HistomicsTK Documentation

Release 0.1.0

Kitware, Inc.

Feb 23, 2017

Contents

HistomicsTK is a Python and REST API for the analysis of Histopathology images in association with clinical and genomic data.

Histopathology, which involves the examination of thin-slices of diseased tissue at a cellular resolution using a microscope, is regarded as the gold standard in clinical diagnosis, staging, and prognosis of several diseases including most types of cancer. The recent emergence and increased clinical adoption of whole-slide imaging systems that capture large digital images of an entire tissue section at a high magnification, has resulted in an explosion of data. Compared to the related areas of radiology and genomics, there is a dearth of mature open-source tools for the management, visualization and quantitative analysis of the massive and rapidly growing collections of data in the domain of digital pathology. This is precisely the gap that we aim to fill with the development of HistomicsTK.

Developed in coordination with the Digital Slide Archive and large_image, HistomicsTK aims to serve the needs of both pathologists/biologists interested in using state-of-the-art algorithms to analyze their data, and algorithm researchers interested in developing new/improved algorithms and disseminate them for wider use by the community.

HistomicsTK can be used in two ways:

- **As a pure Python package**: This is intended to enable algorithm researchers to use and/or extend the analytics functionality within HistomicsTK in Python. HistomicsTK provides algorithms for fundamental image analysis tasks such as color normalization, color deconvolution, cell-nuclei segmentation, and feature extraction. Please see the *api-docs* and *examples* for more information.
- **As a server-side Girder plugin for web-based analysis**: This is intended to allow pathologists/biologists to apply analysis modules/pipelines containerized in HistomicsTK's docker plugins on data over the web. Girder is a Python-based framework (under active development by Kitware) for building web-applications that store, aggregate, and process scientific data. It is built on CherryPy and provides functionality for authentication, access control, customizable metadata association, easy upload/download of data, an abstraction layer that exposes data stored on multiple backends (e.g. Native file system, Amazon S3, MongoDB GridFS) through a uniform RESTful API, and most importantly an extensible plugin framework for building server-side analytics apps. To inherit all these capabilities, HistomicsTK is being developed to act also as a Girder plugin in addition to its use as a pure Python package. To further support web-based analysis, HistomicsTK depends on three other Girder plugins:
  - girder_worker: A Girder plugin for distributed task execution.
  - large_image: A Girder plugin to create/serve/display large multi-resolution images produced by whole-slide imaging systems and a stand-alone Python package for reading these images.
  - slicer_cli_web: A Girder plugin for providing web-based RESTFul access to image analysis pipelines developed as slicer execution model CLIs and containerized using Docker.

This work is funded by the National Cancer Institute (NCI) grant U24-CA194362-01.

CHAPTER 1

Installation

As mentioned in the *index*, HistomicsTK can be used both as a pure Python toolkit for algorithm development and as server-side Girder plugin for web-based analysis. Here, we describe how to install HistomicsTK for both these scenarios.

Installing HistomicsTK as a Python toolkit

HistomicsTK depends on large_image for reading large multi-resolution whole-slide images in a tiled fashion. Please see the Github repo of large_image to find out how to install it as a Python toolkit/package.

HistomicsTK also leverages the functionality of a number of scientific python packages including numpy, scipy, scikit-image, scikit-learn, and pandas. We recommend installing anaconda to ease the cross-platform installation of these packages all of which are listed in requirments_c_conda.txt.

Once large_image is installed as a python package, HistomicsTK can be installed as follows:

```
$ git clone https://github.com/DigitalSlideArchive/HistomicsTK.git
$ cd HistomicsTK
$ conda config --add channels https://conda.binstar.org/cdeepakroy
$ conda install --yes libgfortran==1.0 ctk-cli==1.3.1 --file requirements_c_conda.txt
$ pip install --no-cache-dir -r requirements.txt -r requirements_c.txt
$ python setup.py build_ext --inplace
$ pip install .
```

We are working on releasing HistomicsTK on PyPI so it can easily be pip installed from there.

Installing HistomicsTK as a server-side Girder plugin using Vagrant

When HistomicsTK is used as a server-side Girder plugin for web-based analysis, the following three Girder plugins need to be installed:

- girder_worker: A Girder plugin for distributed task execution.

- large_image: A Girder plugin to create/serve/display large multi-resolution images produced by whole-slide imaging systems and a stand-alone Python package to read/write these images.
- slicer_cli_web: A Girder plugin for providing web-based RESTFul access to image analysis pipelines developed as slicer execution model CLIs and containerized using Docker.

We used Vagrant and Ansible to ease the installation of these plugins in addition to HistomicsTK as follows:

- Download and install virtual box - https://www.virtualbox.org/wiki/Downloads
- Download and install vagrant - https://www.vagrantup.com/downloads.html
- `pip install ansible`
- `git clone https://github.com/DigitalSlideArchive/HistomicsTK.git`
- `cd HistomicsTK && vagrant up`

The Girder instance can then be accessed at http://localhost:8009. Any image placed in the sample_images subdirectory of the directory where HistomicsTK is cloned directory will be seen in the TCGA collection of Girder.

The front-end UI that allows you to apply analysis modules in HistomicsTK's docker plugins on data stored in Girder can be accessed at http://localhost:8009/histomicstk.

You can also ssh into the vagrant virtual box using the command `vagrant ssh`. HistomicsTK and its dependencies are installed at the location `/opt/histomicstk`.

CHAPTER 2

API Documentation

histomicstk

This histomicstk package contains algorithms for fundamentation image analysis tasks including color normalization, color deconvolution, filtering to enhance commonly found objects/structures, cell/nuclei segmentation, and feature extraction.

histomicstk.utils

histomicstk.preprocessing

histomicstk.preprocessing.color_conversion

histomicstk.preprocessing.color_deconvolution

histomicstk.preprocessing.color_normalization

histomicstk.filters

histomicstk.filters.edge

histomicstk.filters.shape

histomicstk.segmentation

Used by autodoc_mock_imports.

histomicstk.segmentation.**label**
Used by autodoc_mock_imports.

histomicstk.segmentation.**label**
Used by autodoc_mock_imports.

histomicstk.segmentation.**label**
Used by autodoc_mock_imports.

histomicstk.segmentation.**label**
Used by autodoc_mock_imports.

histomicstk.segmentation.**label**
Used by autodoc_mock_imports.

histomicstk.segmentation.**label**
Used by autodoc_mock_imports.

histomicstk.segmentation.**label**
Used by autodoc_mock_imports.

histomicstk.segmentation.**label**
Used by autodoc_mock_imports.

histomicstk.segmentation.**label**
Used by autodoc_mock_imports.

** Sub-packages **

histomicstk.segmentation.label

Used by autodoc_mock_imports.

histomicstk.segmentation.label.**trace_boundaries_cython**
Used by autodoc_mock_imports.

histomicstk.segmentation.label.**trace_boundaries_cython**
Used by autodoc_mock_imports.

histomicstk.segmentation.label.**trace_boundaries_cython**
Used by autodoc_mock_imports.

histomicstk.segmentation.label.**trace_boundaries_cython**
Used by autodoc_mock_imports.

histomicstk.segmentation.label.**trace_boundaries_cython**
Used by autodoc_mock_imports.

histomicstk.segmentation.label.**trace_boundaries_cython**
Used by autodoc_mock_imports.

histomicstk.segmentation.label.**trace_boundaries_cython**
Used by autodoc_mock_imports.

histomicstk.segmentation.label.**trace_boundaries_cython**
Used by autodoc_mock_imports.

histomicstk.segmentation.label.**trace_boundaries_cython**
Used by autodoc_mock_imports.

histomicstk.segmentation.label.**trace_boundaries_cython**
Used by autodoc_mock_imports.

histomicstk.segmentation.label.**trace_boundaries_cython**
Used by autodoc_mock_imports.

`histomicstk.segmentation.label.`**`trace_boundaries_cython`**
Used by autodoc_mock_imports.

`histomicstk.segmentation.label.`**`trace_boundaries_cython`**
Used by autodoc_mock_imports.

`histomicstk.segmentation.label.`**`trace_boundaries_cython`**
Used by autodoc_mock_imports.

`histomicstk.segmentation.label.`**`trace_boundaries_cython`**
Used by autodoc_mock_imports.

`histomicstk.segmentation.label.`**`trace_boundaries_cython`**
Used by autodoc_mock_imports.

`histomicstk.segmentation.label.`**`trace_boundaries_cython`**
Used by autodoc_mock_imports.

`histomicstk.segmentation.label.`**`trace_boundaries_cython`**
Used by autodoc_mock_imports.

`histomicstk.segmentation.label.`**`trace_boundaries_cython`**
Used by autodoc_mock_imports.

histomicstk.segmentation.level_set

histomicstk.segmentation.nuclear

histomicstk.features

CHAPTER 3

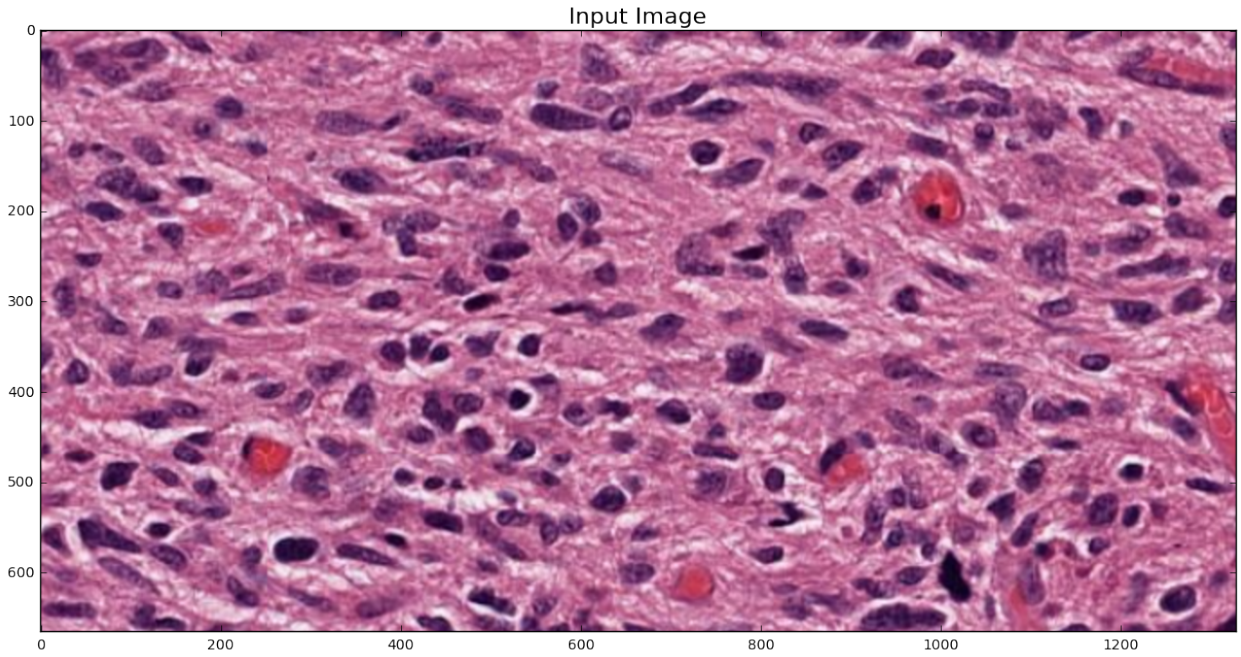
Examples

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
```

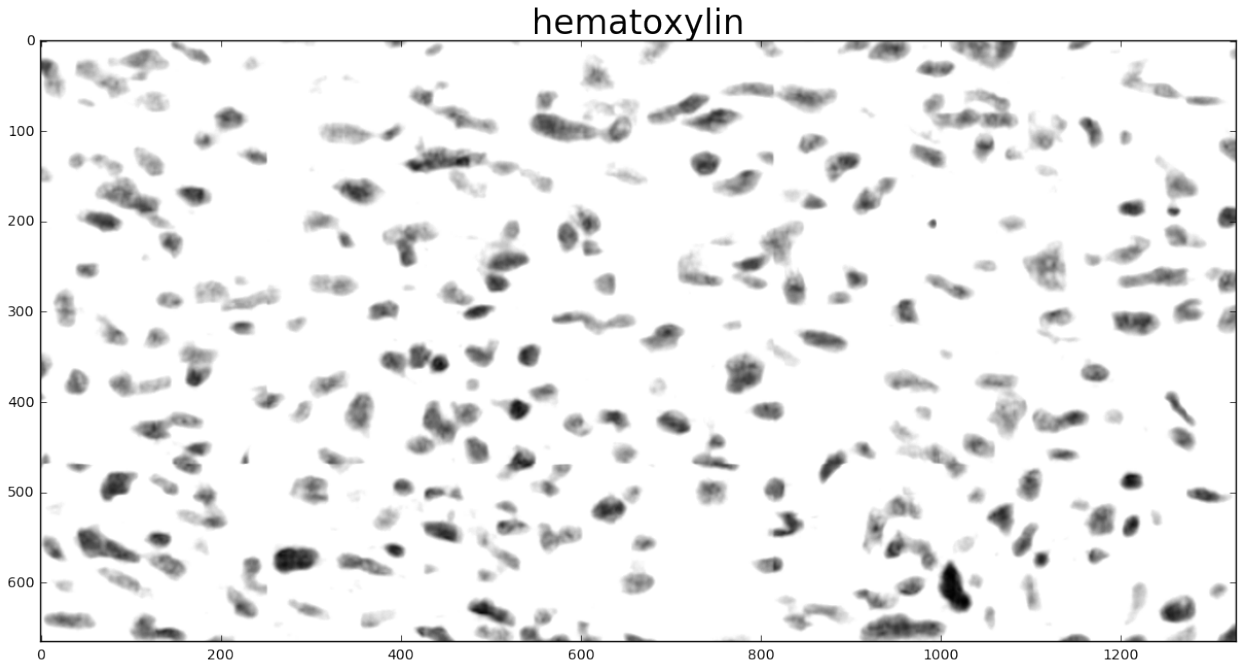
Load input image

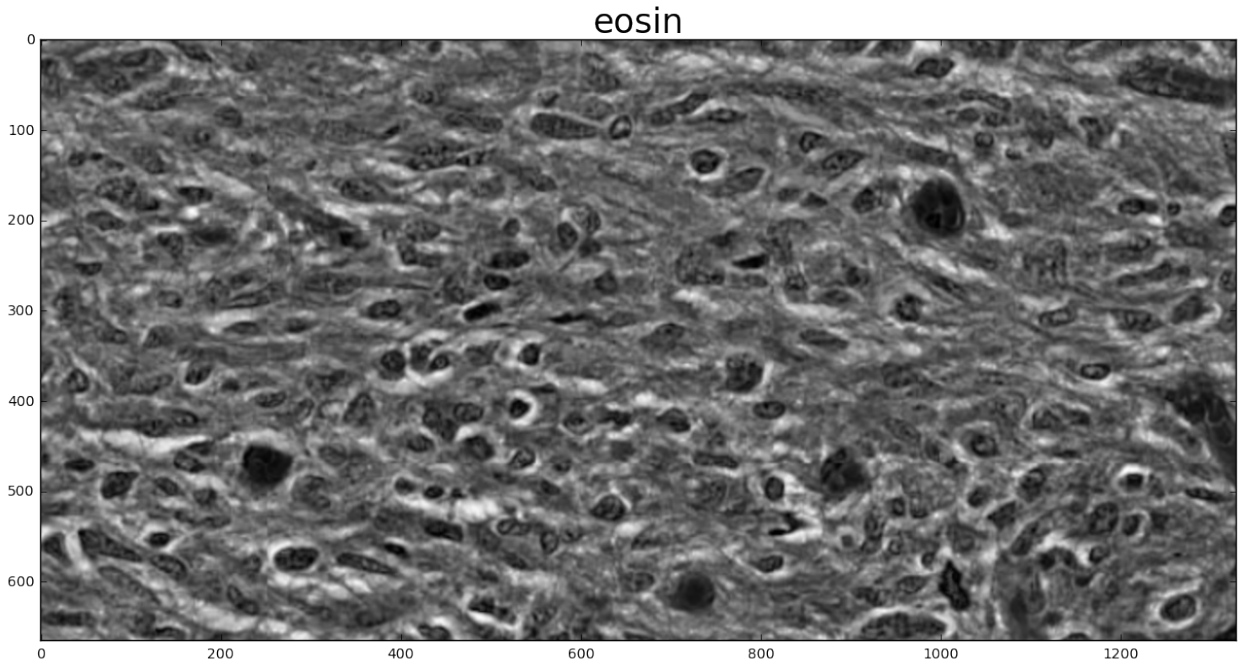
```
In [51]: inputImageFile = ('https://data.kitware.com/api/v1/file/'
                           '57802ac38d777f12682731a2/download')  # H&E.png

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

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



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, W)

         # 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)
```



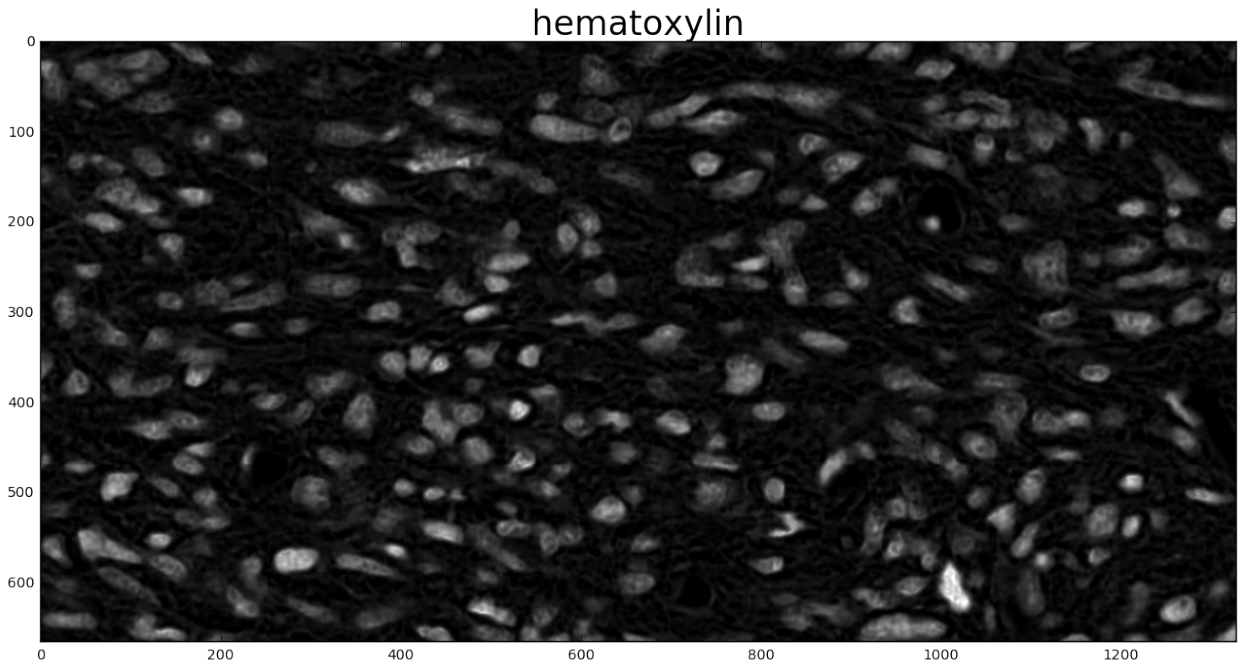


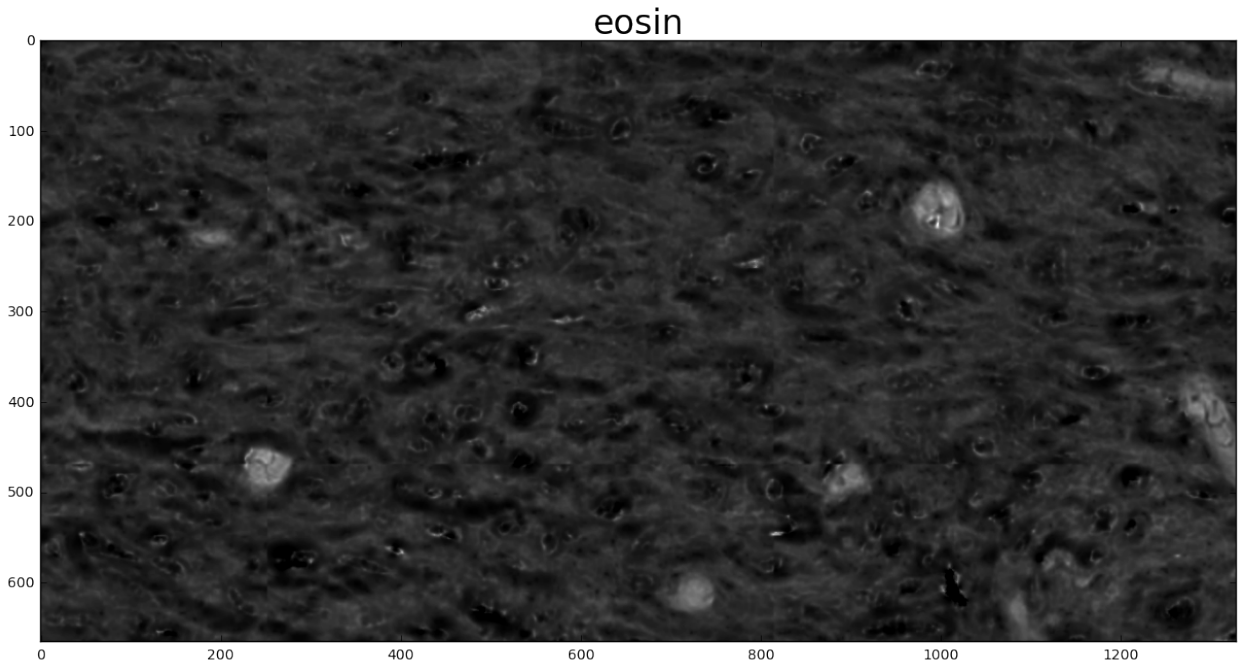
Unsupervised color deconvolution using sparse non-negative matrix factorization

```
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.SparseColorDeconvolution(
```

```
            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]]
```



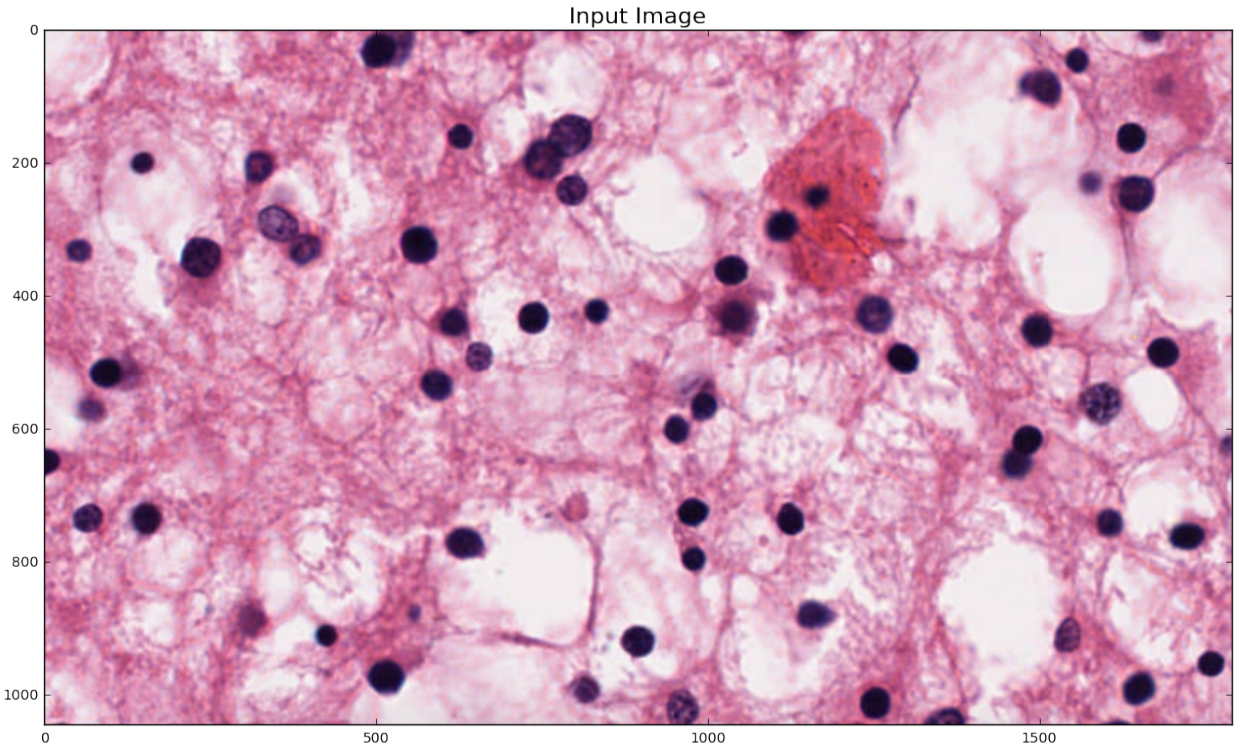


Nuclei Segmentation

```
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
```

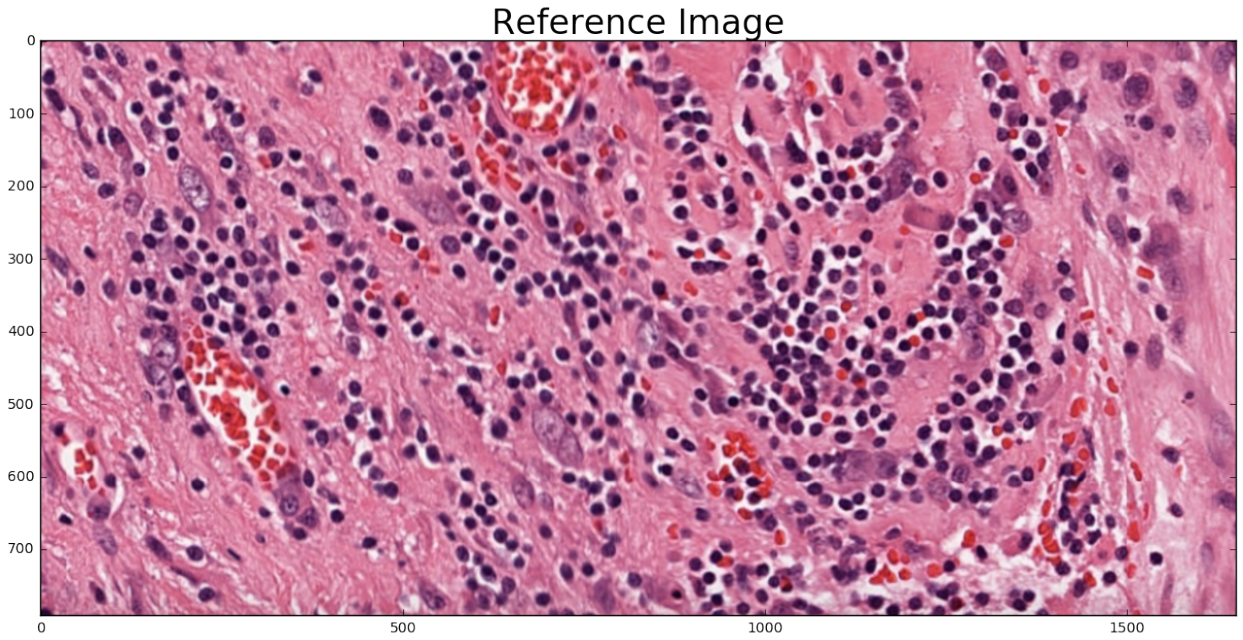
Load input image

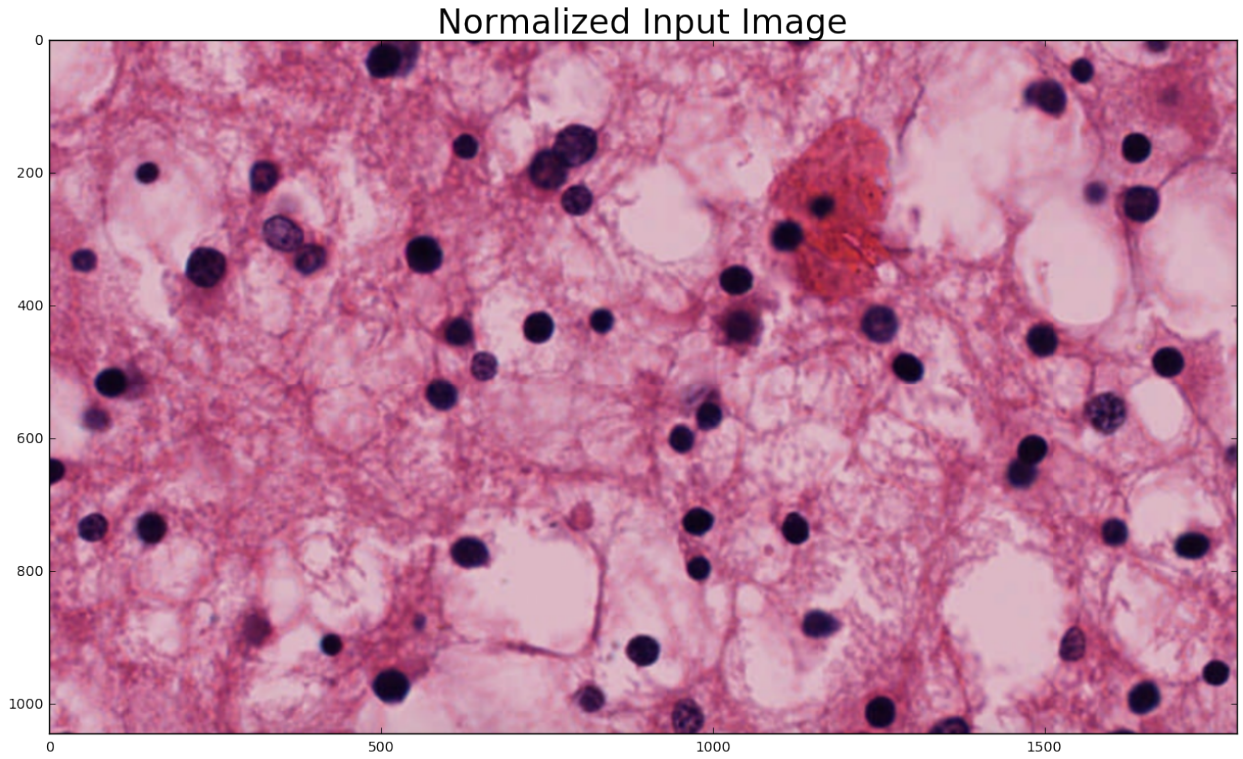
```
In [28]: inputImageFile = ('https://data.kitware.com/api/v1/file/'
                           '576ad39b8d777f1ecd6702f2/download')  # Easy1.png

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

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



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)
```

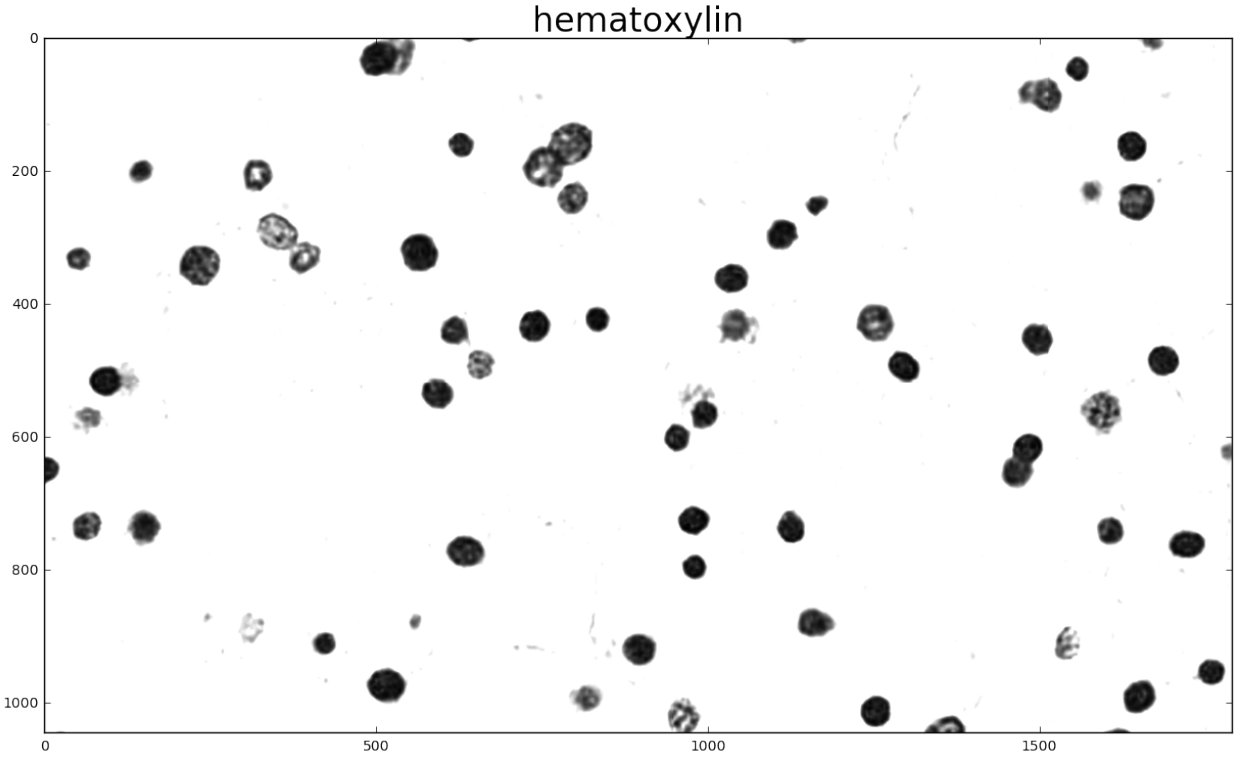


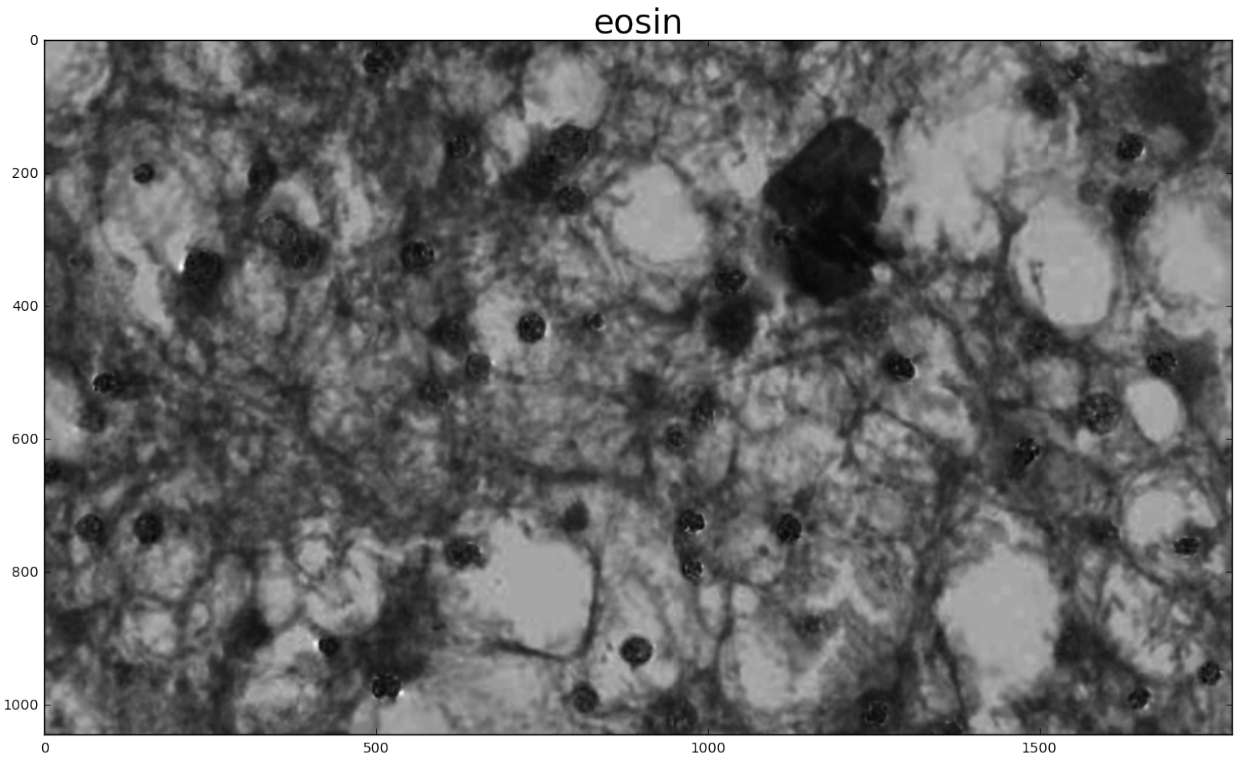


Perform color deconvolution

```
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)
```



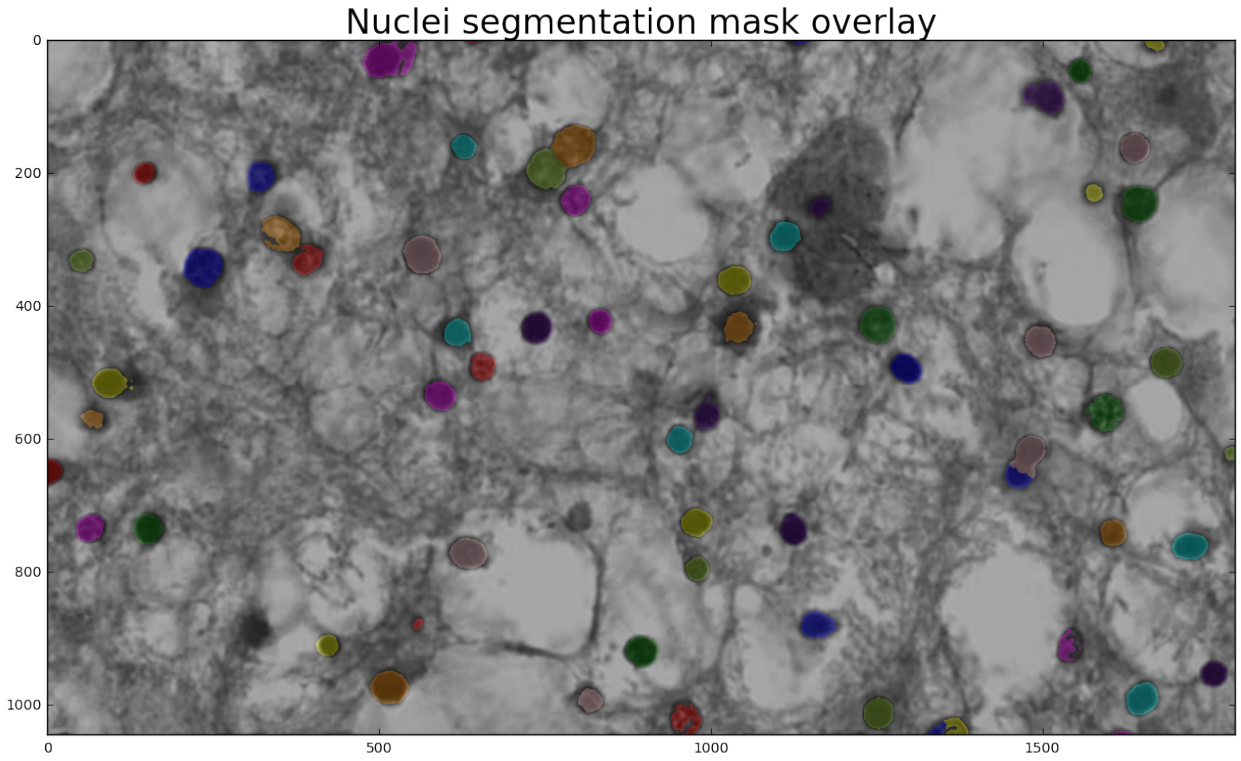


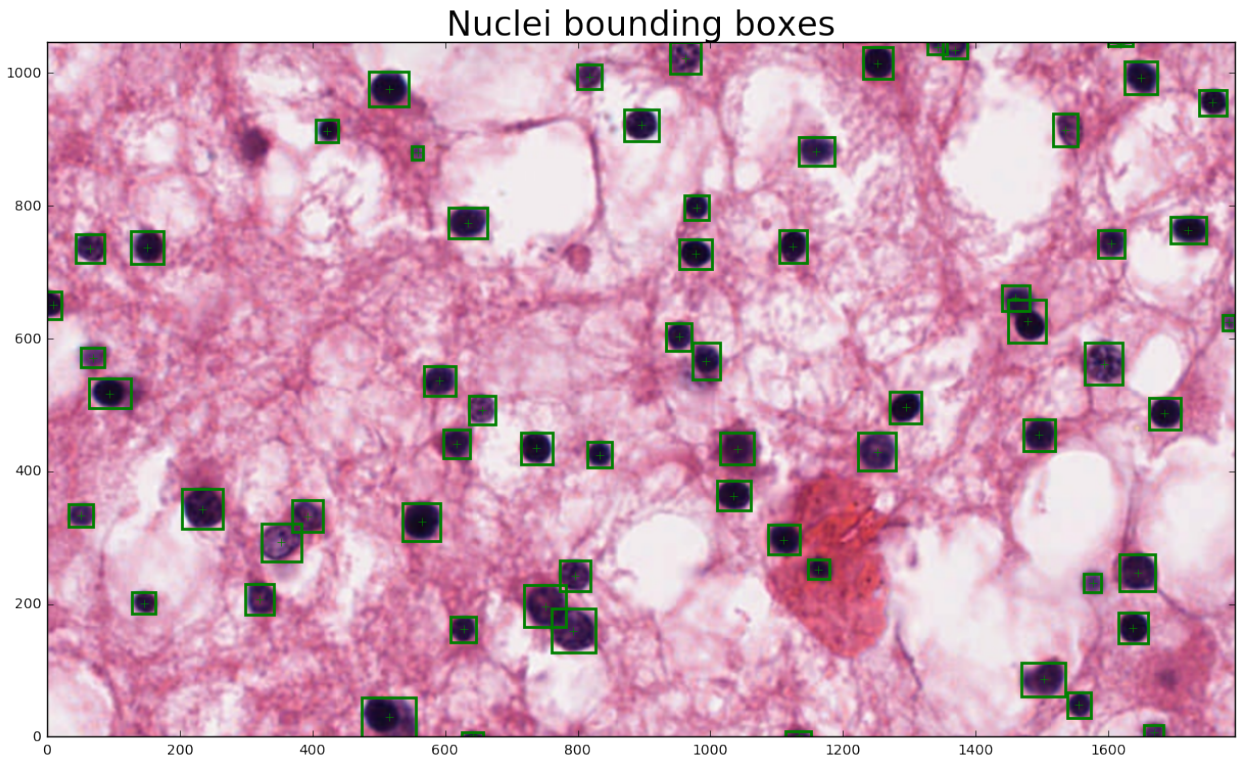
Segment Nuclei

```
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
1000
800
600
400
200
0
0
200
400
600
800
1000
1200
1400
1600

CHAPTER 4

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:

Types of Contributions

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.

Fix Bugs

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

Implement Features

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

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.

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 :)

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. 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.

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.

CHAPTER 5

Credits

Development Lead

- Brian Helba <brian.helba@kitware.com>
- David Gutman <david.gutman@kitware.com>
- David Manthey <david.manthey@kitware.com>
- Deepak Roy Chittajallu <deepak.chittajallu@kitware.com>
- Jonathan Beezeley <jonathan.beezeley@kitware.com>
- Lee Cooper <lee.cooper@emory.edu>
- Sanghoon Lee <sanghoon.lee@emory.edu>
- Zach Mullen <zach.mullen@kitware.com>

Contributors

None yet. Why not be the first?

CHAPTER 6

Indices and tables

- genindex
- modindex
- search

Python Module Index

h

Index