CS 6400 Interim Project Report

ColorWorld ---- a Color-based Image Indexing and Retrieval System

Group No. 3

Ke Wang 903058889 wangke@gatech.edu

Qichao Chu 903060535 qc.chu@gatech.edu

Ke is a Master’s student majoring in Computer Science. He achieved his Bachelor of Science degree in China. Bach then he took one fundamental database course, where he learnt some basic database knowledge, and he took part in a small project. Now he plan to concentrate in Database and Software Engineering as he desires to learn more on database.

Qichao is a first year ECE student with limited knowledge about database. He focused on control system in electrical engineering before, and now as he is admitted in the track of Computer System & Software in ECE, he tries to learn more about computer science. Thus, he decide to participate in a more challenging project rather than individual project.

Project GitHub Repository:

https://github.com/wangkekekexili/ColorWorld

1. Problem Statement

Nowadays digital cameras and smartphones with advanced photography features are ubiquitous. People enjoy sharing and publishing image contents. As a result, there is an increasing need to store and fetch these exciting content as well as search them online with efficiency and quality. Baidu, a famous Chinese web service company, provides image retrieval service to the public. However, the service is not adorable as Baidu often returns results that are unrelated to the search keyword and one can guess that Baidu still collect and index images according to the text description.

But advanced image retrieval systems are emerging. IBM QBIC system, developed by IBM Almaden Research Center, and BlobWorld System, developed at UC Berkeley, both takes image contents into consideration.

Our team would like to explore the image retrieval systems, which is quite different from traditional database systems. An image is a large binary object. Lots of unique attributes are related to images, which should be taken into consideration when building the system. It’s a fascinating area with lots of amazing work. Our team haven’t encountered such topics before so we decide to implement an image indexing and retrieval system, through which we can gain knowledge about image attributes, image indexing methods and image retrieval methods. Color, shape, texture and specific object are attributes to an image. They appear in ascending difficulty. As we have little experience before, we would like to explore the basics first and concentrate on color attributes. Quite a lot of work discussing image indexing and retrieval system concerns about color attributes. QBIC system, for example, partly relies on color to decide pattern and content.

In short, we are going to implement a color-based image indexing and retrieval system in Java programming language. Color attribute will be made full use of. Indexing and retrieval methods published in a couple of papers will be studied and implemented. Also, we will design some tests to tell the performance of our system.

1. Overview of Functionality

2.1 In general, we will implement a color-based image indexing and retrieval system. Currently we only take colorful images into consideration and we pick up animals and landscapes images. The image data we use is discussed in detail in the next section. The indexing algorithms will be related to color histogram (We have implemented part of this). The storage of related information will concern B+ tree, keyword tree, and ISAM (Discussed in algorithm section. The ISAM data structure we will implement is a derived version of the original one to adapt to the specific use case in our system). We do not concern much about physical layer such as the storage of images (we use os file system to store images).

2.2 In the system end, we are trying to bring the function of processing image automatically to reality. We will create a processing unit to deal with image input, which is to analysis the color of the input image. If time permits, we will add the function of shape processing. But the function of texture analysis and other aspects of image processing is beyond our consideration as we won’t be able to finish them in relatively short time. Additionally, the processing unit will compare the input image with stored images to see if they have great similarity. If so, the system will assign the keywords of the stored images to the new image, enabling further querying to reach this new image.

2.3 In the user end, we are trying to achieve the function of querying image with a color, a keyword ,an image or a simple natural language sentence. By input a keyword of the image, such as the word ‘Mount’, the system will find similar images with that keyword. To achieve this, we store the image keyword through initialization and assign keywords to new image by comparing with the stored image . Also, the function of querying by color exists in our plan. This function is designed to achieve a higher query speed as the color information is simple and relatively less than keywords in quantity. With the input color, system will find images with high portion of certain color and show all the results to user from GUI.

1. Data Source

We need a number of images to initialize the system, test all the functionalities and test the system to tell the performance. A simple web crawler, designed specifically for images, will be implemented to achieve a couple of jobs.

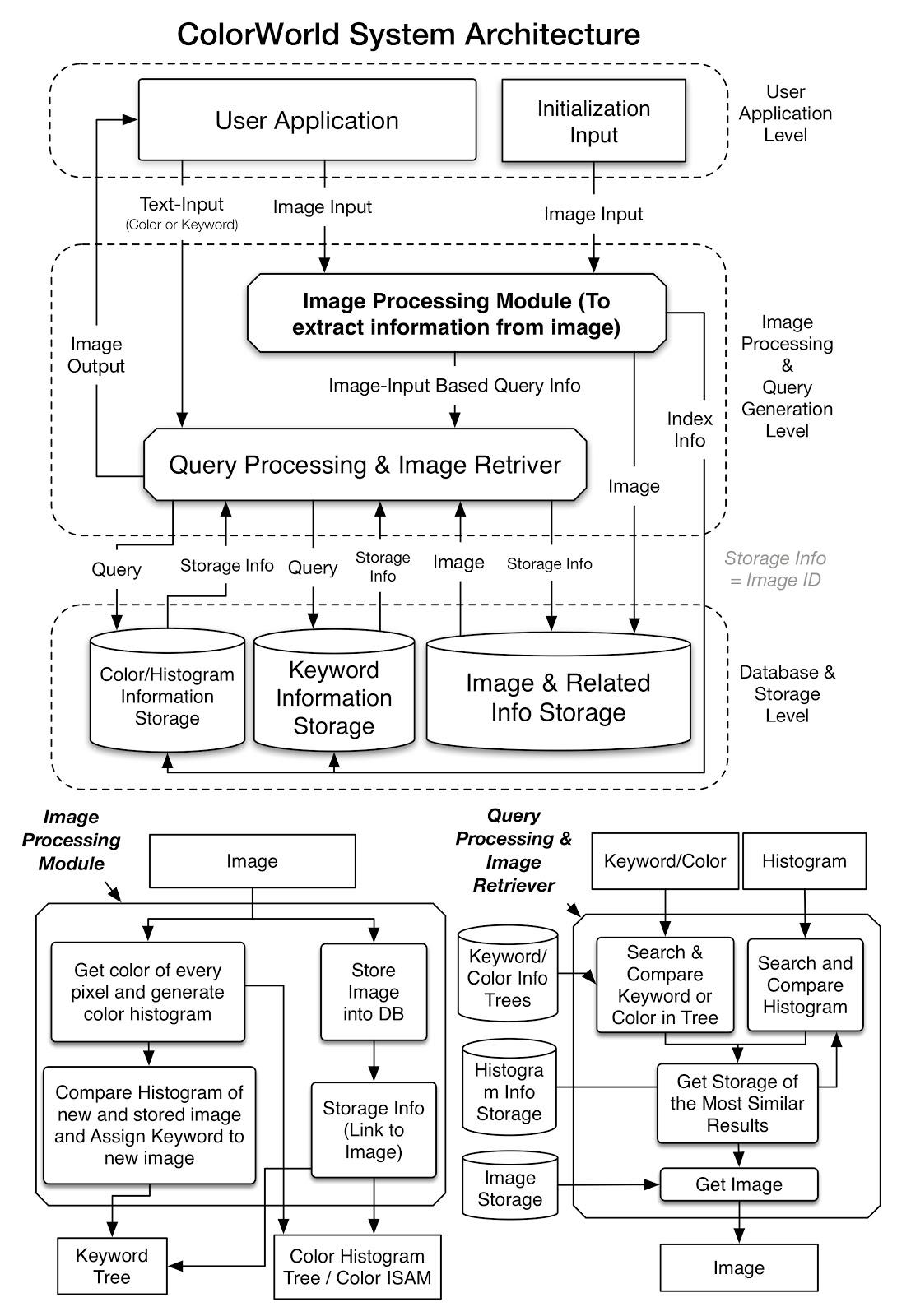
As a constraint, our system is expected to deal with colorful images instead of grayscale images. We will have test for grayscale images. Besides, we use images of animals and landscapes. Because we mainly focus on color, and landscape images seem to work because similar colors infers possible similar landscape. In addition, we think animal images might work because when we use histogram method and intersection distance, background color is more likely to be ignored when comparing two images of a same animal with different background colors.

First, it will be used to automatically obtain images from image research database from other universities. Currently, we are using data from University of Washington (http://www.cs.washington.edu/research/imagedatabase/groundtruth) and CalPhotos image collection from Berkely (http://calphotos.berkeley.edu/). Such data collection is very powerful and we expect to make full use of it.

After initializing the system with images processed, users can use the system through different ways. One of them is that, an image is uploaded by a user to search for similar images. After processing the input image and retrieve similar images, we decide how to deal with the input image according to the result. If a same image is found, then nothing happens. If a similar (we measure this by using different abstract distances) image is found, then the image is added into the system along with information, such as keyword extracted from similar images. If no similar images are found, then the input image is added into the system with no additional information.

The crawler will be used again when there is enough information in the system. Automatically, it obtains images from different websites and let the system to decide how to treat the image.

1. Architecture Overview



The architecture of ColorWorld is showed in the image above. The system is composed of three level: user application level, image processing and query generation level and the database & storage level. The user application level is to let the administrator initialize the system by porting images or let the use querying image in a user-friendly way. The database & storage level is used to store the image and the indexing information about the image. The image processing and query generation level is composed of image processing module and query processing & image retriever unit. The image processing module is to get color of every pixel and generate color histogram. This module is also responsible for assign keywords to the new image and store the image storage information with the indexing information into the keyword tree and color information tree. The query processing & image retriever’s mission is to get image from the image storage by inputting keyword, color or histogram generated from the image processing module.

1. Algorithm & Data Structure Overview

ColorWorld system consists of a couple of algorithms and data structures. As discussed below, Histogram is a simple data structure for storing a histogram of an image, and related algorithms are used to compare two histogram distance. ISAM data structure is used for quick search when giving a specific color. B+ tree is used to search a ID for its related detailed information (including the path to the original image, which stored in the file system). Keyword tree is used to search a keyword for related image IDs. Note: these algorithms and data structure are subject to change while we are developing ColorWorld system and adding more features.

5.1 ISAM (color → id)

The quick color query use ISAM as its data structure. The ISAM data structure allows insertion of overflow pages, which can be used to add more images of certain portion of color to the data structure. Different color will serve as different children of the root node. Different percentage represents the percentage of the color represented by the root node. The overflow pages will be the image id of the images. By using this data structure and limit the number of different color will enable fast query. Different color will similar visual aspect will be merged together to simplify the query for users. For example, user don’t have to distinguish cherry and cinnamon. If the word ‘red’ is inputted, the system will find all the image id(overflow pages) under the red node of the ISAM data structure no matter it has a large portion of cherry color or cinnamon color.

5.2 B+ tree (id → related information)

B+ tree is used to accelerate search for image id to its related information. It shares many things in common with ISAM except that it’s more dynamic, which suites the scenes where often new images will be inserted into the system, so a new entry will be added into the tree to index that image’s ID to its related information.

5.3 Keyword tree (keyword → id+level)

Keyword tree is a dynamic and unbalanced tree structure. It can speed up a keyword query. We can accomplish a keyword query in linear time using keyword tree. Each non-leaf tree node in keyword tree is a character. A sequence of characters from root to a leaf is a keyword for any image IDs stored in the leaf node. Alongwith the image ID is a “level” attribute which represented how much we can trust given the keyword and the image ID. This happens because some keywords are derived from other images and thus are not ground truth. Users can upvote and downvote an result image to improve the system.

5.4 Histogram

A color histogram is a representation of the distribution of colors in an image. It represents the number of pixels of each color (or color range). How to divide the color space is a developer choice. For example, we should design different division for RGB and HSL color space. The Histogram data structure is just a collection of bins. Each color range has its bin. The number contained by the bin is the pixels in that color range.

For example, in RGB color space, we might want to divide by each axis by 32. Thus 32\*32\*32 different colors are regarded as in the same color range and owns one bin. We can use the following algorithm to calculate the histogram for an image.

ALGORITHM: calculate histogram for an image

for (int i = 0;i != image.height;i++) {

for (int j = 0;j != image.width;j++) {

int color = image.getRGB(j,i);

histogram[bin\_for\_color]++;

}

}

After we compute histograms for images, we can compute the distance between two histograms. There are about eight distance measurement. Here we give out two simple measurement.

ALGORITHM: EUCLIDEAN\_DISTANCE(histogram h1, histogram h2)

distance = 0

for each bin in histogram:

distance += (h1[bin]-h2[bin])^2

distance = sqrt(distance)

ALGORITHM: INTERSECTION\_DISTANCE(histogram h1, histogram h2)

distance = 0

minPixels = min(h1.pixels, h2.pixels)

for each bin in histogram:

distance += min(h1[bin], h2[bin])

distance = 1 - distance/minPixels

1. Accomplished Work

We are currently working on two parts of the system: the color querying unit and the image indexing and retrieval unit using histogram method.

The color querying unit can speed up query type “color”, as described in the next section.

We plan to use different method to index and retrieve images. Currently we have completed part of the basic histogram method in RGB color space with several histogram distance calculator. Specifically, we accomplished the work to calculate a basic color histogram for an input image in RGB color space.

Two histograms can be compared using histogram distance calculator to obtain histogram distance. The less distance means two histograms are more similar, and thus the two corresponding images are more likely to be related.

Some additional utilities like sorting the result by distance have been accomplished.

Part of section 8 will demonstrate what we can get from ColorWorld system by far.

Besides, a very fundamental web crawler with limited operations have been implemented. It has been tested and successfully download a couple of files.

1. Example of Queries

ColorWorld system is designed to deal with several queries. The following shows these query types and give some examples about what a user can expect to get. For the part of work we have accomplished, we will give out real data example.

7.1 Color

A specific color (or a combination of colors) can be used as input. A specific data structure is designed for fast retrieval.

Example 1:

INPUT: RED

OUTPUT: Several images that have red as a major color.

Note that “several”, “red” and “major” are all defined further. A default value is designed for how many images will be return. Color will be constrained by design as well. For example, we might use HSL color space to separate each color that user may input. HSL(2, 96%, 68%) will be regarded as RED, as well as HSL(352, 96%, 45%). “major” is defined by a default value as well. If an image has over 40% RED for example, we regard RED as its major color.

Example 2:

INPUT: RGB(100,100,100)

OUTPUT: Several images that have major color near the input color.

The input color will be analyzed to get its nearest color (RED, GREEN for example). Then we search this color as Example 1.

Example 3:

INPUT: PURPLE 20%, GREEN 10%

OUTPUT: Several images that have around 20% purple and around 10% green.

The user input percentage will not be strictly complied. For example, if a user wants an image with 20% brown, we will search for images with 15%-30% brown for him.

7.2 Keyword

When collecting images to initialize ColorWorld system, related information will be collected and stored in the indexing structure. For example, the images collected from University of Washington has a description file which includes ground truth information about an image. These information will be used to build a keyword tree to allow users to search a specific keyword. The (keyword, image) tuple has another attribute “level” which means how much can we rely on this keyword to get the image.

Example 4:

INPUT: CHERRY

OUTPUT: Several images that have keyword with highest “level”.

The keyword tree is searched with argument “cherry”, through which we can get a number of images attached to the keyword “cherry”. However, some images’ keywords are not ground truth (They are derived from other images. For example, one image’s histogram is close to another image’s.). So “level” attribute is used. User can upvote or downvote the result, such refining the keyword search indexing.

7.3 Image

This is the main part of ColorWorld system. ColorWorld can extract more detailed color information (for example, the color distribution) than just plain color (for example, the fact that it has red). These information is obtained by color-based indexing method such as methods mentioned in [2]. When an input image is provided by a user, ColorWorld extract detailed color information from it, and compare it with stored information.

Example 5:

INPUT: an image

OUTPUT: Several images that ColorWorld think related to the input image, sorted by relativity.

We have implemented part of this, as shown below.

Real Image Example:

INPUT: (an image from University of Washington image database)



OUTPUT (the part is still being developed, now it only displays strings):

EUCLIDEAN

14 0.0

24 9860.75656326633

23 13213.389118617524

15 45666.25997385816

12 49508.66301971807

QUADRATIC

14 0.0

24 22762.556847725464

12 27492.421021062313

15 34944.27957920174

4 45706.1475010686

Note: currently, all images are processed in the RGB color space using basic color histogram. We compare the distance by three methods as displayed. For each row that contains two numbers, the first integer represent the id of an image, and the second is the distance compared to the input image. (the distance values of EUCLIDEAN and QUADRATIC are not related)

Parts of IDs and the corresponding images returned in the example are displayed below:



14: the same image as the input.

It shows that if two images are the same, they could be retrieved.



24: this is a truncated image of ID 14 image



12



15

As the result shows, the example shows that although color-based methods are not guaranteed to get best results, in the real test, they could show some good results. Our team will further implement some methods and compare among several histogram distance calculator and color space.

7.4 Natural Language

ColorWorld is designed to cope with simple natural language query.

Example 6:

INPUT: RED CAR

OUTPUT: Several images with keyword “CAR” and has color “RED”

This part needs more study and discussion.

1. Demo Plan

We plan to demonstrate our work by several parts.

First, the algorithms and data structures used in ColorWorld system will be introduced briefly.

Then, the general architecture of our implementation will be give out. The image processing unit will be demonstrated to prove the correctness of color analysis and keyword analysis.

To demonstrate our system’s performance, we will design a couple of tests, showing each our query types, i.e., color, keyword, image and natural language. We will use color and keyword query to show that ColorWorld system can search image in a fast way. We will demonstrate our main query type, by inputting an image, with different algorithms. We will show the result of our comparison of these algorithms.

Reference

[1] Hafner, J., Sawhney, H. S., Equitz, W., Flickner, M., & Niblack, W. (1995). Efficient color histogram indexing for quadratic form distance functions. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, *17*(7), 729-736.

[2] Stricker, M. A., & Orengo, M. (1995). Similarity of color images. *IS&T/SPIE's Symposium on Electronic Imaging: Science & Technology*. International Society for Optics and Photonics.

[3] Pass, G., Zabih, R., & Miller, J. (1997). Comparing images using color coherence vectors. *Proceedings of the fourth ACM international conference on Multimedia*. ACM.

[4] Zhao, Q., & Tao, H. (2005). Object tracking using color correlogram. *Visual Surveillance and Performance Evaluation of Tracking and Surveillance, 2005. 2nd Joint IEEE International Workshop on*. IEEE.