Information Sciences 417 (2017) 143–153

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Local residual similarity for image re-ranking

Shaoyan Sun[a], Ying Li[b], Wengang Zhou[a], Qi Tian[c], Houqiang Li[a,*]

[a] EEIS Department, USTC, Hefei 230027, PR China
[b] ICE School, DUT, Dalian 116023, PR China
[c] One UTSA Circle, San Antonio, TX 78249, U.S.

ARTICLE INFO

Article history:
Received 12 September 2016
Revised 14 June 2017
Accepted 4 July 2017
Available online 5 July 2017

2010 MSC:
00-01
99-00

Keywords:
Image retrieval
Re-ranking
Similarity measure

ABSTRACT

Similarity measurement is an essential component in image retrieval systems. While previous work is focused on generic distance estimation, this paper investigates the problem of similarity estimation within a local neighborhood defined in the original feature space. Specifically, our method is characterized in two aspects, *i.e.*, "local" and "residual". First of all, we focus on a subset of the top-ranked relevant images to a query, with which anchors are discovered by methods such as averaging or clustering. The anchors are then subtracted from the neighborhood features, resulting in residual representations. The proposed Local Residual Similarity (LRS) homogenizes the feature distances within the local neighborhood. Effective and efficient image re-ranking is achieved by calculating LRS between the query and the top-ranked images. The method constrains that relevant images should appear similar in both original and local residual feature space. We evaluate the proposed method on two image retrieval benchmarks with global CNN representations, demonstrating a consistent improvement on performance with very limited extra computational cost.

1. Introduction

In this paper we concentrate on the task of image retrieval [1–6]. Given a query image, we aim to find its similar counterparts from an image database. Typically, for each image a feature vector is extracted, with which the similarities between the query and database images are measured. Then the database images with high similarity scores to the query can be returned as retrieval results.

The image retrieval community has witnessed a number of effective image representations. For example, the Bag-of-Visual-Words representation [1,2,7] follows the practice in text retrieval to group the frequencies of visual word terms into a histogram. This model has dominated the image retrieval frameworks over a decade. To aggregate local visual features into a compact representation, VLAD [8] is proposed to train small visual codebooks and sum up residual vectors of local features with respect to the codewords. Most recently, the convolutional neural network (CNN) is successfully applied in image retrieval for feature extraction and produces superior accuracy [9–11].

As effective feature representations are crucial to improve image retrieval performance, we thereby study on more effective usage of the existing feature representations by exploiting residual vectors with respect to some local anchor-points in the neighborhood. The residual representation has been widely applied in the field of machine. As mentioned above, Jégou

* Corresponding author.
E-mail address: lihq@ustc.edu.cn (H. Li).

http://dx.doi.org/10.1016/j.ins.2017.07.004

et al. [8] propose VLAD to aggregate residual vectors of local features. Arandjelović et al. [12], Liu et al. [13], *etc.*, proceed to further improve this representation. Product Quantization [14] is proposed for encoding quantized residual vectors, and Residual Net [15] exploits residual learning to overcome deeper neural network training problem and achieves state-of-the-art classification accuracy. Bai et al. [16] propose a hierarchical residual framework for feature coding and approximate nearest neighbor search. Bai et al. [17] address 3D shape matching by introducing the residual coding algorithm VLAD into a two-layer coding framework.

Motivated by the latent benefit of residual representations, we propose a novel distance measure for image retrieval re-ranking. We first make an initial query with original feature vectors to identify the neighborhood of the query image, and then represent images in the neighborhood as residual vectors with respect to local anchor-points for an extra re-ranking process. While the residual vectors have equivalent Euclidean distance with respect to the original feature vectors, we use cosine distance to measure included angle value of the residual vectors. We demonstrate the feasibility of designing a more effective distance measure with this strategy by exploiting local feature distribution information.

Our work contributes in three aspects:

(1) We propose a novel similarity measure, Local Residual Similarity (LRS), in a neighborhood of the query defined by its top-ranked images. With this, an efficient online re-ranking method is introduced.
(2) We perform extensive evaluations on various strategies for choosing a neighborhood and computing local anchor-points.
(3) We elaborate two strategies to effectively improve the re-ranking performance by imposing reciprocal neighborhood constraints on our method.

2. Related work

CNN based image retrieval has achieved considerable processes in recent years. In [10], Babenko et al. propose to extract features from CNN activations as *Neural Codes*, and demonstrate its practicality in the image retrieval task. Gong et al. [11] extract CNN features from multiple scales and positions in the image, and aggregate the features by pooling. In [18], the authors evaluate multiple aggregating strategies to improve the CNN representation. Apart from these methods that exploit off-the-shelf CNN features, some works train CNN architectures for some specific image retrieval tasks, *e.g.* landmark recognition [19–21].

With well designed feature representations, re-ranking becomes an effective tool to improve the image retrieval results. To refine results in the ranking list, several supervised re-ranking methods are proposed to learn concepts from manually labeled data [22–27]. Wang et al. [23] learn different semantic spaces (*i.e.*, concepts) and generate semantic signatures by projecting correlate visual features into the same semantic space. Some other works employ multiple attributes as constraints for refining [24], and use feedbacks from user interaction in [25,28]. In order to avoid noisy labels introduced by various causes (*e.g.*, undetermined user intentions), Jain et al. [29] alternatively take the statistical information of click data as a reference to facilitate re-ranking.

As a form of automatic relevance feedback without supervision, query expansion [30] is effective to improve recall, which is transferred from the documental domain to the visual field by [31]. Geometrical consistence restriction is applied in [31] to filter out unrelated images in the candidate list, and the rest is used to construct new queries. Chum et al. [32] improve context expansion for each retrieved image, and [5,33] learn weighting factors through SVM. Some other works apply graph diffusion based reranking methods [34–38]. Zhang et al. [35] exploit the reciprocal neighborhood relationship to build a graph for rank list fusion. Pedronette et al. [36] construct a correlation graph with rank information, and apply unsupervised manifold learning to improve image retrieval performance. Further more, some binary signature based methods [39,40] are proposed in concern of efficiency.

Since supervision information is hardly obtained in practical image retrieval systems, the reciprocal neighborhood relationship is widely applied for unsupervised ground truth verification. Jégou et al. [4,41] propose contextual dissimilarity measure (CDM) based on the average distance of features to their neighborhood to maximize the neighborhood symmetry by iteration. Bai et al. [42] take into account the neighborhood of a given visual word vector when the dissimilarity between visual word vectors are measured. Instead of pursuing the symmetry relationships between the nearest neighbors, Qin et al. [43] design separate similarity metrics for highly relevant neighbors and the rest far away points, respectively. Our method harnesses the reciprocal neighborhood relationships for distance measure designing, and also for extensions by integrating some relevant constraints. Different from CDM, which explicitly assigns a score representing the density around each feature, our method implements a query adaptive online distance measure, which requires no extra storage and runs very fast. In addition, we demonstrate that by extending our method with strategies such as CDM we can further improve the performance.

3. Local residual representation for re-ranking

We elaborate our method in this section. First, we introduce the framework of our local residual representation based re-ranking algorithm in Section 3.1. Then we discuss the definition of neighborhood and anchor-points in our method in Section 3.2. In the last, we make some extensions by imposing reciprocal neighbor constraints in Section 3.3.

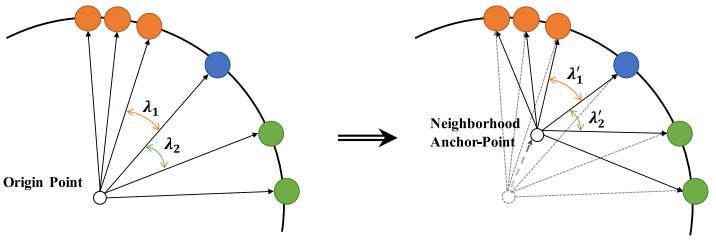


Fig. 1. A toy example of the Local Residual Similarity (LRS). Black curves represent the hypersphere the L_2-normalized features lie on. The blue point denotes the queries, orange points are database images located in a clustered region, and the green ones are scattered database images. All the points are located within a local neighborhood of the query. In the left figure, original feature vectors are used to compute the cosine distance as the similarity, while on the right, residual vectors *w.r.t.* the neighborhood anchor-point are used to compute the cosine distance. We can see that the cosine distances between the query and database features become more homogenized when LRS is applied. In other words, the reversibility of neighborhood relationships of these features is improved. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

3.1. Method overview

In our method, we use CNN to extract global features as the original image representation, which has been demonstrated with superior performance in image retrieval. To be specific, we feed a given image into a CNN model that has been pre-trained on classification tasks (the direct representation scheme as concluded in [9]), and take the middle layer responses as the feature of the whole image. To be more precise, we apply the pre-trained CaffeNet model provided by *caffe* [44], which has been widely used in state-of-the-art approaches. Features from two middle layers are extracted, *i.e.*, 256-dimensional Pool5 layer feature with average pooling on each channel [45] (without aspect ratio resizing of the input image), and 4096-dimensional Fc7 layer feature. The features are rooted dimension-wisely and then L_2-normalized.

Denote the image retrieval database as $\mathcal{D} = \{\boldsymbol{d}_1, \boldsymbol{d}_2, \ldots, \boldsymbol{d}_N\}$, where $\boldsymbol{d}_i$ is the feature representation vector of the ith database image and N is the dataset size. Given a query image $\boldsymbol{q}$, the similarity of $\boldsymbol{q}$ *w.r.t.* database image $\boldsymbol{d}_i$ is computed as $S(\boldsymbol{q}, \boldsymbol{d}_i) = \boldsymbol{q} \cdot \boldsymbol{d}_i$. As all image features are L_2-normalized, *i.e.*, $\|\boldsymbol{q}\|_2 = \|\boldsymbol{d}_i\|_2 = 1$, the similarity $S(\boldsymbol{q}, \boldsymbol{d}_i)$ is equal to the cosine value of the included angle between these two high-dimensional vectors.

In the query phase, we first perform an initial retrieval process with the original image features, and then identify some top-ranked database images as the ***neighborhood*** of the query $\mathcal{N}(\boldsymbol{q})$. After that, we compute one or some ***anchor-points*** with image features inside the neighborhood, denoted as $\mathcal{K}(\boldsymbol{q}) = \left\{\boldsymbol{k}_i^q\right\}, i = 1, \ldots, M$, where M is the number of anchor-points. Then the neighbor images in $\mathcal{N}(\boldsymbol{q})$ and the query $\boldsymbol{q}$ are reformulated as a residual vector *w.r.t.* the anchor-point(s), and their similarities are recomputed using the residual vectors with cosine distance. At last, a re-ranking of the neighborhood images is performed based on the newly computed similarities.

We illustrate this local residual representation with a toy example in Fig. 1. In the left part of the figure, original feature vectors in the high-dimensional space start from the origin $\boldsymbol{0}$, and end at the normalized hypersphere with a radius of 1. In the right part, the features are represented as vectors starting from one neighborhood anchor-point and ending at the same hypersphere. In the shown case, the distance (included angle) between the query and the nearest orange point $\boldsymbol{\lambda}_1$ is enlarged to be $\boldsymbol{\lambda}'_1$, while only marginal variation is observed between $\boldsymbol{\lambda}_2$ and $\boldsymbol{\lambda}'_2$, which indicates the distance (included angle) between the query and the farthest blue point. The toy example shows that with the local residual representation, the feature vectors can have different distance measures. In the next section, we will discuss how to design the proper neighborhood and anchor-points to make a good distance measure for re-ranking.

3.2. Neighborhood & anchor-point

Tow factors are of much importance in the local residual representation. One is the ***neighborhood*** of the query, defining how many database images are involved for computing the anchor-points and for re-ranking. The other one is the ***anchor-point***, with which the residual vectors are generated. To reduce the variable number, we use the same neighborhood for anchor-point computing and re-ranking. Therefore, it should be able to guarantee a high recall while representing the local feature distribution around the query. The anchor-points are crucial to effective re-ranking. They are expected to make a distinctive distance measure to strengthen the matching score between relevant images and eliminate the impact of irrelevant ones. In the following, we introduce the way to choose neighborhood and compute anchor-points.

3.2.1. Neighborhood definition

There are two kinds of popular neighborhood definitions in the field of nearest neighbor search, namely **k-*neighborhood*** and ***ϵ-neighborhood***. In the high-dimensional feature space, k-neighborhood is defined as the top k features most close to the query, while ϵ-neighborhood involves all features that have a distance no larger than ϵ from the query. Since we use cosine distance in this paper, we define ϵ as the minimum similarity of the neighborhood.

Evidently, k-neighborhood contains a fixed number of features regardless of the local feature distribution, while ϵ-neighborhood concerns a fixed-sized range around the query no matter how many features are included, and it is expected

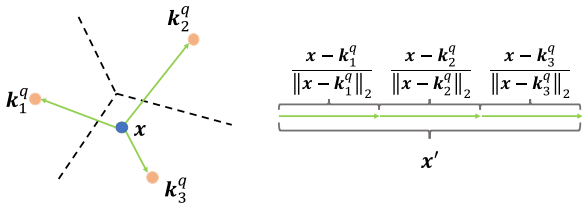


Fig. 2. Illustration of kMean-AP. The residual feature vector $\boldsymbol{x}'$ is formulated as the concatenation of three L_2-normalized residual vector of $\boldsymbol{x}$ *w.r.t.* multiple anchor-points $\left\{\boldsymbol{k}_1^q, \boldsymbol{k}_2^q, \boldsymbol{k}_3^q\right\}$, which are generated with k-mean algorithm.

to adaptively select the neighborhood size according to the feature density around the query. We will evaluate and compare both two neighborhood definitions in the experiments.

3.2.2. Anchor-point computing

The anchor-points mainly aims at making the most of latent information contained in the residual vectors. As we have no supervised information to exploit, it is nontrivial to achieve this objective. However, inspired by the success of contextual dissimilarity measure (CDM) proposed in [4], we target at improving the reversibility of the neighborhood, *i.e*, image features simultaneously appearing in the neighborhood of each other.

As shown in Fig. 1, the reversibility for the query and the farthest green point is not well satisfied in the original feature space, due to the impact of the gathered orange points. However, if we place an anchor-point towards the gathered points, the distance of the query to them will be enlarged (*e.g.*, $\lambda_1 \rightarrow \lambda_1'$), thus improving the reversibility of the neighborhood. Further more, we can see that the orange and green points tend to belong to different classes. But they cannot be distinguished in the view of the query in the original feature space. This is addressed after updating the origin from coordinate $\boldsymbol{0}$ to the selected anchor point, which emphasizes that the true relevant images should have higher similarity score in both the original feature space (in the initial retrieval phase) and local residual feature space (in the re-ranking phase). Meanwhile, re-ranking results in the updated feature space can be more orderly with images from the same class ranked closely.

Intuitively, it is straightforward to set an anchor-point as the mean vector of all features in the neighborhood including the query, which will be closer to the gathered points and make the features distributed more evenly when compared with cosine distance. Also it is possible to set the anchor-point as the median value of these feature vectors. It is suggested to alleviate the impact of alienated feature points around the query. We denote these two anchor-point computing methods as ***Mean-AP*** and ***Median-AP***, respectively. Taking more fine-grained residual information into consideration, we can further use multiple anchor-points. In this case, each feature vector in the neighborhood is represented as concatenated residual vectors *w.r.t.* all anchor-points. Specifically, assuming we have anchor-points $\mathcal{K}(\boldsymbol{q}) = \left\{\boldsymbol{k}_i^q\right\}, i = 1, \ldots, M$, the residual vector of the query or a database image feature $\boldsymbol{x}$ is obtained via

$$\boldsymbol{x}' = \frac{\boldsymbol{x} - \boldsymbol{k}_1^q}{\left\|\boldsymbol{x} - \boldsymbol{k}_1^q\right\|_2} \oplus \cdots \oplus \frac{\boldsymbol{x} - \boldsymbol{k}_M^q}{\left\|\boldsymbol{x} - \boldsymbol{k}_M^q\right\|_2}, \forall \boldsymbol{x} \in \{\boldsymbol{q}, \mathcal{N}(\boldsymbol{q})\}, \tag{1}$$

where the symbol $\oplus$ represents concatenation of vectors. We perform L_2-normalization on each segmentation in the concatenated vector, so that the similarity of two feature vectors $\boldsymbol{q}$ and $\boldsymbol{d}$ can be directly computed with an inner product operation $S(\boldsymbol{q}, \boldsymbol{d}) = \boldsymbol{q} \cdot \boldsymbol{d}$, which is equal to the sum of cosine distances for all segmentations. Apparently, the more anchor-points are used, the higher dimension the residual representation will have.

We use k-mean algorithm to compute M cluster centers as the M anchor-points, representing fine-grained local feature distributions. This multiple anchor-point computing method is denoted as ***kMean-AP***. Note that when $M = 1$, this representation degrades to the aforementioned Mean-AP method. We illustrate the kMean-AP method in Fig. 2. This representation is to some extent similar to the VLAD formulation [8], but different in that our method has query adaptive cluster centers. They reflects the local feature distribution around the query. Also we reserve the complete residual information of image feature rather than average multiple features into one residual vector.

The complete retrieval procedure is summarized in Algorithm 1. The residual feature computation is abstracted as the function *GetResidualFeature*, and the retrieval system is executed in *offline* indexing and *online* query stages, separately.

3.3. Reciprocal neighborhood constraints

As discussed above, the residual feature vector representation is implicitly designed to improve the reversibility of the neighborhood. In this section we discuss two methods to explicitly impose a reciprocal neighborhood constraint following the widely-observed property that relevant images usually appear as one of the nearest neighbors of each other. The first method is ***CDM extension*** and the second is ***database augmentation***.

3.3.1. CDM extension

The contextual dissimilarity measure (CDM) [4] is effective to exploit the local distribution of features for distance computing. This approach assigns one neighborhood distance factor to each feature in the database which represents the

Algorithm 1 The local residual similarity for image retrieval.

```
function GetResidualFeature(Feature x, Anchor Points AP)
    x' = []
    for k ∈ AP do
        x' ← [x', (x − k)/‖x − k‖_2]
    return x'

function RetrievalSystem(Query feature q, Database D = [d_1, ..., d_N])
Offline:
    Nothing
Online:
    D ← rank D with cosine distances w.r.t q
    D_t = &[top T elements in D] // k-neighborhood or ε-neighborhood
    AP = anchor points within D_t
    q ← GetResidualFeature(q, AP)
    for d ∈ D_t do
        d ← GetResidualFeature(d, AP)
        Scores[d] = q · d
    D_t ← Rank D_t by Scores
    return D
```

average distance of the feature to its neighborhood:

$$\lambda(\boldsymbol{d}) = \frac{1}{K_{\mathrm{d}}} \sum_{\boldsymbol{d}_i \in \mathcal{N}(\boldsymbol{d}, K_{\mathrm{d}})} \|\boldsymbol{d}_i - \boldsymbol{d}\|_2, \tag{2}$$

where $\mathcal{N}(\boldsymbol{d}, K_{\mathrm{d}})$ is the K_{d}-nearest neighborhood of the feature $\boldsymbol{d}$. We then compute the distance of the query $\boldsymbol{q}$ with one database image $\boldsymbol{d}$ as

$$D^*(\boldsymbol{q}, \boldsymbol{d}) = D(\boldsymbol{q}, \boldsymbol{d}) / \sqrt{\lambda(\boldsymbol{d})}. \tag{3}$$

Here $D(\cdot)$ denotes Euclidean distance, which is computed as $D(\boldsymbol{q}, \boldsymbol{d}) = 2 - 2\boldsymbol{q} \cdot \boldsymbol{d}$. We apply this CDM weighting for distance computing in both the initial retrieval phase and the re-ranking phase, where the neighborhood distance factor is computed with the original feature vectors and the L_2-normalized residual vectors, respectively. We summarize the retrieval algorithm of our method with CDM extension in Algorithm 2. In the offline stage, the CDM neighborhood distance factors for original features are pre-computed.

Our method improves the reversibility of neighborhood from a different perspective with CDM, and it is expected to attain a more balanced neighborhood and more effective distance measure when they work together.

3.3.2. Database augmentation

With the neighborhood of the query image considered, it is straightforward to further augment all the database images with local residual representation. So far we have been focusing on computing anchor-points in the neighborhood of the query. Actually we can also compute anchor-points around each database image in the offline phase, and use them for another residual representation of the query and the database image. This would impose a stronger constraint for image matching, that in view of both two images, they should be matched with the local residual representation.

Concretely, supposing we compute one anchor-point around the query $\boldsymbol{q}$ as $\boldsymbol{k}^q$, and another around the database image $\boldsymbol{d}$ as $\boldsymbol{k}^d$ (computed offline), then we have residual representation *w.r.t.* $\boldsymbol{k}^q$ as $\boldsymbol{q}'_q = \boldsymbol{q} - \boldsymbol{k}^q, \boldsymbol{d}'_q = \boldsymbol{d} - \boldsymbol{k}^q$, and *w.r.t.* $\boldsymbol{k}^d$ as $\boldsymbol{q}'_d = \boldsymbol{q} - \boldsymbol{k}^d, \boldsymbol{d}'_d = \boldsymbol{d} - \boldsymbol{k}^d$, respectively. With these residual representations, we can compute the similarity between $\boldsymbol{q}$ and $\boldsymbol{d}$ as

$$S(\boldsymbol{q}, \boldsymbol{d}) = \frac{\boldsymbol{q}'_q \cdot \boldsymbol{d}'_q}{\|\boldsymbol{q}'_q\|_2 \|\boldsymbol{d}'_q\|_2} + \frac{\boldsymbol{q}'_d \cdot \boldsymbol{d}'_d}{\|\boldsymbol{q}'_d\|_2 \|\boldsymbol{d}'_d\|_2}. \tag{4}$$

This matching score implies that two features being compared should be similar from the viewpoint of both their respective local neighborhood distributions. We use the summation rather than product of the two scores because they may appear as negative values. The concrete retrieval algorithm for our method with database augmentation is summarized in Algorithm 3.

4. Experiments

We evaluate our method (LRS) on two popular benchmark datasets, *i.e.*, Holidays dataset [46] and UKBench dataset [1]. Mean Average Precision (mAP) is used to evaluate the retrieval accuracy for Holidays dataset, and NS-score (averaged four times top-4 accuracy) is used to measure the retrieval accuracy for UKBench. In the following, we first inspect the impact of

Algorithm 2 The local residual similarity for image retrieval with CDM extension.

function ComputeCDMFactors(Database $\mathcal{D}$)
  for $\boldsymbol{d} \in \mathcal{D}$ **do**
    $\lambda s[\boldsymbol{d}] = \frac{1}{K_d} \sum_{\boldsymbol{d}_i \in \mathcal{N}(\boldsymbol{d}, K_d)} \|\boldsymbol{d}_i - \boldsymbol{d}\|_2$ (Eq. (2))
  return λs

function RankWithCDM(Query feature $\boldsymbol{q}$, Database $\mathcal{D}$, CDM Factors λs)
  for $\boldsymbol{d} \in \mathcal{D}$ **do**
    Dists$[\boldsymbol{d}] = D(\boldsymbol{q}, \boldsymbol{d}) / \sqrt{\lambda s[\boldsymbol{d}]}$ (Eq. (3))
  return Sort $\mathcal{D}$ by Dists

function RetrievalSystemCDM(Query feature $\boldsymbol{q}$, Database $\mathcal{D} = [\boldsymbol{d}_1, \ldots, \boldsymbol{d}_N]$)
Offline:
  $\lambda_{offline} =$ ComputeCDMFactors$(\mathcal{D})$
Online:
  $\mathcal{D} \leftarrow$ RankWithCDM$(\boldsymbol{q}, \mathcal{D}, \lambda_{offline})$
  $\mathcal{D}_t =$ &[top results in $\mathcal{D}$]
  $\boldsymbol{AP} =$ anchor points within $\mathcal{D}_t$
  $\boldsymbol{q} \leftarrow$ GetResidualFeature$(\boldsymbol{q}, \boldsymbol{AP})$
  for $\boldsymbol{d} \in \mathcal{D}_t$ **do**
    $\boldsymbol{d} \leftarrow$ GetResidualFeature$(\boldsymbol{d}, \boldsymbol{AP})$
  $\lambda_{online} =$ ComputeCDMFactors$(\mathcal{D}_t)$
  $\mathcal{D}_t \leftarrow$ RankWithCDM$(\boldsymbol{q}, \mathcal{D}_t, \lambda_{online})$
  return $\mathcal{D}$

Algorithm 3 The local residual similarity for image retrieval with database augmentation.

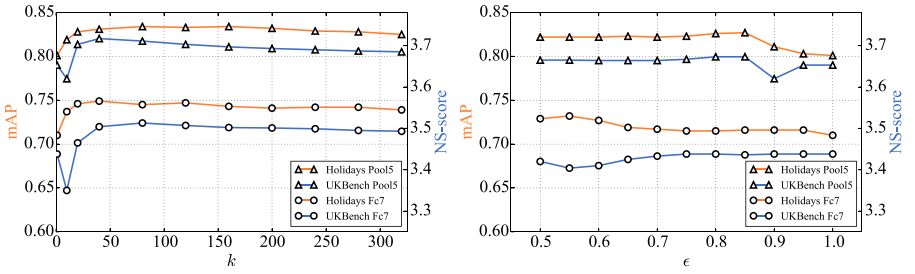
function RetrievalSystemDA(Query feature $\boldsymbol{q}$, Database $\mathcal{D} = [\boldsymbol{d}_1, \ldots, \boldsymbol{d}_N]$)
Offline:
  for $\boldsymbol{d} \in \mathcal{D}$ **do**
    $\mathcal{D} \leftarrow$ rank $\mathcal{D}$ with cosine distances *w.r.t* $\boldsymbol{d}$
    $\mathcal{D}_t =$ &[top T elements in $\mathcal{D}$]
    $\boldsymbol{AP}_{db}[\boldsymbol{d}] =$ anchor points within $\mathcal{D}_t$
    $\boldsymbol{d}_{db} \leftarrow$ GetResidualFeature$(\boldsymbol{d}, \boldsymbol{AP}_{db})$
Online:
  $\mathcal{D} \leftarrow$ rank $\mathcal{D}$ with cosine distances *w.r.t* $\boldsymbol{q}$
  $\mathcal{D}_t =$ &[top T elements in $\mathcal{D}$]
  $\boldsymbol{AP}_q =$ anchor points within $\mathcal{D}_t$
  $\boldsymbol{q}_q \leftarrow$ GetResidualFeature$(\boldsymbol{q}, \boldsymbol{AP}_q)$
  for $\boldsymbol{d} \in \mathcal{D}_t$ **do**
    $\boldsymbol{d}_q \leftarrow$ GetResidualFeature$(\boldsymbol{d}, \boldsymbol{AP}_q)$
    $\boldsymbol{q}_{db} \leftarrow$ GetResidualFeature$(\boldsymbol{d}, \boldsymbol{AP}_{db}[\boldsymbol{d}])$
    Scores$[\boldsymbol{d}] = \boldsymbol{q}_q \cdot \boldsymbol{d}_q + \boldsymbol{q}_{db} \cdot \boldsymbol{d}_{db}$
  $\mathcal{D}_t \leftarrow$ Rank $\mathcal{D}_t$ by Scores
  return $\mathcal{D}$

neighborhood definition and anchor-point computing in Section 4.1. Then we demonstrate the effectiveness of the proposed reciprocal neighborhood constraints in Section 4.2. Finally, we compare our method with other related work in Section 4.3. Note that all the evaluations are performed with the two CNN features introduced in Section 3.1.

4.1. Impact of neighborhood and anchor-points

In general, larger neighborhood ensures higher recall for re-ranking, but may lose some distinct information about the feature local distribution around the query. We first test the image retrieval performance with different settings of k for k-neighborhood and ϵ for ϵ-neighborhood. For these experiments, we generate one anchor-point in the neighborhood by Mean-AP.

We show experimental results on Holdiays and UKBench datasets in Fig. 3. When $k = 1$ or $\epsilon = 1.0$, only the first retrieval result (*i.e.*, the query itself) is used for re-ranking, which is equivalent to the baseline without local residual re-ranking. We



(a) Accuracy with k-neighborhood (b) Accuracy with ϵ-neighborhood

Fig. 3. Impact of the two neighborhood definitions, evaluated with Mean-AP.

Table 1
Comparison of retrieval accuracy (mAP for Holidays and NS-Score for UKBench) for different anchor-point computing methods. kMean-AP/M denotes that the cluster number in the kMean-AP method is set as M.

Method		Median-AP	Mean-AP	kMean-AP/2	kMean-AP/3	kMean-AP/4
Holidays	Pool5	83.4	83.5	**84.0**	**84.0**	83.9
	Fc7	73.7	**74.9**	74.8	74.8	74.7
UKBench	Pool5	3.71	3.72	3.72	**3.73**	3.72
	Fc7	3.49	3.51	**3.52**	**3.52**	3.51

Table 2
Accuracy comparison of the proposed method LRS with the baseline (mAP for Holidays and NS-Score for UKBench), extension with CDM (LRS+CDM) and database augmentation (LRS+DA), and related works. Our best results are marked bold, and the best results among all methods are underlined.

Method		Baseline	LRS	LRS+CDM	LRS+DA	CDM [4]	TR [31]	HN [43]	SCA [42]
Holidays	Pool5	80.1	84.0	**85.5**	84.6	83.8	79.9	80.6	24.9
	Fc7	71.0	74.8	**76.2**	75.7	75.4	71.1	72.2	24.7
UKBench	Pool5	3.65	3.73	3.75	**3.76**	3.73	3.71	3.67	3.80
	Fc7	3.44	3.52	3.54	**3.56**	3.54	3.51	3.45	3.64

see that the baselines for Pool5 layer features are mAP = 80.1% on Holidays, NS-Score = 3.65 on UKBench, and for Fc7 layer features are mAP = 71.0% on Holidays, NS-Score = 3.44 on UKBench.

On the whole, k-neighborhood performs better than ϵ-neighborhood, and it tends to fluctuate less under different parameter settings. We deem that ϵ-neighborhood is vulnerable to the feature distribution around the query. For example, if the query is located in an extremely sparse neighborhood, then very few features are involved in anchor-point computation and re-ranking.

For k-neighborhood, we observe that when k is large enough (no less than 40), our method keeps outperforming the baseline by about 3 percent on Holidays and 8 percent on UKBench. However, when k further increases, the performance drops slightly on both datasets due to lack of local distribution information. I addition, a larger neighborhood will introduce more computation cost for re-ranking. When k is smaller than 40, the performance is slightly better or lower than the baseline ($k = 1$) because not enough local distribution information is provided and the recall is not well achieved for re-ranking.

We then compare different anchor-point computing methods. For Median-AP and Mean-AP where only one anchor-point is used, we use k-neighborhood with $k = 40$. When using kMean-AP, to ensure enough sample features in each cluster for anchor-point computing, we set the parameter k as $40M$, where M is the cluster number.

The comparison is summarized in Table 1. We see that generally the KMean-AP method with 3 cluster centers obtains the best accuracies for each feature. However, when the cluster number M is 2 or 4, the differences with the best results are not dramatic. Mean-AP achieves the best accuracy on Holidays with Fc7 layer feature, and approaches the best performance in other experiments. Among all the methods, Mean-AP has the least computation cost with only one mean value computing process. Median-AP consumes similar computation time but has a relatively low accuracy. kMean-AP method requires cluster center computing and has a longer feature vector representation. However, since the anchor-point computing and re-ranking process only involve no more than hundreds of features and the cluster number is extremely small, the extra cost is very limited.

4.2. Reciprocal neighborhood constraints

The proposed extensions in Section 3.3 aim at improving feature matching accuracy with reciprocal neighborhood constraints. We compare our local residual representation (LRS) with the CDM extension (LRS+CDM) and database augmentation (LRS+DA) in Table 2. For LRS, we show results of the best neighborhood and anchor-point computing practice as proved in

Table 3
Time cost and complexity of different settings. k, N, D are the re-ranked list length, the database size, and the feature dimension, respectively. t is the iteration number in k-means.

	Mean-AP	kMean-AP/3	LRS+CDM	LRS+DA
Time (Pool5) / ms	0.4	10	8	0.7
Time (Fc7) / ms	10	150	30	10
Complexity	$O(k)$	$O(3kt)$	$O(k^2)$	$O(2k)$
Storage	0	0	N	ND

Table 4
Further comparison of our method with CDM. Best significant level value α on LRS, CDM and CDM extension (LRS+CDM) are listed in the middle 3 columns using paired t-tests with the baseline. The Spearman correlation coefficients for the retrieval results produced by LRS and CDM are listed in the last column.

		LRS	CDM	LRS+CDM	Correlation
Holidays	Pool5	0.1	0.1	0.05	0.013
	Fc7	0.1	0.1	0.05	0.006
UKBench	Pool5	0.01	0.01	0.01	0.011
	Fc7	0.01	0.01	0.01	0.011

Section 4.1 (*i.e.*, kMean-AP/3 and k-neighborhood with $k = 40 \times 3 = 120$). For the CDM extension, we set the neighborhood size $K_d = 10$ in Eq. (2). The two extensions are performed with Mean-AP where only one anchor-point is used concerning the efficiency.

According to the table, both two extensions obtain a consistent performance boosting on two datasets with Pool5 and Fc7 layer features. On Holidays dataset, the CDM extension achieves better accuracies, namely 85.5% with Pool5 layer feature and 76.2% with Fc7 layer feature, which are about 1.5 percents higher than LRS and 5.5 percents higher than the baseline, respectively. While on UKBench dataset, the database augmentation extension achieves the best performances, *i.e.*, 3.76 with Pool5 layer feature and 3.56 with Fc7 layer feature, which are about 0.04 higher than LRS and 0.2 higher than the baseline.

We then compare the time costs in the re-ranking process on a PC for different settings in Table 3. As we fix the feature number involved in re-ranking, the evaluation is actually independent of the database size. We see kMean-AP/3 takes more time on re-ranking because it considers a larger number of features and the residual feature vector has a higher dimension. Fc7 layer has 4096 dimensions so that more time is required than Pool5 layer feature. The CDM extension requires online CDM weight computing in the neighborhood, and consumes more time, while database augmentation is mainly conducted offline and needs one more distance computing for each feature pair, which has slight impact on the re-ranking time. As a whole, the extra time cost brought by our method is insignificant, and even a larger number of images involved in re-ranking can be supported. As to memory cost, no extra memory is needed for all LRS settings without extensions. For CDM extension we store only one weight (floating point data) for each database image feature. For database augmentation, one extra anchor-point vector is stored which has the same dimension as the original feature.

4.3. Comparisons

We compare our method with some representative related works in Table 2, including non-iterative CDM [4], Total Recall (TR) [31] with average query expansion, Hello Neighbor (HN) [43], and Sparse Contextual Activation (SCA) [42]. The experiments are performed with the same features used in our method, and the best results for them are picked out after testing different parameters settings.

Among the compared works, CDM has a more consistent accuracy improvement, and outperforms our LRS with Fc7 layer feature. However, our method achieves better results when applying CDM extension. It indicates that the two methods are complementary. TR has no observable accuracy improvement in the two features probably because no geometry verification method is available for global features. HN only has improvement with Pool5 layer feature on UKBench, which is possibly because the recall of near-set is already very high and the far-set makes little difference. Since SCA is a context-based re-ranking algorithm, it requires sufficient reliable neighborhoods to describe the contextual information of the query. Therefore although it has the best performances on UKBench, it fails on Holidays.

To further compare our method with CDM and demonstrate the effectiveness of the extension, we perform statistical paired t-tests between the results of the 3 types of improvements as listed in Table 4 and the baseline. We test the statistical significant level α at 0.01, 0.05, and 0.1, respectively. It can be seen that results of LRS and CDM on Holidays are statistically significant for $\alpha = 0.1$, and insignificant for $\alpha = 0.05$. Nevertheless, the extension (LRS+CDM) can produce significant improvements for $\alpha = 0.05$. On UKBench, all improvements produce significant results for $\alpha = 0.01$. To further examine the complementary property of LRS and CDM, we compute the Spearman correlation coefficients of the retrieval results produced by LRS and CDM. As displayed in Table 4, these two methods have very uncorrelated retrieval results.

Because our method is performed on the online re-ranking process, it requires no extra computation even if new images are inserted into the database. This is also an advantage of our method over CDM. For CDM, when new images are inserted, the original recorded distance factors need to be updated because the neighborhood properties around the database images

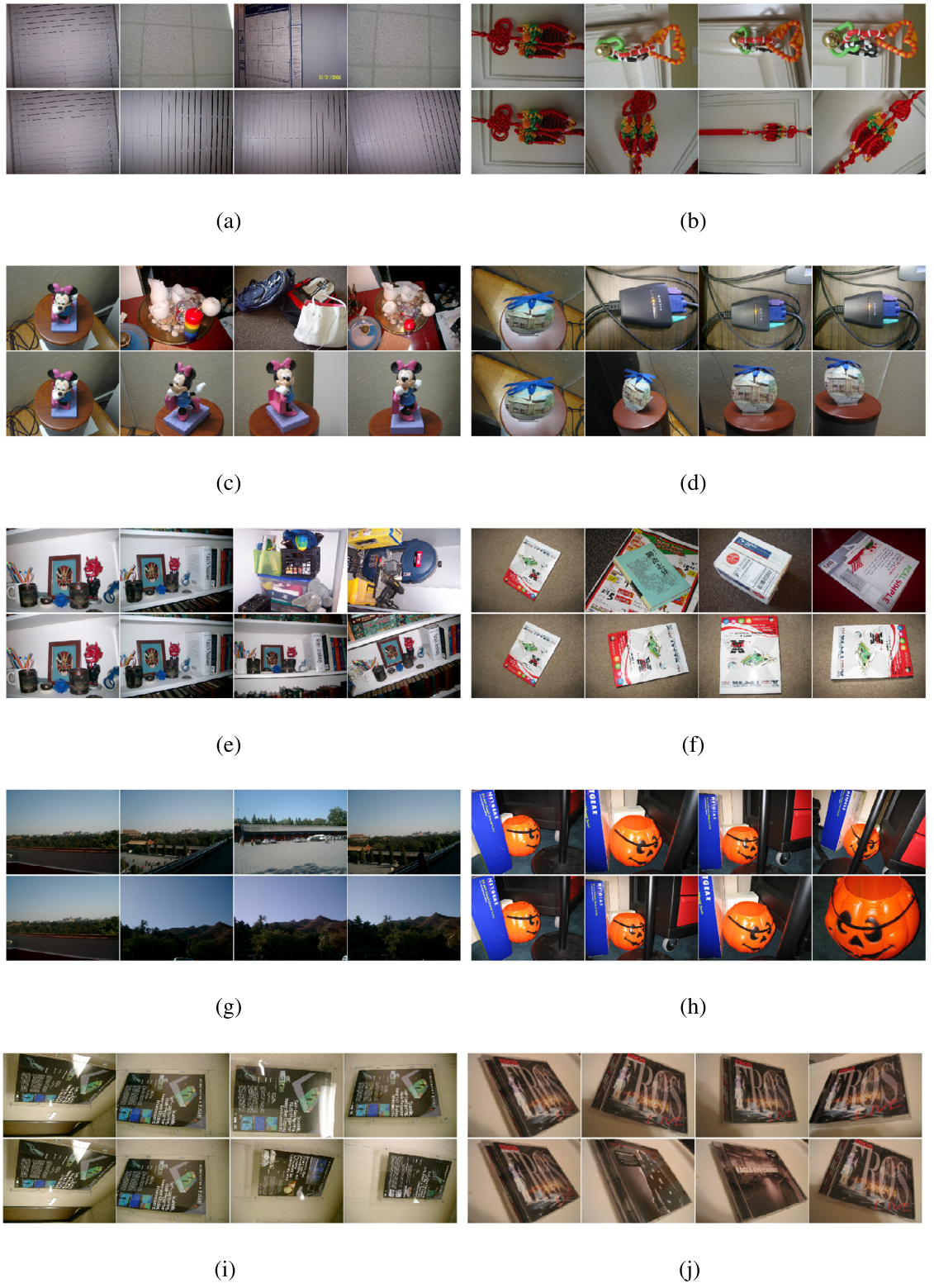


Fig. 4. Top 4 retrieval results on UKBench wo/w LRS re-ranking. In each group, the first column represents both the highest-ranked retrieved database image and the query image, and results without and with re-ranking are given in the first and second rows, respectively. Results in (a)–(f) are cases in which the re-ranking improves accuracy, while (g)–(j) show some failure cases.

can be effected by the new comers. A compromise for CDM could be that the update being performed only after a number of insertions. Note that this also applies to our method with CDM extension and database augmentation extension.

4.4. Visualization and analyses

To intuitively demonstrate the effect of applying the proposed LRS re-ranking, we select a few groups of retrieval results from UKBench dataset in Fig. 4. 6 groups of successful results are shown in Fig. 4(a)–4(f), while 4 groups of failure cases are shown in Fig. 4(g)–4(j). The results are produced by LRS re-ranking with Mean-AP without any extensions.

In Fig. 4(a), the initial query fails to retrieve all 3 ground-truth database images due to the sever 90 ° rotation. Our method, on the other hand, successfully retrieves them, because the 3 database images are mutually similar and are ranked closely in the local residual feature space. Fig. 4(b) is a contrary case. The distractive false positives identified in the original feature space show similar appearances with the query, but the ground-truth database images have great view transformation. This demonstrates that the false positives are excluded from the query in the local residual feature space. Fig. 4(c) and (d) are two cases where the ground-truths are not severely different from the query, but the initial query fails with scattered and clustered distractors, respectively. Our method can handle both the two types of distractors. Two more successful examples are given in Fig. 4(e) and (f).

As to the failure case in Fig. 4(g), we see the original feature recalls 2 ground-truth database images, which have prominent visual content variances *w.r.t.* the query. However, our re-ranking process loses them and retrieves 3 other scenes instead. These 3 images appear mutually identical, and are also very similar to the query. In Fig. 4(h), our method misses one pumpkin toy occluded by the pillar, and returns another image showing the same toy instance with marginal different background. Fig. 4(i) and (j) show two more cases, where the false positives returned by our method are very similar to the query in appearance, and are mutually similar. This contributes to most failure cases to our observation.

5. Conclusion

A novel similarity measurement, Local Residual Similarity (LRS), is proposed and investigated in this paper. LRS features in two aspects. It works in a local neighborhood of the query defined by a subset of the top-ranked images. The residuals are produced by subtracting anchors from the original feature vector, and in turn used in the second-round similarity calculation. This paper discusses various strategies to formulate the neighborhood and anchor computation and provides extensions of LRS with reciprocal neighborhood constraints. Effectiveness of LRS is demonstrated on two benchmarks with limited computational overhead.

Acknowledgement

This work was supported in part to Dr. Houqiang Li by 973 Program under contract No. 2015CB351803, NSFC under contract No. 61325009 and No. 61390514, in part to Dr. Wengang Zhou by NSFC under contract No. 61472378 and No. 61632019, and the Fundamental Research Funds for the Central Universities, and in part to Dr. Qi Tian by ARO grant W911NF-15-1-0290 and Faculty Research Gift Awards by NEC Laboratories of America and Blippar. This work was supported in part by NSFC under contract No. 61429201.

References

[1] D. Nister, H. Stewenius, Scalable recognition with a vocabulary tree, in: CVPR, 2006, pp. 2161–2168.
[2] J. Sivic, A. Zisserman, Video google: a text retrieval approach to object matching in videos, in: ICCV, 2003, pp. 1470–1477.
[3] M. Wang, G. Li, Z. Lu, Y. Gao, T.-S. Chua, When amazon meets google: product visualization by exploring multiple web sources, ACM Trans. Internet Technol. (TOIT) 12 (4) (2013) 12.
[4] H. Jégou, C. Schmid, H. Harzallah, J. Verbeek, Accurate image search using the contextual dissimilarity measure, IEEE Trans. Pattern Anal. Mach. Intell. 32 (1) (2010) 2–11.
[5] R. Arandjelović, A. Zisserman, Three things everyone should know to improve object retrieval, in: CVPR, 2012, pp. 2911–2918.
[6] M. Wang, H. Li, D. Tao, K. Lu, X. Wu, Multimodal graph-based reranking for web image search, IEEE Trans. Image Process. 21 (11) (2012) 4649–4661.
[7] I. Dimitrovski, D. Kocev, S. Loskovska, S. Džeroski, Improving bag-of-visual-words image retrieval with predictive clustering trees, Inf. Sci. 329 (2016) 851–865.
[8] H. Jégou, M. Douze, C. Schmid, P. Pérez, Aggregating local descriptors into a compact image representation, in: CVPR, 2010, pp. 3304–3311.
[9] J. Wan, D. Wang, S.C.H. Hoi, P. Wu, J. Zhu, Y. Zhang, J. Li, Deep learning for content-based image retrieval: acomprehensive study, in: MM, 2014, pp. 157–166.
[10] A. Babenko, A. Slesarev, A. Chigorin, V. Lempitsky, Neural codes for image retrieval, in: ECCV, 2014, pp. 584–599.
[11] Y. Gong, L. Wang, R. Guo, S. Lazebnik, Multi-scale orderless pooling of deep convolutional activation features, in: ECCV, 2014, pp. 392–407.
[12] R. Arandjelović, A. Zisserman, All about VLAD, in: CVPR, 2013, pp. 1578–1585.
[13] Z. Liu, S. Wang, Q. Tian, Fine-residual VLAD for image retrieval, Neurocomputing 173 (2016) 1183–1191.
[14] H. Jégou, M. Douze, C. Schmid, Product quantization for nearest neighbor search, IEEE Trans. Pattern Anal. Mach. Intell. 33 (1) (2011) 117–128.
[15] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: CVPR, 2016, pp. 770–778.
[16] S. Bai, X. Bai, W. Liu, Multiple stage residual model for image classification and vector compression, IEEE Trans. Multimedia 18 (7) (2016) 1351–1362.
[17] X. Bai, S. Bai, Z. Zhu, L.J. Latecki, 3D shape matching via two layer coding, IEEE Trans. Pattern Anal. Mach. Intell. 37 (12) (2015) 2361–2373.
[18] A. Babenko, V. Lempitsky, Aggregating local deep features for image retrieval, in: ICCV, 2015, pp. 1269–1277.
[19] A. Gordo, J. Almazán, J. Revaud, D. Larlus, Deep image retrieval: learning global representations for image search, in: ECCV, Springer, 2016, pp. 241–257.
[20] F. Radenović, G. Tolias, O. Chum, CNN image retrieval learns from bow: unsupervised fine-tuning with hard examples, in: ECCV, Springer, 2016, pp. 3–20.

[21] R. Arandjelovic, P. Gronat, A. Torii, T. Pajdla, J. Sivic, Netvlad: CNN architecture for weakly supervised place recognition, in: CVPR, 2016, pp. 5297–5307.
[22] L. Yang, A. Hanjalic, Supervised reranking for web image search, in: MM, 2010, pp. 183–192.
[23] X. Wang, S. Qiu, K. Liu, X. Tang, Web image re-ranking usingquery-specific semantic signatures, IEEE Trans. Pattern Anal. Mach. Intell. 36 (4) (2014) 810–823.
[24] B. Siddiquie, R.S. Feris, L.S. Davis, Image ranking and retrieval based on multi-attribute queries, in: CVPR, 2011, pp. 801–808.
[25] A. Kovashka, D. Parikh, K. Grauman, Whittlesearch: image search with relative attribute feedback, in: CVPR, 2012, pp. 2973–2980.
[26] D.C.G. Pedronette, J. Almeida, R.D.S. Torres, A scalable re-ranking method for content-based image retrieval, Inf. Sci. 265 (2014) 91–104.
[27] S. Bai, X. Bai, Z. Zhou, Z. Zhang, L. Jan Latecki, Gift: A real-time and scalable 3D shape search engine, in: CVPR, 2016, pp. 5023–5032.
[28] K. Guo, R. Zhang, Z. Zhou, Y. Tang, L. Kuang, Combined retrieval: a convenient and precise approach for internet image retrieval, Inf. Sci. 358 (2016) 151–163.
[29] V. Jain, M. Varma, Learning to re-rank: query-dependent image re-ranking using click data, in: WWW, 2011, pp. 277–286.
[30] R. Arandjelović, A. Zisserman, Multiple queries for large scale specific object retrieval, in: BMVC, 2012, pp. 1–11.
[31] O. Chum, J. Philbin, J. Sivic, M. Isard, A. Zisserman, Total recall: automatic query expansion with a generative feature model for object retrieval, in: ICCV, 2007, pp. 1–8.
[32] O. Chum, A. Mikulik, M. Perdoch, J. Matas, Total recall II: query expansion revisited, in: CVPR, 2011, pp. 889–896.
[33] D. Qin, Y. Chen, M. Guillaumin, L. Van Gool, Learning to rank bag-of-word histograms for largescale object retrieval, in: BMVC, 1, 2014, p. 7.
[34] X. Yang, L. Prasad, L.J. Latecki, Affinity learning with diffusion on tensor product graph, IEEE Trans. Pattern Anal. Mach. Intell. 35 (1) (2013) 28–38.
[35] S. Zhang, M. Yang, T. Cour, K. Yu, D.N. Metaxas, Query specific rank fusion for image retrieval, IEEE Trans. Pattern Anal. Mach. Intell. 37 (4) (2015) 803–815.
[36] D.C.G. Pedronette, R.d.S. Torres, A correlation graph approach for unsupervised manifold learning in image retrieval tasks, Neurocomputing 208 (2016) 66–79.
[37] D.C.G. Pedronette, J. Almeida, R.d.S. Torres, A graph-based ranked-list model for unsupervised distance learning on shape retrieval, Pattern Recognit. Lett. 83 (2016) 357–367.
[38] M. Donoser, H. Bischof, Diffusion processes for retrieval revisited, in: CVPR, 2013, pp. 1320–1327.
[39] G. Tolias, H. Jégou, Visual query expansion with or without geometry: refining local descriptors by feature aggregation, Pattern Recognit. 47 (10) (2014) 3466–3476.
[40] L. Liu, M. Yu, L. Shao, Local feature binary coding for approximate nearest neighbor search, BMVC, 2015.
[41] H. Jégou, H. Harzallah, C. Schmid, A contextual dissimilarity measure for accurate and efficient image search, in: CVPR, IEEE, 2007, pp. 1–8.
[42] S. Bai, X. Bai, Sparse contextual activation for efficient visual re-ranking, IEEE Trans. Image Process. 25 (3) (2016) 1056–1069.
[43] D. Qin, S. Gammeter, L. Bossard, T. Quack, L. Van Gool, Hello neighbor: accurate object retrieval with k-reciprocal nearest neighbors, in: CVPR, 2011, pp. 777–784.
[44] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, T. Darrell, Caffe: convolutional architecture for fast feature embedding, in: MM, 2014, pp. 675–678.
[45] L. Zheng, Y. Zhao, S. Wang, J. Wang, Q. Tian, Good practice in CNN feature transfer, arXiv:1604.00133 (2016).
[46] H. Jégou, M. Douze, C. Schmid, Hamming embedding and weak geometric consistency for large scale image search, in: ECCV, 2008, pp. 304–317.