**Algorithm interface definition**

**Interface**

|  |
| --- |
| seg\_image( input\_img, process\_info) |
| input\_img: input image (type: ndarray)  process\_info: processing flow list (type: list), more below |

**Assumption on the image:**

The object is entirely captured in the images, and placed not at the corners of images. There should be less pixel intensity variation on background.

**Processing flow is constructed by algorithm cascading:**

Choose one processing unit from each column,

|  |  |  |
| --- | --- | --- |
| **First stage segmentation** | **Detecting holes** | **Refining segmentation** |
| Felzenswalb | GrabCut | GrowCut |
| Meanshift | Random Walk |

**Processing information structure:**

|  |
| --- |
| process\_info = [1, MeanShift\_para, 1, GrabCut\_para, 1, GrowCut\_para] |

Where MeanShift\_para stores parameters required for MeanShift algorithm, similar for Felzenswalb, GrabCut, GrowCut and Random Walk. The detail of parameter will be explained later. The type of MeanShift\_para and alike are tuple.

The number before each processing unit parameter are the down-sampling or up-sampling ratio. In this example, there is not any subsampling happened. But for example [0.5,X, 0.5,X], where X is some processing unit, the output image and mask is 0.25=0.5\*0.5 of the original one. The algorithm automatically rescales mask back to its original size, hence there is no explicit rescaling parameter in the final stage.

**Processing unit details:**

**Felzenswalb referred as “Fel” in the code**

|  |  |
| --- | --- |
| Arguments | Notes |
| largeness | Controlling the size of components  Range: positive natural number, typically 100 - infinity |
| gauss\_sigma | Controlling Gaussian blurring effect  Range: positive real number, typically 0.3 – 4 |
| br | Used to determine background, first drawing the diagonal line across image, and on the diagonal deciding 4 points with distance br\*(length of diagonal). Then enforcing those four points as background.  Range: 0-0.5, typically 0.1 |

Example: Fel\_para = ('Fel', 500, 0.5, 0.1)

Link on the implementation and paper:

<http://scikit-image.org/docs/dev/api/skimage.segmentation.html#skimage.segmentation.felzenszwalb>

**MeanShift reference by “MeanShift” in the code**

|  |  |
| --- | --- |
| Arguments | Notes |
| spatial\_radius | Determining bandwidth of spatial kernel (Epanechnikov kernel, see lib code for more) , higher value gives more tolerance on spatial variance  Range: positive number, typically 1 – 20 |
| range\_radius | Determining bandwidth of intensity kernel (Epanechnikov kernel, see lib code for more) , higher value gives more tolerance on intensity variance  Range: positive number, typically 1 – 20 |
| min\_density | Used to threshold the regions, since meanshift is essentially a mode estimation method on pixel pdf. Some threshold is required to group pixel into a region, more on the paper  Typical value: 50 |
| br | The same as Fel argument |

Example: 'MeanShift \_para = ('MeanShift', 5, 10, 50, 0.1)

Link on the implementation and paper:

<https://github.com/fjean/pymeanshift>

<https://courses.csail.mit.edu/6.869/handouts/PAMIMeanshift.pdf>

**GrabCut reference by “GrabCut” in the code**

|  |  |
| --- | --- |
| Arguments | Notes |
| num\_iter | Determining number of iteration with GrabCut, higher iteration takes more time  Using opencv implementation with INITIAL\_WITH\_MASK, not rectangle rough gauss |

Example: GrabCut\_para = ('GrabCut', 10)

Link on the implementation and paper:

<http://docs.opencv.org/3.2.0/d8/d83/tutorial_py_grabcut.html>

<https://cvg.ethz.ch/teaching/cvl/2012/grabcut-siggraph04.pdf>

**GrowCut reference by “GrowCut” in the code**

|  |  |
| --- | --- |
| Arguments | Notes |
| safe\_zone\_len | Using output from grabcut, doing erosion on both foreground mask and background mask, such that they generate a fat boundary of unknown region, this parameter determines the size of erosion box |
| window\_size | Size of neighborhood window |
| num\_points | -1: use all eroded fore-mask pixel as seeds for foreground, same for background  >0: number of uniformly sampled points as seeds |

GrowCut\_para = ('GrowCut', 3, 4, -1)

Link on the implementation and paper:

<https://github.com/nfaggian/growcut>

<http://www.graphicon.ru/oldgr/en/publications/text/gc2005vk.pdf>

Also Useful to look at

https://github.com/nfaggian/growcut/blob/master/notebooks/Image%20Segmentation%20-%203%20-%20GrowCut.ipynb

**Random Walk reference by “RandomWalk” in the code**

|  |  |
| --- | --- |
| Arguments | Notes |
| safe\_zone\_len | Using output from grabcut, doing erosion on both foreground mask and background mask, such that they generate a fat boundary of unknown region, this parameter determines the size of erosion box |
| beta | Random walk compute the probability of a random walking from a unlabeled pixel to labelled pixel, and then do the argmax  the greater beta, the more difficult the diffusion |
| num\_points | -1: use all eroded fore-mask pixel as seeds for foreground, same for background  >0: number of uniformly sampled points as seeds |

RandomWalk\_para = ('RandomWalk', 3, 130, -1)

Link on the implementation and paper:

<http://scikit-image.org/docs/dev/api/skimage.segmentation.html#skimage.segmentation.random_walker>

cns-web.bu.edu/~lgrady/grady2006random.pdf

**Notes on constructing processing flow**

Some suggestions to choose flow combination

1. if image is dark, and you can find a path from background to shadow to object, then use meanshift, subsample in first stage.

2. if object has rigid and clear boundary, then use fel, subsample in the second stage, and do not down-sample too much on the first stage

3. for growcut and randomwalk, non sampling version usually runs faster, no much difference between sampled version

4. for growcut and randomwalk, not much difference in results between two algorithms, but prefer growcut, since randomwalk sometimes gives bad results