cs512 Final project: Report

Imran Ali
Graduate student, A20362180
Department of Computer Science
Illinois Institute of Technology

Arun Rajagopalan
Graduate student, A20360689
Department of Computer Science
Illinois Institute of Technology

November 18, 2015

Abstract:
The problem or the task is to perform content based image retrieval in an efficient manner. Content based image retrieval (CBIR) is a significant research area with myriad applications. It means that the search analyzes the contents of the image rather than the metadata such as keywords, tags, or descriptions associated with the image. CBIR is desirable because searches that rely purely on metadata are dependent on annotation quality and completeness. The term "content" in the context of CBIR might refer to colors, shapes, textures, or any other information that can be derived from the image itself. In this project, we use the color features, texture features and a binary tree to describe the image contents.

1 Problem statement

To improve the performance of content based image retrieval by using binary tree structures to describe high level image features in addition to the usage of color and texture features.

2 Proposed solution

The overall structure of the image retrieval system has seven separated parts: 1. Image database consisting of hundreds or thousands of images among which a query image is searched; 2. Feature Extraction which retrieves features from images and sends them to appropriate parts; 3. Database of extracted features received from part 2; 4. Input Query image. 5. Feature Vectors of query image 6. Search and retrieval part that searches the Feature Vectors database in order to find the images similar to the query image; 7. User interface which shows the retrieved images from part 6.

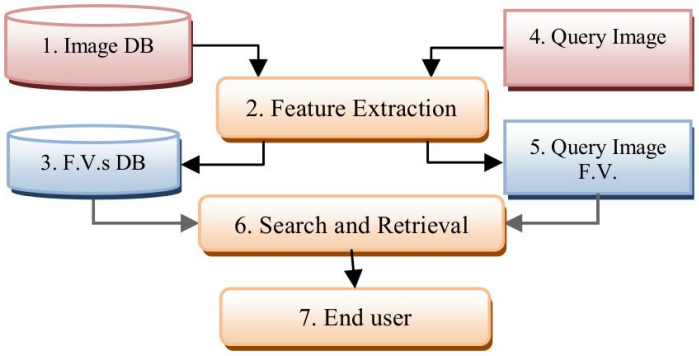


Illustration 1: Overall structure of an image retrieval system

In this project, feature extraction is divided into three steps, color extraction, texture extraction and producing binary tree corresponding to each images.

3 Details on implementation

3.1 Feature Extraction

3.1.1 Color Extraction

Task 1: Image division

First we extract a color feature vector or histogram from the image but, before doing that, we divide the taken image into meaningful parts, so that a feature vector for each part can be found later. For this, the image is divided into 5 parts. The first four parts belong to four corners and the fifth part belongs to the center region of the image. This division is more meaningful since we assume most of the images contain important content in the center so one part covers all the center region. And most of the images will contain a sky and a ground which will be covered by the four corner regions. Later, the comparison is made between the same parts of the target and the query image for better results.

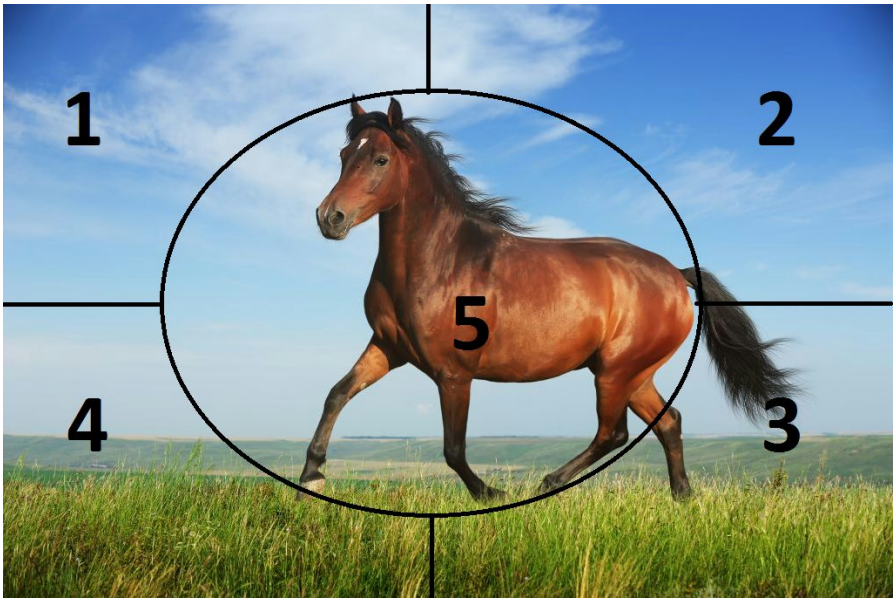


Illustration 2: Dividing each image into several regions before histogram extraction

Task 2: Feature Extraction

Various color spaces such as RGB, HSV, YCbCr, CIE LAB, CIE LUV, etc. are used in CBIR systems. But the HSV color space is used because of its perceptual uniformity. That is, the three components H (Hue), S (Saturation) and V (Value) correspond to the color attributes closely associated with the way that human eye perceives the color. The approach here is to extract two histograms, one for Hue and one for Saturation. Since V is directly related to brightness level, it is not considered in our color measurement approach. The Hue circle is quantized into 32 degrees, and Saturation into 16 levels. Thus their corresponding histograms have 32 and 16 bins. Core routine for color feature extraction follows.

For every file in the image database, do:

For BGR to HSV conversion use:

image = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)

Divide the image into five parts, four corners spaces and one eclipse in the centre:

(h, w) = image.shape[:2]

*(cX, cY) = (int(w * 0.5), int(h * 0.5))*

*segments = [(0, cX, 0, cY), (cX, w, 0, cY), (cX, w, cY, h),(0, cX, cY, h)](axesX, axesY) = (int(w * 0.75) / 2, int(h * 0.75) / 2)*

ellipMask = np.zeros(image.shape[:2], dtype = "uint8")

cv2.ellipse(ellipMask, (cX, cY), (axesX, axesY), 0, 0, 360, 255, -1)

for (startX, endX, startY, endY) in segments:

    cornerMask = np.zeros(image.shape[:2], dtype = "uint8")

    cv2.rectangle(cornerMask, (startX, startY), (endX, endY), 255, -1)

    cornerMask = cv2.subtract(cornerMask, ellipMask)

Calculate the histogram for each section using:

hist = cv2.calcHist([image], [0, 1, 2], mask, (16, 32, 1),[0, 180, 0, 256, 0, 256])

Copy this histogram in the index file hsv.csv

3.1.2 Texture Extraction

To extract texture features, we perform wavelet decomposition of image regions. When we apply wavelet decomposition to grayscale image regions, four sub-images are produced. One is a low resolution copy of the original image and the others are 3 band passed filter results in the horizontal, vertical and diagonal directions respectively. To numerically represent these sub-images which contain texture features, we find mean and variance of each of these sub-images.

Detailed texture extraction process is composed of: 1. dividing image into 8×8 = 64 equal regions; 2. applying Wavelet on each region. 3. calculating Mean and Variation of four sub-images corresponding to each region and concatenating them; 4. concatenating all Mean and Variance feature vectors of every region. At the end of this routine, two vectors will be obtained which describe texture information of image. Both of these vectors are normalized and flattened to get the texture feature histogram. The core routine for texture feature extraction is as follows.

For every file in the image database, do:

For BGR to grayscale conversion and normalization use:

gray_img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

imArray = np.float32(gray_img)

imArray /= 255;

Divide the image into 8x8 regions by using the image shape detail:

(h, w) = gray_img.shape[:2]

*(cX, cY) = (int(w * 0.125), int(h * 0.125))*

Now for each image region, perform wavelet decomposition using db1 mode at level 1:

*coeffs=pywt.wavedec2(imArray[r*cY:r*cY+cY,c*cX:c*cX+cX], 'db1', 1)*

Then for each subimage of each image region,find men and variance and concatenate them to get mean and variance vectors:

(coeffs[0] will be a low res copy of image region and coeffs[1][0..2] will be 3 band passed filter results in horizontal, vertical and diagonal directions respectively)

if i == 0:

tMean.append(np.mean(coeffs[i]))

tVar.append(np.var(coeffs[i]))

else:

tMean.append(np.mean(coeffs[1][i-1]))

tVar.append(np.var(coeffs[1][i-1]))

Finally append the mean and variance vectors of each region into one single feature array, normalize and flatten it.

Save this final array to index file texture.csv

3.1.3 Binary Partitioning Tree

Binary tree is obtained by performing segmentation based feature extraction and to do so, safe color cube is used.

A color palette including fewer colors than real color space is needed to quantize the image. For this purpose Safe color cube has been used because it covers the entire RGB color space. Safe Color Cube consists of 216 colors in RGB mode; each R, G and B can only be 0, 42, 84, 126, 168, 210 or 252. Thus, RGB triples of these values give us (6)3 = 216 colors [10]. To quantize an image, colors of its pixels or a group of pixels (their mean color) will be quantized into safe color pallette. By having an image with 216 specific colors, it is expected to have distinctive regions with homogeneous color. Now the image is converted into grey image. And total area of each of the segments with same grey level is calculated for each of the 6 grey levels.

For every file in the image database, do:

Converting the image into safe color cube:

image = image/42

*image = image*42*

Then converting into grey image

image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

Finding the segmentation feature

hist = cv2.calcHist([image], [0], mask, [6],[0, 253])

Now copy the feature in the index file.

3.1.4 Searching, ranking and retrieval

The search and retrieval is based on measuring the distance between feature vectors (color, texture and tree) of query image and the target image. The rank of each target image is calculated based on their distance from the query image. Chi-squared distance is found between each of the three feature vectors of query image and each database image and a weighted sum of distances is used to rank the database images with respect to the query image as below.

For histograms from the 3 index files (color, texture and tree) x=[x_1,..,x_n] and y=[y_1,...,y_n]

Chi-squared distance d(x,y) = sum((xi-yi)^2 / (xi+yi)) / 2

Now these distances are stored in a dictionary and sorted and finally the top ranks are copied in the result directory.

4 Results and discussion:

To run the program, from the current folder where the scripts are present, the following command is to be issued in the terminal.

To run the index.py file which creates the index files hsv.csv, texture.csv and btree.csv
> python index.py -c hsv.csv -t texture.csv -b btree.csv -d <full path of dataset>

To run the rank.py file which ranks images in the given dataset with respect to the query image (rank.py file also copies the high ranking 15 images to the result directory)
> py rank.py -c hsv.csv -t texture.csv -b btree.csv -q <full path of query image> -d <full path of dataset>

The terminal look when the index.py is run is as follows.

```
rasuishere@ARajago6:~/arun_rajagopalan_as_final_project/code$ python index.py -c
 hsv.csv -t texture.csv -b btree.csv -d /home/rasuishere/arun_rajagopalan_as_fin
al_project/data/dataset
rasuishere@ARajago6:~/arun_rajagopalan_as_final_project/code$
```

The terminal and folder outputs for different query images are as follows.

First query image: query-horse.jpg

Terminal output:

```
rasuishere@ARajago6:~/arun_rajagopalan_as_final_project/code$ python rank.py -c
hsv.csv -t texture.csv -b btree.csv -q /home/rasuishere/arun_rajagopalan_as_fina
l_project/data/query-horse.jpg -d /home/rasuishere/arun_rajagopalan_as_final_pro
ject/data/dataset
query-horse.jpg    1.59451901375e-11
horse-07.jpg    7.80136453804
horse7.jpg    10.077313652
Green-grassland-and-brown-horse.jpg    12.8616361175
horse5.jpg    12.8840331659
horse4.jpg    13.3047788629
Majestic-Stallion.jpg    13.7075429539
Animals_Wallpapers_Horse_Wallpaper-19.jpg    13.7450501754
Horse and Beach White Sand.jpg    13.9073759319
Horse-1.jpg    14.3588706877
sorrel-horse.jpg    14.9610233964
horse_2465281b.jpg    15.3081650396
More-horse-wallpapers-horses-15705243-1600-1200.jpg    15.5218624642
gty_horse_mi_130226_wmain.jpg    15.8392970227
Feeding-the-hot-or-fizzy-horse.jpg    15.9587915035
rasuishere@ARajago6:~/arun_rajagopalan_as_final_project/code$
```

result folder state:

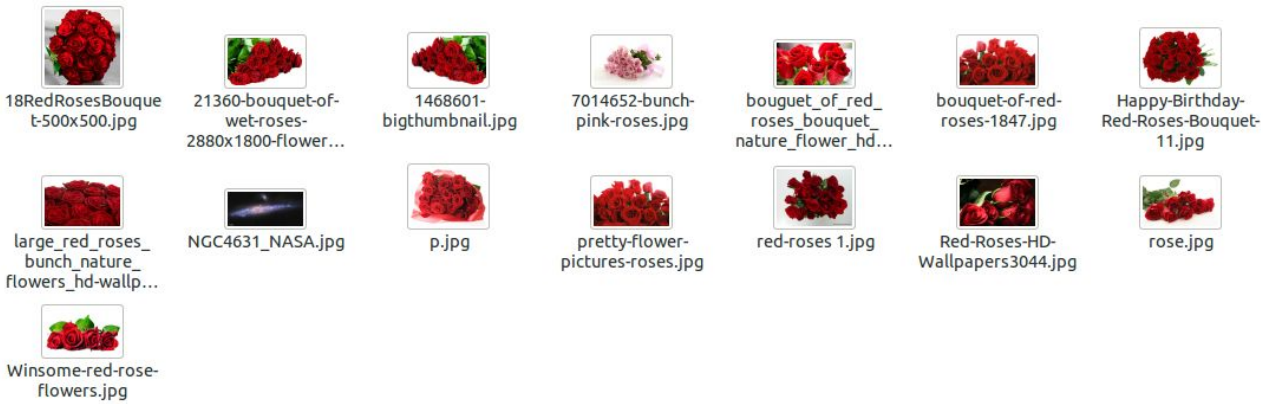
result folder state: 15 out of 15 are correct

Second query image: query-rose.jpg

Terminal output:

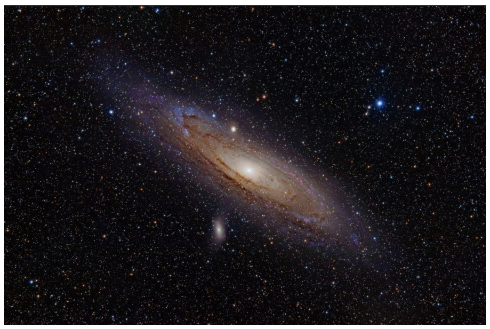
```
rasuishere@ARajago6:~/arun_rajagopalan_as_final_project/code$ python rank.py -c
hsv.csv -t texture.csv -b btree.csv -q /home/rasuishere/arun_rajagopalan_as_fina
l_project/data/query-rose.jpg -d /home/rasuishere/arun_rajagopalan_as_final_proj
ect/data/dataset
21360-bouquet-of-wet-roses-2880x1800-flower-wallpaper.jpg    2.8229362954e-12
Happy-Birthday-Red-Roses-Bouquet-11.jpg    3.75837550533
Winsome-red-rose-flowers.jpg    4.49579848035
18RedRosesBouquet-500x500.jpg    5.37267653197
large_red_roses_bunch_nature_flowers_hd-wallpaper-1487647.jpg    5.5630465852
1468601-bigthumbnail.jpg    7.24096139206
red-roses 1.jpg    7.63418854145
bouguet_of_red_roses_bouquet_nature_flower_hd-wallpaper-1739504.jpg    8.1015494
4141
rose.jpg    8.24337301615
NGC4631_NASA.jpg    8.38873250006
bouquet-of-red-roses-1847.jpg    8.54778063953
p.jpg    8.58551191077
pretty-flower-pictures-roses.jpg    8.65436534747
7014652-bunch-pink-roses.jpg    8.70824062334
Red-Roses-HD-Wallpapers3044.jpg    9.20913409513
rasuishere@ARajago6:~/arun_rajagopalan_as_final_project/code$
```

result folder state:



result folder state: 14 out of 15 are correct

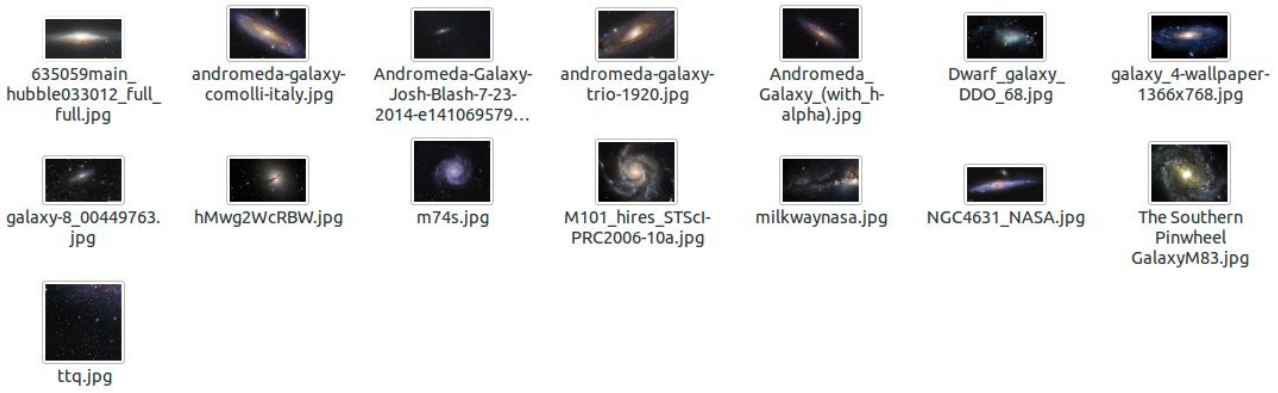
Third query image: query-gal.jpg



Terminal output:

```
rasuishere@ARajago6:~/arun_rajagopalan_as_final_project/code$ python rank.py -c
hsv.csv -t texture.csv -b btree.csv -q /home/rasuishere/arun_rajagopalan_as_fina
l_project/data/query-gal.jpg -d /home/rasuishere/arun_rajagopalan_as_final_proje
ct/data/dataset
Andromeda_Galaxy_(with_h-alpha).jpg    2.0282313824e-11
andromeda-galaxy-comolli-italy.jpg    12.3064438127
m74s.jpg    13.4488634889
M101_hires_STScI-PRC2006-10a.jpg    16.9330488294
galaxy-8_00449763.jpg    17.095485174
milkwaynasa.jpg    17.3081815211
Andromeda-Galaxy-Josh-Blash-7-23-2014-e1410695797991.jpg    17.3409459362
Dwarf_galaxy_DDO_68.jpg    17.3546802127
NGC4631_NASA.jpg    18.0288853073
andromeda-galaxy-trio-1920.jpg    18.6480487219
hMwg2WcRBW.jpg    19.3660523913
635059main_hubble033012_full_full.jpg    19.4061124151
The Southern Pinwheel GalaxyM83.jpg    19.6803879896
ttq.jpg    19.8033246245
galaxy_4-wallpaper-1366x768.jpg    20.1703087976
rasuishere@ARajago6:~/arun_rajagopalan_as_final_project/code$
```

result folder state:



result folder state: 15 out of 15 are correct

Thus the tool correctly identifies same category of images in the dataset as the query image and copies them to result folder. It also displays the filenames of high ranked matching images in the console while printing their rank/score value (lower value indicates close match) next to their names.

5 References:

- Content based image retrieval using knowledge of texture, colour and binary tree structure: Canadian Conference on Electrical and Computer Engineering, 2009. CCECE '09. Authors: Mansoori, Z. ; Sharif Univ. of Technol., Tehran ; Jamzad, M.
  http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5090280

- The complete guide to building an image search engine with Python and OpenCV by Adrian Rosebrock
  http://www.pyimagesearch.com/

- Pywavelets 2D multilevel decomposition using wavedec2
  http://www.pybytes.com/pywavelets/ref/2ddwtandidwt.html