Simple CBIR using Color Histogram Comparison

Author: Pat Kujawa
Purpose: Assignment 1, CSCI 578

Preliminaries

I developed my solution from scratch using both the .net framework **and** python (**2.7**). For the GUI framework, I chose to use WPF, which is the successor to Windows Forms. I did so because of the power and flexibility of WPF while still providing an easy design process. I chose to use python for the core code because it was easy to prototype with and provided essential libraries for efficient data manipulation (numpy) and image deserialization (SimpleCV).

Running the application

Unfortunately, choosing python and its libraries makes for a somewhat painful deployment experience. You should have access to a zip file containing all of the .net and python code. The python script will fail if SimpleCV is not installed on your machine. Luckily, SimpleCV provides a super-installer for windows that will also install python, numpy, scipy, opencv, and a few other libraries that are often used in python development.

(I tried quite hard to find a way to create a single executable, but it appears that there is no easy way to do that (and also take advantage of the aforementioned libraries).)

So overall, to run the application:

- Install SimpleCV (and with it python 2.7, numpy, and opencv)
- Run Cbir.exe
- Follow instructions on GUI
    - Namely, drag an image onto the GUI, drag a folder too, and select the histogram method you want to test

While I was able to keep the UI responsive (and provide a log of activities), I couldn't get the storage of image information on the python side to work when used in conjunction with the GUI (it works when the script is run via commandline, though).

Screenshots of the app in action

Results for image 33:

Color Histogram CBIR

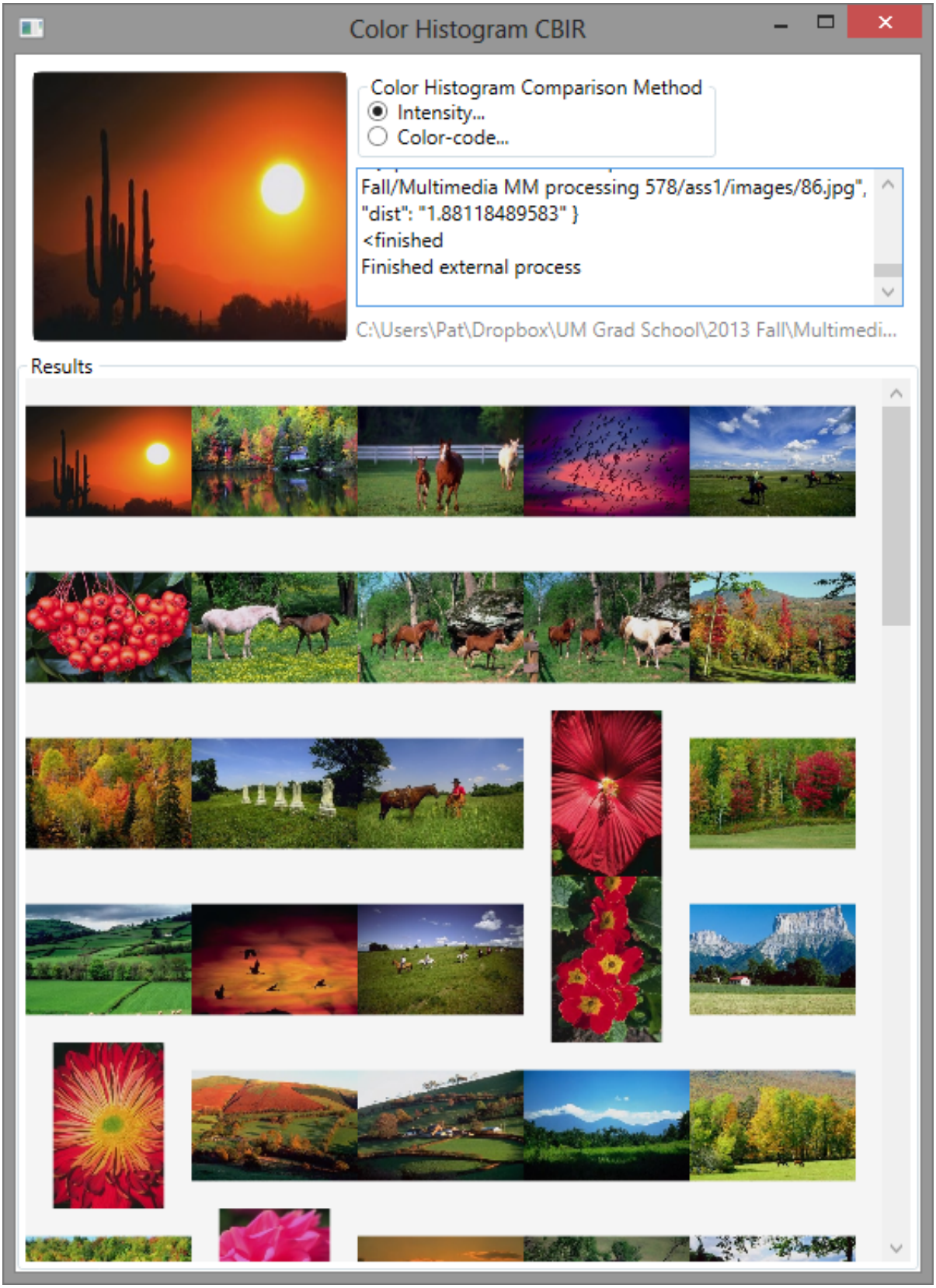
Color Histogram Comparison Method

- ○ Intensity...
- ◉ Color-code...

```
Starting external process
<args: query image file path, db folder path, method,
verbose
<C:\Users\Pat\Dropbox\UM Grad School\2013 Fall
\Multimedia MM processing 578\ass1\33.jpg,C:\Users\Pat
```

C:\Users\Pat\Dropbox\UM Grad School\2013 Fall\Multimedi...

Results



Results for image 93:

Color Histogram CBIR

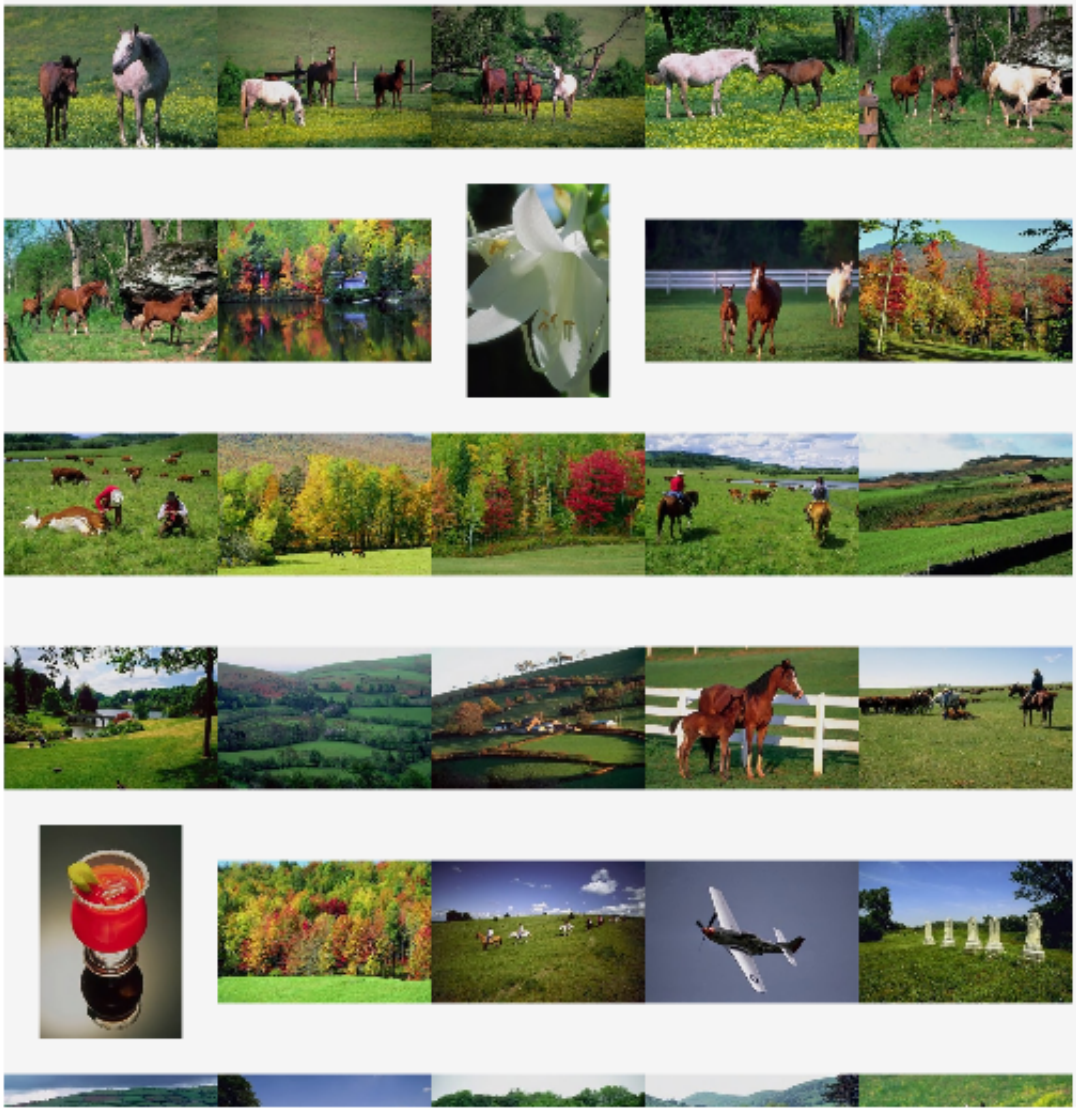
Color Histogram Comparison Method

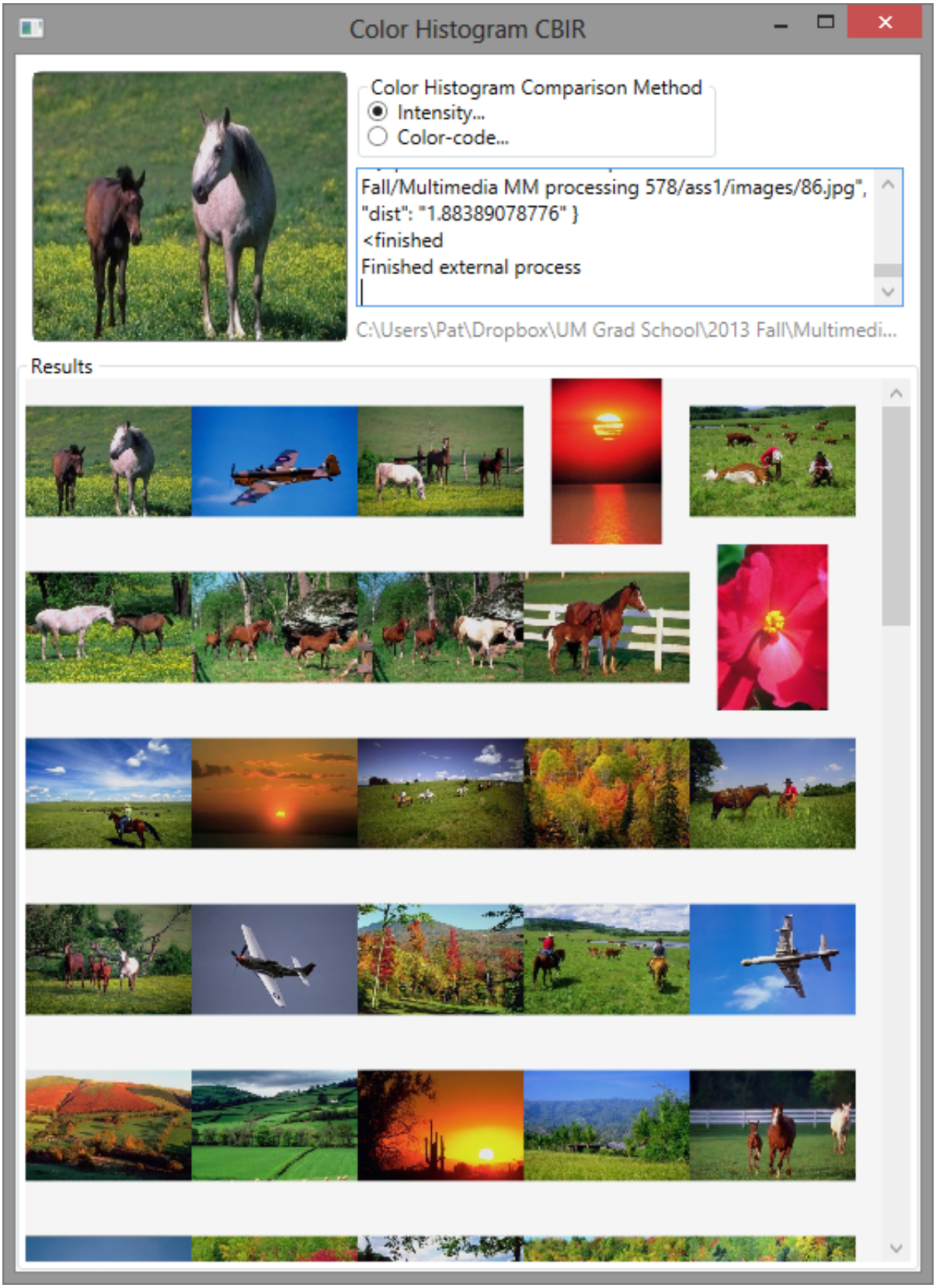
- ○ Intensity...
- ◉ Color-code...

Fall/Multimedia MM processing 578/ass1/images/89.jpg",
"dist": "1.93141682943" }
<finished
Finished external process

C:\Users\Pat\Dropbox\UM Grad School\2013 Fall\Multimedi...

Results





Advantages and limitations of color histogram comparison

Pros:

- Simple and straightforward to implement

Cons:

- Slow to do initial binning
- Results not always clear to users ("Why did this image show up?")

Advances (overcoming limitations)

As an method to simplify users' lives by sorting through their vast trove of images and ordering them by relevance to a query image, color histogram comparison (CHC) falls short. For one thing, it takes far too long to collect the histograms for each image, so the application is useless unless it has already spent significant time cataloging the images before the user runs a query. This slow performance inhibits tweaking of the algorithm (e.g. changing the number of bins or the summary statistic used for binning) because every change requires slow re-computation. (Krishnamachari and Abdel-Mottelab [2] show a novel approach to speed up the process and reduce the memory requirements while still seeing comparable results by reducing the histograms into representative components using the Haar wavelet transformation.)

Another reason CHC falls short is that users often do not see the correlation between their search image and the algorithm's results. If a user queries with an image of a bear, she/he is probably looking for pictures of bears. CHC does not generally perform well in finding localized objects like this, and, furthermore, it provides no simple explanation to the user as to **why** it chose the images it did. Compare that to, say, the system described in [1], wherein specific objects (blobs) are identified and shown to the user in such a way that she/he can interpret (and change) what objects are prioritized in the query.

Now perhaps CHC works well when the user is interested in images with a particular *mood*, as one might expect color palette to contribute greatly to the *feel* of an image, say as a piece of art or landscape imagery. However, I believe that users' desire for this category of results is dwarfed in comparison to the number of queries attempting to find images with the same features/objects as the query image.

Other than changing methods entirely, it would be possible to augment the CHC method with metadata. Shen et al [4] incorporate the text surrounding an image on a website as a *feature* for that image. In [3], Huang et al describe a method to take feedback from the user in order to better understand the priority of result images. Personally, I am much more interested in systems that respond to their users' desires and are able to learn in this way. Had I the time, I would have liked to incorporate some machine learning techniques into my solution. Clearly, though, there are many ways to address the problem of CBIR, and CHC is just one of the simple, decently-effective methods to do so.

References

[1] Carson, C., Belongie, S., Greenspan, H., & Malik, J. (2002). Blobworld: image segmentation using expectation-maximization and its application to image querying. IEEE Transactions on Pattern Analysis

and Machine Intelligence, 24(8), 1026–1038. doi:10.1109/TPAMI.2002.1023800

[2] Krishnamachari, S., & Abdel-Mottaleb, M. (2000). Compact color descriptor for fast image and video segment retrieval. Proceedings of the IS&T/SPIE Storage and Retrieval for Media Databases. Retrieved from http://www.umiacs.umd.edu/~gopal/Publications/spie2000.pdf

[3] Huang, T. S., Ortega, M., & Mehrotra, S. (1998). Relevance feedback: a power tool for interactive content-based image retrieval. IEEE Transactions on Circuits and Systems for Video Technology, 8(5), 644–655. doi:10.1109/76.718510

[4] Shen, H., Zhou, X., & Cui, B. (2005). Indexing text and visual features for WWW images. Web Technologies Research and Development. Retrieved from http://link.springer.com/chapter/10.1007/978-3-540-31849-1_85