Chase Howard
2022-01-15
UWB – Professor Chen
CSS 584 Multimedia Database Systems

Assignment 1: CBIR System

How to run?

1) Locate executable on local machine ‘./app/exe/CBIR.exe’
2) Ensure directory containing executable also contains local image database to analyze
3) Double click executable
4) Wait a while
5) Enjoy!

Note: Screenshots are in the Appendix as well as in the folder ‘./hw1/app/screenshots/’

How to operate?

This interface can be sub-divided into two separate windows. The initial window able to view upon startup of the application will be referred to as the ‘Main Window’ – upon searching for similar images, there will be a secondary window which becomes visible to the user which will be referred to as the ‘Display Window’.

Upon starting the executable, the Main window will be displayed to the user with a collection of image names populating the list box on the left side of the screen. Images names are populated from local storage in a folder entitled ‘images’ within the same directory as the executable launches. Here a user can click to select and preview an image contained within the local database.

Once a user has identified an image of interest, they then have the option to view a more detailed view of the image by selecting the ‘View Details’ button which will open the image in the default viewer within the operating system.

If the user wishes to search for similar images, they have two different search methods to choose from – ‘Color’ or ‘Intensity’. By selecting the Color option, the user will be presented with images in the database most like the previewed image based on the calculated similarity. Intensity will return images with similar intensity histograms generated by flattening each RGB pixel to an intensity value based on the following formula:

$$I = 0.299R + 0.587G + 0.114B$$

Once the user has selected the most appropriate method by which to search, they may select the ‘Search’ button. This will spawn the Display Window which will show the user a preview of the image that was queried along with the search mode that was used. At the bottom of the window will be a grid of twenty images retrieved from the database ranked in order from most similar to least (left to right, then top to bottom). On this page, the user can navigate up or down pages by selecting the ‘Prev’ or ‘Next’ buttons at the bottom of the page.

If desired, a user may update the search parameters (either image or mode) on the Main Window prior to closing the Display Window. The Display window will be updated with the new parameter selected by the user. The user may also close the Display window and reload a new search from the Main Window;

however, if the Main Window is closed, the user will need to re-launch the executable if any additional searches are needed. The user may close out of all windows to shut down the GUI.

Did you use sample code or develop entirely on your own?

I developed the GUI on my own using python and PyQt5. I was certainly inspired by the sample code and retained a similar layout for the search page within my own GUI.

Further Analysis

What are the advantages (what aspects that you like as a system designer and a user)?

The main advantage this system provides as a user is the functionality, specifically with respect to the color-based retrieval method – which appeared to return more user-relevant results when compared to the intensity method of retrieval. The color retrieval method can accurately identify the same and semi-similar colors within images and return results as needed. The user interface, while possibly overly complicated by adding a second window, provides a clear distinction between the user query page and the results page, which can be updated as the user interacts further with the query page. Additionally, this executable loads images from the directory located within the executable to allow for additions to the image database without a need to rebuild – this provides a flexible application to test various types of images with the content-based image retrieval system. One additional useful feature for users is the use of the status bar at the bottom of the page to show the user that their interaction with the UI is being registered as well as displaying the last command sent.

From a Designers perspective, this system follows a Model, View, Controller pattern, making it easier for developers to pick up and understand / maintain the system. Where 'PixInfo.py' serves as the Model, storing the metadata for each of the images after being loaded upon initialization of the executable. 'MainUI.py' serves as the Controller, in which the user can interact with the system to select an appropriate image, select a search mode, and query a search. And finally, 'ResultsDisplay.py' serves as the View, which displays the various results from the image query to the user and can be updated as changes are made to the View while being capable of re-opened / closed as needed. Additionally, this app was written with PyQt5, a popular package for GUI development with extreme similarities between the python and C++ implementation, meaning most developers familiar with GUI's in either python or C++ will be comfortable with making modifications to the application.

What are the limitations and how might you overcome these limitations?

Note: Bibliography located in appendix

- **Limitation**: Slow initialization due to loading of image database and extracting features

Upon execution, the application loads each image and extracts features relating to color and intensity. This results in a slow initialization procedure where each image is located and opened, followed by a M by N scan of pixels for each image to retrieve Intensity and Color information, then the image is closed. The runtime is dependent on the size of the image database and the issue will scale according to the size of the database.

A proposed solution would involve a non-volatile method to store image feature data as binary data, which is significantly less expensive to access. A user can simply reload feature data using a key to the item in a database to avoid reloading, extracting, and storing information for each execution to save time on startup. Each image should have a key which is parsed and checked within the database, if the key doesn't exist, feature data should be extracted from image – if the key does exist, data should be loaded from database with pointer to file. This allows for only feature data to be loaded from the database, saving computing resources by avoiding the superfluous loading of images prior to application start. A specific example includes storage of binary object data with libraries such as 'pickle' in python.

- **Limitation**: Limited search criteria to by which to retrieve images

The user is limited by the features which can be searched 'Color / Intensity'. Different features are commonly used, as well as different methods to extract color and intensity features. Often, users are accustomed to searching images by keywords or by uploading a similar image (reverse image lookup). One possible extension of this system may include other methods by which to search and retrieve images from a database. As well as different metrics by which to compare Color metrics.

The literature search led to a multitude of results of both different features for extraction as well as methods by which to extract color features. Solutions included various extraction methods where the discussion is outside of the scope of this assignment; however, there is a list of them located in the appendix. Cited sources for this include [2],[3] involves the use of MPEG-7 Visual descriptors to describe images. This implies there are numerous methods by which to extract features from images as well as various features by which to describe and relate images. There needs to be a study to identify methods and features relevant for the specific task at hand. In brief, the use of different features to determine similarities generally fall into categories such as { Color, Texture, Shape, Motion, Annotation, Semantic, Sketch, Spatial } [1,2,3,4,5,6]. A more comprehensive list of features come across in the literature search is listed in the appendix. Features can even be combined and weighted to provide tailored results depending on search criteria or user preference. Note, each of the different metrics listed has differing storage implications and extraction time as well – this is something to consider when comparing performance and storage requirements for a specific system.

- **Limitation:** singular distance metric used which may not be applicable for all features of interest.

The user is limited by the distance metric selected, the Manhattan Distance, which evaluates the difference in bin size between two images at each bin location, and sums each of the differences. This may not always be the most accurate distance metric to use, especially if additional search criteria are added to the app in the future. The user may see improved performance with additional distance metrics by which to compare different feature sets.

Alternative methods may include Euclidean Distance or Gaussian Mixture, Quadratic Histogram Distance Measure (DCD-QHDM), Chi-square, Mahalanobis Distance, Histogram Intersection Distance, Merged Palette Histogram Similarity Measure, Minkowski Family distance, and various others [1, 3, 4, 6]. Often distance metrics are suited well for metrics and / or color spaces. Other sources [1, 2] indicate the use of machine learning methods such as SVM / ANN in supervised Learning Classification, or K-means clustering in unsupervised learning. Additionally, deep learning methods have been used such as CNN, Deep Neural Networks, and Boltzmann Machines. These methods often suffer from a high

computational cost and difficult to interpret how decisions are being made. With each application one should investigate to determine appropriate methods by which to determine similarity.

- **Limitation:** Distance values in the RGB color space aren't necessarily indicative of human perception of an image

Distances between colors aren't accurately reflected when using the RGB color space. For example a Brown may be similar to a Darker Red to the human eye; however, their distance metric is may not reflect the same level of similarity.

Proposed solutions in [1,3] describe the use of perceptually uniform color spaces, such as CIE*Luv* where a distance can reflect the perceptual difference between colors. Many articles mention the use of the HSV color space; however, it should be noted there is a discontinuity of Hue at 360 degrees which may cause a small error in distance computations.

- **Limitation:** Concerns around system scalability

The current method of distance calculation requires the computation of distance metrics from one image to all other images, then sorting the distances to determine the shortest distance (e.g. most relevant image). As the image database grows larger, this method is not scalable.

[1, 2, 5] suggests the implementation of relevance feedback techniques to extract an initial retrieval list – the weights for different features would then be updated and fed back into the image dataset for subsequent retrievals. Through an iterative process, the user can identify images of interest without comparing the entire database – and if the user is satisfied with the results, they can end the query process. This allows for a scanning of a subset of the database while still maintaining relevant results, retrieving additional results if needed.

Bibliography

[1] Ibtihaal M. Hameed, Sadiq H. Abdulhussain & Basheera M. Mahmmod | (2021)
Content-based image retrieval: A review of recent trends, Cogent Engineering, 8:1, 1927469, DOI: 10.1080/23311916.2021.1927469

[2] W. K. Ma, "Content Based Image Retrieval Using MPEG-7 Dominant Color Descriptor", *Master Degree Thesis City University of Hong* Kong, 2004

[3] Y. Wu, J. Zhang, W. Cui and H. Shao, "Image Retrieval Based on MPEG-7 Dominant Color Descriptor," in *2008 9th International Conference for Young Computer Scientists*, Hunan, 2008 pp. 753-757.
doi: 10.1109/ICYCS.2008.89

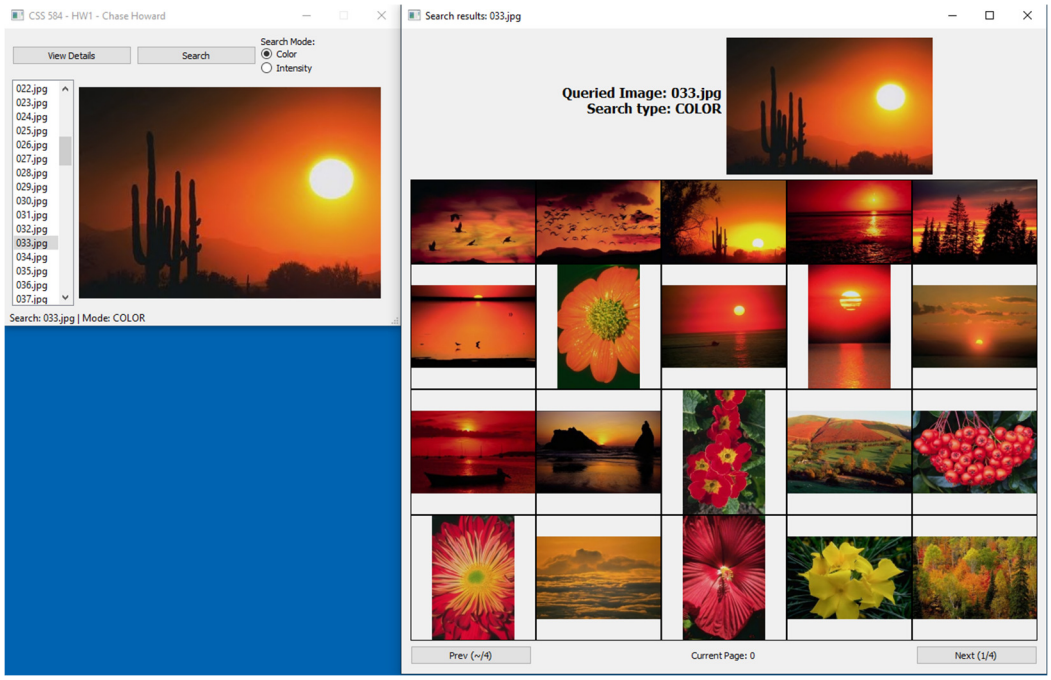
[4] I. Rejeb, S. Ouni and E. Zagrouba, "Image Retrieval Using Spatial Dominant Color Descriptor," in *2017 IEEE/ACS 14th International Conference on Computer Systems and Applications (AICCSA)*, Hammamet, Tunisia, 2017 pp. 788-795.
doi: 10.1109/AICCSA.2017.127

[5] Zhao, T., Tang, L.H., Ip, H.H.S. *et al.* Visual Keyword Image Retrieval Based on Synergetic Neural Network for Web-Based Image Search. *Real-Time Systems* **21,** 127–142 (2001). https://doi.org/10.1023/A:1011147421401

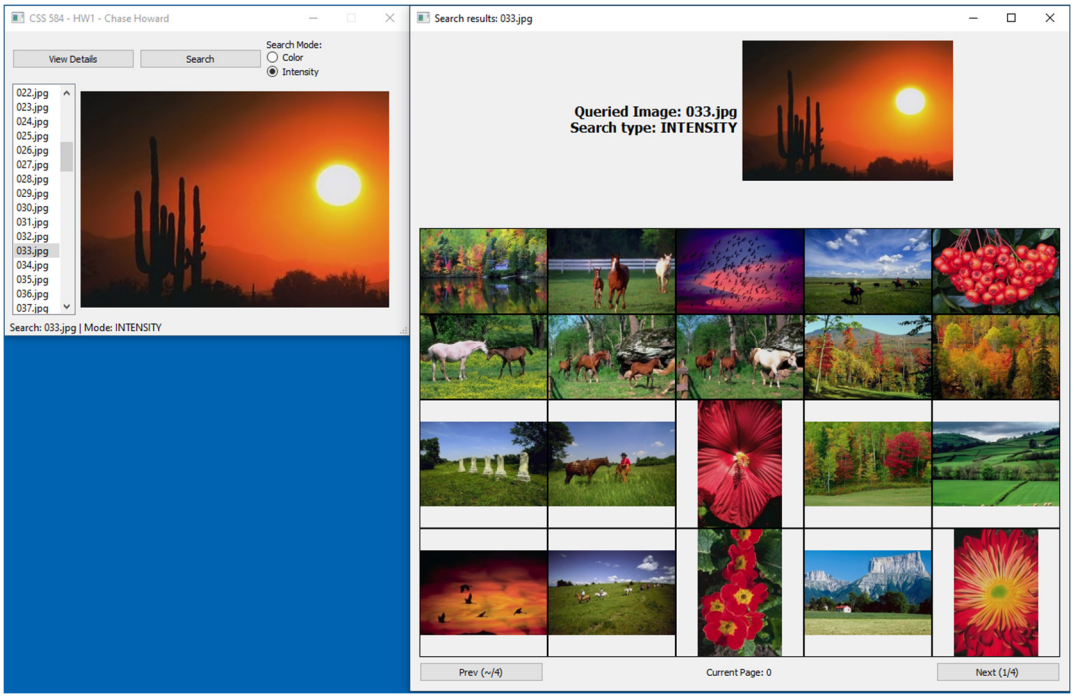
[6] Feng Jing, Mingjing Li, Hong-Jiang Zhang and Bo Zhang, "A unified framework for image retrieval using keyword and visual features," in *IEEE Transactions on Image Processing*, vol. 14, no. 7, pp. 979-989, July 2005, doi: 10.1109/TIP.2005.847289.

Appendix A: Screenshots

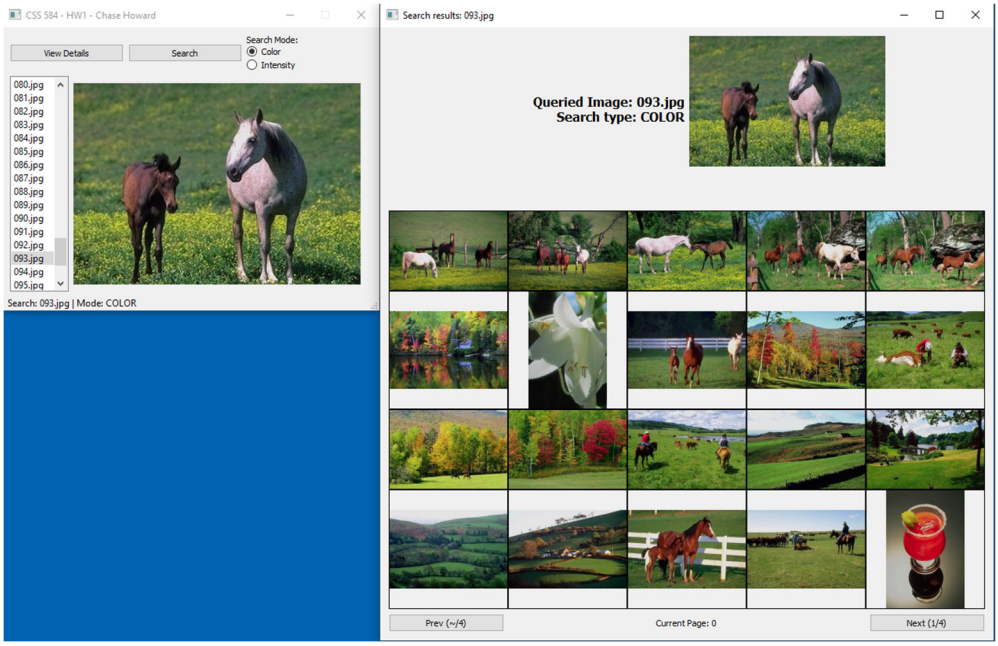
- Image: 33.jpg – Color Query



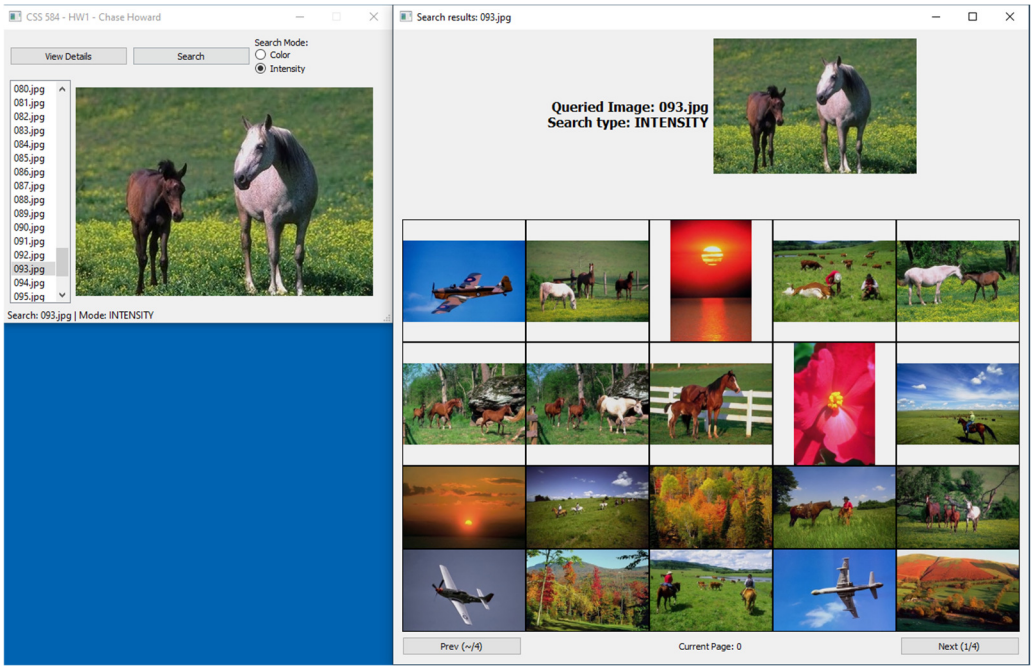
- Image: 33.jpg – Intensity Query



- Image: 93.jpg – Color Query



- Image: 93.jpg – Intensity Query



Appendix B: Alternate Feature List

1) Color Features: Analysis on comparative Histograms of images in the same color space
    - Histogram Intersection [Swain and Ballard]
    - Histogram: Quadratic Histogram Distance Measure (QHDM)
    - Color Spaces Descriptor (CSD)
    - Dominant Color Descriptor (DCD)
    - Scalable Color Descriptor (SCP)
    - Color Layout Descriptor (CLD)
    - Color Structure Descriptor)
    - Group-of-Frames / Group-of-Pictures (GoF / GoP)
2) Texture Features: common visual patterns / structural arrangements of surfaces and their relationship to the environment.
    - Co-occurance matrix [Haralick]
    - Tamura Texture [Tamura et al.]
    - Wavelets [Ma and Manjunath]
    - Homogenous Texture Descriptor (HTD)
    - Edge Histogram Descriptor (EHD)
3) Shape Features: recognizing and grouping objects by their shape
    - Finite Element Method
    - Contour Based methods
    - 3-D Shape Matching
    - Contour Shape Descriptor
    - Region Shape Descriptor
    - 3-D Shape Descriptor
4) Motion Descriptors: description of motions in multimedia content for features of video segments or for features of a moving region.
    - Motion Activity Descriptor
    - Camera Motion Descriptor
    - Motion Trajectory Descriptor
    - Parametric Motion Descriptor
5) Visual Keyword Retrieval
    a. Synergetic Neural Network
    b. Trademark Image Retreival
    c. Affine-invariant feature