LETTER

Local Multi-Coordinate System for Object Retrieval

Go IRIE^{†a)}, Yukito WATANABE[†], Takayuki KUROZUMI[†], and Tetsuya KINEBUCHI[†], Members

SUMMARY Encoding multiple SIFT descriptors into a single vector is a key technique for efficient object image retrieval. In this paper, we propose an extension of local coordinate system (LCS) for image representation. The previous LCS approaches encode each SIFT descriptor by a single local coordinate, which is not adequate for localizing its position in the descriptor space. Instead, we use multiple local coordinates to represent each descriptor with PCA-based decorrelation. Experiments show that this simple modification can improve retrieval performance significantly.

key words: image retrieval, local coordinate system, VLAD

1. Introduction

Object image retrieval is a central topic in computer vision and multimedia research, where the key component is the feature representation of images. Traditional approaches exploit keypoint matching between database images and a query image, typically using SIFT descriptors. However, it is not practical on large-scale databases due to its high computational cost. Modern approaches therefore generate a single vector representation per image. Several approaches have been proposed, such as bag-of-visual-words (BoW) [1]–[3], deep convolutional features [4], [5], and aggregation of SIFT descriptors [6]–[8]. In this paper, we focus on the last category and consider an approach to aggregate a set of SIFT descriptors into a single vector. More specifically, we study local coordinate system (LCS) [9], which is a recent variant of the vector of locally aggregated descriptors (VLAD) [7].

VLAD & LCS. VLAD [7] is a popular approach for aggregation of local descriptors. VLAD first quantizes descriptors into visual words. Unlike BoW which describes an image as a histogram of visual words, VLAD rather computes the sum of the residual vectors. Many variants have also been proposed [9]–[12]. Local coordinate system (LCS) [9], [12] is one of the most recent techniques along this line. The heart of LCS is to put an independent coordinate system around each visual word centroid in the descriptor space and to perform encoding of the descriptors according to the coordinate system [9]. LCS significantly improves retrieval quality, because it can suppress “bursty” visual patterns (e.g., repetitive structures such as building windows or tree leaves) that naturally appear in real images and have harmful effects

Fig. 1 Illustration of our LMCS. (a) In LCS, the position of a SIFT descriptor in the descriptor space is localized using only its single closest centroid; in this example, the angles of $\mathbf{x}_1$ and $\mathbf{x}_2$ are both measured from $\mathbf{c}_1$. However, if two descriptors are almost in the same direction, their positions cannot be distinguished. (b) Our LMCS assigns two or more closest centroids to each descriptor. Incorporating angles from different centroid(s), $\mathbf{c}_2$ in this case, results in a much more accurate localization.

on image similarities [9], [12].

Our contribution. We propose a simple extension of LCS that can improve retrieval performance significantly. The original LCS assigns only a single local coordinate (i.e., single visual word) to each descriptor. However, a single coordinate is not adequate for localizing its global position in the descriptor space (as illustrated in Fig. 1). Instead, our approach called local multi-coordinate system (LMCS) uses multiple local coordinates to enhance the localization quality. This multi-coordinate strategy gives better localization quality. However, it is not sufficient to improve performance, because it simultaneously makes the resulting representation redundant. We therefore decorrelate the coordinates to avoid this redundancy. A relevant idea called multi-assignment [2], [3] has been proposed. However, it is mainly for BoW encoding with a large-scale vocabulary (e.g., 100K+ visual words) and does not improve performance for smaller codebooks. This paper considers LCS encoding with small codebooks (up to 256 visual words). Another related technique is triangulation embedding (Temb) [8]. Unlike Temb, our approach uses only a few visual words (local coordinates) to encode each descriptor hence does not need to store dense high-dimensional codes in its encoding process. Our experiments on two public datasets show that it can improve the performance of LCS significantly.

2. Preliminaries

We review VLAD and LCS which are two relevant approaches to this paper.

Manuscript received May 17, 2016.

Manuscript publicized July 6, 2016.

[†]The authors are with NTT Media Intelligence Laboratories, NTT Corporation, Yokosuka-shi, 239-0847 Japan.

a) E-mail: irie.go@lab.ntt.co.jp

DOI: 10.1587/transinf.2016EDL8109

2.1 VLAD

VLAD is a technique to generate a single vector representation from a set of local descriptors extracted from an image. Suppose we have a codebook with $|C|$ visual word centroids $C = \{\mathbf{c}_j \in \mathbb{R}^d\}_{j=1}^{|C|}$ pre-trained offline (typically by k-means). Given an image, the encoding process is done in four steps: (i) A set of n SIFT descriptors of d -dimensional $\mathcal{X} = \{\mathbf{x}_i \in \mathbb{R}^d\}_{i=1}^n$ is extracted from the image. (ii) Each descriptor is assigned to its closest visual word. Let us denote by $q(\mathbf{x})$ the closest visual word centroid of $\mathbf{x}$, i.e., $q(\mathbf{x}) = \arg \min_{\mathbf{c} \in C} \|\mathbf{x} - \mathbf{c}\|_2$. (iii) For each visual word, the sum of the residuals is computed as

$$\mathbf{v}_j = \sum_{\mathbf{x} \text{ s.t. } q(\mathbf{x})=\mathbf{c}_j} \mathbf{x} - \mathbf{c}_j. \quad (1)$$

(iv) The image is finally represented by a D -dimensional vector $\mathbf{y} = [\mathbf{v}_1^\top \dots \mathbf{v}_{|C|}^\top]^\top$ where $D = d \times |C|$.

In addition, two normalization techniques called residual normalization and power normalization are usually applied to improve performance [7], [9], [11].

Residual Normalization is used so that all descriptors equally contribute to the VLAD vector [9]. Specifically, (1) is replaced by

$$\mathbf{v}_j = \sum_{\mathbf{x} \text{ s.t. } q(\mathbf{x})=\mathbf{c}_j} \frac{\mathbf{x} - \mathbf{c}_j}{\|\mathbf{x} - \mathbf{c}_j\|_2}. \quad (2)$$

This evaluates the angle of the residual from its corresponding centroid which is more robust than the absolute distance used in (1) [8], [9].

Power-law Normalization is used to suppress bursty descriptor patterns [6]. Specifically, each element of the VLAD vector is updated as $y_i \leftarrow |y_i|^\alpha \text{sign}(y_i)$, where α is a parameter. Typically $\alpha = 0.2$ is the best choice for VLAD [9]. The VLAD vector is subsequently ℓ_2 -normalized as $\mathbf{y} \leftarrow \frac{\mathbf{y}}{\|\mathbf{y}\|_2}$.

2.2 Local Coordinate System (LCS)

LCS is a recent VLAD-type encoding technique [9]. The motivation is to enhance the effect of power-law normalization to handle various bursty patterns. In the LCS framework, each visual word centroid is regarded to be the origin of a local coordinate system, and a descriptor is encoded according to the coordinate of its associated visual word. Formally, instead of (2), LCS uses the following formulation.

$$\mathbf{v}_j = \sum_{\mathbf{x} \text{ s.t. } q(\mathbf{x})=\mathbf{c}_j} P_j \frac{\mathbf{x} - \mathbf{c}_j}{\|\mathbf{x} - \mathbf{c}_j\|_2}, \quad (3)$$

where $P_j \in \mathbb{R}^{d \times d}$ is a rotation matrix derived using principal component analysis (PCA), which determines the local coordinate for the j -th visual word. P_j is learned offline on the SIFT descriptors of a training dataset which are assigned

to the j -th visual word. By performing PCA for each visual word, the major bursty pattern around each visual word can be captured by the first PCA direction. Therefore, its magnitude is effectively downweighted through power-law normalization and leads to a performance gain [9].

Eggert et al. [12] propose a simple modification. In the original LCS, PCA rotation matrices are learned on individual SIFT descriptors. Instead, considering that the power-law normalization is applied to aggregated vectors (i.e., $\mathbf{v}_j$ in (3)) but not to individual descriptors, PCA matrices are better to be learned on the aggregated residual vectors. We hereafter assume this version of LCS, unless otherwise noted.

3. Our Approach

3.1 Local Multi-Coordinate System (LMCS)

One shortcoming of the previous LCS approaches may be that each SIFT descriptor is assigned to a single coordinate system. In this case, the position of each SIFT descriptor is represented by the direction (angle) from a single associated centroid. However, one centroid is not enough for localizing the global geometric position of each descriptor in the descriptor space. Let us explain the idea by using the simple illustration given in Fig. 1(a). Suppose we have two SIFT descriptors $\mathbf{x}_1$ and $\mathbf{x}_2$ that are assigned to the same centroid $\mathbf{c}_1$. If they are very close to each other in terms of their angles, these two descriptors are not distinguishable anymore, even though they are distant from each other. Actually, points in similar directions are always indistinguishable, as long as their positions are determined by only a single centroid. Possible solutions would be to use larger or multiple codebooks to get finer localization quality, as is done in hierarchical VLAD [12] or in the multi-vocabulary encoding techniques presented in [10], [11]. However, these approaches make the intermediate/final vector representation much longer, resulting in a slower computation and large memory consumption.

Based on these observations, our idea is to use multiple visual words (i.e., multiple local coordinate systems) to localize each SIFT descriptor without changing the codebook or vector length. As illustrated in Fig. 1(b), this allows us to distinguish the positions of two different descriptors, even if these are close in angle from one centroid, by looking at them from two or more centroids. The processing of LMCS can be summarized as follows: (i) Extract SIFT descriptors from a given image. (ii) For each descriptor, find m closest centroids ($m \geq 1$) and assign them to the descriptor. (iii) Compute an encoding vector for each visual word using

$$\mathbf{v}_j = \sum_{\mathbf{x} \text{ s.t. } \mathbf{c}_j \in Q(\mathbf{x})} P_j \frac{\mathbf{x} - \mathbf{c}_j}{\|\mathbf{x} - \mathbf{c}_j\|_2}, \quad (4)$$

where $Q(\mathbf{x})$ is the set of m closest visual word centroids for $\mathbf{x}$. (iv) Generate the LMCS vector as $\mathbf{y} = [\mathbf{v}_1^\top \dots \mathbf{v}_{|C|}^\top]^\top$. Note that the PCA rotation matrix P_j for LMCS is learned

using SIFT descriptors (from the training set) such that $\forall \mathbf{x} | \mathbf{c}_j \in Q(\mathbf{x})$, leading to a different matrix from that of LCS.

3.2 PCA-Based Decorrelation

LMCS provides much more accurate localization quality of descriptors in the descriptor space. However, this simultaneously brings with it another problem; different local coordinates are undesirably correlated to each other. In Fig. 1(b), the two visual word centroids $\mathbf{c}_1$ and $\mathbf{c}_2$ naturally have much higher correlation than in the case of LCS in Fig. 1(a), because the two centroids explicitly “share” the same descriptors. To see this in more detail, we show the correlation matrices of the LCS and LMCS vectors in Fig. 2. Different visual word (local coordinate) blocks of LMCS vectors (Fig. 2(b)) are more highly correlated with each other in comparison with those of LCS vectors (Fig. 2(a)). Unfortunately, such interference makes the image similarity unreliable and degrades retrieval performance.

We therefore try to decorrelate the dimensions of LMCS vectors while preserving information. This goal can be achieved by applying PCA to the vectors without reducing their dimensionalities. The PCA rotation matrix $A = [\mathbf{e}_1, \dots, \mathbf{e}_D] \in \mathbb{R}^{D \times D}$ is learned on the LMCS vectors of the training images, where $\mathbf{e}_k \in \mathbb{R}^D$ is the k -th eigenvector of the covariance matrix of the vectors. For efficient computation of A , we compute at most the first 1,024 eigenvectors by eigendecomposition of the covariance matrix, and the remaining orthogonal complements up to D -dimensional are filled using Gram-Schmidt orthogonalization [8]. Learning A for $|C| = 64$ from 6K images takes around only 1.5 min. on a Linux PC with Intel Xeon CPU 2.60GHz. The learned matrix A is applied to the original LMCS vector $\mathbf{y}$. Specifically, we compute a new decorrelated LMCS vector $\tilde{\mathbf{y}}$ as $\tilde{\mathbf{y}} = A^T \mathbf{y}$ and $\tilde{\mathbf{y}}$ is subsequently power-law normalized.

The resulting correlation matrix of LMCS vectors after PCA-based decorrelation is visualized in Fig. 2(c). As can be seen, the different blocks of LMCS vectors have been decorrelated. We call this approach LMCS+PCA. As shown later in the experiments, PCA-based decorrelation is essential to LMCS and significantly boosts its retrieval quality, but this is not the case in LCS. LMCS+PCA significantly

outperforms LCS and LCS+PCA.

4. Experiments

We analyze the performance of LMCS on the following two public datasets in a common evaluation protocol [9], [11].

Holidays [3] consists of 1,491 photos of objects and scenes. 500 out of them are used as queries. For training, we use a distinctive set of 60K images collected from Flickr[†].

Oxford5k [13] consists of a total of 5,062 images of buildings in Oxford. 55 images of 11 specific buildings are used as queries. For training, we use the Paris6k dataset [2].

The codebooks and PCA rotation matrices are all learned offline using independent training datasets. The retrieval performance is measured by the mean average precision (mAP).

We use a Hessian-affine detector to extract keypoints. Each detected keypoint is described by RootSIFT with centering, PCA rotation, and ℓ_2 -normalization [9]. For VLAD, LCS, and LMCS, we apply residual normalization and power-law normalization with $\alpha = 0.2$. Similarities of the vectors are measured using the cosine similarity.

4.1 Performance Analysis of LMCS

We analyze the performance of our approach by comparing it with VLAD and LCS baselines. Table 1 shows the results (the codebook is fixed for all the versions listed in the table). We first evaluate our approach as to the number of closest centroids ($m = 2, 3, 5$). We found that LMCS only does not improve the performance, and it is even worse than LCS in some cases. We next apply PCA-based decorrelation to LMCS, i.e., LMCS+PCA. This boosts the performance significantly, and LMCS+PCA outperform the baselines in all the cases. Interestingly, we found that applying PCA to LCS (LCS+PCA) does not improve the performance. These suggest that LMCS can encode more discriminative information than LCS and it becomes effective by PCA.

Figure 3(a) illustrates the performance for various numbers of the closest centroids m . As discussed above,

Fig. 2 Correlation matrices of the final vectors ($|C| = 16$) obtained by (a) LCS and (b) LMCS ($m = 5$) encodings are shown. For brevity, only the first four dimensions of each visual word block (total $4 \times 16 = 64$ dimensions) are visualized in the figures. Due to the nature of LMCS’s multi-coordinate encoding scheme, the cross-correlations between different blocks become much higher than those of LCS. (c) By applying PCA to the LMCS vectors, the blocks become decorrelated and non-redundant.

Table 1 Impact of LMCS and PCA-based decorrelation.

↓ Method	$ C \rightarrow$	Holidays			Oxford5k		
		16	64	256	16	64	256
VLAD		58.6	63.8	67.3	43.7	52.0	59.4
LCS		65.3	67.6	70.7	47.3	57.6	63.7
LMCS ($m = 2$)		57.0	64.5	71.2	45.9	55.4	61.7
LMCS ($m = 3$)		50.0	59.0	69.7	44.1	54.3	61.1
LMCS ($m = 5$)		45.3	68.3	70.8	42.6	53.0	60.4
LCS+PCA		64.7	68.9	70.4	47.5	55.6	60.8
LMCS+PCA ($m = 2$)		65.9	72.2	74.3	49.5	58.3	62.5
LMCS+PCA ($m = 3$)		66.1	72.1	74.9	48.1	56.1	63.6
LMCS+PCA ($m = 5$)		64.0	71.1	75.9	47.3	56.7	64.2

[†]The public Flickr60K dataset [3] does not contain information about which image each descriptor comes from, which is necessary to train P_j . We therefore download 60K images and extract 5M descriptors (the same numbers as in the original Flickr60K) for P_j .

regardless of m , LMCS only does not improve performance, but large improvements are had in combination with PCA. The performance is rather sensitive to m , so tuning by cross-validation may boost performance in practice. Figure 3(b) shows the performance on various power-law normalization coefficients α . As reported in the previous studies [9], the performance dramatically changes depending on α , and it is at a maximum around $\alpha = 0.2$.

4.2 Comparison with the State-of-the-Art

We compare our LMCS+PCA with the state-of-the-art VLAD-type encoding methods including VLAD [7], Fisher [7], VLAD-intra [11], and LCS[†] [9]. We also compare LMCS+PCA with Temb [8], another encoding method to generate dense vectors. Table 2 shows the results. We found that LMCS+PCA outperforms LCS. The gain over LCS is around 10% on both datasets. In addition, our method is better than most of the compared methods. Only one exception is Temb which is better than LMCS+PCA for $|C| = 64$. Note that Temb encoding uses all the visual words and has to store a dense D -dimensional vector

Fig. 3 Parameter study on Holidays. All the results of LMCS and LMCS+PCA are for $|C| = 256$ and $m = 2$.

Table 2 Comparison with state-of-the-art methods.

Method	$ C $	Holidays	Oxford5k
VLAD [7]	64	55.6	37.8
VLAD [7]	256	58.7	-
Fisher [7]	64	59.5	41.8
Fisher [7]	256	62.5	-
VLAD-intra [11]	64	56.5	44.8
VLAD-intra [11]	256	65.3	55.8
LCS [9]	64	65.8	51.7
Temb [8]	64	74.5	63.3
LMCS+PCA ($m = 5$)	64	71.1	56.7
LMCS+PCA ($m = 5$)	256	75.9	64.2

Table 3 Performance on short 128-dimensional signatures.

Method	$ C $	Holidays	Oxford5k
VLAD [7]	64	55.7	28.7
Fisher [7]	64	56.5	30.1
Multivoc-VLAD [10]	4×256	61.4	-
Multivoc-VLAD-intra [11]	4×256	62.5	-
LMCS ($m = 5$)	64	58.9	39.5
LMCS ($m = 5$)	256	60.9	42.1

[†]This is the original LCS proposed in [9].

($D = d \times |C|$) per descriptor in its encoding process, so it may be practical only with small codebooks, e.g., $|C| \leq 64$. Conversely, LMCS still represents each descriptor by using only a few visual words^{††}, hence can be coupled with larger codebooks for better performance. Indeed, LMCS+PCA with $|C| = 256$ can achieve slightly better performances compared with Temb.

We also evaluate the performance for short signatures when the dimensionality of the final vectors is reduced to 128 by the whitening method [10]. LMCS outperforms VLAD and Fisher by 5% on Holidays and 10% on Oxford5k. By contrast, Multivoc-VLAD [10] and Multivoc-VLAD-intra [11] are slightly better than LMCS. This may be because these methods use multiple codebooks to generate their vectors (4 times longer than LMCS), which may allow them to encode richer information.

5. Conclusions

We presented LMCS, a simple extension of LCS for object image retrieval. Unlike LCS, our LMCS uses multiple local coordinates to represent each descriptor for better localization in the descriptor space. Coupled with PCA-based decorrelation, LMCS achieved better retrieval performance than LCS. One future work would be to evaluate its performance when it is combined with different aggregation methods, e.g., democratic aggregation [8]. Using deep convolutional features instead of SIFT descriptors (as done in [5]) would be another interesting future work.

References

- [1] J. Sivic and A. Zisserman, “Video google: A text retrieval approach to object matching in videos,” *Int. Conf. Comput. Vision (ICCV)*, pp.1470–1477, 2003.
- [2] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman, “Lost in quantization: Improving particular object retrieval in large scale image databases,” *Conf. Comput. Vision & Patt. Recog. (CVPR)*, pp.1–8, 2008.
- [3] H. Jégou, M. Douze, and C. Schmid, “Improving bag-of-features for large scale image search,” *Int. J. Comput. Vision (IJCV)*, vol.87, no.3, pp.316–336, 2010.
- [4] A. Babenko, A. Slesarev, A. Chigorin, and V.S. Lempitsky, “Neural codes for image retrieval,” *Euro. Conf. Comput. Vision (ECCV)*, pp.584–599, 2014.
- [5] Y. Gong, L. Wang, R. Guo, and S. Lazebnik, “Multi-scale orderless pooling of deep convolutional activation features,” *Euro. Conf. Comput. Vision (ECCV)*, pp.392–407, 2014.
- [6] F. Perronnin, J. Sánchez, and T. Mensink, “Improving the Fisher kernel for large-scale image classification,” *Euro. Conf. Comput. Vision (ECCV)*, pp.143–156, 2010.
- [7] H. Jégou, F. Perronnin, M. Douze, J. Sánchez, P. Pérez, and C. Schmid, “Aggregating local image descriptors into compact codes,” *IEEE Trans. Pattern Analysis & Machine Intelligence (PAMI)*, vol.34, no.9, pp.1704–1716, 2012.
- [8] H. Jégou and A. Zisserman, “Triangulation embedding and democratic aggregation for image search,” *Conf. Comput. Vision & Patt. Recog. (CVPR)*, pp.3310–3317, 2014.
- [9] J. Delhumeau, P.H. Gosselin, H. Jégou, and P. Pérez, “Revisiting the

^{††}LMCS vectors (before PCA) thus can even be efficiently stored and searched using inverted files, unlike Temb.

- VLAD image representation,” *ACM Multimedia*, pp.653–656, 2013.
- [10] H. Jégou and O. Chum, “Negative evidences and co-occurrences in image retrieval: The benefit of PCA and whitening,” *Euro. Conf. Comput. Vision (ECCV)*, pp.774–787, 2012.
- [11] R. Arandjelović and A. Zisserman, “All about VLAD,” *Conf. Comput. Vision & Patt. Recog. (CVPR)*, pp.1578–1585, 2013.
- [12] C. Eggert, S. Romberg, and R. Lienhart, “Improving VLAD: Hierarchical coding and a refined local coordinate system,” *Int. Conf. Image Proces. (ICIP)*, pp.3018–3022, 2014.
- [13] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman, “Object retrieval with large vocabularies and fast spatial matching,” *Conf. Comput. Vision & Patt. Recog. (CVPR)*, pp.1–8, 2007.
-