ITK2DM – Help

credits

This wrapper library was made by Bernhard Schaffer. The functional code belongs to the Insight Segmentation and Registration Toolkit (ITK), which is an open-source cross-platform library.[1]
email: Bschaffer@superstem.org
version: 2014-11-17

purpose

ITK is a high-end library aiming at providing state of the art computational routines for segmentation and registration of images. This DigitalMicrograph library provides wrapper functions such that the routines can be utilized in the DigitalMicrograph scripting language. No user interface on top of the script commands is currently provided.

Contents

[1] http://www.itk.org/

Setup

Copy the *b_ITK2DM.gtk* file into the plugins folder and restart DigitalMicrograph. On startup, you should receive a message "*Loaded 'ITK to DM' wrapper plugin*" in the results window, and the listed script commands below should be accessible.

Provided functionality

Only a limited amount of the ITK functionality has currently been wrapped. If you have need for a particular functionality, please email me. It is the intention to regularly update this plugin and increase its functionality over time.
Please note that this library only provides glue-code between DigitalMicrograph scripting and ITK[2]. The actual computation is performed solely in the ITK code. Any computational bugs should therefore be provided to the ITK community. Also, for detailed descriptions of the algorithms behind computations, please refer to the ITK documentation and community. In particular, read the ITK Software Guide[3] for the mathematical background of various routines and the meaning of the parameters.

MetaInfo

`ITKVersion`

```
String ITKVersion ( long what )
```

This returns the plugin version information as a string. Use value 0 for the version of the plugin, 1 for the version of linked ITK library.

`ITKProvided`

```
String ITKProvided ( )
```

This returns a string listing all provided script commands for reference.

[2] http://www.itk.org/
[3] http://www.itk.org/ItkSoftwareGuide.pdf

Smoothing

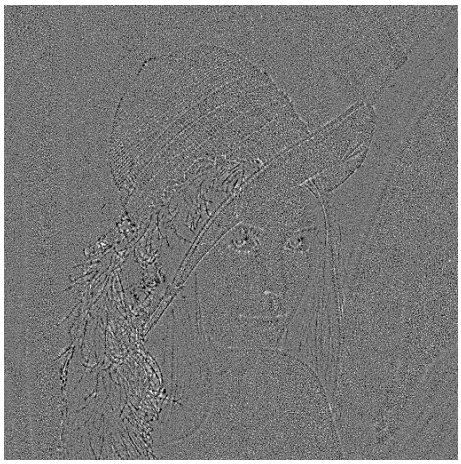
ITKCurvatureFlowImageFilter

Image ITKCurvatureFlowImageFilter(Image src, Number iterations, Number timeStep)

This is an iterative, edge preserving smoothing filter. Input images may be 2D or 3D. The ITK manual suggests *timeStep* values of 0.125 and 0.0625 for 2D and 3D images, respectively.


Orginal image (512 x 512)


10 iterations with timeStep 0.125


difference

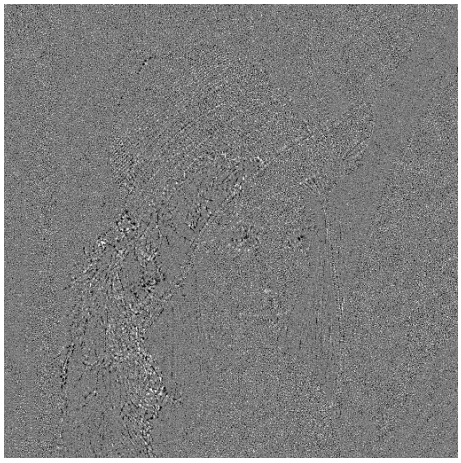
ITKMinMaxCurvatureFlowImageFilter

Image ITKMinMaxCurvatureFlowImageFilter(Image src, Number iterations, Number timeStep, Number stencilRadius)

This is an iterative, edge preserving smoothing filter and a variant of the CurvatureFlowImageFilter. It is supposed to have less diffusion. Input images may be 2D or 3D. The ITK manual suggests *timeStep* values of 0.125 and 0.0625 for 2D and 3D images, respectively. The *stencilRadius* defines a local neighbourhood for the algorithm and can typically be 1. Larger values reduce the filtering effect and slow down processing.


Orginal image (512 x 512)


10 iterations, timeStep 0.125, stencil 1


difference

ITKBilateralImageFilter

```
Image ITKBilateralImageFilter( Image src, Number rangeSigma, Number sigmaX, Number sigmaY )
Image ITKBilateralImageFilter( Image src, Number rangeSigma, Number sigmaX, Number sigmaY, Number sigmaZ )
```

This is an edge preserving smoothing filter using both domain and range neighbourhoods. Input images may be 2D or 3D. The ranges are defined by *rangeSigma* on the intensity scale and by 2 or 3 spatial sigma values for 2D and 3D images, respectively. Larger values increase the smoothing effect and make the filter slower.

Orginal image (512 x 512)

rSigma = 40 and xSigma = ySigma = 10

difference

Segmentation filters

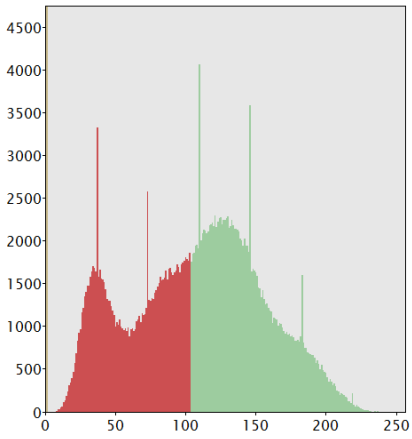
ITKOtsuThresholdImageFilter

```
Image ITKOtsuThresholdImageFilter( Image src, NumberVariable foundThreshold )
```

This filter finds the threshold value which classifies the image into two clusters a ‘background’ and a ‘foreground’. [4] The command returns both the found threshold value and the binary mask image. Input images may be 2D or 3D.

Orginal image (512 x 512)

Mask (threshold = 102.1)



Histogram with threshold

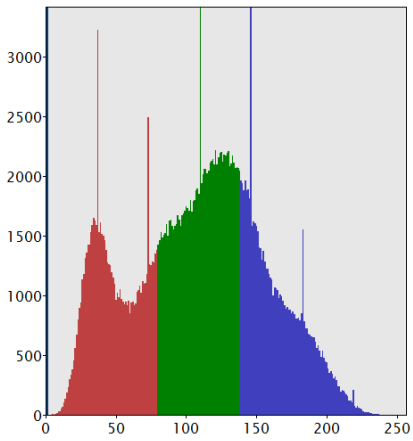
ITKOtsuMultipleThresholdImageFilter

```
Image ITKOtsuMultipleThresholdImageFilter( Image src, Number nValues, TagGroup foundThreashold )
```

This filter extends the OtsuThresholdImageFilter to segment an image into (n+1) segments by n threshold values. The values are stored in a TagGroup. Note, that the algorithm becomes slow very rapidly with increasing n. The returned image has labels from 0 to n. Input images may be 2D or 3D.

Orginal image (512 x 512)

Mask for n = 2



Histogram with thresholds

[4] http://en.wikipedia.org/wiki/Otsu's_method

ITKConnectedThresholdImageFilter

Image ITKConnectedThresholdImageFilter(Image src, Number lowerBound, Number higherBound, TagGroup seedsIndices)

This is a region-growing threshold filter. Input images may be 2D or 3D. Starting from points provided as taglist *seedInidices* pixels which are spatially connected and within the given intensity limits. The filter works best if first combined with a noise reduction/smoothing filter. The *seedIndices* taggroup can be created from point ROIs on the image by the script shown in the appendix.


Smoothed orginal image (512 x 512)


Connected threshold mask (0 – 64)


Simple Treshold mask (0 – 64)

ITKNeighborhoodConnectedImageFilter

Image ITKNeighborhoodConnectedImageFilter(Image src, Number low, Number high, Number radius, TagGroup seedsIndices)

This is a region-growing threshold filter. Input images may be 2D or 3D. It is very similar to the ITKConnectedThresholdImageFilter starting from points provided as taglist *seedInidices* pixels which are spatially connected and within the given intensity limits. Pixels are only accepted if if *all* its neighbours have intensities that fit in this interval. The reason for considering the neighborhood intensities instead of only the current pixel intensity is that small structures are less likely to be accepted in the region.


Smoothed orginal image (512 x 512)


Neighborhoodconnected threshold mask (0 – 64) with radius 1


Neighborhoodconnected threshold mask (0 – 64) with radius 2

ITKConfidenceConnectedImageFilter

Image ITKConfidenceConnectedImageFilter(Image src, Number multiplier, Number nIters, Number initRadius, TagGroup seeds)

This is an iterative region-growing threshold filter. Input images may be 2D or 3D. Starting from the seeds, regions are iteratively grown as long as new pixels from the neighbourhood are added. Pixels are added, if their intensities lie within the a range around the mean value of the already accepted regions. Mean and standard deviation are recomputed with each iteration. *Multiplier* defines a multiplicative factor with which the standard deviation is multiplied to define the valid range. *nIter* defines the maximum number of iterations attempted (if no convergence is reached first) and *initRadius* specifies the initial neighbourhood around the seed for the first iteration.
It is very similar to the ITKConnectedThresholdImageFilter starting from points provided as taglist *seedInidices* pixels which are spatially connected and within the given intensity limits. Pixels are only accepted if if *all* its neighbours have intensities that fit in this interval. The reason for considering the neighborhood intensities instead of only the current pixel intensity is that small structures are less likely to be accepted in the region.[5]


Smoothed orginal image (512 x 512)


ConfidenceConnected threshold mask with radius 1, and multiplier 2


ConfidenceConnected threshold mask with radius 1, and multiplier 3

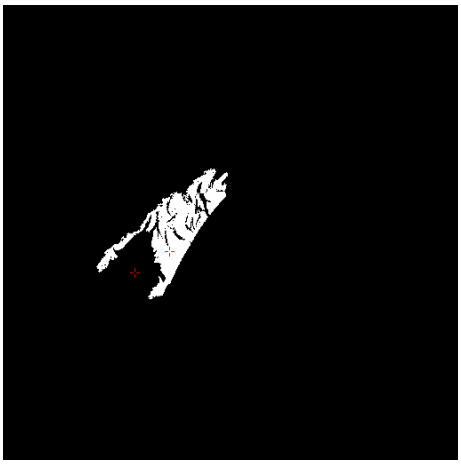
[5] For clarification: If multiple seeds are used, the *total* accepted areas are used to compute a single mean and standard deviation. Hence seeds on pixels of different intensity levels will result in a rather generous acceptance range!

ITKIsolatedConnectedImageFilter

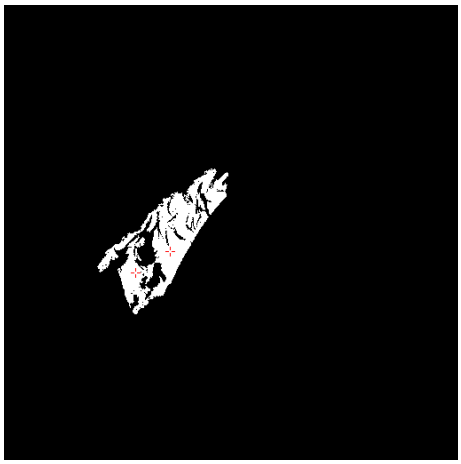
Image ITKIsolatedConnectedImageFilter(Image src, Number lower, Number seed1, Number seed2, NumberVariable isolatedValue)

This is a region-growing threshold filter which is very similar to the ConnectedThresholdImageFilter. It requires the input of a lower threshold limit and two seed points (given as pixel index). Based on this input, a higher threshold value is determined which ensures that the threshold mask includes the first but excludes the second seed point. The threshold value and the mask are returned. Input images may be 2D or 3D.

Smoothed orginal image (512 x 512)



IsolatedConnected threshold mask ,
Lower bound = 0,
found isolation value = 56



Connected threshold mask (0 – 56)

Appendix

Version notes

The plugin currently wraps around the ITK library version number 4.6.0.

Script to compile a seedInidices TagList from point-ROIs on a 2D image

```
image test := GetFrontImage()
number sx,sy
test.getsize(sx,sy)

imagedisplay disp = test.imagegetimagedisplay(0)
number nROI = disp.imagedisplaycountROIs()
number nPROI = 0
for (number n=0;n<nROI;n++)
{
	ROI r = disp.imagedisplaygetROI(n)
	nPROI += r.ROIIsPoint() ? 1 : 0
}
if ( 0 == nPROI )
	Throw( "No points found" )

TagGroup seeds = NewTagList();
for (number n=0;n<nROI;n++)
{
	ROI r = disp.imagedisplaygetROI(n)
	if ( r.ROIIsPoint() )
	{
		number px,py;
		r.ROIGetPoint(px,py)
		seeds.TagGroupInsertTagAsLong( Infinity(), px + py*sx)
	}
}
```

This script is for 2D images. For 3D images, the pixel index of an image of size [sx * sy * sz] at point (px,py,pz) is: px + py*sx + pz*sx*sy.