DP2 Distributed Segmentation

Contact

Author: Richard Giuly
rgiuly@gmail.com
Check for updates here: https://sites.google.com/site/slashsegmentation/

Installation:

1. Install python with supporting modules
2. Install s3cmd for uploading to Amazon's S3 storage
3. Make sure you have an Amazon AWS account with S3 storage
4. Make sure you have an Amazon Mechanical Turk account as a Requester
5. Download and use the dseg.py script

Python Installation Instructions (Ubuntu Linux)

Install the following modules with synaptic: python, python-numpy, python-wxgtk2.8, python-scipy, python-matplotlib
Install opencv and opencv python bindings using the Ubuntu Software Center
Install itk for python: sudo apt-get install python-insighttoolkit3
Install python-graph

Python Installation Instruction (Redhat Linux)

Note: Redhat installation is more time consuming than Linux due to need for compilation of Wrap ITK and orange

Install enthought python suite version 7.1
Install WrapITK with ITK version 3.17
  Ensure that PyBuffer in installed with WrapITK Discussion about this
Install opencv for python

You'll need to configure environmental variables according to the instructions for each module. Here's an example of how I set my environmental variables assuming that EPD is installed at /usr/local/EPD:

- export PYTHON_INCLUDE_PATH=/usr/local/EPD/include/python2.7
- export PYTHON_LIB_PATH=/usr/local/EPD/lib/python2.7
- export LD_LIBRARY_PATH=/usr/lib64:/lib:/lib64:/usr/lib:/usr/local/EPD/lib:/usr/local/lib/

```
InsightToolkit:/usr/local/lib/InsightToolkit/WrapITK/lib
➢     export PATH=/usr/local/EPD/bin:$PATH
```

Usage Example

1. After downloading and expanding the source code tar.gz file, move into the dseg directory and create the test_output folder. You will need to be in the dseg directory when running the dseg script.

```
➢     tar -pxzf imageprocessing.tar.gz
➢     cd imageprocessing
➢     cd dseg
➢     mkdir test_output
```

2. Use the following commands from the imageprocessing/dseg directory to create the Mechanical Turk HITS

These commands will use the stack of png images in the folder test_data to create the images for Mechanical Turk and upload them to S3 storage. The --xyprocess option specifies merge decisions in XY planes. The --zprocess option specifies merge decisions from one XY to the adjacent XY plane.

```
➢     python dseg.py test_data test_output --xyprocess
➢     python dseg.py test_data test_output --zprocess
```

The XY process will generate this file, which defines the HITs for Mechanical Turk. You will upload this csv file to mechanical turk for the xy process:
imageprocessing/dseg/test_output/TestDataName_xy/image_in_plane.csv

The Z process will generate this file, which defines the HITs for Mechanical Turk. You will upload this csv file to mechanical turk for the z process:
imageprocessing/dseg/test_output/TestDataName_z/image_plane_to_plane.csv

3. Optional: create qualifications

These commands will use the stack of png images in the folder test_data to create a qualification test for Mechanical Turk. The --xyprocess option specifies merge decisions in XY planes. The --zprocess option specifies merge decisions from one XY to the adjacent XY plane. You will need to specify your access key and your secret key, which are available on Amazon's AWS website.

```
➢     python dseg.py test_data test_output --xyqual --answers
answers_in_plane_example1.txt --access_key YOURACCESSKEY --secret_key
YOURSECRETKEY

➢     python dseg.py test_data test_output --zqual --answers
answers_plane_to_plane_example1.txt --access_key YOURACCESSKEY --secret_key
YOURSECRETKEY
```

Note 1: Examples above assume you want to put results in test_output. You may specify any folder.

Note 2: The option --answers specifies what the answers should be for examples. You will need to set these answers properly by editing the file. To do this, create the qualification and view it. You will be able to see every example. Then use this to fill in the file correctly. Finally, run the command again to create the qualification with the correct Yes/No answers.
Note 3: When creating your input image stack, you should use 8 bit images with good contrast.

4. Creating your tasks in Mechanical Turk: https://www.mturk.com

In Mechanical Turk, you will need a Design for both XY type and Z type questions. NOTE: You must specify two users per HIT.

Here's an example layout for Z questions in HTML:

```
      <h2>Does the dot stay inside of the cell? (Note: If the dot goes on the border between cells, the answer is No.)</h2>
      <h3>&nbsp;&nbsp;</h3>
      <p><input type="radio" value="Yes" name="same_object" /> Yes</p>
      <p><input type="radio" value="No" name="same_object" /> No</p>
      <p><img src="${image1}" style="margin-right: 30px;" alt="image1" /></p>
```

Here's an example layout for XY questions in HTML:

```
      <h2>Are the dots both inside of the same cell? (Note: if one or both dots are on a cell border, the answer must be No.)</h2>
      <p>(Note: You can look at the information at the bottom of the page to see what a cells and boundaries look like.)</p>
      <p><input type="radio" name="same_object" value="Yes" /> Yes</p>
      <p><input type="radio" name="same_object" value="No" /> No</p>
      <p><img alt="image1" style="margin-right: 30px;" src="${image1}" /></p>
```

Note: Make sure you set the number of users assigned to each HIT to 2.
Note: If you're using qualification, you'll need to set a qualification requirement for the job, which was generated in the previous step.

5. Run your jobs in mechanical turk and you will get csv result files.

6. View your results as images:

This command will render the results for merges in xy planes. In this example, Batch_results1.csv is the results file from Mechanical Turk.

```
➢    python dseg.py test_data test_output --xyrender Batch_results1.csv
```

This command will render the results for merges for one plane to the adjacent plane. In this example, Batch_results2.csv is the results file from Mechanical Turk.

```
➢    python dseg.py test_data test_output --zrender Batch_results2.csv
```

Output will be created as an image stack label volume here:
test_output/TestDataName_render/seg

Rendering test data included for Bioinformatics publication:

The results from the test included in the 2012 Bioinformatics paper can be rendered with this command.

From the imageprocessing/dseg directory:

```
➢    python dseg.py data /tmp --zrender csv/Batch_908647_batch_results.csv
csv/Batch_911615_batch_results.csv
```

Note that this will use the results from an Amazon Mechanical Turk job that was previously run.

Segmentation output will be created as an image stack label volume here:
/tmp/TestDataName_render/seg
