Microscopy Image Analysis: Blob Segmentation using Geodesic Active Contours

Release 1.00

Kishore Mosaliganti, Arnaud Gelas, Alexandre Gouaillard and Sean Megason

March 11, 2009

Systems Biology, Harvard Medical School, Boston, MA-02139, USA

Abstract

An Insight Toolkit (ITK) processing framework for blob segmentation with applications to microscopy images is presented in this paper. Our algorithm is a refinement of the work of Mosaliganti *et al.* [3] for splitting cell clusters. The basic idea is to incorporate as many cues as possible into developing a suitable level-set speed function so that an evolving contour exactly segments a cell/nuclei blob. We use image gradients, distance maps, multiple channel information and a shape model to drive the evolution. The framework consists of a linear pipeline of 6 new ITK filters which are applied in succession to generate the segmentation. The filters extract the cell foreground, construct the speed image, find seed points for evolution and collect cell statistics from the segmentation. We include 2D/3D example code, parameter settings and show the results generated on confocal images of the zebrafish embryo.

Latest version available at the Insight Journal [http://hdl.handle.net/1926/1]

Contents

1 Introduction

Most biological experiments that involve microscopic imaging require cell profiling, counting, segmentation and tracking of cells. Cell/nuclei segmentation is often the first step in any analysis protocol and is becoming an increasingly popular topic in the image analysis community. The nucleus/cell forms the fundamental biological entity of interest. The problem arises when nuclei appear as overlapping or touching each other. Identifying each nucleus separately in a biologically consistent fashion is non-trivial. While some biochemical stains provide viable clues in the form of sharp color-space gradients at the boundaries, others exhibit a narrow *neck* at the site of overlap between two nuclei. We implement an approach that elegantly incorporates available cues and shapes into a viable segmentation pipeline [3]. We use geodesic active contours incorporating shape priors in a level-set framework to provide nuclei segmentations.

The framework is implemented in $2D/3D$ for segmenting blobs (cells or nuclei) and for multiple stains that may be available. For example, often researchers use a nuclear stain and a separate cell membrane stain both of which are captured in two separate channels. The membrane channel can then be used to provides additional help in separating overlapping blobs etc. The implementation consists of 6 ITK filters whose description and use is documented in the rest of the document. Note that for convenience, we use the term nuclei but it can be replaced by any repeating small blob. We also assume that at this point, relevant preprocessing such as deconvolution, filtering and channel unmixing have been carried out to increase the SNR of the data.

2 Foreground Detection

In order to separate the nuclei, we employ a divide-and-conquer strategy to implement the filter `itk::CellForegroundExtraction`. The idea is to first separate nuclear foreground from the background before splitting them into individual entities. We identify nuclear material from the background by drawing information from the following sources: nuclei stain, membrane stain (identified membrane voxels which separate conjoined nuclei) and finally, the convex shape of the nuclei intensity function. A multiple threshold strategy is applied in each case.

The filter `itk::CellForegroundExtraction` is templated over the Input, Foreground and Correlation Im-

age Types. It has a total of 6 parameters whose settings are described below. NOTE: The setting of parameters takes into account the spacing in the images. Hence, if the images are in μm, then the relevant parameters also need to be in the same units.

First, note that the nuclei stain primarily delineates nuclei material from the rest of the image. However, the staining can be inhomogenous and sometimes photon shot noise is captured in the acquisition. An upper and lower threshold is applied in this case. The upper threshold (*m_ThresholdCellmax*) detects a nuclei voxel. The lower threshold rules out a nuclei voxel (*m_ThresholdCellmin*). The intermediate range is further resolved as follows.

`m_ThresholdCellmin` - lower threshold for nuclei stain given by the user in $[0,255]$.
`m_ThresholdCellmax` - upper threshold for nuclei stain given by the user in $[0,255]$.

Other stains, such as the membrane stain may be available, which can also separate the membrane from being classified as the nuclei foreground. A single threshold is applied to classify membrane pixels as being the background.

`m_ThresholdMembrane` - threshold for the membrane stain set by the user in $[0,255]$

We also detect nuclear foreground by detecting the convex shape of the nuclear intensity function by comparing with a Gaussian kernel. Let C_i represent a Gaussian kernel in the image I at location p, its domain (or extent of support) (δ given by *m_SigmaForm*) and peak intensity value equal to unity. It is described by the following equation.

`m_SigmaForm` - Standard deviation of the Gaussian kernel set by the user. Typically, half the size of the nucleus diameter. Usually 3-5μm.

$$\mathscr{C}(x) = \frac{1}{2\pi^{\frac{3}{2}}|\Sigma|^{\frac{1}{2}}} \exp[-\frac{1}{2}(x-p)\Sigma^{-1}(x-p)^T] \tag{1}$$

At all pixels in the intermediate range, a Gaussian kernel with typical parameters, is placed and the Pearsons correlation ($R(p)$) is computed. A high value of the correlation corresponds to the center of the nucleus, a medium value to the background and a negative value to the region in-between the nuclei. Hence, a threshold is again applied on the correlation value to resolve the intermediate voxels.

$$R(p) = \rho(\mathscr{C}(p,\Sigma), I)_{\delta(p)} \tag{2}$$

The correlation ($R(p)$) is computed in a neighborhood patch ($\delta(p)$) that is rectangular. The size of the patch is determined by *m_LargestCellRadius*.

`m_LargestCellRadius` - Largest typical nuclei radius. Usually 5-8μm.

The detected foreground might still contain small holes, which have escaped detection, but which are filled using a hole-filling filter internally.

In order to speed up computation, the images can also be under-sampled during the intermediate computations and resampled back. The ratios for sub-sampling are set using *m_Sampling*.

2.1 Usage

```
const unsigned int Dimension = 3;
```

```
typedef itk::Image< float,Dimension > InputImage;
typedef itk::Image< unsigned short,Dimension > fgImage;
typedef itk::Image< float,Dimension > gfImage;
typedef itk::CellForegroundExtraction< InputImage, fgImage, gfImage > FilterType;

...

// Set sub-sampling ratios
float sampling[Dimension] = {4,4,2};

FilterType::Pointer filter = FilterType::New();
filter->SetInput( 0, cellImg );
filter->SetInput( 1, membraneImg );
filter->SetSigmaForm( 2.0 ); // in real coordinates
filter->SetThresholdCellmin( 70 );
filter->SetThresholdCellmax( 120 );
filter->SetThresholdMembrane( 255 );
filter->SetThresholdForm( 0.5 );
filter->SetLargestCellRadius( 4.0 ); // in real coordinates
filter->SetSampling( sampling );
filter->Update();
```

The output consists of the foreground image (`filter->GetOutput()`) as well as the correlation map $R(p)$ (`filter->GetGaussCorrImage()`). The correlation map has uses later in the pipeline.

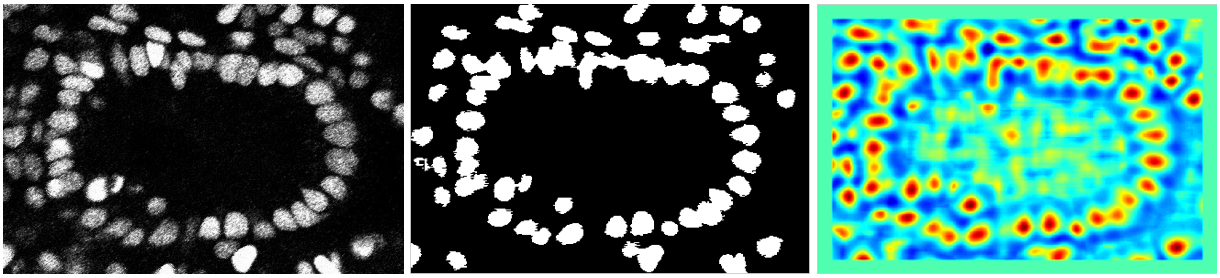
2.2 Results

The results in this example can be obtained by using `CellForegroundExtraction2D.cxx` on the input image `30.png` and `30-memb.png` using the parameter settings given below. The output is written out in the images `30-fg.mha` and `30-gf.mha`. Potential users of this code can execute the example with the following command line:

```
./cellForeground2D 30.png 30-memb.png 30-fg.mha 30-gf.mha

Typical filter settings:
filter->SetSigmaForm( 2.0 ); // in real coordinates
filter->SetThresholdCellmin( 70 );
filter->SetThresholdCellmax( 120 );
filter->SetThresholdMembrane( 255 );
filter->SetThresholdForm( 0.5 );
filter->SetLargestCellRadius( 4.0 ); // in real coordinates
```

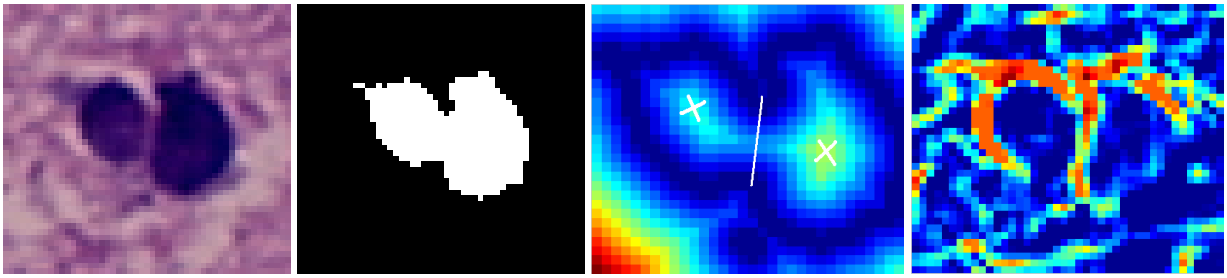
3 Feature Extraction

A key step in using explicit level-set methods is the creation of a suitable feature or speed map f to drive the evolution. This function is designed to be minimized at nucleus boundaries and remains high elsewhere. The function is based on three ideas from literature that were developed independently namely, (i) inter-nuclear boundary gradients and (iii) morphological shape cues. We now describe the implementation of the filter `itkCellFeatureGenerator`.



(a) (b) (c)

Figure 1: (a) Confocal image showing nuclei of the zebrafish ear. (b) Extracted foreground (c) Gaussian correlation map showing convex regions and suppressing non-convex ones.



(a) (b) (c) (d)

Figure 2: (a) A pair of overlapping nuclei. (b) Applying a threshold provides the nuclei foreground I_f and background I_b. (c) Unsigned distance field P of the nuclear contour. The nuclei centers ('x') are located at maxima and the neck region is in a directional minimum (white line segment). (d) High gradient magnitudes $|\nabla I|$ typically exist in the neck region although this is not guaranteed.

The filter `itk::CellForegroundExtraction` is templated over the Input, Foreground and Correlation Image Types. It has a total of 6 parameters whose settings are described below. NOTE: The setting of parameters takes into account the spacing in the images. Hence, if the images are in μm, then the relevant parameters also need to be in the same units.

Let $D(I_f)$ and $D(I_b)$ represent the unsigned distance fields emanating from the nuclei contours that exist outside and within the contour respectively. Consider the image formed by $P = D(I_f) + D(I_b)$. $D(I_b)$ causes the appearance of a directional minimum in the neck region as represented by a white line segment. The appearance of a directional minimum is a direct consequence of the *neck* shape resulting in lower magnitudes of the distance field. Similarly, $D(I_f)$ causes the appearance of a valley region around the nuclei foreground-background interface. Hence, P is a suitable speed function in such cases to control the evolution of a level-set to segment individual cells. Now, consider cases where image gradients are significant.

Some nuclei exhibit high gradients at the site of overlap as shown in Figure 2d. As in standard practice, we take the gradient information, $g(I)$ into account. The gradient information $g(I)$ is the sigmoid function of the derivative-of-Gaussian (doG) gradient magnitudes with an appropriate σ. The σ is automatically determined from the size of the cells (*m_LargestCellRadius*).

`m_SigmaCell` - doG filter σ for nuclear stain. Usually 1/10th the nuclear diameter. `m_SigmaCorrelation` - doG filter σ for correlation map. Usually 1/10th the nuclear diameter. `m_LargestCellRadius` - Largest

typical nuclei radius. Usually 5-8μm.

Finally, we also use the correlation map ($R(p)$) computed as output in *itkCellForegroundExtraction* as yet another source. These three sources are assigned weights to yield the final speed function.

m_DistanceMapWeight, *m_GradientMagnitudeWeight*, *m_GaussCorrWeight* - Weights for the three individual speed functions to create a cumulative speed function. Usually, we use a higher weight for the distance map (0.5) and the gradients (0.3) compared to the correlation map (0.2).

In order to speed up computation, the images can also be under-sampled during the intermediate computations and resampled back. The ratios for sub-sampling are set using *m_Sampling*.

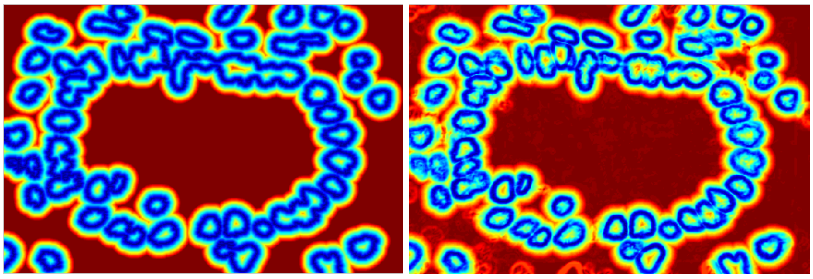
3.1 Usage

```
const int Dimension = 3;
typedef itk::Image< float, Dimension > InputImage;
typedef itk::Image< float, Dimension > FeatureImage;
typedef itk::Image< bool, Dimension > FgImage;

typedef itk::CellFeatureGenerator< InputImage,FeatureImage > FilterType;

float sampling[Dimension] = { 1, 1, 1};

FilterType::Pointer filter = FilterType::New();
filter->SetInput( 0,rawImg );
filter->SetInput( 1,gaussCorrImg );
filter->SetForeground( fgMap );
filter->SetLargestCellRadius( 6.0 );
filter->SetSigmaCell( 0.4 );
filter->SetSigmaCorrelation( 0.4 );
filter->SetSampling( sampling );
filter->SetDistanceMapWeight( 0.5 );
filter->SetGradientMagnitudeWeight( 0.3 );
filter->SetGaussCorrWeight( 0.2 );
filter->Update();
```

3.2 Results

The results in this example can be obtained by using `CellFeatureGenerator2D.cxx` on the input image `30.png`, foreground image `30-fg.png` and correlation image `30-gf.png` using the parameter settings given below. The output is written out in the images `30-feature.mha` and `30-dist.mha`. Potential users of this code can execute the example with the following command line:

```
./cellFeature2D 30.png 30-fg.mha 30-gf.mha 30-feature.mha 30-dist.mha

filter->SetLargestCellRadius( 6.0 );
filter->SetSigmaCell( 0.4 );
filter->SetSigmaCorrelation( 0.4 );
filter->SetSampling( sampling );
filter->SetDistanceMapWeight( 0.5 );
filter->SetGradientMagnitudeWeight( 0.3 );
filter->SetGaussCorrWeight( 0.2 );
```



(a) (b)

Figure 3: (a) Distance map image. (b) Feature/speed function generated.

4 Seed Generation

The next logical step in using level-set methods is to generate a suitable initialization (seed points). In our approach, we generate a list of potential seed points that are used to generate level-set functions. The seed points are chosen as the maxima points in the distance map that are located in the nuclei foreground.

Maxima points are located by dilating the distance map and finding those points that do not change in value. It is easy to see that dilation by a mask does not change the value at only the maxima points. The size of the mask determines the minimum separation distance between any two seed-points. It is set depending on the largest cell diameter supplied by the user (*m_LargestCellRadius*). The filter `itk::SeedExtraction` is templated over the Feature and Foreground Image Types. It has a single parameter. NOTE: The setting of parameters takes into account the spacing in the images. Hence, if the images are in μm, then the relevant parameters also need to be in the same units.

`m_LargestCellRadius` - Largest typical nuclei radius. Usually 5-8μm.

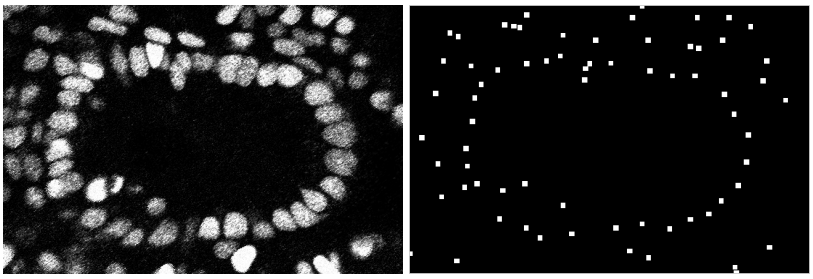
4.1 Usage

```
const unsigned int Dimension = 3;

...

typedef itk::SeedExtraction< featureImage,fgImage > FilterType;

FilterType::Pointer filter = FilterType::New();
filter->SetForeground( fgMap );
filter->SetInput( 0,dist );
filter->SetInput( 1,gf );
filter->SetLargestCellRadius( 2.0 ); // real coordinates
filter->Update();

std::map< float, ImageIndexType > seeds = filter->seeds;
```

4.2 Results

The results in this example can be obtained by using `SeedExtraction2D.cxx` on the fore image `30-fd.png`, foreground image `30-fg.png` and correlation image `30-gf.png` using the parameter settings given below. The output is written out in the images `30-feature.mha` and `30-dist.mha`. Potential users of this code can execute the example with the following command line:

```
./seedExtract 30-fg.mha 30-feature.mha 30-gf.mha 30-seeds.txt

//Filter settings:
filter->SetLargestCellRadius( 2.0 ); // real coordinates
```



(a) (b)

Figure 4: (a) Original image. (b) Seed points generated.

5 Level-set Segmentation

We use the active contour formulation with shape priors developed by Leventon *et al.* [2] for nuclei segmentation. In this formulation, the evolution is given by the following pde:

$$\frac{\partial \psi}{\partial t} = f(I)(\alpha c + \beta \kappa)|\nabla \psi| + \gamma \nabla f.\nabla \psi + \delta(\psi^* - \psi) \tag{3}$$

where ψ is the level-set function and the zero level-set corresponds to the segmentation curve/surface. The first terms include propagation, curvature and advection terms. The final term $(\psi^* - \psi)$ is a recent addition by Leventon *et al.* [2] wherein they incorporated principal components of the segmentation shape model to drive the update equation. The surface ψ^* is the maximum a-posteriori shape given the current segmentation and image information. The parameters α, β, γ are user-defined settings for the relative scaling of the three speeds.

`m_PropagationScaling` - Expansion weight α in the range [0,1].
`m_CurvatureScaling` - Smoothness weight β in the range [0,1].
`m_AdvectionScaling` - Boundary attraction term γ in the range [0,1].

Note that we typically want the level-set to converge and not escape out of the object. Hence, we specify higher values of γ and β relative to α.

Since, we have a large number of cells and each cell has a small finite size, it is encapsulated in a ROI. The size of the ROI, once again, depends on the largest cell diameter(*m_LargestCellRadius*).

`m_LargestCellRadius` - Largest typical nuclei radius. Usually 5-8μm.

Shape Model: Training data consisting of manually segmented nuclei from 3*D* images are used in estimating a PCA-based shape model with user-chosen number of PCA modes of variation. The advantage of using this model is that a good number of cells within a phenotype share similar sizes, shapes and even orientations. A further discussion of the shape parameters is available in the section on level-sets in the Insight Toolkit [1].

`m_ShapePriorScaling` - Shape scaling parameters γ

`m_SeedValue` - Seed value for initializing level-set. Usually 1/3rd the typical nucleus diameter.
`m_Iterations` - User chosen number of maximum iterations. Typically, 100 iterations are sufficient.
`m_MaxRMSChange` - Maximum permissible RMS change to halt evolution.
`m_NumberOfPCAModes` - Number of PCA modes chosen, say 2 or 3 depending on the shape model.

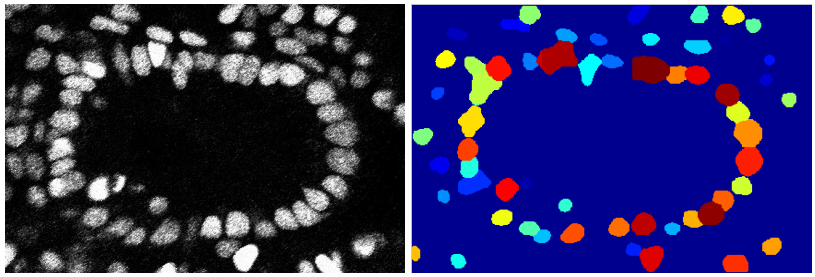
5.1 Usage

```
const unsigned int Dimension = 2;
typedef itk::ShapeLevelSetBasedCellSegmentation< FeatureImage,CompImage >  FilterType;

...

std::vector< FeatureImage::Pointer > shapeModeImages( numberOfPCAModes );
for( int k = 0; k < numberOfPCAModes; k++)
{
  ReaderType::Pointer shapeModeReader = ReaderType::New();
  shapeModeReader->SetFileName(shapeModeFileNames[k].c_str());
  shapeModeReader->Update();
  shapeModeImages[k] = shapeModeReader->GetOutput();
}

...

FilterType::Pointer filter = FilterType::New();
filter->SetInput( featureImg );
filter->SetLargestCellRadius( 6.0 ); // in real coordinates
filter->SetSeedValue( 1.5 );
filter->SetIterations( 1000 );
filter->SetPropagationScaling( 3 );
filter->SetCurvatureScaling( 1 );
filter->SetAdvectionScaling( 3 );
filter->SetMaxRMSChange( 0.01 );
filter->SetShapePriorScaling( 2.0 );
filter->SetNumberOfPCAModes( numberOfPCAModes );

filter->m_meanShape = meanShapeReader->GetOutput();
filter->m_shapeModeImages = shapeModeImages;
```

5.2 Results

The results in this example can be obtained by using `LevelsetBasedCellSegmentation2D.cxx` on the feature image `30-feature.png`, seed image `30-seeds.txt` using the parameter settings given below. The output is written out in the images `30-levelset.mha`. Potential users of this code can execute the example with the following command line:

```
./cellSegment2D 30-feature.mha 30-seeds.txt 30-levelset.mha

//Filter settings:
filter->SetLargestCellRadius( 5.0 ); // in real coordinates
filter->SetSeedValue( 1.5 );
filter->SetIterations( 500 );
filter->SetPropagationScaling( 2 );
filter->SetCurvatureScaling( 1 );
filter->SetAdvectionScaling( 4 );
filter->SetMaxRMSChange( 0.01
```



(a) (b)

Figure 5: (a) Original image. (b) Level-set segmentations.

6 Voronoi Refinement

The level-set evolution partitions the foreground into individual nuclei. Their evolution is guided by the speed function. The affiliation of minute fragments of the nuclear foreground remains unresolved. In order to resolve the affiliation, we assign pixels in a given connected component to the nearest cell available in that component. Some of the unresolved pixels may also come from components that were too small to represent cells. These components are eliminated by using a size threshold (*m_MinComponentSize*). The filter `itk::SeedExtraction` is templated over the Feature and Foreground Image Types. It has a single parameter.

`m_MinComponentSize` - Smallest component size permissible in the given dataset. Set by user to remove spurious components.

6.1 Usage

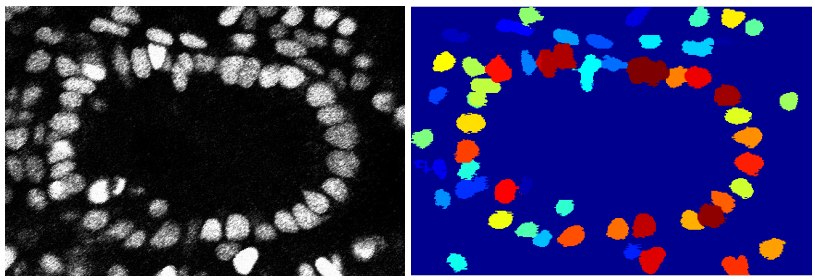
```
const int Dimension = 3;
```

```
typedef itk::VoronoiBasedCellSegmentation< ImageType,ImageType > FilterType;

...

FilterType::Pointer filter = FilterType::New();
filter->SetInput( segmentImg );
filter->SetForegroundImage( fgMap );
filter->SetMinComponentSize( 1000 );
filter->Update();
```

6.2 Results

The results in this example can be obtained by using `VoronoiBasedCellSegmentation2D.cxx` on the level-set image `30-levelset.mha` using the parameter settings given below. The output is written out in the images `30-voronoi.mha` which is the final segmentation. Potential users of this code can execute the example with the following command line:

```
./voronoiSegment2D 30-fg.mha 30-levelset.mha 30-voronoi.mha

//Filter settings:
filter->SetMinComponentSize( 400 );
```



(a) (b)

Figure 6: (a) Original image. (b) Refined segmentations.

7 Cell Statistics Computation

The filter `itk::CellStatistics` computes relevant statistical/shape parameters of each individual segmented cell such as centroid, intensity-weighted centroid, standard deviation of intensities, average intensity, principal directions of the shape etc. It also relabels the cells to have consecutive labels. The filter is templated over the Segmentation Image Type and has no parameters.

7.1 Usage

```
const unsigned int Dimension = 3;
typedef itk::CellStatistics< ImageType,ImageType > FilterType;
```

```
...

FilterType::Pointer filter = FilterType::New();
filter->SetInput( segmentationInput );
filter->SetRawImage( rawImg );
filter->Update();
```

7.2 Results

The results in this example can be obtained by using `CellStatistics2D.cxx` on the segmentation image `30-voronoi.mha` using the parameter settings given below. The tabbed output is written out on standard output which can be redirected to a text file. The output image (`30-segment.mha`) consists of relabeled cells after deleting cells below a threshold size. Potential users of this code can execute the example with the following command line:

```
./cellStatistics2D 30-voronoi.mha 30.png 30-segment.mha

//Filter settings:
filter->SetMinComponentSize( 400 );
```

7.3 Acknowledgments

These classes use the Insight Journal contribution from Lehmann G. named ”Binary Attribute Morphology”. A version of this contribution is included in the code distributed along with this paper. Only the CMakeFiles have been modified to fit into our build framework. More recent version of this contribution might be found here: http://hdl.handle.net/1926/584.

This contribution also use some of the class in the Review directory of ITK. You need to compile ITK with ITK_USE_REVIEW ON to be able to compile our code.

References

[1] L. Ibanez, W. Schroeder, L. Ng, and J. Cates. *The ITK Software Guide*. Kitware, Inc. ISBN 1-930934-10-6, http://www.itk.org/ItkSoftwareGuide.pdf, first edition, 2003.

[2] M. Leventon, W. Grimson, and O. Faugeras. Statistical shape influences in geodesic active contours. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 316–232, 2000.

[3] K. Mosaliganti, L. Cooper, R. Sharp, R. Machiraju, G. Leone, K. Huang, and J. Saltz. Reconstruction of cellular biological structures from optical microscopy data. *IEEE Transactions on Visualization and Computer Graphics*, 14(4):863–876, 2008.