image');
h=figure; imagesc(uint8(out_img1));set(h,'Name','Centroid image');
```

Notes:
This function can handle up to 65535 individual blobs. The process is slow, but labels won't be duplicated up to 65535 blobs.

CornerPoint

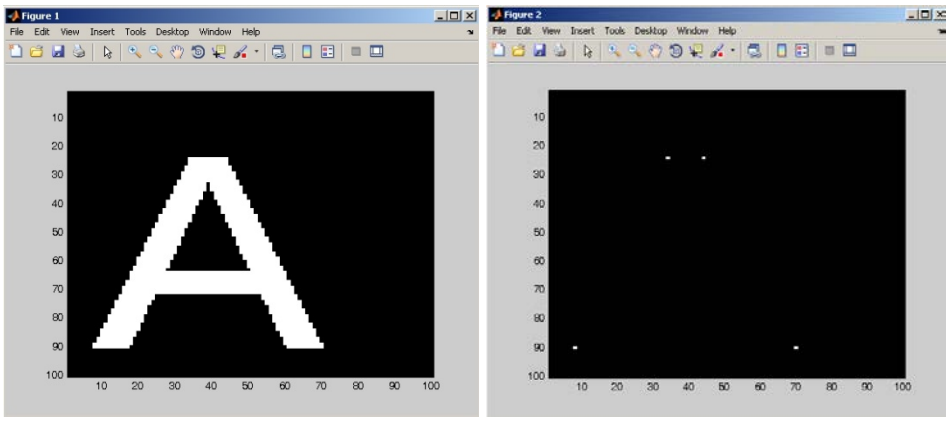
Skeleton corner detection from a binary image based on a 3x3 region.

function call:
[out_img1]=vsg('CornerPoint',in_img1);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.

Description: 0
The CornerPoint function uses 3x3 masks of varying orientation to detect corner points in a binary image. It detects 3×3 corner points and replaces them by a single white pixel if the illustrated patterns occur (Ref). Otherwise, the output pixel (shaded) is set to black. *out_img1* - An image which has dimensions the same as the input image. The output image displays the binary corner points.

Example:
```
img1 = openimage('A.bmp'); %color image
[out_img1]=vsg('Threshold',img1,100);
figure;image(uint8(out_img1));
[out_img1]=vsg('CornerPoint',out_img1);
h=figure;image(uint8(out_img1));
```



Notes:
For non-binary images or colour-planes/slices, corner points must have exactly 0 and 255 values within the corner point detection kernel for that point to be flagged as a corner.

Ref: Paul F. Whelan and Derek Molloy, "*Machine Vision Algorithm in Java – Techniques and implementation*", Springer, 2000.

FWP

Coordinate point representing the location of the first white pixel in the image input image.

function call:
[p1]=vsg('FWP',in_img1);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image.

Return values
p1 – The coordinate of the first white pixel represented as a 2 vector.

Example:
```
img1 = openimage('A.bmp'); %color image
[p1]=vsg('FWP',img1);
```

Notes:
This function handles colour, greyscale and binary images. The binary images should have the range {0,255}, as the function tests for equality with 255.

GreyscalePixelSum

Generates an integer which is the sum of all pixels contained in the input image.

function call:
[num1]=vsg('GreyscalePixelSum',in_img1);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.

Return values
num1 – A number, which is the sum of all grey levels across all image dimensions and colour planes.

Example:

```
img1=openimage('baboon.jpg');
[num]=vsg('GreyscalePixelSum',img1);
```

Connectivity

Connectivity detection in a thinned skeleton binary image. Mark points critical for connectivity in a 3x3 region.

function call:

[out_img1]=vsg('Connectivity',in_img1);

Arguments:

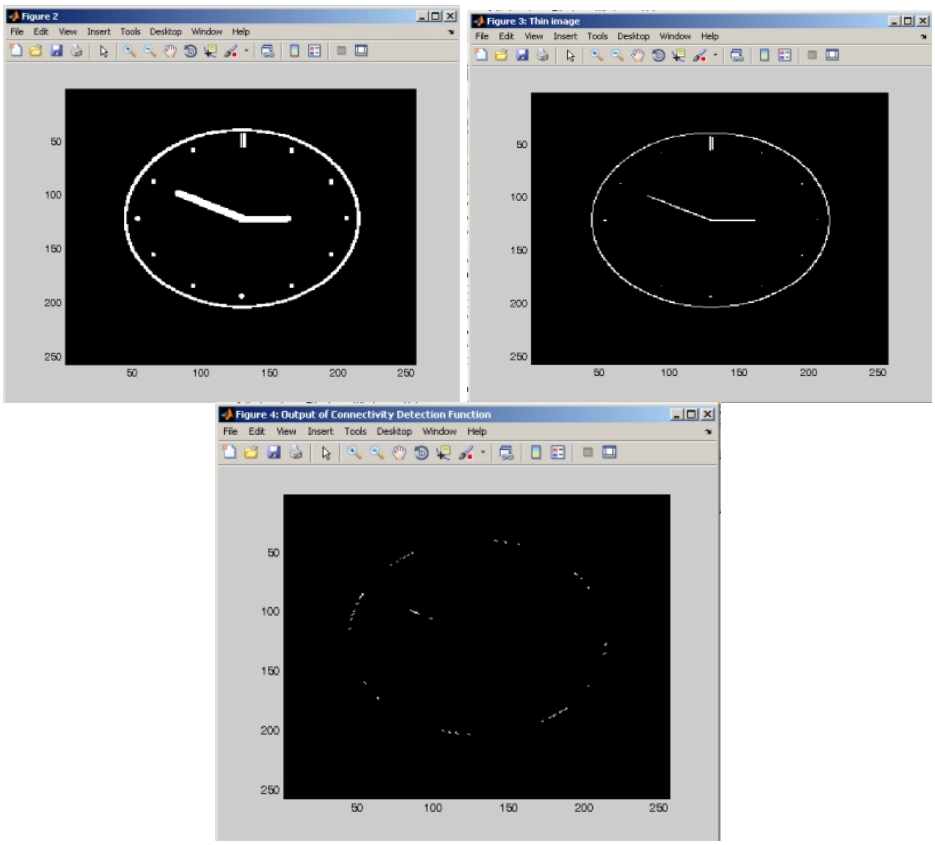
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.

Description:

A connectivity detector shades the output image with 1's to indicate the position of those points which are critical for connectivity (and which are white in the input image). Black points and those which are not critical for connectivity, are mapped to black in the output image. *out_img1* – an image of the same dimensions as the input image and has a white pixel marked on it for every white pixel in the input image which is deemed to be important for object connectivity.

Example:

```
img = openimage('clock.gif'); %color image
h=figure;image(img);set(h,'Name','Input image');
[out_img1]=vsg('Threshold',img,100);
[out_img1]=vsg('FullThin',out_img1);
h=figure;image(uint8(out_img1));set(h,'Name','Thin image');
[out_img1]=vsg('Connectivity',out_img1);
h=figure;image(uint8(out_img1));set(h,'Name','Output of Connectivity
Detection Function');
```



Notes:
A connective pixel is deemed to be a pixel which if removed would cause the object to become broken. It is this pixel that is set to white in the output image. Ensure that binary images have a range of {0,255}.

ConvexHull

Compute the convex hull boundary.

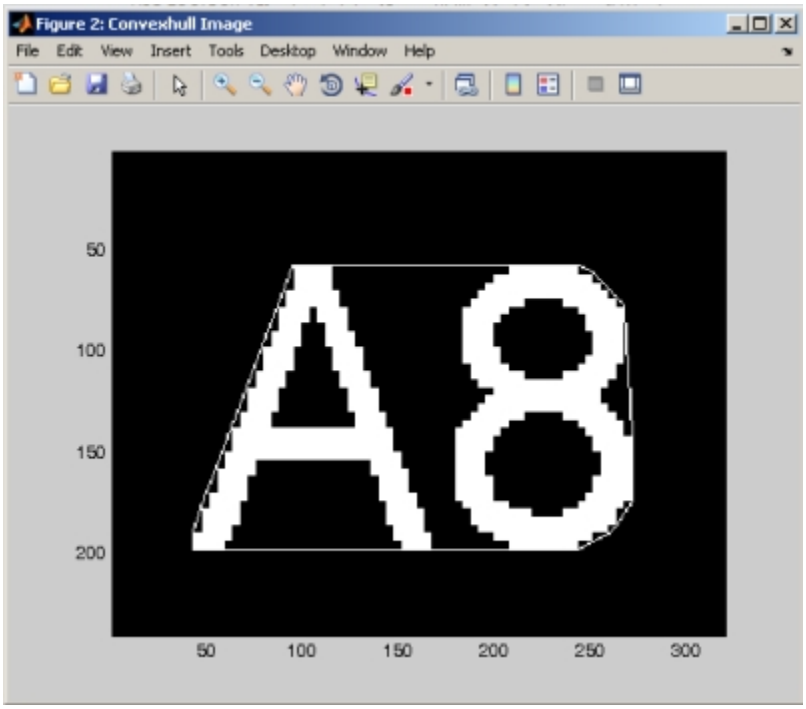
function call:
[out_img1]=vsg('ConvexHull',in_img1);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.

Description:
Consider a single blob in a binary image. The convex hull is that area enclosed within the smallest convex polygon which encloses the shape. *out_img1* – an image of the same dimensions as the input image. The output has the smallest possible enclosing polygon drawn around the objects in the input image.

Examples:

```
img = openimage('A8.bmp');
[out_img1]=vsg('Threshold',img,100);
[out_img1]=vsg('ConvexHull',out_img1);
h=figure;image(uint8(out_img1));set(h,'Name','Convexhull Image');
```



Notes:
For DICOM images, each slice is considered separately. RGB images are worked on as a single plane after edge extraction. Due to the tight fit of the polygon line, single pixels may end up being bays inside the convex hull.

FilledConvex

Compute the filled convex hull.

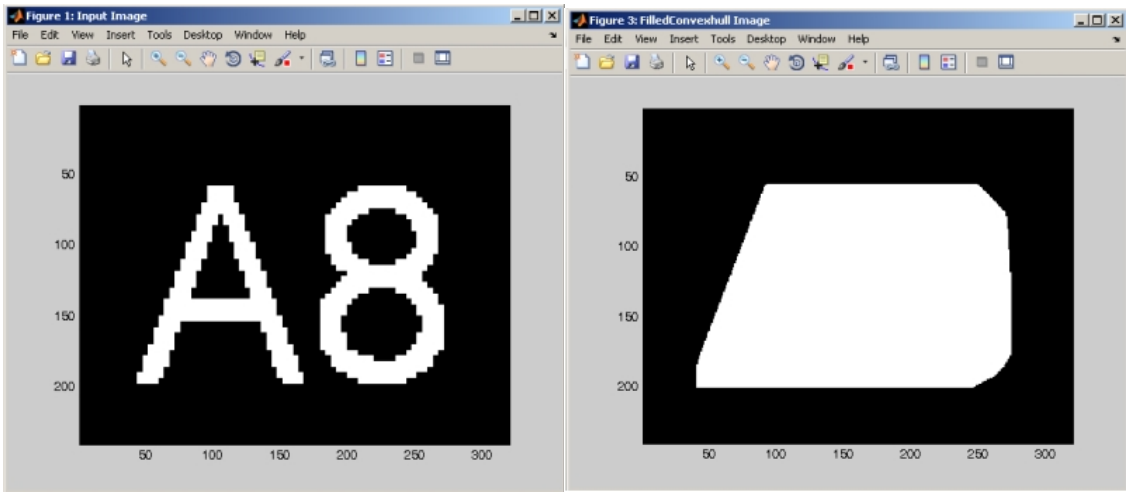
function call:
[out_img1]=vsg('FilledConvex',in_img1);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.

Description:
Consider a single blob in a binary image. The convex hull is that area enclosed within the smallest convex polygon which encloses the shape. *out_img1* – an image of the same dimensions as the input image. The output has the smallest possible enclosing polygon drawn around the objects in the input image. The polygon is filled with white pixels.

Examples:

```
Img1 = openimage('A8.bmp');
h=figure;image(img1);set(h,'Name','Input Image');
[out_img1]=vsg('Threshold',img1,100);
[out_img1]=vsg('ConvexHull',out_img1);
h=figure;image(uint8(out_img1));set(h,'Name','Convexhull Image');
```



Notes:
For DICOM images, each slice is considered separately. RGB images are worked on as a single plane after edge extraction.

DetectCracks

Highlight cracks in the input image.

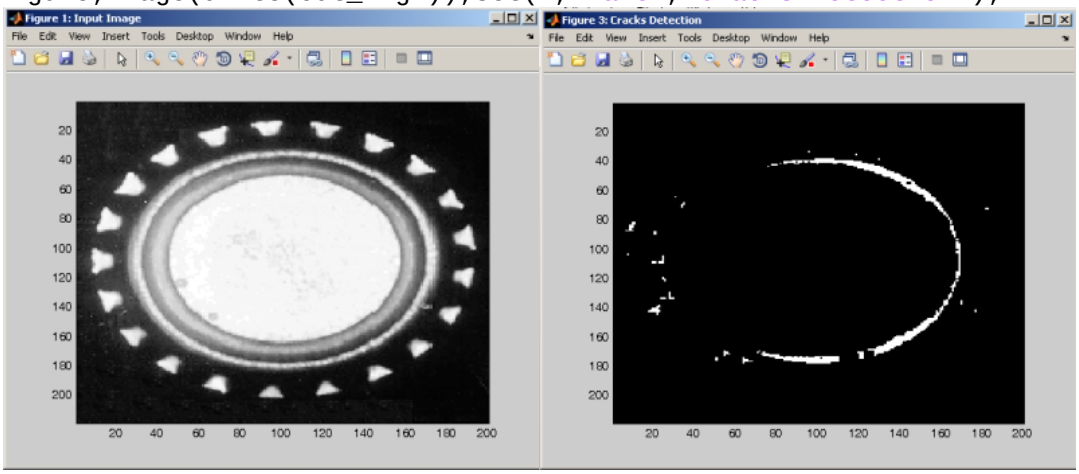
function call:
[out_img1]=vsg('DetectCracks',in_img1);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.

Description
out_img1 – an image of the same dimensions as the input image that shows any cracks in the input image.

Example:

```
img = openimage('crown.bmp'); %color image
h=figure,image(uint8(img));set(h,'Name','Input Image');
[out_img1]=vsg('Threshold',img,100);
[out_img1]=vsg('DetectCracks',out_img1);
h=figure;image(uint8(out_img1));set(h,'Name','Cracks Detection');
```



Notes:
For DICOM images, each slice is considered separately and 2D operators are used. RGB images are worked on plane by plane.

EulerNum

Compute the Euler number from a binary image.

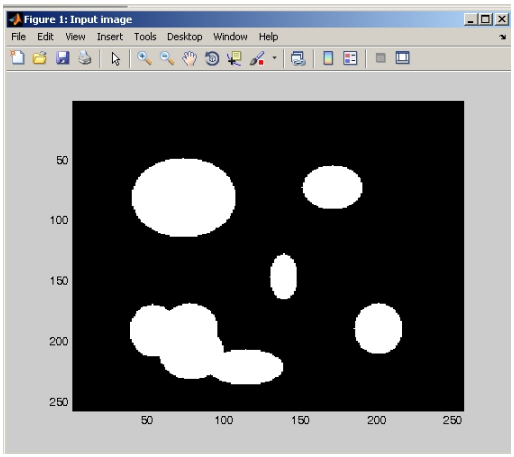
function call:
[num1]=vsg('EulerNum',in_img1);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image.

Return values:
The *num1* – the Euler number of the input image. The number of objects in the image minus the number of holes in those objects.

Example:

```
img = openimage('blobs.jpg'); %color image
[out_img1]=vsg('Threshold',img,100);
[num1]=vsg('EulerNum',out_img1);
```



num1 = 5 (5 objects, 0 holes).

WPCounter

Compute the number of white pixels.

function call:
[num1]=vsg('WPCounter',in_img1);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.

Return values
num1 – the number of white pixels in the input image.

Example:

```
img = openimage('blobs.jpg');
[num1]=vsg('WPCounter',img1);
```

Notes:
Ensure that binary images have a range {0,255}. For DICOM images, this function returns the total number of white pixels across all slices.

Blob functions

BiggestBlob

Extract the biggest white blob from a binary image.

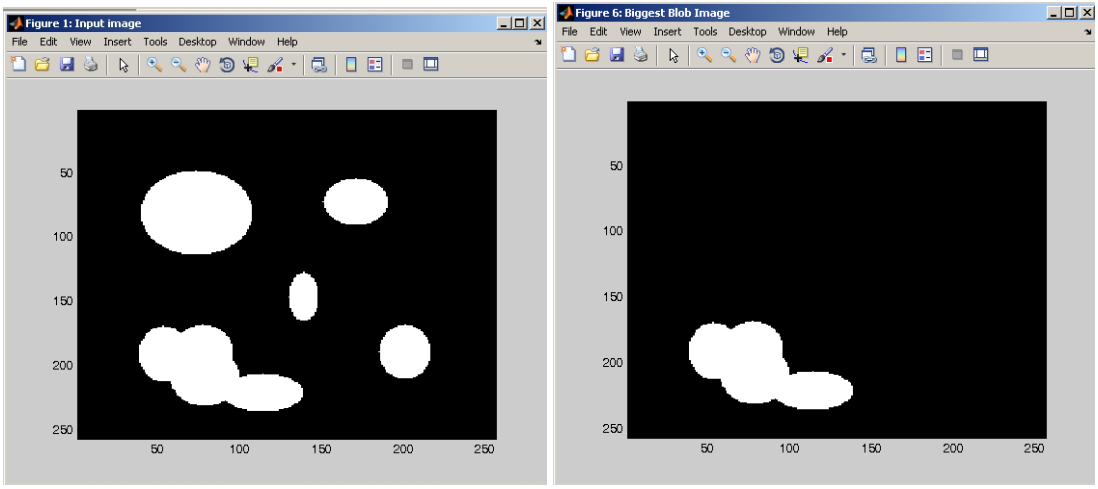
function call:
[out_img1]=vsg('BiggestBlob',in_img1);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.

Description:
Find the largest, non-connecting, white blob in a binary image. *out_img1* - an image which has dimensions the same as the input image. The output image displays the largest of the input white blobs.

Example:

```
img1 = openimage('blobs.bmp');
h=figure;image(img1);set(h,'Name','Input Image');
[out_img1]=vsg('Threshold',img1,100);
[out_img1]=vsg('BiggestBlob',out_img1);
h=figure;image(uint8(out_img1));set(h,'Name','Biggest Blob Image');
```



Notes:
1. For DICOM images, the function operates on each slice separately.
2. For RGB images, the input on all 3 channels must be white for the area to be considered white.

BlobFill

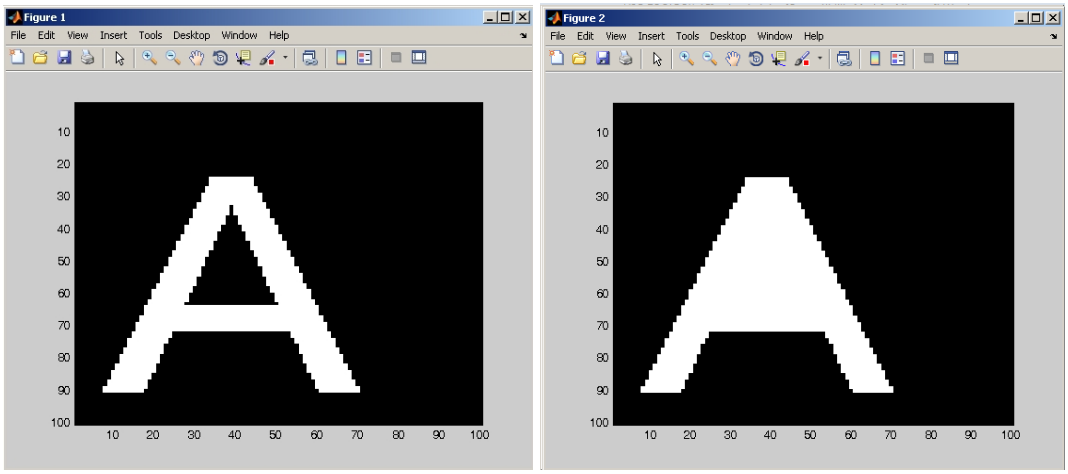
Fill the holes in a binary image.

function call:
[out_img1]=vsg('BlobFill',in_img1);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image or DICOM image.

Description:
Fills the hole in a binary image. out_img1 – returns an image of the same dimensions as the input with each white blob filled so that there are no black spots in any of the white blobs.

Example:
```
img = openimage('A.bmp');
figure;image(img);
[out_img1]=vsg('Threshold',img,100);
[out_img1]=vsg('BlobFill',out_img1);
figure;image(uint8(out_img1));
```



Notes:
Ensure that binary images have a range {0,255}. For DICOM images, each white blob on each slice is filled in, not the 3D blob.

IsolateHoles

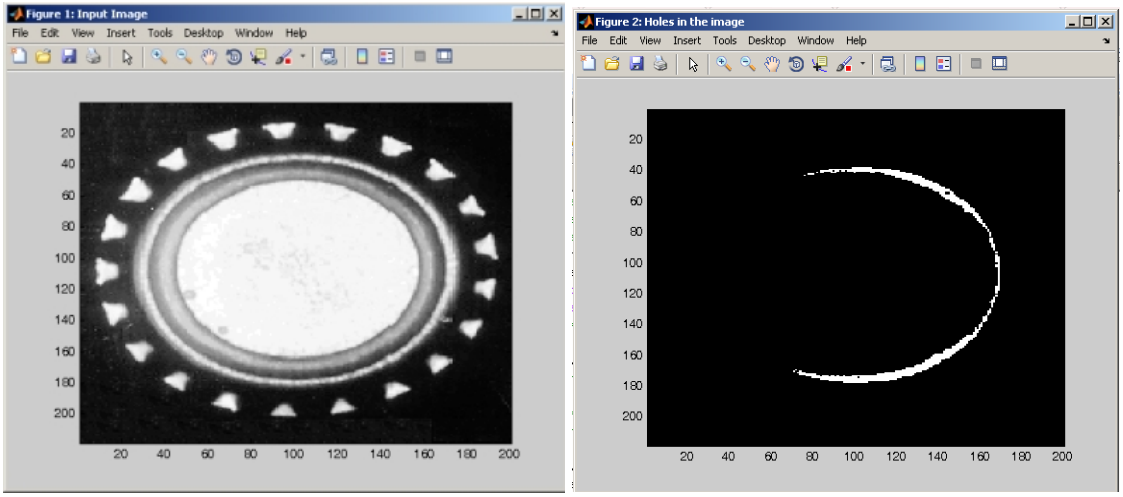
Isolate holes in a binary image.

function call:
[out_img1]=vsg('IsolateHoles',in_img1);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image or DICOM image.

Description:
out_img1 – returns an image of the same dimensions as the input with the holes of each white blob in the input image marked as white on the output image.

Example:
```
img = openimage('crown.bmp');
h=figure;image(img);set(h, 'Name','Input Image');
[out_img1]=vsg('Threshold',img,100);
[out_img1]=vsg('IsolateHoles',out_img1);
h=figure;image(uint8(out_img1));set(h,'Name','Holes in the image');
```



Notes:
Ensure that binary images have a range {0,255}. For DICOM images, the hole in each white blob on each slice is filled displayed, not the hole in the 3D blob.

IsolateBays

Isolate bays in a binary image.

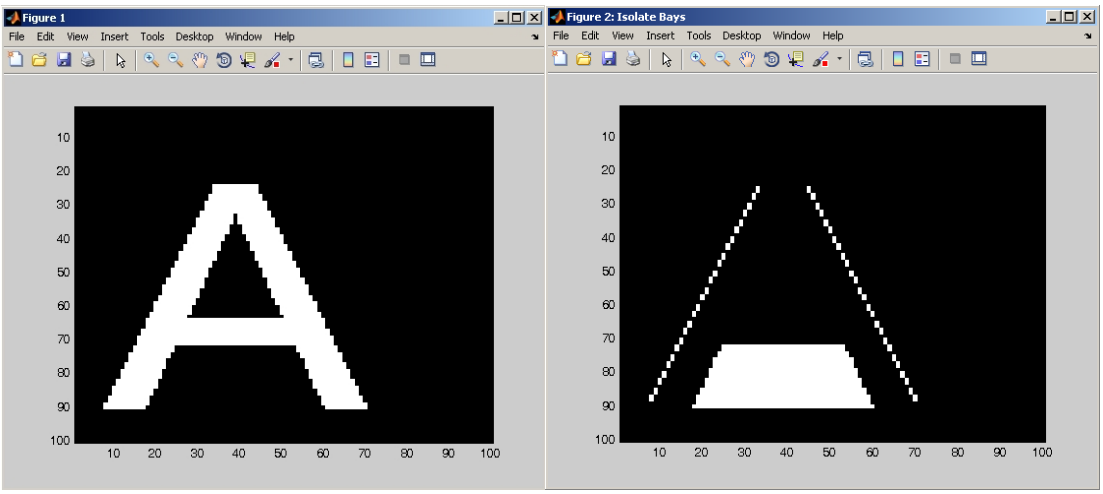
function call:
[out_img1]=vsg('IsolateBays',in_img1);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image or DICOM image.

Description:
The bays are the sections inside the convex hull and outside any enclosed region within the convex hull. The *out_img1* – returns an image of the same dimensions as the input with the bays of the input image returned. The bays are marked as white in the output image.

Example:

```
img = openimage('A.jpg');
figure;image(img);
[out_img1]=vsg('Threshold',img,100);
[out_img1]=vsg('IsolateBays',out_img1);
h=figure;image(uint8(out_img1));set(h,'Name','Isolate Bays');
```



Notes:
Ensure that binary images have a range {0,255}. For DICOM images, the bays on each slice are displayed, not the bays in the 3D blobs.

CountBlobs

Count the number of white bobs in a binary image (Range 0-255).

function call:
[num1]=vsg('CountBlobs',in_img1);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image or DICOM image.

Description:
The *num1* – returns an integer with the number of white blobs.

Example:

```
img = openimage('crown.jpg');
[out_img1]=vsg('Threshold',img,100);
[num1]=vsg('CountBlobs',out_img1)
num1 = 26
```

Notes:
Ensure that binary images have a range {0,255}. For DICOM images, the number of blobs on the first slice is returned.

SmallestBlob

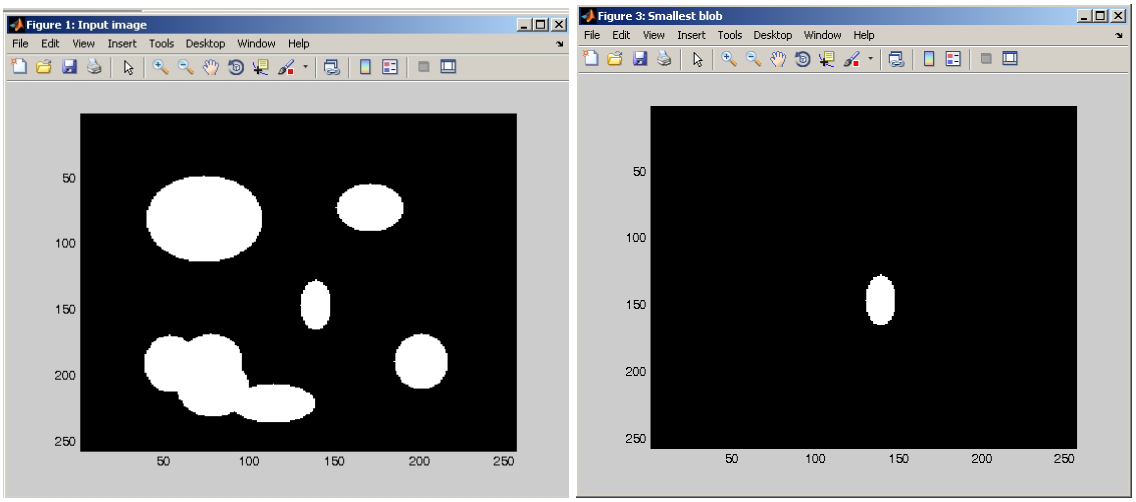
Extract the smallest white blob from a binary image.

function call:
[out_img1]=vsg('SmallestBlob',in_img1);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image.

Description:
Find the smallest, non-connecting, white blob in a binary image. *out_img1* – returns an image of the same dimensions as the input with the smallest input white blob displayed on it.

Example:
```
img = imread('blobs.jpg'); %color image
h=figure;image(img);set(h,'Name','Input Image');
[out_img1]=vsg('Threshold',img,150);
[out_img1]=vsg('SmallestBlob',out_img1);
h=figure;image(uint8(out_img1));set(h,'Name','Smallest blob');
```



Notes:
Ensure that binary images have a range {0,255}. For colour images, only the red channel is examined. If two blobs have the same size, only the first blob encountered in a raster scan direction is returned.

Bounding Box Functions

BoundingBox

Minimum area bounding rectangle.

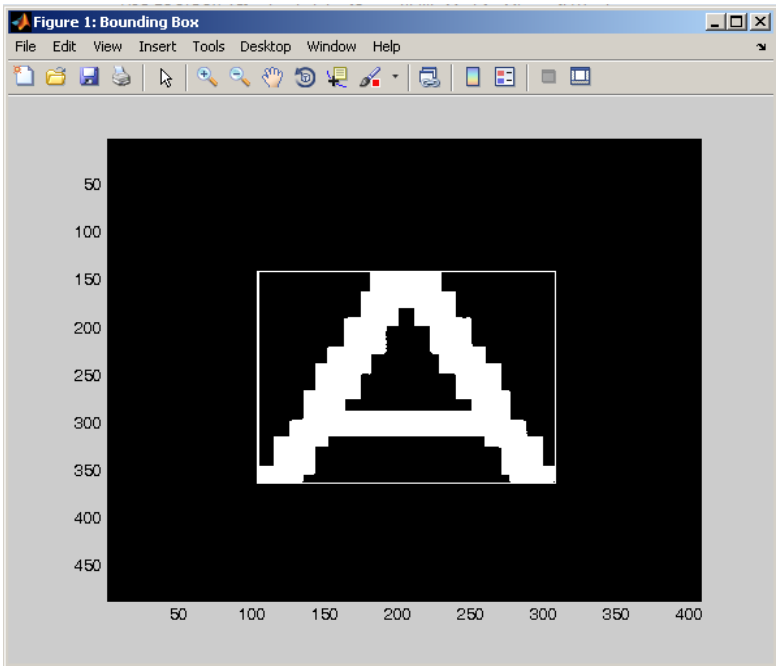
function call:
[out_img1]=vsg('BoundingBox',in_img1);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.

Description:
out_img1 – returns an image of the same dimensions as the input with the smallest possible rectangle that encloses all image objects drawn around the image objects.

Example:

```
img = imread('A.jpg');
[out_img1]=vsg('Threshold',img,100);
[out_img1]=vsg('BoundingBox',out_img1);
h=figure;image(uint8(out_img1));set(h,'Name','Bounding Box');
```



Notes:
For colour images, the bounding box surrounding all coloured objects is drawn. For DICOM images, the bounding box for each slice is calculated and displayed.

FillBounding

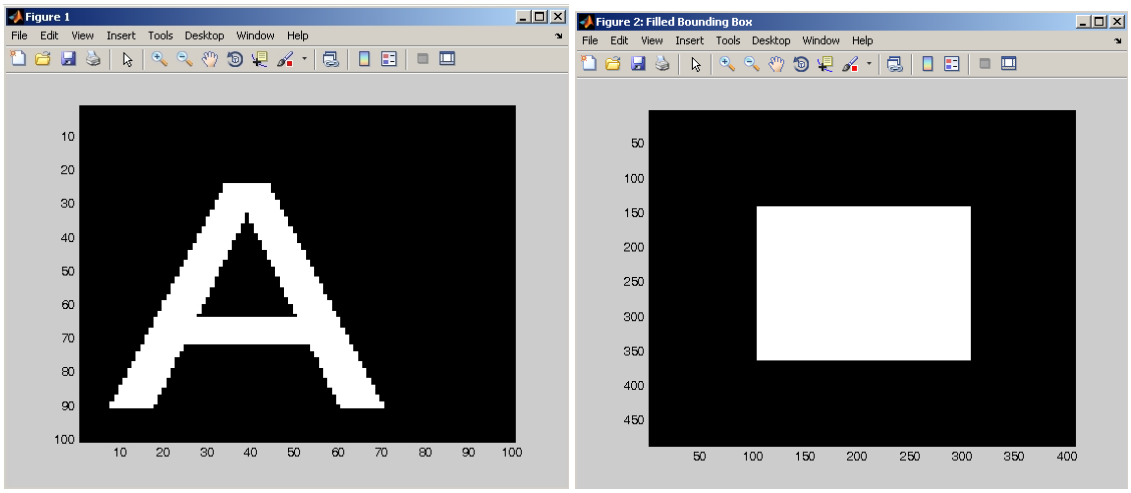
Filled minimum area bounding rectangle.

function call:
[out_img1]=vsg('FillBounding',in_img1);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.

Description:
out_img1 – returns an image of the same dimensions as the input with the smallest possible rectangle that encloses all image objects drawn over the image objects (filled rectangle).

Example:
```
img = imread('A.jpg'); %color image
figure;image(img);
[out_img1]=vsg('Threshold',img,100);
[out_img1]=vsg('FillBounding',out_img1);
h=figure;image(uint8(out_img1));set(h,'Name','Filled Bounding Box');
```



Notes:
For colour images, the bounding box surrounding all coloured objects is drawn. For DICOM images, the bounding box for each slice is calculated and displayed.

Blob Labelling Functions

Labeller

Assigns a unique value to each blob in the input image in a raster order.

function call:
[out_img1]=vsg('Labeller',in_img1);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.

Description:
out_img1 - An image which has dimensions the same as the input image. The output image displays individual blobs which have unique greyscale values for each blob.

Example:
```
img = imread('crown.jpg');
figure;image(img);
[out_img1]=vsg('Threshold',img,100);
[out_img1]=vsg('Labeller',out_img1);
h=figure;imagesc(uint8(out_img1(:,:,1)));set(h,'Name','Lebelled
Image');
```

Notes:
1. For DICOM images, the function operates on each slice separately.
2. For RGB images, the input on all 3 channels must be white for the area to be considered white.

LabelByArea

Assigns a unique value to each blob in the input image in order of area.

function call:
[out_img1]=vsg('LabelByArea',in_img1);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.

Description:
out_img1 - An image which has dimensions the same as the input image. The output image displays individual blobs which have unique greyscale values for each blob. The values increase according to the size of the blobs, i.e. the larger blobs will have larger greyscale values

Example:
```
img = imread('crown.jpg');
figure;image(img);
[out_img1]=vsg('Threshold',img,100);
[out_img1]=vsg('LabelByArea',out_img1);
h=figure;imagesc(out_img1(:,:,1));set(h,'Name','Label By Area Image');
```

Notes:
1. For DICOM images, the function operates on each slice separately.
2. For RGB images, the input on all 3 channels must be white for the area to be considered white.

Arithmetic Functions

Add

Image addition.

function call:
[out_img1]=vsg('Add',in_img1,in_img2);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.
in_img2 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.

Description
[out_img1]=vsg('Add',in_img1,in_img2) adds each element in image *in_img1* with the corresponding element in image *in_img2* and returns the sum in the corresponding element of the output image *out_img1*. The output - *out_img1* has dimensions that are the minimum size to fit both of the input images. Each of the input images are zero padded if their dimensions do not match. The output image is the normalised sum of the greyscale values of the input.

Example:
```
img1 = openimage('baboon.jpg');
img2 = openimage('A.jpg');
[out_img1]=vsg('Add',img1,img2);
figure, image(uint8(out_img1));
```

Notes:
For a single plane greyscale image being added to a 3-plane RGB image, the greyscale image will be added to the red channel of the RGB image because it gets zero-padded along the colour plane axis (i.e. zero padded with a G-plane and B-plane =0).

AND

Boolean AND operation.

function call:
[out_img1]=vsg('And',in_img1,in_img2);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.
in_img2 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.

Description
[out_img1]=vsg('And',in_img1,in_img2) applies bitwise AND operation on each element in image *in_img1* with the corresponding element in image *in_img2* and returns the results of the bitwise AND operation in the corresponding element of the output image - *out_img1*. The output - *out_img1* has dimensions that are the minimum size to fit both of the input images. Each of the input images are zero padded if their dimensions do not match.

Example:
```
img1 = openimage('baboon.jpg');
img2 = openimage('A.jpg');
[out_img1]=vsg('And',img1,img2);
figure, image(uint8(out_img1));
```

Notes:
Note the same padding as the “add” function.

Divide

Image division.

function call:
[out_img1]=vsg('Divide',in_img1,in_img2);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.
in_img2 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.

Description
[out_img1]=vsg('Divide',in_img1,in_img2) divides each element in the image *in_img1* by the corresponding element in image in_img2 and returns the result in the corresponding element of the output image *out_img1*. The output - *out_img1* has dimensions that are the minimum size to fit both of the input images. Each of the input images are zero padded if their dimensions do not match.

Example:
```
img1 = openimage('baboon.jpg');
img2 = openimage('A.jpg');
[out_img1]=vsg('Divide',img1,img2);
figure, image(uint8(out_img1));
```

Notes:
Note the same padding as the “add” function.

Maximum

Maximum of two images.

function call:
[out_img1]=vsg('Maximum',in_img1,in_img2);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.
in_img2 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.

Description
[out_img1]=vsg('Maximum',in_img1,in_img2) Finds the maximum values from a pixel-by-pixel comparison of the input images *in_img1* and *in_img2*. The output - *out_img1* has dimensions that are the minimum size to fit both of the input images. Each of the input images are zero padded if their dimensions do not match.

Example:
```
img1 = openimage('baboon.jpg');
img2 = openimage('A.jpg');
[out_img1]=vsg('Maximum',img1,img2);
figure, image(uint8(out_img1));
```

Notes:
Note the same padding as the “add” function.

Minimum

Minimum of two images.

function call:
[out_img1]=vsg('Minimum',in_img1,in_img2);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.
in_img2 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.

Description
[out_img1]=vsg('Minimum',in_img1,in_img2) Finds the minimum values from a pixel-by-pixel comparison of the input images *in_img1* and *in_img2*. The output - *out_img1* has dimensions that are the minimum size to fit both of the input images. Each of the input images are zero padded if their dimensions do not match.

Example:
```
img1 = openimage('baboon.jpg');
img2 = openimage('A.jpg');
[out_img1]=vsg('Minimum',img1,img2);
figure, image(uint8(out_img1));
```

Notes:
Note the same padding as the “add” function.

Multiply

Image multiply.

function call:
[out_img1]=vsg('Multiply',in_img1,in_img2);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.
in_img2 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.

Description:
[out_img1]=vsg('Multiply',in_img1,in_img2) multiplies each element in image *in_img1* by the corresponding element in image *in_img2* and returns the normalized product in the corresponding element of the output image *out_img1*. The output - *out_img1* has dimensions that are the minimum size to fit both of the input images. Each of the input images are zero padded if their dimensions do not match.

Examples:
```
img1 = openimage('baboon.jpg');
img2 = openimage('A.jpg');
[out_img1]=vsg('Multiply',img1,img2);
figure, image(uint8(out_img1));
```

Notes:
Note the same padding as the “add” function.

NOT

Boolean NOT operation.

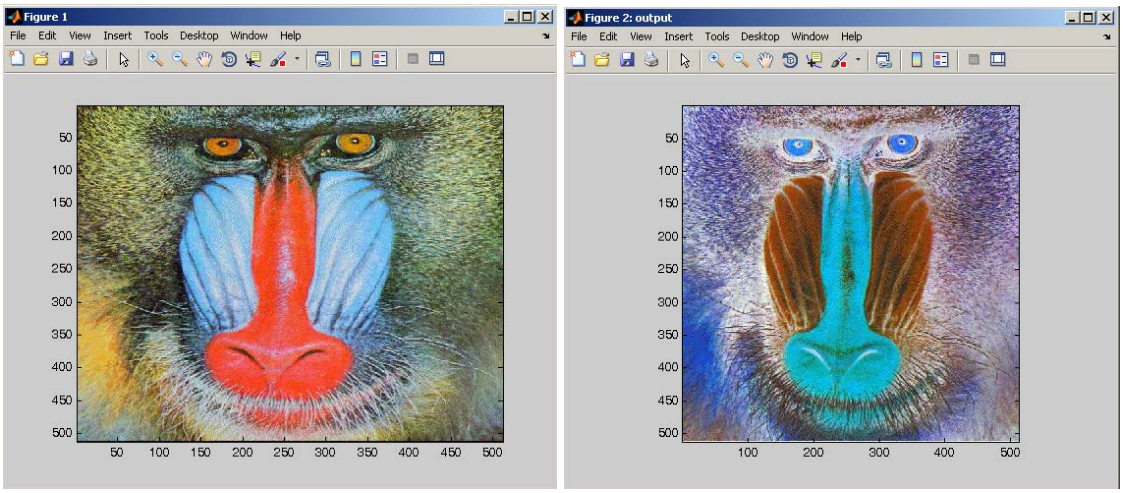
function call:
[out_img1]=vsg('NOT',in_img1);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.

Description:
out_img1 - An image with the same dimensions as the input image. The output image values are 255 minus the in_img1 values.

Examples:

```
img1 = openimage('baboon.jpg');
h=figure;image(uint8(img1));
[out_img1]=vsg('NOT',img1);
h=figure; image(uint8(out_img1));set(h,'Name','output');axis image;
```



OR

Boolean OR operation.

function call:
[out_img1]=vsg('OR',in_img1,in_img2);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.
in_img2 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.

Description
[out_img1]=vsg('OR',in_img1,in_img2) applies bitwise OR operation on each element in image *in_img1* with the corresponding element in image *in_img2* and returns the results of bitwise OR operation in the corresponding element of the output image *out_img1*. The output - *out_img1* has dimensions that are the minimum size to fit both of the input images. Each of the input images are zero padded if their dimensions do not match.

Example:

```
img1 = openimage('baboon.jpg');
```

```
img2 = openimage('A.jpg');
[out_img1]=vsg('OR',img1,img2);
figure, image(uint8(out_img1));
```

Notes:
Note the same padding as the “add” function.

Subtract

Image subtraction.

function call:
[out_img1]=vsg('Subtract',in_img1,in_img2);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.
in_img2 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.

Description:
[out_img1]=vsg('Subtract',in_img1,in_img2) subtracts each element in image *in_img2* from the corresponding element in image *in_img1* and returns the difference in the corresponding element of the output image *out_img1*. The output - *out_img1* has dimensions that are the minimum size to fit both of the input images. Each of the input images are zero padded if their dimensions do not match.

Example:
```
img1 = openimage('baboon.jpg');
img2 = openimage('A.jpg');
[out_img1]=vsg('Subtract',img1,img2);
figure, image(uint8(out_img1));
```

Notes:
Note the same padding as the “add” function.

XOR

Boolean Exclusive OR operation.

function call:
[out_img1]=vsg('XOR',in_img1,in_img2);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.
in_img2 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.

Description
[out_img1]=vsg('XOR',in_img1,in_img2) applies bitwise XOR operation on each element in image *in_img1* with the corresponding element in image *in_img2* and returns the results of bitwise XOR operation in the corresponding element of the output image *out_img1*. The output - *out_img1* has dimensions that are the minimum size to fit both of the input images. Each of the input images are zero padded if their dimensions do not match.

Example:
```
img1 = openimage('baboon.jpg');
img2 = openimage('A.jpg');
```

```
[out_img1]=vsg('XOR',img1,img2);
figure, image(uint8(out_img1));
```

Notes:
Note the same padding as the “add” function.

Filtering Functions

RAFilter

Rectangular Average Filter operation. Size of filter is user defined.

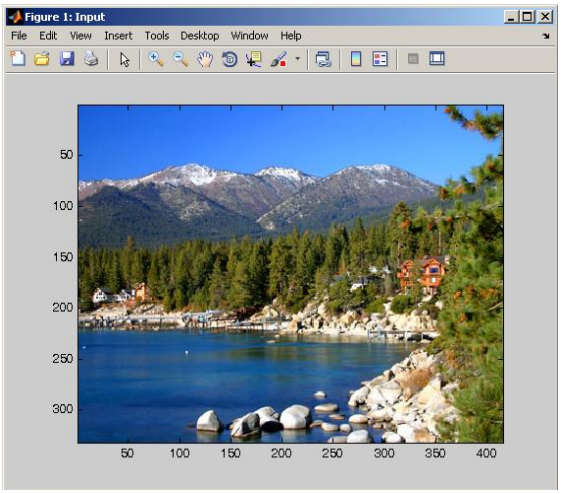
function call:
[out_img1]=vsg('RAFilter',in_img1,size);

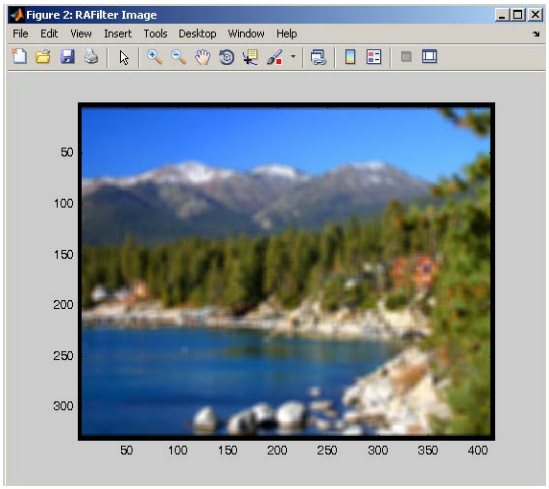
Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image or DICOM image.
size – an integer that specifies the size of the rectangle in which the average is calculated. This number actually represents the number of pixels from the centre which will be considered.

Description:
This function performs a rectangular average filter, of size (2*size+1)*(2*size+1). It is equivalent to a convolution of the specified array in which all the coefficients are ones. *out_img1* – returns an image of the same dimensions as the input with the average of the pixels in every (2*size+1)*(2*size+1) region displayed in the output.

Example:

```
img = openimage('lake.jpg');
h=figure;image(img);set(h,'Name','Input');
[out_img1]=vsg('RAFilter',img,3);
h=figure; image(uint8(out_img1));set(h,'Name','RAFilter Image');
```





AdaptSmooth

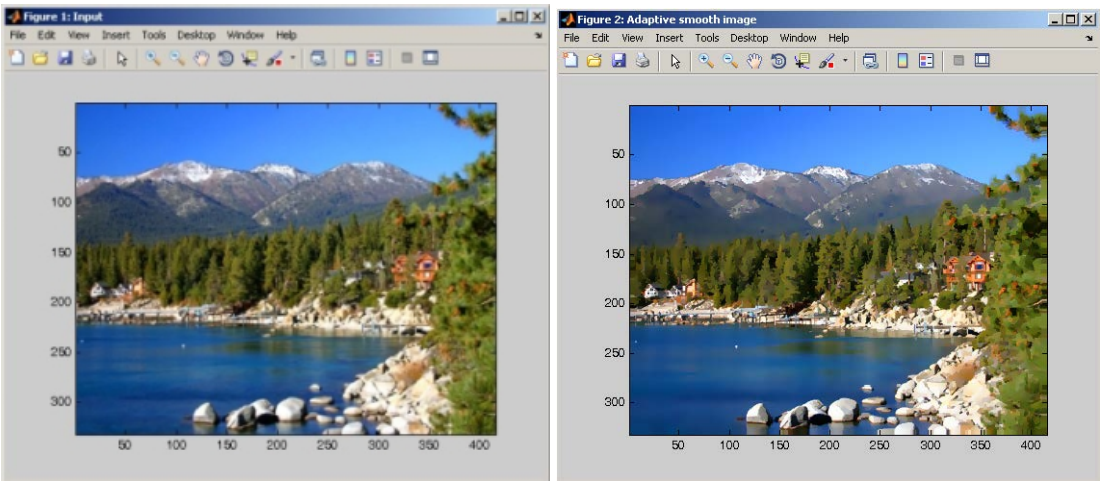
Adaptive smoothing of grey scale images. In order to apply it to colour images, the input image has to be split into RGB components and adaptive smooth has to be applied to each channel. If the colour image is applied directly the algorithm will smooth the average intensity image.

function call:
[out_img1]=vsg('AdaptSmooth',in_img1,num1,num2,num3);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.
num1 – an integer with the number of iterations that smoothing should be performed.
num2 – a double, the variance parameter.
num3 – a double, the diffusion parameter.

Description:
out_img1 - An image which has dimensions the same as the input image. The output image is an adaptively smoothed version of the input image.

Example:
```
img1 = openimage('lake.jpg');
h=figure;image(uint8(img1)); set(h,'Name','Input');
[out_img1]=vsg('AdaptSmooth',img1,10,0.3,10.0);
h=figure;image(uint8(out_img1));set(h,'Name','Adaptive smooth image');
```



Convolution

Convolution of an input image with an odd sized kernel (see Paul F. Whelan and Derek Molloy, "*Machine Vision Algorithm in Java – Techniques and implementation*", Springer, 2000.)

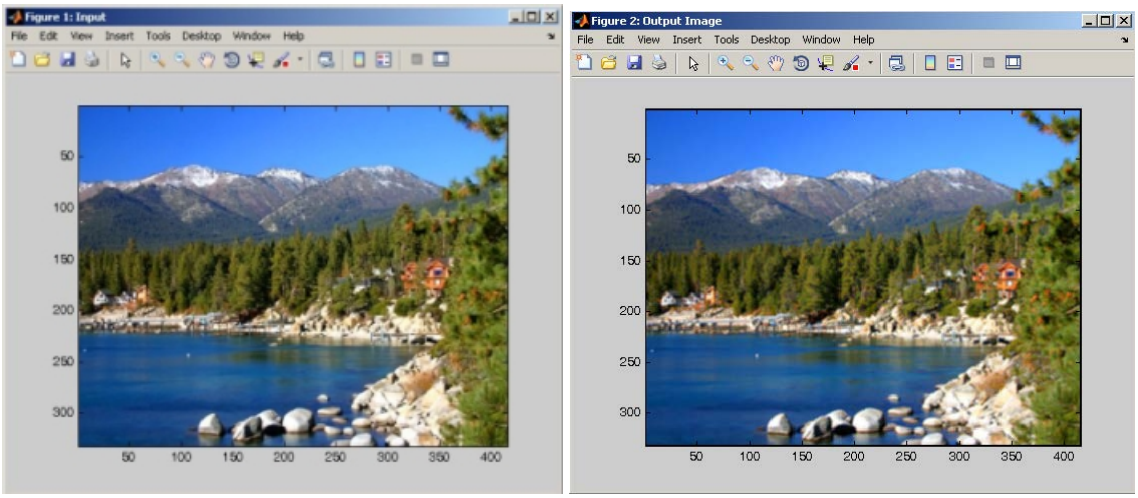
function call:
[out_img1]=vsg('Convolution',in_img1,conv_arr);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.
conv_arr – an array of doubles or integers. Can be non-square, but the dimensions must be odd, i.e. the array must be of size 3x5, 7x1 etc.

Return values:
out_img1 - An image which has dimensions the same as the input image. The output image is the convolution of the input image with the convolution kernel.

Example:
```
img = openimage('lake.jpg');
h=figure;image(uint8(img1)); set(h,'Name','Input');
```

```
conv_arr = [1.0,1.0,1.0;
            1.0,1.0,1.0;
            1.0,1.0,1.0];
[out_img1]=vsg('Convolution',img,conv_arr);
h=figure;image(uint8(out_img1));set(h,'Name','Output Image');
```



Diffusion

Diffusion filters the input image.

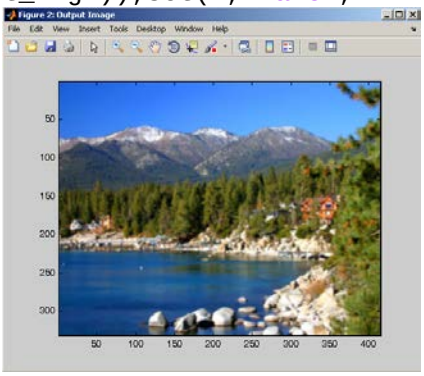
function call:
[out_img1]=vsg('Diffusion',in_img1,num1,num2);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.
num1 – a double specifying the diffusion parameter
num2 – an integer representing the number of iterations of the diffusion function.

Return values
out_img1 - an image which has dimensions the same as the input image. The output image is the input image after diffusion filtering.

Example:

```
img1 = openimage('lake.jpg');
[out_img1]=vsg('Diffusion',img1,5.0,20);
h=figure;image(uint8(out_img1));set(h,'Name','Diffusion');
```



DOLPS

DOLPS – Difference of low pass 3x3 filters. Image A results from applying 3 iterations of the low pass filter. Image B results from applying 6 iterations of the low pass filter. DOLPS = A-B.

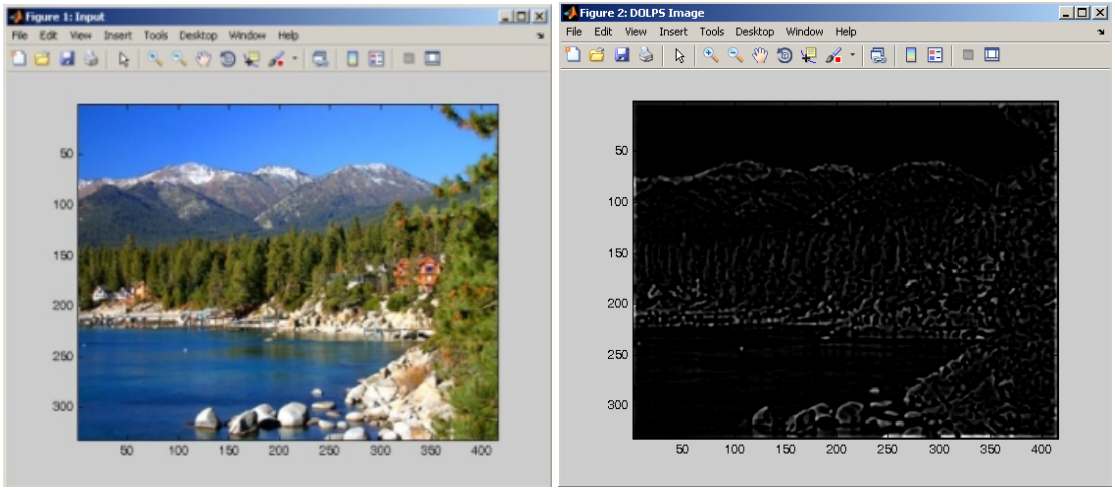
function call:
[out_img1]=vsg('DOLPS',in_img1);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.

Return values:
out_img1 - an image which has dimensions the same as the input image. The output image is the difference between the input image lowpass filtered 3 times and the input lowpass filtered 6 times.

Example:

```
img = openimage('lake.jpg'); %color image
h=figure;image(uint8(img1)); set(h,'Name','Input');
[out_img1]=vsg('DOLPS',img);
h=figure; imagesc(out_img1(:,:,1));colormap(gray);set(h,'Name','DOLPS
Image');
```



Gauss

Gaussian filters the input image.

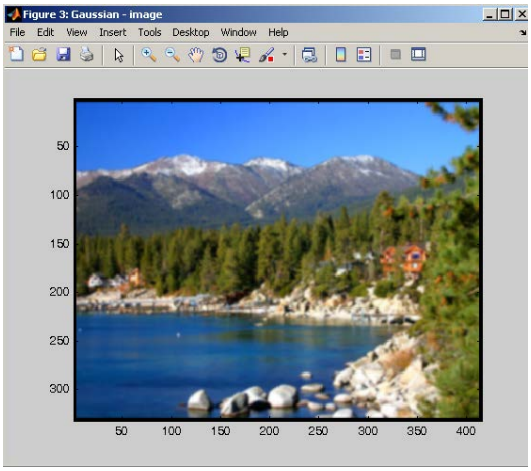
function call:
[out_img1]=vsg('Gauss',in_img1,std);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.
std – a double representing the standard deviation of the Gaussian kernel.

Return values:
out_img1 - an image which has dimensions the same as the input image. The output image is the input image after being Gaussian filtered.

Example:

```
img = openimage('lake.jpg');
[out_img2]=vsg('Gauss',img,1.0);
h=figure; image(uint8(out_img2));set(h,'Name','Gaussian - image');
```



LowPass

Low pass 3x3 filter.

function call:

[out_img1]=vsg('LowPass',in_img1);

Arguments:

in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.

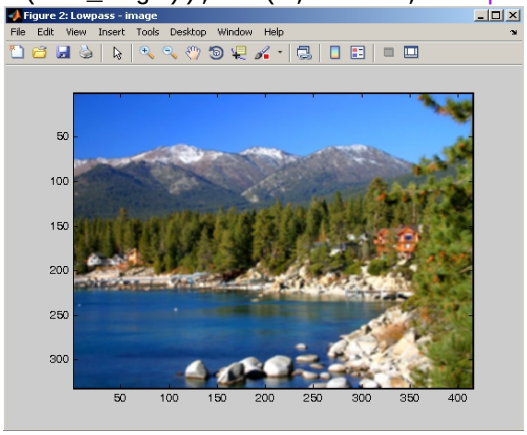
Description:

The lowpass filter is equivalent to a 3×3 convolution with a mask in which all the coefficient are set to 1. The result is normalised by dividing by 9.

out_img1 - an image which has dimensions the same as the input image. The output image is the input image after being LowPass filtered.

Example:

```
img = openimage('lake.jpg');
[out_img1]=vsg('LowPass',img);
h=figure; image(uint8(out_img1));set(h,'Name','Lowpass - image');
```



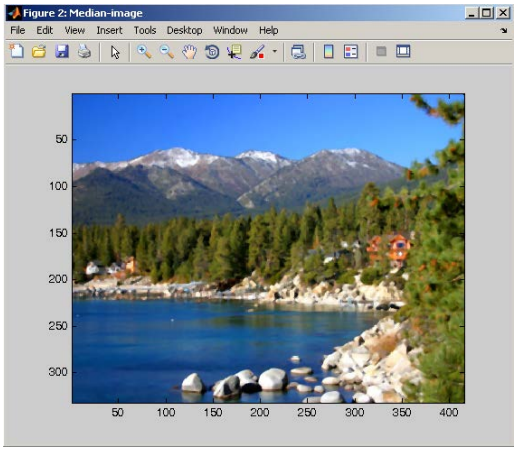
Median
Median 3x3 filter.

function call:
[out_img1]=vsg('Median',in_img1);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.

Description:
The Median filter is particularly useful for reducing the level of noise in an image.
out_img1 - an image which has dimensions the same as the input image. The output image is the input image after being median filtered.

Example:
```
img = openimage('lake.jpg'); %color image
[out_img1]=vsg('Median',img);
h=figure; image(uint8(out_img1));set(h,'Name','Median-image');
```



Midpoint
Midpoint 3x3 filter.

function call:
[out_img1]=vsg('Midpoint',in_img1);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.

Description:
The midpoint filter is based on the mid-value of pixels in a 3×3 neighboured.
out_img1 - an image which has dimensions the same as the input image. The output image is the input image after being midpoint filtered.

Example:
```
img = openimage('lake.jpg'); %color image
h=figure; image(img1);set(h,'Name','Input Image');
[out_img1]=vsg('Midpoint',img);
h=figure; image(uint8(out_img1));set(h,'Name','Midpoint filter');
```

LINFilter

Replace the central pixel of the 3x3 mask with the maximum value.

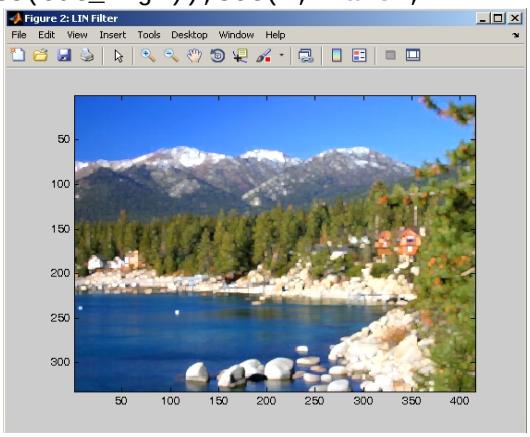
function call:
[out_img1]=vsg('LINFilter',in_img1);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.

Description:
The LinFilter has the effect of spreading bright regions and contracting dark regions.
out_img1 – an image of the same dimensions as the input image that outputs the largest value at each pixel location for that location and its 8 neighbours.

Example:

```
img = openimage('lake.jpg');
[out_img1]=vsg('LINFilter',img);
h=figure; image(uint8(out_img1));set(h,'Name','LIN Filter');
```



Notes:
For DICOM images, each slice is considered separately and 2D operators are used. RGB images are worked on plane by plane.

SINFilter

Replace the central pixel of the 3x3 mask with the minimum value.

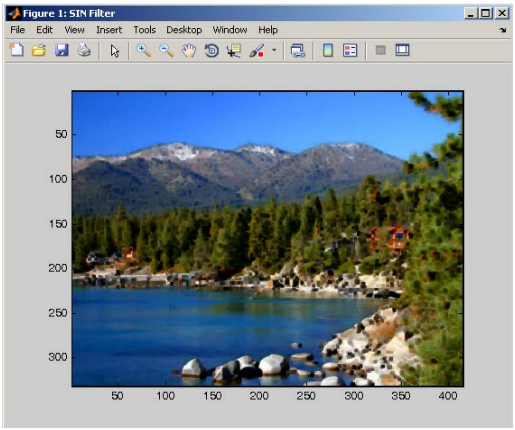
function call:
[out_img1]=vsg('SINFilter',in_img1);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.

Description:
The SINFilter has the effect of spreading dark regions and contracting bright ones.
out_img1 – an image of the same dimensions as the input image that shows outputs the smallest value at each pixel location for that location and its 8 neighbours.

Example:

```
img = openimage('lake.jpg');
[out_img1]=vsg('SINFilter',img);
h=figure; image(uint8(out_img1));set(h,'Name','SIN filter');
```



Notes:
For DICOM images, each slice is considered separately and 2D operators are used. RGB images are worked on plane by plane.

Sharpen

High pass 3x3 filter.

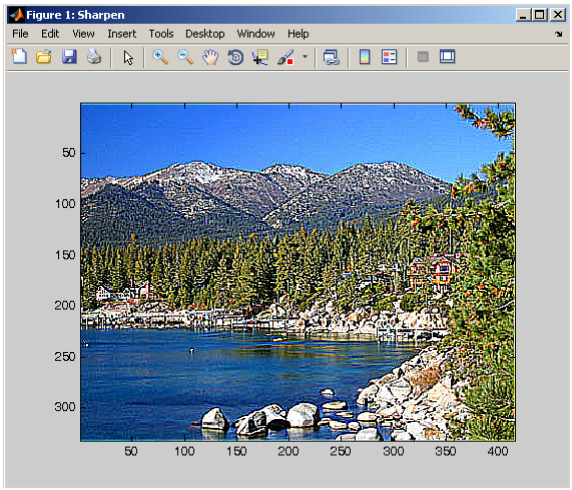
function call:
[out_img1]=vsg('Sharpen',in_img1);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.

Description:
The sharpen function emphasises high frequency regions within an image.
out_img1 - An image which has dimensions the same as the input image. The output image is the input image after being sharpened.

Example:

```
img = openimage('lake.jpg');
[out_img1]=vsg('Sharpen',img);
h=figure; image(uint8(out_img1));set(h,'Name','Sharpen');
```



Edge Functions

ZeroCross

Zero crossings edge detector.

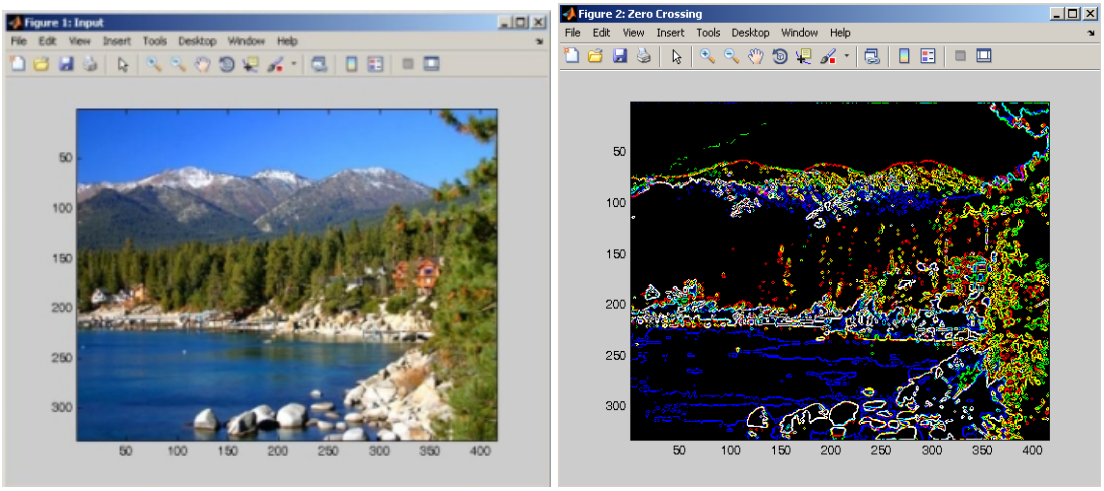
function call:
[out_img1]=vsg('ZeroCross',in_img1);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.

Description:
out_img1 - An image which has dimensions the same as the input image. The output image is the zero-crossing edge regions of the input image.

Example:

```
img = openimage('lake.jpg');
h=figure;image(img); set(h,'Name','Input');
[out_img1]=vsg('ZeroCross',img);
h=figure; image(uint8(out_img1));set(h,'Name','Zero Crossing');
```



Canny

Canny edge detector.

function call:
[out_img1,out_img2]=vsg('Canny',in_img1,std,thresh1,thresh2);

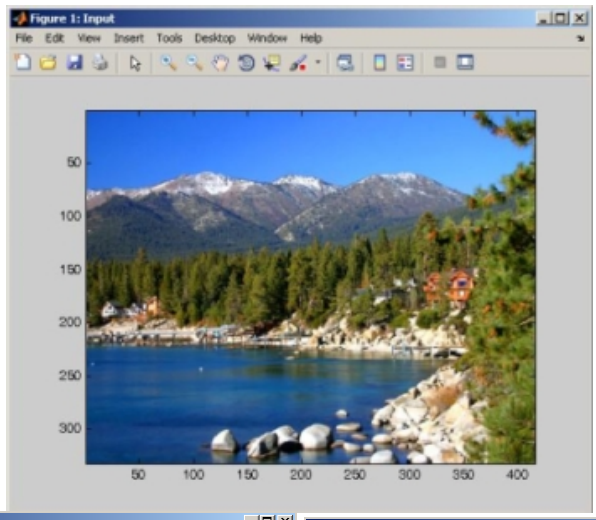
Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.
std – scale (standard deviation of the Gaussian).
thresh1 – lower magnitude threshold.
thresh2 – upper magnitude threshold.

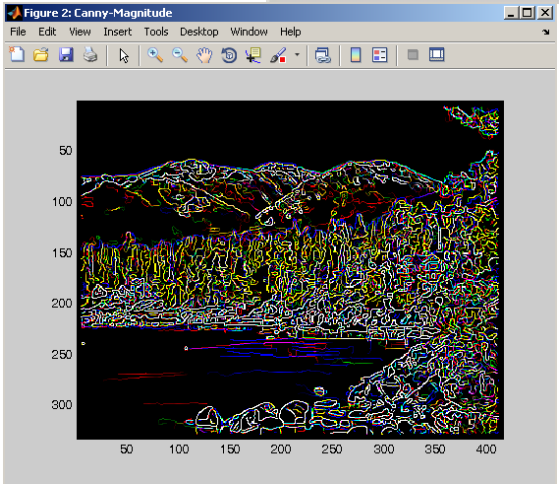
Description:
Canny edge detection involves convolving the input image with a Gaussian of scale std. Spurious response can be reduced by thresholding applied with hysteresis (two threshold are defined). If gradient magnitude of the contour is greater than the higher threshold than it is considered a valid

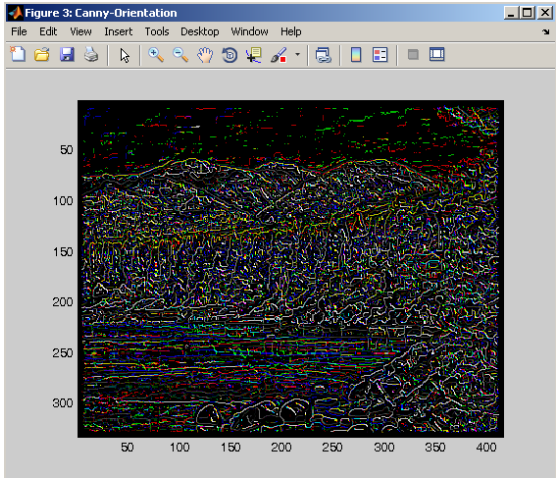
edge point. Any candidate edge pixels that are connected to valid edge points and are above the lower threshold are also considered as edge point. The *out_img1* - An image which has dimensions the same as the input image. The output image contains the magnitude of the edges detected.
out_img2 - An image which has dimensions the same as the input image. The output image contains the orientation of the edges detected.

Example:

```
img = openimage('lake.jpg');
h=figure;image(img); set(h,'Name','Input');
[out_img1,out_img2]=vsg('Canny',img,1.0,5,200);
h=figure; image(uint8(out_img1));set(h,'Name','Canny-Magnitude');
h=figure; image(uint8(out_img2));set(h,'Name','Canny-Orientation');
```







FreiChen

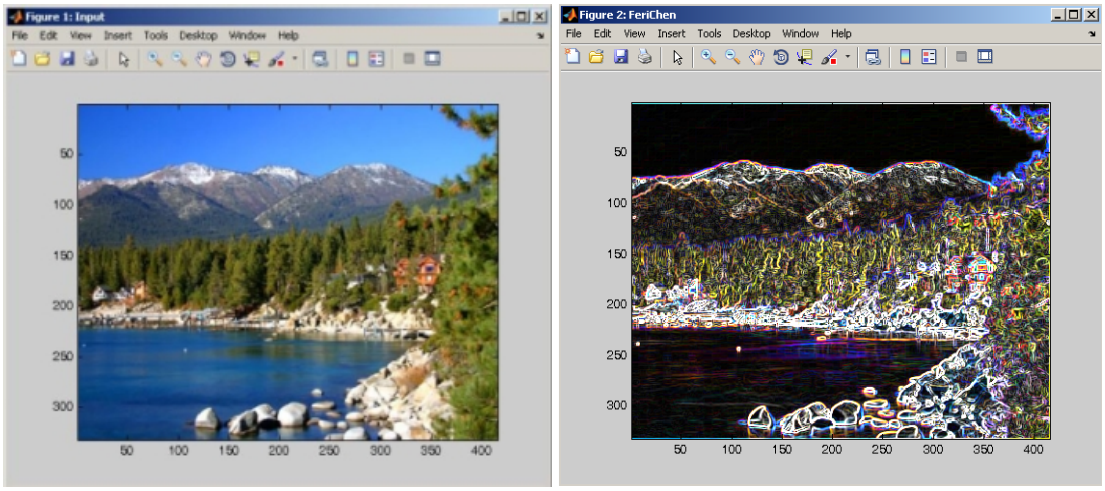
FreiChen edge detector.

function call:

[out_img1]=vsg('FreiChen',in_img1);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.
Description:
out_img1 - An image which has dimensions the same as the input image. The output image displays the result of using a freichen edge detection kernel on the image.

Example:
```
img = openimage('lake.jpg'); %color image
h=figure;image(img); set(h,'Name','Input');
[out_img1]=vsg('FreiChen',img);
h=figure; image(uint8(out_img1));set(h,'Name','FeriChen');
```



Notes:
For non-binary images or colour-planes/slices, corner points must have exactly 0 and 255 values within the cornerpoint detection kernel for that point to be flagged as a corner.

IntensityGradDir

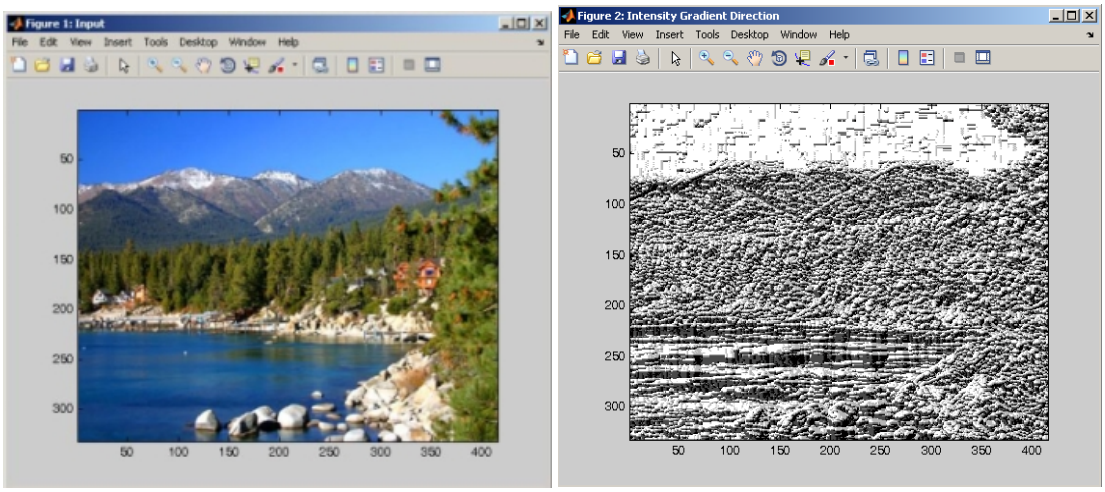
Compute the 3x3 intensity gradient direction. Gradients range from 1 to 8.

function call:
[out_img1]=vsg('IntensGradDir',in_img1);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.

Description:
out_img1 - An image which has dimensions the same as the input image. The output image displays the direction of the gradient of the image according to these values:
1 2 3
4 – 5
6 7 8

Example:
```
img = openimage('lake.jpg');
h=figure;image(img); set(h,'Name','Input');
[out_img1]=vsg('IntensGradDir',img);
h=figure;imagesc(out_img1(:,:,1));colormap(gray);set(h,'Name','Intensit
y Gradient Direction');
```



Notes:
For DICOM and RGB images, the function operates on each channel/slice separately.

Laplacian

Laplacian edge detector. User defined 4-connected or 8-connected neighbourhood.

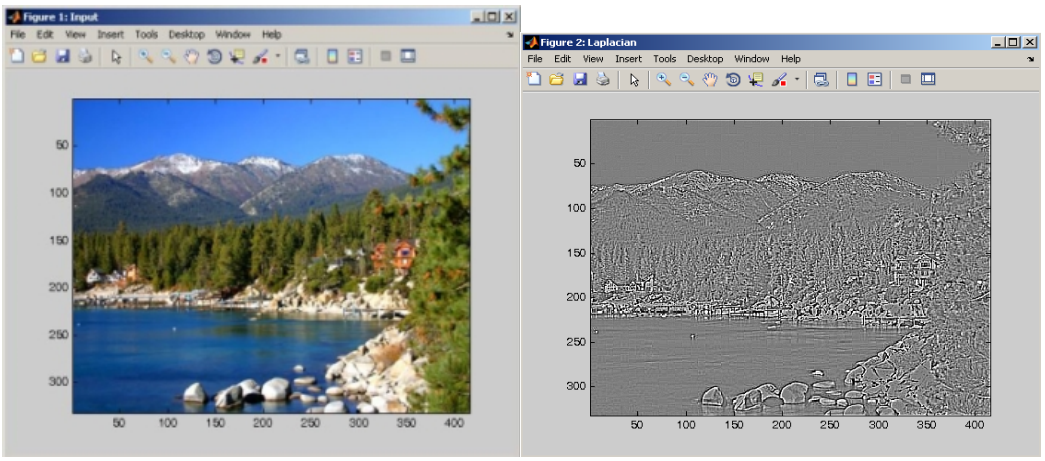
function call:
[out_img1]=vsg('Laplacian',in_img1,connected);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.
connected – an integer specifying whether a 4 or 8 connected scheme should be used.

Description:
The Laplacian function performs second-order derivative on the image. The *out_img1* - An image which has dimensions the same as the input image. The output image displays the result of using a laplacian kernel on the input image.

Example:

```
img = openimage('lake.jpg');
h=figure;image(img); set(h,'Name','Input');
[out_img1]=vsg('Laplacian',img,8);
h=figure;imagesc(out_img1(:,:,1));colormap(gray);set(h,'Name','Laplacia
n');
```



Notes:
For DICOM and RGB images, the function operates on each channel/slice separately.

NonMaxima

Edge detection using non maxima suppression.

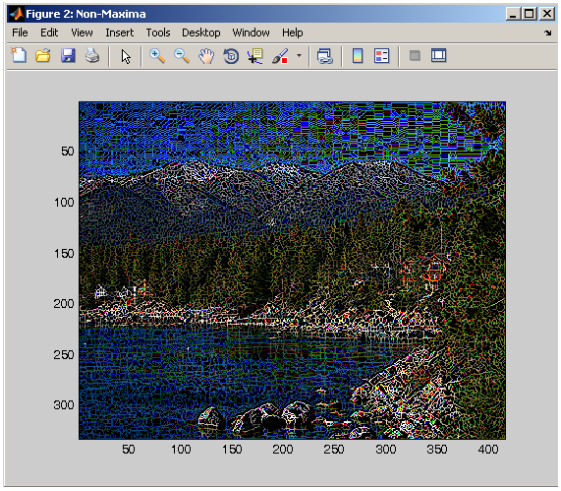
function call:
[out_img1]=vsg('NonMaxima',in_img1);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.

Description:
The *out_img1* - An image which has dimensions the same as the input image. The output image displays the NonMaxima filtered input image.

Example:

```
img = openimage('lake.jpg');
h=figure;image(img); set(h,'Name','Input');
[out_img1]=vsg('NonMaxima',img);
h=figure; image(uint8(out_img1));set(h,'Name','Non-Maxima');
```



Notes:
For DICOM and RGB images, the function operates on each channel/slice separately.

Prewitt

Prewitt edge detector.

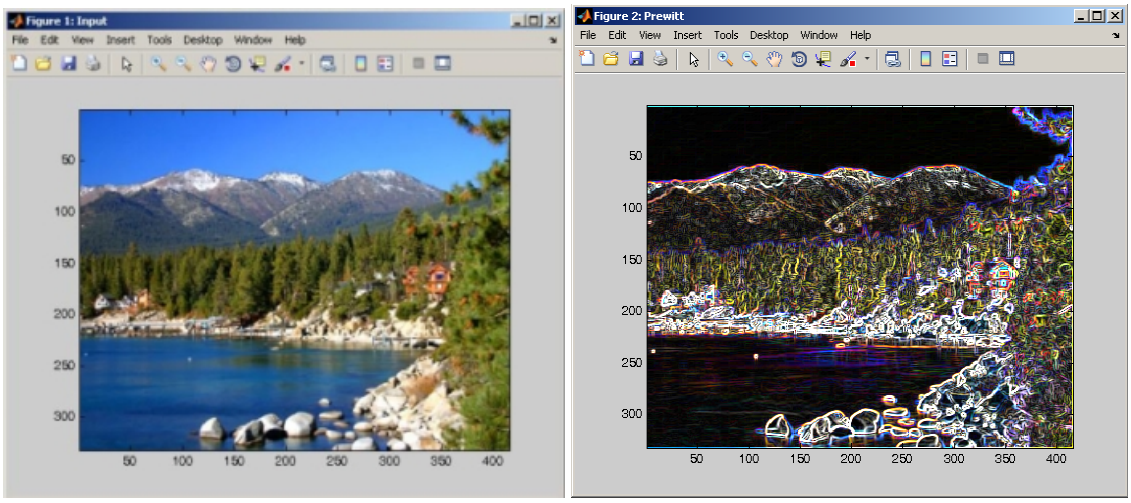
function call:
[out_img1]=vsg('Prewitt',in_img1);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.

Description:
The Prewitt edge detector is similar in nature to the Sobel edge detector, but is more sensitive to noise as it does posses the same inherent smoothing.
out_img1 - An image which has dimensions the same as the input image. The output image displays the prewitt filtered input image.

Example:

```
img = openimage('lake.jpg');
h=figure;image(img); set(h,'Name','Input');
[out_img1]=vsg('Prewitt',img);
h=figure; image(uint8(out_img1));set(h,'Name','Prewitt');
```



Notes:
For DICOM and RGB images, the function operates on each channel/slice separately.

Roberts

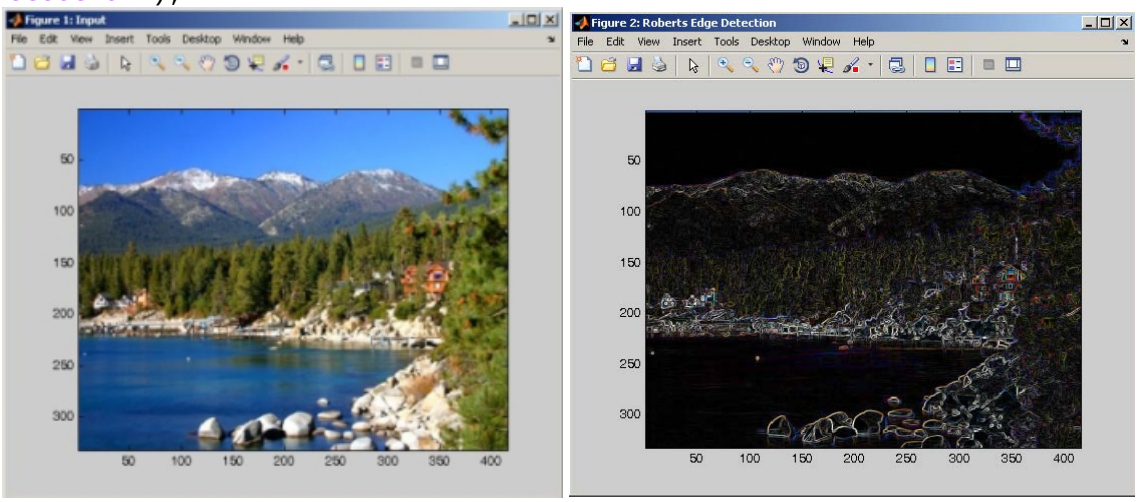
Roberts edge detector.

function call:
[out_img1]=vsg('Roberts',in_img1);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.
connected – an integer specifying whether a 4 or 8 connected scheme should be used.

Description:
The simplest gradient edge detector, the Robert cross operator, computes the first derivatives in two orthogonal directions. This is calculated from cross differences, implemented using two 2×2 convolution kernels.
out_img1 - An image which has dimensions the same as the input image. The output image displays the result of using a roberts kernel on the input image, which highlights edges.

Example:
```
img = openimage('lake.jpg');
h=figure;image(img); set(h,'Name','Input');
[out_img1]=vsg('Roberts',img);
h=figure; image(uint8(out_img1));set(h,'Name','Roberts Edge
Detection');
```



Notes:
For DICOM and RGB images, the function operates on each channel/slice separately.

Sobel
Sobel edge detector.

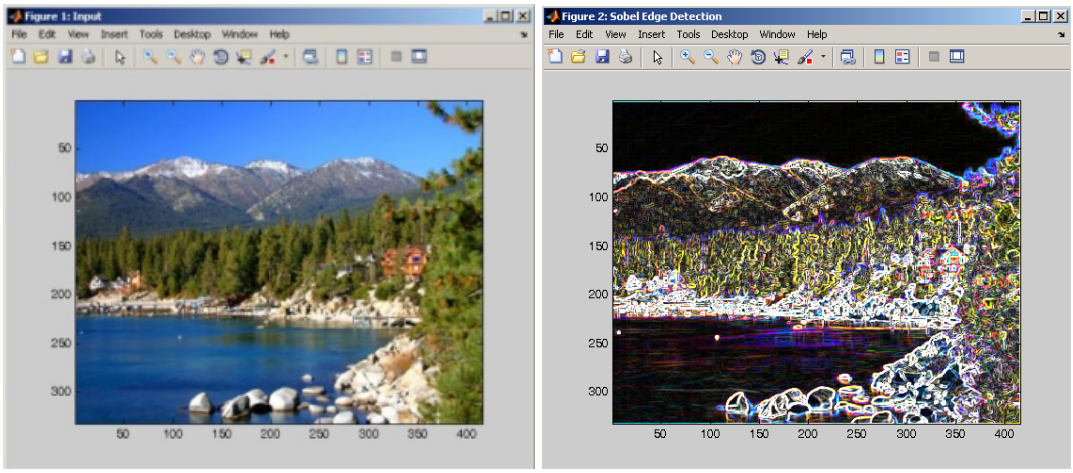
function call:
[out_img1]=vsg('Sobel',in_img1);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.

Description:
The Sobel edge detector uses a 3×3 mask to determine the gradient. The derivatives are calculated using the two convolution masks which are than summed. These have an inherent smoothing effect, which makes the very effective in the presence of noise.
out_img1 - An image which has dimensions the same as the input image. The output image displays the Sobel filtered input image, which highlights edges.

Example:
```
img = openimage('lake.jpg');
h=figure;image(img); set(h,'Name','Input');
[out_img1]=vsg('Sobel',img);
```

```
h=figure; image(uint8(out_img1));set(h,'Name','Sobel Edge Detection');
```



Notes:
For DICOM and RGB images, the function operates on each channel/slice separately.

Features Functions

ThinImg

The input binary image is thinned N times as specified by the user.

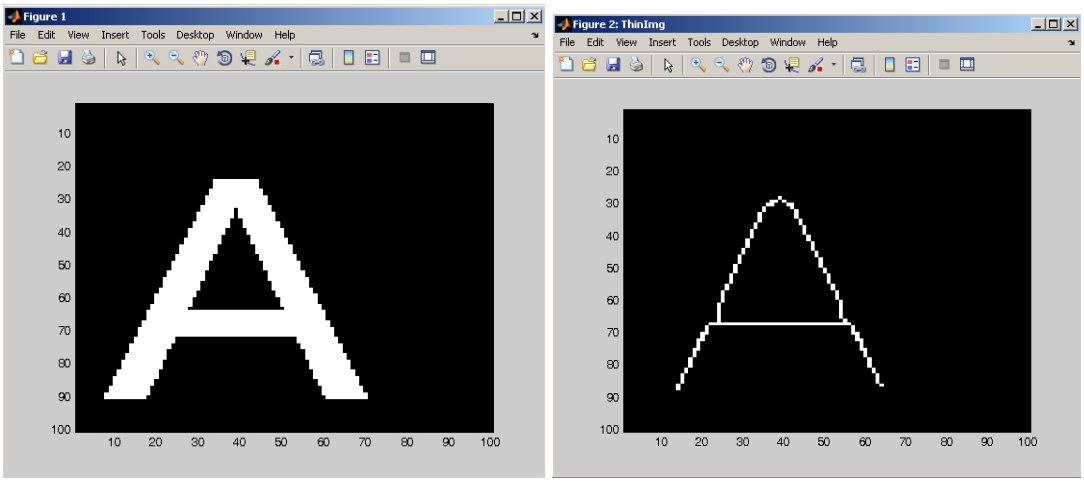
function call:
[out_img1]=vsg('ThinImg',in_img1,iters);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.
iters – an integer specifying the number of iterations to run the thinning algorithm.

Description:
Thinning allows the generation of match-stick like figures while keeping critically connected component.
out_img1 - An image which has dimensions the same as the input image. The output image displays a thinned version of the input white objects.

Example:

```
img = openimage('A.jpg');
h=figure;image(img);
[out_img1]=vsg('Threshold',img,100);
[out_img1]=vsg('ThinImg',out_img1,30);
h=figure; image(uint8(out_img1));set(h,'Name','ThinImg');
```



Notes:
1. For DICOM images, the function operates on each slice separately.
2. For RGB images, the function operates on each channel separately.

FullThin

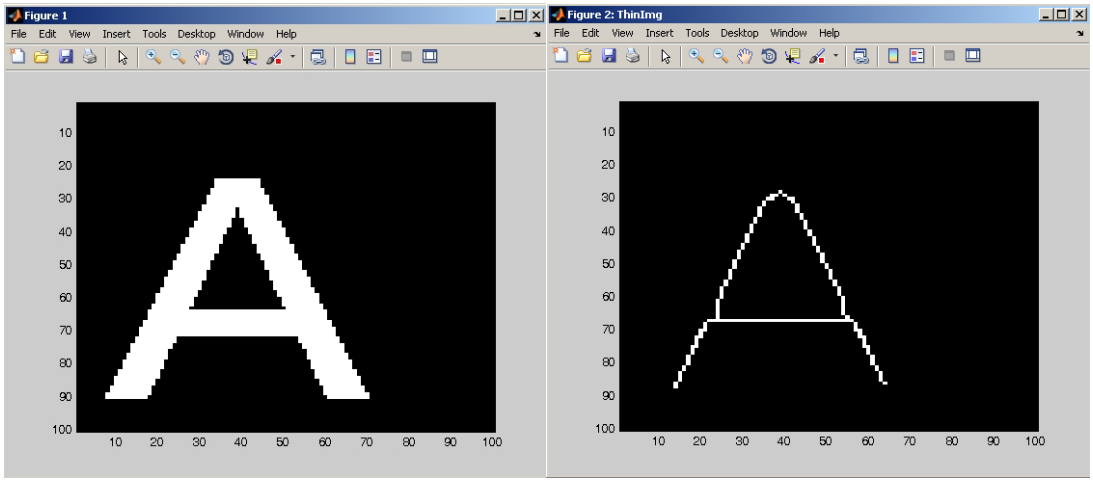
Full application of the thinning algorithm. Thin *till completion* resulting in a skeleton image.

function call:
[out_img1]=vsg('FullThin',in_img1);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.

Description:
Thinning allows the generation of match-stick like figures while keeping critically connected component.
out_img1 - An image which has dimensions the same as the input image. The output image displays a fully thinned version of the input white objects.

Examples:
```
img = openimage('A.jpg');
figure;image(img);
[out_img1]=vsg('Threshold',img,100);
[out_img1]=vsg('FullThin',out_img1);
h=figure;image(uint8(out_img1));set(h,'Name','Full Thin');
```



Notes:
1. For DICOM images, the function operates on each slice separately.
2. For RGB images, the function operates on each channel separately.

Junctions
Skeleton junction detection from a binary image based on a 3x3 region.

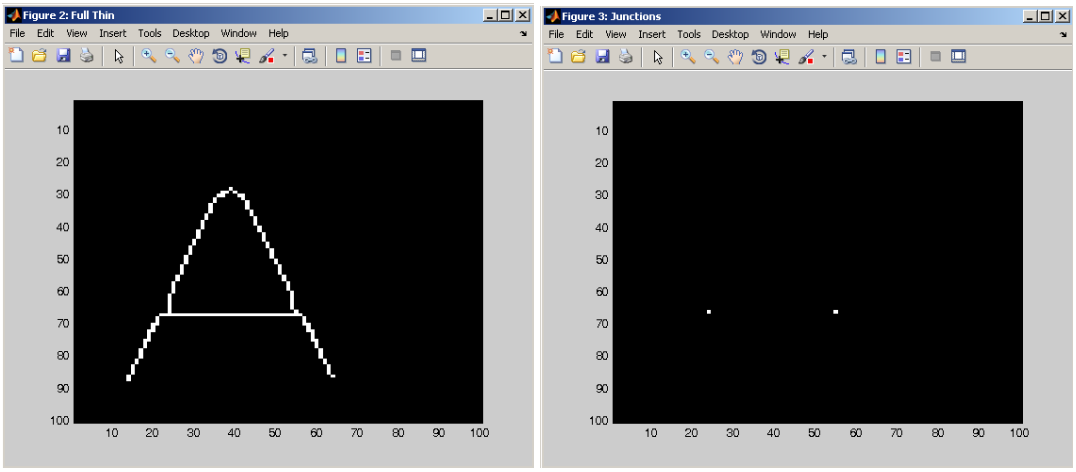
function call:
[out_img1]=vsg('Junctions',in_img1);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.

Description:
The 3×3 masks are designed to detect 3×3 skeleton junction and replace them by a single white pixel if the masks patterns occurs. Otherwise the output pixel is set to black.
out_img1 - An image which has dimensions the same as the input image. The output image displays the junctions of the white regions of the input image.

Example:
```
img = imread('A.jpg');
h=figure;image(img);
```

```
[out_img1]=vsg('Threshold',img,100);
[out_img1]=vsg('FullThin',out_img1);
h=figure; image(uint8(out_img1));set(h,'Name','Full Thin);
[out_img1]=vsg('Junctions',out_img1);
h=figure; image(uint8(out_img1));set(h,'Name','Junctions');
```



Notes:
1. For DICOM images, the function operates on each slice separately.
2. For RGB images, the function operates on each channel separately.

LimbEnds

Skeleton limb end detection from a binary image based on a 3x3 region.

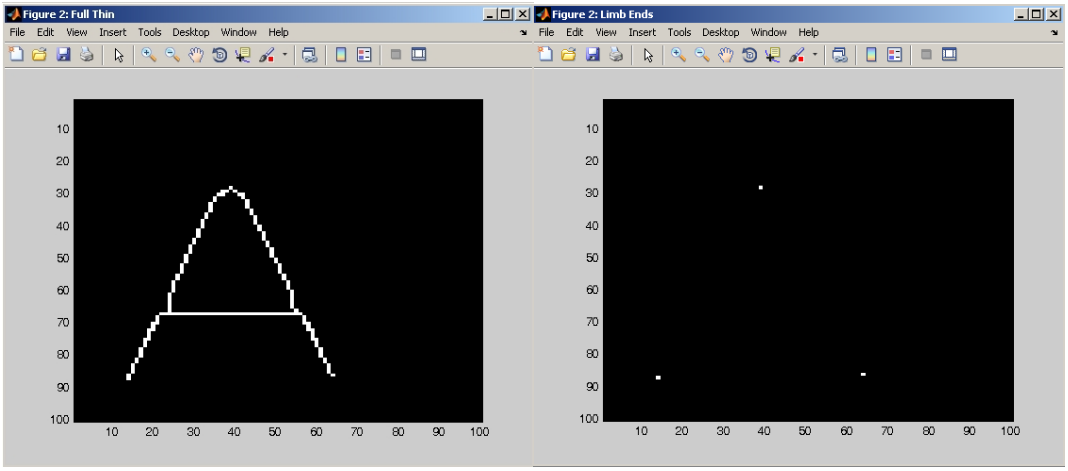
function call:
[out_img1]=vsg('LimbEnds',in_img1);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.

Description:
The LimdEnds function employed predefined masks for detecting 3×3 section end points. Once, detected, these limb end points are replaced by a single white pixel. Otherwise the output pixel is set to black.
out_img1 - An image which has dimensions the same as the input image. The output image displays the limbends of the white regions of the input image.

Example:

```
img = openimage('A.jpg');
[out_img1]=vsg('Threshold',img,100);
[out_img1]=vsg('FullThin',out_img1);
h=figure; image(uint8(out_img1));set(h,'Name','Full Thin');
[out_img1]=vsg('LimbEnds',out_img1);
h=figure; image(uint8(out_img1));set(h,'Name','Limb Ends');
```



Notes:
1. For DICOM images, the function operates on each slice separately.
2. For RGB images, the function operates on each channel separately.

Susan

Grey Scale (SUSAN) corner detector.

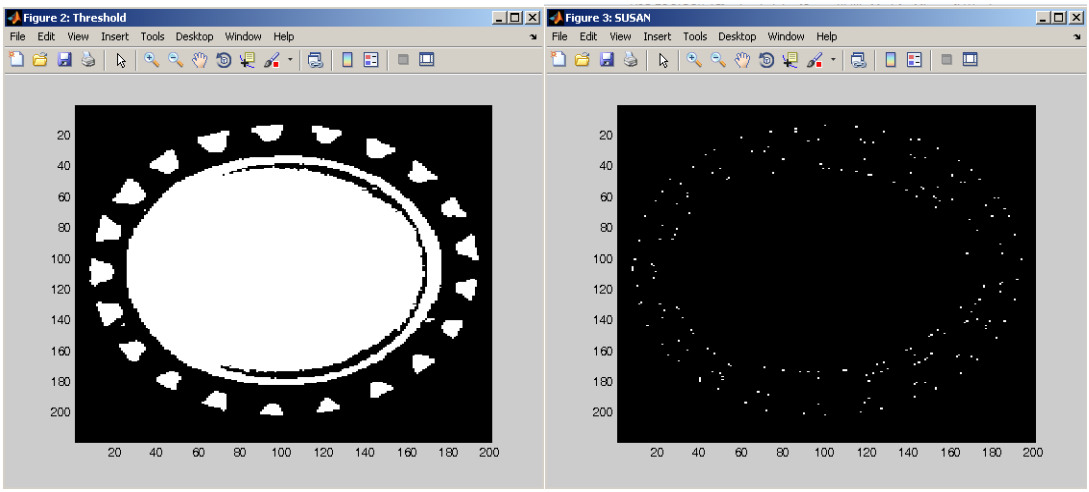
function call:
[out_img1]=vsg('Susan',in_img1,num1,num2);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.
num1 – an integer specifying the brightness threshold.
num2- - an integer specifying the geometric threshold.

Description:
This computationally efficient approach to the detection of gray scale corners is based on the application of a small circular mask to an image. As this mask is moved over the image in a raster scan fashion it examines the local information to determine the likelihood of a valid grey scale corner.
out_img1 - An image which has dimensions the same as the input image. The output image displays SUSAN corners of the input image.

Example:

```
img = openimage('crown.jpg');
h=figure;image(img); set(h,'Name','Input Image');
[out_img1]=vsg('Threshold',img,100);
h=figure; image(uint8(out_img1));set(h,'Name','Threshold');
[out_img1]=vsg('Susan',out_img1,100,10);
h=figure; image(uint8(out_img1));set(h,'Name','SUSAN');
```



Notes:

1. For DICOM images, the function operates on each slice separately.
2. For RGB images, the function operates on each channel separately.

Histogram Functions

AvgIntensity

Compute the average intensity of the input image.

function call:
[num1]=vsg('AvgIntensity',in_img1);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.

Description:
num1 – a double representing the average intensity for the image.

Example:
```
img1 = openimage('lake.jpg');
[num1]=vsg('AvgIntensity',img1);

num1 =  108.3735
```

Notes:
1. For DICOM images, the function finds the average intensity across all slices.
2. For RGB images, the function finds the average value across all colour planes.

EntropyCalc

Compute the entropy of the input image. Entropy is a statistical measure of randomness that can be used to characterise the texture of an input image.

function call:
[num1]=vsg('EntropyCalc',in_img1);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.

Description:
The *num1* – a double representing the greyscale intensity entropy for the image.

Example:
```
img1 = openimage('lake.jpg');
[num1]=vsg('EntropyCalc',img1)
num1 = 5.3307
```

Notes:
1. For DICOM images, the function finds the entropy across all slices.
2. For RGB images, the function finds the entropy across all colour planes.

HighestGrey
Compute the highest grey level from the input image.

function call:
[num1]=vsg('HighestGrey',in_img1);
Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.

Description:
num1 – a double representing the highest greyscale intensity for the image.

Example:
```
img1 = openimage('lake.jpg');
[num1]=vsg('HighestGrey',img1)

num1 =    255
```

Notes:
1. For DICOM images, the function finds the highest value across all slices.
2. For RGB images, the function finds the highest value across all colour planes.

KurtosisCalc
Compute the kurtosis of the input image. Kurtosis is a measure of how outlier-prone a distribution is.

function call:
[num1]=vsg('KurtosisCalc',in_img1);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.

Description:
num1 – a double representing the greyscale intensity kurtosis for the image.

Example:
```
img = openimage('lake.jpg');
[num1]=vsg('KurtosisCalc',img)

num1 = -0.1724
```

Notes:
1. For DICOM images, the function finds the kurtosis value across all slices.
2. For RGB images, the function finds the kurtosis value across all colour planes.

LowestGrey
Compute the lowest grey level from the input image.

function call:
[num1]=vsg('LowestGrey',in_img1);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.

Description:
num1 – a double representing the lowest greyscale intensity for the image.

Example:
```
img1 = openimage('lake.jpg');
[num1]=vsg('LowestGrey',img1)
num1 =  0
```

Notes:
1. For DICOM images, the function finds the lowest value across all slices.
2. For RGB images, the function finds the lowest value across all colour planes.

MSECalc

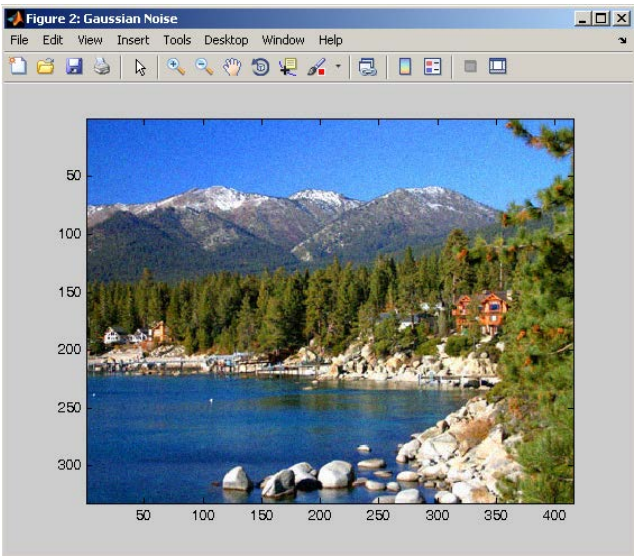
Compare the input images using the mean square error operation.

function call:
[num1]=vsg('MSECalc',in_img1,in_img2);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.
in_img2 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.

Description:
num1 – a double representing the mean squared error per pixel between *in_img1* and *in_img2*. Each image is zero padded to match in size for the calculations.

Example:
```
img1 = openimage('lake.jpg');
[img2]=vsg('GaussianNoise',img1,10.0);
h=figure;image(uint8(img2));set(h,'Name','Gaussian Noise');
[num1]=vsg('MSECalc',img1,img2)
```



```
num1 =   93.7575
```

Notes:
1. For DICOM images, the function calculates the MSE value across all slices and returns a per "voxel" MSE value.
2. For RGB images, the function calculates the MSE value across all colour planes, and returns a per colour plane pixel error.

SkewCalc

Compute the skewness deviation of the input image.

function call:
[num1]=vsg('SkewCalc',in_img1);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.

Description:
num1 – a double representing the greyscale intensity skewness of in_img1.

Example:
```
img = openimage('lake.jpg');
[num1]=vsg('SkewCalc',img)

num1 =   0.6827
```

Notes:
1. For DICOM images, the function calculates the skewness value across all slices
2. For RGB images, the function calculates the skewness value across all colour planes.

STDCalc

Compute the standard deviation of the input image.

function call:
[num1]=vsg('STDCalc',in_img1);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.

Description:
num1 – a double representing the greyscale intensity standard deviation of in_img1.

Example:
```
img = openimage('lake.jpg');
[num1]=vsg('STDCalc',img)

num1 = 65.4649
```

Notes:
1. For DICOM images, the function calculates the std value across all slices
2. For RGB images, the function calculates the std value across all colour planes.

VarianceCalc

Compute the variance of the input image.

function call:
[num1]=vsg('VarianceCalc',in_img1);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.

Description:
num1 – a double representing the greyscale intensity variance of in_img1.

Example:
```
img = openimage('lake.jpg');
[num1]=vsg('VarianceCalc',img)

num1 = 4.2857e+003
```

Notes:
1. For DICOM images, the function calculates the variance value across all slices
2. For RGB images, the function calculates the variance value across all colour planes.

LocalEq3x3

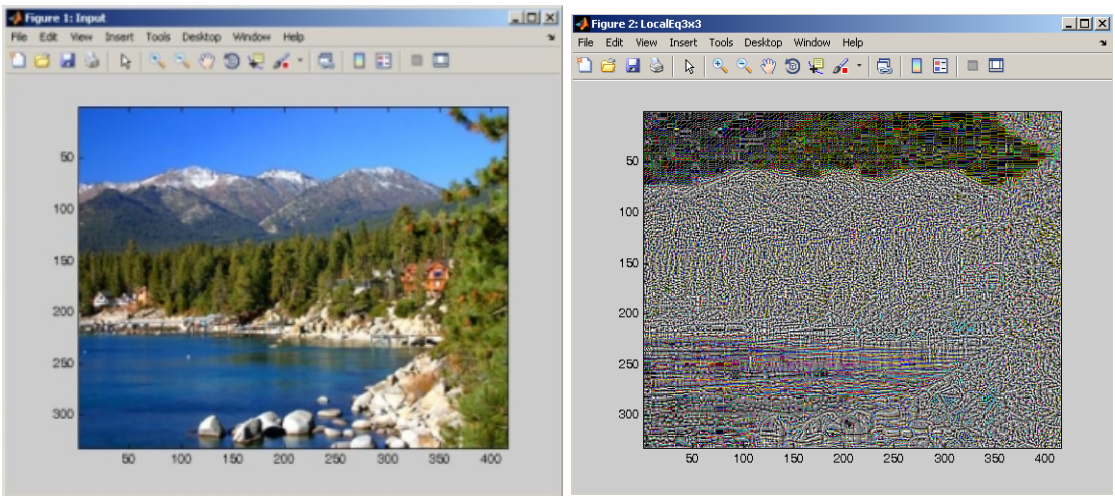
Local histogram equalisation using a 3x3 region.

function call:
[out_img1]=vsg('LocalEq3x3',in_img1);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.

Description:
This function relies upon the application of histogram equalisation within a small (3×3) local area or window (3×3). The number of pixels in a small window (3×3) that are darker than the central pixel are counted. This number defines the intensity at the equivalent point in the output image.
out_img1 – An image which has dimensions the same as the input image. The output image displays local equalisation of the input image.

Example:
```
img = openimage('lake.jpg');
h=figure;image(uint8(img)); set(h,'Name','Input');
[out_img1]=vsg('LocalEq3x3',img);
h=figure;image(uint8(out_img1));set(h,'Name','LocalEq3x3');
```



Notes:
1. For DICOM images, the function operates on all slices separately.
2. For RGB images, the function operates on all colour planes separately.

LocalEq5x5

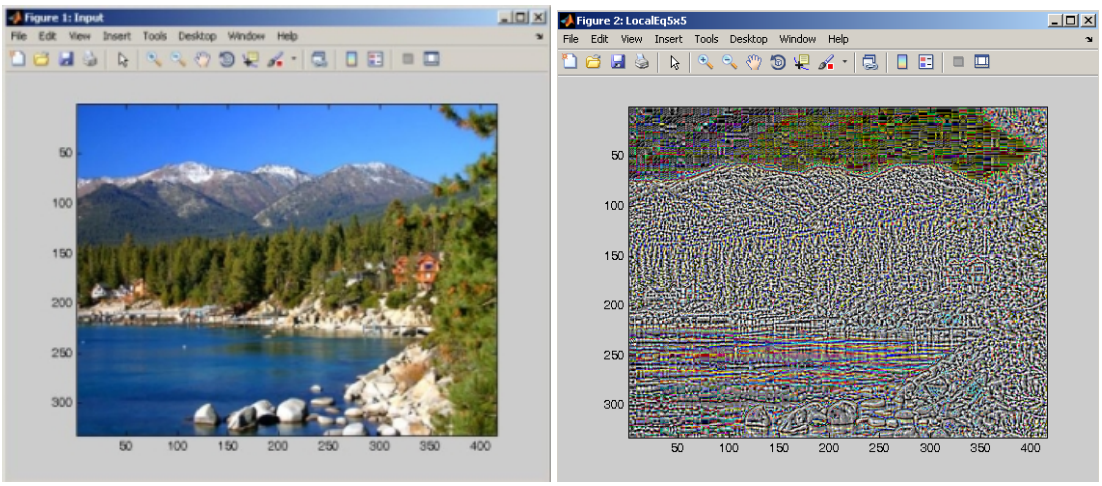
Local histogram equalisation using a 5x5 region.

function call:
[out_img1]=vsg('LocalEq5x5',in_img1);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.

Description:
This function relies upon the application of histogram equalisation within a small (5×5) local area or window (5×5). This number of pixels in a small window (5×5) that are darker than the central pixel are counted. The number defines the intensity at the equivalent point in the output image.
out_img1 – An image which has dimensions the same as the input image. The output image displays local equalisation of the input image.

Example:
```
img = openimage('lake.jpg');
h=figure;image(uint8(img)); set(h,'Name','Input');
[out_img1]=vsg('LocalEq5x5',img);
h=figure;image(uint8(out_img1));set(h,'Name','LocalEq5x5');
```



Notes:

1. For DICOM images, the function operates on all slices separately.
2. For RGB images, the function operates on all colour planes separately.

Morphology Functions

DBLT

Double [Hysteresis] Threshold based reconstruction. Binary Outputs. Seed threshold to reduce noise Mask threshold to maximise signal.

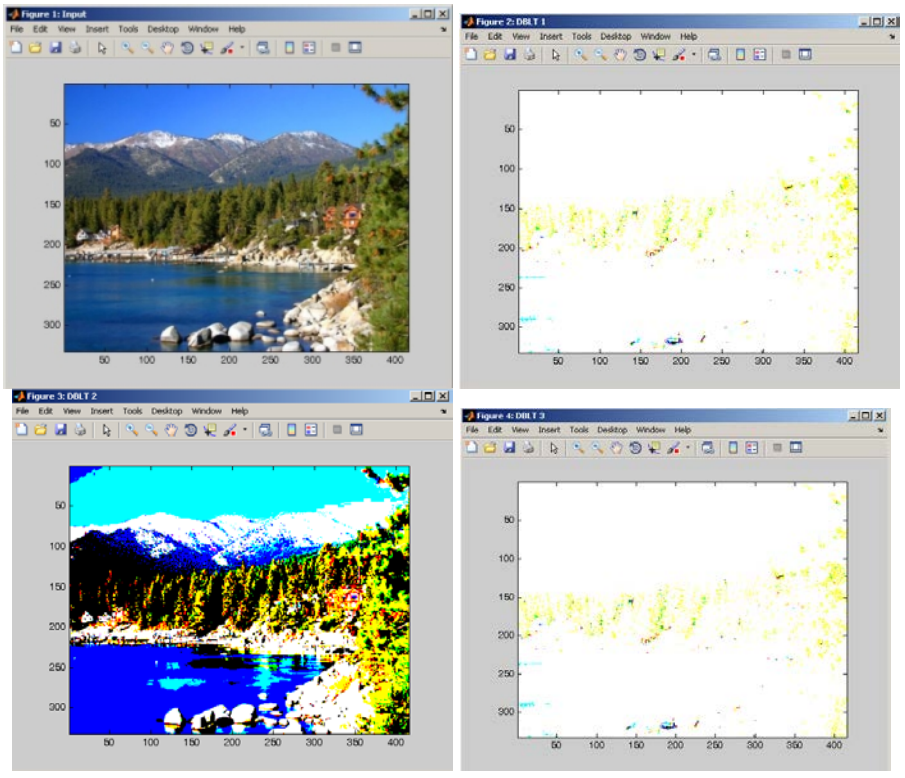
function call:
[out_img1,out_img2,out_img3]=vsg('DBLT',in_img1,num1,num2);

Arguments:
in_img1 mask image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.
num1 - seed threshold, an integer representing the threshold to be used for the seed image.
num2 - mask threshold, an integer representing the threshold to be used for the mask image.

Description:
out_img1 – An image which has dimensions the same as the input image. This is an image which as been reconstructed by dilation using the images formed by thresholding the input image with the seed and mask thresholds.
out_img2 – seed image, the image formed by thresholding the input image at the seed threshold.
out_img3 – mask image, the image formed by thresholding the input image at the mask threshold.

Example:

```
img = openimage('lake.jpg');
h=figure;image(uint8(img));set(h,'Name','Input');
[out_img1,out_img2,out_img3]=vsg('DBLT',img,100,3);
h=figure;image(uint8(out_img1));set(h,'Name','DBLT 1');
h=figure;image(uint8(out_img2));set(h,'Name','DBLT 2');
h=figure;image(uint8(out_img3));set(h,'Name','DBLT 3');
```



Notes:
1. For DICOM images, the function operates on all slices separately.
2. For RGB images, the function operates on all colour planes separately.

Dilation

Dilation operation (user user specifies the connectivity of the structuring element - 4 or 8).

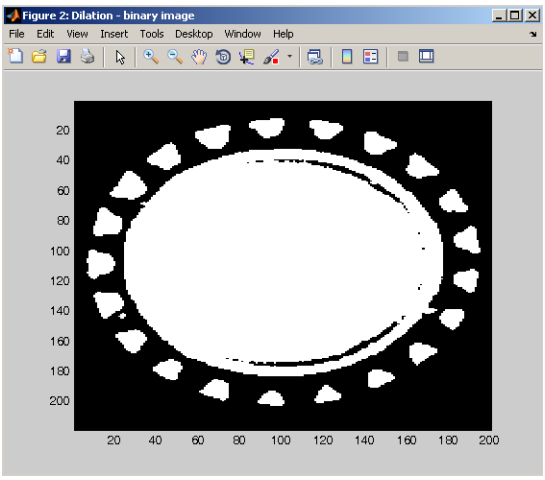
function call:
[out_img1]=vsg('Dilation',in_img1,connected);

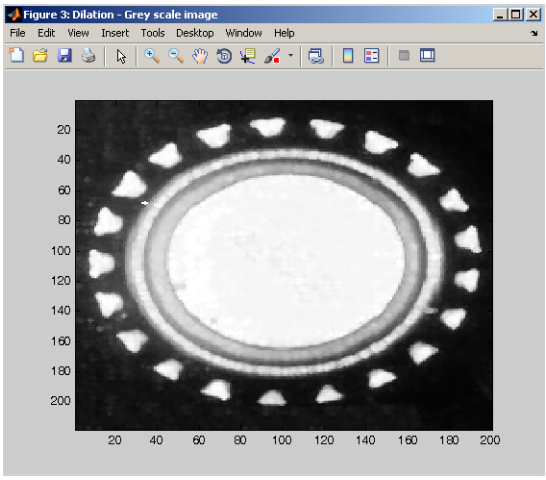
Arguments:
in_img1 mask image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.
connected – an integer specifying whether 4 or 8 connected dilation should be carried out.

Description:
Dilation is also referred to as filling and growing and is the expansion of an image set *A* by a structuring elemt *B*. It is formally viewed as the combination of the two sets using vector addition of the set elements and is equivalent to Minkowski set addition.
out_img1 – An image which has dimensions the same as the input image. This is an image which has been dilated.

Example:

```
img = openimage('crown.jpg');
[out_img2]=vsg('Dilation',out_img1,4);
[out_img3]=vsg('Dilation',img,4);
h=figure;image(uint8(out_img2));set(h,'Name','Dilation - binary
image');
h=figure;image(uint8(out_img3));set(h,'Name','Dilation - Grey scale
image');
```





Notes:
1. For DICOM images, the function operates on all slices separately.
2. For RGB images, the function operates on all colour planes separately.
3. If any number other than 4 or 8 is provided to the function, it defaults to 4.

DilationNxN

Dilation operation with a user defined NxN structuring element.

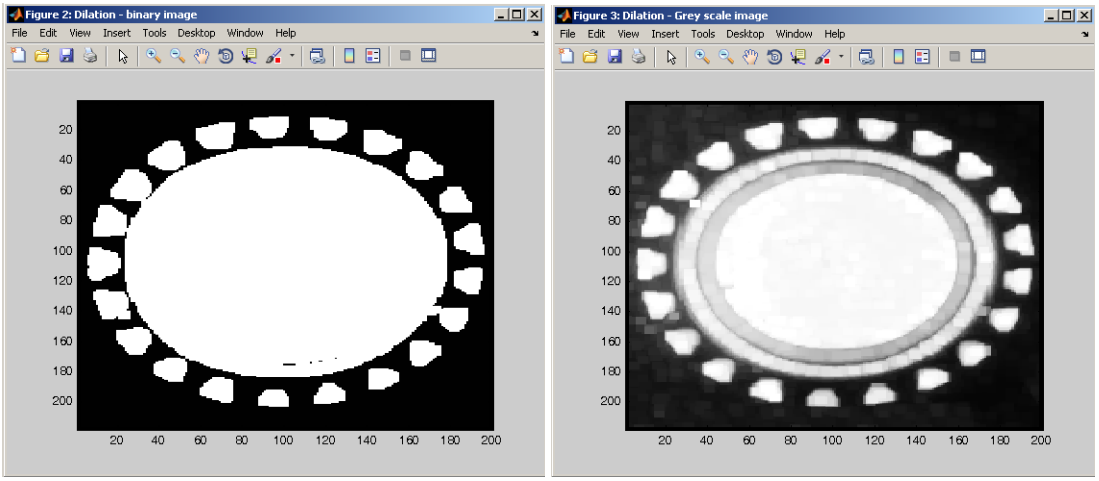
function call:
[out_img1]=vsg('DilationNxN',in_img1,conv_arr);

Arguments:
in_img1 mask image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.
conv_arr – an integer array specifying the mask with which the input image will be dilated.

Description:
Dilation is also referred to as filling and growing and is the expansion of an image set *A* by a SE *B*. It is formally viewed as the combination of the two sets using vector addition of the set elements and is equivalent to Minkowski set addition.
out_img1 – An image which has dimensions the same as the input image. This is an image which has been dilated.

Example:

```
img = openimage('crown.jpg');
[out_img1]=vsg('Threshold',img,100);
array = ones(5,5);
[out_img2]=vsg('DilationNxN',out_img1,array);
[out_img3]=vsg('DilationNxN',img,array);
h=figure;image(uint8(out_img2));set(h,'Name','Dilation – binary
image');
h=figure;image(uint8(out_img3));set(h,'Name','Dilation - Grey scale
image');
```



Notes:
1. For DICOM images, the function operates on all slices separately.
2. For RGB images, the function operates on all colour planes separately.

Erosion

Erosion operation (user specifies the connectivity of the structuring element - 4 or 8).

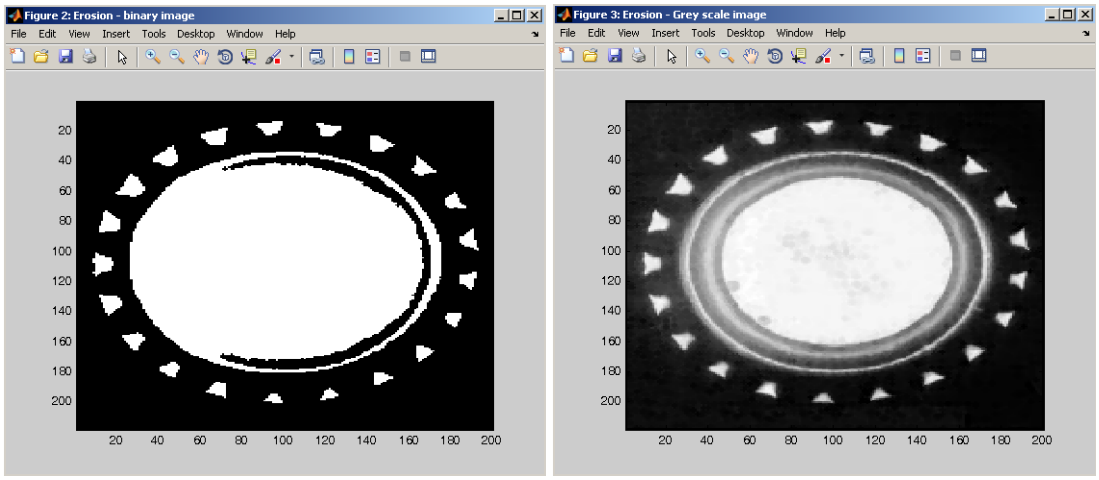
function call:
[out_img1]=vsg('Erosion',in_img1,connected);

Arguments:
in_img1 mask image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.
connected – an integer specifying whether 4 or 8 connected erosion should be carried out.

Description:
Erosion is equivalent to the shrinking of the image set *A* (*in_img1*) by a structuring element *B*. This is a morphological transformation which combines two sets using vector subtraction of set elements. *out_img1* – An image which has dimensions the same as the input image. This is an image which has been eroded.

Example:

```
img = openimage('crown.jpg');
[out_img1]=vsg('Threshold',img,100);
[out_img2]=vsg('Erosion',out_img1,4);
[out_img3]=vsg('Erosion',img,4);
h=figure;image(uint8(out_img2));set(h,'Name','Erosion - binary image');
h=figure;image(uint8(out_img3));set(h,'Name','Erosion - Grey scale
image');
```



Notes:
1. For DICOM images, the function operates on all slices separately.
2. For RGB images, the function operates on all colour planes separately.
3. If any number other than 4 or 8 is provided to the function, it defaults to 4.

ErosionNxN

Erosion operation with a user defined NxN structuring element.

function call:
[out_img1]=vsg('ErosionNxN',in_img1,conv_arr);

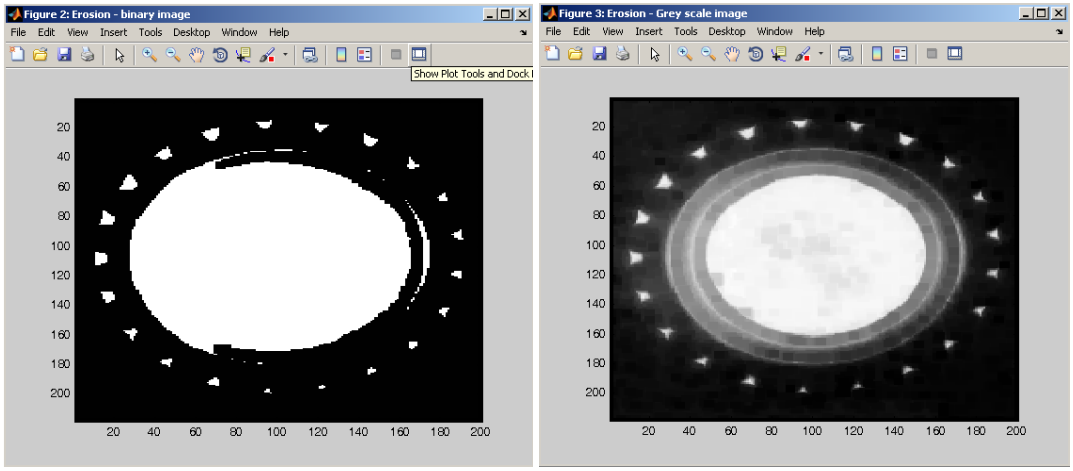
Arguments:
in_img1 mask image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.

conv_arr – an integer array specifying the mask with which the input image will be eroded.

Description:
Erosion is equivalent to the shrinking of the image set *A* (*in_img1*) by a SE *B*. This is a morphological transformation which combines two sets using vector subtraction of set elements.
out_img1 – An image which has dimensions the same as the input image. This is an image which has been eroded.

Example:

```
img = openimage('crown.jpg');
[out_img1]= vsg('Threshold',img,100);
array = ones(5,5);
[out_img2]=vsg('ErosionNxN',out_img1,array);
[out_img3]=vsg('ErosionNxN',img,array);
h=figure;image(uint8(out_img2));set(h,'Name','Erosion - binary image');
h=figure;image(uint8(out_img3));set(h,'Name','Erosion - Grey scale
image');
```



Notes:
1. For DICOM images, the function operates on all slices separately.
2. For RGB images, the function operates on all colour planes separately.

BeucherGrad

Morphological Gradient (user specifies the connectivity of the structuring element - 4 or 8) [Default=8].

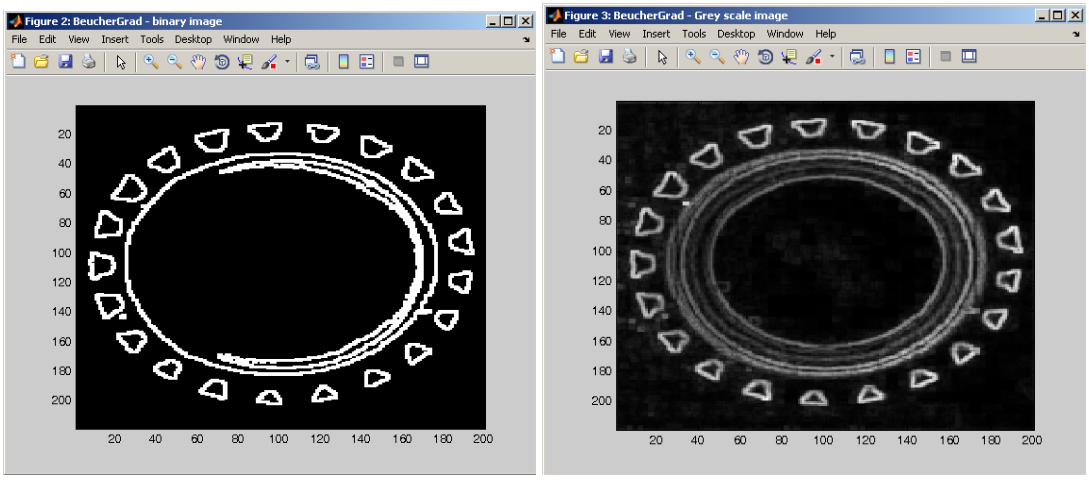
function call:
[out_img1]=vsg('BeucherGrad',in_img1,connected);

Arguments:
in_img1 mask image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.
connected – an integer specifying whether 4 or 8 connected erosion should be carried out.

Description:
out_img1 – An image which has dimensions the same as the input image. This is the morphological gradient of the input image.

Example:

```
img = openimage('crown.jpg');
[out_img1]=vsg('Threshold',img,100);
[out_img2]=vsg('BeucherGrad',out_img1,8);
[out_img3]=vsg('BeucherGrad',img,8);
h=figure;image(uint8(out_img2));set(h,'Name','BeucherGrad - binary
image');
h=figure;image(uint8(out_img3));set(h,'Name','BeucherGrad - Grey scale
image');
```



Notes:

1. For DICOM images, the function operates on all slices separately.
2. For RGB images, the function operates on all colour planes separately.
3. If any number other than 4 or 8 is provided to the function, it defaults to 4.

Close

Closing operation (user specifies the connectivity of the structuring element - 4 or 8).

function call:

[out_img1]=vsg('Close',in_img1,connected);

Arguments:

in_img1 mask image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.
connected – an integer specifying whether 4 or 8 connected erosion should be carried out.

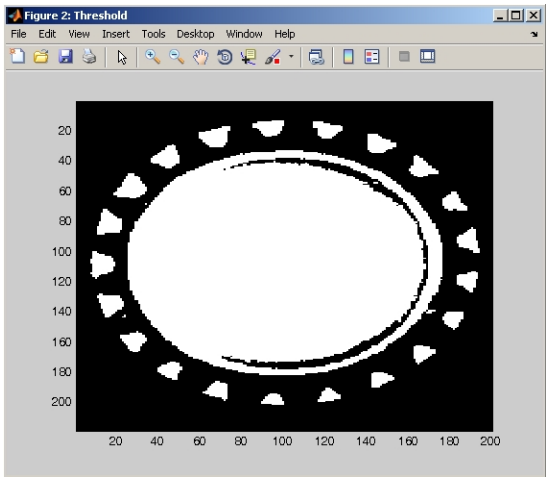
Description:

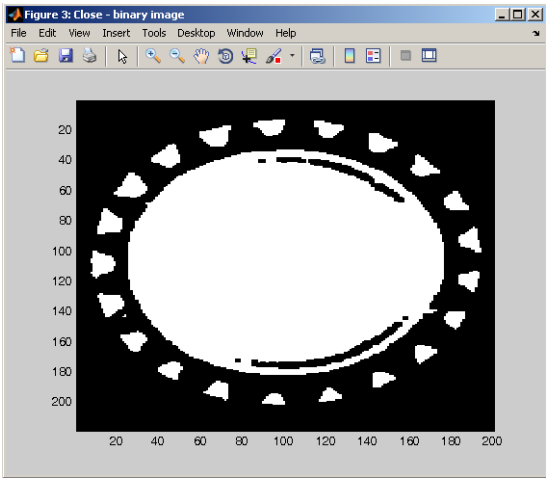
Closing is the dual morphological operation of opening. This transformation has the effect of filing in holes and blocking narrow valleys in the image set *A*, when a structuring element *B* is applied.
out_img1 – An image which has dimensions the same as the input image. This is the result of successive morphological dilation and erosion of the input image.

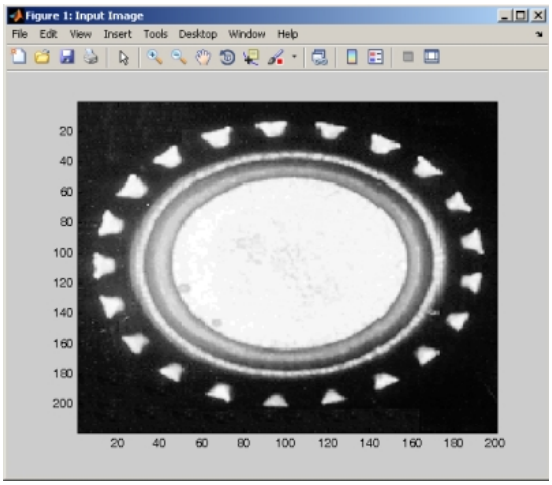
Example:

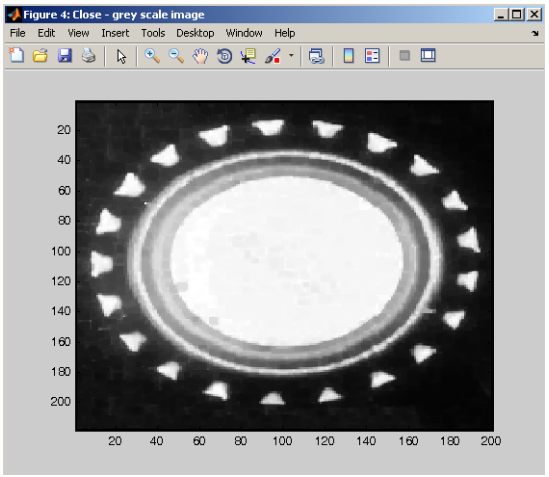
```
img = openimage('crown.jpg');
[h=figure;image(uint8(img));set(h,'Name','Input image');
[out_img1]=vsg('Threshold',img,100);
h=figure;image(uint8(out_img1));set(h,'Name','Threshold');
[out_img2]=vsg('Close',out_img1,8);
```

```
[out_img3]=vsg('Close',img,8);
h=figure;image(uint8(out_img2));set(h,'Name','Close - binary image');
h=figure;image(uint8(out_img3));set(h,'Name','Close - grey scale
image');
```









Notes:
1. For DICOM images, the function operates on all slices separately.
2. For RGB images, the function operates on all colour planes separately.
3. If any number other than 4 or 8 is provided to the function, it defaults to 4.

CloseNxN

Erosion operation with a user defined N×N structuring element.

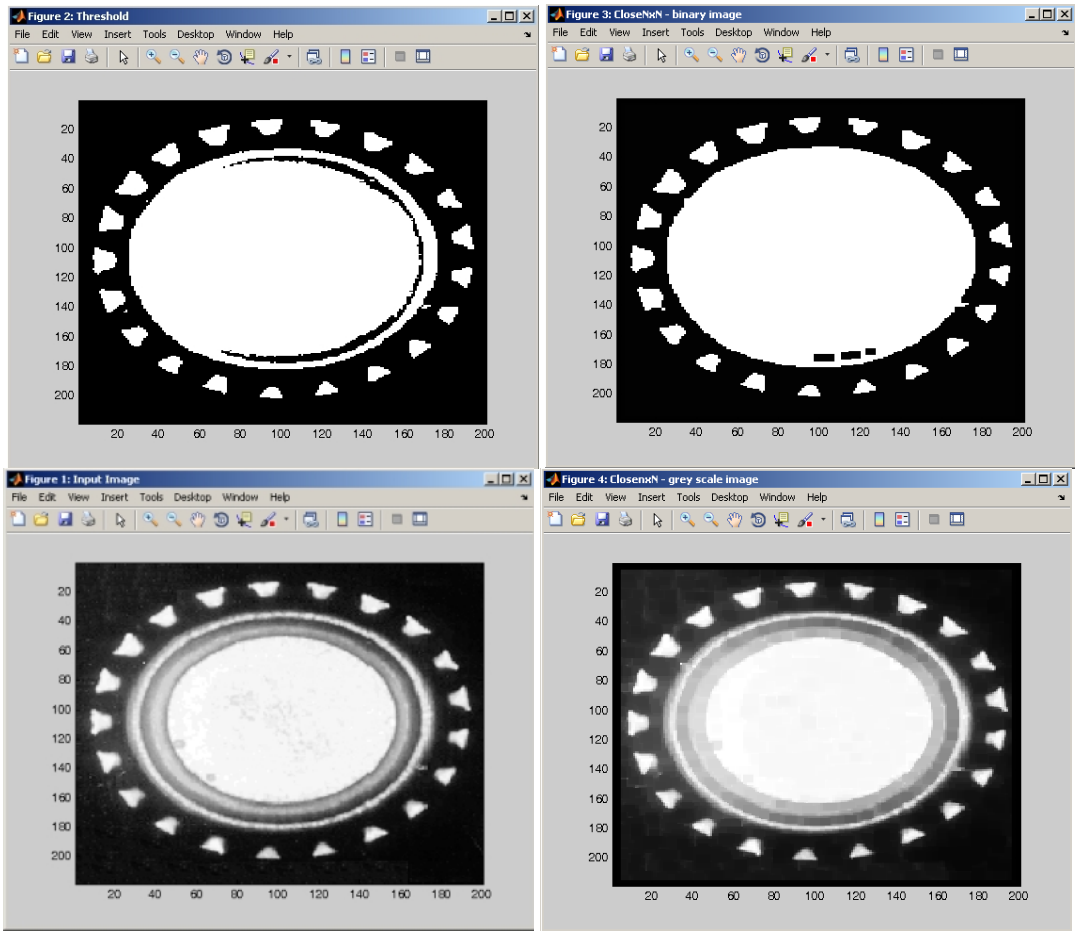
function call:
[out_img1]=vsg('CloseNxN',in_img1, conv_arr);

Arguments:
in_img1 mask image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.
conv_arr – an integer array specifying the mask with which the input image will be processed.

Description:
Closing is the dual morphological operation of opening. This transformation has the effect of filing in holes and blocking narrow valleys in the image set *A*, when a structuring element *B* is applied.
out_img1 – An image which has dimensions the same as the input image. This is the result of successive morphological dilation and erosion of the input image.

Example:

```
img = openimage('crown.jpg');
h=figure;image(uint8(img));set(h,'Name','Input image');
[out_img1]=vsg('Threshold',img,100);
h=figure;image(uint8(out_img1));set(h,'Name','Threshold');
array = ones(5,5);
[out_img2]=vsg('CloseNxN',out_img1,array);
[out_img3]=vsg('CloseNxN',img,array);
h=figure;image(uint8(out_img2));set(h,'Name','CloseNxN - binary
image');
h=figure;image(uint8(out_img3));set(h,'Name','ClosenxN - grey scale
image');
```



Notes:
1. For DICOM images, the function operates on all slices separately.
2. For RGB images, the function operates on all colour planes separately.

Open

Opening operation (user specify connectivity of the structured element 4 or 8).

function call:

[out_img1]=vsg('Open',in_img1,connected);

Arguments:

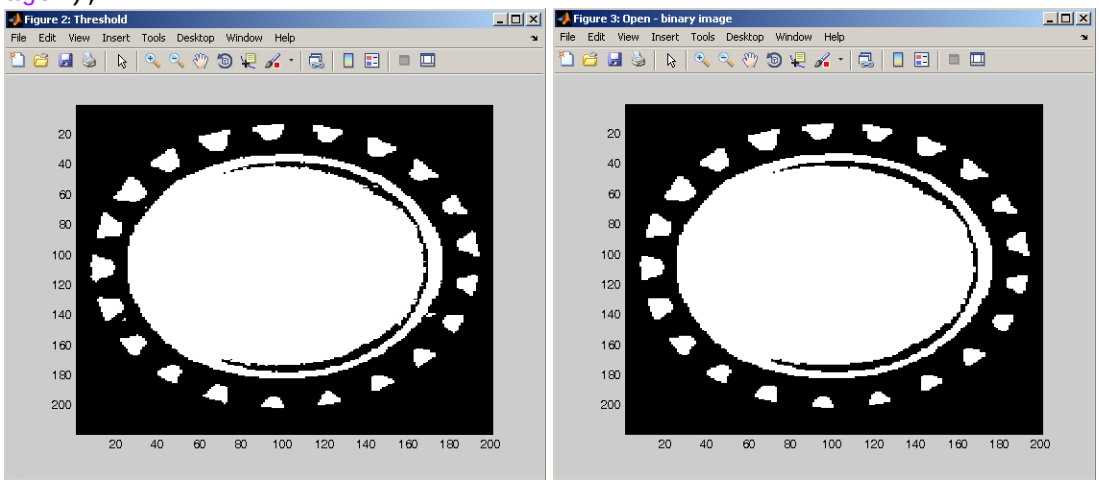
in_img1 mask image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.
connected – an integer specifying whether 4 or 8 connected morphological operations should be carried out.

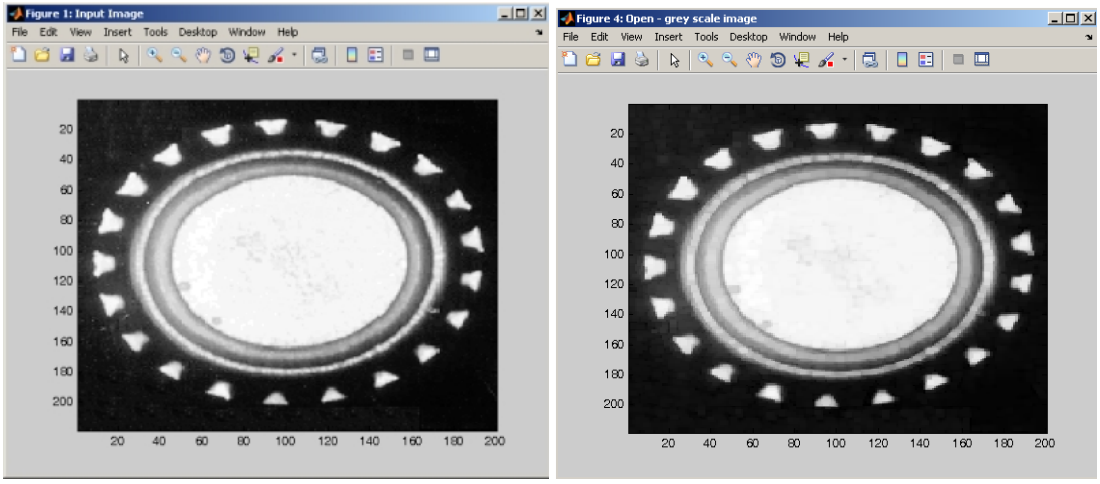
Description:

Opening is a combination of erosion and dilation operations that have the effect of removing isolated spots in the image set *A (in_img1)* that are smaller than the structuring element *B* and those sections of the image set *A* narrower than *B*.
out_img1 – An image which has dimensions the same as the input image. This is the result of successive morphological erosion and dilation of the input image.

Example:

```
img = openimage('crown.jpg');
h=figure;image(uint8(img));set(h,'Name','Input image');
[out_img1]=vsg('Threshold',img,100);
h=figure;image(uint8(out_img1));set(h,'Name','Threshold');
[out_img2]=vsg('Open',out_img1,8);
[out_img3]=vsg('Open',img,8);
h=figure;image(uint8(out_img2));set(h,'Name','Open - binary image');
h=figure;image(uint8(out_img3));set(h,'Name','Open - grey scale
image');
```





Notes:
1. For DICOM images, the function operates on all slices separately.
2. For RGB images, the function operates on all colour planes separately.
3. If any number other than 4 or 8 is provided to the function, it defaults to 4.

OpenNxN

Dilation operation with a user defined NxN structuring element.

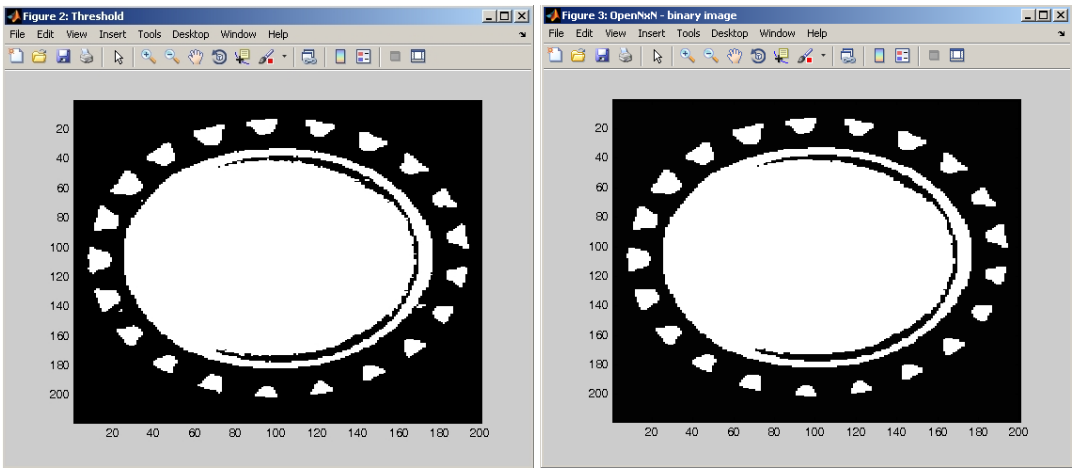
function call:
[out_img1]=vsg('OpenNxN',in_img1, conv_arr);

Arguments:
in_img1 mask image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.
conv_arr – an integer array specifying the mask with which the input image will be processed.

Description:
Opening is a combination of erosion and dilation operations that have the effect of removing isolated spots in the image set *A (in_img1)* that are smaller than the structuring element *B* and those sections of the image set *A* narrower than *B*.
out_img1 – An image which has dimensions the same as the input image. This is the result of successive morphological erosion and dilation of the input image.

Example:

```
img = openimage('crown.jpg'); %color image
[out_img1]=vsg('Threshold',img,100);
figure;image(uint8(out_img1)); set(h,'Name','Threshold');
array = ones(3,3);
[out_img2]=vsg('OpenNxN',out_img1,array);
h=figure;image(uint8(out_img2));set(h,'Name','OpenNxN - binary image');
```



Notes:
1. For DICOM images, the function operates on all slices separately.
2. For RGB images, the function operates on all colour planes separately.

Hit-And-Miss

Hit and miss transformation. Hit and miss array masks must not overlap.

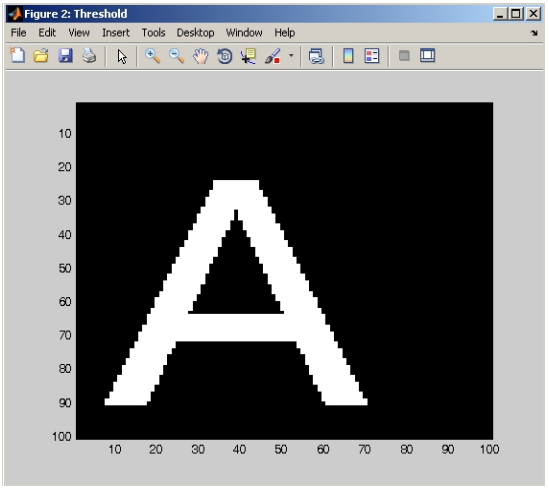
function call:
[out_img1]=vsg('HitAndMiss',in_img1,conv_arr1,conv_arr2);

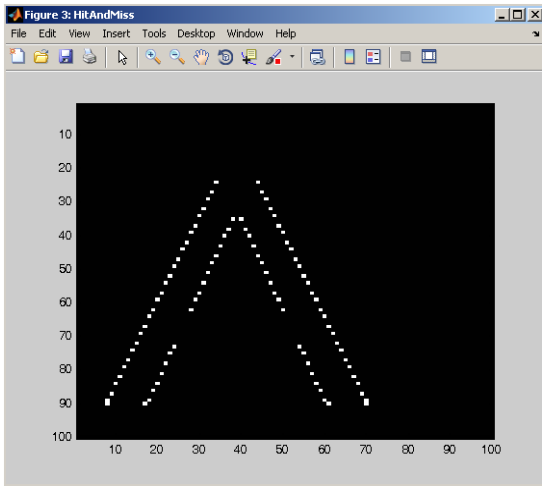
Arguments:
in_img1 mask image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.
conv_arr1 – the "hit" array specifying the mask with which the input image will be processed.
conv_arr2 – the "miss" array specifying the mask with which the input image will be processed.
Description:
The HitAndMiss morphological transform is used to select pixels with certain geometric properties (e.g. corners) while ignoring others.
out_img1 – An image which has dimensions the same as the input image. This is the result of finding the "hit" and "miss values.

Example:

```
img = imread('A.jpg');
[out_img1]=vsg('Threshold',img,100);
h=figure;image(uint8(out_img1));set(h,'Name','Threshold');
H1 = [0,1,0;
      0,1,1;
      0,0,0];
H2 = [0,0,0;
      1,1,0;
      0,1,0];
H3 = [0,1,0;
      1,1,0;
      0,0,0];
H4 = [0,0,0;
      0,1,1;
      0,1,0];
M1 =[0,0,0;
     1,0,0;
     1,1,0];
M2 =[0,1,1;
```

```
     0,0,1;
     0,0,0];
M3 =[0,0,0;
     0,0,1;
     0,1,1];
M4 =[1,1,0;
     1,0,0;
     0,0,0];
[out_img2]=vsg('HitAndMiss',out_img1,H1,M1);
[out_img3]=vsg('HitAndMiss',out_img1,H2,M2);
[out_img4]=vsg('HitAndMiss',out_img1,H3,M3);
[out_img5]=vsg('HitAndMiss',out_img1,H4,M4);
[out_img6]=vsg('OR',out_img2,out_img3);
[out_img6]=vsg('OR',out_img6,out_img4);
[out_img6]=vsg('OR',out_img6,out_img5);
h=figure;image(uint8(out_img6));set(h,'Name','HitAndMiss');
```





Notes:
1. For DICOM images, the function operates on all slices separately.
2. For RGB images, the function operates on all colour planes separately.

TopHat

Morphological top hat operation (user specifies the connectivity of the structuring element - 4 or 8) [Default=8].

function call:
[out_img1]=vsg('TopHat',in_img1,connected);

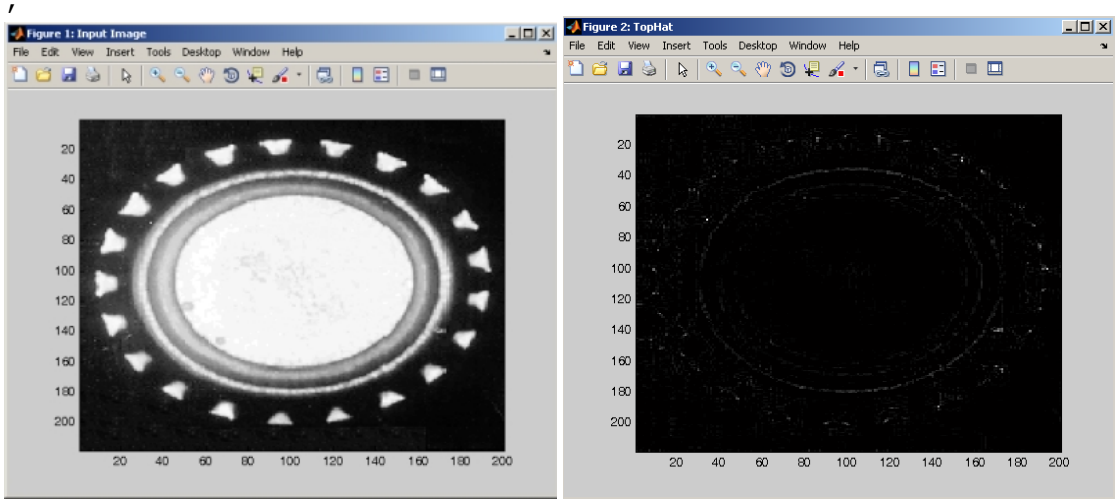
Arguments:
in_img1 mask image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.
connected – an integer specifying whether 4 or 8 connected morphological operations.

Description:
The morphological TopHat uses knowledge about the shape characteristics that are not shared by the relevant image structures. An opening with a structuring element (SE) that does not fit the relevant image structures is then used to remove them from the image. These structures are recovered through the arithmetic difference between the image and its opening.

out_img1 – An image which has dimensions the same as the input image. This is the difference between the original image and the same image after being morphologically opened.

Example:

```
img = openimage('crown.jpg');
h=figure;image(img);set(h,'Name','Input Image');
[out_img1]=vsg('TopHat',img,4);
h=figure;imagesc(out_img1(:,:,1));colormap(gray);set(h,'Name','TopHat')
;
```



Notes:

1. For DICOM images, the function operates on all slices separately.
2. For RGB images, the function operates on all colour planes separately.
3. If any number other than 4 or 8 is provided to the function, it defaults to 4.

Valley

Morphological valley operation (user specifies the connectivity of the structuring element - 4 or 8) [Default=8].

function call:

[out_img1]=vsg('Valley',in_img1,connected);

Arguments:

in_img1 mask image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.
connected – an integer specifying whether 4 or 8 connected morphological operations.

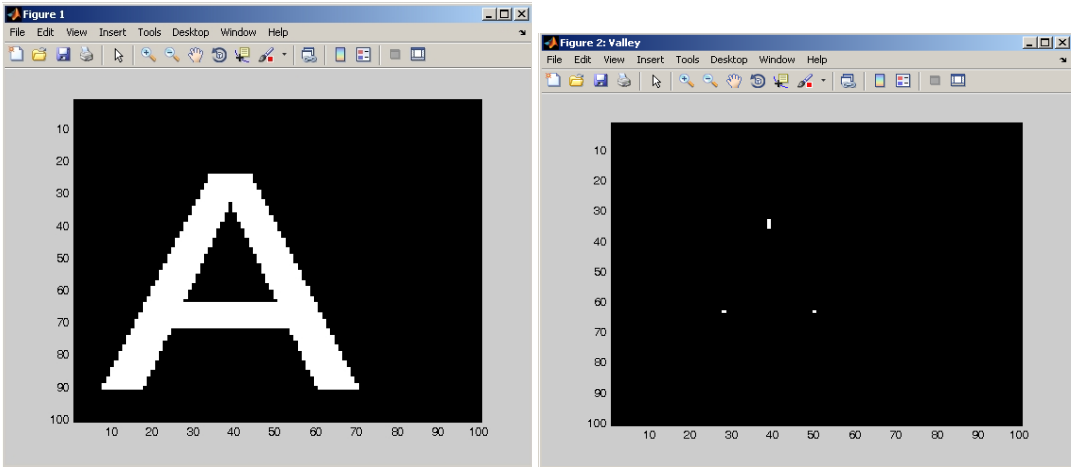
Description:

The valley detector is the dual of the white Top-hat by opening. The valley detector is defined as *BTH(A) = (A●K) – A*; where is A the input image, K is the structuring element and (●) is the closing operation. *out_img1* – An image which has dimensions the same as the input image. This is the difference between the original image and the same image after being morphologically closed.

Example:

```
img = openimage('A.jpg');
h=figure;image(uint8(img));set(h,'Name','Input Image');
[img]=vsg('Threshold',img,100);
[out_img1]=vsg('Valley',img,8);
```

```
h=figure;image(uint8(out_img1));set(h,'Name','Valley');
```



Notes:
1. For DICOM images, the function operates on all slices separately.
2. For RGB images, the function operates on all colour planes separately.
3. If any number other than 4 or 8 is provided to the function, it defaults to 4.

ReconByDil

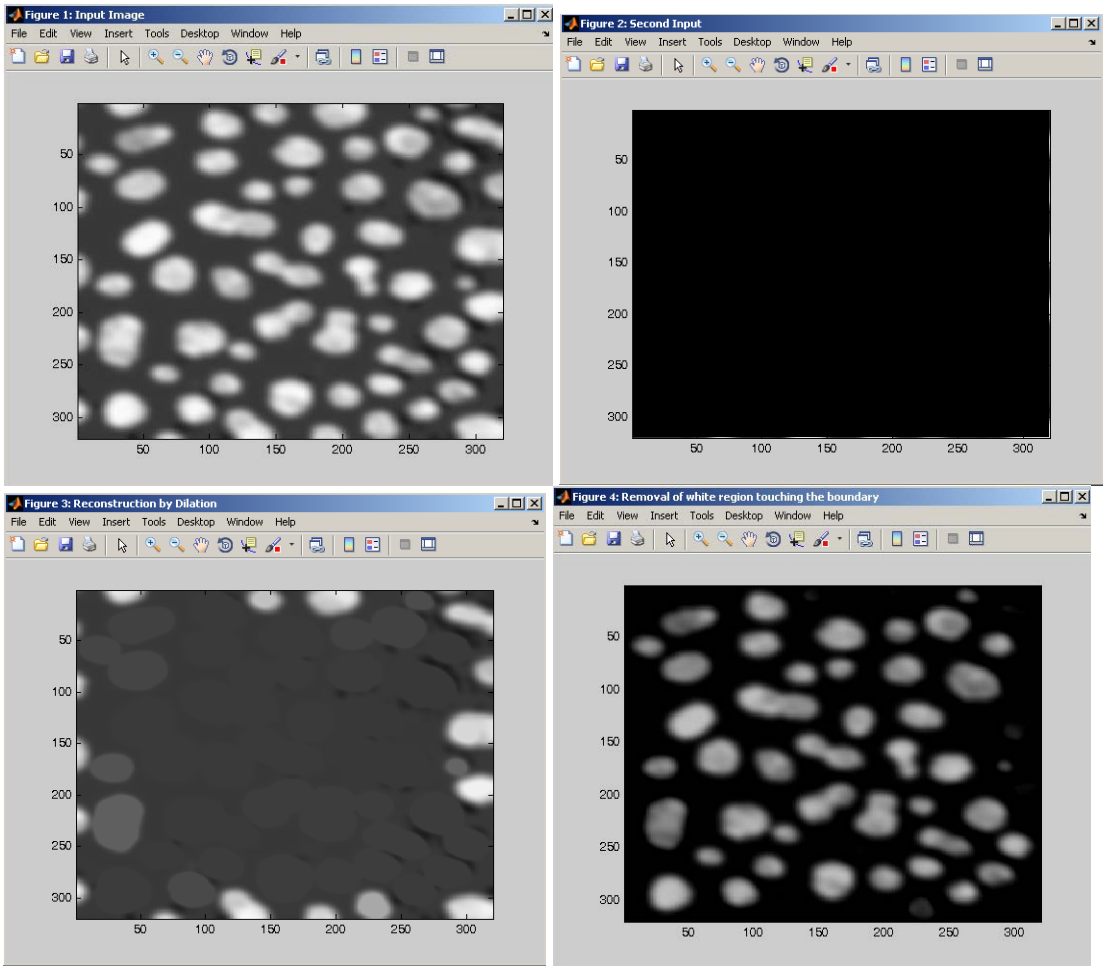
Reconstruction by dilation.

function call:
[out_img1,out_img2]=vsg('ReconByDil',in_img1,in_img2,connected);

Arguments:
in_img1 mask image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.
in_img2 seed image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.
connected – an integer specifying whether 4 or 8 connected morphological operations should be carried out.

Description:
out_img1 – An image which has dimensions the same as the input image. This is the reconstructed image.
out_img2 – An image which has dimensions the same as the input image. This is the remainder of the input mask image less the reconstructed regions.

Example:
```
img1=openimage('bio_sample.gif');
img1 = vsg('Inverse',img1);
h=figure;image(uint8(img1));set(h,'Name','Input Image');
img2 = img1;
img2(2:size(img1,1)-1,2:size(img1,2)-1,:)=0;
h=figure;image(uint8(img2));set(h,'Name','Second Input');
[out_img1,out_img2]=vsg('ReconByDil',img1,img2,connected);
h=figure;image(uint8(out_img1));set(h,'Name','Reconstruction by
Dilation');
h=figure;image(uint8(out_img2));set(h,'Name','Removal of white region
touching the boundary');
```



Notes:
1. For DICOM images, the function operates on all slices separately.
2. For RGB images, the function operates on all colour planes separately.
3. All input images should be binary with a range of {0,255} as it is white regions that are being dilated and it is a white region that should be used as the mask.
3. If any number other than 4 or 8 is provided to the function, it defaults to 4.

Watershed

Watershed transform (return the watershed image and the region boundaries image).

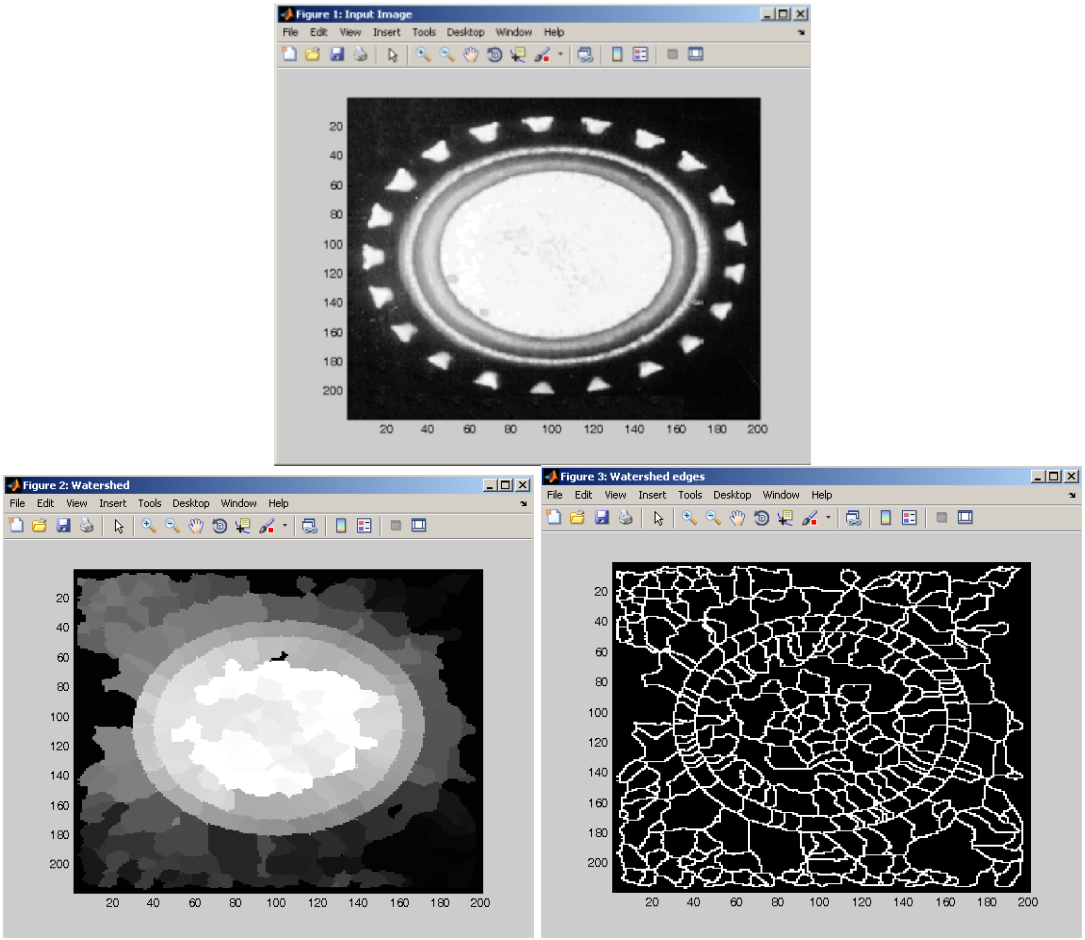
function call:
[out_img1,out_img2]=vsg('Watershed',in_img1);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.

Description:
out_img1 – An image which has dimensions the same as the input image. This is the labelled regions image.
out_img2 – An image which has dimensions the same as the input image. This is the edges of the labelled regions, the watershed lines.

Example:

```
img = openimage('crown.jpg');
h=figure;image(uint8(img));set(h,'Name','Input Image');
[out_img1,out_img2]=vsg('Watershed',img);
h=figure;image(uint8(out_img1));set(h,'Name','Watershed');
```



Notes:

1. For DICOM images, the function operates on all slices separately.
2. For RGB images, the function operates on all colour planes separately.

Noise Functions

AdditiveNoise

Add a user defined level of white noise to the input image.

function call:
[out_img1]=vsg('AdditiveNoise',in_img1,num1);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.
num1 – an integer which specifies the range of the additive noise.

Description:
out_img1 – An image which has dimensions the same as the input image. This is the input image with noise added. Max noise level =num1/2, min noise level =(-num1)/2.

Example:
```
img = openimage('lake.jpg');
[out_img1]=vsg('AdditiveNoise',img,50);
```

Notes:
1. For DICOM images, the function operates on all slices separately.
2. For RGB images, the function operates on all colour planes separately.

GaussianNoise

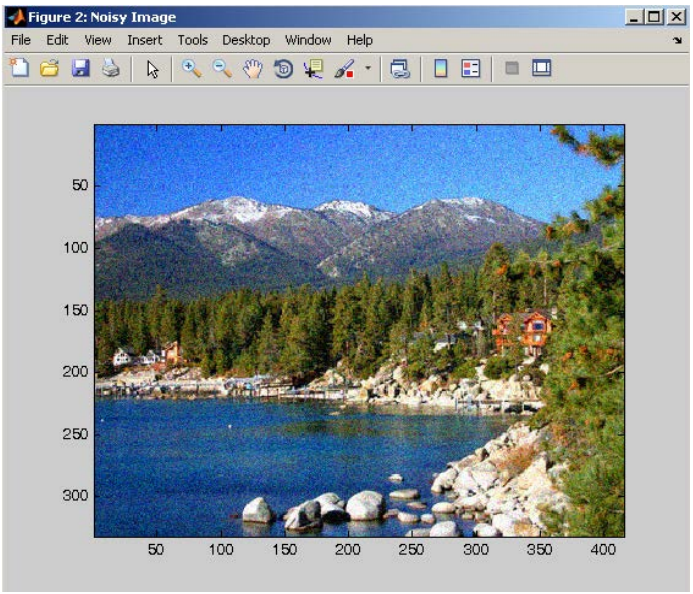
Add a user defined quantity of Gaussian noise to the input image.

function call:
[out_img1]=vsg('GaussianNoise',in_img1,num1);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.
num1 – a double which specifies the standard deviation of the Gaussian noise.

Description:
out_img1 – An image which has dimensions the same as the input image. This is the input image with Gaussian noise added. The Gaussian noise will have a mean of 0 and std of num1.

Example:
```
img = openimage('lake.jpg');
h=figure; image(img);set(h,'Name','Input Image');
[out_img1]=vsg('GaussianNoise',img,10);
h=figure; image(uint8(out_img1));set(h,'Name','Noisy image');
```



Notes:
1. For DICOM images, the function operates on all slices separately.
2. For RGB images, the function operates on all colour planes separately.

PoissonNoise

Add a user defined quantity of Poisson noise to the input image.

function call:
[out_img1]=vsg('PoissonNoise',in_img1,num1);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.
num1 – a double which specifies the standard deviation of the Poisson noise.

Description:
out_img1 – An image which has dimensions the same as the input image. This is the input image with Poisson noise added. The Poisson noise will have a std of num1.

Example:
```
img = openimage('lake.jpg');
h=figure; image(img);set(h,'Name','Input Image');
[out_img1]=vsg('PoissonNoise',img,0.5);%crashed with value of 0 to <1.0
h=figure; image(uint8(out_img1));set(h,'Name','Noisy image');
```

Notes:
1. For DICOM images, the function operates on all slices separately.
2. For RGB images, the function operates on all colour planes separately.

RayleighNoise

Add a user defined quantity of Rayleigh noise to the input image.

function call:
[out_img1]=vsg('RayleighNoise',in_img1,num1);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.
num1 – a double which specifies the standard deviation of the Rayleigh noise.

Description:
out_img1 – An image which has dimensions the same as the input image. This is the input image with Rayleigh noise added. The Rayleigh noise will have a std of num1.

Example:
```
img = openimage('lake.jpg');
h=figure; image(img);set(h,'Name','Input Image');
[out_img1]=vsg('RayleighNoise',img,10);
h=figure; image(uint8(out_img1));set(h,'Name','Noisy Image');
```

Notes:
1. For DICOM images, the function operates on all slices separately.
2. For RGB images, the function operates on all colour planes separately.

S+PNoise

Add salt and pepper noise to the input image

function call:
[out_img1]=vsg('S+PNoise',in_img1,num1);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.
num1 – a double which specifies the fraction of the image that should be covered with salt and pepper noise, range {0.0-1.0}.

Description:
out_img1 – An image which has dimensions the same as the input image. This is the input image with salt and pepper noise added.

Example:
```
img = openimage('lake.jpg');
h=figure; image(img);set(h,'Name','Input Image');
[out_img1]=vsg('S+PNoise',img,0.01);
h=figure; image(uint8(out_img1));set(h,'Name','Noisy Image');
```

Notes:
1. For DICOM images, the function operates on all slices separately.
2. For RGB images, the function operates on all colour planes separately.

Processing Functions

IntegrateC

Integrate Image columns.

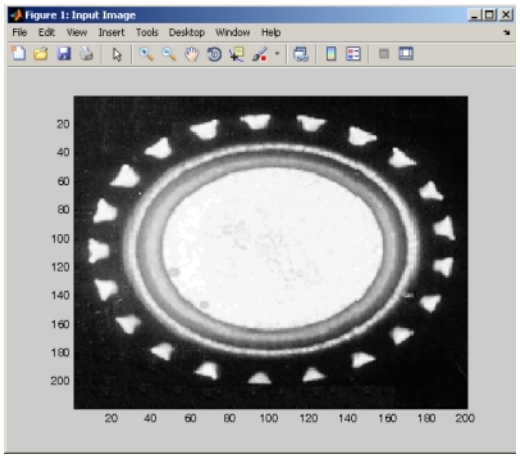
function call:
[out_img1]=vsg('IntegrateC',in_img1);

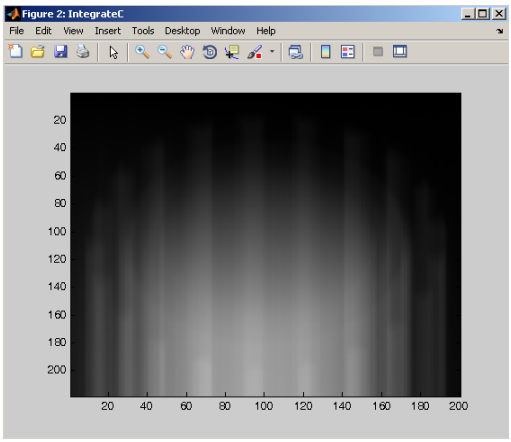
Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.

Return values
out_img1 – An image which has dimensions the same as the input image. This is the running integral on each column, starting from the top, normalised by the height.

Example:

```
img= openimage('crown.jpg');
h=figure;image(img); set(h,'Name','Input Image');
[out_img1]=vsg('IntegrateC',img);
h=figure;image(uint8(out_img1));set(h,'Name','IntegrateC');
```





Notes:
1. For DICOM images, the function operates on all slices separately.
2. For RGB images, the function operates on all colour planes separately.

IntegrateR

Integrate image rows.

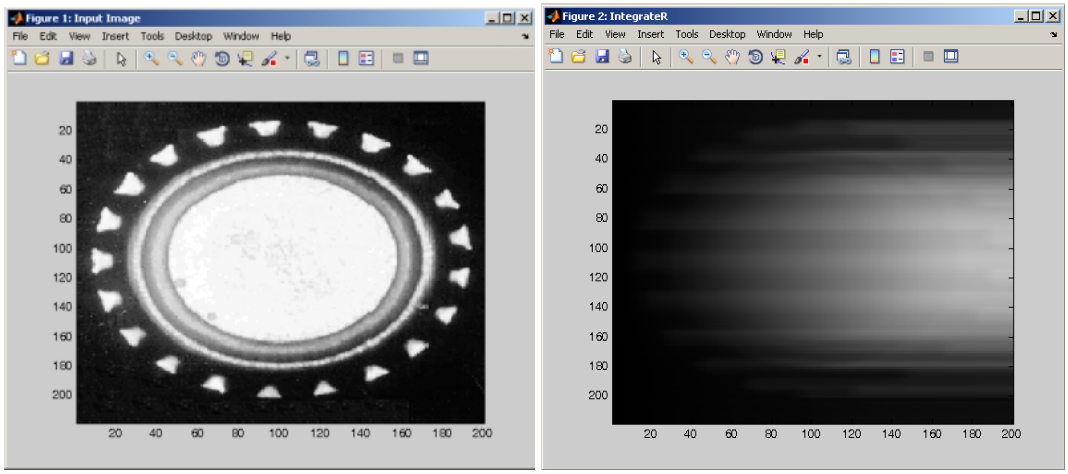
function call:
[out_img1]=vsg('IntegrateR',in_img1);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.

Description:
out_img1 – An image which has dimensions the same as the input image. This is the running integral on each row, starting from the left, normalised by the image width.

Example:

```
img= openimage('crown.jpg');
h=figure;image(img); set(h,'Name','Input Image');
[out_img1]=vsg('IntegrateR',img);
h=figure;image(uint8(out_img1));set(h,'Name','IntegrateR');
```



Notes:

1. For DICOM images, the function operates on all slices separately.
2. For RGB images, the function operates on all colour planes separately.

L2RSum

Pixel summation along the line.

function call:

[out_img1]=vsg('L2RSum',in_img1);

Arguments:

in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.

Description:

out_img1 – An image which has dimensions the same as the input image. This is the running integral on each row, starting from the left, normalised by the image width. Only the right most column of the output image is used to store the sum totals.

Example:

```
img = openimage('crown.jpg');
[out_img1]=vsg('L2RSum',img);
h=figure;imagesc((out_img1(:,:,1)));colormap(gray);
```

Notes:

1. For DICOM images, the function operates on all slices separately.
2. For RGB images, the function operates on all colour planes separately.

RemIso

Remove isolate white pixels (3x3) region).

function call:

[out_img1]=vsg('RemIso',in_img1);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.

Description:
out_img1 – An image which has dimensions the same as the input image. This is the input image with any single white pixels removed.

Example:
```
img = openimage('lake.jpg');
h=figure; image(img);set(h,'Name','Input Image');
[out_img1]=vsg('RemIso',img);
h=figure; image(uint8(out_img1));set(h,'Name','RemIso image');
```

Notes:
1. For DICOM images, the function operates on all slices separately.
2. For RGB images, the function operates on all colour planes separately.
3. Input image should be black and white.

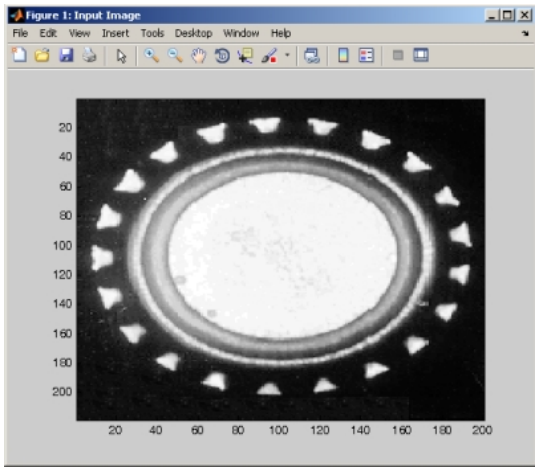
B2TWash
Bottom To Top wash function (once a white pixel is found, all pixels above it in the same column are also set to white).

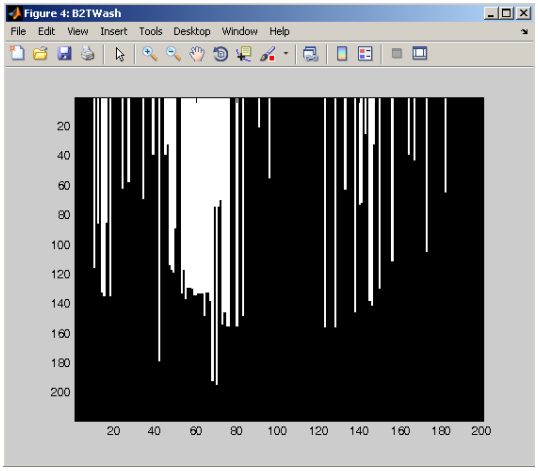
function call:
[out_img1]=vsg('B2TWash',in_img1);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.

Description:
out_img1 – An image which has dimensions the same as the input image. The output image will be set to white from the location of each white pixel in the input image to the top of that column.

Example:
```
img = openimage('crown.jpg');
h=figure;image(uint8(img));set(h,'Name','Input Image');
[out_img1]=vsg('B2TWash',img);
h=figure;image(uint8(out_img1(:,:,1)));colormap(gray);set(h,'Name','B2T
Wash';
```





Notes:
1. For DICOM images, the function operates on all slices separately.
2. For RGB images, the function operates on all colour planes separately.

T2BWash

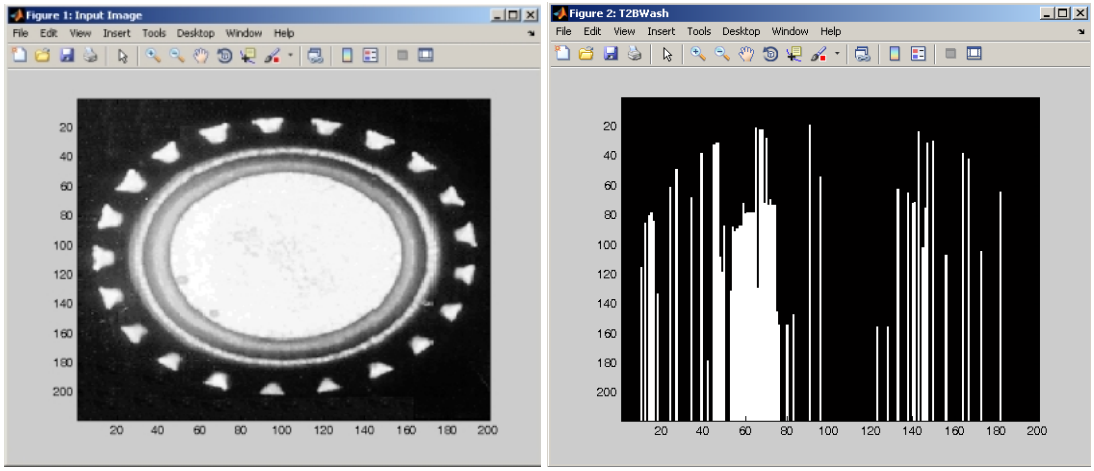
Top To Bottom wash function (once a white pixel is found, all pixels below it in the same column are also set to white).

function call:
[out_img1]=vsg('B2TWash',in_img1);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.

Description:
out_img1 – An image which has dimensions the same as the input image. The output image will be set to white from the location of each white pixel in the input image to the bottom of that column.

Example:
```
img = openimage('crown.jpg');
h=figure;image(uint8(img));set(h,'Name','Input Image');
[out_img1]=vsg('T2BWash',img);
h=figure;imagesc((out_img1(:,:,1)));colormap(gray);set(h,'Name','T2BWas
h');
```



Notes:
1. For DICOM images, the function operates on all slices separately.
2. For RGB images, the function operates on all colour planes separately.

L2RWash

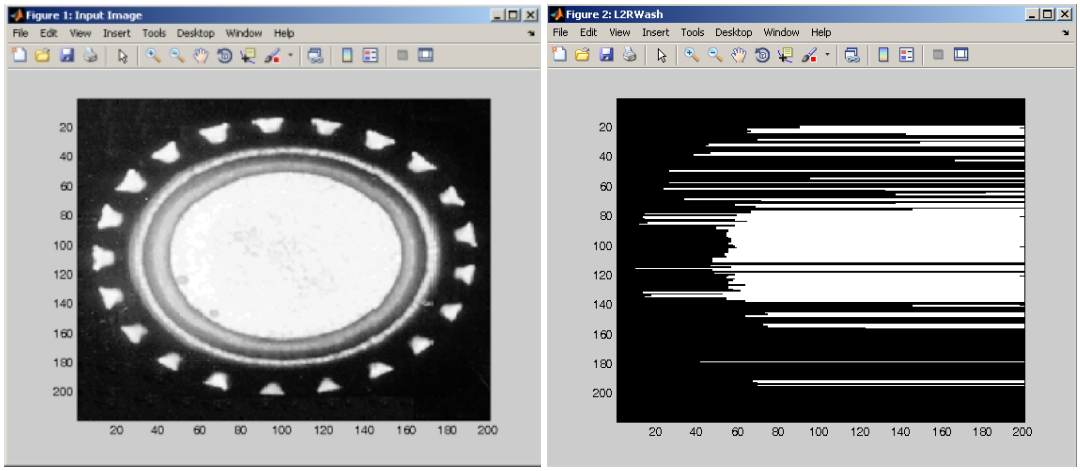
Left To Right wash function (once a white pixel is found, all pixels to its right are also set to white).

function call:
[out_img1]=vsg('B2TWash',in_img1);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.

Description:
out_img1 – An image which has dimensions the same as the input image. The output image will be set to white from the location of each white pixel in the input image and right of that pixel.

Example:
```
img = openimage('crown.jpg');
h=figure;image(uint8(img));set(h,'Name','Input Image');
[out_img1]=vsg('L2RWash',img);
h=figure;imagesc((out_img1(:,:,1)));colormap(gray);set(h,'Name','L2RWas
h');
```



Notes:
1. For DICOM images, the function operates on all slices separately.
2. For RGB images, the function operates on all colour planes separately.

R2LWash
Right To Left wash function (once a white pixel is found, all pixels to its left are also set to white).

function call:
[out_img1]=vsg('B2TWash',in_img1);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.

Description:
out_img1 – An image which has dimensions the same as the input image. The output image will be set to white from the location of each white pixel in the input image and left of that pixel.

Example:
```
img = openimage('crown.jpg');
h=figure;image(uint8(img));set(h,'Name','Input Image');
[out_img1]=vsg('R2LWash',img);
h=figure;imagesc((out_img1(:,:,1)));colormap(gray);set(h,'Name','R2LWas
h');
```

Notes:
1. For DICOM images, the function operates on all slices separately.
2. For RGB images, the function operates on all colour planes separately.

HistEqualiser

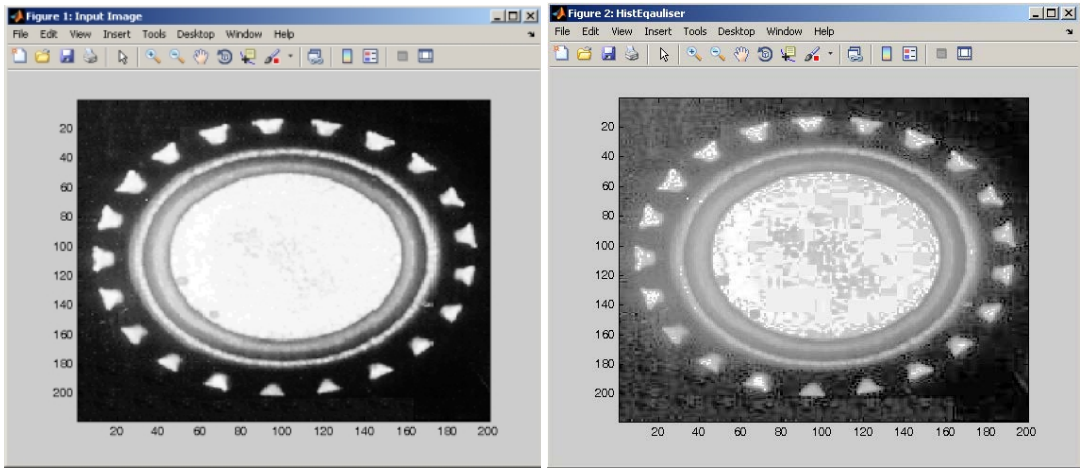
Histogram equalisation.

function call:
[out_img1]=vsg('HistEqualiser',in_img1);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.

Description:
out_img1 – An image which has dimensions the same as the input image. The output image has had its greyscale histogram flattened.

Example:
```
img = openimage('crown.jpg');
h=figure;image(uint8(img));set(h,'Name','Input Image');
[out_img1]=vsg('HistEqualiser',img);
h=figure;image(uint8(out_img1));set(h,'Name','HistEqauliser');
```



Notes:
1. For DICOM images, the function operates on all slices separately.
2. For RGB images, the function operates on all colour planes separately.

Enhancer

Stretch the LUT in order to occupy the entire range between BLACK (0) and WHITE (255).

function call:
[out_img1]=vsg('Enhancer',in_img1);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.

Description:
out_img1 – An image which has dimensions the same as the input image. The output image has had its greyscale histogram stretched from 0 to 255.

Example:
```
img = openimage('crown.jpg');
h=figure;image(uint8(img));set(h,'Name','Input Image');
[out_img1]=vsg('Enhancer',img);
h=figure;image(uint8(out_img1));set(h,'Name','Enhancer');
```

Notes:
1. For DICOM images, the function operates on all slices separately.
2. For RGB images, the function operates on all colour planes separately.

Stretcher

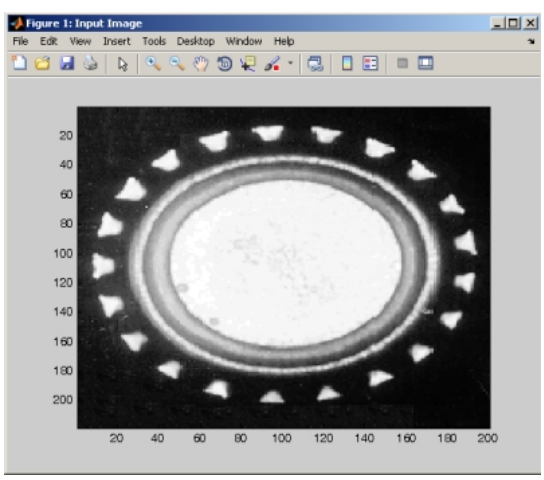
Stretch the LUT between the lower and upper threshold to occupy the entire range between BLACK (0) and WHITE (255).

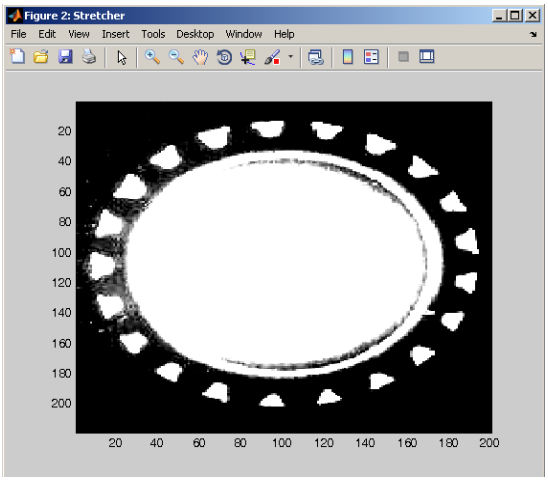
function call:
[out_img1]=vsg('Stretcher',in_img1,num1,num2);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.
num1 – an integer specifying the low greyscale value, below which input pixels will be set to black.
num2 – an integer specifying the high greyscale value, above which input pixels will be set to white.

Description:
out_img1 – An image which has dimensions the same as the input image. The output image is the input image which has had its greyscale histogram from num1 to num2 stretched to fill the full range.

Example:
```
img = openimage('crown.jpg');
h=figure;image(uint8(img));set(h,'Name','Input Image');
[out_img1]=vsg('Stretcher',img,50,100);
h=figure;image(uint8(out_img1));set(h,'Name','Stretcher');
```





Notes:
1. For DICOM images, the function operates on all slices separately.
2. For RGB images, the function operates on all colour planes separately.

Look-Up-Table Functions

Exponential

Transform the linear LUT into exponential.

function call:
[out_img1]=vsg('Exponential',in_img1);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.

Description:
out_img1 – An image which has dimensions the same as the input image. The output pixel values are the normalised exponential of the input pixel values.

Example:
```
img = openimage('crown.jpg');
h=figure;image(uint8(img));set(h,'Name','Input Image');
[out_img1]=vsg('Exponential',img);
h=figure;image(uint8(out_img1));set(h,'Name','Exponential');
```

Notes:
1. For DICOM images, the function operates on all slices separately.
2. For RGB images, the function operates on all colour planes separately.

Inverse

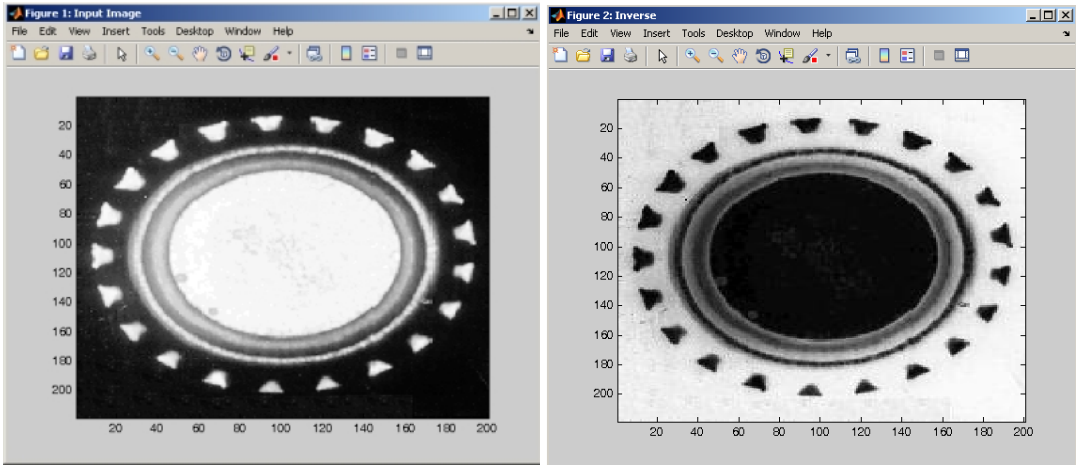
Inverse the LUT of the input image.

function call:
[out_img1]=vsg('Inverse',in_img1);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.

Description:
out_img1 – An image which has dimensions the same as the input image. The output pixel values are 255 minus the input pixel values.

Example:
```
img = openimage('crown.jpg');
h=figure;image(uint8(img));set(h,'Name','Input Image');
[out_img1]=vsg('Inverse',img);
h=figure;image(uint8(out_img1));set(h,'Name','Inverse');
```



Notes:
1. For DICOM images, the function operates on all slices separately.
2. For RGB images, the function operates on all colour planes separately.

Logarithm

Transform the linear LUT into logarithmic.

function call:
[out_img1]=vsg('Logarithm',in_img1);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.

Description:
out_img1 – An image which has dimensions the same as the input image. The output pixel values are the normalised logarithm of the input pixel values.

Example:

```
img = openimage('crown.jpg');
h=figure;image(uint8(img));set(h,'Name','Input Image');
[out_img1]=vsg('Logarithm',img);
h=figure;image(uint8(out_img1));set(h,'Name','Logarithm');
```

Notes:
1. For DICOM images, the function operates on all slices separately.
2. For RGB images, the function operates on all colour planes separately.

Power

The linear LUT is raised to a user specified double value.

function call:
[out_img1]=vsg('Power',in_img1,num1);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.
num1 – a double specifying to which power to raise the input pixels' value.

Description:
out_img1 – An image which has dimensions the same as the input image. The output pixel values are the normalised power function of the input pixel values, i.e. in_img1^num1.

Example:
```
img = openimage('crown.jpg');
h=figure;image(uint8(img));set(h,'Name','Input Image');
[out_img1]=vsg('Power',img,2);
h=figure;image(uint8(out_img1));set(h,'Name','Power');
```

Notes:
1. For DICOM images, the function operates on all slices separately.
2. For RGB images, the function operates on all colour planes separately.

Square
The linear LUT is raised to power of 2.

function call:
[out_img1]=vsg('Square',in_img1);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.

Description:
out_img1 – An image which has dimensions the same as the input image. The output pixel values are the normalised squared input pixel values, i.e. in_img1^2.

Example:
```
img = openimage('crown.jpg');
h=figure;image(uint8(img));set(h,'Name','Input Image');
[out_img1]=vsg('Square',img);
h=figure;image(uint8(out_img1));set(h,'Name','Square');
```

Notes:
1. For DICOM images, the function operates on all slices separately.
2. For RGB images, the function operates on all colour planes separately.

DualThresh
Dual threshold operation. All pixels between the upper and lower thresholds are marked in WHITE.

function call:
[out_img1]=vsg('DualThresh',in_img1,low_th,high_th);

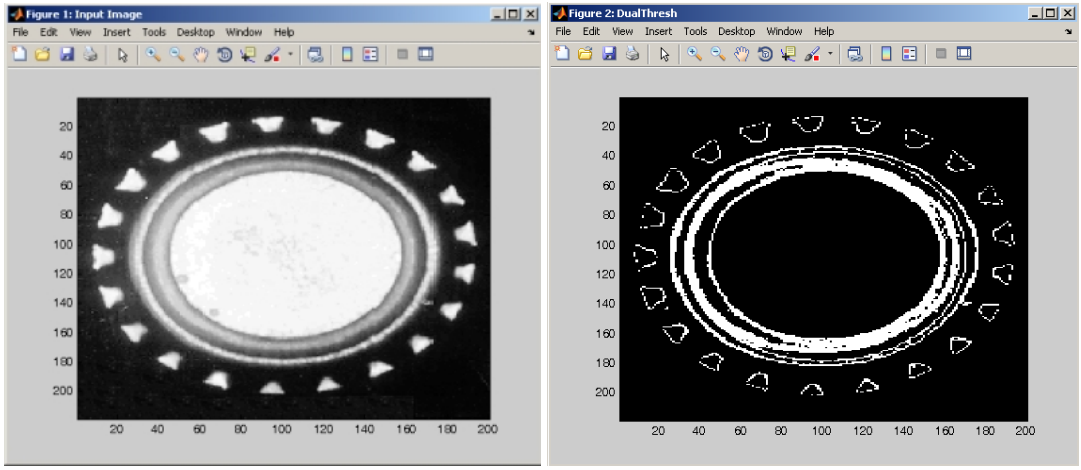
Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.
low_th – an integer or double specifying the lower threshold for the dual threshold function.
high_th – an integer or double specifying the higher threshold for the dual threshold function.

Description:
Values less than the user defined integer value lower and greater than the defined integer value upper are set to BLACK. Otherwise the pixel is set WHITE.

out_img1 – An image which has dimensions the same as the input image. The output pixel values are white if the input pixel values lie between the upper and lower threshold, and black otherwise.

Example:

```
img = openimage('crown.jpg');
h=figure;image(uint8(img));set(h,'Name','Input');
[out_img1]=vsg('DualThresh',img, 105, 184);
h=figure;image(uint8(out_img1));set(h,'Name','DualThresh');
```



Notes:

1. For DICOM images, the function operates on all slices separately.
2. For RGB images, the function operates on all colour planes separately.

EntropicThresh

Compute the entropy based threshold. Relies on maximising the total entropy of both the object and background regions to find the appropriate threshold.

function call:

[num1]=vsg('EntropicThresh',in_img1);

Arguments:

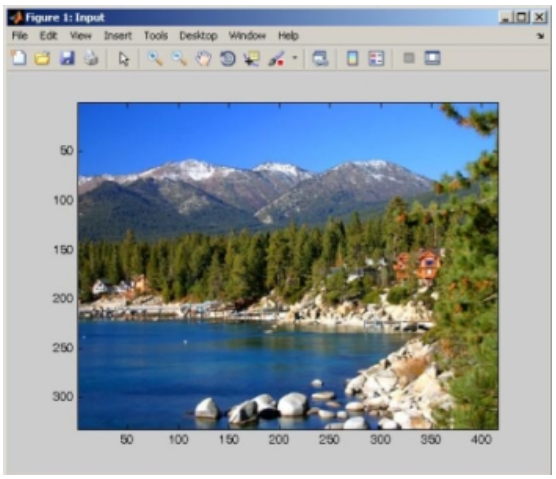
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.

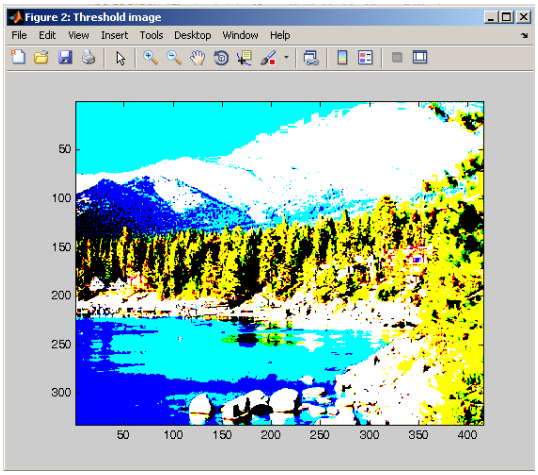
Description:

num1 – returns the optimum threshold value to separate the background from the foreground.

Example:

```
img = openimage('lake.jpg');
h=figure;image(uint8(img));set(h,'Name','Input Image');
[num1]=vsg('EntropicThresh',img)
[out_img1] = vsg('Threshold',img,num1);
h=figure; image(uint8(out_img1));set(h,'Name','Threshold image');

num1 = 79
```





Notes:
For DICOM and RGB images, a single threshold is returned for all planes/slices.

GSCapThresh

Sets all pixels of the input image which are less than the specified threshold to black.

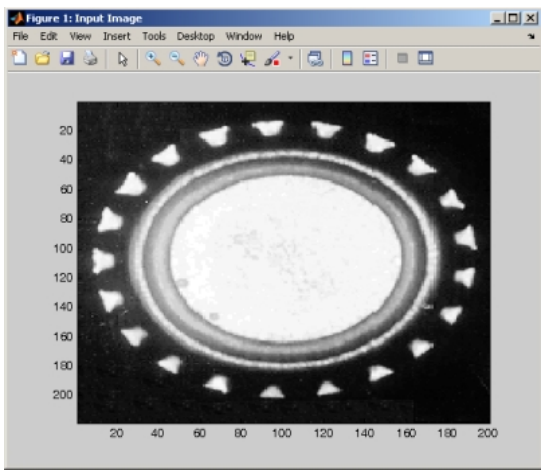
function call:
[out_img1]=vsg('GSCapThresh',in_img1,num1);

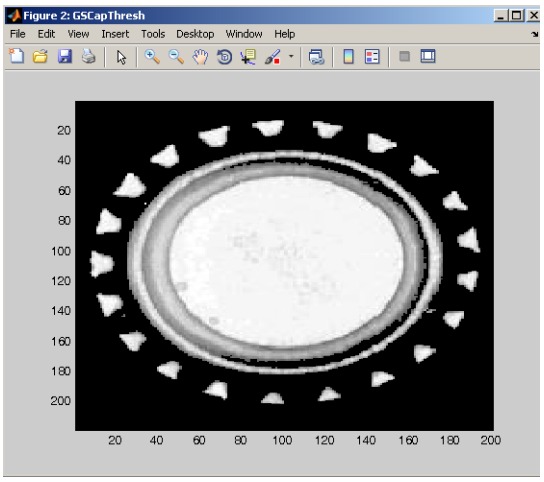
Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.
num1 – an integer or double specifying the threshold for the greyscale cap threshold function.

Description:
out_img1 – An image which has dimensions the same as the input image. The output pixel values are the input pixel values if they are larger than num1, otherwise they are black.

Example:

```
img=openimage('crown.jpg');
h=figure;image(uint8(img));set(h,'Name','Input image');
[out_img1]=vsg('GSCapThresh',img,100);
h=figure;image(uint8(out_img1));set(h,'Name','GSCapThresh');
```





Notes:
1. For DICOM images, the function operates on all slices separately.
2. For RGB images, the function operates on all colour planes separately.

MidThresh

Single threshold operation: threshold level = MIDGREY (127).

function call:
[out_img1]=vsg('MidThresh',in_img1);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.

Description:
out_img1 – An image which has dimensions the same as the input image. The output pixel values are white if they are larger than or equal to 127, otherwise they are black.

Example:
```
img = openimage('crown.jpg');
h=figure;image(uint8(img));set(h,'Name','Input Image');
[out_img1]=vsg('MidThresh',img);
h=figure;image(uint8(out_img1));set(h,'Name','MidThresh');
```

Notes:
1. For DICOM images, the function operates on all slices separately.
2. For RGB images, the function operates on all colour planes separately.

Threshold

function call:
[out_img1]=vsg('Threshold',in_img1,thresh);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.
thresh – an integer or double specifying the threshold for the greyscale threshold function.

Description:
out_img1 – An image which has dimensions the same as the input image. The output pixel values are white if they are larger than or equal to thresh, otherwise they are black.

Example:
```
img = openimage('crown.jpg');
h=figure;image(uint8(img));set(h,'Name','Input Image');
[out_img1]=vsg('Threshold',img,100);
h=figure;image(uint8(out_img1));set(h,'Name','Threshold');
```

Notes:
1. For DICOM images, the function operates on all slices separately.
2. For RGB images, the function operates on all colour planes separately.

3x3Thresh

Adaptive threshold in a 3x3 region.

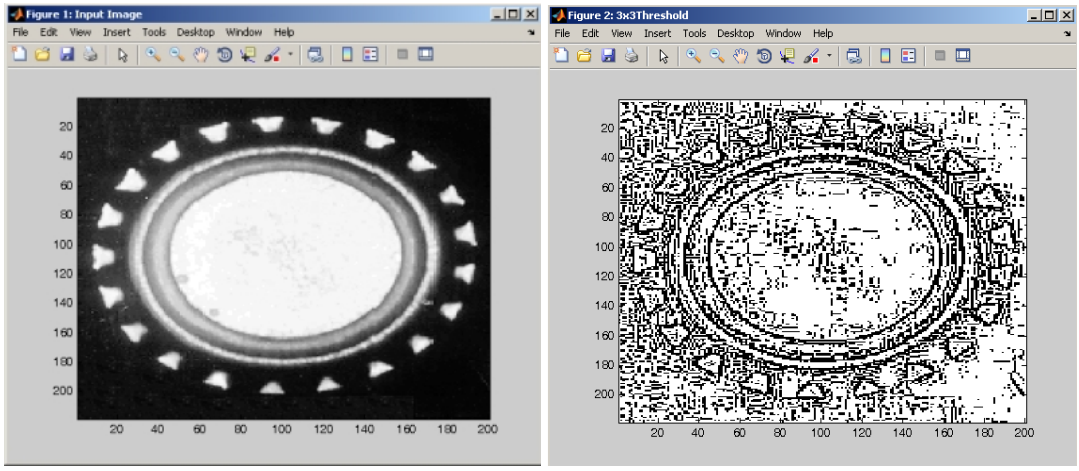
function call:
[out_img1]=vsg('3x3Thresh',in_img1,offset);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.
offset an integer or double specifying the threshold offset for the 3x3 threshold function.

Description:
Threshold = mean of 3×3 region – *offset* (α), where *alpha* is a user defined offset value
out_img1 – An image which has dimensions the same as the input image. The output values are black if the input values are smaller than the 3×3 mean minus the *offset*, otherwise they are white.

Example:

```
img = openimage('crown.jpg');
h=figure;image(uint8(img));set(h,'Name','Input Image');
[out_img1]=vsg('3x3Thresh',img, 0);
h=figure;image(uint8(out_img1));set(h,'Name','3x3Threshold');
```



Notes:
1. For DICOM images, the function operates on all slices separately.
2. For RGB images, the function operates on all colour planes separately.

5x5Thresh

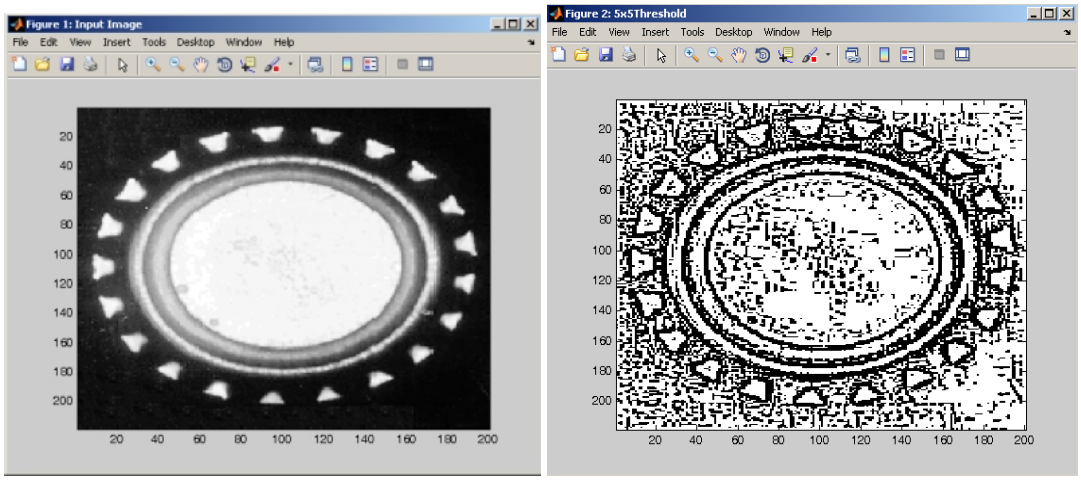
Adaptive threshold in a 5x5 region.

function call:
[out_img1]=vsg('5x5Thresh',in_img1,offset);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.
offset an integer or double specifying the threshold offset for the 5x5 threshold function.

Description:
Threshold = mean of 5×5 region – *offset* (α), where *alpha* is a user defined offset value (in vsg toolbox API it is set to zero).
out_img1 – An image which has dimensions the same as the input image. The output values are black if the input values are smaller than the 5×5 mean minus the offset, otherwise they are white.

Example:
```
img = openimage('crown.jpg');
h=figure;image(uint8(img));set(h,'Name','Input Image');
[out_img1]=vsg('5x5Thresh',img, 0);
h=figure;image(uint8(out_img1));set(h,'Name','5x5Threshold');
```



Notes:
1. For DICOM images, the function operates on all slices separately.
2. For RGB images, the function operates on all colour planes separately.

TripleThresh

This operation produces an LUT in which all pixels below the user specified lower level appear black, all pixels between the user specified lower level and the user specified upper level inclusively appear grey and all pixels above the user specified upper level appear white.

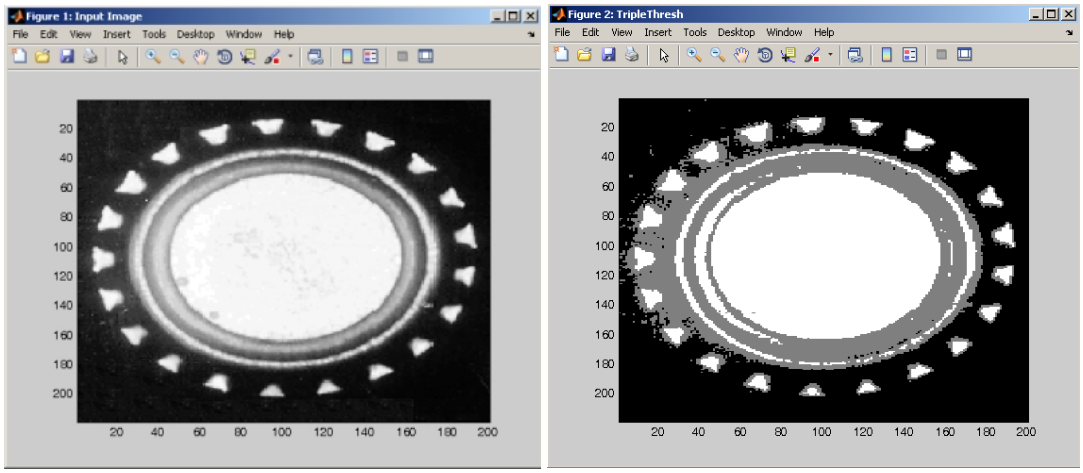
function call:
[out_img1]=vsg('TripleThresh',in_img1,low_th,high_th);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.
low_th – an integer or double specifying the lower threshold for the triple threshold function.
high_th – an integer or double specifying the higher threshold for the triple threshold function.

Description:
out_img1 – An image which has dimensions the same as the input image. Values between BLACK and the user defined integer value *lower* are set to BLACK (0). Values between the user defined integer value *upper* and WHITE (255) are set to WHITE. Otherwise the pixel is set to MIDGREY (midway between the minimum and maximum grey level).

Example:

```
img = openimage('crown.jpg');
h=figure;image(uint8(img));set(h,'Name','Input Image');
[out_img1]=vsg('TripleThresh',img,50,200);
h=figure;image(uint8(out_img1));set(h,'Name','TripleThresh');
```



Notes:

1. For DICOM images, the function operates on all slices separately.
2. For RGB images, the function operates on all colour planes separately.

GreyScaler

Converts RGB image to greyscale

function call:

[out_img1]=vsg('GreyScaler',in_img1);

Arguments:

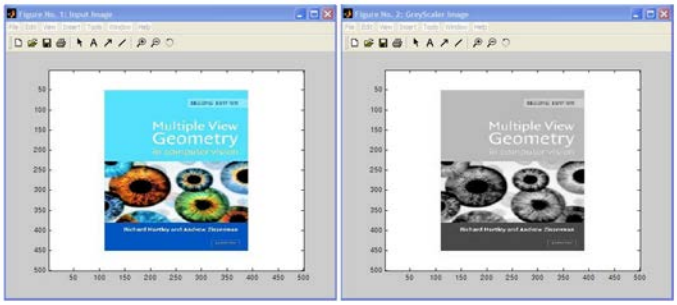
in_img1 input image, 3 channel RGB

Description:

out_img1 – An image which has dimensions the same as the input image. The output values are taken as the average value across all three colour planes.

Example:

```
img = openimage('hartley.jpg');
h=figure;image(uint8(img));set(h,'Name','Input Image');
[out_img1]=vsg('GreyScaler',img);
h=figure;image(uint8(out_img1));set(h,'Name','GreyScaler Image');
```



Transform Functions

BinGrey

Convert WHITE pixels in a binary image to a given greyscale.

function call:
[out_img1]=vsg('BinGrey',in_img1,num1);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.
num1 – an integer or double specifying the output value of input white regions.

Description:
Converts a binary image to a user defined grey scale grey.
out_img1 – An image which has dimensions the same as the input image. The output pixel values are set to num1 if the corresponding input pixels are white.

Example:

```
img=openimage('lake.jpg');
h=figure;image(uint8(img));set(h,'Name','Input image');
[out_img1] = vsg('Threshold',img,100);
h=figure; image(uint8(out_img1));set(h,'Name','Threshold image');
[out_img1]=vsg('BinGrey',out_img1,200);
h=figure; image(uint8(out_img1));set(h,'Name','Grey image');
```

Notes:
1. For DICOM images, the function operates on all slices separately.
2. For RGB images, the function operates on all colour planes separately.

Hough

Line Hough Transform.

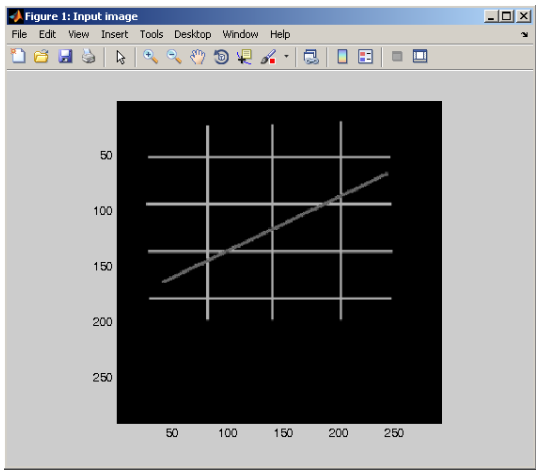
function call:
[out_img1]=vsg('Hough',in_img1);

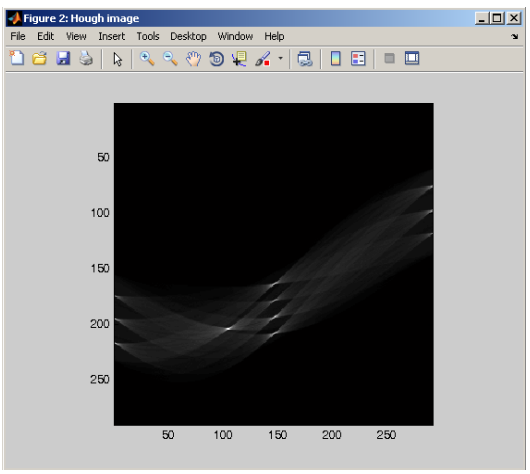
Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.

Description:
The Hough function uses Hough Transform (HT) to detect straight lines from binary image.
out_img1 – An image displaying the Hough transform of the input image. The dimensions are the same as the input image dimensions.

Example:

```
img=openimage('sixth.bmp');
h=figure;image(uint8(img));set(h,'Name','Input image');axis image;
[img] = vsg('Threshold',img,50);
[out_img1]=vsg('Hough',img);
h=figure; imagesc((out_img1(:,:,1)));colormap(gray),set(h,'Name','Hough
image'); axis image;
```





Notes:
1. For DICOM images, the function operates on all slices separately.
2. For RGB images, the function operates on the red colour plane.

InvHough

Inverse Hough Transform. The integer input specifies how many of the brightest pixels shall be taken into account when performing the Inverse Hough operation.

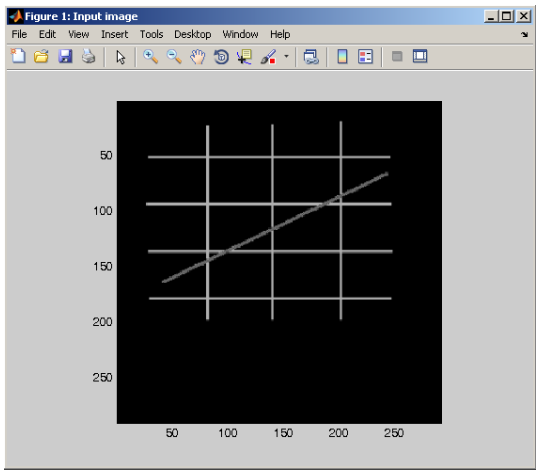
function call:
[out_img1]=vsg('InvHough',in_img1,num1);

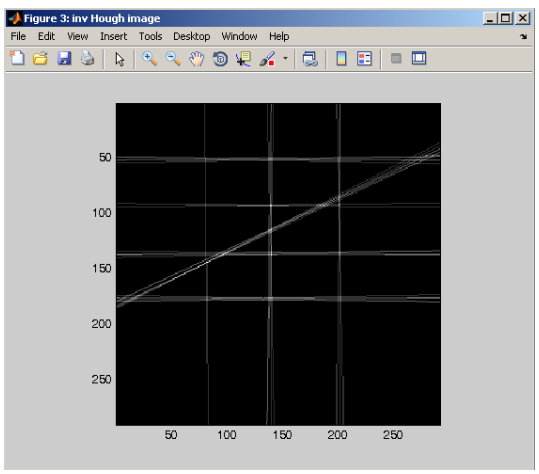
Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image. This should be a Hough space image.
num1 – an integer representing the number of highest input pixels to use in the inverse transform.

Description:
out_img1 – An image which has dimensions which are the same as the input image dimensions. The output image contains the lines corresponding to the num1 most significant pixels of the input Hough space image.

Example:

```
img=openimage('sixth.bmp');
h=figure;image(uint8(img));set(h,'Name','Input image');axis image;
[img] = vsg('Threshold',img,50);
[out_img1]=vsg('Hough',img);
h=figure; imagesc((out_img1(:,:,1)));colormap(gray),set(h,'Name','Hough
image'); axis image;
[out_img2]=vsg('InvHough',out_img1,30);
h=figure; imagesc((out_img2(:,:,1)));colormap(gray),set(h,'Name','inv
Hough image'); axis image;
```





Notes:
1. For DICOM images, the function operates on all slices separately.
2. For RGB images, the function operates on the red colour plane.

CircHough

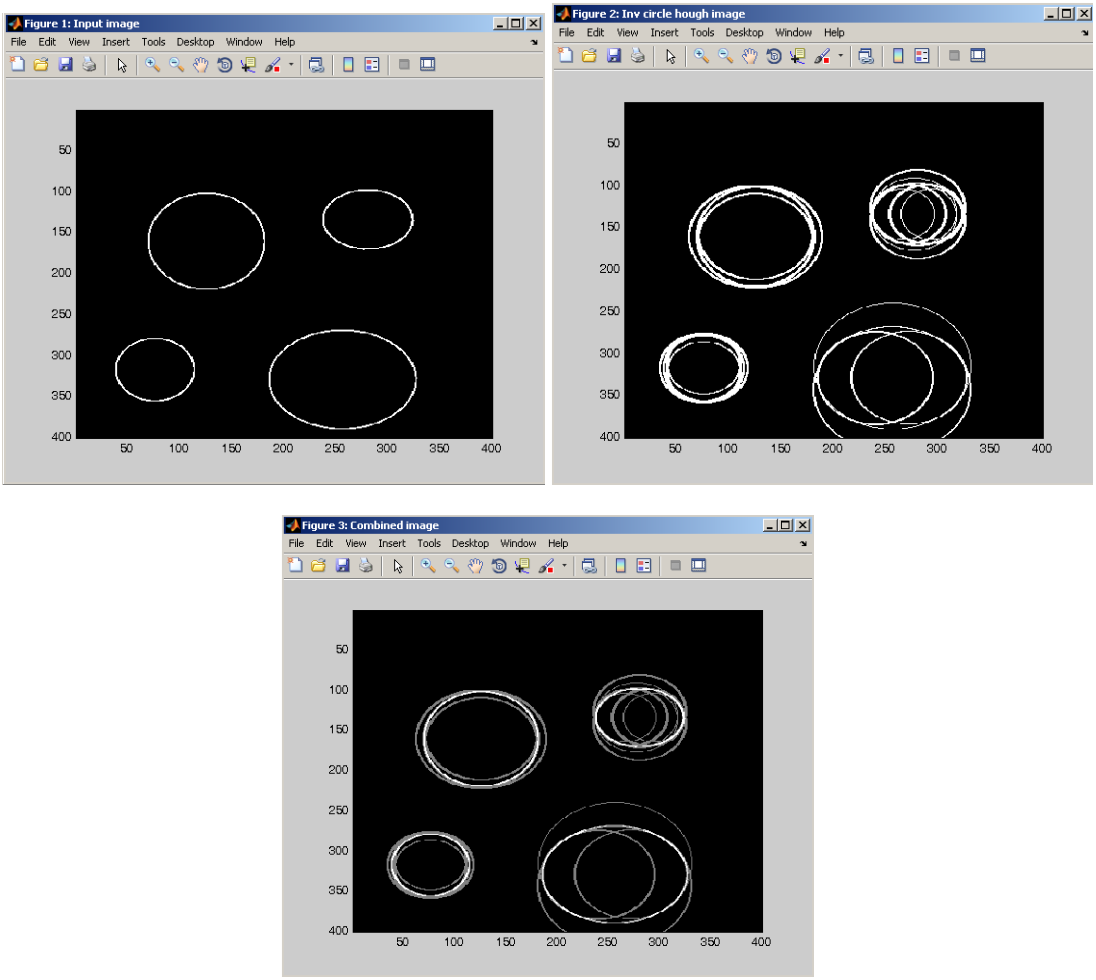
Circular Hough Transform.

function call:
[out_img1]=vsg('CircHough',in_img1,num1,num2,num3);
Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image.
num1 – an integer representing the smallest circle to look for in the input image.
num2 – an integer representing the largest circle to look for in the input image.
num3 – an integer representing the size of the radius increment between num1 and num2.

Description:
The CircHough function applies the generalized Hough Transform (HT) to detect circles from a binary image.
out_img1 – An image displaying the Circular Hough transform of the input image. The dimensions are the same as the input image dimension in the x and y direction, and will have one slice for each size of circle to be found, i.e. (num2-num1)/num3;

Example:

```
in_img1=openimage('circle.jpg');
in_img1=255*(in_img1>128);
h=figure, image(uint8(in_img1)); set(h,'Name','Input image');
tic;[out_img1]=vsg('CircHough',in_img1,10,100,5);toc
[out_img2]=vsg('Threshold',out_img1,0.90*max(max(max(out_img1))));
[out_img3]=vsg('InvCircHough',out_img2,10,100,5);
[out_img4]=vsg('Add',in_img1,out_img3);
h=figure, image(uint8(out_img3)); set(h,'Name','Inv circle hough
image');
h=figure, image(uint8(out_img4));set(h,'Name','Combined image');
```



Notes:
For RGB images, the function operates on the red colour plane.

InvCircHough

Inverse Hough Transform.

function call:
[out_img1]=vsg('InvCircHough',in_img1,num1,num2,num3);

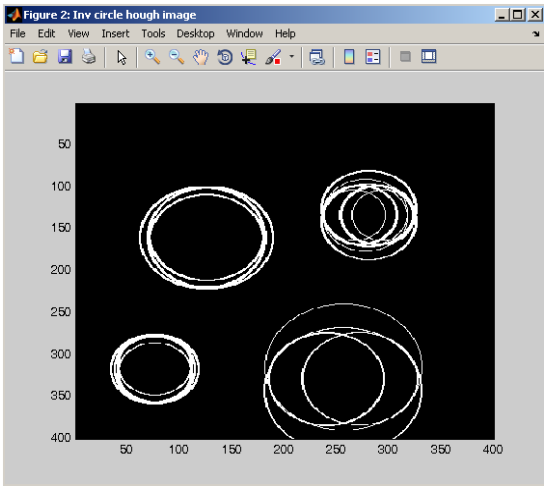
Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image.
num1 – an integer representing the smallest circle radius of the input image.
num2 – an integer representing the largest circle radius of the input image.
num3 – an integer representing the size of the radius increment between num1 and num2.

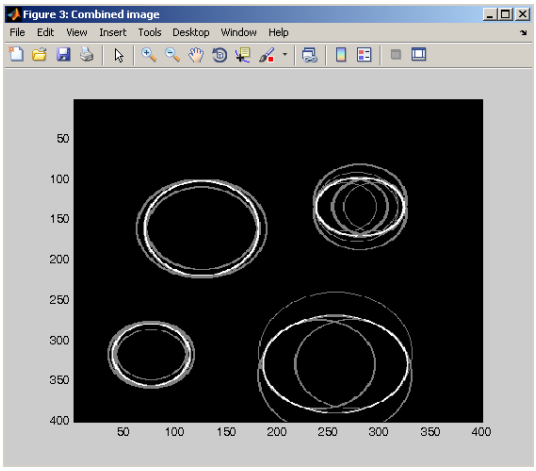
Description:
out_img1 – An image displaying the Inverse Circular Hough transform of the input image. The dimensions are the same as the input image dimension in the x and y direction, and will be a 3 channel RGB image containing the detected circles.

Example:

```
in_img1=openimage('circle.jpg');
```

```
in_img1=255*(in_img1>128);
h=figure, image(uint8(in_img1)); set(h,'Name','Input image');

tic;[out_img1]=vsg('CircHough',in_img1,10,100,5);toc
[out_img2]=vsg('Threshold',out_img1,0.90*max(max(max(out_img1))));
[out_img3]=vsg('InvCircHough',out_img2,10,100,5);
[out_img4]=vsg('Add',in_img1,out_img3);
h=figure, image(uint8(out_img3)); set(h,'Name','Inv circle hough
image');
h=figure, image(uint8(out_img4));set(h,'Name','Combined image');
```





Notes:
For RGB images, the function operates on the red colour plane.

CircDetector

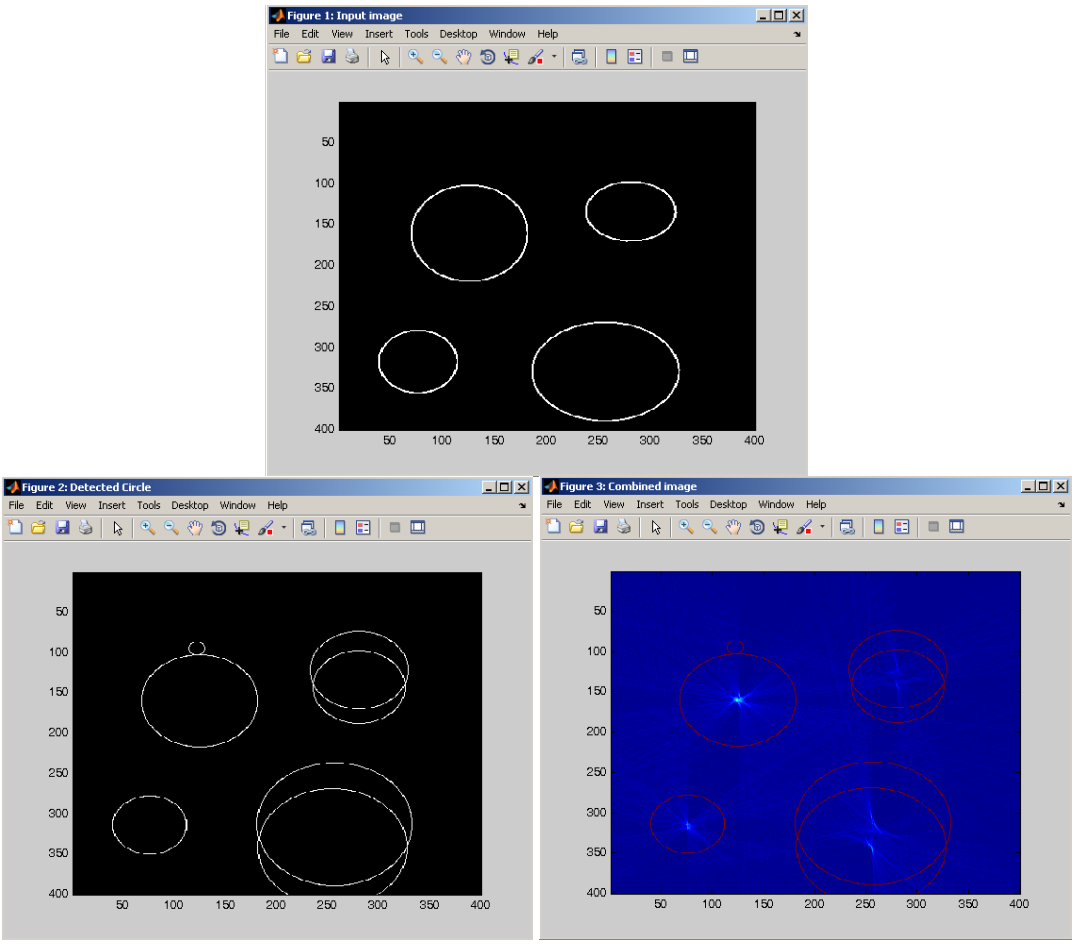
function call:
[out_img1,out_img2]=vsg('CircDetector',in_img1);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.

Description:
out_img1 – An image displaying the detected circles found from the input image, The output image will have the same dimensions as the input image.
out_img2 – An image displaying the detected circles found from the input image superimposed on the radial lines used to find the circles in the first place. The output image will have the same dimensions as the input image.

Example:

```
in_img1=openimage('circle.jpg');
in_img1=255*(in_img1>128);
[in_img1]=vsg('FullThin',in_img1);
h=figure, image(uint8(in_img1)); set(h,'Name','Input image');
[out_img1,out_img2]=vsg('CircDetector',in_img1);
h=figure, image(uint8(out_img1)); set(h,'Name','Detected Circle');
h=figure, imagesc((out_img2(:,:,1)));set(h,'Name','Combined image');
```



Notes:
For RGB images, the function operates on the red colour plane.

MedialAxis

Medial axis transform operation. Binary image showing the simple skeleton.

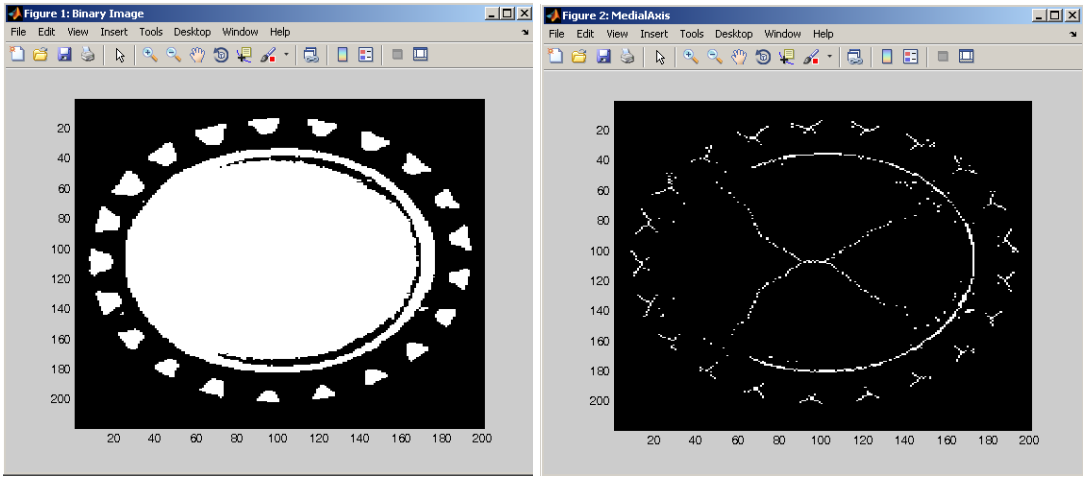
function call:
[out_img1]=vsg('MedialAxis',in_img1);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.

Description:
The function shows the application of Medial Axis Transform (MAT) on binary image.
out_img1 – An image which has dimensions the same as the input image. The centre axis of any white objects in the input image.

Example:

```
img = openimage('crown.jpg');
[out_img1]=vsg('Threshold',img,100);
h=figure;image(uint8(out_img1));set(h,'Name','Binary Image');
[out_img1]=vsg('MedialAxis',out_img1);
h=figure;image(uint8(out_img1));set(h,'Name','MedialAxis');
```



Notes:

1. For DICOM images, the function operates on all slices separately.
2. For RGB images, the function operates on all colour planes separately.

Co-occurrence Matrix Functions

CoOcMatGen

Compute the co-occurrence matrix.

function call:
[out_img1]=vsg('CoOcMatGen',in_img1);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.

Description:
A co-occurrence matrix generation for grey scale or colour images. *out_img1* – An image which has dimensions 256x256 per slice/colour channel and the same number of slices/ colour channels as the input image. The output image represents the greyscale co-occurrence matrix of each channel/slice.

Example:
```
img = openimage('lake.jpg');
h=figure;image(img);set(h,'Name','Input Image'
[out_img1]=vsg('CoOcMatGen',img);
h=figure;imagesc((out_img1(:,:,1)));colormap(gray);set(h,'Name','Cooccu
rrenceMatrixGenerator');
```

Notes:
1. For DICOM images, the function operates on all slices separately.
2. For RGB images, the function operates on all colour planes separately.

CoHomog

Compute the homogeneity of the co-occurrence matrix.

function call:
[num1]=vsg('CoHomog',in_img1);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image. The image should be a co-occurrence matrix of dimension 256x256 per slice/colour-channel.

Description:
num1 – the homogeneity of the image used to create the input co-occurrence matrix image. This value measures the closeness of the distribution of elements in the co-occurrence matrix to its diagonal.

Example:
```
mg = openimage('lake.jpg');
%[img]=convert(img);
h=figure;image(img);set(h,'Name','Input Image');
[out_img1]=vsg('CoOcMatGen',img);
h=figure;imagesc((out_img1(:,:,1)));colormap(gray);set(h,'Name','Cooccu
rrenceMatrixGenerator');
[num1]=vsg('CoHomog',out_img1)

num1 =  9.7808e+004
```

Notes:
1. For DICOM images, the function returns the mean homogeneity per slice
2. For RGB images, the function returns the mean homogeneity per colour plane.

CoEntropy

Compute the entropy of the co-occurrence matrix.

function call:
[num1]=vsg('CoEntropy',in_img1);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image. The image should be a co-occurrence matrix of dimension 256x256 per slice/colour-channel.

Description:
The function CoEntropy computes the entropy of the co-occurrence matrix of an image. num1 – the entropy of the image used to create the input co-occurrence matrix image.

Example:
```
img = openimage('lake.jpg');
h=figure;image(img);set(h,'Name','Input Image');
[out_img1]=vsg('CoOcMatGen',img);
[num1]=vsg('CoEntropy',out_img1)
h=figure;imagesc((out_img1(:,:,1)));colormap(gray);set(h,'Name','Cooccu
rrenceMatrixGenerator');
num1 =  -1.0468e+006
```

Notes:
1. For DICOM images, the function returns the mean entropy per slice
2. For RGB images, the function returns the mean entropy per colour plane.

CoEnergy

Compute the energy of the co-occurrence matrix.

function call:
[num1]=vsg('CoEnergy',in_img1);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image. The image should be a co-occurrence matrix of dimension 256x256 per slice/colour-channel.

Description:
The CoEnergy function calculates the energy of co-occurrence matrix of an image.
num1 – the energy of the image used to create the input co-occurrence matrix image. Returns the sum of squared elements in the co-occurrence matrix.

Example:
```
img = openimage('lake.jpg');
h=figure;image(img);set(h,'Name','Input Image');
[out_img1]=vsg('CoOcMatGen',img);
[num1]=vsg('CoEnergy',out_img1)
h=figure;imagesc((out_img1(:,:,1)));colormap(gray);set(h,'Name','Cooccu
rrenceMatrixGenerator');
num1 = 19256891
```

Notes:
1. For DICOM images, the function returns the mean energy per slice
2. For RGB images, the function returns the mean energy per colour plane.

CoContrast

Compute the contrast of the co-occurrence matrix.

function call:
[num1]=vsg('CoContrast',in_img1);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image. The image should be a co-occurrence matrix of dimension 256x256 per slice/colour-channel.

Description:
num1 – the contrast of the image used to create the input co-occurrence matrix image. Returns a measure of the intensity contrast between a pixel and its neighbour over the whole image.

Example:
```
img = openimage('lake.jpg');
h=figure;image(img);set(h,'Name','Input Image');
[out_img1]=vsg('CoOcMatGen',img);
[num1]=vsg('CoContrast',out_img1);
h=figure;imagesc((out_img1(:,:,1)));colormap(gray);set(h,'Name','Cooccu
rrenceMatrixGenerator');
```

Notes:
1. For DICOM images, the function returns the mean contrast per slice
2. For RGB images, the function returns the mean contrast per colour plane.

Distance Transform Functions

L2RDistance

Left to right distance transform (input binary image).

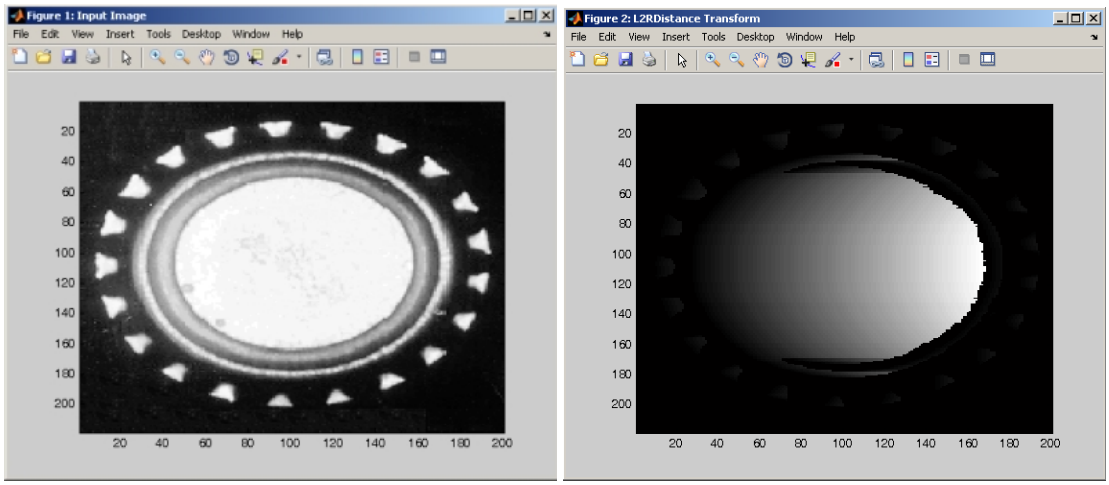
function call:
[out_img1]=vsg('L2RDistance',in_img1);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.

Description:
out_img1 – the output image values are set as, the distance that input white pixels are from the edge of the white region travelling in a left to right direction.

Example:

```
img = openimage('crown.jpg');
h=figure;image(img); set(h,'Name','Input Image');
[out_img1]=vsg('Threshold',img,100);
[out_img1]=vsg('L2RDistance',out_img1);
h=figure;imagesc(out_img1(:,:,1));colormap(gray);set(h,'Name','L2RDistance Transform');
```



Notes:
1. For DICOM images, the function operates on all slices separately.
2. For RGB images, the function operates on all colour planes separately.

R2LDistance

Right to left distance transform (input binary image).

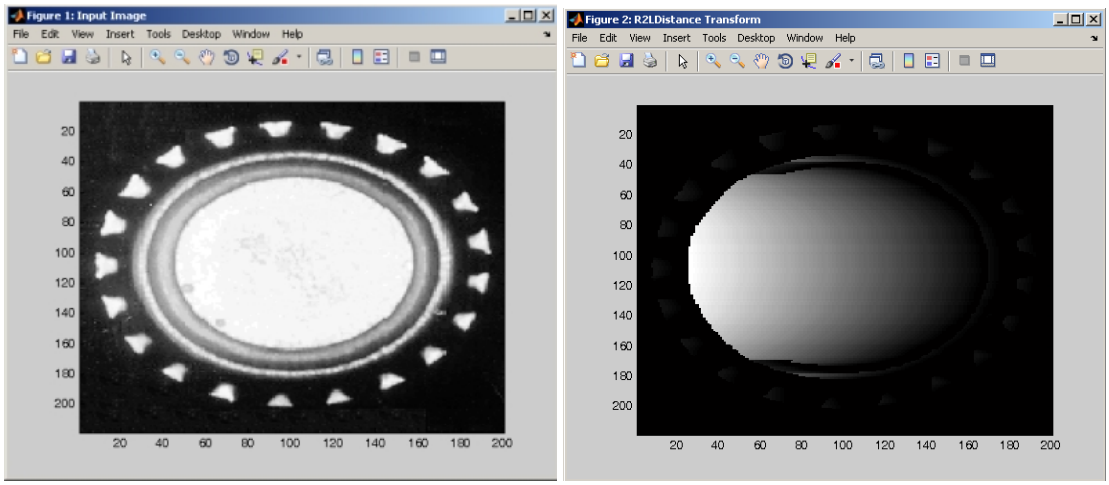
function call:
[out_img1]=vsg('R2LDistance',in_img1);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.
Description:

out_img1 – the output image values are set as, the distance that input white pixels are from the edge of the white region travelling in a right to left direction.

Example:

```
img = openimage('crown.jpg');
h=figure;image(img);set(h,'Name','Input Image');
[out_img1]=vsg('Threshold',img,100);
[out_img1]=vsg('R2LDistance',out_img1);
h=figure;imagesc(out_img1(:,:,1));colormap(gray);set(h,'Name','R2LDista
nce Transform');
```



Notes:

1. For DICOM images, the function operates on all slices separately.
2. For RGB images, the function operates on all colour planes separately.

T2BDistance

Top to bottom distance transform (input binary image).

function call:

[out_img1]=vsg('T2BDistance',in_img1);

Arguments:

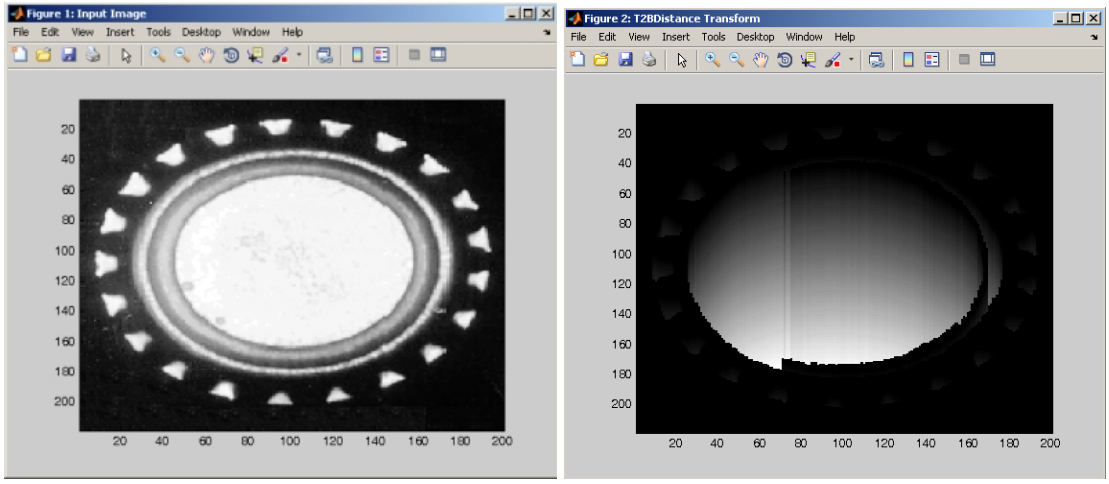
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.

Description:

out_img1 – the output image values are set as, the distance that input white pixels are from the edge of the white region travelling in a top to bottom direction.

Example:

```
img = openimage('crown.jpg');
h=figure;image(img);set(h,'Name','Input Image');
[out_img1]=vsg('Threshold',img,100);
[out_img1]=vsg('T2BDistance',out_img1);
h=figure;imagesc(out_img1(:,:,1));colormap(gray);set(h,'Name','T2BDista
nce Transform');
```



Notes:
1. For DICOM images, the function operates on all slices separately.
2. For RGB images, the function operates on all colour planes separately.

B2TDistance

Bottom to top distance transform (input binary image).

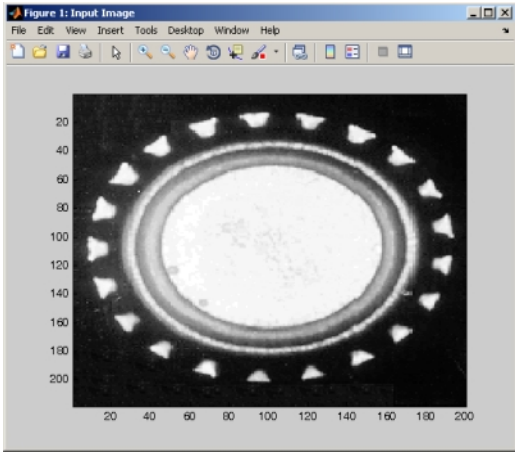
function call:
[out_img1]=vsg('B2TDistance',in_img1);

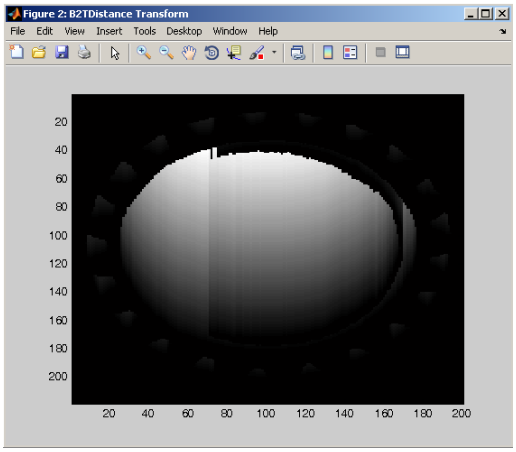
Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.

Description:
out_img1 – the output image values are set as, the distance that input white pixels are from the edge of the white region travelling in a bottom to top direction.

Example:

```
img = openimage('crown.jpg');
h=figure;image(img);set(h,'Name','Input Image');
[out_img1]=vsg('Threshold',img,100);
[out_img1]=vsg('B2TDistance',out_img1);
h=figure;imagesc(out_img1(:,:,1));colormap(gray);set(h,'Name','B2TDista
nce Transform');
```





Notes:
1. For DICOM images, the function operates on all slices separately.
2. For RGB images, the function operates on all colour planes separately.

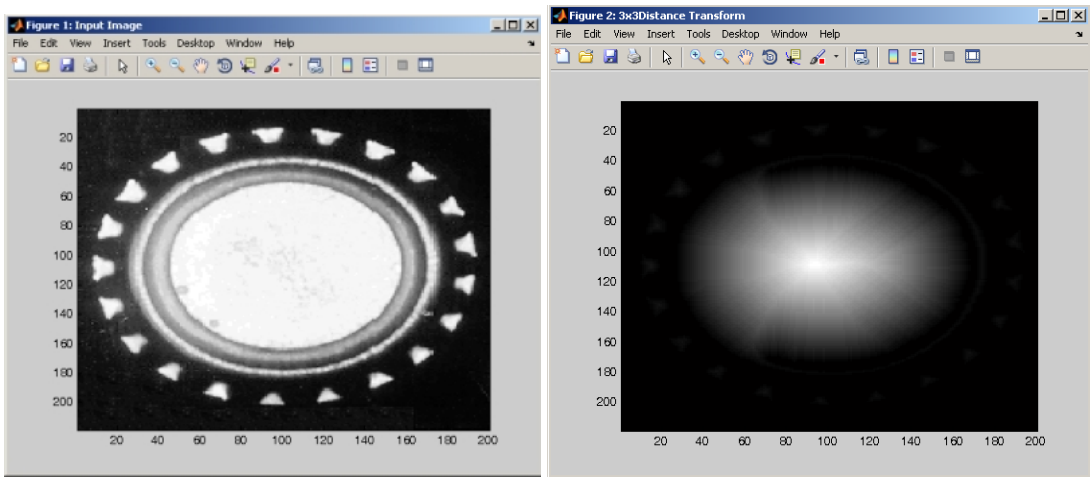
3x3Distance

function call:
[out_img1]=vsg('3x3Distance',in_img1);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.

Description:
out_img1 – the output image values are set as the distance that input white pixels are from the nearest edge of the white region of which they are part of.

Example:
```
img = openimage('crown.jpg');
h=figure;image(img);set(h,'Name','Input Image');
[out_img1]=vsg('Threshold',img,100);
[out_img1]=vsg('3x3Distance',out_img1);
h=figure;imagesc(out_img1(:,:,1));colormap(gray);set(h,'Name','3x3Dista
nce Transform');
```



Notes:
1. For DICOM images, the function operates on all slices separately.
2. For RGB images, the function operates on all colour planes separately.
3. Ensure that binary images have a range {0.255}.

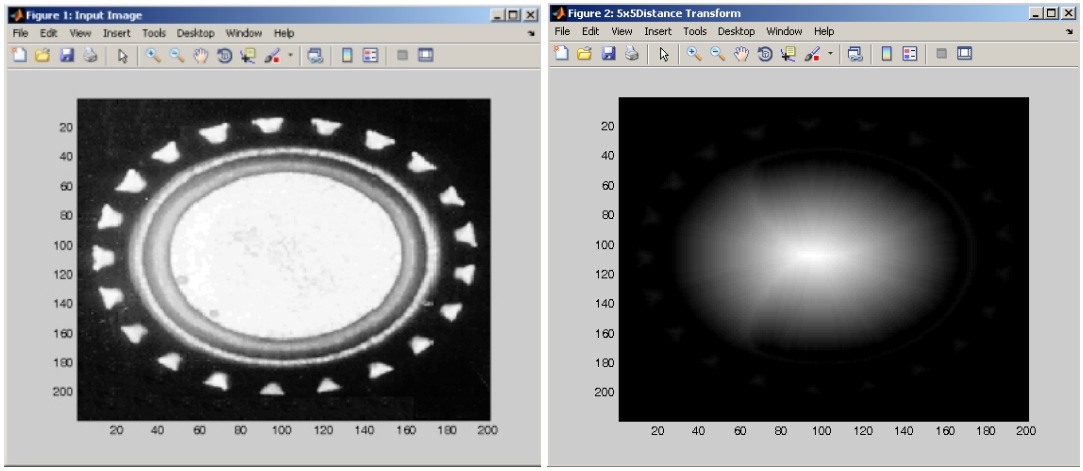
5x5Distance

function call:
[out_img1]=vsg('5x5Distance',in_img1);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.

Description:
out_img1 – the output image values are set as the distance that input white pixels are from the nearest edge of the white region of which they are part of.

Example:
```
img = openimage('crown.jpg');
h=figure;image(img);set(h,'Name','Input Image');
[out_img1]=vsg('Threshold',img,100);
[out_img1]=vsg('5x5Distance',out_img1);
h=figure;imagesc(out_img1(:,:,1));colormap(gray);set(h,'Name','5x5Dista
nce Transform');
```



Notes:
1. For DICOM images, the function operates on all slices separately.
2. For RGB images, the function operates on all colour planes separately.
3. Ensure that binary images have a range {0.255}.

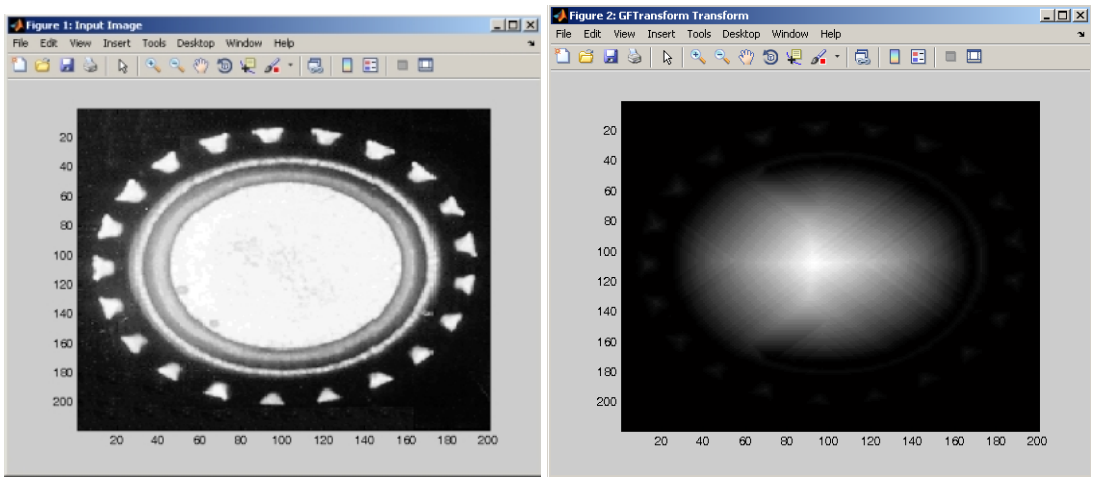
GFTransform

function call:
[out_img1]=vsg('GFTransform',in_img1);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.

Description:
out_img1 – the output image values are set as the distance that input white pixels are from the nearest edge of the white region of which they are part of.

Example:
```
img = openimage('crown.jpg');
h=figure;image(img);set(h,'Name','Input Image');
[out_img1]=vsg('Threshold',img,100);
[out_img1]=vsg('GFTransform',out_img1);
%h=figure;image(uint8(out_img1));set(h,'Name','GFTransform Transform');
h=figure;imagesc(out_img1(:,:,1));colormap(gray);set(h,'Name','GFTransf
orm Transform');
```



Notes:

1. For DICOM images, the function operates on all slices separately.
2. For RGB images, the function operates on all colour planes separately.
3. Ensure that binary images have a range {0.255}.

Cosine Transform Functions

DCT

Direct Cosine Transform operation.

function call:
[out_img1]=vsg('DCT',in_img1);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.

Description:
out_img1 – the output image values are the discrete cosine transform of the input image.

Example:
```
img = openimage('lake.jpg');
h=figure;image(img);set(h,'Name','Input Image');
[out_img1]=vsg('DCT',img);
[out_img2]=vsg('ViewDCT',out_img1);
h=figure; image(uint8(out_img2));set(h,'Name','DCT image');
```

Notes:
1. For DICOM images, the function operates on all slices separately.
2. For RGB images, the function operates on all colour planes separately.

IDCT

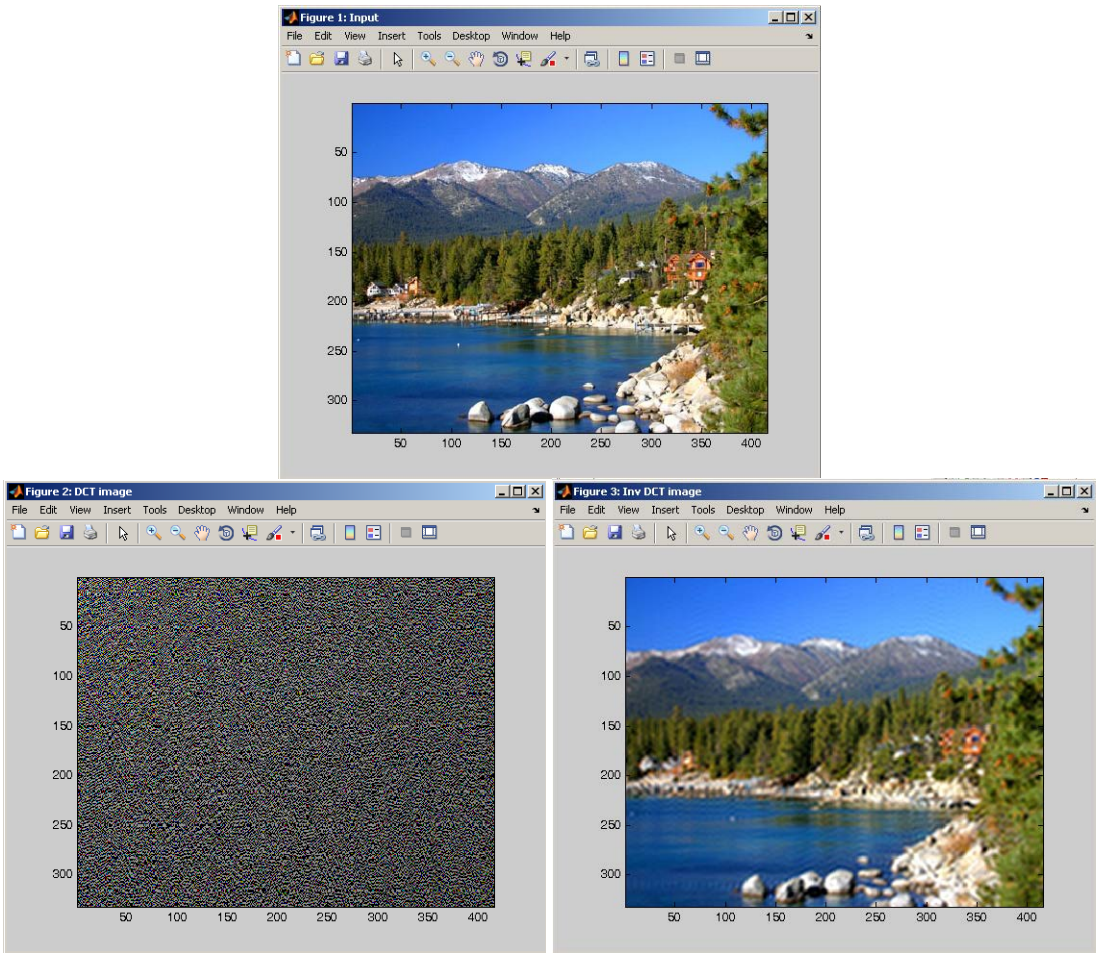
Inverse DCT (filtering factor is user defined).

function call:
[out_img1]=vsg('IDCT',in_img1,val);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.
val – a double representing the filtering factor for *in_img1* (0.0 to 1.0).

Description:
out_img1 – the output image values are the inverse discrete cosine transform of the input image.

Example:
```
img = openimage('lake.jpg');
h=figure;image(img);set(h,'Name','Input Image');
[out_img1]=vsg('DCT',img);
[out_img2]=vsg('ViewDCT',out_img1);
h=figure; image(uint8(out_img2));set(h,'Name','DCT image');
[out_img3]=vsg('IDCT',out_img1,0.30);
h=figure; image(uint8(out_img3));set(h,'Name','Inv DCT image');
[out_img4]=vsg('Subtract',img,out_img3);
h=figure; image(uint8(out_img4));set(h,'Name','Diff image');
```



Notes:
1. For DICOM images, the function operates on all slices separately.
2. For RGB images, the function operates on all colour planes separately.

ViewDCT

Used to view the DCT image.

function call:
[out_img1]=vsg('ViewDCT',in_img1);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image, or DICOM image.

Description:
out_img1 – the output image values are the normalised log of the discrete cosine transform image.

Example:
```
img = openimage('lake.jpg');
h=figure;image(img);set(h,'Name','Input Image');
[out_img1]=vsg('DCT',img);
[out_img2]=vsg('ViewDCT',out_img1);
h=figure; image(uint8(out_img2));set(h,'Name','DCT image');
```

Notes:
1. For DICOM images, the function operates on all slices separately.
2. For RGB images, the function operates on all colour planes separately.

Fourier Transform Functions

FFT

Fast Fourier Transform: FFT.

function call:
[out_img1r, out_img1i]=vsg('FFT',in_img1);

Arguments:
in_img1 input image, 3 channel RGB, 1 channel greyscale or binary image.

Description:
out_img1r – the output image values are the real FFT coefficients of the input image.
out_img1i – the output image values are the imaginary FFT coefficients of the input image.

Example:
```
img = openimage('baboon.jpg');
h=figure;image(img);set(h,'Name','Input Image');
[out_img1,out_img2]=vsg('FFT',img);
```

Notes:
1. For DICOM images, the function operates on only the first slice, this will be updated.
2. For RGB images, the function operates on all colour planes separately.

ViewFFT

function call:
[out_img1]=vsg('ViewFFT',in_img1r,in_img1i);

Arguments:
in_img1r - input image, 3 channel RGB, 1 channel greyscale or binary image of the real FFT coefficients.
in_img1i - input image, 3 channel RGB, 1 channel greyscale or binary image of the imaginary FFT coefficients.

Description:
out_img1 – the output image values are the normalised log of the magnitude of the Fourier transform images.

Example:
```
img = openimage('baboon.jpg');
h=figure;image(img);set(h,'Name','Input Image');
[out_img1,out_img2]=vsg('FFT',img);
[out_img3]=vsg('ViewFFT',out_img1,out_img2);
h=figure;imagesc((out_img3(:,:,1)));colormap(gray);set(h,'Name','FFT');
```

Notes:
For RGB images, the function operates on all colour planes separately.

InvFFT

Inverse Fourier Transform.

function call:
[out_img1]=vsg('InvFFT',in_img1r,in_img1i);

Arguments:
in_img1r - input image, 3 channel RGB, 1 channel greyscale or binary image of the real FFT coefficients.
in_img1i - input image, 3 channel RGB, 1 channel greyscale or binary image of the imaginary FFT coefficients.

Description:
The *out_img1* – the output image values are the inverse Fourier transform of the input images.

Example:
```
h=figure;image(img);axis image;
[out_img1,out_img2]=vsg('FFT',img);
[out_img3]=vsg('ViewFFT',out_img1,out_img2);
h=figure;imagesc((out_img3(:,:,1)));colormap(gray);colormap(gray);set(h
,'Name','FFT');
[out_img4]=vsg('InvFFT',out_img1,out_img2);
h=figure; image(uint8(out_img4));set(h,'Name','InvFFT');
```

FFTAdaptiveLP

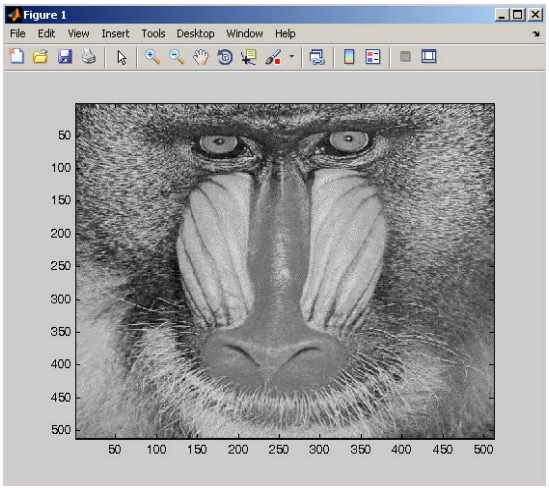
FFT adaptive lowpass filter.

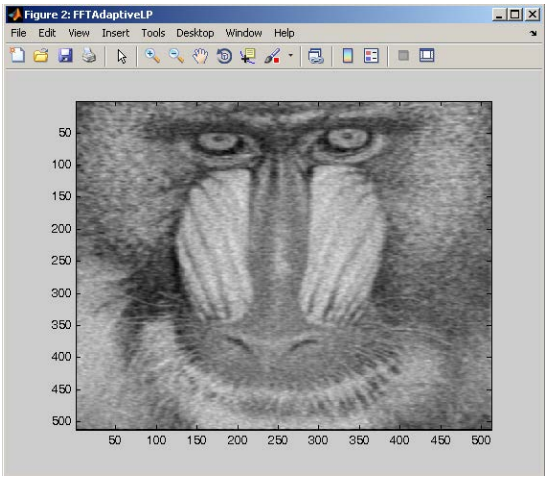
function call:
[out_img1r,out_img1i]=vsg('FFTAdaptiveLP',in_img1r,in_img1i,thresh);

Arguments:
in_img1r - input image, 3 channel RGB, 1 channel greyscale or binary image of the real FFT coefficients.
in_img1i - input image, 3 channel RGB, 1 channel greyscale or binary image of the imaginary FFT coefficients.
Thresh – a double which specifies a magnitude threshold, above which all input image values will be kept. Below this value will have output pixels set to 0.

Description:
The user defined input *limit* is the cutoff frequency for the filter. Unlike the low-pass filter which is based on the position of the filter in Fourier space, this approach is dependent on the real and imaginary values of all the data in the Fourier space.
out_img1r – the output image values are the real FFT coefficients of the input images, which have had a threshold applied to their values.
out_img1i – the output image values are the imaginary FFT coefficients of the input images, which have had a threshold applied to their values.

Example:
```
img = openimage('baboon.jpg');
[out_img1]=vsg('GreyScaler',img);
figure;image(uint8(out_img1));
[out_img1,out_img2]=vsg('FFT',out_img1);
[out_img1r,out_img1i]=vsg('FFTAdaptiveLP',out_img1,out_img2,0.3);
[out_img1]=vsg('InvFFT',out_img1r,out_img1i);
h=figure;image(uint8(out_img1));set(h,'Name','FFTAdaptiveLP');
```





Notes:
For RGB images, the function operates on all colour planes separately.

FFTGaussian

function call:
[out_img1r,out_img1i]=vsg('FFTGaussian',sizex,sizey,std);

Arguments:
sizex – an integer specifying the horizontal dimension of the output image.
sizey – an integer specifying the vertical dimension of the output image.
std – a double specifying the standard deviation of the output Gaussian kernel.

Description:
out_img1r – the output image values are the real FFT coefficients which make up a 2d Gaussian bell curve if the zero frequency was at the centre
out_img1i – the output image values are the imaginary FFT coefficients which are set to zero.

Example:
```
w = 100;
h = 82;
sd = 1.6;
[out_img1r,out_img1i]=vsg('FFTGaussian',w,h,sd);
h=figure;imagesc((out_img1r(:,:,1)));colormap(gray);set(h,'Name','FFTGa
ussian');
```

Notes:
The function always outputs a 3 channel image, all of which are identical. A single channel may be selected if needed.

FFTDivide

Divide one Fourier data file by another.

function call:
[out_img1r,out_img1i]=vsg('FFTDivide',in_img1r,in_img1i,in_img2r,in_img2i);

Arguments:
in_img1r - input image, 3 channel RGB, 1 channel greyscale or binary image of the real FFT coefficients.
in_img1i - input image, 3 channel RGB, 1 channel greyscale or binary image of the imaginary FFT coefficients.
in_img2r - input image, 3 channel RGB, 1 channel greyscale or binary image of the real FFT coefficients.
in_img2i - input image, 3 channel RGB, 1 channel greyscale or binary image of the imaginary FFT coefficients.

Description:
out_img1r – the output image values are the real FFT coefficients of the two input Fourier images divided, i.e. the complex division of in_img1 by in_img2.
out_img1r – the output image values are the imaginary FFT coefficients of the two input Fourier images divided, i.e. the complex division of in_img1 by in_img2.

Example:
```
img1 = openimage('baboon.jpg');
[out_img1r,out_img1i]=vsg('FFT',img1);
out_img2r = out_img1r/2;
out_img2i = out_img1i/2;
[out_img3r,out_img3i]=vsg('FFTDivide',out_img1r,out_img1i,out_img2r,out
_img2;
```

Notes:
For RGB images, the function operates on all colour planes separately.

FFTMultiply
Multiply two Fourier data files.

function call:
[out_img1r,out_img1i]=vsg('FFTMultiply',in_img1r,in_img1i,in_img2r,in_img2i);

Arguments:
in_img1r - input image, 3 channel RGB, 1 channel greyscale or binary image of the real FFT coefficients.
in_img1i - input image, 3 channel RGB, 1 channel greyscale or binary image of the imaginary FFT coefficients.
in_img2r - input image, 3 channel RGB, 1 channel greyscale or binary image of the real FFT coefficients.
in_img2i - input image, 3 channel RGB, 1 channel greyscale or binary image of the imaginary FFT coefficients.

Description:
out_img1r – the output image values are the real FFT coefficients of the two input Fourier images divided, i.e. the complex multiplication of in_img1 by in_img2.
out_img1r – the output image values are the imaginary FFT coefficients of the two input Fourier images divided, i.e. the complex multiplication of in_img1 by in_img2.

Example:
```
img1 = imread('baboon.jpg');
[out_img1r,out_img1i]=vsg('FFT',img1);
out_img2r = out_img1r/10;
```

```
out_img2i = out_img1i/10;
[out_img3r,out_img3i]=vsg('FFTMultiply',out_img1r,out_img1i,out_img2r,o
ut_img2i);
```

Notes:
For RGB images, the function operates on all colour planes separately.

FFTLowPass

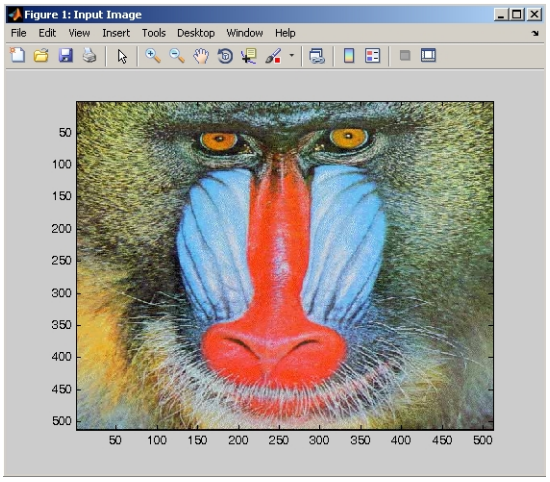
Low pass frequency filter.

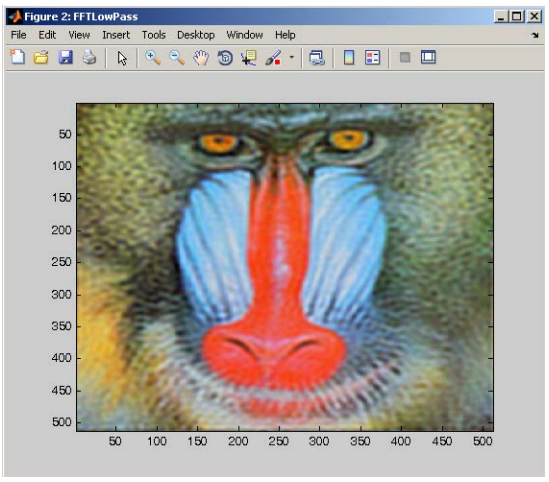
function call:
[out_img1r,out_img1i]=vsg('FFTLowPass',in_img1r,in_img1i,rad1);

Arguments:
in_img1r - input image, 3 channel RGB, 1 channel greyscale or binary image of the real FFT coefficients.
in_img1i - input image, 3 channel RGB, 1 channel greyscale or binary image of the imaginary FFT coefficients.
rad1 – a double which specifies a radius.

Description:
The user defined input value is the cutoff frequency for the low-pass filter.
out_img1r – the output image values are the real FFT coefficients of the input image if they lie within the radius, otherwise they are set to 0.
out_img1i – the output image values are the imaginary FFT coefficients of the input image if they lie within the radius, otherwise they are set to 0.

Example:
```
img = imread('baboon.jpg');
h=figure;image(img); set(h,'Name','Input Image');
[out_img1,out_img2]=vsg('FFT',img);
[out_img1r,out_img1i]=vsg('FFTLowPass',out_img1,out_img2,0.2);
[out_img2]=vsg('InvFFT',out_img1r,out_img1i);
h=figure; image(uint8(out_img2));set(h,'Name','FFTLowPass');
```





FFTHighPass

High pass frequency filter.

function call:

[out_img1r,out_img1i]=vsg('FFTHighPass',in_img1r,in_img1i,rad1);

Arguments:

in_img1r - input image, 3 channel RGB, 1 channel greyscale or binary image of the real FFT coefficients.

in_img1i - input image, 3 channel RGB, 1 channel greyscale or binary image of the imaginary FFT coefficients.

rad1 – a double which specifies a radius.

Description:

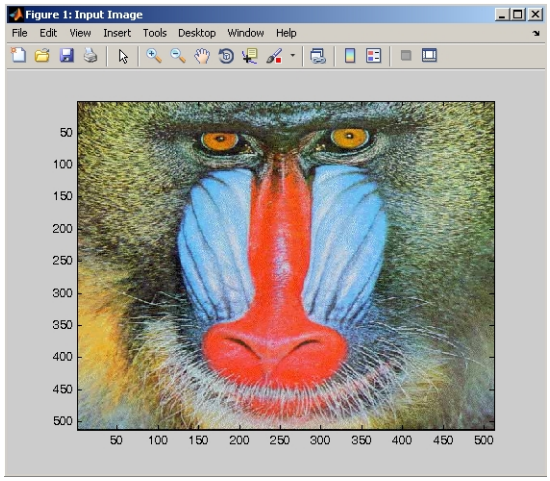
The user defined input value is the cuttoff frequency for the high-pass filter.

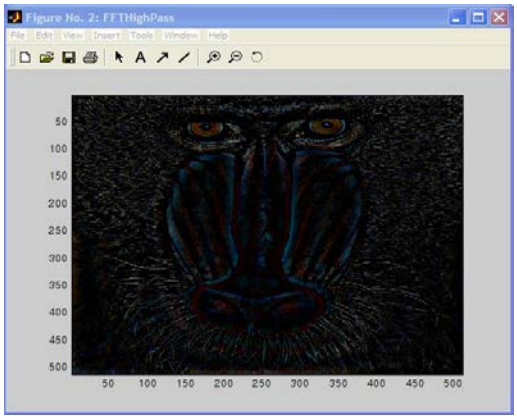
out_img1r – the output image values are the real FFT coefficients of the input image if they lie outside the radius, otherwise they are set to 0.

out_img1i – the output image values are the imaginary FFT coefficients of the input image if they lie outside the radius, otherwise they are set to 0.

Example:

```
img = openimage('baboon.jpg');
h=figure;image(img); set(h,'Name','Input Image');
[out_img1,out_img2]=vsg('FFT',img);
[out_img1r,out_img1i]=vsg('FFTHighPass',out_img1,out_img2,0.05);
[out_img3]=vsg('InvFFT',out_img1r,out_img1i);
h=figure; image(uint8(out_img3));set(h,'Name','FFTHighPass');
```





FFTBandPass

FFT band-pass filter.

function call:

[out_img1r,out_img1i]=vsg('FFTBandPass',in_img1r,in_img1i,rad1,rad2);

Arguments:

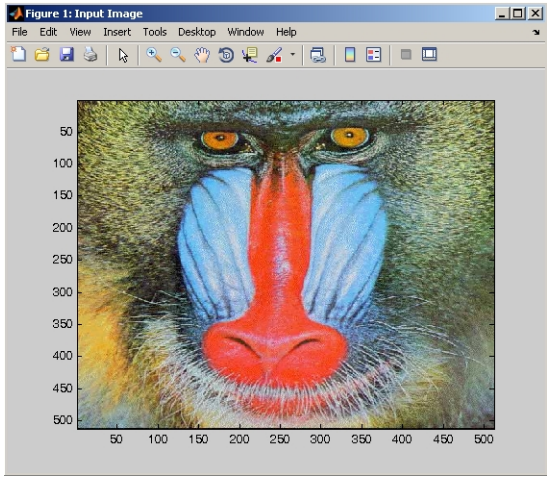
in_img1r - input image, 3 channel RGB, 1 channel greyscale or binary image of the real FFT coefficients.
in_img1i - input image, 3 channel RGB, 1 channel greyscale or binary image of the imaginary FFT coefficients.
rad1 – a double which specifies the smaller radius.
rad2 – a double which specifies the larger radius.

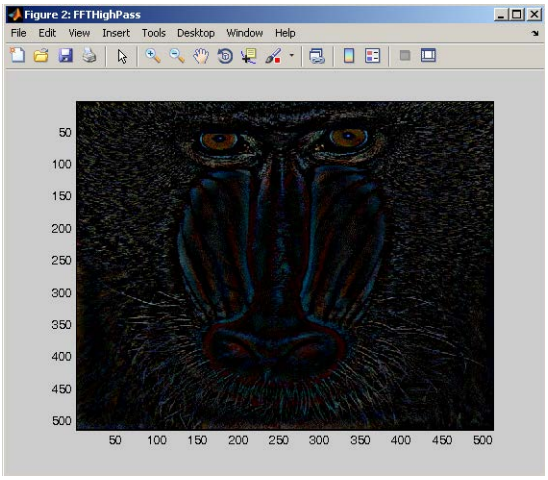
Description:

The user defined inputs *rad1* and *rad2* are the inner and outer bounds of the band-pass region. *out_img1r* – the output image values are the real FFT coefficients of the input image if they lie between radius1 and radius 2, otherwise they are set to 0.
out_img1i – the output image values are the imaginary FFT coefficients of the input image if they lie between radius1 and radius 2, otherwise they are set to 0.

Example:

```
img = openimage('baboon.jpg');
h=figure;image(img); set(h,'Name','Input Image');
[out_img1,out_img2]=vsg('FFT',img);
[out_img1r,out_img1i]=vsg('FFTBandPass',out_img1,out_img2,0.05, 0.5);
[out_img3]=vsg('InvFFT',out_img1r,out_img1i);
h=figure; image(uint8(out_img3));set(h,'Name','FFTBandPass');
```





FFTBandStop

FFT band-stop filter.

function call:

[out_img1r,out_img1i]=vsg('FFTBandStop',in_img1r,in_img1i,rad1,rad2);

Arguments:

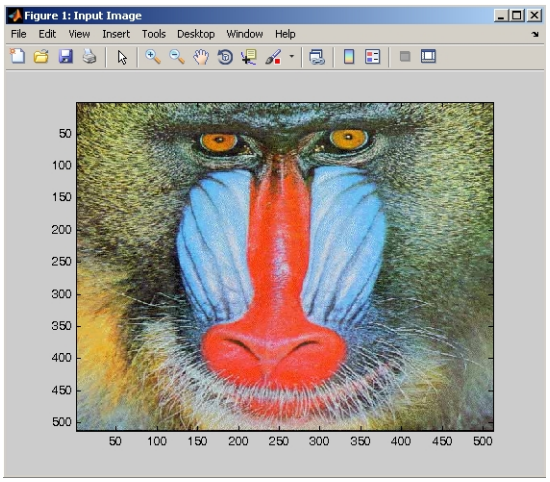
in_img1r - input image, 3 channel RGB, 1 channel greyscale or binary image of the real FFT coefficients.
in_img1i - input image, 3 channel RGB, 1 channel greyscale or binary image of the imaginary FFT coefficients.
rad1 – a double which specifies the smaller radius.
rad2 – a double which specifies the larger radius.

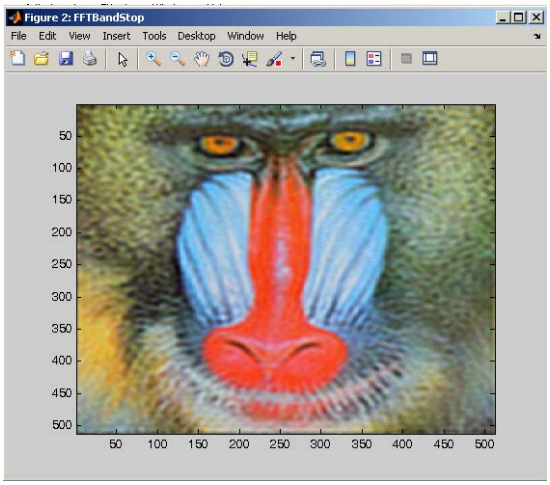
Description:

The user defined inputs *rad1* and *rad2* are the inner and outer bounds of the band-stop region.
out_img1r – the output image values are the real FFT coefficients of the input image if they lie inside radius1 and outside radius 2, otherwise they are set to 0.
out_img1i – the output image values are the imaginary FFT coefficients of the input image if they lie inside radius1 and outside radius 2, otherwise they are set to 0.

Example:

```
img = openimage('baboon.jpg');
h=figure;image(img);set(h,'Name','Input Image');
[out_img1,out_img2]=vsg('FFT',img);
[out_img1r,out_img1i]=vsg('FFTBandStop',out_img1,out_img2,0.2,1.2);
[out_img2]=vsg('InvFFT',out_img1r,out_img1i);
h=figure; image(uint8(out_img2));set(h,'Name','FFTBandStop');
```





FFTSelectivePass

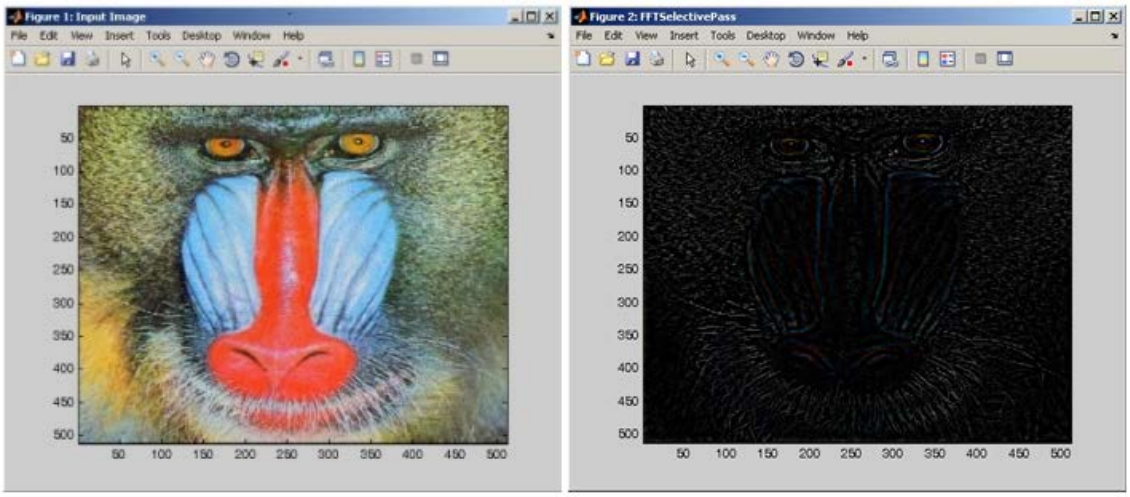
Location specific selective lowpass filter

function call:
[out_img1r,out_img1i]=vsg('FFTSelectivePass',in_img1r,in_img1i,rad1,offsetx,offsety);

Arguments:
in_img1r - input image, 3 channel RGB, 1 channel greyscale or binary image of the real FFT coefficients.
in_img1i - input image, 3 channel RGB, 1 channel greyscale or binary image of the imaginary FFT coefficients.
rad1 – a double which specifies the radius (0-1).
offsetx – an double specifying the x offset for values that may be passed through (0-1).
offsety – an double specifying the y offset for values that may be passed through (0-1).

Description:
The user defined input rad1 is the filters cutoff frequency. Offsetx and offsety are x and y coordinate offset values respectively.
out_img1r – the output image values are the real FFT coefficients of the input image if they lie outside the circle defined by radius1 and its x and y offset, otherwise set to 0.
out_img1i – the output image values are the imaginary FFT coefficients of the input image if they lie outside the circle defined by radius1 and its x and y offset, otherwise set to 0.

Example:
```
img = openimage('baboon.jpg');
h=figure;image(img);set(h,'Name','Input Image');
[out_img1,out_img2]=vsg('FFT',img);
[out_img1r,out_img1i]=vsg('FFTSelectivePass',out_img1,out_img2,0.10,0.01,0.01);
[out_img2]=vsg('InvFFT',out_img1r,out_img1i);
h=figure; image(uint8(out_img2));set(h,'Name','FFTSelectivePass');
```



Colour Space Functions

RGBToHSI

Extract the HSI colour planes.

function call:
[out_img1]=vsg('RGBToHSI',in_img1);

Arguments:
in_img1 - input image, 3 channel RGB, 1 channel greyscale or binary image.

Description:
Hue-Saturation-Intensity (HIS) colour representation from RGB colour plane.
out_img1 – an image with the same dimensions as the input image, which has been transformed into a HSI colour space. The first colour channel is the hue, second is saturation and third intensity.

Example:
```
img = imread('lake.jpg');
[out_img1]=vsg('RGBToHSI',img);
```

HSIToRGB

Create an image from individual HSI planes.

function call:
[out_img1]=vsg('HSIToRGB',in_img1);

Arguments:
in_img1 - input image, 3 channel RGB, 1 channel greyscale or binary image which is in a HSI colour space. The first colour channel should be hue, second is saturation and third intensity.

Description:
out_img1 – an image with the same dimensions as the input image which has been transformed back in to an RGB colour space.

Example:
```
img = imread('lake.jpg');
[out_img1]=vsg('RGBToHSI',img);
[out_img1]=vsg('HSIToRGB',out_img1);
```

Notes:
For single channel images, the single channel image is returned as is.

RGBToCMY

Extract the CMY (Cyan, Magenta, Yellow) colour planes.

function call:
[out_img1]=vsg('RGBToCMY',in_img1);

Arguments:
in_img1 - input image, 3 channel RGB, 1 channel greyscale or binary image.

Description:
out_img1 – an image with the same dimensions as the input image, which has been transformed into a CMY colour space. The first colour channel is the cyan, second is magenta and third yellow.

Example:
```
img = imread('lake.jpg');
[out_img1]=vsg('RGBToCMY',img);
```

Notes:
For single channel images, WHITE-in_img1 is returned.

CMYToRGB

Create an image from individual CMY (Cyan, Magenta, Yellow) planes.

function call:
[out_img1]=vsg('CMYToRGB',in_img1);

Arguments:
in_img1 - input image, 3 channel RGB, 1 channel greyscale or binary image which is in CMY colour space. The first colour channel should be cyan, second is magenta and third yellow.

Description:
out_img1 – an image with the same dimensions as the input image which has been transformed back in to an RGB colour space.

Example:
```
img = openimage('lake.jpg');
[out_img1]=vsg('RGBToCMY',img);
[out_img1]=vsg('CMYToRGB',out_img1);
```

Notes:
For single channel images, WHITE-in_img1 is returned.

RGBToLAB

Extract the Lab colour planes.

function call:
[out_img1]=vsg('RGBToLAB',in_img1);

Arguments:
in_img1 - input image, 3 channel RGB, 1 channel greyscale or binary image.

Description:
out_img1 – an image with the same dimensions as the input image, which has been transformed into a LAB colour space. The first colour channel is the luminance, second is “A” and third is the “B” colour channels.

Example:
```
img = openimage('lake.jpg');
[out_img1]=vsg('RGBToLAB',img);
```

Notes:
For single channel images, the single channel image is returned as is.

LABToRGB

Create an image from individual Lab planes.

function call:
[out_img1]=vsg('LABToRGB',in_img1);

Arguments:
in_img1 - input image, 3 channel RGB, 1 channel greyscale or binary image which is in a LAB colour space. The first colour channel is the luminance, second is “A” and third is the “B” colour channels.

Description:
out_img1 – an image with the same dimensions as the input image which has been transformed back in to an RGB colour space.

Example:
```
img = openimage('lake.jpg');
[out_img1]=vsg('RGBToLAB',img);
[out_img1]=vsg('LABToRGB',out_img1);
```

Notes:
For single channel images, the single channel image is returned as is.

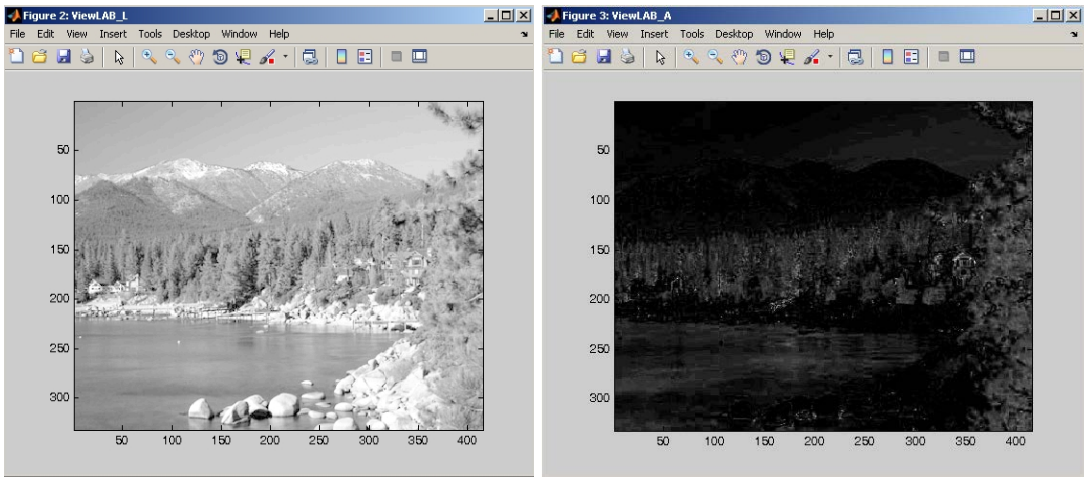
ViewLAB

function call:
[out_img1]=vsg('ViewLAB',in_img1);

Arguments:
in_img1 - input image, 3 channel RGB, 1 channel greyscale or binary image which is in a LAB colour space. The first colour channel is the luminance, second is “A” and third is the “B” colour channels.

Description:
out_img1 – an image with the same dimensions as the input image which is the input image with values normalised to the range of {0-255} on all three channels.

Example:
```
img = imread('lake.jpg');
[out_img1]=vsg('RGBToLAB',img);
[out_img1]=vsg('ViewLAB',out_img1);
h=figure;imagesc((out_img1(:,:,1)));colormap(gray);set(h,'Name','ViewLAB_L');
h=figure;imagesc((out_img1(:,:,2)));colormap(gray);set(h,'Name','ViewLAB_A');
```



Notes:
For single channel images, the single channel image is returned as is.

RGBToOpp

Extract the opponent process colour representation.

function call:
[out_img1]=vsg('RGBToOpp',in_img1);

Arguments:
in_img1 - input image, 3 channel RGB, 1 channel greyscale or binary image.

Description:
The Opponent Process Representation emphasis the difference between red and green, yellow and blue, and black and white colour components in an image. The *out_img1* – an image with the same dimensions as the input image, which has been transformed into the opponent colour space. The first colour channel is the red-green, second is blue-yellow and third is the white-black colour channels.

Example:
```
img = openimage('lake.jpg');
[out_img1]=vsg('RGBToOpp',img);
```

Notes:
For single channel images, the single channel image is returned as is.

OppToRGB

function call:
[out_img1]=vsg('OppToRGB',in_img1);

Arguments:
in_img1 - input image, 3 channel RGB, 1 channel greyscale or binary image which is in the opponent colour space. The first colour channel is the red-green, second is blue-yellow and third is the white-black colour channels.

Description:
out_img1 – an image with the same dimensions as the input image which has been transformed back in to an RGB colour space.

Example:
```
img = openimage('lake.jpg');
h=figure;image(img);axis image;
[out_img1]=vsg('RGBToOpp',img);
[out_img1]=vsg('OppToRGB',out_img1);
```

Notes:
For single channel images, the single channel image is returned as is.

ViewOpp

Normalize (0-255) opponent process colour channels. Used to view the normalized colour (unsaturated) channels.

function call:
[out_img1]=vsg('ViewOpp',in_img1);

Arguments:
in_img1 - input image, 3 channel RGB, 1 channel greyscale or binary image which is in a LAB colour space. The first colour channel is RG, second is BY and third is WB colour channels.

Description:
out_img1 – an image with the same dimensions as the input image which is the input image with values normalised to the range of {0-255} on all three channels.

Example:
```
img = openimage('lake.jpg');
[out_img1]=vsg('RGBToOpp',img);
[out_img1]=vsg('ViewOpp',out_img1);
```

Notes:
For single channel images, the single channel image is returned as is.

RGBToXYZ

Extract the XYZ colour planes.

function call:
[out_img1]=vsg('RGBToXYZ',in_img1);

Arguments:
in_img1 - input image, 3 channel RGB, 1 channel greyscale or binary image.

Description:
out_img1 – an image with the same dimensions as the input image, which has been transformed into an XYZ colour space. The first colour channel is the X component, second is the Y and third is the Z colour component.

Example:
```
img = openimage('lake.jpg');
[out_img1]=vsg('RGBToXYZ',img);
```

Notes:
For single channel images, the single channel image is returned as is.

XYZToRGB

Create an image from individual XYZ planes.

function call:
[out_img1]=vsg('XYZToRGB',in_img1);

Arguments:
in_img1 - input image, 3 channel RGB, 1 channel greyscale or binary image which is in an XYZ colour space. The first colour channel is the X component, second is the Y and third is the Z colour component.

Description:
out_img1 – an image with the same dimensions as the input image which has been transformed back in to an RGB colour space.

Example:
```
img = openimage('lake.jpg');
[out_img1]=vsg('RGBToXYZ',img);
[out_img1]=vsg('XYZToRGB',out_img1);
```

Notes:
For single channel images, the single channel image is returned as is.

ViewXYZ

Normalize (0-255) XYZ channels. Used to view the normalized colour (unsaturated) channels.

function call:
[out_img1]=vsg('ViewXYZ',in_img1);

Arguments:
in_img1 - input image, 3 channel RGB, 1 channel greyscale or binary image which is in an XYZ colour space. The first colour channel is the X component, second is the Y and third is the Z colour component.

Description:
The *out_img1* – an image with the same dimensions as the input image which is the input image with values normalised to the range of {0-255} on all three channels.

Example:
```
img = openimage('lake.jpg');
[out_img1]=vsg('RGBToXYZ',img);
[out_img1]=vsg('ViewXYZ',out_img1);
```

Notes:
For single channel images, the single channel image is returned as is.

RGBToYIQ

Extract the YIQ colour planes.

function call:
[out_img1]=vsg('RGBToYIQ',in_img1);
Arguments:
in_img1 - input image, 3 channel RGB, 1 channel greyscale or binary image.

Description:
out_img1 – an image with the same dimensions as the input image, which has been transformed into a YIQ colour space. The first colour channel is the Y component, second is the I component and third is the Q colour channel.

Example:
```
img = openimage('lake.jpg');
[out_img1]=vsg('RGBToYIQ',img);
```

Notes:
For single channel images, the single channel image is returned as is.

YIQToRGB
Create an image from individual YIQ planes.

function call:
[out_img1]=vsg('YIQToRGB',in_img1);

Arguments:
in_img1 - input image, 3 channel RGB, 1 channel greyscale or binary image which is in a YIQ colour space. The first colour channel is the Y component, second is the I component and third is the Q colour channel.

Description:
out_img1 – an image with the same dimensions as the input image which has been transformed back in to an RGB colour space.

Example:
```
img = openimage('lake.jpg');
[out_img1]=vsg('RGBToYIQ',img);
[out_img1]=vsg('YIQToRGB',out_img1);
```

Notes:
For single channel images, the single channel image is returned as is.

ViewYIQ
Normalize (0-255) YIQ channels. Used to view the normalized colour (unsaturated) channels.

function call:
[out_img1]=vsg('ViewYIQ',in_img1);

Arguments:
in_img1 - input image, 3 channel RGB, 1 channel greyscale or binary image which is in a YIQ colour space. The first colour channel is the Y component, second is the I component and third is the Q colour channel.

Description:

out_img1 – an image with the same dimensions as the input image which is the input image with values normalised to the range of {0-255} on all three channels.

Example:
```
img = imread('lake.jpg');
[out_img1]=vsg('RGBToYIQ',img);
[out_img1]=vsg('ViewYIQ',out_img1);
```

Notes:
For single channel images, the single channel image is returned as is.

RGBToYUV

Extract the YUV colour planes.

function call:
[out_img1]=vsg('RGBToYUV',in_img1);

Arguments:
in_img1 - input image, 3 channel RGB, 1 channel greyscale or binary image.

Description:
out_img1 – an image with the same dimensions as the input image, which has been transformed into a YIQ colour space. The first colour channel is the Y component, second is the I component and third is the Q colour channel.

Example:
```
img = imread('lake.jpg');
[out_img1]=vsg('RGBToYUV',img);
```

Notes:
For single channel images, the single channel image is returned as is.

YUVToRGB

Create an image from individual YUV planes.

function call:
[out_img1]=vsg('YUVToRGB',in_img1);

Arguments:
in_img1 - input image, 3 channel RGB, 1 channel greyscale or binary image which is in a YUV colour space. The first colour channel is the Y component, second is the U component and third is the V colour channel.

Description:
out_img1 – an image with the same dimensions as the input image which has been transformed back in to an RGB colour space.

Example:

Notes:
For single channel images, the single channel image is returned as is.

ViewYUV

Normalize (0-255) YUV channels. Used to view the normalized colour (unsaturated) channels.

function call:
[out_img1]=vsg('ViewYUV',in_img1);

Arguments:
in_img1 - input image, 3 channel RGB, 1 channel greyscale or binary image which is in a YIQ colour space. The first colour channel is the Y component, second is the U component and third is the V colour channel.

Description:
The *out_img1* – an image with the same dimensions as the input image which is the input image with values normalised to the range of {0-255} on all three channels.

Example:

```
img = openimage('lake.jpg');
[out_img1]=vsg('RGBToYUV',img);
[out_img1]=vsg('ViewYUV',out_img1);
```

Notes:
For single channel images, the single channel image is returned as is.

Colour Clustering

Cluster

Cluster a colour image (number of clusters are user defined) using the k-means algorithm.

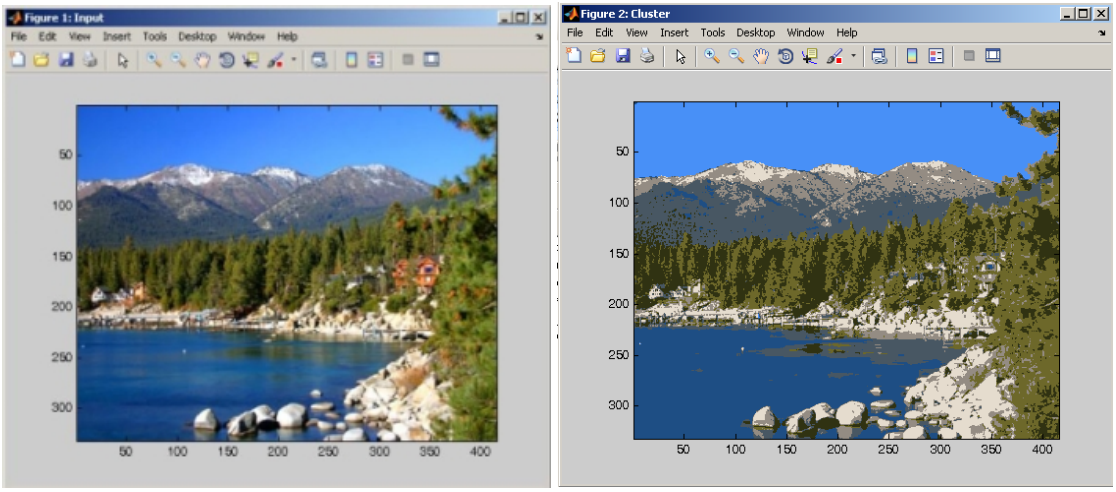
function call:
[out_img1]=vsg('Cluster',in_img1,num1);

Arguments:
in_img1 - input image, 3 channel RGB, 1 channel greyscale or binary image.
num1 – an integer specifying the number of colours to which the input image shall be reduced.

Description:
out_img1 – an image with the same dimensions as the input image which is the input image with a reduced number of colours. The colours are estimated such that the difference between the input and output images is minimised.

Example:

```
in_img1=imread('lake.jpg');
h=figure;image(in_img1);set(h,'Name','Input');
num1=7; % number of colour clusters
[out_img1]=vsg('Cluster',in_img1,num1);
h=figure; image(uint8(out_img1));set(h,'Name','Cluster');
```



Notes:
Clusters may merge and a lower number of output colours may be found.

SICluster

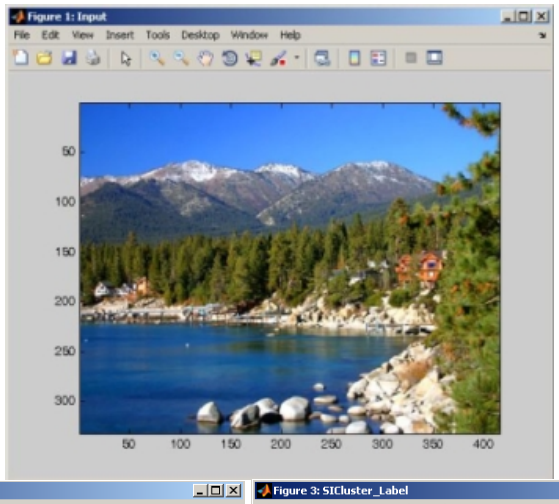
Unsupervised colour clustering using the k-means algorithm.

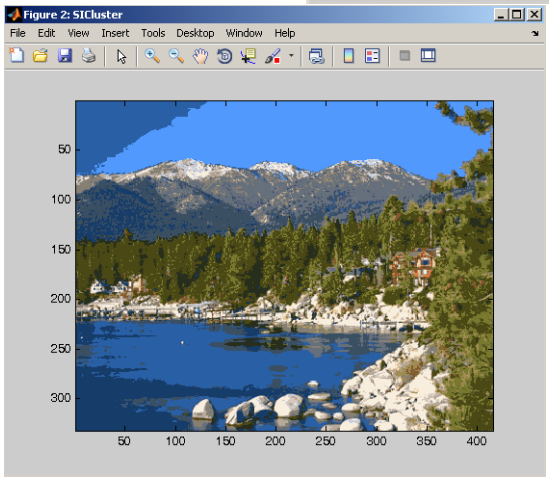
function call:
[out_img1,out_img2,num1]=vsg('SICluster',in_img1);

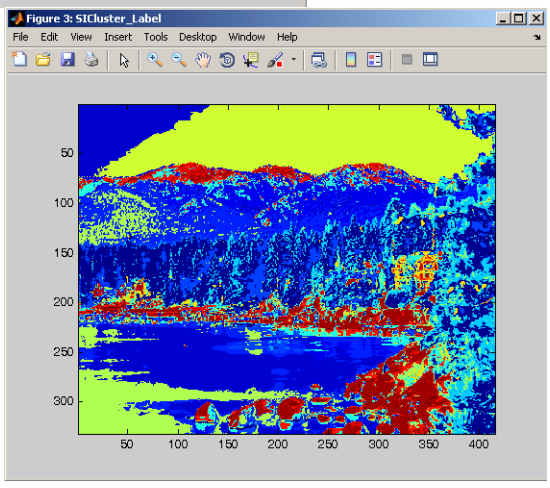
Arguments:
in_img1 - input image, 3 channel RGB, 1 channel greyscale or binary image.

Description:
The function SICluster applies the Unsupervised K-mean clustering for image segmentation.
out_img1 – an image with the same dimensions as the input image which is the input image with a reduced number of colours. The colours are estimated such that the difference between the input and output images is minimised.
out_img2 – an image with the same dimensions as the input image which is an image of the colour labels. i.e. each colour is numbered, and it is these numbers that are output in this image.
The *num* – an integer indicating the number of colour clusters found.

Example:
```
img = imread('lake.jpg');
h=figure;image(img);set(h,'Name','Input');
[out_img1,out_img2,num1]=vsg('SICluster',img);
h=figure; image(uint8(out_img1));set(h,'Name','SICluster');
h=figure; imagesc((out_img2(:,:,1)));set(h,'Name','SICluster_Label');
```







Notes:
This function starts off with 32 colours, all selected based on histogram analysis.

Segementation

KapurSeg

Applies Kapur segmentation to the input image.

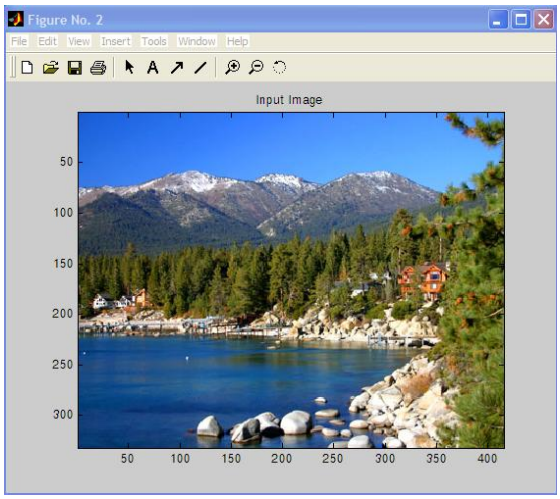
function call:
[out_img1]=vsg('KapurSeg',in_img1);

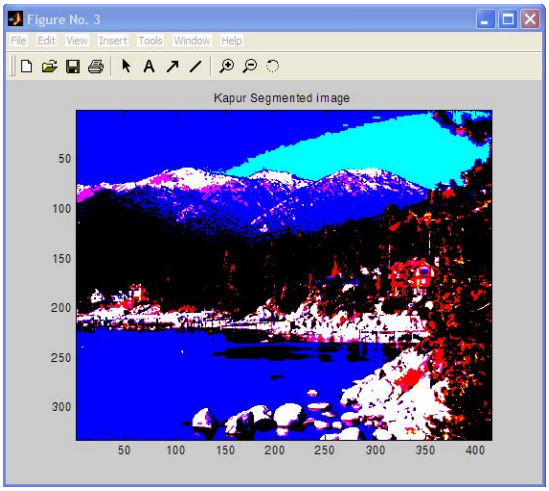
Arguments:
in_img1 - input image, 3 channel RGB, 1 channel greyscale or binary image.

Description:
out_img1 – Returns a binary image which is the input image after it has been thresholded. The threshold level is chosen from the input image using the Kapur algorithm.

Example:

```
in_img1 = openimage('lake.jpg');
[out_img1]=vsg('KapurSeg',in_img1);
figure,image(uint8(in_img1)),title('Input Image');
figure,image(uint8(out_img1)),title('Kapur Segmented image');
```





KittlerSeg

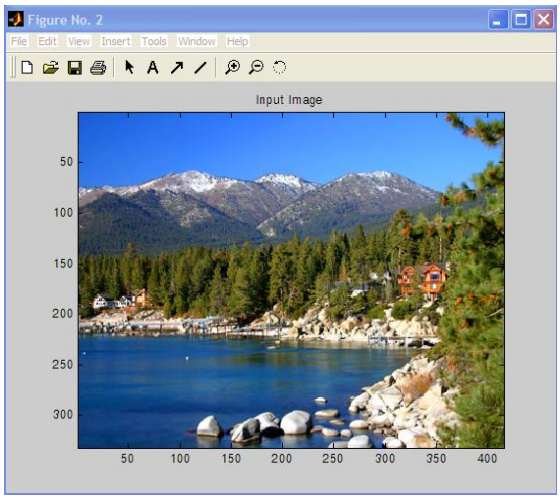
Applies Kittler segmentation to the input image.

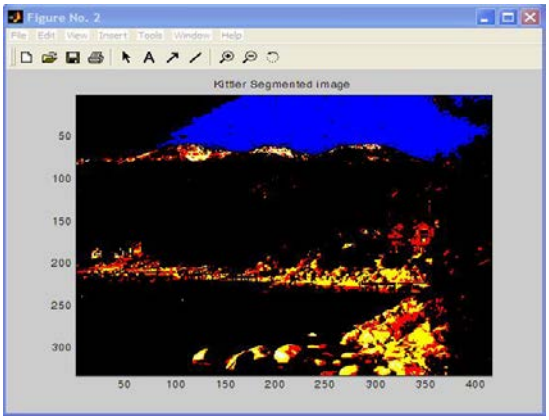
function call:
[out_img1]=vsg('KittlerSeg',in_img1);

Arguments:
in_img1 - input image, 3 channel RGB, 1 channel greyscale or binary image.

Description:
out_img1 – Returns a binary image which is the input image after it has been thresholded. The threshold level is chosen from the input image using the Kittler algorithm.

Example:
```
in_img1 = openimage('lake.jpg');
[out_img1]=vsg('KittlerSeg',in_img1);
figure,image(uint8(in_img1)),title('Input Image');
figure,image(uint8(out_img1)),title('Kittler Segmented image');
```





LloydSeg

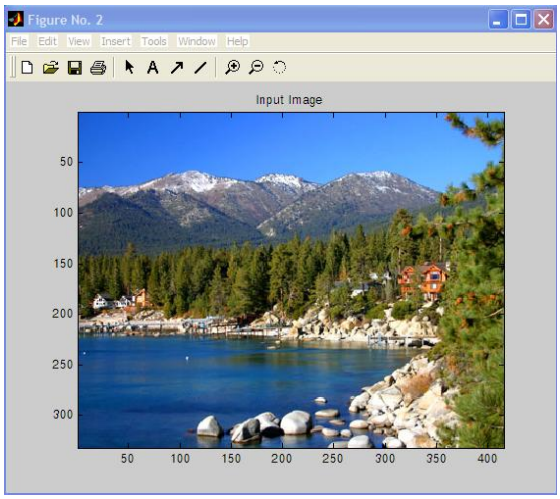
Applies Lloyd segmentation to the input image.

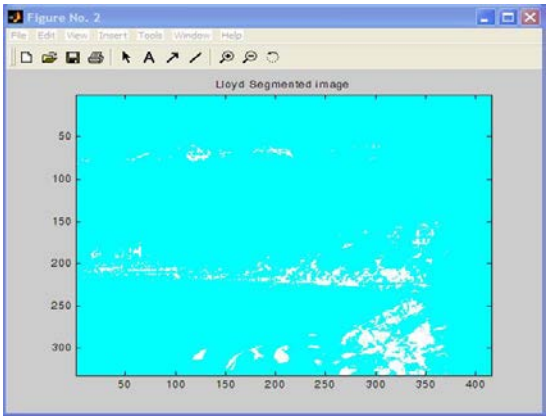
function call:
[out_img1]=vsg('LloydSeg',in_img1);

Arguments:
in_img1 - input image, 3 channel RGB, 1 channel greyscale or binary image.

Description:
out_img1 – Returns a binary image which is the input image after it has been thresholded. The threshold level is chosen from the input image using the Lloyd algorithm.

Example:
```
in_img1 = openimage('lake.jpg');
[out_img1]=vsg('LloydSeg',in_img1);
figure,image(uint8(in_img1)),title('Input Image');
figure,image(uint8(out_img1)),title('Lloyd Segmented image');
```





ModNiblackSeg

Applies ModNiblack segmentation to the input image.

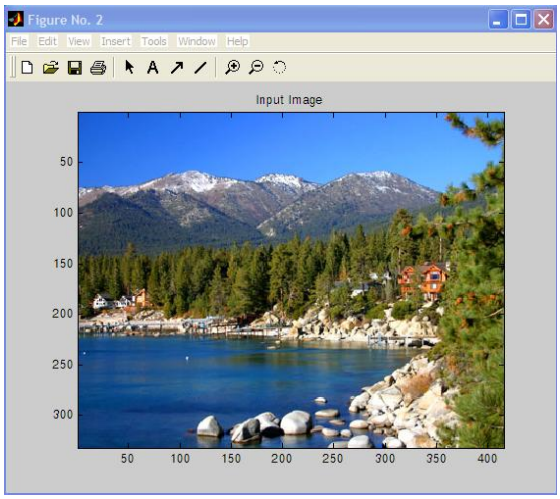
function call:
[out_img1]=vsg(' ModNiblackSeg',in_img1);

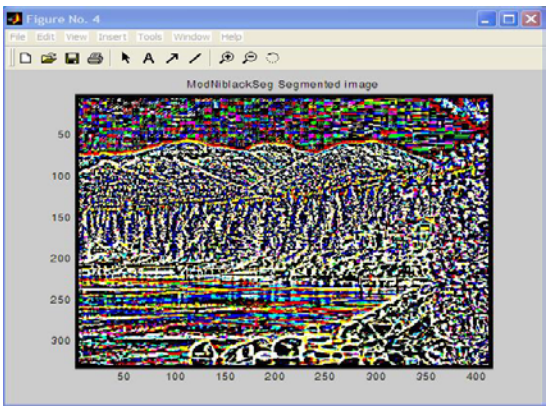
Arguments:
in_img1 - input image, 3 channel RGB, 1 channel greyscale or binary image.

Description:
out_img1 – Returns a binary image which is the input image after it has been thresholded. The threshold level is chosen from the input image using the Modified Niblack algorithm.

Example:

```
in_img1 = openimage('lake.jpg');
[out_img1]=vsg('ModNiblackSeg',in_img1);
figure,image(uint8(in_img1)),title('Input Image');
figure,image(uint8(out_img1)),title(' ModNiblack Segmented image');
```





OtsuSeg

Applies Otsu segmentation to the input image.

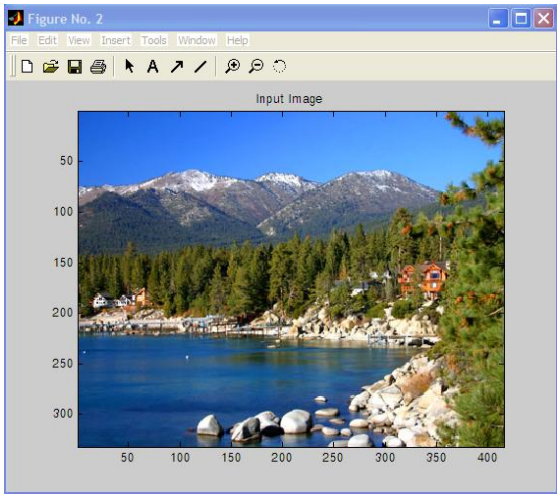
function call:
[out_img1]=vsg(' OtsuSeg',in_img1);

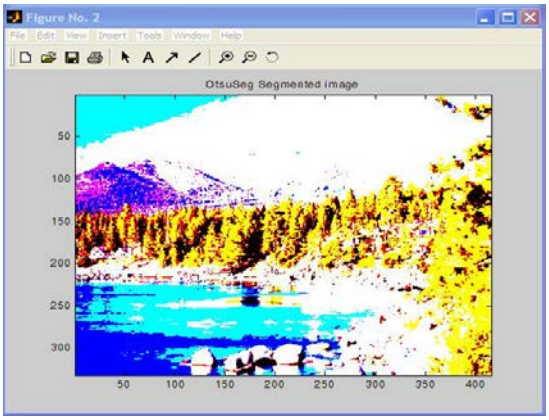
Arguments:
in_img1 - input image, 3 channel RGB, 1 channel greyscale or binary image.

Description:
out_img1 – Returns a binary image which is the input image after it has been thresholded. The threshold level is chosen from the input image using the Otsu algorithm.

Example:

```
in_img1 = openimage('lake.jpg');
[out_img1]=vsg('OtsuSeg',in_img1);
figure,image(uint8(in_img1)),title('Input Image');
figure,image(uint8(out_img1)),title(' Otsu Segmented image');
```





PalumboSeg

Applies Palumbo segmentation to the input image.

function call:
[out_img1]=vsg(' PalumboSeg',in_img1);

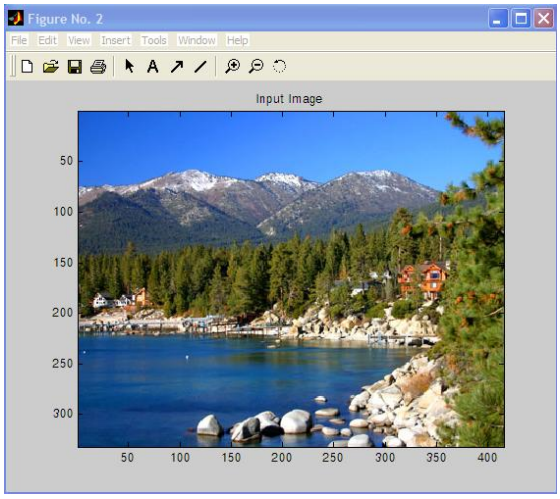
Arguments:
in_img1 - input image, 3 channel RGB, 1 channel greyscale or binary image.

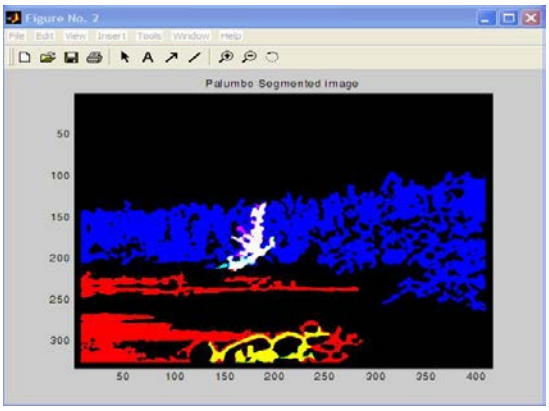
Description:
out_img1 – Returns a binary image which is the input image after it has been thresholded. The threshold level is chosen from the input image using the Palumbo algorithm.

Example:

```
in_img1 = openimage('lake.jpg');
[out_img1]=vsg('PalumboSeg',in_img1);
figure,image(uint8(in_img1)),title('Input Image');
```

```
figure,image(uint8(out_img1)),title('PalumboSeg Segmented image');
```





RiddlerSeg

Applies Riddler segmentation to the input image.

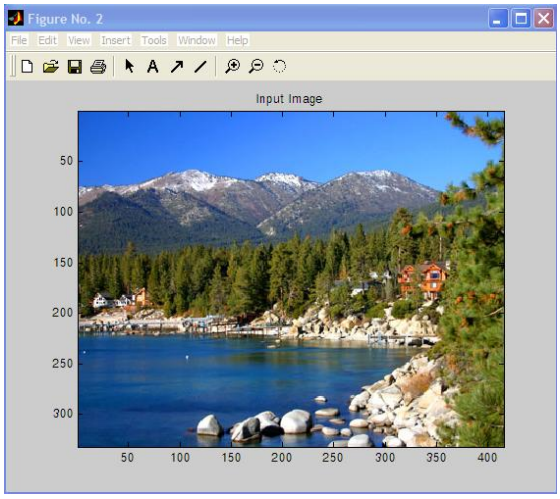
function call:
[out_img1]=vsg('RiddlerSeg',in_img1);

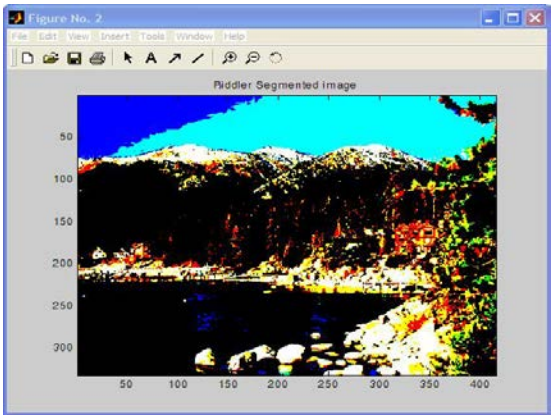
Arguments:
in_img1 - input image, 3 channel RGB, 1 channel greyscale or binary image.

Description:
out_img1 – Returns a binary image which is the input image after it has been thresholded. The threshold level is chosen from the input image using the Riddler algorithm.

Example:

```
in_img1 = openimage('lake.jpg');
[out_img1]=vsg('RiddlerSeg',in_img1);
figure,image(uint8(in_img1)),title('Input Image');
figure,image(uint8(out_img1)),title('Riddler Segmented image');
```





SauvolaSeg

Applies Sauvola segmentation to the input image.

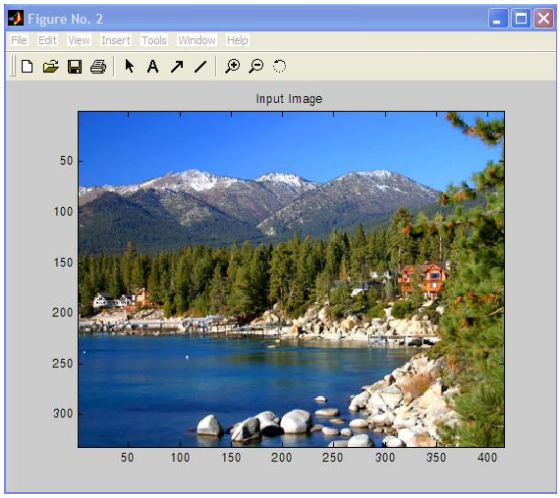
function call:
[out_img1]=vsg('SauvolaSeg',in_img1);

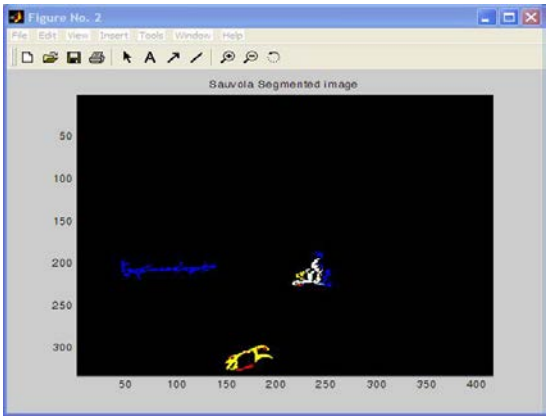
Arguments:
in_img1 - input image, 3 channel RGB, 1 channel greyscale or binary image.

Description:
out_img1 – Returns a binary image which is the input image after it has been thresholded. The threshold level is chosen from the input image using the Sauvola algorithm.

Example:

```
in_img1 = openimage('lake.jpg');
[out_img1]=vsg('SauvolaSeg',in_img1);
figure,image(uint8(in_img1)),title('Input Image');
figure,image(uint8(out_img1)),title(' Sauvola Segmented image');
```





TsaiSeg

Applies Tsai segmentation to the input image.

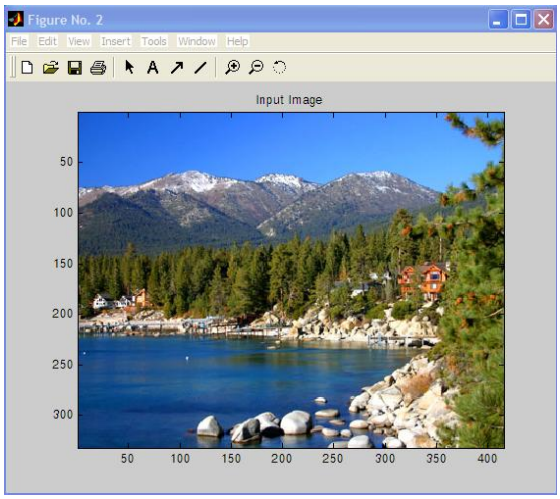
function call:
[out_img1]=vsg('TsaiSeg',in_img1);

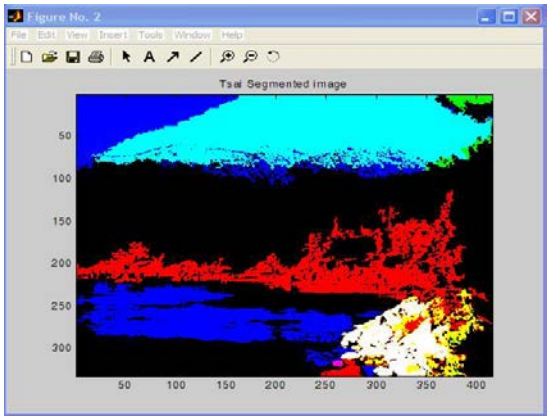
Arguments:
in_img1 - input image, 3 channel RGB, 1 channel greyscale or binary image.

Description:
out_img1 – Returns a binary image which is the input image after it has been thresholded. The threshold level is chosen from the input image using the KTsai algorithm.

Example:

```
in_img1 = openimage('lake.jpg');
[out_img1]=vsg('TsaiSeg',in_img1);
figure,image(uint8(in_img1)),title('Input Image');
figure,image(uint8(out_img1)),title(' Tsai Segmented image');
```





YanniSeg

Applies Yanni segmentation to the input image.

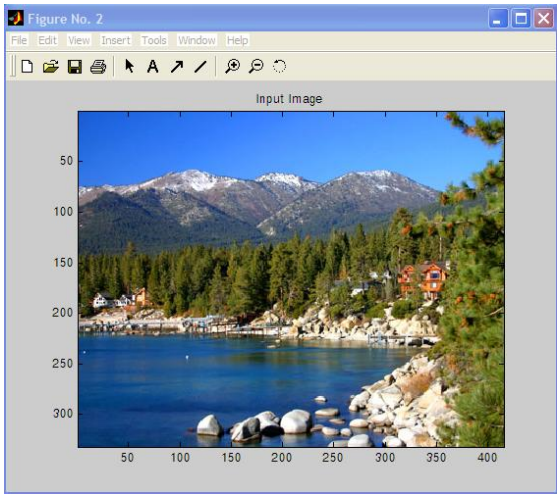
function call:
[out_img1]=vsg('YanniSeg',in_img1);

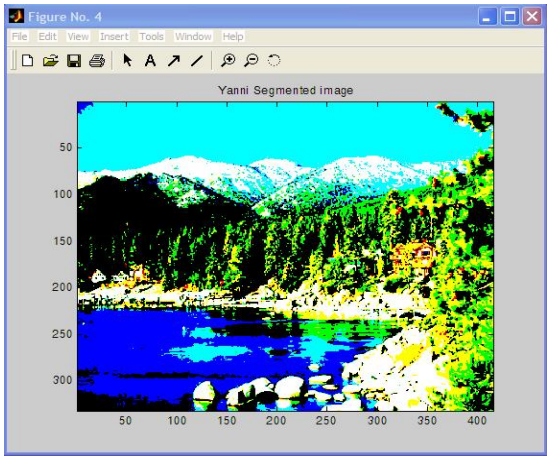
Arguments:
in_img1 - input image, 3 channel RGB, 1 channel greyscale or binary image.

Description:
out_img1 – Returns a binary image which is the input image after it has been thresholded. The threshold level is chosen from the input image using the Kapur algorithm.

Example:

```
in_img1 = openimage('lake.jpg');
[out_img1]=vsg('YanniSeg',in_img1);
figure,image(uint8(in_img1)),title('Input Image');
figure,image(uint8(out_img1)),title(' Yanni Segmented image');
```





DICOM Image Statistics functions

DICOMAvg

Compute the average intensity of the input dicom file.

function call:
[array1]=vsg('DICOMAvg',in_img1);

Arguments:
in_img1 - input image, a DICOM image.

Description:
array1 – one dimensional array of values. Each value is the mean greyscale value of a corresponding slice of the input dicom image.

Example:
```
[filename, chosenDirectory] = uigetfile('*.dcm', 'Pick an M-file');
filePath = [chosenDirectory filename]; %complete path
stackOfImages=vsg('DICOMRead',filePath); %4D array of Data
[array1]=vsg('DICOMAvg',stackOfImages);
array1(1)
array1(2)
```

Notes:
1. For RGB images, the function operates on all colour planes together and returns a single average per slice for all of the colour planes.
2. Requires XMedCon software (version xmedcon-0.10.5-1-win32).
3. DICOM data should be in a separate folder.

DICOMEnt

Compute the entropy of the input dicom file.

function call:
[array1]=vsg('DICOMEnt',in_img1);

Arguments:
in_img1 - input image, a DICOM image.

Description:
array1 – one dimensional array of values. Each value is the entropy greyscale value of a corresponding slice of the input dicom image.

Example:
```
[filename, chosenDirectory] = uigetfile('*.dcm', 'Pick an M-file');
filePath = [chosenDirectory filename]; %complete path
stackOfImages=vsg('DICOMRead',filePath);  %4D array of data
[array1]=vsg('DICOMCoEnt',stackOfImages);
array1(10)

ans = -9.6075e+008
```

Notes:
1. For RGB images, the function operates on all colour planes together and returns a single entropy per slice for all of the colour planes.
2. Requires XMedCon software (version xmedcon-0.10.5-1-win32).
3. DICOM data should be in a separate folder.

DICOMHi

Compute the highest grey level from the input dicom file.

function call:
[array1]=vsg('DICOMHi',in_img1);

Arguments:
in_img1 - input image, a DICOM image.

Description:
array1 – one dimensional array of values. Each value is the highest greyscale value of a corresponding slice of the input dicom image.

Example:
```
[filename, chosenDirectory] = uigetfile('*.dcm', 'Pick an M-file');
filePath = [chosenDirectory filename]; %complete path
stackOfImages=vsg('DICOMRead',filePath);  %4D array of data
[array1]=vsg('DICOMHi',stackOfImages);
array1(1)

ans = 3497
```

Notes:
1. For RGB images, the function operates on all colour planes together and returns a single highest value per slice for all of the colour planes.
2. Requires XMedCon software (version xmedcon-0.10.5-1-win32).
3. DICOM data should be in a separate folder.

DICOMKurt

Compute the kurtosis of the input dicom file.

function call:
[array1]=vsg('DICOMKurt',in_img1);

Arguments:
in_img1 - input image, a DICOM image.

Description:
array1 – one dimensional array of values. Each value is the excess kurtosis greyscale value of a corresponding slice of the input dicom image.

Example:
```
[filename, chosenDirectory] = uigetfile('*.dcm', 'Pick an M-file');
filePath = [chosenDirectory filename]; %complete path
stackOfImages=vsg('DICOMRead',filePath);  %4D array of data
[array1]= vsg('DICOMKurt',stackOfImages);
```

Notes:
1. For RGB images, the function operates on all colour planes together and returns a single kurtosis value per slice for all of the colour planes.
2. Requires XMedCon software (version xmedcon-0.10.5-1-win32).
3. DICOM data should be in a separate folder.

DICOMLow

Compute the lowest grey level from the input dicom file.

function call:
[array1]=vsg('DicomLow',in_img1);

Arguments:
in_img1 - input image, a DICOM image.

Description:
array1 – one dimensional array of values. Each value is the lowest greyscale value of a corresponding slice of the input dicom image.

Example:
```
[filename, chosenDirectory] = uigetfile('*.dcm', 'Pick an M-file');
filePath = [chosenDirectory filename]; %complete path
stackOfImages=vsg('DICOMRead',filePath);  %4D array of data
[array1]=vsg('DicomLow',stackOfImages);
```

Notes:
1. For RGB images, the function operates on all colour planes together and returns a single lowest value per slice for all of the colour planes.
2. Requires XMedCon software (version xmedcon-0.10.5-1-win32).
3. DICOM data should be in a separate folder.

DICOMMSE

Compare the input images using the mean square error operation.

function call:
[array1]=vsg('DicomMSE',in_img1,in_img2);

Arguments:
in_img1 - input image, a DICOM image.
in_img2 input image, a DICOM image.

Description:
array1 – one dimensional array of values. Each value is the greyscale mean squared error value of a pair of corresponding slices of the input dicom images.

Example:
```
[filename, chosenDirectory] = uigetfile('*.dcm', 'Pick an M-file');
filePath = [chosenDirectory filename]; %complete path
stackOfImages=vsg('DICOMRead',filePath);  %4D array of data
[array1]=vsg('DICOMMSE',stackOfImages(:,:,:,1),stackOfImages(:,:,:,2));
```

Notes:
1. For RGB images, the function operates on all colour planes together and returns a single MSE value per slice for all of the colour planes.
2. Requires XMedCon software (version xmedcon-0.10.5-1-win32).
3. DICOM data should be in a separate folder.

DICOMSkew

Compute the skewness deviation of the input dicom file.

function call:
[array1]=vsg('DICOMSkew',in_img1);

Arguments:
in_img1 - input image, a DICOM image.

Description:
array1 – one dimensional array of values. Each value is the greyscale skewness value of a corresponding slice of the input dicom image.

Example:
```
[filename, chosenDirectory] = uigetfile('*.dcm', 'Pick an M-file');
file = [chosenDirectory filename]; %complete path
stackOfImages=vsg('DICOMRead',file); %4D array of data
[array1]=vsg('DICOMSkew',stackOfImages);
```

Notes:
For RGB images, the function operates on all colour planes together and returns a single skewness value per slice for all of the colour planes.
2. Requires XMedCon software (version xmedcon-0.10.5-1-win32).
3. DICOM data should be in a separate folder.

DICOMSTD

Compute the standard deviation of the input dicom file.

function call:
[array1]=vsg('DICOMSTD',in_img1);

Arguments:
in_img1 - input image, a DICOM image.

Description:
array1 – one dimensional array of values. Each value is the greyscale standard deviation value of a corresponding slice of the input dicom image.

Example:
```
[filename, chosenDirectory] = uigetfile('*.dcm', 'Pick an M-file');
filePath = [chosenDirectory filename]; %complete path
stackOfImages=vsg('DICOMRead',filePath);  %4D array of data
[array1]=vsg('DICOMSTD',stackOfImages);
```

Notes:
1. For RGB images, the function operates on all colour planes together and returns a single standard deviation value per slice for all of the colour planes.

2. Requires XMedCon software (version xmedcon-0.10.5-1-win32).
3. DICOM data should be in a separate folder.

DICOMVar

Compute the variance of the input dicom file.

function call:
[array1]=vsg('DICOMVar',in_img1);

Arguments:
in_img1 - input image, a DICOM image.

Description:
array1 – one dimensional array of values. Each value is the greyscale variance value of a corresponding slice of the input dicom image.

Example:
```
[filename, chosenDirectory] = uigetfile('*.dcm', 'Pick an M-file');
filePath = [chosenDirectory filename]; %complete path
stackOfImages=vsg('DICOMRead',filePath);  %4D array of data
[array1]=vsg('DICOMVar',stackOfImages);
```

Notes:
1. For RGB images, the function operates on all colour planes together and returns a single variance per slice for all of the colour planes.
2. Requires XMedCon software (version xmedcon-0.10.5-1-win32).
3. DICOM data should be in a separate folder.

DICOMCoHomog

Compute the homogeneity of the co-occurrence matrix.

function call:
[array1]=vsg('DICOMCoHomog',in_img1);

Arguments:
in_img1 - input image, a DICOM CoOccurrence matrix image, generated with the CoOcMatGen function. Each slice of the input image is the cooccurrence matrix of the original input DICOM image to the CoOcMatGen function.

Description:
array1 – one dimensional array of values. Each value is the greyscale homogeneity value of a corresponding coocurrence matrix slice of the input dicom image. For Dicom images, data should be normalized between 0-255 for Co-Occurrence matrix generation.

Example:
```
[filename, chosenDirectory] = uigetfile('*.dcm', 'Pick an M-file');
filePath = [chosenDirectory filename]; %complete path
stackOfImages=vsg('DICOMRead',filePath);  %4D array of data
maxVal = max(max(max(max(stackOfImages))));
stackOfImages = (stackOfImages/maxVal)*255; %normalizing the data
[stackOfImages]=vsg('CoOcMatGen',stackOfImages);
[stackOfImages]=vsg('DICOMCoHomog',stackOfImages);
```

Notes:
1. For RGB images, the function operates on all colour planes together and returns a single homogeneity value per slice for all of the colour planes.
2. Requires XMedCon software (version xmedcon-0.10.5-1-win32).
3. DICOM data should be in a separate folder.

DICOMCoEnt

Compute the entropy of the co-occurrence matrix.

function call:
[array1]=vsg('DICOMCoEnt',in_img1);

Arguments:
in_img1 - input image, a DICOM CoOccurrence matrix image, generated with the CoOcMatGen function. Each slice of the input image is the cooccurrence matrix of the original input DICOM image to the CoOcMatGen function.

Description:
array1 – one dimensional array of values. Each value is the greyscale entropy value of a corresponding coocurrence matrix slice of the input dicom image. For Dicom images, data should be normalized between 0-255 for Co-Occurrence matrix generation.

Example:
```
[filename, chosenDirectory] = uigetfile('*.dcm', 'Pick an M-file');
filePath = [chosenDirectory filename]; %complete path
stackOfImages=vsg('DICOMRead',filePath);  %4D array of data
maxVal = max(max(max(max(stackOfImages))));
stackOfImages = (stackOfImages/maxVal)*255; %normalizing the data
[stackOfImages]=vsg('CoOcMatGen',stackOfImages);
[array1]=vsg('DICOMEnt',stackOfImages);
```

Notes:
1. For RGB images, the function operates on all colour planes together and returns a single entropy value per slice for all of the colour planes.
2. Requires XMedCon software (version xmedcon-0.10.5-1-win32).
3. DICOM data should be in a separate folder.

DICOMCoEng

Compute the energy of the co-occurrence matrix.

function call:
[array1]=vsg('DICOMCoEng',in_img1);

Arguments:
in_img1 - input image, a DICOM CoOccurrence matrix image, generated with the CoOcMatGen function. Each slice of the input image is the cooccurrence matrix of the original input DICOM image to the CoOcMatGen function.

Description:
array1 – one dimensional array of values. Each value is the greyscale energy value of a corresponding coocurrence matrix slice of the input dicom image. For Dicom images, data should be normalized between 0-255 for Co-Occurrence matrix generation.

Example:

```
[filename, chosenDirectory] = uigetfile('*.dcm', 'Pick an M-file');
filePath = [chosenDirectory filename]; %complete path
stackOfImages=vsg('DICOMRead',filePath);  %4D array of data
maxVal = max(max(max(max(stackOfImages))));
stackOfImages = (stackOfImages/maxVal)*255; %normalizing the data
[out_img1]=vsg('CoOcMatGen',stackOfImages);
[array1]=vsg('DICOMCoHomog',out_img1);
```

Notes:

1. For RGB images, the function operates on all colour planes together and returns a single energy value per slice for all of the colour planes.
2. Requires XMedCon software (version xmedcon-0.10.5-1-win32).
3. DICOM data should be in a separate folder.

DICOMCoContrast

Compute the contrast of the co-occurrence matrix.

function call:

[array1]=vsg('DICOMCoContrast',in_img1);

Arguments:

in_img1 - input image, a DICOM CoOccurrence matrix image, generated with the CoOcMatGen function. Each slice of the input image is the cooccurrence matrix of the original input DICOM image to the CoOcMatGen function.

Description:

array1 – one dimensional array of values. Each value is the greyscale contrast value of a corresponding coocurrence matrix slice of the input dicom image. For Dicom images, data should be normalized between 0-255 for Co-Occurrence matrix generation.

Example:

```
[filename, chosenDirectory] = uigetfile('*.dcm', 'Pick an M-file');
filePath = [chosenDirectory filename]; %complete path
stackOfImages=vsg('DICOMRead',filePath);  %4D array of data
maxVal = max(max(max(max(stackOfImages))));
stackOfImages = (stackOfImages/maxVal)*255; %normalizing the data
[out_img1]=vsg('CoOcMatGen',stackOfImages);
[array1]=vsg('DICOMContrast',out_img1);
```

Notes:

1. For RGB images, the function operates on all colour planes together and returns a single contrast value per slice for all of the colour planes.
2. Requires XMedCon software (version xmedcon-0.10.5-1-win32).
3. DICOM data should be in a separate folder.

Global 3D image operations

AIP

Average intensity projection.

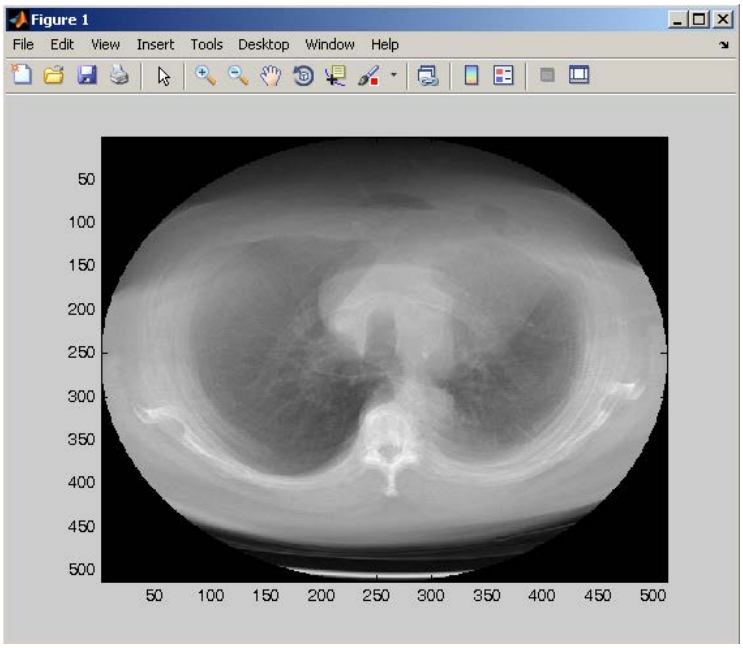
function call:
[out_img1]=vsg('AIP',in_img1);

Arguments:
in_img1 - input image, a DICOM image.

Description:
out_img1 – a 2D greyscale or colour image. Each value of the output image is calculated as the mean intensity value along the z dimension for each x and y location.

Example:

```
[filename, chosenDirectory] = uigetfile('*.dcm', 'Pick an M-file');
filePath = [chosenDirectory filename];
stackOfImages=vsg('DICOMRead',filePath);  %4D array of data
[out_img1]=vsg('AIP',stackOfImages);
```



Notes:
1. For RGB or Dicom images, the function operates on all colour planes or slices separately.
2. Dicom images require XMedCon software (version xmedcon-0.10.5-1-win32).
3. DICOM data must be in a separate folder.

MIP

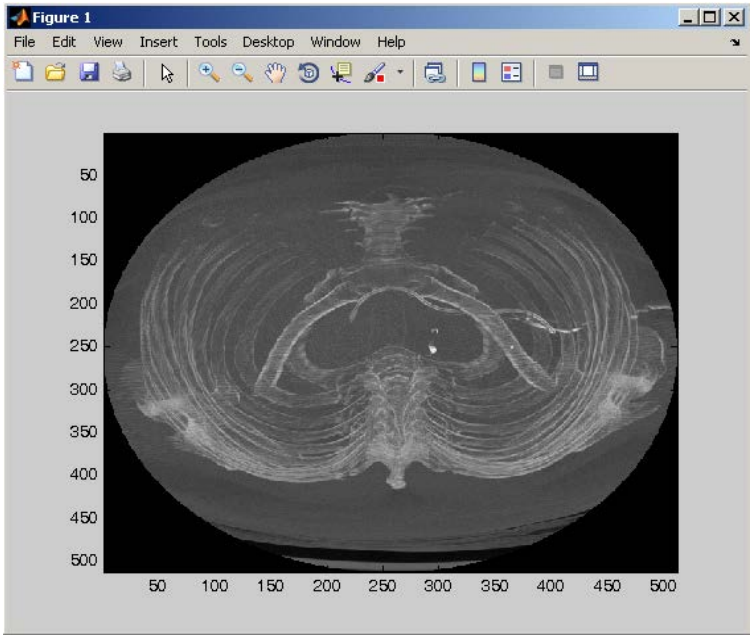
Maximum intensity projection.

function call:
[out_img1]=vsg('MIP',in_img1);
Arguments:
in_img1 - input image, a DICOM image.

Description:
out_img1 – a 2D greyscale or colour image. Each value of the output image is calculated as the maximum intensity value along the z dimension for each x and y location.

Example:

```
[filename, chosenDirectory] = uigetfile('*.dcm', 'Pick an M-file');
filePath = [chosenDirectory filename];
stackOfImages=vsg('DICOMRead',filePath);  %4D array of data
[out_img1]=vsg('MIP',stackOfImages);
```



Notes:
1. For RGB or Dicom images, the function operates on all colour planes or slices separately.
2. Dicom images require XMedCon software (version xmedcon-0.10.5-1-win32).
3. DICOM data must be in a separate folder.

Enhancer3D

Stretch the LUT in order to occupy the entire range between BLACK (0) and num1.

function call:
[out_img1]=vsg('Enhancer3D',in_img1,num1);

Arguments:
in_img1 - input image, a DICOM image.

num1 – an integer specifying the upper limit of the range, to which the input image values will be scaled.

Description:
out_img1 – a DICOM image. Each value is calculated so that the intensity values are scaled from 0 –num1 filling the available colour range.

Example:
```
[filename, chosenDirectory] = uigetfile('*.dcm', 'Pick an M-file');
filePath = [chosenDirectory filename];
stackOfImages=vsg('DICOMRead',filePath);  %4D array of data
[out_img1]=vsg('Enhancer3D',stackOfImages,200);
```

Notes:
1. For RGB or Dicom images, the function operates on all colour planes or slices separately.
2. Dicom images require XMedCon software (version xmedcon-0.10.5-1-win32).
3. DICOM data must be in a separate folder.

Window3D

function call:
[out_img1]=vsg('Window3D',in_img1,num1);

Arguments:
in_img1 - input image, a DICOM image.
num1 – an integer. Which specifies the window threshold.

Description:
out_img1 – a DICOM image. For each input value in the input image that is greater than the threshold, num1, the output value is set to white; otherwise the output value is set to the input value.

Example:
```
[filename, chosenDirectory] = uigetfile('*.dcm', 'Pick an M-file');
filePath = [chosenDirectory filename];
stackOfImages=vsg('DICOMRead',filePath);  %4D array of data
[out_img1]=vsg('Window3D',stackOfImages,200);
```

Notes:
1. For RGB or Dicom images, the function operates on all colour planes or slices separately.
2. Dicom images require XMedCon software (version xmedcon-0.10.5-1-win32).
3. DICOM data must be in a separate folder.

Conv3D

This operation requires coefficients to be specified in the form of a square, odd sized integer array.

function call:
[out_img1]=vsg('Conv3D',in_img1,conv_arr);

Arguments:
in_img1 - input image, a DICOM image.

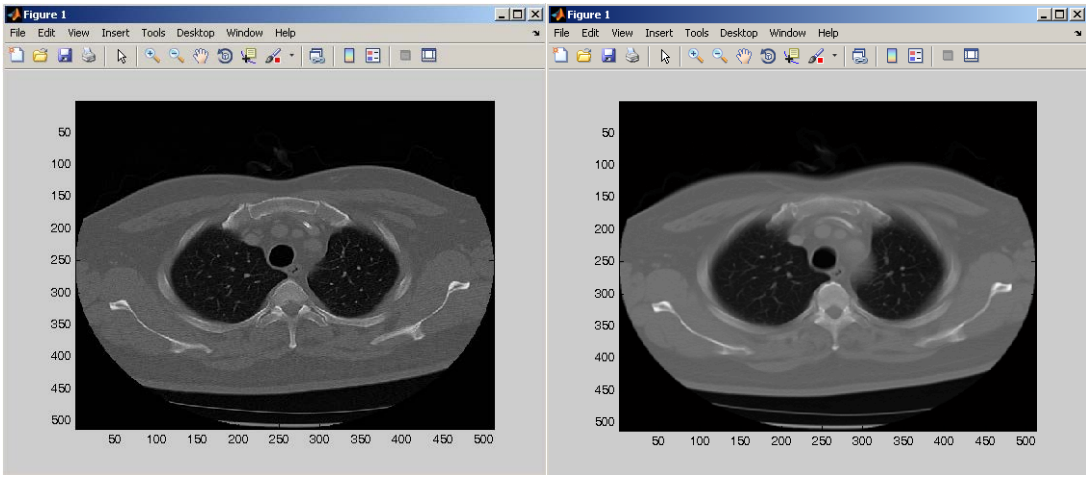
conv_arr – a 9x3 convolution kernel, which is made up of three 3x3's. Each of the 3x3's is one of the slice in the 3x3x3 cube, each one running in the x-y plane.

Description:

out_img1 – a DICOM image. Each value is calculated as the convolution of the input image with the convolution kernel. The values are normalised to the range 0-255.

Example:

```
[filename, chosenDirectory] = uigetfile('*.dcm', 'Pick an M-file');
filePath = [chosenDirectory filename];
stackOfImages=vsg('DICOMRead',filePath);  %4D array of data
arraySize = ones(9,3);
[out_img1]=vsg('Conv3D',stackOfImages,arraySize);
```



Notes:

1. For RGB or Dicom images, the function operates on all colour planes or slices separately.
2. Dicom images require XMedCon software (version xmedcon-0.10.5-1-win32).
3. DICOM data must be in a separate folder.

Volume Labelling

Labeller3D

Label unconnected shapes in a binary image (Range 0-255) by location.

function call:
[out_img1]=vsg('Labeller3D',in_img1);

Arguments:
in_img1 - input image, a DICOM image which is binary, i.e. with values 0 or 255;

Description:
out_img1 – a DICOM image. The input image is processed such that each connected region is given an individual label. The output image is an image of these labels;

Example:
```
[filename, chosenDirectory] = uigetfile('*.dcm', 'Pick an M-file');
filePath = [chosenDirectory filename];
stackOfImages=vsg('DICOMRead',filePath);  %4D array of data
[out_img1]=vsg('Threshold',stackOfImages,500);
[out_img2]=vsg('Labeller3D',out_img1);
```

Notes:
1. For RGB or Dicom images, the function operates on all colour planes or slices separately.
2. Dicom images require XMedCon software (version xmedcon-0.10.5-1-win32).
3. DICOM data must be in a separate folder.

LabelByVol

Label the unconnected shapes in a binary image in relation to their size.

function call:
[out_img1]=vsg('LabelByVol',in_img1);

Arguments:
in_img1 - input image, a DICOM image which is binary, i.e. with values 0 or 255;

Description:
out_img1 – a DICOM image. The input image is processed such that each connected region is given an individual label. These labels are then re-ordered such that the largest blob will have the lowest label and the smallest blob, the highest label. The output image is an image of these labels;

Example:
```
[filename, chosenDirectory] = uigetfile('*.dcm', 'Pick an M-file');
filePath = [chosenDirectory filename];
stackOfImages=vsg('DICOMRead',filePath);  %4D array of data
[out_img1]=vsg('Threshold',stackOfImages,500);
[out_img1]=vsg('LabelByVol',out_img1);
```

Notes:
1. For RGB or Dicom images, the function operates on all colour planes or slices separately.
2. Dicom images require XMedCon software (version xmedcon-0.10.5-1-win32).
3. DICOM data must be in a separate folder.

BigBlob3D

Extract the biggest white blob from a binary image.

function call:
[out_img1]=vsg('BigBlob3D',in_img1);

Arguments:
in_img1 - input image, a DICOM image which is binary, i.e. with values 0 or 255;

Description:
out_img1 – a DICOM image. The input image is processed such that each connected region is given an individual label. These labels are then re-ordered such that the largest blob will have the lowest label and the smallest blob, the highest label. The largest object is then thresholded and returned in the output image.

Example:
```
[filename, chosenDirectory] = uigetfile('*.dcm', 'Pick an M-file');
filePath = [chosenDirectory filename];
stackOfImages=vsg('DICOMRead',filePath);  %4D array of data
[out_img1]=vsg('Threshold',stackOfImages,500);
[out_img2]=vsg('BigBlob3D',out_img1);
```

Notes:
1. For RGB or Dicom images, the function operates on all colour planes or slices separately.
2. Dicom images require XMedCon software (version xmedcon-0.10.5-1-win32).
3. DICOM data must be in a separate folder.

SmallBlob3D

Extract the smallest white blob from a binary image.

function call:
[out_img1]=vsg('SmallBlob3D',in_img1);

Arguments:
in_img1 - input image, a DICOM image which is binary, i.e. with values 0 or 255;

Description:
out_img1 – a DICOM image. The input image is processed such that each connected region is given an individual label. These labels are then re-ordered such that the largest blob will have the lowest label and the smallest blob, the highest label. The smallest object is then thresholded and returned in the output image.

Example:
```
[filename, chosenDirectory] = uigetfile('*.dcm', 'Pick an M-file');
filePath = [chosenDirectory filename];
stackOfImages=vsg('DICOMRead',filePath);  %4D array of data
[out_img1]=vsg('Threshold',stackOfImages,500);
[out_img2]=vsg('SmallBlob3D',out_img1);
```

Notes:
1. For RGB or Dicom images, the function operates on all colour planes or slices separately.
2. Dicom images require XMedCon software (version xmedcon-0.10.5-1-win32).
3. DICOM data must be in a separate folder.

3D Filtering

DOLPS3D

DOLPS – Difference of low pass 3x3x3 filters.

function call:
[out_img1]=vsg('DOLPS3D',in_img1);

Arguments:
in_img1 - input image, a DICOM image.

Return values
out_img1 – a DICOM image. Each value of the DICOM image is calculated as the difference between the input image low pass filtered 6 times and the input image lowpass filtered 3 times. Each low pass filtering is carried out with a 3x3x3 three dimensional kernel.

Example:
```
[filename, chosenDirectory] = uigetfile('*.dcm', 'Pick an M-file');
filePath = [chosenDirectory filename]; %complete path
stackOfImages=vsg('DICOMRead',filePath);  %4D array of data
[out_img1]=vsg('DOLPS3D',stackOfImages);
```

Notes:
1. For RGB images, the function operates on all colour planes together and returns a single average per slice for all of the colour planes.
2. Requires XMedCon software (version xmedcon-0.10.5-1-win32).
3. DICOM data should be in a separate folder.

LIN3D

Replace the central pixel of the 3x3x3 mask with the maximum value.

function call:
[out_img1]=vsg('LIN3D',in_img1);

Arguments:
in_img1 - input image, a DICOM image.

Description:
out_img1 – a DICOM image. Each value of the DICOM image is calculated as the largest intensity value in a 3x3x3 three-dimensional region of the input image low pass filtered with the kernel centred at the location in question.

Example:
```
[filename, chosenDirectory] = uigetfile('*.dcm', 'Pick an M-file');
filePath = [chosenDirectory filename]; %complete path
stackOfImages=vsg('DICOMRead',filePath);  %4D array of data
[out_img1]=vsg('LIN3D',stackOfImages);
```

Notes:
1. For RGB images, the function operates on all colour planes together and returns a single average per slice for all of the colour planes.
2. Requires XMedCon software (version xmedcon-0.10.5-1-win32).

3. DICOM data should be in a separate folder.

LowPass3D

Low pass 3x3x3 filter.

function call:
[out_img1]=vsg('LowPass3D',in_img1);

Arguments:
in_img1 - input image, a DICOM image.

Description:
out_img1 – a DICOM image. Each value of the DICOM image is calculated as the mean intensity value in a 3x3x3 three-dimensional region of the input image.

Example:
```
[filename, chosenDirectory] = uigetfile('*.dcm', 'Pick an M-file');
filePath = [chosenDirectory filename]; %complete path
stackOfImages=vsg('DICOMRead',filePath);  %4D array of data
[out_img1]=vsg('LowPass3D',stackOfImages);
```

Notes:
1. For RGB images, the function operates on all colour planes together and returns a single average per slice for all of the colour planes.
2. Requires XMedCon software (version xmedcon-0.10.5-1-win32).
3. DICOM data should be in a separate folder.

Midpoint3D

Midpoint 3x3x3 filter.

function call:
[out_img1]=vsg('Midpoint3D',in_img1);

Arguments:
in_img1 - input image, a DICOM image.

Description:
out_img1 – a DICOM image. Each value of the DICOM image is calculated as the midpoint between the highest and lowest intensity values in a 3x3x3 three-dimensional region of the input image.

Example:
```
[filename, chosenDirectory] = uigetfile('*.dcm', 'Pick an M-file');
filePath = [chosenDirectory filename]; %complete path
stackOfImages=vsg('DICOMRead',filePath);  %4D array of data
[out_img1]=vsg('Midpoint3D',stackOfImages);
```

Notes:
1. For RGB images, the function operates on all colour planes together and returns a single average per slice for all of the colour planes.
2. Requires XMedCon software (version xmedcon-0.10.5-1-win32).
3. DICOM data should be in a separate folder.

Sharpen3D

High pass 3x3x3 filter.

function call:
[out_img1]=vsg('Sharpen3D',in_img1);
Arguments:
in_img1 - input image, a DICOM image.

Description:
out_img1 – a DICOM image. Each value of the DICOM image is calculated as the difference between a weighted input image and the input image low pass filtered with the a 3x3x3 three-dimensional kernel centred at the location in question.

Example:
```
[filename, chosenDirectory] = uigetfile('*.dcm', 'Pick an M-file');
filePath = [chosenDirectory filename]; %complete path
stackOfImages=vsg('DICOMRead',filePath);  %4D array of data
[out_img1]=vsg('Sharpen3D',stackOfImages);
```

Notes:
1. For RGB images, the function operates on all colour planes together and returns a single average per slice for all of the colour planes.
2. Requires XMedCon software (version xmedcon-0.10.5-1-win32).
3. DICOM data should be in a separate folder.

SIN3D

Replace the central pixel of the 3x3x3 mask with the minimum value.

function call:
[out_img1]=vsg('SIN3D',in_img1);

Arguments:
in_img1 - input image, a DICOM image.

Description:
out_img1 – a DICOM image. Each value of the DICOM image is calculated as the smallest intensity value in a 3x3x3 three-dimensional region of the input with the kernel centred at the location in question.

Example:
```
[filename, chosenDirectory] = uigetfile('*.dcm', 'Pick an M-file');
filePath = [chosenDirectory filename]; %complete path
stackOfImages=vsg('DICOMRead',filePath);  %4D array of data
[out_img1]=vsg('SIN3D',stackOfImages);
```

Notes:
1. For RGB images, the function operates on all colour planes together and returns a single average per slice for all of the colour planes.
2. Requires XMedCon software (version xmedcon-0.10.5-1-win32).
3. DICOM data should be in a separate folder.

FreiChen3D

3D FreiChen edge detector.

function call:
[out_img1]=vsg('FreiChen3D',in_img1);

Arguments:
in_img1 - input image, a DICOM image.
Return values
out_img1 – a DICOM image. Each value of the DICOM image is the edge magnitude value calculated using a three-dimensional FreiChen kernel calculated in a 3x3x3 region of the input image.

Example:
```
[filename, chosenDirectory] = uigetfile('*.dcm', 'Pick an M-file');
filePath = [chosenDirectory filename]; %complete path
stackOfImages=vsg('DICOMRead',filePath);  %4D array of data
[out_img1]=vsg('FreiChen3D',stackOfImages);
```

Notes:
1. For RGB images, the function operates on all colour planes together and returns a single average per slice for all of the colour planes.
2. Requires XMedCon software (version xmedcon-0.10.5-1-win32).
3. DICOM data should be in a separate folder.

Laplace3D

3D Laplacian edge detector.

function call:
[out_img1]=vsg('Laplace3D',in_img1);

Arguments:
in_img1 - input image, a DICOM image.

Description:
out_img1 – a DICOM image. Each value of the DICOM image is the edge magnitude value calculated using a three-dimensional Laplace kernel calculated in a 3x3x3 region of the input image.

Example:
```
[filename, chosenDirectory] = uigetfile('*.dcm', 'Pick an M-file');
filePath = [chosenDirectory filename]; %complete path
stackOfImages=vsg('DICOMRead',filePath);  %4D array of data
[out_img1]=vsg('Laplace3D',stackOfImages);
```

Notes:
1. For RGB images, the function operates on all colour planes together and returns a single average per slice for all of the colour planes.
2. Requires XMedCon software (version xmedcon-0.10.5-1-win32).
3. DICOM data should be in a separate folder.

NonMax3D

3D Edge detection using non maxima suppression.

function call:
[out_img1]=vsg('NonMax3D',in_img1);

Arguments:
in_img1 - input image, a DICOM image.

Description:
out_img1 – a DICOM image. Each value of the DICOM image is the edge magnitude value calculated using a three-dimensional NonMaxima kernel calculated in a 3x3x3 region of the input image.

Example:
```
[filename, chosenDirectory] = uigetfile('*.dcm', 'Pick an M-file');
filePath = [chosenDirectory filename]; %complete path
stackOfImages=vsg('DICOMRead',filePath);  %4D array of data
[out_img1]=vsg('NonMax3D',stackOfImages);
```

Notes:
1. For RGB images, the function operates on all colour planes together and returns a single average per slice for all of the colour planes.
2. Requires XMedCon software (version xmedcon-0.10.5-1-win32).
3. DICOM data should be in a separate folder.

Prewitt3D

3D Prewitt edge detector.

function call:
[out_img1]=vsg('Prewitt3D',in_img1);

Arguments:
in_img1 - input image, a DICOM image.

Description:
out_img1 – a DICOM image. Each value of the DICOM image is the edge magnitude value calculated using a three-dimensional Prewitt kernel calculated in a 3x3x3 region of the input image.

Example:
```
[filename, chosenDirectory] = uigetfile('*.dcm', 'Pick an M-file');
filePath = [chosenDirectory filename]; %complete path
stackOfImages=vsg('DICOMRead',filePath);  %4D array of data
[out_img1]=vsg('Prewitt3D',stackOfImages);
```

Notes:
1. For RGB images, the function operates on all colour planes together and returns a single average per slice for all of the colour planes.
2. Requires XMedCon software (version xmedcon-0.10.5-1-win32).
3. DICOM data should be in a separate folder.

Roberts3D

3D Roberts edge detector.

function call:
[out_img1]=vsg('Roberts3D',in_img1);

Arguments:
in_img1 - input image, a DICOM image.

Decription:
out_img1 – a DICOM image. Each value of the DICOM image is the edge magnitude value calculated using a three-dimensional Roberts kernel calculated in a 2x2x2 region of the input image.

Example:

```
[filename, chosenDirectory] = uigetfile('*.dcm', 'Pick an M-file');
filePath = [chosenDirectory filename]; %complete path
stackOfImages=vsg('DICOMRead',filePath);  %4D array of data
[out_img1]=vsg('Roberts3D',stackOfImages);
```

Notes:
1. For RGB images, the function operates on all colour planes together and returns a single average per slice for all of the colour planes.
2. Requires XMedCon software (version xmedcon-0.10.5-1-win32).
3. DICOM data should be in a separate folder.

Sobel3D

3D Sobel edge detector.

function call:
[out_img1]=vsg('Sobel3D',in_img1);

Arguments:
in_img1 - input image, a DICOM image.

Description:
out_img1 – a DICOM image. Each value of the DICOM image is the edge magnitude value calculated using a three-dimensional Sobel kernel calculated in a 3x3x3 region of the input image.

Example:

```
[filename, chosenDirectory] = uigetfile('*.dcm', 'Pick an M-file');
filePath = [chosenDirectory filename]; %complete path
stackOfImages=vsg('DICOMRead',filePath);  %4D array of data
[out_img1]=vsg('Sobel3D',stackOfImages);
```

Notes:
1. For RGB images, the function operates on all colour planes together and returns a single average per slice for all of the colour planes.
2. Requires XMedCon software (version xmedcon-0.10.5-1-win32).
3. DICOM data should be in a separate folder.

Gauss3D

Gaussian filters the input image across all dimensions.

function call:
[out_img1]=vsg('Gauss3D',in_img1,std);

Arguments:
in_img1 - input image, a DICOM image.
std - a double, which specifies the standard deviation of the Gaussian distribution

Description:
out_img1 – a DICOM image. Each value of the DICOM image is the magnitude value calculated using a three-dimensional Gauss filtering kernel calculated in a 3x3x3 region of the input image.

Example:
```
[filename, chosenDirectory] = uigetfile('*.dcm', 'Pick an M-file');
filePath = [chosenDirectory filename]; %complete path
stackOfImages=vsg('DICOMRead',filePath);  %4D array of data
[out_img1]=vsg('Gauss3D',stackOfImages, 1.0);
```

Notes:
1. For RGB images, the function operates on all colour planes together and returns a single average per slice for all of the colour planes.
2. Requires XMedCon software (version xmedcon-0.10.5-1-win32).
3. DICOM data should be in a separate folder.

Median3D

3D median filter

function call:
[out_img1]=vsg('Median3D',in_img1);

Arguments:
in_img1 - input image, a DICOM image.

Description:
out_img1 – a DICOM image. Each value of the DICOM image is calculated as the median intensity value in a 3x3x3 three-dimensional region of the input image centred at the location in question.

Example:
```
[filename, chosenDirectory] = uigetfile('*.dcm', 'Pick an M-file');
filePath = [chosenDirectory filename]; %complete path
stackOfImages=vsg('DICOMRead',filePath);  %4D array of data
[out_img1]=vsg('Median3D',stackOfImages);
```

Notes:
1. For RGB images, the function operates on all colour planes together and returns a single average per slice for all of the colour planes.
2. Requires XMedCon software (version xmedcon-0.10.5-1-win32).
3. DICOM data should be in a separate folder.

Mask3D

Mask the image over 3 dimensions.

function call:
[out_img1]=vsg('Mask3D',in_img1,border);

Arguments:
in_img1 - input image, a DICOM image.
border – an integer specifying the size of the border which shall be placed on the image.

Description:
out_img1 – a DICOM image. Each value of the DICOM image is the same as the input image, except for the region around the edges in the x, y, and z dimensions where a black border of width border is drawn. The border being applied to the z dimension is equivalent to deleting that number of slices from the beginning and end of the image stack i.e. in the example provided, 20 slices will be masked out in the image stack.

Example:

```
[filename, chosenDirectory] = uigetfile('*.dcm', 'Pick an M-file');
filePath = [chosenDirectory filename]; %complete path
stackOfImages=vsg('DICOMRead',filePath);  %4D array of data
[out_img1]=vsg('Mask3D',stackOfImages,10);
```

Notes:
1. For RGB images, the function operates on all colour planes together and returns a single average per slice for all of the colour planes.
2. Requires XMedCon software (version xmedcon-0.10.5-1-win32).
3. DICOM data should be in a separate folder.

3D Morphology

Dilate3D

Dilation operation in 3D.

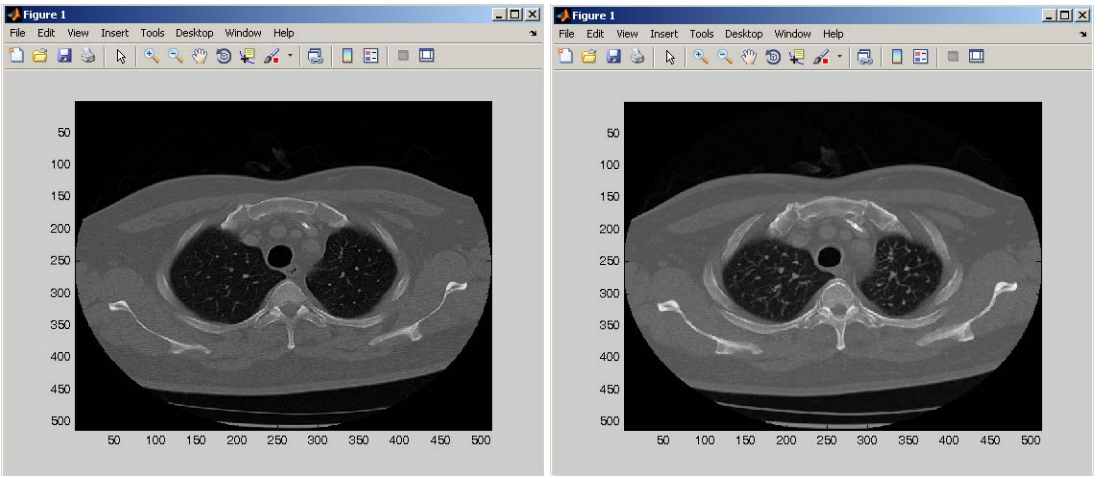
function call:
[out_img1]=vsg('Dilate3D',in_img1,connected);

Arguments:
in_img1 - input image, a DICOM image.
connected – an integer specifying whether 6 or 26 connected is used.

Description:
out_img1 – a DICOM image. Each value of the DICOM image is set as the maximum value in a three-dimensional kernel calculated in a 3x3x3 region of the input image.

Example:

```
[filename, chosenDirectory] = uigetfile('*.dcm', 'Pick an M-file');
filePath = [chosenDirectory filename];
stackOfImages=vsg('DICOMRead',filePath);  %4D array of data
[out_img1]=vsg('Dilate3D',stackOfImages,6);
```



Notes:
1. For RGB or Dicom images, the function operates on all colour planes or slices separately.
2. Dicom images require XMedCon software (version xmedcon-0.10.5-1-win32).
3. DICOM data must be in a separate folder.

Erode3D

Erosion operation in 3D.

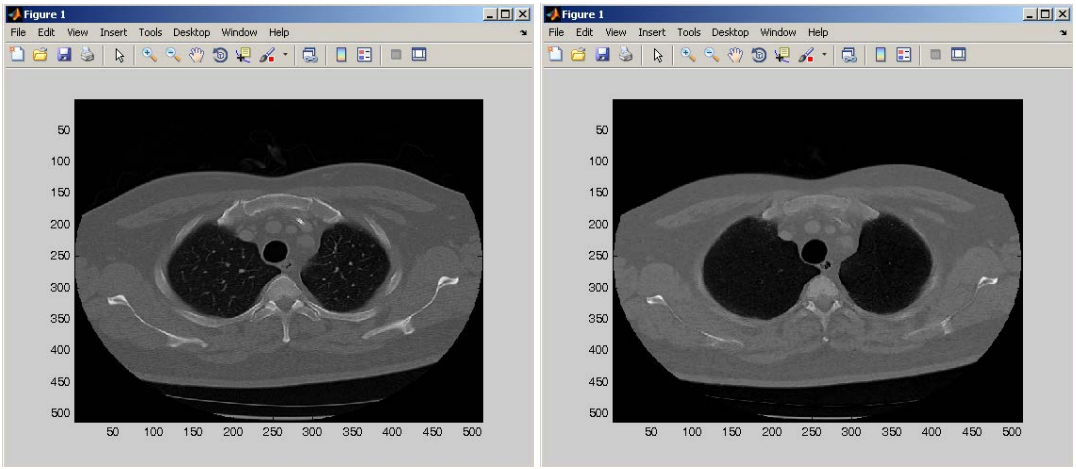
function call:
[out_img1]=vsg('Erode3D',in_img1,connected);

Arguments:
in_img1 - input image, a DICOM image.

connected – an integer specifying whether 6 or 26 connected is used

Description:
out_img1 – a DICOM image. Each value of the DICOM image is set as the minimum value in a three-dimensional kernel calculated in a 3x3x3 region of the input image.

Example:
```
[filename, chosenDirectory] = uigetfile('*.dcm', 'Pick an M-file');
filePath = [chosenDirectory filename];
stackOfImages=vsg('DICOMRead',filePath);  %4D array of data
[out_img1]=vsg('Erode3D',stackOfImages,6);
```



Notes:
1. For RGB or Dicom images, the function operates on all colour planes or slices separately.
2. Dicom images require XMedCon software (version xmedcon-0.10.5-1-win32).
3. DICOM data must be in a separate folder.

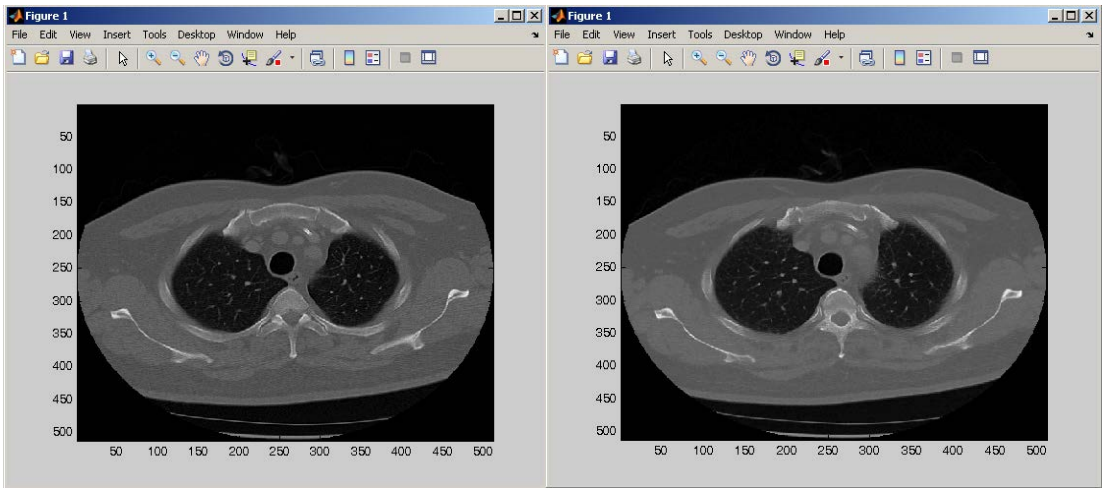
Close3D
Close operation in 3D.

function call:
[out_img1]=vsg('Close3D',in_img1,connected);

Arguments:
in_img1 - input image, a DICOM image.
connected – an integer specifying whether 6 or 26 connected is used

Description:
out_img1 – a DICOM image. Each value of the DICOM image is the result of first doing a 3D dilate, followed by a 3D erode operation.

Example:
```
[filename, chosenDirectory] = uigetfile('*.dcm', 'Pick an M-file');
filePath = [chosenDirectory filename];
stackOfImages=vsg('DICOMRead',filePath);  %4D array of data
[out_img1]=vsg('Close3D',stackOfImages,6);
```



Notes:
1. For RGB or Dicom images, the function operates on all colour planes or slices separately.
2. Dicom images require XMedCon software (version xmedcon-0.10.5-1-win32).
3. DICOM data must be in a separate folder.

Open3D

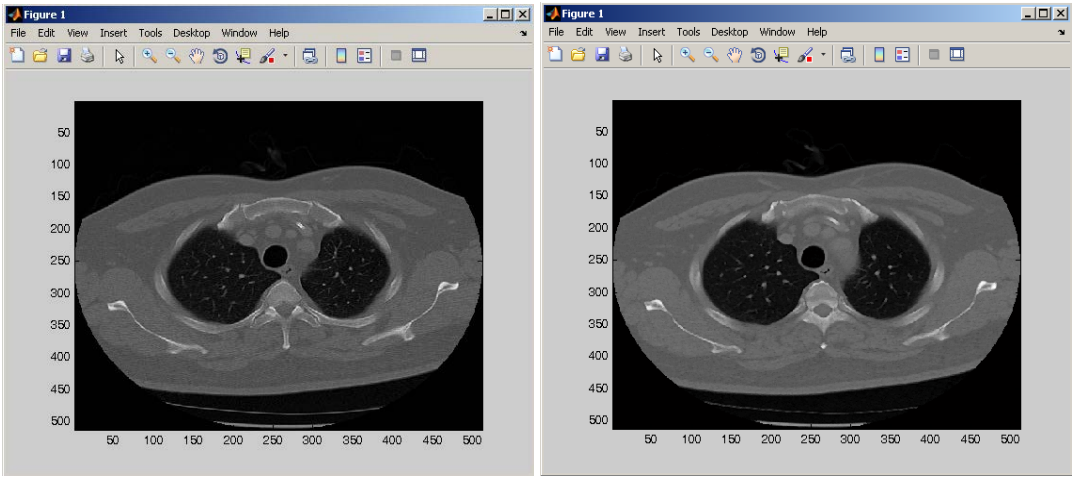
Open Operation in 3D.

function call:
[out_img1]=vsg('Open3D',in_img1,connected);

Arguments:
in_img1 - input image, a DICOM image.
connected – an integer specifying whether 6 or 26 connected is used.

Description:
out_img1 – a DICOM image. Each value of the DICOM image is the result of first doing a 3D erode, followed by a 3D dilate operation.

Example:
```
[filename, chosenDirectory] = uigetfile('*.dcm', 'Pick an M-file');
filePath = [chosenDirectory filename];
stackOfImages=vsg('DICOMRead',filePath);  %4D array of data
[out_img1]=vsg('Open3D',stackOfImages,6);
```



Notes:
1. For RGB or Dicom images, the function operates on all colour planes or slices separately.
2. Dicom images require XMedCon software (version xmedcon-0.10.5-1-win32).
3. DICOM data must be in a separate folder.

BeucherGrad3D

Morphological Gradient (user specify connectivity of the structured element 6 or 26) [Default=6].

function call:
[out_img1]=vsg('BeucherGrad3D',in_img1,connected);

Arguments:
in_img1 - input image, a DICOM image.
connected – an integer specifying whether 6 or 26 connected is used.

Description:
out_img1 – a DICOM image. Each value of the DICOM image is the result of subtracting a 3D eroded input image from a 3D dilated input image. DICOM data must be in a separate folder.

Example:
```
[filename, chosenDirectory] = uigetfile('*.dcm', 'Pick an M-file');
filePath = [chosenDirectory filename];
stackOfImages=vsg('DICOMRead',filePath);  %4D array of data
[out_img1]=vsg('BeucherGrad3D',stackOfImages,8);
```

Notes:
1. For RGB or Dicom images, the function operates on all colour planes or slices separately.
2. Dicom images require XMedCon software (version xmedcon-0.10.5-1-win32).
3. DICOM data must be in a separate folder.

ReconByDil3D

Reconstruction by dilation in 3D (user specify connectivity of the structured element 6 or 26) [Default=6].

function call:
[out_img1,out_img2]=vsg('ReconByDil3D',in_img1,in_img2,connected);

Arguments:
in_img1 - input image, a DICOM image, the mask image.
in_img2 - input image, a DICOM image, the seed image.
connected – an integer specifying whether 6 or 26 connected is used.

Description:
out_img1 – a DICOM image which has been reconstructed by dilation.
out_img2 – a DICOM image, the remaining parts of the image that were excluded from the reconstruction.

Example:
```
[filename, chosenDirectory] = uigetfile('*.dcm', 'Pick an M-file');
filePath = [chosenDirectory filename];
stackOfImages=vsg('DICOMRead',filePath);  %4D array of data
stackOfImages1 = stackOfImages;
stackOfImages1(20:size(stackOfImages1,1)-20,20:size(stackOfImages1,2)-
20,:,1:size(stackOfImages1,4)-1)=0;
connected=6; % 6 or 26 connected
[out_img1,out_img2]=vsg('ReconByDil3D',stackOfImages,stackOfImages1,con
nected);
```

Notes:
1. For RGB or Dicom images, the function operates on all colour planes or slices separately.
2. Dicom images require XMedCon software (version xmedcon-0.10.5-1-win32).
3. DICOM data must be in a separate folder.

Additional

DICOMRead

Reads dicom file.

function call:
img = vsg('DICOMRead','filename');

Arguments:
file - input file name of a DICOM file format data.

Description:
The function reads a dicom file or all the dicom files from the specified folder suing the UID of DICOM files.
img – a two dimensional array of values or a three dimensional array of values.

Example:

```
[filename, chosenDirectory] = uigetfile('*.dcm', 'Pick an M-file');
filePath = [chosenDirectory filename]; %filePath contains the complete
path of the dicom file.
stackOfImages=vsg('DICOMRead',filePath);  %one file or all the dicom
file from the choosen directory
h=figure;imagesc(stackOfImages(:,:,:,1));colormap(gray);set(h,'Name','1
');
```

Notes:
1. For RGB or Dicom images, the function operates on all colour planes or slices separately.
2. Dicom images require XMedCon software (version xmedcon-0.10.5-1-win32).
3. DICOM data must be in a separate folder.
4. DICOM data must be in separate slice per file format and not in multiple slice per file format.

DICOMReadSingle

Reads a single dicom file.

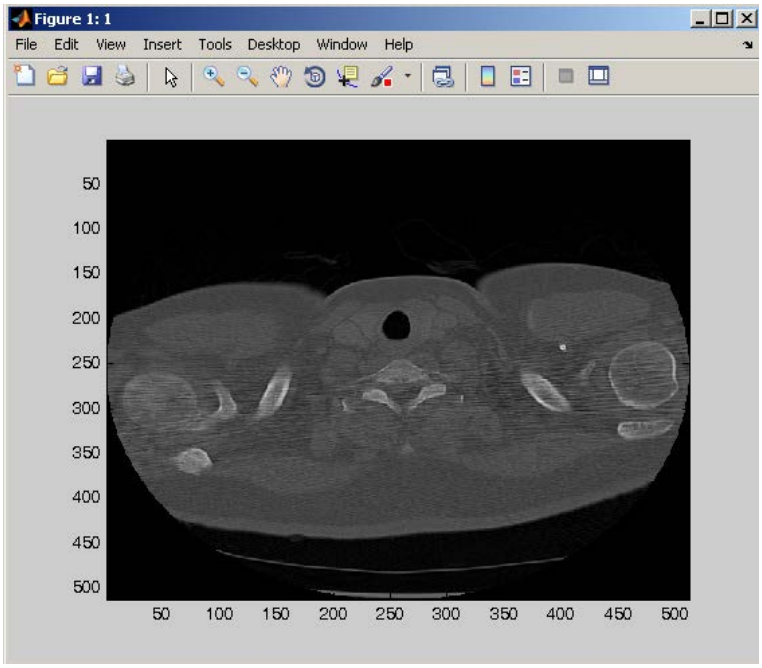
function call:
img = vsg('DICOMReadSingle','filename');

Arguments:
file - input file name of a DICOM file format data.

Description:
The function reads a single DICOM file or all the dicom file.
img – a two dimensional array of values of the DICOM image.

Example:

```
img=vsg('DICOMReadSingle', 'D:\work\testDICOMimage.dcm');
h=figure;imagesc(img),colormap(gray);set(h,'Name','1');
```



Notes:
1. For RGB or Dicom images, the function operates on all colour planes or slices separately.
2. Dicom images require XMedCon software (version xmedcon-0.10.5-1-win32).
3. DICOM data can be in either of the DICOM formats.

EjectFrac

Returns a double which represents the ejection fraction of a heart.

function call:
[frac]=vsg('EjectFrac',in_img1,in_img2,p1,p2);

Arguments:
in_img1 - input image, a DICOM image. A DICOM image of the heart in systolic state

in_img2 - input image, a DICOM image. A DICOM image of the heart in diastolic state.
p1 - a 2-vector specifying the (x,y) location of the approximate centre of the left ventricle in the first slice of the systolic heart image.
p2 - a 2-vector specifying the (x,y) location of the approximate centre of the left ventricle in the first slice of the diastolic heart image.

Description:
frac – a double which returns the estimated ejection fraction of the heart in the input image. This is calculated by segmenting and calculating the volume of the ventricle in both systolic and diastolic states.

Example:
```
stackOfImages1=vsg('DICOMRead','D:\single\patient_17_systolic.dcm');
stackOfImages2=vsg('DICOMRead','D:\second\patient_17_diastolic.dcm');
x1=129;y1=100;
x2=129;y2=106;
[frac]=vsg('EjectFrac',stackOfImages1,stackOfImages2,[x1 y1],[x2 y2])
frac = 0.1899
```

Notes:
1. For RGB or Dicom images, the function operates on all colour planes or slices separately.
2. Dicom images require XMedCon software (version xmedcon-0.10.5-1-win32).
3. DICOM patient data must be in a separate folder.

LevelSet

Estimates and extracts the contour in an image based on modelling the evolving contour as a signed function.

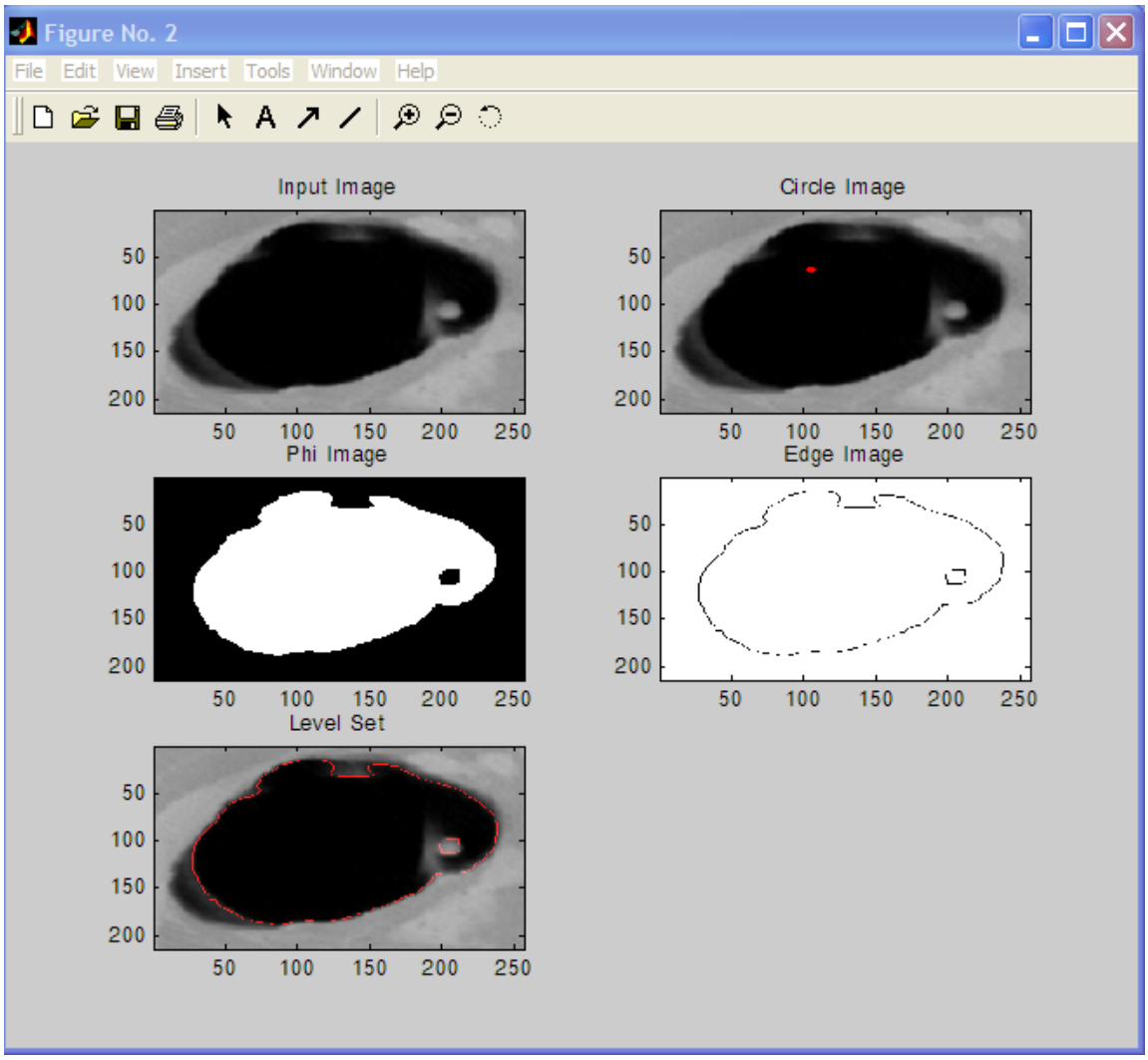
function call:
[out_img1,out_img2,out_img3]=vsg('LevelSet',in_img1,x,y);

Arguments:
in_img1 – a greyscale or RGB image wich is to be segmented.
x – an integer specifying the x location of the seed point for the level-set operation
y – an integer specifying the y location of the seed point for the level-set operation

Description :
out_img1 – a greyscale or RGB image. This image contains the input image with the level-set marked on it.
out_img2 – a binary image. This image contains the edge of the level-set region.
out_img3 – a greyscale image. This image contains the phi image, i.e. the distance of each pixel from the edge of the level-set region.

Example:
```
img = openimage('tumor.gif'); %color image
figure,subplot(3,2,1);image(img),title('Input Image');hold on
x = 104;
y = 63;
[out_img1]=vsg('FillCircle',img,[x y],3,[255 0 0]);
subplot(3,2,2),image(uint8(out_img1)),title('Circle Image');
[out_img2 out_img3 out_img4]=vsg('LevelSet',img,x,y);
subplot(3,2,3); image(uint8(out_img3)),colormap(gray),title('Phi
Image');
```

```
subplot(3,2,4); image((out_img4)),colormap(gray),title('Edge Image');
subplot(3,2,5); image((uint8(out_img2))),colormap(gray),title('Level
Set');
```



Notes:
1. For RGB images, the function operates on all colour planes together and returns a single levelset for the image.

Key References:

1. Ovidiu Ghita, Dana E. Ilea and Paul F. Whelan (2012), "An Adaptive Noise Removal Approach for Restoration of Digital Images Corrupted by Multimodal Noise", **IET Image Processing**. Volume: 6, Issue: 8 Page(s): 1148-1160 November 2012.
2. Ovidiu Ghita, Dana Ilea, Antonio Fernandez and Paul F. Whelan (2012), "Local binary patterns versus signal processing texture analysis: a study from a performance evaluation perspective", **Sensor Review**, Volume 32 issue 2, 149-162
3. Dana E. Ilea and Paul F. Whelan (2011), "Image Segmentation based on the Integration of Colour-Texture Descriptors - A Review", **Pattern Recognition** 44 (2011) 2479–2501. **Most downloaded article in Pattern Recognition January to December 2011**
4. Paul F. Whelan and B.G. Batchelor (2011), "Basic Image Processing Techniques", in **Machine Vision Handbook**, Ed. B.G. Batchelor, Springer-Verlag London Limited, ISBN: 978-1-84996-169-1 [*Invited book chapter*]
5. Paul F. Whelan, R. Sadleir and O Ghita (2011), "NeatVision: A Development Environment for Machine Vision Engineers", in **Machine Vision Handbook**, Ed. B.G. Batchelor, Springer-Verlag London Limited, ISBN: 978-1-84996-169-1 [*Invited book chapter*]
6. B.G. Batchelor and Paul F. Whelan (2011), "Proverbs", in **Machine Vision Handbook**, Ed. B.G. Batchelor, Springer-Verlag London Limited, ISBN: 978-1-84996-169-1 [*Invited book chapter*]
7. O Ghita, T Carew and Paul F. Whelan (2011), "A machine vision system for quality grading of painted slates", in **Machine Vision Handbook**, Ed. B.G. Batchelor, Springer-Verlag London Limited, ISBN: 978-1-84996-169-1 [*Invited book chapter*]
8. Ovidiu Ghita, Paul F. Whelan (2010), "A new GVF-based image enhancement formulation for use in the presence of mixed noise", **Pattern Recognition** Volume 43, Issue 8, August 2010, Pages 2646-2658
9. Ovidiu Ghita, Dana E Ilea and Paul F. Whelan (2009), "Image feature enhancement based on the time-controlled total variation flow formulation", **Pattern Recognition Letters** Volume 30, Issue 3, 1 February 2009, pp. 314-320
10. Dana. E. Ilea and Paul F. Whelan (2008), "CTex - An adaptive unsupervised segmentation algorithm based on colour-texture coherence", **IEEE Transactions on Image Processing** 17(10), pp. 1926-1939, OCTOBER 2008
11. Paul F. Whelan and O Ghita (2008), "Chapter 5: Color Texture Analysis" in **Handbook of Texture Recognition**, edited by Majid Mirmehdi, Xianghua Xie and Jasjit Suri, ISBN 978-1-84816-115-3 / 1-84816-115-8], Imperial College Press, Nov 2008. [*Invited book chapter*]
12. O Ghita and Paul F. Whelan (2008), "A Systems Engineering Approach to Robotic Bin Picking" in **Stereo Vision**, edited by Asim Bhatti, ISBN 978-953-7619-22-0, 372 pages, InTech, November, 2008. [*Invited book chapter*]
13. Paul F. Whelan (2003), "Automated cutting of natural products: A practical packing strategy", Chapter 11 in **Machine Vision for the Inspection of Natural Products**. Mark Graves and Bruce Batchelor (Eds.), Springer (London). , ISBN: 1-85233-525-4, pp 307-329 [*Invited book chapter*]
14. Tarik A. Chowdhury, Paul F. Whelan and Ovidiu Ghita (2006), "The use of 3D surface fitting for robust polyp detection and classification in CT colonography", Computerized Medical Imaging and Graphics, 30 (2006) 427-436
15. O Ghita, K Robinson, M Lynch and Paul F. Whelan (2005), "MRI diffusion-based filtering: A note on performance characterisation", Computerized Medical Imaging and Graphics; 29(4):267-277
16. Ovidiu Ghita, Paul F. Whelan and John Mallon (2005), "Computational approach for depth from focus", Journal of **Electronic Imaging** 14(2), April 2005, 023021 (1-8)
17. Paul F. Whelan, Robert J. T. Sadleir, and Ovidiu Ghita, (2004) "**Informatics in Radiology (*info*RAD): NeatVision: Visual Programming for Computer-aided Diagnostic Applications**" **Radiographics;** 24(6):1779-1789
18. K. Robinson and Paul F. Whelan (2004), "Efficient Morphological Reconstruction: A Downhill Filter", **Pattern Recognition Letters**, 25(15), November 2004, pp: 1759-1767
19. Robert J. T. Sadleir, Paul F. Whelan, Padraic MacMathuna, and Helen M. Fenlon (2004) "Informatics in Radiology (infoRAD): Portable Toolkit for Providing Straightforward Access to Medical Image Data" **Radiographics.** 2004 Jul-Aug:24(4):1193-1202
20. Paul F. Whelan and R.J.T. Sadleir (2004), "A Visual Programming Environment for Machine Vision Engineers", **Sensor Review**, 24(3), pp 265-270
21. O. Ghita and Paul F. Whelan (2002), "A computationally efficient method for edge thinning and linking using endpoints", **Journal of Electronic Imaging**, 11(4), Oct. 2002, pp 479-485.
22. A. Drimbarean and Paul F. Whelan (2001), "Experiments in colour texture analysis", **Pattern Recognition Letters**, 22(10), pp 1161-1167
23. Paul F. Whelan and D. Molloy (2001), **Machine Vision Algorithms in Java: Techniques and Implementation**, Springer (London), 298 Pages. ISBN 1-85233-218-2. (Reprinted Aug. 2001)

24. Paul F. Whelan and Bruce G. Batchelor (2001), “Basic machine vision techniques”, Chapter 2 in **Intelligent Machine Vision: Techniques, Implementation & Interfacing**, B.G. Batchelor and F. Waltz, Springer-Verlag UK, 448 pages, ISBN: 3-540-762248, pp 31-64 [*Invited book chapter*]
25. Paul F. Whelan (2001), “Visual programming for machine vision”, Chapter 8 in **Intelligent Machine Vision: Techniques, Implementation & Interfacing**, B.G. Batchelor and F. Waltz, Springer-Verlag UK, 448 pages, ISBN: 3-540-762248, pp 229-320 [*Invited book chapter*]
26. O. Ghita and Paul F. Whelan (2000), "Real time 3D estimation using depth from defocus", Vision, MVA (SME), 16(3), pp 1-6.
27. B.G. Batchelor, Paul F. Whelan and R. Hack (1999), “Machine vision for inspection and robotic control”, **Wiley Encyclopaedia of Electrical and Electronics Engineering**, J. Webster (Ed.), John Wiley and Sons Inc., 24 Volume Set, ISBN 0-471-13946-7, Volume 11, pp 659-673.
28. Paul F. Whelan (1997), "Remote access to continuing engineering education - RACeE", **IEE Engineering Science and Education Journal**, 6(5), pp 205-211.
29. Paul F. Whelan (1997), "System engineering issues in industrial inspection", **IEE Colloquium on Industrial Inspection** (Ref. No.1997/041), IEE, London (UK), pp 1/1-1/6.
30. Bruce G. Batchelor and Paul F. Whelan (1997), **Intelligent Vision Systems for Industry**, Springer-Verlag (London), 457 pages, ISBN 3-540-19969-1.
31. B.G. Batchelor and Paul F. Whelan (1997), "Real-time colour recognition for machine vision systems", **John Dalton’s Colour Vision Legacy**, Editors C. Dickinson, I. Murray & D. Carden, Taylor & Francis (London), ISBN 0-7484-0310-8, Chap. 10.5, pp 687-693.
32. B.G. Batchelor and Paul F. Whelan (1996), "Machine vision systems: Proverbs, principles, prejudices and priorities", **Proc. Applied Machine Vision '96**, Machine Vision Association (Society of Manufacturing Engineers), Cincinnati (USA), pp. 7-19. Reprinted in part by SRC VISION (USA) and Integral Vision, Inc. (USA).
33. Paul F. Whelan and B.G. Batchelor (1996), "Automated packing systems - A systems engineering approach", **IEEE Trans. on Systems, Man and Cybernetics - Part A [Systems and Humans]**; 26(5):533-544.
34. Bruce G. Batchelor and Paul F. Whelan (Eds.) (1994), **Selected Papers on Industrial Machine Vision S***ystems*, SPIE Milestone Series MS 97, SPIE Optical Engineering Press, 629 pages. ISBN 0-8194-1580-4 (hard bound); ISBN 0-8194-1579-0 (soft bound).
35. B.G. Batchelor and Paul F. Whelan (1994), "Introduction to industrial machine vision systems", Introduction to **Selected Papers on Industrial Machine Vision Systems**, SPIE Milestone Series MS 97, SPIE Optical Engineering Press, pp xi-xxiv.

http://www.cipa.dcu.ie/code.html