User Manual for the Modified ERS

Wei-Chih Tu

December 2017

1 Introduction

We modify the ERS[1] algorithm so that it takes pixel affinities as input to generate superpixels. We also write a Python interface for the modified ERS algorithm. To compile the code, go to /ERS_Python and run

```
python compile_cpp.py build
```

After compilation, a library file called ERSModule.so will be generated.

2 Usage of ERSModule.so

Two functions are provided in the ERS library. One is ERS() for the original algorithm, which takes a color image as input and produce superpixel labels. The other is ERSWgtOnly(), which takes pixel affinities as input to compute superpixels. Here is an example of using ERS():

```
from ERSModule import *
import cv2
import numpy as np

nC = 300
conn8 = 1
lamb = 0.5    # lambda
sigma = 5.0

img = cv2.imread('input.png')
h = img.shape[0]
w = img.shape[1]

# Convert input image into a 1D list
img_list = img.flatten().tolist()
label_list = ERS(img_list, h, w, nC, conn8, lamb, sigma)
label = np.reshape(np.asarray(lab_list), (h, w));
```

Note that the Python interface takes list as input, so we need to convert the numpy array into a list. The function also returns a list, so we need to convert it back to numpy array.

[1] https://github.com/mingyuliutw/EntropyRateSuperpixel

Here is another example for ERSWgtOnly():

```
from ERSModule import *
import cv2
import numpy as np
import math

nC = 300
conn8 = 1
lamb = 0.5    # lambda

img = cv2.imread('input.png')
h = img.shape[0]
w = img.shape[1]

img = np.float32(img)
img = img / 255.
sigma = 5./255.
twoSigmaSquare = 2.0*sigma*sigma

# Compute pixel affinities from gradients
wgt = np.zeros([2, h, w])
for y in range(0, h-1):
    for x in range(0, w-1):
        dist = (np.absolute(img[y, x, 0]-img[y, x+1, 0]) +
                np.absolute(img[y, x, 1]-img[y, x+1, 1]) +
                np.absolute(img[y, x, 2]-img[y, x+1, 2])) / 3.0
        wgt[0, y, x] = math.exp(-dist*dist/twoSigmaSquare)
        dist = (np.absolute(img[y, x, 0]-img[y+1, x, 0]) +
                np.absolute(img[y, x, 1]-img[y+1, x, 1]) +
                np.absolute(img[y, x, 2]-img[y+1, x, 2])) / 3.0
        wgt[1, y, x] = math.exp(-dist*dist/twoSigmaSquare)

wgt_list = wgt.flatten().tolist()
label_list = ERSWgtOnly(wgt_list, h, w, nC, conn8, lamb)
label = np.reshape(np.asarray(lab_list), (h, w));
```

The output is an integer map ranging from 0 to nC - 1. We can save the output label map as single-channel 16-bit png images by

```
cv2.imwrite('label.png', np.uint16(label))
```

To load the 16-bit png image in your code, simply use

```
label = cv2.imread('label.png', -1)
```