Media-based Querying and Searching

Introduction

We have implemented a media-based image querying and search program that accepts two RGB files –one search image and one query image – as input and produces a yes/no output of whether or not the query image was found in the search image. If the query image is found, the program also displays the region of the search image where the query image was detected.

Approach to Content-Based Image Search

Using a color descriptor approach to media-based search, we implement color histograms of normalized hue value ranges to represent and compare the distribution of pixel colors in each image.

Preprocessing of Search and Query Images

As the input images are originally in RGB color space, we first convert the pixels of both images to HSV color space. In doing so, we are able to compare the image pixels more efficiently using a 1-dimmensional hue histogram (instead of the 3-dimmensional histogram needed for the three color channels of RGB). In addition, by comparing the images in the HSV color space, we are able to more robustly compare search images in varying light levels by using hue values in our color histogram.

Next, we generate the (global) color histogram of the query image. Color pixels (non-black, white, or gray pixels) are placed into one of 30 “color” bins that contains the pixel’s hue value in its hue value range. Because black, white, gray, and red color shades all have a hue value of 0 in HSV color space, we instead place black, white, and gray pixels based on their saturation and value (intensity) into one of four “gray” bins. Instead of evenly distributing black/white/gray pixels into these bins (as is done for color pixels), we follow tighter requirements for placing a pixel in a designated black or white bins. For pixels with value (intensity) < 0.1 (where value ranges from 0 to 1), the pixel is placed in the black bin. For pixels with saturation < 0.1 and value > 0.8, the pixel is placed in the white bin. For all other gray pixels with saturation < 0.1 and value $<= 0.8$, the pixels are placed into the remaining “gray” buckets based on value ranges.

If an alpha image file is available for the query image, only the query image pixels specified in the alpha image are added to the color histogram.

Because the search image can contain a scaled or transformed form of the query image, we normalize the color histogram by the total number of pixels in histogram. This normalization zeroes out “outlier” bin counts (small amounts of color in the image that aren’t important in respect to the other dominant hues in the image for comparison purposes). In addition, this normalization allows us to use the ratios of bin values to total pixels in the histogram when comparing the query and search images (much more accurate when comparing the query image to a scaled version of itself in the search image).

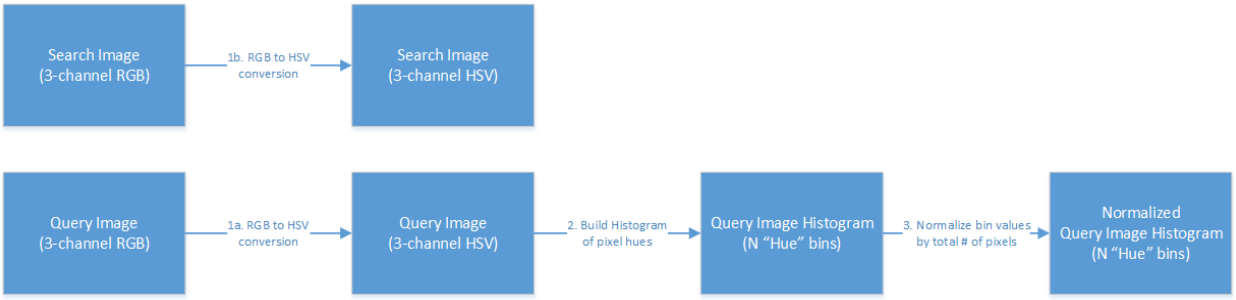


Figure 1. Diagram of preprocessing steps performed on search image and query image.

Searching and Comparing Images

After performing the preprocessing steps on the query and search images, we use the histogram equalization technique of back projection to determine which areas of the search image (if any) are likely to contain the query image. We build a 2D map of pixel brightness values for each pixel in the search image by getting the query image histogram bin value (normalized to a 0-1 value that corresponds to the percent of pixels in the query image that are in this hue bin) that corresponds to the hue of each search image pixel and multiplying it by 255. This "pixel brightness value" corresponds to the probability of this hue appearing in the query image.

After calculating all the pixel brightness values for the search image, we iterate through the back projection map to find clusters of search image pixels that have brightness values > 30 (threshold value for brightness) and are within a radius of 2 pixels from one another using a depth first search. In our implementation, only pixel clusters with 100+ pixels are saved as potential matches. As part of the program output, a rectangle showing each of these potential match regions is drawn on the search image (even if none of the regions are determined to be a true match with the query image).

Next, we iterate through each of the search image pixel clusters (following the order of largest pixel cluster to smallest) and check if it matches the query image. For each pixel cluster, we build a normalized color histogram following the same process as the color histogram generated for the query image. Next, we compute the Bhattacharyya distance between the cluster image and query image. If the distance is less than our threshold value of 0.7 and both images have the same maximum bin value (dominant hue color) within a bin threshold of +/- 2, we determine the cluster image to be a match. If multiple cluster images "match" the query image, we select the cluster image with the smaller Bhattacharyya distance to be the detected region.

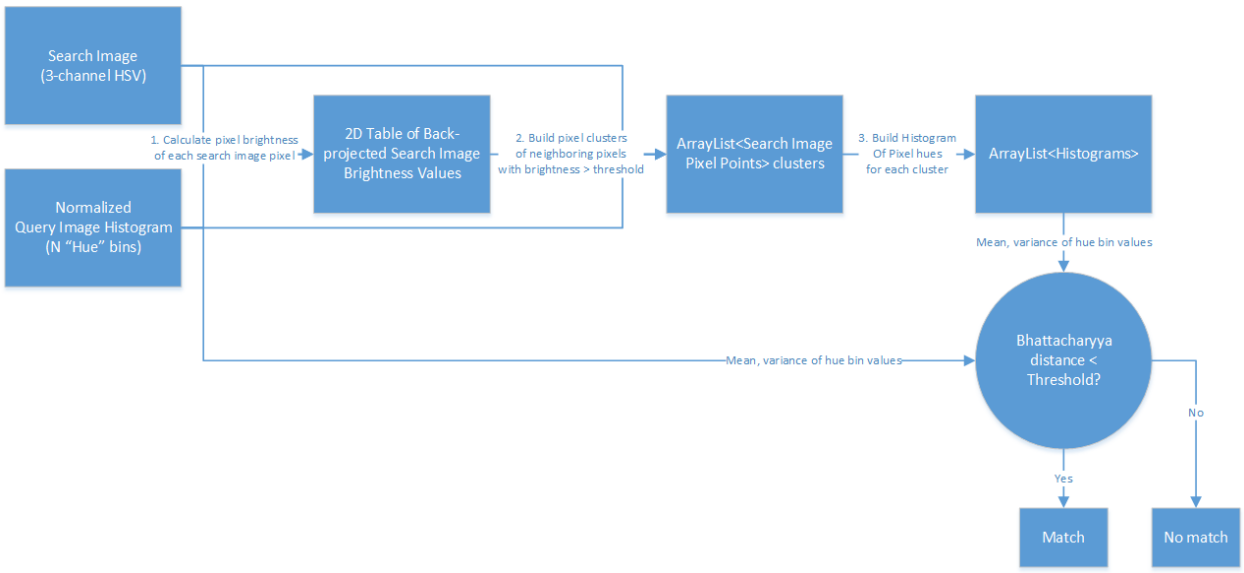


Figure 2. Diagram of search and comparison steps performed on the search image

Example: Normalized histograms of query image and matching search image cluster

Computed Bhattacharyya distance of query and cluster histograms: 0.22468

Hue Histogram Bin:	Query Image:	Search Image Cluster:
Bin: 0	Value: 854	Value: 292
Bin: 1	Value: 0	Value: 47
Bin: 2	Value: 0	Value: 31
Bin: 3	Value: 0	Value: 26
Bin: 4	Value: 0	Value: 20
Bin: 5	Value: 0	Value: 8
Bin: 6	Value: 0	Value: 2
Bin: 7	Value: 0	Value: 2
Bin: 8	Value: 0	Value: 3
Bin: 9	Value: 0	Value: 4
Bin: 10	Value: 0	Value: 3
Bin: 11	Value: 0	Value: 7
Bin: 12	Value: 0	Value: 9
Bin: 13	Value: 0	Value: 32
Bin: 14	Value: 0	Value: 22

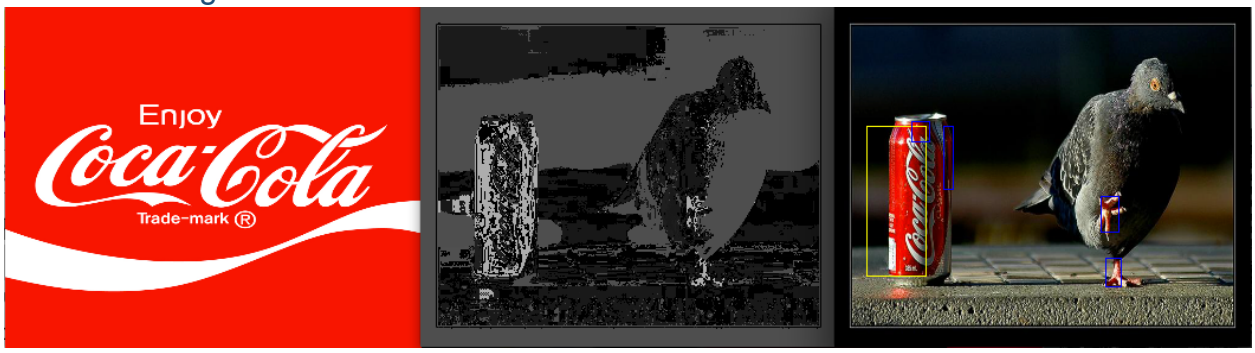
Bin: 15	Value: 0	Value: 2
Bin: 16	Value: 0	Value: 3
Bin: 17	Value: 0	Value: 1
Bin: 18	Value: 0	Value: 0
Bin: 19	Value: 0	Value: 0
Bin: 20	Value: 0	Value: 0
Bin: 21	Value: 0	Value: 0
Bin: 22	Value: 0	Value: 0
Bin: 23	Value: 0	Value: 1
Bin: 24	Value: 0	Value: 0
Bin: 25	Value: 0	Value: 0
Bin: 26	Value: 0	Value: 1
Bin: 27	Value: 0	Value: 2
Bin: 28	Value: 0	Value: 11
Bin: 29	Value: 1	Value: 164
Bin: 30 (Black)	Value: 0	Value: 207
Bin: 31 (White)	Value: 145	Value: 51
Bin: 32 (Gray)	Value: 0	Value: 32
Bin: 33 (Gray)	Value: 0	Value: 17

Results

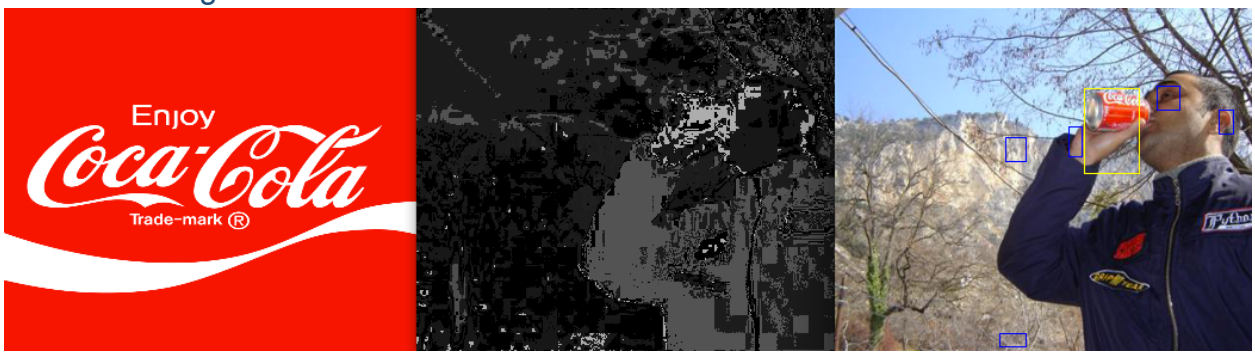
Successes in Search Image Test Set

Our search algorithm was able to correctly detect the logo images in these search images in the test set:

Coca-Cola Logo: Level 1

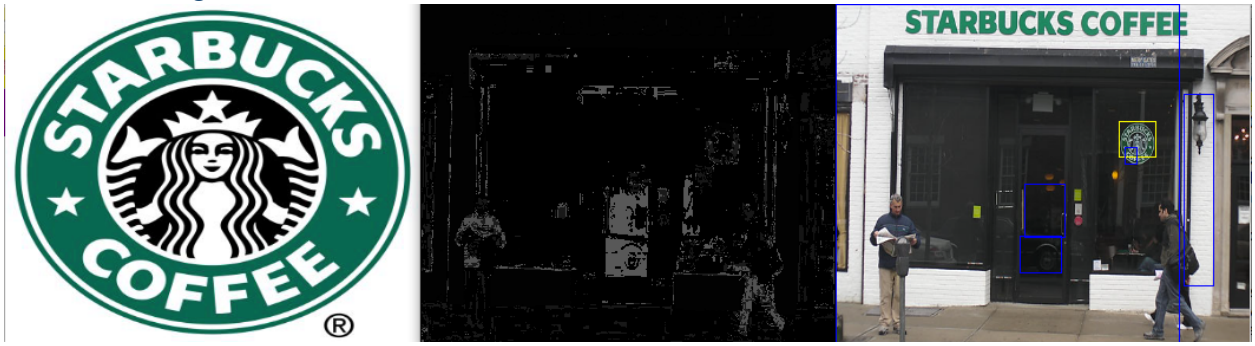


Coca-Cola Logo: Level 2



Starbucks Logo: Level 0

Starbucks Logo: Level 2



Orange Logo: Level 0

US Flag Logo: Level 0

US Flag Logo: Level 1

US Flag Logo: Level 2

Results of False Positive Tests

In addition to testing using the logo query image and its respective search images from the test set, we also tested for “false positives” by testing query images against search images from the set that did not contain the query image. Our implementation behaved correctly in most of the cases; however, we saw “false positive” results for the Coca-Cola logo in i.) US flag search images and ii.) orange search images. We determined these false positives occur when i.) all hue colors of the query image being contained in similar pixel ratio amount in the search image and ii.) the dominant hue color in the query image is placed in the same histogram hue bin as the dominant color in the search image.

Analysis of Failures

Our search algorithm was not able to detect the respective logo images in these search images:

Flowers Logo: Level 1

Our query and search algorithm returns no but marked some of the yellow flowers as potential match regions. Due to the large variety of colors in the source/query image, we were only able to find potential match regions around the yellow flowers in the search image. However, since the mean and distribution of the hue values in the query and search image differ significantly, our algorithm was not able to match on any of the potential regions because the Bhattacharyya distance was above the threshold. Note: if we had raised our distance threshold to 0.8, we would have matched but raising our threshold to this level introduced false positive matches in other queries.

Flowers Logo: Level 2

Our query and search algorithm returns no and does not find potential match regions. Due to the large variety of colors in the source image and variety of colors in the background of the search image, we were not able to discover any potential match regions of 100+ pixels to check the query image against.

Orange Logo: Level 1

Our query and search algorithm returns yes but on the wrong region of the image. Because we only use 30 bins of hue value ranges, the redness in the woman's face and the color of the orange are placed into the same hue bin. As a result, the woman's cheek is determined to be a better match than the orange in the image.

Orange Logo: Level 2

Our query and search algorithm returns yes but on the wrong region of the image. Because we only use 30 bins of hue value ranges, the redness of the child on the left's face (and redness/brownness of her hair) and the color of the orange are placed into the same hue bin. As a result, the child on the left is determined to be a better match than the orange in the image.

Starbucks Logo: Level 1

Our query and search algorithm returns no. Lighting in Starbucks level 1 search image is much darker than in the Starbucks logo image. As a result, the hues of the green and white parts of the logo in the search image are too different from the hues in the query image to detect a match.

Conclusion

Our approach to media-based search and query uses color histograms to compare the query image with regions of the search image that are "likely" to match (based on back projection of query image histogram onto the search image). Our approach works very well for query images with a non-black, white, or gray dominant hue bin value and for query images with more "distinct" hue values (farther than two apart in histogram hue bins). Our approach did not work as well for query images containing white or gray hues when the search image had darker lighting (resulting in the hue values of the logo in the search image to be much different than in the query image). To address this issue in our algorithm, we may increase the number of "gray" bins used in the algorithm to be more robust to lighting changes for black, white, and gray hues.

References

[1] HSL and HSV. Wikipedia. <http://en.wikipedia.org/wiki/HSL_and_HSV>.

[2] Color Histogram. Wikipedia. <http://en.wikipedia.org/wiki/Color_histogram>.

[3] Histogram Equalization. Wikipedia. <http://en.wikipedia.org/wiki/Histogram_equalization>.

[4] Bhattacharyya Distance. Wikipedia <http://en.wikipedia.org/wiki/Bhattacharyya_distance>.