IMPLEMENTATION OF CONTENT-BASED IMAGE RETRIEVAL USING THE CFSD ALGORITHM

Nagaraja. G. S¹, Samir Sheriff², Raunaq Kumar³

¹ Associate Professor, ²,³ VII Semester B.E., Dept of Computer Sci. Eng., R.V. College of Engineering, Bangalore, India
nagarajags@rvce.edu.in, samiriff@gmail.com, devilronny@gmail.com

Abstract

Content-based image retrieval (CBIR) is the application of multimedia-processing techniques to the image retrieval problem, that is, the problem of searching for digital images in large databases. There is a growing interest in CBIR because of the limitations inherent in metadata-based systems. The most common method for comparing two images in CBIR is using an image distance measure based on color, texture, shape and others. In this paper, we describe an object-oriented graphical implementation of a system, backed by a MySQL database, that computes distance measures using the CFSD algorithm, based on color similarity.

1. INTRODUCTION

"Content-based" means that the search will analyze the actual contents of the image rather than the metadata such as keywords, tags, and/or descriptions associated with the image. The term 'content' in this context might refer to colors, shapes, textures, or any other information that can be derived from the image itself.

The system we have implemented uses color-based feature extraction. Computing distance measures based on color similarity is achieved by computing a color histogram for each image that identifies the proportion of pixels within an image holding specific values. Examining images based on the colors they contain is one of the most widely used techniques because it does not depend on image size or orientation.

Our program allows the user to choose between two algorithms

_ Color Histogram Method: In image processing and photography, a color histogram is a representation of the distribution of colors in an image. For digital images, a color histogram represents the number of pixels that have colors in each of a fixed list of color ranges, that span the image’s color space, the set of all possible colors.

_ Color Frequency Sequence Difference Method: The high dimensionality of feature vectors of the Color Histogram method results in high computation cost and space cost. CFSD expresses the color image in terms of numerical values for each color channel, greatly improving comparison time and computation costs.

2. THE COLOR HISTOGRAM METHOD

1) Color Quantization: For every image in the database, colors in the RGB model are quantized, to make later computations easier. Color quantization reduces the number of distinct colors used in an image. In our application, the user can select the quantization bucket size for red, green and blue separately.

2) Compute Histogram: For every quantized image: Calculate a color histogram, which is a frequency distribution of quantized RGB values of each pixel of an image. In our application, we have used hashmaps to store histogram values as <string, double > pairs.
3) Query Image Feature Extraction: Compute the quantized color-values and the resulting histogram of the query image.

4) Compute Distance Measure: Compare the histogram of the query image with the histograms of every image of the retrieval system. Let $h$ and $g$ represent two color histograms. If $n = 2B - 1$, where $B$ is the maximum number of bits used for representing quantized color components $a$, $b$, and $c$, then the Euclidean distance between the color histograms $h$ and $g$ can be computed as:

$$ X_n \left( a=0 \right) \not{= \forall} \left( a; b; c \right) \not{= \forall} \left( a; b; c \right) \cdot \left( \left( a; b; c \right) \not{= \forall} \left( a; b; c \right) \right) \cdot \left( 1 \right) $$

In this distance formula, there is only comparison between the identical bins in the respective histograms. Two different bins may represent perceptually similar colors but are not compared crosswise. All bins contribute equally to the distance.

5) Find Similar Images: The similarity between two images is inversely proportional to the (color histogram) distance between them.

3. THE COLOR FREQUENCY SEQUENCE DIFFERENCE

METHOD

1) Color Space Conversion: Translate the representation of all colors in each image from the RGB space to the HSV space. HSV color space has two distinct characteristics: one is that lighness component is independent of color information of images; the other is that hue and saturation component is correlated with manner of human visual perception.

2) Color Quantization: For every image in the database, colors in the HSV model are quantized, to make later computations easier. Color quantization reduces the number of distinct colors used in an image. In our application, the user can select the quantization bucket size for each color component separately.

3) Compute Histogram (Sequence): For every quantized image, a color histogram is calculated, which is a frequency distribution of quantized HSV values of each pixel of an image. In our application, we have used treemaps to store histogram values as <string, double> pairs. The values are ordered alphabetically by their HSV strings. For instance, (H, S, V) string (0, 0, 0) is followed by (0, 0, 1).

4) Compute Sorted Histogram (Frequency): Take the color histogram of each image and sort the keys in descending order of frequency values. Each image now has 2 histograms:

- One sorted by color sequence Call this $P$
- One sorted by color frequency Call this $Q$

However, in our application, we maintain only the sorted histogram as the other histogram can be derived implicitly.

5) Query Image Feature Extraction: Convert and quantize the color space of the query image. Then, compute the sequence histogram as well as the frequency histogram of the query image.

6) Compute SFD: In this algorithm, SFD is an alias for distance measure, which is calculated as follows, for each component of the HSV model:

For every color combination $x$ in Histogram $P$,

$$ \text{sfd} = \sum_{i=1}^{N} w(i) h(i) \left( 2 \right) $$

where

- $i$ is the index of color combination $x$ in histogram $Q$ and $i$ is the index of $x$ in histogram $P$
- $w(i) = 1$
- $j(i) - i + 1$. $w$ is a function of the difference between the indices $i$ and $i$.
- $h$ is a function of the frequency of color combination $x$ in the image.

7) Compute Distance with SFD (Difference): Let $h$ and $g$ represent two images which have SFD values for each component of the HSV model. The SFD distance between $h$ and $g$ can be computed as:

$$ D_{sfd}(a; b) = (sfdH(a; b) + sfdS(a; b) + sfdV (a; b)) = 3 $$

where

- $sfdT (a; b)$ is the corresponding SFD value of component $T$ of the HSV model representing image $a$
- $T_fH; S; V$ are the corresponding SFD values of the component $T$ of the HSV model of the two images $a$ and $b$.

$D_{sfd}(a; b) = \left( SFD(a; b) + SFD(b; a) \right)$ which is the distance between the corresponding SFD values of the component $T$ of the HSV model of the two images $a$ and $b$. 

$T_fH; S; V$ are the corresponding SFD value of component $T$ of the HSV model representing image $a$. 


8) Find Similar Images: Compare the SFDs of each HSV component of the query image with the corresponding SFDs of every image of the retrieval system. The similarity between two images is inversely proportional to the SFD distance between them.

4. OBJECT-ORIENTED DESIGN

The Java Programming Language has been used to create our Content-Based Image Retrieval System, with MySQL for the back-end. For displaying graphical objects, we have used the ACM package (created by the Java Task Force (JTF)), which provides a set of classes that support the creation of simple, object-oriented graphical displays. The object-oriented design of our program below:

A. Class Design

We have segregated the code into two packages for easy management:

_retriever - This package contains classes that provide methods to read a database of images, store status information. The implementation of the algorithms mentioned in the previous sections can be found in this package.

_gui - As the name suggests, this package contains all the code necessary for the graphical user interface of our retrieval system. It makes use of methods provided by the ACM library as well as the Swing library.

Now that you know the organization of code, let us dive deeper into the actual object-oriented design. The retriever package consists of the following classes:

1) ColorFrequency - This class is used while storing color histograms in hashmaps/treemaps, to store <value, frequency> tuples for different color combinations. It implements the Comparable Interface so that the Collections.sort() method provided by the java.util package to sort histograms in decreasing order of frequency, as required by the CFSD algorithm.

2) ImageData - This class keeps track of the status of a single image of the database. It stores information and provides methods to:

_ store the two-dimensional representation of an image using the GImage class of the ACM package

_ store the path of this image

_ quantize every color model component of the given pixel of this image

_ create color histograms with ColorFrequency mappings for individual components as well as a combination of all components of the color space of this image.

_ calculate SFD values for every component of the color space of this image.

_ establish or disable a connection with a database and retrieve status information if feature values have already been calculated for this image during a previous run of the program. This is done to prevent unnecessary computation of SFD values whenever the program is run repeatedly.

3) ImageCollection - This class keeps track of the Image-Data of a number of images in the database. In essence, an ImageCollection Object is a collection of ImageData objects. It provides methods to:

_ initialize an ImageData collection with images from a specified directory, which is traversed to a folder depth of 1

_ find images that are most similar to a query image, computed based on the Color Histogram method and Euclidean distance, top are similar since their sfd values are quite close. This similarity cannot be detected by the normal color histogram method.

Fig.2. SFD Calculation for one component of a given image

Fig.3. Finding Similar Images based on SFD Distance. The two images on top are similar since their sfd values are quite close. This similarity cannot be detected by the normal color histogram method.
formula. A list of images, sorted based on decreasing order of similarity is returned.
- find images that are most similar to a query image, computed based on the CFSD method and SFD distance formula. A list of images, sorted based on decreasing order of similarity is returned.

4) UserVariables - This class only maintains the values of all the variables that the user can set in the GUI window. The ImageData and ImageCollection objects read these values during computation. It provides methods to get and set:
- the number of red, green and blue bins for color quantization
- the number of similar images to be returned by ImageCollection
- a boolean variable denoting whether the Color Histogram Algorithm or CFSD algorithm should be used.

The gui package consists of only one class, namely:

1) MainGraphics - This class uses the power of the widgets provided by the ACM graphics package and the Javaswing library to create a user-friendly interface for our Image Retrieval System. It creates a window consisting of a white area where the query image selected by the user, and the three most similar images are displayed, in their normal forms as well as their quantized forms.

The northern portion of the window consists of:
- Two Buttons - One to open a query image and another to start the execution of the selected algorithm, which is the Color Histogram Algorithm, by default.
- Three sliders - Each of which can be varied from 1 to 60. The sliders are used to choose the bin sizes of each component of the RGB (HSV) model for the Color Histogram (CFSD) algorithm.
- One checkbox - If checked, then the CFSD algorithm is run to find similar images. Otherwise, the Color Histogram method is used.

For each of these widgets, the corresponding variables of the UserVariable class are set.

B. Schema Design

Since comparisons in the CFSD algorithm are done using a few SFD values instead of comparing entire histograms as in the normal color histogram method, the CFSD algorithm can be made to operate more efficiently by calculating the SFD values of an image only once (when it is encountered for the very first time) and storing them in a database which can be accessed for all future references. Please note that whenever we say red, green or blue in the CFSD algorithm, we actually refer to the H, S or V values of the image respectively. This was done for compatibility.

The database that we constructed for our system consists of two tables:

1) imagedata - It consists of the following attributes:
- Image ID - which is the primary key of this table.
- It is an auto-incremented variable
- Image Path - A string with a variable number of characters that stores the path of the given image, relative to the directory of the application.
- redSFD - The SFD calculated using the red (H) histogram of the image.
- greenSFD - The SFD calculated using the green (S) histogram of the image.
- blueSFD - The SFD calculated using the blue (V) histogram of the image.

2) quantizedvalues - It consists of the following attributes:
- Image ID - which is the foreign key of this table that references the primary key of the imagedata table.
- qRed - The value of the bin used to quantize red (H).
- qGreen - The value of the bin used to quantize green (S).
- qBlue - The value of the bin used to quantize blue (V).

C. Summary and Results

In order to test out our image retrieval system, we have provided a sample set of images, organized in seven folders with five images each. Feel free to add more folders if necessary, but ensure that each folder contains exactly five images at any given instant of time. Using our sample database, it was found that this application could retrieve at least three similar images to a supplied query image, chosen from the database itself. The speed of retrieval varied depending on the algorithm selected, with the CFSD algorithm running faster than the naive Color Histogram algorithm.

5. FUNCTIONALITY

First ensure that all the above requirements are met. Then, to start the application, go to the installation folder and click on a file named CBIR.jar. In the new window that pops up, select the appropriate query image using the open button.

Afer selecting the appropriate number of bins for quantization of each component of the color space, and checking/unchcking the CFSD box, click on the Retrieve Similar Images button to retrieve the three most similar images that will be displayed in the lower portion of the window. Figures 4, 5 and 6 show the state of the window at different instances of time.

Fig4. Application after the query image has been selected by the use
Fig 5 Application after images similar to the query image have been retrieved using the CFSD algorithm

However, it was found that the accuracy diminished when more than three similar images were sought. This leads us to conclude that we should also consider other features such as shape, etc., while comparing images in a content-based retrieval system.

ACKNOWLEDGMENTS

The authors wish to express their gratitude to Dr. Nagaraja G. S., who offered invaluable assistance, support and guidance throughout the development of this tool.

REFERENCES


CONCLUSIONS

This paper discussed briefly the software implementation of a Content-Based Image Retrieval System using the CFSD algorithm as well as a naive Color Histogram method. This software tool was implemented in Java, relying heavily on Object-Oriented concepts without which code management wouldn’t have been easy at all. We also had the opportunity to find out that the naive Color Histogram algorithm is not as fast as the CFSD algorithm. In fact, the CFSD algorithm, coupled with a database, forms a highly efficient system which retrieves at most three similar images quite accurately.