Reverse Image Search

Shubham Kumar(14BCE0324)
Srijan Singh(14BCE0313)
Slot-B1,Prof-Don S

November 6, 2016

1 Abstract

Reverse image search is a content-based image retrieval (CBIR) query technique that involves providing the CBIR system with a sample image that it will then base its search upon; in terms of information retrieval, the sample image is what formulates a search query. Content-Based Image Retrieval (CBIR) uses the visual contents of an image such as color, shape, texture, and spatial layout to represent and index the image. General techniques for image retrieval are color, texture and shape. These techniques are applied to get an image from the image database.

2 Introduction

CBIR is the process of retrieving images from a database or library of digital images according to the visual content of the images. In other words, it is the retrieving of images that have similar content of colors, textures or shapes. Images have always been an inevitable part of human communication and its roots millennia ago. Images make the communication process more interesting, illustrative, elaborate, understandable and transparent .In CBIR system, it is usual to group the image features in three main classes: color, texture and shape. Ideally, these features should be integrated to provide better discrimination in the comparison process. Color is by far the most common visual feature used in CBIR, primarily because of the simplicity of extracting color information from images. To extract information about shape and texture feature are much more complex and costly tasks, usually performed after the initial filtering provided by color features.

3 About (CBIR)

A regular image search on any search engine like Google begins by entering a keyword like Eiffel Tower, which is followed by a set of images related to Eiffel

Tower in Paris. However, a lot of times, there is a need for searching with a given image and finding related images i.e. instead of giving a text as an input to search, an image is given as an input. For example, an image of Eiffel Tower is given as input and then we would like to see images not only of Eiffel Tower but also of similar towers. Existing techniques generates results which contains exactly matching towers i.e. only Eiffel Tower in Paris. They do not generate similar towers from around the world in the result.

4 Aim of Project

Our aim is to find the similar images from the database from the input give image on the basis of color. We would input the image to the gui of the matlab code and on the basis of color and texture the code will extract the similar images from the database.

5 Process of Extracting Images from Database

5.1 Color Feature Based Retrieval

Several methods for retrieving images on the basis of color similarity have been described in the literature, but most are variations on the same basic idea. Each image added to the collection is analyzed to compute a color histogram, which shows the proportion of pixels of each color within the image. The color histogram for each image is then stored in the database. At search time, the user can either specify the desired proportion of each color (75 percent olive green and 25 percent red, for example), or submit an example image from which a color histogram is calculated. Either way, the matching process then retrieves those images whose color histograms match those of the query most closely.

5.2 RGB to HSV Conversion

The obtainable HSV colors lie within a triangle whose vertices are defined by the three primary colors in RGB space.

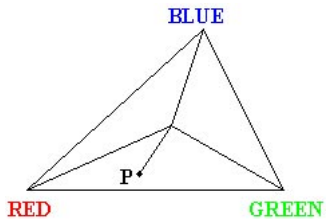


Figure 1: rgb triangle

The hue of the point P is the measured angle between the line connecting P to the triangle center and line connecting RED point to the triangle center.The saturation of the point P is the distance between P and triangle center.The value (intensity) of the point P is represented as height on a line perpendicular to the triangle and passing through its center. The grayscalepoints are situated onto the same line.

5.3 Proposed Method

5.3.1 Color Histogram

Step1. Convert RGB color space image into HSV color space.

Step2. Color quantization is carried out using color histogram by assigning 8 level each to hue, saturation and value to give a quantized HSV space with 8x8x8=512 histogram bins.

Step3. The normalized histogram is obtained by dividing with the total number of pixels.

Step4. Repeat step1 to step3 on an image in the database.

Step5. Calculate the similarity matrix of query image and the image present in the database.

Step6. Repeat the steps from 4 to 5 for all the images in the database.

Step7. Retrieve the images.

6 Results

Through this project we found the CBIR technique based on colour and texture.The similarity measure by a given query image involves searching the database for similar coefficients.Quadratic distance is suitable and effective method which is widely used in image retrieval area. The images in the database are ranked according to their distance d to the query image in ascending orders, and then the ranked images are retrieved. The computed distance is ranked according to closest similar; in addition, if the distance is less than a certain threshold set, the corresponding original images is close or match the query.

7 Conclusion

In this we are able to extract the images based on their color feature through a matlab code.We are able to find similar images from database successfully.

8 References

1. "Histology Image Retrieval in Optimized Multifeature Spaces" by Qianni Zhang and Ebroul Izquierdo, Senior Member of IEEE, published in IEEE Journal of Biomedical and Health Informatics , vol,17,No.1 , January 2013.

2. "Content Based Image Retrieval Using Color and Shape Features" by Reshma Chaudhari,A. M. Patil in International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering,Vol. 1, Issue 5, November 2012.

3. Shamik Sural, Gang Qian and Sakti Pramanik, "Segmentation and Histogram Generation Using the HSV Color Space for Image Retrieval", International Conference on Image processing, Vol. 2, pp. 589-592, 2002.

4. http://in.mathworks.com/

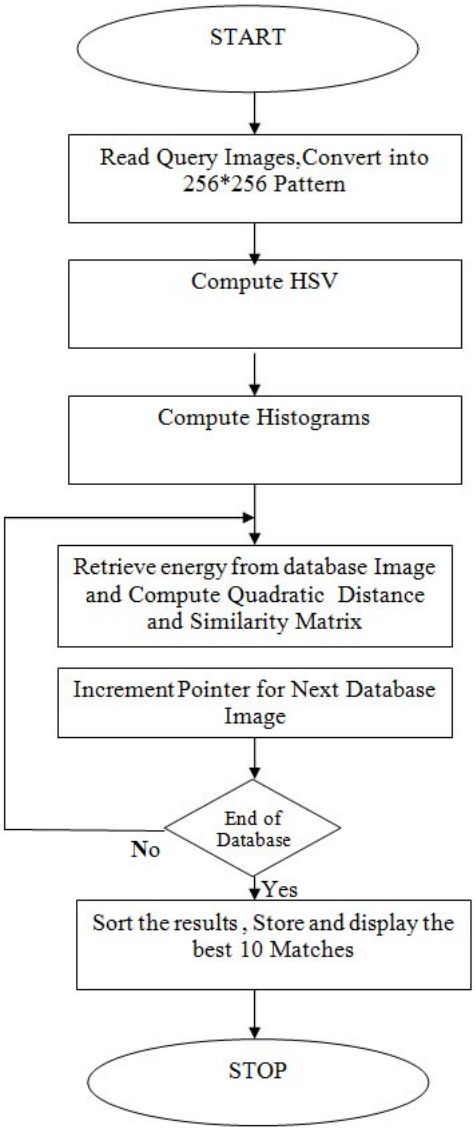


Figure 2: Flowchart of color retrival

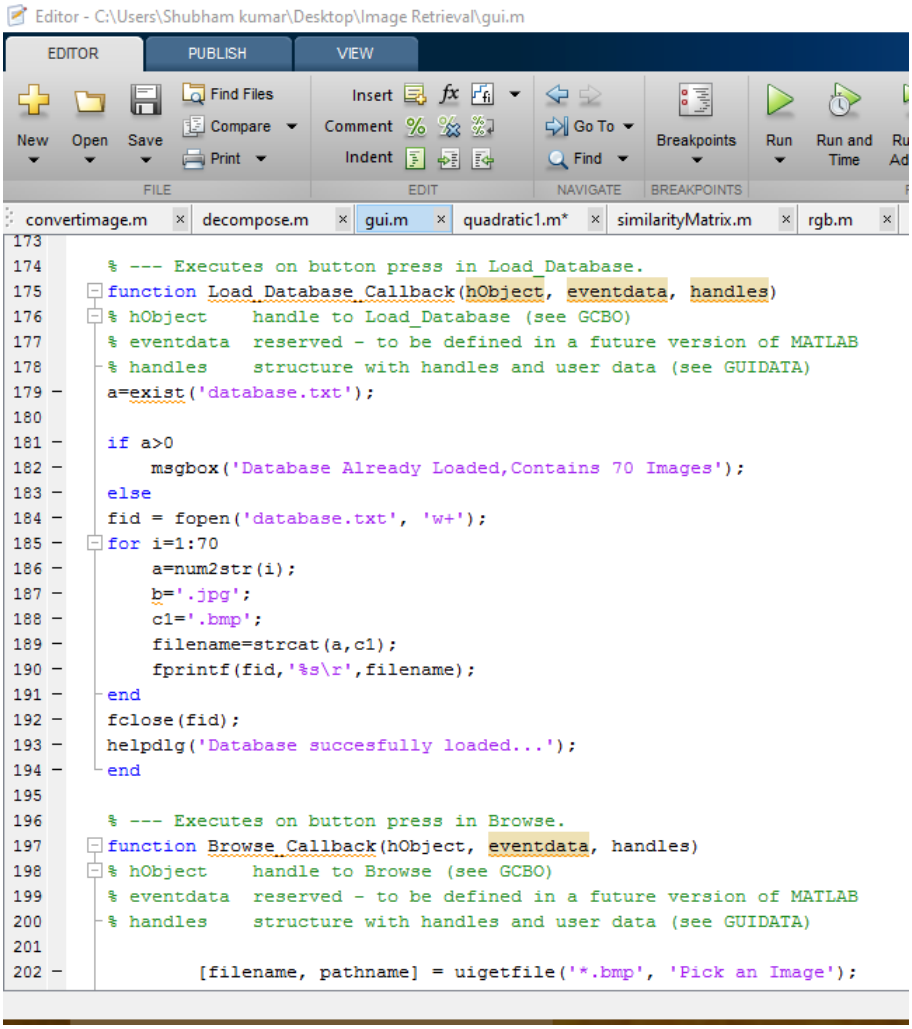
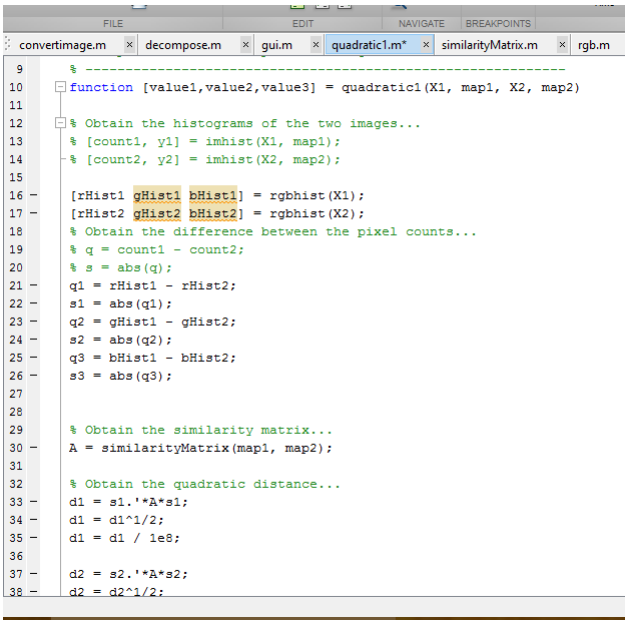
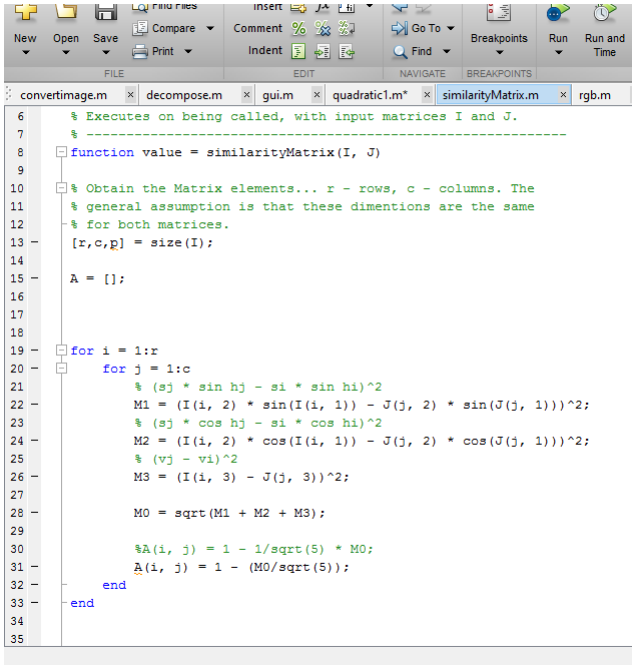

Editor - C:\Users\Shubham kumar\Desktop\Image Retrieval\gui.m
EDITOR
PUBLISH
VIEW
New
Open
Save
Find Files
Compare
Print
Insert
Comment
Indent
Go To
Find
Breakpoints
Run
Run and Time
FILE
EDIT
NAVIGATE
BREAKPOINTS
convertimage.m
decompose.m
gui.m
quadratic1.m*
similarityMatrix.m
rgb.m
173
174 % --- Executes on button press in Load_Database.
175 function Load_Database_Callback(hObject, eventdata, handles)
176 % hObject handle to Load_Database (see GCBO)
177 % eventdata reserved - to be defined in a future version of MATLAB
178 % handles structure with handles and user data (see GUIDATA)
179 a=exist('database.txt');
180
181 if a>0
182 msgbox('Database Already Loaded,Contains 70 Images');
183 else
184 fid = fopen('database.txt', 'w+');
185 for i=1:70
186 a=num2str(i);
187 b='.jpg';
188 c1='.bmp';
189 filename=strcat(a,c1);
190 fprintf(fid,'%s\r',filename);
191 end
192 fclose(fid);
193 helpdlg('Database succesfully loaded...');
194 end
195
196 % --- Executes on button press in Browse.
197 function Browse_Callback(hObject, eventdata, handles)
198 % hObject handle to Browse (see GCBO)
199 % eventdata reserved - to be defined in a future version of MATLAB
200 % handles structure with handles and user data (see GUIDATA)
201
202 [filename, pathname] = uigetfile('*.bmp', 'Pick an Image');

Figure 3: GUI



```
convertimage.m | decompose.m | gui.m | quadratic1.m* | similarityMatrix.m | rgb.m

% ------------------------------------------------------------
function [value1,value2,value3] = quadratic1(X1, map1, X2, map2)

% Obtain the histograms of the two images...
% [count1, y1] = imhist(X1, map1);
% [count2, y2] = imhist(X2, map2);

[rHist1 gHist1 bHist1] = rgbhist(X1);
[rHist2 gHist2 bHist2] = rgbhist(X2);
% Obtain the difference between the pixel counts...
% q = count1 - count2;
% s = abs(q);
q1 = rHist1 - rHist2;
s1 = abs(q1);
q2 = gHist1 - gHist2;
s2 = abs(q2);
q3 = bHist1 - bHist2;
s3 = abs(q3);


% Obtain the similarity matrix...
A = similarityMatrix(map1, map2);

% Obtain the quadratic distance...
d1 = s1.'*A*s1;
d1 = d1^1/2;
d1 = d1 / 1e8;

d2 = s2.'*A*s2;
d2 = d2^1/2;
```

Figure 4: Quadratic histogram



```
New | Open | Save | Compare | Print | Comment | Indent | Go To | Find | Breakpoints | Run | Run and Time
FILE | EDIT | NAVIGATE | BREAKPOINTS

convertimage.m | decompose.m | gui.m | quadratic1.m* | similarityMatrix.m | rgb.m

% Executes on being called, with input matrices I and J.
% ------------------------------------------------------------
function value = similarityMatrix(I, J)

% Obtain the Matrix elements... r - rows, c - columns. The
% general assumption is that these dimentions are the same
% for both matrices.
[r,c,p] = size(I);

A = [];



for i = 1:r
    for j = 1:c
        % (sj * sin hj - si * sin hi)^2
        M1 = (I(i, 2) * sin(I(i, 1)) - J(j, 2) * sin(J(j, 1)))^2;
        % (sj * cos hj - si * cos hi)^2
        M2 = (I(i, 2) * cos(I(i, 1)) - J(j, 2) * cos(J(j, 1)))^2;
        % (vj - vi)^2
        M3 = (I(i, 3) - J(j, 3))^2;

        M0 = sqrt(M1 + M2 + M3);

        %A(i, j) = 1 - 1/sqrt(5) * M0;
        A(i, j) = 1 - (M0/sqrt(5));
    end
end
```

Figure 5: Similarity matrix

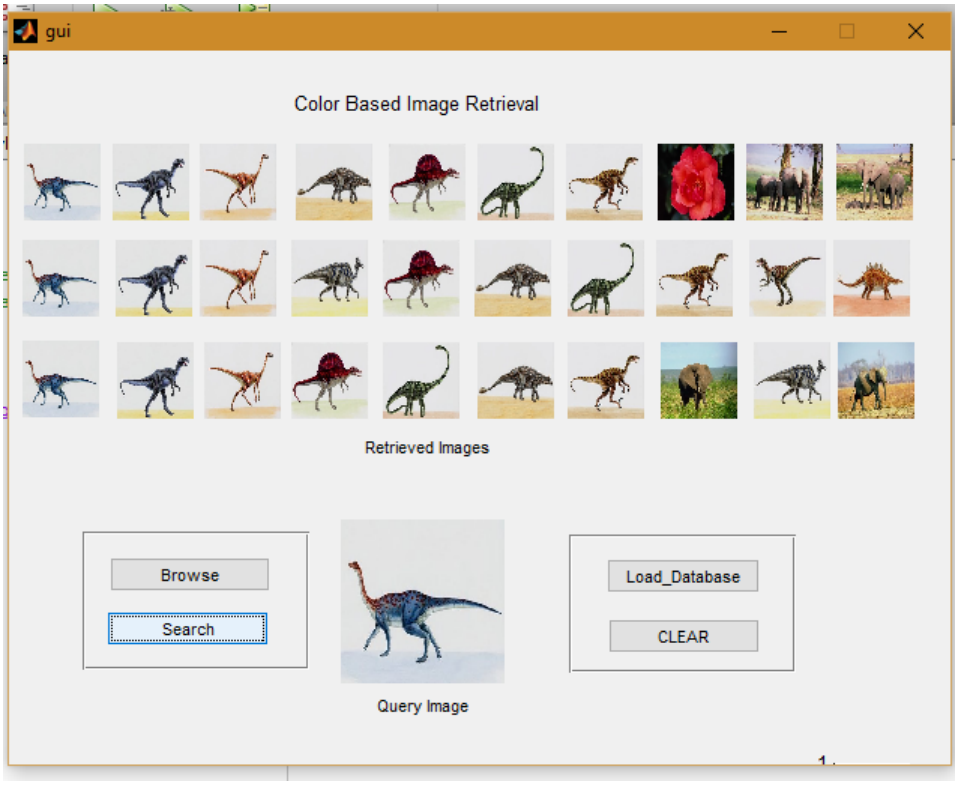


Figure 6: Output In gui