CS558 HW3

Name: Yiqun Peng

Problem 1:

1. Load image from ImClass directory:

   Code:

```
for i in range(int(traning_num / 3)):
    num = str(i+1).zfill(1)
    filename1 = "coast_train" + num
    filename2 = "forest_train" + num
    filename3 = "insidecity_train" + num
```

```
for i2 in range(int(test_num / 3)):
    num = str(i2+1).zfill(1)
    filename1 = "coast_test" + num
    filename2 = "forest_test" + num
    filename3 = "insidecity_test" + num
```

```
def loadImage(labelName,bin):
    imgname = "ImClass/" + labelName + ".jpg"
    pre_image = cv2.imread(imgname)
    row, col, cha = pre_image.shape
```

2. Compute the histograms:

   Code:

```
b, g, r = cv2.split(pre_image)
hist_b = np.zeros([1, bin])
hist_g = np.zeros([1, bin])
hist_r = np.zeros([1, bin])
for i in range(row):
    for j in range(col):
        ind_b = int(np.floor(b[i][j] / (256 / bin)))
        ind_g = int(np.floor(g[i][j] / (256 / bin)))
        ind_r = int(np.floor(r[i][j] / (256 / bin)))
        hist_b[0, ind_b] = hist_b[0, ind_b] + 1
        hist_g[0, ind_g] = hist_g[0, ind_g] + 1
        hist_r[0, ind_r] = hist_r[0, ind_r] + 1
```

3. The Euclidean distance in the 24D histogram space is used to calculate the distance of the three-channel histogram. (Using the k-nearest neighbor classifier):

   Code:

```
def k_nearest_neighbor(training_X, training_y, test_X, test_y, k):
    row, col = test_X.shape
    predicts = []
    for i in range(row):
        dis = np.sqrt(np.sum((test_X[i,:] - training_X) ** 2, axis=1))
        ind = dis.argsort()[0: k]
        candidate = training_y[ind].reshape((-1,),order='F')
        vote = np.bincount(candidate)
        if vote.size == 4 and vote[1]==vote[2] and vote[2]==vote[3]:
            predicts.append(candidate[0])
        else:
            predicts.append(vote.argmax())
        print("Test image",i ,"of class ",test_y[i]," has been assigned to class "
```

4. Compute the accuracy of the classifier:

   Code:

```
error = 0
for i in range(test_num):
    if test_y[i, 0] != predict1[i]:
        error += 1
accuracy = (test_num - error) / test_num
```

Results: (class 1: coast, class 1: forest, class 3: insidecity)

Use 8 bins for each histogram and 1-nearest neighbor classifier:

```
-------------------------
Test image 0 of class  [1]  has been assigned to class  1
Test image 1 of class  [2]  has been assigned to class  2
Test image 2 of class  [3]  has been assigned to class  2
Test image 3 of class  [1]  has been assigned to class  1
Test image 4 of class  [2]  has been assigned to class  2
Test image 5 of class  [3]  has been assigned to class  2
Test image 6 of class  [1]  has been assigned to class  1
Test image 7 of class  [2]  has been assigned to class  2
Test image 8 of class  [3]  has been assigned to class  3
Test image 9 of class  [1]  has been assigned to class  1
Test image 10 of class  [2]  has been assigned to class  2
Test image 11 of class  [3]  has been assigned to class  3
[1, 2, 2, 1, 2, 2, 1, 2, 3, 1, 2, 3]
Accurate( 8  bins with 1-nearest neighbor):  0.83
```

Use 4 bins for each histogram and 1-nearest neighbor classifier:

```
Test image 0 of class  [1]  has been assigned to class  1
Test image 1 of class  [2]  has been assigned to class  2
Test image 2 of class  [3]  has been assigned to class  2
Test image 3 of class  [1]  has been assigned to class  1
Test image 4 of class  [2]  has been assigned to class  2
Test image 5 of class  [3]  has been assigned to class  3
Test image 6 of class  [1]  has been assigned to class  1
Test image 7 of class  [2]  has been assigned to class  2
Test image 8 of class  [3]  has been assigned to class  3
Test image 9 of class  [1]  has been assigned to class  1
Test image 10 of class  [2]  has been assigned to class  2
Test image 11 of class  [3]  has been assigned to class  1
[1, 2, 2, 1, 2, 3, 1, 2, 3, 1, 2, 1]
Accurate( 4  bins with 1-nearest neighbor):  0.83
```

Use 16 bins for each histogram and 1-nearest neighbor classifier:

```
Test image 0 of class  [1]  has been assigned to class  1
Test image 1 of class  [2]  has been assigned to class  2
Test image 2 of class  [3]  has been assigned to class  2
Test image 3 of class  [1]  has been assigned to class  1
Test image 4 of class  [2]  has been assigned to class  2
Test image 5 of class  [3]  has been assigned to class  2
Test image 6 of class  [1]  has been assigned to class  1
Test image 7 of class  [2]  has been assigned to class  2
Test image 8 of class  [3]  has been assigned to class  2
Test image 9 of class  [1]  has been assigned to class  1
Test image 10 of class  [2]  has been assigned to class  2
Test image 11 of class  [3]  has been assigned to class  3
[1, 2, 2, 1, 2, 2, 1, 2, 2, 1, 2, 3]
Accurate( 16  bins with 1-nearest neighbor):  0.75
```

Use 32 bins for each histogram and 1-nearest neighbor classifier:

```
Test image 0 of class  [1]  has been assigned to class  1
Test image 1 of class  [2]  has been assigned to class  2
Test image 2 of class  [3]  has been assigned to class  2
Test image 3 of class  [1]  has been assigned to class  1
Test image 4 of class  [2]  has been assigned to class  2
Test image 5 of class  [3]  has been assigned to class  2
Test image 6 of class  [1]  has been assigned to class  1
Test image 7 of class  [2]  has been assigned to class  2
Test image 8 of class  [3]  has been assigned to class  2
Test image 9 of class  [1]  has been assigned to class  1
Test image 10 of class  [2]  has been assigned to class  2
Test image 11 of class  [3]  has been assigned to class  3
[1, 2, 2, 1, 2, 2, 1, 2, 2, 1, 2, 3]
Accurate( 32  bins with 1-nearest neighbor):  0.75
```

Use 8 bins for each histogram and 3-nearest neighbor classifier:

```
Test image 0 of class  [1]  has been assigned to class  1
Test image 1 of class  [2]  has been assigned to class  2
Test image 2 of class  [3]  has been assigned to class  2
Test image 3 of class  [1]  has been assigned to class  1
Test image 4 of class  [2]  has been assigned to class  2
Test image 5 of class  [3]  has been assigned to class  2
Test image 6 of class  [1]  has been assigned to class  1
Test image 7 of class  [2]  has been assigned to class  2
Test image 8 of class  [3]  has been assigned to class  2
Test image 9 of class  [1]  has been assigned to class  1
Test image 10 of class  [2]  has been assigned to class  2
Test image 11 of class  [3]  has been assigned to class  3
[1, 2, 2, 1, 2, 2, 1, 2, 2, 1, 2, 3]
Accurate( 8  bins with 3-nearest neighbor):  0.75
```

Problem 2:

1. Use gimp to obtain a mask:

2. Separate sky from non-sky pixels:

   Code:

```
def generateTrainingData(trainImag, mask):
    row, col ,cha = trainImag.shape
    sky = []
    non_sky = []
    for i in range(row):
        for j in range(col):
            if mask[i][j][0] == 255 and mask[i][j][1] == 255 and mask[i][j][2] == 255:
                sky.append(trainImag[i][j])
            else:
                non_sky.append(trainImag[i][j])
    return sky, non_sky
```

3. Run k-means separately on the sky and non-sky sets with k = 10 to obtain 10 visual words for each class:

   Code:

```
def generateVisualWords(sky, non_sky):
    sky_10 = KMeans(n_clusters=10)
    non_sky_10 = KMeans(n_clusters=10)
    sky_10.fit(sky)
    non_sky_10.fit(non_sky)
    centroids_sky = sky_10.cluster_centers_
    centroids_non_sky = non_sky_10.cluster_centers_
    sky_label = np.ones((10, 1))
    non_sky_label = np.zeros((10, 1))
    words = np.vstack((centroids_sky, centroids_non_sky))
    labels = np.vstack((sky_label, non_sky_label))
    return words, labels
```

4. For test image find the nearest word and classify it as sky or non-sky:

   Code:

```
for t in range (len(test_img)):
    num = str(t + 1).zfill(1)
    test = test_img[t]
    cv2.imshow("test_" + num, test)
    row, col, cha = test.shape
    for i in range(row):
        for j in range(col):
            minIndex = -1
            minDist = np.inf
            for w in range(len(words)):
                sub = test[i][j][:] - words[w]
                dis = np.sqrt(sub[0] ** 2 + sub[1] ** 2 + sub[2] ** 2)
                if dis < minDist:
                    minDist = dis
                    minIndex = w
```

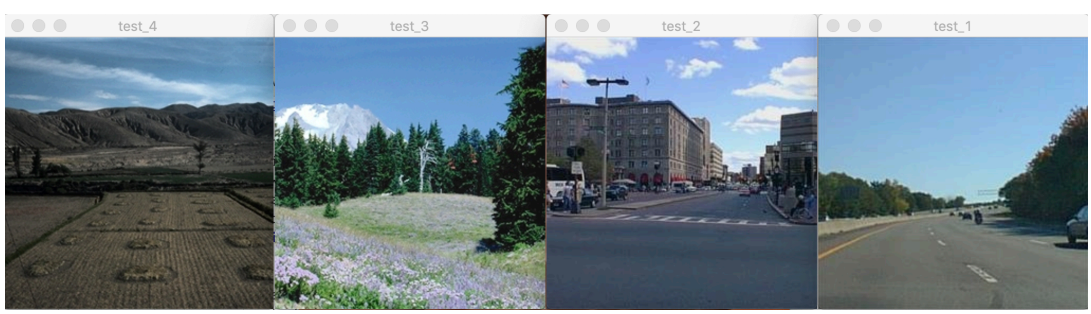
5. Generate an output image in which sky pixels are painted with a distinctive color:

   Code:

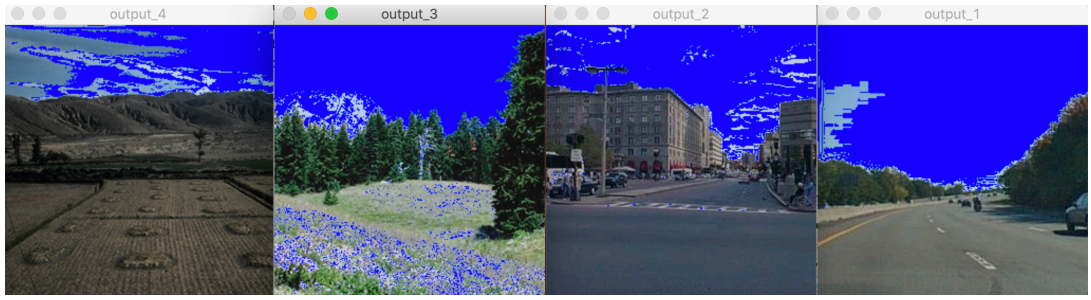
```
            if labels[minIndex] == 1:
                test[i][j][0] = 255
                test[i][j][1] = 0
                test[i][j][2] = 0
cv2.imshow("output_"+num, test)
```

Results:

Test image:



Output image:



Discussion:

It works because, by calculating the distance, each pixel in the test image is divided into its closest visual word label.

As you can see, some parts of the sky are not fully classified as sky word labels (blue), while some non-sky parts (such as zebra crossings) are classified as sky labels (blue). This could be because the distance between the sky words in the pixels of some clouds in the sky is greater than the distance between the sky labels in the pixels of zebra crossing, thus causing this failure.

In order to improve the accuracy of the classifier, the value of k in the k-mean algorithm can be changed to change the vocabulary size, but it should not be too large or too small. Secondly, the number of pixels in the training set can be increased to make the model more powerful.

The Source Code:

Problem 1:

```
from PIL import Image
import cv2
import math
import numpy as np
import matplotlib.pyplot as plt

def loadImage(labelName,bin):
    imgname = "ImClass/" + labelName + ".jpg"
    pre_image = cv2.imread(imgname)
    row, col, cha = pre_image.shape
    b, g, r = cv2.split(pre_image)
    hist_b = np.zeros([1, bin])
    hist_g = np.zeros([1, bin])
    hist_r = np.zeros([1, bin])
    for i in range(row):
        for j in range(col):
            ind_b = int(np.floor(b[i][j] / (256 / bin)))
            ind_g = int(np.floor(g[i][j] / (256 / bin)))
            ind_r = int(np.floor(r[i][j] / (256 / bin)))
            hist_b[0, ind_b] = hist_b[0, ind_b] + 1
            hist_g[0, ind_g] = hist_g[0, ind_g] + 1
            hist_r[0, ind_r] = hist_r[0, ind_r] + 1

    # hist_bb = cv2.calcHist([pre_image],[0],None,[8],[0,255])
    # hist_gg = cv2.calcHist([pre_image],[1],None,[8],[0,255])
    # hist_rr = cv2.calcHist([pre_image],[2],None,[8],[0,255])
    hist = np.hstack((hist_b,hist_g,hist_r))
    # print(hist)
    # print(hist.shape)

    return hist

def k_nearest_neighbor(training_X, training_y, test_X, test_y, k):
    row, col = test_X.shape
    predicts = []
    for i in range(row):
        dis = np.sqrt(np.sum((test_X[i,:] - training_X) ** 2, axis=1))
```

```
        ind = dis.argsort()[0: k]
        candidate = training_y[ind].reshape((-1,),order='F')
        vote = np.bincount(candidate)
        if vote.size == 4 and vote[1]==vote[2] and vote[2]==vote[3]:
            predicts.append(candidate[0])
        else:
            predicts.append(vote.argmax())
        print("Test image",i ,"of class ",test_y[i]," has been assigned to class
",predicts[i])
        # print("dis",i,":",dis)
        # print("sort", i, ":", ind)
        # print("candidate:",candidate)
        # print("vote",vote)
    print(predicts)
    return predicts


if __name__ == "__main__":

    '''using 4, 8, 16, 32 bins'''
    for b in range(2,6):
        bb = 2 ** b
        traning_num = 12
        test_num = 12
        bin = bb
        training_X = np.zeros([traning_num, bin * 3])
        training_y = np.zeros([traning_num, 1],dtype=np.int)
        test_X = np.zeros([test_num, bin * 3])
        test_y = np.zeros([test_num, 1],dtype=np.int)
        iteration = 0

        ''' training data'''
        for i in range(int(traning_num / 3)):
            num = str(i+1).zfill(1)
            filename1 = "coast_train" + num
            filename2 = "forest_train" + num
            filename3 = "insidecity_train" + num
            hist1 = loadImage(filename1, bin)
            training_X[iteration,:] = hist1
            training_y[iteration,:] = 1
            iteration += 1
```

```
        hist2 = loadImage(filename2, bin)
        training_X[iteration, :] = hist2
        training_y[iteration, :] = 2
        iteration += 1
        hist3 = loadImage(filename3, bin)
        training_X[iteration, :] = hist3
        training_y[iteration, :] = 3
        iteration += 1

    ''' test data'''
    iteration2 = 0
    for i2 in range(int(test_num / 3)):
        num = str(i2+1).zfill(1)
        filename1 = "coast_test" + num
        filename2 = "forest_test" + num
        filename3 = "insidecity_test" + num
        hist1 = loadImage(filename1, bin)
        test_X[iteration2, :] = hist1
        test_y[iteration2, :] = 1
        iteration2 += 1
        hist2 = loadImage(filename2, bin)
        test_X[iteration2, :] = hist2
        test_y[iteration2, :] = 2
        iteration2 += 1
        hist3 = loadImage(filename3, bin)
        test_X[iteration2, :] = hist3
        test_y[iteration2, :] = 3
        iteration2 += 1

    ''' do 1-nearest neighbor '''
    predict1 = k_nearest_neighbor(training_X, training_y, test_X, test_y, 1)
    error = 0
    for i in range(test_num):
        if test_y[i, 0] != predict1[i]:
            error += 1
    accuracy = (test_num - error) / test_num
    print("Accurate(",bb ," bins with 1-nearest neighbor): ",
np.round(accuracy,2))
    print("----------------------------")

    if bin == 8:
```

```
        ''' do 3-nearest neighbor '''
        predict3 = k_nearest_neighbor(training_X, training_y, test_X, test_y,
3)
        error = 0
        for i in range(test_num):
            if test_y[i, 0] != predict3[i]:
                error += 1
        accuracy = (test_num - error) / test_num
        print("Accurate(",bb ," bins with 3-nearest neighbor): ",
np.round(accuracy, 2))
        print("---------------------------")
```

Problem2:

```
from PIL import Image
import cv2
import math
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans

def loadImage(labelName):
    imgname = "sky/" + labelName + ".jpg"
    pre_image = cv2.imread(imgname)
    print(labelName,pre_image)
    print(pre_image.shape)
    return pre_image

def generateTrainingData(trainImag, mask):
    row, col ,cha = trainImag.shape
    sky = []
    non_sky = []
    for i in range(row):
        for j in range(col):
            if mask[i][j][0] == 255 and mask[i][j][1] == 255 and mask[i][j][2] ==
255:
                sky.append(trainImag[i][j])
            else:
```

```
            non_sky.append(trainImag[i][j])
    return sky, non_sky

def generateVisualWords(sky, non_sky):
    sky_10 = KMeans(n_clusters=10)
    non_sky_10 = KMeans(n_clusters=10)
    sky_10.fit(sky)
    non_sky_10.fit(non_sky)
    centroids_sky = sky_10.cluster_centers_
    centroids_non_sky = non_sky_10.cluster_centers_
    sky_label = np.ones((10, 1))
    non_sky_label = np.zeros((10, 1))
    words = np.vstack((centroids_sky, centroids_non_sky))
    labels = np.vstack((sky_label, non_sky_label))
    return words, labels


if __name__ == "__main__":
    '''Load image'''
    train_img = loadImage("sky_train")
    mask = loadImage("sky_train_gimp")
    test_img = []
    for i in range(4):
        num = str(i + 1).zfill(1)
        test_img.append(loadImage("sky_test"+num))

    '''genetate training data '''
    sky, non_sky = generateTrainingData(train_img, mask)

    '''generate visual words'''
    words, labels = generateVisualWords(sky, non_sky)

    '''find the nearest word'''
    for t in range (len(test_img)):
        num = str(t + 1).zfill(1)
        test = test_img[t]
        cv2.imshow("test_" + num, test)
        row, col, cha = test.shape
        for i in range(row):
            for j in range(col):
                minIndex = -1
```

```
            minDist = np.inf
            for w in range(len(words)):
                sub = test[i][j][:] - words[w]
                dis = np.sqrt(sub[0] ** 2 + sub[1] ** 2 + sub[2] ** 2)
                if dis < minDist:
                    minDist = dis
                    minIndex = w
            '''Generate an output image '''
            if labels[minIndex] == 1:
                test[i][j][0] = 255
                test[i][j][1] = 0
                test[i][j][2] = 0
    cv2.imshow("output_"+num, test)
cv2.waitKey()
cv2.destroyAllWindows()
```