Randomized Locality Sensitive Vocabularies for Bag-of-Features Model*

Yadong Mu[1], Ju Sun[1,2], Tony X. Han[3], Loong-Fah Cheong[1,2], Shuicheng Yan[1]

[1]Electrical and Computer Engineering, National University of Singapore, Singapore
[2]Interactive & Digital Media Institute, National University of Singapore, Singapore
[3]Electrical and Computer Engineering, University of Missouri-Columbia, USA
Email: {elemy, idmsj, eleclf, eleyans}@nus.edu.sg, hantx@missouri.edu

Abstract. Visual vocabulary construction is an integral part of the popular Bag-of-Features (BOF) model. When visual data scale up (in terms of the dimensionality of features or/and the number of samples), most existing algorithms (e.g. k-means) become unfavorable due to the prohibitive time and space requirements. In this paper we propose the *random locality sensitive vocabulary* (RLSV) scheme towards efficient visual vocabulary construction in such scenarios. Integrating ideas from the Locality Sensitive Hashing (LSH) and the Random Forest (RF), RLSV generates and aggregates multiple visual vocabularies based on random projections, without taking clustering or training efforts. This simple scheme demonstrates superior time and space efficiency over prior methods, in both theory and practice, while often achieving comparable or even better performances. Besides, extensions to supervised and kernelized vocabulary constructions are also discussed and experimented with.

1 Introduction

The *bag-of-features* (BOF) model (also known as the *bag-of-words*) has gained much empirical success in producing orderless representations of feature-rich vision data. In this model, in order to obtain uniform representations for feature sets of varying cardinalities, one performs feature quantization for each primitive feature referring to a learnt "visual vocabulary", and then summarizes each set into histograms. Despite its simplicity, the BOF model has shaped the current paradigms towards many high-level vision problems, e.g., object recognition and content-based image retrieval (CBIR). In fact, state-of-the-art approaches to object recognition are to first extract local descriptors such as SIFT [1] from interest points (i.e., local extrema in scale space pyramid) in images, and then devise Mercer kernels such as Pyramid Match Kernels (PMK) or Histogram Intersectional Kernels (HIK) between pairwise feature histograms. Finally, sophisticated classifiers such as the Supporting Vector Machines (SVM) are used for per-sample decision.

* Support of IDMPO Grant R-705-000-018-279 Singapore and NRF/IDM Program under research Grant NRF2008IDMIDM004-029 are gratefully acknowledged.

Visual vocabulary construction is critical to the BOF model. In ideal cases, every visual word in the vocabulary bears concrete meaning, or semantic, inspired from the similar idea of *bag-of-words* models in text analysis. In practice, however, quality of the vocabulary depends on numerous factors associated with the vocabulary creation process, such as the source of samples, the number of visual words specified, and the similarity metric. Hence, a practical criterion for a proper visual vocabulary could be visual features near to the same visual word bear some similarities. This in essence turns the vocabulary construction problem into partitioning the visual feature space according to a few data samples. Numerous methods (as described in Sec-1.1) have been proposed to address this partition problem.

Complications associated with vision data, however, stem from the typical scale issue including huge amount and high dimensionality. For example, the popular SIFT descriptor [1] used for local visual feature description has 128 dimensions, while a typical image at normal resolution can produce 1k $\sim$ 2k such primitive feature descriptors. The complexity quickly explodes for real-world vision databases that usually consist of millions or even billions of such images. Moreover, peculiarities with high dimensionality, such as concentration of distribution [2][3], show up frequently and deserve special investigations. Questing into large-scale problems, most techniques that work nicely in low-dimensional spaces with small amount of data may not healthily scale up. This calls for novel solutions that are efficient and dedicate for large-scale data while producing acceptable levels of performance.

1.1 Prior Work

There are intimate connections between the vocabulary construction and the clustering/space partitioning problem, the latter of which is widely studied in machine learning. Hence various unsupervised clustering methods have been applied to this particular problem, among which k-means clustering[1] is the most popular. Other methods include mean-shift [4], tree-based coding (e.g., tree induced by hierarchical k-means) [5]. On the other hand, for problems where supervision or prior knowledge are available, e.g., labels or segmentations for images, sparsity in representation, supervision is applied to partially guide the unsupervised clustering (e.g., random forest [6] based methods such as ERC-Forest [7]), or to learn informative vocabularies directly (e.g.,[8][9]). Nevertheless, most of the supervised vocabulary learning techniques are very expensive and unlikely to scale up nicely. Hence we will focus on unsupervised clustering techniques (e.g. k-means) and extremely efficient and flexible supervised techniques (e.g. the random forest family).

Sparse coding and its various applications [10] are perhaps the most inviting work into high-dimensional spaces for vision research. In fact for clustering [2] reveals that in high-dimensional spaces, clustering methods such as k-means tend

[1] Hereafter we default k-means to the hierarchical k-means algorithm due to its efficiency until otherwise stated.

to return many similar centers to a query, which makes these methods rather unstable. This is partially explained by the concentration of finite number of samples as reported by many authors, e.g. [2][3]. In this case, aggregation of distinct clustering results prove useful to enhancing the stability. This is the underlying principle for the random forest method [6], which also inspires our method.

Randomized algorithms have been widely studied and analyzed in algorithm designs [11]. For large-scale problems, [3] has tried to contrast the random projection with the classic PCA technique. Moreover random vectors sampled from the unit sphere have served as the hashing vectors for one family of LSH [12]. In this vein, there is fruitful research work on LSH theory (e.g.,[13]) and applications (e.g., image database indexing and search[14]). Our method build from the idea of LSH.

1.2 Our Approach

We propose and empirically validate the idea of *randomized locality sensitive vocabulary* (RLSV), which is inspired by both RF and LSH. Instead of building visual vocabularies by optimizing a global objective, our proposed method generates visual words by applying a sequence of random linear bipartitions. Assume all samples are originally embedded in a specific metric space (e.g., Hamming space, ℓ_p-normed space). For any sample pair, theoretic analysis of LSH guarantees that their collision probability in the resulting vocabulary is tightly related to the pairwise similarity or distance. Furthermore, to reduce the inherent randomness, multiple random vocabularies are independently created using the same method, and the final inter-sample distance or similarity is based on the consensus of all vocabularies.

Compared with existing methods, the proposed RLSV has the following merits: 1) No time-consuming training or clustering stage in RLSV. The major computational cost comes from the random hash function generation and histogram binning operations, which can be efficiently handled. 2) Noise resistance and stability by exploiting the ensemble technique. RLSV generates an ensembles of several vocabularies to mitigate the effect of randomness and noise. 3) As stated above, nearest-prototype based methods (e.g., k-means) tend to suffer from the so-called *curse of dimensionality* problem. In contrast, the performance of RLSV is stable for high-dimensional data. 4) Compared to methods such as RF which require supervision, RLSV is unsupervised in nature but can be readily extended to supervised and kernerlized cases.

2 Randomized Locality Sensitive Vocabularies

In this section and the next we elaborate on the proposed vocabulary construction algorithm, and discuss its relationship to existing methods. We provide theoretic analysis on the time and space complexity in contrast with k-means and RF, and also present extensions to the supervised or kernelized cases.

2.1 Preliminaries

A key ingredient to many visual recognition and retrieval applications is to search k-nearest-neighbors (k-NN) for the query (or testing sample). In many cases the performance of the whole system heavily hinges on the efficiency of the k-NN procedure. In practice, exact k-NN is somewhat unnecessary, since in many scenarios approximate nearest neighbors (ANN) results in nearly the same performance. Several efficient algorithms are known for low-dimensional cases (e.g., up to 10 to 20), such as the kd-tree [15] algorithm. However, for high-dimensional cases, these methods often provide little improvement over a linear scan algorithm, which is known to be the phenomenon "the curse of dimensionality". In recent years, various *locality sensitive hashing* (LSH) [16][13] methods are proposed to tackle this problem.

Let $\mathcal{H}$ be an LSH family defined on metric space $\mathbb{R}^d$. For any $x,\ y \in \mathbb{R}^d$, the following relationship holds[2]:

$$\forall h \in \mathcal{H}, \quad Pr[h(x) = h(y)] = \kappa(x, y), \tag{1}$$

where $\kappa(\cdot, \cdot)$ denotes the similarity measure between samples x and y. In other words, x and y's collision probability (i.e., being mapped to the same hash bucket) is monotonically increasing with their similarity value, which is known as the "locality sensitive" property. Several LSH families have been developed for various distances (or similarities). Here we list some representative work:
Arccos distance: for real-valued feature vectors lying on hypersphere $S^{d-1} = \{x \in \mathbb{R}^d \ \mid \ \|x\|_2 = 1\}$, an angle-oriented distance can be defined as $\Theta(x, y) = \arccos\left(\frac{x \cdot y}{\|x\|\|y\|}\right)$. Charikar et al. [12] proposes the following LSH family:

$$h(x) = \begin{cases} 0, \text{ if } & \rho^\top x < 0 \\ 1, \text{ if } & \rho^\top x \geq 0 \end{cases} \tag{2}$$

where the hashing vector ρ is uniformly sampled from the unit hypersphere S^{d-1}. The collision probability is $Pr[h(x) = h(y)] = 1 - \Theta(x, y)/\pi$.
ℓ_p **distance**: for linear vector spaces equipped with the ℓ_p metric, i.e., $D_{\ell_p}(x, y) = \left(\sum_{i=1}^{d} |x_i - y_i|^p\right)^{\frac{1}{p}}$, Datar et al. [16] proposes a hashing algorithm based on linear projections onto a 1-dimensional line and chopping the line into equal-length segments, as below:

$$h(x) = \left\lfloor \frac{\rho^\top x + b}{W} \right\rfloor, \tag{3}$$

where the hashing vector $\rho \in \mathbb{R}^d$ is randomly sampled from the *p-stable distribution* and $\lfloor \cdot \rfloor$ is the flooring function for rounding. W is the data-dependent window size and b is sampled from the uniform distribution $U[0, W)$.

[2] Note that other definitions of LSH exist, such as the one in [17]. However, they are fundamentally equivalent to the current.

We employ the arccos distance family in the current work, and the locality sensitivity property will be key to ensuring that our hashing vectors properly partition the feature space such that near neighbors are grouped together with high probability.

2.2 Overview

K-means is probably the most popular due to its empirical success. The goal of k-means is to seek K prototypes (or cluster centers) that minimizes a pre-specified functional value. These prototypes constitute a Voronoi diagram per se and each sample is assigned to its nearest prototype according to some specific distance metric. Despite its popularity, for an input feature set of size n, the classic k-means requires $\mathcal{O}(Knd)$ operations per iteration and typically costs tens of iterations before convergence, which is computationally forbidden for massive data source, high dimensional feature spaces and large vocabulary sizes. Tree structured vocabulary [5][7] requires shorter training time compared with k-means, yet consuming exponentially increasing memory space (w.r.t. the tree depth) to store the splitting information (random dimension, threshold etc.) of inner tree nodes.

Compared with these aforementioned structures, the proposed RLSV method is superior in terms of both memory requirement and training time. The major weakness is the inferior discriminant ability of single vocabulary resulting from the intrinsic randomness in visual word generation. To mitigate it, a straightforward solution is to collect an ensemble of independent random vocabularies, similar to the idea in ERC-Forest [7].

2.3 RLSV Construction

The algorithmic pipeline for RLSV can be described in three consequent steps as follows:

Step-1: visual word generation Assume the similarity between any two samples $p, q \in \mathbb{R}^d$ can be measured by $\kappa(p, q)$. Previous studies (see [18] for a brief survey) reveal the existence of LSH families for many well-known metrics or similarities such as ℓ_p. Formally, let $\mathcal{H}$ be an LSH family such that for any hash function $h \in \mathcal{H} : \mathbb{R}^d \to \{0, 1\}$, the locality-sensitive property holds. Suppose B hash functions are independently generated from $\mathcal{H}$, obtaining $H = \langle h_1, h_2, \ldots, h_B \rangle$ after direct concatenation. We proceed to give a formal definition for "random visual word" as below:

Definition 1. *(**random visual word**): there is a bi-mapping between any visual word w_i and valid permutation π_i from $\{0, 1\}^B$. Any two samples $p, q \in \mathbb{R}^d$ belong to the same visual word if for any $i \in \{1, \ldots, B\}$, there is $h_i(p) \oplus h_j(q) = 0$, where $\oplus$ denotes the XOR bit operation.*

In ideal case, B hash functions are able to produce at most 2^B unique visual words. However, in practice the evolutionary curve of visual word count

seldom demonstrates such an exponentially growing tendency. The phenomena can be geometrically understood, since it is almost impossible for the hyperplane induced by a hash function intersects with all other polyhedrons produced by other hash functions. Since the relationship between B and the number of valid vocabulary entries cannot be accurately determined, practically we maintain the record of vocabulary sizes and continue adding new hash functions until a pre-defined vocabulary size M is reached.

Step-2: visual word filtering In Fig. 1 we plot the sample counts corresponding to distinct visual words. It can be seen that it roughly follows a power-law

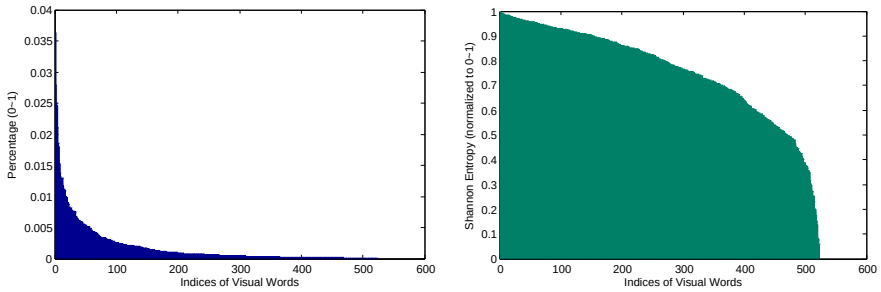

Fig. 1. Illustration of the typical distribution of features (**left**, sorted descendingly) and Shannon entropy of labels (**right**, sorted descendingly) for a 1024-word visual vocabulary in a multi-class problem. Empty words are omitted. Non-informative bins of the vocabulary can be filtered out accordingly by throwing away low-frequency bins.

distribution and thus part of the vocabulary can be trimmed without notable information loss. Moreover, in the supervised (or semi-supervised) settings (discussed in Sec-3.2), we can also abandon the visual words that have weak discriminating power. For multi-class problems, useful statistics can be calculated based on the entropy of class-label distribution for each valid visual word. In Fig. 1 we plot the entropy distribution on the data set of Caltech-101 (vocabulary size M = 1000). In the above two cases, a simple threshold-based visual word filtering will benefit outlier removal and yield more compact representations.

Step-3: histogram binning and normalization In practice, we maintain L independent random vocabularies to ensure the stability and enhance the performance. After vocabulary construction, each feature bag can be transformed into uniform histogram representations by casting its elements into visual words and then perform counting and normalization. For each feature, the binary hash bits $H = \langle h_1, h_2, \ldots, h_B \rangle$ determine a unique decimal value in $[0, 2^B]$. Recall that there are actually no valid visual words corresponding to most decimal values, we maintain a *word-key mapping table* $\mathcal{T}: \{0,1\}^B \rightarrow \{1, \ldots, M\}$ for the purpose of efficient binning.

2.4 Complexity Analysis

Here we provide theoretic comparisons amongst RLSV, ERC-Forest (ERCF), and Hierarchical k-means (HK), in terms of the vocabulary construction time complexity, storage requirements, and the query complexity (i.e., a new vector gets assigned to one of the bins in the vocabulary). Table 1 presents these results in Big-$\mathcal{O}$ notation. Here we assume all tree-structures are with splitting factor of 2, and D for feature dimension, N for total number of available samples, K number of desired cluster centers. For simplicity, we further assume $K = 2^d$, where $d+1$ will be the tree depth for binary splitting trees as we have assumed. Note that c is an undetermined constant in $(1, 2)$, accounting for the empty

Table 1. Comparison of time and space complexity of different methods.

Algorithm	Construction Time	Space Complexity per Word	Query Complexity
RLSV	$\mathcal{O}(D \log_c K)$	$\mathcal{O}(\frac{\log_c K}{K} D)$	$\mathcal{O}(D \log_c K)$
ERCF	$\mathcal{O}(\sqrt{D} N \log_2 K)$	$\mathcal{O}(\frac{K-1}{K} D)$	$\mathcal{O}(\log_2 K)$
HK	$\mathcal{O}(2DN \log_2 K)$	$\mathcal{O}(\frac{2(K-1)}{K} D)$	$\mathcal{O}(2D \log_2 K)$

buckets that have been generated and trimmed after random projections. For the time complexity, RLSV is independent of N, so it can scale up nicely even when the data set is huge. For the storage requirement, KLSH approaches 0 for space per word as K goes up, whereas the other two methods remain constant for large K. It is unfortunate that the query complexity of RLSV is not as low as the ERCF, which could be hurt for very high-dimensional data.

3 Extensions

The algorithm presented in subsection 2.3 targets unsupervised cases in finite-dimensional linear feature spaces. However, both kernel tricks and supervision information are ubiquitous in computer vision databases. For the former, the pairwise similarity is gauged via the inner product in *reproducing kernel Hilbert space* (RKHS) [19]. While for the latter, supervision information from manual labeling or annotations is available to regularize the constructed visual words. Both of them are common scenarios in real-world applications. In this section we discuss the extensions to these cases.

3.1 Kernelized RLSV (K-RLSV)

Note that choice of an LSH family in an application depends on the underlying metric in the feature space. Here we focus on ℓ_p distance when $p = 2$ and Arccos distance. Recall that in both cases LSH is feasible based on sampling from standard Gaussian distribution which belongs to the p-stable distribution. Generally, sampling in RKHS is difficult owing to the lack of explicit feature representation. However, we have the following observation (similar to the theoretic results in [14] yet no zero-centered assumption on the Gram matrices

here), which reveals that sampling from Gaussian distribution in Hilbert space is feasible:

Theorem 1. ***(ℓ_2-keeping projection in RKHS)*** *Denote $\kappa(\cdot,\cdot)$ as the inner product in Hilbert space $\mathcal{K}$. Given an m-cardinality data set $\mathcal{X}$ and corresponding Gram matrix G, the ℓ_2-metric keeping projection can be expressed as $p(x) = \sum_{i=1}^{m} \omega(i)\kappa(x, x_i)$, where $\omega(i)$ only relies on G.*

Proof. Denote the implicit Hilbert mapping function as ψ. The geometric mean can be computed as $\mu_\psi = \frac{1}{m}\sum_{i=1}^{m}\psi(x_i)$. For a t-cardinality subset $\mathcal{S} \subset \{1 \dots m\}$, let $z = \frac{1}{t}\sum_{i\in S}\psi(x_i)$ and $\widetilde{z} = \sqrt{t}(z-\mu_\psi)$. According to the central limit theorem, *$\widetilde{z}$ is distributed as Gaussian $\Phi(0, \Sigma)$, where Σ is the covariance matrix of $\mathcal{X}$. Further applying a whitening transform, we can obtain the desired hash vector in $\mathcal{K}$, i.e. $r = \Sigma^{1/2}\widetilde{z}$. For any datum x, $h(x) = \psi(x)^\top \Sigma^{1/2}\widetilde{z}$.*

Given Gram matrix $G = \Psi^\top\Psi$, where each column of Ψ corresponds to a feature vector in data set $\mathcal{X}$. It is easily verified that

$$\widetilde{z}^\top \Sigma^{1/2}\psi(x) = \widetilde{z}^\top(\Psi Q)(QGQ)^{-\frac{1}{2}}(\Psi Q)^\top\psi(x) \tag{4}$$

where $Q = I - \frac{1}{m}ee^\top$. Substituting $\widetilde{z} = \sqrt{t}\Psi(\frac{1}{t}\delta_S - \frac{1}{m}e)^\top$, where δ_S is a binary indicator vector for subset $\mathcal{S}$. Finally we get

$$p(x) = \left[\sqrt{t}(\frac{1}{t}\delta_S - \frac{1}{m}e)GQ(QGQ)^{-\frac{1}{2}}Q^\top\right]\Psi^\top\psi(x) \tag{5}$$

Let $\omega \triangleq \left[\sqrt{t}(\frac{1}{t}\delta_S - \frac{1}{m}e)GQ(QGQ)^{-\frac{1}{2}}Q^\top\right]$, thus the conclusion holds. □

The complexity of the above procedure is low since $m \ll n$ where n is the sample count of the whole database (e.g., $m = 200 \sim 1000$ and n is probably on the order of million or billion). From the property of p-stable distribution, for samples x, y, projection difference $|p(x)-p(y)| = \left|\sum_{i=1}^{m}\omega(i)\kappa(x,x_i) - \sum_{i=1}^{m}\omega(i)\kappa(y,x_i)\right|$ sustains their distance in the implicit RKHS induced by $\kappa(\cdot,\cdot)$, which makes the LSH algorithms mentioned in Section 2.1 feasible.

3.2 Discriminative RLSV (D-RLSV)

Denote the ensemble of all feature bags as $\mathcal{B} = \{b_i\}$, where each bag $b_i = \{x_{i_1}, \dots, x_{i_n}\}$. In the supervised case, a unique label y_i is assigned to bag b_i. For tractability, we adopt the same assumption to [7], i.e., assuming all features in a bag share the same label. Recall that the scheme in subsection 2.3 is totally random. A possible improvement is to sequentially select an optimal hashing vectors from a candidate pool according to pre-specified label-oriented criterion, which motivates the Discriminative RLSV (D-RLSV) here. The proposed method works as follows: suppose k hashing vectors have been generated and denote the resulting visual words as $V_k = \{w_i, i = 1 \dots n_k\}$ (n_k is the vocabulary

size). To choose the $(k+1)$-th hashing vector, for each candidate $\widetilde{h}$ we calculate a score based on the Shannon entropy as suggested in [20], defined as:

$$S_{k+1}(\widetilde{h}) = \frac{1}{n_k} \sum_{i=1}^{n_k} \frac{2 \cdot I_i(\widetilde{h})}{H_C(\omega_i) + H_S(\widetilde{h}, \omega_i)}, \tag{6}$$

where $H_i^C(\widetilde{h})$ denotes the entropy of the class distribution in the i-th visual word. Note that adopting $\widetilde{h}$ will split each of the existing visual words into two. $H_i^T(\widetilde{h})$ describes the split entropy of the i-th visual word. Formally,

$$H_C(\omega_i) = -\sum_{l \in \mathcal{L}} \frac{n_l}{n} \log_2 \frac{n_l}{n}, \quad \text{and} \quad H_S(\widetilde{h}, \omega_i) = -\sum_{p=1}^{2} \frac{n_p}{n} \log_2 \frac{n_p}{n}. \tag{7}$$

The maximum of $H_i^T(\widetilde{h})$ is reached when the two partitions have equal size. Based on the entropy of a given visual word, the impurity can be calculated by the mutual information of the split, i.e.,

$$I_i(\widetilde{h}) = H_C(\omega_i) - \frac{n_+}{n} H_C(\omega_i^+) - \frac{n_-}{n} H_C(\omega_i^-), \tag{8}$$

where ω_i^+, ω_i^- are the split of ω_i by $\widetilde{h}$, and n_+, n_- denote the number of features belonging to each new visual word respectively. Finally, the optimal hashing vector for the $(k+1)$-th iteration can be determined via $h^* = \arg\max_{\widetilde{h}} S_{k+1}(\widetilde{h})$.

4 Evaluation

In this section, we evaluate the proposed RLSV and its extensions on real-world data sets under three different task settings, i.e., action recognition in video, object recognition, and near-duplicate video detection. Our main concerns include: 1) time used to construct visual vocabularies. 2) memory storage used to keep vocabulary-related information. 3) performance in terms of accuracy. All the experiments are conducted on our common PC with Due-core 3.0Ghz CPU and 8GB physical memory. We choose two representative methods, i.e., Hierarchical K-means and ERC-Forest [7] for comparison. For the former, we adopt a tree branching factor of 2. All statistics are obtained by averaging multiple independent runs.

Experiment-1: KTH Video Database

The KTH video database was developed by Schuldt et al. [21] in 2004 and is one of the popular benchmarks for human action recognition. It contains six different actions captured with appearance variations and mild camera motions such as zooming in and zooming out. The actions are performed by 25 subjects in 4 different scenarios. Each video clips are segmented into 4 sub-clips, resulting in 2400 video sequences in total. See Fig. 2 for the illustration of KTH video clips.

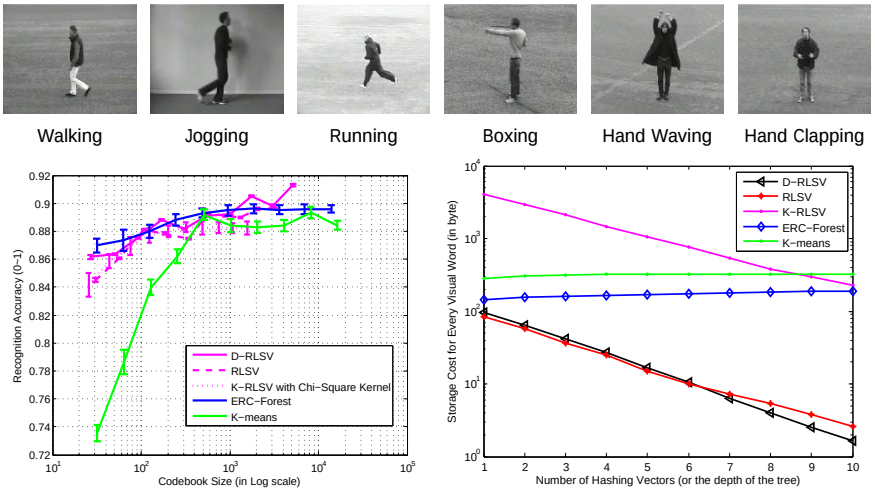

Fig. 2. Top: Example frames from KTH video database. **Bottom Left**: Averaged recognition rates. **Bottom Right**: Storage cost for each visual word. RLSV can consistently achieve comparable performance (with little deviation) with ERC, while consuming less memory. By comparison, k-means is worse in both performance and space efficiency. Please refer to the color pdf for better view.

We use the same dataset splitting as in [21], which contains a training set (8 persons), a validation set (8 persons) and a test set (9 persons). For local feature descriptors, we describe each video segment using Space-time interest points (STIP) as in [21], around which *histogram of gradient* (HOG) and *histogram of flow* (HOF) features are extracted. Both of the counts of independent vocabularies in RLSV or trees in ERC-Forest are set to be 50. Recognition accuracies are presented on the bottom left of Fig. 2. RLSV algorithm family demonstrates comparable discriminating ability to ERC-Forest, but both are obviously better than K-means. D-RLSV reaches a peek performance of 91.4% with around 4000 visual words, which is in sync with the setting and results reported in [22]. In the bottom right of Fig. 2, we illustrate the averaged memory storage for the vocabulary-related information per visual word, which well validates our previous complexity analysis.

Experiment-2: Caltech-101

Caltech-101 is constructed to test object recognition algorithms for semantic categories of images. The data set contains 101 object categories and 1 background category, with 40 to 800 images per category. As pre-processing, the maximum dimension of each image is normalized to be 480-pixel. Most objects contained in the images are of similar scale and orientation. However, considering the large inter-category variation on appearance, lighting and occlusion, the recognition

task on Caltech-101 is still challenging and well suitable to testing various local features and visual vocabularies.

For fair comparison between different vocabularies, we guarantee that they share the same type of features, and any post-processing on them. Specifically, we extract 3000 SIFT descriptors from each image. Unlike the traditional dense sampling strategy on a uniform 2-D image grid, we determine both the locations and scales of each SIFT feature in a random way [23], ignoring more complex and effective sampling schemes such as [24]. However, the experimental results for such a simple scheme are amazingly good, as shown in Fig. 3.

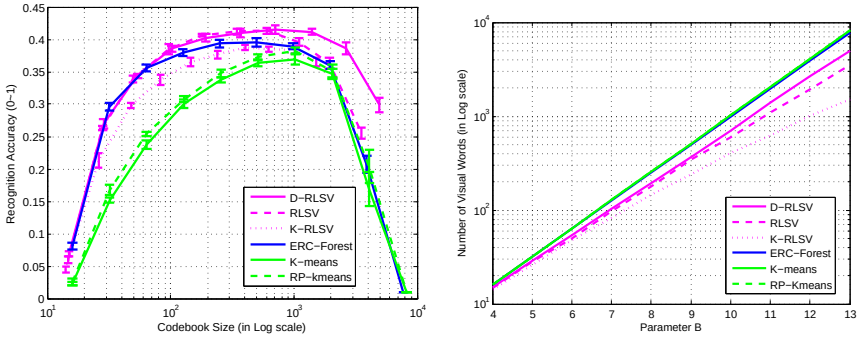

Fig. 3. Left: Performance with 15 training samples per category. Our RLSV family shows consistently (with little deviation) comparable or even better performance than the other methods. **Right**: Vocabulary sizes under varying number of hashing vectors in RLSV or tree depth in ERC-Forest. Please refer to the color pdf for better view.

We test the proposed method in two settings, either with 15 or 30 training samples per category. Here we only present the former due to the space constraint and also the observation that the latter case concurs with the former in terms of algorithmic behaviors as compared to other methods. The peak performance (41.4% for 15 training samples per class) appears with roughly 700 visual words. The performance drastically decreases with extremely smaller or larger vocabularies, which is consistent with previous study in computer vision and our experimental settings. For parameter setting, we use 20 independent visual vocabularies for RLSV and its extensions, and 20 trees for the ERC-Forest. In the classification stage, we regress the histogram feature of the testing sample on the column space spanned by all the training samples in each category, and measure the distance with the regression residue. The overall distance is computed as the summation over individual visual vocabularies or random trees. Classification methods like SVM or NN produce similar yet slightly worse results. As seen in Fig. 3, the accuracies produced by RLSV-related methods and ERC-Forest are comparable, and all are superior to hierarchial K-means. Moreover, it is also observed that the vocabulary sizes roughly linearly increase with respect to the number of hashing vectors in RLSV and tree depth in ERC-Forest or K-means,

although the increasing rate of RLSV-related methods are much smaller than others.

An interesting comparison is between k-means and random projection (into a 60-dimensional lower space) followed by k-means (RP-kmeans). The random projection is meant to be a lightweight replacement of the PCA for dimensionality reduction as suggested in [3]. And the simple scheme consistently outperforms the simple k-means for all vocabulary sizes. This can be partially explained by the property of reduced eccentricity exhibited by the random projection discussed in [3]. Nevertheless, there are still significant performance gaps between RP-kmeans and RLSV or ERC-Forest methods. Moreover, there is no fundamental change in the time and space complexity.

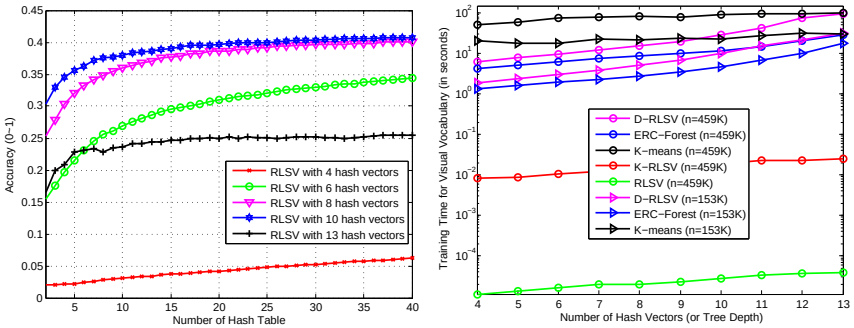

Fig. 4. Left: Performance evolution curve with respect to the number of hash tables in RLSV. **Right**: Time spent to construct visual vocabularies for various approaches.

A random vocabulary as in RLSV or ERC-Forest is very easy to generate, but yet a single one typically performs inferiorly even as compared with k-means in the classification phase. However, we argue that the ensemble of independent "weak" vocabularies significantly outperforms a single elaborately-designed vocabulary, in the same spirit to the bagging algorithm developed in machine learning. We plot the performance evolution curve w.r.t. the number of hashing vectors for RLSV on the left of Fig. 4. As can been seen, the performance of RLSV ensemble hikes rapidly and overruns k-means with only a small number of weak vocabularies.

Fig. 4 (right) also provides an illustration of the time cost for vocabulary construction consumed by various methods. Time spent on vocabulary construction roughly follows a log-linear rule. Among them, RLSV and K-RLSV are order-of-magnitude faster as compared to others.

Experiment-3: Near-Duplicate Video Detection

We also validate the proposed RLSV method for detecting near-duplicate video clips. Being intrinsically unsupervised, ERC-Forest and D-RLSV are not suitable. For such applications, RLSV beats K-means owing to its fast speed and

high flexibility. The near duplicate video benchmark is provided by Wu [25]. The benchmark comprises 12876 video clips, divided into 24 sets. The groundtruth of each video set is manually labeled, and the most relevant video is treated as the query video. The task is to distinguish near-duplicate videos to the query among each video set.

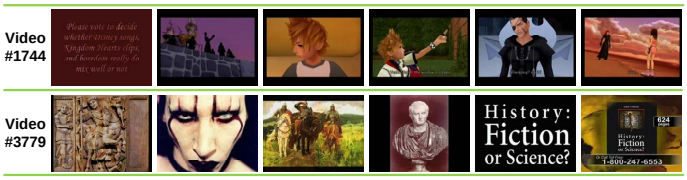

Fig. 5. Selected video clips from Wu's near-duplicate detection database.

Table 2. Comparison of different approaches on near-duplicate video detection.

	K-means	RLSV	K-LSV
Mean Average Precision (MAP)	0.9280	0.9411	0.9442
Time for vocabulary construction (in second)	7.76	2.29×10^{-4}	1.53
Time for code generation (in second)	1.85×10^{-2}	3.4×10^{-3}	2.09×10^{-2}

We adopt a key frame based approach. Each video is first segmented and one key frame is taken from a segment, resulting in 30 key frames per video clip on average. We extract a simple HSV 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 [25]. As seen in Figure 5, the HSV histogram can greatly vary among different key frames, thus the bag-of-feature model well fits this scenario. We test the Mean Average Precision (MAP) over all 24 queries, and together provide a comparison between time spent on vocabulary construction and code generation of each video clip. A K-RLSV method is also included in the experiment, where the Chi-Square kernel is applied. Table 2 indicates RLSV is a good tradeoff between MAP and speed.

5 Conclusion

We have presented a simple yet effective algorithm for visual vocabulary construction combining the idea of LSH and RF. It avoids the severe problems occurring in most existing methods, such as the slow training (e.g., in K-means), or huge storage (e.g., in ERC-Forest). The proposed method strikes a good balance between accuracy and efficacy, and is supposed to be applicable to many real-world applications in computer vision. Moreover, extensions to kernerlized and supervised cases are also presented. We plan to extend the work to online settings. Moreover, currently our method is not advantageous in terms of query complexity. This motivates us to investigate the possibility of synergizing construction and query process towards higher efficiency.

References

1. Lowe, D.: Distinctive image features from scale-invariant keypoints. International journal of computer vision **60** (2004) 91–110
2. Beyer, K.S., Goldstein, J., Ramakrishnan, R., Shaft, U.: When is "nearest neighbor" meaningful? In: ICDT. (1999) 217–235
3. Dasgupta, S.: Experiments with random projection. In: UAI. (2000) 143–151
4. Jurie, F., Triggs, B.: Creating efficient codebooks for visual recognition. In: ICCV. (2005) 604–610
5. Nistér, D., Stewénius, H.: Scalable recognition with a vocabulary tree. In: CVPR (2). (2006) 2161–2168
6. Breiman, L.: Random forests. Machine Learning **45** (2001) 5–32
7. Moosmann, F., Nowak, E., Jurie, F.: Randomized clustering forests for image classification. IEEE Trans. Pattern Anal. Mach. Intell. **30** (2008) 1632–1646
8. Yang, L., Jin, R., Sukthankar, R., Jurie, F.: Unifying discriminative visual codebook generation with classifier training for object category recognition. In: CVPR. (2008)
9. Lee, H., Battle, A., Raina, R., Ng, A.: Efficient sparse coding algorithms. NIPS **19** (2007) 801
10. Wright, J., Yang, A., Ganesh, A., Sastry, S., Ma, Y.: Robust face recognition via sparse representation. PAMI (2009) 210–227
11. Cormen, T., Leiserson, C., Rivest, R., Stein, C.: Introduction to algorithms. The MIT press (2001)
12. Charikar, M.: Similarity estimation techniques from rounding algorithms. In: STOC. (2002) 380–388
13. Andoni, A., Indyk, P.: Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. Commun. ACM **51** (2008) 117–122
14. Kulis, B., Grauman, K.: Kernelized locality-sensitive hashing for scalable image search. In: ICCV. (2009)
15. Bentley, J.L.: Multidimensional binary search trees used for associative searching. Commun. ACM **18** (1975) 509–517
16. Datar, M., Immorlica, N., Indyk, P., Mirrokni, V.: Locality-sensitive hashing scheme based on p-stable distributions. In: SCG. (2004) 253–262
17. Indyk, P., Motwani, R.: Approximate nearest neighbors: Towards removing the curse of dimensionality. In: STOC. (1998) 604–613
18. Andoni, A., Indyk, P.: Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. Commun. ACM **51** (2008) 117–122
19. Scholkopf, B., Smola, A.J.: Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond. MIT Press (2001)
20. Geurts, P., Ernst, D., Wehenkel, L.: Extremely randomized trees. Machine Learning **63** (2006) 3–42
21. Schuldt, C., Laptev, I., Caputo, B.: Recognizing human actions: a local SVM approach. In: ICPR. (2004)
22. Laptev, I., Marszałek, M., Schmid, C., Rozenfeld, B.: Learning realistic human actions from movies. (2008)
23. Nowak, E., Jurie, F., Triggs, B.: Sampling strategies for bag-of-features image classification. In: ECCV (4). (2006) 490–503
24. Marée, R., Geurts, P., Piater, J.H., Wehenkel, L.: Random subwindows for robust image classification. In: CVPR (1). (2005) 34–40
25. Wu, X., Hauptmann, A.G., Ngo, C.W.: Practical elimination of near-duplicates from web video search. In: ACM Multimedia. (2007) 218–227