CSE3018 : Content Based Image and Video Retrieval

Lab - 6

Naga Harshith Bezawada
19BCE1547

Derive the color coherence vector of the given image. Implement a CBIR using these features and City Block distance metric.

CODE :

```
clc

clear all

close all

query_image_path = './images/blue3.jpg';

names = {'file_name'};

for i=1:16

names{end+1} = sprintf('%s%d', 'C-', i);

names{end+1} = sprintf('%s%d', 'NC-', i);

end

names{end+1} = 'city_block_distance';

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

query_ccv_feature = getCCVfeature(query_image_path);

D = './images';

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

for k=1:numel(S)

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

image_ccv_feature = getCCVfeature(image_path);

city_block_distance = 0;
```

```
for i = 1:32

     city_block_distance = city_block_distance +
abs(image_ccv_feature{i} - query_ccv_feature{i});

end

image_feature = [S(k).name, image_ccv_feature, city_block_distance];

info_table = [info_table; image_feature];

end

info_table = sortrows(info_table, 'city_block_distance');

writetable(info_table, 'lab6.xls');

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(I);

title(char(file_names(i)));

end


function[x] = getCCVfeature(image_path)

image = imread(image_path);

image = imresize(image, [64, 64]);

img = rgb2gray(image);

img = imgaussfilt(img, 2);
```

```
steps = 256/16;

levels = steps:steps:256;

img = imquantize(img, levels);

T = cell2table(cell(0, 2), 'VariableNames', {'Intensity',
'Frequency'});

[s0, s1] = size(img);

CCV = zeros(16, 3);

for i=1:16

CCV(i, 1) = i;

end

VISITED = zeros(s0, s1);

for i=1:s0

for j=1:s1

     if VISITED(i, j) == 1

          continue

     else

          [ni, nv] = getCCV(img, i, j, VISITED);

          VISITED = nv;

     end

     newrow = {img(i, j), ni};

     T = [T;newrow];

end

end


tao = 250;


for i=1:size(T, 1)

if (T{i, 2} >= tao)
```

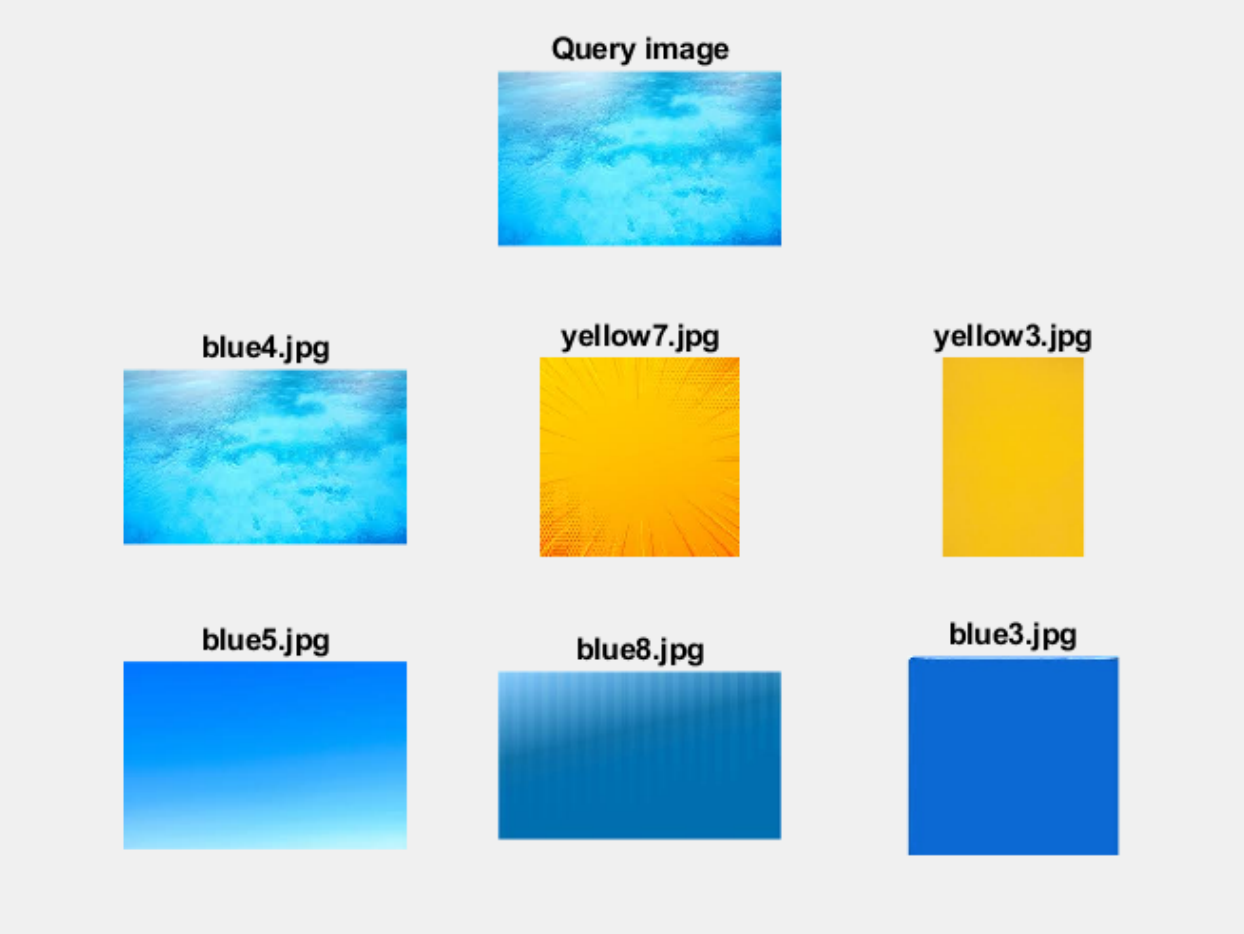
```
     CCV(T{i, 1}, 2) = CCV(T{i, 1}, 2) + T{i, 2};
else
     CCV(T{i, 1}, 3) = CCV(T{i, 1}, 3) + T{i, 2};
end
end
ccv_feature = {};
for i=1:size(CCV, 1)
   ccv_feature = [ccv_feature, CCV(i, 2), CCV(i, 3)];
end
x = ccv_feature;
end



function[x,y] = getCCV(img, i, j, VISITED)
[s0, s1] = size(img);
if i<1 || i>s0 || j<1 || j>s1
x = 0;
y = VISITED;
return
end

if VISITED(i, j) == 1
x = 0;
y = VISITED;
return
end
```

```
if numel(unique(img)) == 1
   x = s0*s1;
   y = ones(s0, s1);
   return
end
n = 1;  % Initializing the frequency of the current patch
VISITED(i, j) = 1;  % Marking the current pixel as visited
if i ~= 1
if img(i, j) == img(i-1, j)
      [ni, nv] = getCCV(img, i-1, j, VISITED);
      n = n + ni;
      VISITED = nv;
end
end
if i ~= s0
if img(i, j) == img(i+1, j)
      [ni, nv] = getCCV(img, i+1, j, VISITED);
      n = n + ni;
      VISITED = nv;
end
end
if j ~= 1
if img(i, j) == img(i, j-1)
      [ni, nv] = getCCV(img, i, j-1, VISITED);
      n = n + ni;
      VISITED = nv;
end
```

```
end
if j ~= s1
if img(i, j) == img(i, j+1)
      [ni, nv] = getCCV(img, i, j+1, VISITED);
      n = n + ni;
      VISITED = nv;
end
end
if (i~=1 && j~=1)
if img(i, j) == img(i-1, j-1)
      [ni, nv] = getCCV(img, i-1, j-1, VISITED);
      n = n + ni;
      VISITED = nv;
end
end
if (i~=1 && j~=s1)
if img(i, j) == img(i-1, j+1)
      [ni, nv] = getCCV(img, i-1, j+1, VISITED);
      n = n + ni;
      VISITED = nv;
end
end
```

```
if (i~=s0 && j~=1)
if img(i, j) == img(i+1, j-1)
      [ni, nv] = getCCV(img, i+1, j-1, VISITED);
      n = n + ni;
      VISITED = nv;
end
end
if (i~=s0 && j~=s1)
if img(i, j) == img(i+1, j+1)
      [ni, nv] = getCCV(img, i+1, j+1, VISITED);
      n = n + ni;
      VISITED = nv;
end
end
x = n;
y = VISITED;
end
```

OUTPUT : Input - blue4.jpg



OUTPUT : Input - yellow10.jpg