Indexing Tamper Resistant Features for Image Copy Detection

Peter Mork[†], Beitao Li[‡], Edward Chang[‡], Junghoo Cho[†], Chen Li[†], and James Wang[†*]

Abstract

In this paper we present the *image* copy detection service we have built. We call this service RIME, for Replicated IMage dEtector. Given an image registered by its creator or distributor, RIME checks if *near-replicas* of the image (i.e., copies of the image that may have gone through alterations) exist on the Internet and returns a list of *suspect* URLs. The core technologies that the RIME project develops are effective image characterization for copy detection, and efficient image indexing for finding images with similar characteristics.

1 Introduction

Advances in the Internet and World-Wide Web technology have led to the growth of the dissemination of digital media at an unprecedent pace. This has made copyright enforcement more difficult [3], which has deterred some content providers from placing digital media on-line.

In our demonstration, we plan to show an *image* copy detection service that we have been building. We call this service RIME, for Replicated IMage dEtector [1, 2]. Given an image registered by its creator or distributor, RIME checks if *near-replicas* of the image (i.e., copies of the image that may have gone through alterations) exist on the Internet and returns a list of *suspect* URLs. The creator or distributor of the image can then check if the suspect images are indeed unlawful copies.

The detection of image copies faces two conflicting performance requirements: *accuracy* and *speed*. On the one hand, it aims at achieving high *precision* and high *recall*, which often means high computational cost. On the other hand, it requires short processing time so that more images can be compared given limited time and computing resources. To meet these performance requirements, RIME faces two design challenges: First, it needs an effective feature vector that can tell apart different images. Second, it needs an efficient indexing scheme that can find the suspect images accurately and quickly (i.e., with the minimum number of IOs). The contribution of our research project lies in two areas:

1: We devise effective image characterization for copy detection, and

2: We propose efficient image indexing for finding images with similar characteristics.

In characterizing images for copy detection, the feature vector must be able to survive subterfuge attacks, including image format conversion (e.g., from JPEG to GIF and vice versa), resampling, requantization, geometric transformations (e.g., translation, scaling, and rotation) and alterations (e.g., adding a border or a frame). Our demo shows that the new feature vector that we use is more tamper resistant than the conventional feature vectors and can detect unlawful copies effectively. Our demo also shows that a novel indexing scheme we employ [8] can index $500,000$ high-dimensional Internet images characteristized by high-dimensional feature vectors efficiently.

1.1 Related Work

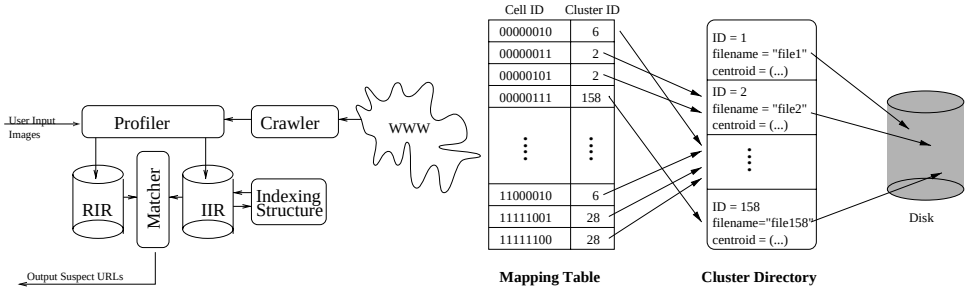
Color histograms have been widely used to index images in large image databases. Color histograms, however, do not contain spatial information about the colors. Thus, color histograms are inappropriate for image copy detection.

Several methods have been proposed to remedy the shortcomings of color histograms. Pass and Zabih [9, 10] developed the *color coherence vector* (CCV) method, which partitions each color into a coherent and an incoherent portion. Hsu et al. [4] selected a set of dominant colors from an image and partitioned the image into rectangular regions, each containing one color. Smith and Chang [11] employed histogram back-projection [12] to partition the image into regions. Huang et al. [7, 6] tried to encode the spatial correlation of color-bin pairs by the "color correlogram."

2 Architecture

Figure 1(a) shows the architecture of the RIME system. As shown in the figure, RIME receives registered images from users, profiles the images, and finds matching ones in its repository. If matches are found, RIME returns a list

*† Computer Science Department, Stanford University. ‡ Elec. & Computer Engineering, U.C. Santa Barbara.



(a) RIME's Components (b) The Indexer

Figure 1: Architecture.

of suspect URLs. To build the repository that stores the image profiles, RIME crawls the web periodically, profiles newly discovered images, and then indexes them based on their features.

RIME consists of the following six components.

- The Crawler crawls images from the WWW.
- The Profiler extracts feature vectors from images, either user input images, or images from the crawler.
- The Registered Image Repository (RIR) contains the feature vectors and URLs of the images registered by the owner or distributor for copy detection. RIME performs copy detection periodically (e.g., after a new web crawl) for these registered images.
- The Internet Image Repository (IIR) stores the feature vectors and URLs of the crawled web images. Notice that RIR and IIR may be two logical databases on top of one physical database.
- The Indexer indexes the feature vectors in a high-dimensional structure for efficient searching and storage.
- The Matcher performs nearest-neighbor search in the indexing structure to return the suspect URLs of a given query image.

In the remainder of the section, we briefly describe two key components where we have novel contributions: the indexer and the profiler.

2.1 The Indexer

RIME uses an indexing structure to support efficient similarity search. Similar images are grouped into clusters so that we can find the similar images of a query image by retriving the cluster where the query image resides. The indexer is built in two steps (we discuss the details in [8]):

1: We run a clustering algorithm to group similar objects into clusters.

2: We store each cluster sequentially on disk.

Figure 1(b) shows that the indexing structure includes two parts: a *cluster directory* and a *mapping table*. The cluster directory keeps track of the information about all the clusters, and the mapping table is maintained to map a cell to the cluster where the cell resides.

All the objects in a cluster are stored in sequential blocks on disk, so that these objects can be retrieved by one efficient sequential IO. The cluster directory records the information about all the clusters, such as the cluster ID, the name of the file that stores the cluster and a flag indicating whether the cluster is an outlier cluster. The cluster's centroid, which is the center of all the cells in the cluster, is also stored. With the information in the cluster directory, the objects in a cluster can be retrieved from disk once we know the ID of the cluster.

The mapping table is used to support fast lookup from a cell to the cluster where the cell resides. Given an image, RIME follows the mapping table to return the near-by clusters of the query image. Experiments showed that the indexer typically can achieve 90% recall after performing just a few sequential IOs (to retrieve a few near-by clusters) [8]. It also provides a way to incorporate a user's relevant feedback to perform query refinement by moving the subsequent search towards a more relevant cluster.

2.2 The Profiler

Our approach attempts to incorporate the spatial information of the colors into a color-based feature vector. By introducing the variances of colors into a color histogram, we compose the Variance-Augmented Color Histogram (VACH). The use of variance can be traced back to the seven moments method proposed by Hu [5], who used the variances of colors as one of the seven moments to do shape analysis. Using the variance also circumvents the algorithmically complex problem of segmenting the image into homogeneous color regions.

Color variances are a reasonable indicator of the global distribution of colors; they provide a decent approximation of which colors are more concentrated. In addition, the variance does not depend on the exact position of the pixels. This makes the color variance robust despite a variety of image modifications.

The VACH is also algorithmically simple. It can be computed in a single pass over the image, making the computing time comparable to a color histogram. The VACH is especially effective at finding the similarity between an image and the same image with an additional border because it weights color regions with high variance less than those with low variance. VACH can also serve as a general-purpose feature vector in the image retrieval system.

The basic VACH feature vector measures the ratio between intensity and variance. Each RGB pixel is mapped to one of 64 colors based on the 2 most significant bits of each scale (2 bits $\times$ 3 scales = 6 bits). A color histogram measures just the intensity of each of these colors. The VACH divides this value by the variance of that color. This weights concentrated regions of color more highly, since they will have a smaller variance.

For experimental purposes, we extracted both a conjoint and disjoint VACH. The conjoint VACH is the ratio a color's intensity to the sum of the variance along the X and Y axes. The disjoint VACH is the sum of the ratios of intensity to X-variance and intensity to Y-variance.

Color Histogram for color $i = Intensity(i)$

Conjoint VACH for color $i = \frac{Intensity(i)}{VarX(i)+VarY(i)}$

Disjoint VACH for color $i = \frac{Intensity(i)}{VarX(i)} + \frac{Intensity(i)}{VarY(i)}$

Since the total variance in an image will depend on its size, images crawled from the web are scaled to a uniform 200 by 200 pixels before the feature vectors are extracted. Another alternative would be to scale the intensity and variances by the total size of the image. The first collection of experiments we ran was on a controlled collection of images, each of which was the same size, so scaling was not required.

3 Experiments

We began with a collection of $3,000$ images. (At the end of this section we show some preliminary results with half a million images.) This size was chosen to facilitate an exhaustive comparison of a single modified image with every unmodified image. A larger database would have resulted in prohibitive search time, and a smaller database would not have been sufficiently large (and diverse).

A collection of three feature vectors was extracted from each image. The first feature vector was a simple color histogram. The conjoint and disjoint VACH were also calculated. A database was constructed containing all three feature vectors for each of the original $3,000$ (unmodified) images. In practice a single feature vector is selected. Each image requires 512 bytes of storage for the feature vector (assuming 8-byte double-precision values are used): $64 \times 8 = 512$. It may not be necessary to maintain this precision since each value in the feature vector is close to 1.

A random sample of 100 images was then chosen. Each of these images was modified in some way (e.g., a border was added, the image was cropped, etc.). The feature vectors of the modified images were then calculated. These modifications were allowed to affect the total size of the image. Cropping decreased image size; framing increased it. Bordering, which included both cropping and framing, did not affect the size of the image.

For each modified image (the query image), the Euclidean distance between its feature vector and the feature vector for each of the $3,000$ original images was calculated. This result was sorted and the location (λ) of the image corresponding to the query image was determined (i.e., the image that was modified to produce the query image). Thus, a k-nearest neighbor search of the original database would return the original image provided k was greater than or equal to λ.

This process was repeated for each of the 100 random images under a number of potential modifications. The cumulative accuracy for increasing values of k was thereby determined. In practice a good feature vector would have near perfect accuracy for very small values of k. This would allow a k-nearest neighbor search to be performed with an excellent chance of returning modified images, if they existed. The interesting values of k are those that are extremely small. Given a large database (say 1,000,000 images) a nearest-neighbor search for the closest 0.1% of the images would return 1,000 images. For our much smaller database, this corresponds to the 3 nearest neighbors.

The result of a k-nearest neighbor search could be scanned visually for actual duplicates (stressing the need for an extremely small value for k). Note that a perfect feature vector would not be fooled by the modifications, and λ would consistently be equal to 1 (i.e., the original would be the most similar image to the query image).

Modification Styles (Figures 2,3,4):

The first collection of tests involved a variety of modifications to the outermost 10 pixels of each image. The modifications performed were:

1: Bordering: Replace the outermost pixels with a black border. The image does not change in size, but 19.1% of the image is converted to black.

2: Framing: A 10 pixel black frame is added to the image. The image thus increases in size by 21.0%.

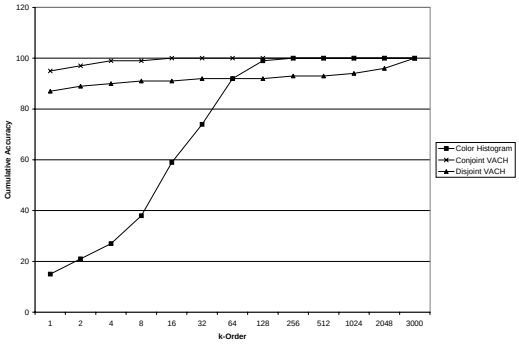


Figure 2: Effect of bordering

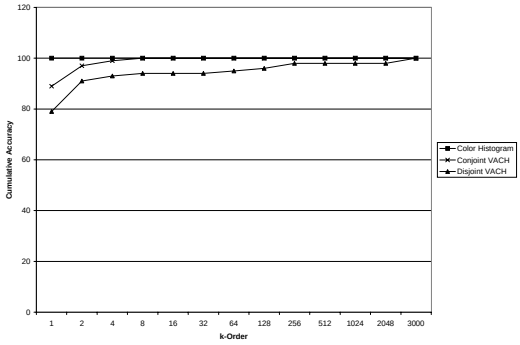


Figure 3: Effect of cropping

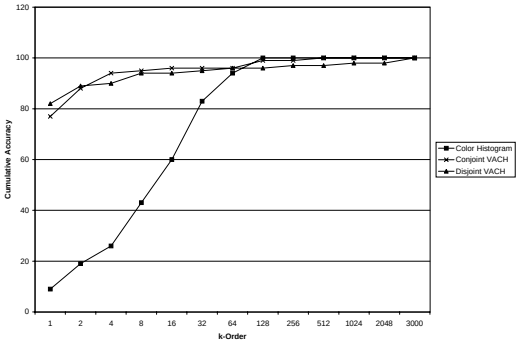


Figure 4: Effect of framing

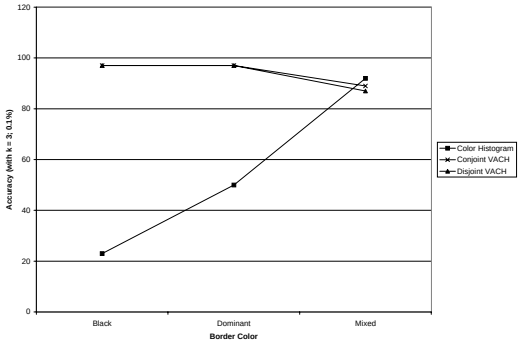


Figure 5: Effect of changing border colors

3: Cropping: The outermost 10 pixels are completely removed from the image. The image decreases in size by 19.1%.

The conjoint VACH is effective at detecting both borders and frames. The color histogram is better at detecting cropping, but the VACH is quite effective as well. A very narrow nearest-neighbor search would be able to detect most copies in this database.

Effect of Color Choice (Figure 5):

Under the assumption that a black border might not be aesthetically pleasing, the next pair of tests changed the 10 pixel black border to a:

1: Dominant Border: The single most dominant color (of the 64) present in the image.

2: Mixed Border: An even mixture of the 5 most dominant colors present in the image.

A single-color border can easily be detected. It appears that as more colors are added to the border, the efficacy of the VACH drops off. It should be noted, however, that a uniform mixture of pixels that may vary widely in color, is probably not a very likely modification. It is more reasonable to assume that the graphic designer will choose a more or less solid border. Since numerous RGB colors are mapped to a single color, there can be a fair amount of variety in a mono-chromatic border.

Effect of Border Size (Figure 6):

The next collection of tests altered the size of the border in order to determine whether or not the VACH feature vectors could deal with extremely large borders. The smallest border (5 pixels) corresponds to 9.8% of the image. A 10 pixel border corresponds to 19.1% of the image, a 15 pixel border to 27.9% of the image and a 20 pixel border to 36.2%. Beyond this size of a border, too much of the image was destroyed to merit testing larger borders. The color histogram is quickly thwarted by the presence of a border. The VACH is able to detect borders with an approximately linear loss of accuracy (based on the percentage of the original image that was replaced by a border).

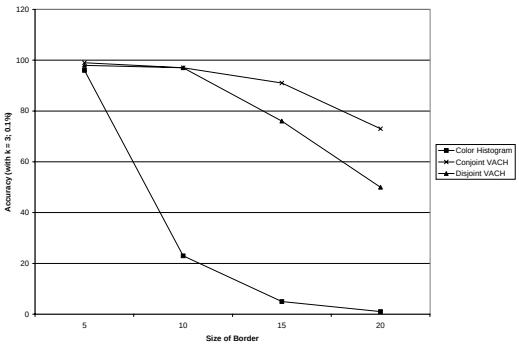


Figure 6: Effect of changing border sizes

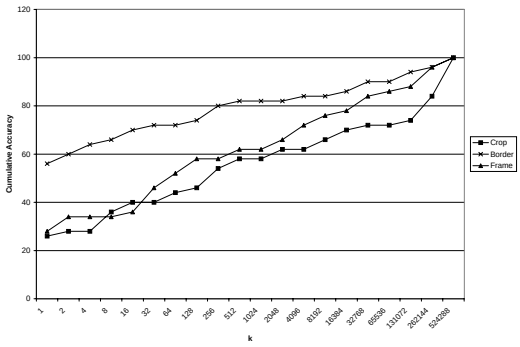


Figure 7: Results on a large database

Very large database (Figure 7):

The first experiment was then repeated on a collection of $500,000$ images crawled from the WWW. In this case, the modified image was added to the database. The query image was the unmodified original. Figure 7 indicates the results of the conjoined VACH after discarding all exact matches. These results were less promising for cropping and framing. Bordering, which is the logical composite of cropping and framing, can be detected with a high degree of accuracy. A search of the most similar 0.1% of the database has roughly an 80% chance of detecting the modified image. Even a much more restrictive search has a better than 60% accuracy.

4 Conclusion

There are at least two merits that our demo shows:

- Our VACH feature appears to be tamper resistant and is good for image copy detection. When attempting to detect images modified with a border, experimental results indicate that a color histogram returns the correct closest match less than 20% of the time whereas the conjoined VACH has better than 80% accuracy . The advantage of the VACH over the color histogram is sustained even in a large database.
- Our indexer can assist finding near-replicas of an image fast with high accuracy.

References

[1] E. Chang, C. Li, J. Wang, P. Mork, and G. Wiederhold. Searching near-replicas of images via clustering. *Proc. of SPIE Symposium of Voice, Video, and Data Communications, Multimedia Storage and Archiving Systems VI, Boston, MA*, September 1999.

[2] E. Chang, J. Z. Wang, C. Li, and G. Wiederhold. Rime: A replicated image detector for the world-wide web. *Proc. of SPIE Symposium of Voice, Video, and Data Communications, Multimedia Storage and Archiving Systems VI, Boston, MA*, November 1998.

[3] H. Garcia-Molina, S. Ketchpel, and N. Shivakumar. Safeguarding and charging for information on the internet. *Proceedings of ICDE*, 1998.

[4] W. Hsu, T. S. Chua, and H. K. Pung. An integrated color-spatial approach to content-based image retrieval. *ACM Multimedia Conference*, pages 305–313, 1995.

[5] M. Hu. Visual pattern recognition by moment invariants. *IRE Trans. Information Theory ,IT-8, 2*, pages 179–187, 1962.

[6] J. Huang, S. R. Kumar, and M. Mitra. Combining supervised learning with color correlograms for content-based image retrieval. *ACM Multimedia Conference*, pages 325–334, 1997.

[7] J. Huang, S. R. Smith, M. Mitra, W.-J. Zhu, and R. Zabih. Image indexing using color correlograms. *Proc. IEEE Comp. Soc. Conf. Comp. Vis. and Patt. Rec.*, pages 762–768, 1997.

[8] C. Li, E. Chang, H. Garcia-Molina, J. Wang, and G. Wiederhold. Clindex: Clustering for similarity queries in high-dimensional spaces (extended version). *Stanford Technical Report SIDL-WP-1998-0100*, Feb. 1999.

[9] G. Pass and R. Zabih. Histogram refinement for content-based image retrieval. *IEEE Workshop on Applications of Computer Vision*, pages 96–102, 1996.

[10] G. Pass, R. Zabih, and J. Miller. Comparing images using color coherence vectors. *Proceedings of the Fourth ACM Multimedia Conference*, pages 65–73, 1996.

[11] J. Smith and S.-F. Chang. Tools and techniques for color image retrieval. *Volume 2670 of SPIE Proceedings*, pages 1630–1639, 1996.

[12] M. Swain and D. Ballard. Color indexing. *International Journal of Computer Vision*, pages 7(1):11–32, 1991.