CSE3018 : Content Based Image and Video Retrieval

Lab - 10

Naga Harshith Bezawada
19BCE1547

CBIR Using Local Binary Patterns

METHOD 1 : Using extractLBPFeatures( )

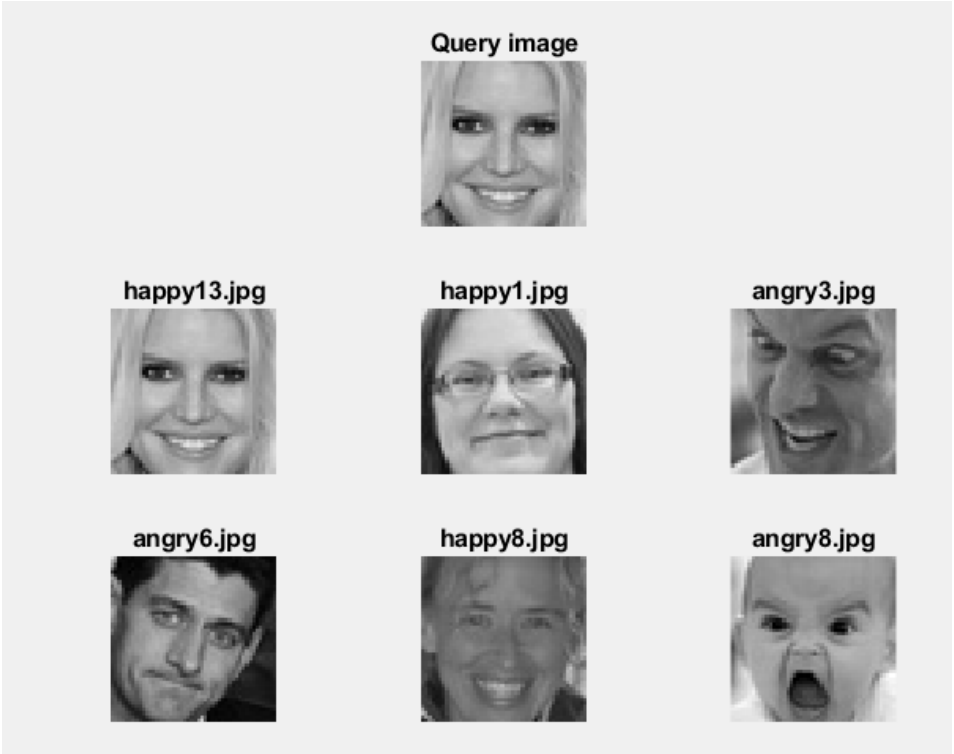
CODE :

```
clc
clear all
close all
tic;
% Reading in the query image and extracting it's LBP features
query_image = imread('./Faces/happy13.jpg');
query_image_features = extractLBPFeatures(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 = 1:59
     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 = extractLBPFeatures(I);
     euclidean_distance = sqrt(sum((image_features -
query_image_features).^2));
     imageFeatures={S(k).name};
     for i=1:59
     imageFeatures{end+1}=image_features(i);
     end
     imageFeatures{end+1} = euclidean_distance;
     info_table = [info_table; imageFeatures];
end

% Sorting the entries of the table based on ascending order of
% euclidean_distance
info_table = sortrows(info_table, 'Euclidean Distance');
writetable(info_table, 'lab10.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 : Elapsed time is 1.643879 seconds.



METHOD 2 : Using LBP( ) Function

CODE :

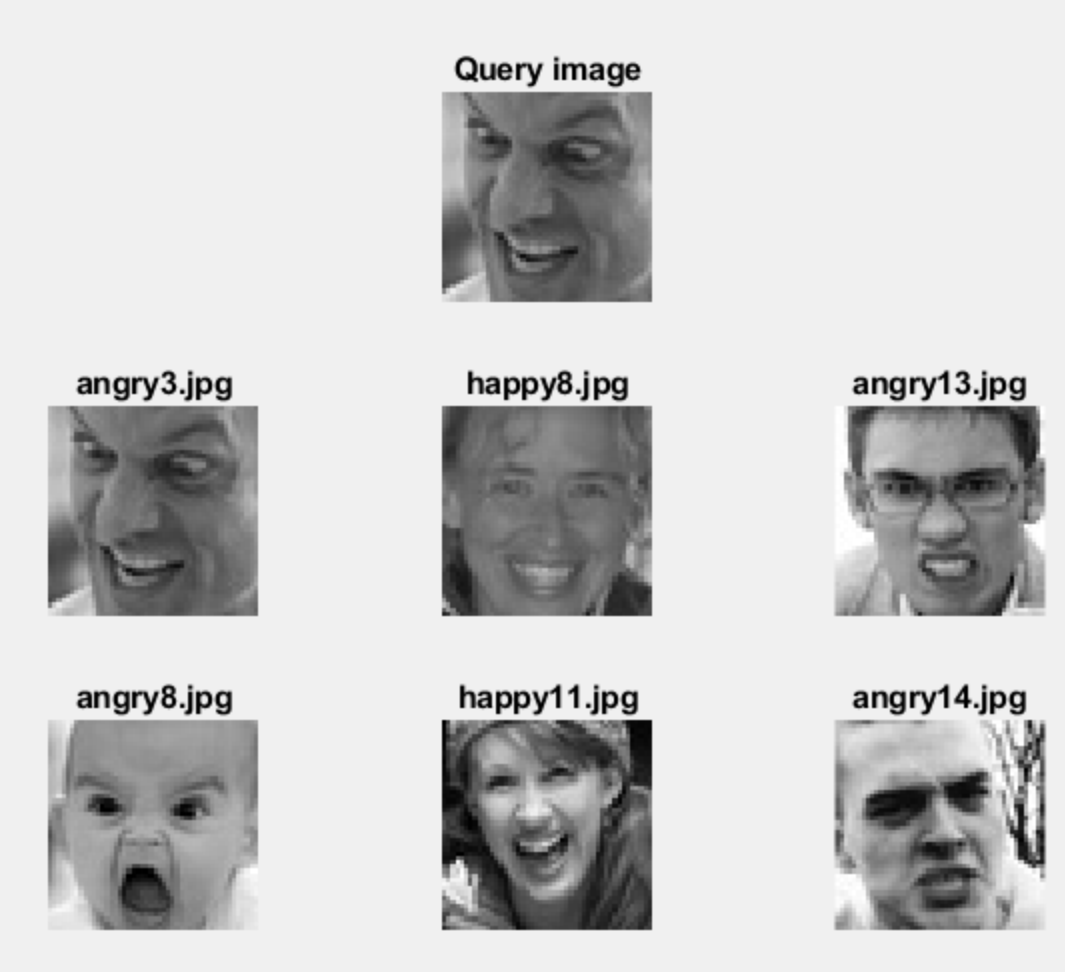
```
clc
clear all
close all
tic;
% Reading in the query image and extracting it's LBP features
query_image = imread('./Faces/happy13.jpg');
query_image_features = lbp(query_image,1);

% 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 = 0:255
     CNames{end+1} = sprintf('Bin - %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 = lbp(I,1);
     euclidean_distance = sqrt(sum((image_features -
query_image_features).^2));
     imageFeatures={S(k).name};
     for i=1:256
     imageFeatures{end+1}=image_features(i);
     end
     imageFeatures{end+1} = euclidean_distance;
     info_table = [info_table; imageFeatures];
end

% Sorting the entries of the table based on ascending order of
% euclidean_distance
info_table = sortrows(info_table, 'Euclidean Distance');
writetable(info_table, 'lab10.xlsx','Sheet',2);

% 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;
function feat = lbp(img, distance)
%   LBP Extract Local Binary Features
%   Extract Local Binary Pattern histogram features
%   Pattern for a grayscale image
%    img = im2gray(img);
     [h,w] = size(img);

     feat = zeros(1, 2^(8*distance));

     cnv_size = 2*distance + 1; % cnv_size -> convertor matrix
size
     cnv = zeros(cnv_size); % cnv -> convertor
     % Add numbers in top & bottom
     for j=1:cnv_size
     cnv(1,j) = (j) - 1; % top
     cnv(end, j) = (2*(cnv_size-1) + cnv_size - j + 1) - 1;  %
bottom
     end
     % Add numbers in left & right
```

```
     for i=2:cnv_size-1
     cnv(i, cnv_size) = (cnv_size + i - 1) - 1;  % right
     cnv(i, 1) = (8*distance - i + 2) - 1;   % left
     end
   % Raise each element to the power of 2 for binary conversion
     cnv = 2.^cnv;
     % Set all elements to zero, except the edge elements
     cnv(2:end-1, 2:end-1) = 0;
% |~~ Convertor generated! (45 min to develop this algo :|) ~~|
     for i=distance+1:h-distance
     for j=distance+1:w-distance
          % Extract the window
          window =
img(i-distance:i+distance,j-distance:j+distance);
          window(2:end-1, 2:end-1) = 0;
          % Thresholding
          tmp = zeros(cnv_size);
          tmp(window >= img(i,j)) = 1;
          % Convert to binary value using the convertor
          tmp = tmp.*cnv;
          pixel_value = sum(tmp(:));
          % Increment the bin value
          feat(1, pixel_value+1) = feat(1, pixel_value+1) + 1;
     end
     end
end
```

OUTPUT : Elapsed time is 1.899829 seconds.



Sheet 1 - Features extracted from exractLPBFeatures( )

Sheet 2 - Features extracted from lbp( ) function

Dataset - Happy and angry images from FER Dataset in Kaggle