CSE3018 : Content Based Image and Video Retrieval

Lab - 8

Naga Harshith Bezawada
19BCE1547

Q1 . Find out GLCM – Horizontal and Vertical, for each quantization level and obtain the following features :

i. Energy ii. Entropy iii. Contrast iv. Inverse Difference Moment

CODE :

```
clc
clear all
close all
query_img_path = "./images/blue1.jpg";
for i = [8,16,32,64]
query_features = getGLCMFeatures(query_img_path, i);
out = sprintf(" QUANTIZATION LEVEL - %d",i);
disp(out);
      out = sprintf(" Horizontal Energy - %f",query_features(1));
      disp(out);
      out = sprintf(" Horizontal Entropy - %f",query_features(2));
      disp(out);
      out = sprintf(" Horizontal Contrast - %f",query_features(3));
      disp(out);
      out = sprintf(" Horizontal Inverse Difference Moment -
%f",query_features(4));
      disp(out);
```

```
     out = sprintf(" Vertical Energy - %f",query_features(5));
     disp(out);
     out = sprintf(" Vertical Entropy - %f",query_features(6));
     disp(out);
     out = sprintf(" Vertical Contrast - %f",query_features(7));
     disp(out);
     out = sprintf(" Vertical Inverse Difference Moment -
%f\n",query_features(8));
     disp(out);
end
function features = getGLCMFeatures(imgFilePath, levels)
if ~exist('levels', 'var')
levels = 32;
end
directions = 3;

img = imread(imgFilePath);
img = im2gray(img);
[h, w] = size(img);

% Initialize the GLCM
GLCM = zeros(levels, levels, directions);

% Quantize the image
partitions = 256/levels;
img = imquantize(img, partitions:partitions:256);

for i=1:h
```

```
for j=1:w
      % Horizontal
      if j ~= w
            GLCM(img(i,j),img(i,j+1),1) = GLCM(img(i,j),img(i,j+1),1)
+ 1;
      end
      % Vertical
      if i ~= h
            GLCM(img(i,j),img(i+1,j),2) = GLCM(img(i,j),img(i+1,j),2)
+ 1;
      end
      % Leading Diagonal
      if i ~= h && j ~=w
            GLCM(img(i,j),img(i+1,j+1),3) =
GLCM(img(i,j),img(i+1,j+1),3) + 1;
      end
end
end


features = zeros(4,3);


% Calculate features for resp. directions
for d=1:directions
GLCMDR = GLCM(:,:,d); % GLCM Direction Resp.
% Normalize
GLCMDR = GLCM(:,:,d)./sum(sum(GLCMDR));


% Calculate energy
tmp = GLCMDR.^2;
```

```
features(1,d) = sum(tmp(:));


% Calculate entropy
tmp = GLCMDR.*log(GLCMDR);
tmp(isnan(tmp)) = 0; % To avoid calc errors
features(2,d) = -1 * sum(tmp(:));


% Calculate contrast & IDM
for i=1:levels
     for j=1:levels
          % Contrast
          features(3,d) = features(3,d) + ((i-j)^2*GLCMDR(i,j));
          % Inverse Difference Moment
          features(4,d) = features(4,d) + (GLCMDR(i,j)/(1+(i-j)^2));
     end
end
end
features = features(:);
end
```

OUTPUT FROM COMMAND WINDOW : Input Image - blue1.jpg

```
QUANTIZATION LEVEL - 8
 Horizontal Energy - 1.000000
 Horizontal Entropy - -0.000000
 Horizontal Contrast - 0.000000
 Horizontal Inverse Difference Moment - 1.000000
 Vertical Energy - 1.000000
 Vertical Entropy - -0.000000
```

Vertical Contrast - 0.000000

Vertical Inverse Difference Moment - 1.000000

QUANTIZATION LEVEL - 16

Horizontal Energy - 1.000000

Horizontal Entropy - -0.000000

Horizontal Contrast - 0.000000

Horizontal Inverse Difference Moment - 1.000000

Vertical Energy - 1.000000

Vertical Entropy - -0.000000

Vertical Contrast - 0.000000

Vertical Inverse Difference Moment - 1.000000

QUANTIZATION LEVEL - 32

Horizontal Energy - 0.496592

Horizontal Entropy - 0.730633

Horizontal Contrast - 0.006837

Horizontal Inverse Difference Moment - 0.996582

Vertical Energy - 0.497069

Vertical Entropy - 0.727948

Vertical Contrast - 0.006372

Vertical Inverse Difference Moment - 0.996814

QUANTIZATION LEVEL - 64

Horizontal Energy - 0.387485

Horizontal Entropy - 1.134275

Horizontal Contrast - 0.012918

```
Horizontal Inverse Difference Moment - 0.993541
Vertical Energy - 0.387551
Vertical Entropy - 1.129367
Vertical Contrast - 0.011131
Vertical Inverse Difference Moment - 0.994434
```

Elapsed time is 0.167332 seconds.

OUTPUT FROM COMMAND WINDOW : Input Image - yellow10.jpg

```
QUANTIZATION LEVEL - 8
Horizontal Energy - 0.936583
Horizontal Entropy - 0.194195
Horizontal Contrast - 0.030673
Horizontal Inverse Difference Moment - 0.985021
Vertical Energy - 0.956108
Vertical Entropy - 0.143116
Vertical Contrast - 0.009454
Vertical Inverse Difference Moment - 0.995273

QUANTIZATION LEVEL - 16
Horizontal Energy - 0.297917
Horizontal Entropy - 1.442338
Horizontal Contrast - 0.394203
Horizontal Inverse Difference Moment - 0.817473
Vertical Energy - 0.439292
Vertical Entropy - 1.076402
Vertical Contrast - 0.099952
Vertical Inverse Difference Moment - 0.950549
```

```
QUANTIZATION LEVEL - 32
Horizontal Energy - 0.169237
Horizontal Entropy - 2.235947
Horizontal Contrast - 0.944726
Horizontal Inverse Difference Moment - 0.720818
Vertical Energy - 0.291128
Vertical Entropy - 1.675129
Vertical Contrast - 0.181255
Vertical Inverse Difference Moment - 0.919655

QUANTIZATION LEVEL - 64
Horizontal Energy - 0.052900
Horizontal Entropy - 3.479888
Horizontal Contrast - 3.375216
Horizontal Inverse Difference Moment - 0.553950
Vertical Energy - 0.125255
Vertical Entropy - 2.671182
Vertical Contrast - 0.528043
Vertical Inverse Difference Moment - 0.851577
```

Elapsed time is 0.180377 seconds.

Q2 a. Implement a CBIR for Texture Images using these GLCM Features, for an optimal quantization level (32).

Converting Image to GreyScale So we have 12 features for each image.

CODE

```
clc

clear all

close all

tic;

D = './images';

S = dir(fullfile(D,'*.jpg')); % pattern to match filenames.


query_image_path = 'images/blue10.jpg';

queryImgFeatures = getGLCMFeatures(query_image_path,32);

names = ['file_name',"H_Energy", "H_Entropy",
"H_Contrast","H_InverseDifferenceMoment","V_Energy", "V_Entropy",
"V_Contrast","V_InverseDifferenceMoment","LD_Energy", "LD_Entropy",
"LD_Contrast","LD_InverseDifferenceMoment","Euclidean Distance"];

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


for k=1:numel(S)

image_path = sprintf('images/%s', S(k).name);

image_GLCM_feature = getGLCMFeatures(image_path,32);

image_feature = [];

image_path = S(k).name;

image_feature{end+1} = image_path;

% Calculating the Euclidean distance between the image and the query

euclidean_distance = 0;

     for i = 1:12

     euclidean_distance = euclidean_distance +
(image_GLCM_feature(i)^2 - queryImgFeatures(i)^2);

     image_feature{end+1} = image_GLCM_feature(i);
```

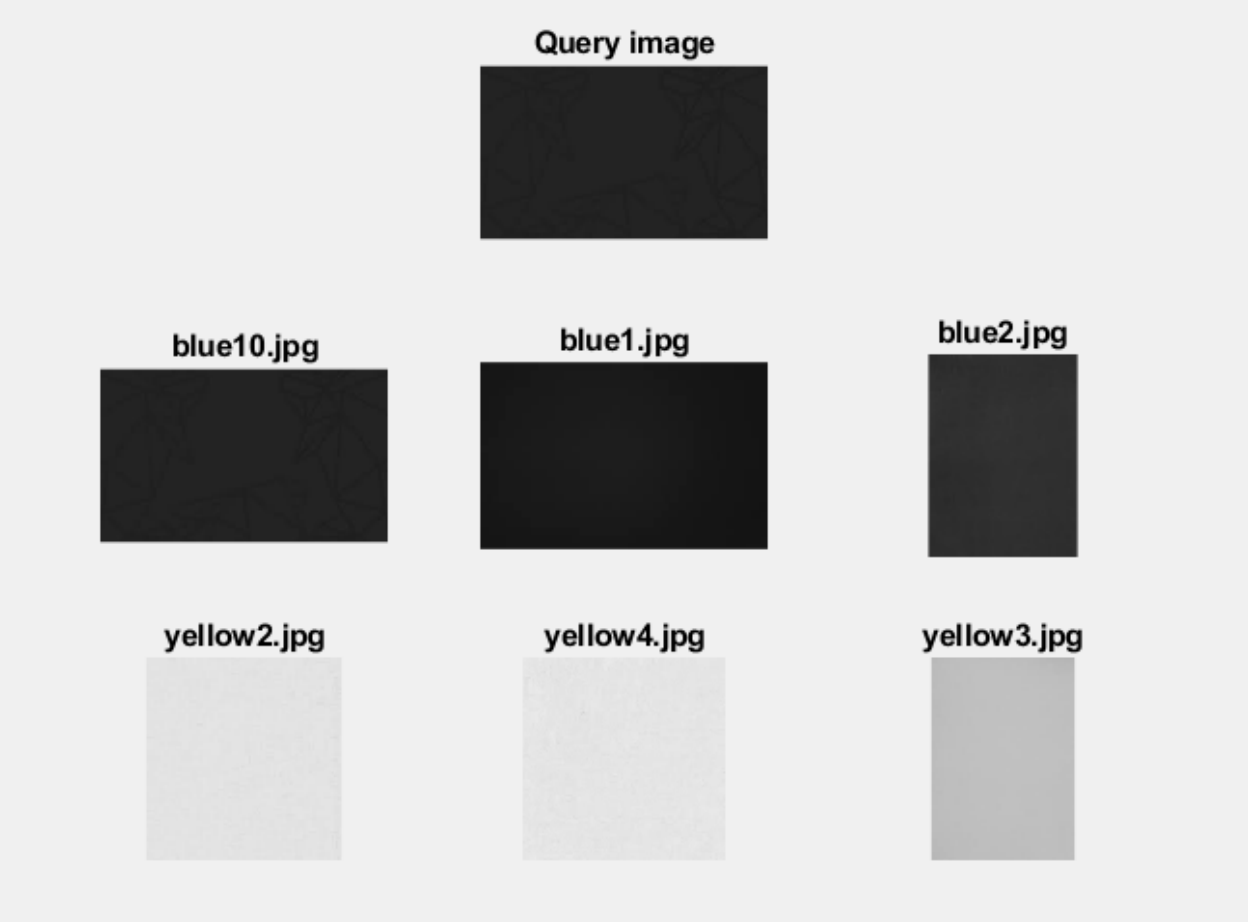
```
      end
      euclidean_distance = sqrt(euclidean_distance);
image_feature{end+1} = euclidean_distance;
% Appending the result in the table
   info_table = [info_table; image_feature];
end
info_table = sortrows(info_table, 'Euclidean Distance');
writetable(info_table, 'lab8.xls','Sheet',1);

% Displaying the first 6 nearest image
subplot(3, 3, 2);
query_image = imread(query_image_path);
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(im2gray(I));
title(char(file_names(i)));
end
toc;

function features = getGLCMFeatures(imgFilePath, levels)
if ~exist('levels', 'var')
levels = 32;
```

```
end
directions = 3;

img = imread(imgFilePath);
img = im2gray(img);
[h, w] = size(img);

% Initialize the GLCM
GLCM = zeros(levels, levels, directions);

% Quantize the image
partitions = 256/levels;
img = imquantize(img, partitions:partitions:256);

for i=1:h
for j=1:w
      % Horizontal
      if j ~= w
            GLCM(img(i,j),img(i,j+1),1) = GLCM(img(i,j),img(i,j+1),1)
+ 1;
      end
      % Vertical
      if i ~= h
            GLCM(img(i,j),img(i+1,j),2) = GLCM(img(i,j),img(i+1,j),2)
+ 1;
      end
      % Leading Diagonal
      if i ~= h && j ~=w
```

```
            GLCM(img(i,j),img(i+1,j+1),3) =
GLCM(img(i,j),img(i+1,j+1),3) + 1;
      end
end
end

features = zeros(4,3);

% Calculate features for resp. directions
for d=1:directions
GLCMDR = GLCM(:,:,d); % GLCM Direction Resp.
% Normalize
GLCMDR = GLCM(:,:,d)./sum(sum(GLCMDR));

% Calculate energy
tmp = GLCMDR.^2;
features(1,d) = sum(tmp(:));

% Calculate entropy
tmp = GLCMDR.*log(GLCMDR);
tmp(isnan(tmp)) = 0; % To avoid calc errors
features(2,d) = -1 * sum(tmp(:));

% Calculate contrast & IDM
for i=1:levels
      for j=1:levels
            % Contrast
            features(3,d) = features(3,d) + ((i-j)^2*GLCMDR(i,j));
```

```
            % Inverse Difference Moment
            features(4,d) = features(4,d) + (GLCMDR(i,j)/(1+(i-j)^2));
      end
end
end

features = features(:);
end
```

OUTPUT : Input image - blue10.jpg : Data in Sheet 1 of lab8.xls



Elapsed time is 2.050695 seconds.

Q2 b. Implement a CBIR for Texture Images using these GLCM Features, for an optimal quantization level (32).

Splitting Image to Color Planes So we have 3*12, 36 features for each image.

CODE

```
clc
clear all
close all
tic;
D = './images';
S = dir(fullfile(D,'*.jpg')); % pattern to match filenames.

query_image = imread('images/blue10.jpg');

%Color Plane Slicing
red = query_image(:,:,1);
green = query_image(:,:,2);
blue = query_image(:,:,3);
%Extracting GLCM Features from each Plane
r=[];
g=[];
b=[];
r = getGLCMFeatures(red,32);
g = getGLCMFeatures(green,32);
b = getGLCMFeatures(blue,32);
```

```
queryImgFeatures=[];

for i = 1:12

queryImgFeatures{end+1} = r(i);

end

for i = 1:12

queryImgFeatures{end+1} = g(i);

end

for i = 1:12

queryImgFeatures{end+1} = b(i);

end


names = ['file_name',"R_H_Energy", "R_H_Entropy",
"R_H_Contrast","R_H_InverseDifferenceMoment","R_V_Energy",
"R_V_Entropy",
"R_V_Contrast","R_V_InverseDifferenceMoment","R_LD_Energy",
"R_LD_Entropy",
"R_LD_Contrast","R_LD_InverseDifferenceMoment","G_H_Energy","G_H_Entr
opy", "G_H_Contrast","G_H_InverseDifferenceMoment","G_V_Energy",
"G_V_Entropy",
"G_V_Contrast","G_V_InverseDifferenceMoment","G_LD_Energy",
"G_LD_Entropy",
"G_LD_Contrast","G_LD_InverseDifferenceMoment","B_H_Energy","B_H_Entr
opy", "B_H_Contrast","B_H_InverseDifferenceMoment","B_V_Energy",
"B_V_Entropy",
"B_V_Contrast","B_V_InverseDifferenceMoment","B_LD_Energy",
"B_LD_Entropy",
"B_LD_Contrast","B_LD_InverseDifferenceMoment","Euclidean Distance"];


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


for k=1:numel(S)

image_path = sprintf('images/%s', S(k).name);

img = imread(image_path);
```

```
%Color Plane Slicing
red = img(:,:,1);
green = img(:,:,2);
blue = img(:,:,3);

%Extracting GLCM Features from each Plane
r = getGLCMFeatures(red,32);
g = getGLCMFeatures(green,32);
b = getGLCMFeatures(blue,32);

image_GLCM_feature=[];
image_feature = [];

for i = 1:12
     image_GLCM_feature{end+1} = r(i);
end
for i = 1:12
     image_GLCM_feature{end+1} = g(i);
end
for i = 1:12
     image_GLCM_feature{end+1} = b(i);
end

image_path = S(k).name;
image_feature{end+1} = image_path;
% Calculating the Euclidean distance between the image and the query
euclidean_distance = 0;
```

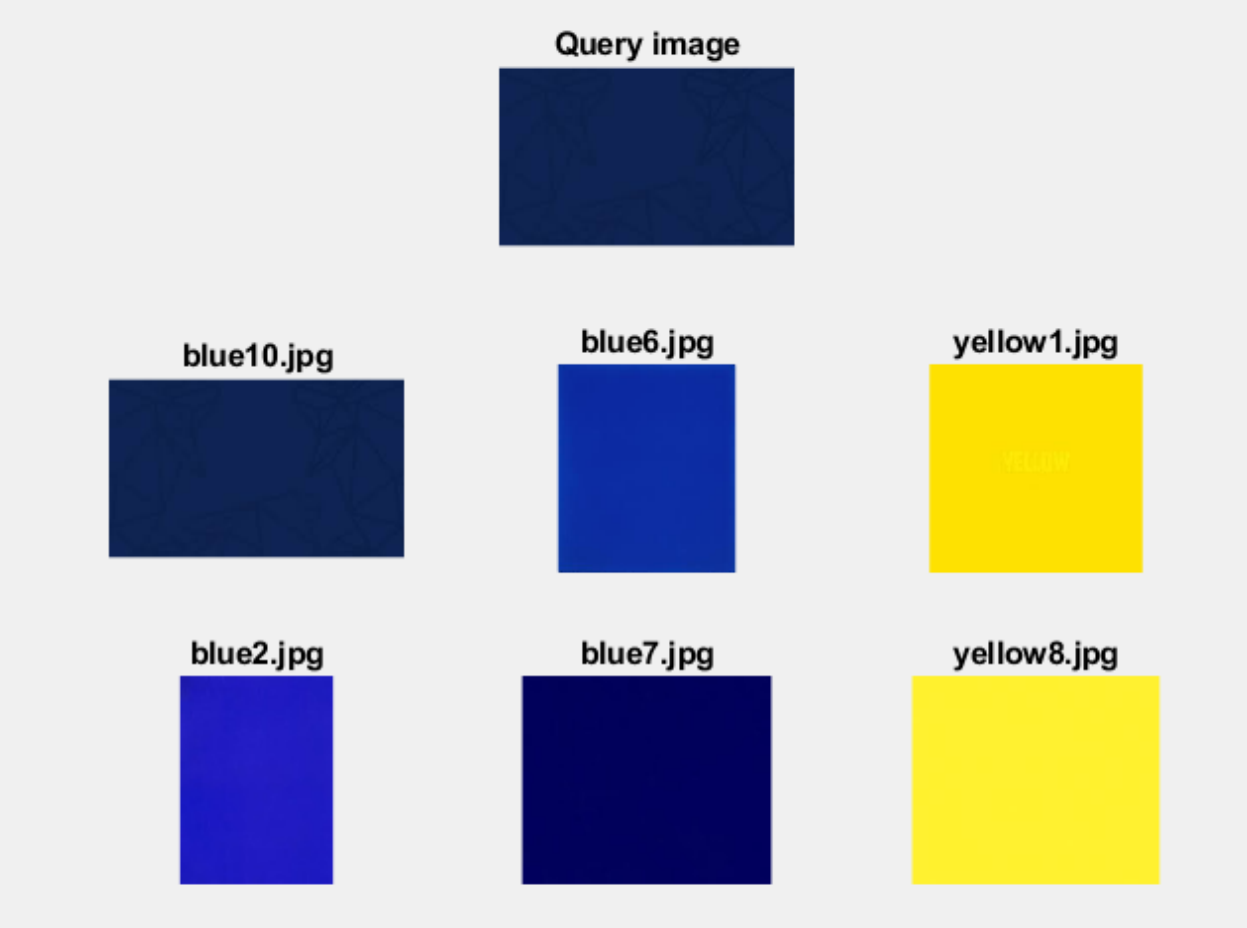
```
      for i = 1:36
      x = image_GLCM_feature{i};
      y = queryImgFeatures{i};
      euclidean_distance = euclidean_distance + (x*x - y*y);
      image_feature{end+1} = image_GLCM_feature(i);
      end
      euclidean_distance = sqrt(euclidean_distance);
image_feature{end+1} = euclidean_distance;

% Appending the result in the table
info_table = [info_table; image_feature];
end
info_table = sortrows(info_table, 'Euclidean Distance');
writetable(info_table, 'lab8.xls','Sheet',2);

% Displaying the first 6 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 features = getGLCMFeatures(imgFile, levels)
if ~exist('levels', 'var')
levels = 32;
end
directions = 3;

img = imgFile;

[h, w] = size(img);

% Initialize the GLCM
GLCM = zeros(levels, levels, directions);

% Quantize the image
partitions = 256/levels;
img = imquantize(img, partitions:partitions:256);

for i=1:h
for j=1:w
      % Horizontal
      if j ~= w
            GLCM(img(i,j),img(i,j+1),1) = GLCM(img(i,j),img(i,j+1),1)
+ 1;
      end
      % Vertical
      if i ~= h
```

```
            GLCM(img(i,j),img(i+1,j),2) = GLCM(img(i,j),img(i+1,j),2)
+ 1;

      end

      % Leading Diagonal

      if i ~= h && j ~=w

            GLCM(img(i,j),img(i+1,j+1),3) =
GLCM(img(i,j),img(i+1,j+1),3) + 1;

      end

end

end


features = zeros(4,3);


% Calculate features for resp. directions

for d=1:directions

GLCMDR = GLCM(:,:,d); % GLCM Direction Resp.

% Normalize

GLCMDR = GLCM(:,:,d)./sum(sum(GLCMDR));


% Calculate energy

tmp = GLCMDR.^2;

features(1,d) = sum(tmp(:));


% Calculate entropy

tmp = GLCMDR.*log(GLCMDR);

tmp(isnan(tmp)) = 0; % To avoid calc errors

features(2,d) = -1 * sum(tmp(:));
```

```
% Calculate contrast & IDM
for i=1:levels
      for j=1:levels
            % Contrast
            features(3,d) = features(3,d) + ((i-j)^2*GLCMDR(i,j));
            % Inverse Difference Moment
            features(4,d) = features(4,d) + (GLCMDR(i,j)/(1+(i-j)^2));
      end
end
end

features = features(:);


end
```

OUTPUT : Input image - blue10.jpg : Data in Sheet 2 of lab8.xls



Elapsed time is 2.362187 seconds.