Efficiently Matching Sets of Features with Random Histograms

Wei Dong, Zhe Wang, Moses Charikar, Kai Li
Department of Computer Science, Princeton University
35 Olden Street, Princeton, NJ 08540, USA
{wdong,zhewang,moses,li}@cs.princeton.edu

ABSTRACT

As the commonly used representation of a feature-rich data object has evolved from a single feature vector to a set of feature vectors, a key challenge in building a content-based search engine for feature-rich data is to match feature-sets efficiently. Although substantial progress has been made during the past few years, existing approaches are still inefficient and inflexible for building a search engine for massive datasets. This paper presents a randomized algorithm to embed a set of features into a single high-dimensional vector to simplify the feature-set matching problem. The main idea is to project feature vectors into an auxiliary space using locality sensitive hashing and to represent a set of features as a histogram in the auxiliary space. A histogram is simply a high dimensional vector, and efficient similarity measures like L_1 and L_2 distances can be employed to approximate feature-set distance measures.

We evaluated the proposed approach under three different task settings, *i.e.* content-based image search, image object recognition and near-duplicate video clip detection. The experimental results show that the proposed approach is indeed effective and flexible. It can achieve accuracy comparable to the feature-set matching methods, while requiring significantly less space and time. For object recognition with Caltech 101 dataset, our method runs 25 times faster to achieve the same precision as Pyramid Matching Kernel, the state-of-the-art feature-set matching method.

Categories and Subject Descriptors

H.3 [**Information Storage and Retrieval**]; I.2.10 [**Artificial Intelligence**]: Vision and Scene Understanding

General Terms

Algorithms, Experimentation

Keywords

set of features, random histogram, locality sensitive hashing

MM'08, October 26–31, 2008, Vancouver, British Columbia, Canada.

1. INTRODUCTION

Using sets of local features to represent multimedia data objects for content-based similarity search or object recognition has been gaining popularity during the past few years because of the superior discriminating power of local features over global features. However, matching sets of unordered high-dimensional feature vectors imposes much higher complexity in both space and time. The challenge is to develop a method to substantially reduce the complexity of the set-matching operation while maintaining the search or recognition quality of using sets of features.

For example, the state-of-the-art approach to object recognition is to extract local features such as SIFT [21] from "interesting" points in an image and form a set of feature vectors for each image. The number of such feature vectors for each image varies, depending on its content, but it is common for an image to have a set of hundreds of SIFT features where each feature vector has 128 dimensions. Recent advanced search systems for audio [23], video [29] and sensor data also use sets of unordered high-dimensional feature vectors as their metadata representations, thus it is important to develop a flexible, effective and space and time efficient approach to represent and perform similarity match for sets of features.

There are two challenging aspects in solving the problem of matching sets of features. The first is that its memory space requirement is high. Local feature vectors are high dimensional. A SIFT feature vector for example has 128 dimensions. A typical MFCC feature vector, a commonly-used feature extraction for audio data, typically has about 200 dimensions. The number of SIFT feature vectors or MFCC feature vectors for each image or audio sentence is typically on the order of hundreds and sometimes thousands. The space requirement for such set of feature vectors can easily exceed 100KB which is often larger than the size of the original data object.

The second is that the complexity to compute a commonly-used set similarity measure is high. Since the set representation disregards the spatial orders or semantic relationships among the feature vectors, the similarity between two sets of feature vectors is typically defined as an overall measure of the similarity between the matched points under certain condition from the two sets. A commonly-used similarity measure is the sum of point distances of the one-to-one best matched points from the two sets [27]. The complexity of computing this measure is $O(n^3)$, where n is the average number of feature vectors in a set. If a search engine had to compare the set of features of a query image with that

of each of billions of images, the computation cost would be formidably expensive, especially when n is on the order of hundreds and sometimes thousands.

Previous content-based data retrieval methods [22, 10] have take the filtering-then-ranking approach. The filtering step uses a fix number of indices to index a subset of feature vectors of the query data object to obtain a candidate set of data objects. The ranking phase will rank the candidate set by computing the similarity measure between the query object's set of features with that of each data object in the candidate set. This approach is reasonably efficient though the ranking phase can become a performance bottleneck if the candidate set is large. Also, the space requirement for the indices can be significant.

The state-of-the-art light-weight set matching algorithm is Pyramid Matching Kernel (PMK) [11, 13]. Its main idea is to uniformly quantize the feature space in multiple resolutions, and to represent a set of features as a list of the space quanta that include a set member. Such lists are ordered, and can be matched in linear time, which is a substantial improvement over the polynomial set-matching algorithms. However, PMK has two drawbacks. First, it works with feature space of limited dimensionality. For high-dimensional feature vectors, it requires applying a dimension reduction method first which may cause significant loss of information if the intrinsic feature dimensionality is high. Second, the sparse and hierarchical representation imposes a significant space overhead.

This paper presents a randomized algorithm to embed a set of features into a single high-dimensional vector to simplify the feature-set matching problem. The main idea is to project all feature vectors into an auxiliary space using locality sensitive hashing [14, 9] and to represent a set of features as histograms, which are simply high dimensional vectors. Histogram similarity measures will be employed to approximate feature-set distance measures. By doing so, the challenging feature-set matching problem is reduced to a simpler similarity match problem in a single high-dimensional space.

To evaluate the effectiveness of the proposed approach, we have implemented it in a toolkit[1] and conducted experiments under three different task settings, *i.e.* content-based image similarity search, image object recognition, and near-duplicate video clip detection. The experimental results show that the proposed approach is indeed effective and flexible. It can achieve search and recognition accuracy comparable to the feature-set matching methods, while requiring significantly less space and time. For object recognition with Caltech 101 dataset, our method achieves the same precision as the feature-set matching method, but runs 25 times faster than Pyramid Matching Kernel, the state-of-the-art kernel for feature-set matching.

In addition, the proposed approach is flexible and can easily be used to construct search systems for multiple feature-rich data types with different similarity measures. By using different LSH families, one can approximate different point distance functions. Users can define different histogram distance metrics to satisfy desired set matching conditions. Since histograms are straightforward vectors, it is easy to pass them to existing software modules. Since the dimensionality of the histogram vectors does not explicitly depend on the feature vector's dimensionality nor the set size, users can configure it according to their needs.

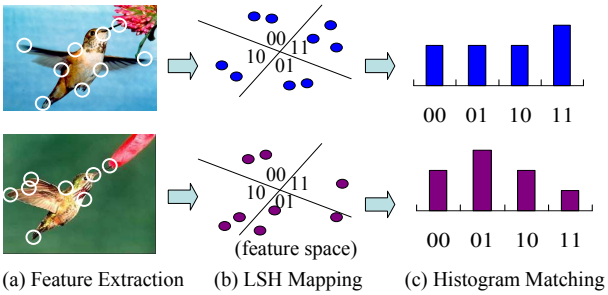

Figure 1: Illustration of the proposed approach. (a) Features are extracted from interesting points. (b) Objects are represented as sets of points in the feature space. These points are mapped to discrete values via locality preserving random mappings (LSH). A mapping is equivalent to a partitioning of the feature space. (c) Sets of features are embedded to histograms over the values mapped to. The histograms are matched with standard vector similarity measures.

2. PRELIMINARIES

In this section we recap the various concepts related to similarity measures, and the main technique, LSH, which we use to create random histograms.

2.1 Similarity, Distance and Kernel

Set similarities are defined in terms of point similarities. Let X and Y be two finite sets in an arbitrary feature space, and let n be the average set size. In this paper, we use s to denote an arbitrary point similarity ($s(x, y)$ is the similarity between points x and y, larger values indicate higher similarity). We use S for the corresponding set similarity ($S(X, Y)$ is the similarity between sets X and Y of features). We study the following two set similarities which are commonly used in practice.

The first set similarity S_1 is defined in a similar way as the Earth Mover's Distance [27]:

$$S_1(X,Y) = \max \sum_{x\in X}\sum_{y\in Y} a_{x,y}s(x,y)$$

$$s.t. \quad \sum_{x\in X} a_{x,y} \le 1, \quad \forall y \in Y$$

$$\sum_{y\in Y} a_{x,y} \le 1, \quad \forall x \in X$$

$$a_{x,y} \in \{0,1\}, \quad \forall x \in X, y \in Y.$$

The above assignment problem formation (a special case of integer linear programming) searches for an optimal one-to-one matching between the set members. The coefficient $a_{x,y} = 1$ means that the point x is matched to the point y, and a point can be used at most once. The best known method to find the optimal matching is the Hungarian algorithm, which takes $O(n^3)$ time. A similar measure can be defined for weighted sets which can be expressed as the optimum of a linear program (in fact min cost flow), which can also be solved in polynomial time.

The second set similarity S_2 is for one-to-many matching,

which is defined as

$$S_2(X,Y) = \sum_{x\in X}\sum_{y\in Y} s(x,y).$$

The above similarity obviously takes $O(n^2)$ time.

Distance and kernel are two related concepts in measuring similarity (for either points or sets). Distances are more often used in data retrieval tasks, which are commonly formulated as the k-nearest neighbor problem. Though a properly defined distance metric should be positive, symmetric, and satisfy the triangle inequality, certain measures violating one or more of these conditions, like the Kullback–Leibler divergence, are also considered distance measures in practice. Kernels are most commonly used with support vector machine (SVM) related methods. A well defined kernel should follow the Mercer condition (positive definiteness), which ensures the convergence of the SVM learning process. Taking a Mercer kernel between two objects is equivalent to taking the inner product between vectors obtained by mappings of these objects into certain high dimensional L_2 space. This equivalence allows SVM to apply to arbitrary feature space so long as there is a well-defined kernel. Note that higher similarity means a larger kernel value, and a smaller distance.

For set similarity, our proposed approach explicitly maps (embeds) sets of features to high-dimensional vectors, and is thus a complement to the kernel method, where the mapping is implicit. The Mercer condition is automatically guaranteed by the explicit mapping to a vector space. One can essentially try out any distances and kernels with the embedded vectors. In this paper, we limit our discussion to L_1 and L_2 distance for set similarity. Given two arbitrary points $x, y \in \mathbb{R}^d$, the L_1 and L_2 distances are defined as follows:

$$L_1(x,y) = \sum_{i=1}^{d} |x_i - y_i| \qquad L_2(x,y) = \{\sum_{i=1}^{d} |x_i - y_i|^2\}^{1/2}$$

We also define the corresponding K_1 and K_2 kernels as follows:

$$K_1(x,y) = \sum_{i=1}^{d} \min\{x_i, y_i\} \qquad K_2(x,y) = \sum_{i=1}^{d} x_i \cdot y_i$$

The following two relationships can be easily verified:

$$L_1(x,y) = K_1(x,x) + K_1(y,y) - 2 \times K_1(x,y)$$
$$[L_2(x,y)]^2 = K_2(x,x) + K_2(y,y) - 2 \times K_2(x,y)$$

The above two equalities essentially have the same form. For normalized vectors, we have $K_i(x,x) = K_i(y,y) = 1, i = 1,2$, and the relationships are exactly linear. For unnormalized vectors, there is still a conceptual equivalence. Thus later in this paper when set similarity is concerned, we will only discuss in the kernel setting, knowing that the results apply to the corresponding distance metric.

2.2 Locality Sensitive Hashing

Unlike set similarity, which we measure via the embedding vectors by distance or kernel functions, the point similarity is directly incorporated into the embedding method by Locality Sensitive Hashing(LSH).

An Locality Sensitive Hashing[1] is a random mapping defined on a set of objects, such that the probability of two object being mapped to the same value reflects the similarity between these objects. Formally, given a feature space $\mathbb{M}$, we say $\mathcal{H}$ is a LSH family if for arbitrary $x, y \in \mathbb{M}$,

$$\Pr_{h\in\mathcal{H}}[h(x) = h(y)] = s_{\mathcal{H}}(x,y),$$

where $s_{\mathcal{H}}$ is a similarity measure of $\mathbb{M}$. Note $s_{\mathcal{H}}$ may not be a kernel or distance defined in a standard fashion; it can be an increasing function of a kernel, like K_1 and K_2, or a decreasing function of a distance, like L_1 and L_2. We say that the similarity measure $s_{\mathcal{H}}$ is induced by LSH family $\mathcal{H}$.

Given an arbitrary locality sensitive hash function $h \in \mathcal{H}$, we also define the corresponding similarity measure as

$$s_h(x,y) = \delta_{h(x),h(y)} = \begin{cases} 1 & \text{if } h(x) = h(y) \\ 0 & \text{if } h(x) \neq h(y), \end{cases}$$

such that $E_{h\in\mathcal{H}}[s_h(x,y)] = s_{\mathcal{H}}(x,y)$. The Kronecker delta δ will appear again later.

LSH families have been devised for various similarity measures. Some of the most popular examples are bit sampling for hamming distance [14], min-wise independent permutation for set similarity (Jaccard index) [2], random hyperplane for cosine similarity [4] and random projection for L_2 distance [5]. In Section 4, we will present two locality sensitive hashing families based on [7] as case studies. These two LSH families will be used in our experimental studies.

3. THE PROPOSED APPROACH

In this section we formalize the random histogram construction algorithm, and establish a correspondence between the histogram kernels K_1, K_2 and the set similarity measures S_1, S_2.

3.1 Random Histograms Construction

Our approach is to represent a set of features as a histogram. However, creating a histogram in the original feature space is not always possible if the feature space is not simply a vector space, or may lead to several issues: the histogram can be extremely large and sparse if the dimensionality is high, and multiple resolutions may be needed to attain reasonable accuracy. To avoid such issues, we propose to first project the feature space into an auxiliary space, and create histograms on the auxiliary space. The mapping used for projection should preserve the locality information, so point similarity is preserved in the auxiliary space. LSH is a perfect tool for this purpose.

We first formalize the concept of *histogram* as used in this paper. Given a finite set $\mathbb{A}$, let $\{\mathbf{e}[i] \mid i \in \mathbb{A}\}$ be the standard basis of the $|\mathbb{A}|$-dimensional vector space, so that for any $i, j \in \mathbb{A}$,

$$\mathbf{e}[i] \cdot \mathbf{e}[j] = \delta_{i,j}.$$

A histogram G over the set $\mathbb{A}$ is represented as a point in the $|\mathbb{A}|$-dimensional vector space, *i.e.* $G = \sum_{i\in\mathbb{A}} \alpha_i \mathbf{e}_i$, with α_i being the "count" of bin i.

We then formalize how to embed a set of features into a histogram. Let $\mathbb{M}$ be an arbitrary feature space, and $\mathcal{H}$ be an LSH family of mappings from $\mathbb{M}$ to $\mathbb{A}$. Note that $\mathbb{M}$ can

[1]We use the definition of LSH given in [4], which better fits our purpose, rather than the one given in [14, 9].

be any space that admits an LSH family, not necessarily a vector space. We define the family of mappings

$$\mathcal{G}_{\mathcal{H}} = \{g_h : 2^{\mathbb{M}} \to \mathbb{R}^{|\mathbb{A}|} \mid h \in \mathcal{H}\},^2$$

such that for any $X \subset \mathbb{M}$ and $h \in \mathcal{H}$, X is mapped to a histogram on $\mathbb{A}$:

$$g_h(X) = \sum_{x \in X} \mathbf{e}[h(x)].$$

Because the histogram $g_h(X)$ is determined by the hash function h chosen at random from $\mathcal{H}$, we call it a *random histogram*.

The above scheme can be easily extended to work with sets of weighted features. If $X' \subset \mathbb{R} \times \mathbb{M}$ is a set of weighted features, then we define

$$g_h(X') = \sum_{\langle w,x \rangle \in X'} w \times \mathbf{e}[h(x)].$$

We assume an unweighted set for the rest of this paper because this extension is straightforward.

In practice, we maintain N independent random histograms to improve accuracy. Specifically, we choose N hash functions $H = \langle h_1, \ldots, h_N \rangle$ from $\mathcal{H}$ independently, and concatenate them to form a "super" histogram:

$$g_H(X) = \langle g_{h_1}(X), \ldots, g_{h_N}(X) \rangle.$$

We extend any similarity measure function $f(\cdot, \cdot)$ on single histograms to this "super" histogram:

$$f[g_H(X), g_H(Y)] = \frac{1}{N} \sum_{i=1}^{N} f[g_{h_i}(X), g_{h_i}(Y)].$$

In doing so, we keep the expectation the same, while lowering variance by a factor of $1/N$.

For most applications, the scaling factor $1/N$ will not affect the search or learning result, and the "super" histogram can be treated as a single vector.

3.2 Matching Random Histograms

Recall that we set out to describe our technique to efficiently estimate similarity between sets of feature vectors. So far we have explained the representation of sets of features as histograms. Now we explain how we compute the similarity for these histograms. In principle, one can use any histogram matching distances or kernels with the random histogram. In this paper, we discuss two special cases, *i.e.* the K_1 and K_2 kernels, corresponding to two set similarity measures S_1 and S_2 (introduced earlier) that have been proved practically effective. Given two sets $X, Y \subset \mathbb{M}$, and a mapping $h \in \mathcal{H}$, the kernels $K_{1,h}$ and $K_{2,h}$ induced by the mapping h are just the K_1 and K_2 kernels in the histogram space:

$$K_{1,h}(X,Y) = \sum_{i \in \mathbb{A}} \min\{g_h(X) \cdot \mathbf{e}[i],\ g_h(Y) \cdot \mathbf{e}[i]\}$$

$$K_{2,h}(X,Y) = \sum_{i \in \mathbb{A}} \{g_h(X) \cdot \mathbf{e}[i]\} \times \{g_h(Y) \cdot \mathbf{e}[i]\}.$$

We will explain how these two histogram kernels relate to the two set similarity measures S_1 and S_2.

[2]For an arbitrary set $\mathbb{M}$, the power set $2^{\mathbb{M}}$ is the set of all subsets of $\mathbb{M}$.

3.2.1 One-to-One Matching

The kernel $K_{1,h}$ corresponds to one-to-one matching. The intuition is that points hashed into the same bin by h can be matched to each other, and for each bin, the set with fewer points in this bin is always guaranteed to have its members matched. Formally, the point similarity measure induced by h is s_h and the corresponding set similarity measure for one-to-one matching is $S_{1,h}$. It can be easily verified that

$$K_{1,h}(X,Y) = S_{1,h}(X,Y) \quad \forall X, Y \in \mathbb{M}.$$

However, this equality does not strictly carry over to the similarity $s_{\mathcal{H}}$ and $S_{1,\mathcal{H}}$ which are actually interesting. Though $s_{\mathcal{H}}(X,Y) = E[s_h(X,Y)]$, we only have $E[K_{1,h}(X,Y)] \geq S_{1,\mathcal{H}}(X,Y)$. In the worst case, $E[K_{1,h}(X,Y)]$ can be significantly larger than $S_{1,\mathcal{H}}(X,Y)$.

Nevertheless, s_h does contain information about $s_{\mathcal{H}}$. In practice, using $K_{1,h}$ as a biased approximation of $S_{1,\mathcal{H}}(X,Y)$ yields good results when one-to-one matching is desirable.

3.2.2 One-to-Many Matching

The kernel $K_{2,h}$ corresponds to one-to-many matching, and we show a direct connection between the histogram kernel $K_{2,h}$ and the set similarity measure $S_{2,\mathcal{H}}$. We have

$$E_{h \in \mathcal{H}}[K_{2,h}(X,Y)] = S_{2,\mathcal{H}}(X,Y), \quad \forall X, Y \subset \mathbb{M}.$$

This can be seen by the following:

$$\begin{aligned} & K_{2,h}[X,Y] \\ =& g_h(X) \cdot g_h(Y) = \left\{ \sum_{x \in X} \mathbf{e}[h(x)] \right\} \cdot \left\{ \sum_{y \in Y} \mathbf{e}[h(y)] \right\} \\ =& \sum_{x \in X} \sum_{y \in Y} \mathbf{e}[h(x)] \cdot \mathbf{e}[h(y)] = \sum_{x \in X} \sum_{y \in Y} \delta_{h(x),h(y)} \end{aligned}$$

Because $E_{h \in \mathcal{H}}[\delta_{h(x),h(y)}] = \Pr_{h \in \mathcal{H}}[h(x) = h(y)] = s_{\mathcal{H}}(x,y)$, the relationship can be obtained by linearity of expectation.

4. POINT SIMILARITY: CASE STUDIES

In this section, we present LSH families for two point similarities, *i.e.* L_2 distance and cosine similarity. In order to facilitate tuning of histogram sizes to provide a size/performance trade-off, we build LSH families from 0/1 valued *atomic* LSH families. Let $\mathcal{F}$ be an atomic LSH family, we concatenate B independent atomic hash functions to make a B-bit hash function: $\mathcal{H} = \{\langle f_1, \ldots, f_B \rangle \mid f_i \in \mathcal{F}\}$. We use $\mathcal{H}$ to build random histograms, which are of size 2^B.

Concatenating B atomic mapping has the effect of enhance locality sensitiveness. The similarities induced by $\mathcal{H}$ and $\mathcal{F}$ have the relationship $s_{\mathcal{H}}(x,y) = [s_{\mathcal{F}}(x,y)]^B$ for any $x, y \in \mathbb{M}$. Figure 2 shows such a relationship. The effect of a larger B is to have a smaller range (at the high end) of point similarity that actually contributes to the set similarity (i.e. only very similar pairs of points have an effect on the set similarity value). In practice, this often means higher discriminative power, and thus higher retrieval or recognition accuracy, but at the same time, a larger histogram size is required. Such threshold effect has been proved effective in previous studies [22, 25].

An easier approach to tune histogram size is to use mappings that produce an integer value, and use the modulo operation to limit the range of this value. One potential drawback of this approach is that the mapping might con-

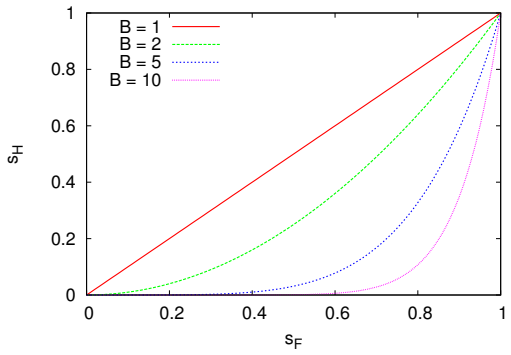

Figure 2: Concatenating B atomic mappings enhances the locality sensitiveness.

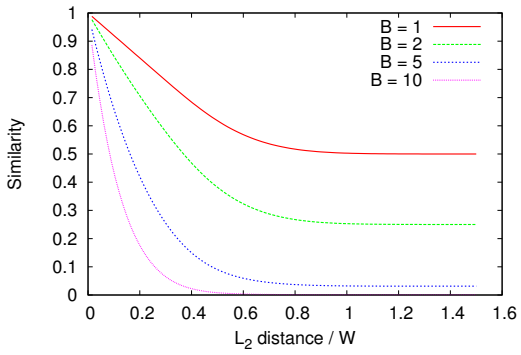

Figure 3: The similarity induced by LSH (B bits) as a decreasing function of the L_2 distance.

centrate on a few values, resulting in a biased histogram with some bins more often used than others.

In the following two subsections, we present the atomic LSH families for L_2 distance and cosine similarity. The same mappings have been used to compress high dimensional vectors into bit vectors [7]. Both schemes assume that the feature space is d dimensional vector space, *i.e.* $\mathbb{M} = \mathbb{R}^d$. The choice between different LSH families depends on the underlying point similarity measure of interest.

4.1 L_2 Distance

The atomic LSH family for L_2 distance is based on random projection. For a point $p \in \mathbb{R}^d$, we define the following atomic LSH function:

$$f(p) = \lfloor \frac{A \cdot p + b}{W} \rfloor \mod 2$$

where $A \in \mathbb{R}^d$ is a random vector with each dimension sampled independently from the standard Gaussian distribution $N(0,1)$, and $b \in R$ is sampled from the uniform distribution $U[0, W)$. W is called the window size, which controls the distance range that the mapping is sensitive to.

The similarity $s_{\mathcal{F}}$ is a function of L_2 distance d between two points, and the following relationship holds [7]:

$$s_{\mathcal{F}} = E_{f \in \mathcal{F}}[s_f] = 1 - \int_0^1 \int_0^1 \frac{W}{d} \sum_{k \in \mathbb{Z}} \phi \left[\frac{W}{d}(2k + x + y) \right] dxdy,$$

where ϕ is the probability density function of the standard Gaussian distribution. See Figure 3 for this relationship. We can see that when the distance is small, there is a near-linear relationship between d and $s_{\mathcal{F}}$, and after certain threshold, $s_{\mathcal{F}}$ quickly converges to a small value. We can also see the effect of concatenating B atomic mappings.

4.2 Cosine similarity

Some applications, like text document retrieval, use the angle, or the cosine of the angle (cosine similarity), between two feature vectors as a similarity measure. Cosine similarity can also be used as a set similarity measure, when sets are represented as 0/1 vectors. The cosine similarity is also closely related to the widely used statistical indicator Pearson correlation. For any $p, q \in \mathbb{R}^d$, the cosine similarity is defined by $d_{cos}(p,q) = \frac{p \cdot q}{\|p\|_2 \|q\|_2}$, and the angle between two vectors is defined accordingly.

The LSH family to approximate the angle between vectors is based on the random hyper-plane technique [4]. For a point $p \in \mathbb{R}^n$, we define the atomic LSH as

$$f(p) = \begin{cases} 0 & \text{if } \rho \cdot p < 0 \\ 1 & \text{if } \rho \cdot p \geq 0 \end{cases}$$

where ρ is a random d-dimensional vector sampled from the unit hyper-sphere $\{v \in \mathbb{R}^d \mid \|v\|_2 = 1\}$. Suppose the angle between two points p and q is θ, we have

$$s_{\mathcal{F}}(p,q) = E_{f \in \mathcal{F}}[s_f(p,q)] = 1 - \frac{\theta}{\pi}.$$

5. PRACTICAL ISSUES

5.1 Compact Histogram Representation

Usually the histogram size needs to be large for high discriminative power. The result is that 1) the histogram can be very sparse and 2) the count of each bin is small. Next, we describe two practical techniques for compact histogram representation. These methods are specially designed to work with random histograms for unweighted features. Other general dimension reduction methods like PCA, and sketches [7] can be applied to the histograms for lossy compression.

5.1.1 *Folding*

The folding technique applies to the K_2 histogram kernel (inner product). The main objective of folding is to produce a more accurate estimator of similarity without increasing the space and time requirement. Basically the histograms corresponding to several different hash functions are added together; hence the name. The K_2 histogram kernel applied to this folded histogram has roughly the same expectation as the kernel applied to each individual histogram, but has lower variance. This technique is based on the following observations: the same LSH function causes similar points to collide, but if two points are mapped with different LSH functions, the values are independent. Specifically, for two points x, y, and two different B-bit LSH functions h_1, h_2, if a random atomic hash function is equally likely to map a point to 0 or 1 and if the atomic hash functions are picked independently of each other, we have

$$\Pr[h_1(x) = h_2(y)] = 2^{-B}$$

The probability is close to 0 when B is large. As a result, for two sets X and Y, $g_{h_1}(X) \cdot g_{h_2}(Y)$ should also be small:

$$E[g_{h_1}(X) \cdot g_{h_2}(Y)] = \sum_{x \in X} \sum_{y \in Y} \Pr[h_1(x) = h_2(y)] = 2^{-B}|X|\,|Y|.$$

Our folding technique is simply to add multiple independent random histograms together to make a single vector. Given M independent LSH mappings $H = \{h_1, h_2, \ldots, h_M\} \subset \mathcal{H}$, we use the following conceptual random histogram

$$g_H(X) = \frac{1}{\sqrt{M}} \sum_{i=1}^{M} g_{h_i}(X).$$

We have

$$\begin{aligned}
&E[g_H(X) \cdot g_H(Y)] \\
=&E[\left\{ \frac{1}{\sqrt{M}} \sum_i g_{h_i}(X) \right\} \cdot \left\{ \frac{1}{\sqrt{M}} \sum_j g_{h_j}(Y) \right\}] \\
=&E[\frac{1}{M} \sum_i g_{h_i}(X) \cdot g_{h_i}(Y) + \frac{1}{M} \sum_{i \neq j} g_{h_i}(X) \cdot g_{h_j}(Y)] \\
=&E[g_{h_1}(X) \cdot g_{h_1}(Y) + (M-1)E[g_{h_1}(X) \cdot g_{h_2}(Y)] \\
=&E[g_{h_1}(X) \cdot g_{h_1}(Y)] + (M-1)2^{-B}|X|\,|Y|.
\end{aligned}$$

Thus the value of the folded estimator is essentially an unbiased main term plus an error term. The error term only depends on the sizes of X and Y, and is not locality sensitive. The effect of folding is that the error is reduced by lowering the variance of the main term (roughly by a factor of M), with a small amount of extra error introduced. When the similarity value is higher than a certain threshold, the main term is dominant, and the overall effect of folding is positive. In practice, high similarity values are more important, and thus folding can potentially improve the end performance. The limitation is that folding is only helpful when the histograms are very sparse, and the parameter M cannot be too large. We use both folding and concatenation in our final scheme.

5.1.2 Bit-Coding

If bin counts of the histogram are still small after folding, it will be wasteful to use full integers or floating point numbers to represent such small counts. We can fit multiple bins into a single byte via bit-encoding. Assume we encode each bin with $C = 1, 2, 4, 8$ bits, then each byte can hold $8/C$ bins. If the count of a bin is larger than 2^C, the exceeding part is trimmed. When computing the kernel functions online, it would be slow if we first extract the bin counts from the bit-encoding. We thus use a lookup table of 256×256 entries with pre-computed kernel values. By doing so, the $8/C$ multiplications or minimum operations are substituted with a single memory lookup. However, bit-encoding is a specialized representation which might not be supported by existing software, and we do not use it in the experiments reported in this paper.

5.2 Complexity and Parameter Tuning

Figure 4 illustrates the whole scheme of our proposed method, showing all the tunable parameters.

The complexity of our method has two aspects — offline and online, and in practice, the latter is usually more important. Assume the feature space has a dimension of d, and the average set size is n. The off-line part involves generating the hash values and adding to the histogram bins. It is straightforward that the time complexity to embed a single set is $O(dnBMN + 2^B N)$ (for dense representation, $2^B N$ is necessary for histogram initialization). The online part depends entirely on the histogram size, and thus takes time $O(2^B N)$. The online part does not explicitly depend on the set size or the dimensionality, and can be potentially very fast.

We briefly comment on how to tune each of the parameters for a given dataset.

- The parameter B determines the size of a single histogram. Larger B usually improves the accuracy, but exponentially enlarges the histogram size. So B is more limited by available space. A good starting point for tuning B is to make the histogram size (2^B) similar to the average set size.

- The parameter N lowers the variance of the result. This is necessary as our method is a randomized algorithm. Our suggested range of values for N is 10 to 50.

- The parameter M determines the number of histograms folded together. It depends on the histogram size, and can be slightly enlarged as the histogram expands. Our experiments show that M is mostly useful when less than 5.

- The LSH family used actually has the largest impact on the result. The right LSH family depends on the dataset, and different choices of LSH need to be tested if the feature extraction algorithm does not suggest an obvious choice.

6. EVALUATION

In this section, we evaluate the random histogram technique with real-life datasets under three different task settings, *i.e.* object recognition, content-based image retrieval, and near-duplicate video clip detection. The tasks are experimented with different configurations of parameters to illustrate the flexibility of our method—they either use different LSH families or different set similarity measure. We show the comparable accuracy of our method with the state-of-the-art set matching techniques, mainly EMD [27] and PMK [13], as well as our superiority in speed. The effect of different parameters to accuracy is evaluated in detail with the Caltech-101 dataset to serve as a guide for parameter tuning.

6.1 Object Recognition

This task is to recognize the semantic category of images. We evaluate our method with two datasets, *i.e.* Caltech-101 and Graz-01. The former focuses on distinguishing between different object categories, and the latter emphasizes on predicting whether an object of certain single category appears in the image or not,

Following the previous studies, we use SVM for machine learning. Because the random histograms are just high-dimensional vectors, off-the-shelf software like [16, 3] can be directly applied.

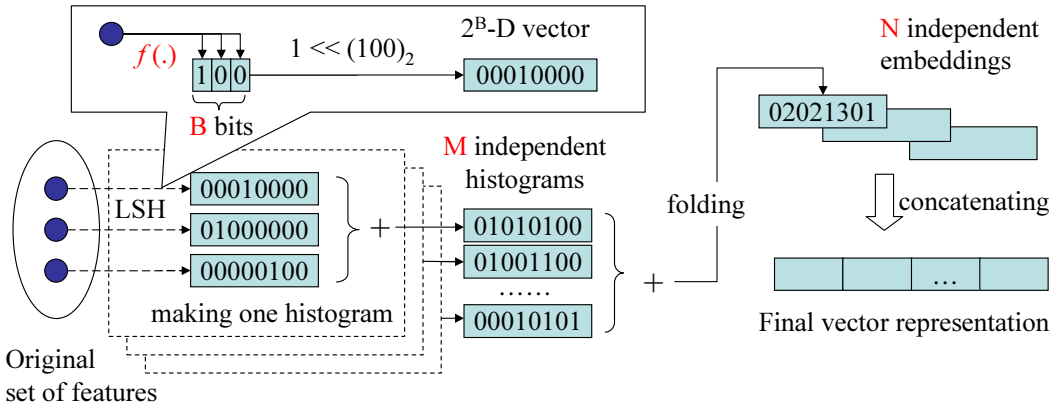

Figure 4: Overview of the embedding scheme

6.1.1 *Caltech-101*

The Caltech-101 dataset [8] contains 101 object categories, with 40 to 800 images per category. The names of the categories were chosen from the Webster Collegiate Dictionary and the images were manually selected from Google image search results. Most of the images are of medium resolution, *i.e.* about 300×300 pixels, and contain little or no clutter. The objects tend to lie in the center of the images and to be presented in similar poses. Images from some categories are artificially rotated and have partially black background.

We evaluate our method with the same features as used in [11]. The features were extracted on a uniform grid from the images, and each set has on average 1140 features. Each feature is the concatenation of a 10D PCA-SIFT [17] descriptor and a 2D positional feature. The standard evaluation protocol with the dataset is used, *i.e.* 15 images from each category are selected as training example, and a model is trained with the training images of all categories. The model is then used to predict the category of the remaining images. The recognition rate of each category is averaged as the overall performance.

Table 1 compares the performance of our method and some of the previous methods based on the set-of-feature model. Zhang [30] use an EMD based kernel, but their feature sets are slightly different from ours. First, they use four combinations of interesting point detectors (Harris + Laplacian) and descriptors (SIFT and SPIN), each channel is matched individually and the sum of the channel similarity is used as the kernel value for learning. Second, when matching in a single channel, the features of a image are first clustered, and EMD is applied upon the clusters as weighted features. This method uses more information than [13] and our method, and is the best result so far achieved by a pure set-of-feature model. The Spatial Pyramid Matching Kernel proposed by Lazebnik [19] represents the best result achieved on the dataset among all existing approaches. They divide the features into different channels by clustering the features of all the images, and then do pyramid match in each channel with the positional features. We see in Table 1 that our fast configuration achieves roughly the same performance of PMK, and the slow configuration approaches that of Wang [30]. We run our experiment on a Pentium 4 2.2GHz CPU, and the fast configuration takes only 4×10^{-6} second to match a pair of images, while the number reported for PMK on a slightly faster machine (2.4GHz) was 1×10^{-4} [13].

Grauman [13]	Zhang [30]	Lazebnik [19]	Ours A	Ours B
0.50	0.539	0.646	0.497	0.541

Table 1: Comparison of different methods with Caltech-101 benchmark. We test two different configurations of our methods. Both use the LSH for L_2 distance and match histograms with K_2 kernel. Other parameters are: (A, for speed) $B = 6, M = 5, N = 20$; (B, for accuracy) $B = 10, M = 20, N = 20$.

Thus our method is 25 times the speed of PMK for matching images from Caltech-101 benchmark to achieve the same recognition performance.

Lazebnik [19] out-performs all other methods in this task. This is partly due to that the images in the dataset tend to present in similar poses, and Lazebnik [19] is specially optimized for such tasks. We can see that with the Graz-01 dataset, which has more variations, the performance difference is not this significant.

Figure 5 shows the effects of different parameters on the performance. These curves are just as expected in our discussion in the previous section. Generally, enlarging the single histogram size (B), folding more histograms (M) and concatenating more histograms (N) all lead to higher performance, and all have a threshold beyond which the performance improvement becomes very slow. For B the threshold is around $8 \sim 10$, and for N it is 20. For M, the threshold depends on the histogram size, and folding tends to add more to the performance when histograms are large.

We also evaluated our method on a similar but easier dataset ETH-80 [20] with the same protocol as used by [11]. We achieve an accuracy of 84.3%, using the configuration $B = 8, M = 20, N = 20$, which is similar to the performance of PMK, which is 84%.

6.1.2 *Graz-01*

The Graz-01 dataset [26] contains two object classes, bike (373 images), person(460 images), and a background class(270 images). This dataset is of high intra-class scale and pose variation. We use the Difference-of-Gaussian (DoG) detector and SIFT descriptor for feature extraction [21].

We perform two-class (object vs. background) categorization using the same experimental setup of [26]. For each ob-

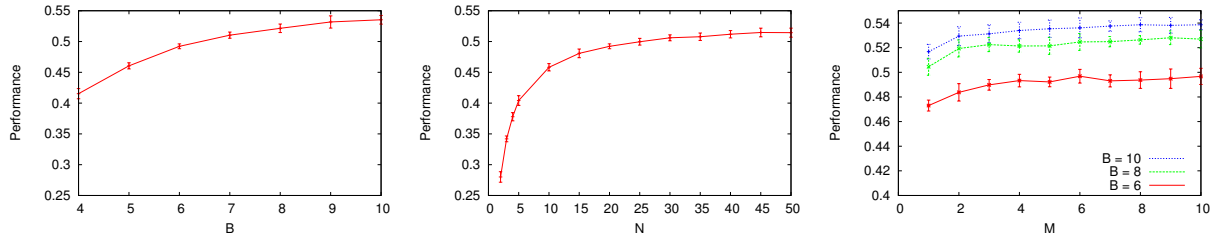

Figure 5: Recognition performance vs. different embedding parameters. The embedding has $N \times 2^B$ dimensions, and the figures indicate that accuracy generally increases with dimensionality. When histograms are sparse (with large B), folding (increasing M) can also improve accuracy without extra space requirement.

	Opelt [26]	Zhang [30]	Lazebnik [19]	Ours
Bike	0.865	0.920	0.863	0.883
Person	0.808	0.880	0.823	0.805

Table 2: Comparison of different methods with Graz-01 benchmark. We use LSH for L_2 distance and match histograms with K_2 kernel. Other parameters are: $B = 12, M = 20, N = 20$

Method	SIMPLIcity	EMD	EMD*	Ours
Avg. Prec.	0.331	0.548	0.615	0.548
Time/match	N/A	5.0e-5	5.0e-5	3.1e-7

Table 3: Comparison of SIMPLIcity, EMD, EMD* and out method with a 10K image benchmark. We use LSH for L_2 distance, with $W = 1$ and match histograms with K_2 kernel. Other parameters are: $B = 4, M = 1, N = 50$.

ject class, we train a model on 100 positive and 100 negative images (of which 50 are drawn from the other object class and 50 from the background class). We generate ROC curves by thresholding raw SVM output, and report the ROC equal error rate, which is the point on the ROC curve where

$$\frac{\text{true positive}}{\text{positive}} = 1 - \frac{\text{false positive}}{\text{negative}}.$$

We show the recognition rates in Table 2. We optimize the parameters of our method for accuracy, and our result is on a similar level as other methods. Here the gap between our method and Zhang [30] is more obvious, showing accuracy loss of our randomized algorithm and its potential limitation in application to tasks requiring very high accuracy.

6.2 Content-Based Image Retrieval

We use the same image collection and feature extraction algorithm as in [22], where content-based image retrieval with a tweaked version of EMD distance is studied. The image collection consists about 10,000 general-purpose images. We use 32 sets of similar images as the ground truth in our evaluation. These sets are manually selected, and each of these sets contains images that are visually similar. We use JSEG [6] to segment the images into regions. The average number of regions per image is 7.16. We extract from each region a feature vector of 14-dimensions, including 9 dimensions of color moment information and 5 dimensions of bounding box information. We use the region weight that is proportional to the square-root of the region's size, which as shown by [22] is better than the raw region's size.

We use average precision to measure the effectiveness of the retrieving method. Given a query and k relevant items excluding the query, let rank_i be the rank of the ith retrieved relevant item, then average precision is defined as following:

$$\text{average precision} = \frac{1}{k}\sum_{i=1}^{k}\frac{i}{\text{rank}_i}.$$

Table 3 compares four methods: SIMPLIcity [28], EMD, EMD* and random histogram. The performance of random histogram is the same as EMD. The highest performance is achieved by EMD*, which involves a thresholding tweak that is highly specialized to the particular feature extraction method.

Though we cannot directly compare the image retrieval speed with [22] without implementing the system, we note that the retrieval process of [22] need to re-rank 5% the whole dataset with EMD. Thus, even if we scan all the histograms in brute-force, our method only cost the time of their ranking phase, and is around 3.5 times faster than SIMPLicity [28]. In practice, our method can be further accelerated by various high-dimensional indexing data structures.

6.3 Near-Duplicate Video Detection

We also experiment our embedding methods with the task of near duplicate detection of video clips. The near duplicate video clips benchmark is provided by Wu [29]. The benchmark consists of 24 sets of video clips downloaded from websites such as Youtube using specific keywords. Each set of videos are manually labeled by human to find near duplicate video clips, the first relevant video clip is used as the query video to find other near duplicate videos within the set. Due to the difference in the time that the videos are downloaded, our video collection (7127 video clips) is a subset of the video clips used in paper [29].

We first segment the video and take one key frame from each segment, resulting in an average of 124 key frames per video clip. Then we extract a simple HSV based color histogram from each key frame. The 24 dimensional HSV color histogram is concatenated with 18 bins for Hue, 3 bins for Saturation and 3 bins for Value as described in [29]. Finally we embed this set-of-feature representation into our histogram representation. Because video clips involve high occurrence of near-duplicate features, which can dominate the set similarity easily under one-to-many match, we use K_1 kernel to match histograms.

	Wu SIG_CH	Wu SET_NDK	ours SIG_CH	ours Embedding
Average Precision	0.892	0.952	0.835	0.893
Speed (second)	"fast"	"minutes"	1.7e-4	4.6e-3

Table 4: Performance comparison for near duplicate video detection. We use LSH for cosine similarity and match histograms with K_1 kernel. Other parameters are: $B = 8, M = 3, N = 10$.

Wu's paper proposed two methods to do near duplicate detection. One is to use a global signature with HSV color histogram (SIG_CH) which is fast but with less precision; the other is to use expensive local feature descriptor based pairwise comparison among key frames (SET_NDK). The performance metric used is the average of Mean Average Precision for each video set. Table 4 shows that by using our embedding method to embed the set of color histogram, we can achieve similar percentage improvement over SIG_CH while maintaining high speed. Note that due to the difference in video clips and segmentation method, our SIG_CH result is not directly comparable to Wu's SIG_CH.

7. RELATED WORK

Existing methods for measuring set similarity roughly fall into two categories. Methods in the first category depend on a given point similarity measure, and directly work on the sets by matching the member points. The S_1 similarity for one-to-one matching [27] and S_2 similarity for one-to-many matching [25] are two commonly used approaches. Such methods usually involve high on-line computational cost. Further more, storing raw sets of high-dimensional features can easily outgrow the limited main memory, leading to an out-of-core implementation and further slowing down the computation.

Methods in the second category, on the other hand, calculate set similarity via certain intermediate representation. They assume that the points are from certain metric space, often the Euclidean space, making the point similarity measure implicit. The intermediate representation can be a histogram created by quantizing the space. For example, Pyramid Matching [11, 13, 15] corresponds to scalar quantization, and the bag-of-word model commonly used in computer vision often use vector quantization [30]. These methods either require large space to store the high dimensional sparse histograms, or long running off-line clustering. The intermediate representation can also be parameters of certain distributional model, *e.g.* Gaussian-mixture [24], though such methods also tend to be CPU intensive, and are not as general.

Among the various methods mentioned above, the pyramid method [11, 13, 15] and the bag-of-word model are the two that are most related to our work. Note that [15] and [11, 13] are essentially the same method developed by two groups of people, the former in the distance metric setting and the latter in the kernel setting. They scalar-quantize the feature space in multiple resolutions and maintain a histogram for each resolution. Though in practice a sparse data structure is used, the cost in space is still significant when the feature space is of high dimensionality, and dimension reduction methods like PCA need to be applied first. A recent work [12] alleviates the problem at the cost of a time-consuming off-line clustering phase and a slower online matching process.

The bag-of-word model works by clustering the feature vectors into a fixed number of key words, turning a set of points into a bag of words. The clustering process is essentially vector-quantization of the feature space. The mapping from feature vectors to key words induced by the clustering result can be viewed as a deterministic locality sensitive hashing, while from the other perspective, our method is just a randomized version of the bag-of-word model.

The idea of LSH was original proposed to solve the high-dimensional indexing problem [14, 9]. The 0/1 valued mappings used in this paper are based on [7], where they are used to construct sketches for high dimensional vectors. LSH can also be used to construct bloom filters like ordinary hash function [18]. Such bloom filters have a special property that allows one to estimate the distance between a point and the sets of point embedded in the bloom filter. The random histogram proposed in this paper can be viewed as an extended version of such bloom filter. We not only record whether each bucket is empty, but also keep a count for each bucket.

The compact data structure proposed in [22] to approximate EMD distance is actually a specialized version of our method. Their final data structure is more compact than ours via a further level of sampling and sketching, but they need to maintain the raw data for re-ranking due to the precision loss of compression. With our method, the raw data is no longer needed.

8. CONCLUSION

In this paper we propose an efficient method for representing and estimating similarities between sets of features. Representing multimedia objects as sets of local features has been quite successful recently, but suffers from high storage requirements and high time complexity. Our main idea is to project the feature points into a compact auxiliary space with LSH mappings, and then represent the sets as histograms on the auxiliary space. The histograms are just standard high-dimensional vectors. Our method has the flexibility that allow the user to approximate different measures of point similarity by choosing corresponding LSH families, and implement different measures of set similarity with appropriate choice of histogram distances/kernels. Experiments with various datasets under different task settings show that our method has accuracy comparable to the best-known set matching algorithms, and has much higher speed. For example, to achieve the same recognition performance on the Caltech-101 dataset, our method is 25 times the speed of Pyramid Matching Kernel for set matching, the latter being the fastest known set matching algorithm. Because our histogram embedding is essentially a global feature extracted from the intermediate local feature representation, our results show that global features, when properly extracted, can also achieve high accuracy while maintaining a significant speed advantage.

Acknowledgments

This work is supported in part by NSF grants EIA-0101247, CCR-0205594, CCR-0237113, CNS-0509447, DMS-0528414 and by research grants from Google, Intel, Microsoft, and Yahoo!. Wei Dong is supported by Gordon Wu Fellowship. We would like to thank John Lee for providing us with the Caltech 101 feature data used in [13].

9. REFERENCES

[1] `http://www.cs.princeton.edu/cass`.

[2] A. Z. Broder, M. Charikar, A. M. Frieze, and M. Mitzenmacher. Min-wise independent permutations (extended abstract). In *STOC '98: Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 327–336, New York, NY, USA, 1998. ACM.

[3] C.-C. Chang and C.-J. Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[4] M. S. Charikar. Similarity estimation techniques from rounding algorithms. In *STOC '02: Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 380–388, New York, NY, USA, 2002. ACM.

[5] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *SCG '04: Proceedings of the twentieth annual symposium on Computational geometry*, pages 253–262, New York, NY, USA, 2004. ACM.

[6] Y. Deng and B. S. Manjunath. Unsupervised segmentation of color-texture regions in images and video. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 23(8):800–810, 2001.

[7] W. Dong, M. Charikar, and K. Li. High dimensional similarity search with sketches. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research & Development on Information Retrieval*, Singapore, July 2008.

[8] L. Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. *Computer Vision and Image Understanding*, In Press, Corrected Proof.

[9] A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions via hashing. In *VLDB '99: Proceedings of the 25th International Conference on Very Large Data Bases*, pages 518–529, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

[10] K. Grauman and T. Darrell. Pyramid match hashing: Sub-linear time indexing over partial correspondences. *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, pages 1–8, 17-22 June 2007.

[11] K. Grauman and T. Darrell. The pyramid match kernel: Discriminative classification with sets of image features. In *ICCV '05: Proceedings of the Tenth IEEE International Conference on Computer Vision*, pages 1458–1465, Washington, DC, USA, 2005. IEEE Computer Society.

[12] K. Grauman and T. Darrell. Approximate correspondences in high dimensions. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 505–512. MIT Press, Cambridge, MA, 2007.

[13] K. Grauman and T. Darrell. The pyramid match kernel: Efficient learning with sets of features. *J. Mach. Learn. Res.*, 8:725–760, 2007.

[14] P. Indyk and R. Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *STOC '98: Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 604–613, New York, NY, USA, 1998. ACM.

[15] P. Indyk and N. Thaper. Fast image retrieval via embeddings. In *3rd International Workshop on Statistical and Computational Theories of Vision*, 2003.

[16] T. Joachims. Training linear svms in linear time. In *KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 217–226, New York, NY, USA, 2006. ACM.

[17] Y. Ke and R. Sukthankar. Pca-sift: A more distinctive representation for local image descriptors. *cvpr*, 02:506–513, 2004.

[18] A. Kirsch and M. Mitzenmacher. Distance-sensitive bloom filters. In *ALENEX '06: Proceedings of the Ninth Workshop on Algorithm Engineering and Experiments*, pages 41–50, Philadelphia, PA, USA, 2006. Society for Industrial and Applied Mathematics.

[19] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR '06: Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2169–2178, Washington, DC, USA, 2006. IEEE Computer Society.

[20] B. Leibe and B. Schiele. Analyzing appearance and contour based methods for object categorization. *cvpr*, 02:409, 2003.

[21] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, 2004.

[22] Q. Lv, M. Charikar, and K. Li. Image similarity search with compact data structures. In *CIKM '04: Proceedings of the thirteenth ACM international conference on Information and knowledge management*, pages 208–217, New York, NY, USA, 2004. ACM.

[23] Q. Lv, W. Josephson, Z. Wang, M. Charikar, and K. Li. Ferret: a toolkit for content-based similarity search of feature-rich data. In *EuroSys '06: Proceedings of the 1st ACM SIGOPS/EuroSys European Conference on Computer Systems 2006*, pages 317–330, New York, NY, USA, 2006. ACM.

[24] S. Lyu. A kernel between unordered sets of data: The gaussian mixture approach. In *European Conference on Machine Learning (ECML)*, Porto, Portugal, 2005.

[25] S. Lyu. Mercer kernels for object recognition with local features. In *CVPR '05: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 2*, pages 223–229, Washington, DC, USA, 2005. IEEE Computer Society.

[26] A. Opelt, M. Fussenegger, A. Pinz, and P. Auer. *Weak Hypotheses and Boosting for Generic Object Detection and Recognition*, volume 3022/2004 of *Lecture Notes in Computer Science*, pages 71–84. Springer Berlin / Heidelberg, 2004.

[27] Y. Rubner, C. Tomasi, and L. J. Guibas. The earth mover's distance as a metric for image retrieval. *Int. J. Comput. Vision*, 40(2):99–121, 2000.

[28] J. Z. Wang, J. Li, and G. Wiederhold. Simplicity: Semantics-sensitive integrated matching for picture libraries. *IEEE Trans. Pattern Anal. Mach. Intell.*, 23(9):947–963, 2001.

[29] X. Wu, A. G. Hauptmann, and C.-W. Ngo. Practical elimination of near-duplicates from web video search. In *MULTIMEDIA '07: Proceedings of the 15th international conference on Multimedia*, pages 218–227, New York, NY, USA, 2007. ACM.

[30] J. Zhang, M. Marszałek, S. Lazebnik, and C. Schmid. Local features and kernels for classification of texture and object categories: A comprehensive study. *Int. J. Comput. Vision*, 73(2):213–238, 2007.