User Manual for Superpixel Evaluation

Wei-Chih Tu

December 2017

1 Overview

The main evaluation code is written in C++ with a Python interface. Functions for computing the ASA and BR scores are included in the C++ library. The script eval.py is for evaluating the results in a folder. Another python script eval_par.py for parallel evaluation is also provided.

2 Compiling the C++ Library

Go to /cpp and run

```
python compile_cpp.py build
```

After compilation, a C++ library called EvalSPModule.so will be generated.

3 Usage of EvalSPModule.so

There are two functions in EvalSPModule.so. One is computeASA() and the other is computeBR(). To use these functions, the superpixel labels are loaded in a Python script and converted into an integer list. For example:

```
from EvalSPModule import *
import cv2

# Assume the superpixel labels are saved as a single
# channel 16-bit png file
spLabel = cv2.imread('spLabel.png', -1)
spList = spLabel.faltten().tolist()

# GT segmentation map
gtLabel = cv2.imread('gtLabel.png', -1)
gtList = gtLabel.faltten().tolist()
h, w = gtLabel.shape

# Compute ASA
asa = computeASA(spList, gtList, 0)

# Compute BR
r = 1
br = computeBR(spList, gtList, h, w, r)
```

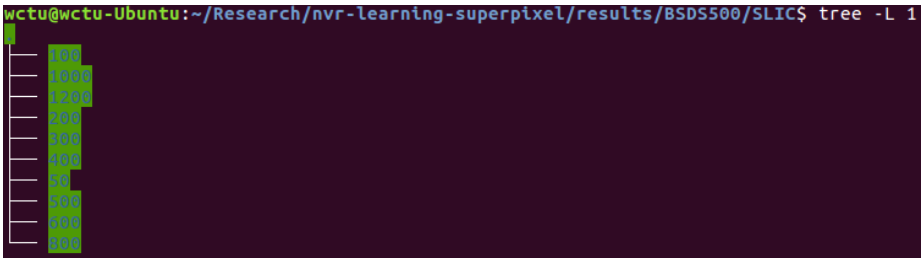

Figure 1: Folder layout of the superpixel labels.

The third argument of computeASA() indicates whether to return a segmentation error list or not. It is used for the training process of the segmentation-aware loss. For the evaluation mode, just set the argument as 0. If it is set as 1, the function returns the ASA score as well as a error list. The error map can be further reshaped as an error map.

```
asa, errorList = computeASA(spList, gtList, 1)
errorMap = np.reshape(np.asarray(errorList), (h, w))
```

4 Batch Evaluation

The Python script eval.py is used for batch evaluation of a folder. The script finds GT maps in gtseg_dir and evaluates scores for superpixel labels in label_dir. By default, all maps are assumed to be saved as 16-bit png files.

The Python script eval_par.py is used to evaluate the results of different number of superpixels for the same dataset. You may need to run

```
pip install joblib
```

first to enable the use of multi-threading in the script. Note that nC_list in the script defines the target superpixel numbers.

```
nC_list = [100, 200, 300, 400, 500, 600]
```

There should be corresponding sub-folders in the root result folder as shown in Figure 1 and each sub-folder should contain resulting superpixel labels for all images in the dataset.