ECE 661 Homework 7

Weichen Xu

xu1363@purdue.edu

1. Theory Questions

1.1 Short explanation on three methods for measuring image textures

1) **Gray Level Co-Occurrence Matrix (GLCM):** GLCM is a statistical method of measuring the spatial relationship of pixel. It calculates how often pairs of pixel with specific values and specific spatial relationship in an image, yielding the GLCM. Thus, multiple statistics can be derived from GLCM using properties of co-occurrence matrix, which give the textual information.

2) **Local Binary Pattern (LBP):** Since it has been shown that runs that consist of a single run of 0s and a following single run of 1s can carry most of the information among different textures. The goal of LBP is to first quantize the inter-pixel variations with binary vectors, then converting it to the rotation-invariant representation. The histogram of various LBP is used to represent the texture distribution.

3) **Gabor Filter Family:** Gabor filters can localize repetitive small patterns with different frequencies, it applies a Gaussian decay function to pixels. Essentially Gabor filter is a convolutional operator, a localized Fourier transform, which applies exponential decay factor to pixels away local pixels. So a set of Gabor filters with different frequencies and orientations can localize the image textures.

1.2

1) RGB and HSI are just linear variants of each other. **(Wrong)**

2) The color space L*a*b* is a nonlinear model of color perception. **(Right)**

3) Measuring the true color of the surface of an object is made difficult by the spectral composition of the illumination. **(Wrong)**

2. LBP Feature Extraction

Local Binary Pattern (LBP) is feature extraction method used to obtain translation and rotation-invariant features in grayscale images, the features can be then used for image classification.

2.1 Binary representation for P neighbors

We each pixel in the image, we define its neighborhood based on two parameters, R is the radius of the circle around the pixel, P is the number of points on the circle with equal distance. Here we assume $R = 1,\ P = 8$. The surrounding pixel coordinate can be expressed as follows,

$$du = Rcos\left(\frac{2\pi p}{P}\right)$$

$$dv = Rsin\left(\frac{2\pi p}{P}\right)$$

$$x_p = x + du = x + Rcos\left(\frac{2\pi p}{P}\right)$$

$$y_p = y + dv = y + Rsin\left(\frac{2\pi p}{P}\right)$$

With $p = 0,1,2\cdots, P-1$.

When $p = 0,4$, it corresponds to the bottom and top pixel of the center pixel. However, not every neighbor pixel resides exactly on the integer pixel, we need to use the bilinear interpolation to represent the pixel intensities of P neighbors.

$$I_p = (1-du)(1-dv)A + (1-du)dv \cdot B + du(1-dv)C + du \cdot dv \cdot D$$

Then for the P neighbors, we will threshold neighbors' intensities w.r.t the center pixel value, i.e. values no smaller than center pixel value will be 1, while values smaller than center pixel value will be 0. Thus, a binary vector for each pixel is generated.

2.2 Rotation-invariant pattern

Since the rotation of image will not significantly change how we percept the image, we will convert the binary pattern to a new form that is rotation-invariant. So the implementation is circularly shifting the binary vector until the minimum decimal integer representation is obtained because of the rotation-invariant feature.

2.3 Encoding the image

The rotation-invariant binary patterns will then be encoded by integer representation from 0 to $P+2$, based on their number of runs in binary patterns. If it has only one run with all 0s, it is encoded with 0. If it has only one run with all 1s, it is encoded with P. If it has two runs, the number of 1s will be used to encode. If it has more than two runs, it is encoded with $P+1$.

2.4 Histogram of encoded image

Since the image pixel is encoded with the runs of pattern, the image histogram will have $P+2$ bins, then we will normalize each bin by the total sum of pixel representations. The normalized histogram value can be treated as the feature vector of the image.

3. K Nearest Neighbors Based on Euclidean Distance

The purpose of k-nearest neighbor classifier is to classify the test image based on the result of calculating the Euclidean distance of test image's feature vector with all training images' feature vectors. Then a set of 5 training images resulting the least Euclidean distances with the test image will be given. The classification result if given by the class that appears for the most times in the set of 5 candidates.

4. Experiment Results and Observations

Image name	LBP feature vector									
beach/1.jpg	0.0368	0.0892	0.0812	0.1557	0.1496	0.0936	0.0372	0.0681	0.1690	0.1195
building/01.jpg	0.0528	0.1077	0.0435	0.1396	0.1304	0.0801	0.0357	0.1047	0.1425	0.1630
car/01.jpg	0.0346	0.0726	0.0485	0.1197	0.1236	0.0614	0.0288	0.0670	0.3497	0.0939
mountain/01.jpg	0.0567	0.0742	0.0565	0.1130	0.0734	0.0613	0.0440	0.0673	0.3272	0.1264
tree/01.jpg	0.0941	0.0978	0.0631	0.0752	0.0991	0.0854	0.0699	0.0985	0.1137	0.2031

Table 1: Sample images in all classes and Their LBP feature vectors

		Predictions				
		beach	building	car	mountain	tree
Ground Truth	beach	**2**	0	0	2	1
	building	0	**4**	0	1	0
	car	1	2	**2**	0	0
	mountain	0	0	0	**4**	0
	tree	1	0	0	0	**4**

Table2: Confusion matrix for 25 test images (Overall accuracy is 64%)

In this experiment, I choose the parameter $R = 1, P = 8$ for LBP feature extraction and choose $k = 5$ for the total number of candidates which decide the final match. We can see the **overall accuracy is $\mathbf{64}\%$**, which is much higher than the baseline accuracy 20%, it shows that at least the combination of LBP feature extraction and k-NN classifier makes a difference on the result of image classification.

5. Source Code

```
#!/usr/bin/env python
# coding: utf-8



import numpy as np
import cv2
import time
import os
from BitVector import BitVector
import statistics
from statistics import mode



def get_inter(A, B, C, D):
    # use bilinear interpretation to calculate fractional location's intensity
    dx,dy = np.sqrt(2)/2, np.sqrt(2)/2
    return (1-dx)*(1-dy)*A + dx*(1-dy)*B + (1-dx)*dy*C + dx*dy*D

def get_bitvec(patch):
    # return the 8 neighbors intensity using the 3*3 image patch
    v1 = patch[2,1]
    v2 = get_inter(patch[1,1], patch[1,2], patch[2,1], patch[2,2])
    v3 = patch[1,2]
    v4 = get_inter(patch[1,1], patch[1,2], patch[0,1], patch[0,2])
    v5 = patch[0,1]
    v6 = get_inter(patch[1,1], patch[1,0], patch[0,0], patch[0,0])
    v7 = patch[1,0]
    v8 = get_inter(patch[1,1], patch[1,0], patch[2,1], patch[2,0])

    vec = np.array([v1, v2, v3, v4, v5, v6, v7, v8])
    vec = vec >= patch[1,1] #threshold according to the center pixel value
    return vec.astype(int)

def get_pattern(vec):
    # get the encoding of local binary vector based on rotation-invariant feature
    # reference: Avinash Kak, Measuring Texture and Color in Images
    bv = BitVector.BitVector( bitlist = vec )
    intvals_for_circular_shifts = [int(bv << 1) for _ in range(len(vec))]
    minbv = BitVector.BitVector( intVal = min(intvals_for_circular_shifts), size = len(vec))
    bvruns = minbv.runs()

    if len(bvruns) > 2:
        return 9
    elif len(bvruns) == 1 and bvruns[0][0] == '1':
        return 8
    elif len(bvruns) == 1 and bvruns[0][0] == '0':
        return 0
    else:
```

```
    return len(bvruns[1])

def LBP(img):
    # Whole procedure of obtaining the P+2 bins LBP feature histogram
    R = 1
    P = 8
    hist = [0,0,0,0,0,0,0,0,0,0]
    for i in range(1, img.shape[0] - 1):
        for j in range(1, img.shape[1] - 1):
            vec = get_bitvec(img[i-1: i+2, j-1: j+2])
            idx = get_pattern(vec)
            hist[idx] += 1

    hist = np.array(hist).astype(np.float32) / np.sum(hist)
    return hist.tolist()

def train(labels):
    training_path = '/home/xu1363/Documents/ECE 661/hw7/imagesDatabaseHW7/training/'
    for label in labels:
        path = training_path + label + '/'
        hist_all = []
        for name in os.listdir(path):
            img = cv2.imread(path + name)
            img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
            hist = LBP(img)
            hist_all.append(hist)
            print(label + name + ' is finished')
        with open(training_path + label + '.txt', 'w') as filehandle:
            for i in range(len(hist_all)):
                content = ''
                hist = hist_all[i]
                for j in range(len(hist) - 1):
                    content += str(hist[j]) + ','
                content += str(hist[j+1]) + '\n'
                filehandle.write(content)

def test(labels):
    matrix_confusion = np.zeros((5,5))
    testing_path = '/home/xu1363/Documents/ECE 661/hw7/imagesDatabaseHW7/testing/'
    for i in range(len(labels)):
        test_label = labels[i]
        for num in range(5):
            name = test_label + '_' + str(num+1) + '.jpg'
            #print(name)
            img = cv2.imread(testing_path + name)
            img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
            hist_test = LBP(img)

            values = []

            # compare current image's feature vector with all training images' feature vectors
            for label in labels:
                training_path = '/home/xu1363/Documents/ECE 661/hw7/imagesDatabaseHW7/training/'
                with open(training_path + label +'.txt', 'r') as fh:
                    lines = fh.readlines()
```

```
            for k in range(len(lines)):
                line = lines[k]
                temp = line.split(',')
                a = []
                for j in range(len(temp)):
                    a.append(float(temp[j]))
                a = np.array(a)
                b = np.array(hist_test)
                value = np.linalg.norm(a-b)
                values.append(value)

        # sort from the minimum Euclidean distance
        values_sorted = sorted(values)
        val_min = values_sorted[0]
        idx = values.index(val_min)
        pred = int(idx/20)
        print(i, pred)
        print('input is ' + test_label + ', pred is ' + labels[pred])
        matrix_confusion[i, pred] += 1

    return matrix_confusion


labels = ['beach', 'building', 'car', 'mountain', 'tree']
train(labels)
matrix_confusion = test(labels)
print(matrix_confustion)
```