**EIE4100 Assignement 2**

**Feature Extraction and Image Classification**

Tles Usserbayev, 15100295d

**Objectives:**

Extract HoG features from training images from five classes to make a model. Extract HoG features from testing images and classify them into classes according to the training results.

In this assignment there are five classes, five training for each class and five testing images for each class.

Practical task to learn important commands:

>> 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

Size of **A** = 254x384x3.

Size of **A\_gray** = 256x384.

In the above example, the functions of some commands are:

**rbg2gray** is used to convert from color into grayscale image.

c = **imfilter**(a, b) is used to apply a filter b to a matrix a and store a result in c.

**mat2gray**(a) is used to convert a matrix a into a grayscale image.

**Procedure**

To achieve the objectives of the assignment the following methodology is proposed.

1. Read the testing image.
2. Apply Sobel filters.
3. Find Magnitude.
4. Find Gradient Orientation (unsigned).
5. Find Image HoG and
6. Store HoG in Training\_HoG.
7. Repeat 1-6 for each training image in increasing order of class number and name so that the Training\_HoG contains five HoG’s of images of class 1, 2, 3, 4, 5 in a row.
8. Repeat 1-5 for testing image.
9. Calculate the difference of HoG of testing image with every training image stored in Training\_HoG and find the smallest. Use L1 metric, L2 metric and Chi-square distance to measure the difference.
10. Store the minimum difference according to each of the measures into testClasses\_by\_L1, testClass\_by\_L2, testClass\_by\_Chi respectively.
11. Repeat 8-10 for each testing image.

In the end, we have three arrays with the class numbers according to the three difference measuring techniques. Ideally, by the design of a program it should contain consecutive five ‘1’s, five ‘2’s, five ‘3’s, five ‘4’s and five ‘5’s.

To evaluate the impact of the size of blocks in HoG extraction, 2x2 and 3x3 blocks are used in this assignment. Also, the number of histogram bins is 9. The whole program code is attached in the Appendix.

**Results**

2x2 blocks

Feature vector dimension is 1x36

|  |  |  |  |
| --- | --- | --- | --- |
| **Class** | **Correctly classified by L1** | **Correctly classified by L2** | **Correctly classified by Chi-square** |
| 1, Face | 4 | 4 | 4 |
| 2, Car | 2 | 2 | 2 |
| 3, People | 4 | 3 | 4 |
| 4, Chair | 3 | 3 | 2 |
| 5, Texture | 4 | 5 | 5 |

Percentage by measure measure method

L1 = (4+2+4+3+4) / 25 = 17/25

L2 = (4+2+3+3+5) / 25 = 17/25

Chi square = (4+2+4+2+5) / 25 = 17/25

Most accurate classification by Class is Texture.

Less accurate classification by Class is Car.

3x3 blocks

Feature vector dimension is 1x81

|  |  |  |  |
| --- | --- | --- | --- |
| **Class** | **Correctly classified by L1** | **Correctly classified by L2** | **Correctly classified by Chi-square** |
| 1, Face | 5 | 5 | 5 |
| 2, Car | 2 | 2 | 2 |
| 3, People | 2 | 2 | 2 |
| 4, Chair | 3 | 2 | 2 |
| 5, Texture | 5 | 5 | 5 |

Percentage by measure measure method

L1 = (5+2+2+3+5) / 25 = 17/25

L2 = (5+2+2+2+5) / 25 = 16/25

Chi square = (5+2+2+2+5) / 25 = 16/25

Most accurate classifications by Class are Face and Texture.

Less accurate classifications by Class are Car and People.

**Conclusions**

From the results obtained from the experiment it is clear that HoG extraction is definitely a valid way of image classifications. It was decided to use unsigned orientations and 9 bins for the histogram. The only variable is the size of blocks, 2x2 and 3x3. Evaluation is based on L1 metric, L2 metric and Chi-square.

The measuring method does not affect or gives an insignificant impact on the performance of the classifier in most cases.

The dimensions of blocks, however, may change the classification results. It affects the performance for certain classes. Since the number of blocks is defined by the user it is important to use appropriate value for each class. From our experiment 3x3 suits best for face recognition. The texture can be correctly identified with both 2x2 and 3x3 blocks mainly because it has very smooth look and has uncertain edges and details. For car and people identification, it is necessary to try different sizes to find the best HoG classifier.

**Appendix**

Source Code of the main file

nclass = 5;

nimage = 5;

nr\_b = 2;

nc\_b = 3;

nbin = 9;

Sx = [-1 0 1; -2 0 1; -1 0 1];

Sy = [-1 -2 -1; 0 0 0; 1 2 1];

Training\_HoG = {};

c = 1;

minDifference = 0;

testClasses\_by\_L1 = zeros(1, 25);

testClasses\_by\_L2 = zeros(1, 25);

testClasses\_by\_Chi = zeros(1, 25);

%HoG for Training

for class\_no = 1:nclass

for image\_no = 1:nimage

filepath = strcat(num2str(class\_no), '\', num2str(class\_no), num2str(image\_no), '\_Training.bmp');

I = imread(filepath);

Ix = imfilter(double(I), Sx);

Iy = imfilter(double(I), Sy);

I\_mag = sqrt(Ix.^2 + Iy.^2);

[nr, nc] = size(I\_mag);

% Gradient orientation of training images

I\_angle = Get\_Gradient\_Orientation(Ix, Iy, nr, nc);

% number of histogram bins

Image\_HoG = Get\_HoG(I\_mag, I\_angle, nr\_b, nc\_b, nbin);

Training\_HoG{c} = Image\_HoG;

c = c + 1;

end

end

testClass = 0;

c = 1;

%HoG for Testing

for class\_no = 1:nclass

for image\_no = 1:nimage

filepath = strcat(num2str(class\_no), '\', num2str(class\_no), num2str(image\_no), '\_Test.bmp');

I = imread(filepath);

Ix = imfilter(double(I), Sx);

Iy = imfilter(double(I), Sy);

I\_mag = sqrt(Ix.^2 + Iy.^2);

[nr, nc] = size(I\_mag);

% Gradient orientation of training images

I\_angle = Get\_Gradient\_Orientation(Ix, Iy, nr, nc);

nbin = 9; % number of histogram bins

Test\_HoG = Get\_HoG(I\_mag, I\_angle, nr\_b, nc\_b, nbin);

testClass = Class\_by\_L1(Test\_HoG, Training\_HoG);

testClasses\_by\_L1(1, c) = testClass;

testClass = Class\_by\_L2(Test\_HoG, Training\_HoG);

testClasses\_by\_L2(1, c) = testClass;

testClass = Class\_by\_Chi(Test\_HoG, Training\_HoG);

testClasses\_by\_Chi(1, c) = testClass;

c = c + 1;

end

end

testClasses\_by\_L1

testClasses\_by\_L2

testClasses\_by\_Chi

Source Code of the functions

Get\_Gradient\_Orientation

function [I\_angle] = Get\_Gradient\_Orientation(Ix, Iy, nr, nc)

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

Get\_HoG

function [Image\_HoG] = Get\_HoG(I\_mag, I\_angle, nr\_b, nc\_b, nbin)

[nr, nc] = size(I\_mag);

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

HoG1

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 = 180/nbin; % the interval for a bin, and what is ì????î?

for i = 1:nr1

for j = 1:nc1

index = floor(Ip(i, j)/interval)+1;

ghist(index) = ghist(index) + Im(i, j);

end

end

end

Histogram\_Normalization

function [ nhist ] = Histogram\_Normalization( ihist )

% Normalize input histogram ihist to a unit histogram

total\_sum = sum(ihist);

nhist = ihist/total\_sum;

end

Class\_by\_L1

function [testClass] = Class\_by\_L1 (Test\_HoG, Training\_HoG)

minDifference = 0;

for c = 1:25

d = abs(Test\_HoG - Training\_HoG{c});

d1 = sum(d);

if (d1 <= minDifference) || (minDifference == 0)

minDifference = d1;

if (c > 0 && c < 6)

testClass = 1;

end

if(c > 5 && c < 11)

testClass = 2;

end

if (c > 10 && c < 16)

testClass = 3;

end

if(c > 15 && c < 21)

testClass = 4;

end

if (c > 20 && c < 26)

testClass = 5;

end

end

end

Class\_by\_L2

function [testClass] = Class\_by\_L2 (Test\_HoG, Training\_HoG)

minDifference = 0;

for c = 1:25

d = (Test\_HoG - Training\_HoG{c}).^2;

d1 = sum(d);

if (d1 <= minDifference) || (minDifference == 0)

minDifference = d1;

if (c > 0 && c < 6)

testClass = 1;

end

if(c > 5 && c < 11)

testClass = 2;

end

if (c > 10 && c < 16)

testClass = 3;

end

if(c > 15 && c < 21)

testClass = 4;

end

if (c > 20 && c < 26)

testClass = 5;

end

end

end

Class\_by\_Chi

function [testClass] = Class\_by\_Chi (Test\_HoG, Training\_HoG)

minDifference = 0;

for c = 1:25

d = (Test\_HoG - Training\_HoG{c}).^2 / (Test\_HoG + Training\_HoG{c});

d1 = sum(d);

if (d1 <= minDifference) || (minDifference == 0)

minDifference = d1;

if (c > 0 && c < 6)

testClass = 1;

end

if(c > 5 && c < 11)

testClass = 2;

end

if (c > 10 && c < 16)

testClass = 3;

end

if(c > 15 && c < 21)

testClass = 4;

end

if (c > 20 && c < 26)

testClass = 5;

end

end

end

Outputs for 2x2:

testClasses\_by\_L1 =

Columns 1 through 20

1 1 1 1 3 1 4 2 2 1 3 5 3 3 3 3 4 4 2 4

Columns 21 through 25

5 5 5 3 5

testClasses\_by\_L2 =

Columns 1 through 20

1 1 1 1 5 1 4 2 2 1 3 5 3 5 3 3 4 4 2 4

Columns 21 through 25

5 5 5 5 5

testClasses\_by\_Chi =

Columns 1 through 20

1 1 1 1 5 3 4 2 2 1 3 3 3 5 3 3 4 4 2 1

Columns 21 through 25

5 5 5 5 5

Outputs for 3x3:

testClasses\_by\_L1 =

Columns 1 through 20

1 1 1 1 1 3 4 2 2 1 3 5 5 5 3 3 4 4 2 4

Columns 21 through 25

5 5 5 5 5

testClasses\_by\_L2 =

Columns 1 through 20

1 1 1 1 1 3 4 2 2 1 3 5 5 5 3 3 4 4 2 1

Columns 21 through 25

5 5 5 5 5

testClasses\_by\_Chi =

Columns 1 through 20

1 1 1 1 1 3 4 2 2 1 3 5 5 5 3 3 4 4 2 1

Columns 21 through 25