HistomicsTK Documentation

Release 0.1.0

Kitware, Inc.

Sep 12, 2016

Contents

HistomicsTK is a Python and REST API for the analysis of Histopathology images. Developed in coordination with the Digital Slide Archive, HistomicTK is intended for consumers of histopathology workflows and algorithms developers alike. It provides algorithms for fundamental image analysis tasks, algorithm pipelines for common histopathology workflows, and a framework for the easy distribution and integration of algorithms and pipelines.

To see examples of HistomicsTK's capabilities, check out the algorithm and pipeline examples (link) or visit the algorithm library (link) to learn about the filtering, normalization, segmentation and feature extraction capabilities.

Installing HistomicsTK is easy with pip. See the :doc:installation page for more details.

Integrating your algorithms into HistomicsTK and the Digital Slide Archive is made simple with Docker and the Slicer Execution Model. This framework gives developers the freedom to create portable algorithms and automatically generate DSA UI elements for their algorithm, and exposes their work to a broad community of users. Read more about this in the :doc:api-docs.

CHAPTER 1

Installation

At the command line type:

```
$ pip install histomicstk
```

Or, if you have virtualenvwrapper installed:

```
$ mkvirtualenv histomicstk
$ pip install histomicstk
```

CHAPTER 2

Usage

To use histomicstk in a project:

```
import histomicstk
```

CHAPTER 3

API Documentation

3.1 histomicstk

3.1.1 histomicstk.features

This package contains functions to computing a variety of image-based features that quantify the appearance and/or morphology of an objects/regions in the image. These are needed for classifying objects (e.g. nuclei) and regions (e.g. tissues) found in histopathology images.

histomicstk.features.**ComputeFSDFeatures**(*im_label*, *K=128*, *Fs=6*, *Delta=8*, *rprops=None*)

Calculates *Fourier shape descriptors* for each objects.

Parameters

- **im_label** (*array_like*) – A labeled mask image wherein intensity of a pixel is the ID of the object it belongs to. Non-zero values are considered to be foreground objects.
- **K** (*int, optional*) – Number of points for boundary resampling to calculate fourier descriptors. Default value = 128.
- **Fs** (*int, optional*) – Number of frequency bins for calculating FSDs. Default value = 6.
- **Delta** (*int, optional*) – Used to dilate nuclei and define cytoplasm region. Default value = 8.
- **rprops** (*output of skimage.measure.regionprops, optional*) – rprops = skimage.measure.regionprops(im_label). If rprops is not passed then it will be computed inside which will increase the computation time.

Returns **fdata** – object/label.

Return type Pandas data frame containing the FSD features for each

References

histomicstk.features.**ComputeGradientFeatures**(*im_label*, *im_intensity*, *num_hist_bins=10*, *rprops=None*)

Calculates gradient features from an intensity image.

Parameters

- **im_label** (*array_like*) – A labeled mask image wherein intensity of a pixel is the ID of the object it belongs to. Non-zero values are considered to be foreground objects.

- **im_intensity** (*array_like*) – Intensity image
- **num_hist_bins** (*int, optional*) – Number of bins used to computed the gradient histogram of an object. Histogram is used to energy and entropy features. Default is 10.
- **rprops** (*output of skimage.measure.regionprops, optional*) – rprops = skimage.measure.regionprops(im_label). If rprops is not passed then it will be computed inside which will increase the computation time.

Returns **fdata** – A pandas dataframe containing the gradient features listed below for each object/label.

Return type pandas.DataFrame

Notes

List of gradient features computed by this function:

Gradient.Mag.Mean [float] Mean of gradient data.

Gradient.Mag.Std [float] Standard deviation of gradient data.

Gradient.Mag.Skewness [float] Skewness of gradient data. Value is 0 when all values are equal.

Gradient.Mag.Kurtosis [float] Kurtosis of gradient data. Value is -3 when all values are equal.

Gradient.Mag.HistEnergy [float] Energy of the gradient magnitude histogram of object pixels

Gradient.Mag.HistEnergy [float] Entropy of the gradient magnitude histogram of object pixels.

Gradient.Canny.Sum [float] Sum of canny filtered gradient data.

Gradient.Canny.Mean [float] Mean of canny filtered gradient data.

References

histomicstk.features.**ComputeHaralickFeatures**(*im_label*, *im_intensity*, *offsets=None*, *num_levels=None*, *gray_limits=None*, *rprops=None*)

Calculates 26 Haralick texture features for each object in the given label mask.

These features are derived from gray-level co-occurence matrix (GLCM) that is a two dimensional histogram containing the counts/probabilities of co-occurring intensity values with a given neighborhood offset in the region occupied by an object in the image.

Parameters

- **im_label** (*array_like*) – An ND labeled mask image wherein intensity of a pixel is the ID of the object it belongs to. Non-zero values are considered to be foreground objects.
- **im_intensity** (*array_like*) – An ND single channel intensity image
- **offsets** (*array_like, optional*) – A (num_offsets, num_image_dims) array of offset vectors specifying the distance between the pixel-of-interest and its neighbor. Note that the first dimension corresponds to the rows.

  See *histomicstk.features.graycomatrixext* for more details.
- **num_levels** (*unsigned int, optional*) – An integer specifying the number of gray levels For example, if *NumLevels* is 8, the intensity values of the input image are scaled so they are integers between 1 and 8. The number of gray levels determines the size of the gray-level co-occurrence matrix.

Default: 2 for binary/logical image, 32 for numeric image

- **gray_limits** (*array_like, optional*) – A two-element array specifying the desired input intensity range. Intensity values in the input image will be clipped into this range.

  Default: [0, 1] for boolean-valued image, [0, 255] for integer-valued image, and [0.0, 1.0] for-real valued image

Returns **fdata** – A pandas dataframe containing the haralick features.

Return type pandas.DataFrame

Notes

This function computes the following list of haralick features derived from normalized GLCMs (P) of the given list of neighborhood offsets:

Haralick.ASM.Mean, Haralick.ASM.Range [float] Mean and range of the angular second moment (ASM) feature for GLCMS of all offsets. It is a measure of image homogeneity and is computed as follows:

$$ASM = \sum_{i,j=0}^{levels-1} p(i,j)^2$$

Haralick.Contrast.Mean, Haralick.Contrast.Range [float] Mean and range of the Contrast feature for GLCMs of all offsets. It is a measure of the amount of variation between intensities of neighboiring pixels. It is equal to zero for a constant image and increases as the amount of variation increases. It is computed as follows:

$$Contrast = \sum_{i,j=0}^{levels-1} (i-j)^2 p(i,j)$$

Haralick.Correlation.Mean, Haralick.Correlation.Range [float] Mean and range of the Correlation feature for GLCMs of all offsets. It is a measure of correlation between the intensity values of neighboring pixels. It is computed as follows:

$$Correlation = \sum_{i,j=0}^{levels-1} p(i,j) \left[\frac{(i-\mu_i)(j-\mu_j)}{\sigma_i \sigma_j} \right]$$

Haralick.SumOfSquares.Mean, Haralick.SumOfSquares.Range [float] Mean and range of the SumOfSquares feature for GLCMs of all offsets. It is a measure of variance and is computed as follows:

$$SumofSquare = \sum_{i,j=0}^{levels-1} (i-\mu)^2 p(i,j)$$

Haralick.IDM.Mean, Haralick.IDM.Range [float] Mean and range of the inverse difference moment (IDM) feature for GLCMS of all offsets. It is a measure of homogeneity and is computed as follows:

$$IDM = \sum_{i,j=0}^{levels-1} \frac{1}{1+(i-j)^2} p(i,j)$$

Haralick.SumAverage.Mean, Haralick.SumAverage.Range [float] Mean and range of sum average feature for GLCMs of all offsets. It is computed as follows:

$$SumAverage = \sum_{k=2}^{2levels} kp_{x+y}(k), \qquad where$$

$$p_{x+y}(k) = \sum_{i,j=0}^{levels-1} \delta_{i+j,k} p(i,j)$$

$$\delta_{m,n} = \begin{cases} 1 & \text{when } m = n \\ 0 & \text{when } m \neq n \end{cases}$$

Haralick.SumVariance.Mean, Haralick.SumVariance.Variance [float] Mean and range of sum variance feature for the GLCMS of all offsets. It is computed as follows:

$$SumVariance = \sum_{k=2}^{2levels} (k - SumEntropy) p_{x+y}(k)$$

Haralick.SumEntropy.Mean [float] Mean and range of the sum entropy features for GLCMS of all offsets. It is computed as follows:

$$SumEntropy = -\sum_{k=2}^{2levels} p_{x+y}(k) \log(p_{x+y}(k))$$

Haralick.Entropy.Mean, Haralick.Entropy.Range [float] Mean and range of the entropy features for GLCMs of all offsets. It is computed as follows:

$$Entropy = -\sum_{i,j=0}^{levels-1} p(i,j) \log(p(i,j))$$

Haralick.DifferenceVariance.Mean, Haralick.DifferenceVariance.Range [float] Mean and Range of the difference variance feature of GLCMs of all offsets. It is computed as follows:

$$DifferenceVariance = \text{variance of } p_{x-y}, \qquad where$$

$$p_{x-y}(k) = \sum_{i,j=0}^{levels-1} \delta_{|i-j|,k} p(i,j)$$

Haralick.DifferenceEntropy.Mean [float] Mean and range of the difference entropy feature for GLCMS of all offsets. It is computed as follows:

$$DifferenceEntropy = \text{entropy of } p_{x-y}$$

Haralick.IMC1.Mean, Haralick.IMC1.Range [float] Mean and range of the first information measure of cor-

relation feature for GLCMs of all offsets. It is computed as follows:

$$IMC1 = \frac{HXY - HXY1}{\max(HX, HY)}, \qquad where$$

$$HXY = -\sum_{i,j=0}^{levels-1} p(i,j)\log(p(i,j))$$

$$HXY1 = -\sum_{i,j=0}^{levels-1} p(i,j)\log(p_x(i)p_y(j))$$

$$HX = -\sum_{i=0}^{levels-1} p_x(i)\log(p_x(i))$$

$$HY = -\sum_{j=0}^{levels-1} p_y(j)\log(p_y(j))$$

$$p_x(i) = \sum_{j=1}^{levels} p(i,j)$$

$$p_y(j) = \sum_{j=1}^{levels} p(i,j)$$

Haralick.IMC2.Mean, Haralick.IMC2.Range [float] Mean and range of the second information measure of correlation feature for GLCMs of all offsets. It is computed as follows:

$$IMC2 = [1 - \exp(-2(HXY2 - HXY))]^{1/2}, \qquad where$$

$$HXY2 = -\sum_{i,j=0}^{levels-1} p_x(i)p_y(j)\log(p_x(i)p_y(j))$$

References

histomicstk.features.**ComputeIntensityFeatures**(*im_label*, *im_intensity*, *num_hist_bins=10*, *rprops=None*)

Calculates intensity features from an intensity image.

Parameters

- **im_label** (*array_like*) – A labeled mask image wherein intensity of a pixel is the ID of the object it belongs to. Non-zero values are considered to be foreground objects.
- **im_intensity** (*array_like*) – Intensity image.
- **num_hist_bins** (*int, optional*) – Number of bins used to computed the intensity histogram of an object. Histogram is used to energy and entropy features. Default is 10.

- **rprops** (*output of skimage.measure.regionprops, optional*) – rprops = skimage.measure.regionprops(im_label). If rprops is not passed then it will be computed inside which will increase the computation time.

Returns **fdata** – A pandas dataframe containing the intensity features listed below for each object/label.

Return type pandas.DataFrame

Notes

List of intensity features computed by this function:

Intensity.Min [float] Minimum intensity of object pixels.

Intensity.Max [float] Maximum intensity of object pixels.

Intensity.Mean [float] Mean intensity of object pixels

Intensity.Median [float] Median intensity of object pixels

Intensity.MeanMedianDiff [float] Difference between mean and median intensities of object pixels.

Intensity.Std [float] Standard deviation of the intensities of object pixels

Intensity.IQR: float Inter-quartile range of the intensities of object pixels

Intensity.MAD: float Median absolute deviation of the intensities of object pixels

Intensity.Skewness [float] Skewness of the intensities of object pixels. Value is 0 when all intensity values are equal.

Intensity.Kurtosis [float] Kurtosis of the intensities of object pixels. Value is -3 when all values are equal.

Intensity.HistEnergy [float] Energy of the intensity histogram of object pixels

Intensity.HistEntropy [float] Entropy of the intensity histogram of object pixels.

References

histomicstk.features.**ComputeMorphometryFeatures**(*im_label*, *rprops=None*)

Calculates morphometry features for each object

Parameters

- **im_label** (*array_like*) – A labeled mask image wherein intensity of a pixel is the ID of the object it belongs to. Non-zero values are considered to be foreground objects.
- **rprops** (*output of skimage.measure.regionprops, optional*) – rprops = skimage.measure.regionprops(im_label). If rprops is not passed then it will be computed inside which will increase the computation time.

Returns **fdata** – A pandas dataframe containing the morphometry features for each object/label listed below.

Return type pandas.DataFrame

Notes

List of morphometry features computed by this function:

Size.Area [int] Number of pixels the object occupies.

Size.MajorAxisLength [float] The length of the major axis of the ellipse that has the same normalized second central moments as the object.

Size.MinorAxisLength [float] The length of the minor axis of the ellipse that has the same normalized second central moments as the region.

Size.Perimeter [float] Perimeter of object which approximates the contour as a line through the centers of border pixels using a 4-connectivity.

Shape.Circularity: float A measure of how similar the shape of an object is to the circle

Shape.Eccentricity [float] A measure of aspect ratio computed to be the eccentricity of the ellipse that has the same second-moments as the object region. Eccentricity of an ellipse is the ratio of the focal distance (distance between focal points) over the major axis length. The value is in the interval [0, 1). When it is 0, the ellipse becomes a circle.

Shape.EquivalentDiameter [float] The diameter of a circle with the same area as the object.

Shape.Extent [float] Ratio of area of the object to its axis-aligned bounding box.

Shape.MinorMajorAxisRatio [float] A measure of aspect ratio. Ratio of minor to major axis of the ellipse that has the same second-moments as the object region

Shape.Solidity [float] A measure of convexity computed as the ratio of the number of pixels in the object to that of its convex hull.

histomicstk.features.**ComputeNucleiFeatures**(*im_label*, *im_nuclei*, *im_cytoplasm=None*, *fsd_bnd_pts=128*, *fsd_freq_bins=6*, *cyto_width=8*, *num_glcm_levels=32*, *morphometry_features_flag=True*, *fsd_features_flag=True*, *intensity_features_flag=True*, *gradient_features_flag=True*, *haralick_features_flag=True*)

Calculates features for nuclei classification

Parameters

- **im_label** (*array_like*) – A labeled mask image wherein intensity of a pixel is the ID of the object it belongs to. Non-zero values are considered to be foreground objects.
- **im_nuclei** (*array_like*) – Nucleus channel intensity image.
- **im_cytoplasm** (*array_like*) – Cytoplasm channel intensity image.
- **fsd_bnd_pts** (*int, optional*) – Number of points for boundary resampling to calculate fourier descriptors. Default value = 128.
- **fsd_freq_bins** (*int, optional*) – Number of frequency bins for calculating FSDs. Default value = 6.
- **cyto_width** (*float, optional*) – Estimated width of the ring-like neighborhood region around each nucleus to be considered as its cytoplasm. Default value = 8.
- **num_glcm_levels** (*int, optional*) – An integer specifying the number of gray levels For example, if *NumLevels* is 32, the intensity values of the input image are scaled so

they are integers between 0 and 31. The number of gray levels determines the size of the gray-level co-occurrence matrix.

Default: 32

- **morphometry_features_flag** (*bool, optional*) – A flag that can be used to specify whether or not to compute morphometry (size and shape) features. See histomicstk.features.ComputeMorphometryFeatures for more details.
- **fsd_features_flag** (*bool, optional*) – A flag that can be used to specify whether or not to compute Fouried shape descriptor (FSD) features. See *histomicstk.features.ComputeFSDFeatures* for more details.
- **intensity_features_flag** (*bool, optional*) – A flag that can be used to specify whether or not to compute intensity features from the nucleus and cytoplasm channels. See *histomicstk.features.ComputeFSDFeatures* for more details.
- **gradient_features_flag** (*bool, optional*) – A flag that can be used to specify whether or not to compute gradient/edge features from intensity and cytoplasm channels. See *histomicstk.features.ComputeGradientFeatures* for more details.
- **haralick_features_flag** (*bool, optional*) – A flag that can be used to specify whether or not to compute haralick features from intensity and cytoplasm channels. See *histomicstk.features.ComputeHaralickFeatures* for more details.

Returns **fdata** – A pandas data frame containing the features listed below for each object/label

Return type pandas.DataFrame

Notes

List of features computed by this function

Morphometry (size and shape) features of the nuclei See histomicstk.features.ComputeMorphometryFeatures for more details. Feature names prefixed by *Size.* or *Shape.*.

Fourier shape descriptor features See *histomicstk.features.ComputeFSDFeatures* for more details. Feature names are prefixed by *FSD*.

Intensity features for the nucleus and cytoplasm channels See *histomicstk.features.ComputeFSDFeatures* for more details. Feature names are prefixed by *Nucleus.Intensity.* for nucleus features and *Cytoplasm.Intensity.* for cytoplasm features.

Gradient/edge features for the nucleus and cytoplasm channels See *histomicstk.features.ComputeGradientFeatures* for more details. Feature names are prefixed by *Nucleus.Gradient.* for nucleus features and *Cytoplasm.Gradient.* for cytoplasm features.

Haralick features for the nucleus and cytoplasm channels See *histomicstk.features.ComputeHaralickFeatures* for more details. Feature names are prefixed by *Nucleus.Haralick.* for nucleus features and *Cytoplasm.Haralick.* for cytoplasm features.

See also:

histomicstk.features.ComputeMorphometryFeatures(), *histomicstk.features.ComputeFSDFeatu*
histomicstk.features.ComputeIntensityFeatures(), *histomicstk.features.ComputeGradientFe*
histomicstk.features.ComputeHaralickFeatures()

histomicstk.features.**graycomatrixext**(*im_input*, *im_roi_mask=None*, *offsets=None*, *num_levels=None*, *gray_limits=None*, *symmetric=False*, *normed=False*, *exclude_boundary=False*)

Computes gray-level co-occurence matrix (GLCM) within a region of interest (ROI) of an image. GLCM is a

2D histogram/matrix containing the counts/probabilities of co-occuring intensity values at a given offset within an ROI of an image.

Read the documentation to know the default values used for each of the optional parameter in different scenarios.

Parameters

- **im_input** (*array_like*) – Input single channel intensity image
- **im_roi_mask** (*array_like, optional*) – A binary mask specifying the region of interest within which to compute the GLCM. If not specified GLCM is computed for the the entire image.

  Default: None
- **offsets** (*array_like, optional*) – A (num_offsets, num_image_dims) array of offset vectors specifying the distance between the pixel-of-interest and its neighbor. Note that the first dimension corresponds to the rows.

  Because this offset is often expressed as an angle, the following table lists the offset values that specify common angles for a 2D image, given the pixel distance D.

Angle (deg)	offset [y, x]
0	[0 D]
45	[-D D]
90	[-D 0]
135	[-D -D]

  Default - 1D: np.array([1]) - 2D : numpy.array([[1, 0], [0, 1], [1, 1], [1, -1]]) - 3D and higher: numpy.identity(num_image_dims)
- **num_levels** (*unsigned int, optional*) – An integer specifying the number of gray levels For example, if *NumLevels* is 8, the intensity values of the input image are scaled so they are integers between 1 and 8. The number of gray levels determines the size of the gray-level co-occurrence matrix.

  Default: 2 for binary/logical image, 32 for numeric image
- **gray_limits** (*array_like, optional*) – A two-element array specifying the desired input intensity range. Intensity values in the input image will be clipped into this range.

  Default: [0, 1] for boolean-valued image, [0, 255] for integer-valued image, and [0.0, 1.0] for-real valued image
- **symmetric** (*bool, optional*) – A boolean value that specifies whether or not the ordering of values in pixel pairs is considered while creating the GLCM matrix.

  For example, if *Symmetric* is True, then while calculating the number of times the value 1 is adjacent to the value 2, both 1,2 and 2,1 pairings are counted. GLCM created in this way is symmetric across its diagonal.

  Default: False
- **normed** (*bool, optional*) – A boolean value specifying whether or not to normalize glcm.

  Default: False
- **exclude_boundary** (*bool, optional*) – Specifies whether or not to exclude a pixel-pair if the neighboring pixel in the pair is outside *im_roi_mask*. Has an effect only when *im_roi_mask* is specified.

  Default: False

Returns **glcm** – num_levels x num_levels x num_offsets array containing the GLCM for each offset.

Return type array_like

References

3.1.2 histomicstk.filters

This package contains functions for enhancing different kinds of structures (e.g. edges/membrane, blobs/nuclei, vessels) in images.

histomicstk.filters.edge

This package contains functions to enhance edges in images.

histomicstk.filters.edge.**GaussianGradient**(*I*, *Sigma*)

Performs smoothing with derivative gaussian kernel.

Uses seperable convolution to simultaneously smooth and calculate the gradient of a grayscale image.

Parameters

- **I** (*array_like*) – An intensity image.
- **sSigma** (*double*) – Standard deviation of smoothing kernel used in gradient calculation.

Returns

- **dX** (*array_like*) – An intensity image of the X gradient component.
- **dY** (*array_like*) – An intensity image of the Y gradient component.

Notes

Return values are returned as a namedtuple

histomicstk.filters.shape

This package contains functions to enhance and/or detect objects of different shapes (e.g. blobs, vessels)

histomicstk.filters.shape.**cLoG**(*I*, *Mask*, *SigmaMin=42.419999999999995*, *SigmaMax=70.7*)

Constrainted Laplacian of Gaussian filter.

Takes as input a grayscale nuclear image and binary mask of cell nuclei, and uses the distance transform of the nuclear mask to constrain the LoG filter response of the image for nuclear seeding. Returns a LoG filter image of type float. Local maxima are used for seeding cells.

Parameters

- **I** (*array_like*) – A hematoxylin intensity image obtained from ColorDeconvolution. Objects are assumed to be dark with a light background.
- **Mask** (*array_like*) – A binary image where nuclei pixels have value 1/True, and non-nuclear pixels have value 0/False.
- **SigmaMin** (*float*) – A scalar defining the minimum scaled nuclear radius. Radius is scaled by sqrt(2). Default value = 30 * 2 ** 0.5.

- **SigmaMax** (*float*) – A scalar defining the maximum scaled nuclear radius. Radius is scaled by sqrt(2). Default value = 50 * 2 ** 0.5.

Returns **Iout** – A color image of type unsigned char where boundary pixels take on the color defined by the RGB-triplet 'Color'.

Return type array_like

References

histomicstk.filters.shape.**gLoG**(*I*, *Alpha=1*, *Range=array([1.5, 1.6875, 1.875, 2.0625, 2.25, 2.4375, 2.625, 2.8125, 3.])*, *Theta=0.7853981633974483*, *Tau=0.6*, *Eps=0.6*)

Performs generalized Laplacian of Gaussian blob detection.

Parameters

- **I** (*array_like*) – A hematoxylin intensity image obtained from ColorDeconvolution.
- **Alpha** (*double*) – A positive scalar used to normalize the gLoG filter responses. Controls the blob-center detection and eccentricities of detected blobs. Larger values emphasize more eccentric blobs. Default value = 1.
- **Range** (*array_like*) –
- **Theta** (*double*) – Angular increment for rotating gLoG filters. Default value = np.pi / 6.
- **Tau** (*double*) – Tolerance for counting pixels in determining optimal scale SigmaC
- **Eps** (*double*) – Range to define SigmaX surrounding SigmaC

Returns

- **Rsum** (*array_like*) – Sum of filter responses at specified scales and orientations
- **Maxima** (*: array_like*) – A binary mask highlighting maxima pixels

Notes

Return values are returned as a namedtuple

References

histomicstk.filters.shape.**Vesselness**(*I*, *Sigma*)

Calculates vesselness measure for grayscale image *I* at scale *Sigma*. Also returns eigenvalues and vectors used for vessel salience filters.

Parameters

- **I** (*array_like*) – M x N grayscale image.
- **Sigma** (*double*) – standard deviation of gaussian kernel.

Returns

- **Deviation** (*array_like*) – M x N image of deviation from blob
- **Frobenius** (*array_like*) – M x N image of frobenius norm of Hessian - measures presence of structure.
- **E** (*array_like*) – M x N x 2 eigenvalue image - see Eigenvalues.py.

- **Theta** (*array_like*) – M x N eigenvector angle image for E(:,:,0) in radians see Eigenvalues.py. Oriented parallel to vessel structures.

References

3.1.3 histomicstk.preprocessing

This package contains functions to pre-process histopathology images.

histomicstk.preprocessing.color_conversion

This package contains utility functions to convert images between different color spaces.

histomicstk.preprocessing.color_conversion.**lab_mean_std**(*im_input*)

Computes the mean and standard deviation of the intensities of each channel of the given RGB image in LAB color space. The outputs of this function are needed for reinhard color normalization.

Parameters **im_input** (*array_like*) – An RGB image

Returns

- **mean_lab** (*array_like*) – A 3-element array containing the mean of each channel of the input RGB in LAB color space.
- **std_lab** (*array_like*) – A 3-element array containing the standard deviation of each channel of the input RGB in LAB color space.

See also:

histomicstk.preprocessing.color_conversion.rgb_to_lab(), histomicstk.preprocessing.color_conversion.reinhard()

References

histomicstk.preprocessing.color_conversion.**lab_to_rgb**(*im_lab*)

Transforms an image from LAB to RGB color space

Parameters **im_lab** (*array_like*) – An image in LAB color space

Returns **im_rgb** – The RGB representation of the input image 'im_lab'.

Return type array_like

See also:

histomicstk.preprocessing.color_conversion.rgb_to_lab(), *histomicstk.preprocessing.color_normalization.reinhard()*

References

histomicstk.preprocessing.color_conversion.**od_to_rgb**(*im_od*)

Transforms input optical density image *im_od* into RGB space

Parameters **im_od** (*array_like*) – A floating-point image of optical density values obtained from rgb_to_od.

Returns **im_rgb** – A floating-point multi-channel image with intensity values in the range [0, 255].

Return type array_like

See also:

histomicstk.preprocessing.color_conversion.rgb_to_od(),
histomicstk.preprocessing.color_deconvolution.ColorDeconvolution(),
histomicstk.preprocessing.color_deconvolution.ColorConvolution()

histomicstk.preprocessing.color_conversion.**rgb_to_lab**(*im_rgb*)
Transforms an image from RGB to LAB color space

Parameters **im_rgb** (*array_like*) – An RGB image

Returns **im_lab** – LAB representation of the input image *im_rgb*.

Return type array_like

See also:

histomicstk.preprocessing.color_conversion.lab_to_rgb(),
histomicstk.preprocessing.color_normalization.reinhard()

References

histomicstk.preprocessing.color_conversion.**rgb_to_od**(*im_rgb*)
Transforms input RGB image *im_rgb* into optical density space for color deconvolution.

Parameters **im_rgb** (*array_like*) – A floating-point RGB image with intensity ranges of [0, 255].

Returns **im_od** – A floating-point image of corresponding optical density values.

Return type array_like

See also:

histomicstk.preprocessing.color_conversion.od_to_rgb(),
histomicstk.preprocessing.color_deconvolution.ColorDeconvolution(),
histomicstk.preprocessing.color_deconvolution.ColorConvolution()

histomicstk.preprocessing.color_deconvolution

This package contains implementation of methods to deconvolve or separate the stains of histopathology images.

histomicstk.preprocessing.color_deconvolution.**ColorConvolution**(*I*, *W*)
Performs Color Convolution Reconstructs a color image from the stain matrix *W* and the individual images stored as channels in *I* and generated by ColorDeconvolution.

Parameters

- **I** (*array_like*) – An RGB image where in each channel contains image of one stain
- **W** (*array_like*) – A 3x3 matrix containing the stain colors in its columns. In the case of two stains, the third column is zero and will be complemented using cross-product. The matrix should contain a minumum two nonzero columns.

Returns **IOut** – Reconstructed RGB image with intensity values ranging from [0, 255], suitable for display.

Return type array_like

See also:

histomicstk.preprocessing.color_deconvolution.ComplementStainMatrix(),
histomicstk.preprocessing.color_deconvolution.ColorDeconvolution(),
histomicstk.preprocessing.color_conversion.rgb_to_od(),
histomicstk.preprocessing.color_conversion.od_to_rgb()

histomicstk.preprocessing.color_deconvolution.**ColorDeconvolution**(*I*, *W*)

Performs color deconvolution. The given RGB Image *I* is first first transformed into optical density space, and then projected onto the stain vectors in the columns of the 3x3 stain matrix *W*.

For deconvolving H&E stained image use:

W = array([[0.650, 0.072, 0], [0.704, 0.990, 0], [0.286, 0.105, 0]])

Parameters

- **I** (*array_like*) – Input RGB Image that needs to be deconvolved.
- **W** (*array_like*) – A 3x3 matrix containing the color vectors in columns. For two stain images the third column is zero and will be complemented using cross-product. Atleast two of the three columns must be non-zero.

Returns

- **Stains** (*array_like*) – An rgb image where in each channel contains the image of the stain of the corresponding column in the stain matrix *W*. The intensity range of each channel is [0, 255] suitable for displaying.
- **StainsFloat** (*array_like*) – An intensity image of deconvolved stains that is unbounded, suitable for reconstructing color images of deconvolved stains with ColorConvolution.
- **Wc** (*array_like*) – A 3x3 complemented stain matrix. Useful for color image reconstruction with ColorConvolution.

See also:

histomicstk.preprocessing.color_deconvolution.ComplementStainMatrix(),
histomicstk.preprocessing.color_deconvolution.ColorConvolution(),
histomicstk.preprocessing.color_conversion.rgb_to_od(),
histomicstk.preprocessing.color_conversion.od_to_rgb()

histomicstk.preprocessing.color_deconvolution.**ComplementStainMatrix**(*W*)

Generates a complemented stain matrix Used to fill out empty columns of a stain matrix for use with ColorDeconvolution. Replaces right-most column with normalized cross-product of first two columns.

Parameters **W** (*array_like*) – A 3x3 stain calibration matrix with stain color vectors in columns.

Returns **WComp** – A 3x3 complemented stain calibration matrix with a third orthogonal column.

Return type array_like

See also:

histomicstk.preprocessing.color_deconvolution.ColorDeconvolution()

histomicstk.preprocessing.color_deconvolution.**SparseColorDeconvolution**(*I*, *Winit*, *Beta*)

Performs adaptive color deconvolution.

Uses sparse non-negative matrix factorization to adaptively deconvolve a given RGB image into intensity images representing distinct stains. Similar approach to `ColorDeconvolution` but operates adaptively. The input RGB image *I* consisting of RGB values is first transformed into optical density space as a row-matrix, and then is decomposed as $V = WH$ where W is a 3xk matrix containing stain vectors in columns and H is a k x

m*n matrix of concentrations for each stain vector. The system is solved to encourage sparsity of the columns of :math"H i.e. most pixels should not contain significant contributions from more than one stain. Can use a hot-start initialization from a color deconvolution matrix.

Parameters

- **I** (*array_like*) – An RGB image of type unsigned char, or a 3xN matrix of RGB pixel values.
- **Winit** (*array_like*) – A 3xK matrix containing the color vectors in columns. Should not be complemented with ComplementStainMatrix for sparse decomposition to work correctly.
- **Beta** (*double*) – Regularization factor for sparsity of H - recommended 0.5.

Returns

- **Stains** (*array_like*) – An rgb image with deconvolved stain intensities in each channel, values ranging from [0, 255], suitable for display.
- **W** (*array_like*) – The final 3 x k stain matrix produced by NMF decomposition.

Notes

Return values are returned as a namedtuple

See also:

histomicstk.preprocessing.color_deconvolution.ColorDeconvolution()

References

histomicstk.preprocessing.color_normalization

This package contains functions to correct non-uniform staining issues in histopathology images.

histomicstk.preprocessing.color_normalization.**reinhard**(*im_src*, *target_mu*, *target_sigma*, *src_mu=None*, *src_sigma=None*)

Performs Reinhard color normalization to transform the color characteristics of an image to a desired standard.

The standard is defined by the mean and standard deviations of the target image in LAB color space defined by Ruderman. The input image is converted to Ruderman's LAB space, the LAB channels are each centered and scaled to zero-mean unit variance, and then rescaled and shifted to match the target image statistics. If the LAB statistics for the input image are provided (*src_mu* and *src_sigma*) then these will be used for normalization, otherwise they will be derived from the input image *im_src*.

Parameters

- **im_src** (*array_like*) – An RGB image
- **target_mu** (*array_like*) – A 3-element array containing the means of the target image channels in LAB color space.
- **target_sigma** (*array_like*) – A 3-element array containing the standard deviations of the target image channels in LAB color space.
- **src_mu** (*array_like, optional*) – A 3-element array containing the means of the source image channels in LAB color space. Used with ReinhardSample for uniform normalization of tiles from a slide.

- **src_sigma** (*array, optional*) – A 3-element array containing the standard deviations of the source image channels in LAB color space. Used with ReinhardSample for uniform normalization of tiles tiles from a slide.

Returns **im_normalized** – Color Normalized RGB image

Return type array_like

See also:

histomicstk.preprocessing.color_conversion.rgb_to_lab(), *histomicstk.preprocessing.color_conversion.lab_to_rgb()*

References

histomicstk.preprocessing.color_normalization.**ReinhardSample**(*File*, *Magnification*, *Percent*, *Tile*)

Samples a whole-slide-image to determine LAB colorspace statistics (mean, variance) needed to perform global Reinhard color normalization.

Normalizing individual tiles independently creates a significant bias in the results of segmentation and feature extraction, as the color statistics of each tile in a whole-slide image can vary significantly. To remedy this, we sample from the entire whole-slide image in order to obtain the global mean and variance of the LAB colorspace channels that can then be used when processing individual tiles for uniformity. This function can also be used to obtain the global target statistics from an ideal slide that can serve as the normalization standard.

Parameters

- **File** (*str*) – path and filename of slide.
- **Magnification** (*scalar*) – Desired magnification for sampling. Defaults to native scan magnification.
- **Percent** (*double*) – Percentage of pixels to sample (range (0, 1]).
- **Tile** (*int*) – Tile size used in sampling high-resolution image.

Returns

- **TargetMu** (*array_like*) – A 3-element array containing the means of the target image channels in LAB color space.
- **TargetSigma** (*array_like*) – A 3-element list containing the standard deviations of the target image channels in LAB color space.

Notes

Returns a namedtuple.

See also:

histomicstk.preprocessing.color_conversion.rgb_to_lab(), *histomicstk.preprocessing.color_conversion.lab_to_rgb()*

3.1.4 histomicstk.segmentation

This package contains functions for segmenting a variety of objects/structures (e.g. nuclei, tissue, cytoplasm) found in histopathology images.

histomicstk.segmentation.**EmbedBounds**(*I, Bounds, Color=[255, 0, 0]*)

Embeds object boundaries into an RGB color, grayscale or binary image, returning a color rendering of the image and object boundaries.

Takes as input a grayscale or color image, a perimeter mask of object boundaries, and an RGB triplet, and embeds the object boundaries into the input image at the prescribed color. Returns a color RGB image of type unsigned char. If the input image is type double, and has pixels inside range [0, 1], then it will be scaled to the range [0, 255]. Otherwise it will be assumed to be in the range of an unsigned char image.

Parameters

- **I** (*array_like*) – A color or grayscale image.
- **Bounds** (*array_like*) – A binary image where object perimeter pixels have value 1, and non-perimeter pixels have value 0.
- **Color** (*array_like*) – A 1 x 3 array of RGB values in the range [0, 255].

Returns **Iout** – A color image of type unsigned char where boundary pixels take on the color defined by the RGB-triplet 'Color'.

Return type array_like

See also:

histomicstk.segmentation.label.LabelPerimeter()

histomicstk.segmentation.**GraphColorSequential**(*Adjacency*)

Generates a coloring of an adjacency graph using the sequential coloring algorithm. Used to bin regions from a label image into a small number of independent groups that can be processed seperately with algorithms like multilabel graph cuts or individual active contours. The rationale is to color adjacent objects with distinct colors so that their contours can be co-evolved.

Parameters **Adjacency** (*array_like*) – A binary matrix of size N x N, where N is the number of objects in Label. A value of 'True' at Adjacency(i,j) indicates that objects 'i' and 'j' are neighbors. Does not contain entries for background objects.

Returns **Colors** – A list of colors for the objects encoded in 'Adjacency'. No two objects that are connected in 'Adjacency' will share the same color.

Return type array_like

See also:

histomicstk.segmentation.LabelRegionAdjacency(), *histomicstk.segmentation.RegionAdjacenc*

histomicstk.segmentation.**LabelRegionAdjacency**(*Label*, *Neighbors=4*)

Constructs a region adjacency graph for a label image using either 4-neighbor or 8-neighbor connectivity. Background pixels are not included (Label == 0). Not intended to build large graphs from individual pixels.

Parameters

- **Label** (*array_like*) – Label image where positive values (Label > 0) correspond to foreground objects of interest.
- **Neighbors** (*float*) – The neighbor connectivity to use, either '4' or '8'. Default value = 4.

Returns **Adjacency** – A binary matrix of size N x N, where N is the number of objects in Label. A value of 'True' at Adjacency(i,j) indicates that objects 'i' and 'j' are neighbors.

Return type array_like

histomicstk.segmentation.**RegionAdjacencyLayer**(*Adjacency*)

Adds an additional layer of dependence to a region adjacency graph, connecting each node to the neighbors of its immediate neighbors.

Parameters **Adjacency** (*array_like*) – A binary matrix of size N x N, where N is the number of objects in Label. A value of 'True' at Adjacency(i,j) indicates that objects 'i' and 'j' are neighbors.

Returns **Layered** – A version of 'Adjacency' with additional edges to connect 2-neighbors.

Return type array_like

See also:

histomicstk.segmentation.LabelRegionAdjacency()

histomicstk.segmentation.**SimpleMask**(*I*, *BW=2*, *DefaultBGScale=2.5*, *DefaultTissueScale=30*, *MinPeak=10*, *MaxPeak=25*, *Percent=0.1*, *MinProb=0.05*)

Performs segmentation of the foreground (tissue) Uses a simple two-component Gaussian mixture model to mask tissue areas from background in brightfield H&E images. Kernel-density estimation is used to create a smoothed image histogram, and then this histogram is analyzed to identify modes corresponding to tissue and background. The mode peaks are then analyzed to estimate their width, and a constrained optimization is performed to fit gaussians directly to the histogram (instead of using expectation-maximization directly on the data which is more prone to local minima effects). A maximum-likelihood threshold is then derived and used to mask the tissue area in a binarized image.

Parameters

- **I** (*array_like*) – An RGB image of type unsigned char.
- **BW** (*double, optional*) – Bandwidth for kernel density estimation - used for smoothing the grayscale histogram. Default value = 2.
- **DefaultBGScale** (*double, optional*) – Standard deviation of background gaussian to be used if estimation fails. Default value = 2.5.
- **DefaultTissueScale** (*double, optional*) – Standard deviation of tissue gaussian to be used if estimation fails. Default value = 30.
- **MinPeak** (*double, optional*) – Minimum peak width for finding peaks in KDE histogram. Used to initialize curve fitting process. Default value = 10.
- **MaxPeak** (*double, optional*) – Maximum peak width for finding peaks in KDE histogram. Used to initialize curve fitting process. Default value = 25.
- **Percent** (*double, optional*) – Percentage of pixels to sample for building foreground/background model. Default value = 0.10.
- **MinProb** (*double, optional*) – Minimum probability to qualify as tissue pixel. Default value = 0.05.

Returns **Mask** – A binarized version of *I* where foreground (tissue) has value '1'.

Return type array_like

See also:

histomicstk.utils.Sample()

** Sub-packages **

histomicstk.segmentation.label

This package contains functions for post-processing labeled segmentation masks produced by segmentation algorithms.

histomicstk.segmentation.label.**AreaOpenLabel**(*Label*, *Area*)

Removes small objects from label image.

Parameters

- **Label** (*array_like*) – A uint32 type label image generated by segmentation methods.
- **Area** (*int*) – Area threshold for objects. Objects with fewer than 'Area' pixels will be zeroed to merge with background.

Returns **Split** – A uint32 label where objects with pixels < Area are removed.

Return type array_like

Notes

Objects are assumed to have positive nonzero values. Label image will be condensed during processing.

See also:

histomicstk.segmentation.label.CondenseLabel(), *histomicstk.segmentation.label.ShuffleLa*
histomicstk.segmentation.label.SplitLabel(), *histomicstk.segmentation.label.WidthOpenLab*

histomicstk.segmentation.label.**CompactLabel**(*Label*, *Compaction=3*)

Performs a thinning operation on a label image to remove thin protrusions from objects that are in contact with the background. Applies a distance transform and sequentially removes pixels with small distances to background that are not connected to higher distance pixels.

Parameters

- **Label** (*array_like*) – A labeled segmentation mask
- **Compaction** (*int*) – Factor used in compacting objects to remove thin protrusions. Refered to as d in the reference below. Default value = 3.

Notes

Implemented from the reference below.

Returns **Compact** – A labeled segmentation mask with thin protrusions removed.

Return type array_like

See also:

histomicstk.segmentation.label.AreaOpenLabel(), *histomicstk.segmentation.label.Condensel*
histomicstk.segmentation.label.ShuffleLabel(), *histomicstk.segmentation.label.SplitLabel*
histomicstk.segmentation.label.WidthOpenLabel()

References

histomicstk.segmentation.label.**ComputeNeighborhoodMask**(*im_label*, *neigh_width=8*)

Computes a label mask highlighting a ring-like neighborhood of each object or region in a given label mask

Parameters

- **im_label** (*array_like*) – A labeled mask image wherein intensity of a pixel is the ID of the object it belongs to. Non-zero values are considered to be foreground objects.
- **neigh_width** (*float, optional*) – The width of the ring-like neighborhood around each object.

Returns **im_neigh_label** – A labeled mask image highlighting pixels in a ring-like neighborhood of width upto *neigh_width* around each object in the given label mask. The intensity of each pixel in the ring-like neighborhood is set equal to the label of the closest object in the given label mask. other pixels (including the ones inside objects) are set to zero.

Return type array_like

histomicstk.segmentation.label.**CondenseLabel**(*Label*)

Shifts labels in a label image to fill in gaps corresponding to missing values.

Parameters **Label** (*array_like*) – A label image generated by segmentation methods.

Returns **Condensed** – A label image where all values > 0 are shifted down to fill gaps.

Return type array_like

See also:

histomicstk.segmentation.label.ShuffleLabel()

histomicstk.segmentation.label.**DeleteLabel**(*Label*, *Index*)

Deletes objects with values in 'Index' from label image, writing them over with zeros to assimilate with background.

Parameters

- **Label** (*array_like*) – A label image generated by segmentation methods.
- **Index** (*array_like*) – An n-length array of strictly positive integer values to delete from 'Label'.

Returns

- **Deleted** (*array_like*) – A label image where all values in 'Index' are set to zero.
- *Notes*
- ——
- *A call to CondenseLabel can squeeze label image values to fill in gaps from*
- *deleted values.*

See also:

histomicstk.segmentation.label.CondenseLabel()

histomicstk.segmentation.label.**LabelPerimeter**(*L*, *Connectivity=4*)

Converts a label or binary mask image to a binary perimeter image.

Uses 4-neighbor or 8-neighbor shifts to detect pixels whose values do not agree with their neighbors.

Parameters

- **L** (*array_like*) – A label or binary mask image.
- **Connectivity** (*double or int*) – Neighborhood connectivity to evaluate. Valid values are 4 or 8. Default value = 4.

Returns **Mask** – A binary image where object perimeter pixels have value 1, and non-perimeter pixels have value 0.

Return type array_like

See also:

histomicstk.segmentation.EmbedBounds()

histomicstk.segmentation.label.**ShuffleLabel**(*Label*)

Shuffles labels in a label image to improve visualization and enhance object boundaries.

Parameters **Label** (*array_like*) – A label image generated by segmentation methods.

Returns **Shuffled** – A label image where all values > 0 are randomly shuffled.

Return type array_like

See also:

histomicstk.segmentation.label.CondenseLabel()

histomicstk.segmentation.label.**SplitLabel**(*Label*, *Connectivity=8*)

Re-labels objects that have multiple non-contiguous portions to create a new label image where each object is contiguous.

Parameters

- **Label** (*array_like*) – A uint32 type label image generated by segmentation methods.
- **Connectivity** (*int*) – Neighborhood connectivity to define contiguity. Valid values are 4 or 8. Default value = 4.

Notes

Objects are assumed to have positive nonzero values.

Returns **Split** – A uint32 label where discontiguous objects are split and relabeled.

Return type array_like

See also:

histomicstk.segmentation.label.AreaOpenLabel(), *histomicstk.segmentation.label.Condense*
histomicstk.segmentation.label.ShuffleLabel()

histomicstk.segmentation.label.**TraceBounds**(*Mask*, *Connectivity=4*, *XStart=None*, *YStart=None*, *MaxLength=inf*)

Performs exterior boundary tracing of a single object in a binary mask. If a starting point is not provided then a raster scan will be performed to identify the starting pixel.

Parameters

- **Mask** (*array_like*) – A boolean type image where foreground pixels have value 'True', and background pixels have value 'False'.
- **Connectivity** (*int*) – Neighborhood connectivity to evaluate. Valid values are 4 or 8. Default value = 4.
- **XStart** (*int*) – Starting horizontal coordinate to begin tracing. Default value = None.
- **YStart** (*int*) – Starting vertical coordinate to begin tracing. Default value = None.
- **MaxLength** (*int*) – Maximum boundary length to trace before terminating. Default value = np.inf.

Notes

The Improved Simple Boundary Follower (ISBF) from the reference below is used for 4-connected tracing. This algorithm provides accurate tracing with competitive execution times. 8-connected tracing is implemented using the Moore tracing algorithm.

Returns

- **X** (*array_like*) – A 1D array of horizontal coordinates of contour seed pixels for tracing.
- **Y** (*array_like*) – A 1D array of the vertical coordinates of seed pixels for tracing.

References

Following Method for Image Sensors" in Sensors,vol.16,no.353, doi:10.3390/s16030353, 2016.

histomicstk.segmentation.label.**TraceLabel**(*Label*, *Connectivity=4*)

Performs exterior boundary tracing of a multiple objects in a label image.

Parameters

- **Mask** (*array_like*) – A boolean type image where foreground pixels have value 'True', and background pixels have value 'False'.
- **Connectivity** (*int*) – Neighborhood connectivity to evaluate. Valid values are 4 or 8. Default value = 4.

Returns

- **X** (*array_like*) – A list of 1D arrays of horizontal coordinates for each object in 'Label'.
- **Y** (*array_like*) – A list of 1D arrays of horizontal coordinates for each object in 'Label'.

Notes

Objects should be made contiguous using SplitLabel prior to tracing. Returns lists with length Label.max(). Object values missing from 'Label' will have corresponding boundaries with value 'None' in the outputs. Condensing the label image values prior to boundary tracing can prevent this.

Uses the Improved Simple Boundary Follower (ISBF) from the reference below for 4-connected tracing. This algorithm provides accurate tracing with competitive execution times. 8-connected tracing is implemented using the Moore tracing algorithm.

See also:

histomicstk.segmentation.label.SplitLabel(), *histomicstk.segmentation.label.CondenseLabe*
histomicstk.segmentation.label.TraceBounds()

References

Following Method for Image Sensors" in Sensors,vol.16,no.353, doi:10.3390/s16030353, 2016.

histomicstk.segmentation.label.**WidthOpenLabel**(*Label*, *Width*)

Removes thin objects from label image using maximum of distance transform values within each object.

Parameters

- **Label** (*array_like*) – A uint32 type label image generated by segmentation methods.

- **Width** (*int*) – Width threshold for objects. Objects with fewer than 'Area' pixels will be zeroed to merge with background.

Notes

Objects are assumed to have positive nonzero values. A binary mask is generated for each object setting all other objects to the background value (0). The maximum chamfered distance transform value of this mask is used to represent the object width.

Returns **Split** – A uint32 label where objects with pixels < Area are removed.

Return type array_like

See also:

histomicstk.segmentation.label.CondenseLabel(), *histomicstk.segmentation.label.ShuffleLa*
histomicstk.segmentation.label.SplitLabel(), *histomicstk.segmentation.label.AreaOpenLabe*

histomicstk.segmentation.level_set

This package contains functions that implement commonly used level-set based methods for segmenting objects/regions in images.

histomicstk.segmentation.level_set.**ChanVese**(*I*, *Mask*, *Sigma*, *dt=1.0*, *Mu=0.2*, *Lambda1=1*, *Lambda2=1*, *It=100*)

Region-based level sets.

Region-based level set implementation based on the Chan-Vese method. Provides cost terms for boundary length, the variance of intensities inside the zero level-set, and the variance of external intensities. Robust to initialization.

Parameters

- **I** (*array_like*) – A floating-point intensity image.
- **Mask** (*array_like*) – A binary mask initializing the level set image. Regions corresponding to the interior of the level-set function have value 1, with other regions having value 0.
- **Sigma** (*double*) – Standard deviation of smoothing filter for input image I.
- **dt** (*double*) – Time step for evolving the level-set function Phi. Default value = 1.0.
- **Mu** (*double*) – Boundary length weight for energy function. Default value = 0.2.
- **Lambda1** (*double*) – Internal variance weight for energy function. Default value = 1.
- **Lambda2** (*double*) – External variance weight for energy function. Default value = 1.
- **It** (*double*) – Number of iterations to evolve curve level set function over. Default value = 100.

Returns **Phi** – An intensity image where the zero level set defines object boundaries. Can be further processed with fast marching methods or other to obtain smooth boundaries, or simply thresholded to define the object mask.

Return type array_like

See also:

histomicstk.segmentation.nuclear.GaussianVoting()

References

histomicstk.segmentation.level_set.**DregEdge**(*I*, *Phi*, *Well='double'*, *Sigma=1.5*, *dt=1.0*, *Mu=0.2*, *Lambda=1*, *Alpha=-3*, *Epsilon=1.5*, *It=100*)

Distance-regularized edge-based level sets.

Distance-regularization is used in this edge-based level set implementation to avoid numerical problems requiring costly re-initialization. Provides cost terms for boundary length, area, and regularization of the level set function. Foreground objects are assumed to have larger intensity values than background.

Parameters

- **I** (*array_like*) – A floating-point intensity image.
- **Phi** (*array_like*) – A floating-point initalization at the level-set image. Interior values are set to -c0, and exterior values set to c0, where c0 > 0.
- **Well** (*string*) – Choice of well function for regularization. Can be set to either 'single' or 'double' for single-well or double-well regularization, or any other value for no regularization. Default value = 'double'.
- **Sigma** (*double*) – Standard deviation of smoothing filter for input image I.
- **dt** (*double*) – Time step for evolving Phi. Default value = 1.0.
- **Mu** (*double*) – Regularization weight for energy function. Default value = 0.2.
- **Lambda** (*double*) – Boundary length weight for energy function. Default value = 1.0.
- **Alpha** (*double*) – Area weight for energy function. A negative value is used to seed the interior of the foreground objects and then evolve the boundary outwards. A positive value assumes that the boundary begins outside the foreground objects and collapses to their high-gradient edges. Default value = -3.
- **Epsilon** (*double*) – Coefficient used to smooth the Dirac and Heaviside functions. Default value = 1.5.
- **It** (*double*) – Number of iterations to evolve curve level set function over. Default value = 100.

Returns **Phi** – An intensity image where the zero level set defines object boundaries. Can be further processed with fast marching methods or other to obtain smooth boundaries, or simply thresholded to define the object mask.

Return type array_like

See also:

histomicstk.segmentation.nuclear.GaussianVoting()

References

histomicstk.segmentation.nuclear

This package contains implementations of state-of-th-art methods for segmenting nuclei from histopathology images.

histomicstk.segmentation.nuclear.**GaussianVoting**(*I*, *rmax=35*, *rmin=10*, *sSigma=5*, *Tau=5*, *bw=15*, *Psi=0.3*)

Performs nuclear detection using Gaussian kernel voting.

Uses a gaussian kernel to localize the centroids of cell nuclei. Takes as input a hematoxylin-deconvolved image and uses the gradient signal to cast directed votes towards the center of cell nuclei. These votes are blurred by a gaussian kernel, and are spatially clustered using the mean-shift algorithm. Convolutions are performed separably to reduce compute time.

Parameters

- **I** (*array_like*) – A hematoxylin intensity image obtained from ColorDeconvolution.
- **rmax** (*double*) – Upper-limit on voting area extent. Default value = 35.
- **rmin** (*double*) – Lower-limit on voting area extent. Default value = 10.
- **sSigma** (*double*) – Standard deviation of smoothing kernel used in gradient calculation. Default value = 5.
- **Tau** (*double*) – Lower limit on gradient magnitude for casting a vote. Default value = 5.
- **bw** (*double*) – Bandwidth parameter for mean-shift clustering. Default value = 15.
- **Psi** (*double*) – Lower limit threshold on votes. Expressed as a percentage of the maximum vote, ranges from [0, 1). Default value = 0.3.

Returns

- **Centroids** (*array_like*) – An N x 2 array defining the (x,y) coordinates of cell centroids.
- **Votes** (*array_like*) – An intensity image containing the blurred votes obtained by voting.

Notes

Return values are returned as a namedtuple

See also:

histomicstk.preprocessing.color_deconvolution.ColorDeconvolution()

References

histomicstk.segmentation.nuclear.**GradientFlow**(*I*, *Mask*, *K=1000*, *Diffusions=10*, *Mu=5*, *Lambda=5*, *Iterations=10*, *dT=0.05*)

Performs gradient-field tracking to segment smoothed images of cell nuclei.

Takes as input a smoothed intensity or Laplacian-of-Gaussian filtered image and a foreground mask, and groups pixels by tracking them to mutual gradient sinks. Typically requires merging of sinks (seeds) as a post processing steps.

Parameters

- **I** (*array_like*) – Smoothed intensity or log-filtered response where nuclei regions have larger intensity values than background.
- **Mask** (*array_like*) – Binary mask where foreground objects have value 1, and background objects have value 0. Used to restrict influence of background vectors on diffusion process and to reduce tracking computations.
- **K** (*float*) – Number of steps to check for tracking cycle. Default value = 1000.
- **Mu** (*float*) – Weight parmeter from Navier-Stokes diffusion - weights divergence and Laplacian terms. Default value = 5.

- **Lambda** (*float*) – Weight parameter from Navier-Stokes diffusion - used to weight divergence. Default value = 5.
- **Iterations** (*float*) – Number of time-steps to use in Navier-Stokes diffusion. Default value = 10.
- **dT** (*float*) – Timestep to be used in Navier-Stokes diffusion. Default value = 0.05.

Returns

- **Segmentation** (*array_like*) – Label image where positive values correspond to foreground pixels that share mutual sinks.
- **Sinks** (*array_like*) – N x 2 array containing the (x,y) locations of the tracking sinks. Each row is an (x,y) pair - in that order.

See also:

histomicstk.utils.GradientDiffusion(), *histomicstk.segmentation.nuclear.MergeSinks()*, *histomicstk.segmentation.label.ShuffleLabel()*

References

histomicstk.segmentation.nuclear.**MaxClustering**(*Response*, *Mask*, *r=10*)

Local max clustering pixel aggregation for nuclear segmentation. Takes as input a constrained log or other filtered nuclear image, a binary nuclear mask, and a clustering radius. For each pixel in the nuclear mask, the local max is identified. A hierarchy of local maxima is defined, and the root nodes used to define the label image.

Parameters

- **Response** (*array_like*) – A filtered-smoothed image where the maxima correspond to nuclear center. Typically obtained by constrained-LoG filtering on a hematoxylin intensity image obtained from ColorDeconvolution.
- **Mask** (*array_like*) – A binary mask of type boolean where nuclei pixels have value 'True', and non-nuclear pixels have value 'False'.
- **r** (*float*) – A scalar defining the clustering radius. Default value = 10.

Returns

- **Label** (*array_like*) – Label image where positive values correspond to foreground pixels that share mutual sinks.
- **Seeds** (*array_like*) – An N x 2 array defining the (x,y) coordinates of nuclei seeds.
- **Maxima** (*array_like*) – An N x 1 array containing the maximum response value corresponding to 'Seeds'.

See also:

histomicstk.filters.shape.cLoG()

References

histomicstk.segmentation.nuclear.**MergeSinks**(*Label*, *Sinks*, *Radius=5*)

Merges attraction basins obtained from gradient flow tracking using sink locations.

Parameters

- **Segmentation** (*array_like*) – Label image where positive values correspond to foreground pixels that share mutual sinks.
- **Sinks** (*array_like*) – N x 2 array containing the (x,y) locations of the tracking sinks. Each row is an (x,y) pair - in that order.
- **Radius** (*float*) – Radius used to merge sinks. Sinks closer than this radius to one another will have their regions of attraction merged. Default value = 5.

Returns **Merged** – Label image where attraction regions are merged.

Return type array_like

See also:

histomicstk.utils.GradientDiffusion(), *histomicstk.segmentation.label.ShuffleLabel()*

histomicstk.segmentation.nuclear.**MinimumModel**(*I*, *Delta=0.3*, *MaxLength=255*, *Compaction=3*, *MinArea=100*, *MinWidth=5*, *MinDepth=2*, *MinConcavity=inf*)

Performs a nuclear segmentation using a gradient contour tracing and geometry splitting algorithm. Implemented from the reference below.

Parameters

- **I** (*array_like*) – An intensity image used for analyzing local minima/maxima and gradients. Dimensions M x N.
- **Delta** (*float*) – Percent difference threshold between minima/maxima pairs to be included in seed point detection. Percentage difference ([0, 1]) in total image range e.g. Delta = 0.3 with a uint8 input would translate to 0.3 * 255. Default value = 0.3.
- **MaxLength** (*int*) – Maximum allowable contour length. Default value = 255.
- **Compaction** (*int*) – Factor used in compacting objects to remove thin spurs. Refered to as 'd' in the paper. Default value = 3.
- **MinArea** (*int*) – Minimum area of objects to analyze. Default value = 100.
- **MinWidth** (*int*) – Minimum max-width of objects to analyze. Default value = 5.
- **MinDepth** (*float*) – Minimum depth of concavities to consider during geometric splitting. Default value = 2.
- **MinConcavity** (*float*) – Minimum concavity score to consider when performing for geometric splitting. Default value = np.inf.

Notes

Objects are assumed to be dark (as nuclei in hematoxylin channel from color deconvolution). Smoothing improves accuracy and computation time by eliminating spurious seed points. Specifying a value for 'Delta' prevents shallow transitions from being included, also reducing computation time and increasing specificity.

Returns

- **X** (*array_like*) – A 1D array of horizontal coordinates of contour seed pixels for tracing.
- **Y** (*array_like*) – A 1D array of the vertical coordinates of seed pixels for tracing.
- **Min** (*array_like*) – A 1D array of the corresponding minimum values for contour tracing of seed point X, Y.
- **Max** (*array_like*) – A 1D array of the corresponding maximum values for contour tracing of seed point X, Y.

See also:

histomicstk.segmentation.label.TraceBounds()

References

3.1.5 histomicstk.utils

This package contains utility functions that are widely used by functions in all other sub-packages of histomicstk

histomicstk.utils.**Del2**(*X*)

Discrete Laplacian with edge-value extrapolation.

Calculates the discrete Laplacian of an input image. Edge values are calculated by using linear extrapolation of second differences. This is consistent with the way that Matlab calculates the discrete Laplacian.

Parameters **X** (*array_like*) – A floating-point intensity image.

Returns **L** – The discrete Laplacian of L.

Return type array_like

See also:

histomicstk.segmentation.level_set.DregEdge(),*histomicstk.segmentation.level_set.ChanVe*

histomicstk.utils.**Eigenvalues**(*H*)

Calculates the eigenvectors of the hessian volumes 'H' generated by Hessian.py

Parameters **H** (*array_like*) – M x N x 4 hessian matrix - H[:,:,0] = dxx, H[:,:,1] = H[:,:,2] = dxy, H[:,:,3] = dyy.

Returns

- **Lambda** (*array_like*) – M x N x 2 image of eigenvalues.
- **V1** (*array_like*) – M x N x 2 eigenvector for Lambda(:,:,0)
- *V2* – M x N x 2 eigenvector for Lambda(:,:,1)

histomicstk.utils.**GradientDiffusion**(*dX*, *dY*, *Mask*, *Mu=5*, *Lambda=5*, *Iterations=10*, *dT=0.05*)

Diffusion of gradient field using Navier-Stokes equation. Used for smoothing/denoising a gradient field.

Takes as input a gradient field image (dX, dY), and a mask of the foreground region, and then iteratively solves the Navier-Stokes equation to diffuse the vector field and align noisy gradient vectors with their surrounding signals.

Parameters

- **dX** (*array_like*) – Horizontal component of gradient image.
- **dY** (*array_like*) – Vertical component of gradient image.
- **Mu** (*float*) – Weight parmeter from Navier-Stokes equation - weights divergence and Laplacian terms. Default value = 5.
- **Lambda** (*float*) – Weight parameter from Navier-Stokes equation - used to weight divergence. Default value = 5.
- **Mask** (*array_like*) – Binary mask where foreground objects have value 1, and background objects have value 0. Used to restrict influence of background vectors on diffusion process.

- **Iterations** (*float*) – Number of time-steps to use in solving Navier-Stokes. Default value = 10.
- **dT** (*float*) – Timestep to be used in solving Navier-Stokes. Default value = 0.05.

Returns

- **vX** (*array_like*) – Horizontal component of diffused gradient.
- **vY** (*array_like*) – Vertical component of diffused gradient.

See also:

histomicstk.segmentation.nuclear.GradientFlow()

References

histomicstk.utils.**Hessian**(*I*, *Sigma*)

Calculates hessian of image I convolved with a gaussian kernel with covariance C = [Sigma^2 0; 0 Sigma^2].

Parameters

- **I** (*array_like*) – M x N grayscale image.
- **Sigma** (*double*) – standard deviation of gaussian kernel.

Returns M x N x 4 hessian matrix - H[:,:,0] = dxx, H[:,:,1] = H[:,:,2] = dxy, H[:,:,3] = dyy.

Return type H - array_like

histomicstk.utils.**MergeColinear**(*X*, *Y*)

Processes boundary coordinates in polyline with vertices X, Y to remove redundant colinear points. Polyline is not assumed to be open or closed.

Parameters

- **X** (*array_like*) – One dimensional array of horizontal boundary coordinates.
- **Y** (*array_like*) – One dimensional array of vertical boundary coordinates.

Returns

- **XOut** (*array_like*) – X with colinear boundary points removed.
- **YOUt** (*array_like*) – Y with colinear boundary points removed.

histomicstk.utils.**PoissonMixture**(*I*, *Mu=None*, *InitialTau=None*, *Tol=0.1*)

Generates a Poisson mixture model to fit pixel intensities for foreground/background masking.

Takes as input an array or intensity image 'I' and optimizes a two-component poisson model describing foreground and background intensity models. This model can be used to describe the probability that a pixel comes from foreground versus background. The poisson distribution assumes discrete values and so is suitable for integral valued intensity images. Assumes that foreground intensities are lower (darker) than background.

Parameters

- **I** (*array_like*) – A hematoxylin intensity image obtained from ColorDeconvolution.
- **Mu** (*double*) – Optional mean value of signal to optimize. Calculated from input if defined as 'None'. Default value = None.

Returns

- **Tau** (*double*) – Optimal threshold for distinguishing foreground and background.

- **Foreground** (*array_like*) – An intensity image with values in the range [0, 1] representing foreground probabiities for each pixel.
- **Background** (*array_like*) – An intensity image with values in the range [0, 1] representing background probabiities for each pixel.

References

histomicstk.utils.**Sample**(*File*, *Magnification*, *Percent*, *Tile*, *MappingMag=1.25*, *Coverage=0.1*)

Generates a sampling of pixels from a whole-slide image.

Useful for generating statistics or Reinhard color-normalization or adaptive deconvolution. Uses mixture modeling approach to focus sampling in tissue regions.

Parameters

- **File** (*str*) – path and filename of slide.
- **Magnification** (*double*) – Desired magnification for sampling (defaults to native scan magnification).
- **Percent** (*double*) – Percentage of pixels to sample. Must be in the range [0, 1].
- **Tile** (*int*) – Tile size used in sampling high-resolution image.
- **MappingMag** (*double, optional*) – low resolution magnification. Default value = 1.25.
- **Coverage** (*double, optional*) – minimum percent of tile covered by tissue to be included in sampling. Ranges between [0,1). Default value = 0.1.

Returns **Pixels** – A 3xN matrix of RGB pixel values sampled from the slide at *File*.

Return type array_like

See also:

histomicstk.preprocessing.color_normalization.reinhard(), *histomicstk.preprocessing.color_deconvolution.SparseColorDeconvolution()*

histomicstk.utils.**SimpleMask**(*I*, *BW=2*, *DefaultBGScale=2.5*, *DefaultTissueScale=30*, *MinPeak=10*, *MaxPeak=25*, *Percent=0.1*, *MinProb=0.05*)

Performs segmentation of the foreground (tissue) Uses a simple two-component Gaussian mixture model to mask tissue areas from background in brightfield H&E images. Kernel-density estimation is used to create a smoothed image histogram, and then this histogram is analyzed to identify modes corresponding to tissue and background. The mode peaks are then analyzed to estimate their width, and a constrained optimization is performed to fit gaussians directly to the histogram (instead of using expectation-maximization directly on the data which is more prone to local minima effects). A maximum-likelihood threshold is then derived and used to mask the tissue area in a binarized image.

Parameters

- **I** (*array_like*) – An RGB image of type unsigned char.
- **BW** (*double, optional*) – Bandwidth for kernel density estimation - used for smoothing the grayscale histogram. Default value = 2.
- **DefaultBGScale** (*double, optional*) – Standard deviation of background gaussian to be used if estimation fails. Default value = 2.5.
- **DefaultTissueScale** (*double, optional*) – Standard deviation of tissue gaussian to be used if estimation fails. Default value = 30.

- **MinPeak** (*double, optional*) – Minimum peak width for finding peaks in KDE histogram. Used to initialize curve fitting process. Default value = 10.
- **MaxPeak** (*double, optional*) – Maximum peak width for finding peaks in KDE histogram. Used to initialize curve fitting process. Default value = 25.
- **Percent** (*double, optional*) – Percentage of pixels to sample for building foreground/background model. Default value = 0.10.
- **MinProb** (*double, optional*) – Minimum probability to qualify as tissue pixel. Default value = 0.05.

Returns **Mask** – A binarized version of *I* where foreground (tissue) has value '1'.

Return type array_like

See also:

histomicstk.utils.Sample()

CHAPTER 4

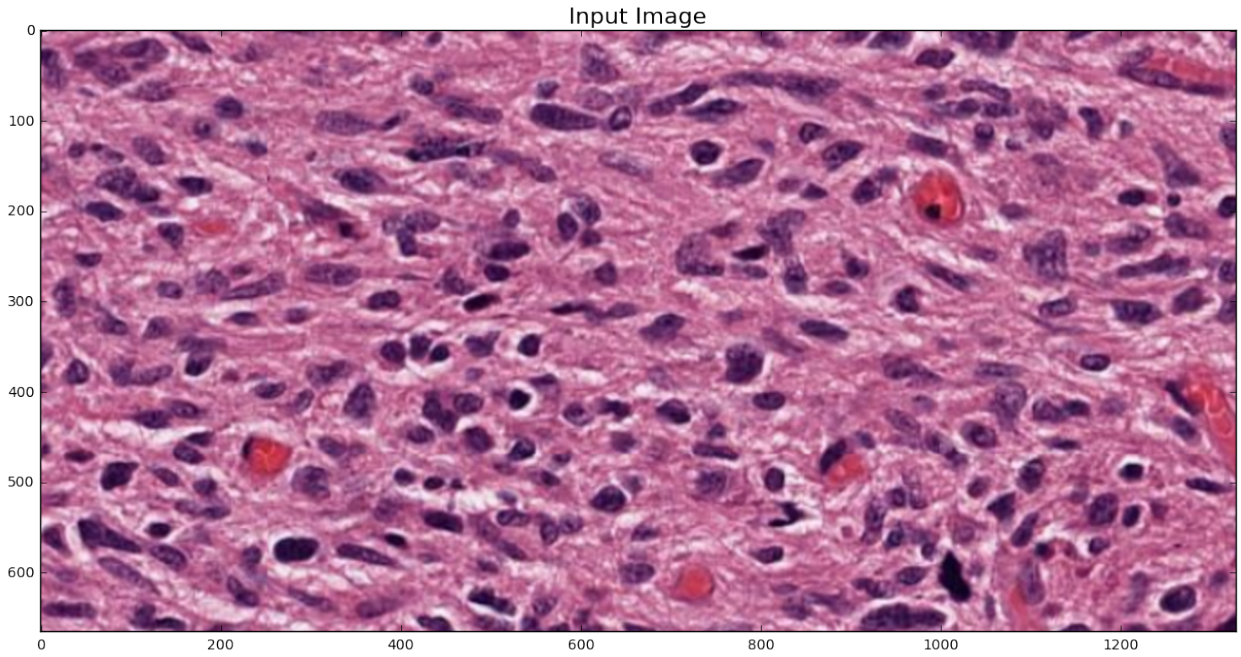
Examples

4.1 Color Deconvolution

```
In [50]: import histomicstk as htk

         import numpy as np
         import scipy as sp

         import skimage.io
         import skimage.measure
         import skimage.color

         import matplotlib.pyplot as plt
         import matplotlib.patches as mpatches
         %matplotlib inline

         #Some nice default configuration for plots
         plt.rcParams['figure.figsize'] = 15, 15
         plt.rcParams['image.cmap'] = 'gray'
         titlesize = 24
```

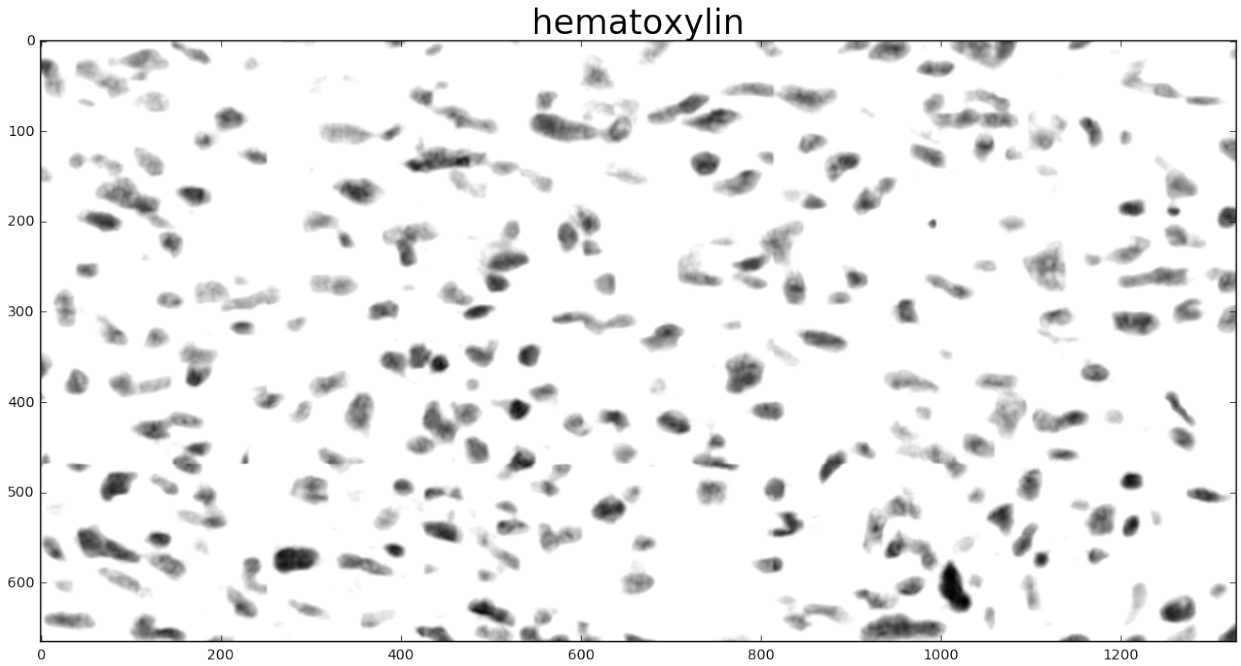
4.1.1 Load input image

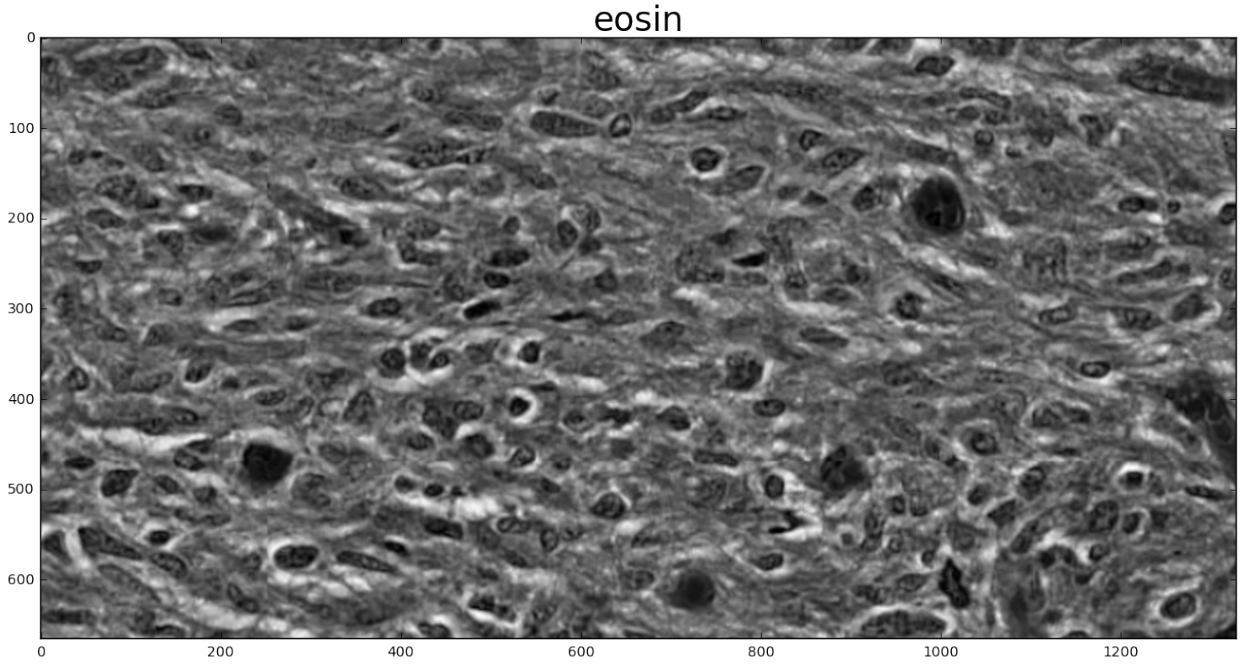
```
In [51]: inputImageFile = ('https://data.kitware.com/api/v1/file/'
                           '57802ac38d777f12682731a2/download')  # H&E.png

         imInput = skimage.io.imread(inputImageFile)[:, :, :3]

         plt.imshow(imInput)
         _ = plt.title('Input Image', fontsize=16)
```



4.1.2 Supervised color deconvolution with a known stain matrix

```
In [52]: # create stain to color map
         stainColorMap = {
             'hematoxylin': [0.65, 0.70, 0.29],
             'eosin':       [0.07, 0.99, 0.11],
             'dab':         [0.27, 0.57, 0.78],
             'null':        [0.0, 0.0, 0.0]
         }

         # specify stains of input image
         stain_1 = 'hematoxylin'   # nuclei stain
         stain_2 = 'eosin'         # cytoplasm stain
         stain_3 = 'null'          # set to null of input contains only two stains

         # create stain matrix
         W = np.array([stainColorMap[stain_1],
                       stainColorMap[stain_2],
                       stainColorMap[stain_3]]).T

         # perform standard color deconvolution
         imDeconvolved = htk.preprocessing.color_deconvolution.ColorDeconvolution(imInput,

         # Display results
         plt.figure()
         plt.imshow(imDeconvolved.Stains[:, :, 0])
         plt.title(stain_1, fontsize=titlesize)

         plt.figure()
         plt.imshow(imDeconvolved.Stains[:, :, 1])
         _ = plt.title(stain_2, fontsize=titlesize)
```



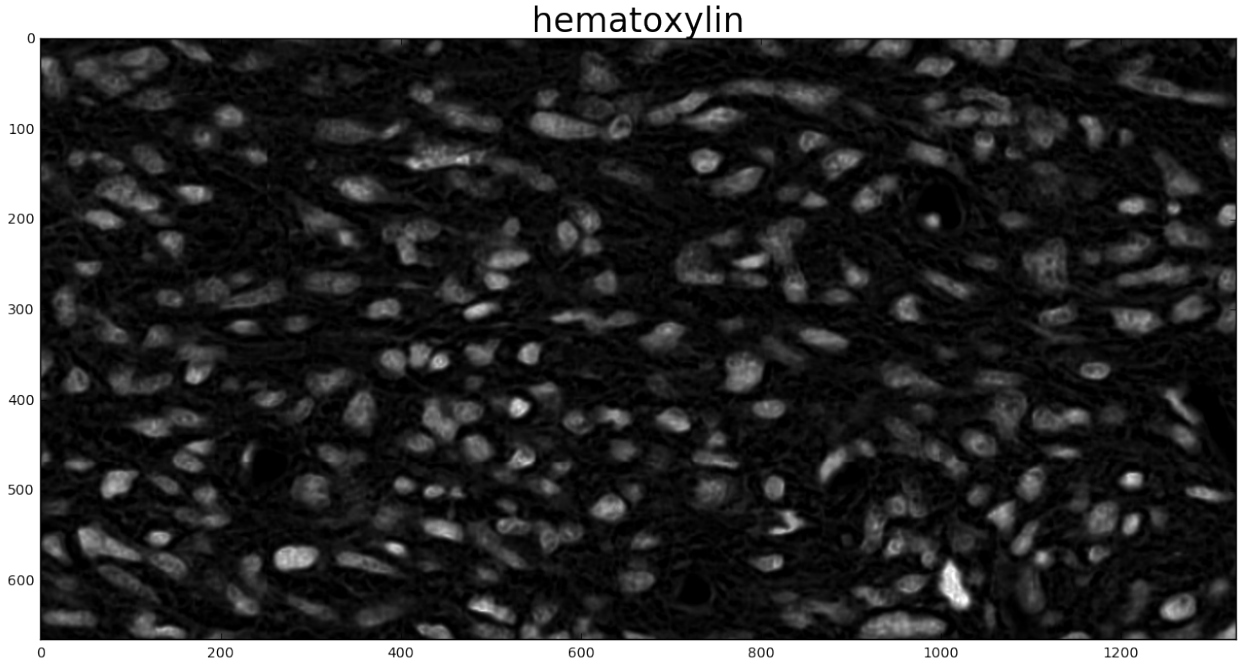


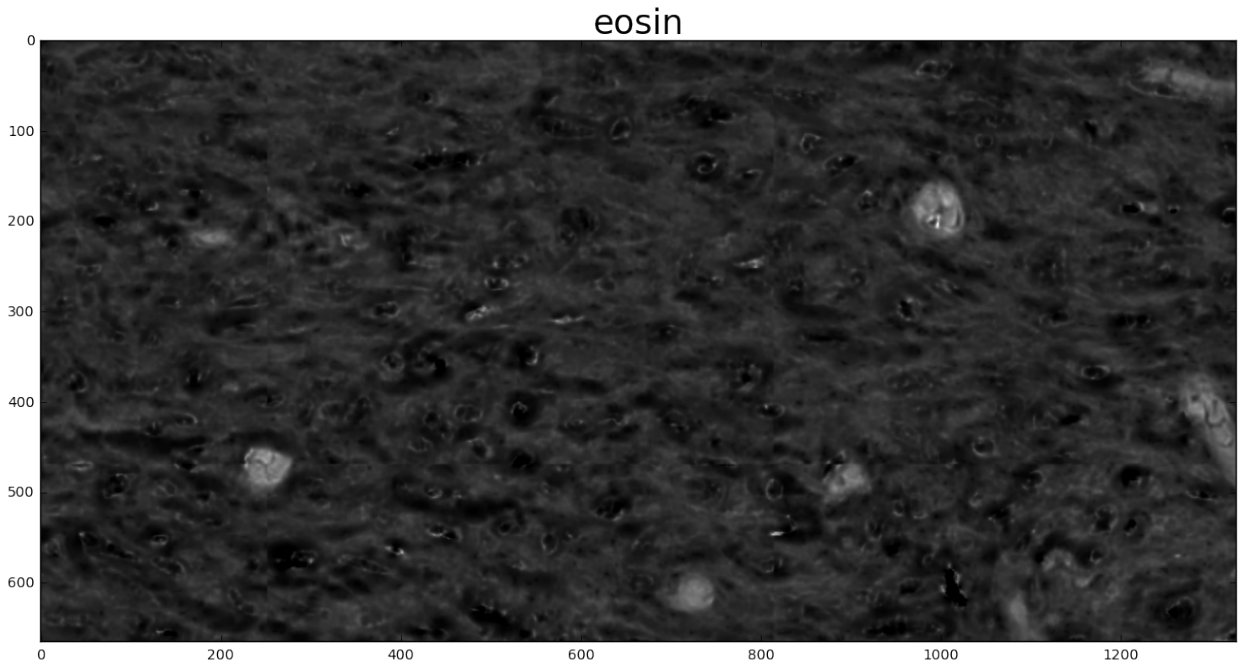
4.1.3 Unsupervised color deconvolution using sparse non-negative matrix factorization

```
In [53]: # create initial stain matrix
         W_init = np.array([stainColorMap[stain_1],
                            stainColorMap[stain_2]]).T

         # perform sparse color deconvolution
         sparsity_factor = 0.5
```

```
        imDeconvolved, W_est = htk.preprocessing.color_deconvolution.SparseColorDeconvolut
            imInput, W_init, sparsity_factor)

        print 'Estimated stain colors (in rows): '
        print W_est.T

        # Display results
        plt.figure()
        plt.imshow(imDeconvolved[:, :, 0])
        plt.title(stain_1, fontsize=titlesize)

        plt.figure()
        plt.imshow(imDeconvolved[:, :, 1])
        _ = plt.title(stain_2, fontsize=titlesize)
```

```
Estimated stain colors (in rows):
[[ 0.48475365  0.78114376  0.39348231]
 [ 0.18976762  0.81925652  0.54111645]]
```



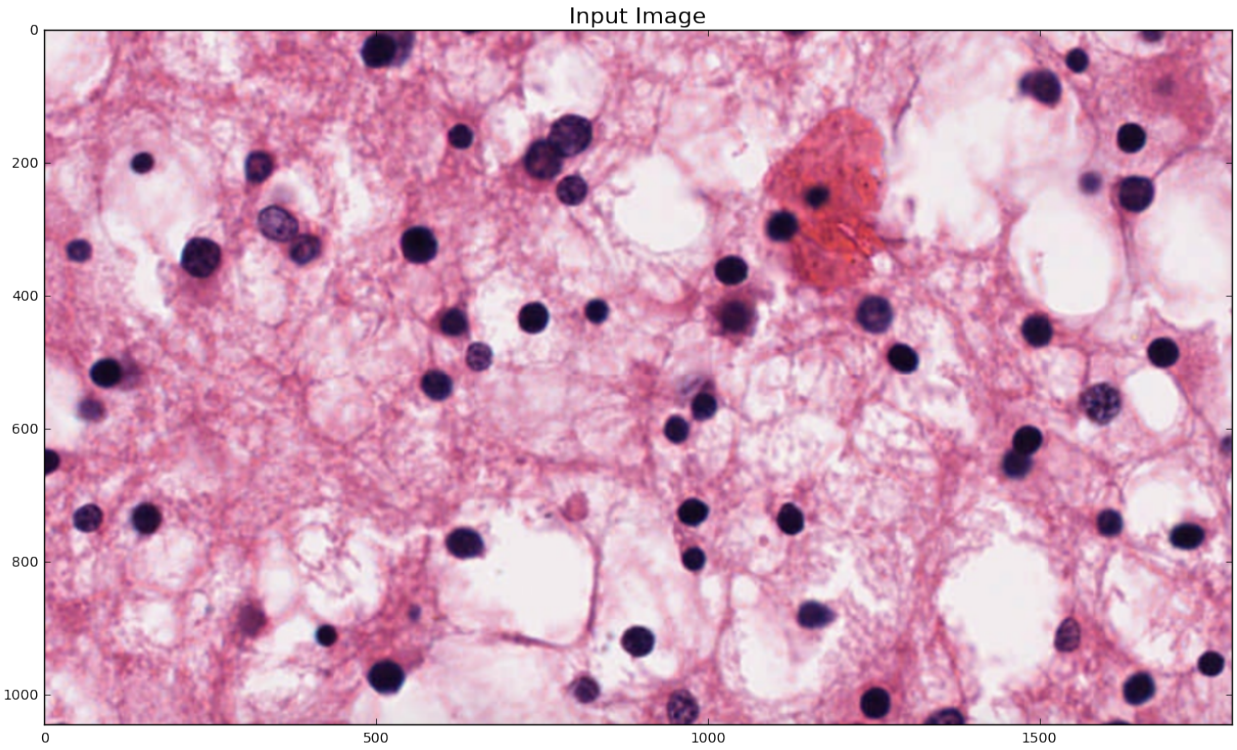


4.2 Nuclei Segmentation

```
In [27]: import histomicstk as htk

         import numpy as np
         import scipy as sp

         import skimage.io
         import skimage.measure
         import skimage.color

         import matplotlib.pyplot as plt
         import matplotlib.patches as mpatches
         %matplotlib inline

         #Some nice default configuration for plots
         plt.rcParams['figure.figsize'] = 15, 15
         plt.rcParams['image.cmap'] = 'gray'
         titlesize = 24
```

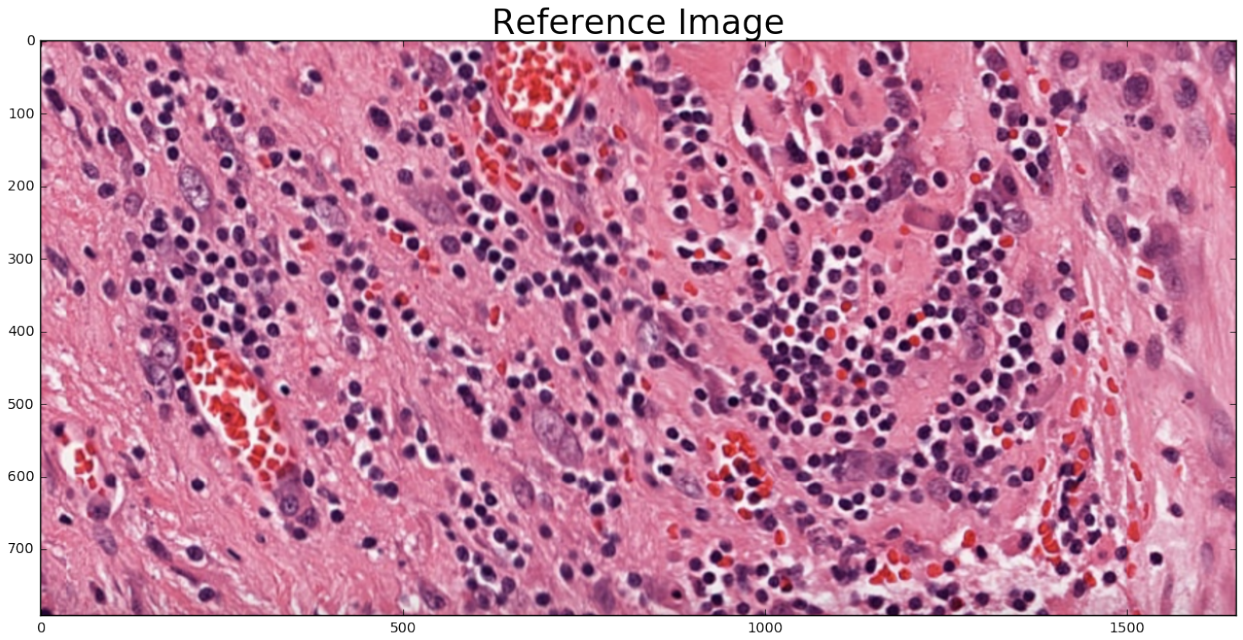
4.2.1 Load input image

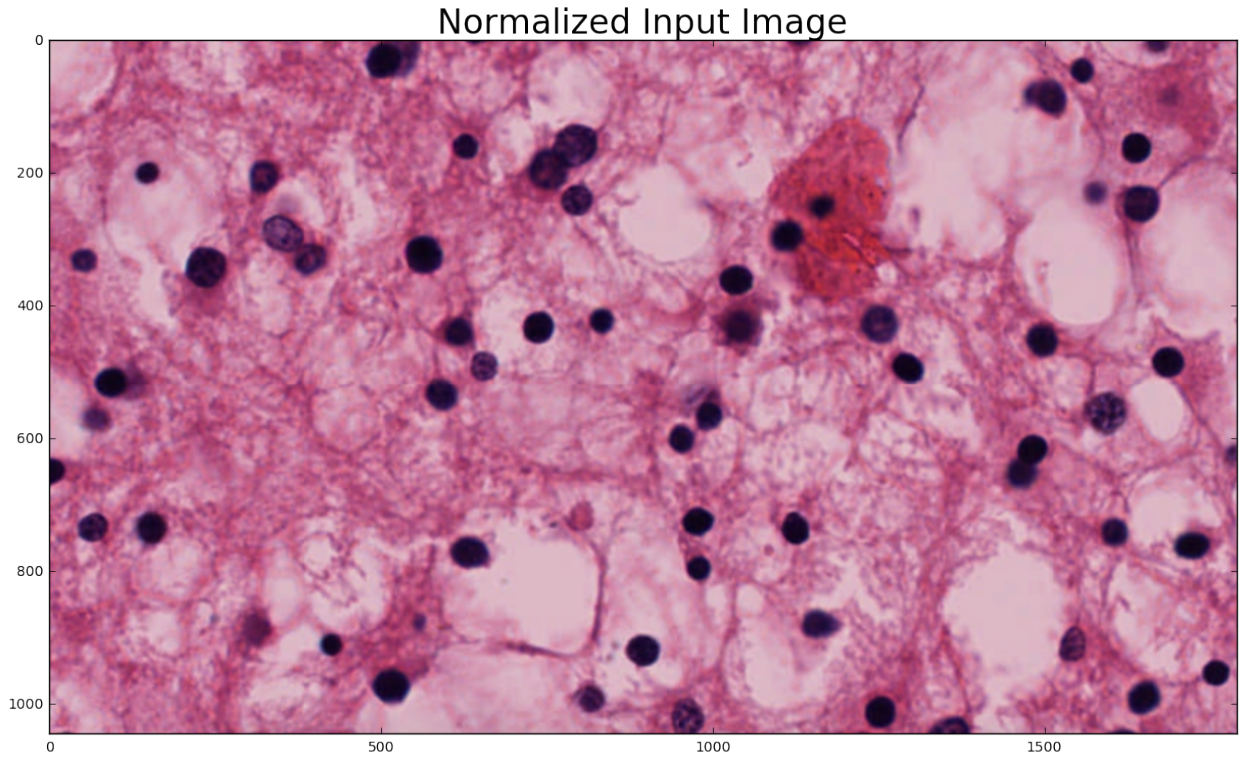
```
In [28]: inputImageFile = ('https://data.kitware.com/api/v1/file/'
                           '576ad39b8d777f1ecd6702f2/download')  # Easy1.png

         imInput = skimage.io.imread(inputImageFile)[:, :, :3]

         plt.imshow(imInput)
         _ = plt.title('Input Image', fontsize=16)
```



4.2.2 Perform color normalization

```
In [29]: # Load reference image for normalization
         refImageFile = ('https://data.kitware.com/api/v1/file/'
                         '57718cc28d777f1ecd8a883c/download')  # L1.png

         imReference = skimage.io.imread(refImageFile)[:, :, :3]

         # get mean and stddev of reference image in lab space
         meanRef, stdRef = htk.preprocessing.color_conversion.lab_mean_std(imReference)

         # perform reinhard color normalization
         imNmzd = htk.preprocessing.color_normalization.reinhard(imInput, meanRef, stdRef)

         # Display results
         plt.figure()
         plt.imshow(imReference)
         _ = plt.title('Reference Image', fontsize=titlesize)

         plt.figure()
         plt.imshow(imNmzd)
         _ = plt.title('Normalized Input Image', fontsize=titlesize)
```

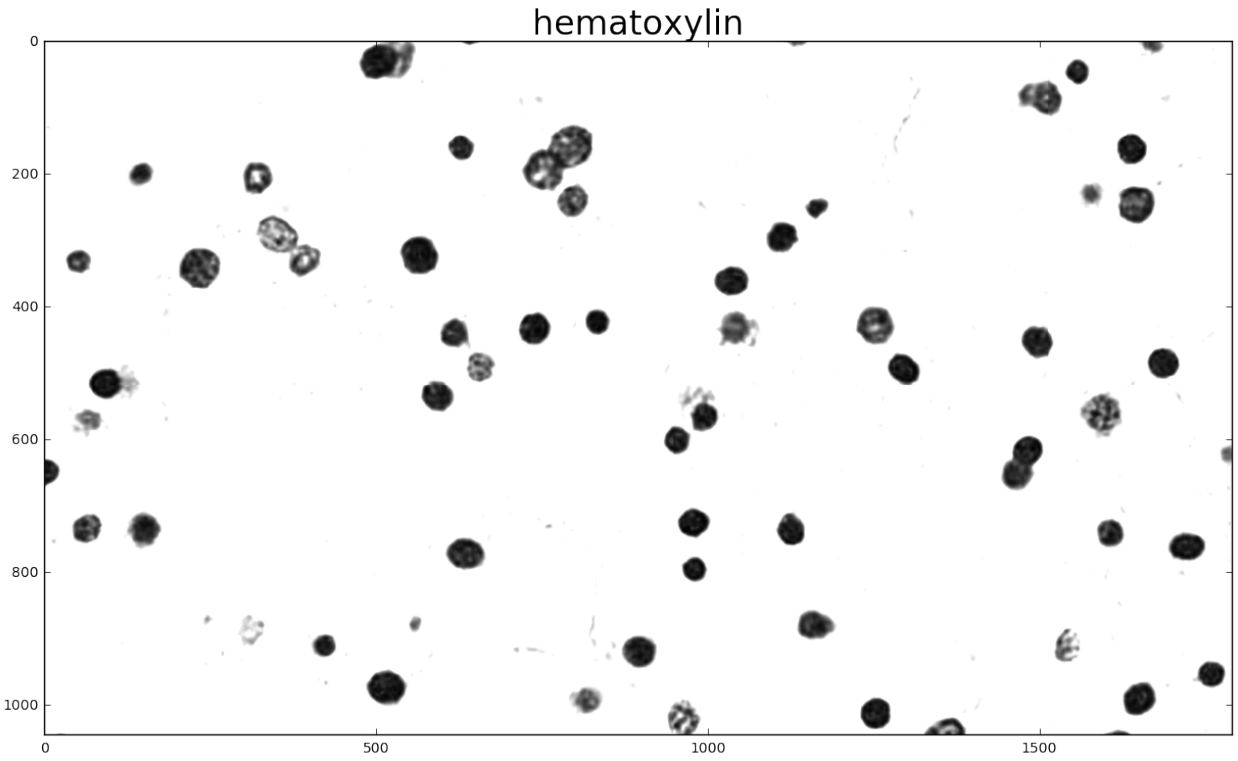


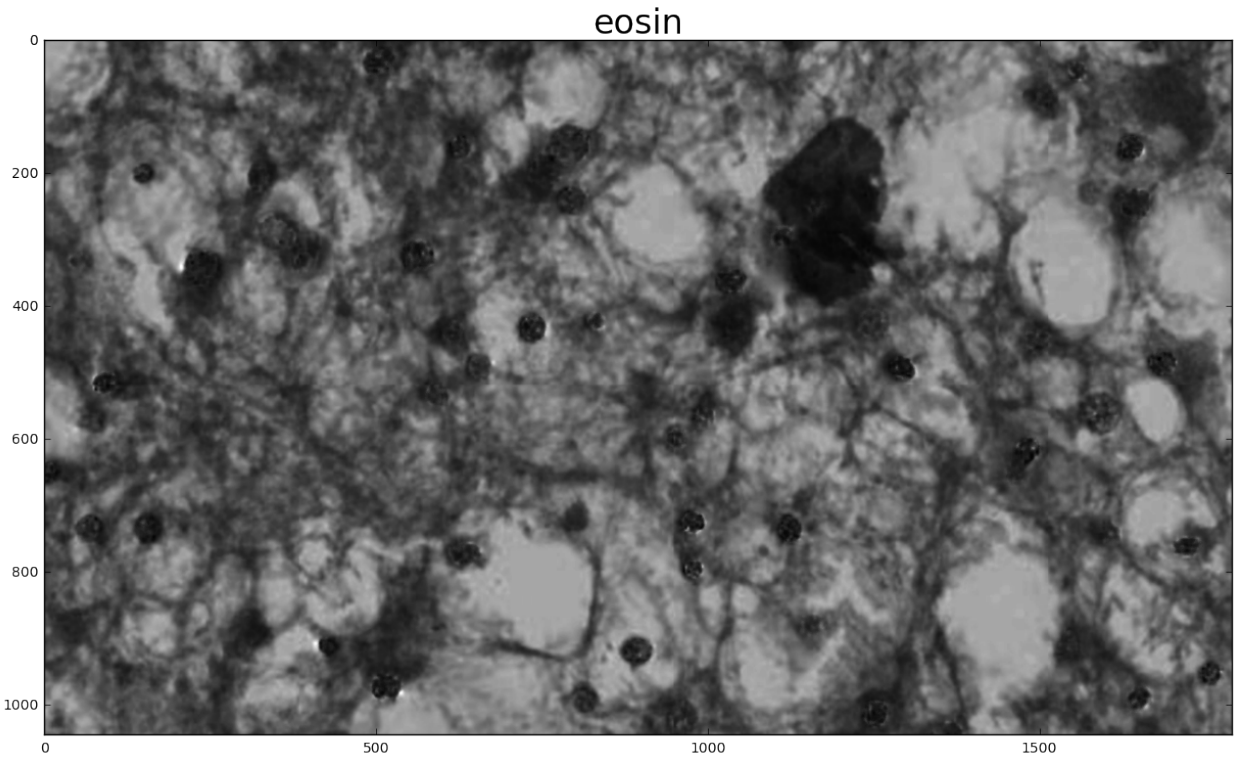


4.2.3 Perform color deconvolution

```
In [30]: # create stain to color map
         stainColorMap = {
             'hematoxylin': [0.65, 0.70, 0.29],
             'eosin':       [0.07, 0.99, 0.11],
             'dab':         [0.27, 0.57, 0.78],
```

```
    'null':         [0.0, 0.0, 0.0]
}

# specify stains of input image
stain_1 = 'hematoxylin'   # nuclei stain
stain_2 = 'eosin'         # cytoplasm stain
stain_3 = 'null'          # set to null of input contains only two stains

# create stain matrix
W = np.array([stainColorMap[stain_1],
              stainColorMap[stain_2],
              stainColorMap[stain_3]]).T

# perform standard color deconvolution
imDeconvolved = htk.preprocessing.color_deconvolution.ColorDeconvolution(imNmzd, W

# get nuclei/hematoxylin channel
imNucleiStain = imDeconvolved.Stains[:, :, 0].astype(np.float)

# Display results
plt.figure()
plt.imshow(imDeconvolved.Stains[:, :, 0])
plt.title(stain_1, fontsize=titlesize)

plt.figure()
plt.imshow(imDeconvolved.Stains[:, :, 1])
_ = plt.title(stain_2, fontsize=titlesize)
```



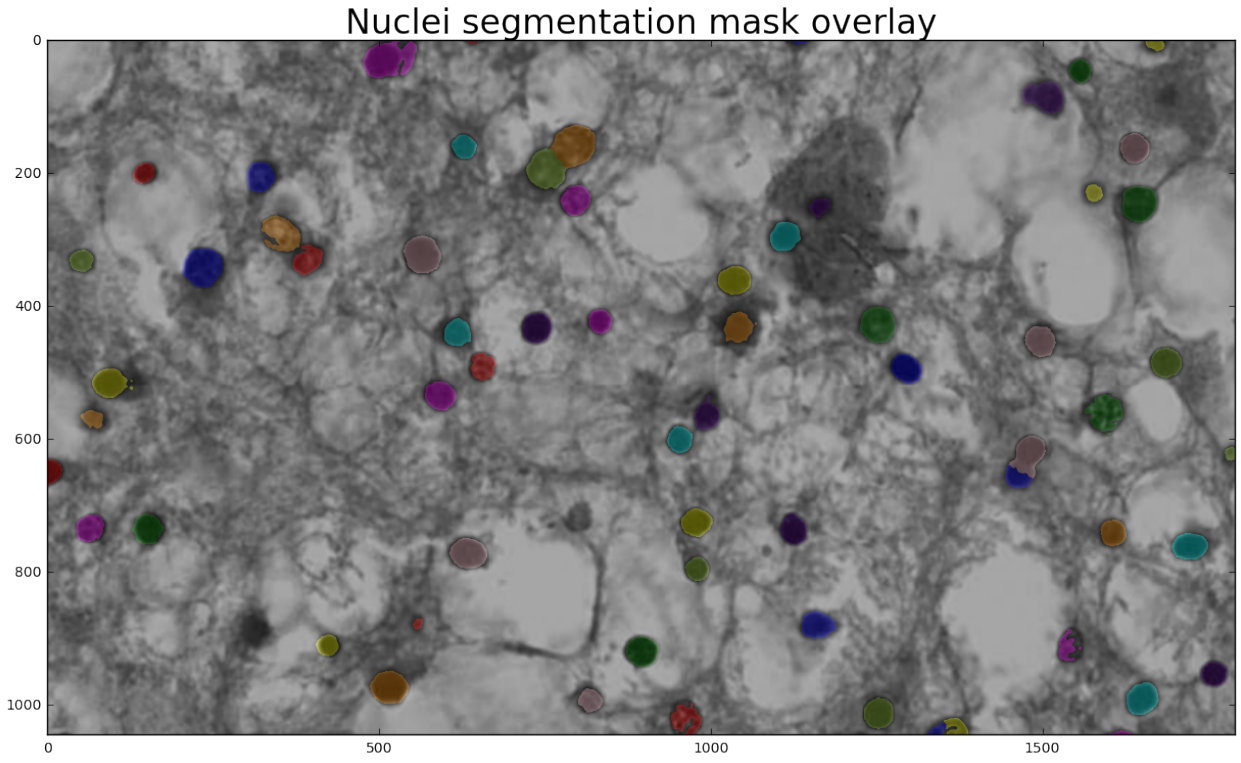


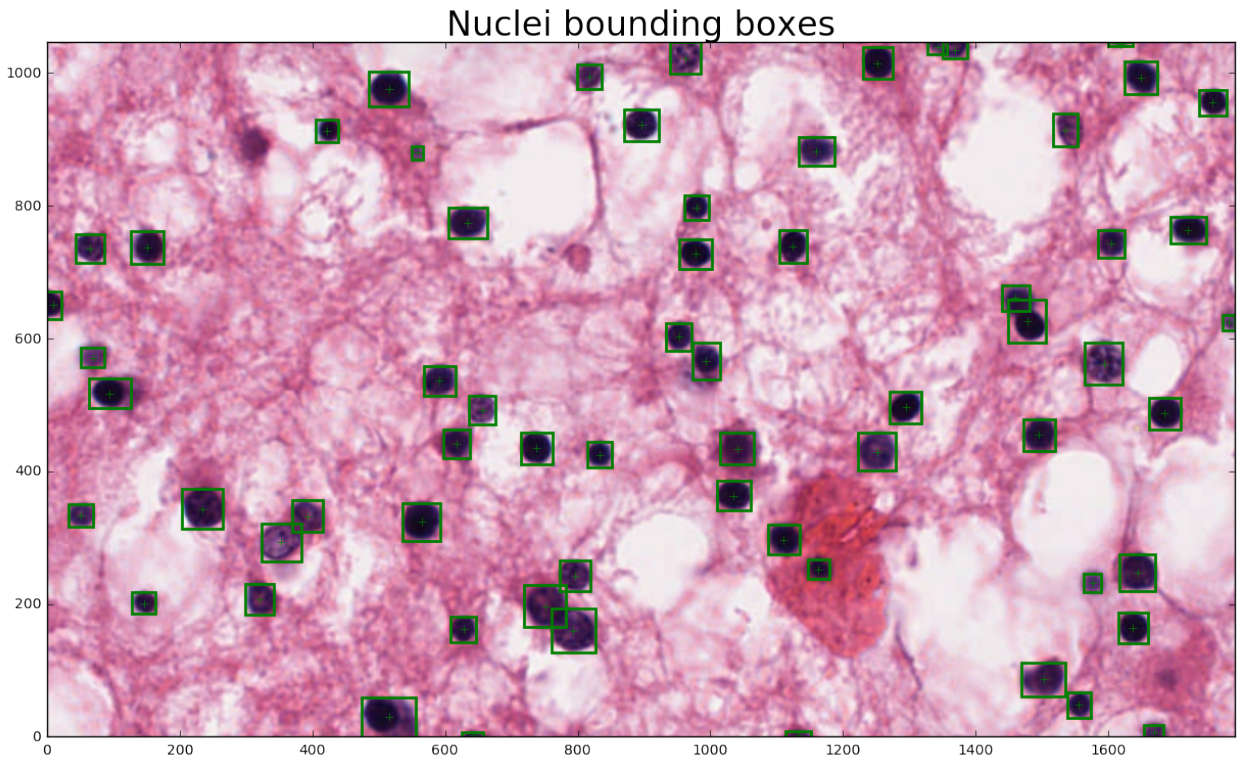
4.2.4 Segment Nuclei

```
In [31]: # segment foreground
         foreground_threshold = 160

         imFgndMask = sp.ndimage.morphology.binary_fill_holes(
             imNucleiStain < foreground_threshold)

         # run adaptive multi-scale LoG filter
         min_radius = 10
         max_radius = 15

         imLog = htk.filters.shape.cLoG(imNucleiStain, imFgndMask,
                                        SigmaMin=min_radius * np.sqrt(2),
                                        SigmaMax=max_radius * np.sqrt(2))

         # detect and segment nuclei using local maximum clustering
         local_max_search_radius = 10

         imNucleiSegMask, Seeds, Max = htk.segmentation.nuclear.MaxClustering(
             imLog, imFgndMask, local_max_search_radius)

         # filter out small objects
         min_nucleus_area = 80

         imNucleiSegMask = htk.segmentation.label.AreaOpenLabel(
             imNucleiSegMask, min_nucleus_area).astype(np.int)
```

```
        # compute nuclei properties
        objProps = skimage.measure.regionprops(imNucleiSegMask)

        print 'Number of nuclei = ', len(objProps)

        # Display results
        plt.figure()

        plt.imshow(skimage.color.label2rgb(imNucleiSegMask, imInput, bg_label=0))
        plt.title('Nuclei segmentation mask overlay', fontsize=titlesize)

        plt.figure()
        plt.imshow( imInput )
        plt.xlim([0, imInput.shape[1]])
        plt.ylim([0, imInput.shape[0]])
        plt.title('Nuclei bounding boxes', fontsize=titlesize)

        for i in range(len(objProps)):

            c = [objProps[i].centroid[1], objProps[i].centroid[0], 0]
            width = objProps[i].bbox[3] - objProps[i].bbox[1] + 1
            height = objProps[i].bbox[2] - objProps[i].bbox[0] + 1

            cur_bbox = {
                "type":         "rectangle",
                "center":       c,
                "width":        width,
                "height":       height,
            }

            plt.plot(c[0], c[1], 'g+')
            mrect = mpatches.Rectangle([c[0] - 0.5 * width, c[1] - 0.5 * height] ,
                                       width, height, fill=False, ec='g', linewidth=2)
            plt.gca().add_patch(mrect)
```

```
Number of nuclei =  64
```


Nuclei segmentation mask overlay
0
200
400
600
800
1000
0
500
1000
1500


Nuclei bounding boxes
0
200
400
600
800
1000
0
200
400
600
800
1000
1200
1400
1600

CHAPTER 5

Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

5.1 Types of Contributions

5.1.1 Report Bugs

Report bugs at https://github.com/DigitalSlideArchive/HistomicsTK/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

5.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" is open to whoever wants to implement it.

5.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "feature" is open to whoever wants to implement it.

5.1.4 Write Documentation

HistomicsTK could always use more documentation, whether as part of the official HistomicsTK docs, in docstrings, or even on the web in blog posts, articles, and such.

5.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/DigitalSlideArchive/HistomicsTK/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

5.2 Get Started!

Ready to contribute? Here's how to set up *HistomicsTK* for local development.

1. Fork the *HistomicsTK* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/HistomicsTK.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv HistomicsTK
$ cd HistomicsTK/
$ python setup.py develop
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 HistomicsTK tests
$ python setup.py test
$ tox
```

   To get flake8 and tox, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

5.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 2.6, 2.7, and for PyPy. Check https://travis-ci.org/DigitalSlideArchive/HistomicsTK/pull_requests and make sure that the tests pass for all supported Python versions.

5.4 Tips

To run a subset of tests:

```
$ python -m unittest tests.test_HistomicsTK
```

CHAPTER 6

Credits

6.1 Development Lead

- Brian Helba <brian.helba@kitware.com>
- Deepak Roy Chittajallu <deepak.chittajallu@kitware.com>
- Lee Cooper <lee.cooper@emory.edu>
- Zach Mullen <zach.mullen@kitware.com>

6.2 Contributors

None yet. Why not be the first?

CHAPTER 7

History

CHAPTER 8

Indices and tables

- genindex
- modindex
- search

h

L

M

O

P

R

S

T

V

W