The Hong Kong Polytechnic University
Department of Electronic and Information Engineering

EIE4100 CVPR Assignment 2:

Feature Extraction and Image Classification

In this assignment, you will be given a number of images, which have been allocated to five different classes. You will use the histogram of oriented gradients (HoG) to classify the images, and then measure the performances using the HoG method with different distance metrics.

Software Tools: MATLAB is used throughout this exercise. You may refer to the HELP menu for the MATLAB commands used in this exercise.

Procedures:

There are different visual features that can be used to represent images, which are then used for image object detection and classification. Commonly used features include Gabor wavelets, HoG, local binary patterns (LBP), etc. In this assignment, we consider using the HoG features. A HoG feature vector is extracted from each image, which describes the distribution of shape structure in the image. In this exercise, a small database, which contains five different classes, with each class having five images, is employed. To evaluate the performances of HoG with different structures and distance metrics, five testing images for each class are used. The five classes are denoted as Classes 1, 2, 3, 4, and 5, which correspond to images of “face”, “vehicle”, “people”, “chair”, and “texture”, respectively. The training images and testing images are placed in the folder “image”, and images of Class “i” are stored in the folder named “i”. An image file named “ij_training.bmp” means that it is the jth training image of Class “i”, while “ij_test.bmp” is a testing image. Figure 1 shows some of the images used in the experiment.

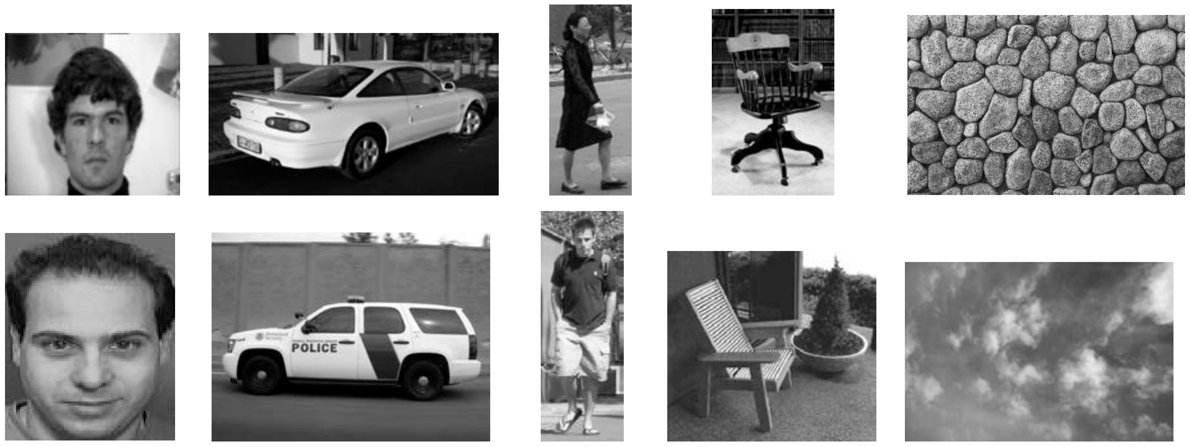
(a) *Image Processing using MATLAB*

In image processing, the first step is to read an image file, then operate on the image pixels, and show and save the processed images. Use the following MATLAB codes to read a color jpeg file. Note that you should change your current directory in MATLAB to where the images are placed, and that a color image contains three color components.

```
>> A = imread('test_color.jpg');
>> size(A)
>> A_gray = rgb2gray(A);
>> [nr nc] = size(A_gray)
>> imwrite(A_gray, 'test_gray.jpg');
>> Sx=[-1 0 1; -2 0 2; -1 0 1];      % Sobel operator
>> Sy=[-1 -2 -1; 0 0 0; 1 2 1];      % Sobel operator
>> Ax=imfilter(double(A_gray), Sx);
>> Ay=imfilter(double(A_gray), Sy);
>> figure, imshow(A), impixelinfo
>> figure, imshow(A_gray), impixelinfo
>> figure, imshow(mat2gray(Ax)), impixelinfo
>> figure, imshow(mat2gray(Ay)), impixelinfo
```

The image test_color.jpg, its gray-scale version, as well as horizontal and vertical gradients, should be displayed. Note that the three color components of A are represented as A(:, :, 1), A(:, :, 2), and A(:, :, 3), respectively.

What are the sizes of A and A_gray? What is the function of rgb2gray, imfilter, mat2gray, respectively?



“face” “vehicle” “people” “chair” “texture”

Figure 1: Images from five image classes.

(b) *Extracting the HoG feature*

The images used are gray-scale images, and in the bmp format. To read the image “11_Training.jpg” in folder “1” (i.e. Class 1), you may use the following MATLAB codes:

```
>> class_no = 1;
>> image_no = 1;
>> filepath = strcat(num2str(class_no), ‘\’, number2str(class_no), number2str(image_no),
     ‘_Training.bmp’);
>> imread(filepath)
```

Now, the gradient magnitude and orientation for each pixel are to be computed. Assume that Ix and Iy are the horizontal and vertical gradient images, respectively, and the unsigned gradient is used to represent the HoG histogram. Study the following MATLAB codes to compute the magnitudes and orientations. Note that Ix and Iy must be of the type double.

```
>> I_mag=sqrt(Ix.^2+ Iy.^2);                % I_mag: gradient magnitude
>> % Gradient orientation
>> for j=1:nr
>>   for i=1:nc
>>     if abs(Ix(j,i))<=0.0001 & abs(Iy(j,i))<=0.0001
>>       I_angle(j, i) = 0.00;
>>     else
>>       if Ix(j,i)~=0
>>         Ipr(j, i) = atan(Iy(j,i)/Ix(j,i));
>>         I_angle(j, i) = Ipr(j, i)*180/pi;
>>         if I_angle(j, i) < 0
>>           I_angle(j, i)=180+I_angle(j, i);
>>         end
>>       else
>>         Ipr(j, i) = pi/2;
>>         I_angle(j, i) = 90;
>>       end
>>     end
>>   end
>> end
```

To extract HoG features from an image, we usually divide it into a number of blocks. Assume that the blocks do not overlap with each other, and the numbers of blocks in a column and in a row are nr_b and nc_b, respectively. Study the following MATLAB codes, which read I_mag and I_angle at each pixel position to determine the orientation bin of Image_HoG to be added with the gradient magnitude.

```
>> % Use 2x2 blocks
>> nr_b = 2;
>> nc_b = 2;
>> nbin = 9;                            % number of histogram bins
>> nr_size = nr/nr_b
>> nc_size = nc/nc_b
>> Image_HoG = zeros(1, nbin*nr_b*nc_b);
>> for i=1:nr_b
>>   for j=1:nc_b
>>     I_mag_block = I_mag((i-1)*nr_size+1:i*nr_size, (j-1)*nc_size+1:j*nc_size);
>>     I_angle_block = I_angle((i-1)*nr_size+1:i*nr_size, (j-1)*nc_size+1:j*nc_size);
>> % HoG1 is a function to create the HoG histogram
>>     gh=HoG1(I_mag_block, I_angle_block, nbin);
>> % Histogram_Normalization is a function to normalize the input histogram gh
>>     ngh=Histogram_Normalization(gh);
>>     pos = (j-1)*nbin+(i-1)*nc_b*nbin+1
>>     Image_HoG(pos:pos+nbin-1) = ngh
>>   end
>> end
```

In order to execute the above MATLAB codes, you have to develop the functions HoG1( ) and Histogram_Normalization( ). Some of the MATLAB codes are given as follows:

```
function [ ghist ] = HoG1( Im, Ip, nbin )
% Compute the HoG of an image block, with unsigned gradient (i.e. 0-180)
% Im: magnitude of the image block
% Ip: orientation of the image block
% nbin: number of orientation bins
ghist = zeros(1, nbin);
[nr1 nc1] = size(Im);
% Compute the HoG
interval = ????;            % the interval for a bin, and what is “????”?
for I = 1:nr1
  for j = 1:nc1
    index = floor(Ip(i, j)/interval)+1;
    ghist(index) =????;     % what is “????”?
  end
end
end

function [ nhist ] = Histogram_Normalization( ihist )
% Normalize input histogram ihist to a unit histogram
total_sum = sum(ihist);
nhist = ????;               % what is “????”?
end
```

Using the above MATLAB codes, you can compute the HoG histogram for each of the training images. Now, you should extract the HoG features of those testing images. After that, you compare the HoG feature of a testing image to each training image based on a distance metric. The testing image will be assigned to the class of the training image with the smallest distance.

(c) *Matching of HoG histogram*

Having constructed the HoG histograms of the images, we can perform image classification, based on a distance metric or a similarity measure. In the following, we will consider the L_1-metric, L_2-metric, and Chi-square distance to measure the distance between two histograms, **I** and **Q**, which is defined as follows:

$$d_{L1}(\mathbf{I},\mathbf{Q}) = \sum_{i=1}^{n} \left| I_i - Q_i \right| , \qquad (1)$$

$$d_{L2}(\mathbf{I},\mathbf{Q}) = \sum_{i=1}^{n} \left(I_i - Q_i \right)^2 \text{ , and} \qquad (2)$$

$$\chi^2(\mathbf{I},\mathbf{Q}) = \sum_{i=1}^{n} \frac{\left(I_i - Q_i \right)^2}{I_i + Q_i} , \qquad (3)$$

where n is the number of bins, and I_i and Q_i represent the i^{th} bin in the HoG histograms **I** and **Q**, respectively. Following are the codes to compute the L_1 distance between two histograms, h1 and h2.

```
>> d = abs(h1 – h2);        % absolute differences of the corresponding bins
>> d1 = sum(d);             % sum of absolute differences
```

(d) *Image Classification and Evaluation*

The classification accuracy of using HoG will be evaluated, based on a different numbers of blocks and distance metrics.

(i) Set nr_b = 2 and nc_b = 2, evaluate the accuracy of classifying the 25 testing images using the distance metrics: L_1-metric, L_2-metric, and Chi-square distance. What is the dimension of the HoG feature vector?

(ii) Repeat (i) by setting nr_b = 3 and nc_b = 3. What is the dimension of the HoG feature vector?

Discuss your results.