CSE3018 : Content Based Image and Video Retrieval

Lab - 7

Naga Harshith Bezawada
19BCE1547

Q1. Implement a CBIR for Texture Images using these Tamura Features.

CODE :

```
clc;
clear all;
close all;
query_image_path = '1.jpg';
queryImgFeatures =
getTamuraFeatures(getImgFilePath(query_image_path));
names = ['file_name',"Contrast", "Directionality",
"Coarseness","Linelikeness", "Regularity", "Roughness","City
Block Distance"];
info_table = cell2table(cell(0, size(names,2)), 'VariableNames',
names);
% Reading the images of textures from the image base
D = './img2';
S = dir(fullfile(D,'*.jpg')); % pattern to match filenames.
% Looping through all the images in the directory
for k=1:numel(S)
     image_path = S(k).name;
```

```
     image_tamura_feature =
getTamuraFeatures(getImgFilePath(image_path));
     image_feature = [];
     image_feature{end+1} = image_path;
     % Calculating the City Block distance between the image and
the query
     city_block_distance = 0;
     for i = 1:6
          city_block_distance = city_block_distance +
abs(image_tamura_feature(i) - queryImgFeatures(i));
          image_feature{end+1} = image_tamura_feature(i);
     end
     image_feature{end+1} = city_block_distance;
     % Appending the result in the table
   info_table = [info_table; image_feature];
end
info_table = sortrows(info_table, 'City Block Distance');
writetable(info_table, 'lab7.xls');

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

% |#| Function to retrieve file path
function filePath = getImgFilePath(imgName)
     imgSetPath = "./img2/";
     filePath = sprintf('%s%s', imgSetPath, imgName);
end
```

getTamuraFeatures.m

```
function feat = getTamuraFeatures(imgFilePath)
%GETTAMURAFEATURES Extracts the Tamura features of the given image
%   Returns Tamura Features of the given image as a 1x6 matrix in
order of
%
Contrast,Directionality,Coarseness,Linelikeness,Regularity,Roughness
%   References:
%
https://github.com/MarshalLeeeeee/Tamura-In-Python/tree/master/refere
nced-matlab-code
```

```
feat = double(zeros(1,6));

img = imread(imgFilePath);

img = im2gray(img);


DLMI = double(img(:)); % Double Linear Matrix Image


% Contrast

alpha = 0.25;

feat(1, 1) = var(DLMI)/(kurtosis(DLMI)^alpha);

% Directionality

[feat(1, 2), sita] = directionality(img);

% Coarseness

feat(1, 3) = coarseness(img, 5);

% Linelikeness

feat(1, 4) = linelikeness(img, sita, 4);

% Regularity

feat(1, 5) = regularity(img, 64);

% Roughness

feat(1, 6) = feat(1, 1) + feat(1, 3);


end
```

coarseness.m

```
function Fcrs = coarseness( graypic,kmax )

[h,w]=size(graypic);

A=zeros(h,w,2^kmax);

for i=2^(kmax-1)+1:h-2^(kmax-1)

for j=2^(kmax-1)+1:w-2^(kmax-1)

      for k=1:kmax


A(i,j,k)=mean2(graypic(i-2^(k-1):i+2^(k-1)-1,j-2^(k-1):j+2^(k-1)-1));

      end

end

end

for i=1+2^(kmax-1):h-2^(kmax-1)

for j=1+2^(kmax-1):w-2^(kmax-1)

      for k=1:kmax

            Eh(i,j,k)=abs(A(i+2^(k-1),j,k)-A(i-2^(k-1),j));

            Ev(i,j,k)=abs(A(i,j+2^(k-1),k)-A(i,j-2^(k-1)));

      end

end

end

for i=2^(kmax-1)+1:h-2^(kmax-1)
```

```
for j=2^(kmax-1)+1:w-2^(kmax-1)
      [maxEh,p]=max(Eh(i,j,:));
      [maxEv,q]=max(Ev(i,j,:));
      if maxEh>maxEv
            maxkk=p;
      else
            maxkk=q;
      end
      Sbest(i,j)=2^maxkk;
end
end
Fcrs=mean2(Sbest);
end
```

directionality.m

```
function [Fdir,sita]=directionality(graypic)
[h w]=size(graypic);
GradientH=[-1 0 1;-1 0 1;-1 0 1];
GradientV=[ 1 1 1;0 0 0;-1 -1 -1];
MHconv=conv2(graypic,GradientH);
MH=MHconv(3:h,3:w);
MVconv=conv2(graypic,GradientV);
```

```
MV=MVconv(3:h,3:w);

MG=(abs(MH)+abs(MV))./2;

validH=h-2;

validW=w-2;

for i=1:validH

for j=1:validW

      sita(i,j)=atan(MV(i,j)/MH(i,j))+(pi/2);

end

end

n=16;

t=12;

Nsita=zeros(1,n);

for i=1:validH

for j=1:validW

      for k=1:n

            if sita(i,j)>=(2*(k-1)*pi/2/n) &&
sita(i,j)<((2*(k-1)+1)*pi/2/n) && MG(i,j)>=t

                  Nsita(k)=Nsita(k)+1;

            end

      end

end
```

```
end

for k=1:n

HD(k)=Nsita(k)/sum(Nsita(:));

end

[maxvalue,FIp]=max(HD);

Fdir=0;

for k=1:n

Fdir=Fdir+(k-FIp)^2*HD(k);

end

end
```

contrast.m

```
function Fcon=contrast(graypic)

graypic=double(graypic);

x=graypic(:);

M4=mean((x-mean(x)).^4);

delta2=var(x,1);

alfa4=M4/(delta2^2);

delta=std(x,1);

Fcon=delta/(alfa4^(1/4));

end
```

likeness.m

```
function Flin=linelikeness(graypic,sita,d)

n=16;

[h,w]=size(graypic);

PDd1=zeros(n,n);

PDd2=zeros(n,n);

PDd3=zeros(n,n);

PDd4=zeros(n,n);

PDd5=zeros(n,n);

PDd6=zeros(n,n);

PDd7=zeros(n,n);

PDd8=zeros(n,n);

for i=d+1:h-d-2

for j=d+1:w-d-2

      for m1=1:n

            for m2=1:n

                  if (sita(i,j)>=(2*(m1-1)*pi/2/n) &&
sita(i,j)<((2*(m1-1)+1)*pi/2/n)) && (sita(i+d,j)>=(2*(m2-1)*pi/2/n)
&& sita(i+d,j)<((2*(m2-1)+1)*pi/2/n))

                  PDd1(m1,m2)=PDd1(m1,m2)+1;

                  end
```

```
                if (sita(i,j)>=(2*(m1-1)*pi/2/n) &&
sita(i,j)<((2*(m1-1)+1)*pi/2/n)) && (sita(i-d,j)>=(2*(m2-1)*pi/2/n)
&& sita(i-d,j)<((2*(m2-1)+1)*pi/2/n))

                PDd2(m1,m2)=PDd2(m1,m2)+1;

                end

                if (sita(i,j)>=(2*(m1-1)*pi/2/n) &&
sita(i,j)<((2*(m1-1)+1)*pi/2/n)) && (sita(i,j+d)>=(2*(m2-1)*pi/2/n)
&& sita(i,j+d)<((2*(m2-1)+1)*pi/2/n))

                PDd3(m1,m2)=PDd3(m1,m2)+1;

                end

                if (sita(i,j)>=(2*(m1-1)*pi/2/n) &&
sita(i,j)<((2*(m1-1)+1)*pi/2/n)) && (sita(i,j-d)>=(2*(m2-1)*pi/2/n)
&& sita(i,j-d)<((2*(m2-1)+1)*pi/2/n))

                PDd4(m1,m2)=PDd4(m1,m2)+1;

                end

                if (sita(i,j)>=(2*(m1-1)*pi/2/n) &&
sita(i,j)<((2*(m1-1)+1)*pi/2/n)) && (sita(i+d,j+d)>=(2*(m2-1)*pi/2/n)
&& sita(i+d,j+d)<((2*(m2-1)+1)*pi/2/n))

                PDd5(m1,m2)=PDd5(m1,m2)+1;

                end

                %右上方向

                if (sita(i,j)>=(2*(m1-1)*pi/2/n) &&
sita(i,j)<((2*(m1-1)+1)*pi/2/n)) && (sita(i-d,j+d)>=(2*(m2-1)*pi/2/n)
&& sita(i-d,j+d)<((2*(m2-1)+1)*pi/2/n))

                PDd6(m1,m2)=PDd6(m1,m2)+1;
```

```
                end
                %左下方向
                if (sita(i,j)>=(2*(m1-1)*pi/2/n) &&
sita(i,j)<((2*(m1-1)+1)*pi/2/n)) && (sita(i+d,j-d)>=(2*(m2-1)*pi/2/n)
&& sita(i+d,j-d)<((2*(m2-1)+1)*pi/2/n))
                PDd7(m1,m2)=PDd7(m1,m2)+1;
                end
                %左上方向
                if (sita(i,j)>=(2*(m1-1)*pi/2/n) &&
sita(i,j)<((2*(m1-1)+1)*pi/2/n)) && (sita(i-d,j-d)>=(2*(m2-1)*pi/2/n)
&& sita(i-d,j-d)<((2*(m2-1)+1)*pi/2/n))
                PDd8(m1,m2)=PDd8(m1,m2)+1;
                end
            end
      end
end
end
f=zeros(1,8);
g=zeros(1,8);
for i=1:n
for j=1:n
      f(1)=f(1)+PDd1(i,j)*cos((i-j)*2*pi/n);
      g(1)=g(1)+PDd1(i,j);
```

```
      f(2)=f(2)+PDd2(i,j)*cos((i-j)*2*pi/n);

      g(2)=g(2)+PDd2(i,j);

      f(3)=f(3)+PDd3(i,j)*cos((i-j)*2*pi/n);

      g(3)=g(3)+PDd3(i,j);

      f(4)=f(4)+PDd4(i,j)*cos((i-j)*2*pi/n);

      g(4)=g(4)+PDd4(i,j);

      f(5)=f(5)+PDd5(i,j)*cos((i-j)*2*pi/n);

      g(5)=g(5)+PDd5(i,j);

      f(6)=f(6)+PDd6(i,j)*cos((i-j)*2*pi/n);

      g(6)=g(6)+PDd6(i,j);

      f(7)=f(7)+PDd7(i,j)*cos((i-j)*2*pi/n);

      g(7)=g(7)+PDd7(i,j);

      f(8)=f(8)+PDd8(i,j)*cos((i-j)*2*pi/n);

      g(8)=g(4)+PDd8(i,j);

end

end

tempM=f./g;

Flin=max(tempM);

end
```

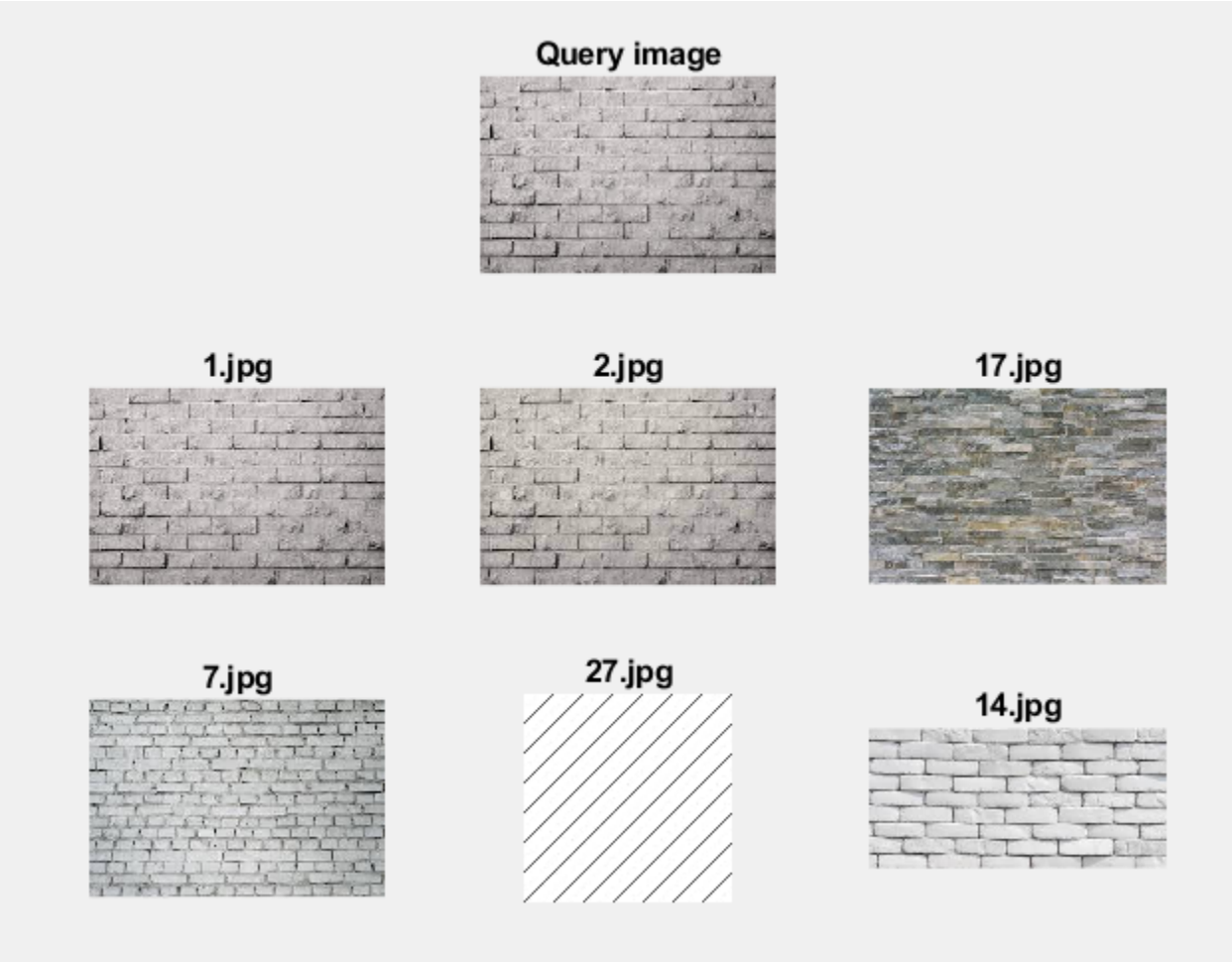
regularity.m

```
function Freg=regularity(graypic,windowsize)

[h,w]=size(graypic);

k=0;

for i=1:windowsize:h-windowsize

     for j=1:windowsize:w-windowsize

          k=k+1;
crs(k)=coarseness(graypic(i:i+windowsize-1,j:j+windowsize-1),5);

con(k)=contrast(graypic(i:i+windowsize-1,j:j+windowsize-1));
[dire(k),sita]=directionality(graypic(i:i+windowsize-1,j:j+windowsize
-1));
lin=linelikeness(graypic(i:i+windowsize-1,j:j+windowsize-1),sita,4)*1
0;

     end

end

% Find the standard deviation of the above parameters

Dcrs=std(crs,1);

Dcon=std(con,1);

Ddir=std(dire,1);

Dlin=std(lin,1);



Freg=1-(Dcrs+Dcon+Ddir+Dlin)/4/100;

end
```

roughness.m

```
Frgh=Fcrs+Fcon;
```

OUTPUT :



Elapsed time is 97.436472 seconds.

Q2. Color Plane Separation

CODE

```
clc;

clear all;
```

```
close all;

tic;

query_image_path = '10.jpg';

queryImgFeatures =
getTamuraFeaturesRGB(getImgFilePath(query_image_path));

names = ['file_name',"Red_Contrast", "Red_Directionality",
"Red_Coarseness","Red_Linelikeness", "Red_Regularity",
"Red_Roughness","Green_Contrast", "Green_Directionality",
"Green_Coarseness","Green_Linelikeness", "Green_Regularity",
"Green_Roughness","Blue_Contrast", "Blue_Directionality",
"Blue_Coarseness","Blue_Linelikeness", "Blue_Regularity",
"Blue_Roughness","City Block Distance"];

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

% Reading the images of textures from the image base

D = './img2';

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

% Looping through all the images in the directory

for k=1:10

image_path = S(k).name;

image_tamura_feature =
getTamuraFeaturesRGB(getImgFilePath(image_path));

image_feature = [];

image_feature{end+1} = image_path;
```

```
% Calculating the City Block distance between the image and the query

city_block_distance = 0;

     for i = 1:18

     city_block_distance = city_block_distance +
abs(image_tamura_feature(i) - queryImgFeatures(i));

     image_feature{end+1} = image_tamura_feature(i);

     end

image_feature{end+1} = city_block_distance;

% Appending the result in the table

   info_table = [info_table; image_feature];

end

info_table = sortrows(info_table, 'City Block Distance');

writetable(info_table, 'lab7.xls','Sheet', 2);


% Displaying the first 6 nearest image

subplot(3, 3, 2);

query_image = imread(getImgFilePath(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(I);

title(char(file_names(i)));

end

toc;


% |#| Function to retrieve file path

function filePath = getImgFilePath(imgName)

      imgSetPath = "./img2/";

filePath = sprintf('%s%s', imgSetPath, imgName);

end
```

getTamuraFeaturesRBG.m

```
function feat = getTamuraFeaturesRGB(imgFilePath)

feat = double(zeros(1,6));

img = imread(imgFilePath);

red = img(:,:,1);

green = img(:,:,2);

blue = img(:,:,3);
```

```
R_DLMI = double(red(:)); % Double Linear Matrix Image

G_DLMI = double(green(:)); % Double Linear Matrix Image

B_DLMI = double(blue(:)); % Double Linear Matrix Image


% Red

% Contrast

alpha = 0.25;

feat(1, 1) = var(R_DLMI)/(kurtosis(R_DLMI)^alpha);

% Directionality

[feat(1, 2), sita] = directionality(red);

% Coarseness

feat(1, 3) = coarseness(red, 5);

% Linelikeness

feat(1, 4) = linelikeness(red, sita, 4);

% Regularity

feat(1, 5) = regularity(red, 64);

% Roughness

feat(1, 6) = feat(1, 1) + feat(1, 3);


% Green

% Contrast
```

```
alpha = 0.25;

feat(1, 7) = var(G_DLMI)/(kurtosis(G_DLMI)^alpha);

% Directionality

[feat(1, 8), sita] = directionality(green);

% Coarseness

feat(1, 9) = coarseness(green, 5);

% Linelikeness

feat(1, 10) = linelikeness(green, sita, 4);

% Regularity

feat(1, 11) = regularity(green, 64);

% Roughness

feat(1, 12) = feat(1, 7) + feat(1, 9);



% Blue

% Contrast

alpha = 0.25;

feat(1, 13) = var(B_DLMI)/(kurtosis(B_DLMI)^alpha);

% Directionality

[feat(1, 14), sita] = directionality(blue);

% Coarseness

feat(1, 15) = coarseness(blue, 5);
```

```
% Linelikeness

feat(1, 16) = linelikeness(blue, sita, 4);

% Regularity

feat(1, 17) = regularity(blue, 64);

% Roughness

feat(1, 18) = feat(1, 13) + feat(1, 15);

end
```

OUTPUT