Fast approximate kernel-based similarity search for image retrieval task

David GORISSE[1], Matthieu CORD[2], Frederic PRECIOSO[1], Sylvie PHILIPP-FOLIGUET[1]
[1] *ETIS, CNRS, ENSEA, Univ Cergy-Pontoise, France,* [2] *LIP6, UPMC-P6, Paris, France*
{*david.gorisse, frederic.precioso, philipp*}*@ensea.fr, matthieu.cord@lip6.fr*

Abstract

In content based image retrieval, the success of any distance-based indexing scheme critically depends on the quality of the chosen distance metric. We propose in this paper a kernel-based similarity approach working on sets of vectors to represent images. We introduce a method for fast approximate similarity search in large image databases with our kernel-based similarity metric. We demonstrate our algorithm on image retrieval task and show it to be accurate and faster than linear scanning.

1. Introduction

In computer vision, a lot of techniques are now based on unordered sets of local features to represent the image content. In [10], Lowe shows that features based on local description like Points of Interest (PoI), provide a good representation of images.

Content based image retrieval requires a similarity function between images in order to achieve database ranking thanks to any query image. Designing powerful similarities and ranking schemes on unordered sets of local features is a very challenging task. Lowe proposed a two-steps technique, a fast k-Nearest Neighbors (k-NN) search for all the PoIs of the query, and a voting strategy to rank the images by counting their number of matched PoIs.

One step further has been done by K. Grauman and T. Darrell [8]: they propose to use a kernel function, the pyramid match kernel, to compute the similarity between sets of PoIs. They speed up their kernel ranking scheme using an approximate k-NN search [2]. Their scheme gives good results, but are limited to kernels that can be explicitly expressed as a dot product in an induced space[1]. Extensions proposed by K. Grauman [7] as the vocabulary-guided pyramid scheme also suffers of the same limitations.

We investigate here another way to get fast scheme retrieval for kernel functions without explicit formulation in the induced space. The first requirement is to define admissible kernels working on sets of PoI vectors. We use the class of kernels on sets of vectors derived from kernels on vectors (see for instance chap. 9, [13]). The aforementioned kernels have been successfully adapted for object retrieval [11, 6]. The major interest is that we are no more limited for the choice of the kernel.

We propose in this paper an original fast retrieval scheme, similar to the Lowe's voting scheme, but using our kernel-based similarity framework. We adapt the sublinear Locality Sensitive Hashing (LSH) scheme (proposed in [9] to approximate k-NN search) to select a subset of images that should be relevant enough to be at the top of the similarity ranking. To decrease the computational complexity, the kernel is only computed in this subset. The resulting scheme is an approximation of the exact similarity ranking. After introducing our algorithm, we evaluate the accuracy of the approximation and the efficiency of our approach.

2. Fast approximate kernel-based search

2.1. Kernels on bags

Local-based image analysis provides a powerful data representation framework. Relevant features are extracted on local patches to characterize the objects embedded in the image. Image I_j is represented by a bag B_j composed of s vectors $\mathbf{b}_{sj}$. The similarity between two bags B_i and B_j is measured by the following class of kernels on bags K [13]:

$$K(B_i, B_j) \triangleq \sum_{\mathbf{b}_{ri} \in B_i} \sum_{\mathbf{b}_{sj} \in B_j} k(\mathbf{b}_{ri}, \mathbf{b}_{sj}) \qquad (1)$$

where k is a kernel function. This kernel (called minor kernel) is defined as a dot product in the space induced

[1]The mapping function between the original space and the Hilbert space is explicit. This assumption greatly limits the usable classes of kernels.

by the embedding function ϕ:

$$k(\mathbf{b}_{ri}, \mathbf{b}_{sj}) = \langle \phi(\mathbf{b}_{ri}), \phi(\mathbf{b}_{sj}) \rangle \qquad (2)$$

Although K kernels are quite relevant to evaluate image similarity between bags [11], they are computationally expensive. Such a kernel computation becomes intractable for image retrieval task in large databases, especially when feature bags contain many vectors. For example, 100 to 1000 SIFT vectors are usually extracted from one image [10].

To avoid the computational complexity of this kernel while keeping its similarity efficacy, we propose to do an approximate search: we do not compute the similarity for all the images of the database but only for a carefully chosen subset. The selection of the subset of interest is presented in the following section.

2.2. Approximate similarity search

Let I_q be a query image represented by bag $Q = \{\mathbf{q}_r\}$ containing r vectors. Using K as similarity, the retrieval problem in database $\mathcal{B}$ of bags B_j can be written as:

$$\underset{B_j \in \mathcal{B}}{Sort}(K(Q, B_j)) \qquad (3)$$

In precision-oriented retrieval applications [3], only top rank of similarity scores are interesting. It is not necessary to provide the full ranking of the database but only the TOPN[2]. Our idea is to quickly select the images that have a *high probability* to be at the top of the ranking, *i.e.* the images *very similar* to the query I_q.

We assume that two images are *very similar* if at least two local features are similar. This local similarity is expressed using a k-NN approach: the vector $\mathbf{b}_{sj} \in B_j$ from image I_j is similar to the vector $\mathbf{q}_r \in Q$ from query I_q if $\mathbf{b}_{sj} \in$ k-NN$(\mathbf{q}_r)$. If $\mathcal{S}$ denotes the subset of images from $\mathcal{B}$ that have at least one vector similar to one $\mathbf{q}_r$

$$\mathcal{S} = \{B_j | \exists(\mathbf{q}_r, \mathbf{b}_{sj}) \in Q \times B_j, \mathbf{b}_{sj} \in \text{k-NN}(\mathbf{q}_r)\}$$

the optimization problem in (3) is modified as follows:

$$\underset{B_j \in \mathcal{S}}{Sort}(K(Q, B_j)) \qquad (4)$$

This low-restrictive definition of *very similar* images allows us to avoid missing true relevant images in S. This modified optimization scheme is interesting if the computation of S may be fast. Instead of doing a linear scan for the k-NN search, we use a efficient indexing scheme based on LSH. In this way, the computational time to retrieve the closest neighbors will be negligible with respect to the time to compute the true kernel. The approximate search in (4) will be then about $\frac{|\mathcal{B}|}{|\mathcal{S}|}$ faster than a brute-force linear scan in (3).

[2]N is usually fixed by the user. Details are provided in our experiments.

2.3. LSH indexing

We shortly report in this section the basic LSH functionalities to explain how we use it in our context.

LSH solves the $(R, 1+\epsilon)$-NN problem: find at least one vector $\mathbf{b}'$ in the ball $B(\mathbf{q}, (1+\epsilon)R)$ if there is a vector $\mathbf{b}$ in the ball $B(\mathbf{q}, R)$. $\mathbf{b} \in B(\mathbf{q}, R)$ if $||\mathbf{b} - \mathbf{q}|| \leq R$. Indyk and Motwani [9] solved this problem for the Hamming metric with a complexity of $O(n^{1/(1+\epsilon)})$ where n is the number of vectors of the database. Datar and al. [4] proposed an extension of this method that solves this problem with the Euclidian metric and with similar time performances. The method generates some hash tables of points, where the hashing function works on tuples of random projections of the form: $h_{\mathbf{a},c}(\mathbf{b}) = \lfloor \frac{\mathbf{a}.\mathbf{b}+c}{w} \rfloor$ where $\mathbf{a}$ is a random vector whose each entry is chosen independently from a Gaussian distribution, c is a real number chosen uniformly in the range $[0, w]$, w specifies a bin width (which is set to be constant for all projections) and $\mathbf{b}$, a vector.

A tuple of projections specifies a partition of the space where all points inside the same part have the same key. All points with the same key are in the same bucket C. Clearly, if the number of projections is carefully chosen, then two points which hash into the same bucket C will be nearby in the feature space. To avoid boundary effects, many hash tables are generated, each using a different tuple of projections. In practice, a proportion of these points (called "false matches") will be at a distance greater than R from the query point $\mathbf{q}$. That is why, a check (computation of the Euclidian distance between all points $\mathbf{b}$ of bucket C and $\mathbf{q}$) is carried out to remove "false matches".

In our case, we do not want to find just one vector $\mathbf{b} \in B(\mathbf{q}, (1+\epsilon)R)$ but all of them. For that, we use a method from E^2LSH [1] which is a modified version of [4] to solve the $(R, 1-\delta)$-near neighbor problem: each vector $\mathbf{b}$ satisfying $||\mathbf{b} - \mathbf{q}||_2 \leq R$ has to be found with a probability $1-\delta$. Thus, δ is the probability that a near neighbor $\mathbf{b}$ is not reported.

2.4. Fast approximate similarity search scheme

The algorithm is composed of three stages: *(i)* computation of buckets, *(ii)* selection of images, *(iii)* computation of the kernel on selected images.

(i) In the first stage, the hash functions are generated to split the set of all $\mathbf{b}_{sj}$ of the database into buckets.

This stage may be time consuming but it is done off-line.

(ii) For a query $Q = \{\mathbf{q}_r\}$, the set of keys are computed for each vector $\mathbf{q}_r$ with the hash functions generated during the off-line stage *(i)*. Each key allows to quickly select a bucket C_i^r containing vectors $\mathbf{b}_{sj}$ that have a high probability to be in the ball $B(\mathbf{q}_r, R)$. A check is carried out to eliminate vectors that are at a distance greater than R of $\mathbf{q}_r$. All remaining bags B_j containing at least one vector $\mathbf{b}_{sj}$ in a bucket C_i^r are candidates:

$$\hat{\mathcal{S}} = \{B_j | \exists(r,s,i) : (\mathbf{b}_{sj}, \mathbf{q}_r) \in [C_i^r \cap B(\mathbf{q}_r, R)] \times C_i^r\}$$

(iii) The similarity is computed only for these images; the approximate search is defined by:

$$\underset{B_j \in \hat{\mathcal{S}}}{Sort}(K(Q, B_j)) \tag{5}$$

As seen in section 2.3, some of ”good matches” (vectors in ball $B(\mathbf{q}_r, R)$) are not reported, thus $\hat{\mathcal{S}} \subsetneq \mathcal{S}$. However, more similar B_j and Q, more numerous vectors $\mathbf{b}_{sj}$ close to vectors $\mathbf{q}_r$. As one match is enough to select the corresponding image, we see that a close target image is more likely to be selected, and $\hat{\mathcal{S}}$ should be a relevant estimation of $\mathcal{S}$.

The parameter R allows to tune the approximation of the search and the time complexity of the algorithm. The bigger R, the greater the number of selected images ($|\hat{\mathcal{S}}| \approx |\mathcal{B}|$), the lower the approximation of the search, but the lower the improvement of computation time compared to the true search ($\frac{|\hat{\mathcal{S}}|}{|\mathcal{B}|} \approx 1$).

In the next section, we empirically evaluate the tradeoff between the approximation and the speed up according to R.

3. Experiments

In this section we show that our approximate search is efficient in a content-based image retrieval. As Gosselin in [6] and Lyu in [11] have already proved the efficacy of this class of kernels on bags, we focus our results on the evaluation of our approximation: deterioration of the ranking and computational time saving compared to the search on all the database by linear scan.

We first show our approximation is almost equivalent to the true search, by comparing the ranking of similarity obtained with the approximate search relative to the ranking we would obtain with the search on the whole database. We focus our evaluation on the first N images of the ranking called TOPN. The evaluation is obtained by computing the accuracy of TOPN, defined as the number of images of the TOPN obtained with the true search actually found in the TOPN obtained with the approximate search.

Then we evaluate the computational time saving by using our approximation by measuring the ratio of the measuring number of images selected $|\hat{\mathcal{S}}|$ divided by the number of images in the database $|\mathcal{B}|$. Finally, we gave for various radii the actual time improvement factor obtained between the approximate and true search.

We display results of the first two evaluations with ”box and whisker plots”: each box has lines at the lower quartile, median value, and upper quartile values, whiskers extend from each end of the box to show the extent of the rest of data, and outliers are denoted with pulses. Circles are added to show mean values.

To evaluate our method, we used VOC2006 database [5] which contains 5,304 images. The database was indexing using the well-know approach combining MSER region detectors and SIFT descriptors [12]. Each image is described by a bag of about one hundred PoI. And each PoI is represented by a 128 dimensional SIFT descriptor representing the 16 8-bin histograms of image gradient orientations. For all our experiments we chose as parameter of E^2LSH [1] $\delta = 0.1$, which corresponds to 400 hash tables of 20 projections and we tested for various radii between 4.0 and 6.0. We tested the kernel of Eq.(1) with a minor kernel Gaussian L_2. All results are obtained with 200 randomized queries.

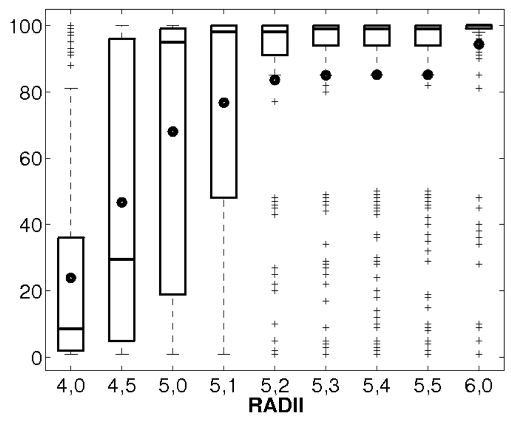

Figure 1. Accuracy of TOP100 **for various radii of search around query points.**

Figure (1) displays plotted results of the deterioration measure induced by the approximation for $N = 100$. With a search radius greater than 5.2, on average, more than 80% of images of the TOP100 are reported, and for 50% of queries, it reaches 98% of accuracy. In comparison, on Caltech-101 database Grauman obtained a accuracy of 76% for a TOP5 between her hashing retrieval and a true search [8]. Although

Table 1. Speed improvement factor regards to the true search.

Radius	4	5	5.2	6
factor	122.17	14.85	10.03	3.19

the mean accuracy quickly decreases for lower radius, some queries reach good accuracy. For example, with a radius of 5.0 and a mean accuracy of 68%, we have all the same 95% of accuracy for 50% of queries.

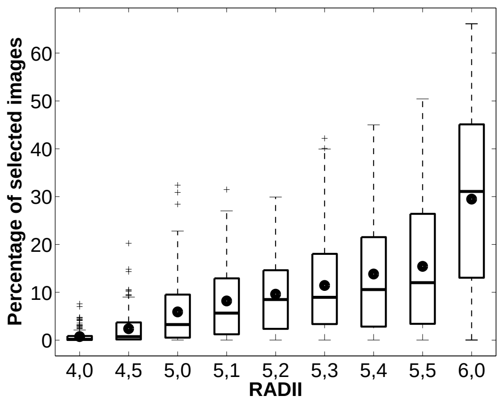

Figure 2. Percentage of selected images for various radii.

Figure (2) displays the percentage of selected images of the database. This figure helps to explain the last result. Indeed, for a radius of 5.1 and for 25% of the queries, less than 1.23% of images of the database (i.e. less than 65 images) were selected. Therefore, the accuracy could not exceed 0.65 for 25% of the queries. For some queries, we have thus selected less than 100 images which explains the bad results.

This figure also shows that for a radius of 5.2 that gives good result, on average, only 9.6% of images of the database were selected. We must therefore have an improvement of computation time of a factor of 10 towards the computation time of the true search.

We have checked that the improvement of computing time was well reached by comparing obtained computation time for the true search and the approximate one (tab.1). For small radii the time improvement factor is important and it goes to one as R increases.

The average time to compute the true search for a query is about 95 seconds on a machine with a 3.2 GHz processor and 8 GB of memory. For a radius of 5.2, the selection of $\hat{\mathcal{S}}$ takes 0.14 second and the similarity on this selection takes 9.26 seconds. The use of LSH to select images makes the computational time of this step negligible compared to computation time of the true search, so we are $\frac{|\hat{\mathcal{S}}|}{|\mathcal{B}|} \approx 10$ faster for a radius of 5.2. This approximation makes computing time acceptable for an online learning scheme.

4. Conclusion

In this paper, we introduced a method to efficiently achieve similarity retrieval in large image databases. Our technique is based on a powerful similarity using local-based image representation and kernel functions. We short cut the full database linear scan by only computing the kernel-based similarity on a carefully chosen subset of images. We combined this strategy with a LSH scheme to efficiently compute the pre-seleted image subset. Experiments on image datasets demonstrated that our method achieves a good tradeoff between accuracy and efficiency for the image similarity search task. We are currently working on the embedding of this strategy in online category learning scheme.

References

[1] A. Andoni. E2lsh. http://www.mit.edu/∼andoni/LSH/.

[2] M. Charikar. Similarity estimation techniques from rounding algorithms. *ACM*, pages 380–388, 2002.

[3] Y. Chiaramella, P. Mulhem, M. Mechkour, I. Ounis, and M. Pasca. Towards a fast precision-oriented image retrieval system. *ACM*, pages 383–384, 1998.

[4] M. Datar, N. Immorlica, P. Indyk, and V. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. *SCG*, pages 253–262, 2004.

[5] M. Everingham, A. Zisserman, C. K. I. Williams, and L. Van Gool. The PASCAL Visual Object Classes Challenge 2006 (VOC2006).

[6] P.-H. Gosselin, M. Cord, and S. Philipp-Foliguet. Kernel on bags for multi-object database retrieval. *ACM CIVR*, 2007.

[7] K. Grauman. *Matching Sets of Features for Efficient Retrieval and Recognition*. PhD thesis, MIT, 2006.

[8] K. Grauman and T. Darrell. Pyramid match hashing: Sub-linear time indexing over partial correspondences. *CVPR*, pages 1–8, 2007.

[9] P. Indyk and R. Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. *ACM*, pages 604–613, 1998.

[10] D. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, 2004.

[11] S. Lyu. Mercer kernels for object recognition with local features. *CVPR*, 2:223–229, 2005.

[12] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. Gool. A comparison of affine region detectors. *IJCV*, 65(1):43–72, 2005.

[13] J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.