Content Based Image Retrieval Project

CPSC 4660

Logan Martin

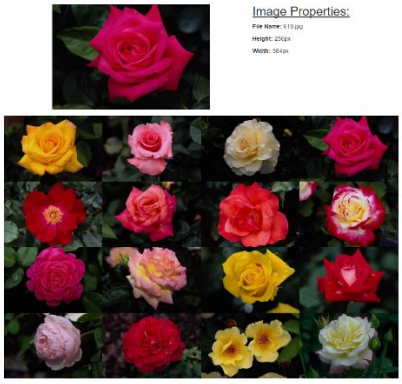
For my project, I implemented a content based image retrieval solution that was proposed in “An integrated approach to Content Based Image Retrieval” (Choudhary, Raina, Chaudhary, Chauhan, Goudar; 2014). The proposed solution combines two commonly used CBIR methods, Local Binary Patterns(LBP’s) and Color Moments, in order to get more accurate results than one of these solutions would get on their own. To implement searching by Color Moments, what we first have to do is convert our image if it’s RGB, to HSV (hue, saturation, value). We use HSV because it works much more similarly to human color perception as opposed to RGB (Huang, Chan, NG, Yeung; 2010). Once converted, we calculate the Mean, Standard Deviation, and Skewness of the image for each color channel (h, s and v). The result will be nine values, which makes up our Color Moment feature vector for an image. Local Binary Patterns are used for texture and feature analysis in images, and was first introduced by Ojala, Pietikinen, and Harwood. It’s most commonly used in facial recognition software as it can be used for feature recognition on a spatial basis (i.e. it knows where the features are in the image, not just that they’re there). To calculate the LBPs of an image, we first convert the image to greyscale, as we only need the grey levels. Next we section off the image into sub-images; this is used for spatial awareness. Now, for each pixel in each sub-image, we calculate its “central pixel” value. This is done by getting the grey level for said pixel, as well as for all its neighboring pixels. If a neighboring pixel’s grey level is greater than that of the central pixel, it’s given a weight, based on a power of two, and added to our total value. The total value can end up being between 0 and 255. We add each pixel to a histogram for each sub-image, and then concatenate each sub-image’s histogram into one. This makes up our LBP feature vector for an image. For the proposed CBIR solution, we then combine the two feature vectors for an image to make up our final feature vector. Now to retrieve images that are similar to our initial or queried image, we simply compare the final feature vectors by finding the Euclidean distance of each element in the feature vectors. If an image is within a certain distance threshold, it is determined to be a match.

For my implementation of this CBIR solution, I decided to develop it mainly using JavaScript and HTML5. The main reason for this was simplicity and portability, not having to

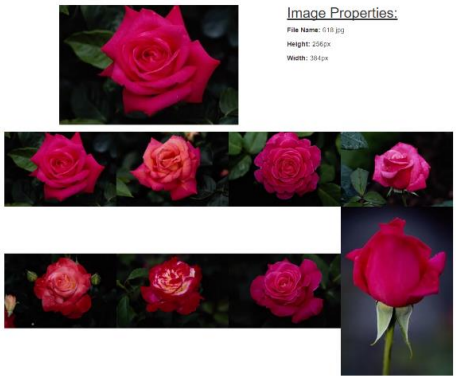
compile every time I made a change, or manage other libraries that I've included, and to be able to quickly make the GUI. I used HTML5's Canvas elements in order to load images and have easy access to the per-pixel data that I needed for my algorithms, and was easily changeable once we calculated the greyscale values for example. For the images that I used for my comparisons, I used the Wang Database (Wang, Li, Wiederhold; 2001). It's a database of 1000 image that are roughly broken up into categories (e.g. pictures of beaches, flowers, animals, etc.) which makes them ideal for CBIR algorithms, because you already have a rough idea of what images you'd want to see returned to you when you run your algorithm. These images are just stored in a folder with the rest of the source files, instead of stored in a more elaborate database format. To calculate the LBP feature vector, I essentially did exactly as described by Choudhary et al[1]. I went with a 3x3 grid of each image for the spatial information, as it seemed good enough for this project. If we increased it to a 4x4 grid or bigger, theoretically we would be able to make even more precise comparisons. I got the values of each pixel's neighbor, calculated the central pixel's value, added it to that sub-image's histogram, and then concatenated all 9 of the histograms together at the end. For the Color Moment feature vector, I converted each RGB pixel to HSV, then used this data to calculate the mean, standard deviation, and skewness of each color channel, to make up my 9-value feature vector. The proposing paper couldn't seem to make its mind up whether the second value was just the variance or the standard deviation, but other sources[3,4] indicated that we wanted the latter. Now to search for images based on other images, I first calculated the feature vector for the queried image, then looped through the first 1000 images in the "imageDB" folder, calculating and comparing the feature vector for each of these images to the queried image. If it fell within the difference threshold, then I added it to an array of matching images, to be displayed to the user. I played around with the threshold value until it seemed right, as there didn't seem to be an indication of what it should be by Choudhary et al[1]. They also said to increase the contrast of the image before we did feature extraction, but I opted not to, as it seemed to skew the results negatively when I did. For the proposed solution, I simply ran both comparisons, and then found which matches were in both results. I did loosen the thresholds on the LBP and Color

Moment results for the proposed solution, as they have the added information from each other to further filter out differing images.

For evaluation, I ended up comparing the proposed solution to simply searching images based on the LBP feature vector, or Color Moment feature vector individually. Here's an example of how the results between them differed (Queried image on top).



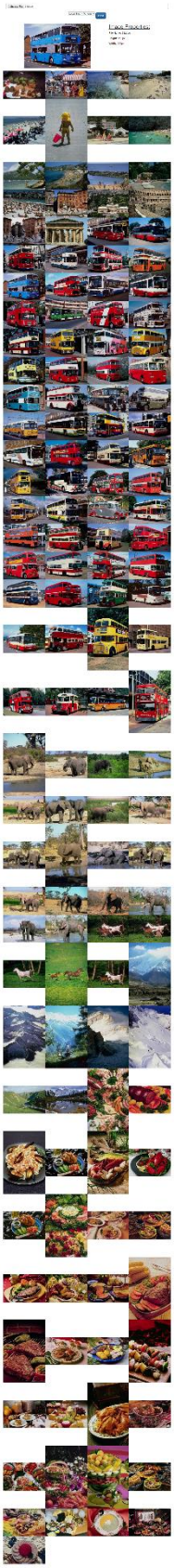
Local Binary Pattern



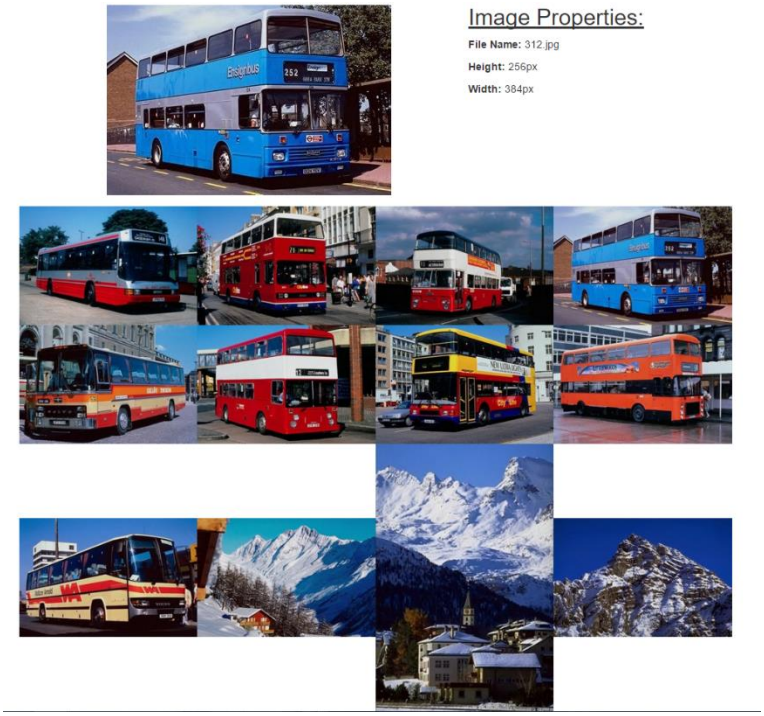
Color Moment

Proposed Solution(combination)

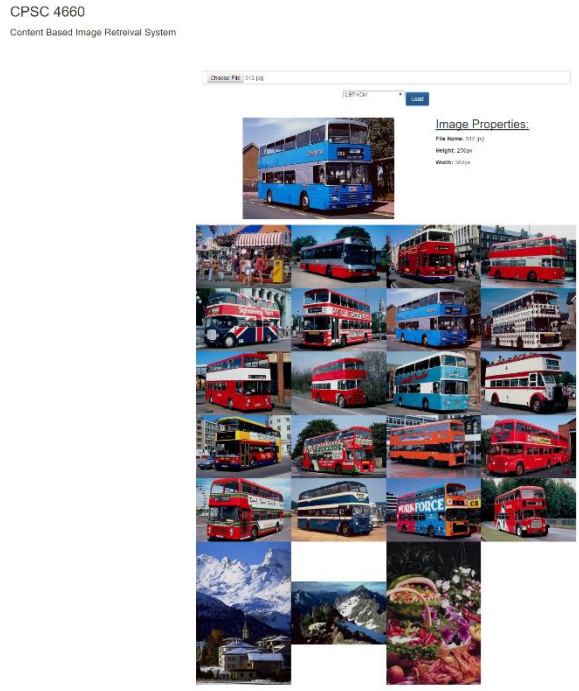
As you can see from a first glance, the LBP search has images with similar content, but the color could be anything, the color moment search worked quite well on this image, as all the matching images are also of roses, and our proposed solution is even more strict, showing only roses that are displayed similarly. Though in this case it seems like they all do a fairly good job. Here's an example where the proposed solution is clearly better.



Local Binary Pattern



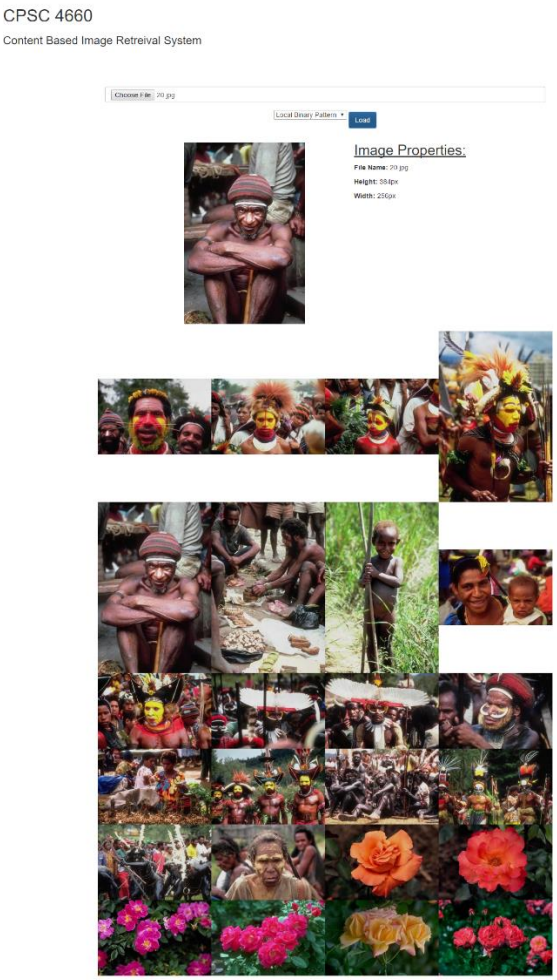
Color Moment



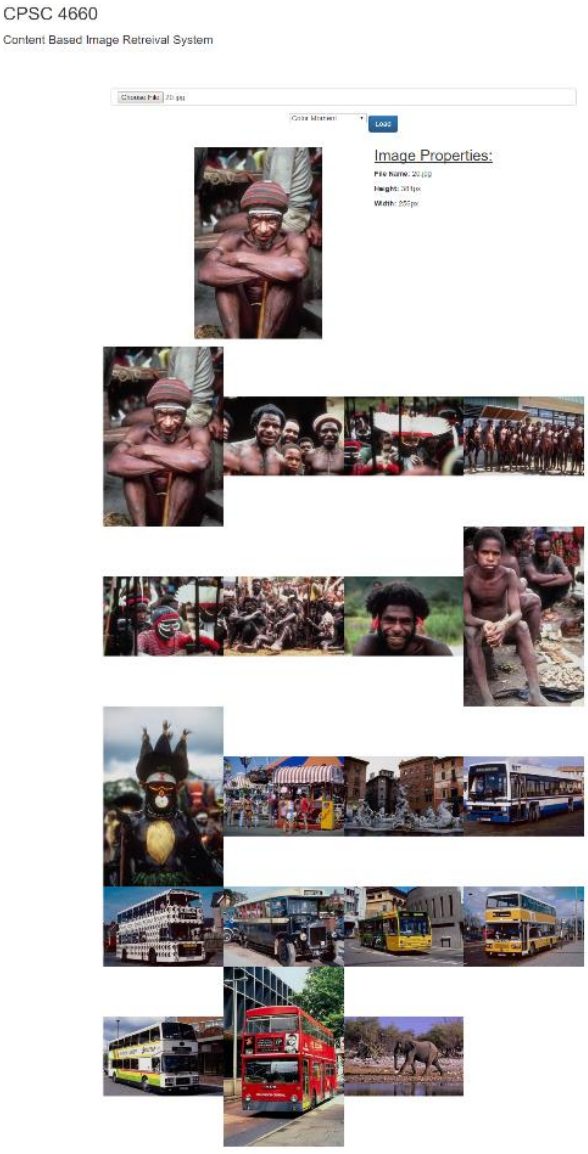
Proposed Solution

As you can see, Local Binary Patterns in this case proved fairly ineffective compared to the other two. Even though it got a bunch of busses back, it also got a lot of other unrelated matches. Color Moment returned a good number of busses and a few mountain ranges, and our proposed solution returned mainly busses, with a couple of other mismatched images. This shows that none of these solutions is perfect, though they can likely get closer with some tweaking of the thresholds.

Here's one more example



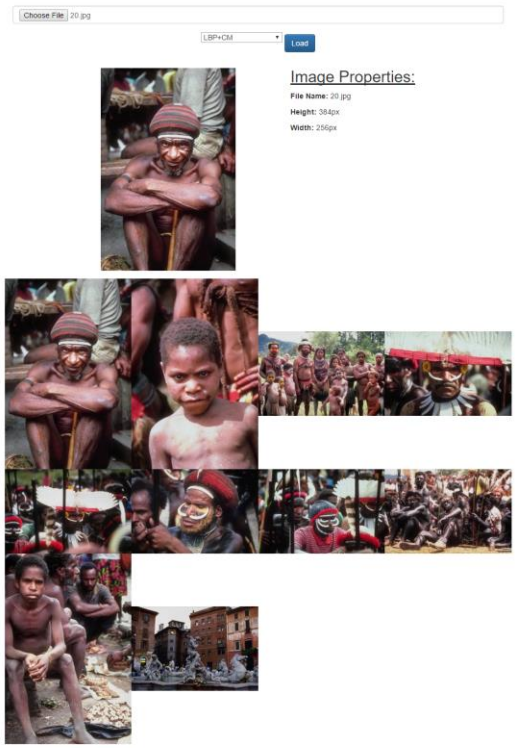
Local Binary Pattern



Color Moment

CPSC 4660

Content Based Image Retreival System



Proposed Solution

Unfortunately, I don't have any comparable data for the accuracy each search method, aside from just looking at the results, as the distance values for Local Binary Patterns and Color Moments are so different that they're incomparable. What I could compare though is the time it takes to run each algorithm. The average runtime for each algorithm, based on four tests each, was as follows:

	Local Binary Pattern	Color Moment	Proposed Solution
Time per image	25.25ms	60.25ms	78ms
Total Time	25.25s	60.25s	78s

One of the downfalls to the way I implemented this system is that we're required to wait for each image to fully load as an HTML5 canvas before we can start processing it. This means that there could be drastically different runtimes based on how long it takes to load each image. For example, compared to running this program locally, running it from a web server could take two to five times longer.

It seems to me that with the proper threshold tweaking that the proposed solution is much better than either of the other solutions on their own, though the individual solutions can still do a pretty good job. The proposed solution does have the problem of basically having the runtime of the other two combined. I believe a solution to this problem may be to store the feature vectors of an image in the database when uploaded, then you would only have to calculate the one feature vector of the queried image before you can compare them.

References

1. Roshi Choudhary, Nikita Raina, Neeshu Chaudhary, Rashmi Chauhan, R H Goudar, “An integrated approach to Content Based Image Retrieval”, Advances in Computing, Communications and Informatics (ICACCI, 2014 International Conference on 24-27 Sept. 2014), p. 2404-2410.

2. Zhi-chun Huang, Patrick P.K. Chan, Wing W.Y. NG, Daniel S. Yeung, “Content-based image retrieval using color moment and Gabor texture feature” Machine Learning and Cybernetics (ICMLC), p. 719-724, 2010.

3. S. Mangijao Singh, K. Hemachandran, “Content-Based Image Retrieval using Color Moment and Gabor Texture Feature”, IJCSI International Journal of Computer Science Issues, Vol.9, Issue 5, No. 1, p. 299-309, 2012.

4. Markus Stricker, Markus Orengo, “Similarity of Color Images”, In SPIE Conference on Storage and Retrieval for Images and Video Databases III, volume 2420, p. 381-392, 1995.

5. Jia Li, James Z. Wang, “Automatic linguistic indexing of pictures by a statistical modeling approach” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 25, no 9, p. 1075-1088, 2003.

6. James Z. Wang, Jia Li, Gio Wiederhold, “SIMPLIcity: Semantics-sensitive Integrated Matching for Picture Libraries” IEEE Trans. On Pattern Analysis and Machine Intelligence, vol 23, no. 9, p. 947-963, 2001.