mean_value+sobel

February 16, 2020

1 Introduction

Here we aim to classify images into daylight landscapes, nightlight landscapes and portraits by designing features and using kNN classifier.

[16]:
```
# all imports
import cv2
import matplotlib.pyplot as plt
import numpy as np
import os
from sklearn.decomposition import KernelPCA
```

1.1 Image resizing

Since we aim to use Sobel filter to generate feature vectors, it is necessary to resize all the images to the same dimensions so that the generated feature vectors are of same length. Resizing a image may cause loss of information. To minimize this loss as well as avoid padding any image, we resize all the images to the minimum height and width.

[17]:
```
# get minimum dimensions
def get_min_dimensions():
    MIN_WIDTH = 500
    MIN_HEIGHT = 500
    for class_folder in ['portrait', 'day-landscape', 'night-landscape']:
        for file in os.listdir('../Train Images/' + class_folder):
            img = cv2.imread('../Train Images/' + class_folder + '/' + file)
            if (img.shape[0] < MIN_HEIGHT):
                MIN_HEIGHT = img.shape[0]
            if (img.shape[1] < MIN_WIDTH):
                MIN_WIDTH = img.shape[1]

    for file in os.listdir('../Test Images/'):
        img = cv2.imread('../Test Images/' + file)
        if (img.shape[0] < MIN_HEIGHT):
            MIN_HEIGHT = img.shape[0]
        if (img.shape[1] < MIN_WIDTH):
            MIN_WIDTH = img.shape[1]
```

```
    return (MIN_HEIGHT, MIN_WIDTH)
```

[18]:
```
# resize images
def resize_images(img, min_dims):
    resized_img = cv2.resize(img, min_dims, cv2.INTER_AREA)
    return resized_img
```

1.2 Average brightness

In the HSV colorspace, the 'value' of each pixel corresponds to the brightness of that pixel. Since, we know that night landscapes are inherently dark and vice versa, we can use average value of the images as a feature to distinguish between the two.

[19]:
```
# get mean value
def get_mean_value(img):
    hsv_img = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
    mean_value = np.mean(hsv_img[:,:,2])
    return hsv_img, mean_value
```

1.3 Vertical Sobel filter

We know that, in a portrait image, there would be significant number of vertical edges as compared to a landscape image. Thus, we use Sobel filter responses as a feature vector.

[20]:
```
# apply sobel filter
def apply_sobel(img):
    blur_img = cv2.GaussianBlur(img, (3, 3), 0)
    gray_img = cv2.cvtColor(blur_img, cv2.COLOR_BGR2GRAY)
    grad_y = cv2.Sobel(gray_img, cv2.CV_16S, 1, 0, ksize=3, scale=1, delta=0,␣
 ↪borderType=cv2.BORDER_DEFAULT)
    abs_gray_y = cv2.convertScaleAbs(grad_y)
    return abs_gray_y
```

[21]:
```
# image preprocessing
min_dims = get_min_dimensions()
print(min_dims)
for class_folder in ['portrait', 'day-landscape', 'night-landscape']:
    for file in os.listdir('../Train Images/' + class_folder):
#         print(file)
        img = cv2.imread('../Train Images/' + class_folder + '/' + file)
        resize_image = resize_images(img, min_dims)
        plt.imsave('./Resized Images/' + class_folder + '/' + file,␣
 ↪resize_image)

for file in os.listdir('../Test Images/'):
#     print(file)
    img = cv2.imread('../Test Images/' + file)
    resize_image = resize_images(img, min_dims)
    plt.imsave('./Resized Images/Test Images/' + file, resize_image)
```

```
(138, 182)
```

[22]:
```
# class dictionary
class_dict = {
    'portrait': 0,
    'day-landscape': 1,
    'night-landscape': 2
}

# get image features
train_feats = []
labels = []

for class_folder in ['portrait', 'day-landscape', 'night-landscape']:
    for file in os.listdir('./Resized Images/' + class_folder):
        img = cv2.imread('./Resized Images/' + class_folder + '/' + file)
        hsv_img, mean_value = get_mean_value(img)
        plt.imsave('./HSV Images/' + class_folder + '/' + file, hsv_img)
        abs_gray_y = apply_sobel(img)
        plt.imsave('./Sobel Responses/' + class_folder + '/' + file,␣
 ↪abs_gray_y, cmap='gray')
        abs_gray_y_flatten = abs_gray_y.flatten()
        abs_gray_y_flatten_list = abs_gray_y_flatten.tolist()
        abs_gray_y_flatten_list.append(mean_value)
        feat_vec = np.array(abs_gray_y_flatten_list)
        train_feats.append(feat_vec)
        labels.append(class_folder)

train_feats = np.array(train_feats, dtype=np.float32)


labels = list(map(lambda class_label : class_dict.get(class_label), labels))
labels = np.array(labels, dtype=np.float32)
print(train_feats.shape, labels.shape)
```

```
(42, 25117) (42,)
```

[23]:
```
# get test features
test_feats = []
test_file_names = []
for file in os.listdir('./Resized Images/Test Images'):
    img = cv2.imread('./Resized Images/Test Images/' + file)
    hsv_img, mean_value = get_mean_value(img)
    plt.imsave('./HSV Images/Test Images/' + file, hsv_img)
    abs_gray_y = apply_sobel(img)
    plt.imsave('./Sobel Responses/Test Images/' + file, abs_gray_y, cmap='gray')
    abs_gray_y_flatten = abs_gray_y.flatten()
```

```
    abs_gray_y_flatten_list = abs_gray_y_flatten.tolist()
    abs_gray_y_flatten_list.append(mean_value)
    feat_vec = np.array(abs_gray_y_flatten_list)
    test_feats.append(feat_vec)
    test_file_names.append(file)

test_feats = np.array(test_feats, dtype=np.float32)

print(test_feats.shape)
```

```
(12, 25117)
```

[24]:
```
# train knn classifier

knn = cv2.ml.KNearest_create()
knn.train(train_feats, cv2.ml.ROW_SAMPLE, labels)
```

[24]:
```
True
```

[25]:
```
# test kNN
ret, results, neighbours ,dist = knn.findNearest(test_feats, 3)
print( "result:  {}\n".format(results), test_file_names )
```

```
result:  [[2.]
 [2.]
 [1.]
 [1.]
 [2.]
 [2.]
 [2.]
 [1.]
 [1.]
 [1.]
 [2.]
 [2.]]
 ['test_1.jpeg', 'test_10.jpeg', 'test_11.jpeg', 'test_12.jpeg', 'test_2.jpeg',
'test_3.jpeg', 'test_4.jpeg', 'test_5.jpeg', 'test_6.jpeg', 'test_7.jpeg',
'test_8.jpeg', 'test_9.jpeg']
```

[ ]:
```

```