Clustering Near-Duplicate Images in Large Collections

Jun Jie Foo
School of Computer Science
and Info. Technology, RMIT
University, Australia
jufoo@cs.rmit.edu.au

Justin Zobel
School of Computer Science
and Info. Technology, RMIT
University, Australia
jz@acm.org

Ranjan Sinha
Dept. of Computer Science &
Software Eng., University of
Melbourne, Australia
rsinha@csse.unimelb.edu.au

ABSTRACT

Near-duplicate images introduce problems of redundancy and copyright infringement in large image collections. The problem is acute on the web, where appropriation of images without acknowledgment of source is prevalent. In this paper, we present an effective clustering approach for near-duplicate images, using a combination of techniques from invariant image local descriptors and an adaptation of near-duplicate text-document clustering techniques; we extend our earlier approach of near-duplicate image pairwise identification for this clustering approach. We demonstrate that our clustering approach is highly effective for collections of up to a few hundred thousand images. We also show — via experimentation with real examples — that our approach presents a viable solution for clustering near-duplicate images on the Web.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Clustering*; I.5.3 [**Pattern Recognition**]: Clustering—*Algorithms*

General Terms

Algorithms, Experimentation, Performance

Keywords

clustering near-duplicates, image clustering, near-duplicate detection

1. INTRODUCTION

Duplicate and near-duplicate images — that is, variants derived from the same original image — are common amongst the vast numbers of images on the web. The existence of such near-duplicates in web image searches indicate the presence of redundancy, and may also represent violations of copyright. The detection of such instances is challenging due to the multitude of possible variations of images. Manipulation and re-rendering of images is straightforward, as simple digital editing operations (for example conversion to grayscale, change in colour balance, rescaling, rotating, and cropping), easily defeats simple image duplicate detection methods.

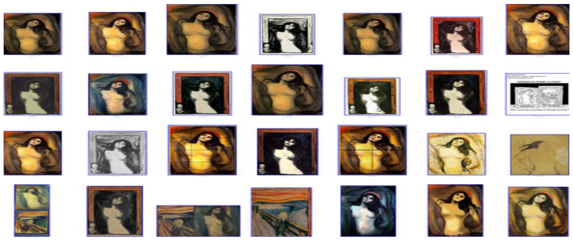

Figure 1: A snapshot of the first 28 image results returned by Google Images using the query "*Edvard Munch Madonna*".

Consider the example in Figure 1, which shows the first 28 image results returned by the Google image search engine[1] for the text-based image query "Edvard Munch Madonna". Most of the returned images are nearly identical; others are clearly derived from one another, reflecting revisions of the same image. From the perspective of a user, it is attractive to be able to group these duplicate and near-duplicate instances into a set, so that a greater variety of relevant images can be presented more effectively; this also allows the user to avoid viewing re-occurrences of the same image (or variants of it) in the result set. Grouping these image variants into their respective sets also enables infringements of copyright to be efficiently detected; each set can be further perused to identify suspect images. Thus, the ability to identify such variants with a reasonable degree of reliability and accuracy would allow detection of copyright violations, and reduce redundancy within collections and during the presentation of search results.

In this paper, we propose a clustering approach — adapted from near-duplicate text document clustering [4] — to identify and cluster near-duplicate images; we show this adaptation by extending our previous method of near-duplicate pairwise image identification [9]. First, we describe the adaptation of this clustering technique to the image domain where images are characterized using *PCA-SIFT* [16], a type of invariant local descriptor. We then provide empirical evidence to show that our proposed clustering approach

MIR'07, September 28–29, 2007, Augsburg, Bavaria, Germany.
Copyright 2007 ACM 978-1-59593-778-0/07/0009 ...$5.00.

[1] http://images.google.com

is highly effective for clustering near-duplicates in large image collections (of up to 300 000 images). Significantly, we demonstrate that it manages such tasks using only modest processing time. Furthermore, we show that our clustering technique offers an effective solution that is indeed practical for web images; we observe high effectiveness in our experiments in clustering real-world near-duplicate examples gathered from the web.

2. BACKGROUND

In the research literature, near-duplicate images have been categorized into three main groups based on the nature of modifications from the original [7, 14, 26]: *scene*, which are changes effected by movement, addition, or occlusion of image subjects or objects; *camera*, which are changes due to differing camera viewpoint or angle; and *image*, which describe changes due to digital manipulation such as filtering, cropping, colour, contrast, or resolution alteration. In this work, we focus on the detection of image variants through image changes, where we assume that the original image and its variants share the same digital source, and that the co-derivation is evident to a human observer. Examples of such include multiple photographs of the same scene, or multiple photographs of a painting; these are near-duplicates that seem almost identical without a common digital source. Although they are not syntactically identical, they may be of interest for the detection of copyright violation.

For near-duplicate image detection, Ke et al. [17] propose using interest-points that are highly robust for object-recognition and *correspondence matching* for the matching of near-duplicate images. For this task, they demonstrate near-perfect accuracy on a collection of 18 000 images using their Scale Invariant Feature Transform (SIFT) [18], characterized by PCA-SIFT local descriptors [16], but scalability was poor even for a moderate-sized collection. To improve scalability, we showed in previous work that pruned SIFT features that are characterized using the same PCA-SIFT descriptors can lead to great gains in efficiency with negligible loss in accuracy for large collections [8].

Meng et al. [20] propose perceptual distance functions — also known as Dynamic Partial Functions (DPF) — for near-duplicate retrieval using CBIR colour and texture image features, wherein they observe limited effectiveness; this method has also been observed to suffer from low effectiveness for severe image alterations [10]. Qamra et al. [23] show that a higher level of effectiveness can be achieved with this method using statistical analysis and sampling, but its efficiency is unclear. Lu and Hsu [19] demonstrate mesh-based image hashing technique for retrieval of near-duplicate images on a collection of 20 000 images, but the method showed only limited effectiveness on even less severe alterations; scalability remains an issue.

Most research in this field approach the problem from the query-based retrieval perspective, whereby, given a query example, the aim is to identify near-duplicate instances — with respect to the query image — in an image collection; clustering near-duplicate images has not been extensively investigated. In early work, Chang et al. [5] proposed the RIME system for clustering of near-duplicate images using a cell-based indexing scheme, where they show high effectiveness for 10 near-duplicate images in a moderate-sized collection of 30 000 images. The effectiveness of their approach was tested on a small number of near-duplicate images containing minor alterations [5]; efficacy on more variations of near-duplicates in larger collections is unknown. Zhang and Chang [26] demonstrate that attributed relational graphs can be applied for near-duplicate image detection using a combination of machine learning and stochastic models to assess the similarity between two near-duplicate images; they use this method to automatically identify near-duplicate pairs within a video dataset. The results observed on the TRECVID video keyframes using this method suffers from low effectiveness [26]; the efficiency is also unclear as only a small dataset of 600 images were tested. Nevertheless, Zhang and Chang [26] were perhaps the first to address the challenges of automatically identifying all near-duplicate image instances within a given collection without the use of a query example.

In more recent work [9], we propose using hash-based probabilistic counting — originally used for near-duplicate text-document detection [24] — for near-duplicate images. We undertook a preliminary investigation in the adaptation of this technique, and showed that it can be used for accurate pairwise identification of near-duplicate images — characterized by local descriptors — in a moderate-sized collection. In this work, by extending this method, we propose a new clustering technique for near-duplicate images by borrowing another technique originally proposed by Broder et al. [4] for clustering near-duplicate text-documents. We describe the adaptation of this clustering algorithm for our application in Section 6. Next, we describe the invariant image local descriptors used in our work.

3. INVARIANT LOCAL DESCRIPTORS

The idea of local descriptors is to detect interesting regions surrounding a certain point that possess properties that are robust to photometric changes and geometric variation, so that each region can be computed as distinctive local descriptors. Broadly, the process of computing local descriptors can be divided into two steps: first, characteristic and robust image points or regions are detected (using interest point detectors [22]) from an image, then distinctive features can be computed based on these detected points or surrounding regions — typically based on the analysis of pixel statistics or gradient information [21] — to be used for matching purposes; then, each local descriptor is usually summarized into a high-dimensional feature vector that can be matched in an L_p-metric space. Thus the number of descriptors is dictated by the cardinality of the set of detected interest points from an image.

Several types of local descriptor have been proposed, including SIFT [18], PCA-SIFT [16], SURF [2], and GLOH [21]. Although some work has been done to evaluate the effectiveness of these descriptors for object recognition and correspondence matching [21, 22]; there is no consensus on the most appropriate choice of local descriptor for the task of near-duplicate image detection. In this work, we apply the PCA-SIFT local descriptors that has been shown to be highly accurate for near-duplicate image matching [8, 17], and more compact than other descriptors [16].

The PCA-SIFT local descriptors were originally proposed as an alternative way to compute SIFT detected interest points or regions.[2] The processes of SIFT interest point detection, and PCA-SIFT local descriptor generation, are as

[2] We use the terms *points* and *regions* interchangeably.

follows. In the initial stage of the SIFT algorithm, all local peaks are identified in various locations and scales using a difference-of-Gaussian (DoG) function computed by a pyramidal scheme. Then, by thresholding using contrast levels and ratios of principal curvature [18], poorly localized and unstable peaks are discarded. Next, after all stable peaks are found, each peak is assigned a dominant orientation for rotation invariance; where orientation is computed using gradients of the regions centered around identified peaks. A stable peak (scale and location information) along with its dominant orientations, is known as a *keypoint* [18]. To compute the local descriptors, the PCA-SIFT uses the vertical and horizontal gradients of the regions surrounding each detected interest keypoint; the x and y gradients are sampled using a 39×39 patch, producing a 3042 element feature vector for each region. This vector is then projected to 36 dimensions using PCA; two local descriptors are deemed a match in this high-dimensional space if their L_2-distance is within 3 000 [16].

In our previous work [8], we showed that the highly distinctive PCA-SIFT descriptors can be used on even a severely pruned set of DoG interest regions. This is achieved using a simple ranking scheme to threshold and reject poorly localized peaks so that only a subset of interest points are considered. We showed that using approximately 100 interest points gives the same level of effectiveness as the original scheme, and results in great gains in efficiency for near-duplicate image matching; we use the pruned version in this work. As a large collection of images can produce up to hundreds of thousands of local descriptors, all local descriptors are first indexed (prior to clustering) using a high-dimensionality index structure as in the work of Ke et al. [17] so that each local descriptor can be more efficiently compared when needed; we describe indexing of local descriptors in the next section.

4. INDEXING LOCAL DESCRIPTORS

We now describe the process of indexing the local descriptors for a given image collection. To index a set of high-dimensional local descriptors, we use the Locality Sensitive Hashing (LSH) — an index scheme [1, 11, 13] for approximate nearest-neighbour matching in high-dimensional spaces. The idea of LSH is to use a family of hash functions to ensure that the probability of collision of two points is correlated to the distance between them.

Given a set of local descriptors (points) $p \in P$, an LSH hash function can be defined as $g_i(p) = (h_1(p), \ldots, (h_k(p))$, for $i = 1, \ldots, l$, where k determines the probability of collision, and l determines the fraction of false negatives [13]. A family of functions can be efficiently computed with a Hamming space H^d for d dimensions [11], whereby each d-dimensional vector $p(x_1, \ldots, x_d)$ can be mapped to a Hamming cube $H^{d'}$ with $d' = Cd$ (where C denotes the largest coordinate in P), turning vector p to a binary Hamming string p' while preserving the properties of a hash function. Given a transformed vector of p', a hash $g_i(p)$, for $i = 1, \ldots, l$, can be obtained by a projection of vector p' onto the coordinate set I_i (where I consists of randomly sampled k elements from $\{1, \ldots, d'\}$) essentially hashing point p to bucket $g_i(p)$. The number of hash buckets, by this approach, can be large depending on the cardinality of set P; thus, a second level of standard hashing is used to map the contents of $g_i(p)$ to a hash table [11]. Hence, there are a total of l LSH indexes, each using the LSH functions from the $H^{d'}$ family. The size of each hash table M is determined using: $M = \frac{n}{\alpha B}$ where n is the total number of points in a collection, and B is maximum number of points in each hash bucket; the utilization parameter to 0.5 as it has been shown to perform well for near-duplicate image detection [17, 8].

When comparing two points using the LSH index, the hash value for a single point is used to retrieve the corresponding hash bucket; the neighbourhood search is limited to only those points that fall within that bucket. Thus, the neighbourhood of an approximate match is reduced considerably to those that share identical hash values. The probability of collision within the index is dictated by the k random bits that are selected to create the hash. A large k value results in a low collision probability; whereas a small k value causes the opposite effect. Using l independent indexes increases accuracy as those that are missed by one index may be retrieved by other indexes. The tunable parameters for the LSH index are l and k, both of which are critical for efficacy. We experiment with these parameters for our clustering algorithm in Section 8.

Using the LSH index, the cost of comparing two images largely depends on the cardinality of the set of descriptors P in any given image. For example, an image with 1000 local descriptors requires 1000 point queries to the LSH index for evaluation. As each image is treated as a *bag-of-points*, this approach can be computationally intensive given a large number of local descriptors. All points that collide within a given hash table are approximated, by the L_1-distance (embedded in the Hamming space), to be closer to each other than those that do not. As most local descriptors are designed to be matched within the L_2-norm [21], there can also be a considerable number of false positive matches as a result of this approximation — one of the drawbacks of this scheme. An additional L_2-norm verification can be used to discard false positive matches under the L_1 [17], but has been observed to have limited impact on effectiveness [11, 9]. Any additional verification during clustering incurs an overhead to efficiency — a crucial factor for clustering of large image collections. In this work, we use the LSH index proposed by Gionis et al. [11] where we assume similarity of two local descriptors by a hash-collision in the L_1-norm, as it has been shown to perform well for retrieval of near-duplicate images [17], and automatic identification of near-duplicate image pairs [9].

5. PROBABILISTIC COUNTING

The concept of hash-based probabilistic counting (HPC) [3, 24] was originally proposed for near-duplicate text document detection; it was adapted in our previous work for the task of identifying near-duplicate image pairs within an image collection [9]. Here, we briefly describe our adaptation of probabilistic counting to local descriptors; then, by extending this approach, we propose a clustering algorithm in the following section.

Using the LSH index, the post-indexing phase of local descriptors of an image can be mapped to a series of *hash-keys* across l indexes, such that two features sharing identical keys are, with high confidence, close to each other (under the L_1-norm) in a d-dimensional space. Our approach is similar to the *visual vocabulary* approach [12, 25], in that

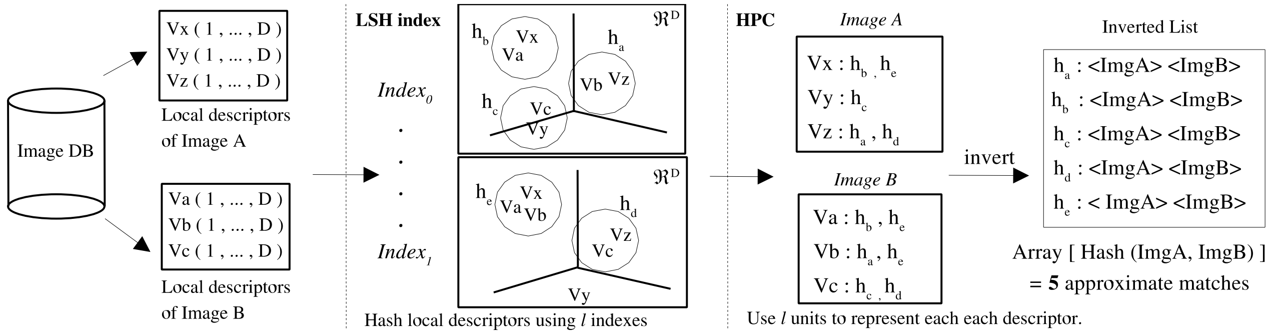

Figure 2: Process of hash-based probabilistic counting of local descriptors using the LSH index and the inverted list.

a set of local descriptors are mapped to cluster centers (or a single representative units) by vector quantization using k-means on some training data [25]. Instead of mapping descriptors to a single unit, we map each local descriptor to l representative units by utilizing the LSH indexes; this has been shown to yield high accuracy [9, 10]. Using this approach, each hash-key can be treated as a set of l units, akin to words in text, and thus an image is transformed into a series of representative units that can be stored in an *inverted list* [27]; wherein each entry consists of a list of images containing the local descriptor that share identical hash-keys. The rationale is that image pairs that share large quantities of identical hash-keys are highly likely to be near-duplicates of each other.

Using the HPC method, all possible image pairs in every entry of the inverted list can be rapidly hashed to an array A of m counters using a universal hash function h [24]. For a pair of images with identifiers (ID) of imgID_1 and imgID_2, that share co-occurring entries (units) in the inverted list, $A[h(\text{imgID}_1, \text{imgID}_2)]$ is incremented; this is essentially a *probabilistic* hash-counter. That is, it may sometimes generate spurious pairs due to hash collisions when image pairs with no co-occurring descriptors share identical hash locations within an LSH index with the ones that do. This phenomenon is typically observed with high probability when incommensurate hash-counters are used. As we are only concerned with matches between two images with at least a minimum number of matching descriptors [9], thresholding was introduced to reject image pairs that do not have sufficient matches. Thus, a small range of threshold values (T) can be implemented such that the width of a hash counter is upper-bounded to reduce the effects of hash-collisions. Once the appropriate pairs have been approximated using the probabilistic hash-counter, matching features between a pool of vastly reduced image pairs can be counted exactly. That is, a tally is kept for every unique image pair using a static structure with no possibility of collision, thereby producing a list of <imgID_1, imgID_2, counter> triplets. As such, each counter reflects the actual number of local descriptor matches between any two images amongst l LSH indexes; this process is illustrated in Figure 2.

In previous work, we demonstrate the applicability of this approach for the identification of near-duplicate image pairs and show high effectiveness [9]. Given that the HPC method uses the LSH index, the efficacy of this method is directly affected by the tunable LSH parameters l and k, and the threshold value T. Later in Section 8, we show — via experimentation of these parameters — that the HPC method can be extended to accurately cluster near-duplicate images into non-overlapping sets within large image collections. We describe the clustering algorithm in the following section.

6. THE CLUSTERING ALGORITHM

The identification of near-duplicate image pairs can be conceptualized as a graphing problem [3, 9], where each node represents a unique image, and the presence of an undirected edge between two nodes reflects an existing near-duplicate relationship between the images. As such, a collection of N images can be represented as a weighted graph of $G = (V, E)$; where a set of nodes $V = \{1, \ldots, N\}$ represents the images, and $E = \{s(i, j) : i, j \in V\}$ represents the set of edges between every pair of nodes in the graph. $s(i, j)$ denotes the approximated local descriptor matches between two nodes of i and j.

Once an image collection is processed by the HPC method, each triplet $\langle \text{imgID}_1, \text{imgID}_2, \text{counter} \rangle$ can be seen as two nodes with a weighted edge between them, where the number of approximated local descriptor matches (henceforth referred to as *similarity*) is reflected by the edge weight, Thus the aim is to find an efficient solution for discriminating between unique images and near-duplicates, and to accurately form non-overlapping clusters for each near-duplicate set; this is akin to identifying disjoint partitions within a graph.

We propose adapting the clustering approach by Broder et al. [4] — originally used for text documents — for near-duplicate images. They show that using triplets of ⟨docID, docID, count⟩ generated from text documents can be used for effective clustering. By examining each triplet, they unite (add an edge in between) two nodes (documents) from an *union-find* algorithm if the pair of the examined triplet has a count of common *shingles* [4] exceeding a certain threshold. A shingle is simply the smallest unit by which a text document can be partially represented, similar to the representative units of hash-keys used in the HPC method. When all triplets have been examined, clusters are created by identifying all the disjoint sets of connected nodes. Although the generation of the triplets (as used by Broder et al.) uses exact counting of common shingles, which is less scalable than the probabilistic approach of Shivakumar and Garcia-Molina [3, 24] (as adapted in the HPC method), the proposed clustering method remains viable. That is, the scalability issues imposed by the algorithm of Broder et al. is limited to the process of identifying document pairs.

We conjecture that near-duplicate images form natural clusters that can be accurately identified by this approach; a high similarity between two near-duplicate images is often indicated by a large number of matching local descriptors. Thus, by using a suitable threshold, we can effectively gather near-duplicates into their respective clusters. We can then create m clusters of $\{C_1, \ldots, C_m\}$ where two non-overlapping clusters can be defined as: $C_a \bigcap C_b = \emptyset$, $a \neq b$ and $\bigcup_{a=1}^{m} C_a = V'$. Using this algorithm, clusters are formed for only the set of nodes (V') with non-zero weighted edges, that is, image pairs without any matching local descriptors are not considered during clustering. Thus, the clustering algorithm is performed on $G'(V', E')$, for $V' \subseteq V$ where $E' = \{s(i,j) \geq T : i, j \in V\}$, and T is the inclusion threshold for the nodes in each triplet.

The threshold T used for clustering is similar to that used for identifying image pairs (as discussed in Section 5). However for clustering, we conjecture the value of T to be significantly different due to the transitive property of the near-duplicate relationship. That is, for three near-duplicate images of A, B, and C, where A shares high similarity with B but not with C (due to its indirect derivation from B), images A and C remains associated due to the approximated matches between images B and C that exceed the threshold. Thus a pair of nodes within a cluster may have an edge weight below a certain threshold T but are indirectly associated via another node within the same cluster. In this work, we empirically determine a suitable threshold for clustering near-duplicates by experimenting with the parameter T, the *cluster threshold*.

7. EVALUATION METHODOLOGY

We now describe the experiments used to empirically demonstrate the effectiveness of our clustering approach. First, we evaluate the effect of varying the parameters l and k — the choice of which is critical for large image collections — of the LSH index to determine the optimal settings by which to perform clustering. The aim is to study the impact of these parameters on the effectiveness of near-duplicate pairwise image identification — the HPC method — that are subsequently used for clustering.

The effectiveness of this experiment can be measured with the *coverage* measure using artificially generated edges from the nodes (images) in a test collection. Artificial edges (or reference edges [9]) can be generated using a collection of known or predefined near-duplicate image pairs. For example, 10 groups of 5 near-duplicate images each will generate $10 \times \frac{5\times4}{2} = 100$ edges. Thus, coverage — similar to the recall metric in document retrieval literature — can be defined as [3, 9]: Coverage $= \frac{E_I}{E_R}$, where E_I denotes the total number of algorithm-identified edges that also appear in the set of artificial edges, and E_R indicates the total number of artificial edges. Second, using a series of experiments, we observe the effects of l and k on the clustering algorithm both in terms of accuracy, size of identified clusters, and the number of identified edges for a given collection; we also perform timing experiments. We study the effect of varying the threshold value T to determine its effects on the quality of clusters.

To measure the quality of the generated near-duplicate clusters, we borrow two measures from Chen et al. [6], namely *purity* and *entropy*. Given a set of N images that belong to c distinctive near-duplicate groups — denoted by $1, \ldots, c$ — that are formed into m clusters C_j, $j = 1, \ldots, m$, purity for a single cluster C_j can be defined as [6]:

$$p(C_j) = \frac{1}{|C_j|} \max_{k=1,\ldots,c} |C_{j,k}|$$

where $|C_j|$ is the size of the cluster (number of images), and $C_{j,k}$ denotes a set of images in cluster C_j that belong to a near-duplicate set k. Note that c and m are independent variables as the ideal number of clusters and the number of algorithm-identified clusters do not necessarily coincide.

Purity is the ratio of the dominant near-duplicate set within a cluster to the cluster size. This measure is similar to the precision metric in retrieval literature, since $C_{j,k}$ returns the number of relevant images in the dominant near-duplicate set of the cluster. Entropy for cluster C_j is defined as [6]:

$$h(C_j) = -\frac{1}{\log c} \sum_{k=1}^{c} \frac{|C_{j,k}|}{|C_j|} \log \frac{|C_{j,k}|}{|C_j|}$$

This measure is used to quantify the distribution of the different near-duplicate sets within a cluster; values are normalized to be between 0 and 1. A low-entropy value indicates that the cluster consists of primarily a single near-duplicate set, whereas a high-entropy value reflects a mixture of different sets. Thus, in contrast to purity, an ideal algorithm will form all clusters with entropy values of 0, as each cluster contains only images that are near-duplicate instances of one other. Additionally, we also measure the average purity and entropy of each near-duplicate set, respectively defined as:

$$p(k) = \frac{1}{m_k} \sum_{j=1}^{m_k} p(C_j) \quad , \quad h(k) = \frac{1}{m_k} \sum_{j=1}^{m_k} h(C_j)$$

where m_k denotes the number of clusters containing images from near-duplicate set k.

Although purity and entropy are informative measures, they are insufficient indicators of the content of each cluster [6]. To this end, we also measure and report recall and precision — used in standard IR literature — of the near-duplicate set, which are respectively defined as:

$$R(k) = \frac{\sum_{j=1}^{m_k} |C_{j,k}|}{|k|} \quad , \quad Pr(k) = \frac{\sum_{j=1}^{m_k} |C_{j,k}|}{\sum_{j=1}^{m_k} |C_j|}$$

where $|k|$ indicates the size of the known near-duplicate set k. When used with purity and entropy along with m_k, recall and precision can provide an accurate indicator as to the quality of a cluster in terms of completeness and clustering precision.

Image Collections

To generate artificial clusters for our experiments, we select 200 unique images from the Corel Photo CD collection and apply 50 alterations to each image. Thus each of the selected 200 images has 50 altered versions, resulting in 10 000 *seed* images that form 200 unique non-overlapping clusters. To create our first collection C_1, we use all 10 000 seed images and randomly select an additional 140 000 images from the SPIRIT collection [15] to serve as noise, thereby creating a collection of 150 000 images. We then create a larger collection C_2 of 300 000 by adding a completely different set of 290 000 randomly picked images from the SPIRIT collection.

We discard images that share identical checksums or URLs to reduce the possibility of identical images within the noise collection; also, we choose only images of similar sizes to the seed collection to ensure uniform quality. A separate collection is used to create the seed collection, as those from the SPIRIT collection are gathered from the web. This way, we reduce the likelihood of the seed images having duplicates or near-duplicates within the SPIRIT collection unbeknownst to us, the presence of which could affect the experimental results.

The list of alterations — similar to that of the works of Ke et al. [17] and Qamra et al. [23] — is as follows:

1. `format change`: format change from .jpg to .gif (1 type)
2. `colorize`: each of the red, green, and blue channels are tinted by 10% (3 types)
3. `contrast`: increase and decrease contrast (2)
4. `severe contrast`: increase and decrease contrast 3× of original image (2)
5. `crop`: crop 95%, 90%, 80%, and 70% of image, preserve center region (4)
6. `severe crop`: crop 60%, 50%, and 10% of image, preserve center region (3)
7. `despeckle`: apply "despeckle" operation of ImageMagick (1)
8. `frame`: a frame size 10% of image is added using random colors (4)
9. `rotate`: rotate image (by 90^o, 180^o, and 270^o) about its center (3)
10. `scale-up`: increase scale by 2×, 4×, and 8× (3)
11. `scale-down`: decrease scale by 2×, 4×, and 8× (3)
12. `saturation`: alter saturation by 70%, 80%, 90%, 110%, and 120% (5)
13. `intensity`: alter intensity level by 80%, 90%, 110%, 120% (4)
14. `severe intensity`: alter intensity level by 50% and 150% (2)
15. `rotate+crop`: rotate image (by 90^o, 180^o, and 270^o), crop 50% in center region (3)
16. `rotate+scale`: rotate image (by 90^o, 180^o, and 270^o), decrease scale 4x (3)
17. `shear`: apply affine warp on both x and y axes using 5^o, and 15^o (4)

All images are first converted to grayscale and resized to 512 pixels on the longer edge prior to extraction of local descriptors. The number in parentheses indicates the number of instances for each alteration type.[3]

To test our clustering algorithm on real near-duplicate examples on the Web, we use approximately 14 000 images retrieved from 20 queries using Google Images from the web. The 20 subjects used in this collection are selected from Google ZeitGeist 2005:[4] *50 cent*, *Angelina Jolie*, *David Beckham* , *Carmen Electra*, *Britney Spears*, *The Simpsons*, *South Park*, *Garfield*, *Ferrari*, *Lamborghini*, *Batman*, *Harry Potter*, *Yoda*, *Spiderman*, *Superman*, *Bob Marley*, *Tupac*, *Kurt Cobain*, *Aaliyah*, and *Terri Schiavo*. We use the same collection as in our previous work, where this collection was manually clustered by their near-duplicates [10]. Each of the 20 queries were used to retrieve the image results; each result set is then manually evaluated using the top 20 unique images as examples by which to find other near-duplicate instances within the result set. We use only 190 human-evaluated clusters from this collection; we discard singleton clusters and those that contain images that are unusable.[5]

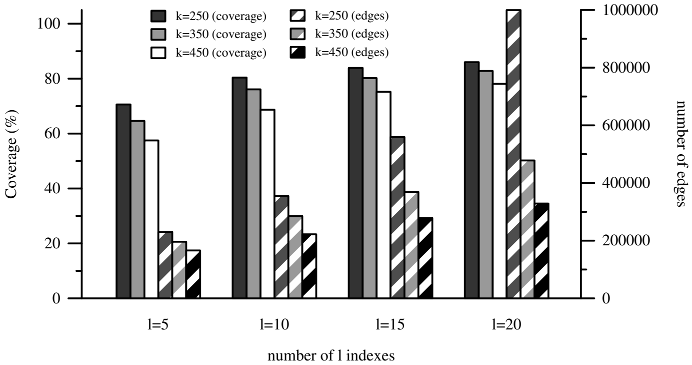

Figure 3: Effects of l and k on the identification of all image pairs and clusters on collection C_1 of 150 000 images; we report coverage (%) and the number of identified edges; results are reported for variations of LSH parameters l (from 5 to 20) and k (from 250 to 450) using a threshold $T = 8$.

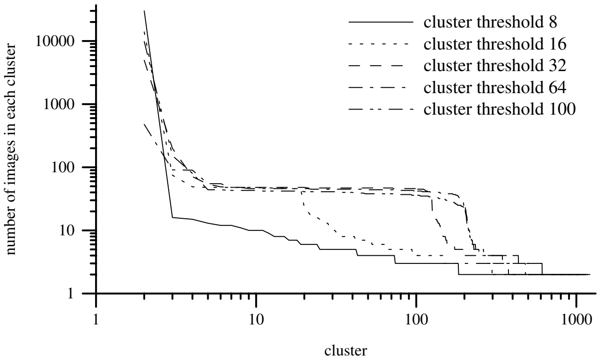

Figure 4: Distribution of the number of images for the identified clusters using LSH parameters of $l = 10$ and $k = 250$; each line in the graph denotes a different threshold value (from 8 to 100).

All experiments are run on a two-processor Xeon 3 GHz machine with 4 GB of main memory running Linux 2.4.

8. RESULTS

We now present results for our clustering algorithm on collections C_1, C_2, and C_3. We first provide experimental results using varying thresholds of l and k on collection C_1. Then, using the optimal settings of l and k, we present clustering results on the larger collection C_2 and the web collection C_3, in the following subsections.

In Figure 3, we show the effects of varying l and k on the number of identified edges and the coverage for 150 000 images (collection C_1). These results are for a single threshold $T = 8$, similar trends are observed for clustering threshold values ranging from 8 to 100. As expected, a larger l in-

[3] All alterations are created using ImageMagick, `http://www.imagemagick.com`.

[4] `http://www.google.com/press/zeitgeist/archive.html`

[5] Images were evaluated using their thumbnails; thus, not all evaluated exists within the collection [10]. The list of image URLs returned by the Google index is available at `http://www.cs.rmit.edu.au/~jufoo/ZeitGeistURLS.tgz`

creases the number of identified edges, as more image pairs are likely to have matching local descriptors that exceed the given threshold; this trend is also observed for the coverage of the near-duplicate edges. We find that the number of edges can be sharply reduced by increasing the parameter k with only a slight drop in coverage.

As shown in Figure 3, using $l = 20$ and $k = 250$ yields a coverage of 86% with nearly one million edges; this is undesirable given that there are only 245 000 artificial edges in the test collection — $200 \times \frac{50(49)}{2}$. Using too small a value for l may reduce the number of identified edges and adversely affect coverage. However, we observe that using $l = 10$ indexes and $k = 250$ strikes a good balance between the number of identified edges and the level of coverage; $l = 10$ and $k = 250$ yields coverage of 70% with approximately 230 000 edges. We use this configuration for subsequent clustering experiments.

Figure 4 shows the distribution of the number of images in the formed clusters as observed on collection C_1. On average, over all threshold values, approximately 1 000 non-overlapping clusters are formed for this collection. This observation is expected as the majority of the images are gathered from unknown sources and we expect some clusters to be formed from existent duplicates or near-duplicates unbeknownst to us. As shown in Figure 4, for clustering threshold values ranging from 8 to 100, the number of images (approximately 40–50) remains consistent for approximately 200 clusters. The observed results also indicate that the majority of the clusters contain only a single pair of near-duplicate images. This is a satisfying result as the collection C_1 contains 200 artificial near-duplicate groups, each containing 50 near-duplicate images. Even though these clusters do not explicitly reflect the 200 artificial groups, the distribution of the images provide an indication of cluster formation.

We observe that the number of clusters declines more gradually with increasing clustering threshold value; for instance, the difference between threshold values of 8 and 100 is evident with the size of the largest cluster. The relatively skewed distribution of images in clusters formed by $T = 8$ and $T = 16$ hints that a small threshold value may be counter-productive for clustering. The transitive property of the edges can cause images to be less selectively linked to one another, resulting in large clusters to be formed. As discussed in Section 7, an ideal algorithm would form m clusters such that $m = c$, where $c = 200$ artificial groups. However, the number of identified clusters exceeds that of the artificial groups, hinting that some groups may have been over-clustered.

Figure 5 shows the average number of clusters formed for each of the 200 artificial near-duplicate groups. These results are observed for clustering threshold T ranging from 8 to 100. We observe that, on average, large clustering thresholds tend to over-cluster; the average number of clusters for each group (known as m_k) approaches 3. We believe this result can be explained by the various alterations of near-duplicate images that exist within each artificial group; we observe that the more severe alterations of near-duplicate images tend to form individual clusters due to low-weighted edges that do not rise above the threshold. Nevertheless, this is a positive result considering the relatively low number of algorithm-formed clusters per group (m_k) against the number and variation of near-duplicate images in each group.

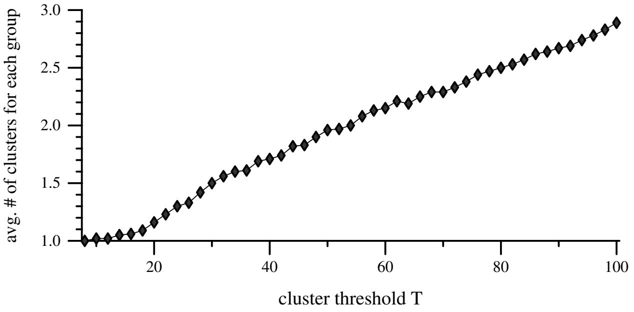

Figure 5: Average number of algorithm-formed clusters for each of the 200 artificial groups, for different clustering threshold T (from 8 to 100) using LSH parameters of $l = 10$ and $k = 250$.

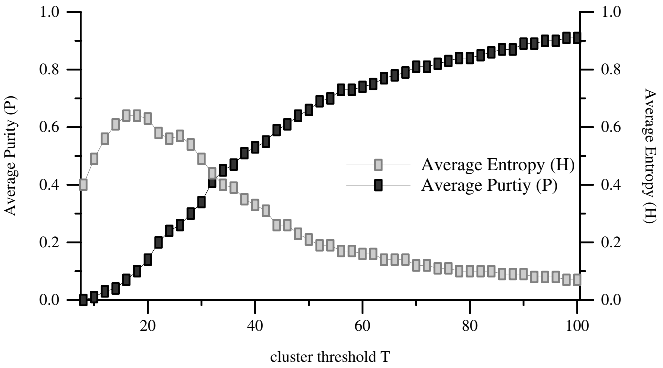

Figure 6: Average purity $p(k)$ and average entropy $h(k)$ over 200 near-duplicate groups, for different clustering threshold T (from 8 to 100) and LSH parameters of $l = 10$ and $k = 250$.

8.1 Cluster Quality and Accuracy

Figure 6 shows the average purity and average entropy of the clusters in collection C_1; these measure are defined in Section 7). Figure 7 shows the average recall and average precision of the clusters (over all the 200 artificial groups). The results in these two figures are observed over a range of clustering threshold T values, ranging from 8 to 100. As shown in Figure 6, we observe that by increasing the threshold value, average purity of the clusters approaches 1 whereas average entropy approaches 0. It also indicates that small clustering threshold values do not produce quality clusters; we begin to observe higher cluster quality with threshold values of approximately 70 and above.

Figure 7 shows that small clustering threshold values yield low average precision and high average recall; large threshold values yield the opposite effect. We observe that the reductions in average recall is relatively gradual and remains above 80% with the largest threshold value of 100, where it converges with average precision of 80%. The results from both figures indicate that our clustering algorithm favours large threshold values, yielding high average recall and precision, with high purity and low entropy. Thus, overall, we observe that our clustering algorithm excels at forming high quality clusters, given reasonably large threshold values.

The results presented so far indicate the average cluster

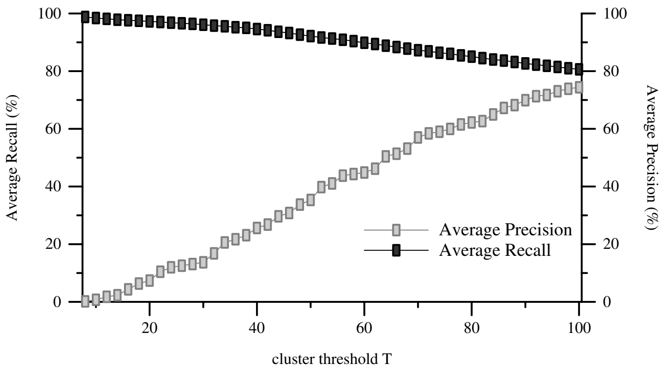

Figure 7: Average recall and precision over 200 near-duplicate groups, for different clustering threshold T (from 8 to 100) using LSH parameters of $l = 10$ and $k = 250$.

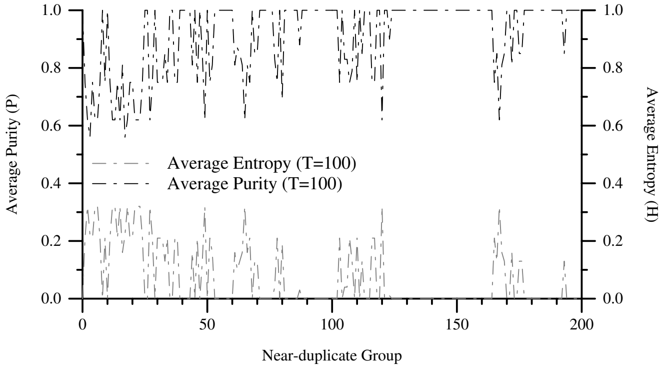

Figure 8: Average purity $p(k)$ and average entropy $h(k)$ for each of the 200 near-duplicate groups using cluster threshold of $T = 100$, with LSH parameters of $l = 10$ and $k = 250$.

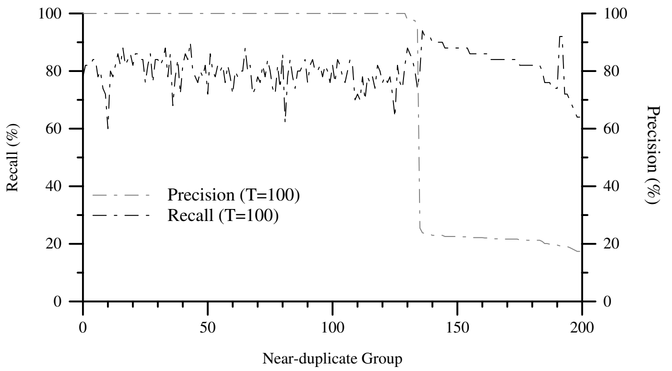

Figure 9: Recall and precision (%) (sorted by precision) for each of the 200 near-duplicate groups using cluster threshold of $T = 100$, with LSH parameters of $l = 10$ and $k = 250$.

quality over 200 artificial groups. To further evaluate the quality of clusters, we report the average entropy, purity, recall, and precision for each of 200 near-duplicate groups; these results are presented in Figure 8 and Figure 9. The results are observed using a cluster threshold (T) of 100, as it yields the best tradeoff amongst all measures. As indicated in Figure 8, the majority of identified clusters for each of the 200 near-duplicate groups contain high concentration of correctly identified images; this is indicated by an overall high average purity and low average entropy. We also observe that the majority of the near-duplicate groups have average purity over 0.8 and average entropy below 0.2 — indicative of the high effectiveness of our clustering algorithm.

Figure 9 shows that the recall and precision for clusters in majority of the near-duplicate groups are pleasingly high. For all groups with 100% precision — approximately 145 of 200 groups — the level of recall is observed to range between 60% to 90%. The drop in precision of clusters in some of the near-duplicate groups indicate that our clustering algorithm occasionally fails to correctly categorize images into their respective clusters, resulting in a mixed bag of images within each cluster, thus, yielding a relatively low level of precision albeit a slightly higher level of recall. The two relatively distinct levels of precision can be attributed to the precision measure (see Section 7), where it measures the ratio of correctly identified images in all algorithm-formed clusters for each of the 200 near-duplicate groups. This results in a more stringent measure, as a low precision level of a single cluster (with the group) can effect an abrupt drop in the overall precision.

Overall, the results indicate that our clustering algorithm is indeed effective for accurately distinguishing relevant images into their respective clusters most of the time — on a large collection (150, 000 images) — at the expense of approximately 10 misses (on average) for every 50 near-duplicate images considered during clustering; as indicated by the average recall of all near-duplicate groups.

8.2 Further Studies

To study the effectiveness of our clustering approach on an even larger collection, we perform the same experiments on collection C_2 of 300, 000 images using the optimal settings ($l = 10$, $k = 250$, and $T = 100$). We find the overall level of effectiveness to be nearly lossless in comparison to that of collection C_1. Using cluster threshold T of 100 for collection C_2, we observe that the average number of clusters formed for each of the 200 near-duplicate groups remains at 2.89, which indicates that although the collection has doubled in size, the identified number of clusters for each artificial near-duplicate group remains relatively constant. As with collection C_1, our clustering algorithm for collection C_2 achieves average recall and average precision of 80.7% and 76.4%, respectively, for the clusters within the 200 near-duplicate groups. The average recall and precision for collection C_1 were 80.6% and 74.4%, respectively. The average purity and average entropy for collection C_2 were observed to be 0.89, and 0.09, respectively; the average purity and average entropy were, respectively, 0.91, and 0.07 for collection C_1. Overall, these results demonstrate that our proposed clustering technique is indeed effective and that the observed efficacy is less dependent to changes in collection size.

To gain insight into the efficiency and performance of our clustering technique with regards to collection C_2, we compare the total time required to identify and cluster the near-duplicate images for both collection C_1 and C_2. We also compare the number of identified edges in each collection to determine the impact of the larger collection on efficiency. Figure 10 shows the total run-time and the number of identified edges for both collections using cluster threshold T of

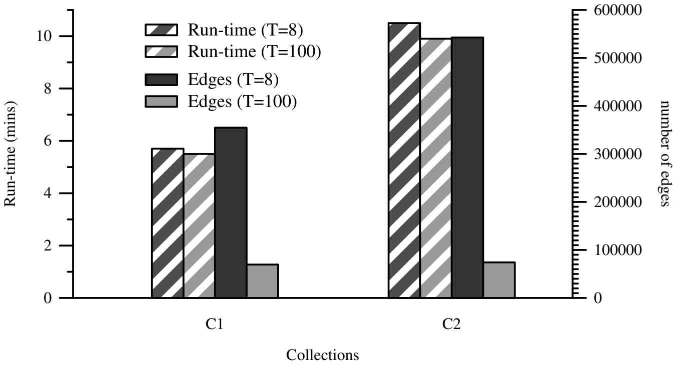

Figure 10: Cost of identification and clustering of all near-duplicate images within two image collections C_1 of $150,000$ images, and C_2 of $300,000$ images; the figure shows the timing results and the number of identified edges using cluster threshold of $T = 8$ and $T = 100$, with LSH parameters of $l = 10$ and $k = 250$.

8 and 100. We use two different threshold values to observe the impact of small and large threshold values on efficiency. As shown in Figure 10, using cluster threshold of 8, the time to process collections C_1 and C_2 are 5.7 minutes and 10.5 minutes, respectively.

We also observe in Figure 10, a linear growth in the number of edges from approximately 360 000 to 540 000. A cluster threshold T of 100 significantly reduces the number of edges to 70 000 and 74 000, respectively, for collections C_1 and C_2. This is an important result and shows that a large cluster threshold value can both restrict the number of edges as the collection size is increased, as well as maintain high effectiveness in clustering. The relatively unaffected run-time with regards to the cluster threshold T (from 8 to 100 in both collections) is within our expectation; the hash-counters used during pairwise identification of all image pairs has a cost of $O(n)$; it requires two images to be approximately counted for every matching local descriptor in each of the l LSH indexes to determine the number of edges within the given threshold. Thus, run-time remains unaffected regardless of the cluster threshold value and shows that a large collection can be processed and effectively clustered using our clustering algorithm with modest processing time.

To further validate the results of our clustering algorithm, we perform the same clustering experiments using collection C_3 (contains images retrieved from Google Images). Table 1 reports the results for our clustering algorithm on this collection for each of the 20 subject queries used. Columns two and three list the number of human-evaluated clusters and the number of algorithm-formed clusters for each subject query, respectively. The measures average recall, average precision, average purity, and average entropy for the algorithm-formed clusters within each of the 20 subjects are shown in columns four to seven. The results in this table show that the average number of algorithm-formed clusters for each human-evaluated cluster is approximately 1.7. This indicates that with our clustering algorithm, each human-evaluated clusters are not over-clustered into more than 2 groups, on average. Although the average recall is not as high as those observed for collections C_1 and C_2, it remains at a competitive level of 70% recall and 80% precision, on average.

Table 1: Clustering results for the collection C_3 (Web collection) for 20 different subjects; each subject consists of c human-evaluated clusters. Column three shows the total number of algorithm-identified clusters that contain images from the c clusters of each category. Columns four and five report the average recall and precision (%) of the identified clusters, respectively. Average purity and average entropy are also reported, respectively, in the final two columns.

Query Subject	c	Total m_k	Avg. Rec.(%)	Avg. Prec.(%)	Avg. p(k)	Avg. h(k)
50 cent	14	25	77.57	68.80	0.71	0.04
aaliyah	11	24	74.27	72.63	0.79	0.04
angelina	10	6	86.11	100.00	1.00	0.00
batman	8	11	65.83	84.64	0.78	0.03
marley	14	20	54.58	81.14	0.78	0.03
spears	6	3	63.33	68.97	0.70	0.05
electra	11	8	70.79	88.89	0.89	0.02
beckham	9	6	94.44	100.00	1.00	0.00
ferrari	7	7	80.95	89.29	0.89	0.02
garfield	7	5	82.50	79.55	0.60	0.06
potter	11	11	57.19	76.07	0.85	0.02
cobain	12	23	68.34	73.01	0.76	0.03
lamborghini	7	8	75.00	93.97	0.94	0.01
south park	10	13	71.50	67.50	0.63	0.05
spiderman	7	11	51.91	65.04	0.75	0.03
superman	7	10	69.92	78.92	0.75	0.03
schiavo	10	22	68.37	81.05	0.77	0.04
simpsons	8	5	68.33	85.42	0.83	0.02
tupac	10	16	58.22	84.64	0.88	0.02
yoda	11	11	66.36	85.16	0.87	0.03

We expect this level of effectiveness for this collection as there exists some very severe alterations within each of the subject groups in the collection [10]. The relatively lower recall value can also be explained by the smaller number of near-duplicate images in each cluster as compared to the artificial collections; hence, the fluctuation in recall level is more abrupt. Also indicated in Table 1, the quality of the clusters in each subject group remains high, at an observed average of 0.82 and 0.03 for purity and entropy, respectively. Overall, the results are highly pleasing as they indicate the effectiveness of our clustering technique for both large image collections, and real world examples on the Web. We show some examples of the algorithm-identified clusters in Figure 11.

9. CONCLUSIONS AND FUTURE WORK

We have demonstrated a new image clustering algorithm that combines techniques from computer vision and text document clustering methods. We have successfully adapted a text-oriented clustering algorithm to the image domain. The results indicate that our approach yields high effectiveness — even for large collections — when coupled with PCA-SIFT invariant local descriptors; our clustering algorithm has been shown to effectively generate non-overlapping clusters containing large concentrations of relevant near-duplicate images belonging to the same set. Overall, our clustering

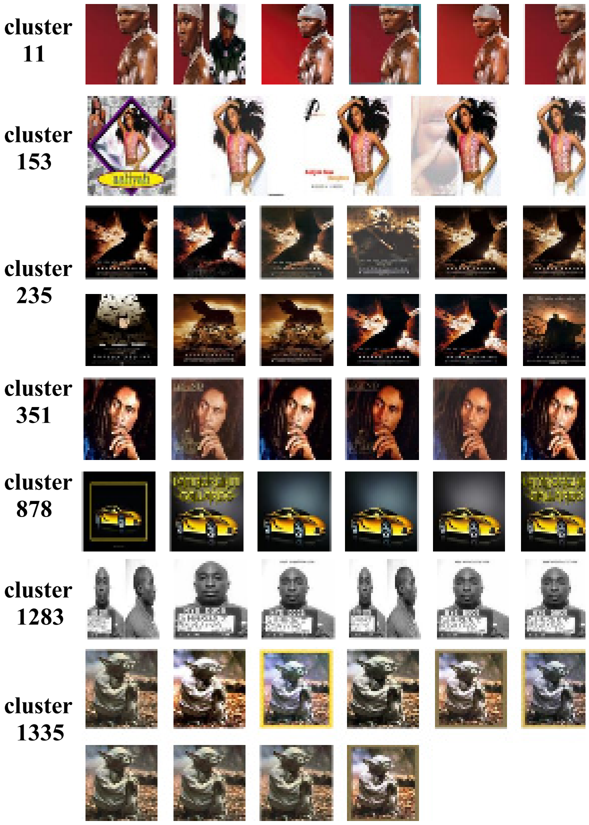

Figure 11: Examples of some clusters that are identified by our approach — using the same settings as those reported in our experiments — on images retrieved from the Web (Google Images). The numbers of the left column denote the cluster IDs.

algorithm offers a promising approach for organizing near-duplicate images in large collections such as the Web, and presents a plausible solution to the challenges of image redundancy inherent in image search engines. In future work, we intend to explore more efficient detection and retrieval of near-duplicate images using this clustering approach; we plan to investigate the scalability of our approach, and also to perform a comparative evaluation against predominant clustering techniques such as the k-means.

10. ACKNOWLEDGMENTS

This work was supported by the Australian Research Council.

11. REFERENCES

[1] M. Bawa, T. Condie, and P. Ganesan. LSH forest: Self-tuning indexes for similarity search. In *Proc. WWW*, pages 651–660, May 2005.

[2] H. Bay, T. Tuytelaars, and L. V. Gool. Surf: Speeded up robust features. In *Proc. ECCV*, May 2006.

[3] Y. Bernstein and J. Zobel. A scalable system for identifying co-derivative documents. In *Proc. SPIRE*, pages 55–67, October 2004.

[4] A. Z. Broder, S. C. Glassman, M. S. Manasse, and G. Zweig. Syntactic clustering of the web. *Computer Networks*, 29(8-13):1157–1166, 1997.

[5] E. Y. Chang, C. Li, J. Z. Wang, P. Mork, and G. Wiederhold. Searching near-replicas of images via clustering. In *Proc. SPIE*, pages 281–292, September 1999.

[6] Y. Chen, J. Z. Wang, and R. Krovetz. CLUE: Cluster-based retrieval of images by unsupervised learning. *IEEE Trans. on Image Processing*, 14(8):1187–1201, 2005.

[7] O. Chum, J. Philbin, M. Isard, and A. Zisserman. Scalable near identical image and shot detection. In *Proc. CIVR*, July 2007.

[8] J. J. Foo and R. Sinha. Pruning SIFT for scalable near-duplicate image matching. In *Proc. ADC*, pages 63–71, January 2007.

[9] J. J. Foo, R. Sinha, and J. Zobel. Discovery of image versions in large collections. In *Proc. MMM*, pages 433–442, Jaunary 2007.

[10] J. J. Foo, J. Zobel, R. Sinha, and S. Tahaghoghi. Detection of near-duplicate images for web search. In *Proc. CIVR*, July 2007.

[11] A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions via hashing. In *Proc. VLDB*, pages 518–529, September 1999.

[12] K. Grauman and T. Darrell. Efficient image matching with distributions of local invariant features. In *Proc. CVPR*, pages 627–634, June 2005.

[13] P. Indyk and R. Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proc. STOC*, pages 604–613, May 1998.

[14] A. Jaimes, S.-F. Chang, and A. C. Loui. Duplicate detection in consumer photography and news video. In *Proc. MM Int. Conf. on Multimedia*, pages 423–424, December 2002.

[15] H. Joho and M. Sanderson. The spirit collection: an overview of a large web collection. *ACM SIGIR Forum*, 38(2):57–61, 2004.

[16] Y. Ke and R. Sukthankar. PCA-SIFT: A more distinctive representation for local image descriptors. In *Proc. CVPR*, pages 506–513, June 2004.

[17] Y. Ke, R. Sukthankar, and L. Huston. An efficient parts-based near-duplicate and sub-image retrieval system. In *Proc. MM*, pages 869–876, October 2004.

[18] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. Journal of Computer Vision*, 60(2):91–110, 2004.

[19] C.-S. Lu and C.-Y. Hsu. Geometric distortion-resilient image hashing scheme and its applications on copy detection and authentication. *Multimedia Systems*, 11(2):159–173, 2005.

[20] Y. Meng, E. Y. Chang, and B. Li. Enhancing dpf for near-replica image recognition. In *Proc. CVPR*, pages 416–423, June 2003.

[21] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. In *Proc. CVPR*, pages 257–263, June 2003.

[22] K. Mikolajczyk and C. Schmid. Scale and affine invariant interest point detectors. *Int. Journal of Computer Vision*, 60(1):63–86, 2004.

[23] A. Qamra, Y. Meng, and E. Y. Chang. Enhanced perceptual distance functions and indexing for image replica recognition. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 27(3):379–391, 2005.

[24] N. Shivakumar and H. Garcia-Molina. Finding near-replicas of documents and servers on the web. In *Proc. WebDB*, pages 204–212, March 1998.

[25] J. Sivic and A. Zisserman. Video Google: A text retrieval approach to object matching in videos. In *Proc. ICCV*, volume 2, pages 1470–1477, October 2003.

[26] D. Zhang and S.-F. Chang. Detecting image near-duplicate by stochastic attributed relational graph matching with learning. In *Proc. MM*, pages 877–884, October 2004.

[27] J. Zobel and A. Moffat. Inverted files for text search engines. *Computing Surveys*, 38:1–56, 2006.