```python
import pdb
import os
import numpy as np
import matplotlib.pyplot as plt
import pydensecrf.densecrf as dcrf
from pydensecrf.utils import unary_from_softmax, create_pairwise_bilateral
import PIL.Image as Image
import multiprocessing
from evaluate import fm_and_mae
from skimage.segmentation import slic
from tqdm import tqdm
import cv2
from myfunc import make_graph
from scipy.sparse import coo_matrix, dia_matrix, eye
from scipy.sparse.linalg import inv, spsolve
from functools import reduce

# 7202, 7299 -> 7616
#

theta = 10.0
alpha = 0.99

sal_set = 'ECSSD'
img_root = '../data/datasets/saliency_Dataset/%s/images'%sal_set
prob_root1 = '../ROTS2files/cap-init'
prob_root2 = '../ROTS2files/cls-init'
output_root = '../ROTS2files/init-sp-mr'

if not os.path.exists(output_root):
    os.mkdir(output_root)

files = os.listdir(img_root)


def mr_func(imgs, probs1, probs2):
    _, hh, ww, _ = imgs.shape
    msks = []
    for i, img in enumerate(imgs):
        prob1 = probs1[i]
        prob2 = probs2[i]

        # superpixel
        img_lab = cv2.cvtColor(img, cv2.COLOR_RGB2LAB).astype(np.float) / 255.0
        sp_label = slic(img_lab, n_segments=200, compactness=20)
        # in case of empty superpixels
        sp_onehot = np.arange(sp_label.max() + 1) == sp_label[..., None]
        sp_onehot = sp_onehot[:, :, sp_onehot.sum(0).sum(0) > 0]
        rs, cs, num = np.where(sp_onehot)
        for i, n in enumerate(num):
            sp_label[rs[i], cs[i]] = n
        sp_num = sp_label.max() + 1
        sp_prob1 = []
        sp_prob2 = []
        sp_img = []
        for i in range(sp_num):
            sp_prob1.append(prob1[sp_label == i].mean())
            sp_prob2.append(prob2[sp_label == i].mean())
            # superpixel vector holds Lab value
```

```
60              sp_img.append(img_lab[sp_label == i, :].mean(0,
    keepdims=False))
61          sp_img = np.array(sp_img)
62          sp_prob1 = np.array(sp_prob1)
63          th1 = sp_prob1.mean()
64          sp_prob2 = np.array(sp_prob2)
65          th2 = sp_prob2.mean()
66          seed = np.ones(sp_num)
67          seed[sp_prob1<th1] = 0
68          seed[sp_prob2<th2] = 0
69          # affinity matrix
70          edges = make_graph(sp_label)
71
72          weight = np.sqrt(np.sum((sp_img[edges[:, 0]] - sp_img[edges[:,
    1]]) ** 2, 1))
73          weight = (weight - np.min(weight, axis=0, keepdims=True)) \
74                   / (np.max(weight, axis=0, keepdims=True) -
    np.min(weight, axis=0, keepdims=True))
75          weight = np.exp(-weight * theta)
76
77          W = coo_matrix((
78              np.concatenate((weight, weight)),
79              (
80                  np.concatenate((edges[:, 0], edges[:, 1]), 0),
81                  np.concatenate((edges[:, 1], edges[:, 0]), 0)
82              )))
83          dd = W.sum(0)
84          D = dia_matrix((dd, 0), (sp_num, sp_num)).tocsc()
85
86          optAff = spsolve(D - alpha * W, eye(sp_num).tocsc())
87          optAff -= dia_matrix((optAff.diagonal(), 0), (sp_num, sp_num))
88
89          """stage 2"""
90          fsal = optAff.dot(seed)
91          fsal = (fsal - fsal.min()) / (fsal.max() - fsal.min())
92          th = fsal.mean()
93          fsal[fsal>th] = 1
94          fsal[fsal<=th] = 0
95          msk = np.zeros((hh, ww))
96          for i in range(sp_num):
97              msk[sp_label==i] = fsal[i]
98          msks += [msk]
99      msks = np.stack(msks, 0)
100     return msks
101
102
103 def thisfunc(img_name):
104     img = Image.open(os.path.join(img_root,
    img_name[:-4]+'.jpg')).convert('RGB')
105     ww, hh = img.size
106     img = np.array(img, dtype=np.uint8)
107     probs1 = Image.open(os.path.join(prob_root1, img_name[:-4]+'.png'))
108     probs1 = probs1.resize((ww, hh))
109     probs1 = np.array(probs1)
110     probs1 = probs1.astype(np.float)/255.0
111
112     probs2 = Image.open(os.path.join(prob_root2, img_name[:-4]+'.png'))
113     probs2 = probs2.resize((ww, hh))
114     probs2 = np.array(probs2)
115     probs2 = probs2.astype(np.float)/255.0
```

```
116
117     # superpixel
118     img_lab = cv2.cvtColor(img, cv2.COLOR_RGB2LAB).astype(np.float) / 255.0
119     sp_label = slic(img_lab, n_segments=200, compactness=20)
120     # in case of empty superpixels
121     sp_onehot = np.arange(sp_label.max() + 1) == sp_label[..., None]
122     sp_onehot = sp_onehot[:, :, sp_onehot.sum(0).sum(0) > 0]
123     rs, cs, num = np.where(sp_onehot)
124     for i, n in enumerate(num):
125         sp_label[rs[i], cs[i]] = n
126     sp_num = sp_label.max() + 1
127     sp_prob1 = []
128     sp_prob2 = []
129     sp_img = []
130     for i in range(sp_num):
131         sp_prob1.append(probs1[sp_label == i].mean())
132         sp_prob2.append(probs2[sp_label == i].mean())
133         # superpixel vector holds Lab value
134         sp_img.append(img_lab[sp_label == i, :].mean(0, keepdims=False))
135     sp_img = np.array(sp_img)
136     sp_prob1 = np.array(sp_prob1)
137     th1 = sp_prob1.mean()
138     sp_prob2 = np.array(sp_prob2)
139     th2 = sp_prob2.mean()
140     seed = np.ones(sp_num)
141     seed[sp_prob1<th1] = 0
142     seed[sp_prob2<th2] = 0
143     # bg1 = np.zeros(sp_num)
144     # bg2 = np.zeros(sp_num)
145     # bg1[sp_prob1<th1] = 1
146     # bg2[sp_prob2<th2] = 1
147
148     # affinity matrix
149     edges = make_graph(sp_label)
150     # edges = np.concatenate((np.stack((np.arange(sp_num),
    np.arange(sp_num)), 1), edges), 0)
151
152     weight = np.sqrt(np.sum((sp_img[edges[:, 0]] - sp_img[edges[:, 1]])
    ** 2, 1))
153     weight = (weight - np.min(weight, axis=0, keepdims=True)) \
154              / (np.max(weight, axis=0, keepdims=True) - np.min(weight,
    axis=0, keepdims=True))
155     weight = np.exp(-weight * theta)
156
157     W = coo_matrix((
158         np.concatenate((weight, weight)),
159         (
160             np.concatenate((edges[:, 0], edges[:, 1]), 0),
161             np.concatenate((edges[:, 1], edges[:, 0]), 0)
162         )))
163     dd = W.sum(0)
164     D = dia_matrix((dd, 0), (sp_num, sp_num)).tocsc()
165
166     optAff = spsolve(D - alpha * W, eye(sp_num).tocsc())
167     optAff -= dia_matrix((optAff.diagonal(), 0), (sp_num, sp_num))
168
169     # """stage 1"""
170     # bds = [bg1, bg2]
171     # bsal = []
172     # for bd in bds:
```

```
173     #     seed = np.zeros(sp_num)
174     #     seed[bd] = 1
175     #     _bsal = optAff.dot(seed)
176     #     _bsal = (_bsal - _bsal.min()) / (_bsal.max() - _bsal.min())
177     #     bsal.append(1 - _bsal)
178     # bsal = reduce(lambda x, y: x * y, bsal)
179     # bsal = (bsal - bsal.min()) / (bsal.max() - bsal.min())
180 
181     """stage 2"""
182     fsal = optAff.dot(seed)
183     fsal = (fsal - fsal.min()) / (fsal.max() - fsal.min())
184 
185     msk = np.zeros((hh, ww))
186     for i in range(sp_num):
187         msk[sp_label==i] = fsal[i]
188     msk = (msk*255).astype(np.uint8)
189     msk = Image.fromarray(msk)
190     msk.save(os.path.join(output_root, img_name[:-4]+'.png'), 'png')
191 
192 
193 if __name__ == '__main__':
194     # for file in tqdm(files):
195     #     thisfunc(file)
196 
197     print('start crf')
198     pool = multiprocessing.Pool(processes=8)
199     pool.map(thisfunc, files)
200     pool.close()
201     pool.join()
202     print('done')
203     fm, mae, _, _ = fm_and_mae(output_root,
    '../data/datasets/saliency_Dataset/%s/masks'%sal_set)
204     print(fm)
205     print(mae)
206 
```