Scalable Near Identical Image and Shot Detection

Ondřej Chum[1] James Philbin[1] Michael Isard[2] Andrew Zisserman[1]

[1]Department of Engineering Science, University of Oxford
[2]Microsoft Research, Silicon Valley

ABSTRACT

This paper proposes and compares two novel schemes for near duplicate image and video-shot detection. The first approach is based on global hierarchical colour histograms, using Locality Sensitive Hashing for fast retrieval. The second approach uses local feature descriptors (SIFT) and for retrieval exploits techniques used in the information retrieval community to compute approximate set intersections between documents using a min-Hash algorithm.

The requirements for near-duplicate images vary according to the application, and we address two types of near duplicate definition: (i) being perceptually identical (e.g. up to noise, discretization effects, small photometric distortions etc); and (ii) being images of the same 3D scene (so allowing for viewpoint changes and partial occlusion). We define two shots to be near-duplicates if they share a large percentage of near-duplicate frames.

We focus primarily on scalability to very large image and video databases, where fast query processing is necessary. Both methods are designed so that only a small amount of data need be stored for each image. In the case of near-duplicate shot detection it is shown that a weak approximation to histogram matching, consuming substantially less storage, is sufficient for good results. We demonstrate our methods on the TRECVID 2006 data set which contains approximately 165 hours of video (about 17.8M frames with 146K key frames), and also on feature films and pop videos.

Categories and Subject Descriptors

H.3.1 [**Content Analysis and Indexing**]: Indexing methods; I.4 [**Image Processing and Computer Vision**]: Image Representation—*hierarchical, statistical*; E.2 [**Data Storage Representations**]: Hash-table representations

General Terms

Algorithms, Theory

Keywords

Near duplicate detection, LSH, Min Hash, Large image databases

1. INTRODUCTION

An image is called a near-duplicate of a reference image if it is "close", according to some defined measure, to the reference image. Near duplicate image detection (NDID) and retrieval is a vital component for many real-world applications.

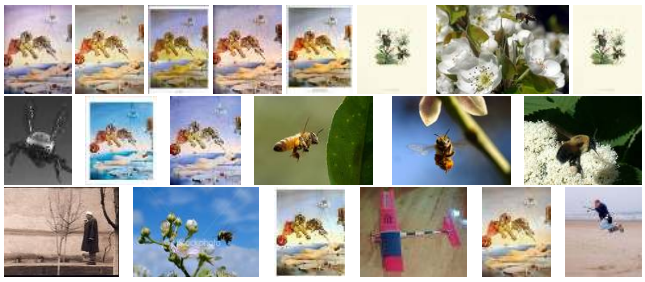


Figure 1: First page of results for the query 'flight of a bee' using Google Images.

Figure 2: First page of results for the query 'Munch Vampire' using Google Images.

Consider the following example. Searching for the phrase 'flight of a bee' in a popular internet image search engine (here, Google Images), gives the first page of results shown in figure 1. Many of these results show Salvador Dali's painting 'Dream Caused by the Flight of a Bee Around a Pomegranate One Second Before Awakening,' and are perceptually identical. A user might prefer the search engine to "collapse" these images of the painting into a set, represented visually by a single reference image, so that a greater diversity of images is initially displayed. If the user wants an image of the painting, he could then click on the reference example to explore the near-duplicate set. If the painting isn't desired, he doesn't have to view many near-duplicate occurrences of it. However, the images are not identical. They differ in size, color adjustment, compression level, etc. Therefore, exact duplicate detection (at the pixel level) will not be able to group all similar results together.

A second example of an image search result is shown in figure 2. There exist several different versions of the painting, 'The Vampire,' by Edvard Munch, but this is not immediately apparent from the search results. Grouping all the near-duplicates together so that distinct versions appear as distinct groups is preferable.

Video processing is another area where NDID can prove extremely useful. Detection of identical frames or shots (sequences

CIVR'07, July 9–11, 2007, Amsterdam, The Netherlands.

of frames) can reduce the amount of data that need to be stored and speed up further video analysis once the duplicates are removed. This is especially true in video streams composed of news clips and advertisements (repeated segments, logos, etc).

We can extend our notion of similarity to near-duplicate shot detection (NDSD) in video. Given a reference shot, this can be used to find all shots in a database that are near-duplicates of the reference, where we define this to mean that a high proportion of images in the reference shot have near-duplicates in the returned shot. A reliable NDSD system could be used, for example, to detect copyright violations of digital video. YouTube [22], a video-sharing website, currently contains a large amount of copyrighted television and films. Once one example of a copyrighted video is identified, a NDSD system can be used to automate or partially automate the detection and removal of all the copies of that copyrighted clip.

2. NEAR DUPLICATE IMAGES AND SHOTS

The definition of a near duplicate image varies depending on what photometric and geometric variations are deemed acceptable. This depends on the application. In the case of exact duplicate detection, no changes are allowed. At the other extreme, a more general definition is to require that images be of the same scene, but with possibly different viewpoints and illumination. In this paper, we focus on images that appear, to a human observer, to be identical or very similar. Therefore, the proposed methods can handle images with digitization artifacts, high levels of jitter, differing levels of compression, mild photometric distortions and small amounts of occlusion. In section 7 we explore how well the methods cope when the definition of near-duplicate is expanded to include partial scene-matching.

There are two main duplicate detection tasks that we would like to perform. The first is to enumerate all the duplicates of a given query image in a corpus. The second is simply to evaluate a predicate: is there any duplicate of the query image in a set of images? Both of these tasks arise in the image search example given in the introduction.

The predicate evaluation is typically used as a subroutine in the query process. When a corpus is very large and constantly being updated (as with a web search engine) it is prohibitively expensive to pre-calculate and store duplicate sets for every image.[1] Instead, at query time, the returned list is pruned to eliminate duplicates. A query may match millions of images, but only a few top ranked results are shown to the user. The pruning to find these k top-ranked results works as follows: each image with rank $2 \leq i \leq k$ is compared (using the near-duplicate predicate operation) to all the higher-ranking images. If it is a near-duplicate of any higher-ranked image, it is discarded, pulling the next best-ranked image into the top-k set.

Detection of near duplicate images in *large databases* imposes two challenging constraints on the methods used. Firstly, for each image, only a small amount of data (a fingerprint) can be stored; secondly, queries must be very cheap to evaluate. Ideally, enumerating all the duplicates of an image should have complexity close to linear in the number of duplicates returned[2] and predicate evaluation should take close to constant time. Since predicate evaluations are performed at query time, and the number of evaluations is proportional to the number of duplicates returned (which may be tens of thousands for a common image in a large corpus), this constant must also be very small, ideally no more than a few microseconds.

[1]Note that since the duplicate relation is not transitive, duplicate detection cannot generally be used directly to cluster a corpus into non-overlapping sets of near-duplicates.

[2]Web search engines typically scale by distributing queries in parallel among a set of computers, where the size of this set grows linearly with the size of the corpus. Therefore the search complexity is generally linear in the size of the corpus, although the constant of proportionality is extremely small.

2.1 Efficient image representations

Most approaches to near-duplicate detection share a similar pattern. Firstly, an image representation and a distance measure are defined, which affects both the amount of data stored per image and the time complexity of a database search. The amount of stored data ranges from a (small) constant amount of data per image to large sets of image features, whose size often far exceeds the size of the images themselves. When searching the database for a near duplicate image, algorithms of different time complexity are used, the most naive approach being computing the difference to every image in the database.

We propose and compare two methods for near duplicate image detection, both storing only a small constant amount of data per image. Both have near-constant time complexity for predicate evaluation and a complexity for duplicate enumeration that is close to linear in the number of duplicates returned. The image representation and the distance measure are chosen in both cases so that search for near duplicate images can be efficiently approximated by a randomized procedure.

The first method (section 3) represents the image using a hierarchical tiled colour histogram and measures similarity using the Euclidean distance between descriptors. Efficient retrieval is achieved using Locality Sensitive Hashing (LSH). The second method (section 4) represents the image by a sparse set of visual words. Here the similarity is computed using a set overlap measure and efficient retrieval is achieved using a min-Hash algorithm.

2.2 Near-duplicate shot detection

Here, we extend the concept of near-duplicate image detection to video shots. Given a query shot, the task is to find near-duplicate shots in the corpus that contain a large proportion of images that are near-duplicates of images in the query. We achieve this using a variant of the Hough transform.

We first initialize a voting table whose size is the number of shots in the dataset. We take each frame from the query shot in turn and search for its near-duplicates, sorting them chronologically (the order they would have appeared in the original video). Each returned image from the dataset acts like a voting permit and can be used to vote for a particular shot only once for all the images in the query shot. Once all the images in the query shot have been processed, we use an empirically derived threshold on the percentage of votes found in the voting table to return all near-duplicate shots. We could enforce a more stringent shot-level test by examining the temporal ordering between the query and target shots, but empirically we find that this is not needed for accurate detection.

The aggregation of many votes makes near-duplicate shot detection quite robust to individual near-duplicate image false positives. As we show in section 6, this allows us to use weaker, but cheaper, approximation schemes for near-duplicate image enumeration when NDID is used as a subroutine for NDSD.

2.3 Related work

Ke et al. [10] demonstrate near-duplicate detection and sub-image retrieval by using sparse features, taken from each image, coupled with a disk-based LSH for fast approximate search on the individual feature descriptors. They demonstrate the efficacy of their method on a synthetic database of "corrupted" images but show the system only scaling to handle 18K images with query times many times slower than both our methods. Zhang & Chang [23] use a

parts-based representation of each scene by building Attributed Relational Graphs (ARG) between interest points. They then compare the similarity of two images by using Stochastic Attributed Relational Graph Matching, to give impressive matching results. Unfortunately, they only demonstrate their method on a few hundred images and don't discuss any way to scale their system to larger datasets of images.

In the case of NDSD, others have considered a stronger temporal constraint than we employ here. For example, [1, 6, 24] use edit distance. However, these methods are only applied to a corpus consisting of tens of hours of video at most. The largest system for near-duplicate shot detection to our knowledge is that of Joly *et al.* [8, 9] for 30,000 hours of video. Their method involves representing each keyframe by a set of 20 dimensional spatio-temporal descriptors computed about Harris interest points. They then use a Hilbert curve for efficient approximate search for these features, which has sub-linear search complexity, and a shot voting procedure to flag duplicates. However, there are a number of limitations: (i) a large amount of data must be stored per keyframe (possibly hundreds of Harris points and their descriptors); (ii) it is not suitable for real time operations. Our approach differs in being applicable to both images and video, having time complexity for enumerating duplicates that is close to linear in the number of duplicates, and only requiring a small, constant amount of data to be stored per image, independent of the number of detected features.

3. COLOUR HISTOGRAMS AND LSH

3.1 An image in 384 bytes

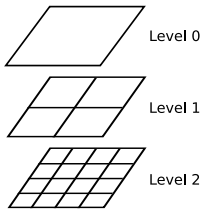
In order to be able to deal efficiently with millions of images, while still being able to keep a sizeable portion of the data in main memory, we need to generate an extremely compressed feature vector for each image. Here we propose using a colour histogram combined with a spatial pyramid over the image to jointly encode global and local information. We use a colour model which is partially colour normalized and simple to compute, known as the opponent colour model [4]:

$$\begin{aligned} I &= (R+G+B)/3 \\ O_1 &= (R+G-2B)/4+0.5 \\ O_2 &= (R-2G+B)/4+0.5 \end{aligned}$$

The spatial pyramid is arranged so that we use 128 bytes of data in describing each level. These are appended to create the final feature vector. On descending to the next level in the pyramid, the number of segments we take histograms over increases four-fold. Therefore, to maintain the size constraints for each level, the number of bytes used to describe each channel per segment is quartered. This places a desirable bias on the importance of the levels. We require the histograms of two candidate regions at the top level to agree quite accurately for them to be matched, but we have more "give" at the lower levels, due to the lower histogram resolution. As individual frames often suffer from jitter and noise, this is a necessary requirement for ensuring we match near duplicate frames. The representation is illustrated in figure 3.

We use double the amount of data for storing the intensity channel, I, as the other opponent colour channels, O_1 and O_2. This is a common practice as generally more information is contained in the intensity information. We also use three pyramid levels and sum the colour components for each pixel, into a running histogram for each segment. We then L_1 normalize each histogram and represent each bin by a single byte.

As well as giving us a very reduced representation for each image, computing the histogram is very efficient. We only accumulate



Level	I	O_1	O_2
0	64	32	32
1	16	8	8
2	4	2	2

Figure 3: The left side of this figure shows the spatial subdivision of the image at each level of the histogram pyramid. The table to the right shows the amount of data stored for each segment and colour channel in bytes.

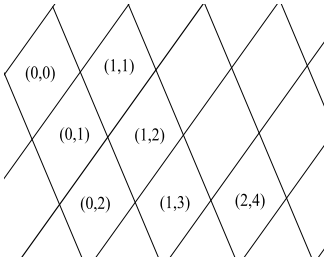


Figure 4: Segmenting the space using non-orthogonal random projections. Each cell is represented by a tuple.

pixels explicitly into the histograms at the lowest level and higher levels are found by summing these histograms from lower levels. Additionally, the opponent transform can be computed using vectorized integer operations, which enable the histograms for video material to be computed at approximately 150 fps for frame sizes of 320×240 pixels on a 2GHz machine.

To find near duplicate frames, it is assumed that Euclidean distance between their feature vectors is a meaningful measure of similarity. The problem then simplifies to: given a query frame, find all the frames within some specified distance from the query. Brute force search is completely impractical given the high dimensionality and size of the data, so we use the LSH scheme described in the following section to efficiently find all the points within the distance.

3.2 Locality Sensitive Hashing overview

Locality Sensitive Hashing (LSH) [3] is a method developed for efficiently answering approximate near neighbour queries, of the form "find all points within a given radius, R, from a query point with high probability". The method works by generating a number of hash tables of points, where the hashing function works on tuples of random projections of the form:

$$h_{\mathbf{a},b}(\mathbf{v}) = \left\lfloor \frac{\mathbf{a}\cdot\mathbf{v}+b}{w} \right\rfloor$$

where $\mathbf{a}$ is a random vector whose components are independently drawn from a Gaussian distribution, b is a random number drawn from $[0, w)$, w specifies a bin width (which is set to be a constant for all projections), and $\mathbf{v}$ is the query point. A Gaussian distribution is used so that the difference between the projections of two points is approximately L_2 distance preserving, as the Gaussian is a 2-stable distribution [7]. Each projection splits the space along planes perpendicular to $\mathbf{a}$, separated by distance w and a tuple of projections specifies a segmentation of the space akin to a grid, but where the axes are non-orthogonal (see figure 4). Clearly, if the number of projections is chosen carefully, then two points which hash into the same bin will be nearby in the space. To avoid boundary effects, many hash tables are generated, each using a different

tuple of projections (although, we can improve run-time speed by re-using some of these projections).

The time complexity for querying every hash table is constant and this returns a set of candidate points which lie near to the query point in space. In practice, a proportion of these points will be at a greater distance than R from the query point. The experiments in section 5 show that for some applications these "false matches" can be tolerated, in which case the histograms themselves do not need to be consulted at query time, and so they need not be stored. This leads to a storage cost of 5.3 bytes per hash table per image.

If pruning is required, we need to explicitly compute the distance to each of the returned points. The total number of points and therefore the number of distances to compute grows as $O(n^\rho)$, where $\rho = \frac{\ln 1/p_1}{\ln 1/p_2}$, p_1 is a lower bound on the probability that two points within R will hash to the same bucket and p_2 is an upper bound on the probability that two points not within R will hash to the same bucket. Clearly if $p_1 \gg p_2$, then ρ will be very small. Thus the complexity of enumerating duplicates is close to linear in the number of duplicates. When this pruning is needed, we must store an additional 384 bytes per image at query time.

4. SPARSE FEATURES AND MIN-HASH

4.1 Image description

Local features and descriptors have been developed for image to image matching [11, 14]. These are designed to be invariant to illumination and geometric transformations varying from scale to a full affine transformation, as might arise from a viewpoint change. They have been used successfully in model based recognition systems [11]. Furthermore, by quantizing the descriptors into visual words, 'bag-of-words' representations have also been used successfully for matching images and scenes [15, 19]. We build on these approaches in our design of a sparse feature based near duplicate image detector.

The difference of Gaussians (DoG) [11] operator is used as a feature (region) detector. Each region is then represented by a SIFT [11] descriptor using the image intensity only. SIFT features have proven to be insensitive to small local geometric and photometric image distortions [13].

A 'visual vocabulary' [19] – a set of visual words $\mathcal{V}$ – is constructed by vector quantizing the SIFT descriptors of features from the training data using K-means. A random subset of the database can be used as the training data. The K-means cluster centres define visual words. The SIFT features in every image are then assigned to the nearest cluster centre to give the visual word representation.

Assume a vocabulary $\mathcal{V}$ of size $|\mathcal{V}|$ where each visual word is encoded with unique identifier from $\{1, \ldots, |\mathcal{V}|\}$. Each image is represented as *a set* $\mathcal{A}_i$ of words $\mathcal{A}_i \subset \mathcal{V}$. Note, that a set of words is weaker representation than a bag of words, as it doesn't record the frequency of occurrence of visual words in the image.

The distance measure between two images is computed as the similarity of sets $\mathcal{A}_1$ and $\mathcal{A}_2$, which is defined as the ratio of the number of elements in the intersection of the representations over their union:

$$\text{sim}(\mathcal{A}_1, \mathcal{A}_2) = \frac{|\mathcal{A}_1 \cap \mathcal{A}_2|}{|\mathcal{A}_1 \cup \mathcal{A}_2|}. \quad (1)$$

To efficiently retrieve NDID under this distance measure a min-Hash algorithm is used. This allows us to approximately find all images whose similarity is above a threshold for a given query in constant time. We describe the search algorithm in the following sub-section.

4.2 Min Hash review

In this section, we describe how we adapt a method originally developed for text near-duplicate detection [2] to near-duplicate detection of images. We describe it using textual words, and then explain the adaptation to visual words in the following sub-section.

Two documents are near duplicate if the similarity $\text{sim}(\mathcal{A}_1, \mathcal{A}_2)$ is higher than a given threshold ρ. The goal is to retrieve all documents in the database that are similar to a query document. This section reviews an efficient randomized procedure that retrieves near duplicate documents in time proportional to the number of near duplicate documents (i.e. time complexity is independent of the size of the database). The outline of the algorithm is as follows: First a list of min-hashes are extracted from each document. A min-hash is a single number having the property that two sets $\mathcal{A}_1$ and $\mathcal{A}_2$ have the same value of min-hash with probability equal to their similarity $\text{sim}(\mathcal{A}_1, \mathcal{A}_2)$. For efficient retrieval the min-hashes are grouped into n-tuples called sketches. Identical sketches are then efficiently found using a hash table. Documents with at least m identical sketches (sketch hits) are considered as possible near duplicate candidates and their similarity is then estimated using all available min-hashes.

min-Hash. First, a random permutation of word labels π is generated. For each document $\mathcal{A}_i$ a min-hash $\min \pi(\mathcal{A}_i)$ is recorded. Consider the following example: vocabulary $\mathcal{V} = \{\text{A,B,C,D,E,F}\}$ and three sets {A,B,C}, {B,C,D}, and {A E F}. Four independent random permutations and corresponding min-hashes follow in the table.

	A	B	C	D	E	F	A B C	B C D	A E F	
permutations	3	6	2	5	4	1	2	2	1	min-hashes
	1	2	6	3	5	4	1	2	1	
	3	2	1	6	4	5	1	1	3	
	4	3	5	6	1	2	3	3	1	

The method is based on the fact, which we show later on, that the probability that $\min \pi(\mathcal{A}_1) = \min \pi(\mathcal{A}_2)$ is

$$P(\min \pi(\mathcal{A}_1) = \min \pi(\mathcal{A}_2)) = \frac{|\mathcal{A}_1 \cap \mathcal{A}_2|}{|\mathcal{A}_1 \cup \mathcal{A}_2|} = \text{sim}(\mathcal{A}_1, \mathcal{A}_2).$$

To estimate $\text{sim}(\mathcal{A}_1, \mathcal{A}_2)$, N independent random permutations π_j are used. Let l be the number of how many times $\min \pi_j(\mathcal{A}_1) = \min \pi_j(\mathcal{A}_2))$. We estimate $\text{sim}(\mathcal{A}_1, \mathcal{A}_2) = l/N$. In our example, the sets {A,B,C} and {B,C,D} have three identical min-hashes and the estimated similarity will be 0.75, while the exact similarity is 0.5. The sets {A,B,C} and {A,E,F} share one min-hash and their similarity estimate is 0.25 (0.2 is exact).

How does it work? Consider drawing $X = \arg\min \pi(\mathcal{A}_1 \cup \mathcal{A}_2)$. Since π is a random permutation, each element of $\mathcal{A}_1 \cup \mathcal{A}_2$ has the same probability of being the least element. Therefore, we can think of X as being drawn at random from $\mathcal{A}_1 \cup \mathcal{A}_2$. If X is an element of both $\mathcal{A}_1$ and $\mathcal{A}_2$, i.e. $X \in \mathcal{A}_1 \cap \mathcal{A}_2$, then $\min \pi(\mathcal{A}_1) = \min \pi(\mathcal{A}_2) = \pi(X)$. If not, say $X \in \mathcal{A}_1 \setminus \mathcal{A}_2$, then $\pi(X) < \min \pi(\mathcal{A}_2)$. Therefore, for random permutation π it follows

$$P(\min \pi(\mathcal{A}_1) = \min \pi(\mathcal{A}_2)) = \frac{|\mathcal{A}_1 \cap \mathcal{A}_2|}{|\mathcal{A}_1 \cup \mathcal{A}_2|}. \quad (2)$$

Sketches. For efficiency of the retrieval, the min-hashes are grouped into n-tuples. Let Π be an n-tuple $(\pi_1, \ldots, \pi_n)$ of different independent random permutations of $\mathcal{V}$. Let $S_\Pi(\mathcal{A}_1)$ be a *sketch* $(\min \pi_1(\mathcal{A}_1), \ldots, \min \pi_n(\mathcal{A}_1))$. The probability that two sets $\mathcal{A}_1$ and $\mathcal{A}_2$ have identical sketches $S_\Pi(\mathcal{A}_1) = S_\Pi(\mathcal{A}_2)$ is $\text{sim}(\mathcal{A}_1, \mathcal{A}_2)^n$ since the permutations Π (and hence the min-hashes in the sketch) are independent. Grouping min-hashes significantly reduces the probability of false positive retrieval. The retrieving procedure then

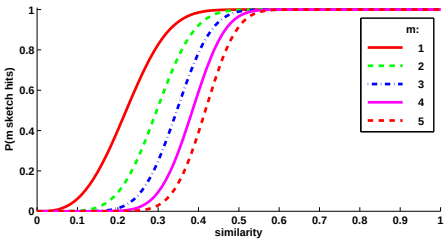


Figure 5: The probability of at least m sketch hits for $k = 64$, $n = 3$.

Figure 6: A random sample of TRECVID key frames.

estimates $\text{sim}(\mathcal{A}_1, \mathcal{A}_2)$ only for image pairs that have at least m identical sketches out of k sketches $(\Pi_1, \ldots, \Pi_k)$. The probability $P(\mathcal{A}_1 \overset{m}{\sim} \mathcal{A}_2)$ that the sets $\mathcal{A}_1$ and $\mathcal{A}_2$ have at least m identical sketches out of k is given

$$P(\mathcal{A}_1 \overset{m}{\sim} \mathcal{A}_2) = \sum_{i=m}^{k} \binom{k}{i} p^{in} (1 - p^n)^{k-i}, \qquad (3)$$

where $p = \text{sim}(\mathcal{A}_1, \mathcal{A}_2)$. If we considered sketches of size $n = 2$ in the example, the sets {A,B,C}, {B,C,D}, and {A,E,F} would be represented by sketches $((2,1),(1,3))$, $((2,2),(1,3))$, and $((1,1),(3,1))$ respectively. The only sketch hit is between the sets {A,B,C} and {B,C,D} (in the second sketch). No other pair would be considered as a possible near duplicate.

A plot, showing the dependency of $P(\mathcal{A}_1 \overset{m}{\sim} \mathcal{A}_2)$ on p for $k = 64$, $n = 3$ and varying m is shown in figure 5. Note, that the parameters k and n have to be fixed at the database design time, whereas m can be specified at query time.

4.3 Text versus images

In the case of text near duplicate search, the size $|\mathcal{V}|$ is typically very large (especially for shingling [2], that works with sequences of words rather than single words). Two text documents are typically considered identical if their similarity is above 90% [5].

In our case, the visual vocabulary is significantly smaller, typically of the order of tens of thousands of visual words. Additionally, many visual words are unstable and can disappear between apparently identical images. Therefore, in near duplicate image retrieval, we are interested in image pairs with a similarity greater than 35% (or even less). For NDID it is sufficient to only store the min-hashes. Using $k = 64$ sketches of length $n = 3$ requires 384 bytes per image, which, including the necessary hash tables, leads to around 724 bytes per image in our experiments.

5. EXPERIMENTS

We do not have access to ground-truth data for our experiments, since we are not aware of any large public corpus in which near-duplicate images have been annotated. We do show representative results to illustrate the typical performance of the methods; however most of our experiments focus on evaluating the effect of the approximations we have used to speed up retrieval.

For each method, we consider three sets of near-duplicates of a given image. The first is the "true similarity" set: this is the set of images that matches our similarity definition. For the colour histogram method (henceforth referred to as CH-LSH) this is the set of images whose histograms are within a given distance of the reference histogram. For the image feature method (henceforth referred to as SF-mH) it is the set of images whose similarity as defined in (1) is above a threshold. This true similarity set is constructed by exhaustive search of the corpus and is used purely for evaluation of our approximate methods since it would be impractically expensive for real applications.

The second set of near-duplicates is the "raw approximate similarity" set. In the case of CH-LSH, this is the set of images that are retrieved by an LSH query using the reference image's histogram. In the case of SF-mH it is the set of images that have at least one matching sketch. This set includes many false positives for both methods, however it is the cheapest to compute. If only the raw approximate similarity set is required it is not necessary to store the actual colour histograms (for CH-LSH) or min-hashes (for SF-mH).

The third set is the "verified approximate similarity" set. This is a filtered version of the raw approximate similarity set. In the case of CH-LSH, the distance between the reference histogram and each histogram in the raw set is computed, and any histograms that differ by more than the selected threshold are discarded. Thus the verified approximate similarity set for histograms contains no false positives compared with the true similarity set, though there may be missed near-duplicates. In the case of SF-mH the similarity estimate is computed using all min-hashes and any images with estimated similarity smaller than a threshold are discarded. Since min-hashes only provide an estimate of the similarity, both false positives and false negatives w.r.t. the true similarity set can occur.

We demonstrate our methods for NDID on the TRECVID 2006 data set. TRECVID [20] is an annual benchmark, designed to promote progress in the field of video information retrieval. The dataset used for testing is about 165 hours (17.8M frames, 127 GB) of MPEG-1 news footage, recorded from different TV stations from around the world. From this data, a shot boundary detector is run by the organizers and several "representative" keyframes are taken from each shot and provided separately as JPEG's, giving 146,588 keyframes. Each frame is normally of quite low quality and at a resolution of 352×240 pixels. The frames suffer from compression artifacts, jitter and noise typically found in highly compressed video. A random sample of the data (figure 6) shows that it contains a huge variety of different objects, scenes and people. We compare our two methods for NDID on TRECVID using just the keyframes, without using temporal information. In this setting, near duplicates for every key-frame are searched for. We also show results for NDSD over the full 17.8M frames of TRECVID, using the CH-LSH method.

5.1 LSH (CH-LSH) retrieval

For this experiment, we defined near duplicate images as images having an L_2 distance between their histograms of less than 200. For the LSH, 55 random projections were computed and combined in tuples to create 36 hash tables (taking 539 bytes per image including the histogram). Using these parameters, the average false negative rate (the fraction of images in the true similarity set that are not in the raw approximate similarity set) was 0.98%. The ratio of sizes between the raw approximate similarity set and the verified approximate similarity set was 9.09, on average.

To show that a small change of the threshold does not result in retrieving almost the whole corpus we drew a large number of ran-

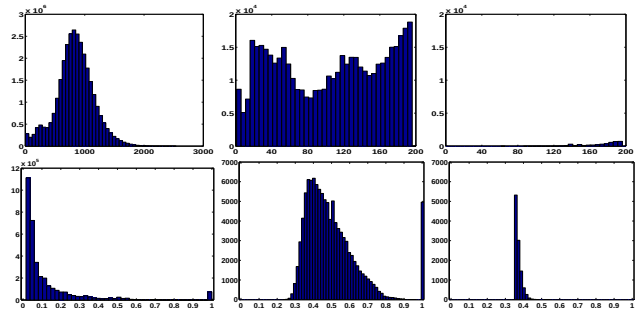


Figure 7: Histograms of image pair distances in the CH-LSH (top), and true similarity in the SF-mH method (bottom) from TRECVID data. Left to right: raw approximate similarity set, verified approximate set, and false negatives.

Figure 8: Examples of images form the TRECVID database. The query image is in the first column, other columns show images and their distance for the color histogram method.

dom pairs of images and measured their similarity. The similarity between a random pair of images is larger than 500 in 99.9% of cases. Figure 7(top) shows histograms of distances, and demonstrates that for NDID, verifying the raw results returned from the LSH is necessary to ensure accurate retrieval. Figure 8 shows some example images from the TRECVID dataset along with a selection of retrieved images showing the typical perceptual similarity between images with varying histogram distances.

On a 2GHz commodity laptop, building the hash tables and finding all near-duplicates for the 150K TRECVID keyframes took only 15s + 15s = 30s.

5.2 Min Hash (SF-mH) retrieval

For this experiment, we detected DoG features [11] and vector-quantized its SIFT description into vocabulary of 2^{16} words. We defined near duplicate images as images having similarity (1) above 35%. For retrieval we used $k = 64$ sketches of $n = 3$ min-hashes (taking 724 bytes per image – 384 for min-hashes and 5.3 bytes for each of 64 hash tables). Images were placed in the raw approximate similarity set given a single sketch hit. The verified approximate similarity set removed all images with estimated similarity lower than 35%. Using these parameters, 0.75% of images in the true similarity set were missing from the raw approximate similarity set. On average 5.9% of images in the verified approximate similarity set were not in the true similarity set, and 4.95% of the true similarity set was missing from the verified approximate set. The average ratio of sizes between the raw approximate similarity set and the verified approximate similarity set was 10.04.

The similarity between random image pairs is less than 5% in 99.9% of cases. Figure 7 (bottom row) shows histograms of similarities, and demonstrates that for the SF-mH method verifying the raw results returned by sketches is necessary for accurate retrieval.

Figure 9 shows some example images from the TRECVID dataset along with a selection of retrieved images showing the typical perceptual similarity between images with varying feature similarities.

Figure 9: Examples of images form the TRECVID database. The query image is in the first column, other columns show images and their similarity for the Min Hash method.

Figure 10: Images with 30-40 detected near duplicates: samples from images retrieved by both methods (common ND / colour histogram only / min Hash only).

5.3 Near duplicate definition comparison

We have selected several sets of 30-40 near duplicate images and have compared results of the two proposed methods on them. This experiment compares the ability of the two representations (colour histograms and image features) to encode the information necessary to detect near-duplicate images. Image samples from the sets are shown in figure 10. Three numbers are overlaid on the images: the number of images retrieved by both methods, the number of images retrieved using CH-LSH only, and the number of images retrieved by SF-mH only. Manual verification shows that all returned images are perceptually duplicates in this limited trial, in other words we did not see any false positives. We have no ground truth to determine the exact number of false negatives, but as the figure shows, each method had a small number of false negatives compared with the other. No obvious trends emerged from these false negatives, however we know that each method is sensitive to certain failure modes:

Occlusion. Since L_2 is not a robust distance, local occlusions can cause a significant increase of the colour histogram distance, making CH-LSH sensitive to occlusion. For the min-hashes, occlusions typically insert and remove some visual words into the image representation. Therefore, SF-mH tolerates occlusions that preserve a sufficiently high percentage of visual words.

Noise and blur. The SF-mH method is heavily dependent on the quality of the feature detection. Therefore, any image deformation that affects the firing of the feature detector can alter the performance of the whole method. Such deformations include strong artifacts / noise (which increases the number of features) and image blur (which decreases the number of features). The CH-LSH method is fairly insensitive to these types of image deformations.

6. NEAR-DUPLICATE SHOT DETECTION

In addition to the keyframe experiment, we also performed shot-based near-duplicate detection for the TRECVID data on the full 17.8M frames, using CH-LSH. For this experiment, we used 36 random projections, combined into 8-tuples for 36 hash tables, and used the raw approximate similarity sets directly, which required only 190 bytes per image of storage. On a 2GHz machine, the

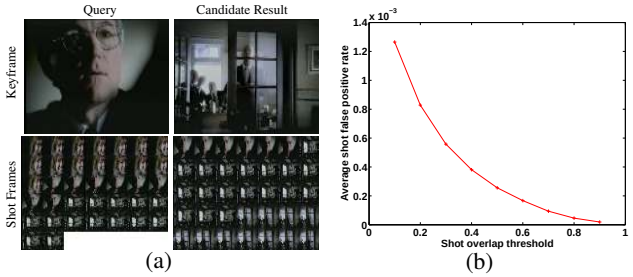


Figure 11: (a) shows a near-duplicate shot matched from the frames, which wouldn't have been found by examining the keyframes alone. (b) compares how the false positive duplicate rate changes with varying overlap threshold on a small subset of 10 hours of TRECVID data.

method took approximately 90mins to search for all 17.8M frames in the dataset.

Figure 11(a) demonstrates the need to perform near-duplicate shot detection as opposed to using the keyframes alone. This figure shows that, in general, matching shots on individual frames and matching shots on keyframes will give different results. In figure 11(b), we examine the ability of the shot-voting to act as a proxy for histogram verification on a small sample of 10 hours of TRECVID data. This shows that when the overlap threshold is high, histogram verification is not required to achieve low false positive rates for near-duplicate shot detection. For performing NDSD over the whole dataset, we use a shot overlap threshold of 0.7, which gives us an average false positive rate of 9.4×10^{-4}.

7. SCENE REPETITION IN FILMS

In this section we consider a looser definition of NDID – that of identifying images of the same scene. In this case the images *may* differ perceptually. We take as our application detecting those frames in a video that were shot at the same location. Previous methods for this application have used the feature film 'Run Lola Run' [Tykwer, 1999] [18] and the music video 'Come Into My World' [Gondry, 2002] [16], since both videos contain a time loop. We use both these videos for the experimental evaluation.

Kylie Minogue: Come Into My World. The video contains four repeats of Kylie walking around a city area (with superposition), and a short 5th appearance at the end. A full description is given in [21]. For the experiments the video is represented as key frames by extracting every 15^{th} frame, giving 423 frames. We compare the performance of the CH-LSH and SF-mH methods to the ground truth by computing frame similarity matrices.

For the SF-mH method a new vocabulary of 10,000 visual words is generated from the key frames. Each frame is then represented by 384 min-hashes, and sketches of 2 min-hashes are generated from these. It is necessary to use a smaller n (the number of min-hashes in a sketch) in this case to avoid false negatives. The CH-LSH method is unchanged from the TRECVID implementation.

Figure 12 top row shows similarity matrices for CH-LSH (distance threshold 450) and SF-mH (similarity threshold 15%) respectively. In all similarity matrices, self-matching frames (diagonal) are not displayed. Matches along the diagonal are matches between consecutive frames. Both methods are successful in capturing the contiguous scene repetitions as demonstrated by the parallel diagonal lines. Note that the similarity matrix for SF-mH is slightly denser, especially in the fourth repetition where the frames are more occluded. Also, the lines of repetition are thicker, since several consecutive frames are matched despite viewpoint change. The viewpoint change is captured in the samples from repeated frames

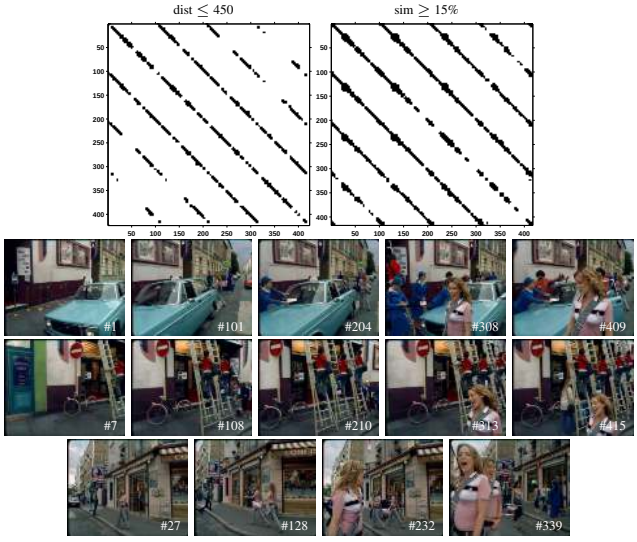


Figure 12: Top row: the similarity matrices for key frames from 'Come Into My World' for CH-LSH (left) and SF-mH (right). Bottom: samples of detected similar frames by SF-mH. Four repetitions (plus a few frames of a fifth) are evident (diagonal and off diagonal lines). The frame number is overlaid on the frames.

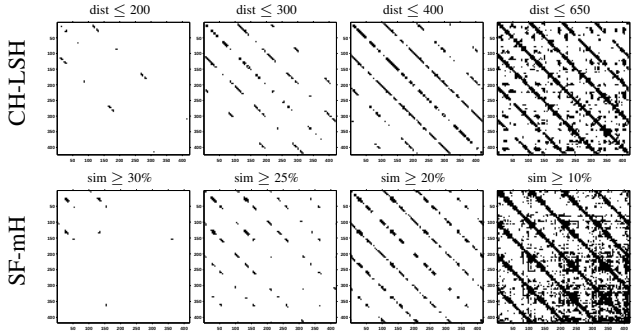


Figure 13: Dependance of the scene repetition retrieval in 'Come Into My World' on varying thresholds.

in figure 12 bottom. This demonstrates the tolerance to viewpoint change built in by design to the SF-mH method.

The dependance of the performance on the distance / similarity threshold for the two methods is shown in figure 13. For CH-LSH, the distance thresholds used were 200, 300, 400, and 650; for SF-mH, the similarity thresholds were 30%, 25%, 20%, and 10%. Note that there is a "sweet spot" threshold that reveals the story repetition for both the methods. This fact suggests that our distance/similarity measure corresponds to some extent to a human's perception of image similarity.

Run Lola Run. The story in the film repeats three times, with many of the repeated shots being of identical locations although the camera viewpoint can differ. The video is represented as key frames by extracting every 25^{th} frame, giving 4285 frames.

For the SF-mH method in order to give extra tolerance to viewpoint change the feature detector is changed here from DoG to Hessian Affine [12, 17]. Again SIFT descriptors are taken and vector quantized into 10,000 visual words. The matching time over all pairs of near duplicate images (not including the feature detection) is less than 1 second for our Matlab implementation on a 2GHz machine. The CH-LSH implementation is unchanged from the TRECVID case.

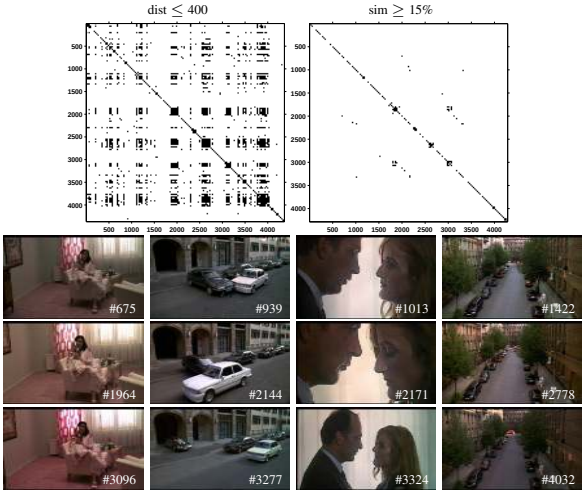


Figure 14: Top row: the similarity matrices for CH-LSH and SF-mH for key frames in 'Run Lola Run'. The raw data has been processed by morphological dilation for display purposes. Three repetitions are evident (diagonal and off diagonal curves) in the right plot. Bottom: examples of near duplicate frames found by SF-mH.

Figure 14 shows the key-frame similarity matrix for both methods. In the case of SF-mH this matches the results of [18] (Fig. 20), reflecting the ground truth of the movie. The comparison between the CH-LSH and SF-mH methods is instructive here – in the CH-LSH case there are so many false positives that it is not possible to see the true structure. Setting the threshold to 400 gave reasonable results for scene-matching in the Kylie video because in that video, different views of the scene had very different histogram distributions, but this is not indicative of most television or film data. For Lola, the histograms are more similar throughout the film, so setting the threshold so high gives many false positives. A threshold of 200 does not reveal any of the structure in the film. We have also tried using our NDSD CH-LSH method for Lola, but again, because our histograms are not designed to be viewpoint invariant, we are unable to correctly match scenes in the film.

8. DISCUSSION

Our preliminary experiments on NDID in a large video corpus suggest that the performance of CH-LSH and SF-mH is interchangeable. If this result is repeatable on a corpus of photographs we will conclude that CH-LSH is preferable as a duplicate-detection mechanism for image search engines. The processing time per frame for CH-LSH is much smaller than for SF-mH, the method is dependent on many fewer parameters and it does not require the construction of a feature vocabulary.

Our algorithm for NDSD provides a very time-efficient approximation to CH-LSH and is a promising step towards automatically locating duplicate video clips in large corpora, a feature that is currently missing from large-scale commercial video search engines.

As the definition of near-duplicate images is changed to include more perceptually dissimilar examples, the performance of SF-mH starts to dominate that of CH-LSH. We believe this may be valuable for certain image search tasks. For example, the results of a query for images of Paris might benefit from duplicate detection that collapsed all images of the Eiffel tower into a single set, regardless of weather, time of day, lighting conditions, and with some invariance to viewpoint. Our experiments in section 7 suggest that SF-mH may be a promising approach for this application.

We conclude by noting that although we have proposed two different image representations with associated distance measures and search methods, in principle the search method of one could be applied to the other. For example, the bag of visual words sparse feature representation can be cast as a normalized histogram, and near duplicates within an L_2 distance may then be found using the LSH search method.

Acknowledgement. We are grateful for support from the Royal Academy of Engineering, the EU Visiontrain Marie-Curie network, and the EPSRC.

9. REFERENCES

[1] M. Bertini, A. D. Bimbo, and W. Nunziati. Video clip matching using mpeg-7 descriptors and edit distance. In *CIVR*, pages 133–142, 2006.

[2] A. Broder. On the resemblance and containment of documents. In *SEQS: Sequences '91*, 1998.

[3] M. Datar, N. Immorlica, P. Indyk, and V. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *SCG*, pages 253–262, 2004.

[4] J. Geusebroek, R. van den Boomgaard, A. Smeulders, and H. Geerts. Color invariance. *PAMI*, 23(12):1338–1350, 2001.

[5] M. Henzinger. Finding near-duplicate web pages: a large-scale evaluation of algorithms. In *SIGIR '06*, pages 284–291, New York, NY, USA, 2006. ACM Press.

[6] T. C. Hoad and J. Zobel. Fast video matching with signature alignment. In *MIR*, pages 262–269, 2003.

[7] P. Indyk. Stable distributions, pseudorandom generators, embeddings and data stream computation. In *IEEE Symposium on Foundations of CS*, 2000.

[8] A. Joly, O. Buisson, and C. Frélicot. Content-based copy detection using distortion-based probabilistic similarity search. *IEEE Transactions on Multimedia*, to appear, 2007.

[9] A. Joly, C. Frelicot, and O. Buisson. Robust content-based video copy identification in a large reference database. In *Proc. CIVR*, 2003.

[10] Y. Ke, R. Sukthankar, and L. Huston. Efficient near-duplicate detection and sub-image retrieval. In *ACM Multimedia*, 2004.

[11] D. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, 2004.

[12] K. Mikolajczyk and C. Schmid. An affine invariant interest point detector. In *Proc. ECCV*. Springer-Verlag, 2002.

[13] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. In *Proc. CVPR*, 2003.

[14] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. Van Gool. A comparison of affine region detectors. *IJCV*, 65(1/2):43–72, 2005.

[15] D. Nister and H. Stewenius. Scalable recognition with a vocabulary tree. In *Proc. CVPR*, 2006.

[16] T. Quack, V. Ferrari, and L. Van Gool. Video mining with frequent itemset configurations. In *Proc. CIVR*, 2006.

[17] F. Schaffalitzky and A. Zisserman. Multi-view matching for unordered image sets, or "How do I organize my holiday snaps?". In *Proc. ECCV*, 2002.

[18] F. Schaffalitzky and A. Zisserman. Automated location matching in movies. *CVIU*, 92:236–264, 2003.

[19] J. Sivic and A. Zisserman. Video Google: A text retrieval approach to object matching in videos. In *Proc. ICCV*, 2003.

[20] TRECVID. http://trecvid.nist.gov/.

[21] Wikipedia. Come into my world. http://en.wikipedia.org/wiki/Come_Into_My_World.

[22] YouTube. http://www.youtube.com/.

[23] D. Zhang and S. Chang. Detecting image near-duplicate by stochastic attributed relational graph matching with learning. In *ACM Multimedia*, 2004.

[24] J. Zhou and X.-P. Zhang. Automatic identification of digital video based on shot-level sequence matching. In *ACM MM*, pages 515–518, 2005.