# Experiment Report

Project title: Content Based Image Retrieval

Students' Name: Jian Ru 10389231

Jin Luo 10389230

Xiaodan Lin 10389228

Yifang Li 09388441

Shangjun Wu 10389201

Date due: 1/10/2013

Date handed in: Before due

**Abstract:** The aim of this experiment is devising an approach to retrieve images of the same kind as the queried image from a set of images. This report demonstrates our approach and result. The advantages of our approach are simplicity, intuitiveness, and the hit-rates are relatively good and the disadvantages of our approach are inability of comprehending contents.

# Workload Distribution

**Jian Ru 10389231**

Implementation of the image retrieval algorithm

Implementation of the command-line interface

Writing work of the "Technical Discussion" part of this experiment report

**Jin Luo 10389230**

Implementation of the GUI

Writing work of the "How to Use?" part of this experiment report

**Xiaodan Lin 10389228**

Experiment report writing

**Yifang Li 09388441**

Experiment report writing

**Shangjun Wu 10389201**

Experiment report writing

# Technical Discussion

In order to tweak this project out, I spent a lot of time in reading theses and testing. The interesting thing is that the implementation takes little time, while, most of the time is spent in testing. I tried three approaches: common color histogram, coherent color vector (CCV), and color correlogram. Moreover, I tried two color models: RGB and HSV models. Finally, I settle down at the combination of crude color histogram and CCV because this approach gives the best average hit-rate and its computational burden is lower than the color correlogram approach.

Before discussing the final settlement in detail, I'd like to discuss the other approaches that I tried and the reason why they are inferior to our ultimate choice.

- **Common Color Histogram Only**

When only the color histogram approach is used, to our surprise, the result is not bad (much better than color correlogram which is way more computational intensive). You can see the table below which lists the average accuracy for the ten genres of images and the overall average hit-rate.

| Genre | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Hit-Rate** | 0.5372 | 0.2756 | 0.35861 | 0.47494 | 0.99823 | 0.33355 | 0.46665 | 0.6747 | 0.288 | 0.48896 | 0.4896 |

**Fig. 1. 1.** The hit-rate for each genre and the overall average hit-rate when only color histogram is adopted. Genre n means the collection of the images with name "100(n – 1).jpg" to "100n – 1.jpg".

In this case, RGB model is used and the RGB cube is quantized into 216 sub-cubes, that is, 216 colors. So you can see, when only color histogram is used, you get an overall hit-rate of 0.4896, that is, about 49%.

- **Coherent Color Vector (CCV)**

The drawback of color histogram is that it is a global attribute and it contains only the color information. So it is spontaneous to think whether we can take the spatial information into consideration. CCV is one a kind of improvement of the color histogram. It divides the pixels of the same color into two categories: coherent and incoherent. The definition of coherency is not mounted so the implementer can choose the suitable one pragmatically. Our definition will be explained latter in detail so I am not going to place it here. Instead, I'd like to demonstrate the hit-rates of this approach.

| Genre | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Hit-Rate** | 0.52163 | 0.26674 | 0.3669 | 0.50024 | 0.99907 | 0.32709 | 0.52006 | 0.66433 | 0.26073 | 0.49143 | 0.4918 |

**Fig. 1. 2.** The hit-rate for each genre and the overall hit-rate when only color histogram is adopted.

Again, RGB model is used and the cube is divided into 216 sub-cubes. The table shows that you get a little bit more accurate, 0.4918, when CCV is used. Notice that there is a relatively huge boost of accuracy for the 4-th and the 7-th genres. The meanings, to human, of these two genres are buses and flowers, respectively. Scrutinizing this two set of images, you will

discover that the colors of buses are diverse but their shapes are basically the same and this is more salient for those flower images. So the colors of buses and flowers are actually misleading information here. Since CCV takes some spatial information into account, CCV is more capable to recognize "shape-dominant" images. But nevertheless, this advantage is obtained in the expense of the accuracy "non-shape-dominant" images. For instance, the hit-rate of the first genre is higher when color histogram is used.

- **Color Correlogram**

Initially, I had high expectation of this method because the thesis which introduces this approach says that it out-performs the CCV approach. Nevertheless, the outcome of my implementation is not pleasing to any extent. I did not record the accuracies for all the genres when this approach is used but I still remember the overall hit-rate is about 43% when HSV model is used and about 42% when RGB model is used. I think the problem is my implementation, not the approach, but I if my implementation is correct, then I think it is the following reason that makes this approach be inferior to the two stated above. The first reason the parameters chosen. Since there are two parameters: the number of colors and the distance for evaluation, I may missed the right parameters. Moreover, since the computational intensity is higher than the two above, each change of the parameters will cause a large amount of computation. This prevents me to find a better parameter because that will cost me too much time. The second reason is that color correlogram contains two much spatial information, at least much more than CCV. This may be inhibitive in this experiment because the aim of our search is very general. We not only want to retrieve those images of the same objects which may be rotated, or resized; but also we want to retrieve those images of different objects but of the same meaning (i.e. they are all images of buses). Since the requirement is so general, the program may think two images containing different objects of the same kind (i.e. two images containing a roasted turkey and a roasted lobster, respectively) as images of different genre even though they are all food to human. So, in this case, excessive spatial information may be disadvantageous. For the technical detail of this approach, since its performance is not good for this experiment, I am not going to further explain it. But if you are interested, you can refer to the reference at the end of this report for further information.

- **The Combination of Color Histogram and CCV**

Since color histogram in expressing the global color distribution while CCV is more sensitive to spatial information of an image, we can combine them to get the best of them. In this approach, RGB model is used and the cube is subdivided into 216 sub-cubes uniformly. The mathematical expression of the algorithm of color quantization is

$$I = \left\lfloor \frac{R}{N} \right\rfloor \times n^2 + \left\lfloor \frac{G}{N} \right\rfloor \times n^1 + \left\lfloor \frac{B}{N} \right\rfloor \times n^0$$

where I is the index of the quantized color, n is the number of segments of each edge of the RGB cube after quantization, N is the step width and N = 256 / n, and R, G, and B are the R, G, and B values of the RGB image. The maximum number of colors after quantization is

$$(\left\lfloor \frac{256}{n} \right\rfloor + 1)^3$$

After quantization each pixel in the image is associated with a color index I and the pixels have the same index are thought as having the same color. Quantization is important, before quantization there are $2^{24}$ number of colors. It is impossible, for any reason, to deal with this number of colors and we don't need that many either.

Next the color histogram is computed. Since the computation of color histogram is simple, I will skip its formulation here.

Next we compute the CCV for the one thousand images and store the result in a binary file. Using this pre-computation optimization, we don't need to construct the database each time when the program is started and this will save a huge amount of time.

The definition of coherency is that if a pixel in the image connects to more than 1% pixels in the image in the sense of 4-connectivity, then the pixel is thought to be coherent otherwise it is incoherent. The algorithm for checking coherency can be implemented using recursion intuitively but recursion wastes memory and it is slow due to frequent function calls so we'd simulate the recursive process using a stack. Furthermore, if a pixel is coherent, then all the pixels that are 4-connective to it are also coherent and if it is incoherent, all the pixels that are 4-connective to it are incoherent as well. Thus instead of checking coherency for each pixel, we can check coherency for a set of pixels in just one pass and hence avoiding redundant computation. After all these improvement, the algorithm is very efficient such that the time needed for each query is less than one second. That is, you will see the results with nearly no delay. To make things complete, the CCV is an array of 2-tuples, like

$$\mathrm{CCV} = [< a_0, b_0 >, \ldots, < a_i, b_i >, \ldots, < a_m, b_m >]$$

where $a_i$ is the number of coherent pixels of color i, $b_i$ is the number of incoherent pixels of color i, and m the number of colors used.

To describe the similarity of images, we need some standard such as distances: the $L_1$ distance, the Euclidean distance, the $L_\infty$ distance, etc. And we choose $L_1$ distance because it gives the best performance. The formula for computing distance is

$$D_{12} = .5 \times \mathrm{sum}(|h_1 - h_2|) + .5 \times \mathrm{sum}(|\mathrm{CCV}_1 - \mathrm{CCV}_2|)$$

where $D_{12}$ is the $L_1$ distance of images 1 and 2, $h_1$ is the color histogram of the query image, $h_2$ is the color histogram of an arbitrary image in the image database, $\mathrm{CCV}_1$ is the CCV of the query image and $\mathrm{CCV}_2$ is the CCV of an arbitrary image in the database. The function sum() adds all the entries together and thus its return value is a scalar. So if $D_{xy} < D_{xz}$, then image y is regarded as more similar to image x than image z is.

Finally, the one thousand distances are sorted in ascending order and displayed in order.

# Discussion of Results

The hit-rates for each genre and the overall hit-rate is listed in the following table.

| Genre | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Hit-Rate** | 0.53513 | 0.27363 | 0.36638 | 0.4948 | 0.99879 | 0.33291 | 0.49846 | 0.67478 | 0.27723 | 0.49382 | 0.4946 |

**Fig. 2. 1.** The hit-rate of each genre and the overall hit-rate for the settled approach.

The improvement is very slim globally but as you can see that the accuracies of the 4-th and the 7-th genre are better than only-color-histogram, in the mean while, the accuracies which deteriorate when the approach is shifted from color histogram to CCV degenerate less this time. The overall accuracy is about 50%, nearly 100% for the 5-th genre, but the hit-rates for the 2-nd and the 9-th genres are dismal. The extremely high hit-rate of the 5-th kind is caused

by the almost homogeneous background color of all the images of the 5-th genre, yet, the dismal accuracies of the 2-nd and the 9-th kinds are caused by their similarity in the sense of color (both sky blue). More need to be done to push the accuracy ahead but since time is limited, we will stop here now.

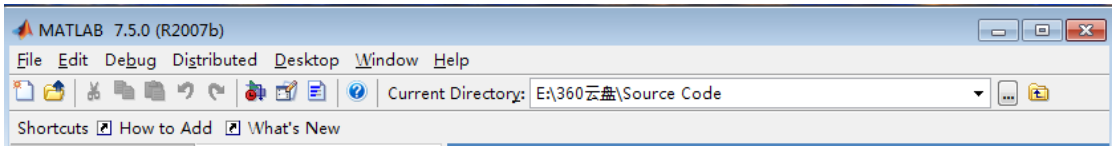
In conclusion, our approach is simple, relatively accurate, and fast but nevertheless, it is not good enough, especially in understanding the contents of images.

# How to Use?

Firstly, put the folder named MART at the root of disk D:

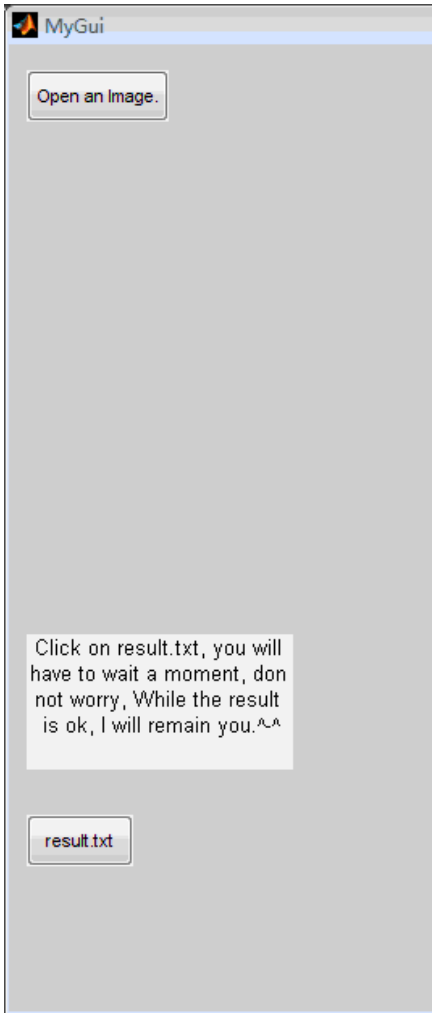

The folder named Source Code can be put anywhere.

Then open your software MATLAB.exe, set its currently path to the path of folder named Source Code (or you can add the path where the source code resides to the search path of MATLAB using "path(path, 'SOURCE_CODE_PATH')").



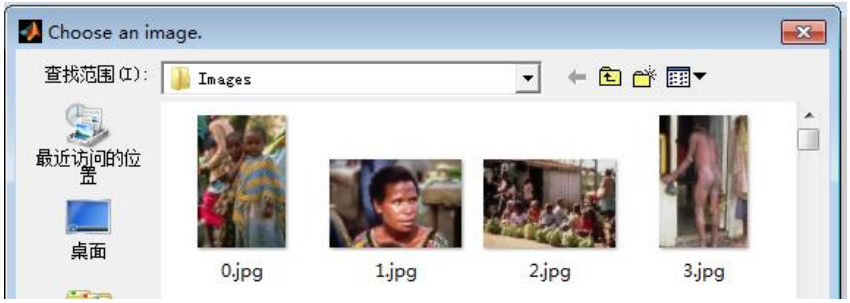

At the command line mode, type in command 'gui', you can enter our graphic user interface.


Our GUI will appear:




Click on button ‘Open an Image’ to choose a testing image.

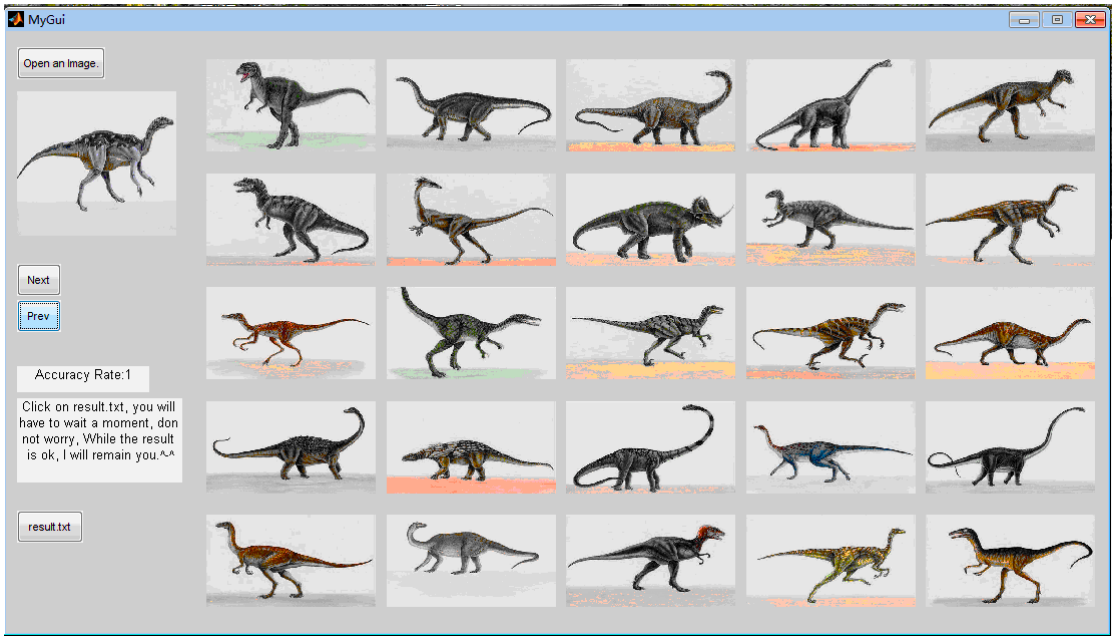
Click on button ‘result.txt’, you will have to wait a moment, don not worry, while the file result.txt is ok, I will remind you.



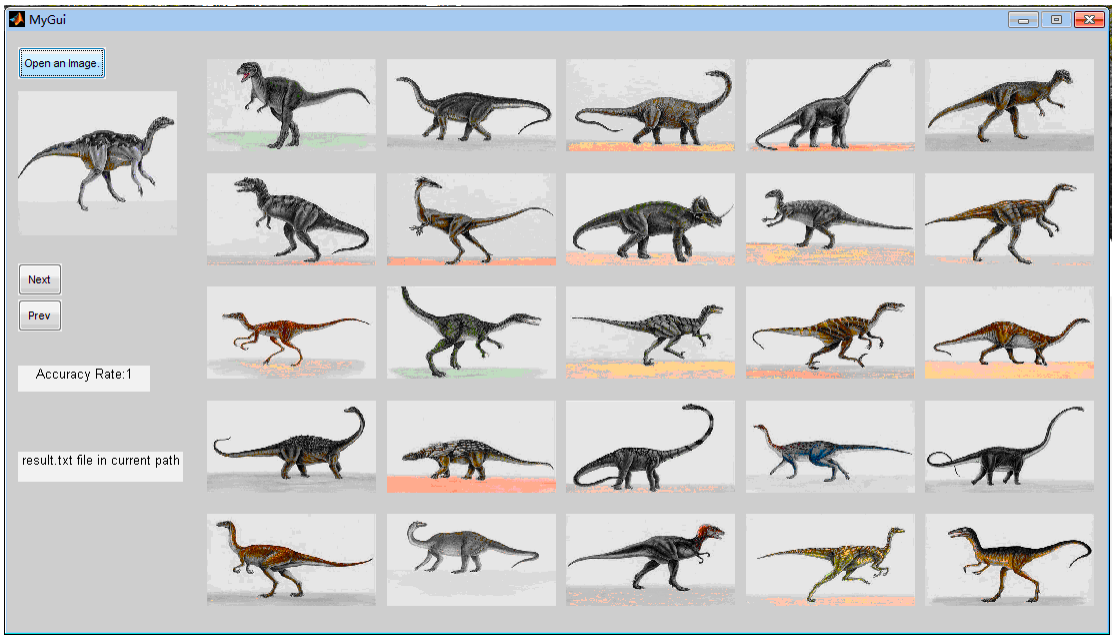

Choose an image to test.


Click button ‘Run’, the program will execute.


After execution, the GUI will show you the result page as well as accuracy rate. Click button ‘Next’, the GUI will show you the next result page.

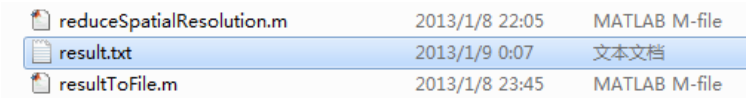



Click button ‘Prev’, the GUI will show you the previous result page that you just view.



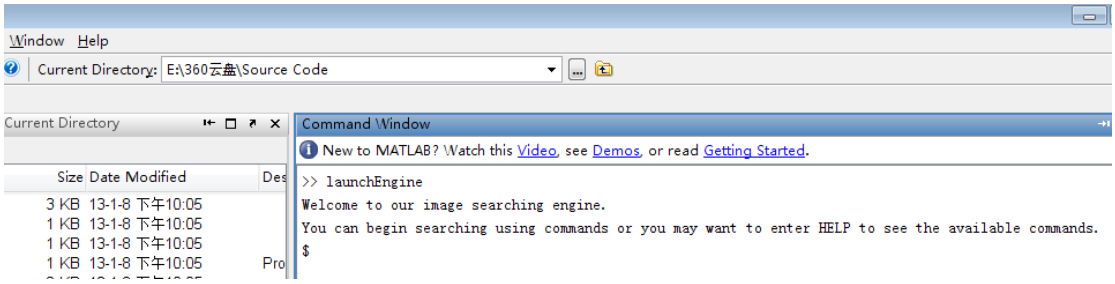

If you click on the button ‘result.txt’, you will have to wait for a moment, don’t worry, I will remind you as above picture. I will show you that “result.txt file in current path”.

Now you can enter the path of folder named Source Code to view the file result.txt




What’s more, you can also use command line interface.

For the command line interface, type ‘launchEngine’, follow the prompt, and then you can play around with the command line interface. The command line interface provides you the ability to do batch test. For example, if you want to test the average hit-rate for the images of the first genre, then the CLI will enable you to do that.


Window Help
Current Directory: E:\360云盘\Source Code
Current Directory
Size Date Modified
3 KB 13-1-8 下午10:05
1 KB 13-1-8 下午10:05
1 KB 13-1-8 下午10:05
1 KB 13-1-8 下午10:05
Command Window
New to MATLAB? Watch this Video, see Demos, or read Getting Started.
>> launchEngine
Welcome to our image searching engine.
You can begin searching using commands or you may want to enter HELP to see the available commands.
$

# Reference

[1] G. Pass and R. Zabih. Histogram refinement for contentbased image retrieval. *IEEE Workshop on Applications of Computer Vision*, pages 96–102, 1996.

[2] J. Huang and S. R. Kumar, M. Mitra, W. Zhu, and R. Zabih. Image indexing using color correlograms. *IEEE Workshop on Applications of Computer Vision*, pages 1–7, 1997.