Enhanced Perceptual Distance Functions and Indexing for Near-Replica Image Recognition

Arun Qamra

Department of Computer Science

Yan Meng, Edward Y. Chang

Electrical & Computer Engineering Department

University of California Santa Barbara

Santa Barbara, CA 93106

arun@cs.ucsb.edu, echang@ece.ucsb.edu

Abstract

The proliferation of digital images, and the widespread distribution of digital data that has been made possible by the Internet, has increased problems associated with copyright infringement on digital images. Watermarking schemes have been proposed to safeguard copyrighted images, but watermarks are vulnerable to image processing and geometric distortions, and may not be very effective. Thus, the content-based detection of pirated images has become an important application. In this paper, we discuss two important aspects of such a near-replica detection system: distance functions for similarity measurement, and scalability. We extend our previous work on perceptual distance functions, which proposed the Dynamic Partial Function (DPF), and present enhanced techniques that overcome limitations of DPF. These techniques include the Thresholding, Sampling and Weighting schemes. Experimental evaluations show superior

performance compared to DPF and other distance functions. We then address the issue of using these perceptual distance functions to efficiently detect replicas in large image datasets. The problem of indexing is made challenging by the high-dimensionality and the non-metric nature of the distance functions. We propose using Locality Sensitive Hashing (LSH) to index images while using the above perceptual distance functions, and demonstrate good performance through empirical studies on a very large database of diverse images.

Keywords: *Indexing methods, Image databases, Image Retrieval*

1. Introduction

Advances in the Internet and World-Wide Web technology have led to exponential proliferation of information, disseminated at an unprecedented pace. However, the resulting widespread distribution of data (text, music, imagery and video) has increased the problems associated with copyright infringement.

To safeguard copyrighted images and trademarks, many watermark schemes have been proposed [1]. Though useful, a watermark is vulnerable to image processing, geometric distortions, and subversive attacks [2]. Therefore, we propose a content-based scheme to safeguard trademarks and copyrighted images. Given an image registered by its creator or distributor, the system checks to find out whether near-replicas of the image exist on the Internet and reports a list of suspect URLs. The rightful owner of the images can then check the suspect images to determine whether they are indeed unauthorized copies.

To build such a system, and to achieve efficient and high detection accuracy, the following three issues need to be addressed. First, representative and informative features that represent an image's perceptual features, must be extracted for each image. Second, an effective method must be employed to accurately quantify the perceptual similarity between images. Third, the system must be scalable, and must be capable of efficiently searching near-replicas in very large databases. Our previous work [11] explored the first two issues. We identified perceptual fea-

tures and demonstrated their appropriateness for the problem of image copy detection through extensive empirical studies. We also presented the Dynamic Partial Function (DPF) to measure perceptual similarity between images, and showed that it performs better than several traditional, widely used distance functions. In this paper, we extend our work on similarity measures, address drawbacks of the DPF distance measure, and present enhanced perceptual distance functions. We then address the third issue, that of efficiently searching large databases for image near-replicas.

An image can be pirated in many ways, including format conversion (e.g, from JPEG to GIF), resampling, scaling, changing orientation, cropping, framing, colorizing, etc. Yet, it would be unlikely for a pirate to change the image substantially and thereby lose its perceptual similarity to the original. Thus, the features used to represent each image should be such that they represent the perceptual characteristics of the image. We identified an appropriate set of features [11], which we will summarize in Section 2.

To measure perceptual similarity between images, many distance functions such as Minkowski-like metrics [7] (including Euclidean, L1, and Fractional distance functions [8]), Histogram Cosine distance [9], and Fuzzy logic [10], have been used. However, in [11], we showed that these methods frequently overlook obviously similar images and hence are not adequate for recognizing near-replica images. We proposed the Dynamic Partial Function (DPF) for characterizing similarity between images. Unlike other current feature-selection methods [13, 14, 15, 16] that select the same features for all the images, DPF dynamically activates features (with minimum differences) in a pair-wise fashion. By searching through an effective subspace for comparable images, DPF has been shown to perform significantly better than Minkowski-like metrics. The superior performance of DPF can be understood from ideas in cognitive psychology, a discussion of which can be found in Section 3.1 and [11].

Though achieving high accuracy, DPF suffers from the limitation of one-size-fits-all: it imposes the same number of dimensions on all pairwise comparisons. Ideally, the optimal

number of features should be determined in a pairwise fashion. A simple example will illustrate this point. Suppose we would like to find objects similar to a red bell-pepper. Among several possibilities, a strawberry and a green bell-pepper may be chosen. The first pair (the strawberry and the red bell-pepper) are similar in color (red), and the second pair (the green and the red bell-peppers) are similar in shape and texture. Thus we see that the number of similar features might differ for different image pairs, as might the different types of features. Therefore, using a fixed number of features to capture various characteristics of different image pairs may not yield optimal results. Image pairs with different characteristics may have a different number of features contributing to similarity.

Based on the above observations, we propose a number of schemes for enhancing DPF [12], including *Thresholding*, *Sampling*, *Weighting*, and some hybrid schemes of these three basic approaches. These methods adaptively activate different numbers of features for different image pairs, which can better capture the different characteristics of different image pairs. Our experimental results show that all these methods can enhance DPF's performance in recognizing near-replica images. Among them, the Weighted-Sampling-Thresholding method can most accurately capture pairwise image similarity. For example, at 80% recall rate, it can improve DPF's precision by as much as 16%, outperforming L1 distance function by 43%.

In addition to feature extraction and distance measurement, the third critical issue that needs to be addressed when building a useful image replica detection system, is the ability to efficiently search through a large number of images. With the widespread proliferation of digital equipment(cameras and scanners, etc.) and fast dissemination through the Internet, there has been an exponential increase in the number of digitized images available on the Web. A sequential scan through such a dataset is inefficient and not scalable. We need efficient indexing schemes that allow replica detection by searching just a small subset of the entire dataset. The large number of features describing each image, and the non-metric nature of perceptual similarity, make indexing a challenging problem. In this work, we study the use of Locality Sensitive Hashing(LSH) [24] to index images, and for use with the perceptual distance

functions. In LSH, a hash is created for each point (each image is represented as a point in a high-dimensional space) in the database such that the probability of collision is much higher for points that are close to each other than for those that are far apart. This indexing technique has been used successfully for some high-dimensional applications, as in [27] and [28]. Here, we demonstrate that this is a promising indexing technique for image piracy detection.

The rest of this paper is organized as follows: In Section 2, we describe the features used for representing each image. We discuss distance functions for measuring image similarity in Section 3, and address the problem of indexing images for replica recognition in Section 4. Section 5.1 describes the testbed we used for experimental evaluations, and Section 5 presents the results of our empirical study. We offer concluding remarks in Section 6.

2. Feature Selection and Extraction

To describe images, it is important to select a set of features that can adequately represent the perceptual characteristics of each image. We have addressed this problem in previous work [32, 11], which employs a multi-resolution feature set that includes color and texture features. A number of studies [32, 33, 34, 11] have shown this set of features to be robust and effective for image retrieval. Hence we use this feature set in our current work. With this feature set, we achieved acceptable recall even with the Euclidean distance function, and recall was drastically improved when used with perceptual distance functions. Below, we describe this feature set. For further details, please consult [11].

As some researchers [17] have suggested, human perception works in a multi-resolution fashion. In our study, we employ a similar multi-resolution image representation scheme and describe each image with a 279-dimensional feature vector, including three types of perceptual features: color, texture and shape. Specifically, we use 147 global color, texture, and shape features, and 132 color-based local features to characterize each image. Each feature component is then normalized to be in the range of $[0, 1]$ using Gaussian normalization [18].

Color. Color is one of the most important elements for human vision [21]. To capture the perceptual properties of images, it is desirable to use a color space in which small changes in the space will correspond to small perceptual differences. In this study, we employ HSV color space, which has been shown to be capable of representing human color perception substantially [21].

Research [17] shows that, throughout the visible spectrum (roughly 380nm to 780nm), colors that can be named by all cultures are generally limited to eleven—namely *black, white, red, green, blue, brown, yellow, pink, purple, gray,* and *orange.* We use these "culture-colors" to divide the entire color space into 11 bins, each corresponding to a culture-color. Since all colors inside each bin are referred to be the same name, colors falling in one bin can be considered to be perceptually similar. We want to stress here that human perception and vision is quite complex any such division of the color space can only be an approximation. However, we believe this is a reasonable approximation since words in languages can be expected to reflect how humans perceive color. Also, experimental studies in [11] have found such a feature representation to show good performance for the replica detection problem.

We thus divide HSV space into 11 culture-color bins. Within each bin, we extract features in the following way. At the coarsest resolution, we characterize color using a color histogram. At finer resolutions, we record eight features for each color. These eight features are color means (in H, S and V channels), color variance (in H, S and V channels) and two shape characteristics: elongation and spreadness [2]. Color elongation characterizes the shape of a color, and color spreadness characterizes how that color scatters within an image. Altogether, we extracted 99 color features.

Texture. Texture is another important element for human vision. As shown by many studies [3, 4, 5, 6], characterizing texture features in terms of structuredness, orientation, and scale (coarseness) fits well with models of human perception. There is a great variety of texture features available to system designers. For ease of computation, we choose a discrete wavelet transformation (DWT) using quadrature mirror filters [5].

Each wavelet decomposition on a 2-D image yields four subimages: a $(\frac{1}{2} \times \frac{1}{2})$ scaled-down version of the input image and its wavelets in three orientations: horizontal, vertical, and diagonal. Decomposing the scaled-down image further, we obtain the tree-structured or wavelet packet decomposition. The wavelet image decomposition provides a representation that is easy to interpret. Every subimage contains information of a specific scale and orientation and also retains spatial information. We obtain texture features from subimages of four scales and three orientations. At each scale and each orientation, we extract four global texture features: energy mean, energy variance, elongation, and spreadness. In all, we obtain 48 texture features.

Local features. Since observers are often interested in the spatial layout of the colors, textures, and shapes of an image, or because they may be interested only in certain objects, using global features alone can be inadequate [10]. Therefore, we also extract 132 local features to capture some local information. We first extract 11 subimages from each image, according to the culture colors. Each of these sub-images contain only one culture color. Next, from each subimage, we extract texture energy features in three orientations (horizontal, vertical, and diagonal) and four scales, and normalize the texture energy features with the color areas [19].

3. Distance Functions for Similarity Measurement

A distance function is used to quantify (dis)similarity between two instances (images). Each image can be represented by a p dimensional feature vector $(x_1, x_2, \ldots, x_p)$, where each dimension represents an extracted feature. In the following, we define the feature distance in feature channel i as $\Delta d_i = |x_i - y_i|$,for $i = 1, 2, \ldots, p$. Since we normalize the feature values to be in the range of [0, 1], the feature distance is also in the range of [0, 1].

To measure similarity, Minkowski-like distance functions [7] are commonly used [5, 8, 18]. In these metrics, the distance between any two images X and Y is defined as

$$D(X, Y) = \left(\sum_{i=1}^{p} \Delta {d_i}^r \right)^{\frac{1}{r}}. \quad (1)$$

When we set $r = 1$, the above distance is the City-block or L1 distance. When we set $r = 2$, it is the Euclidean or L2 distance. And when r is a fraction, it defines a fractional function [8]. In general, a Minkowski-like distance function considers all features for similarity measurement.

However, extensive study [11] has shown that the above assumption is questionable. Instead, in [11], the Dynamic Partial Function(DPF) was proposed. Here, we recapitulate DPF, discuss cognitive psychology that validates the approach, and identify its limitations. We then propose advanced distance functions that address the limitations of DPF.

3.1 Dynamic Partial Function(DPF)

If we define the set

$$\Delta_m = \text{The smallest } m \ \Delta d_i\text{'s of } (\Delta d_1, \Delta d_2, \ldots, \Delta d_p),$$

then the dynamic partial distance between two images X and Y, is defined as

$$D(X, Y) = \left(\sum_{\Delta d_i \in \Delta_m} {\Delta d_i}^r \right)^{\frac{1}{r}}, \tag{2}$$

where m and r are tunable parameters. When $m = p$, DPF degenerates to a Minkowski-like metric. When $m < p$, it counts the smallest m feature distances between two images, and hence uses only the similar aspects of the images to measure similarity. Thus, DPF activates different features for different image pairs. The activated features are those with minimum differences—those that provide coherence between the objects. If there is sufficient coherence (because a sufficient number of features are similar), then the distance is small and the images can be inferred to be similar. The parameter m can be selected by training a large image dataset and computing an "on-average optimal" m value. For r, commonly used values are 1 and 2, as in the case of Minkowski measures.

The dynamic and partial approach used by DPF is strongly grounded in ideas from research in cognitive psychology. Studies have shown [31, 20] that the Minkowski approach, where all

aspects are employed to measure similarity, does not model human perception. Goldstone [30] explains that similarity is the process that determines the respects for measuring similarity. In other words, similarity is measured by selecting the appropriate respects, at the time of comparison. Human perception infers overall similarity based on the aspects that are similar among the compared objects rather than based on the dissimilar aspects. For example, we are likely to find a red car similar to a red truck based on the color, and also find a red car similar to a black car based on shape. As we see here, similarity in both cases is measured based on a different set of aspects—the ones that are most similar for the compared pair. DPF works in a similar way. It measures similarity based on the subset of features that are most similar for a given pair of compared objects. Hence it models human perception more accurately.

For further details about the Dynamic Partial Function, please refer to [11], which presents extensive experimental studies that validate the effectiveness of the DPF approach. Also, the performance of DPF has been shown to be superior to that of widely-used distance functions such as the fractional function, histogram cosine distance function, and the Euclidean and Manhattan distance functions. For instance, for a recall of 80%, the retrieval precision of DPF is shown to be 84% for the test dataset used. In comparison, the histogram cosine function and the best fractional function yield precisions of 79% and 25% respectively; Euclidean and Manhattan distance functions yield precisions of 50% and 70% respectively. Since studies in [11] have already shown performance of DPF to be superior to other distance functions, in this paper, we compare the performance of enhanced perceptual distance functions only with DPF.

3.2 DPF Enhancements

Even though DPF works much better than Minkowski-like metrics, it is limited by the fact that the number of features used, m is fixed. This is a limited model since the similarity of different pairs of objects may depend on different number of features. Using a fixed number of features may include features that contribute some noise or might overlook some features that actually

contribute to similarity.

In this work, we devise techniques to overcome these limitations. Below, we propose three basic remedies—Thresholding, Sampling, and Weighting—for adaptively selecting m. We also discuss hybrid schemes that combine these basic remedies. We present experimental results in Section 5 for evaluation.

3.2.1 Thresholding Method

The Thresholding method activates different numbers of "similar" features for measuring similarity by employing a threshold θ. After comparing two images in every feature, a threshold θ is introduced to categorize their p features into a similar feature-set and a dissimilar feature-set. The features that have a difference smaller than θ belong to the similar feature-set, and the other features belong to the dissimilar set. We compute the distance of the image pair using only the features in the similar feature-set. More specifically, let A_θ denote the similar feature-set. The number of similar features can be written as $|A_\theta|$. The Thresholding distance of image pair (X, Y) is then defined as

$$D(X,Y) = \left(\frac{1}{|A_\theta|} \sum_{\Delta d_i \in A_\theta} {\Delta d_i}^r \right)^{\frac{1}{r}}. \tag{3}$$

A proper θ must be chosen so that the resulting distance function can work effectively. The procedure for setting θ is similar to that for setting m: we experiment with different θ values on a training dataset, then select the θ that can achieve the on-average lowest missing rate. Even though the Thresholding method is not tune-free, it is more adaptive to the characteristics of different image pairs.

3.2.2 Sampling Method

For measuring similarity, DPF uses an "on-average optimal" m, and Thresholding uses an "on-average optimal" θ. Ideally, the optimal m and the optimal θ should be determined in a pairwise

fashion. Our sampling method aims to achieve this goal for DPF and for Thresholding.

Our Sampling-DPF adaptively finds the optimal m for each image pair. (Once Sampling-DPF is presented, Sampling-Thresholding simply replaces m_n with θ_n.) The main idea of Sampling-DPF is to try different m settings simultaneously, and to anticipate that some of the sampling points can be near the optimal m. Suppose we sample N settings of m, denoted as m_n, $n = 1, \cdots, N$. The sampling DPF method ranks the similarity of all images in the dataset (denoted as Φ) with respect to query X using m_n. This step is repeated N times for N different m_n sampling points[1]. At each sampling m_n, a ranking list, denoted as $R_n(\Phi, X)$, is generated. Finally, the sampling method aggregates N ranking lists and produces the final ranking, denoted as $R_r(\Phi, X)$.

There are two issues concerning the Sampling method: how frequently and where to sample, and how to aggregate the final ranking. Regarding sampling frequencies and sampling locations, our empirical study (see Section 5.2.3) will provide a rule of thumb. Regarding ranking, we suggest the following two methods:

1. Using the smallest rank as the final rank (SR).

The SR ranking relies on information at one sampling point that gives an image the best rank. For instance, suppose we have three sampling points, and image I_i ranks at these sampling points 7^{th}, 11^{th}, and 121^{st}, respectively. Its SR rank is $\min\{7, 11, 121\} = 7$.

2. Using the average distance (or rank) as the final distance (AR).

The AR approach takes all sampling points into consideration. The AR distance $D_{AR}(X, Y)$ of image pair (X, Y) is the average of the aggregated distance at all N sampling points, or

$$D_{AR}(X, Y) = \frac{1}{N} \sum_{n=1}^{N} \Big(\sum_{\Delta d_i \in \Delta_{m_n}} \Delta {d_i}^r \Big)^{\frac{1}{r}}. \tag{4}$$

For the Sampling-Thresholding method, its AR distance can be written as

[1]For efficiency, we first ranked N sampling points in ascending order. After computing feature distances for each image pair, we sorted them and incrementally computed and aggregated the distances under these samples in one pass.

$$D(X,Y) = \frac{1}{N}\sum_{n=1}^{N}(\frac{1}{|A_{\theta_n}|}\sum_{\Delta d_i \in A_{\theta_n}} \Delta {d_i}^r)^{\frac{1}{r}}. \quad (5)$$

3.2.3 Weighting Method

The Weighting method can be used with Sampling-DPF, Sampling-Thresholding, or other methods. In addition to setting different m or θ values for different image pairs, the Weighting method takes into consideration the fact that different types of features can have different degrees of importance for measuring similarity, and thus should be given different weights accordingly.

Let w_i denote the weight of the i^{th} similar feature. In the Weighted-Sampling-DPF method, the distance of image pair (X, Y) is defined as

$$D(X,Y) = \frac{1}{N}\sum_{n=1}^{N}(\sum_{\Delta d_i \in \Delta_{m_n}} w_i \Delta {d_i}^r)^{\frac{1}{r}}, \quad (6)$$

and in the Weighted-Sampling-Thresholding method, it is defined as

$$D(X,Y) = \frac{1}{N}\sum_{n=1}^{N}(\frac{1}{|A_{\theta_n}|}\sum_{\Delta d_i \in A_{\theta_n}} w_i \Delta {d_i}^r)^{\frac{1}{r}}. \quad (7)$$

To set the weight w_i for the i^{th} feature, we use the inverse of the standard deviation of all feature values in that feature *among similar images*, which has been shown to be effective in modeling feature relevance, and has been widely used in the image database community [5, 10, 18].

4. Indexing Images for Replica Detection

For a practically useful image replica detection system, the ability to efficiently search for replicas of a query image in a large database of images is crucial. With the large number of digital images available, the system may have to search through millions of images. A sequential scan through such a large dataset is not feasible. Thus, indexing of image databases is important,

so that the system can quickly find the relevant images from among the entire dataset. An ideal indexing scheme would allow quickly retrieving only all the nearest neighbors of the query image. However, indexing images is challenging [26] because of the high dimensionality of the feature vectors and because of the non-metric nature of the perceptual distance functions.

4.1. Locality Sensitive Hashing(LSH)

Locality Sensitive Hashing was proposed by Gionis et. al. [24] to index points for approximate nearest-neighbor search in a high-dimensional space. In this section, we summarize the technique. Please refer [24] and [29] for an elaborate description of the algorithm, and a theoretical treatment.

A locality-sensitive hashing function is defined as one for which the probability that two points will collide is closely related to the distance between them. The hash function creates a hash such that the probability of collision is much higher for points that are close to each other than for points that are far apart. In [24], the authors present a family of locality-sensitive hash functions for points embedded in the Hamming cube.

This family of functions can be applied for nearest-neighbor searches in the Euclidean space by mapping each point to a point in the Hamming cube. Let P be the set of points to be indexed, each represented as a d-dimensional vector, and let C be the largest coordinate in all points in P. Then the set P can be embedded into the Hamming cube H having dimension d' where $d' = Cd$, by representing each coordinate x as a sequence of x ones followed by $C - x$ zeroes.

The hash functions are defined by selecting a random subset I from $\{1, ..., d'\}$. The hash is created by the projection of each vector p on the coordinate set I. This is done by selecting from p, the bits corresponding to the dimensions in I. Hence the hash is a sequence of bits. Since the total number of hash buckets may be large, a second level hash is used to map the LSH buckets into a smaller hash table. This is done only for better implementation and does not change the properties of the LSH hash.

When querying the index structure, the hash for the query point is created and the corresponding hash bucket is retrieved. Near neighbors of the query point would be in this retrieved bucket with a high probability. Further processing is then done only on this retrieved bucket, and hence we need to visit only a small subset of the entire image database.

To increase accuracy of retrieval, we create l independent hash functions from the family of hash functions described above, and construct an independent index structure corresponding to each of the hash functions. Doing so reduces false negatives, since neighbors left out by one index may be retrieved through the other indexes. As an illustration, consider the following example. Suppose an index suffers from 40% false negatives. Using 5 such indexes that are created independently using different hash functions, we expect the fraction of false negatives to be reduced from 0.4 to 0.4^5 or around 0.01, since each index has independent errors. We will show in 5 that the value of l must be carefully set. A smaller l results in a high incidence of false negatives, but a high l may be counter-productive because of reading in the same points multiple times. Note that when a query is processed, l hash buckets are retrieved, one from each index.

4.2. Addressing Challenges to Indexing

There are two significant challenges raised by the problem of indexing images. The first arises because of the large number of features that describe each image. The well known *curse of dimensionality* makes it difficult to index points. LSH is able to index points in high dimensional search, but since it is a probabilistic scheme, it returns only the approximate neighborhood of the query point, and may miss some neighbors. But for a quick search, an approximate search is the only option and is acceptable for practical purposes. If higher recall is required, the number of LSH indices used can be increased, or a sequential scan can be performed.

The second challenge is that perceptual distance functions are non-metric in nature, as the work in this paper shows. Developing indexing methods for use with non-metric distance

measures is a hard problem. DynDex, a clustering based algorithm for indexing in non-metric spaces was introduced in [25]. With DynDex, frequent insertions or deletions may cause index quality to degrade, and when that happens, reindexing is required.

Since LSH allows finding approximate near neighbors in the Euclidean space, we cannot directly get perceptually similar points (ie. nearest neighbors using DPF or other perceptual distance functions). Hence we employ the following scheme. The index structure is used to retrieve a large Euclidean neighborhood of the query point such that most of the perceptually similar points are retrieved. Then the retrieved points are sorted using the advanced perceptual distance functions to select the most similar images. Thus, we use the LSH index structure for prefiltering and then select the best matches based on the perceptual distance functions. In Section 5.3.2, we study the size of the Euclidean neighborhood that will have to be retrieved for such a scheme, and the tradeoffs involved, between efficiency and effectiveness, when using this method.

Also, it is worth noting that LSH may be an appropriate indexing scheme for non-metric DPF-like functions because of the inherent partial nature of LSH. LSH works by selecting a small subset of dimensions of the embedded binary vector, which is somewhat analogous to activating just a subset of dimensions, as in DPF.

5. Empirical Study

We start this section by describing the testbed that we used for empirical evaluation. Then we present detailed experimental results. To test the effectiveness of the enhanced perceptual distance functions, we conducted experiments using DPF and each of the enhanced functions to evaluate if near-replicas of the query image could be retrieved with better precision/recall compared to DPF. Note that we present comparisons only with DPF, since DPF has already been shown in [11] to be superior to a variety of distance functions. Then, to evaluate whether these methods scale well to a large dataset, we tested the best method with a one-million-

image database. The next set of experiments was conducted, using the million-image dataset, to investigate the use of LSH and to find optimal LSH parameters for indexing in a near-replica detection system.

5.1 Experimental Testbed

To study the effectiveness of the proposed functions and methods, we require a testbed containing a large number of diverse images, including a variety of labeled original and pirated images. Unfortunately, no such experimental testbed is available, and for thorough evaluations we have to resort to an artificially constructed dataset. To construct an artificial but realistic testbed, we used a base set of original images, and performed a set of 40 transformations on each of these images. These 40 transformations span a wide variety of transformations that an image pirate is likely to make. Considering the variety of transformations performed, and the large number of diverse base images used, we believe that this artificial dataset is a good simulation of real-world test data, and can be used for a reliable evaluation of our methods.

To create the set of original images, we collected images from Corel image CDs and the Internet, and normalized them into two sizes: either 384×256 or 256×384. The following is a list of transformations performed on each of these images.

1.Colorizing. Each image is colorized through the red, green, and blue channels by 10% respectively.

2.Changing contrast. The intensity differences between the lighter and darker elements of each image are increased or decreased.

3.Cropping. We first symmetrically remove the outer borders of each image to reduce its size by 5%, 10%, 20%, and 30%, and then scale the cropped image back to its original size.

4.Despeckling. The amount of speckles within each image is reduced.

5.Downsampling. Each image is downsampled by seven different percentages: 10%, 20%, 30%, 40%, 50%, 70%, and 90%.

6.Flopping. We obtain the mirror images by reflecting the seed images along the horizontal direction.

7.Changing format. We obtain the GIF version of each JPEG image.

8.Framing. For each image, we produce four framed images, each with a randomly selected frame color. The frame covers 10 percent of the total area of the framed image.

9.Rotating. We rotate each image by 90^o, 180^o and 270^o.

10.Scaling

- *Scale up then down.* Each image is scaled up by 2, 4 and 8 times, and then scaled back to its original size.
- *Scale down then up.* Each image is scaled down by 2, 4 and 8 times, and then scaled back to its original size.

11.Changing saturation. We alter the saturation amplitude of each image by five percentages: 70%, 80%, 90% 110%, and 120%.

12.Changing intensity. We vary the intensity amplitude of each image by four percentages: 80%, 90%, 110%, and 120%.

Training dataset A training dataset is required by the enhancement schemes where a threshold is utilized. It is also used to obtain a suitable range of m or θ values for the sampling schemes so that similar and dissimilar images can be effectively separated. For this purpose we used 2000 randomly picked images and their 40 replicas, thus creating a training dataset with 82000 images.

Testing dataset To evaluate the effectiveness of our enhancement schemes, we constructed a testing dataset with images that did not appear in the training dataset. This testing dataset was composed of 20,000 randomly picked images, of which 500 were selected to be used as query images. A replica image set was then created for each query image using the 40 transformations, yielding an additional 20,000 images. The replica images were added to the test dataset to give a the total of 40,000 images.

Scalability and indexing test dataset A much larger dataset was created to test the scalability of the distance functions and to investigate LSH as an indexing scheme. This dataset, containing over a million images, was created by applying the 40 transformations to each of about 25,000 randomly picked images.

We also applied the distance functions and methods described here to a real-world situation. A software prototype using these methods was developed, and used to search Web-sites on the Internet for replicas of proprietary images owned by a certain company. This was done for each of $300,000$ images in a database of original images. However, because of copyright issues, we cannot provide much detail about this work and the results obtained. Snapshots of the software in action, and some brief information, are provided in Section 5.4.

5.2. Distance Functions

Here, we present the performance of DPF using its fixed "on-average optimal" m value, and then show experimental results for the enhanced distance functions.

5.2.1 DPF

In our mining work, we took Euclidean distance function as the baseline, and set $r = 2$ for DPF. To find the "on average optimal" m value, we trained DPF in the following procedure. We used our 82,000 training image dataset and the 2,000 seed-images as queries. For each query, we retrieved the top-40 most similar images and tallied how many of the 40 transformed images were missing. We computed the average missing rate for each similar image category (colorizing, changing contrast, cropping, despeckling, downsampling, flopping, changing format, framing, rotating, changing saturation, scaling, and changing intensity). The training results (Figure 1) show that the best m value is 189, where the on-average missing rate is the lowest.

But, as Figure 1 shows, when we went to individual m values, we found that for different types of transformed images, the optimal m values were not equal. For example, the best m

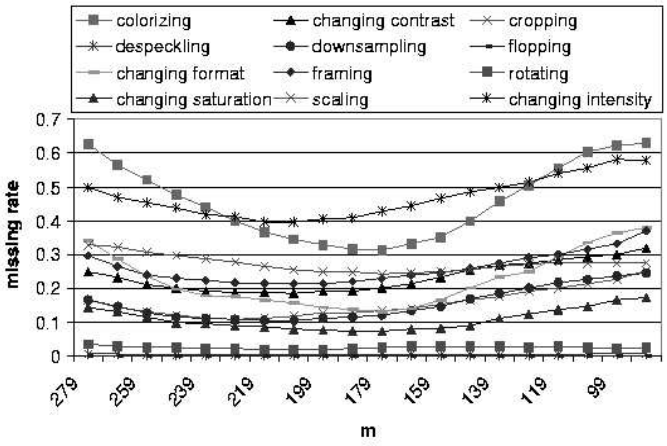


Figure 1: Predicting optimal m

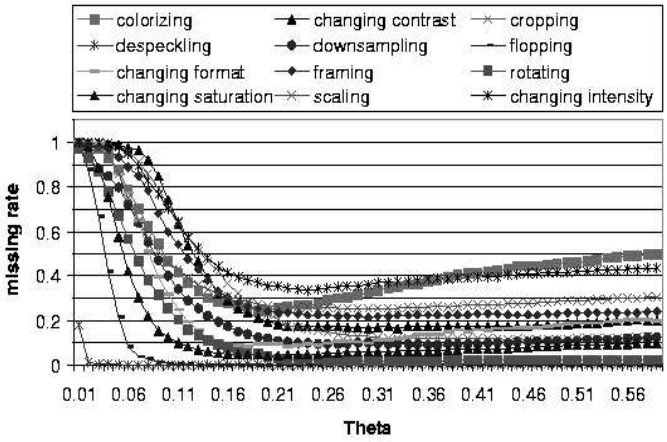
value is $m = 179$ for colorizing images, and 229 for cropping images. This means that using 189 as the best m value for colorizing images may involve 10 dissimilar features, which actually add noise and contribute to dissimilarity; using 189 as the best m value for cropping images might overlook 40 similar features that could contribute to similarity.

Through the above experiments, we learned that when using a fixed number of features, DPF faces difficulties in capturing various characteristics of different image pairs. Further enhancement is therefore desirable.

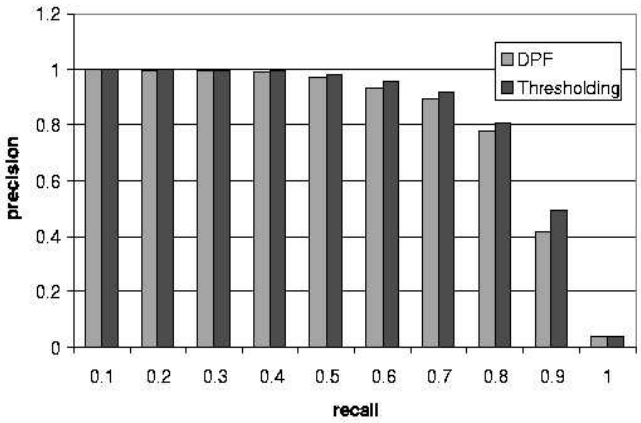
5.2.2 Thresholding Method

By employing a threshold θ to divide all features into similar and dissimilar categories, the Thresholding method sets different m values for different image pairs. To find the "on-average optimal" θ, we trained the distance function (Equation 3) using the same procedure as in Section 5.2.1. The training result (Figure 2(a)) shows that the optimal θ is ($\theta = 0.28$), at which value the on-average missing rate for all kinds of transformed images achieves the lowest rate. When a smaller θ ($\theta = 0.1$) is chosen, the missing rates get higher due to ignoring some features that could contribute to similarity. And using a larger θ ($\theta = 0.5$) also increases the missing rates, because it involves some noisy features that contribute to dissimilarity. Compared with the mining results of DPF (Figure 1), the missing rates of all types of transformed images are

reduced in the Thresholding method. For example, for colorizing images, the lowest missing rate for DPF is 31.5%, whereas for the Thresholding method, it is 25.4%.



(a) Predicting optimal θ



(b) Comparison of Thresholding and DPF

Figure 2: The Thresholding method

Then, we compared the Thresholding method with DPF, using the testing dataset. After setting $\theta = 0.28$ based on the training results, we used the 500 query images. For each query, we recorded the precision/recall of searching for its near replicas. The results in Figure 2(b) show that the Thresholding method outperforms DPF. When the recall rate is 90% (thirty-six out of forty transformed images have been found), the Thresholding method achieves 49.7% precision, whereas DPF achieves only 41.6%.

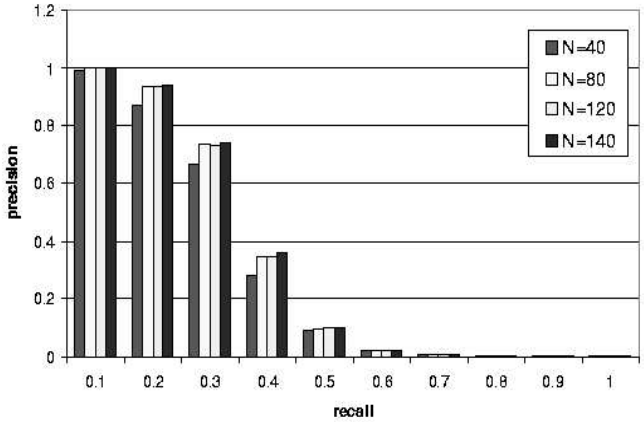
5.2.3 Sampling Methods

Next, we studied Sampling-DPF and Sampling-Thresholding methods. We used the same testing dataset and the 500 seed-images as queries, and recorded the precision/recall. Based on the training results of DPF and Thresholding methods (Figure 1 and Figure 2(a)), we chose the sampling range [100, 240] for the Sampling-DPF method, and [0.1, 0.4] for the Sampling-Thresholding method. Within these ranges, the missing rates of all kinds of transformed images attain the lower values. We randomly sampled 40, 80, 120, and 140 points.

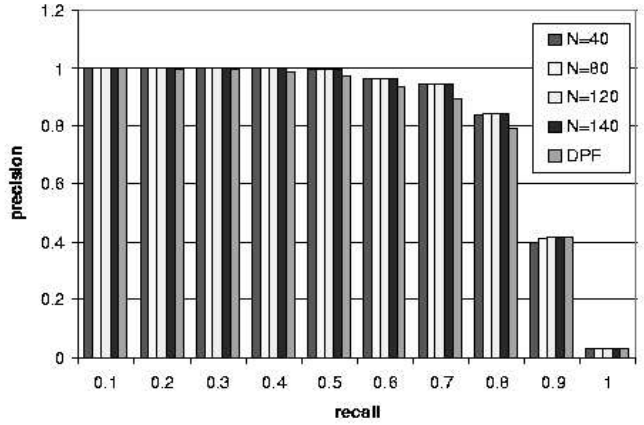
We first compared two aggregation methods: the smallest ranking (SR) and the average ranking (AR). Figures 3(a) and 4(a) present the testing results of using SR, and Figures 3(b) and 4(b) show the results of using AR. These figures demonstrate that for both Sampling-DPF and Sampling-Thresholding methods, AR works better than SR. The superior performance of AR can be attributed to its statistical property. It uses the mean distance obtained from multiple samples (see Equations 4 and 5), which is statistically better than using a single value. Since AR works better, we employed it in our subsequent experiments.

Then, we studied the effect of the sampling frequency. Figure 3(b) and Figure 4(b) depict that the more points sampled (from $N = 40$ to $N = 120$), the higher the precision. When the recall rate is 90%, the accuracy of the Sampling-DPF method increases from 38.1% (when $N = 40$) to 41.8% (when $N = 120$), and the accuracy of the Sampling-Thresholding method increases from 61.5% to 63.0%. As more points are sampled (from $N = 120$ to $N = 140$), the speed of the accuracy increment in both methods slows down. Considering the trade-off between accuracy and computational cost, we do not need to sample as many points as possible. In our case, $N = 120$ is good enough, since it already can achieve high precision. In subsequent work, we used the results based on $N = 120$ points.

We also compared the two sampling methods with DPF and Thresholding methods. Figure 3(b) and Figure 4(b) show that the sampling methods work more effectively. When the recall is 80%, the precisions for DPF, Thresholding, Sampling-DPF, and Sampling-Thresholding meth-



(a) The SR method



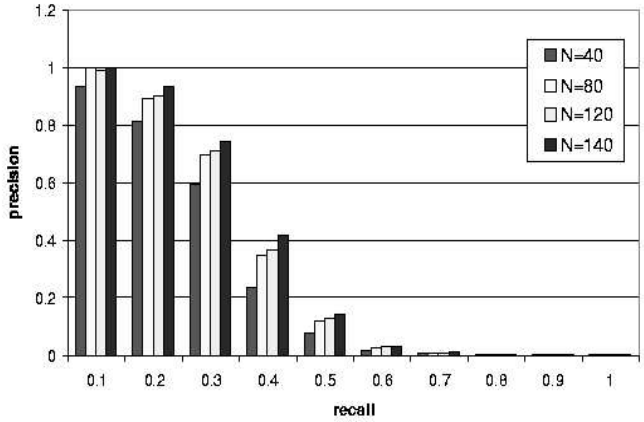
(b) The AR method

Figure 3: The Sampling-DPF method

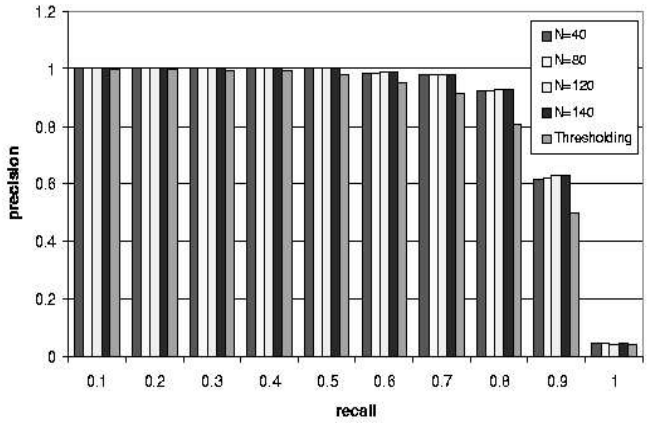
ods are 78.8%, 80.5%, 84.2%, and 92.6%, respectively.

5.2.4 Weighted-Sampling Methods

In addition to considering that different image pairs may have different numbers of similar features, Weighted-Sampling-DPF and Weighted-Sampling-Thresholding methods assign different weights to different features. We used the inverse of the standard deviation of all feature values in the i^{th} feature among similar images as the weight w_i (Equations 6 and 7). Based on the previous study, we randomly sampled $N = 120$ m and θ values within the ranges of [100, 240] and [0.1, 0.4] respectively. Figure 5 presents the comparative results of using the testing



(a) The SR method



(b) The AR method

Figure 4: The Sampling-Thresholding method

image dataset and the 500 seed-images as queries. The figure shows that Weighted-Sampling-DPF and Weighted-Sampling-Thresholding work even better than other methods in recognizing near-replica images. For instance, when the recall rate is 80%, the precisions of DPF, Sampling-DPF, Weighted-Sampling-DPF, Sampling-Thresholding, and Weighted-Sampling-Thresholding methods are 78.8%, 84.2%, 85.4%, 92.6%, and 94.8%, respectively.

To summarize, by adaptively capturing the different characteristics of various image pairs, all new methods (Thresholding, Sampling-DPF, Sampling-Thresholding, Weighted-Sampling-DPF, and Weighted-Sampling-Thresholding) can enhance the performance of DPF by large amounts, outperforming Minkowski-like metrics substantially. Among them, the Weighted-

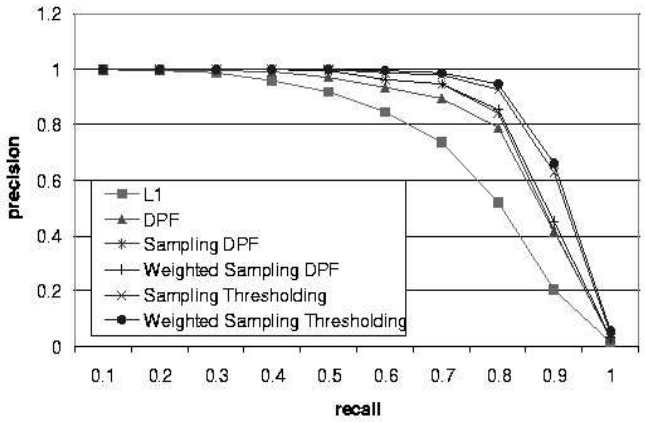


Figure 5: The weighted sampling methods

Sampling-Thresholding method performs best, because of its highest achievement in capturing various image-pair characteristics. When the recall rate is 80%, it improves DPF's performance by 16%, outperforming the L1 distance function by as much as 43%.

5.3. Scalability and Indexing

As shown above, the Weighted-Sampling-Thresholding method performs the best. The Sampling-Thresholding method, as can be seen in Figure 5, performs almost as well, but is simpler to implement and is less computationally expensive. Hence we used the Sampling-Thresholding method as a representative scheme to evaluate how well the proposed distance functions scale to much larger datasets. The precision-recall graph obtained with the million-image dataset was quite similar to that in Figure 5, thus demonstrating that the enhanced perceptual distance functions scale well to much larger datasets. However, we obtained these results by sequentially scanning through the entire dataset, which is inefficient and does not scale well in terms of retrieval time.

Next, we evaluate Locality Sensitive Hashing, and its effectiveness and efficiency when applied to an image replica detection system. We study its suitability to index images, and determine the optimal parameters. We conducted several experiments, each of which is detailed below with results.

The two tunable parameters for LSH are k, the number of random bits selected to create the hash, and l, the number of independent indexes used. We experiment with values of k set to 15, 30 and 45, and that of l ranging from 1 to 10. Values for k greater than 45 are ineffective in retrieving much of the neighborhood, and using k less than 15 results in very large hash buckets so that a very large number of points are retrieved when querying the index, thus making it inefficient. Thus we restrict ourselves to these values.

5.3.1 Indexing in High-D spaces with LSH

Before investigating the use of LSH for indexing images, we study the ability of LSH to retrieve the Euclidean neighborhood of a query point in high-dimensional spaces. Note that in this experiment, we are concerned with only retrieving Euclidean neighbors, and ignore the issue of perceptual similarity.

LSH is an approximate NN hashing scheme in which near neighbors are expected to collide with high probability, but collision is not guaranteed. This probability of collision varies with the value of k. Larger values of k mean that the hash-buckets would be smaller, and a larger proportion of the neighborhood would be left out. In this experiment, for each k, we check the bucket retrieved from the index for a query point, and count the number of points in the bucket that lie within varying radii around the query point. We compare this with the number of points found within these radii by a sequential scan of the entire dataset. The objective is to measure how well LSH can capture neighborhood of a query point. In Figures 6 and 7, we plot these counts vs the radii for each of the LSH indexes and for the sequential scan.

In the plots, the curves for the LSH indexes, compared to that for a sequential scan, depict the proportion of the neighborhood of a query point that is captured by the LSH indexes. This tells us the probability of near neighbors colliding in the LSH hash buckets. From this, we can estimate the number of independent indexes we must use so that we can expect all nearest neighbors to be retrieved.

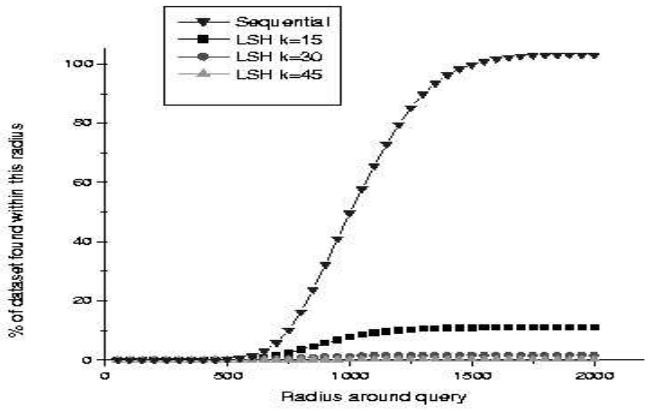


Figure 6: LSH effectiveness

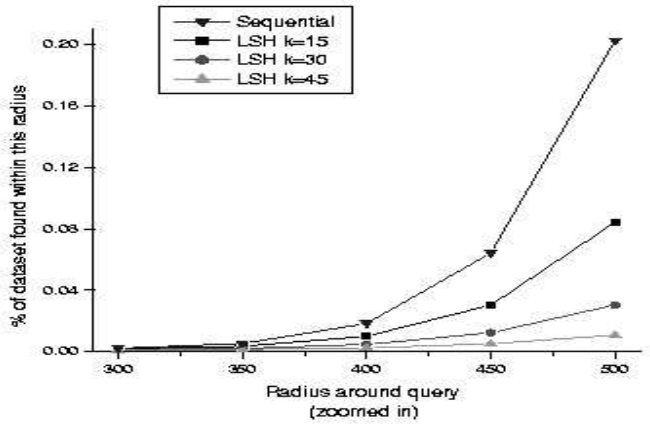


Figure 7: LSH effectiveness - zoomed in

For instance, for points within a radius of 500 units around the query point (see figure 7), the probabilities of colliding with the query point are about 0.4, 0.125 and 0.05 for k set to 15, 30 and 45 respectively. From this we infer that a larger number of independent indexes would be required when k is 45, than when it is 15. Note however that a smaller number of indices used may not mean higher efficiency, since the total number of data points that have to be read also depend on the size of each hash bucket. The number of points in the hash bucket for k set to 45 would be much smaller than for k set to 15. Hence to select optimal values, we have to compare efficiency and effectiveness across a range of both k and l. We do so in Section 5.3.3.

A second observation from the plots is that the curves for the LSH indices are almost flat after certain radii, which indicates that very few of the points retrieved in a LSH bucket are

not in the near neighborhood of the query point. This confirms that the locality-sensitive hash functions indeed preserve locality.

5.3.2 Indexing with LSH for perceptual distance functions

In Section 4.2, we proposed the following method for using LSH to retrieve perceptually similar images in an efficient manner. Use the index structure to retrieve a large Euclidean neighborhood of the query point that includes most of the perceptually similar points. Then the points retrieved from the LSH hash buckets are sorted using the advanced perceptual distance functions to select the most similar images.

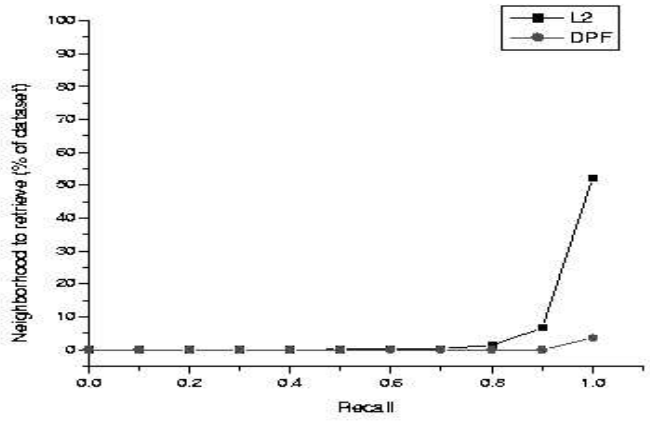


Figure 8: Limitation of L2 index

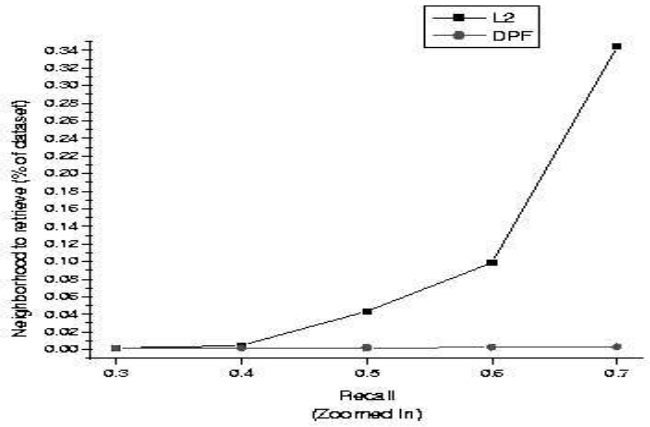


Figure 9: Limitation of L2 index - zoomed in

We present results here from experiments conducted to study the size of the Euclidean

neighborhood that would have to be retrieved as above, for different desired recall. We compare this with the ideal situation, where the indexing mechanism would allow retrieving neighbors directly using the DPF measure.

We rank all points in the dataset by distance from the query point, using L2 and DPF respectively, and count the number of nearest neighbors we must scan, for each, to achieve various recall. We plot the percentage of data retrieved vs. recall, for both L2 and DPF in Figure 8 and Figure 9.

From the figures, we can see, for instance, that to get a recall of 0.6, we need to retrieve only about 0.1 percent of the dataset, when retrieving L2 neighborhood. Thus a scheme like LSH can be used to achieve a reasonable recall by retrieving a very small percentage of the database. If an indexing scheme that allowed retrieving DPF neighbors directly was available, much higher recall could be achieved very efficiently. However, indexing in non-metric spaces is still an open research problem, and we have to resort to this hybrid scheme.

5.3.3 Scalable Image Replica Detection Using LSH

We used the following performance measures to evaluate indexing—hit rate and retrieval time. We define hit rate as the fraction of relevant points (i.e. near-replicas of the query) found in the top 40 points from the retrieved set. We choose this measure of performance over the more commonly used recall because we believe that in a real system, the user would not want to look through a large number of points but would rather see results in the top 40 candidates. (40 would be a reasonable number, since that is the number of replicas in the database.)

Figure 10 shows hit rate achieved with different parameter settings for l and k. As stated before, we present results with values of k set to 15, 30 and 45, and that of l ranging from 1 to 10.

To measure retrieval time, we use a count of the number of data points read. Our experiments were done in a memory-based setting where all data points are stored in memory before

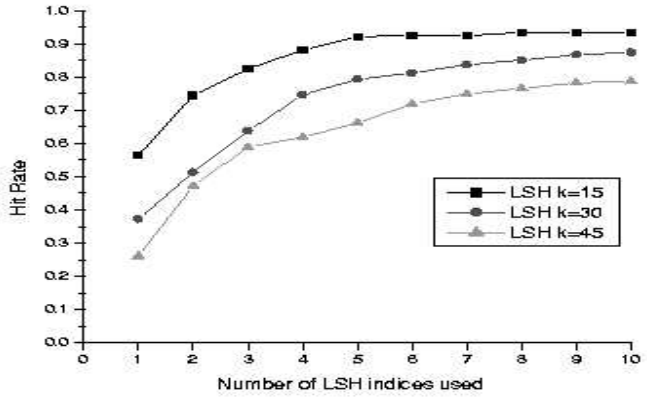


Figure 10: Hit Rates achieved with LSH

processing. Thus retrieval time is directly proportional to the number of points read. Figure 11 shows the number of points read for different l and k.

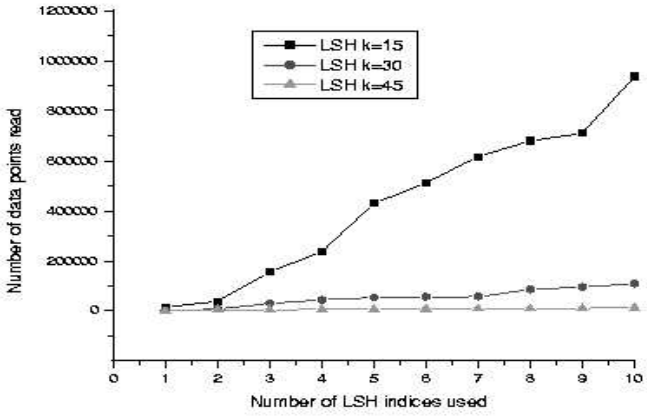


Figure 11: In-memory performance of LSH

Figures 10 and 11 show us that LSH can be used to efficiently achieve high performance. Clearly there is a tradeoff between efficiency and hit rate. Higher hit rates can be achieved at the cost of some loss in efficiency, if required by a particular application. The parameter settings $k = 45$ and $l = 9$ can be considered optimal settings. With these parameter settings, an average hit rate of 0.78 is achieved by retrieving about 1% of the dataset. Our experiment on a Pentium-III 1GHz workstation shows a query time of about 3 to 5 seconds at this setting, which is acceptable to an interactive user.

5.3.4 Summary

From the above experiments on indexing and scalability, we conclude that Locality Sensitive Hashing is a promising technique for efficient detection of pirated images in large image databases. LSH does indeed preserve locality, but its probabilistic nature causes some neighbors to be missed. However, this can be easily taken care of by the use of multiple independent LSH indices. High recalls can be achieved efficiently by using the perceptual distance functions in conjunction with LSH. This can be done by retrieving a larger neighborhood using the LSH indices and then using the perceptual distance functions to order retrieved points. Such a hybrid scheme allows achieving high hit rates (almost 80%) in interactive time ($3 - 5$ seconds). We found the best parameters to be $k = 45$ and $l = 9$.

5.4. Experiments with proprietary database

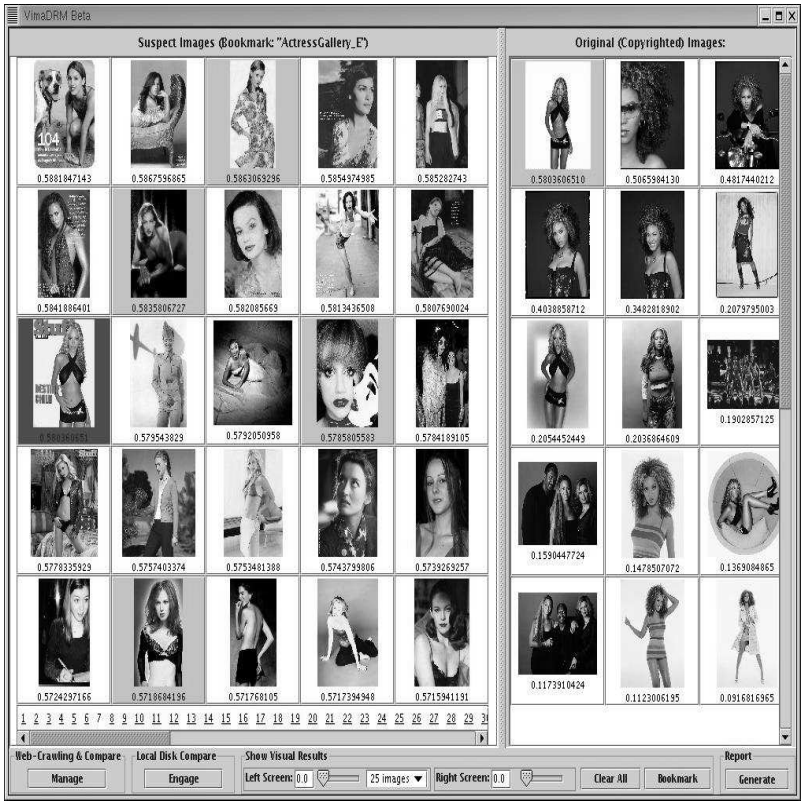


Figure 12: Detecting replicas of proprietary images - 1

We now present some snapshots (Figures 12 and 13) from replica search experiments with

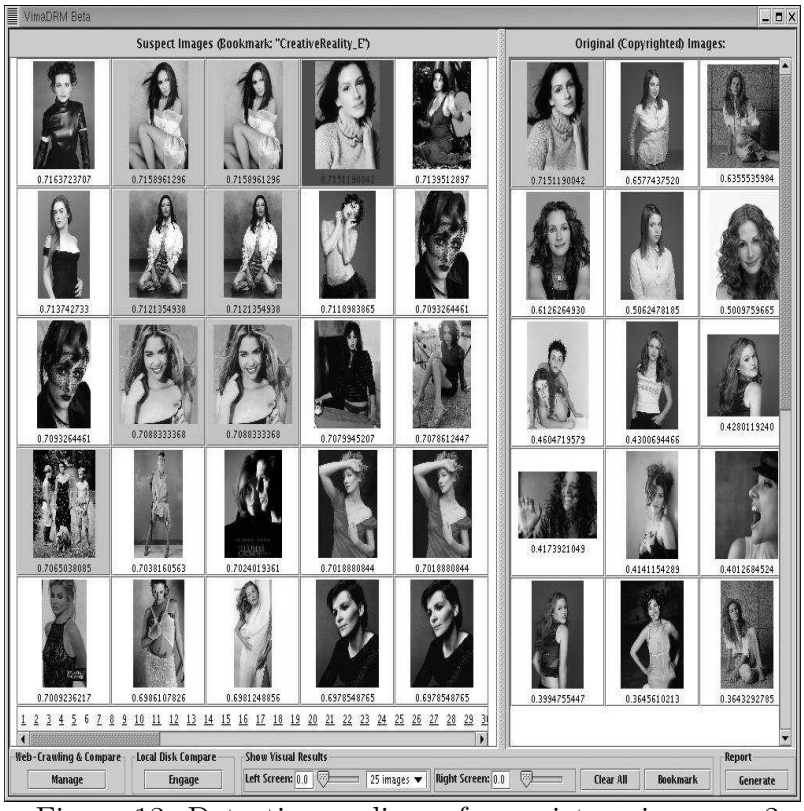


Figure 13: Detecting replicas of proprietary images - 2

the proprietary dataset mentioned in Section 5.1. In the pictures, the left half of the UI shows the near-replica suspects (found on the Internet) in yellow frames. When an image on the left is selected by the user (selected image seen in a red frame), the right half of the UI displays the top 15 matching images from the proprietary image dataset. For instance, in Figure 12, the near-replica found on the Internet (framed in red on the left) is a cropped near-replica of an original image (the top image returned by the system on the right) with some added text. In Figure 13, we see that even when the replica (first row, 4th image on the left) has been transformed to change the color of clothes, our distance functions can successfully recognize it as a near replica of an original proprietary image (which is shown on the right).

Based on manual searches of Web-sites, it was found that the image replica detection system could find replicas with an accuracy of upto $60 - 70\%$, thus demonstrating that the methods proposed in this work are effective for finding image near-replicas for real-world datasets.

Unfortunately, due to copyright issues we are unable to present a detailed study using this

dataset. However, we believe that thorough empirical evaluation presented above using the artificial dataset, and this brief information about performance with real-world images, give a reliable indication of the usefulness of the methods presented here.

5.5. Note

A very recent paper [35], presents a parts-based technique for image replica detection. In the technique proposed by [35], a large number of feature vectors (in the order of 1000) are extracted per image for characterizing salient points, resulting in high feature extraction time (1 to 10 seconds per image, which is 10 to 100 times our feature-extraction time). Another drawback of the parts-based is that images which contain the same or similar objects (e.g., the White House) as a query image, are likely to be retrieved as replicas, even though they are not near-replicas. We believe that the approach of [35] could yield better precision/recall due to its more robust feature extraction at the cost of higher computational time. An interesting research topic is to apply DPF on the salient-point features and examine the effectiveness of the combined scheme.

6. Conclusions

In this work, we addressed two important issues for near-replica image detection: measuring perceptual similarity for images, and indexing image data so as to efficiently retrieve perceptually similar images. We enhanced the DPF perceptual distance measure proposed previously through three advanced schemes: Thresholding, Sampling and Weighting. Experimental results have demonstrated that all these approaches improve DPF significantly and the Weighted-Sampling-Thresholding method gives the best performance. For indexing images for efficient replica detection, we proposed a scheme to use Locality Sensitive Hashing with the perceptual distance functions presented, and demonstrated that for very large image datasets, this scheme can be used to efficiently achieve high hit rates.

In the future, we plan to compare DPF and the enhanced methods with other statistical

measures—such as *k-median*, *M-estimator* [22], χ^2*-statistics*, and *Jeffrey-divergence* [23]— for quantifying image similarity. On the indexing aspect, we plan to work on further improving indexing mechanisms for non-metric distance functions like DPF.

7. Acknowledgement

We would like to thank support of NSF grants IIS-0133802 (CAREER) and IIS-0219885 (ITR).

References

[1] H. Garcia-Molina, S. P. Ketchpel, and N. Shivakumar, "Safeguarding and Charging for Information on the Internet", *Proceedings of the Fourteenth International Conference on Data Engineering*, 182-189, 1998

[2] Y. Meng and E. Chang, "Image Copy Detection Using DPF", *IS&T/SPIE International Conference on Storage and Retrieval for Media Databases*, San Jose, 176-186, 2003

[3] R. Picard, "A Society of Models for Video and Image Libraries", *IBM Systems Journal*, 35(3):292-312, 1996

[4] B. S. Manjunath, P. Wu, S. Newsam, and H. Shin, "A Texture Descriptor for Browsing and Similarity Retrieval", *Journal of Signal Processing: Image Communications*, 16(1-2):33-43, 2000

[5] J. Z. Wang, G. Wiederhold, O. Firschein, and S. X. Wei, "Content Based Image Indexing and Searching Using Daubechies' Wavelets", *Journal of Digital Libraries*, 1(4):311-328, 1998

[6] A. Jain and G. Healey, *A Multiscale Representation Including Opponent Color Features for Texture Recognition*, IEEE Transactions on Image Processing, 7(1):124-128,1998

[7] M. W. Richardson, "Multidimensional Psychophysics", *Psychological Bulletin*, 35:639–660, 1938

[8] C. C. Aggarwal, A. Hinneburg and D. A. Keim, "On the Surprising Behavior of Distance Metrics in High Dimensional Space", *International Conference on Database Theory*, 420-434, 2001

[9] I. Witten, A. Moffat, and T. Bell, *Managing Gigabytes: Compressing and Indexing Documents and Images*, Van Nostrand Reinhold, NY, 1994

[10] Y. Chen, J. Z. Wang, "A Region-based Fuzzy Feature Matching Approach to Content-based Image Retrieval", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(9):1252-67,2002

[11] B. Li, E. Chang and Y. Wu, "Discovery of A Perceptual Distance Function for Measuring Image Similarity", *ACM Multimedia Journal Special Issue on Content-Based Image Retrieval*, 8(6):512-522, 2003

[12] Y. Meng, E. Chang and B. Li, "Enhancing DPF for Near-replica Image Recognition", *International Conference on Computer Vision and Pattern Recognition*, 416-423, June 2003

[13] H. Liu and H. Motoda, *Feature Selection for Knowledge Discovery and Data Mining*, Kluwer Academic Publishers, Norwell MA, 1998

[14] A. Blum and P. Langley, "Selection of Relevant Features and Examples in Machine Learning", *Artificial Intelligence*,97:245-271, 1997

[15] T. Dietterich, "Machine Learning Research: Four Current Directions", *AI magazine*, 18:97-136, 1998

[16] R. Kohavi, "Wrappers for Feature Subset Selection", *Artificial Intelligence*, 97(1-2):273:324, 1997

[17] E. B. Goldstein, *Sensation and Perception*, (5th Edition), Brooks/Cole, 1999

[18] Y. Rui, T. S. Huang and S. Mehrotra, "Content Based Image Retrieval with Relevance Feedback in MARS", *International Conference on Image Processing*,(2):815-818, 1997

[19] J. Smith and A. Natsev, "Spatial and Feature Normalization for Content Based Retrieval", *Proceedings of IEEE Intl. Conference on Multimedia and Expo*, 1:193-196, 2002

[20] A. Tversky, "Features of Similarity", *Psychological Review*, 84(4):327-352, 1977

[21] M. S. Lew, *Principles of Visual Information Retrieval*, Springer, 2001

[22] P. Huber, *Robust Statistics*, Wiley, 1981

[23] J. Puzicha, J. M. Buhmann, Y. Rubner and C. Tomasi, "Empirical Evaluation of Dissimilarity Measures for Color and Texture", *I*nternational Conference on Computer Vision, 1165-1172, 1999

[24] A. Gionis, P. Indyk, R. Motwani, "Similarity Search in High Dimensions via Hashing", *The VLDB Journal*, 518-529, 1999

[25] K. Goh, B.Li and E. Chang, "DynDex: A Dynamic and Non-metric Space Indexer", *ACM Multimedia*, 466-475, 2000

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

[27] C. Yang, "MACS: Music Audio Characteristic Sequence Indexing For Similarity Retrieval", *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, 2001

[28] J. Buhler, "Efficient Large-Scale Sequence Comparison by Locality-Sensitive Hashing", *Bioinformatics*, 17(5) 419-428, 2001

[29] P. Indyk and R. Motwani, "Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality", *Symposium on Theory of Computing*, 604-613, 1998

[30] R. L. Goldstone, "Similarity, interactive activation, and mapping", *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 20:3-28, 1994

[31] D. L. Medin, R. L. Goldstone, and D. Gentner, "Respects for Similarity", *Psychological Review*, 100(2):254-278, 1993

[32] S. Tong, and E. Chang, "Support Vector Machine Active Learning for Image Retrieval", *Proceedings of the ninth ACM International Conference on Multimedia*, 107-118, 2001

[33] B. Li, K.-S. Goh, and E. Chang, "Confidence-based Dynamic Ensemble for Image Annotation and Semantics Discovery", *Proceedings of ACM International Conference on Multimedia*, 195-206, 2003

[34] B. Li, E. Chang, and C.-S. Li, "Learning Image Query Concepts via Intelligent Sampling", *Proceedings of the IEEE International Conference on Multimedia*, 1168-1171, 2001

[35] Y. Ke, R. Sukthankar, and L. Huston, "An Efficient Parts-based Near-duplicate and Sub-image Retrieval System", *Proceedings of ACM International Conference on Multimedia*, 2004