Building Compact Local Pairwise Codebook with Joint Feature Space Clustering

Nobuyuki Morioka[1,3] and Shin'ichi Satoh[2]

[1] The University of New South Wales, Australia
nmorioka@cse.unsw.edu.au
[2] National Institute of Informatics, Japan
satoh@nii.ac.jp
[3] NICTA, Australia

Abstract. This paper presents a simple, yet effective method of building a codebook for pairs of spatially close SIFT descriptors. Integrating such codebook into the popular bag-of-words model encodes local spatial information which otherwise cannot be represented with just individual SIFT descriptors. Many previous pairing techniques first quantize the descriptors to learn a set of visual words before they are actually paired. Our approach contrasts with theirs in that each pair of spatially close descriptors is represented as a data point in a joint feature space first and then clustering is applied to build a codebook called Local Pairwise Codebook (LPC). It is advantageous over the previous approaches in that feature selection over quadratic number of possible pairs of visual words is not required and feature aggregation is implicitly performed to achieve a compact codebook. This is all done in an unsupervised manner. Experimental results on challenging datasets, namely 15 Scenes, 67 Indoors, Caltech-101, Caltech-256 and MSRCv2 demonstrate that LPC outperforms the baselines and performs competitively against the state-of-the-art techniques in scene and object categorization tasks where a large number of categories need to be recognized.

Keywords: bag-of-words, spatial pyramid matching, higher-order spatial features, local pairwise codebook.

1 Introduction

For scene and object categorization tasks, the bag-of-words model has received much attention due to its simplicity and robustness. However, because of its orderless representation of local features, there have been numerous works [7,8,10,11,15,17,18,25] that consider spatial arrangement of the features to discover higher-order structures inherent in scenes and objects for improved performance. Amongst the many, one simple method is to consider pairs of spatially close visual words [8,10,11]. This is commonly done by first obtaining a codebook where each feature descriptor is mapped to a visual word and then representing occurrences of local pairs of visual words with the bag-of-words model.

K. Daniilidis, P. Maragos, N. Paragios (Eds.): ECCV 2010, Part I, LNCS 6311, pp. 692–705, 2010.

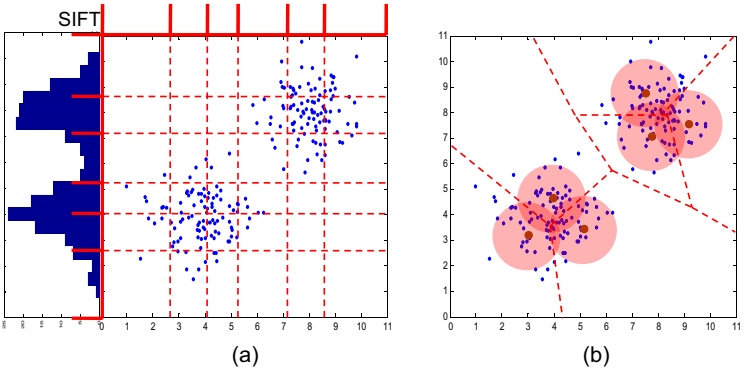

Fig. 1. (a) Traditional pairwise approach: The distribution of SIFT descriptors is viewed as one dimensional data and intervals shown in red indicate the derived clusters. The clusters are then used to determine the partition of the local pairwise distribution which is depicted in the dotted red line. (b) Our pairwise approach: The underlying local pairwise distribution is partitioned directly by joint feature space clustering to achieve a compact local pairwise codebook.

However, we speculate several issues with building the codebook prior to the pairing process. *First*, the number of possible pairs of visual words grows in quadratic with respect to the codebook size. This is depicted in Fig. 1 (a) where high dimensional descriptors like SIFT [12] are seen as one dimensional data and plotted along the x and y axes. Such quadratic growth results in a high dimensional histogram representation where in the case of a few training samples available, a classifier may over-fit and fail to generalize over testing data. *Second*, while feature selection [8,11] can be applied on these pairs of visual words to avoid the over-fitting problem, it often requires additional information like class labels and does not always lead to performance improvement [8]. *Third*, as illustrated in Fig. 1 (a), the traditional pairing approaches ignore the underlying distribution of pairs of nearby local feature descriptors, as they have already determined the partition of such distribution from the single feature codebook. We suspect that this way of over-partitioning the distribution may lead to poor generalization - degrading the recognition as previously seen in [8].

Given the above-mentioned issues, this paper describes a different approach, rather a reversed approach, in discovering a compact set of local pairwise feature clusters called *Local Pairwise Codebook* (LPC). We first concatenate each pair of spatially close descriptors and treat it as a data point in a joint feature space. Then, clustering is applied on the data to build a codebook as depicted in Fig. 1 (b). Our approach considers the underlying distribution explicitly, thus the partitioning is data-driven. In contrast to the traditional approaches where the number of pairwise feature clusters is determined by the single feature codebook, the size of LPC is controlled directly by a clustering algorithm and can be set to a moderate size. Thus, feature selection is not required to achieve a compact

codebook which is significantly smaller than quadratic number of possible pairs of visual words formed from a moderate-sized single feature codebook. With LPC, we directly encode local structure information into the bag-of-words model to boost the recognition in categorization tasks.

In summary, there are three main contributions in this paper. *Firstly*, we perform joint feature space clustering to build a compact local pairwise codebook based on SIFT descriptors to overcome issues discussed above. *Secondly*, when assigning the nearest cluster to each pair of nearby descriptors in an image, its time complexity grows in the number of pairs formed. Thus, to significantly reduce the complexity, we propose an efficient pairwise cluster assignment technique. *Thirdly*, we combine LPC with spatial pyramid matching kernel [9] to demonstrate that local and global spatial information complement each other to achieve competitive results on two scenes and three object categorization datasets, namely 15 Scenes [9], 67 Indoors [16], Caltech-101 [3], Caltech-256 [6] and MSRCv2 [22] datasets.

This paper is organized as follows. In Sect. 2, we cover some of the related work. Then Sect. 3 presents how to learn and use LPC in detail. This is followed by the experimental results in Sect. 4 to demonstrate the effectiveness of LPC. Finally, Sect. 5 concludes our paper with possible future work.

2 Related Work

The idea of simply pairing up local features within their spatial local neighborhood is not new. While Lazebnik et. al. [8] have attempted to categorize objects by pairing visual words of SIFT descriptors in their maximum entropy framework, their results have shown that the pairwise features with frequency-based feature selection do not necessarily improve the performance over just using the visual words. Similar results are obtained by Lan et. al. [7]. Interestingly, both [7] and [8] have commented that higher-order spatial features might be more useful in locating and segmenting out objects of interest.

In contrast, Liu et. al. [11] have proposed an efficient feature selection method based on boosting which progressively mines higher-order spatial features. This has resulted in performance improvement up to 2^{nd} order features, i.e. pairs of visual words. Savarese et. al. [17] have presented correlaton which captures the distribution of the distances between pairs of visual words based on clustering a set of correlogram elements. When this is combined with the appearance based bag-of-words model, it achieves promising results on some challenging object class recognition datasets. Instead of using actual spatial distance between local features, Proximity Distribution Kernel [10] and its variant [21] have reduced such information into ranks to achieve scale-invariance in their representation. Quak et. al. [15] have used frequent itemset mining to discover a set of distinctive spatial configurations of visual words to learn different object categories. In addition, such higher-order spatial features have shown to be also effective in unsupervised object discovery [18] and image retrieval [2,19].

All the above-mentioned techniques thus far assume that a set of visual words is already learned before considering pairs of features to represent higher-order

spatial information - indicating the issues discussed in the previous section. Our work is different from the above techniques in that we apply quantization in the joint feature space of local pairs of descriptors. A compact codebook of such pairs can be built by discovering clusters that encode correlation between spatially close descriptors. We believe that this captures better generalization of local pairwise feature distribution. Furthermore, it is possible to combine our work with the above techniques such as modeling the distribution of the distances between descriptors. From one perspective, our work can be seen as an extension of spatial pyramid matching kernel proposed by Lazebnik et. al. [9], as both approaches are based on densely sampled feature descriptors, but we consider local pairs of such descriptors to learn a more discriminative codebook.

3 Local Pairwise Codebook

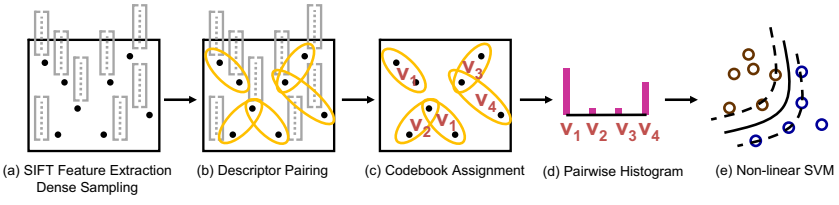

Fig. 2. An overview of our approach. (a) Features are densely sampled and described by SIFT. (b) Features are paired locally and descriptors are concatenated. (c) A codeword is assigned to each local pair of features. (d) A histogram of pairwise codewords are established per image. (e) Non-linear SVM is used for image classification.

3.1 Baseline: Pairing Visual Words

Before describing our approach, we first outline how pairs of visual words are usually formed in the previous approaches. Note that we only model occurrences of pairs of visual words within predefined neighborhood denoted as γ. In other words, other spatial information such as direction and distance between the pairs is not explicitly modeled. Each key-point f_i in an image is encoded as $(x_i, y_i, \theta_i, \sigma_i, \mathbf{d}_i)$ representing x and y coordinates, dominant orientation, scale and SIFT descriptor respectively. Similar to [9], features are densely sampled from single scale and are rotationally variant, thus θ and σ are ignored. From a randomly sampled set of the descriptors, a codebook of size K, $\{c_k\}_K$ is learned using K-means clustering. Then, every feature descriptor $\mathbf{d}_i$ is mapped to the closest cluster c_k in the feature space measured by Euclidean distance. To form a local pair of visual words, two visual words must be within γ pixels away from each other spatially and such pair is represented as:

$$f_{(i,j)} = \left(\frac{(x_i + x_j)}{2}, \frac{(y_i + y_j)}{2}, (c_i, c_j) \right) \tag{1}$$

The ordering of i and j is determined by $c_i \leq c_j$ which is just a comparison between cluster indices. Given a set of visual word pairs formed for an image I, we establish a histogram H_I where each bin stores the occurrence of a particular pair of visual words (c_i, c_j). Once histograms are generated for all training images, a kernel is computed. This is further discussed in Sect. 3.4.

Given K number of clusters, there are $K \times (K+1)/2$ number of pairwise feature clusters which can potentially be large even if the size of K is moderate. In the case of [8], they set K to be 1000, so the number of possible pairs is around 500K. While feature selection [8,11] can be used to reduce the number of such clusters, it is often done in a supervised manner. This precisely motivates our work on Local Pairwise Codebook (LPC) as we learn a compact set of pairwise feature clusters in an unsupervised manner and its codebook size is not governed by the number of single feature clusters. Although supervised feature selection may of course be applied to LPC to further increase discrimination if possible, we will not discuss this in the paper. We introduce LPC in the next section.

3.2 Pairing Spatially Close Feature Descriptors

As described previously, each key-point f_i in an image is represented as a tuple of $(x_i, y_i, \mathbf{d}_i)$. In LPC, we first pair up features which are within γ pixels away from each other. Each of such pairs, also referred to as local pairwise feature, is encoded as follows:

$$f_{(i,j)} = \left(\frac{(x_i + x_j)}{2}, \frac{(y_i + y_j)}{2}, [\mathbf{d}_i \mathbf{d}_j] \right) \tag{2}$$

The ordering of i and j can be achieved by any total order to ensure order invariance when descriptors are paired. In the case of our work, we determine such ordering by comparing the values of the first non-equal dimension of two descriptors $\mathbf{d}_i$ and $\mathbf{d}_j$ encountered. For this ordering to be robust against noise, we have applied discretization on each descriptor by $\lceil \mathbf{d}_i / \delta \rceil$ where δ is a small constant. Once the descriptors are concatenated as $\mathbf{d}_{ij}$, it can be viewed as a data point in the joint feature space. We use $[\mathbf{d}_i \mathbf{d}_j]$ and $\mathbf{d}_{ij}$ interchangeably. After the descriptor pairing process, K-means is applied on a random collection of local pairwise feature descriptors to learn a codebook $C = \{c_k\}_K$ where K denotes the codebook size. Then, for every image, a histogram where each bin represents occurrences of one local pairwise codeword c_k is established. An illustrated example of how our approach works is given in Fig. 2.

The construction of LPC is done independently from the single feature codebook. Thus, the size of LPC is directly controlled by K-means and feature selection is not required if K is set to a moderate size. Since the distribution of the local pairwise features is considered explicitly during clustering, the clusters are likely to capture local structure information, specifically correlation between spatially close descriptors. Therefore, compared to quadratic growth in the number of pairwise feature clusters, LPC is a *relatively compact* and *general* codebook that can be learned unsupervised.

3.3 Efficient Pairwise Cluster Assignment

With LPC, cluster assignment is done on a paired descriptor $\mathbf{d}_{ij}$ instead of $\mathbf{d}_i$ and $\mathbf{d}_j$ individually. Given N descriptors of D dimension in an image, if P number of pairs are formed per descriptor on average, then the time complexity of finding the nearest clusters for $N \times P$ paired descriptors by naïvely computing their Euclidean distances to K clusters would be $O(PDNK)$. When compared to using the single feature codebook, it grows linear in P. While hierarchical K-means [13] or K-means with approximate nearest neighbor search [14] might be an attractive solution to efficiently look for the nearest cluster, they are designed to reduce $O(K)$ to $O(log(K))$ with different approximations and are more suitable in image retrieval tasks where a large codebook is required to increase discrimination. Here, we outline an exact and efficient nearest neighbor search algorithm specific to LPC. It is based on the observation that data redundancy is present in the pairs of descriptors formed within an image. The resulting algorithm decouples P and D and its time complexity becomes $O((P + D)NK)$. Thus, with high dimensional descriptors like SIFT and a moderate number of pairs formed (e.g. 50), the efficiency of LPC is greatly improved.

In order to search for the nearest cluster of a paired descriptor $\mathbf{d}_{ij}$, we need to compute its squared Euclidean distance to each cluster $\mathbf{c}_k$ and find the one with the minimum distance. Since each pairwise feature cluster is an exemplar of local pairwise features, its vector $\mathbf{c}_k$ can be seen as a concatenation of two single feature descriptor exemplars $[\mathbf{c}_{k1}\mathbf{c}_{k2}]$. Thus, the squared distance between $\mathbf{c}_k$ and $\mathbf{d}_{ij}$ can be rewritten as a sum of two squared distances given in (4). Since both $\mathbf{d}_i$ and $\mathbf{d}_j$ are likely to be paired again with some other descriptors within their local neighborhood, we can cache $\|\mathbf{c}_{k1} - \mathbf{d}_i\|^2$ and $\|\mathbf{c}_{k2} - \mathbf{d}_j\|^2$ into matrices Q^1 and Q^2 respectively so to avoid recomputing these partial distances as stated in (5). Both matrices have the size of $K \times N$ and are indexed by cluster and descriptor indices. Thus, if we have the two matrices calculated prior to the pairwise feature cluster assignment, then the distance calculation for each pair becomes just a sum of two values in the matrices.

$$\|\mathbf{c}_k - \mathbf{d}_{ij}\|^2 = \|[\mathbf{c}_{k1}\mathbf{c}_{k2}] - [\mathbf{d}_i\mathbf{d}_j]\|^2 \tag{3}$$

$$= \|\mathbf{c}_{k1} - \mathbf{d}_i\|^2 + \|\mathbf{c}_{k2} - \mathbf{d}_j\|^2 \tag{4}$$

$$= Q^1_{k1,i} + Q^2_{k2,j} \tag{5}$$

Computation of the matrices Q^1 and Q^2 takes $O(DNK)$ time as distances between N descriptors and $2K$ single feature descriptor exemplars are calculated. Then, we have $N \times P$ descriptor pairs to look up the distances to K clusters to find the closest one and this takes $O(PNK)$ time. By combining the two operations, we get the time complexity of $O((P + D)NK)$. In the case of our experiments, we have used $P = 48$ ($\gamma = 24$) and $D = 32$ by default and the efficiency of LPC has increased by 20 folds using this proposed technique.

3.4 Computing Kernel

Once a codebook is learned, a histogram or its variant is established for each image to compute either histogram intersection kernel (HIK) or spatial pyramid matching kernel (SPMK) [9] for classification. For a pair of ℓ_1 normalized histograms H_X and H_Y representing images X and Y respectively, HIK is computed as given in (6) where i denotes a histogram bin index.

$$K_{HIK}(H_X, H_Y) = \sum_{i=1}^{K} min(H_X(i), H_Y(i)) \tag{6}$$

For SPMK, each image is partitioned into $M \times M$ equal-sized regions over multiple levels of resolution. At each level l, by defining the resolution M to be 2^l, $M \times M$ histograms are established and concatenated to produce H^l. Constructing such histograms essentially captures global spatial information at different granularity. Finally, (7) computes the similarity between two histograms H_X and H_Y where we set the maximum level L to be 2.

$$K^L_{SPM}(H_X, H_Y) = \frac{1}{2^L} K_{HIK}(H^0_X, H^0_Y) + \sum_{l=1}^{L} \frac{1}{2^{L-l+1}} K_{HIK}(H^l_X, H^l_Y) \tag{7}$$

4 Experimental Results

This section presents the experimental results of LPC on five challenging datasets. For all datasets, SIFT descriptors (32 dimension, 2×2 grids, 8 orientation bins) are densely sampled at every 8 pixel step and are described from 16×16 patches. In [20], their experimental results have shown that SIFT descriptors with 2×2 grids perform competitively against the one with 4×4 grids. We have also conducted our own experiments on the datasets and verified that the 2×2 performs competitively against the 4×4. By default, the neighborhood threshold γ is set to 24 pixels which forms 48 pairs per descriptor on average. We use the following baselines to compare against our LPC:

1. **QPC**: Quadratic Pairwise Codebook as described in Sect. 3.1. We have evaluated with $K = \{80, 200\}$ resulting in $\{3240, 20100\}$ number of pairwise feature clusters.
2. **QPC+Sel**: Quadratic Pairwise Codebook with frequency-based feature selection outlined in [8]. 1000 most frequently appearing pairwise feature clusters are chosen from each category.
3. **Sgl**: A single feature codebook approach. This baseline is our reimplementation of Lazebnik et. al. [9].

In fact, we have also tried other methods to compare against LPC, such as applying supervised feature aggregation [4] to compress QPC. However, due to a large number of categories to be learned and a relatively few training examples available, it has resulted in a performance similar to QPC+Sel. Thus, we report

the results of the above three baselines only. To construct the single feature codebook and LPC for each dataset, 100K and 250K descriptor samples are used respectively. For all experiments, we have repeated eight times with different sets of training and testing images randomly sampled.

4.1 Scene Categorization

For scene categorization, we have used the 15 Scenes [9] and 67 Indoors [16] datasets. 15 Scenes contains images of 15 categories ranging from indoor scenes (e.g. living room and kitchen) to outdoor scenes (e.g. mountain and street). 100 images per category are used for training and the rest is used for testing. It is clear from the results shown in Table 1 that LPC outperforms Sgl and QPC(+Sel) in both HIK and SPMK. Although QPC has performed better than Sgl in the case of HIK, it has failed to do so in SPMK. QPC+Sel has selected 2562 and 4400 feature clusters from $K = 80$ (3240) and $K = 200$ (20100) respectively, but with no improvement observed. While the performance of Sgl reaches its peak when K is 1600 and 400 for HIK and SPMK respectively, LPC continues to improve as we increase K in both settings. For 67 Indoors dataset, the same experimental setup in [16] is used. In general, as similar to the results of 15 Scenes, LPC performs better in both HIK and SPMK settings.

Table 1. Recognition accuracy (%) on 15 Scenes and 67 Indoors. HIK refers to histogram intersection kernel and SPMK refers to spatial matching kernel.

		15 Scenes		67 Indoors	
Methods	K	HIK	SPMK	HIK	SPMK
QPC	80	76.61±0.44	**81.41±0.56**	27.28±1.44	34.83±0.97
	200	**78.25±0.45**	80.93±0.36	30.75±1.38	**36.15±0.95**
QPC+Sel	80	**76.03±0.46**	**81.11±0.55**	26.96±1.76	**34.27±0.93**
	200	74.69±0.60	78.71±0.60	**27.29±0.60**	32.91±0.60
Sgl	400	74.75±0.66	**81.76±0.47**	23.89±0.92	33.82±1.10
	800	75.62±0.58	81.50±0.48	26.11±0.98	34.82±0.83
	1600	**76.67±0.61**	81.40±0.42	27.67±1.03	**35.13±1.07**
	3200	76.13±0.51	80.34±0.37	**28.77±1.85**	34.71±1.13
LPC	800	76.78±0.53	82.50±0.63	27.30±1.44	35.80±0.86
	1600	78.42±0.23	83.07±0.50	29.71±0.70	37.16±1.20
	3200	**79.76±0.38**	**83.40±0.58**	**32.35±0.68**	**38.36±0.63**

4.2 Caltech-101/256 Datasets

The Caltech-101 dataset [3] comprises of 101 different object classes. In our experiments, we have used 30 images per category for training and the rest for testing, excluding Background class. For an efficiency reason, we have downsized images preserving their aspect ratios if their longer sides are greater than 1000 pixels. The results are given in Table 2. Compared with the results of

Table 2. Recognition accuracy (%) on Caltech-101 and Caltech-256

		Caltech-101		Caltech-256	
Methods	K	HIK	SPMK	HIK	SPMK
QPC	80	51.91±1.20	**67.27±1.00**	23.12±0.42	31.50±0.52
	200	**54.68±0.63**	66.04±0.76	**25.57±0.43**	**32.13±0.37**
QPC+Sel	80	**51.87±1.20**	**67.16±0.93**	23.17±0.30	**31.10±0.35**
	200	51.04±0.89	63.34±0.96	**26.18±0.62**	30.48±0.66
Sgl	400	45.40±0.80	**65.38±1.16**	18.67±0.42	29.06±0.26
	800	47.38±0.67	64.45±0.44	19.10±0.49	**29.14±0.38**
	1600	47.74±0.40	62.88±0.83	19.63±0.47	28.44±0.35
	3200	**47.95±0.97**	60.25±1.09	**20.00±0.46**	27.75±0.41
LPC	800	53.06±0.99	69.90±0.48	24.69±0.34	33.93±0.54
	1600	55.50±0.82	70.48±0.73	26.33±0.37	35.08±0.47
	3200	**57.08±0.88**	**71.00±0.48**	**28.11±0.38**	**35.74±0.41**

15 Scenes, the performance boost for LPC over Sgl is much more obvious - improving the results by around 9% and 6% for HIK and SPMK respectively. We suspect that objects tend to have more local structures present than scenes and LPC has exploited these to obtain boost in the recognition. Overall, LPC has outperformed both Sgl and QPC(+Sel) showing similar trends seen in the previous experiment.

The Caltech-256 dataset [6] succeeds the Caltech-101 dataset with several improvements including increased intra-class variability and variability in object pose and location. We have used 30 training and 25 testing images per category, excluding Clutter class. We have down-sized images if their longer sides are greater than 300 pixels. The results are presented in Table 2. The performance increase of LPC over Sgl is akin to the previous results. Interestingly, the best result obtained with LPC HIK ($K = 3200$) is close to Sgl SPMK ($K = 800$). Since the number of bins required for LPC HIK is 3200 and Sgl SPMK is 16800 which is 5 times more, this implies the effectiveness of exploiting local structures over global structure when variability of object pose and location is high. Of course, LPC SPMK, the combination of both local and global information, has shown to perform the best on this dataset as well.

4.3 MSRCv2 Dataset

The MSRCv2 dataset is a relatively small object class database compared to Caltech-101/256, but is considered be much more difficult dataset due to its high intra-class variability [22]. We have simply followed the experimental setup used in [25], except we have used dense sampling instead of an interest point dectector to extract feature descriptors. Nine out of fifteen classes are chosen (i.e. cow, airplanes, faces, cars, bikes, books, signs, sheep and chairs) where each class contains 30 images. For each experiment, we have randomly sampled 15 training and 15 testing images class and no background is removed from the images. We have used HIK in this experiment. LPC has performed better

against the baselines with an accuracy of $83.9 \pm 2.9\%$ with K being 3200. The highest accuracy obtained by each baseline is as follows: QPC ($81.8 \pm 3.4\%$), QPC+Sel ($80.8 \pm 3.4\%$) and Sgl ($81.7 \pm 2.8\%$). Our results are higher than the results reported in [25] where 2^{nd} (pairwise) and 10^{th} order spatial features have obtained accuracies of $78.3 \pm 2.6\%$ and $80.4 \pm 2.5\%$ respectively.

4.4 Comparison with Large Patch SIFT Descriptors

We have experimented the baseline Sgl with three different configurations of SIFT descriptor to be sampled at every 8 pixels, i.e. 24×24 patch with 3×3 grids ($D = 72$), 32×32 patch with 4×4 grids ($D = 128$), 40×40 patch with 5×5 grids ($D = 200$). These try to mimic the effect of LPC with different neighborhood thresholds, i.e. $\gamma = \{8, 16, 24\}$. For an example, with 40×40 patch SIFT descriptors, they potentially cover all possible pairings of 16×16 SIFT descriptors with $\gamma = 24$. The results are presented in Table 3. For all datasets, these large patch SIFT descriptors improve over the Sgl baseline evaluated earlier. However, LPC performs better across all datasets using just 16×16 patch SIFT descriptors, especially, the performance gap is significant for Caltech-256 which is considered to be a hard dataset. Not only this implies the robustness of LPC, it is also relatively more efficient compared to these baselines if time required to do feature extraction, codebook construction and cluster assignment is considered. In fact, it is possible to apply LPC on larger patches as well. For an instance, LPC with 24×24 patch with 2×2 grids ($D = 32$) has achieved $72.15 \pm 0.68\%$ accuracy on Caltech-101.

Table 3. Recognition accuracy (%) of Sgl with large patch SIFT descriptors. Although the performance of Sgl has increased, LPC that uses 16×16 patches still consistently performs better than these baselines.

Methods	K	15 Scenes	67 Indoors	Caltech-101	Caltech-256
Sgl (24x,72D)	800	81.96±0.44	36.56±0.78	**67.34±1.11**	31.32±0.34
	1600	**82.25±0.65**	37.42±1.23	66.87±0.96	**31.50±0.41**
	3200	81.77±0.43	**37.57±0.85**	66.07±0.42	31.14±0.49
Sgl (32x,128D)	800	81.25±0.59	36.04±0.75	**69.18±0.86**	31.57±0.48
	1600	**81.74±0.80**	36.08±0.87	68.82±0.73	**32.26±0.21**
	3200	81.17±0.74	**36.86±0.92**	68.38±0.66	32.22±0.43
Sgl (40x,200D)	800	80.40±0.68	34.23±1.66	68.95±1.26	31.08±0.46
	1600	**80.75±0.48**	**35.38±1.08**	**69.47±0.69**	31.52±0.43
	3200	80.61±0.48	35.08±0.85	69.40±1.00	**31.87±0.25**
LPC	3200	**83.40±0.58**	**38.36±0.63**	**71.00±0.48**	**35.74±0.41**

4.5 Comparison with Different Neighborhood Thresholds γ

In this section, we have also experimented LPC with different neighborhood thresholds $\gamma = \{8, 16, 24, 32\}$ using SPMK. Table 4 shows the results on the first four datasets. Overall, with any γ and codebook size tried, it has improved over

Table 4. The performance comparison with different γ. Each γ shows the average number of pairs formed per descriptor in brackets.

		γ (#pairs)			
Dataset	K	8 (8)	16 (24)	24 (48)	32 (80)
15 Scenes	1600	82.83±0.62	82.92±0.44	83.07±0.50	82.84±0.52
	3200	82.68±0.47	83.18±0.63	**83.40±0.58**	83.28±0.49
67 Indoors	1600	37.29±0.96	37.76±1.43	37.16±1.20	36.81±1.38
	3200	37.46±1.43	**38.64±1.32**	38.36±0.63	37.75±0.67
Caltech-101	1600	68.49±0.91	69.92±0.64	70.48±0.80	70.89±0.84
	3200	67.95±0.76	70.08±0.56	71.00±0.48	**71.05±0.85**
Caltech-256	1600	32.69±0.40	34.28±0.39	35.08±0.47	34.68±0.43
	3200	32.75±0.52	34.80±0.71	**35.74±0.41**	35.57±0.52

Table 5. The performance comparison with different step sizes

		step size		
Dataset	K	4	6	8
15 Scenes	1600	81.31±0.38	81.39±0.47	83.07±0.50
	3200	81.97±0.45	82.00±0.43	**83.40±0.58**
67 Indoors	1600	37.58±1.04	37.98±1.21	37.16±1.20
	3200	38.93±0.48	**39.63±0.69**	38.36±0.63
Caltech-101	1600	72.40±0.79	71.65±0.52	70.48±0.73
	3200	**72.94±0.54**	72.01±0.49	71.00±0.48
Caltech-256	1600	36.85±0.56	36.06±0.54	35.08±0.47
	3200	**37.71±0.47**	37.10±0.46	35.74±0.41

Sgl (c.f. Tables 1 and 2). This shows the benefit of capturing local interaction between feature descriptors. For 15 Scenes, any γ larger than 8 pixels achieve similar performance. On the contrary, with Caltech-101 and Caltech-256, the performance tends to improve as γ is increased for any codebook size tried.

4.6 Comparison with Different Step Sizes

In this section, LPC is evaluated on different step sizes used by dense sampling. The neighborhood threshold γ is set to 24 pixels. We report the results on SPMK in Table 5. For both Caltech-101 and Caltech-256, the performance increases as the step size gets smaller.

4.7 Comparison with Previously Published Results

This section compares LPC with the previously published results which were or are known to be the state-of-the-art for the datasets. For each dataset, we report the highest recognition accuracy of LPC obtained from the previous sections. For the comparison to be fair, we have only included results that are obtained from using a single descriptor or cue. As shown in Table 6, LPC has performed

Table 6. Performance comparison with the state-of-the-art methods based on a single descriptor. Results for a single image scale is reported wherever possible.

Methods	15 Scenes	67 Indoors	Caltech-101	Caltech-256
KC [5]	76.67±0.39	-	64.14±1.18	27.17±0.46
SPM [9]	81.40±0.50	-	64.60±0.80	-
KSPM [6]	-	-	67.40	34.10
NBNN [1]	-	-	70.40	-
HC [23]	82.30±0.49	-	-	-
HG [26]	**85.20**	-	73.10	-
ScSPM [24]	80.28±0.93	-	**73.20±0.54**	34.02±0.35
ROI+Gist[16]	-	>30.00	-	-
LPC	83.40±0.58	**39.63±0.69**	72.94±0.54	**37.71±0.47**

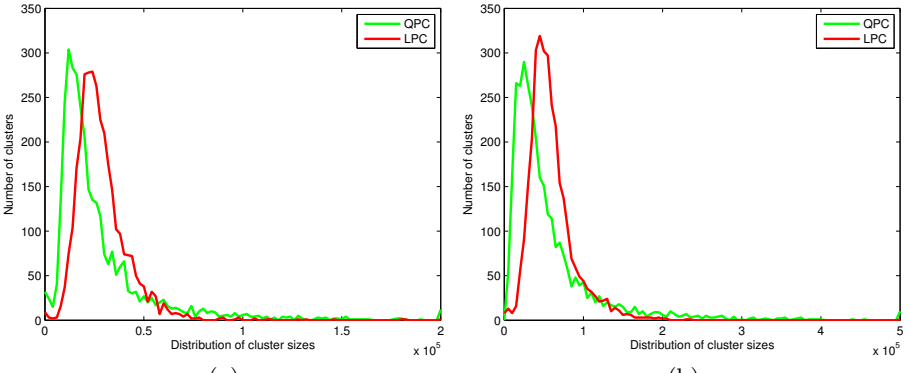

Fig. 3. Distributions of cluster sizes estimated on (a) 15 Scenes and (b) Caltech-101

competitively against other methods across all datasets. For Caltech-256, it has outperformed the state-of-the-art methods.

4.8 On the Distribution of Local Pairwise Features

In this section, we have estimated the distributions of the cluster sizes for the traditional approaches (QPC) and our approach (LPC) using the 15 Scenes and Caltech-101 datasets. To be unbiased, we have used roughly the same number of clusters for both approaches, i.e. $K = 80$ (3240 pairwise clusters) for QPC and $K = 3200$ for LPC. We have extracted pairs of features from all images and used them to plot the distributions. In total, approximately 90 million and 190 million such features are extracted for 15 Scenes and Caltech-101 respectively.

As depicted in Fig. 3, the distributions of QPC plotted in green has an earlier peak than LPC plotted in red. This means that QPC has placed a lot of its clusters in regions where there are not many data points. Also, the distributions are heavy-tailed due to many points assigned to only a few number of clusters. In contrast, the estimated distributions obtained by LPC seem to be tighter than

the former distributions. Based on this observation, we can infer that LPC has explicitly considered the underlying distribution of the local pairwise features and balanced its cluster sizes as much as possible. Therefore, we believe that this observation supports our intuition presented earlier with Fig. 1.

5 Conclusion

In this paper, we have presented a simple, yet effective method of building a compact codebook that encodes local spatial information with joint feature space clustering called Local Pairwise Codebook. For it being a simple method, LPC has outperformed the methods with which we have compared, and performed competitively against the previously published results. Our future work involves building LPC based on interest point detectors instead of the dense sampling strategy used in our experiments as well as incorporating additional spatial information like direction and distance.

Acknowledgments

NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Centre of Excellence program.

References

1. Boiman, O., Shechtman, E., Irani, M.: In defence of Nearest-Neighbor based image classification. In: CVPR (2008)
2. Chum, O., Perd'och, M., Matas, J.: Geometric min-Hashing: Finding a (thick) needle in a haystack. In: CVPR (2009)
3. Fei-Fei, L., Fergus, R., Perona, P.: Learning Generative Visual Models from Few Training Examples: An Incremental Bayesian Approach Tested on 101 Object Categories. In: CVPR Workshop (2004)
4. Fulkerson, B., Vedaldi, A., Soatto, S.: Localizing Objects with Smart Dictionaries. In: Forsyth, D., Torr, P., Zisserman, A. (eds.) ECCV 2008, Part I. LNCS, vol. 5302, pp. 179–192. Springer, Heidelberg (2008)
5. van Gemert, J., Veenman, C., Geusebroek, J.: Visual Word Ambiguity. In: TPAMI (2010)
6. Griffin, G., Holub, A., Perona, P.: Caltech 256 object category dataset. Technical Report UCB/CSD-04-1366, California Institute of Technology (2006)
7. Lan, X., Zitnick, C., Szeliski, R.: Local Bi-gram Model for Object Recognition MSR-TR-2007-54. Technical Report, Microsoft Research (2007)
8. Lazebnik, S., Schmid, C., Ponce, J.: A Maximum Entropy Framework for Part-Based Texture and Object Recognition. In: ICCV (2005)
9. Lazebnik, S., Schmid, C., Ponce, J.: Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories. In: CVPR (2006)
10. Ling, H., Soatto, S.: Proximity Distribution Kernels for Geometric Context in Category Recognition. In: ICCV (2007)

11. Liu, D., Hua, G., Viola, P., Chen, T.: Integrated feature selection and higher-order spatial feature extraction for object categorization. In: CVPR (2008)
12. Lowe, D.: Distinctive Image Features from Scale-Invariant Keypoints. In: IJCV (2004)
13. Nister, D., Stewenius, H.: Scalable Recognition with a Vocabulary Tree. In: CVPR (2006)
14. Philbin, J., Chum, O., Isard, M., Sivic, J., Zisserman, A.: Object retrieval with large vocabularies and fast spatial matching. In: CVPR (2007)
15. Quak, T., Ferrari, V., Leibe, B., Gool, L.V.: Efficient Mining of Frequent and Distinctive Feature Configurations. In: ICCV (2007)
16. Quattoni, A., Torralba, A.: Recognizing indoor scenes. In: CVPR (2009)
17. Savarese, S., Winn, J., Criminisi, A.: Discriminative Object Class Models of Appearance and Shape by Correlatons. In: CVPR (2006)
18. Sivic, J., Russell, B., Efros, A., Zisserman, A., Freeman, W.: Discovering objects and their location in images. In: ICCV (2005)
19. Torresani, L., Szummer, M., Fitzgibbon, A.: Learning query-dependent prefilters for scalable image retrieval. In: CVPR (2009)
20. Uijlings, J., Smeulders, A., Scha, R.: Real-time Bag of Words. In: CIVR (2009)
21. Vedaldi, A., Soatto, S.: Relaxed matching kernels for robust image comparison. In: CVPR (2008)
22. Winn, J., Criminisi, A., Minka, T.: Object Categorization by Learned Universal Visual Dictionary. In: CVPR (2005)
23. Wu, J., Rehg, J.: Beyond the Euclidean distance: Creating effective visual codebooks using the histogram intersection kernel. In: ICCV (2009)
24. Yang, J., Yu, K., Gong, Y., Huang, T.: Linear spatial pyramid matching using sparse coding for image classification. In: CVPR (2009)
25. Yimeng, Z., Tsuhan, C.: Efficient Kernels for identifying unbounded-order spatial features. In: CVPR (2009)
26. Zhou, X., Cui, N., Li, Z., Liang, F., Huang, T.: Hierarchical Gaussianization for Image Classification. In: ICCV (2009)