CS-663 Assignment 3 Q2

Soham Naha (193079003)
Akshay Bajpai (193079002)
Mohit Agarwala (19307R004)

October 18, 2020

2 (45 points) Image Segmentation using Mean Shift.

Input images:

- *2/data/baboonColor.png*
- *2/data/bird.jpg*
- *2/data/flower.jpg*

Take each image, smooth it using Gaussian convolution with a standard deviation of 1 pixel-width unit, and sub-sample the smoothed image by a factor of 2 in each spatial dimension to produce a smaller image. Use these smaller-sized images for the following experiment. If these images still lead to a computational cost that is beyond your computer's capabilities to allow for a sufficiently comprehensive experimentation, then you may resize further.

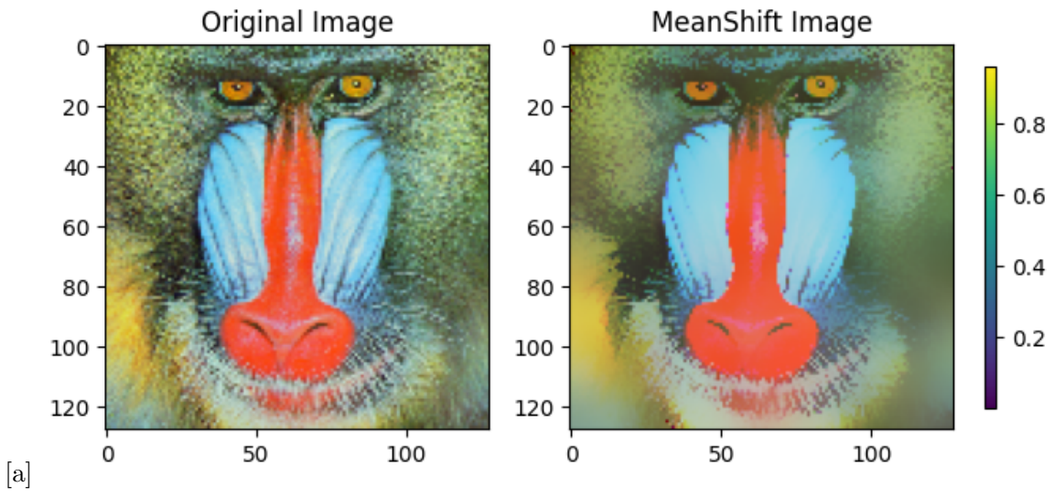
- (24 points) Write a function myMeanShiftSegmentation.m to implement the algorithm for mean-shift image segmentation using both color (RGB) and spatial-coordinate (XY) features. Tune parameters suitably to get a segmented image with at least 5 segments and no more than 50 segments. To improve code efficiency, you may use Matlab functions like *knnsearch(), bsxfun()*, etc. For these images, about 20–40 iterations should be sufficient for reaching close to convergence, but feel free to tune this stopping criterion as suitable. You may select a random subset of nearest neighbors, in feature space, for the mean-shift updates to reduce running time. Each iteration can run in about 10-40 seconds on a typical personal computer.
- (12 points) For each input image, display the (i) original image along with (ii) the segmented image that shows color-coded pixels (and, thus, segments) using the color component of the converged feature vectors.
- (9 points) For each input image, report the following parameter values:
  - i Gaussian kernel bandwidth for the color feature
  - ii Gaussian kernel bandwidth for the spatial feature
  - iii number of iterations.

myMeanShiftSegmentation.py

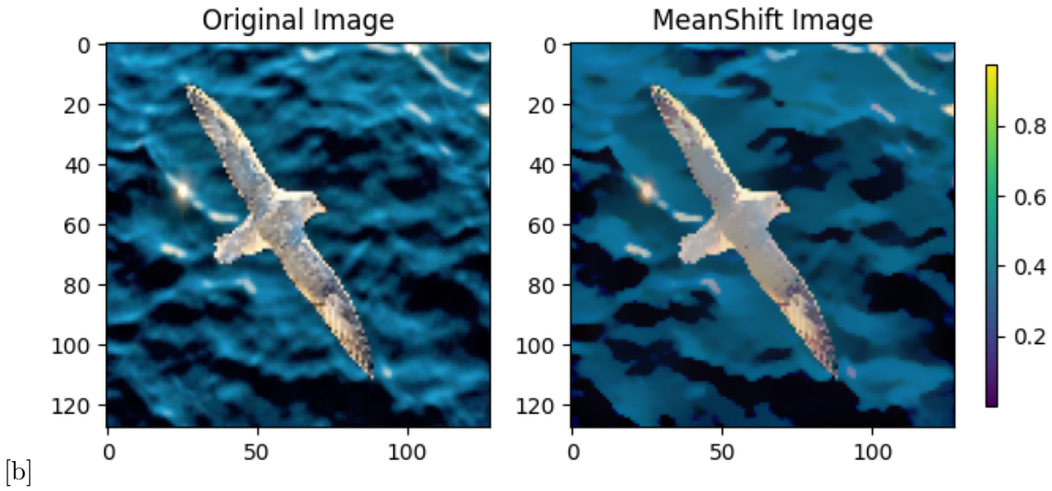
```python
import cv2
import numpy as np
import sys,os
from tqdm import tqdm
from matplotlib import pyplot as plt

def plot_and_save(output_image,input_image,filename):
    name = filename.rstrip("\n").split("/")[-1].split(".")[0]
    fig,axes = plt.subplots(1,2, constrained_layout=True)
    axes[0].imshow(input_image)
    axes[0].axis("on")
    axes[0].set_title("Original Image")
    im = axes[1].imshow(output_image)
    axes[1].axis("on")
    axes[1].set_title("MeanShift Image")
    cbar = fig.colorbar(im,ax=axes.ravel().tolist(),shrink=0.45)
```

```
17
18      plt.savefig("../images/"+name+"MeanShiftcombined_.png",bbox_inches="tight",pad=-1)
19
20      plt.imsave("../images/" + name+"MeanShift.png",output_image)
21
22  def im2double(im):
23      """Normalize the image using min-max Scaling
24      :param im : the input image
25      :return out : the normalized images
26      """
27      im1 = im[:,:,0]
28      im2 = im[:,:,1]
29      im3 = im[:,:,2]
30      out1 = (im1.astype(np.float64) - np.min(im1.ravel())) / (np.max(im1.ravel()) - np.min(im1.ravel()))
31      out2 = (im2.astype(np.float64) - np.min(im2.ravel())) / (np.max(im2.ravel()) - np.min(im2.ravel()))
32      out3 = (im3.astype(np.float64) - np.min(im3.ravel())) / (np.max(im3.ravel()) - np.min(im3.ravel()))
33      out = cv2.merge((out1,out2,out3))
34      return out
35
36  def meanShift(filename,intensity_sigma=0.1,spatial_sigma=11.0,num_iter=30):
37      print("Running Mean-Shift for file: ",filename)
38      img = cv2.imread(filename)
39      img = cv2.cvtColor(img,cv2.COLOR_BGR2RGB)
40
41      r,c,d = img.shape
42      img = im2double(img)
43      gaussian_blur = cv2.GaussianBlur(img, (5,5), 1.0)
44
45      row,col = 128,128
46      newimg = cv2.resize(img,(row,col))
47
48
49      result1 = np.zeros((row,col))
50      result2 = np.zeros((row,col))
51      result3 = np.zeros((row,col))
52
53      #intensity Gaussian standard deviation
54      h=intensity_sigma
55
56      #spatial Gaussian parameters
57      sigma = spatial_sigma
58      window_size = 7
59      padded = np.concatenate((np.concatenate((np.zeros((row,window_size,3)), newimg),axis=1),
60              np.zeros((row,window_size,3))),axis=1)
61      padded = np.concatenate((np.concatenate((np.zeros((window_size,col+2*(window_size),3)),padded),axis=0),
62      np.zeros((window_size, col+2*(window_size),3))),axis=0)
63      #padded = np.zeros((row+2*window_size,col+2*window_size))
64      r,c,d = padded.shape
65      #r,c = padded.shape
66      print(padded.shape)
67
68      #isotropic
69      #Spatial
70      spatial = np.zeros((2*window_size+1,2*window_size+1))
71      for is1 in range(2*window_size+1):
72          for is2 in range(2*window_size+1):
73              spatial[is1][is2] = ((is1-window_size)**2+(is2-window_size)**2)**0.5
74      spatial = np.exp(-(spatial/sigma)**2)
75
76
77      for idx1 in tqdm(range(window_size,r-window_size)):
78          for idx2 in range(window_size,c-window_size):
79              window = padded[idx1-window_size:idx1+window_size+1, idx2-window_size:idx2+window_size+1]
80              #print(newimg[idx1][idx2])
81              (x1,x2,x3) = newimg[idx1-window_size][idx2-window_size]
```

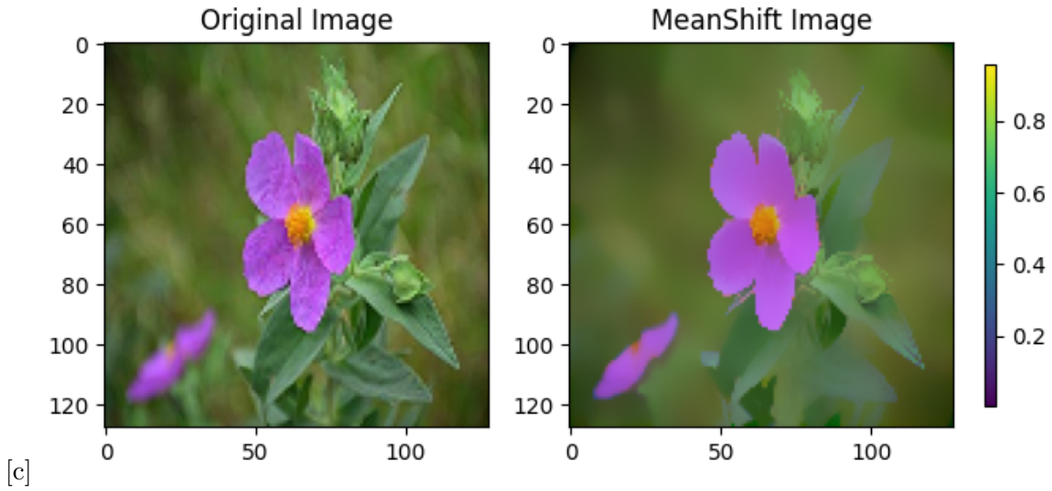
```
            N1 = 0.0 #numerator
            D1 = 1.0 #denominator
            N2 = 0.0 #numerator
            D2 = 1.0 #denominator
            N3 = 0.0 #numerator
            D3 = 1.0 #denominator


            for itern in range(num_iter): # number of iterations
                for idx3 in range(2*window_size+1):
                    for idx4 in range(2*window_size+1):
                        (x_i1,x_i2,x_i3) = window[idx3][idx4]
                        diff1 = abs(x1-x_i1)
                        diff2 = abs(x2-x_i2)
                        diff3 = abs(x3-x_i3)

                        d1 = np.exp(-(diff1/h)**2)*spatial[idx3][idx4]
                        n1 = x_i1*d1
                        N1 += n1
                        D1 += d1

                        d2 = np.exp(-(diff2/h)**2)*spatial[idx3][idx4]
                        n2 = x_i2*d2
                        N2 += n2
                        D2 += d2

                        d3 = np.exp(-(diff3/h)**2)*spatial[idx3][idx4]
                        n3 = x_i3*d3
                        N3 += n3
                        D3 += d3

                x1 = float(N1)/D1 # in each iteration, x changes
                x2 = float(N2)/D2 # in each iteration, x changes
                x3 = float(N3)/D3 # in each iteration, x changes

            result1[idx1-window_size][idx2-window_size] = x1
            result2[idx1-window_size][idx2-window_size] = x2
            result3[idx1-window_size][idx2-window_size] = x3

    result = cv2.merge((result1,result2,result3))
    #print(result)
    plot_and_save(result,newimg,filename)

if __name__=="__main__":
    filename = sys.argv[1]
    meanShift(filename)
```



[a]



[b]



[c]

Figure 1: (a) Mean-Shift Image for 2/data/baboonColor.png (b)Mean-Shift Image for 2/data/bird.jpg (c)Mean-Shift Image for 2/data/flower.jpg

Parameters Used :

[***Note :*** **For all the images, we resized them to 128x128 RGB images for computational conveniences.**]

For all the images, we chose a common parameter of 0.1 as the standard deviation for the intensity gaussian kernel, i.e. in case of the weighted sum.
For all the images we used the standard deviation of the spatial kernel to be 11.0.
Other parameters are :

- *2/data/baboonColor.png* :: num_iterations : 30
- *2/data/bird.jpg* :: num_iterations : 40
- *2/data/flower.jpg* :: num_iterations : 40

myMainScript.py

```
from myMeanShiftSegmentation import meanShift

filenames = ["baboonColor.png","bird.jpg","flower.jpg"]
foldername = "../data/"
iterations = [30,40,40]

for i in range(len(filenames)):
    meanShift(foldername+filenames[i],intensity_sigma=0.1,num_iter=iterations[i])
```