Still-Image Copy Detection

Edwin Babbitt, Nathan Catell

Boston University
Department of Electrical and Computer Engineering
8 Saint Mary's Street
Boston, MA 02215
www.bu.edu/ece

May 2, 2019

Technical Report No. ECE-2019-01

Contents

Table of Contents

List of Figures

List of Tables

1 Introduction to Problem

There are many applications in copy detection such as determining what images are being used for, copyright management, advertisement targeting, image search optimization, and data presentation. In this paper we focus only on copyright management. Today, with the increase of shared media, developing these applications has assumed an increased sense of urgency to combat the upload of copied content to the internet. An example of shared media that makes this application clear is YouTube videos. In this case, videos can be copied and uploaded, and YouTube needs a way for automatic copy detection so that these videos can be removed from the site.

Still Image Copy Detection has two main techniques. The first is digital watermarking, which is the process of embedding information on the media before distribution. This allows copyright detection to be easier. One of the issues with watermarking is that it is possible (although difficult) to remove the watermark, rendering it useless. Even if the water mark is not removed, the person uploading a copied image can apply small distortions to the image which could alter the watermark and make it useless as well. Even slight distortions that would still allow the image to be enjoyed by the viewer could destroy the watermark embedded into the image. This led to content-based copy detection which not only gives a more robust way to manage copyrights, but also gives a solution for the other applications described above. Content-based copy detection is the process of describing an image by some of its features, and comparing the same features of another image. This is much more tolerable to distortions that may be applied to an image and uploaded to the internet. This paper looks into four different algorithms that can help in the process of finding these copied images from a library of masters and distorted images.

2 Literature Review

The issue of image copy detection has been addressed in several papers using content-based metrics to determine image similarity. “Similarity” may be defined by multiple metrics, such as distance (Euclidean, Manhattan, min/max intersections, etc.)

between extracted feature vectors. Multiple distance measurements have been explored by Hampapur and Bolle [1], ranging from the smoothed image difference which utilizes a normalized moving average of pixel color values and is the simplest metric, to color histogram intersection comparisons, to multiple edge detection and comparison methods. Regardless of the metric, similar images will have minimized distance measurements between them.

Color histograms have been used extensively as low-cost feature vectors [1] [2] [3] [4] [5]. Color histograms tend to be fairly insensitive to noise deformation, perhaps from rescaling or compression [2], but generally lack spatial information. Mork et al demonstrated the computation efficiency of using the Variance-Augmented Color Histogram (VACH) [3] which does not rely on the exact position of pixels and color regions, though these cases were limited to fairly benign distortion of the image borders only, and Chum et al have proposed a histogram coupled with a spatial construct to encode both color and position data.

More elaborate feature detection methods may be used to compare specific image contents, not just color distribution and concentration. Scale-invariant feature transforms (SIFT), which are fairly robust against deformation, have been used to compile "visual words" which may be compared against the set of words that is associated with another image [2]. Hampapur et al utilized ordinal intensity signatures to compare the distances between gray levels in partitioned blocks of scenes from a video [4].

Many sources have stressed the need for computational efficiency, and certain techniques may be used to improve search speed, feature vector extraction, and matching. While not the focus of this investigation, further reading is needed on this area to ensure a functional demonstration is achieved.

Spatial information of an image is vital when looking for image copies with distortion. One way to extract spatial information is with Gabor Filters. These filters are used to segment images based on localized textures. These features can be pulled out of an image and used for clustering to segment images using the well-known K-Means algorithm [6]. Doing this allows for copy detection by comparing the cluster's differences with those of another image.

One application of feature vectors is the covariance matrix comparison. Feature vectors may contain pixel information such as coordinate, color, derivatives, luminance, etc. [9]. The covariance comparison is particularly useful since each image, no matter what size, feature vector of length *N*, can be represented by an *N x N* covariance matrix. The covariance matrix conveniently combines multiple types of image data, potentially reducing the number of steps required to process an image before comparison. Since the covariance matrix does not reside in Euclidean space, the simple difference between two covariance matrices would not adequately describe their relationship. It has been proposed to use the *l*2 vector norm of the matrix-log difference instead [10].

3 Problem Statement

3.1 Problem Details

The simplest way to perform copy detection is to do a pixel by pixel comparison of two images and measure how similar they are based on the average difference of pixel values. This works in certain situations such as comparing an image with itself filtered, or transformed in some way. However, applications that require going through a large data set and looking at images of different sizes and containing different distortions cannot use pixel by pixel comparison. Images could be compressed, have added noise, be shifted, have a change in contrast or brightness, be converted in color scheme, have an added border or many other changes that make copy detection difficult. In the example of searching for a copyrighted image, an algorithm would still need to determine that an image was a copy with any or all of these distortions. There needs to be an algorithm that can extract content from an image to create an image representation with its features. This is then compared to a data set of other image representations with a distance metric that can handle any of the distortions that an image might obtain. There are many different distance metrics that can be applied and each perform differently with different algorithms. For this reason, different metrics need to be tested and the highest performer must be chosen for the given algorithm implementation. There also needs to be some type of threshold of similarity selected to accurately determine if two images are a copy of each other. This creates the difficulty of dealing with false positives. There is always a

tradeoff between missed detections of copies, and false positives which needs to be optimized through experimentation.

A copy of an image even with slight distortions is easy for a human to pick out. This means that an image or video could be copied and uploaded on the internet and other people could enjoy them. However, for a computer, finding these copies is a difficult task. This is why these algorithms need to be implemented to give the computer some sort of visual ability to pick out the copies.

All of these difficulties together lead us to our goal which is to implement automatic copy detection given a target and reference library of images.

3.2 Proposed Solutions

To solve the problem stated above we developed four different algorithms and compared their results in the following sections. Each of the four algorithm's Flowcharts are shown below in this section.

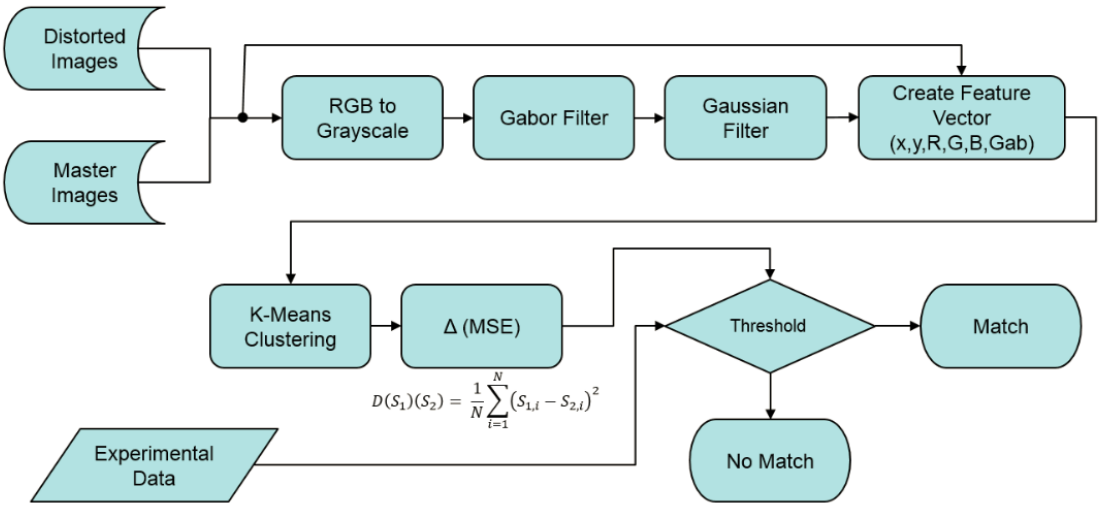


Figure 1 Flowchart of K-Means Clustering Algorithm

Figure 1 above shows the flowchart of the K-Means Clustering Algorithm that was implemented. In the following section this algorithm is described in detail.

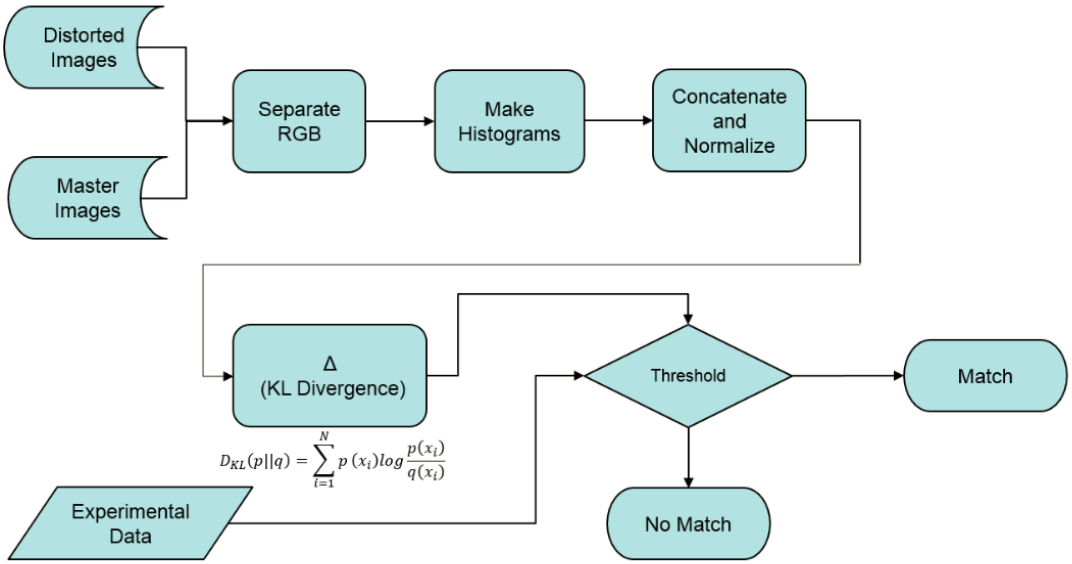


Figure 2 Flowchart of Color Histogram Algorithm

Figure 2 above shows the flowchart of the Color Histogram Algorithm that was implemented. In the following section this algorithm is described in detail.

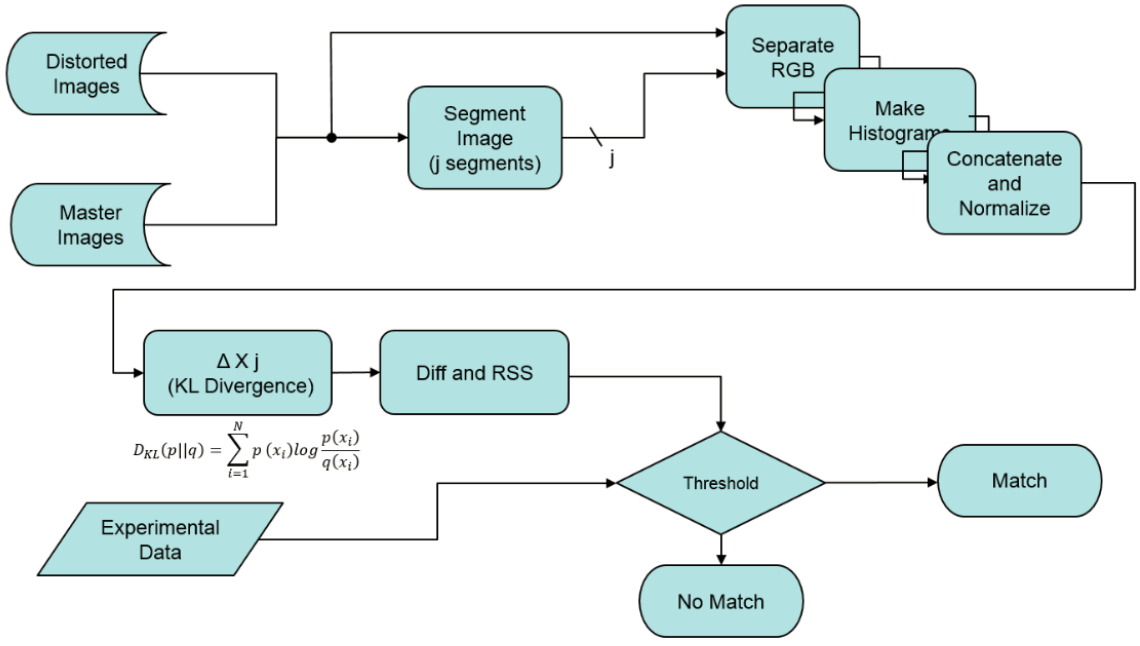


Figure 3 Flowchart of Segmented Histograms Algorithm

Figure 3 above shows the flowchart of the Segmented Histograms Algorithm that was implemented. In the following section this algorithm is described in detail.

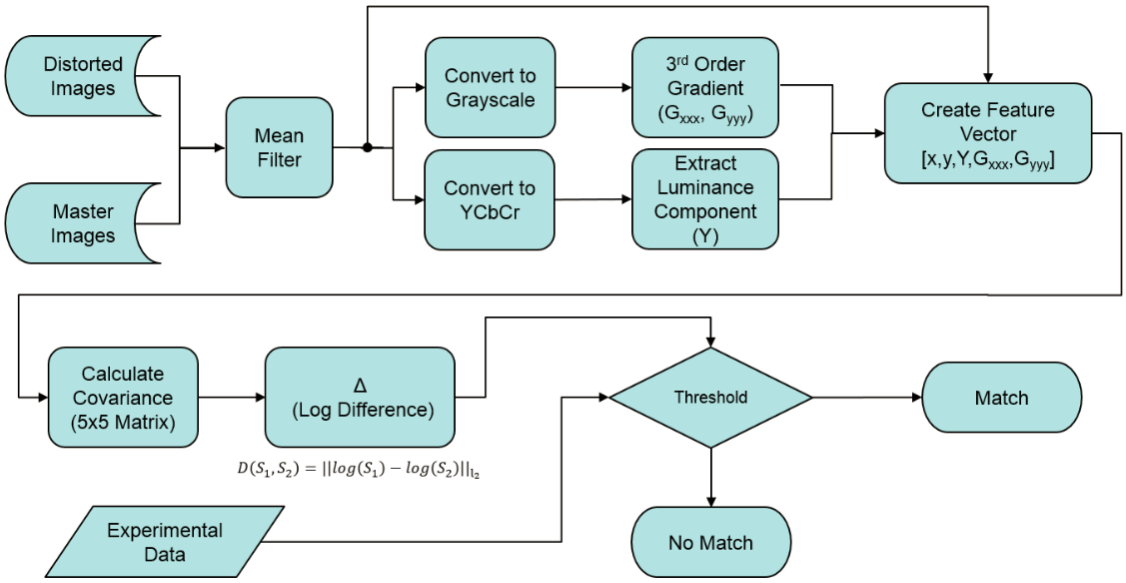


Figure 4 Flowchart of Feature Covariance Matrix Algorithm

Figure 4 above shows the flowchart of Feature Covariance Matrix Algorithm that was implemented. In the following section this algorithm is described in detail.

3.3 Assumptions and Constraints

While developing the algorithms shown above there were some assumptions and constraints that were made. The first was that to reduce complexity, still images are used instead of video. Next, all images in our master image library that were to be searched through for matches were the same size, and the distorted versions of these master images only had distortions applied to the extent that would allow for viewers to have the ability to still enjoy the image. This is because for this type of application, people who are copying and uploading images or videos to the internet want people to view or watch what they are uploading. People will not want to view an image or video with too intense of distortions. We also ignored computational efficiency of our algorithms, and purely

designed for performance. Lastly, a match was constrained to be a binary decision. Two images either match or they do not.

4 Implementation

4.1 Gathering and Using Dataset

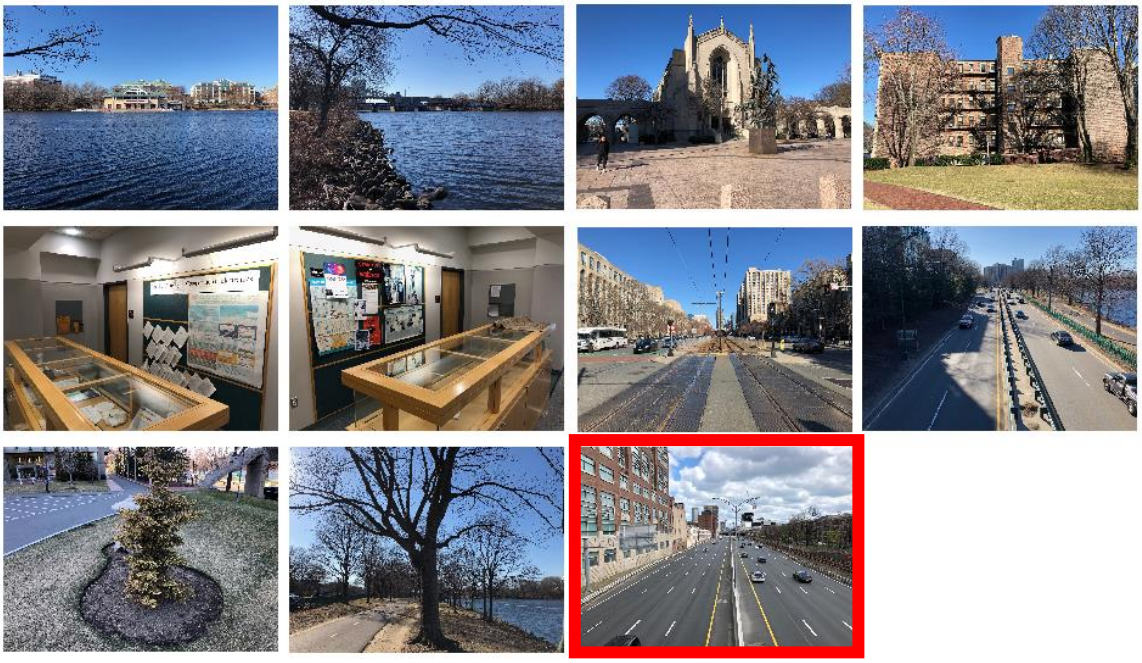


Figure 5 Master Images

We developed our master image library by capturing a directory of 11 photos shown above in Figure 5. The 10 images that are not enclosed in red are the original master images that were all taken on the same day, around the same time and in the same general area. This was done so that the lighting was similar between each image to make the copy detection more difficult. Each of these 10 images has an associated “like” pair. For example, the two images on the top left are one of these pairs. One is an image of the BU boathouse, and the other is of a bridge near the boathouse. These images have the same river, and a similar sky and trees. Other pairs are the two building, the two cases in a hallway at different angles, the highway and railroad tracks, and the two trees. These pairs are of similar scenes and have similar composition. This is intended to provide a likely false-positive candidate, to demonstrate the effectiveness of our developed algorithms.

We developed code to create distorted versions of each of the 10 original master images. We have divided these modifications into “Photometric” and “Geometric” distortions. These distortions are listed in Table 1.

Table 1 Distortion Types

Distortion	**Parameter**	**Type**
Added Border	20 Pixels Wide	Geometric
Gaussian Noise	$\mu = 0$, $\sigma^2 = 0.01$	Photometric
Salt & Pepper Noise	Density = 0.05	Photometric
Low-Pass Filtered	Wp = 0.45, Ws = 0.55, n = 11	Photometric
Modified Brightness	+15%	Photometric
Rotation	1°	Geometric
Compression	JPEG Quality = 25	Lossy

This results in a directory of 70 distorted images (7 distorted images per master image). The last addition to the data was a dummy image. This can be seen in Figure 5 as the image that is enclosed in red. This image was put into the master directory as a test for our algorithms. It was taken on a different day as the other images, however it was still a bright day and of a highway. It shows similarities to the some of the other images. This is especially true for the other image of a different highway. We wanted to use this this image as a true negative to make sure our algorithms would not match any distorted image to this dummy image.

Each of the algorithms that follow in this section were tested with the same methodology. Each of the distorted images were compared to all of the master images attempting to detect which ones were copies based on the experimentally found threshold. This means that each distorted image could result in a match against all of the master images, none of the imagines, or anything in between. There can be three different results for each comparison. First is a true positive which means that a distorted image is predicted as a match with the master image it was produced from. Second, a false positive is when it is predicted that a distorted image is a match with a master that it does

not truly match. Lastly, a miss is when a distorted image is not predicted as a match with any master image.

4.2 K-Means Clustering

Figure 6 BU Boathouse RGB, Grayscale and Clustered Image

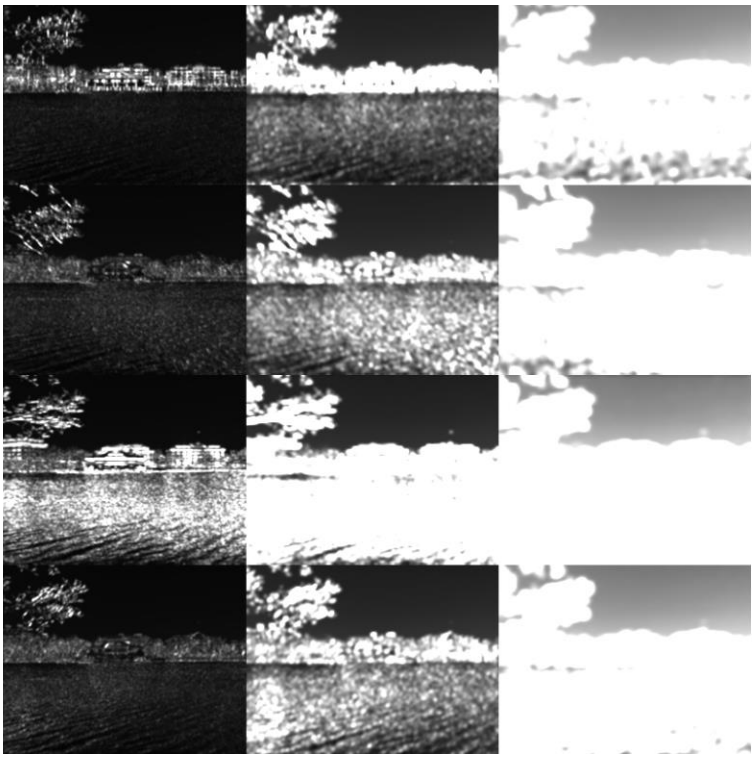


Figure 7 Output of 12 Gabor Filters performed on the BU Boathouse Image

A K-Means Clustering algorithm was developed to segment the master and distorted images into clusters based on variations in textures, RGB values and pixel location as its features. An example of this is shown in Figure 6 above which shows the master image of the BU Boathouse segmented into clusters alongside the original RGB and Grayscale image. It is shown that the buildings and trees are separated from each other and put in their own clusters. It also should be noticed that both the sky and water have a lighter blue and darker blue portion which are clustered separately from each other. This all makes sense because not only do all of these clusters have different RGB values, but they also are varying in textures. The algorithm successfully segments each of these into separate clusters.

The first step in the algorithm is to resize the distorted image to the size of the master image. This is because the features being extracted for comparison are each the size of the image. Once the images are the same size the algorithm extracts features from both the master and distorted images which are being compared. The features give a representation of both images under analysis. The first five features are simply the pixel location, and the R, G, and B component of the images. This gives us color information

and where the colors are located. When these were the only features the performance was extremely poor, and we quickly knew that we needed more information. MathWorks suggested the use of Gabor Filters [7] to gain texture information.

A Gabor Filter bank was created which is used for texture analysis by finding frequencies in specific directions to make clustering of certain textures possible. We selected six different frequencies and 4 different orientations/directions for the bank of 24 total filters. Each filter corresponds to a different frequency at a different orientation. The images were converted to grayscale and fed into this filter bank. Each of the 24 filter outputs is a magnitude response that shows these different components of textures in the image which are each used as a feature for clustering. An example of the output of a Gabor filter bank with 12 filters performed on the BU Boathouse image is show above in Figure 7.

The output of the Gabor filters are then further filtered by a Gaussian filter to smooth local variations of the image to create better features for clustering. Even though filtering causes some information to be lost it proved to help copy detection by smoothing the features. The magnitude responses of the filters along with the pixel location and RGB information results in 29 total features.

Given a number of clusters the images are then clustered with the K-Means algorithm [7]. This means that each pixel is clustered with others that are most similar based on these 29 features. Lastly, the centroids of each cluster are made into a matrix that represents that image (the result is a number of clusters by 29 matrix). The MSE between the two matrices of the images being compared is calculated and the two images are deemed a match if the error is under an experimentally found threshold.

The MSE equation is shown below:

$$MSE = D(S_1)(S_2) = \frac{1}{N}\sum_{i=1}^{N}\left(S_{1,i} - S_{2,i}\right)^2$$

It should be noted that although our own implementation of the Gabor filter was attempted, the built in MATLAB function was used because it was fully optimized.

Finding the optimal numbers of clusters to use was a great challenge. At first, it was thought that we could simply use 5 clusters and this would give sufficient information to perform accurate copy detection. Although this gave reasonable results, they were far

from optimized. It was found that changing the number of clusters used based on the master image that a distorted image was being compared to yields far better results. This optimal number of clusters was found by experimentation, however can still further be adjusted for better performance. The issue is that running the algorithm and performing all the comparisons takes around 5 hours so the experimentation and adjustments take a long time.

Another difficulty was finding the correct distance metric and threshold to use. The two distance metrics that were used and compared were MSE and the Log Difference which is explained further in the Feature Covariance Matrix algorithm implementation section below. At first, the Log difference metric seemed very promising, but was inconsistent with different images. Through testing, and adjusting the threshold based on results it was found that MSE performed best as a distance metric and a threshold of 6.8 was optimal.

4.3 Color Histogram

A normalized histogram provides the distribution of values in each color component of an image, e.g. red, blue and green (RGB). Similar images should contain similar distributions of color components. This approach generates a concatenated histogram of each color component of an image *I* and normalizes the histogram by the size of the image, i.e. the number of pixels multiplied by the number of color channels,

$$hist_T(I) = [hist_R(I), hist_G(I), hist_B(I)] \Big/ numel(I)$$

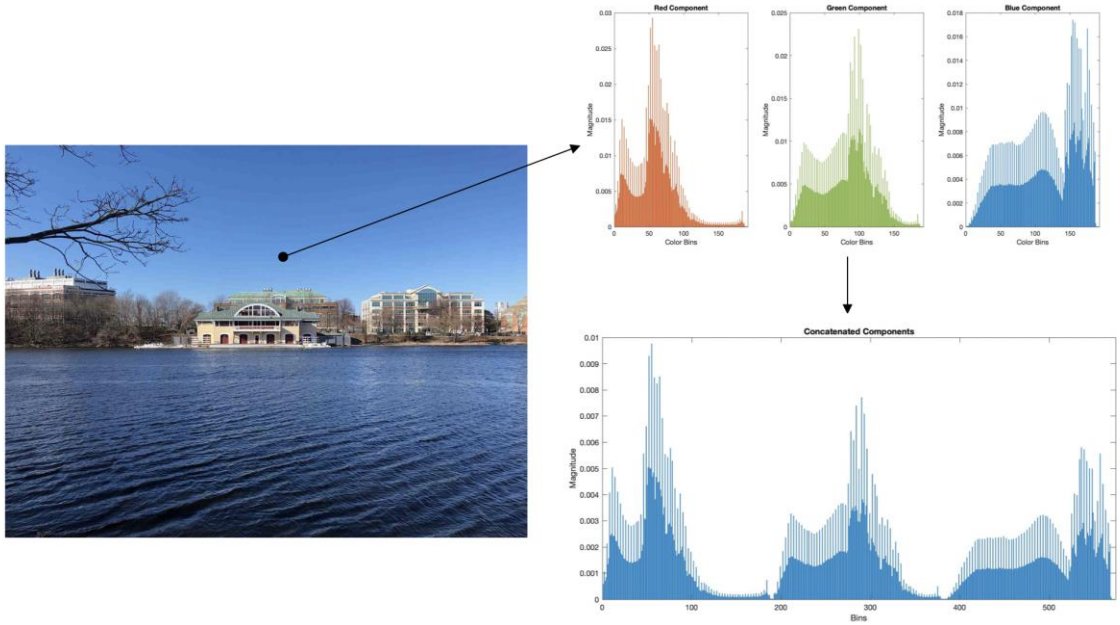


Figure 8 Concatenated Histogram for BU Boathouse

Figure 8 demonstrates this process for the master image of the BU Boathouse. Histogram differences were first compared by calculating the histogram overlap of the two candidates as shown in Figure 9 and by,

$$D_{OL}(I_1, I_2) = 1 - \sum_{bins} \min(hist_T(I_1), hist_T(I_2))$$

Only 75% of the total number of color magnitudes [0-255] were used for the bins in creating each histogram. This "condensed" the final distribution and improved the overlap scoring between master and distorted images. A threshold of $D_{OL} < 0.45$ was determined experimentally for a matching histogram.

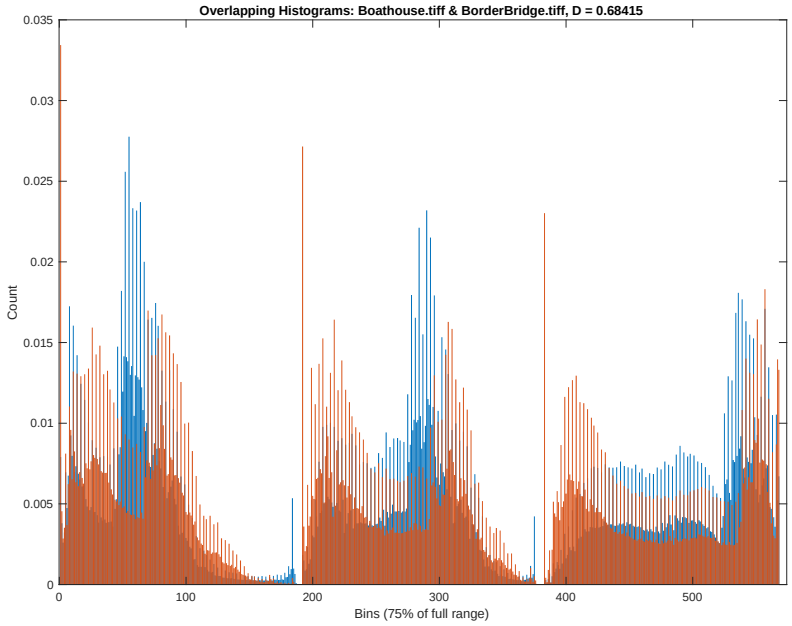


Figure 9 Histogram Overlap

The histogram overlap metric did not prove to be a robust measure of image similarity. Using the fact that the concatenated histograms approximate a probability density function (PDF), a second distance metric, the Kullback-Leibler (K-L) divergence, was then used. The K-L divergence calculates the expected value of the log difference between two distributions,

$$D_{KL}(p||q) = \sum_{i=1}^{N} p\,(x_i) log \frac{p(x_i)}{q(x_i)}$$

where p and q are the calculated PDFs of the two candidate images,

$$p = hist_T(I_1), \qquad q = hist_T(I_2)$$

A more stringent threshold of $D_{KL} < 0.05$ was determined experimentally for a matching histogram using the new metric.

This approach was very sensitive to photometric distortions, so a second approach was developed in an attempt to find a more robust color comparison. The algorithm was modified to incorporate location data within the analysis by computing ratios between segments of each image to the whole. We call this approach the Segmented Histograms method. Theoretically, the ratio between each segment and the whole image should remain fairly constant even through photometric distortions.

Each image *I* was divided into six non-overlapping sections $I_{j,segment}$, j = [1,6]. Concatenated RGB histograms were created for each segment and for the original image. The set of overlaps between the segments and the whole image, described by vector $S_{OL} = [D_{OL}(I,I_{j,segment})]$ was then calculated. Figure 10 demonstrates this process for the master image of the BU Boathouse for six sections.

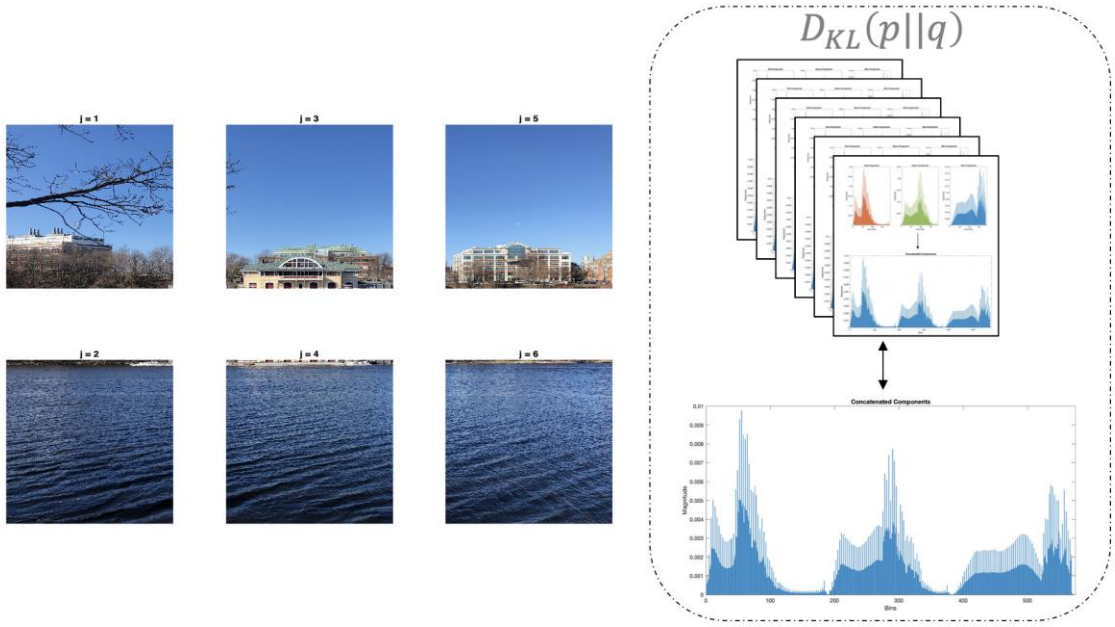


Figure 10 Segmented Image and Segment-Image PDF Comparison

The Euclidean distance *D* between master and distorted S_{OL} vectors was used to compare images. A threshold of $D < 0.2$ was determined experimentally for a matching histogram. Again, the overlap proved not to be a robust difference metric, and so the KL divergence was used instead, producing the vector $S_{KL} = [D_{KL}(hist(I)||hist(I_{j,segment}))]$. This resulted in an optimal threshold of $D < 0.058$.

4.4 Feature Covariance Matrix

A covariance matrix calculated from an image feature vector is a compact structure with which to represent the image. Given an image with *n* pixels, and a feature vector *v* containing *M* features, the covariance matrix is an *M x M* matrix given by,

$$C = \frac{1}{n-1}\sum_{i=1}^{n}(v_i - \bar{v})(v_i - \bar{v})^T$$

where $\bar{v}$ is the mean of the feature vector. As long as the number of features *M* is held constant, the *M x n* dimensional matrix $\{v_i\}_{i=1\ldots n}$ (each feature is defined per pixel) will produce an *M x M* covariance matrix via the outer product, regardless of the size of *n*. As mentioned previously, covariance matrices do not reside in the Euclidean space, and so the relationship between two covariance matrices is described by the *l*2 vector norm of the matrix-log difference,

$$D_{||mlog||}(C_1, C_2) = ||mlog(C_1) - mlog(C_2)||_{l_2}$$

Multiple features were examined for covariance analysis. The following pixel features were considered: coordinates, color components, luminance, and first, second, and third order partial derivatives ($x, y, R, G, B, Y, \delta_x, \delta_y, \delta_{xx}, \delta_{yy}, \delta_{xxx}, \delta_{yyy}$). A five-feature vector was selected for testing purposes, and combinations of the selected features were tested. It was experimentally found that a vector using coordinates, luminance, and third order partial derivates produced the best results,

$$v = [x, y, Y, \delta_{xxx}\delta_{yyy}]$$

Interestingly, including the third order derivatives did not necessarily reduce the $D_{//mlog//}$ measure towards zero for matching images, but instead it increased the distance measure between non-matching images.

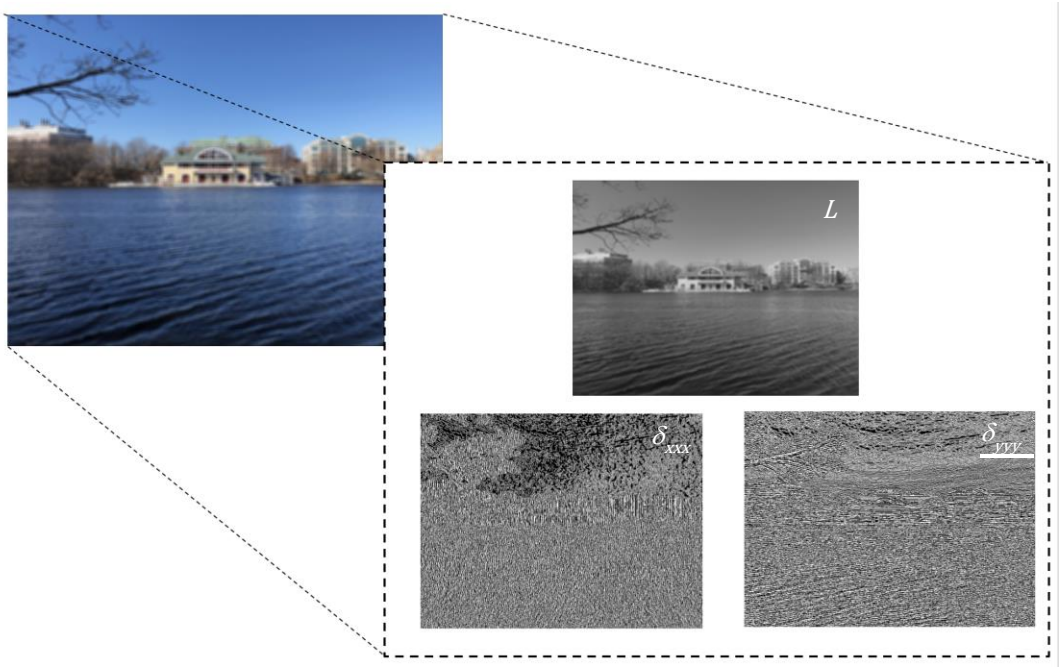


Figure 11 Feature Extraction from Filtered BU Boathouse

Prior to feature extraction, each image was filtered using a 2-D averaging filter with a 15 x 15 mask to reduce noise. The most effective mask size across all images in the dataset was determined experimentally. The image was then converted to both YCbCr to extract the luminance components and Grayscale to calculate the 3rd order partial derivatives, and the feature vectors were constructed. The covariance matrices for the master and distorted images were then calculated and the difference $D_{//m log//}$ between the two was found and compared to the experimentally determined threshold. Multiple tests for the covariance distance metric resulted in a threshold of $D_{//m log//} < 0.3$.

5 Experimental Results

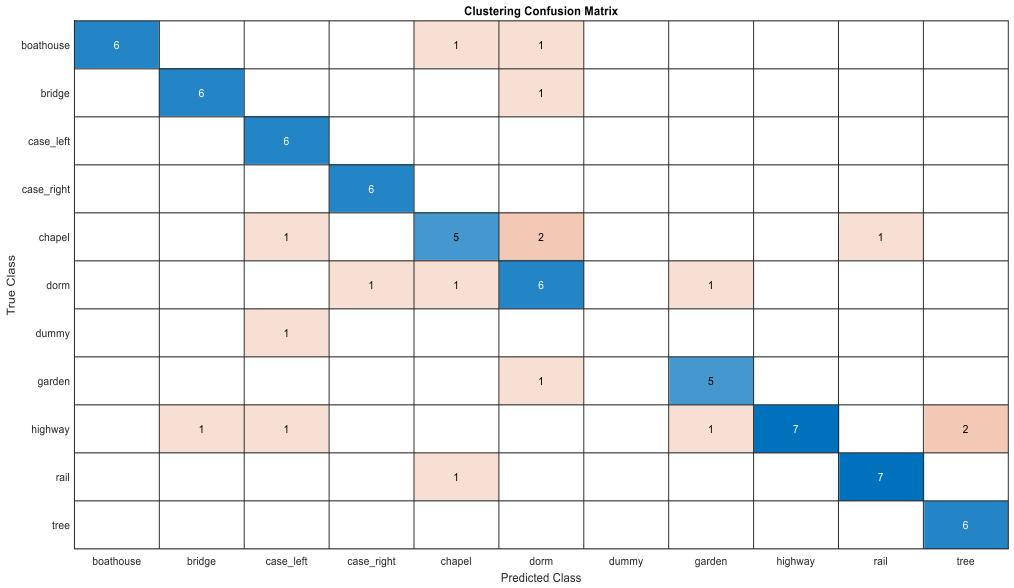
As the experiments described above in the Implementation section were performed, each match, as determined by the threshold levels, was logged along with the type of match. The performance of the algorithm is determined by evaluating the confusion matrix and performance bar graphs for each algorithm. Along the diagonal of the confusion matrix with blue squares shows the true positives. The darker the blue means there are more correct hits. There is also a number that indicates this number of hits on each blue square. The red squares are false positives. Similarly, the darker red indicates more false positives and there is also a number for indication. The scoring code is structured in such a way that only distortions that were caught as hits or false positives were logged for the confusion matrix. If each distortion was caught, the sum of each column of the confusion matrix should be seven, otherwise misses were not logged. This bug leaves out the information regarding the missed distortions and should be addressed in a future update. This bug was addressed by creating the performance bar graphs. The graphs below show the totally number of true positives, false positives, and misses for each image. Each algorithm shows both a confusion matrix and performance bar graph.

It should be noted that we met our fallback goals, and most of our reach goals. From our intermediate report our fallback goal was to keep optimizing the Color Histogram, Segmented Histograms, and K-Means Clustering algorithm. This goal was achieved and we also went on to meet most of our reach goals. One of these goals was to add

compressions as a distortion and another was to develop the Feature Covariance Matrix algorithm. Both of these goals were met.

5.1 K-Means Clustering

Table 2 K-Means Clustering Confusion Matrix



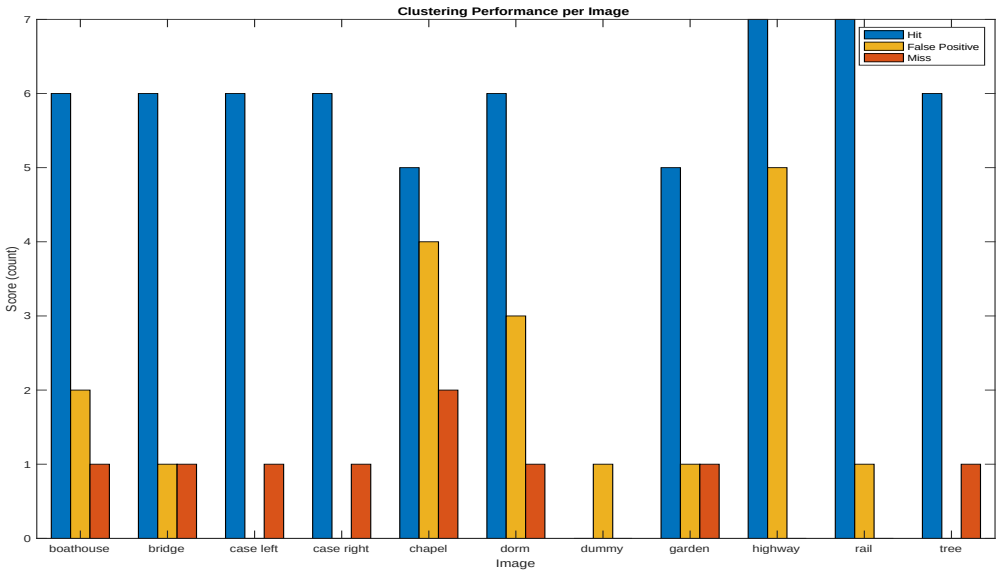


Figure 12 K-Means Clustering Performance Bar Graph

K-Means Clustering produced good results and showed promise to be able to be optimized even further. Table 2 above is a confusion matrix that shows the results of the K-Means Clustering algorithm. This shows that the lowest number of true positives for any case was five out of seven however, in most cases six out of seven or even all seven true detections were found. The two distortions that were sometimes missed by the algorithm were Salt and Pepper noise and rotation. We feel that these two distortions are the least likely distortions for the copy detection application because they make images appear aesthetically worse than the other distortions. This would potentially mean that viewers would not want to view the image. Although it would be better to be able to catch these copies as well in all cases, we do feel it is best to miss these two distortions.

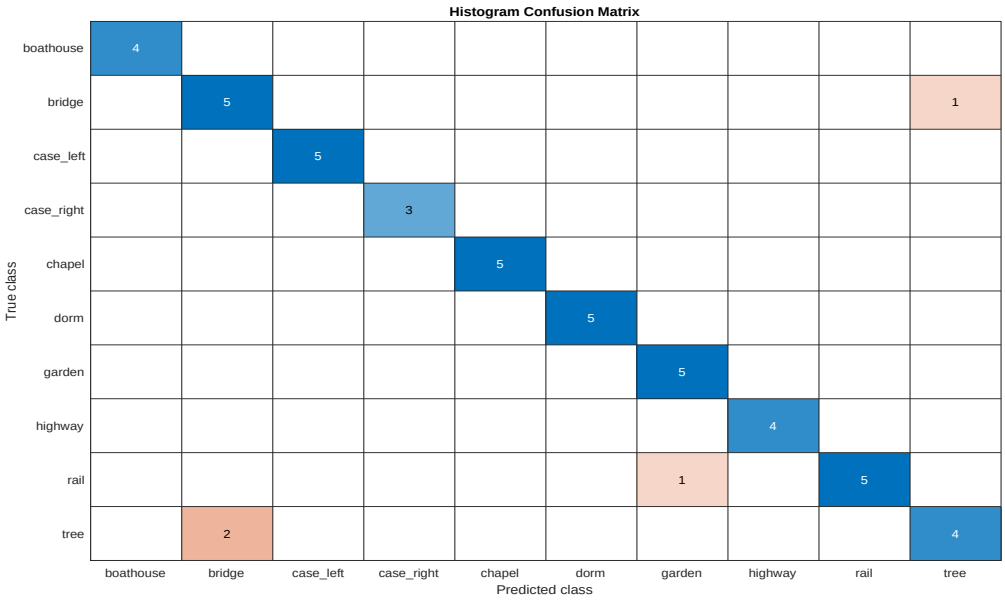
From figure 12, it can also be seen that there are quite a bit of false positives in some cases. This is partially because the threshold was designed to be set high to catch more true positives, and partly because the number of clusters is not optimized in each case. Finding the optimal number of clusters for an image is a difficult task, however, doing so would result in even better results than seen above. The difficulty with this algorithm is its computational efficiency. It takes ~5 hours to run through all 70 comparisons which makes it extremely difficult to tune the algorithm. Even though tuning the algorithm was difficult it did perform well, and has shown room for improvement.

5.2 Color Histogram

Color Histograms produced fairly poor results, captured in the confusion matrix in Table 3. At most only 5 images out of 7 were correctly identified, and in the worst case only 3 images were correctly identified. Notably, while this approach identified relatively few false positives (4 total out of 70 candidates), this is due to the fact that there were a significant number of misses, as shown in Figure 13. According to our methodology, we prioritize a false positive over a miss. These results mean that with Color Histograms, several copies could slip through the system.

The histogram comparison performed fairly well for the "geometric" distortion category, but completely failed to detect both the brightened and lowpass-filtered distortions. Several of the cases missed the gaussian noise distortion as well. Since this algorithm relies completely on the color content of an image, it makes sense that brightening or lowpass-filtering, which may cause significant changes to the color content and distribution, and adding random noise to the color distribution, would prove difficult.

Table 3 Color Histogram Confusion Matrix



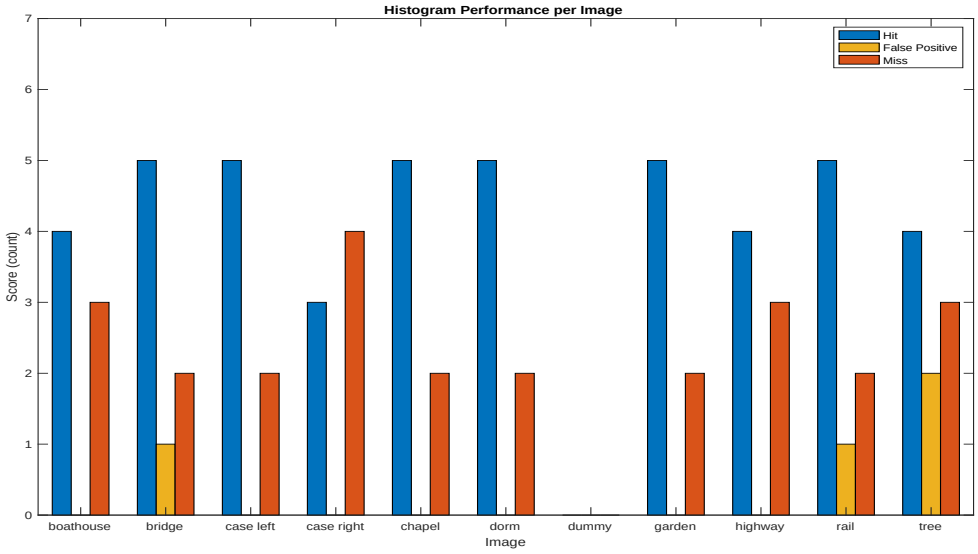
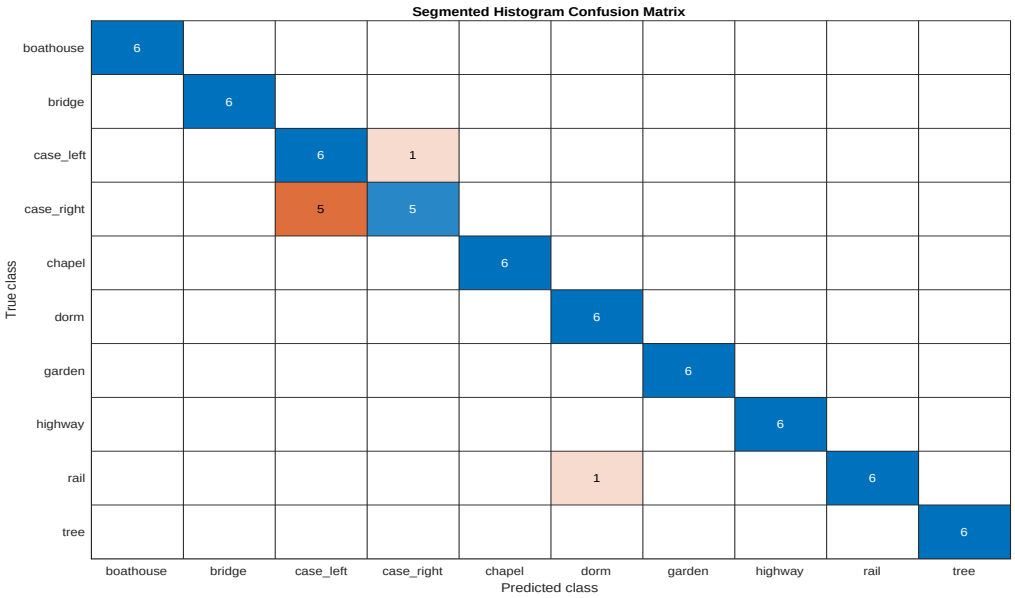


Figure 13 Color Histogram Performance Bar Graph

Segmented Histograms, however, reliably detected matching images through both "photometric" and "geometric" distortions. Interestingly, Figure 14 shows that this implementation failed to detect a match (a "miss") for gaussian noise in nearly every case. The single case, "rail", that detected the gaussian noise distortion resulted in a false positive, identifying the image as "dorm". As shown in the confusion matrix in Table 4, the segmented-histogram method is more sensitive to very similar images, for instance in

the case of the false positives when comparing case_right.tiff and case_left.tiff. These particular images were selected to present an especially difficult case, so it is unsurprising that that they may be misidentified. It was found that a lower threshold eliminated the false-positives, but increased the number of misses. The higher threshold was selected for this report since the copy-detection application should prioritize false positives over misses. To fully evaluate this algorithm different numbers of image segments, *j*, should be evaluated as part of future work.

Table 4 Segmented Histograms Confusion Matrix



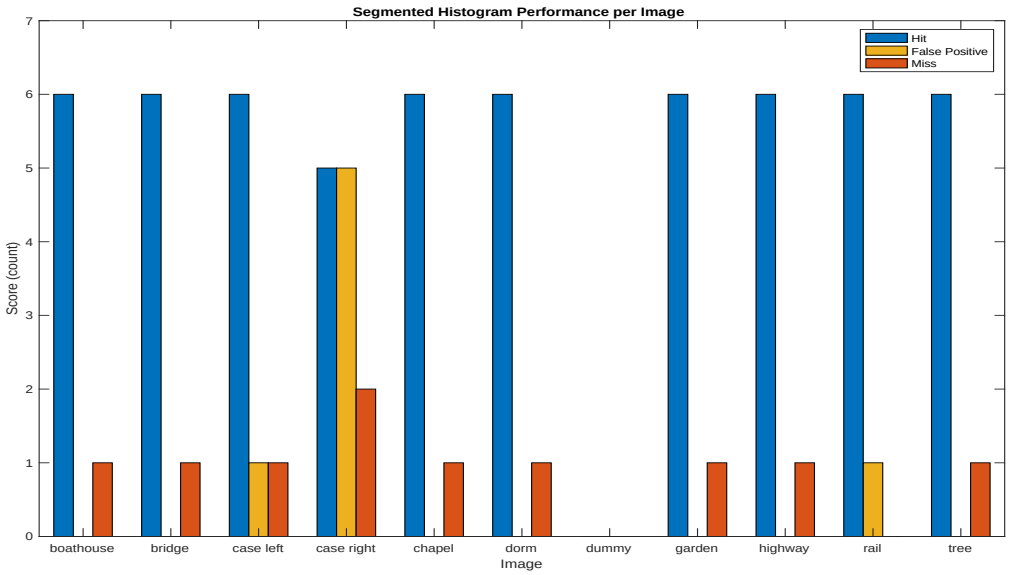


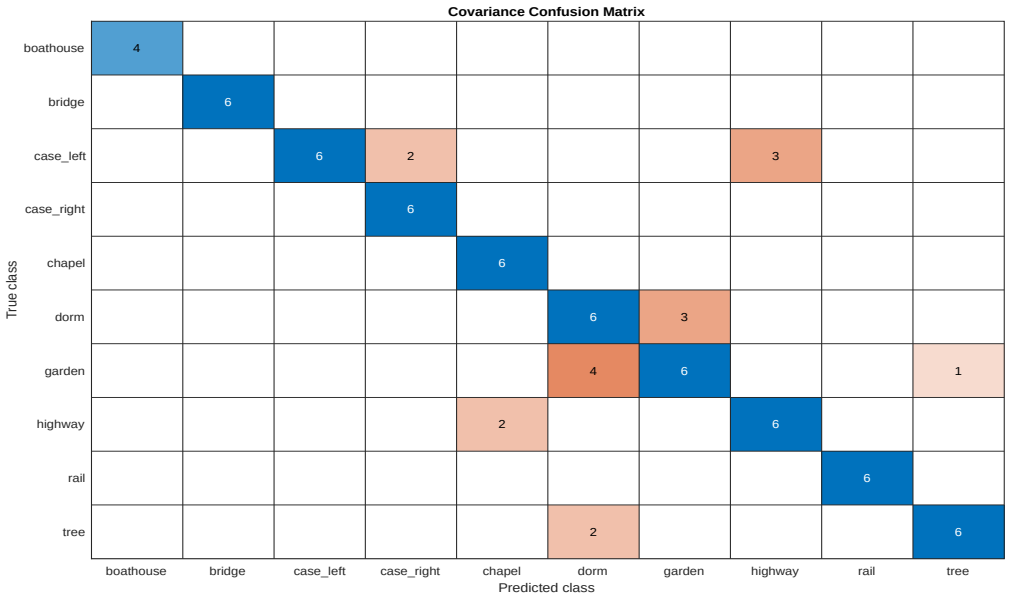
Figure 14 Segmented Histograms Performance Bar Graph

5.3 Feature Covariance Matrix

Feature Covariance Matrix detected matching images through most "photometric" and "geometric" distortions. Similarly to Segmented Histograms, Figure 15 shows that most cases had a single "miss" which we have identified as the "low-pass filter" distortions. We theorize that this is due to the low-pass filter blurring the image and reducing the edge detail, essentially removing edge information from the distorted image. As shown in the confusion matrix in Table 5, this approach identified a significant number of false positives but this is due to the threshold prioritizing a match over a miss. Feature Covariance particularly struggled with the "boathouse" case. This is likely due to the fairly uniform color within the image.

The pre-processing involved in the algorithm, both filtering and feature extraction, increases the runtime of significantly. Practically, this limited the investigation for the sake of reducing processing time and increasing the number of tests run. Processing time could be reduced by storing the master image feature vectors and only performing feature extraction on the distorted images as needed. While the 5-feature vector selected performed best, the *x*-*y* direction partial derivatives and other features should be considered in future work. Larger feature vectors, and thus larger covariance matrices, may improve the results and should also be considered.

Table 5 Feature Covariance Confusion Matrix



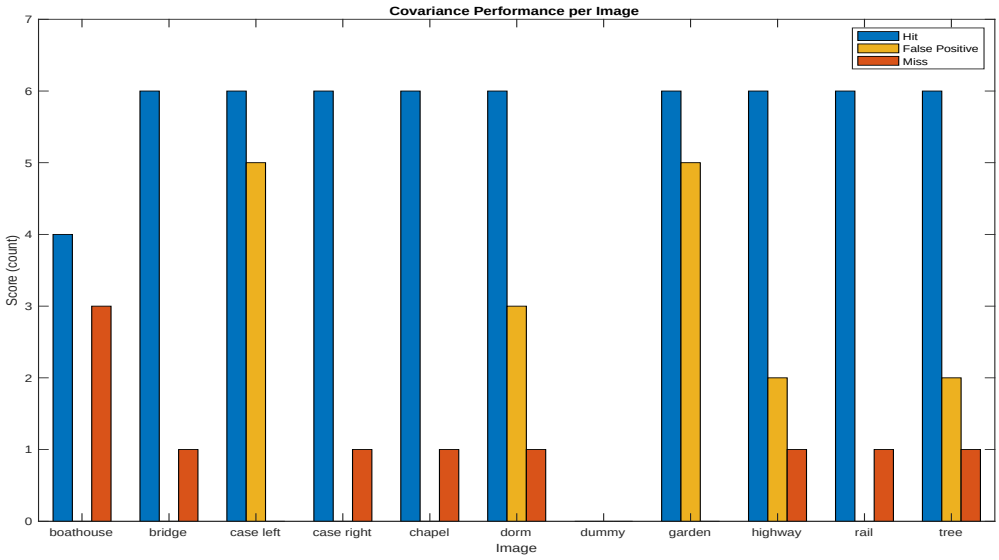


Figure 15 Feature Covariance Performance Bar Graph

6 Conclusions

6.1 Takeaways

During the progression of this project a lot was learned. Not only was the application of such algorithms and their importance in today's society studied, but also the difficulty of the task was explored. As stated earlier copy detection of images is something that is simple for the human eye which allows people to apply the simple distortions explained above and upload them as their own. These copies can then be enjoyed by viewers while not experiencing much degradation of quality to the human eye. To a computer, however, it is difficult to find these copies with distortions. There are many algorithms that have been attempted for use in this application. In this paper, four such algorithms are described, but there are many others that can be implemented. This field of research is important because there has not been an algorithm that has been found that performs perfectly at solving the problem.

Selecting the correct distance metric can be almost as important as selecting what algorithm to use. As seen above, there are three different distance metrics that were used in our four algorithms. In each case many distance metrics were tested, but one seemed to always work better. It was interesting to find that the same distance metric was not always the best. This proves that finding the correct distance metric takes time and testing because each can be better in different scenarios.

The importance of selecting a correct threshold was also learned from this project. When optimizing an algorithm, the threshold will change and this needs to be accounted for. Also, it should be noted that through experimentation we learned that using different features greatly effects the threshold that needs to be selected. Some features can give more information and lower the difference measure between two images greatly as a match, but they also may lower this difference of non-matches. Other features may widen the margin of difference between matching images and non-matches. The best example of this we found was in our research of the Feature Covariance Matrix algorithm. Using the 3rd order gradient instead of the 1st or 2nd order did not lower the error between matches, however it widened the margin by creating a larger measure of difference between non-matches.

Finding these optimal distance metrics and thresholds turned out to be difficult because there are so many options. Also, in most cases one metric would be better in certain cases but worse in others. This led to the need of finding the best choice for most cases.

During our implementation of the Feature Covariance Matrix and K-Means Clustering algorithms we ran into an efficiency problem. Although we decided to ignore computational efficiency, the time it took to run these algorithms created difficulty for us to test and optimize them.

Specifically, for K-Means clustering there was a major problem with finding the optimal number of clusters. The performance of the algorithm turned out to be greatly depended on the number of clusters chosen. Each master image has a specific number of clusters that is optimal for copy detection, and knowing that number needs to be found through testing. This is a difficult task, however, once done could make it the best performer of the bunch.

We have ideas to improve the other algorithms and develop new ones, however, currently our best performer is Segmented Histograms as seen from the results above.

6.2 Future Improvements

Moving forward we would like to continue to optimize our existing algorithms. Particularly, the K-Means Clustering still has room to grow. Testing will be continued to find the optimal number of clusters for each master image to improve its performance. Also, research will be done to try and find a way to automatically find the optimal number of clusters. There are methods out there such as the Elbow and Silhouette method which gives an optimal number of clusters. These techniques will be looked into and a decision will be made if they can help us or not. Another improvement we want to try is adding more features to our Feature Covariance Matrix algorithm.

We want to make improvements to our dataset as well by gathering more images to grow our libraries for testing. This would give us more data, and allow us to do more rigorous tests on our developed algorithms. It is possible this could point out flaws that could be improved upon.

Lastly, there are still other methods of copy detection that we want to explore. The first being a twist on our existing Segmented Histograms and Feature Covariance Matrix methods. This would result in a Segmented Feature Covariance algorithm, comparing covariances of multiple regions of images instead of the whole. The hope would be that this would have the benefits of both algorithms and provide better results. After this we will start looking into data-driven methods and Speeded Up Robust Feature (SURF) comparisons.

References

[1] A. Hampapur, K. Hyun and R. Bolle, "Comparison of Sequence Matching Techniques for Video Copy Detection." *Storage and Retrieval for Media Databases 2002*, 2001.

[2] O. Chum, J. Philbin, M. Isard and A. Zisserman, "Scalable near Identical Image and Shot Detection." *Proceedings of the 6th ACM International Conference on Image and Video Retrieval - CIVR 07*, 2007

[3] P. Mork, B. Li, E. Chang and J. Cho, "Indexing Tamper Resistant Features for Image Copy Detection." 1999

[4] A. Hampapur, and R.M. Bolle. "Comparison of Distance Measures for Video Copy Detection." *IEEE International Conference on Multimedia and Expo*, 2001. ICME 2001., 2001.

[5] Bovik, Al. "9.1 - Image and Video Indexing and Retrieval." Handbook of Image Video Processing, Elsevier-Academic Press, 2005, pp. 687–703.

[6] Aradhya, V.n. Manjunath, and M.s. Pavithra. "A Comprehensive of Transforms, Gabor Filter and k-Means Clustering for Text Detection in Images and Video." Applied Computing and Informatics, vol. 12, no. 2, 2016, pp. 109–116.

[7] Mathworks. "Iimsegkmeans." Mathworks, www.mathworks.com/help/images/ref/imsegkmeans.html

[8] Lu, Xu, Li, Chang, and Wang. "A Robust Image Copy Detection Method Based OnFeature Extraction Algorithm." *Journal of Information Hiding and Multimedia Signal Processing*, vol. 7, no. 5, September 2016

[9] Serra, Giuseppe, et al. "Covariance of Covariance Features for Image Classification." *Proceedings of International Conference on Multimedia Retrieval - ICMR '14*, 2014.

[10] Dornaika, Fadi, et al. "Handbook of Neural Computation." 2017, pp. 591–606.