Near-Duplicate Detection for Images and Videos

Xin Yang Qiang Zhu Kwang-Ting Cheng
Dept. of Electrical and Computer Engineering, University of California, Santa Barbara, CA 93106, USA
xinyang@umail.ucsb.edu qzhu@ece.ucsb.edu timcheng@ece.ucsb.edu

ABSTRACT
In this paper, we describe a system for detecting duplicate images and videos in a large collection of multimedia data. Our system consists of three major elements: Local-Difference-Pattern (LDP) as the unified feature to describe both images and videos, Locality-Sensitive-Hashing (LSH) as the core indexing structure to assure the most frequent data access occurred in the main memory, and multi-steps verification for queries to best exclude false positives and to increase the precision. The experimental results, validated on two public datasets, demonstrate that the proposed method is robust against the common image-processing tricks used to produce duplicates. In addition, the memory requirement has been addressed in our system to handle large-scale database.

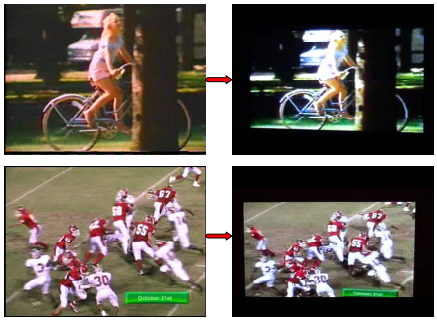
Categories and Subject Descriptors: I.4.9 [Image Processing and Computer Vision]: Application

General Terms Algorithms, Experimentation

Keywords
Duplicate, Feature, Indexing, Query, Database, Local-Similarity-Pattern, Locality-Sensitive-Hashing

1. INTRODUCTION
One of the major issues encountered in the new century for internet is the shear growth in volume of multimedia data, as seen by the growth of the media-acquired hardware. This growth in data also increased the importance of duplicate data detection from the perspectives of data storage, data mining and search engines.

1.1 Duplicate Data
What constitutes a *duplicate* could have several different definitions. For instance, a duplicate can be defined as the exact syntactic terms and sequence, with or without formatting differences. Thus, there are either no-differences, i.e. identical, or only formatting differences, and the content of the data is exactly the same. In such a restrictive definition of duplicates, we can calculate a unique hash value or each data item and then examine for duplications by looking up the hash value (checksum) in either an in-memory hash or a persistent lookup system. Specific to images/videos, a *duplicate* is often not an identical copy but rather a transformed copy of the original source of images/videos using digital photometric or geometric transformations. In the literatures, the transformed copies are sometimes referred to as *near duplicates* of the source images/videos. While some of the transforemations are quite exotic – to ensure that each variant is unique and thus would not be detected by existing algorithms using cryptographic hash of the data, but visually all variants are very similar to the source image and to each other. Some real examples are shown in Figure 1.



Near duplicates in two replica videos selected from MUSCLE-VCD-2007 [1], a corpus used for CIVR 2007 video copy detection evaluation.

Near duplicates selected from the Art Image Database [2], a public dataset for evaluation of image copy detection purpose. Left to right, four techniques (framing, down-sampling, rotating and cropping) used to produce duplicates.

Figure 1: Near duplicates of image/video in the real world

1.2 Applications
The ability of finding near duplicates efficiently in large data collections is of strong demand for several applications:

- **Reduce redundant contents on web**: recent reports indicate that a high percentage, up to 40%, of the web contents is duplicates. This fact results in a huge waste of storage spaces and a longer retrieval time for a search query.
- **Image spam detection:** Much of the recent increase in undetected spam can be attributed to the emergence of new and more sophisticated forms of image spam. Early anti-spam attempts to detect image spam included creation of a cryptographic hash for images in spam e-mails, so that, the same images could be detected when they are sent again. However, this technique was easily cracked by the spammers who create a unique variant of the spam image for each recipient by automatically adding some random visual effects

LS-MMRM'09, October 23, 2009, Beijing, China.

to each source image. Therefore, identifying duplicates is core task for image spam detection.

- **Illegal use of images/videos:** Some studies reveal a large portion of the data on the internet is used illegally and infringe the copyrights. In fact, any image used on a website is a potential copyright infringement unless it is a personal photo, a graphic file generated by the owner of the source, or an image acquired from a photobank. Detection of duplicates in conjunction with a web crawler can be used to automate the detection of online copyright infringements.

2. OVERIVEW

2.1 Related Work

2.1.1 Duplicate-Image Detection

Detection of near-duplicate images has been investigated for years. In [3], Zhang et al. proposed near-duplicate detection by using the Stochastic Attribute Relational Graphing technique. With an image-to-image comparison scheme, they show good results in terms of robustness and discrimination; however, the high computational cost limits its application to large-scale databases. In [4], Meng et al. employed the traditional Content-Based-Image-Retrieval (CBIR) framework for duplicate detection in a large data collection. Although their system is able to handle large-scale database, the proposed global image representation, e.g. color/texture feature, is not fully optimized on near-duplicate detection problem. Recent surveys [6][7] conducted an extensive comparison between global feature and local feature in near duplicate detection application. Both works conclude that local image descriptors are in general superior to global image descriptors in terms of accuracy, while at the cost of higher complexity of computing the features and more space for storing them.

The design of using local image descriptors for duplicate detection was originated in [5] which has attracted good attentions since its publication. The success of their framework results from its three core features: using PCA-SIFT as a compact and distinctive local descriptor [8] to characterize the images, using Locality-Sensitive Hashing (LSH) [9] to cope with the large number of local image features, and using L2 distance and RANSAC [10] as the post-verification steps to further eliminate false-matched features. To further speed up the query response, the authors proposed a hierarchically storage scheme to fit the core indexing structure, i.e. the Hash Tables, into the main-memory. However, their scheme requires a large number of hash tables (they used 20 hash tables in their experiments) in order to achieve the desirable level of accuracy. This requirement results in significant increase in both storage space and computation time, which in turn limit its application to large databases. Our system is mostly inspired by Ke et al's system [5], but particularly addresses the scalability limitation in their approach. Our unique feature design and efficient indexing structure allow us to handle 20X larger database compared to Ke et al's method. Recent frameworks employing local feature and Locality Sensitive Hashing have been reported in [20, 23] with focus of accuracy other than addressing the scalability problem.

2.1.2 Duplicate-Video Detection

The explosive growth in broadcasting digital video content through internet to various end-user devices accelerates the demand for search of video copies in large video databases. MUSCLE-VCD-2007 [1], a public video copy detection benchmark which contains about 100 hours of video materials assembled from a number of different sources, provides researchers an ideal driver for research and for evaluation of the various techniques with respect to the same data and settings. This live benchmark was organized during the ACM International Conference on Image and Video Retrieval (CIVR2007) in which a number of research teams participated in the public evaluation using this benchmark.

For video duplicate detection, it has also been reported that local-feature-based methods outperform the global-feature-based approaches in terms of accuracy [11]. For example, the near-duplicate video detection system reported in [12], an enhancement from its earlier version of [13], demonstrated that the elaborated local spatiotemporal feature, combined with their specific matching algorithm, is robust with respect to the most complex and challenging transformations such as cropping and inserting. However, the system still suffers from the high overhead of computation time and storage space which limits its deployment to large databases and real-time applications.

2.1.3 Discussions & Summary

The choice of an image representation and an indexing structure affects both the amount of data stored per image and the time complexity of database search. In this section, we will give a brief summary of these two key issues that need to be addressed for building a near duplicate detection system.

- **Data Representation:** There exist many known features for describing an individual image. An effective representation utilizes a set of features that can effectively capture the "essence" of any source image and that is sufficiently robust in presence of random noises. In particular, our application requires a design of image/video features which are relatively invariant with respect to the typical transformations applied to images/videos for producing near duplicates. In addition, the descriptor should be as compact as possible in order to address the scalability issue for handling large multimedia databases. Prior studies have concluded that local-feature-based methods consistently outperform global-feature-based methods, especially for duplicates resulting from complex transformations.
- **Indexing Structure:** An efficient indexing structure is the most important factor for addressing the scalability problem of a database design.To our knowledge, one of the best-known and popular indexing methods is the Locality Sensitive Hashing (LSH) [9] which achieves high efficiency with very little compromise in accuracy for the duplicate detection applications [5].

2.2 Our System

The near duplicate detection system proposed in this paper, illustrated in Figure 2, adopts a local-feature-based framework along with a hierarchical indexing structure using the LSH technique. We represent an image or a video-frame using a set of local image patches. For each local patch, we design a descriptor, called Local-Difference-Pattern (LDP), which is further encoded as a binary bit-string for hashing. The database indexing follows a hierarchical structure which consists of three individual tables: the "File Table" and the "Feature Table" are stored in the hard disk to index the large collection of images/videos, and the "Hash Table", the core part of

the indexing structure, is stored in the memory and is used for quick selection of candidate features for further L2/RANSAC verification during the search. The experimental results on two public datasets widely used for duplicates detection evaluation show that the proposed system can achieve a detection accuracy comparable to the best results reported in the literatures, while using significantly less processing time and storage space than the prior approaches, Overall, the key strengths of the proposed system can be summarized as follows:

- Our unique feature, Local-Difference-Pattern (LDP), is the key contributor for the significant reduction in both computation cost and storage space. The LDP feature is encoded using only 72 binary-bits which facilitates very efficient hashing. In contrast, the feature used in Ke's system [5] requires 9180 binary-bits for hashing.
- Based on the LSH technique, the hierarchical indexing structure assures that the most frequent data access to the hash table occurs in the main memory. Only a small subset of "candidate" features are selected from the feature set for further L2/RANSAC verification which typically involves hard-disk access. This design greatly reduces I/O costs yet still capable of finding near duplicates accurately.
- The proposed framework is directly applicable to both still images and videos. To the best of our knowledge, this is the first system which reports results on both image and video datasets without special modification and customization with respect to specific media types while, at the same time, achieving an accuracy level comparable to the best prior results reported in the literatures for each media type.

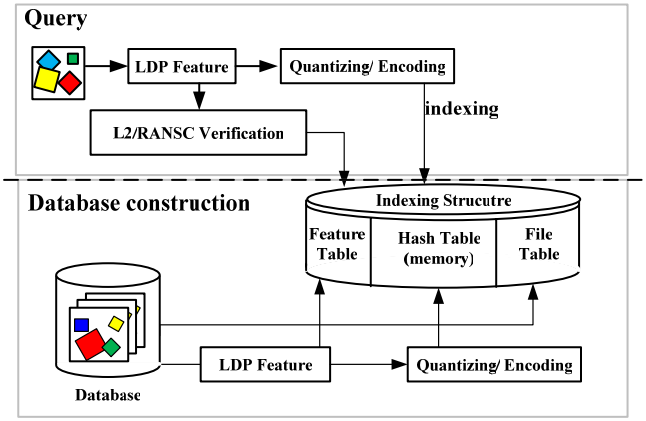


Figure 2: Our system for near duplicate detection

In the remainder of the paper, we first introduce the design of our features for characterizing images and its extension to video sequences, followed by the details of our indexing structure and the query process. We present the experimental results on two public datasets in Section 6 and conclude with a short discussion.

3. FEATURE DESIGN

Ideally, an image descriptor should be distinctive (for reliably distinguishing one image of interest from others), compact (to minimize storage space), and robust with respect to small shifts and random noises. In this section, we first describe details of the Local-Difference-Pattern (LDP) feature, followed by a discussion on its strengths and suitability for solving the target problem of duplicate detection. Finally we conduct a comparison, between the PCA-SIFT feature [8] used in [5] and our feature, using some real data to validate the power of the proposed LDP feature.

3.1 Local-Difference-Pattern (LDP)

We depict the key extraction steps of the Local-Difference-Pattern (LDP) feature in Figure 3 with the following details:

Image Patch Detector: First we detect a set of interest points across the target image using the key-point localization algorithm described in [14] where the Scale-space extreme detection and the Orientation assignment are used to assure that the feature descriptors to be extracted are scale-rotation invariant.

Image Patch Descriptor: We divide each detected image patch into a 3 x 3 equal-sized grid and compute the sum of the differences (SD) between corresponding pixels for each pair of grids. Then we compute the Average Difference Per Pixel (ADPP) by dividing SD by the number of pixels in a gird to construct a symmetric 9 x 9 average difference matrix, each entry of which corresponds to the ADPP derived from a pair of grids. After discarding the diagonal entries (which have a value of zero) and the entries in the lower part of the matrix (as they are identical to the upper part in this symmetric matrix), we have our LDP descriptor of 36 dimensions. By employing the Integral Image [15] calculation technique, these features can be derived very efficiently – approximately 10X faster than the time required for extracting the original PCA-SIFT descriptor.

Quantization and Encoding: We further quantize each of the 36 LDP feature value into two binary-bits with respect to a threshold T: if the feature value is between *T* and 255, it will be encoded as '11'. Similarly, it will be encoded as '10', '01' and '00' if the feature value is in the ranges of (0, T), (0, -T), and (-T, -255) respectively. As a result, the 36-dimensional descriptor is encoded into a 72-bit string, i.e. 9-byte long.

It should be noted that if we encode each feature value by only one bit (i.e. "1" for a positive feature value and "0" for a negative feature value), the resulting 36-bit string will be nicely embedded into a Hamming space. In a Hamming space, the number of bits with different bit values between two bit-strings is directly proportional to the L1 distance between the two corresponding image descriptors. This special property is used in the development of the Locality-Sensitive-Hashing technique when designing our indexing structure. We will discuss this point in Section 4. Quantizing a feature value into two bits, instead of just one bit, offers greater discriminating power for the resulting encoded features. While the bit-string will no longer perfectly fit into the Hamming space, the bit-count difference of two bit-strings is still, statistically, strongly correlated to the L1 distances between the two feature descriptors.

Extension for Video Sequences: As illustrated in Figure 3, we further extend the ADPP operation to two consecutive frames (thus, dealing with two 3x3 grids which results in 18x18 pairs of grids). This straightforward extension which establishes a temporal-spatial descriptor across the video frames only increases the size of the resulting bit-string from 72 bits to 288 bits without altering the process of computing the descriptor. With this straightforward extension, our indexing structure can be used for both image and video databases.

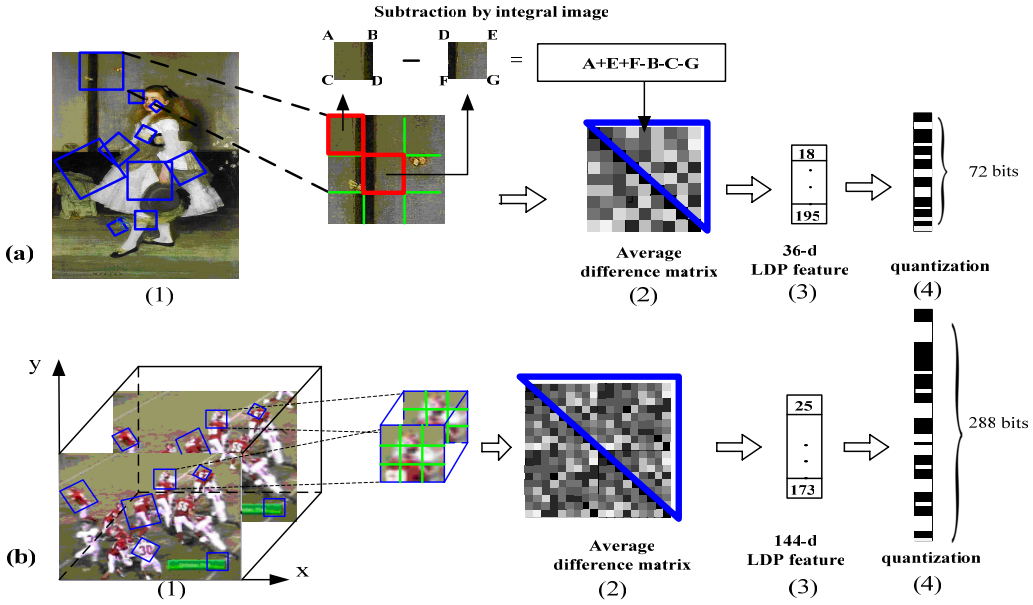


Figure 3: Extraction of Local-Difference-Pattern (LDP) and encoding. (a) At an image pixel. (b) At a video pixel.

1)The keypoint localization algorithm described in SIFT paper[14] to detect a set of scale-rotation invariant image patches; 2)Applying ADPP operation to the pairs of a 3x3 equal-sized grid to construct a average difference matrix(ADM) . Integral image technique is employed in this step to speed-up the process; 3) Constructing LDP feature based on ADM; 4) Quantizing each of the LDP value into two binary-bits with respect to a threshold T to form a bit-string for the following hashing indexing.

3.2 Motivations and Strengths

Our feature design was mostly inspired by the direct observations made on the duplicates commonly seen in practice. Most of the common transformations (color/contrast adjustments, blurring, down-sampling, etc.) can be simply compensated by computing the difference between two sub-regions. While noises added to the source image change some of its characteristics (color, intensity and even quality) as a whole, some of the intrinsic structures within the image are often not changed. The closet work to our LDP feature is *self-similarity descriptor* proposed by Eli and Michal in [16]. *"self-similarity descriptor"* was designed to efficiently capture the internal self-similarities of any interest point within a target image. In their approach a local area was divides into grids and "Sum of Square Differences" between the center grid and the others was computed. However their particular design is not applicable for our application due to two reasons: First, the Integral Image technique cannot be applied to their "Sum of Square Differences" operation and thus their feature extraction process cannot be easily sped up to meet our needs. In addition, they simply concatenated all the difference values into one big feature vector without any quantization/encoding. Therefore, their feature descriptor is not compatible with the Locality-Sensitive-Hashing indexing structure.

3.3 Invariant to Transformations

For making a direct comparison between our LDP feature and the PCA-SIFT feature [8] used in [5], we designed two experiments based on the traditional image-matching task. The database used for this comparison contains 150 source images randomly selected from cgfa art image database [2] and 6000 duplicates (produced by applying 40 different transformations to each source).

The first experiment was inspired by the matching scheme proposed in the original SIFT paper [14]. In their work, for each query feature, they calculate the ratio between its 1st and 2nd nearest-neighbors with respect to a collection of existing features. If the ratio is smaller than a pre-selected threshold, the 1st nearest-neighbor is considered as a match of the query. The rationale behind this rule is that a match to the query feature must have a sufficiently small distance to it. Such a scheme avoids the need for a threshold T to determine a match, and is also efficient for excluding the outliers. In our experiment, we first filter out the matches with the ratio between its 1st and 2nd nearest-neighbors below a threshold (0.8 suggested in [14]) to find matched candidates of a source image to its 40 transformed copies. For further removing false matches, we apply RANSAC [10] as post-verification step. A matched-point-ratio, defined as the number of matched features to the total number of features detected in a source image, is calculated for each source image. A good feature design which is robust with respect to various transformations should have a high matched-point-ratio for all source images. In the experiment, we compute the average ratio over all 150 source images with respect to each of the 40 transformations. Figure 4 shows the comparison of the two ratio curves between the PCA-SIFT feature and the LDP feature. Overall, the LDP feature consistently achieves comparable matched-point-ratio to that of the PCA-SIFT feature, and for some transformations the LDP performs even better than PCA-SIFT feature.

The second experiment directly examines the L2 distance between the PCA-SIFT feature and the LSP feature under the same scenario. We simply calculate the pair-wise L2 distances between the feature set derived from the source image and the one derived from its transformed copy. Using a very strict threshold to determine a match, we identified matched pairs for further analysis. We compute

the average L2 distances derived from the matched pairs over 150 source images with respect to each of the 40 transformations. Figure 5 shows the comparison of two L2-distance curves between the PCA-SIFT feature and the LSP feature. We observed that the LSP features result in smaller L2 distances for all 40 transformations.

In addition to the superior performance shown in Figures 4 and 5, deriving the LDP feature is about 10X faster and the feature size is about 130X more compact (72 bits vs. 9180 bits) for hashing than those for the PCA-SIFT feature.

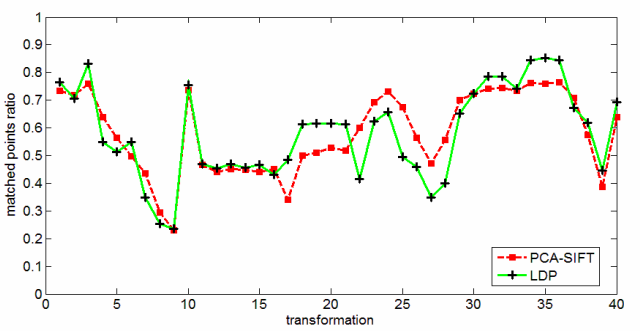


Figure 4: Matched-points-ratio for 40 transformations over 150 source images selected from Art Image Database

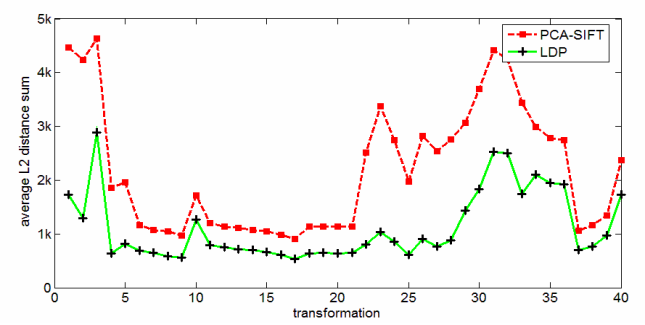


Figure 5: L2 distance for 40 transformations over 150 source images selected from Art Image Database

4. DATABASE ORGANIZATION

Indexing schemes have been widely studied in the domain of document and image retrieval systems. While the tree-based indexing structures [18] [19] return accurate results for any given query, most of them are not suitable for high-dimensional data such as audios, images and videos. Recently, Locality Sensitive Hashing [9] have been employed as the core indexing for approximate similarity search in large databases to achieve high efficiency without significant degradation in accuracy[5][6][20]. One major drawback of the LSH techniques is the requirement for a large number of hash tables in order to achieve sufficient search quality. In this paper, we introduce the integration of the proposed LDP feature into our indexing structure based on the LSH technique. We will explain how we achieve a dramatic reduction in the number of required hash tables, from 20 to 1, while maintaining a high accuracy.

In this section, we first introduce our database organization, followed by the key concept of Locality-Sensitive Hashing for the indexing.

4.1 Table Structure

In Figure 6, we illustrate the structure of our database and also present a table which presents the storage requirements for an exemplar database containing 10k images. While this design is similar to the database structure proposed in [5], our key contribution is in the reduction on the number of hash tables required for achieving a desirable accuracy level which greatly improves the overall LSH efficiency. We briefly describe the objective and the design of each table in the following:

- **The File Table**: This table stores a list of file names to locate the images or videos stored in the hard disk. Each record is 200 bytes in length and the entire table occupies 2MB storage in disk for a database containing 10k records.
- **The Feature Table**: In a local-feature-based framework, each image is represented by hundreds feature vectors. This table stores all local descriptors extracted from all images. In each entry, we use 4 bytes for the file ID to record the source image of this feature, 12 bytes for some geometry information (X and Y locations, scale, and orientation) for later RANSAC verification, and 36 bytes for the LDP descriptor. Assuming an average of 1,000 features per image for a dataset of 10K images, the Feature Table requires a storage space of approximately 520M. Apparently, this table has to be in the hard disk to assure the system scalability for much larger video/image databases in real applications.
- **The Hash Table**: Locality Sensitive Hashing was originally designed to achieve its efficiency assuming the data are in the main memory, for which random access is fast. In our database structure, the key indexing structure is the ***Hash Table*** which basically establishes a direct mapping to the ***Feature Table*** by hashing a subset of the feature values using the LSH technique. Each entry of this table includes the feature ID, a link back to the ***Feature Table***, and a 32-bits checksum for excluding false matches during hashing. This table helps effectively narrow the feature matches down to a small subset of candidates by hashing similar features in the high-dimensional space into the same bucket. Then more sophisticated distance metrics (or even taking the geometry information into account) are employed to further improve search precision. We discuss our hashing choice, the LSH technique, in the following.

4.2 Locality Sensitive Hashing

4.2.1 Preliminaries

Locality Sensitive Hashing (LSH), proposed in [9], is an approximate nearest-neighbor search technique that works effectively even for high-dimensional data. A LSH scheme defines a family of hash functions operating on a collection of objects, such that for two objects x and y,

$$\Pr_{h \in F}[h(x) = h(y)] = sim(x, y), \quad (1)$$

Where *sim(x,y)* is some similarity function defined on the collection of objects. In other words, highly similar objects will be hashed into the same bucket in the hash table with a high probability. The key task in an LSH indexing method is to design a set of specific hash functions that satisfy the criteria defined in Formula (1), so that we can approximate the nearest-neighbors for any query in O (1) time. This is a desirable property for any retrieval system which contains

a large collection of objects. One of the easiest ways to construct an LSH family is by bit sampling. This approach works for the Hamming distance over d-dimensional binary vectors $\{0,1\}^d$. Here, the family $\mathcal{F}$ of hash functions is simply the family of all the projections of points on one of the d coordinates, i.e.

$$\mathcal{F} = \{h : \{0,1\}^d \rightarrow \{0,1\} \mid h(x) = x_i, i = 1...d\} \quad (2)$$

where x_i is the i^{th} coordinate of x. A random function h from $\mathcal{F}$ simply selects a random bit from its input vectors.

4.2.2 Implementation details

In [5], Ke et al. adopted a set of locality-sensitive hash functions using the Hamming distance, originally introduced by Gionis *et al.* in [21]. They first map each 36-dimensional PCA-SIFT feature vector into a 9180-dimensional Hamming space by concatenating the unary representation of each (discredited) coordinate for each feature value in [0, 255]. They then randomly select a subset (450 bits in their system) out of 9180 bits for hashing. Although their method is straightforward and easy to implement, it requires 20 hash tables for their implementation, each of which utilizes a unique subset for hashing, in order to achieve a desirable accuracy level. The LSH family employed in our hashing scheme follows a similar principle but has the following two distinct aspects:

1) We map the 36-dimensional LDP vector into a 72-dimensional near-Hamming space through a clever encoding scheme (described in Section 3.1), rather than concatenating the unary representation of each (discredited) coordinate for each feature dimension. This results in a significantly more compact representation (reducing from 9180 bits to 72 bits). In our experiment, we randomly chose 36 bits out of the 72-bit string for hashing. With this choice, one hash table would be sufficient to achieve the same level of accuracy as that achieved by the setting of 20 hash tables reported in [5]. This dramatic improvement indicates that our 72-bit signature carries as much useful information as that in their 9180-bit signature. Therefore, one hash table in their system carries a much smaller fraction of the information (450 out of 9180 bits) than that in our system (36 out of 72 bits).

2) While we might be able to apply the same encoding/quantizing to the PCA-SIFT features to avoid the long bit string, the amount of information carried in the resulting shorter string would likely be significantly reduced. That is, the encoding works uniquely well for our LDP feature – it is not just a mathematical transformation, also making great sense as part of the feature self. After quantizing the difference between any pair of grids into a two-bit vector, the resulting bit-string does carry information regarding the intrinsic structure around each interest point. For example, a string with frequent patterns of "00" or "11" would indicate an image patch with high contrast between grids.

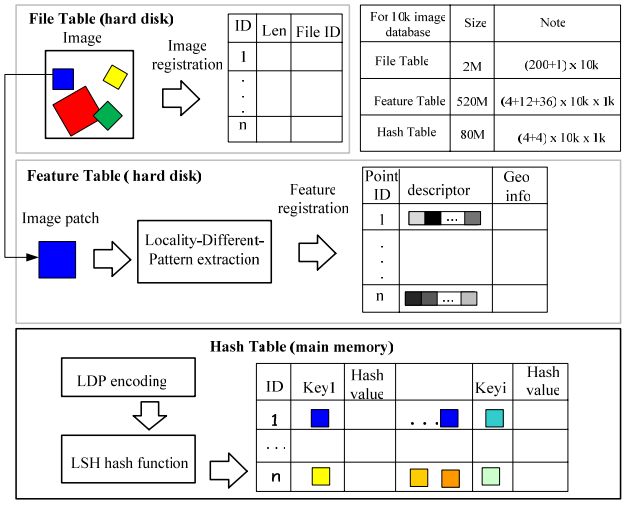


Figure 6: Database organization with tables and exemplar storage space requirements for a database containing 10k images

5. RESPONSE TO A QUERY

Once we construct the indexing structure for the entire database we can process any query according to the following steps:

1) Locate interest points on the query image and extract/encode the LDP features as described in Section 3.

2) Perform LSH hashing to obtain a set of candidate features and acquire them in the feature table from the disk to the main memory as described in Section 4.

3) To eliminate falsely matched features, compute the L2 distance between the query feature and each of the candidates, and discard those candidates with a distance above a pre-defined threshold.

4) Look up the file table to acquire a set of candidate images for which the number of matches in the feature level is larger than a threshold. Further employ the RANSAC [10] algorithm for geometric verification to minimize the false matches in the image level.

6. EXPERIMENTS

In this section, we describe the experiment settings and present the experimental results on two widely used datasets: a) the Art Image Dataset, and b) MUSCLE-VCD-2007.

6.1 Art Image Database

Our first experiment is based on the Art Image Database [2] used for image duplicate detection evaluation. We randomly selected 150 images, synthesized 40 duplicates for each selected image and added the rest in the database to the gallery as distractors to construct a database containing 10k images. We choose this standard database for a fair comparison to other existing work, however, the proposed method is able to handle much larger-scale database The experiment setting is nearly identical to that described in [5]. For reporting the results, we also use the same evaluation metrics of *recall* and *precision* defined in [5]. All of our

experiments are implemented in C++ and run on a PC with a 2.8 GHz CPU and 4GB memory.

In Table 1 we show the accuracy performance under the same experimental setting for five different approaches. As reported in [5], the system proposed in [4] achieved a recall of 90% and a precision of 67% for the Art Image Database. The next two reported numbers are directly cited from [22] which employs DCT fingerprint and FMT fingerprint respectively. The above two systems use a global-feature-based representation, and neither of which can achieve a desirable accuracy level on both recall and precision. The two systems, reported in the last two rows of the table, employ a local-feature-based framework along with the LSH technique as the core indexing structure. The results clearly indicate that the local-feature-based systems perform extremely well in terms of accuracy. We should point out that the system proposed in [5] was re-implemented for our experiments, and therefore the reported accuracy numbers, while very close, are not completely identical to the data reported in [5] (with a recall of 99.85% and a precision of 100%). We believe this minor reduction in accuracy is mainly due to: 1) we used a different SIFT point detector for the purpose of easier integration, instead of the original Lowe's binary detector used in [5]; 2) the random selection of the query images for the experiments inevitably introduces some randomness to the final results.

Method	Recall	Precision
Meng et al's system [4]	90%	67%
DCT fingerprint [22]	59.6%	100%
FMT fingerprint [22]	33.7%	100%
Ke et al's system [5]: using 20 hash tables (using 1 hash table)	99.4% (92%)	99.6% (99.8%)
Our system using LSP + LSH: using 1 hash table	**98.9**%	**99.9**%

Table 1: Accuracy comparison for duplicate detection methods on the Art-Image Database

In comparison with the system reported in [5], while achieving comparable results on accuracy, our system outperforms significantly in terms of the **storage space** and the **computation cost**:

- In Ke et al's system, 20 hash tables are used to achieve the reported accuracy in Table 1. If only 1 hash table is used for their system, the recall drops to 92%. Given the same, limited memory resources to store the core indexing of hash tables, our system can therefore handle 20X larger databases while achieving the same accuracy level.
- As for computation time, we greatly speed up the extraction of the LSP feature by using the Integral Image technique -- approximately 10X faster than extracting a PCA-SIFT descriptor. Another significant saving in processing time results from the significantly less hashing time, which is proportional to the number of hash tables.

6.2 MUSCLE-VCD-2007

The corpus used for CIVR 2007 video copy detection evaluation is referred as MUSCLE-VCD-2007 [1]. It provides researchers an ideal driver for research and evaluation of various techniques. This dataset contains about 100 hours of video materials coming from different sources: web video clips, TV archives, and movies. The evaluation consists of two separate tasks (ST1 and ST2):

Video Query (ST1): Copies of an entire video chip (from 5 minutes to 1 hour) as the query. The transformations includes re-encoded, noised, or slightly re-edited. The most challenging queries could be movies re-acquired by a camcorder.

Video Stream Query (ST2): Queries include parts of several videos belonging (or not belonging) to the database. Sequences belonging to the database must be identified and localized by their start and end times. The length of an inserted segment is within the range of 1 second to 1 minute.

The following new features are necessary to adapt our system for detecting video duplicates:

- **Frame Sampling Strategy**: Given 100 hours of video materials with the standard 25 fps, the total number of video frames is close to 10 millions. For a local-feature-based framework, we have to design a clever sampling strategy for selecting a small subset of, while most valuable, frames to represent the target video. In our implementation, we compute the pixel difference between two consecutive video frames and only select those frame pairs with a large enough difference values for incorporation into the database. Such a sampling scheme will select frames typically occurred in the transitions of the videos. In our experiments, we found a small number of frame pairs (from 30 to 50) are sufficient to achieve a good accuracy for finding the duplicate videos in the database.
- **Extended LSP Feature:** As described in Section 3.1, we establishes a temporal-spatial descriptor across the video frames which only increases the size of the resulting bit-string from 72 bits to 288 bits without altering the process of computing the LSP descriptor. With this straightforward extension, we do not need modify our indexing structure in the scenario of detecting video duplicates.
- **Voting Function:** The direct output of our system still predicts a duplicate, or not, in the frame-level. To accumulate the frame-level results to a video-level label, we need a voting scheme. In our experiments, the simple majority voting is proven sufficient for achieving a desirable accuracy.

Methods	ST1 score	ST1 search time
Best Team in CIVR07 [1]	0.86	44 minutes
Video mining system [12] *	0.93	23 minutes
Our system using LSP + LSH	**0.93**	**25 minutes**

*The reported time seems only accounting for the indexing part. If this is the case, the indexing in our system only takes seconds. The major computation is for decoding and feature extraction.

Table 2: Performance comparison for duplicate detection methods for ST1 on MUSCLE-VCD-2007

We should mention that, with a sparse frame-sampling strategy, our system is not very suitable for the task of ST2 whose duplicates could be of only 1-second long. Therefore, we only perform a comparison on the ST1 task, presented in Table 2. Our system still achieves the state-of-the-art performance in terms of both accuracy and computation time.

7. CONCLUSION

We demonstrate a system of detecting duplicated images and videos that matches the detection performance of existing state-of-the-art methods while using significantly less storage space and lower CPU runtime. Such reductions are achieved by integrating the LSH concept with the novel LSP feature design. Central to the success of our approach is the use of a much more compact feature (72 bits vs. 9180 bits) which in turn results in a reduction of the required hash tables.

8. REFERENCES

1. MUSCLE-VCD-2007, http://www.rocq.inria.fr/imedia/civr-bench/index.html.
2. CGFA - A Virtual Art Museum, http://cgfa.sunsite.dk/.
3. D. Q. Zhang, S. F. Chang. Detecting Image Near-Duplicate by Stochastic Attributed Relational Graph Matching with Learning, In *Proceedings of ACM International Conference on Multimedia*, New York, USA, October 2004.
4. Y. Meng, E. Y. Chang, and B. Li. Enhancing DPF for near-replica image recognition. In *Proceedings of IEEE International Conference on Computer Vision*,, Madison, Wisconsin, June 2003.
5. Y. Ke, R. Sukthankar, and L. Huston. Efficient Near-duplicate Detection and Sub-image Retrieval. In *Proceedings of ACM International Conference on Multimedia*, New York, USA, October 2004.
6. J. J. Foo, R. Sinha, and J. Zobel. Discovery of image versions in large collections. In *Proceedings of ACM International Conference on Multimedia Modeling*, Singapore, January 2007.
7. B. Thomee, M. J. Huiskes, E. M. Bakker, and Michael S. Lew. Large scale image copy detection evaluation. In *Proceedings of ACM International Conference on Multimedia Information Retrieval*, Vancouver, Canada, October 2008.
8. Y. Ke and R. Sukthankar. PCA-SIFT: A more distinctive representation for local image descriptors. In *Proceedings of IEEE Computer Vision and Pattern Recognition*, Washington, DC, June, 2004.
9. P. Indyk and R. Motwani. Approximate nearest neighbor - towards removing the curse of dimensionality. In *Proceedings of Symposium on Theory of Computing*, Dallas, Texas, May1998.
10. M. Fischler and R. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6), June 1981.
11. L. T. Julien, C. Li, J. Alexis, L. Ivan , B. Olivier, G. B. Valérie, B. Nozha, and S. Fred. Video copy detection: a comparative study. In *Proceedings of ACM International Conference on Image and Video Retrieval, Amsterdam, Netherlands, July* 2007
12. S. Poullot and O. Buisson. Scalable mining of large video database using copy detection. In *Proceedings of ACM International Conference on Multimedia*, Vancouver, Canada, October 2008
13. S. Poullot and O. Buisson. Z-grid-based probabilistic retrieval for scaling up content-based copy detection. In *Proceedings of ACM International Conference on Image and Video Retrieval, Amsterdam, Netherlands, July* 2007
14. D. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2), pp. 91-110, 2004.
15. P. Viola and M. Jones. Robust Real-Time Face Detection, *International Journal of Computer Vision*, 57(2), pp. 137-154, 2004.
16. E. Shechtman and M. Irani. Matching Local Self-Similarities across Images and Videos. *IEEE Conference on Computer Vision and Pattern Recognition,* Minneapolis, Minnesota, June 2007.
17. T. Ojala, M. Pietikainen, and T. Maenpaa. Multiresolution gray-scale and rotation invariant textureclassification with local binary patterns. In *IEEE Trans. Pattern Analysis and Machine Intelligence,* pages 971-987, Jul. 2002.
18. Beygelzimer, S. Kakade, and J. Langford. Cover trees for nearest neighbor. In *Proceedings of the 23rd International Conference on Machine Learning*, Pittsburgh, Pennsylvania, June, 2006
19. R. Krauthgamer and J. R. Lee. Navigating nets: Simple algorithm fir proximity search. In *Proceedings of the 15th ACM-SIAM Symposium on Discrete Algorithms*, New Orleans, LA, January 2004.
20. P. Jain, B. Kulis, and K. Grauman. Fast Image Search for Learned Metrics. In *Proceeding of the IEEE Conference on Computer Vision and Pattern Recognition*, Anchorage, Alaska, June 2008.
21. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions via hashing. In *Proceedings of International Conference on Very Large Databases*, Edinburgh, Scotland, UK, September, 1999.
22. S.H. Srinivasan, Neela Sawant, Finding near-duplicate images on the web using fingerprints, In *Proceedings of ACM International Conference on Multimedia*, Vancouver, Canada, October 2008.
23. O. Chum, J. Philbin, and A. Zisserman, Near-duplicate image detection: min-hash and tf-idf weighting, *In Proceedings of British Machine Vision Conference*, 2008.