CSE3018 : Content Based Image and Video Retrieval

# Lab - 12

Naga Harshith Bezawada
19BCE1547

**Implement a CBIR system that uses features derived from HOG Features.**

**CODE:**

**getHOGFeatures.m**

```
function [feature] = getHOGFeatures(im)

if size(im,3)==3

im=rgb2gray(im);

end

im=double(im);

rows=size(im,1);

cols=size(im,2);

Ix=im; %Basic Matrix assignment

Iy=im; %Basic Matrix assignment


% Gradients in X and Y direction. Iy is the gradient in X direction
and Iy

% is the gradient in Y direction

for i=1:rows-2
```

```
Iy(i,:)=(im(i,:)-im(i+2,:));

end

for i=1:cols-2

Ix(:,i)=(im(:,i)-im(:,i+2));

end


gauss=fspecial('gaussian',8); %% Initialized a gaussian filter with
sigma=0.5 * block width.


angle=atand(Ix./Iy); % Matrix containing the angles of each edge
gradient

angle=imadd(angle,90); %Angles in range (0,180)

magnitude=sqrt(Ix.^2 + Iy.^2);


% figure,imshow(uint8(angle));

% figure,imshow(uint8(magnitude));


% Remove redundant pixels in an image.

angle(isnan(angle))=0;

magnitude(isnan(magnitude))=0;


feature=[]; %initialized the feature vector
```

```
% Iterations for Blocks

for i = 0: rows/8 - 2

for j= 0: cols/8 -2

      %disp([i,j])


      mag_patch = magnitude(8*i+1 : 8*i+16 , 8*j+1 : 8*j+16);

      %mag_patch = imfilter(mag_patch,gauss);

      ang_patch = angle(8*i+1 : 8*i+16 , 8*j+1 : 8*j+16);


      block_feature=[];


      %Iterations for cells in a block

      for x= 0:1

            for y= 0:1

                  angleA =ang_patch(8*x+1:8*x+8, 8*y+1:8*y+8);

                  magA   =mag_patch(8*x+1:8*x+8, 8*y+1:8*y+8);

                  histr  =zeros(1,9);


                  %Iterations for pixels in one cell

                  for p=1:8
```

```
                for q=1:8
%
                    alpha= angleA(p,q);


                    % Binning Process (Bi-Linear Interpolation)
                    if alpha>10 && alpha<=30
                        histr(1)=histr(1)+
magA(p,q)*(30-alpha)/20;
                        histr(2)=histr(2)+
magA(p,q)*(alpha-10)/20;
                    elseif alpha>30 && alpha<=50
                        histr(2)=histr(2)+
magA(p,q)*(50-alpha)/20;
                        histr(3)=histr(3)+
magA(p,q)*(alpha-30)/20;
                    elseif alpha>50 && alpha<=70
                        histr(3)=histr(3)+
magA(p,q)*(70-alpha)/20;
                        histr(4)=histr(4)+
magA(p,q)*(alpha-50)/20;
                    elseif alpha>70 && alpha<=90
                        histr(4)=histr(4)+
magA(p,q)*(90-alpha)/20;
```

```
                        histr(5)=histr(5)+
magA(p,q)*(alpha-70)/20;
                    elseif alpha>90 && alpha<=110
                        histr(5)=histr(5)+
magA(p,q)*(110-alpha)/20;
                        histr(6)=histr(6)+
magA(p,q)*(alpha-90)/20;
                    elseif alpha>110 && alpha<=130
                        histr(6)=histr(6)+
magA(p,q)*(130-alpha)/20;
                        histr(7)=histr(7)+
magA(p,q)*(alpha-110)/20;
                    elseif alpha>130 && alpha<=150
                        histr(7)=histr(7)+
magA(p,q)*(150-alpha)/20;
                        histr(8)=histr(8)+
magA(p,q)*(alpha-130)/20;
                    elseif alpha>150 && alpha<=170
                        histr(8)=histr(8)+
magA(p,q)*(170-alpha)/20;
                        histr(9)=histr(9)+
magA(p,q)*(alpha-150)/20;
                    elseif alpha>=0 && alpha<=10
                        histr(1)=histr(1)+
magA(p,q)*(alpha+10)/20;
```

```
                        histr(9)=histr(9)+
magA(p,q)*(10-alpha)/20;
                    elseif alpha>170 && alpha<=180
                        histr(9)=histr(9)+
magA(p,q)*(190-alpha)/20;
                        histr(1)=histr(1)+
magA(p,q)*(alpha-170)/20;
                    end



                end
                end
                block_feature=[block_feature histr]; % Concatenation
of Four histograms to form one block feature


            end
        end
        % Normalize the values in the block using L1-Norm
        block_feature=block_feature/sqrt(norm(block_feature)^2+.01);


        feature=[feature block_feature]; %Features concatenation
end
end
```

```
feature(isnan(feature))=0; %Removing Infinitiy values


% Normalization of the feature vector using L2-Norm

feature=feature/sqrt(norm(feature)^2+.001);

for z=1:length(feature)

if feature(z)>0.2

     feature(z)=0.2;

end

end

feature=feature/sqrt(norm(feature)^2+.001);

% toc;
```

**Main.m**

```
clc

clear all

close all

tic;

% Reading in the query image and extracting it's LBP features

query_image = imread('./Faces/angry20.jpg');

query_image_features = getHOGFeatures(query_image);
```

```
% Initializing the path of the image base and getting the directory
listing

D = './Faces';

S = dir(fullfile(D, '*.jpg'));


%Column Names

CNames = {'file_name'};

for i = 1:900

CNames{end+1} = sprintf('%d',i);

end

CNames{end+1} = 'Euclidean Distance';

info_table = cell2table(cell(0, size(CNames,2)),
'VariableNames',CNames);


% Calculating the euclidean distance between every image in the image
base and the query image

for k=1:numel(S)

F = fullfile(D, S(k).name);

I = imread(F);

image_features = getHOGFeatures(I);

if size(image_features, 2) == size(query_image_features, 2)

     euclidean_distance = sqrt(sum((image_features -
query_image_features).^2));
```

```
      imageFeatures={S(k).name};
      for i=1:900
            imageFeatures{end+1}=image_features(i);
      end
      imageFeatures{end+1} = euclidean_distance;
      info_table = [info_table; imageFeatures];
end
end

% Sorting the entries of the table based on ascending order of
% euclidean_distance
info_table = sortrows(info_table, 'Euclidean Distance');
writetable(info_table, 'lab12.xlsx','Sheet',1);

% Displaying the first 4 nearest image
subplot(3, 3, 2);
imshow(query_image);
title('Query image');
file_names = info_table(:, 'file_name').file_name; % Extracting the
filenames of the images
for i = 1:6
```

```
F = fullfile(D,char(file_names(i)));

I = imread(F);

subplot(3, 3, i+3);

imshow(I);

title(char(file_names(i)));

end

toc;
```

**OUTPUT**

**Query 1**


**Precision = 5/(5+5) = 0.5**

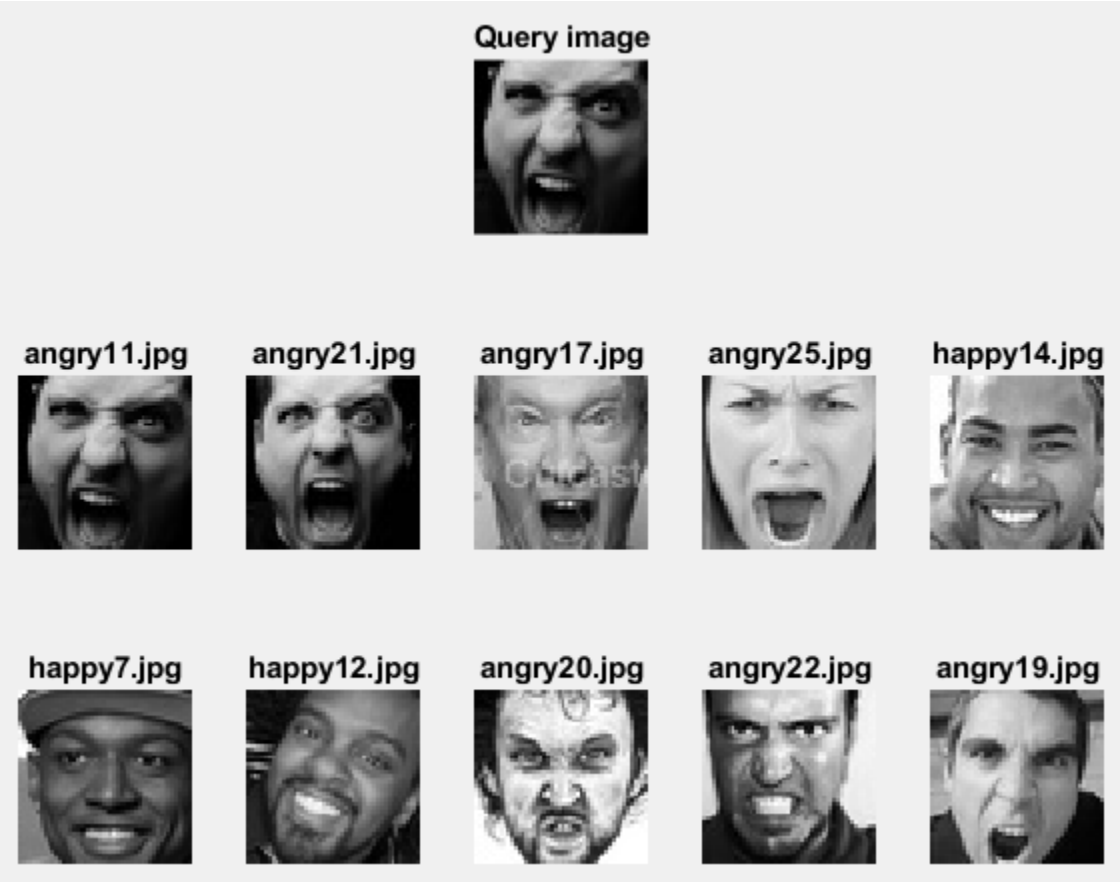
**Recall = 5/(5+(25-5)) = 5/25 = 0.2**

**Query 2**


**Precision = 7/(7+3) = 0.7**

**Recall = 7/(7+(25-7)) = 7/25 = 0.28**

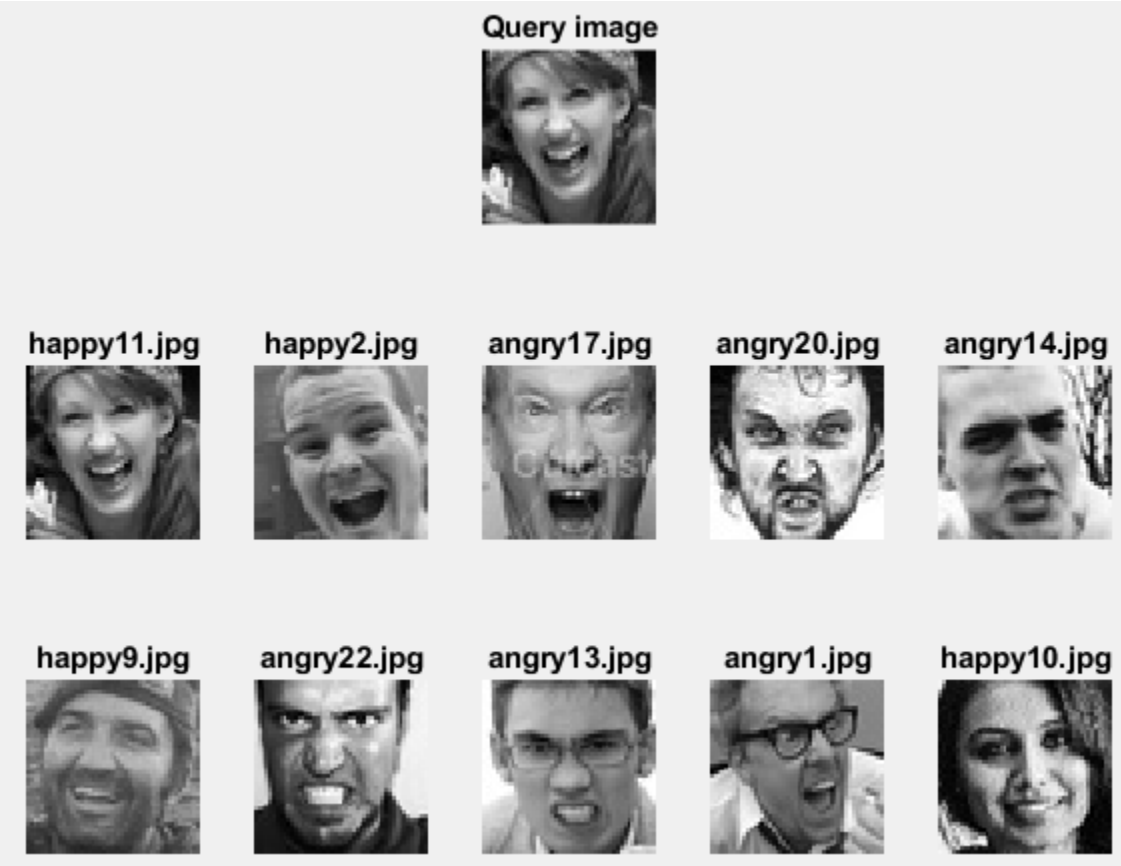
**Query 3**




**Precision = 7/(7+3) = 0.7**

**Recall = 7/(7+(25-7) = 0.28**

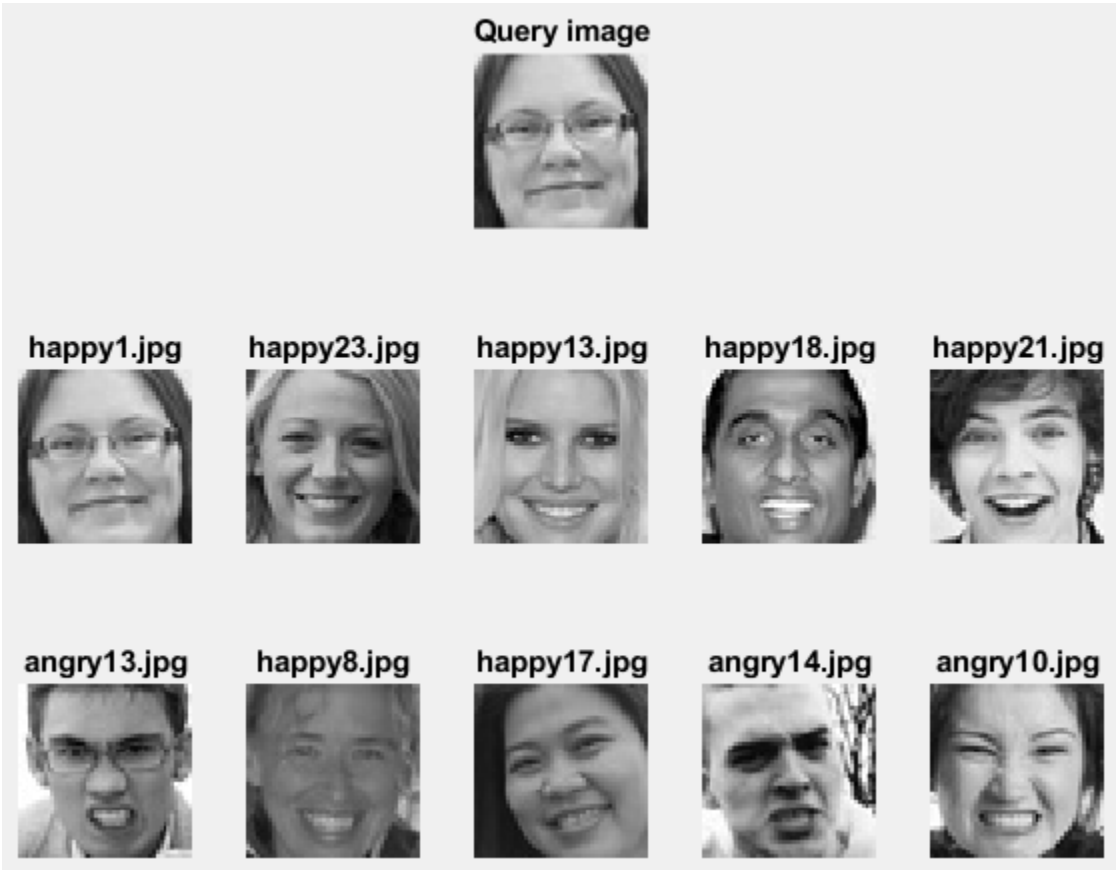
**Query 4**




**Precision = 4/(4+6) = 0.4**

**Recall = 4/25 = 0.16**

**Query 5**




**Precision = 7/(7+3) = 0.7**

**Recall = 7/25 = 0.28**

**Table**

| Query | Precision | Recall |
|---|---|---|
| angry20.jpg | 0.5 | 0.2 |
| happy20.jpg | 0.7 | 0.28 |
| angry11.jpg | 0.7 | 0.28 |
| happy4.jpg | 0.4 | 0.16 |
| happy1.jpg | 0.7 | 0.28 |

**Average Precision = 0.6**

**Average Recall = 0.24**